Justification Mining: Developing a Novel Machine Learning Method for Identifying Representative Sentences and Summarising Sentiment in Financial Text

Kevin Patel

Linacre College

University of Oxford

Supervised by

Professor Bob Coecke and Dr. Edwin Simpson

Submitted: Hilary Term,

11 March 2021

This thesis is submitted to the Department of Computer Science, University of Oxford, in partial fulfilment of the requirements for the degree of Doctor of Philosophy

Declaration of Authorship

I declare that this thesis is entirely my own work, and except where otherwise stated, describes my own research.

To my family

Table of Contents

1 Acknowledgements

I would like to thank both my supervisors, Professor Bob Coecke and Dr. Edwin Simpson, for their invaluable support and guidance throughout this process. I would also like to extend my gratitude for all the other academics at Oxford who have helped me along the way, including Professor Marina Jirotka, Dr. Philip Inglesant, Professor Stephen Roberts, Professor Amir Amel-Zadeh, Professor Martin Schmalz, Dr. Martin Tegnér, Dr. Jan Calliess, and Dr. Jason Nurse. Other than academics, many other people have supported me throughout this journey at Oxford: my college advisor, Dr. Catherine Walter; everyone at Linacre College and the Disability Advisory Service; all the amazing people at the Department of Computer Science; Collette Byrne from OCD Action; and, finally, of course, my family for all their love and support. Thank you all; without your encouragement, I could not have made it this far!

2 Abstract

As the fields of machine learning, natural language processing, and big data become increasingly important throughout various industries, including finance, it becomes crucial to evaluate how they are being utilised and understand the motivations behind the recommendations of algorithms, in order to make sound decisions based on this information. In finance, particularly, being able to create concise and easy to comprehend tools for understanding machine learning models is extremely beneficial. These tools could serve multiple purposes, including allowing managers to better explain to and convince investors of machine learning based trading strategies. Decision-makers themselves might also be able to act according to the information provided by these tools, which could assist in, for example, making more ethically informed decisions. Thus, this type of research not only has implications in the areas of machine learning, Natural Language Processing (NLP), and finance but also in fields like AI explainability, model interpretability and Responsible Research and Innovation.

The central problem this thesis addresses is whether machine learning methods can be used to mine representative sentences from financial text that are able to capture the majority of sentiment in a full document with only a few sentences. These mined sentences are referred to as 'justifications', and the process has been called 'justification mining'. The full documents used here are data taken from 10-K filings. Before examining justification mining, however, transfer learning methods suitable for training data annotated at a different level than testing data (sentence- versus document-level) are assessed. The purpose of this is to address a problem that often occurs in NLP research, where no annotated training data perfectly suited to the research is readily available. Therefore, methods must be created to make use of what is available, in this case sentence-annotated training data being employed to train classifiers for prediction on document-level testing data. These transfer learning methods are then employed in the next step, which focuses on justification mining. The process of justification mining itself is first developed using transformer models to encode embeddings and, then, using clustering algorithms and cosine similarity to extract or mine justifications. This process is evaluated by comparing sentiment from mined justifications to sentiment from full documents, in order to assess the ability of justification mining to capture or summarise sentiment. It is also evaluated by correlating predicted sentiment from mined justifications and full documents to future stock returns, to gauge whether justification mining offers any benefit in identifying signals in the data for financial purposes.

Little work has been done previously that evaluates the results of financial sentiment analyses in this way. In the NLP domain, the best way of extracting aggregate justifications for sentiment is still an open question. Moreover, few research papers attempt to apply transfer learning from lower-level data to entire documents. Nor are 10-K filings widely studied in this context, as they are difficult to parse. The methods created in this thesis might offer a novel means of providing information that can assess the motivations behind sentiment analyses.

In the transfer learning process, feature engineering and preprocessing steps were modified to obtain accuracies up to 0.903. For justification mining, considering statistically significant ($p \leq 0.05$) correlations of $|r| > 0.2$, sentiment from mined justifications more often correlated with future stock returns than did full document sentiment. Although the correlations (r) were somewhat weak overall, they could potentially be combined with traditional alpha signals to enhance these signals (see Section 11 for further discussion). Moreover, high degrees of similarity were found between aggregated

sentiment from mined justifications and full documents, with similarity scores up to $0.99\overline{9}$, supporting the efficacy of justification mining in capturing full document sentiment. In these evaluations, transformer models performed better in numericising text for input into ML models than traditional approaches like Bag of Words. In fact, every statistically significant sentiment to stock return correlation bar one, as well as every mined justification to full document similarity and correlation score above 0.7, used transformer models for numericisation.

These results imply that justification mining might be successful in eliminating sentiment noise in financial data, as well as in capturing the majority of sentiment from a full document. Moreover, transformer models might provide an advantage over traditional approaches like Bag of Words for numericisation of text. Mined justifications, themselves, provide an easily interpretable and presentable means of explaining the output of sentiment analysis and have numerous uses, including identifying the driving factors behind sentiment in a financial document, which could be helpful for making more ethically informed decisions or building investor trust in the methodology of sentiment analysis algorithms.

3 Introduction

The field of intelligent algorithms has become increasingly important in both the realm of academia and that of business. Spanning everything from financial strategies to computer vision, from search engines to self-driving cars, it is set to have a far-reaching impact on society in the immediate future. People engage with machine learning algorithms almost daily in simple tasks ranging from using search engines to having their emails automatically filtered for spam. Improvements in computational power, natural language processing (NLP) and machine learning have allowed financial firms to venture into the field of big data. In the past, most firms focused on traditional financial data, such as securities movement and tick data, but now many are realising the importance of social media and other news along with the aggregation of information from all these sources. However, as the importance of this information grows, the motivations and justifications behind the sentiment in data have not been sufficiently studied.

In this thesis, I first discuss the literature on natural language processing (NLP), sentiment analysis, transformer models and semantically meaningful sentence embeddings. The applications of these fields in finance are outlined along with the implications of this research, for example addressing Responsible Research and Innovation (RRI) concerns utilising machine learning by discussing AI explainability in finance through natural language processing. Probably the biggest contribution of this research is the creation of a tool capable of capturing the sentiment of a document in just a few sentences and potentially utilising such summarised sentiment as a signal for future stock returns. This can have substantial implications not only for the fields of machine learning and NLP but also for RRI, AI explainability and finance, where these methods have not been so widely applied.

The research investigation is organised around three research questions. Research Question 1 addresses methods for applying transfer learning between two financial documents annotated at different levels. The main goal is to devise a system of feature engineering and other steps that works best for a sentence- to document-level transfer learning task. Once this method has been trialled and assessed, it will then be utilised in tackling Research Questions 2 and 3.

For research questions 2 and 3, a new NLP method termed *justification mining* will be developed, presented, and evaluated. The goal of this method will be to summarise the sentiment of a document, capturing the majority of the sentiment contained in an entire document in just a small number of sentences. Addressing research questions 2 and 3 comprises the bulk of the research investigation and constitutes the main focus of this thesis.

Section 4 introduces the research questions. Section 5 presents an initial literature review, containing much of the literature that influenced the goals behind developing justification mining. Section 6 begins addressing research question 1, discussing and evaluating the results of transfer learning. Section 7 examines the development and assessment of justification mining, which focuses on research questions 2 and 3. Section 8 discusses the benefits of this research and implications of potential success. Section 9 outlines the contributions of this thesis and gaps in the literature addressed by this research, including a discussion of AI explainability. Section 10 presents various challenges that arose during the research investigation and limitations of this research. Section 11 discusses possible directions of future work, including how justification mining could potentially help

in making more ethically informed decisions in finance. Finally, Section 12 draws overall conclusions for this research.

4 Research Questions and Goals

Before addressing the research questions, a few qualifying pieces of information should be noted. As mentioned previously, the focus of research question 1 is to examine transfer learning, mainly utilising sentence-annotated training data to make predictions on document-level data. This document-level data comprises SEC 10-K filings, which constitutes most of the testing datasets in this thesis. Research question 1 is addressed in Section 6.

After this, a new method that I have named justification mining is presented. Justification mining is the focus of both research questions 2 and 3 and is presented and evaluated in Section 7. It involves utilising transformer models to extract semantically meaningful sentence embeddings from financial text and clustering these embeddings in order to mine representative sentences (which I call justifications). Here, another objective is the evaluation of Sentence-BERT, the tool used to encode the sentence embeddings with deep learning transformer models. Encoded sentence embeddings from Sentence-BERT are not only employed in justification mining itself but are also evaluated as a form of text numericisation to create features for input into classifier algorithms. The central question is research question 2, which prompts the most important findings and results of this thesis. It focuses on justification mining's ability to capture the majority of sentiment in a document by comparing sentiment analyses on mined justifications versus full documents. Research question 3 addresses a second form of evaluating justification mining, by examining the correlation of predicted sentiment to future stock returns and comparing the correlation when using mined justifications versus full documents.

4.1 Research Question 1

Can sentiment analysis models trained on financial sentence-annotated data be used to make predictions on document-level financial data?

4.1.1 Sub-question 1-1*:* What are the challenges and limitations present in performing sentiment analysis utilising long document-level financial testing data (such as 10-K filings)?

4.2 Research Question 2

Can machine learning be utilised to mine justifications/representative sentences from financial text data for the purposes of capturing the sentiment of a full document with only a few sentences?

4.2.1 Sub-question 2-1*:* Can mined justifications function as an easily interpretable form of sentiment summarisation?

4.2.2 Sub-question 2-2*:* Do transformer models and Sentence-BERT (when utilised as a means for feature engineering and numericising text as semantically meaningful sentence embeddings for input into machine learning algorithms) offer value over traditional approaches, such as Bag of Words or TFIDF, for the goal of comparing mined justification sentiment scores and full document sentiment scores?

4.2.3 Sub-question 2-3*:* Can mined justifications provide information that could make machine learning or sentiment analysis based financial strategies more explainable and convincing to investors? Can this information also be employed to help make more ethically informed decisions in finance?

4.2.4 Sub-question 2-4*:* What does a successful machine learning application here mean? What are the consequences and benefits of it? What are the critiques of such a tool?

4.3 Research Question 3

Do mined justifications provide an advantage over using full documents for informing future stock returns via sentiment analysis, perhaps through eliminating noise in the sentiment?

4.3.1 Sub-Question 3-1: Do transformer models and Sentence-BERT also present value over traditional approaches (Bag of Words/TFIDF) as a feature engineering/numericisation tool for apprising future returns?

5 Initial Literature Review

This section contains the initial literature review. As such, the focus mainly revolves around the materials that lead to the formulation of and purpose for developing justification mining. Additional technical literature related to sentiment analysis, natural language processing, transfer learning, justification mining and machine learning is presented throughout Sections 6 and 7 (which cover the research investigation) in relevant and related parts, rather than here in the initial literature review.

Instead, here, the literature pertaining to ethical decision-making utilising machine learning and frameworks for doing so is reviewed. General concepts in machine learning and sentiment as well as semantic analysis are introduced. State-of-the-art techniques in justification, argument and opinion mining are discussed. The principal definitions of RRI are evaluated. The origins of RRI leading eventually to the study of RRI in robotics are outlined. Several themes in these fields, including responsibility, culpability and ethics, are also discussed. Finally, the study of ethics in Natural Language Processing is reviewed.

5.1 Ethical Decision-Making Utilising Machine Learning

Ethical decision-making has presented a grand challenge in artificial intelligence for a long time (Wallach and Allen, 2008). Much of the difficulty in doing this work is caused by the unavailability of any formal specification of ground-truth ethical principles. Dwork et al. (2012), in reference to fairness in machine learning, argued that an "*approximation as agreed upon by society*" must be utilised when such ground-truth ethical principles are absent (Dwork et al., 2012, p. 214).

There has been some work in creating tools to automate ethical decision-making, as well as frameworks for moral decision-making in machine learning (Noothigattu et al., 2017; Conitzer et al., 2017). One such method attempts first to learn a model based on societal preferences and then aggregate these to make an ethical choice (Noothigattu et al., 2017). Noothigattu et al. (2017) create a mathematical model based on swap-dominance efficient voting rules and trial their method in the field of self-driving cars. Their methodology contains four steps: data collection, learning, summarisation and aggregation (Noothigattu et al., 2017). Utilising data from 1.3 million voters from the Moral Machine platform, Noothigattu et al. (2017) accomplished an accuracy on 3000 test instances of 98.2% at two alternatives per ethical dilemma instance and 95.1% at ten alternatives. They argue that while their method allows for credible ethical decision making on dilemmas in the self-driving car domain, it is not an end-all solution (Noothigattu et al., 2017).

Conitzer et al. (2017) also refer to voting based methods on the moral values of multiple people. They argue that while game and decision theory have allowed for much domain-independent research in machine learning, this generalisability is lacking for moral or ethical decision-making (Conitzer et al., 2017). Conitzer et al. (2017) state that a framework is required for a machine agent to utilise in making ethical decisions in a wide variety of contexts. The majority of machine learning research falls within relatively straightforward ethical frameworks such as consequentialism and utilitarianism (Conitzer et al., 2017). Conitzer et al. (2017) instead discuss generalisable frameworks based on game theory and machine learning on instances labelled by people. While their machine learning approach

is more flexible than the game-theoretic method, the two can also complement each other by utilising the outputs from game-theoretic approaches as the features in machine learning methods (Conitzer et al., 2017).

While none of this work has involved word/sentence embedding or accounting for end-user specifications, it does provide a useful context for building a methodology. For instance, both Conitzer et al. (2017) and Dwork et al. (2014) allude to the importance of annotation by end-users. In future work, the techniques researched in this thesis could lead to a tool for incorporating such information towards the goal of more ethically informed decision-making (see Section 11.1).

5.2 Introduction to Machine learning, Sentiment and Semantic Analysis

Every machine learning algorithm has certain characteristics in common. Machine Learning differs from traditional methods in a number of key ways. Firstly, machine learning utilises data to determine the form of an algorithm or function, rather than utilising the function to create data. Secondly, in traditional computing, a program is usually utilised to create output. With machine learning, however, a program is written to create another program. Machine learning algorithms are either linear or nonlinear. Linear algorithms make the most assumptions on the form of the algorithm and gradient descent can be utilised for optimisation here. Nonlinear algorithms make fewer assumptions. There are also two general types of outputs, real values and classification-based outputs (such as the typical binary option). Machine learning algorithms are all based on the idea of a function that transforms an input into an output, one of the most basic concepts learnt in early mathematics education. In computer science, this function can also be called a program (Brownlee, 2017). The form of this algorithm or program is what machine learning tries to deduce by determining what best fits with the data.

Another important concept in machine learning is that of an irreducible error function, as there is never a function that ideally and fully characterises the relationship between output and input data (Brownlee, 2017). Machine learning algorithms are further separated into two other categories, parametric and nonparametric algorithms. Parametric models make stronger assumptions on the form of a model and nonparametric make less assumptions. This means that nonparametric models are often more flexible and powerful but more complex, slower and sometimes overfit to the data so that new data is potentially not characterised well by the algorithm (Brownlee, 2017). This could lead to the need for generalisation or regularisation techniques so the algorithm can function better on new data.

The last categorisation of techniques in machine learning is in regard to supervised versus unsupervised machine learning. Here the difference rests in whether the training data has both outputs and inputs (supervised) or just inputs (unsupervised). Machine learning algorithms are trained on this data in order to guess the best form or parameters of an algorithm before they are tried on new data to make new estimates. For instance, in the scope of this thesis, a typical classification algorithm could be utilised in determining whether a justification from financial text is seen as indicating a rise or fall in stock price (as determined by an annotator). It can utilise historic data for training and testing as well as future data for novel estimations (Brownlee, 2017). This is of course a simplification of

strategies utilised at the moment in machine learning but creates a useful starting point for discussions of justification mining.

Finally, one of the most important concepts in the crafting of machine learning algorithms is the bias-variance issue (Brownlee, 2017). Bias describes the assumptions an algorithm makes to simplify, and variance is the sensitivity of an algorithmic model to the training data (Brownlee, 2017). Increasing one will always decrease the other. For instance, if the training data were sporadic with many short peaks and troughs but displayed a general linearly negative trend, a simple negatively sloped line would show large bias but low variance, whereas a line that perfectly connected each training point (thus not being straight but having many peaks and troughs as well) would show large variance and low bias. Parametric and linear algorithms contain large bias and small variance, and nonparametric and nonlinear algorithms have small bias and large variance. It is the balancing of these two that is one of the main battles in the creation and utilisation of machine learning algorithms (Brownlee, 2017).

Often hand-in-hand with machine learning and discussing natural language processing is sentiment and semantic analysis. Sentiment analysis involves determining overall sentiment towards a specific prediction task by combining the large amount of data available in social media, blogs, news feeds, etc. (often referred to as Big Data) to make guesses on classification style machine learning problems (Mäntylä et al., 2018). Semantic analysis goes a step beyond this to look at meaning (Foltz, 1996; Landauer et al., 1998). In this thesis, although I utilise sentiment analysis in many parts of the research investigation, steps involving justification mining can fall within the realm of semantic analysis.

In the NLP domain, extracting and aggregating justifications for sentiment from data is a new area of research. Correlating mined justifications to future stock returns and comparing sentiment from mined justifications to full documents are also novel concepts. This thesis attempts to address both of these areas.

5.3 Justification Mining, Argument Mining and Opinion Mining

One of the more recent intersections of Machine Learning and NLP is opinion/argument mining (Pang and Lee, 2008; Brandseye, 2017; Naderi and Hirst, 2016; Kenter et al., 2016). For this thesis, I intend to utilise the principles of argument mining as a starting point to create a methodology for mining representative sentences capable of capturing the sentiment of an entire document. As such, my focus is on mining what I call ‘justifications’ rather than ‘arguments’. They are related but not identical tasks and utilise the same kinds of methods and algorithms. While justifications could be part of arguments, they need not be argumentative in nature. Rather, the term justification is applied here as an attempt to refer to the reasons behind why sentiment is the way it is, in this case sentiment derived from financial text data. Thus, in finance for instance, justification mining could attempt to understand or explain why a specific investment strategy was recommended by a sentiment analysis model. In this way, justification mining is very similar to opinion mining, which attempts to realise the reasons behind the way people feel (Pang and Lee, 2008; Brandseye, 2017). Again, however, while justifications could contain opinions, they may not necessarily relate to how people feel, as many justifications could be entirely fact-based. Justification mining is a term I created in order to best

describe the type of information being mined, as it does not adequately fit into the fields of argument or opinion mining. The term 'justification' is more accurate in the context of this thesis.

In the first step of the methodology, justification mining could be performed using word or sentence embeddings with methods such as skip-thoughts or Siamese CBOW (Continuous Bag-of-Words), which are often utilised for state-of-the-art argument mining (Naderi and Hirst, 2016; Kenter et al., 2016). Instead, however, I will turn to newer techniques that involve transformer models to extract sentence embeddings (Reimers and Gurevych, 2019). These methods turn the words or sentences into numerical vectors to allow for comparisons of semantics and meaning and offer a computational benefit over using one-hot encoding or other sparse word representations (Goldberg, 2017; Brownlee, 2017b). Different aspects of a word or sentence are associated by a real-valued feature vector in a vector space. The main benefit is that the number of these features is much lower than the entire vocabulary size (Bengio et al., 2003). Other methods for deriving word and sentence embeddings involve utilising existing available tools such as word2vec and GloVe (Heuer, 2015; Pennington et al., 2014). Once sentences embeddings are extracted, embeddings with similar themes can be clustered using clustering algorithms, such as K-Means or a hierarchical Dirichlet process, the latter of which is popular because it does not require a set number of topics (Batmanghelich et al., 2016). From these clusters, a representative sentence, or justification, can be selected (this methodology will be expanded upon in Section 7).

Sentiment analysis would then be required to correlate mined justifications with future stock returns as well as compare sentiment between mined justifications and full documents. Bi-LSTM (Bidirectional Long Short-Term Memory) is a deep learning model that is state-of-the-art for learning classifiers of text and has been used effectively in literature (Eger et al., 2017; Li et al., 2017). Word embedding methods have successfully improved the efficacy of recurrent neural networks in the field of opinion mining (Irsoy and Cardie, 2014). In this thesis, I utilise instead sentence embedding methods to numericise text in an effort to enhance the efficacy of classifier models. When there is not enough data for a neural net strategy, Bayesian approaches or simple ML models like Random Forests and Support Vector Machines may be utilised instead, or a hybrid of the two (Xie et al., 2019; Cliche, 2017; Fan et al., 2008; Joachims, 1998). With these methods, models are trained on annotated datasets from literature to then predict which justifications and documents indicate a rise or fall in stock prices of the corresponding companies. These techniques offer a unique approach that has not previously been utilised in providing explanation behind sentiment analysis investment strategies in finance.

5.4 Origin and Evolution of RRI

As mentioned previously, the study of RRI literature led to the idea of justification mining. It was conceived as a tool that could potentially help incorporate RRI concepts and improve ethical decision-making in finance by examining the justifications behind sentiment analysis classification. However, due to time restrictions based on the time it took to manually parse 10-K filing data and the time it would take to properly crowdsource end-user annotations (as well as the need to remain within the scope and length of a DPhil thesis), there was simply not enough time to explore these RRI concepts within the actual research investigation. Instead, the idea of using justification mining to develop a tool for incorporating end-user specifications to automatically provide information that could help investors in making more ethically informed decisions is discussed as a potential direction of future

work in Section 11.1. Literature pertaining to RRI and ethics in ML and NLP is still discussed in this section as it provides the foundation and motivation for developing justification mining.

RRI is by no means a novel concept. It is a field that has gone through a complex evolution with many different key researchers and fields of application. Owen et al. (2012) state that the term itself originated in the early 2000s (Guston, 2004, Owen et al., 2009, Owen and Goldberg, 2010, von Schomberg 2011, von Schomberg, 2011b, Lee, 2012 and Armstrong et al., 2012). It can be traced back even further if originating terms such as 'responsible development' are counted. These terms themselves started from discussions about legal, ethical and social implications of fields such as genomics, as well as discussions of collaborations between natural, social and physical sciences such as in the fifth and sixth EU Framework Programmes (Owen et al., 2012). Indeed, the origins of RRI in EU policy discourse can be traced back to the fifth and sixth EU Framework Programmes (Angelaki, 2016). Owen et al. (2012) also discuss other early roots of RRI, including technology assessment (Schot and Rip, 1996) and anticipatory governance (Karinen and Guston, 2010).

In 2011, a reflection group workshop was held by the Directorate-General for Research with the goal of developing a combined understanding of RRI and creating policy recommendations in the European Research Area (Angelaki, 2016). May of 2011 marked the first statement in public of the importance of RRI in new EU policy by the Directorate-General for Research. Only a week later, at another international workshop, the EC also outlined concrete intentions for RRI development and utilisation (Owen et al., 2012).

Owen et al. (2012) believe that a key factor in the growth of RRI could be the recognition of the shortcomings of existing policies to oversee ethically problematic areas such as genetically modified foods (Grove-White et al., 2000) and information and communication technology (von Schomberg, 2011b). Jirotka et al. (2017) discuss the various fields of application for RRI during its history and evolution, which first began with the goal of evaluating novel areas of research, starting with fields such as nanotechnology (Floridi, 2010) then moving forward to health and environmental sciences such as geoengineering (Stilgoe et al., 2013) and synthetic biology (Tucker and Zilinskas, 2006). More recently, Jirotka et al. (2017) state that the field of RRI has grown to incorporate areas such as robotics, informatics, computer science and information and communications technology (ICT) (Grimpe, Hartswood and Jirotka, 2014, Jirotka et al., 2017). From discussions in areas such as Robotics and ICT, machine learning becomes a natural stepping stone in which to discuss RRI. Of course, the story of RRI evolution is by no means complete, as the field continues to grow and evolve.

5.5 Definition of Responsible Research and Innovation

Responsible Research and Innovation (RRI) is an area that has not yet been associated extensively with intelligent algorithms and machine learning and certainly not with justification, argument and opinion mining. At the moment, similar research focuses on physical robots or machine learning in the context of quantum computing (McBride and Stahl, 2014; Jirotka and Inglesant, 2016). RRI concerns, however, are crucial for several reasons, including the evaluation of the impact of technology and advancements on society as the tools utilized in computing become more sophisticated, leading to a greater need to evaluate what measures should be taken to ensure that this innovation matches the larger goals of society. RRI is not intended to create new ways to oversee

research and innovation but instead focuses on expanding the reach and importance of existing principles and activities by linking them (Jirotka et al., 2017).

Proper use of RRI could lead to better implementation of policies and an increase in acceptance of the research process and system. Much of the current literature on machine learning and natural language processing in this area focuses on ethics and social implications (Anderson and Anderson, 2011; Burton, 2016; Burton, 2016b; Schnoebelen, 2017; Schnoebelen, 2017b). Responsible research and innovation, however, includes ethics but goes beyond this field in a number of ways. RRI covers a much broader scope, which involves promoting research in a way that is better for the overall advancement of society. RRI requires a widening of scope to incorporate the governance of innovation itself (Jirotka et al., 2017). In the European Commission's RRI Framework, ethics is one of the six main "*keys*" of RRI, with other keys including areas such as science education and gender equality (The Directorate-General for Research and Innovation of the European Commission, 2012; See Section 5.5.2).

The concept of moral appropriateness and ethics is a crucial and complex part in the acceptance of novel technology, although it is only one part. This complexity in ethics is one of the reasons why RRI could provide a better approach for this discussion (Stahl et al., 2014). Moreover, ethics might not be able to provide a language or lexicon that is academically solid and practical (Stahl et al., 2014). RRI allows for new and novel opportunities for incorporating ethical considerations, allowing researchers to see beyond usual boundaries and consider broader societal goals (Stahl et al., 2014). The interdisciplinary nature of ICT, for instance, makes it difficult to have a coherent and sustained discussion of ethics because it combines a number of fields and technologies (Stahl et al., 2014). Similarly, machine learning also has an interdisciplinary foundation.

5.5.1 Definition by von Schomberg

The most basic definition of responsible research and innovation (RRI) comes from von Schomberg (von Schomberg, 2011). RRI is a research process where the societal stakeholders (researchers, policymakers, citizens, businesses, etc.) work together to align the research and its outcomes with the desires and expectations of society. The von Schomberg definition contains three main elements: sustainability, ethical acceptability and social desirability. In this sense, RRI is both broader in scope than corporate responsibility because it evaluates societal requirements but also narrower in scope because it does not look at the entire business cycle, only at research and innovation (Iatridis and Schroeder, 2016). The impact on society is thus taken into account very upstream in the research and innovation process, as is the involvement of society (Stilgoe et al., 2013). A variety of other aspects involving the connection between society and innovation are also inherently explored, including ethics, open access to information, gender equality, public and societal engagement, and formal and informal science education (Science with and for Society, 2014). There are many organizations throughout Europe and the United States that have already begun to focus on RRI, including the United Kingdom's Engineering and Physical Sciences Research Council and the Netherlands Responsible Innovation Program (Owen, 2014).

5.5.2 Policy Agendas – EC RRI Definition

The RRI framework was summarized by the European Commission into six "*keys*". This framework has been the guiding principle for much RRI research (The Directorate-General for Research and Innovation of the European Commission, 2012). The first of these is overall engagement. This calls for RRI to involve the engagement of all societal actors, including researchers, policymakers, citizens, and education and industry officials. Secondly, RRI seeks to promote gender equality and to balance the under-representation of women in research organizations as well as balancing the representation of masculine and feminine values. The third key is ethics. RRI calls for adherence to a standard of ethics beyond just legality but also a general ethical acceptability of new research by society as a whole. The fourth item is open access, which calls for free and early access to scientific research in order to garner innovation and collaboration by all societal actors. Next is science education. This key not only calls for a promotion of scientific vocations to increase the number of researchers but more importantly to boost the overall education process and interest in science, technology and mathematics. The last key point in the policy agenda is governance. This point discusses the responsibility of policymakers in preventing unethical directions as well as handling variable outcomes in research and innovation (Geoghegan-Quinn, 2012).

5.5.3 Framework for RRI – Stilgoe Definition of RRI

To research these outcomes, the process of responsible research and innovation must align itself with certain requirements or dimensions (Stilgoe et al., 2013). There are four that have been identified by current RRI researchers: anticipation, reflexivity, inclusion and responsiveness. Stilgoe et al. (2013) state that this framework is derived from questions often asked of scientists by society. They divide these questions into three categories, product questions such as "*What other impacts can we anticipate?*", process questions such as "*Who is in control?*" or "*Who is taking part?*", and lastly purpose questions such as "*Who will benefit?"* and *"What are they going to gain?*" (Stilgoe et al., 2013, p. 1572). RRI becomes a method of innately embedding these questions into the research and innovation process (Stilgoe et al., 2013).

Anticipation requires researchers to take on new viewpoints and be able to imagine a number of outcomes in order to create socially robust products (van den Hoven et al., 2013). Anticipation is especially important in areas such as nanotechnology, where hype in the present might not match realities in the future, and some kind of responsibility is necessary in discerning between the two (Rip et al., 2012). Some techniques that incorporate anticipation well are Constructive Technology Assessment (Rip et al., 1995), Upstream public engagement (Wilsdon and Willis, 2004) and Real-Time Technology Assessment (Guston and Sarewitz, 2002). Guston (2012) also points out that anticipation might be met with some hesitation as scientists usually seek to preserve their independence.

Reflexivity calls for innovators to evaluate their own ethical, social and political assumptions and biases to allow for a better understanding of public dialogue and the existence of assumptions that may differ from their own (van den Hoven et al., 2013). For example, in many places, public forums on issues in science and innovation are utilized to move beyond the involvement of the typical stakeholders to include a wider public. These forums serve not only to involve the public very upstream in the innovation process but are also to question the authority of "*expert, top-down policy-*

making" (Stilgoe et al, 2013, p. 1571). Callon et al. (2009) created three criteria by which to measure public engagement: i) intensity, how early in the innovation process the public is consulted; ii) openness, the level of diversity of the group and who is represented; and iii) quality, the gravity of the dialogue. Of course, the entire process of inclusion has its own problems as the many participants, from the ones commissioning the public dialogue to the participants themselves, may have varying expectations of the benefits and outcomes, including substantive, normative and instrumental benefits (Stirling, 2008).

Inclusion asks for the involvement of a more diverse group of stakeholders in order to broaden the sources of expertise and allow for novel views that may not otherwise have been reached (van den Hoven et al., 2013). In this way it can diversify the basis of perspectives and expertise (Wilsdon and Willis, 2004).

Finally, responsiveness calls for research to be adaptable in order for it to align with ethical or societal requirements (van den Hoven et al., 2013). Responsiveness goes hand in hand with diversity. Diverse systems are by their very nature responsive (Stirling, 2007) and thus RRI must not only include diversity but garner and build it (Stilgoe et al., 2013).

5.5.4 Definition by van den Hoven

Another definition of RRI was discussed by van den Hoven et al. (2013). RRI is a means for researchers and innovators to consider the consequences of their actions at an early stage, realise the options available to them and evaluate these options with reference to moral values. They can then utilise these aspects in the further design and adaptation of novel research (van den Hoven et al., 2013). The key aspects of this definition of RRI are ethical acceptability and a focus on societal needs (van den Hoven et al., 2013). Like others (Iatridis and Schroeder, 2016), van den Hoven et al. (2013) argue that RRI is not an individual effort but is rather an inclusive, collective and system-wide method. This definition, in particular, is used as a basis for the future work method involving employing justification mining to help make decisions in finance that incorporate RRI concepts, such as being more ethically/morally informed (discussed in Section 11.1).

5.5.5 Hermeneutic Approach - Definition by Grunwald

Unlike previous definitions of RRI, Grunwald begins to look at the definition of RRI in the circumstance of very little or no relevant knowledge being available about the future consequences of novel technologies and innovations (Grunwald, 2014). New instruments are required for the evaluation of the future in order to realise the full benefits from visionary communication (Grunwald, 2007). Grunwald (2014) suggests taking a hermeneutic approach in defining RRI for new technologies and innovations in science, as well as in evaluations of the ethical, social and legal implications of these innovations (Grunwald, 2014). This definition comes into practice in ascribing meaning to the emerging field of quantum computing (Coenen and Grunwald, 2017). Coenen and Grunwald (2017) evaluate initial RRI discourse in quantum computing, which they state is particularly significant because it is at this stage that meaning is given to a novel innovation.

5.5.6 Survey of RRI definitions by Sutcliffe

Sutcliffe (2011) surveys current RRI literature and summarises the evolving definitions of RRI into five key points. The first point states that societal and environmental benefits are a deliberate focus of research. The second underscores the involvement of societal overall. The third emphasises focusing on ethical, social and environmental impacts of novel innovations along with the commercial and technical benefits. The fourth point states the need for regulatory mechanisms that are adaptable. The last key point emphasises transparency and openness in the innovation method (Sutcliffe, 2011).

5.5.7 Goals of RRI

It is important to discuss the nomenclature of RRI in reference to its goals. By calling this branch of research and innovation 'responsible' it is separated from the rest of research and innovation. In a discussion about ethical consumerism and whether the fashion industry is environmentally friendly, one designer asks why terms like ethically correct fashion, green fashion, fair fashion or eco-fashion should have to be created. This designer argues that upholding standards in fair pay and labour, safe materials, and environmental soundness of fabrics should be the standard rather than require a special term to set it apart (Iatridis and Schroeder, 2016). Of course, while this is an interesting debate in theory and does have some merit, it would in practice take a long time for the ideals of RRI to become the baseline of standards and involve a large amount of change in education and the way research is approached. Creating a tool to automatically increase ethical consideration in financial decision-making would represent a significant step in that direction (see Section 11.1).

Based on these definitions, RRI researchers have created three overall outcome goals, the first of these being outcomes related to learning (Geoghegan-Quinn, 2012). Responsible research and innovation should lead to an overall increase in education, empowerment and responsibility among all societal actors (Geoghegan-Quinn, 2012). The second goal relates to the outcomes of the research and innovation itself. RRI should lead to outcomes that are not only ethically acceptable but also sustainable and in line with the desires of society (Stilgoe et al., 2013). Lastly, and perhaps most importantly, RRI should be utilized to address some of the 'Grand Challenges' of society, such as energy and climate issues (Guston, 2012).

Although the scope of innovation can be large, especially innovation that addresses ethical and societal concerns, RRI should not be seen as an impossible task. RRI does not call for prediction but rather anticipation and robust and adaptable innovation (Jirotka and Inglesant, 2016). The NQIT (Networked Quantum Information Technologies) 2016 policy brief for RRI in quantum computing underscores this point by attempting to break down the gargantuan task of evaluating RRI concerns in such a complex area by creating a set of concrete recommendations (Jirotka and Inglesant, 2016). It is part of the UK National Quantum Technology Programme, which strives to develop a quantum computer demonstrator and realise this novel sector of technology (*Networked Quantum Information Technologies* n.d.; *UK National Quantum Technologies Programme,* 2013).

One of the methods for accomplishing these goals is the encouragement of partnerships between social scientists and other specialists, thus taking the burden off scientists alone (Jirotka and Inglesant, 2016; Innova Integra, 2017). The method described for future work (see Section 11.1) incorporates

this concept by involving decisionmakers at financial firms in the annotation process. While it is important to discuss the goals of RRI, there has also been much discussion of its value and necessity (van den Hoven et al., 2013; García et al., 2016; McBride and Stahl, 2014; Owen et al., 2013). There have certainly been a number of unsuccessful (van den Hoven et al., 2013; Gaskell et al., 2010) and successful (Schuurbiers, 2014; van den Hoven et al., 2013; Iatridis and Schroeder, 2016; García et al., 2016; The SPICE Project, 2017) novel innovations as a result of RRI, as well as discussions of areas where RRI would be of value (McBride and Stahl, 2014) to make a strong argument for the necessity of RRI.

5.5.8 Comparing and Evaluating the Definitions of RRI

In this section, six principal definitions for RRI have been presented: the early RRI definition by von Schomberg (2011); the definition created by the European Commission summarized into six policy agendas (The Directorate-General for Research and Innovation of the European Commission, 2012); the four dimensions of RRI presented by Stilgoe et al. (2013); the definition by van den Hoven et al. (2013); the hermeneutic approach by Grunwald (2007; 2014; Coenen and Grunwald 2017); and the survey of RRI definitions by Sutcliffe (2011). The von Schomberg definition provides a good starting point and is easy to comprehend. It allows one to begin to see why RRI is important in many areas of research. Stilgoe et al.'s framework provides some clear benefits when evaluating technical fields. It creates a guideline for conducting research that allows researchers to expand their focus beyond just their individual area of expertise as well as incorporating input from a number of societal actors not often taken into account. The hermeneutic approach by Grunwald and Coenen creates a novel method for situations where it is difficult to realise the future consequences of innovations. The survey by Sutcliffe provides a good summary of the key points from various definitions of RRI throughout the literature.

For evaluating the creation of practical tools that relate to RRI concerns, the EC's six policy agendas and the definition by van den Hoven are the most useful. The European Commission's policy agendas provide a solid framework for considering RRI with the goals of policymakers and practical applications in mind and focus on macro ideas such as science education, gender equality, ethics, engaging the public and open access to research to promote discovery and innovation. These points are excellent tools for driving research in ways that can benefit society and lead to very feasible and tangible agenda goals for policymakers and industry members. The definition by van den Hoven emphasises ethical acceptability, which, as previously mentioned, plays a focal role in the methodology outlined in Section 11.1 for future research.

All the definitions have this in common: a process and outcome dimension (Iatridis and Schroeder, 2016). The process dimension specifies that responsibility is a collective effort and outcome refers to RRI focusing not on what is ethically and socially unacceptable but instead on what creates public value and benefit (Iatridis and Schroeder, 2016). RRI should also be seen as a meta-responsibility, in other words a step above individual responsibility. Framing RRI in this manner sets it apart from other means of evaluating innovation governance (Stahl, 2013). This view of RRI is compatible with other definitions of RRI, including those put forth by von Schomberg (2011), Sutcliffe (2011) and Grunwald (2007; 2014; Coenen and Grunwald, 2017) (Stahl, 2013). There is, certainly, no single best definition; rather, each has distinct goals and areas in which it excels in reaching those goals. While

one might be better for machine learning and practical tools, another might be more suitable for a different field.

5.6 RRI in Robotics

McBride and Stahl use the Penfield Robotics approach to discuss robot engineering methodology. They introduce RRI concepts as early as the first step, identifying a problem (McBride and Stahl, 2014). In the nature of defining a problem, they bring up a set of questions that would logically follow: "*Who sees it as a problem? What is the goal of the robot engineering? For whom is that goal being pursued? What are the social and economic goals? What is the overall purpose of the engineering process?*" (McBride and Stahl, 2014, p. 2). It is clear that taking this approach allows the engineers to naturally bring an ethical and social environment into play in the research and points towards the need for an ethical agenda (McBride and Stahl, 2014). The entire robot engineering methodology they outline involves ethical and social considerations from start to finish (McBride and Stahl, 2014).

The field of robotics engineering involves the utilisation of machine learning, and thus is an important field to review for this thesis. Care must be taken, however, not to assume that machine learning and robotics are the same; while robotics does require the utilisation of intelligent algorithms, there are numerous considerations that are only important to the manufacturing of a physical robot that would not be important in general machine learning, where algorithms are often written without the goal of any kind of physical manufacturing (McBride and Stahl, 2014). However, there are still many techniques that can be applicable in both, namely the embedding of RRI considerations in almost every step of the design rather than just something to be considered at the end.

5.7 Responsibility, Culpability and Ethics in Machine Learning

A large theme discussed within the scope of RRI and new technologies, particularly applicable in the field of machine learning, is the concept of culpability (Stilgoe, 2017; Stilgoe, J., 2017b; McBride and Stahl, 2014; Stahl et al., 2014). It is not enough to understand why it is important but also both whose responsibility it is to address RRI and culpability when things do not work out. Certainly, it is clear how such a topic could be important in the field of machine learning, where there are many human and computer actors. However, it is probably even more important in other innovations brought about by machine learning, perhaps none more than self-driving cars. Such a concept has been lighting up imaginations in science fiction for many years, with examples as early as "*The Living Machine*" by David Keller in 1935 (Keller, 1935).

Stilgoe (2017) references the case of Joshua Brown, who hit a truck in his Tesla Model S at 74mph because the car could not differentiate between the white truck and the bright white sky behind it. Stilgoe emphasises that the situation underscores the complexity of new technologies but also the gaps in governance (Stilgoe, 2017). He states that self-driving cars would be one of the most disruptive new technologies of the early 21st century (Stilgoe, 2017). Certainly, the technology has many benefits, including safety, energy concerns, and congestion. Stilgoe argues though that the benefits must be counted at the same time as addressing the issues by improving governance, an area

which is often discussed only in hindsight. RRI strives to address these issues by embedding them directly into the innovation process (Stilgoe, 2017).

In a technology like self-driving cars, which involves machine learning, a large part of the learning must occur in the intended environment, in this instance out on the road. Tesla calls this 'fleet learning' (Stilgoe, 2017). Consequently, much of the governance and RRI discussion has been limited. Stilgoe argues for a change in the way researchers approach governance and RRI discussions in fields like these (Stilgoe, 2017). However, the question remains as how best to do so when even the final release of a technology is still learning and improving. Champions of self-driving cars - and there are many as a plethora of other companies including Toyota, Volva, BMW, Audi, Google, Intel, Ford, Nissan and even Uber race to begin what could be called a revolution - could argue that these technologies cannot be fully judged until they have the opportunity to learn on the field. But this then raises many ethical questions as researchers and policymakers weigh the issues that could arise during this learning with the potential and vast benefits a world of self-driving cars could bring. Former U.S. President Barack Obama stated in an interview, "*What are the values that we're going to embed in the cars… and how do we make the public comfortable with it?*" (Obama, 2016). He talks about swerving into a wall to avoid a pedestrian (an iteration of the classic trolley problem: Jarvis Thomson, 1985), and how one can allow a machine to make such a moral decision, calling it not just a pure utilitarian decision (Obama, 2016). Certainly, as a topic that could impact many people, self-driving cars have been an important part of the social and ethical discussions as well as RRI discussions in the field of machine learning (Stilgoe, 2017; Stilgoe, J., 2017b).

Difficulty arises when intelligent algorithms begin to alter their code in response to new information, often information taken from increasingly complex systems that are unfeasible to fully comprehend as they involve the interaction of countless machine and human actors. Often these algorithms are even "*black boxes*" in cases like nonparametric machine learning, where the form and parameters of the algorithm are too complex to model or detail. Furthermore, there is the issue of the adaptability of these algorithms, and how they might alter or be altered during practical use in industry.

Ethics research could be approaching its limit in reference to innovations in computer technologies. "*Traditional concepts of responsibility that focus on the clearly delineated action of the individual are increasingly difficult to apply to the mediated and networked world of computing technologies*" (Stahl et al., 2014, p. 814). This is especially true when the lines of human and machine actors blur in a field like machine learning. These limitations can explain why the RRI discourse is particularly apt when discussing ethics (Stahl et al., 2014). One can even go a step beyond this and look at other stakeholders in novel technologies such as investors or legislators. As industries such as social media grow and with new legislation in the United States under the Trump administration and a Republican-led Congress that permits internet providers to sell customer internet browsing data, privacy has moved to the forefront of concerns in society. Stahl argues that while it is not a novel concern, it has become more important recently due to the new ways of linking data and is therefore a key concern for RRI (Stahl, 2013). Privacy, however, also underscores a lot of the key issues in evaluating and embedding RRI. Privacy further emphasises the importance of legislation and its role in the regulation of RRI. In stark contrast to the United States, for instance, in Europe privacy is recognised as a human right (Stahl, 2013). Van den Hoven et al. (2013) believed that because markets do not do much in the way of incentivising RRI, policy agendas must do so (van den Hoven et al., 2013).

Often it is difficult to assign blame to an individual (Van de Poel, 2015). For instance, in sentiment analysis utilised to estimate merger and acquisition success, there are many different actors present in the collective including the algorithms themselves, their creators/innovators, the management or policymakers that put them in place, the general social mass that propagates the sentiment over social media and perhaps even the social media platform itself. This is not a novel concept, as the problem of many hands is a well-discussed idea (Van de Poel et al., 2012). Furthermore, there is the question of the impact on non-participants, such as employees of the merging firms that are not involved in the automated trading around the merger but whose company equity options might nevertheless be impacted. This becomes even more important if sentiment analysis strategies and other automated trading strategies impact the overall outcome of a merger.

5.8 Natural Language Processing and Ethics

There are a number of debates that often arise when discussing ethical concerns in the area of natural language processing. One of these debates concerns the social and ideological uses of language (Schnoebelen, 2017). To ignore these would amount to ignoring some of the applications of NLP. For instance, many companies use machine learning and NLP to build models to determine approval of loan applications. Sometimes certain individuals are restricted access to credit because of how they speak, or what they look like. In the justice system as well, non-standard dialects can be the reason important testimony is ignored (Rickford and King, 2016). Linguistic profiling is seen in many other areas, including housing (Baugh, 2016). While not always strictly unethical, these issues can be seen as undesirable, especially from an RRI viewpoint. They often arise from the notion that most people who create novel technologies (such as in machine learning and NLP) are from privileged backgrounds. This makes it hard for these technologists to empathise with or even imagine everyone their innovations would impact (Crawford, 2016).

Another issue to examine when discussing machine learning and NLP is the maintenance of the status quo. When systems are already consistent, they tend towards further consistency, often exacerbating what is already there. Every algorithmic or human choice allows the option to disturb the status quo but the overwhelming majority of the time they just reiterate what came before (Schnoebelen, 2017).

When specifically examining justification mining, there are justifications that could be considered concerning, especially in the field of finance. These include: "*Everyone else is doing it and we have to keep up, No one else is doing it and we can lead the pack, It makes money, It's legal, It's inevitable*" (Schnoebelen, 2017, p. 91). Again, these many not necessarily be considered unethical or unlawful, but they could be labelled as problematic justifications (Pope and Vasquez, 2016). Although the law can sometimes promote ethical behaviour, it is usually far behind new ethical dilemmas considered by data scientists, computational linguists and machine learning researchers (Moor, 1985). Justification mining could be a means for bringing these concerns to the attention of decisionmakers in finance (see Section 11.1).

There are two ethical dimensions that can be associated with justification mining, NLP, and machine learning. The first is whether the justifications mined from the financial text data are themselves ethical in nature (as discussed in the previous paragraph). The second refers to the implications of utilising machine learning to make certain decisions, such as financial decisions. It considers the

ethics of using black box ML techniques that could make mistakes. The future work approach outlined in Section 11.1 strives to address the first dimension by permitting decisionmakers to consider the motivations behind sentiment analysis classification via justification mining, thus allowing for an innate RRI aspect to the method (Geoghegan-Quinn, 2012; The Directorate-General for Research and Innovation of the European Commission, 2012). The second dimension is much more complex and likely requires collaboration and consideration by innovators, policymakers and even society as a whole as the fields of machine learning an AI continue to expand.

5.9 Summary of Motivation Behind Development of Justification Mining

Advances in natural language processing and machine learning have allowed financial firms to expand beyond traditional financial data (such as securities prices and tick data) into the area of big data collected from social media and other sources. Companies have begun looking at strategies using tools such as sentiment and semantic analysis along with machine learning, yet the motivations and justifications behind these analyses are not often studied. Machine learning is utilised to automate decision-making tasks in many applications, including finance where it can make predictions of key securities, indices, trends and events in markets and even generate alpha signals. However, many of the latest machine learning technologies are black boxes in nature. Providing some accountability for decisions could have a large impact when evaluating areas such as ethics (a topic included in every definition of RRI) and bias or during a financial crash. For instance, if there is an algorithm that works by aggregating sentiment and it tries to provide advice to policymakers, what if any of that sentiment is justified by reasons that could be considered unethical? An example of this could be utilising sentiment analysis to gauge the approval rating of a new bill, such as congressional re-zoning (i.e., manipulating zoning districts to minimise the impact of minority voters) or immigration bans. Even in cases where sentiment is favourable and the approval rating for such bills could be high, the justification behind the sentiment could be racially motivated and therefore considered unethical.

Some of the justifications, opinions and arguments that could be mined in finance and other areas might include reasoning such as: "*Everyone else is doing it and we have to keep up, No one else is doing it and we can lead the pack, It makes money, It's legal, It's inevitable*" (Schnoebelen, 2017, p. 91). While some of these may not be considered unlawful, they could be classified as problematic justifications (Pope and Vasquez, 2016). Even if the law promotes ethical behaviour, such as the Sarbanes-Oxley Act (United States, 2002), it often lags behind novel ethical concerns that data scientists, computational linguists and machine learning researchers discover (Moor, 1985). While people are often aware of the importance of ethical decision-making in computational finance (Prentice, 2007; Walter, 2016), there is a lack of practical tools to take these aspects into account.

While these ethical considerations are discussed as a direction for future work in Section 11.1, the motivation behind developing justification mining that is principally discussed in this thesis falls into the realm of model interpretability and AI explainability (which will be expanded upon in Section 9.1). In finance, when a strategy does not work, it can be difficult to discover why, given the complexity of financial markets. Moreover, machine learning strategies, such as sentiment analysis, can be combined with more traditional financial features such as rolling average momentum and mean reversion to generate alpha signals. A strategy might work initially but begin to falter as time passes and the financial climate alters. Using justification mining to reveal the motivation behind sentiment might allow for analysts to identify when and why a strategy is not performing as desired and enable

them to take steps to develop not only better financial strategies but more flexible and robust ones. Here the term justification is utilised to mean the explanation behind why a particular ML sentiment classification from financial text is the way it is, or, put another way, why that sentiment is recommending a particular financial investment decision. It is a similar concept to that at play in opinion mining, which tries to understand the reasons behind the way people feel (Pang and Lee, 2008; Brandseye, 2017). Understanding these motivations or justifications, particularly if they are simple in nature, might also allow for a practical tool that can help explain to and convince non-expert investors of financial strategies founded upon ML.

5.10 Literature Review Conclusions

I have outlined some of the current tools and frameworks for ethical decision-making utilising machine learning. General concepts common to most machine learning strategies were introduced along with factors involved in sentiment and semantic analysis. State-of-the-art methods in opinion and argument mining were reviewed along with their application towards justification mining. Three definitions of RRI were evaluated. The origin and evolution of RRI leading to a discussion of RRI in robotics was outlined. Notions of responsibility, culpability and ethics were evaluated. Ethical concerns and difficulties in NLP were reviewed. Finally, how these concepts led to conceiving the idea of justification mining was discussed. The next step is to devise a method for combining the benefits of machine learning and justification mining towards a successful technique for sentiment summarisation (capturing the majority of sentiment in a full document with only a few sentences), correlation to future stock returns, and potentially also consideration of ethics in computational finance decision-making. Further literature review is integrated at relevant parts in the research investigation sections that follow.

6 Research Investigation - Transfer Learning

Section 6 addresses research question 1 and sub-question 1-1, focusing on transfer learning as a means to overcome issues that arise due to scarcity of annotated financial data, where training data is annotated at the sentence-level and testing data is document-level.

The details of compiling a financial text corpus are outlined and the process of annotation discussed. The transfer learning research methodology and investigation is presented in detail, including the following: the preprocessing data steps, the experiment set-up including feature engineering, training the classifier on sentence-annotated data, and finally applying transfer learning to novel data (where the classifier makes predictions on 10-K data). Results are reported and performance measures, such as accuracy and F1 scores, are presented and evaluated. Finally, the causes of misclassification are outlined and discussed.

6.1 Research Methodology Design Overview/Summary

Before discussing the research method in detail, it is beneficial to provide an overview of the research design and organisation of the research investigation. The experiment part of this thesis can be broken down into two tasks. The first task is to perform a sentiment analysis on a suitable financial text corpus, mainly focusing on feature engineering and transfer learning between sentence-annotated training data and document-level testing data (this is covered in Section 6). Next, justification mining is performed and evaluated using various methods (in Section 7). Code was written in Jupyter Notebook using Python 3, as well as in Python *.py* module files imported into the Notebook. Jupyter Notebook allowed for alternating between code and markdown cell blocks, which provided clarity and seemed preferable for academic work. Python 3 allows for the use of several libraries beneficial to data science and machine learning.

Originally, it had been intended that the research investigation would start with developing and exploring justification mining. However, upon beginning research, several issues arose in performing sentiment analysis on the chosen text corpus (10-K SEC filings) due to the scarcity of available annotated financial training data. It became necessary to devise a strategy to train models on data annotated at sentence-length and use these models to predict on large documents (see Sections 6.3 and 6.4 for details). This raised an interesting question worthy of further research and it was therefore decided to delve further into this issue, making it the focus of this section. These issues and subsequent solutions are discussed throughout Section 6.

6.2 Identifying Stakeholders

There are a number of stakeholders for whom this research could be relevant and important. This includes the engineers at financial firms that utilise machine learning as well as management teams and investors at these companies. Policymakers and academics in the field will also have an interest in the findings. Lastly, the general public should also be considered as stakeholders. "*Ongoing public confidence will be central to realising the benefits that machine learning promises, and continued*

engagement between machine learning researchers and practitioners and the public will be important as the field develops" (The Royal Society, 2017).

6.3 Financial Text Corpus and Annotation

6.3.1 Online Text Corpora Availability

The first major challenge for this thesis was obtaining an appropriate financial text corpus. It needed to be suitable for sentiment analysis, justification mining (i.e. sentence embedding and clustering tasks) and correlation to future stock returns. Primarily, it must contain information that could inform future stock movement as well as be large enough over a long enough period of time so the results of the various analyses performed are statistically significant. Thus, the ideal text corpus would be of document length (as opposed to corpora that are composed of only individual sentences or words). Moreover, to allow for adequate correlation to future returns, it had to be gathered from many different companies across similar time periods (ideally over at least a ten-year period). When initially discussing obtaining data with my supervisor, it was suggested that the best course of action would be to find open-source annotated datasets from other research papers on which classifiers could be trained. This is a method often utilised in machine learning research (Raffel et al., 2019; Munikar et al., 2019; Cer et al., 2019; Xie et al., 2019; Cliche, 2017; Zhang et al., 2019; Farha et al., 2019; Bender, 2012). However, almost all research papers covering financial sentiment analysis utilise annotated data created from Twitter tweets, Stocktwits or news headlines - in other words, data at a sentence-annotated level (Cortis et al., 2017; Bollen et al., 2010; Malo et al., 2013). A few others scraped news agency websites for articles. For example, Sun et al. (2014) utilised data at a document-level, parsed from the Gulf Keystone Petroleum Ltd stock forum database and the Irish financial sentiment dataset (sourced from RTE, The Irish Times, the Irish Independent), originally compiled and annotated by the University College Dublin Machine Learning Group in Brew et al. (2010). However, in these document-level data research papers the datasets were not made public. Even had they been, they would not have been suitable here as both the companies represented in the data and the time intervals over which it was gathered varied too much.

Another paper (Ren et al., 2013) presented a viable option. It utilised 10-K forms that companies in the United States are required to file with the Securities and Exchange Commission (SEC), available to the public via the Electronic Data Gathering, Analysis, and Retrieval (EDGAR) database on the SEC website (https://www.sec.gov/edgar/searchedgar/webusers.htm). These forms provided a number of attractive benefits. They are available from a wide variety of companies, cover the same time periods (released annually), are required by law to be truthful and thus might be more objective than other news sources, are readily available from the SEC website utilising Python's Requests library and can be parsed with the BeautifulSoup library (detailed further in Section 6.3.3 below). However, like the previously mentioned document-level datasets, Ren et al.'s (2013) dataset was not made public. In any case, they only annotated sentences from the Management Discussion and Analysis subsection from two firms (Microsoft and Coach) for just two years and so their dataset would not be suitable here as a longer timeframe and more firms is ideally required.

Over and above the aforementioned benefits, choosing 10-K data also allows for this thesis to utilise a scarcely studied (Malo et al., 2013) and potentially vast source of financial information, which has the possibility of creating novel contributions to financial text analysis. Not many published papers use

10-K data, and only a small number of working papers and student theses do so. One of the reasons for this may be that high-quality labelled data is expensive to gather and obtain, especially when it demands specialised expertise in the field to annotate (Ren et al., 2013), which can be difficult to locate (Brew et al., 2010). Furthermore, when viewing financial research as a whole, machine learning and statistical methods in financial sentiment analysis are not widely utilised (Malo et al., 2013). Malo et al. (2013) speculate this could be due to the difficulty in implementing these strategies for practical applications and the lack of good quality training data. Especially in finance, annotated training data is scarce and often reserved only for proprietary use (Malo et al., 2013; O'Hare et al., 2009).These are some of the challenges related to using 10-K data, which relates back to research sub-question 1-1.

However, these difficulties should not be a deterrent to using 10-K data. Loughran and McDonald (2016) argue that 10-K data has the potential to be a great source of financial text data. 10-K filings have also been receiving increasing amounts of attention in other accounting and finance literature (Loughran and McDonald, 2016), and thus should not be ignored as a source of data for financial sentiment analysis or other machine learning tasks. Although it might be apocryphal, famed investor Warren Buffett, chairman and CEO of Berkshire Hathaway, is often cited as making his fortunes based on the information he gathered reading 10-K filings (Root, 2019; MacBride, 2014). He is even quoted as saying "*[he] read 500 pages like this every day. That's how knowledge builds up, like compound interest*" (MacBride, 2014). It stands to reason that successful sentiment analysis along with other NLP tasks on this data could have tremendous benefit to the fields of finance and machine learning.

6.3.2 Sentiment Analysis Using 10-K Forms

While 10-K forms provided a readily available and viable financial text corpus for this type of study, it soon became apparent why there were so few Natural Language Processing and Sentiment Analysis research papers in finance that utilise this data (Cortis et al., 2017; Bollen et al., 2010; Malo et al., 2013; Sun et al., 2014; Brew et al., 2010; Ren et al., 2013; Loughran and McDonald, 2011; Bodnaruk et al., 2015; Loughran and McDonald, 2016; Ball et al., 2012). In this sub-section and the next several, I will outline some of the limitations and challenges of utilising 10-K filings data for ML (research sub-question 1-1) including difficulties related to readability, annotation and parsing.

To start, 10-K forms can be difficult to annotate and require expert-level annotators due to their complexity, which as previously stated can be difficult and costly to obtain (Ren et al., 2013; Brew et al., 2010). This complexity occurs due to companies being required by law to report liabilities, risks and issues in detail to avoid being liable for deceptive reporting (United States Federal Government, 2019). In order to calculate their complexity, readability indices can be utilised (Zhou et al., 2017; Partow-Navid and Ardekani, 1991; Humphreys and Humphreys, 2013). These indices are generally based on two heuristic characteristics: lengthier sentences and longer words are less readable. In order to evaluate the length of words either the number of unique phonemes or syllables can be utilised. For these 10-K filings, two common readability indices were used: Flesch-Kincaid Readability Index and Gunning-Fog Index, given by the following formulae:

Flesch-Kincaid Readability Index: $$0.39\frac{total\ words}{total\ sentences} + 11.8\frac{total\ syllables}{total\ words} - 15.59$$

Gunning-Fog Index: $0.4[\left(\frac{words}{sentences}\right) + 100\left(\frac{complex\ words*}{words}\right)]$

*complex words refer to words with three or more syllables

The results of both indices correspond to the U.S. grade level required to read and understand the document, with a score of 16 corresponding to a College senior and 17 to a College graduate (Zhou et al., 2017; Partow-Navid and Ardekani, 1991; Humphreys and Humphreys, 2013). Thus, complex academic works would typically have scores of 16 or above. In order to gauge the complexity of 10-Ks, filings from Starbucks, Amazon and JP Morgan filed between 2012 and 2017 were compared to several Wikipedia articles in the followed advanced fields: Statistical Thermodynamics, Theoretical Physics, Special Relativity and Stochastic Calculus (Wikipedia Contributor, 2019a-d).

The data was parsed with Python's requests and BeautifulSoup (using the *lxml* parser) libraries. In order to calculate the two indices on the data, a number of preprocessing steps had to be taken (Zhou et al., 2017; Partow-Navid and Ardekani, 1991; Humphreys and Humphreys, 2013). The data was then tokenised and cleaned with Python's nltk library. It was stemmed with *SnowballStemmer*, lemmatised with *WordNetLemmatizer*, word-tokenised with *RegexpTokenizer* (in order to select all alphabets with the following code: r'[^\d\W]+'), and sentence-tokenised with *sent_tokenize* (both were required since both word count and sentence count were necessary). Word syllables were counted using cmudict in the nltk library. These values were utilised to calculate the total word count, sentence count, syllable count and number of complex words. With these values, functions for both the Flesch-Kincaid Readability Index and Gunning-Fog Index were coded and then a for loop was used to calculate the indices of all the gathered documents (Zhou et al., 2017; Partow-Navid and Ardekani, 1991; Humphreys and Humphreys, 2013). The Wikipedia articles resulted in scores between 16 and 20, whereas the 10-K filings' scores ranged between 18 and 22. This provides some indication of the high complexity of 10-K filings and the level of expertise and deep domain knowledge (not only in finance but likely in law and the relevant company industry as well) required to annotate them. This makes popular annotation methods in economics like crowdsourcing (Brew et al., 2010; Malo et al., 2013), particularly with experts (Ren et al., 2013; Brew et al., 2010) difficult and not viable in most circumstances for 10-K data. Brew at al. (2010) had to annotate with non-experts; Malo et al. (2013) did use expert annotators but only on a small dataset of sentence-level material; and Ren et al. (2013) annotated 10-K data but only on one subsection and only over two years for two companies (in other words, a total of four 10-K filings).

Moreover, using pre-packaged parsing tools as well as parsing specific data from 10-K filings can be difficult if not impossible (Loughran and McDonald, 2016). Loughran and McDonald (2016) found that pre-packaged parsing tools like Fathom in Perl or the Talburt method do not work well on 10-K data, which might deter researchers. In a personal email communication with Professor McDonald, he recommended that I should not try to automatically parse subsections of 10-K forms either (McDonald, 2019b). He found that only about 80 percent of the Management Discussion and Analysis subsection could be captured correctly and there were too many tripwires that make it impossible to accurately segment 10-Ks (Loughran and McDonald, 2016; McDonald, 2019b). There are no algorithms that can solve this parsing problem unless code is written for every exception. Moreover, mistakes in automatic parsing also substantially skew performance results (McDonald, 2019b). Commercial vendors might provide this data but it is typically manually parsed by people, is expensive and not likely to have a long history (McDonald, 2019b). This makes obtaining data from commercial vendors unsuitable for this study as a sufficiently long time frame is required.

While these difficulties provided challenging impediments, they were not impossible to overcome. In doing so, this thesis strives to provide more analysis and data on this rarely tapped (in the context of machine learning tasks) and rich source of financial data (Malo et al., 2013; Loughran and McDonald, 2016; Root, 2019; MacBride, 2014). Another approach was required to accomplish the sentiment analysis classification of 10-K filings necessary to evaluate the subsequent justification mining related tasks. As discussed previously, the requirements for the dataset needed in this thesis were many and prevented using any of the publicly available annotated datasets. All the annotated financial datasets that were made public and which I could locate involved annotation at the word- or sentence-level (Malo et al., 2013; Cortis et al., 2017; Bollen et al., 2010; Ren et al., 2013). This led me to wonder if transfer learning methods could be applied in order to train classifiers on this sentence-level data and then make predictions on the document-level 10-K data, which is the matter presented in research question 1. When I discussed this with my supervisor, he mentioned that being able to apply transfer learning strategies from sentence-level data to document-level data could be the subject of a thesis in itself, which piqued my interest. I realised there was an opportunity here to explore methods that could allow further study and utilisation of the vast source of potentially signal-rich and valuable financial information contained within 10-K filings (Malo et al., 2013; Loughran and McDonald, 2016; Root, 2019; MacBride, 2014). Therefore, I expanded what was meant to be an initial small step of my thesis in order to research methods to accomplish this transfer learning task.

In transfer learning, the model is first trained on a data-rich task before it is fine-tuned and utilised on a downstream task. It is a powerful and effective technique in natural language processing (Raffel et al., 2019; Munikar et al., 2019; Cer et al., 2018). Moreover, transfer learning utilising sentence embeddings often outperforms word-level transfer (Cer et al., 2018). Cer et al. (2018) demonstrated good results using transfer learning via sentence embeddings even with minimal quantities of supervised, labelled training data. These findings presented good arguments for attempting classification on document-level data with sentence-level training sets, especially since sentence embeddings would be utilised in justification mining later.

Of the annotated training datasets that were made public and which I could locate (Malo et al., 2013; Cortis et al., 2017; Bollen et al., 2010; Ren et al., 2013), Malo et al.'s (2013) dataset stood out. This dataset was created from English news covering all listed firms in the OMX Helsinki. It contained a mix of small and large cap companies, different news sources as well as different industries. Three labels were utilised: favourable sentiment, unfavourable sentiment and neutral corresponding to a rise, fall or no change in stock price respectively (Malo et al., 2013). Other financial sentiment analysis research papers also tend to favour three class labels over a binary classification (Ren et al., 2013), as a neutral class offers the flexibility of labelling sentences that offer no insight into the companies' underlying performance and stock value. Moreover, Malo et al. (2013) utilised an extensive and impressive annotation method. Annotators approached sentences from the viewpoint of an investor, judging the news by how it could impact stock prices. The dataset was annotated by a total of 16 people with high levels of expertise and background in finance (three researchers and 13 master's students with majors in economics, finance and accounting at Aalto University School of Business) (Malo et al., 2013). Crowd-sourcing annotation work in economics and finance has had much success and benefits (Brew et al., 2010). Thus, this dataset not only had the advantage of being annotated via crowdsourcing but also of being annotated by experts. It was made freely available for academic use and came in four formats, denoting different degrees of accordance between the annotators (ranging from 50 percent agreement to 100). Malo et al. (2013) found similar levels of performance across all four categories of agreement. For this thesis, 75 percent agreement was chosen as it would allow for

an adequate majority while not diminishing the number of sentences too greatly. The final training dataset contains 3453 sentences, as opposed to the original total of 4840 sentences.

6.3.3 Parsing Data

For the training data obtained from Malo et al. (2013), no parsing was required. Data was provided from their link in a *.txt* file. Sentences and sentiment were separated with an '@' delimiter, making for simple reading in of the data in Python. Data was read in and stored in a list object. After this, the sentences (X) were separated into one list and the sentiment (y) into another of equal length.

Parsing the 10-K data proved more difficult. However, a variety of research papers provided detailed methods for doing so (Loughran and McDonald, 2011; Bodnaruk et al., 2015; Loughran and McDonald, 2016; Ball et al., 2012; Ren et al., 2013). Loughran and McDonald (2016) state that pre-packaged parsing tools (like Fathom in Perl or the Talburt method) do not perform well on 10-K filings. Consequently, these types of methods were not utilised for parsing. McDonald (2019a) provides an online software repository that contains a variety of useful information and code for downloading data from the SEC website. It has a variety of Python code that can be utilised to download EDGAR files from the SEC by form type (including 10-K filings). It also contains a large data repository of 10-K and 10-Q filings from 1994-2017, already parsed and cleaned. However, for this study, data from only a few companies, randomly chosen from the S&P 500 index, was parsed. More recent filings are also better in terms of consistency in structure, which is why filings only as far back as 1994 (when available), similar to Professor McDonald's repository data (2019a), were parsed. A total of 17 companies were chosen, ranging from different industries, sizes, and locations to match the characteristics of Malo et al.'s (2013) training data. The names, tickers, date ranges and number of filings for each company can be found in Table 1 below. The overall number of 10-K filings ended up being 326, because not all companies had filings as far back as 1994 and others were missing filings in html format for certain years. The text from entire filings was used rather than segmenting a specific subsection of the 10-K form. As mentioned in Section 6.3.2, parsing subsections from 10-K filings can be problematic (Loughran and McDonald, 2016; McDonald, 2019b).

Using the companies' unique CIK numbers, data was pulled from the SEC website via the corresponding URLs and Python's requests library. Once retrieved, the data was fed into a BeautifulSoup (another Python library) parse tree, using the *lxml* parser, in order to facilitate data preprocessing and removing html tags later. The correct section for 10-K filings was found within the <*DOCUMENT*> html tag, so these sections were separated from the rest using this tag. Within the <*DOCUMENT*> tag, the 10-K filing could be found under the <*TYPE*> tag, which is followed by the word '*10-K*' for all 10-K filings. Python's RegEx library was utilised to find and separate out only the 10-K filings themselves from the entire html file. Unlike the training dataset, the 10-K data was initially put into a Pandas (a Python library) DataFrame to keep track of which company and filing date belonged to which 10-K filing. From there, the actual text of the 10-K filings was separated into a list, similar to the training data, but this time with each item in the list corresponding to the full 10-K document rather than an individual sentence (McDonald, 2019a). At various points during parsing, both the training dataset from Malo et al. (2013) as well as the parsed 10-K data were saved into pickle dumps utilising Python's pickle library.

Table 1: List of companies included in the 10-K dataset. Table includes the ticker symbol, full name, date range parsed and total number of filings per company obtained. Note: Date ranges may not match the number of filings parsed as sometimes earlier dates did not have html full-text formats to parse.

Ticker	Name	Date Range	Total #
AMD	Advanced Micro Devices Inc	1994-2017	20
AMZN	Amazon.com Inc.	1999-2017	17
CHTR	Charter Communications	2000-2017	17
CNP	CenterPoint Energy	2003-2017	15
CVX	Chevron Corp.	1994-2017	21
EIX	Edison Int'l	1994-2017	24
EFX	Equifax Inc.	1995-2017	18
FRT	Federal Realty Investment Trust	1994-2017	19
GM	General Motors	2010-2017	8
GPS	Gap Inc.	1994-2017	22
HON	Honeywell Int'l Inc.	1995-2017	20
HSY	The Hershey Company	1994-2017	23
JPM	JPMorgan Chase & Co.	1994-2017	18
JWN	Nordstrom	1994-2017	24
MAR	Marriott Int'l.	1998-2017	18
SBUX	Starbucks Corp.	1996-2017	21
WELL	Welltower Inc.	1995-2017	21

6.3.4 Annotating 10-K Forms

While crowdsourcing has been shown to have great success in economic text annotation (Brew et al., 2010; Malo et al., 2013), it was not a viable option for the annotation of a large corpus of 10-K filings. As mentioned in Sections 6.3.1 and 6.3.2, these filings are complex and would likely require annotation by experts with deep domain knowledge. Brew et al. (2010) had to utilise non-experts to annotate their large corpus of economic news text while Malo et al. (2013) did use expert annotators but only for a small collection of 4840 sentences from financial news sources. For this thesis, the process of securing expert annotators for such a large annotation task proved to be too costly. Thus, manual annotation was utilised in combination with other metrics (TFIDF and cosine similarity) that will be outlined later in this subsection.

Very limited efforts have been made to build annotated corpora encompassing the finance and economics fields (Malo et al., 2013). To the best of my knowledge, none of the existing financial datasets have document-level annotations for any type of document (news, government filings, etc). Aside from gauging the effectiveness of the transfer learning employed in this study, annotating the 10-K data collected provides a contribution to the availability of financial data for machine learning and NLP tasks. To begin the annotation process here, guidance was taken from the annotation method proposed by Malo et al. (2013). Annotation was approached from the viewpoint of an investor, focusing on whether the information contained within the 10-K filing might influence a rise, drop or no change in the price of the company's equity/stock (Malo et al., 2013). Neutral labels were given in two circumstances: either the document was not relevant from the perspective of an investor or it indicated both favourable and unfavourable sentiment equally. The following guidelines from Wiebe

et al. (2005) were also taken into consideration: There are no fixed rules on the annotation of certain words, prior knowledge and speculation should be avoided, and the annotations should be consistent across all tasks.

To make sense of the lengthy and complex 10-K filings, it was decided to put more focus and weight on important subsections. The Management Discussion and Analysis section was likely to be the most important, as it was not only the focus of numerous research papers (Loughran and McDonald 2011; Loughran and McDonald, 2016; Ren et al., 2013; Ball et al., 2012; Bodnaruk et al., 2015) but known to hold valuable information by members of the finance industry (MacBride, 2014). Daniel Kane, member of the three Artisan Partners Asset Management Value Funds (a fund acknowledged for its performance), also suggests the Chairman's letter as an interesting place to begin in evaluating 10-K filings. He also suggests focusing on the footnotes to get a better understanding of a company's debt (MacBride, 2014). W. Lee Norton, senior investment analyst in the portfolio review department of Vanguard Group, believes the risk factors subsection to be one of the most useful in 10-Ks. Norton also emphasised the importance of the properties and legal proceedings subsections (MacBride, 2014). Keeping all this in mind, while reading each 10-K filing, sentiment scores were assigned to each subsection, or sometimes two scores for lengthier or contrasting sentiment subsections. Notes were also taken for each document regarding the relevant information for annotation in case this information might be of use later (such as during the justification mining tasks in Section 7). Each section's sentiment score was also scaled by a value between one and ten, based on either the importance of the section (as discussed above) or, in some cases, if the sentiment was particularly strong or might heavily influence future stock prices. These scaled sentiment scores were then averaged to obtain a final sentiment score for each filing.

Moreover, due to the complexity of this annotation task, it was decided that further statistical calculations could assist in the annotation process. Calculating Term Frequency-Inverse Document Frequency (TFIDF) scores and cosine similarities between sequential TFIDF scores could provide some guidance on whether the annotations were done well (namely, whether annotations matched the information derived from calculating these two metrics). In order to calculate TFIDF scores, the *TfidfVectorizer* function from Python's scikit-learn (sklearn) was utilised. Before utilising the *TfidfVectorizer*, the 10-K data required a few preprocessing steps. Before lemmatisation with NLTK's *WordNetLemmatizer*, html tags were removed with BeautifulSoup's *html.parser*, common abbreviations were replaced with the full words using RegEx and the text was lowercased. Finally, English stop words were removed by downloading NLTK's *stopwords* list (Fan et al., 2008). Most of this was done with for loops and list comprehensions.

Before the *TfidfVectorizer* could be fit to the 10-K filings, a vocabulary was required to be inputted. For this, the popular Loughran-McDonald sentiment lists for finance were utilised (Loughran and McDonald 2011; Bodnaruk et al., 2015; Loughran and McDonald, 2016; Ball et al., 2012). Loughran and McDonald (2011) found that word lists for other fields often misclassify financial text corpora. In their study, almost three-quarters of words classified by the then widely-used Harvard Dictionary were not actually considered negative in the context of finance. Thus, Loughran and McDonald (2011) developed an alternative negative (as well as five other sentiments) word list that better suited classifying financial text. Accordingly, the Loughran-McDonald word lists were downloaded (from McDonald, 2019a) and the same preprocessing actions were applied to them. After fitting the programmed *TfidfVectorizer* on the 10-K data, the scores from each company for each sentiment were

summed. This data was recorded in a Pandas DataFrame, the first five rows of which are presented below in Table 2.

Table 2: First five rows of the results from the TFIDF analysis with Loughran-McDonald word lists on the 10-K data. Numbers are from AMD, the first company listed on the table. The values represent the summation of the scores for each sentiment for each filing. Each row is one 10-K filing, the dates of which are shown in the leftmost column.

AMD	Negative	Positive	Litigious	Uncertainty	Constraining	Interesting
21/02/2017	6.715098	4.369003	4.628768	2.85737	3.483957	2.143152
18/02/2016	6.46062	4.289379	5.087544	2.894389	3.546442	2.161341
19/02/2015	6.295396	4.257435	4.946737	2.85645	3.401976	2.092404
18/02/2014	6.376984	4.253102	4.762886	2.980516	3.387282	2.048123
21/02/2013	6.003314	4.220801	4.300099	3.017173	3.503672	2.14602

Aside from the TFIDF values themselves, another metric that was utilised in order to assist in annotation was the cosine similarities of neighbouring TFIDF vectors. Because the TFIDF values were stored as vectors in a matrix in temporal order, cosine similarities were calculated between neighbouring values/vectors using the *cosine_similarity* function from Python's sklearn library. Quantifying changes to financial documents over time can be useful as drastic changes in these documents from one period to the next could indicate changes in the company or market. This could be indicative of changes to the risk values of the underlying stock. Thus, annotation/sentiment scores from years where there are large changes can be examined more closely. Cosine similarity measures the angle between two document vectors in vector space. It is a measure of orientation not magnitude and is given by the following formula (Al-Hagery, 2016):

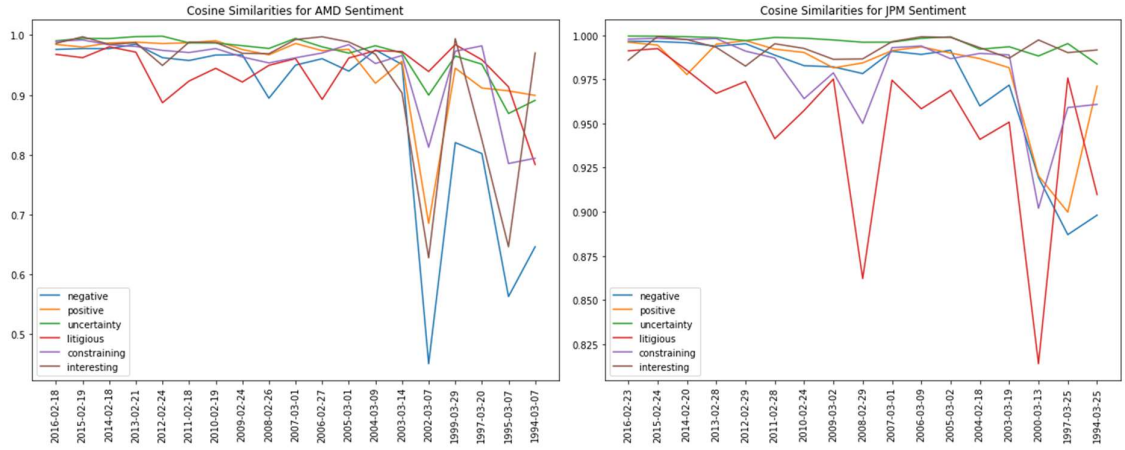
$$similarity = \cos\theta = \frac{X \cdot Y}{|X||Y|}$$

While these similarity values could be utilised as alpha signals themselves, in this case they were utilised to detect any interesting changes to sentiment over time and assist in the annotation process of the 10-K filings. Their values are presented in Table 3 and their graphs were plotted with Python's Matplotlib library. Two of these plots, for AMD and JPM, are shown below in Plot 1 and the rest are presented in Appendix 1 (See Section 14.1). As you can see from Plot 1, there are several points where sentiment values change drastically (particularly in the earlier years), which presents valuable information to assist in the annotation of the 10-Ks from these years.

Table 3: First five rows of the Cosine similarity scores for neighbouring 10-K filings, segmented by company and TFIDF sentiment (from the Loughran-McDonald list).

Ticker	**Date**	**TFIDF Sentiment**	**Cosine Similarity Score**
AMD	2016	Negative	0.97603
AMD	2015	Negative	0.97755
AMD	2014	Negative	0.97743
AMD	2013	Negative	0.98482
AMD	2012	Negative	0.96232

Plot 1: Cosine similarity scores plotted for AMD and JPM. Y-axis represents the cosine similarity score, X-axis is the date of the filing and each coloured line represents a TFIDF sentiment from the Loughran-McDonald list with the key in the bottom left.



After these two additional metrics were calculated, they were compared to the manually annotated sentiment scores to observe any large discrepancies. In the case of drastic discrepancies, the 10-K filing was annotated again and sometimes final sentiment scores were modified accordingly. All these things in combination allowed for a better overall sentiment score for each filing.

6.4 Experiment Design – Transfer Learning

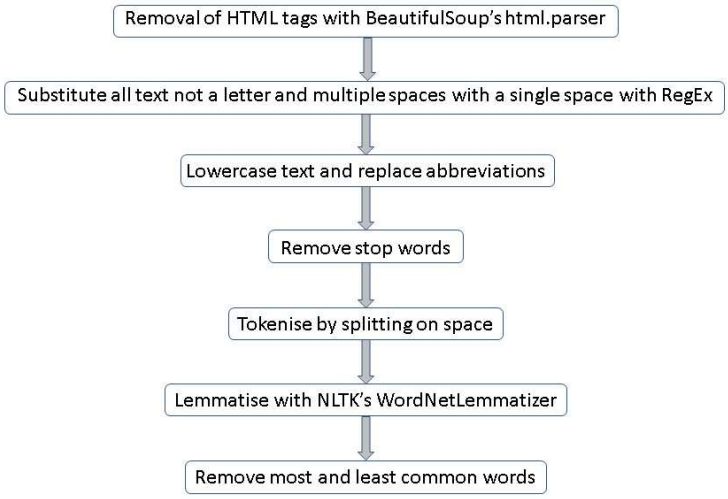
6.4.1 Preprocessing Steps and Feature Engineering

With the data parsed and annotated, it was read into a Jupyter Notebook and stored in list objects, one list for the sentences from the training data (Malo et al., 2013), another for the documents from the 10-K filings (the unprocessed X's for ML algorithms), and two more lists containing the sentiment values for both of these datasets (the y's). The sentiment values were changed to numeric values: 0 for neutral, 1 for favourable sentiment, and 2 for negative sentiment (because negative numbers raise errors in some machine learning models).

From there a number of preprocessing steps were taken in order to get the data ready for training and testing. At various points during the preprocessing and feature engineering steps, data was dumped into pickle files to save time between work sessions. For the sentence-level training data (the sentences from Malo et al., 2013), RegEx was utilised to substitute any character that was not a letter with a space, and all text was lowercased. Abbreviations were replaced with full words as well. The sentences were then tokenised into words by splitting on spaces. They were finally lemmatised using NLTK's *WordNetLemmatizer*. Next, a bag of words from the data was created using a list comprehension and Python's *Counter* function from the collections library. Then, the most and least common words in the bag of words were removed (the least common were removed by a calculated percentage and the five most common words [*'the'*, *'of'*, *'in'*, *'and'*, *'to'*] were removed). Finally, the *enumerate* function in Python was utilised to create a dictionary mapping each unique word to an integer and the features for training were created by replacing each word in the original filtered list of

sentences with its corresponding numeric value. For preprocessing and feature engineering of the 10-K data (testing dataset), the same steps were taken as for the training data, with a few exceptions (as this data was a list of documents rather than list of sentences). In addition to these steps, BeautifulSoup's *html.parser* was utilised to remove html tags from the data. Due to the formatting of 10-K data, there were also many locations with multiple consecutive spaces, thus RegEx was utilised to replace all two or more consecutive spaces with a single space. Due to the size of the 10-K data, stop words from NLTK's *stopwords* library were also removed. Finally, of course, the mapping dictionary/vocabulary from the training data was employed to remove any words from the test set that are not present in the training dataset. Aside from these changes, both datasets followed the same steps, and at the end two tokenised datasets were readied, one at the sentence-level and one at the document-level (Sun et al., 2014; Loughran and McDonald, 2011; Bodnaruk et al., 2015; Loughran and McDonald, 2016; McDonald, 2019a). The preprocessing steps are summarised in Flowchart 1 below.

Flowchart 1: Steps taken for preprocessing, does not include the numericisation steps.



6.4.2 Classifier Models

Models are only trained on the sentence-annotated data from Malo et al. (2013), referred to as the training dataset, and not on any of the annotated 10-K data, referred to as the testing dataset. In Section 6.4.2, five-fold cross-validation is utilised on the training dataset to evaluate the performance of various models and ascertain whether performance similar to that achieved by Malo et al. (2013) is possible. The validation sets for this are generally referred to as the validation sets from the training dataset as they are directly created from within the training data. The goal here is to obtain comparable performance to Malo et al. (2013) before moving on to the transfer learning task. The averaged metrics obtained via this cross-validation can also be used as a basis for evaluating metrics obtained from transfer learning. In Section 6.4.3, the transfer learning method is outlined and models trained on Malo et al. (2013)'s annotated dataset are used to make predictions on the 10-K testing dataset. This 10-K testing dataset is also split into two equal sets. The first, generally referred to as the 10-K validation set, is used to fine-tune the transfer learning methodology and model

hyperparameters. The second, generally referred to as the 10-K testing set, is unseen by the models during fine-tuning and, thus, used to evaluate the performance of the final trained models at the end. In the transfer learning task, the validation and testing sets are obtained from the same data (10-K filings), while the training dataset is obtained from a different source (annotated dataset from Malo et al., 2013).

6.4.2.1 Deep Learning Models

Before deciding on which classifier models to utilise, I began researching current state-of-the-art techniques for text classification, particularly those involving data annotated at the sentence- or document-level.

The BERT and RoBERTa methods have set new standards of performance for sentence-pair regression tasks. These techniques belong to a new type of neural network known as transformer models, which release big pretrained models (trained to produce embeddings) that can be fine-tuned to new machine learning tasks with fairly small amounts of data. They have quickly become state-of-the-art in extracting sentence embeddings (Devlin et al., 2018; Lui et al., 2019) as well as other machine learning and NLP tasks (Munikar et al., 2019; Xie et al., 2019). However, they create massive computational overhead (Reimers and Gurevych, 2019). In fact, it takes approximately 65 hours to compute the required 50 million inference computations to find similar sentence pairs in a corpus of 10,000 sentences. This makes BERT inappropriate for unsupervised machine learning tasks such as clustering (Reimers and Gurevych, 2019). Moreover, they often require a GPU cluster with lots of memory to implement, to which I did not have access.

Even though BERT and other methods have become current state-of-the-art, there is still much evidence for the success of Long Short-Term Memory Recurrent Neural Networks (LSTMs) (Cliche, 2017; Farha and Magdy, 2019) and bidirectional LSTMs (Zhang et al., 2019; Bender, 2012; Katiyar and Cardie, 2016; Eger et al., 2017; Li et al., 2017) in achieving similar performance levels and even outperforming BERT on a variety of tasks, such as fine-grained opinion mining (Zhang et al., 2019), foreign text sentiment analysis (Farha and Magdy, 2019) and Twitter sentiment analysis (Cliche, 2017). Thus, LSTMs and bidirectional LSTMs with Bag of Words numericisation instead of embeddings were viable options for a starting point.

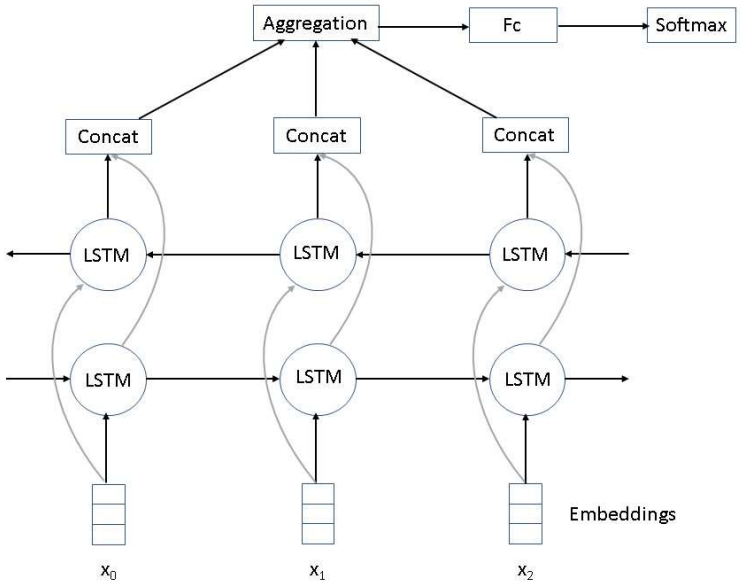
LSTMs are a type of recurrent neural network (RNN) that work well on sequential data. Unlike other kinds of RNNs, LSTMs can create long-term dependencies and apply that knowledge to the present task. Principally, LSTMs deal with the problem of exploding and vanishing gradients. To explain, if the model is backpropagating over an extremely large number of timesteps, the gradient turns into a product of numerical values that are all less than or greater than 1. This causes the gradient to become very large and explode or became very small and vanish (Cliche, 2017; Farha and Magdy, 2019; Katiyar and Cardie, 2016). The LSTM model was stored in a class utilising Python's Pytorch library (specifically the *nn* class). The main class of Pytorch is *torch.tensor*. Its structure consisted of the following layers: an embedding layer, an LSTM layer, a dropout layer, a linear fully connected layer and, lastly, a softmax layer for the final output (since there were three class labels instead of two). An embedding layer is required for mapping to low-dimension, dense vectors. It also changes the shape of the input vectors to be of the three dimensions required by Pytorch for LSTM models (Katiyar and

Cardie, 2016). The dropout layer would help the model generalise to new data, which would be crucial for transfer learning. The initial weights of the hidden layers were set to 0, and both the output and hidden state were returned during the forward pass. A *dataloader* function was utilised in order to batch the data and labels, change them to the appropriate tensor format and modify their length to be equal, left-padding the shorter sentences with 0s and truncating longer sentences (Cliche, 2017; Farha and Magdy, 2019; Katiyar and Cardie, 2016).

Bidirectional LSTMs are a type of LSTM that can further improve performance for sequence classification tasks, which are very typical in finance. There are several options in Python for implementing bidirectional LSTMs. The Keras, TFLearn and Pytorch libraries all have bidirectional LSTM implementations. In bidirectional LSTMs, the outputs of both the normal RNN and reverse RNN must be concatenated, thus it is twice the size of the input embedding size. In the Pytorch library, once the outputs are concatenated, this aggregation is then fed back into the subsequent dense neural network. For Keras and TFLearn, in the default setting only the last sequence output is returned, so the parameter *return_seq* (TFLearn) or *return_sequences* (Keras) must be set to *True* in the initial LSTM layer(s). In TFLearn, the sequence must then be passed to an aggregation layer. TFLearn has a *Merge* layer implementation for this. In Keras, there is instead a bidirectional wrapper, where you can specify the merge mode as either a summation, multiplication, concatenation or averaging (Katiyar and Cardie, 2016; Bender, 2012).

For this study, as with the regular LSTM implementation, Pytorch was utilised to implement a bidirectional LSTM class. The basic structure remained the same as the LSTM model described above, with some modifications. First, *bidirectional* was set to *True* within the parameters of the definition of the LSTM layer. The *hidden state* or *LSTM size* parameter was multiplied by two when defining the fully connected layer. In the initialisation of the hidden states to 0s, the *number of layers* parameter was also multiplied by two. During the forward function itself, the output of the LSTM layer was then split utilising *torch.chunk* into two outputs. These outputs were then reshaped, concatenated back together and then fed into the fully connected layer before finally being put through the softmax layer (Katiyar and Cardie, 2016). This process can be summarised by the diagram in Figure 1.

Figure 1: Representation of the bidirectional LSTM model utilised.



Both the LSTM and bidirectional LSTM models were trained over 20 epochs in order to ascertain after how many epochs the models achieved their best accuracy scores on the validation sets. Accuracy scores were calculated by comparing the true labels to the predicted label via Pytorch's *topk* function. All training was also done on GPU (*cuda*). The following hyperparameters were tuned over many attempts in order to obtain the best model: batch size, sequence length, dropout percentage, number of epochs, learning rate, number of LSTM hidden layers, and the sizes of the embedding and LSTM layers. Grid search with five-fold cross-validation on the training dataset was utilised to determine the optimal hyperparameters, using accuracy as the scoring criterion. Due to the softmax output, negative log likelihood loss was utilised for the loss criterion and *Adam* was utilised for the optimiser (Cliche, 2017; Farha and Magdy, 2019; Katiyar and Cardie, 2016; Bender, 2012; Zhang et al., 2019). For both models, a sequence length of 50 and batch size of 20 yielded the best results. A batch size of 20 was chosen because larger batch sizes tend to drastically degrade the quality of the model and its ability to generalise, something that would be important for transfer learning. This inability to generalise well results from large-batch models often converging to sharp minimisers of the training functions (Keskar et al., 2017). The LSTM model achieved its highest accuracy after three epochs (58.7%), while the bidirectional LSTM achieved its highest accuracy after five epochs (59.2%), see Table 5 in the next subsection. These accuracies refer to the averaged accuracy scores of held-out validation sets obtained from five-fold cross-validation on the training dataset.

For the two deep learning models, only accuracy was initially calculated. When it became apparent the accuracy scores for these models were too low, it did not make sense to calculate the remaining performance metrics. Thus, only accuracy scores for the LSTM models are presented in the results tables (Tables 5 to 10).

6.4.2.2 Simple Models

With the lack of success in predicting on 10-Ks using LSTM models, another approach was necessary. Generally, deep learning models require large training datasets to be successful (Xie et al., 2019; Cliche, 2017), another limitation of sentiment analysis using 10-K filings as testing data (research sub-question 1-1). Representing sentences as bag of words and then training a linear classifier is simple and efficient and can work better for sentence classification when training data is limited (Fan et al., 2008; Joachims, 1998). In fact, Malo et al. (2013) primarily utilised Support Vector Machines (SVMs) in their original research with this dataset. Thus, four simpler classification models were chosen to train the data: Decision Trees, Random Forests, Naïve Bayes (Multinomial) and SVMs. Python's scikit-learn (sklearn) library was utilised to train these models, as well as measure performance metrics. The same preprocessing steps were taken; sentences were again made to be of equal length by left-padding and truncating until all had a length of 30. For each of the models, certain hyperparameters were tuned during training using grid search with five-fold cross-validation in order to improve performance. Again, accuracy was employed as the scoring criterion in the grid search. These hyperparameters are listed in Table 4 below. Four metrics of performance were calculated for each training and validation run: accuracy, recall, precision and F1-score. Accuracy is a measure of how often the classifier predicted correctly (i.e., matched the annotated labels). Recall measures how many data points that were of a certain label were actually classified as that label. Precision measures how many data points that were classified a certain label were actually that label. The F1-score is a weighted mean of both recall and precision. Typically, the accuracy and F1-score are the most important performance metrics (Malo et al., 2013). For accuracy, both the training and average cross-validation accuracy scores were calculated and plotted in learning curves, in order to ascertain if the model was overfitting or underfitting and how well it would generalise to new data. Table 5 lists the averaged metrics of the held-out validation sets from the training dataset calculated during five-fold cross-validation using optimal hyperparameters. Although SVMs did not perform well, in contrast to the original Malo et al. (2013) study (likely due to the choice of model, hyperparameters or preprocessing steps), certain classifiers did obtain metrics similar to the original paper. Recall, this was the goal of performing cross-validation on the training data. On this basis, it was decided to begin the transfer learning process. The performance metrics in Table 5 can also serve as a basis for evaluating the metrics obtained in subsequent transfer learning tasks. The scores in Table 5 are obtained from performing training and predictions on the same dataset (all training and validation sets come from the Malo et al., 2013, annotated dataset), and thus, can be a suitable goal for scores to obtain via transfer learning. In Section 6.5, Tables 6 to 10 use a similar table format to report the performance metrics calculated using models trained on Malo et al. (2013) annotated data to instead predict on 10-K filings (testing datasets). These are the crucial metrics for answering research question 1.

Table 4: Classifier hyperparameters that were tuned for the sklearn models using grid search.

Classifier	**Hyperparameters**
Decision Trees	max_depth, min_samples_split, min_samples_leaf, max_leaf_nodes
Random Forests	n_estimators, max_depth, min_samples_split, min_samples_leaf,
Naïve Bayes	alpha, fit_prior
SVM	C, kernel, probability, max_iter, cache_size

Table 5: Averaged performance metrics on validation datasets (sentence-level). Five-fold cross-validation utilised on the training dataset from Malo et al. (2013). Accuracy and F1 scores are most important. Bold values represent the highest score in each metric.

	LSTM	Bidirectional LSTM	Decision Trees	Random Forests	Naïve Bayes	SVM
Accuracy	0.587	0.592	0.753	**0.932**	0.792	0.678
Recall			0.662	**0.795**	0.771	0.672
Precision			**0.734**	0.522	0.719	0.528
F1-Score			0.696	0.630	**0.744**	0.591

6.4.3 Transfer Learning

As discussed in Section 6.3.2, transfer learning can be a powerful tool in NLP tasks (Raffel et al., 2019; Munikar et al., 2019; Cer et al., 2018). Cer et al. (2018) even found that transfer learning at the sentence-level outperformed word-level transfer and worked even with minimal amounts of labelled training data. This provided a promising option in view of the fact that the training data for this study is limited. This subsection discusses the steps taken to apply transfer learning to the 10-K filings. Sections 6.4.3 and 6.5 address research question 1, attempting to research and evaluate ways in which sentence-level annotated training data can make good performance predictions on document-level testing data. Section 6.4.3 focuses on the methodology and steps taken, while Section 6.5 discusses and evaluates all the results.

In order to apply transfer learning, the first major impediment was the disparity in the length of input between the training data (sentences) and the 10-K data (documents). Recall, models are trained on Malo et al. (2013)'s 75% agreement sentence-annotated dataset. There were two solutions to this: change the training data to be approximately the length of the documents with left-padding (Section 6.4.3.1) or predict sentiment on each sentence in a 10-K filing and aggregate the sentiment (Sections 6.4.3.2, 6.4.3.3 and 6.4.3.4). Both methods were attempted. Training was performed using the entire training dataset. In order to not skew the performance metrics, the 10-K testing dataset was split evenly into two sets, with both containing 163 10-K filings. The first set is used as a validation set to evaluate the methodology and further preprocessing steps. The performance on this first set is employed to fine tune the transfer learning methodology and utilises optimised classifier model hyperparameters. This is done so the models remain generalisable and are not simply biased to work well on the chosen 10-K dataset. Tables 6 to 9 in Section 6.5 list performance metrics for this first 10-K validation set. After the last batch of further preprocessing and cleaning steps are evaluated on the 10-K validation set, the final models are trained with all optimal settings and used to make predictions on the second 10-K testing dataset. This second 10-K dataset was previously unseen by the models during fine-tuning and is, therefore, best for ascertaining overall performance. These final performance metrics are presented in Table 10 in Section 6.5 and are key to addressing research question 1.

The coding styles (for loops, list comprehensions, embedded functions, etc), as well as Python libraries utilised in the following processing and cleaning steps, are similar to those utilised in previous subsections. The only major difference is that embedded functions and embedded list comprehensions were more frequent in the aggregation methods (Sections 6.4.3.2, 6.4.3.3 and 6.4.3.4 below) as all preprocessing steps on the 10-K data were applied at the sentence-level there, instead of

document-level, while still maintaining the original structure of the data. Thus, in the interest of length, preprocessing steps are not discussed in detail again.

6.4.3.1 Method 1: Length to 75,000

The first method attempt involved setting the length of the sentence-level training data to be closer to document length. The approximate average document length of the individual filings in the 10-K validation dataset was 75,000. Thus, all the sentences in the training data were left-padded with 0s to be of length 75,000 and all the documents in the first validation 10-K dataset were left-padded and truncated to be this length as well. Although this method did not make much theoretical sense, as it introduced an enormous amount of noise into the training data, it was worth trying. However, as anticipated, the results were disappointing (see Table 6 in Section 6.5). Most of the trained models did no better than random guessing for a three-label classification problem, although Naïve Bayes did surprisingly better than expected. The deep learning models, however, would not even run as the GPU ran out of memory and CUDA-related errors were returned. Trying to run these LSTM models on CPU did not work either. These results are further elaborated upon in Section 6.5 below. Because the performance was generally no better than random guessing, the trained models from this first method were not used to make predictions on the second 10-K set, the testing dataset.

6.4.3.2 Method 2: Aggregation of Sentiment

The second method was to split each individual 10-K filing within the 10-K validation dataset into sentences, thus creating another dimension to this data structure. This was done utilising the *split()* function, and splitting on three types of punctuation (periods, question marks and exclamation marks). The sentences were then classified one at a time with every trained model (simple models and deep learning models). Next, the sentence-level sentiments from each individual 10-K filing were aggregated together with two methods. The mean and mode (using Scipy's *stats* class) of all the sentence-level sentiments in each 10-K filing were calculated and this sentence-aggregated mean or mode was utilised as the final sentiment for the entire document (note: mean values were also rounded with the built-in *round()* function so they could be compared to the annotated labels). This was initially attempted without any further preprocessing (beyond that outlined in Section 6.4.1) to the 10-K validation data or training data, other than splitting the 10-K filings into sentences.

After this initial attempt using the aggregation method, the performance metrics were adequate for a three-label classification problem (see Table 7 in Section 6.5), certainly above the 33 percent theoretically obtained from random guessing. But I felt the metrics could be better, so decided to further examine the scores each model was predicting, both at the aggregated document-level and some at the individual sentence-level. I noticed in some of the models very high levels of neutral labels were predicted and sometimes very low levels of either of the other two label classes. This indicated there was likely something wrong with how the models were making their predictions.

Therefore, it was necessary to re-evaluate the training and 10-K data as well as the processing steps in order to ascertain areas which could be improved. A wide variety of further preprocessing and cleaning steps were undertaken, each performed one at a time with metrics recalculated after each step (in order to determine whether the steps taken improved or worsened the performance of the trained

models). Models were retrained every time the training dataset was modified in any way. In the interest of length, however, only the steps that successfully increased performance of the trained models on 10-K data classification are reported in Sections 6.4.3.3 and 6.4.3.4 below. The steps were also grouped into two larger batches of changes, with performance metrics only being reported after each batch, (rather than results tables for each individual step/change) so as not to clutter the results reporting with overwhelming amounts of tables/data. The performance metrics after these two batches of further processing steps are presented in Tables 8 and 9 in Section 6.5.

6.4.3.3 First Batch of Further Processing Steps

The first set of further processing steps began by removing short sentences of length 10 and below from the 10-K validation dataset. Upon re-evaluating the arrays of sentences in the 10-K data, I noticed most of the shorter sentences were nonsensical. Some of them were often parts of headers or titles, a limitation of the data itself (which relates back to research sub-question 1-1). Others occurred in places where the split by punctuation (periods, question marks and exclamation marks) utilised to segment the 10-K documents into sentences was erroneous for one reason or another, a limitation of the tokenisation method. For instance, in some cases the period was a decimal and not the end of a sentence. Two typical examples of tokenised, short, nonsensical sentences would be: ['10', 'k', '1', 'amd', '12312016x10k'] or ['c']. Removing these sentences lowered the noise in the data and allowed for the signal to better come through, which also seemed to work best when utilising mode-aggregation of the sentence-level sentiments to determine overall document sentiment. Removing these short sentences did not greatly impact the size of the 10-K validation dataset as only a small percentage of sentences had a length of 10 or below.

Other than reassessing the 10-K data, I also decided it was necessary to re-evaluate the training dataset. The process of interrogating training data is not only crucial for building effective models but also for creating ethical ones (Schnoebelen, 2017). Thus, the next further processing step involved rebalancing the original training dataset because I noticed there were twice as many neutral sentences as the other two labels combined. The mean of the three scores (recall the data was labelled as follows: 0 for neutral sentiment, 1 for favourable sentiment and 2 for unfavourable sentiment) was around 0.5, which indicated high levels of neutral sentences. Thus, neutral-labelled sentences were randomly dropped (utilising the *random* method from Python's random library) until the mean was closer to 1 and the number of sentences per sentiment label were closer in value (in other words closer to a 1:1:1 ratio). The rebalanced dataset contained nearly an equal amount of each of the three labels (although not exactly equal as I did not want to remove too much training data). Other processing steps introduced in this batch included the removal of more of the most common and least common words from the 10-K data, as well as removing the most common words of the original training dataset from the 10-K validation dataset. All of this was an effort to reduce the noise in the data and increase the signal, which might alleviate the trained model issues presented previously. While the performance metrics drastically improved after this batch of changes (see Table 8 in Section 6.5), further processing changes were implemented as I continued to experiment with what worked and what did not.

6.4.3.4 Second Batch of Further Processing Steps

The second batch of further processing steps started with increasing the length of the padded/truncated sentences from 30 to 40 (in both the training data and 10-K validation data). This was a result of noticing the sentences in the 10-K documents were generally much longer than the sentences from the training data and truncating them too much might lose significant amounts of valuable information. The biggest gains to overall performance occurred during the last step, however, which was further removing shorter sentences from the 10-K validation data. This time, all sentences that were length 20 or shorter were removed. When I evaluated the data one more time, I noticed that many of the sentences between 10 and 20 length were still filled with nonsensical information that likely occurred due to title and header sections, or errors resulting from the parsing and preprocessing methodology. This last batch of further preprocessing and cleaning steps improved performance of the trained models on the 10-K validation data even more (see Table 9 in Section 6.5).

The final models were then trained and used to make predictions on the second 10-K set, the testing dataset (see Table 10 in Section 6.5). Both the Malo et al. (2013) training dataset and this second 10-K testing dataset are processed using all the corresponding further preprocessing steps outlined in both batches. The optimal model hyperparameters obtained from fine-tuning are also used to train these final classification models. This second 10-K dataset was previously unseen and, thus, not involved in any of the model tuning. The results on the second 10-K testing dataset in Table 10 show performance only slightly behind the metrics calculated on the first 10-K validation set in Table 9, and still generally outperform the metrics calculated after all previous steps (Tables 6 to 8). Moreover, the highest scores in Table 10 are close in value to the highest scores obtained from cross-validation on the training dataset presented in Section 6.4.2 Table 5.

The metrics in Tables 9 and 10 are probably similar because the 10-K validation and testing sets both come from 10-K filings. Recall, unlike typical ML tasks, in this transfer learning task, the training, validation and testing sets are not all created from the same original dataset. Instead, the training set is from Malo et al. (2013), and, therefore, the validation and testing sets are likely more similar to each other than to the training dataset. Unlike cross-validation, the models here are also never directly trained on but only fine-tuned using data from the validation set.

6.5 Experiment Results and Performance Metrics – Transfer Learning

This subsection discusses and evaluates the performance of the various transfer learning methods and steps outlined in the previous subsections, thus, directly addressing research question 1. It also reviews the metrics from cross-validation on the training dataset from Section 6.4.2 Table 5 for comparison. In addition to the simpler ML models, the LSTM models were also trained on the full training dataset and, after, employed to make predictions on the 10-K validation and testing datasets. As with the metrics presented in Sections 6.4.2.1 and 6.4.2.2, for the two deep learning models, only accuracy scores are presented in this subsection. These numbers were calculated in case there were any surprises with how the trained deep learning models performed on the 10-K data, but as seen from Tables 6 to 10, both the LSTM model and bidirectional LSTM model did not perform well compared to the simpler models. Of note, however, bidirectional LSTMs did outperform LSTMs.

The main reason for the poor performances from both deep learning models can be attributed to deep learning models still needing large training datasets to be successful (Xie et al., 2019; Cliche, 2017). Cliche (2017) had to utilise a large amount of unlabelled data in order to pre-train word embeddings successfully. LSTMs were unlikely to work well here but were still worth trying for comparison. As discussed in Section 6.4.2.2, simpler linear classifiers ended up performing better given limited training data (Fan et al., 2008; Joachims, 1998).

Four metrics of performance were recorded for the simpler models in the following tables: accuracy, recall, precision and F1-score. These are defined in Section 6.4.2.2. During cross-validation on the training dataset, random forests achieved the best metrics. However, Naïve Bayes did obtain the highest F1-score. SVMs performed the poorest (see Table 5 in Section 6.4.2.2). In the first transfer learning method (where all data was padded or truncated to a length of 75,000), every method performed poorly and could not be considered better than random guessing (see Table 6). Both LSTM models could not run due to GPU memory constraints. Interestingly though, Naïve Bayes did achieve relatively high accuracy and precision scores. During the initial attempt at the aggregation method, Decision Trees and Random forests outperformed Naïve Bayes and SVMs in both mean- and mode-aggregation. Moreover, in all but one category (recall for Random Forests), mode-aggregation obtained better scores than mean-aggregation (see Table 7). After the first batch of further preprocessing steps (see Section 6.4.3.3), numbers improved but models performed similarly to before: Decision Trees and Random Forests performed the best (with Decision Trees being on top), followed by Naïve Bayes and lastly SVMs (see Table 8). Finally, following the second batch of further preprocessing (see Section 6.4.3.4), Naïve Bayes outperformed all other models in accuracy, precision and F1-score. After that decision trees and random forests achieved second best metrics (with decision trees achieving the highest recall score) and SVMs again lagged behind, although did obtain a relatively high recall score (see Table 9). Tables 6 to 9 represent prediction metrics using the first 10-K set, the validation dataset.

With evaluation on the 10-K validation dataset completed, models were used to make predictions on the second set, the previously unseen 10-K testing dataset, using all the further preprocessing and cleaning steps from both batches along with optimal model hyperparameters from previous fine-tuning. This was done to ensure the models generalised well to new data and were not skewed to work only on my specific 10-K dataset. The results of this final evaluation are presented in Table 10. The models performed similarly to Table 9, but SVMs obtained the highest recall score (although it should be noted the recall scores between Decision Trees and SVMs were close in Tables 9 and 10). As stated previously, the similarity in scores between the two tables is likely because both the 10-K validation set in Table 9 and the 10-K testing set in Table 10 come from the overall 10-K data corpus and in both cases the exact same further preprocessing and cleaning steps are utilised (i.e., all steps from both batches).

The results in Table 10 are the most important metrics for addressing research question 1 as they are calculated on data previously unseen by the trained models. The highest metrics in Table 10 are not only comparable to the highest scores obtained via cross-validation on the training dataset in Section 6.4.2 (Table 5) but also to the highest scores obtained by Malo et al. (2013) in their original study (which does not utilise transfer learning), supporting the efficacy of the transfer learning approach in this thesis.

Out of all the mode-aggregation methods, Naïve Bayes ended up working best, followed by decision trees. Out of the mean-aggregation methods, random forests worked best, followed by Naïve Bayes. However, as previously mentioned, mean-aggregation lagged behind mode-aggregation across nearly all metrics. What is interesting though, is that until the final batch of further processing and cleaning steps, Random Forests and Decision Trees outperformed Naïve Bayes and SVMs in almost every situation and metric.

Even though Malo et al. (2013) utilised SVMs most successfully in their original research with the sentence-annotated dataset, they did not perform very well during the training dataset cross-validation in Section 6.4.2.2 and tended to underperform during the transfer learning as well. This could be due to differences in both the preprocessing methods as well as hyperparameters chosen between their study and mine.

On the other hand, Ren et al. (2013), whose study shared many common features with both Malo et al. (2013) and this thesis, found success with Naïve Bayes. While they did not employ transfer learning and used a small dataset from a limited time range, Ren et al. (2013) did also utilise 10-K data, evaluated at the sentence-level and utilised the same three labels for annotation and classification as this study. Thus, there is prior evidence for Naïve Bayes being successful given these set of circumstances. However, Naïve Bayes also tended to work poorly initially, until much of the noise was removed from the training and 10-K data. The biggest gain to its performance occurred after the removed of sentences of length 20 and shorter from the 10-K validation dataset (see Section 6.4.3.4), after which Naïve Bayes started outperformed all other models in nearly every metric.

Table 6: Performance metrics of trained models on 10-K validation data when both the initial training set (sentence-level) and 10-K data (document-level) are set to length 75,000 (Note: GPU ran out of memory and could not run deep learning models with this method).

	LSTM	Bidirectional LSTM	Decision Trees	Random Forests	Naïve Bayes	SVM
Accuracy	n/a	n/a	0.316	0.319	**0.451**	0.302
Recall			**0.361**	0.010	0.203	0.225
Precision			0.315	0.317	**0.447**	0.356
F1-Score			**0.336**	0.019	0.279	0.276

Table 7: Performance metrics of trained models on 10-K validation data utilising aggregation methods. Since the training data is trained/classified by sentence, 10-K data was split and classified by sentence and then sentiment was aggregated using both mean and mode. This is the first attempt where similar preprocessing methods were used for both the training data and 10-K validation data.

		LSTM	Bidirectional LSTM	Decision Trees	Random Forests	Naïve Bayes	SVM
Mean aggregated method	Accuracy	0.321	0.409	**0.614**	0.573	0.492	0.404
	Recall			**0.552**	0.500	0.528	0.211
	Precision			0.294	**0.457**	0.397	0.405
	F1-Score			0.384	**0.478**	0.453	0.277
Mode aggregated method	Accuracy	0.464	0.471	**0.646**	0.644	0.516	0.421
	Recall			**0.615**	0.475	0.596	0.344
	Precision			0.367	**0.547**	0.411	0.530
	F1-Score			0.460	**0.508**	0.487	0.417

Table 8: Performance metrics of trained models on 10-K validation data utilising aggregation methods after first batch of further preprocessing steps on 10-K data (see Section 6.4.3.3 for details). Mode aggregation outperformed mean aggregation in every metric, thus only mode metrics are reported below. The gap in the metrics between mean and mode increased as further preprocessing and cleaning steps were taken.

	LSTM	Bidirectional LSTM	Decision Trees	Random Forests	Naïve Bayes	SVM
Accuracy	0.577	0.581	**0.801**	0.712	0.675	0.609
Recall			**0.779**	0.577	0.428	0.562
Precision			0.714	**0.835**	0.761	0.333
F1-Score			**0.745**	0.682	0.548	0.418

Table 9: Performance metrics of trained models on 10-K validation data utilising aggregation methods after second batch of further preprocessing steps (see Section 6.4.3.4 for details). These results are calculated from predictions on only the first 10-K validation dataset used for fine-tuning and incorporate both batches of further preprocessing steps. Mode aggregation outperformed mean aggregation in every metric again, thus only mode metrics are reported below. The gap in the metrics between mean and mode increased even further in this evaluation.

	LSTM	Bidirectional LSTM	Decision Trees	Random Forests	Naïve Bayes	SVM
Accuracy	0.624	0.649	0.820	0.800	**0.910**	0.672
Recall			**0.801**	0.627	0.671	0.791
Precision			0.670	0.781	**0.828**	0.511
F1-Score			0.730	0.696	**0.741**	0.621

Table 10: Final performance metrics of trained models on 10-K testing data utilising aggregation methods incorporating all further preprocessing steps from both batches (see Section 6.4.3.4 for details). These results are calculated from predictions on the second 10-K testing dataset, the previously unseen dataset. Mode aggregation outperformed mean aggregation in every metric again, thus only mode metrics are reported below.

	LSTM	Bidirectional LSTM	Decision Trees	Random Forests	Naïve Bayes	SVM
Accuracy	0.601	0.622	0.814	0.782	**0.903**	0.651
Recall			0.734	0.615	0.632	**0.822**
Precision			0.674	0.799	**0.820**	0.403
F1-Score			0.703	0.695	**0.714**	0.541

6.6 Sources of Inaccuracy and Misclassification in Model

After the experiment phase of research was completed, it was decided to conduct an error analysis to determine sources of inaccuracy. The method for doing so was modelled on the error analysis utilised in Malo et al. (2013). A random sample of misclassified 10-K filings was selected from both the 10-K validation and testing datasets. These were then assessed based on the nature of error made by the classifier. To do so, I employed the sectional notes I had taken during the initial annotation. Unlike Malo et al.'s (2013) study, however, I focused more on macro trends since I was evaluating lengthy documents, whereas many of their sources of error could only be applicable at the sentence level.

The first trend I noticed was older 10-K filings were more likely to be misclassified than newer ones. This could be because the older filings were often less structured. In my personal email communication with Professor McDonald, he advised utilising primarily recent filings as they are better in terms of consistency in structure (McDonald, 2019b). Another trend that led to misclassification was that often large events in the company or changes in the company's structure (both in terms of assets and executives) were described in favourable wording. However, no indication was given as to whether such events might raise or lower equity/stock value. Other times, the nature of some events was ambiguous. It could be construed as bad news by some parties and good news by others. In some cases, documents with a large amount of conflicting sentiment across various sections tended to be misclassified as well. Another surprising trend of misclassification arose from the industry of the companies. The models tended to make more mistakes on companies from the Real Estate sector. This could be due to an underrepresentation of this industry in the original training data combined with a high level of industry-specific jargon utilised in the 10-K filings themselves (Malo et al., 2013).

Lastly, 10-K filings with more numeric data tended to be misclassified more often. In their original study, Loughran and McDonald (2011) excluded tables and exhibits as they often contain template language that is not useful in determining tone. Exhibits also typically included a number of negative words (Loughran and McDonald, 2011). However, in some cases, there were instances of such types of information that did not appear within the standard *Table* and *Exhibit* html tags (which is how they are generally removed, see McDonald, 2019a). When there was data similar to table and exhibit information leftover in the 10-K filings even after processing, this also tended to lead to more misclassification.

6.7 Closing Statements – Transfer Learning

Sentiment analysis is important for many reasons, from evaluation in areas such as brand sentiment to prediction in areas like political polls, the stock market, box office sales and even the likelihood of social uprisings. The transfer learning and sentiment aggregation methods outlined in this section have the potential to allow a wider use of sentiment analysis when training data is limited, from a different source than the testing data, and potentially not annotated at the same level as the testing data.

In Section 6, I have answered research question 1, outlining a method for sentiment and semantic analysis and transfer learning between sentence-level training data and document-level testing data, as well as begun to address research sub-question 1-1, presenting many limitations of sentiment analysis using long, document-level 10-K filings (further discussion in Section 10.1). The lack of availability of high quality annotated financial data was discussed and methods to overcome this issue were presented. The parsing and annotation of 10-K filings was outlined. Next, experiment design was summarised. The steps taken to preprocess and engineer features were detailed. Several classifiers were trained and evaluated. These classifiers were utilised for transfer learning on 10-K filings where the methodology was fine-tuned through further preprocessing steps. Performance metrics were presented and evaluated. Finally, the sources of misclassification were discussed.

7 Research Investigation - Justification Mining

The focus of this section is justification mining and addressing research questions 2 and 3. The goal is to pull out or extract the justifications or motivations behind sentiment analysis predictions on a financial document. The annotation method and class labels assigned to these documents is the same as in Section 6. The specifics of parsing and collecting the new dataset are reported. The methods and models utilised in justification mining, including transformer models, Sentence-BERT, and clustering algorithms, are reviewed. The research method for this section is outlined in detail, including the process for justification mining (clustering sentences and obtaining the justifications from each cluster), feature engineering and training the classifiers for the experiment, making predictions on mined justifications and full documents, and finally evaluating justification mining (utilising correlation to future stock returns and similarity tests to full document sentiment scores). Results are then reviewed and performance metrics outlined.

7.1 Data Collection

The first task and major hurdle of Section 7 is the creation of a new dataset, which was a difficult challenge in Section 6 as well. The research investigation in Section 6 showed that entire 10-K filings present several problems as sources of data for NLP and machine learning tasks. They are lengthy, contain too much extraneous information that does not inform the sentiment of the document (thus, contain a high noise to signal ratio), and are difficult to annotate in large amounts. Therefore, it might be better here to choose a particular subsection of each 10-K filing instead. Loughran and McDonald (2016) argue in favour of focusing on one subsection of the 10-K (they recommend the MD&A subsection) rather than the entire filing for economic research. The Management Discussion and Analysis (MD&A) subsection of 10-K filings has often been viewed as one of the most telling parts of a 10-K (Loughran and McDonald 2011; Loughran and McDonald, 2016; Ren et al., 2013; MacBride, 2014; Ball et al., 2012). Head of global equity research at Fidelity Investments, Christopher Bartel states the MD&A is where companies discuss everything from types of risks to financial requirements. It is also where one can understand how a company functions. Regarding this subsection, Bartel states, "I can anticipate how the company starts to perform in different environments, because I know what the levers are" (MacBride, 2014). The MD&A subsection is also believed to be where management is most probable to reveal information through the tone they employ (Loughran and McDonald, 2011). Furthermore, this subsection has often been segmented and utilised in financial research involving textual analysis (Loughran and McDonald, 2011; Bodnaruk et al., 2015; Loughran and McDonald, 2016; Ball et al., 2012; Ren et al., 2013).

While this subsection is impossible to parse with code (McDonald, 2019b), a combination of parsing with code and manual checking could be an adequate solution. Trying to parse the 10-K filings with Python's requests library using a combination of HTML tag searching (with Python's BeautifulSoup library) for the "*p*", "*table*" and "*div*" tags and specifying only the correct subsection title for Management Discussion and Analysis (always under the title "*Item 7*") might be a feasible method to isolate the majority of these subsections. It would also be necessary to exclude instances of length five or greater so only the headers are returned and not cases where the item number appears elsewhere. This dataset could then manually be checked and corrected when inaccurate to create a repository of just the MD&A subsections from the original 326 10-K filings. The code to do this was written in Python 3. While this may be more time-consuming than typical parsing methods, the creation of such

an annotated dataset and a reliable method for creating new similar datasets could provide valuable contributions to financial text analysis and the use of machine learning methods in finance. Moreover, the annotation of these subsections is not too time-consuming with the original notes from annotation of full 10-K filings in Section 6.

However, once this automatic parsing method was put into practice, several key issues arose. To parse the MD&A subsections, I began by trying the method outlined above. As discussed, the idea was to search the BeautifulSoup parse tree for the section item number, which only appears a handful of times and exclude instances of length 5 or more so only the headers would be identified and not instances where the section item number appeared within a sentence in another area. However, as Professor McDonald advised me (see section 6.3.2; McDonald, 2019b), this method ended up proving unfruitful. When it did work, it worked very well, and I was able to automatically parse the entire subsection of text with no errors. However, the process would often breakdown and almost never worked completely while parsing over even a single company's filings. Sometimes it could find the start tag of "*Item 7*" but not the end tag of "*Item 7a*" (between which was the MD&A text). Other times, only the first few most recent years of a particular company would parse but not the rest. In the end, for a dataset of the size I needed, manual parsing proved to be not only much faster, but the only feasible solution. It was simply always quicker to manually parse the data than it was to check which data was incorrectly parsed by the code and fix those mistakes. However, for a larger dataset, this would obviously not be an option as manual parsing was time consuming.

Since it was decided to manually gather the data, there were more options and flexibility in what could be done. Thus, I decided to choose only one of the parts from the MD&A subsection. Upon reviewing this MD&A subsection across multiple years and companies, many of these subsections contained too much uninformative data, accounting data and numerous tables. Most of them did, however, contain an introductory segment that was titled "*Overview*," which summarised all the important points from the MD&A subsection. I felt this "*Overview*" segment (referred to hereafter as the Overview section or subsection) would be a good data source for justification mining. Its length was sufficient to fulfil what was required and the information it contained was also ideal. Manually parsing the MD&A Overview subsections also alleviated some of the sources of misclassification discussed in Section 6 (see Section 6.6). In particular, it would help with issues that arose before such as large amounts of conflicting sentiment across various sections in a single filing (since only one subsection is being used) and various amounts of numeric data left over after automatic parsing (since the data is being manually parsed no tables or exhibits are left in the dataset).

One downside, however, is that manual parsing of data is extremely slow and requires a large time commitment. During initial parsing, I was able to make note of how long each filing needed in terms of time, and in order to complete the dataset, it would take at least approximately two months of dedicated work. Thus, before moving forward with manually parsing the entire dataset, it was necessary to trial justification mining on some of the first gathered pieces of data. As justification mining was a new method devised for this thesis, this would allow me to properly evaluate if it was not only working as intended but also if the conclusions from it were useful or needed to be modified in any way before proceeding. In this way, before dedicating the time to gather the full the dataset, it was possible to assess several aspects, including: whether the "*Overview*" section of the MD&A is indeed a suitable choice; if it is long enough; if the code is working as intended; whether the code runs in a reasonable amount of time (since there is quite a lot of deep learning involved, I anticipated even smaller datasets would take a significant amount of time using CUDA/GPU); etc. Being able to work

on these concerns on a small selection of data might not provide any concrete conclusions towards the efficacy of justification mining, but the goal is simply to adjust and modify the algorithms instead. These initial trials are discussed throughout Sections 7.2 to 7.5. Before that, however, the remaining details of the full data collection process must be outlined.

For Section 7, it was intended for data to be collected from the same firms utilised in Section 6 of this thesis. However, after the first few companies, it became apparent that the format of older filings was inconsistent, not only between different companies but within the same company as well. Thus, I was not able to manually parse Overview subsection data for all 326 filings used in Section 6. Data collection was limited to filings from years that contained an Overview or summary segment within the MD&A subsection of the 10-K. Table 11 shows which companies and how many filings from each were used in the final dataset. Note that Chevron Corp. (CVX) did not have any 10-K filings with an MD&A Overview subsection, so instead Alphabet Inc. (GOOGL) and FleetCor Technologies Inc. (FLT), other firms from the S&P500, were chosen in order to compensate. In the end, 238 total MD&A Overview subsections from 10-K filings were parsed. However, when compiled, this turned out to be more than enough data for the purposes of evaluating justification mining. The data that was collected from each filing included the unique CIK numbers, ticker symbol, date and text from the Overview subsections. Data was organised into a .csv file for ease of use with Pandas and separated with a unique delimiter (.@) so that it could be cleanly read into a Pandas dataframe. As predicted previously, after the initial trials for justification mining were completed, manual parsing and checking of the data ended up taking approximately two and a half months.

Table 11: List of all the firms and how many filings from each firm were utilised in Section 7 of this thesis.

Ticker	**Name**	**Total #**
AMD	Advanced Micro Devices Inc	16
AMZN	Amazon.com Inc.	16
CHTR	Charter Communications	8
CNP	CenterPoint Energy	13
EIX	Edison Int'l	10
EFX	Equifax Inc.	18
FLT	FleetCor Technologies Inc.	10
FRT	Federal Realty Investment Trust	15
GM	General Motors	10
GOOGL	Alphabet Inc.	4
GPS	Gap Inc.	13
HON	Honeywell Int'l Inc.	16
HSY	The Hershey Company	15
JPM	JPMorgan Chase & Co.	16
JWN	Nordstrom	14
MAR	Marriott Int'l.	16
SBUX	Starbucks Corp.	14
WELL	Welltower Inc.	15

7.2 Encoding Sentences into Sentence Embeddings

7.2.1 Encoder-Decoders and the Transformer Model

The process of justification mining involves clustering semantically similar sentences in order to extract those that represent the actual justifications. This is done by first utilising sentence embedding methods. These methods turn a sentence into a numerical vector, allowing for the comparison of sentences that are semantically similar. The distance between these vectors is computed to determine similarities, often with a deep-neural network. To understand this process, it is useful to look at word embeddings. In the field of opinion mining (an area similar to justification mining), word embedding techniques have been utilised successfully to improve the efficacy of recurrent neural networks (Irsoy and Cardie, 2014). Instead of using the full word corpus, assigning vectors allows for a much more computationally feasible approach. The vectors are dense and low-dimensional, which often works better with neural network toolkits. Thus, there is a clear computational benefit to this method (Goldberg, 2017). Each word is associated with a real-valued vector with many dimensions. Different aspects of the word are represented by this feature vector and each word is linked with a point in the vector space. The advantage is that the number of features is much smaller than the vocabulary size (Bengio et al., 2003). An example of utilising such vectors would be to take the word 'duke', subtract the 'male-ness' vector and put in the 'female-ness' vector to arrive at the word 'duchess' or close to it (Mikolov et al., 2013). Some examples of popular word embedding tools include already trained word embeddings such as *word2vec* (in Python's Gensim library) or *GloVe*, both of which are available under a permissive licence for academic use. These word embedding concepts can be expanded to both the sentence- and document-level. Python's Gensim library also has applications for both of these (*sent2vec* and *doc2vec*). However, for this thesis, I turn to more recent, state-of-the-art transformer models to extract sentence embeddings.

Transformer models are expansions of the older encoder-decoder architecture (Vaswani et al., 2017). One of the main purposes of the encoder-decoder architecture was to deal with sequence-to-sequence (Seq2Seq) issues, primarily when the input and output sequences had varying lengths. Typically, the encoder part of an encoder-decoder is a Recurrent Neural Network (LSTMs are a popular choice for this). The decoder, similarly, is also usually an LSTM. The decoder LSTM network's purpose is to read the full target sequence one word at a time and make predictions of the equivalent sequence but offset by one timestep. In this way, based on the previous timestep, the decoder is trained to make a prediction of the subsequent timestep. In NLP, these timesteps are words (Ilya et al., 2014; Costa-jussa et al., 2018; Lusetti et al., 2018). This was one of the things that initially drew me to the encoder-decoder models, as the process sounds very similar to time series prediction on Open-High-Low-Close ticker data.

One of the main problems with encoder-decoder models was that they were unable to memorise particularly lengthy sequences into fixed-sized vectors. This is where attention comes into the picture (Bahdanau et al., 2015). Bahdanau et al. (2015) first presented attention in order to answer the question: While generating a word at timestep *t*, how much attention must be paid to every word that is part of the input sequence? This question helps understand the main insight behind the methodology and theory of attention.

From the concept of attention, Transformer models arose. They are a novel method in Natural Language Processing, created with the goal of processing Seq2Seq tasks and concurrently managing long-range dependencies (Vaswani et al., 2017). The principle behind transformer models is to completely manage these long-term dependencies that occur between input and output sequences by utilising recurrence and attention. Transformer models are comprised of numerous identical encoders and decoders. These identical encoders and decoders are stacked on top of each other to form blocks (Vaswani et al., 2017). One of the more interesting capabilities of Transformer models is predicting next sentences, a process that could potentially also have many applications in financial time series prediction.

7.2.2 BERT and SBERT

Extending the Transformer model and concept of attention even further, Google developed a novel model called BERT, which is a Transformer encoder model that utilises multi-layer bi-directional architecture. The BERT (Bidirectional Encoder Representations from Transformers) model contains a self-attention layer that can execute attention in both directions (similar in concept to bi-directional LSTMs discussed previously). The amount of stacked Transformer model layers varies in BERT, but is typically 12, while for BERT large it is 24. Pairs of sentences are utilised as training data by BERT for pre-training language models (Devlin et al., 2018).

However, BERT does have several notable limitations. In particular, the architecture of BERT does not allow it to perform a variety of tasks. These include unsupervised learning tasks such as clustering, large-scale semantic similarity comparison, as well as semantic searching for the purposes of information retrieval (Reimers and Gurevych, 2019). Many of these limitations arise because BERT has a model structure that does not create independent sentence embeddings, one of its main shortcomings. This, in turn, means BERT faces significant challenges in extracting sentence embeddings. One method researchers used to get around these challenges was to feed single sentences into BERT for the purpose of extracting a fixed-size vector. However, there have been no studies to evaluate whether this methodology is capable of creating semantically meaningful sentence embeddings (Reimers and Gurevych, 2019). Moreover, large computational power, frequently in the form of GPU clusters, is necessary for BERT, making it fairly slow for practical applications when such computational power is unavailable (Reimers and Gurevych, 2019).

In order to address the issues surrounding the time and processing power required by BERT and RoBERTa, a robustly optimised BERT created at Facebook (Devlin et al., 2018; Lui et al., 2019), colleagues of my second supervisor have developed a novel sentence embedding technique called Sentence-BERT (SBERT) that requires considerably (by several orders of magnitude) less time and processing power (Reimers and Gurevych, 2019). While it was originally planned to utilise Python Gensim library's *sent2vec* for extracting sentence embeddings for justification mining, Sentence-BERT offers a newer, state-of-the-art approach with promising results that outperforms other state-of-the-art sentence embedding techniques (Reimers and Gurevych, 2019). Also, as colleagues of my second supervisor, the developers of this method were accessible to me (as opposed to the developers of Gensim), providing an invaluable source of information and discussion about their technique.

Sentence-BERT is "a modification of the pretrained BERT network that use[s] siamese and triplet network structures to derive semantically meaningful sentence embeddings that can be compared using cosine-similarity" (Reimers and Gurevych, 2019, p. 1). With the aim of extracting fixed-size sentence embeddings, SBERT inserts a pooling operation to BERT's output. Reimers and Gurevych (2019) produced siamese and triplet networks (Schroff et al., 2015) for the purpose of updating the model weights and fine-tuning BERT. The goal of doing so, as stated, was to create semantically meaningful sentence embeddings that were capable of being compared with cosine-similarity or other similarity tests. In the siamese network architecture, the weights of two pooled BERT networks are tied. For the Wikipedia triplet network/dataset, two sentences are pulled from the same Wikipedia section and a third (negative) sentence is taken from a different section. The goal of the triplet network structure is to end up with the two embeddings from the same section closer to each other in vector space than to embedding from the third/negative sentence. A loss function is utilised to optimise this goal (Reimers and Gurevych, 2019).

Reimers and Gurevych (2019) went on to briefly evaluate SBERT for classification tasks using SentEval datasets and comparing the results with other popular methods. In their trials, SBERT obtained the best performance in 71% of the tasks. Compared to InferSent and the Universal Sentence Encoder, performance improved by approximately two percentage points. Sentence-BERT is capable of surpassing similar state-of-the-art sentence embedding models (Reimers and Gurevych, 2019).

SBERT also reduces the computational overhead and drastically improves the speed of discovering the most similar sentence pairs (Reimers and Gurevych, 2019). With this approach, Reimers and Gurevych (2019) were able to maintain the accuracy of BERT and RoBERTa while reducing the time of finding the most similar pair down from 65 hours to 5 seconds (Reimers and Gurevych, 2019). The details of this approach have been made available for academic use on the authors' Github, along with tutorials on how to utilise their novel methods. I adapted this code in Python 3 to apply to this study. In addition, they also provide several pretrained models that can be used to extract sentence-level embeddings. Large NLP models are trained to produce these embeddings which can be utilised for other ML tasks (Reimers and Gurevych, 2019). Of these pretrained models, I started with the Wikipedia dataset, which is optimised for subsequent sentence clustering tasks. Training can be fine-tuned by choosing different pre-trained datasets, different pooling methods (such as mean, max, using embeddings from cls tokens, or a combination of these), or different loss functions. The training can be done from scratch or can be continued on one of the pretrained models by utilising another dataset.

As discussed previously, I started by utilising a small sample from the total MD&A Overview data to trial various aspects of the justification mining methodology. This sample contained the MD&A Overview subsection from 2017 Nordstrom 10-K filing. The initial code I wrote takes as input a single Overview subsection (in this case from the Nordstrom 2017 filing) in the form of a string, which is then split into sentences. Next, an embedder is required to encode the sentences into numerical vectors. Sentence-BERT provides two readers for reading in your own training dataset for the embedder. The first requires sentences with labels from pairwise cosine similarity scores and the other reader requires a triplet structure, with two binary classification example sentences and an anchor sentence (as outlined earlier). Both of these would be extremely expensive and/or take a Herculean crowd-sourcing effort to produce on a 10-K dataset at anywhere near the scale of the NLI, Wikipedia, or other datasets provided by Reimers and Gurevych (2019). Therefore, as discussed earlier, instead of training the embedder on a 10-K dataset, I chose the pre-trained Wikipedia dataset

model, titled *'bert-base-wikipedia-sections-mean-tokens'*, and initially used this embedder to encode the sentences (while later adding the NLI dataset pre-trained model as well).

The next three subsections (7.3, 7.4 and 7.5) include analysis of the methodology and results of this initial trial on the 2017 Nordstrom MD&A Overview data.

7.3 Clustering Sentences

After sentences in a document are encoded into embeddings, sentences that are close in the embedding vector space are then clustered. These sentences would ideally express a similar sentiment score and thus a similar justification. While sentence clustering is not a novel technique (Sarkar, 2009; Kotlerman et al., 2012), to the best of my knowledge none of the literature clusters sentence embeddings created using transformer models. Kotlerman et al. (2012) clustered sentences vectorised by turning each word into a weight based on its term frequency while Sarkar (2009) clustered sentences based on their number of matching words and sentence length.

The actual clustering or topic analysis can be performed with a few different available methods. Python's Gensim library, for instance, has options for hierarchical Dirichlet process clustering, while the NLTK, scipy and sklearn libraries have options for K-means, Agglomerative and other clustering models. Hierarchical Dirichlet processes as well as sklearn's MeanShift and DBSCAN models are attractive options because they do not require one to fit in a number of topics/clusters in advance.

These clusters might need to be modified if they are not informative enough or do not make sense. Thus, this process could need altering and repeating to find different justifications. The goal is some kind of summary of all the sentiment in a particular document. To choose a justification, the algorithm begins by choosing a sentence from each cluster with the embedding closest to the centre of the cluster's vector space. This could be done by computing cosine similarity, which calculates the distance between one point in the embedding space to another. Cosine similarity is utilised frequently as a similarity metric between sentences (Sarkar, 2009; Kotlerman, 2012). In this case, the cluster centre represents the embeddings vector of an imaginary sentence that theoretically perfectly exemplifies that cluster. The point closest to the centre can be located by computing cosine similarities between all the points in the cluster and the centre of the cluster. This point or sentence then becomes the actual example of the cluster, potentially representing the justification. However, depending on the outcome after the initial evaluations using the Nordstrom 2017 data, there could be a need to discard sentences which are not of interest. The full outcome and conclusions from these initial trials are discussed later. This step is not only important but also the one that requires the most trial and error as new methods of choosing justifications would need to be compared to assess performances. For instance, if cosine similarities do not yield meaningful justifications, other approaches will have to be tested and evaluated. Perhaps different similarity indices might work better, or entirely new directions could be employed.

I began with agglomerative clustering from the sklearn library because it was the method used by the original authors of Sentence-BERT (Reimers and Gureyvch, 2019). With the number of clusters set to five, the sentences in each cluster were printed. After this, however, I switched to K-Means clustering,

also from the sklearn library. Unlike agglomerative, K-Means works by calculating cluster centres, so it seemed like another good model with which to perform initial evaluations. As with agglomerative, predicted labels could be called with the *labels_* attribute, but unlike agglomerative, K-Means also had a *cluster_centers_* attribute which could be used to get the vectors for the five cluster centres. To get the cluster centres when using agglomerative clustering, I used the mean of the vectors in each cluster.

The model was fit on the corpus embeddings and both the predicted cluster labels and cluster centres were stored in new variables. The sentences in the clusters were again printed using a for loop. The annotated themes/sentiment for each cluster and the differences between the two clustering models (agglomerative and K-Means) are summarised in Table 12 (note: the cluster numbers were reassigned in one model so similar clusters between models would have the same number for ease of comparison). As with transfer learning in Section 6, sentiment was evaluated from the perspective of an investor, i.e., whether the information in the sentence or cluster would cause an increase, decrease or no change in stock or equity value. The overall sentiment for the 2017 Nordstrom MD&A Overview subsection was annotated to be favourable, with most of the sentences indicating an increase in stock price and the rest being neutral. As you can see, in both clustering models, three of the clusters (clusters 2, 3 and 5, which also contained the majority of the sentences in the document) indicated favourable sentiment as well. Furthermore, the clusters of both models were very similar. Clusters 2 and 5, which in my opinion contained the most relevant information for informing stock movement, were exactly the same in both models (both of these clusters are presented in full in Table 13, with the chosen justification sentences in bold and underlined). The third favourable sentiment cluster, cluster 3, was almost identical in both models as well, and while it contained favourable sentiment tones, the information in these sentences was slightly more subjective and not as factual as clusters 2 and 5. The main differences in the models came from the remaining 2 clusters (one of which only contained a single sentence when using K-Means clustering), which covered themes such as generic store information, generic customer experience information, and customer services. This type of generic, neutral information is likely why these two clusters varied between models. It is surmised that this clustering approach is not particularly useful when information has no common theme. This is likely a result from the way the embeddings were creating with Sentence-BERT, which creates semantically meaningful sentence embeddings. If, however, there are no semantic similarities, then the clustering models will likely vary in how they deal with non-thematic clusters. Fortunately, the clusters of interest are the favourable tone clusters, not the neutral ones. A solution to this might be to decrease the number of clusters to four, as this would allow for the uninformative sentences to be in the same cluster, which could then be separated or even discarded. Another solution, as discussed previously, is to utilise the clustering models that do not require specifying the number of clusters in advance.

Table 12: Themes for each cluster in the Nordstrom 2017 MD&A Overview data for both Agglomerative clustering and K-Means clustering models as well as a comparison of the two models.

Cluster #	Agglomerative Model Themes/Sentiment	Comparison with K-Means Model
1	General store info, no information that would inform the movement of stocks; **Neutral sentiment**	Same theme as agglomerative, but only one of the sentences from first model's cluster was in this cluster, rest were split into clusters 3 and 4
2	Primarily sales-related with two sentences about the Nordstrom Rewards programme. All sentences, however, describe some kind of increase or expansion; **Favourable sentiment**	Exact same as cluster from agglomerative model
3	Sentences about improving customer experience, customer offerings and increasing market share; **Favourable sentiment**	Same theme as agglomerative cluster, with almost exactly the same sentences
4	About customer services Nordstrom offers; **Neutral sentiment**	Sentences about what goods Nordstrom offers; Neutral sentiment
5	Covers improvements in the operations side; **Favourable sentiment**	Exact same as cluster from agglomerative model

Table 13: Two most favourable tone clusters from the Agglomerative and K-Means models, which were the same in both models. Note the sentences chosen by the algorithm as the justification/representative sentence are in bold and underscored.

Cluster #	Text contained in Cluster
2	['Our 2016 earnings per diluted share of $2.02, which included a goodwill impairment charge of $1.12, exceeded our outlook of $1.70 to $1.80.', 'We reached record sales of $14.5 billion for the year, reflecting a net sales increase of 2.9% and comparable sales decrease of 0.4% primarily driven by full-line stores.', 'Our expansion into Canada where we currently have five full-line stores, including two that opened last fall, contributed total sales of $300 in 2016.', 'Nordstrom.com sales reached over $2.5 billion, representing approximately 25% of full-price sales.', **'Our off-price business reached $4.5 billion, with growth mainly driven by our online net sales increase of 32% and 21 new store openings.'**, 'Off-price continues to be our largest source of new customers, gaining approximately 6 million in 2016.', 'Our expanded Nordstrom Rewards program, which launched in the second quarter, drove a strong customer response with 3.7 million customers joining through our non-tender offer', 'We ended the year with a total of 7.8 million active Nordstrom Rewards customers.']
5	['Our results were driven by continued operational efficiencies in inventory and expense execution and demonstrated our team's speed and agility in responding to changes in business conditions.', 'We achieved the following milestones in multiple growth areas.', 'Our working capital improvements contributed to the $1.6 billion in operating cash flow and $0.6 billion in free cash flow.', 'In 2016, we made focused efforts to improve our productivity, particularly around our technology, supply chain and marketing.', 'In technology, we increased the productivity of delivering features to enhance the customer experience.', 'In supply chain, we focused on overall profitability by reducing split shipments and editing out less profitable items online.', **'Through these efforts, we made significant progress in improving operational efficiencies, reflected by moderated expense growth of 10% in these three key areas, relative to an annual average of 20% over the past five years.'**]

7.4 Obtaining/Mining Justifications from Clusters

The next step was to develop a method to obtain justifications or representative sentences for each cluster. In this subsection, the details of the technique introduced in Section 7.3 will be presented.

Sarkar (2009) utilised four methods for choosing the representative sentence from each cluster: selecting randomly, selecting the longest sentence, selecting based on sentence closeness to a centroid, and selecting based on thematic importance. While the third option used by Sarkar (2009) sounds similar to the technique presented in Section 7.3 where the sentence closest to the cluster centre is found using cosine similarity, the actual methodology is very different. Because Sarkar (2009) clustered sentences based on number of matching words and sentence length, the centroid was calculated as a collection of words whose weights exceeds a certain term frequency (calculated using log(1+term frequency)). Then Sarkar (2009) summed the weights of these words for each sentence and chose the highest sum as the representative sentence. Because I, instead, clustered semantically meaningful sentence embeddings, it was possible to directly calculate the distance of each embeddings vector to the actual centre of its cluster in vector space using cosine similarity. Thus, this is the method that was utilised to select representative sentences/justifications.

To start, the cosine similarity scores between each sentence embeddings vector and corresponding cluster centre vector needed to be computed. A variety of methods were tried. I first attempted to rearrange the embeddings into their clusters before calculating cosine similarities (with sklearn) between these embeddings and the corresponding cluster centre. I was able to obtain the max cosine score, but it was difficult to keep track of the original sentences with this method. Also, this method required many lists and for loops, which are rather slow and not very pythonic. Another method I thought of trying, because operations over dictionaries are generally faster, was to put the data into key/value pairs in a dictionary and iterating over that dictionary. However, it required multiple values to each key and the organisation ended up being too messy. Finally, I decided a Pandas DataFrame might be best suited to store the different types of information throughout the various operations. I stored initially the sentences, embeddings and labels in a dataframe. After, I tried to calculate cosine similarity over entire columns with the *cluster_centres* list, but this did not work. I also tried to utilise the *iterrows* Pandas function to include the *cluster_centres* into the dataframe but ran into errors there too. Finally, I was able to use a list comprehension to create a new list with the *cluster_centres* for each sentence in order and add that list as a column in the dataframe. From there it was a simple matter of using the *zip* function to calculate the cosine similarity scores between the embeddings column and cluster centres column and store these scores into a new column on the dataframe. Several rows from this final dataframe are presented in DataFrame 1, with the first row corresponding to column labels.

DataFrame 1: Several rows from the Pandas DataFrame compiled during justification mining for Nordstrom 2017 MD&A Overview data. Ellipses indicate rows and data omitted in the interest of space.

Index	Sentences	embeddings	labels	cluster_centres	cosine_similarities_to_centre
0	Nordstrom is a leading fashion specialty retailer offering apparel, shoes, cosmetics and accessories for women, men, young adults and children.	[2.44740415e-02 3.30607951e-01 9.68608186e-02 5.29339015e-02…]	3	[6.20663296e-02 3.40147207e-01 7.99355122e-02 7.43937890e-02…]	0.995056538
1	We offer an extensive selection of high-quality brand-name and private label merchandise through our various channels, including Nordstrom U.S.	[9.33215022e-02 3.49529207e-01 9.45549011e-02 9.34226960e-02…]	3	[6.20663296e-02 3.40147207e-01 7.99355122e-02 7.43937890e-02…]	0.996777456
…	…	…	…	…	…
19	Our strategies around product differentiation include our ongoing efforts to grow limited distribution brands such as Ivy Park, J.Crew and Good American, in addition to our Nordstrom exclusive offering.	[2.60058176e-02 3.46508294e-01 5.87883182e-02 3.06536481e-02…]	2	[2.93034106e-02 3.45817196e-01 6.13398984e-02 5.97825285e-02…]	0.996245148
20	In 2016, we made focused efforts to improve our productivity, particularly around our technology, supply chain and marketing.	[3.92976813e-02 3.46630603e-01 5.66811338e-02 6.98716566e-02…]	4	[-2.54182970e-02 3.42263371e-01 2.72638609e-02 8.03621230e-02…]	0.997267435
…	…	…	…	…	…

The final task was to obtain the highest cosine similarity score for each label and return the sentence corresponding to that score. The first method I tried in order to do this was the *groupby* function sorted by *max*. I used *groupby* on the labels column and initially it seemed to work. It returned a new smaller dataframe with one row for each label, along with one sentence and one cosine similarity score. The cosine score was indeed the max score for each label; however, upon investigating the data, I realised the sentence did not match the sentence from this score. Instead, the sentence from each cluster that began with the alphabetically last letter was returned, likely because it was counted as the equivalent of a "*max*" sentence. It seemed this method was returning the max for each value, including the sentence, rather than just the max cosine similarity score and its accompanying sentence and cluster label. After this, in order to troubleshoot, I tried three other methods on a simple toy/dummy dataframe whose columns and information mimicked the actual dataframe. The first method used the *groupby* and *idxmax* functions to print only the rows with the maximum scores. The second method used the *groupby* and *transform(max)* functions to get the index corresponding to the max scores, from which I could obtain the original sentences from the main dataframe. The last method utilised the *sort_values* function on both labels and cosine scores, dropping duplicates for

labels, in order to create a new, smaller dataframe of just the rows with the max cosine scores. All three methods worked on the toy data, but only the *sort_values* method worked on the actual data. I imagine the first two methods raised errors due to the *numpy* array data structure present in the actual data that is missing from the toy data (some columns in the actual dataframe utilised *numpy* arrays to house data). Using the *sort_values* method, I was able to easily return the correct original sentences with max cosine scores for each cluster. Encoding the embeddings (utilising deep learning models) and other calculations were also surprisingly fast, taking on average only 7 seconds for the entire trial. This aligns with previous discussions regarding Sentence-BERT, which was reported to be orders of magnitude faster than BERT and RoBERTa (Reimers and Gurevych, 2019).

7.5 Methods for Evaluating Justification Mining

Unlike in Section 6, where supervised learning methods were evaluated with annotated data, the goal here is to evaluate justification mining itself and not classification models. Thus, metrics like accuracy and F1 score are not useful. Instead, methods that evaluate unsupervised learning, a core aspect of justification mining, must be developed.

7.5.1 Early Trials and Initial Evaluations

In this subsection, the initial evaluations for the early trials and testing of justification mining are outlined. These initial evaluations are utilised to test and fine-tune the algorithms for justification mining to make sure everything is working as intended before moving on to the full investigation. They are performed on the trial data from the 2017 Nordstrom MD&A Overview subsection. The results from justification mining can be evaluated by looking at each step individually and iteratively. Evaluation must assess whether justification mining meets its intended goals, which include deriving sentences that represent the motivation behind the sentiment classification predictions on a document as well as capture the sentiment of a full document (along with assessing if mined justifications provide other benefits over using full documents). These approaches could also be useful in evaluating how beneficial justification mining is for the purposes of explaining ML investment strategies to laypeople (see Section 9.1 for further discussions of AI explainability). In the next subsection, Section 7.5.2, the final evaluation methods for the full dataset are presented.

For the initial trials on the Nordstrom data, there are several points at which evaluation could be performed. The first point would be to evaluate the embeddings and clustering methods. For example, one form of assessment could be to identify how many of the derived clusters contain a common theme between the sentences of that cluster using manual evaluation. Moreover, if they did have a theme, how many sentences in that cluster do not fit that theme. This might give some indication of how well the clustering worked. However, this evaluation might be skewed or influenced by posterior knowledge so another method might be to pair semantically similar sentences from the original data and evaluate how many of the paired sentences end up in the same cluster. This evaluation method has the added benefit of somewhat mimicking the techniques of Sentence-BERT, which utilise siamese and triplet network structures and calculate cosine similarities between sentence pairs (Reimers and Gurevych, 2019).

In Section 7.3, I considered the themes of each cluster for the Nordstrom data when comparing the K-Means and Agglomerative clustering models. Thus, in this subsection, I will instead focus on a second point of evaluation, assessing justification mining itself, as the goal is to fine-tune and test the justification mining algorithms. Evaluating justification mining will prove more difficult as there is no previous literature in this area to use as a basis. Expanding upon the evaluation methods presented above, one idea is to rank all the sentences in each cluster by how representative they are of the theme for that cluster. As clusters have a variable number of sentences, it would likely be best to set rank number one as the highest score and larger numbers as lower scores. Ideally, justification mining would choose the highest ranked sentence as the representative sentence for each cluster. Thus, the ranks of the sentences chosen by the algorithm could be averaged to evaluate how well justification mining works.

Using K-Means as the clustering model for justification mining on the Nordstrom 2017 MD&A Overview data, five clusters were derived since *k* was set to five. The chosen justifications for each of the five clusters, along with the rank that was originally assigned to these chosen sentences within that cluster is presented in Table 14. The three favourable sentiment clusters are also put in bold and italics for clarity. For the first cluster, there was only one sentence, so by default it was ranked as 1, and was of course chosen as the representative sentence as well. For the second cluster, one of the favourable tone clusters, the sentence chosen by the program was also ranked as first. It described the off-price business, which was Nordstrom's largest source of new customers for that year. There was a sentence in this cluster that discussed a net sales increase, which I had almost chosen for the first rank as it seemed to more generally discuss why the overall sentiment for this document was favourable, whereas the other sentence discussed only one part of the business. However, upon further review, the sentence that discussed a net sales increase also discussed a small sales decrease, which would have indicated unfavourable or negative sentiment. This conflicting sentiment within the same sentence may have been why it was not closer to the cluster centre (since the cluster had overall favourable sentiment). The conflicting sentiment is also the reason why I did not choose it to be ranked first as it would make a poor choice to justify the reason why the overall sentiment for the document was favourable. For the third cluster, another favourable tone cluster, the chosen sentence was also ranked first, as it best summarised the theme of this cluster. In the fourth cluster, a neutral sentiment cluster, the chosen sentence was ranked second. However, the first ranked sentence was only slightly better. Both sentences discussed the extent of goods Nordstrom offered and, upon reviewing the cluster, it could be seen that either one could have worked well as the representative sentence of that cluster. In fact, when rerunning the program, sentences in the two neutral sentiment clusters would often shift and the first ranked sentence for this cluster would sometimes be chosen instead. In the final cluster, the last favourable sentiment cluster, the chosen sentence ranked first, summarising the operational improvements and representing favourable tone the best out of the sentences in this cluster. As can be seen, based on these early findings, the justification mining algorithm was able to mostly select sentences that were also manually ranked as first. While this is only a small initial trial and cannot provide much support towards the overall efficacy of justification mining, it does at least fulfil the goal of this trial, which was to make sure the algorithm itself was functioning properly. From these early evaluations, it can be concluded that the code and algorithm for justification mining was working as intended and I was able to get a better understanding of the results and how it functioned, as well as begin to think about final evaluation methods and implications of potential success for the research. The findings in this subsection also help address research sub-question 2-1. Since mined justifications are outputted by the algorithm simply as sentences, they are easy to interpret and provide tone or sentiment information for the overall 10-K MD&A Overview document, demonstrating their potential for sentiment summarisation.

Table 14: Mined justifications from Nordstrom 2017 MD&A Overview subsection along with manually annotated assigned ranks. The three favourable sentiment clusters are in bold and italics.

Cluster #	Justifications	Rank
1	As of January 28, 2017, our stores are located in 40 states throughout the United States and in three provinces in Canada.	1
2	***Our off-price business reached $4.5 billion, with growth mainly driven by our online net sales increase of 32% and 21 new store openings.***	***1***
3	***Moving forward, we believe our strategies give us a platform for enhanced capabilities to better serve customers and increase market share.***	***1***
4	We offer an extensive selection of high-quality brand-name and private label merchandise through our various channels, including Nordstrom U.S.	2
5	***Through these efforts, we made significant progress in improving operational efficiencies, reflected by moderated expense growth of 10% in these three key areas, relative to an annual average of 20% over the past five years.***	***1***

7.5.2 Final Evaluation Methods

In this subsection, options for the final evaluation methods of justification mining on the full dataset are presented and discussed based on further literature review and consultation with experts in the field. As justification mining is a novel idea, there are no set or best forms of evaluation that are perfectly suited to it.

As discussed previously, in order to evaluate justification mining, there is a need to formulate a method that evaluates both the clustering algorithms as well as the mined justifications. There are numerous forms of evaluation for clustering models that exist in the literature, including the Elbow Method, Silhouette Analysis, downstream supervised learning tasks, evaluating how compact and well-separated clusters are, intra- and inter-cluster variance, F-measure, and Adjusted Rand Index (Manning et al., 2009; Rodriguez et al., 2019; Xu and Tian, 2015; Palacio-Nino and Berzal, 2019). It was a difficult task to know which one would be best suited for this study since justification mining, as stated before, is a new idea and none of the models being evaluated in the literature could perfectly match it. In order to help assess evaluation methods, I sought out the advice of experts. I knew of one expert in particular with a wide scope of knowledge in the machine learning domain: Professor Stephen Roberts, head of the Oxford Robotics Group and the Oxford-Man Institute of Quantitative Finance, who has worked extensively on unsupervised learning for decades (Roberts, 2020). Through email and video conference correspondence, we discussed my approach and the most suitable evaluation methods for it (Roberts, 2020).

Professor Roberts stated that the problem with the approaches I had researched through literature review was that they all make subjective assumptions of what success looks like in the data. For

instance, if using compact and well-separated clusters as a form of evaluation, one is assuming that more compact and separated means more successful clustering and, thus, successful justification mining. But this "success" does not actually mean much. Instead, it might be better to consider a form of evaluation with the goals of justification mining and the research questions in mind (Roberts, 2020). We also discussed the initial evaluation methods outlined in Section 7.5.1, which involved analysing the theme of each cluster and annotating ranks for each sentence in a cluster. Manual annotating and checking are useful for deep understanding, but this applies a human subjective measure of similarity (Sarkar, 2009; Roberts, 2020). With this approach, a massive crowd-sourcing effort is required (Sarkar, 2009; Roberts, 2020) along with randomising the trials and performing them thousands of times (Roberts, 2020). This method corresponds to a psychologist approach of evaluation: Do the results conform to what human beings agree as being correct (Roberts, 2020)? However, this evaluation method would likely be the subject of a thesis in itself and there was neither the time nor the resources to take this approach.

According to Professor Roberts, this form of manual evaluation would not be required for this thesis and instead a loss/reward measure could be utilised (Roberts, 2020). In any research that is carried out scientifically, there must be a clear measure of success or failure. If the question is whether the clusters and mining conform to human standards, then crowdsourcing is required. This is different for loss/reward measures and forecasts. Forecasts as an evaluation method are fair and cannot be argued with (Roberts, 2020). If observing a single asset sentiment over time, does this correlate to the share price movement? In this way, future stock returns are the tool used for evaluating justification mining. The questions that are being asked then include: Is the nature of sentiment in mined justifications informative? Does it appear to correspond to future returns/share prices or, at least, correspond more than sentiment from full documents (Roberts, 2020)? Utilising financial metrics in a downstream evaluation task is possible due to the data being financial text from publicly traded firms.

This approach also avoids issues with the various assumptions different clustering algorithms make. For instance, K-Means makes the naïve assumption that distances are Euclidean (clusters are circular) and Agglomerative clustering makes the assumption of Euclidean as the norm (Roberts, 2020). Evaluation methods need to take careful note of mathematical assumptions made by different algorithms. Because of this, it is necessary to look at some kind of reward measure in the outcome space of the problem being assessed which, in this case, would be a risk-reward measure. Without this downstream measure of failure, evaluation becomes too subjective. In this way, the evaluation method proposed by Professor Roberts is an empirical measure of performance that does not make a biased assumption of what success looks like in the data. Thus, downstream tasks are a useful option for measuring success for unsupervised machine learning models, such as justification mining. There are methods where supervised learning tasks can be used as this downstream task, but because I am utilising financial data, it is much simpler to look at future returns as a form of downstream evaluation (Roberts, 2020).

Based on these discussions, a good first option for evaluating justification mining seemed to be to assess the correlation between mined justification sentiment and future returns. To do so, transfer learning methods from Section 6 can again be utilised. A classifier can be trained on annotated financial sentences (such as those from Malo et al., 2013) and then used to make predictions on the mined justifications. The results of this sentiment analysis could then be correlated to the future returns of each stock, using various methods like similarity tests. In order to evaluate whether the

results are meaningful and whether enough data has been used, statistical significance testing would also need to be performed (Roberts, 2020).

After consulting with Professor Roberts, I went on to elaborate and discuss these ideas with another professor at Oxford with experience in unsupervised learning and financial text data, Professor Amir Amel-Zadeh, professor of accounting in the Business School and member of the Oxford-Man Institute (Amel-Zadeh, 2020). Professor Amel-Zadeh agreed that the best option would be some form of aggregated sentiment from mined justifications, which could then be correlated to future returns. 10-K filings get published as results from the past financial year and, in the MD&A subsection, there is forward-looking information, which makes it ideal for this form of evaluation (Amel-Zadeh, 2020). Professor Amel-Zadeh suggested that one option would be to correlate to the following one-year stock returns. Would aggregated sentiment scores from mined justifications have any correlation to these returns? Professor Amel-Zadeh also elaborated on what a good time-frame for stock returns might be, suggesting up to a year would be suitable, based on the kind of information MD&A subsections typically include. He also suggested a good number of filings would fall between 200-300, ideally checking each chosen firm as far back as possible in order to avoid only checking periods of large-scale economic trends (Amel-Zadeh, 2020).

In this thesis, aggregated sentiment falls into two broad categories, summarised and full sentiment scores. Full sentiment is aggregated from full MD&A Overview documents while summarised sentiment is aggregated from mined justifications (which are meant to capture the sentiment of a full document with only a few sentences). Another categorisation used in this research is weighted versus unweighted sentiment scores. Unweighted scores are what is typical in sentiment analyses, where each sentiment score is an integer value. Weighted sentiment scores, instead, can be integers as well as float values, and are possible based on the type of aggregation method used during transfer learning. For instance, one method for obtaining weighted sentiment scores from justification mining involves first classifying mined sentences with trained models. From there, a numeric, weighted sentiment value could be calculated based on how many of the mined sentences were classified with the majority classification. For instance, if five sentences were mined from five clusters and three were classified negative, the weighted sentiment score would be -3/5 or -0.6 instead of the usual score of -1 utilised to represent negative classification in most sentiment analysis. Professor Amel-Zadeh also suggested that the same correlation analysis to future returns could be performed using unweighted sentiment (so in the example given above, using -1 instead of -0.6) in order to assess if correlation to future returns is better or worse with unweighted sentiment scores (Amel-Zadeh, 2020). In this way it is possible to assess whether weighing the sentiment scores provides extra benefit, which would be an evaluation of the sentiment aggregation methods. The above approach for obtaining weighted and unweighted scores, however, does not take into account what the sentiment of the remaining two sentences is (i.e. it would be the same whether both the remaining sentences were neutral sentiment or if only one was). In order to address this, other options for obtaining weighted sentiment scores would be to use the mean of individual scores, while using the mode or rounded mean could be options for obtaining unweighted scores. Similar to comparing correlation of weighted and unweighted scores to future returns, full and summarised sentiment scores can also be compared to each other to assess which category has better correlation to future stock returns. Professor Amel-Zadeh agreed comparing metrics on full document versus mined justification sentiment would be a suitable approach for evaluation of justification mining (Amel-Zadeh, 2020). Unlike weighted versus unweighted scores which assesses aggregation methods, this approach directly compares the product of justification mining (mined sentences) to full documents and addresses research question 3 (see Section 4.3).

Based on input from both professors, one path to evaluate justification mining became clear. An area where the professors' opinions differed, however, was about how to look at future returns. Professor Roberts suggested simply looking at future returns utilising straightforward percent change as the method (Roberts, 2020). In this approach, the percent change in stock prices over various time periods would be correlated to aggregated sentiment scores. Comparing the correlation scores and amount of statistically significant correlations between stock returns and sentiment from mined justifications versus full documents would become the means for evaluating justification mining. Professor Amel-Zadeh, instead, proposed an alternate approach of creating a portfolio of stocks in the dataset, at least 300 companies over the last ten years, and taking long and short positions in each company in the dataset with a set initial investment (for instance $1 or $100,000). The returns could then be calculated for the following year after the last 10-K filing in the dataset. The stocks would be held at the position as indicated from the sentiment analysis of the mined justifications for the entire year and returns could be checked for this fictional investment at the end of that year. This method is often utilised in similar financial studies (Amel-Zadeh, 2020). Likewise, the returns can also be calculated holding stocks at positions based on sentiment analysis of full documents. Comparing these returns would then serve as a means for evaluating justification mining and a correlation analysis would not be required. However, while this is an interesting alternate approach, the main issue with it is that it would be unfeasible to manually parse that amount of data in a reasonable amount of time, given how much time remained after the Confirmation viva. For reference, just 18 firms took approximately 2.5 months to manually parse, so 300 would be impossible within the time remaining. Thus, Professor Roberts' approach using percent change was chosen as the best option for this research and the scope of a DPhil thesis.

The details of how this methodology was implemented are elaborated upon in the following subsections where the method and results are outlined more thoroughly.

The evaluation method outlined above, however, only helps answer research question 3. Another form of evaluation was required to address research question 2 and its sub-questions. For this, it was necessary to find a means of comparing the aggregated sentiment scores from mined justifications versus the aggregated sentiment scores from full MD&A Overview documents. After further discussion with Professor Roberts as well as with my second supervisor, Dr. Edwin Simpson, two metrics for calculating this comparison were chosen, cosine similarity scores and Pearson correlation coefficients (Roberts, 2020; Simpson, 2020). Being able to compare the two aggregated scores would allow assessment of how well justification mining is able to capture most of the sentiment from the full document with only a few sentences. The idea is that the more similar the aggregated sentiment scores are between mined justifications and full documents, the better justification mining is at capturing or summarising the sentiment of the full document (Roberts, 2020; Simpson, 2020). This second form of evaluation helps address research question 2 and is arguably the biggest contribution of this research investigation, since research question 2 is the most important. Thus, the methodology and further details of this second evaluation approach are discussed at length in *Section 7.6.4 Making Predictions on Mined Justifications and Full Documents* and *Section 7.7.2 Similarity between Mined Justification Sentiment and Full Document Sentiment*.

7.6 Experiment Design – Justification Mining

7.6.1 Training Classifier Models – Bag of Words approach

After the data was collected and manually checked, it was then transferred to a *.csv* format file. Most of the code for various algorithms, models and calculations were written in Python 3 *.py* module files, while the final data manipulations and running the models was done in a Jupyter Notebook. The data was read into a dataframe object from the *.csv* file to the Jupyter Notebook using the Pandas library. The columns of the dataframe included the unique CIK number for each firm, the ticker symbol, the date the 10-K filing was released (parsed in *datetime* format), and finally the parsed text itself. Several logistics checks were also performed to ensure the data was read properly, such as checking for missing values in the dataframe.

After the data was properly stored in the dataframe object, it was time to train the classifier that would be utilised to make predictions on the mined justifications. The process for this training was carried out in similar fashion to the transfer learning tasks in Section 6, using the same Malo et al. (2013) sentence-annotated data. This being the case, the method for how these classifiers were trained is not outlined again. Instead, the focus is on the differences between the training in this section and Section 6. The major difference in classifier training here was simply that two different numericising techniques for creating features from the text were used, unlike in Section 6, where only the Bag of Words method was used for feature engineering. Moreover, fewer machine learning model options were utilised, focusing only on the ones that worked best previously. Preprocessing steps for the Bag of Words method and general model hyperparameters are based on the choices that worked optimally for the transfer learning task in Section 6.

In Section 6, four different classifiers were used: decision trees, random forests, Naïve Bayes, and Support Vector Machines. Mean and mode aggregation were utilised for getting a final document sentiment score from the sentence level classification predictions and four metrics of evaluation were calculated (accuracy, recall, precision and F1-score). Based on the findings from Section 6, deep learning models were not trained for classification tasks in Section 7 as the same training dataset was being used and they did not work well on this amount of training data.

To start, two Bag of Words based models were trained, a random forests model and a Naïve Bayes model. To set up the Bag of Words method, training and testing sets needed to be created. The Malo et al. (2013) data was read into a Pandas dataframe and the sentiment for each sentence was reassigned from its original string values (favourable, unfavourable and neutral sentiment) to numeric values (1, -1 and 0 respectively) using the *iterrows()* function. The data was then split into two separate lists, one with the sentences themselves and the other with their corresponding sentiments. This was done via list comprehensions. The list with the actual sentences was then pre-processed similar to Section 6, lowercasing, substituting any non-words, splitting on the word to tokenise the text, and lemmatisation using the NLTK library's *WordNetLemmatizer* (Check Section 6.4 for full details). After this the *Counter()* function in the collections library was utilised to set up the actual Bag of Words. A dictionary was created using this method with keys being the words in the vocabulary and values being their count. With this, most common and least common words were removed from the bag of words by calculating word frequency. This was done to help improve the classifier training. The low cut-off frequency for least common words was set to a percentage equal to

0.00029 and only the five most common words were removed. From there, two mapping dictionaries were created that could be utilised to transform text to their corresponding numerical id (using the *enumerate()* function on a list of words created from the *Counter()* dictionary keys) and back from the numerical id into the corresponding word. Using *enumerate()*, words were simply assigned a number starting from 1 until the total number of unique words in the data. With these dictionaries, the original tokenised text sentences were then filtered and finally numericised. These numeric sentences became the training data for the classifiers, but before they were final, they were standardised to a single length, in this case 30. This was done by truncating any sentences longer than 30 and adding 0's to sentences shorter than 30 (similar to Section 6). A fixed length is needed for machine learning tasks. One other optional function was written into the code but ultimately not utilised due to finding an easier alternate method for doing this, and that was a balancing function. The balance function was created to remove random sentences of a given sentiment label until all three labels had a similar ratio. It was discovered that a manual function for balancing was not required as the sklearn library had options for this built-in, which will be discussed shortly. In the end, the function for creating the training dataset returned the mapping dictionary, Bag of Words method tokenised and numericised text (i.e. the training dataset itself or *X_train*), and corresponding sentiment labels (*y_train*).

To create the test set was a simpler process, as the training mapping vocabulary/dictionary could be used to filter the testing data. The text was again preprocessed and then filtered to include only words in the training vocabulary using the training mapping dictionary. This represents one of the biggest shortcomings of the Bag of Words method; namely, that the test set cannot contain any words not part of the training set vocabulary (which would raise errors during model training). As will be seen later, this is a problem that numericisation via Sentence-BERT and transformer models does not have. Once the test set has been filtered, the same process is utilised to create numericised sentences and truncate or pad sentences to be the same length of 30. Unlike with the training set, however, this function only returns the test set (*X_test*) and nothing else.

Finally, a function was created for the actual training of the classifier models. Options for four different models were coded, similar to Section 6 (decision trees, random forests, Naïve Bayes and Support Vector Machines). The classifiers were utilised from the sklearn library (*DecisionTreeClassifier, RandomForestClassifier, MultinomialNB* and *SVC)*. Various hyperparameters of these models were tuned during training, again like in Section 6. For convenience, the table of hyperparameters is given once more below, see Table 15. Notice, however, the main difference versus Section 6 is that this time the *class_weight* hyperparameter was utilised. As discussed earlier, originally a balancing function was written, but this hyperparameter offered an easier and more readily manipulatable option for balancing the amount of sentiment labels without removing sentences from the dataset (which was the case in the manual balancing function). Recall, in Section 6.4.3.3, manually balancing the labels removed some of the training data, potentially affecting the performance of the subsequent trained models. The *class_weight* parameter instead implicitly replicates the smaller class labels until ratios are similar. The models are then trained on the training set and corresponding class labels, with the trained model and mapping vocabulary again being returned so that this vocabulary can be used to transform text back to its original state.

Table 15: Classifier hyperparameters that were tuned for the sklearn models in Section 7.

Classifier	Hyperparameters
Decision Trees	max_depth, min_samples_split, min_samples_leaf, max_leaf_nodes, class_weight
Random Forests	n_estimators, max_depth, min_samples_split, min_samples_leaf, class_weight
Naïve Bayes	alpha, fit_prior
SVM	C, kernel, probability, max_iter, cache_size, class_weight

While this was all programmed in Python files, within the Jupyter Notebook itself, the models were actually trained through importing the created functions, and objects for the mapping vocabulary and trained model were formed. These model and vocabulary objects were saved into pickle files on the local machine so that models did not need to be retrained. Only two promising machine learning models from the Section 6 conclusions were utilised with the Bag of Words numericising method: Naïve Bayes and random forests. This was done because the final predictions would be made in many different variations, and so in order to limit the size of the total calculations within realistic and feasible parameters, only two models were trained here. Moreover, in all Bag of Words models, the class weights or labels were balanced (setting the hyperparameter *class_weight=balanced*).

7.6.2 Training Classifier Models – Sentence Embeddings Approach

In order to train classifiers that utilised text numericised with Sentence-BERT, a similar process to above was utilised. However, as sentence embeddings were being created with pre-trained datasets, the process was much simpler and straightforward. As before, the annotated data from Malo et al. (2013) was read in, sentence labels changed to numbers, and data split into two separate lists via list comprehensions (one for the sentences and another for their corresponding labels). However, unlike before, it was not necessary to tokenise or preprocess the text in any way. Instead, an embedder was instantiated utilising Sentence-BERT with a pretrained model. As discussed in Section 7.2, Sentence-BERT utilises transformer models to create semantically meaningful sentence embeddings. The idea in this subsection was to utilise these semantically meaningful sentence embeddings directly as numericised training data on which to train classifiers. Because they are all created at a fixed length and provide semantic information, they seem to be an ideal choice for numericising text as input into ML models. Moreover, unlike the Bag of Words approach, no data was lost or added to sentences to standardise their length. This could prove to be a driving force in their performance if they outperformed Bag of Words models. In a subsequent subsection, the method for using Sentence-BERT for mining justifications is discussed in greater detail, but for this subsection, utilising Sentence-BERT for numericising text for input into machine learning classification models is the focus.

Two pre-trained models were chosen for instantiating the embedder, 'bert-base-wikipedia-sections-mean-tokens' and 'bert-base-nli-mean-tokens'. The model for 'bert-base-nli-mean-tokens' is part of the set of models optimised for Semantic Textual Similarity tasks. This particular model used mean pooling on the large NLI training dataset with the bert-base-uncased base model. It was also fine-tuned with the STS Benchmark train set (Reimers and Gurevych, 2019). On the other hand, the 'bert-base-wikipedia-sections-mean-tokens' pre-trained model was trained on a Wikipedia dataset originally from Dor et al. (2018) and uses the triplet network structure outlined earlier (by learning whether two

sentences are part of the same Wikipedia section). This pre-trained model was chosen because it is ideal for fine-grained clustering (Reimers and Gurevych, 2019). Both of these are the same two pre-trained models that are utilised for justification mining later, as the justification mining process involves clustering sentences (in this case clustering similar sentences into topics or sections). Once the embedder is instantiated, it is used to encode the sentences from the training data. The function then returns the encoded embeddings (*X_train*) and their corresponding sentiment labels (*y_train*).

The process for creating the test set was similar. Since I was utilising massive pre-trained models to create the embedder, saved mapping dictionaries or similar objects from the Bag of Words training process were not needed to create the test set here. Instead, the pre-trained models were again utilised to instantiate an embedder which directly encoded the sentences in the test set similar to the training process. The function then returned these test set embeddings (*X_test*).

Although the training of the classifier models themselves was similar to the Bag of Words approach, more variations were utilised here as testing how well Sentence-BERT worked as a numericiser was essential to addressing research sub-questions 2-2 and 3-1. The function again allowed use of the four different classifier models mentioned before and also had an optional parameter to choose whether or not to balance class labels (via the *class_weight* hyperparameter). The sklearn models and the hyperparameters that were tuned during training are the same as the ones listed previously in the Bag of Words subsection above (see Table 15). This time the model that was primarily utilised was random forests along with some minor focus on Support Vector Machines as well. As this was the first time testing Sentence-BERT for numericisation, the classifier model was mostly kept fixed to properly compare other variables. The first random forest model variation used balanced class labels and the Wikipedia pre-trained model. The second random forest model also used the Wikipedia pre-trained model but instead did not balance class labels. The final random forests model utilised the NLI pre-trained model and again used balanced class labels. For the Support Vector machine models, only balanced class labels were used for two models, one using the Wikipedia pre-trained model and another using the NLI pre-trained model. This settled to a total of five different classifiers trained using the Sentence-BERT embedding method for numericising text. These trained models were again saved into pickle dumps, which could be loaded throughout the rest of the research process without constantly retraining the model and skewing the results. It was important to do so, as retraining the classifiers every time the Notebook was reopened would have led to inconsistency in the results and conclusions that could not be generalised well or compared accurately.

7.6.3 Justification Mining

Before the trained classifiers from the previous section could be utilised, it was first necessary to mine the actual justification sentences from the text. This subsection discusses the process for justification mining and details behind its code. As outlined in Section 7.2, the process of justification mining also utilises transformer models and Sentence-BERT. The method outlined in Section 7.2 was used as the basis for formalising the process into a single function and later into a Python Class. Thus, in this section, the details of the justification mining process that were discussed earlier are not discussed again. Instead, the focus is on how the entire process was brought together in order to be utilisable in a practical sense.

A function was created in Python to perform all the necessary calculations and data manipulations. The first task was to determine if any preprocessing would be required to encode the sentences with Sentence-BERT. According to a personal correspondence with Professor Reimers, however, it was determined that no preprocessing of the input text would be required in this case. This allows the embeddings method to be more consistent between different models, whereas normally in NLP tasks (as was clear from Section 6 where many preprocessing steps were taken) a variety of preprocessing steps are taken before text can be utilised with machine learning models (Reimers, 2020). Before encoding the sentences in each MD&A Overview subsection, however, the data still needed to be split into sentences, as Sentence-BERT takes input in this form. This was done with Python's nltk library using the *sent_tokenize* method from the *tokenize* class. Recall, in Sections 6.4.3.2, 6.4.3.3 and 6.4.3.4, there were various issues using the *split()* function to tokenise documents into sentences. Thus, a more sophisticated method from the nltk library, *sent_tokenize()*, is used here. Next, sentences were again encoded into semantically meaningful embeddings using the two pre-trained models that were utilised in the previous section, 'bert-base-wikipedia-sections-mean-tokens' and 'bert-base-nli-mean-tokens.' The expectation was that the Wikipedia based pre-trained model would outperform the pre-trained model from the NLI dataset here as the Wikipedia dataset model is designed for subsequent downstream clustering tasks (Reimers and Gurevych, 2019).

Once the sentences had been encoded, the next step was to cluster them in order to derive justifications later. Four different clustering models were used as options for this part: K-Means clustering, Agglomerative clustering, DBSCAN (Density-Based Spatial Clustering of Applications with Noise) and MeanShift. K-Means and Agglomerative clustering are popular approaches often utilised in literature for clustering tasks. Both models were utilised from Python's sklearn library. Agglomerative clustering is a type of hierarchical clustering that uses a bottom-up approach to cluster. It begins with each object in the data as a separate group and continuously merges groups close to each other until it reaches the specific k number of clusters. Because it is a sequential process, Agglomerative clustering is hierarchical (Reddy et al., 2017). On the other hand, K-Means is a simultaneous clustering model. The objective of K-Means is again to split the data into a prespecified k number of clusters utilising cluster centres or means. The goal is minimisation of the squared Euclidean distances (Kaushik and Mathur, 2014). While both of these models require the specification of k number of clusters beforehand, there are some important differences. K-Means is generally able to handle large amounts of data better and is usually faster. Hierarchical clustering, however, is usually more reproducible as K-means initialises each training instance randomly. K-Means also works best when clusters are spherical in vector space (Reddy et al., 2017; Kaushik and Mathur, 2014).

Python's sklearn documentation also confirms that K-Means is one of the fastest clustering algorithms but recommends restarting the model several times to verify results (sklearn.cluster.KMeans, 2020). Unlike when training classifiers, the number of hyperparameters tuned for these clustering models was small. For the K-Means algorithm, the main hyperparameter tuned was *n_clusters*, which allows the specification for the number of clusters the algorithm will produce. This hyperparameter also, consequently, determines the number of justifications that will be mined, since each cluster will produce one justification or representative sentence. The other hyperparameter that was modified was *random_state*. Given that Sentence-BERT and transformer models in general utilise deep learning methods, it was important to set the *random_state* to make the randomness deterministic and allow for results to be reproduced (sklearn.cluster.KMeans, 2020). For Agglomerative clustering, the *n_clusters* hyperparameter was again set. This time, however, two other hyperparameters were also tuned, *linkage* and *affinity*. Linkage is related to the distance that is used. In this case the options that were

trialled were *ward, average* and *complete.* Affinity refers to the metric utilised in the linkage calculation. The options that were used in fine-tuning this hyperparameter included: *Euclidean, L1, L2* and *Manhattan* (sklearn.cluster.AgglomerativeClustering, 2020).

In both models, the *labels_* attribute was used to extract the cluster label for each sentence. In the K-Means model, the model itself also calculates the cluster centres and stores them in the *cluster_centers_* attribute. This model allowed a slightly easier way to get these, saving some time. In Agglomerative clustering, however, these cluster centres had to be calculated independently. For both models, the first step was to instantiate an instance of the model. This model object was then fit to the sentence embeddings encoded utilising Sentence-BERT. For K-Means, the labels and cluster centres were then saved in separate numpy arrays. In the Agglomerative clustering approach, the labels were also saved in a numpy array, but further steps were required to obtain the cluster centres. An empty list of lists was instantiated utilising a list comprehension, with an *n_clusters* number of lists. The labels were then sorted into this list using a for loop and the *enumerate* function to get the correct embedding from the original corpus embeddings list into the proper cluster based on its assigned label. The cluster centres were then calculated using a list comprehension by taking the mean of each cluster (*axis=0*) for each list/cluster in the list of lists that was created and filled previously. The end result was similar to what the K-Means algorithm did automatically and did not add too much extra time to the overall process. Note that in both K-Means and Agglomerative clustering, the number of clusters, *k*, was set to five by default. This number was determined by writing a quick algorithm in Jupyter Notebook to obtain the average number of sentences in each data entry (in other words, in each MD&A Overview subsection) using nltk tokenize's *sent_tokenize* function. Based on this average, it seemed like five would be a good starting place for separating into large enough clusters that still maintained enough variance to have separate topics. This number was also chosen based on the initial trials that were outlined earlier, using the Nordstrom 2017 MD&A Overview data. Finally, one other method was used for determining optimal model hyperparameters, including *n_clusters*. As this was an unsupervised learning task, sklearn's grid searching functionality was not appropriate. Instead, twenty randomly chosen MD&A Overview subsections were manually annotated by splitting the documents into clusters and assigning each sentence in a cluster the same label. Various *n_clusters* values (from two through seven) along with the other hyperparameter options listed for each model previously were trialled on these twenty subsections and optimal values were chosen based on which ones worked best for the clustering task (split the clusters closest to the manual annotations). The optimal number for *n_clusters* ended up being five in both clustering models. This method was chosen because it seemed like typical clustering scoring methods, for instance Silhouette coefficient, were unsuitable for this study as they did not provide information on whether the subsequent justification mining task would perform well (Roberts, 2020).

The other two clustering algorithms used, DBSCAN and MeanShift, differed from K-Means and Agglomerative clustering in that they did not require specifying a *k* number of clusters beforehand (Kaushik and Mathur, 2014, Ester et al., 1996). This allowed the testing of whether the number of clusters that was set for the first two models (five) made sense. It also allowed for an alternate approach to justification mining where the number of justification sentences returned by the function would vary between each MD&A Overview subsection (both between different firms but also even between different years within the same firm). Since the sentiment would later be aggregated, it did not matter if the number of returned justifications varied, only if they were able to capture the overall sentiment of the document. Both DBSCAN and MeanShift were implemented using the sklearn library in Python. DBSCAN works by utilising density, locating areas of high density and developing clusters from that starting point (Ester et al., 1996). Three hyperparameters were tuned during various

trials. The first and most important was *eps,* which determined the maximum distance between samples for them to be neighbours. For this, values between 0.3 and 0.7 were tested. The other two hyperparameters were *min_samples* (values between 3 and 7) and *leaf_size* (values between 20 and 40). As with before, the optimal values were found utilising manual annotation of the 20 random MD&A Overview subsections. The optimal values were found to be 0.5 for *eps* and 4 and 25 respectively for *min_samples* and *leaf_size*.

As with before, the DBSCAN model was first instantiated with the chosen hyperparameters and then fit on the sentence embeddings. The cluster labels were again saved into a numpy array from the *labels_* attribute. Just as with Agglomerate clustering, DBSCAN did not inherently calculate cluster centres, thus a similar method had to be used to manually calculate them. Unlike Agglomerate clustering though, DBSCAN creates a cluster of all noisy samples and assigns these a label of -1. As this noisy samples cluster would not yield any meaningful justification sentence (remember, justification sentences are chosen by selecting the sentence closest to the cluster centre in vector space and utilised as a representation of the information/sentiment in that cluster, which would not be the case for a noisy samples cluster), it was necessary to remove it from the data. This was done by creating a new numpy array for the cluster labels containing only those samples that were not labelled -1 and simultaneously removing the corresponding index from the original sentences' embeddings list, so that both the sentence embeddings list iterable and the cluster labels numpy array iterable would remain the same length (effectively just removing the sentences that DBSCAN determined were 'noisy' from the data entirely). Next, since the number of clusters varied when using DBSCAN, the number of clusters had to be calculated by taking the length of the set created from the cluster labels array (using the *len* and *set* functions in Python). The process returned an integer giving the number of clusters derived from the algorithm. Then, as before, a list of lists was created using a list comprehension. Again, a for loop and the *enumerate* function were used to sort the sentence embeddings into the corresponding clusters. Finally, the mean along *axis=0* was calculated for each cluster and stored into a new list corresponding to the cluster centres.

Like DBSCAN, MeanShift also does not require specification of the *k* number of clusters beforehand and works by utilising centroids (Kaushik and Mathur, 2014, Ester et al., 1996). The only hyperparameter that was tuned for MeanShift was *min_bin_freq,* which was trialled with values between 1 and 4 using the same methods outlined previously. It was found to work best at the default value of 1. MeanShift, like K-Means, returned both a *labels_* and a *cluster_centers_* attribute. Thus, it simply had to be instantiated then fit to the sentence embeddings and these two attributes could be saved into numpy arrays directly.

In the end, each clustering algorithm would return two numpy arrays, one containing the class labels assigned to each sentence embedding from the originally encoded sentence embeddings list, and another containing a smaller numpy array with the vector values for each cluster centre returned by the model. To summarise the various hyperparameters that were tuned, Table 16 contains a list of hyperparameters for each model along with the optimal values found using a manually annotated random subset of the data (with the method described previously).

Table 16: Clustering algorithm hyperparameters that were tuned for the sklearn models in Justification Mining.

Clustering Model	Hyperparameters	Optimal Values (respectively)
K-Means	n_clusters, random_state	5, 42
Agglomerative	n_clusters, linkage, affinity	5, ward, euclidean (note: euclidean must be selected for *affinity* when *linkage* is ward)
DBSCAN	eps, min_samples, leaf_size	0.5, 4, 25
MeanShift	min_bin_freq	1

As discussed in earlier subsections, the easiest way to manipulate this data was to combine everything into a Pandas dataframe object. The next step was calculating the cosine similarity between each sentence embedding and the corresponding cluster centre from the cluster to which that sentence belongs. A list comprehension was thus, first, utilised in order to create a new list of the same length as the array of sentence embeddings, with each index containing the cluster centre vector corresponding to the sentence at that index. This task was performed by iterating through the class labels list and pulling the cluster centre at the index corresponding to that label (this list would populate the '*cluster_centres*' column as described below). Then, an empty dataframe was created and various columns filled with all the information obtained up to that point. The original sentences from the text corpus get placed in a column labelled '*sentences*', their corresponding embeddings in an '*embeddings*' column, the labels given to each sentence by the clustering algorithm in a '*labels*' column, and finally the cluster centre vector for each sentence based on that label in a '*cluster_centres*' column. In this way, each row of the dataframe contained all the pertinent information for each sentence calculated during the justification mining process.

Cosine similarity is then calculated utilising the sklearn metrics pairwise module. In this regard, cosine similarity is utilised as a metric for computing the distance between one point in the vector/embedding space to another. It can, therefore, be used as a means for finding the point or embedding closest to the cluster centre by calculating the cosine similarity between all points in a cluster and the cluster centre and then taking the sentence with the max similarity score from each cluster. The formula was given earlier in Section 6.3.4 and is as follows, simply the normalised dot product between X and Y:

$$similarity = \cos\theta = \frac{X \cdot Y}{|X||Y|}$$

The goal would be to create another column in the dataframe containing all the calculated cosine similarities for each sentence. To start with, first the sentence embeddings and their corresponding cluster centres are zipped together using Python's *zip* function so that they can be input into the cosine similarity function iteratively. Because each array input into the function contains a single sample, based on the way the cosine similarity function works, they also had to be reshaped using *array.reshape(1,-1)*. Finally, a list comprehension was utilised to put the final outcomes into a new list, taking only the final values in the float dtype instead of an array type by calling the [0][0] index. These final float values were placed into a new column in the dataframe titled '*cosine_similarities_to_centre*'. As discussed previously in Section 7.4, the best method that was found to work in order to obtain the sentence with the highest cosine similarity score from each cluster involved Pandas' *sort_values* function. The new dataframe was sorted on the columns '*labels*' and '*cosine_similarities_to_centre*' in descending order and duplicates were dropped on the '*labels*'

column. This returned a dataframe containing only the rows with sentences containing the highest cosine similarity scores and from there it was a simple task to obtain just the sentences themselves using the *tolist()* function on the '*sentences*' column of this new dataframe. The function then returns this list of mined justification sentences at the end of the process. As mentioned before, this method ended up being surprisingly fast for each MD&A Overview subsection inputted during the initial trials, only about 7 seconds per subsection, making it a very viable approach to use with any kind of hardware, with no requirement for GPU clusters. It should be noted, however, that in this research, mining was done using CUDA and GPU processing instead of CPU; however, such conditions should be replicable on most modern computers with dedicated Nvidia graphics cards. There was also an optional parameter included within the algorithm to print the clusters themselves, in other words the sentences grouped into their clusters, for the purposes of testing and sanity checks.

7.6.4 Making Predictions on Mined Justifications and Full Documents

Before outlining the next step, it is necessary to elaborate on the decision to calculate sentiment on the full MD&A Overview subsections along with the mined justifications. As mentioned earlier, through correspondence with Professor Stephen Roberts (Roberts, 2020), it was decided that one form of evaluation for justification mining would be a downstream task involving correlating sentiment from mined justification sentences to future stock returns. To do so, the trained classifiers would only need to make predictions on the mined justification sentences themselves and the full document could be ignored. However, it occurred to me that there would be immense value in also using the trained classifiers to get aggregated sentiment scores on the full documents as well. In this way, many factors could be compared between the mined justification sentences' aggregated sentiment scores and the full document's aggregated sentiment scores, including the difference in the correlation to future stock returns. Moreover, the mined justification scores and full document scores could also be directly compared, bearing in mind that one of the goals of justification mining was to determine if the mined justification sentences could capture the sentiment of the full document using only a few representative sentences. In this way, justification mining could be evaluated as a form of sentiment summarisation (similar in concept to methods that involve document summarisation in current literature: Neto et al., 2002; Sarkar, 2009; Cohan et al., 2018; Gidiotis and Tsoumakas, 2020). However, the goal of sentiment summarisation here is different from that of document summarisation, allowing justification mining to make a novel contribution to the literature in this area. So, it was important to devise methods for evaluating how well mined justifications were able to encompass the sentiment from the full document in only a few sentences. More on this is discussed in *Section 7.7.2 Similarity between Mined Justification Sentiment and Full Document Sentiment* later. For now, I will outline the process itself.

In order to begin making predictions on the mined justifications and full documents, several functions were created to organise the calculations and simplify the enormous number of variations and iterations that could be performed. In normal sentiment analysis on full documents, scores of -1, 0 and 1 are usually the outcome of the predictions. With the transfer learning methods from Section 6, I can instead utilise several aggregation methods to obtain values between -1 and 1, allowing calculation of weighted sentiment, as discussed earlier. Note, these transfer learning techniques can be utilised to create weighted sentiment scores for both mined justifications and full documents. The difference between these two is not that one can be weighted and the other cannot but, instead, that predictions on justifications represent summarised sentiment scores and predictions on full documents do not. From the experiments in Section 6 involving transfer learning, the viability of several aggregation methods was determined. In this section, these methods are expanded upon and further options are

tried in a more formalised and organised approach. In short, I can take what I learnt in Section 6 and apply it now more succinctly and clearly.

To begin, two functions were created within the Python module file itself to obtain the final sentiment scores for all predictions. The first was a general function to obtain the sentiment score for mined justifications from a document, titled *get_mined_sentiment*. The function takes in various input parameters, including the text document (MD&A Overview subsection), the trained classifier model, the mapping dictionary/vocabulary associated with that model (obtained earlier during training of all the models using the Malo et al., 2013, dataset), the clustering model to be utilised (K-Means, Agglomerative, DBSCAN or MeanShift), the feature engineering or numericisation technique (Bag of Words or Sentence-BERT embeddings), the pretrained model to use if using Sentence-BERT to numericise text (Wikipedia or NLI) and finally the sentiment score aggregation method to use (mean, mode or rounded mean; which are discussed momentarily). The function first calls upon the other function created to mine justifications discussed in the previous subsection. This returns a list of the mined justifications (from the full document) whose length varies depending on which clustering algorithm was selected (for K-Means and Agglomerative clustering, the number of clusters was set to five, so five justifications were also returned; however, for DBSCAN and MeanShift clustering the number would vary).

Then, using the functions coded earlier for creating the test set, predictions were made using the previously trained classifier models (see Sections 7.6.1 and 7.6.2 for details). If the Bag of Words method is selected, then the test set is created using the mined justifications and the mapping vocabulary dictionary obtained during the creation of the training set (in order to remove any words present in the test set that are not present in the training set). This test set is basically each word mapped to its corresponding index number based on the training data. As can be seen, this may eliminate a lot of data from the test set, both due to new words that appear and truncating each sample to be of length 30. The trained classifier models then make predictions on this test set and return an array of the prediction scores. If the Sentence-BERT embeddings method is used as input, then the test set is instead created using the mined justifications and the chosen pre-trained model (Wikipedia or NLI). This test set, unlike the Bag of Words model, simply contains semantically meaningful sentence embeddings for each sentence, in other words a vector for each embedding. The model (which, recall, were trained on sentence embeddings created using the corresponding pre-trained model) then makes predictions on this embeddings test set and returns the sentiment scores as before.

Before the final aggregated sentiment score for each text (in this case each set of mined justifications) can be returned, an aggregation method is chosen. There are three options: mean, mode and rounded mean aggregation. The first method, mean aggregation, is based on the evaluation method that was outlined earlier during the trials with the Nordstrom 2017 MD&A data. At that time, counting the dominant sentiment among the justifications and creating a weighted sentiment score based on that was discussed (thus, if, for instance, three of five mined justification sentences had negative sentiment, the weighted sentiment score would be -3/5 or -0.6). In this case, however, it was decided that using the mean might be a better option. In the counting scenario, the effect of sentences with the opposite sentiment are ignored. Because there are three labels (1, 0 and -1), if only the most prevalent sentiment is counted and divided by the total, the effects of the other two sentiment values are ignored. Taking the mean instead considers all the sentiment scores that are returned in order to create the aggregate weighted sentiment score. The dominant score still prevails but is dampened if there are

a lot of samples with opposite or neutral scores. Mean was calculated using the *mean* function in Python's statistics library.

The other two aggregation methods, mode and rounded mean, remove this weighted aspect to the sentiment score. While they remove the weighted aspect, they are still useful for providing additional methods to determine if the mined justification sentences can capture the full sentiment of a document or provide advantage over full documents in correlating to future stock returns. Mode was calculated using the *stats* function in the scipy library. It would return the most prevalent sentiment score among the justifications. While this does ignore the effects of other sentiment scores that might be present (as it is unweighted), I can still use mode aggregation as another way to determine if the overall sentiment of the mined justifications is similar to the sentiment derived from the full document. The rounded mean aggregation method simply utilises Python's *round()* function to round the weighted values obtained before from mean aggregation. It has a similar purpose to mode aggregation but provides a different method for doing so. Both of these will be beneficial as alternative aggregation options, to check if they are useful in correlation to future returns or comparison between mined justification and full document sentiment scores.

In the end, the *get_mined_sentiment* function simply returns a single float or integer value of the aggregated sentiment score for each input text (i.e. set of mined justifications) based on the combination of parameters chosen.

The second function created within the module file is similar to the function above but utilised for obtaining aggregated sentiment scores on the full document (i.e. full MD&A Overview subsection) instead, titled *get_full_sentiment*. It again takes in numerous parameters including the full text of the MD&A Overview subsection, the trained model and mapping dictionary as before, the desired feature engineering or numericisation method (Bag of Words or Sentence-BERT), the pretrained model to use for Sentence-BERT (Wikipedia or NLI) and finally the aggregation method (mean, mode and rounded mean). This time, instead of mining justifications, the sentences in the full document are split using the nltk *tokenize* module's *sent_tokenize* function. If Bag of Words was selected, as before, these split sentences are then used to create a test set using the mapping dictionary/vocabulary (remember, this eliminates a lot of information in the text). The trained model then makes predictions. With Sentence-BERT, again, the test set is created from the sentences with the chosen pre-trained model to create semantically meaningful embeddings on which the trained classifier model makes predictions. Finally, the aggregation is carried out the same way as before, with the options to obtain a mean, mode or rounded mean aggregated final sentiment score for the full document.

With the module file functions completed, I returned to the Jupyter Notebook for the actual calculations. To begin, three empty lists were created to keep everything organised, named 'all_sentiments_list', 'mined_sentiments_list' and 'full_sentiments_list'. These lists would hold various information between mined justification and full document sentiment scores in order to make things easier later. Each time a sentiment score calculation was performed, it would be appended to the original dataframe under a new column (whose rows correspond to each individual filing in the testing dataset). The name for this column would contain information on which options were selected for that calculation. As evident from the two functions created in the module file, there are several options that can be selected between the clustering model, the numericising method, the pre-trained model for embeddings and the aggregation method. In addition to all these options, recall, I also

trained several different classifier models earlier. There are too many combinations of function parameter options for it to be realistic to calculate them all, so it was necessary to pick and choose options. In order to keep these choices organised, the options that were selected were incorporated directly into the name of the column added to the original dataframe. Thus, each column of predicted sentiment scores would be labelled with a name that would contain information on how that score was calculated. This would make organisation much easier for evaluating the results and drawing conclusions. This column name would also be added to the appropriate empty listed created earlier (all column names would be added to the 'all_sentiments_list', while mined justification sentiment scores are added to the 'mined_sentiments_list' and full document sentiment scores added to the 'full_sentiments_list').

The first function written in this section of the Jupyter Notebook, titled *calculate_mined_sentiments*, would be used to calculate the sentiment scores for mined justifications in an organised manner. The function would allow the specification of the column name (which, as stated, would include the options selected), as well as include parameters to input all these options (which model and mapping dictionary to use along with which clustering model, numericisation method, pre-trained Sentence-BERT model, and aggregation method to use). To start, an empty list was created to house all the calculated sentiment scores in order. Then the function iterates through all the rows of the original dataframe (containing all the parsed data) using Pandas *iterrows()* method. The *tqdm* module is also used to keep track of how long each calculation takes, in order to make observations on the differences in time between various methods. During the iteration, the loop uses the *get_mined_sentiment* function from the module file to obtain the final aggregated sentiment score for each dataframe row, taking as input the 'TEXT' column from each row (which contains the actual text from each MD&A Overview subsection) along with the specified options for the other parameters from that function (model, mapping dictionary, numericiser, pre-trained SBERT model, and aggregation method). This sentiment score is appended to the sentiment scores list created earlier. Finally, a new column is created in the original dataframe using the inputted column name parameter as the name and the list of sentiment values as the data (since the list was created in order by row, it will populate the column with the sentiment scores for each row in order as well). The column name is then appended to the 'all_sentiments_list' and 'mined_sentiments_list'. The function also prints the chosen column name along with all the calculated sentiment scores. This is done so the results can be viewed and sanity checks can be made to make sure it is performing the calculations as would be expected.

The second function for calculating full document sentiment scores, titled *calculate_full_sentiments*, works similarly. It takes in the same parameter options except clustering model, since a clustering model is not required as this function is not mining justifications. However, the rest of the process is the same: the function again iterates through the original dataframe and this time uses the *get_full_sentiment* function from the module file to calculate sentiment scores. These scores are once more appended into the original dataframe under the new column name chosen, with that column name being added to the 'all_sentiments_list' and the 'full_sentiments_list'. Again, the function will also print the chosen column names and calculated sentiment scores for sanity checks.

In summary, I have two functions whose parameters can be modified to capture all the various combinations of options allowed for calculating sentiment scores (in other words, the different classifier models that were trained and the different options incorporated into the sentiment score functions in the module file). The functions also print all the results for sanity checking, append the

various options/iterations chosen into organised lists and, using *tqdm*, provide the time each calculation takes for comparison. This allows me to simply modify the parameters of these two functions repeatedly rather than having to write out loops and code for each variation of options calculated.

Due to the sheer volume of different combinations that could be created, it was necessary to choose only a limited number of variations to actually calculate. Focus was given first to models that generally worked well in Section 6. While it may seem like more option/variation combinations were used for mined justifications than for full document sentiment calculations, as can be seen in Tables 17 to 24, this is only because justification mining utilised numerous different clustering algorithms. The clustering algorithm itself is the primary variable that changed between the various combinations involving justification mining. Full document sentiment predictions, of course, did not require any clustering algorithms. While most iterations calculated mean, mode and rounded mean aggregations, for the K-Means and MeanShift clustering models only mean aggregation was utilised, as this method returned a weighted sentiment score. Evaluating weighted sentiment scores was more important as this created novel contributions to the literature, as discussed previously. Next, slightly more focus was given to calculations using Sentence-BERT as the numericisation technique, as this is a newer and less evaluated method than Bag of Words, which is much older and more established in NLP research. Finally, the Wikipedia pre-trained model was also utilised more often as it was designed for downstream clustering tasks, and it was therefore hypothesised to work better in these calculations than the NLI pre-trained model.

Because it would be too unwieldy to list every variation of parameter options used, this information is instead consolidated into several tables arranged by the classifier model employed.

The first set of calculations utilised the random forests model, trained using the Bag of Words numericisation method and its corresponding mapping dictionary (both previously saved into pickle dumps). All of the options in this group also employed the Wikipedia pre-trained model, 'bert-base-wikipedia-sections-mean-tokens'. The option or variable combinations are organised into Table 17.

Table 17: First set of variable combinations for sentiment predictions made on both the mined justifications and full documents obtained from all parsed MD&A Overview subsections.

Training Model	Numericising Technique	Mined Justification or Full Sentiment Scores	Clustering Model used in Justification Mining	Pre-Trained SBERT model	Final Sentiment Aggregation Method
Random Forests	Bag of Words	Mined Justifications	Agglomerative Clustering	Wikipedia	Mean
Random Forests	Bag of Words	Mined Justifications	Agglomerative Clustering	Wikipedia	Rounded Mean
Random Forests	Bag of Words	Mined Justifications	Agglomerative Clustering	Wikipedia	Mode
Random Forests	Bag of Words	Mined Justifications	DBSCAN	Wikipedia	Mean
Random Forests	Bag of Words	Mined Justifications	DBSCAN	Wikipedia	Rounded Mean
Random Forests	Bag of Words	Mined Justifications	DBSCAN	Wikipedia	Mode
Random Forests	Bag of Words	Mined Justifications	K-Means	Wikipedia	Mean
Random Forests	Bag of Words	Mined Justifications	MeanShift	Wikipedia	Mean
Random Forests	Bag of Words	Full Document	N/A	Wikipedia	Mean
Random Forests	Bag of Words	Full Document	N/A	Wikipedia	Rounded Mean
Random Forests	Bag of Words	Full Document	N/A	Wikipedia	Mode

The output was returned in arrays of size 238, with an aggregated sentiment score for each filing. However, due to the size of this output, it is not included within this thesis itself. Upon observing the output, however, it was as expected. Values for the scores varied between -1 and 1, with float values in the mean aggregated returns and integers in the mode and rounded mean aggregated returns. Everything appeared to be working as intended. Also as expected, the calculation times for full document sentiment scores were much lower than for mined justifications, as these calculations did not utilise any deep learning to encode embeddings during justification mining. The calculations for sentiment scores of mined justifications were typically around 9 minutes (taking approximately 2.3 seconds per filing), whereas for full document sentiment scores the calculations were around 4 seconds (processing about 50 filings per second). This is to be expected, however, as Bag of Words is a very fast technique. The timing, while important to note, is not as important as the results from the variation correlation tests that are outlined later. It is in these correlation results where justification mining can be compared and contrasted to simply utilising full documents (which is faster). Sentiment scores from justification mining would need to prove more valuable or beneficial in these downstream tasks in order to defend their longer calculation times.

The next set of predictions also utilised the random forests trained classifier with Bag of Words numericisation, but this time the NLI dataset pre-trained model was utilised instead. This was the only change between this and the first set of predictions. Only the pre-trained model for Sentence-BERT was changed in order to have a solid baseline to compare the two pre-trained model datasets. If too many other variables were changed, it would not be possible to have an objective comparison between

the NLI and Wikipedia pre-trained SBERT models for this study. The full list of variable combination choices for the second set of predictions is listed in Table 18 below. The calculation times for both the mined justification predictions and full document predictions remained similar to the first set (around 9 minutes and 5 seconds respectively).

Table 18: Second set of variable combinations for predictions made on both the mined justifications and full documents obtained from all parsed MD&A Overview subsections.

Training Model	**Numericising Technique**	**Mined Justification or Full Sentiment Scores**	**Clustering Model used in Justification Mining**	**Pre-Trained SBERT model**	**Final Sentiment Aggregation Method**
Random Forests	Bag of Words	Mined Justifications	Agglomerative Clustering	NLI	Mean
Random Forests	Bag of Words	Mined Justifications	Agglomerative Clustering	NLI	Rounded Mean
Random Forests	Bag of Words	Mined Justifications	Agglomerative Clustering	NLI	Mode
Random Forests	Bag of Words	Mined Justifications	DBSCAN	NLI	Mean
Random Forests	Bag of Words	Mined Justifications	DBSCAN	NLI	Rounded Mean
Random Forests	Bag of Words	Mined Justifications	DBSCAN	NLI	Mode
Random Forests	Bag of Words	Mined Justifications	K-Means	NLI	Mean
Random Forests	Bag of Words	Mined Justifications	MeanShift	NLI	Mean
Random Forests	Bag of Words	Full Document	N/A	NLI	Mean
Random Forests	Bag of Words	Full Document	N/A	NLI	Rounded Mean
Random Forests	Bag of Words	Full Document	N/A	NLI	Mode

The next set is the final set of predictions using the Bag of Words method for feature engineering/numericisation. This time the Naïve Bayes trained model (since it performed well in Section 6) and the Wikipedia pre-trained model were employed. Again, the timings did not change from the first two sets. The full list of variables is found in Table 19. As discussed in previous subsections, bear in mind that all Bag of Words models utilised balancing for the class labels with the *class_weight* parameter.

Table 19: Third set of variable combinations for predictions made on both the mined justifications and full documents obtained from all parsed MD&A Overview subsections.

Training Model	**Numericising Technique**	**Mined Justification or Full Sentiment Scores**	**Clustering Model used in Justification Mining**	**Pre-Trained SBERT model**	**Final Sentiment Aggregation Method**
Naïve Bayes	Bag of Words	Mined Justifications	Agglomerative Clustering	Wikipedia	Mean
Naïve Bayes	Bag of Words	Mined Justifications	Agglomerative Clustering	Wikipedia	Rounded Mean
Naïve Bayes	Bag of Words	Mined Justifications	Agglomerative Clustering	Wikipedia	Mode
Naïve Bayes	Bag of Words	Mined Justifications	DBSCAN	Wikipedia	Mean
Naïve Bayes	Bag of Words	Mined Justifications	DBSCAN	Wikipedia	Rounded Mean
Naïve Bayes	Bag of Words	Mined Justifications	DBSCAN	Wikipedia	Mode
Naïve Bayes	Bag of Words	Mined Justifications	K-Means	Wikipedia	Mean
Naïve Bayes	Bag of Words	Mined Justifications	MeanShift	Wikipedia	Mean
Naïve Bayes	Bag of Words	Full Document	N/A	Wikipedia	Mean
Naïve Bayes	Bag of Words	Full Document	N/A	Wikipedia	Rounded Mean
Naïve Bayes	Bag of Words	Full Document	N/A	Wikipedia	Mode

After this, the subsequent sets of predictions focused on employing the Sentence-BERT transformer models to extract sentence embeddings for numericisation or feature engineering. I started with the random forests classifier that was trained on this type of embedding. Recall that another parameter was available with the Sentence-BERT models, which was whether or not the class labels were balanced. This first set applied balancing to the class labels with the *class_weight* parameter and utilised the Wikipedia dataset for the pre-trained SBERT model. Table 20 lists the options employed.

Table 20: Fourth set of variable combinations for predictions made on both the mined justifications and full documents obtained from all parsed MD&A Overview subsections.

Training Model	**Numericise Technique**	**Mined Justification or Full Sentiment Scores**	**Clustering Model used in Justification Mining**	**Pre-Trained SBERT model**	**Final Sentiment Aggregation Method**	**Balanced class labels or weight**
Random Forests	Sentence-BERT	Mined Justifications	Agglomerative Clustering	Wikipedia	Mean	Yes
Random Forests	Sentence-BERT	Mined Justifications	Agglomerative Clustering	Wikipedia	Rounded Mean	Yes
Random Forests	Sentence-BERT	Mined Justifications	Agglomerative Clustering	Wikipedia	Mode	Yes
Random Forests	Sentence-BERT	Mined Justifications	DBSCAN	Wikipedia	Mean	Yes
Random Forests	Sentence-BERT	Mined Justifications	DBSCAN	Wikipedia	Rounded Mean	Yes
Random Forests	Sentence-BERT	Mined Justifications	DBSCAN	Wikipedia	Mode	Yes
Random Forests	Sentence-BERT	Mined Justifications	K-Means	Wikipedia	Mean	Yes
Random Forests	Sentence-BERT	Mined Justifications	MeanShift	Wikipedia	Mean	Yes
Random Forests	Sentence-BERT	Full Document	N/A	Wikipedia	Mean	Yes
Random Forests	Sentence-BERT	Full Document	N/A	Wikipedia	Rounded Mean	Yes
Random Forests	Sentence-BERT	Full Document	N/A	Wikipedia	Mode	Yes

Now that Sentence-BERT, and consequently deep learning, was being deployed twice for the mined justifications options (once for the actual justification mining and again to numericise the text for input into the classifier model), the time taken also increased. For all the predictions on mined justifications, the average time was around 17 minutes with approximately 4.3 seconds per filing. The full sentiment scores also utilised Sentence-BERT and deep learning models once for numericisation and ended up averaging around 9 minutes for each variation with about 2.3 seconds per filing. This increase in time is to be expected. Another initial observation of note is that there was a lot more variation in the actual sentiment scores being printed between this set and the previous three sets that employed Bag of Words. Already, it can be seen that using sentence embeddings for feature engineering had an effect. This is further discussed in the evaluation of these results in subsequent subsections.

The next set of variations was the same as the previous set but this time without balanced class labels. This would mean that the *class_weight* parameter was set to *None* and no processing steps were performed during training of the classifier model to account for any bloated class labels. Again, only this one variable was changed to keep the comparison between balanced and unbalanced class labels objective. This set again used the random forests trained classifier and the Wikipedia pre-trained SBERT model. Table 21 summarises the details. Average training times were similar to the previous set that used balanced class weights. And again, a higher degree of variation in the sentiment scores was observed compared to the predictions made by the Bag of Words models.

Table 21: Fifth set of variable combinations for predictions made on both the mined justifications and full documents obtained from all parsed MD&A Overview subsections.

Training Model	**Numericise Technique**	**Mined Justification or Full Sentiment Scores**	**Clustering Model used in Justification Mining**	**Pre-Trained SBERT model**	**Final Sentiment Aggregation Method**	**Balanced class labels or weight**
Random Forests	Sentence-BERT	Mined Justifications	Agglomerative Clustering	Wikipedia	Mean	No
Random Forests	Sentence-BERT	Mined Justifications	Agglomerative Clustering	Wikipedia	Rounded Mean	No
Random Forests	Sentence-BERT	Mined Justifications	Agglomerative Clustering	Wikipedia	Mode	No
Random Forests	Sentence-BERT	Mined Justifications	DBSCAN	Wikipedia	Mean	No
Random Forests	Sentence-BERT	Mined Justifications	DBSCAN	Wikipedia	Rounded Mean	No
Random Forests	Sentence-BERT	Mined Justifications	DBSCAN	Wikipedia	Mode	No
Random Forests	Sentence-BERT	Mined Justifications	K-Means	Wikipedia	Mean	No
Random Forests	Sentence-BERT	Mined Justifications	MeanShift	Wikipedia	Mean	No
Random Forests	Sentence-BERT	Full Document	N/A	Wikipedia	Mean	No
Random Forests	Sentence-BERT	Full Document	N/A	Wikipedia	Rounded Mean	No
Random Forests	Sentence-BERT	Full Document	N/A	Wikipedia	Mode	No

After this, a set of predictions was made using the random forests model, feature engineering via Sentence-BERT, balanced class weights and this time the NLI pre-trained SBERT model instead of the Wikipedia pre-trained model (for both numericisation and justification mining). Table 22 lists the options.

Table 22: Sixth set of variable combinations for predictions made on both the mined justifications and full documents obtained from all parsed MD&A Overview subsections.

Training Model	Numericise Technique	Mined Justification or Full Sentiment Scores	Clustering Model used in Justification Mining	Pre-Trained SBERT model	Final Sentiment Aggregation Method	Balanced class labels or weight
Random Forests	Sentence-BERT	Mined Justifications	Agglomerative Clustering	NLI	Mean	Yes
Random Forests	Sentence-BERT	Mined Justifications	Agglomerative Clustering	NLI	Rounded Mean	Yes
Random Forests	Sentence-BERT	Mined Justifications	Agglomerative Clustering	NLI	Mode	Yes
Random Forests	Sentence-BERT	Mined Justifications	DBSCAN	NLI	Mean	Yes
Random Forests	Sentence-BERT	Mined Justifications	DBSCAN	NLI	Rounded Mean	Yes
Random Forests	Sentence-BERT	Mined Justifications	DBSCAN	NLI	Mode	Yes
Random Forests	Sentence-BERT	Mined Justifications	K-Means	NLI	Mean	Yes
Random Forests	Sentence-BERT	Mined Justifications	MeanShift	NLI	Mean	Yes
Random Forests	Sentence-BERT	Full Document	N/A	NLI	Mean	Yes
Random Forests	Sentence-BERT	Full Document	N/A	NLI	Rounded Mean	Yes
Random Forests	Sentence-BERT	Full Document	N/A	NLI	Mode	Yes

Unsurprisingly, the time taken for making these predictions did not change between the NLI set and the Wikipedia sets. Also of interest is that, just as in the Wikipedia sets, the variation in sentiment scores was greater than the Bag of Words approaches. This might mean that regardless of which pre-trained model is chosen for Sentence-BERT, models trained using numericisation with SBERT will all be closer to each other in outcomes than they are to the outcomes from the Bag of Words methods. This was promising, as it was an early indication that there was a marked difference between numericisation using the sentence embeddings methods and the Bag of Words methods. However, further conclusions could not be drawn before carrying out the evaluation steps (see Section 7.7).

I moved on to utilising a different classifier model, Support Vector Machines. The numericisation method remained sentence embeddings from Sentence-BERT, this time using the pre-trained Wikipedia model. In this set, class labels were also balanced. Table 23 outlines the variables for this set.

Table 23: Seventh set of variable combinations for predictions made on both the mined justifications and full documents obtained from all parsed MD&A Overview subsections.

Training Model	Numericise Technique	Mined Justification or Full Sentiment Scores	Clustering Model used in Justification Mining	Pre-Trained SBERT model	Final Sentiment Aggregation Method	Balanced class labels or weight
SVM	Sentence-BERT	Mined Justifications	Agglomerative Clustering	Wikipedia	Mean	Yes
SVM	Sentence-BERT	Mined Justifications	Agglomerative Clustering	Wikipedia	Rounded Mean	Yes
SVM	Sentence-BERT	Mined Justifications	Agglomerative Clustering	Wikipedia	Mode	Yes
SVM	Sentence-BERT	Mined Justifications	DBSCAN	Wikipedia	Mean	Yes
SVM	Sentence-BERT	Mined Justifications	DBSCAN	Wikipedia	Rounded Mean	Yes
SVM	Sentence-BERT	Mined Justifications	DBSCAN	Wikipedia	Mode	Yes
SVM	Sentence-BERT	Mined Justifications	K-Means	Wikipedia	Mean	Yes
SVM	Sentence-BERT	Mined Justifications	MeanShift	Wikipedia	Mean	Yes
SVM	Sentence-BERT	Full Document	N/A	Wikipedia	Mean	Yes
SVM	Sentence-BERT	Full Document	N/A	Wikipedia	Rounded Mean	Yes
SVM	Sentence-BERT	Full Document	N/A	Wikipedia	Mode	Yes

The first thing I noticed from the output was that the predictions between this SVM set and the previous random forests sets varied significantly. It would be interesting to note how this might impact correlation to future returns or similarity between sentiment scores from mined justifications and full documents. One thing that did not change, however, was the time taken to complete the predictions, which remained very similar to the other Sentence-BERT numericisation based sets.

Because the Support Vector Machine set varied so much from the random forests models, it was decided to do one final set with this classifier model. Thus, the very final set of predictions utilised the SVM trained model, with sentence embeddings from Sentence-BERT for features, balanced class weights and the NLI pre-trained model for SBERT instead of Wikipedia this time. The variations are listed in Table 24.

Table 24: Eighth and final set of variable combinations for predictions made on both the mined justifications and full documents obtained from all parsed MD&A Overview subsections.

Training Model	Numericise Technique	Mined Justification or Full Sentiment Scores	Clustering Model used in Justification Mining	Pre-Trained SBERT model	Final Sentiment Aggregation Method	Balanced class labels or weight
SVM	Sentence-BERT	Mined Justifications	Agglomerative Clustering	NLI	Mean	Yes
SVM	Sentence-BERT	Mined Justifications	Agglomerative Clustering	NLI	Rounded Mean	Yes
SVM	Sentence-BERT	Mined Justifications	Agglomerative Clustering	NLI	Mode	Yes
SVM	Sentence-BERT	Mined Justifications	DBSCAN	NLI	Mean	Yes
SVM	Sentence-BERT	Mined Justifications	DBSCAN	NLI	Rounded Mean	Yes
SVM	Sentence-BERT	Mined Justifications	DBSCAN	NLI	Mode	Yes
SVM	Sentence-BERT	Mined Justifications	K-Means	NLI	Mean	Yes
SVM	Sentence-BERT	Mined Justifications	MeanShift	NLI	Mean	Yes
SVM	Sentence-BERT	Full Document	N/A	NLI	Mean	Yes
SVM	Sentence-BERT	Full Document	N/A	NLI	Rounded Mean	Yes
SVM	Sentence-BERT	Full Document	N/A	NLI	Mode	Yes

The first immediate observation was that this time the predictions did again vary from the predictions made from the random forests models, but surprisingly not as much as with the previous SVM set that employed the Wikipedia pre-trained SBERT model. Another thing to note was that while the time taken for the predictions on mined justifications remained similar to the other SBERT sets, the time taken for the predictions on the full document actually increased to an average of 10 minutes per variation and 2.6 seconds per filing. While this need not be of particular significance, it is still noteworthy.

That concludes the bulk of the predictions made for this section of the thesis. Recall that each of the 11 variations per set listed above outputs 238 predictions (one per each filing) - too much information to include within the thesis itself. In each set, there are some notable patterns. For instance, the mined justification prediction time taken would always be greater than the full document predictions. This is likely caused by utilising Sentence-BERT deep learning models to encode sentence embeddings for justification mining in those variations. Deep learning models often take much more time than simple models to run, even when employing CUDA and GPU processing. Finally, the data, as discussed before, was consolidated into one large Pandas dataframe, which ended up with 92 columns to incorporate all the predictions from the sets discussed above (but remained with the original number of 238 rows). Since each row still represented one individual filing, this also made organisation much simpler for subsequent calculations.

7.6.5 Calculating Future Stock Returns

With all the necessary predictions completed, it was time to move onto calculating future returns so that these could be correlated to the predictions from the previous subsection. A new Python module file was created to house all the functions related to calculating stock price returns. The first step was to devise a function that could get the stock price on any given date for any ticker/stock. For this, the Yahoo database was chosen using the *pandas_datareader* library. This library has a function, *get_data_yahoo*, that would return a dataframe of stock prices when given parameters of the stock ticker and a start and end date. In order to get a single day's stock price, the start and end date were simply set to the same day. After that, the function was set to extract and return the adjusted closing price from that date. This price was chosen as the adjusted closing price encompasses any corporate actions taken and can be suitable for analysing historical performance.

With this function, it was possible to calculate the stock price for all the tickers in my dataset on the dates that the filings were published (information which was stored in my main Pandas dataframe). With this out of the way, however, it was still necessary to devise a method for obtaining the stock price at a date a certain amount of time after the publication date. In order to cover a wide variety of options so that the predictions from the previous subsection could be correlated to the returns in the stock over numerous time windows, several functions were created in the module file. Recall that when the MD&A Overview data was first imported into the Jupyter Notebook file, the dates were parsed into *datetime* format. Because of this, the *datetime* library can be employed to make adjustments to the date. So, the first function written was used to add a certain number of years to a date. It would take as input the date itself and the number of years to add to it. This would be useful for calculating stock returns after a year following the publication of a particular 10-K filing. This function would first try to use the *replace* method in the *datetime* library to replace the years part of a datetime object with the original year plus however many years were selected to be added. This, however, would sometimes raise a *ValueError*, in which case an alternate method would be returned where a year would be directly added to the input date using the *date* method from the *datetime* library (this was performed by simply using the *date* method to add *X* number of years to the input date and then subtract the input date to get *X* number of years, which is then re-added to the original date directly without the *date* method). This try/except specification was able to cover all cases and, through unit testing, was found to work for all dates inputted.

Similar to adding years, the second function was set to add months to an input date. As before, it would take a date (such as the publication date of a 10-K filing) as input along with the number of months to add to it. This function utilised the *relativedelta* method from the *dateutil relativedelta* module. With this method, the number of months specified in the function parameters could be added directly to the date. The final function written was used to add days to the input date. It worked similar to the previous function for adding months, where an input date and a number of days to add would be the parameters. This time, however, the *timedelta* method from the *datetime* library was employed. Like the *relativedelta* method, it was able to add days directly to the date inputted.

Lastly, a percent change formula was coded into the module file, that would take the stock price at the date a 10-K was published along with a future stock price at a specified future date and calculate the stock return between those two dates. Unit tests were employed using Google (GOOGL) and various

start and end dates to make sure all the functions within the module file worked as intended. Manual checking proved that they did work in these initial sanity checks.

With these functions at hand, it was possible to begin calculating all the necessary stock prices required for the stock returns section in the Jupyter Notebook file. The idea was to input the date any given 10-K filing was published (taken from the main Pandas dataframe) and add a certain time period to it. Finally, the stock return between these two periods would be calculated and returned. In order to limit the number of calculations, five time windows were chosen: one week, one month, three months, six months and one year after the publication date of a 10-K filing. These time periods would adequately cover both long and short term returns to test if there is any correlation to any of the predictions made in the previous subsection.

However, once the code was run, an error arose. The functions were unable to obtain the stock price at certain dates in the future. Upon printing out the results along with performing manual checks, it became apparent that this was because any given future date might fall on a day of the week on which markets are closed, or a holiday or any other reason why a price does not exist that date. To solve this, a new function was created within the Notebook. This function would take as input a ticker and a date. It would then use a *while True* loop to try to find the stock price. Within the loop the function would first try to get the stock price from the adjusted close price function written in the module file and if a price was obtained then break. If, however, a *KeyError* occurred, then the date would be updated by adding a single day using the add days function in the module file, after which the loop would restart, and the function would try to get the stock price on this new date (which was the old date plus one day). It would keep adding days until it found a date where a stock price was available. Finally, it would return this stock price and the date on which the price occurred. Using this function would allow me to find the next available date where the markets were open and a stock price is available.

After this, the first step would be to get the stock prices at the date of publication for all the 10-K filings in the dataset. Two new empty lists were created to house both the stock prices returned and the date on which they occurred (in case the markets were closed on the date a 10-K filing was published and the next day, for instance, had to be utilised). Then, I iterated over the rows of the pandas dataframe using the *iterrows()* function and used the functions I had written to obtain the stock price for every ticker on or near the date the 10-K filing was published.

The next step would be to obtain the stock price at the end date of the chosen time windows (one week, one month, three months, six months and one year). Again, ten new empty lists were instantiated to house the dates and stock prices obtained from the subsequent loops. *For* loops and the *iterrows()* method were again employed to iterate over the dataframe. Because I had previously saved all the dates at which the 10-K filings were published into a list in the preceding step, I could simply add a period of time to every date in this list. So, for instance, to calculate the date one week after this date, I could use the *add_days* function from my module file to add 7 days to all the dates on the list and save these dates in a new list. To add a year, I could do the same but with the *add_years* function instead (similarly for the *add_months* function). With this new list of dates in hand, all that remained was to feed these new dates into my function (with the *while* loop) along with the ticker obtained from each row of the dataframe and it would return a new stock price and date. These new stock prices and dates could be appended to the new lists that were created for each time window. This process was

repeated for each of the five time windows of interest (one week, one month, three months, six months and one year), and all the information was saved into the corresponding lists for organisation.

One issue that occurred during this process was an error that was raised when calculating prices for one year after the 10-K filings publication dates. Upon examining the error that was returned, I realised that a single filing at index 209 for Starbucks was published very late in the year. Thus, to get its stock price for the following year was impossible as it had not occurred yet at that time. As a result, this sample point had to be removed from the dataset. The original dataset actually contained 239 filings (including this late Starbucks filing for the year 2019) but was modified to 238 once it was removed (the number listed in all the previous subsections). This has not been mentioned until now to avoid any confusion.

Finally, all that remained was to calculate the stock returns themselves between the date the 10-K filings were published and the new dates saved in the five freshly populated lists. First, as with before, for the purposes of organisation, new empty lists were created to house the output of all the returns calculations, one for each time window. Then the *zip* function was employed to zip together the stock prices at the date of publication and the stock price at a given future date (one week to one year). Using a *for* loop, these were then iteratively passed into the function created in my module file to calculate stock returns. Finally, these returns were appended to their corresponding list. This was performed five times for each time window. Using these now filled-in lists, the one week, one month, three months, six months and one-year stock returns were then appended as new columns to my main Pandas dataframe (which was now 97 columns wide). In this way, all the relevant information from my original data and all the output from the numerous calculations performed was organised into one dataframe with pertinent information arranged by filing (each row was a separate 10-K filing). Not only would this allow for better organisation overall, but calculations over a Pandas dataframe were often faster than other data structures that could have been employed. This would be useful for all the calculations that were necessary for evaluation in the next subsections.

Also of note, the column names for the stock returns were saved in a new list. Previously the column names for the mined justification sentiment predictions and full document sentiment predictions (as well as one for both combined) were also saved in lists. These lists would help perform the next steps for evaluating the results. Now that all the experimental calculations were done, the main Pandas dataframe was saved into a pickle file so that it could be easily read in between work sessions. The following subsection (Section 7.7) outlines the calculations for various evaluation methods.

One other step that was taken, although it was not necessary to complete the research investigation, was to modify the code written to better align it with the standards of how Python modules related to machine learning are typically released. This mostly involved rewriting all the functions into classes and re-organising the module files written into new module files with one class per file. This was completed and the final module files uploaded onto Github. The titles of these files were also changed to align with typical practices (lower case module file names with underscores instead of spaces and Uppercase class names with no spaces). The *.csv* data files, Jupyter Notebook files, and pickle saves were also uploaded. Finally, a Readme was written as well.

7.7 Experiment Results and Performance Metrics – Justification Mining

With all the experimental calculations completed and the data saved into a single Pandas dataframe, it was time to begin calculating the evaluation metrics outlined in previous subsections.

As discussed in earlier subsections, two primary forms of evaluation were chosen, both with different goals in mind. The first arose through discussions with Professor Stephen Roberts and Professor Amir Amel-Zadeh (Roberts, 2020; Amel-Zadeh, 2020). The idea behind this first form of evaluation would be to create an objective downstream task for the sake of evaluating the usefulness of justification mining. Typical evaluation methods for clustering models are often subjective in nature and do not offer any meaningfully practical method for judging the utility of a model (Roberts, 2020). An ideal downstream task given the type of data used (10-K financial documents) would be a correlation task between the sentiment scores and future returns (Roberts, 2020; Amel-Zadeh, 2020). Since financial data was employed for this research, future returns provide the objective evaluation desired, while also providing the potential for justification mining to have a tangible and practical benefit if successful. Moreover, I can also compare the correlation to future returns between mined justification sentiment predictions and full document sentiment predictions. This evaluation method primarily helps address research question 3 and sub-question 3-1.

The second form of evaluation arose while preparing the data and performing the initial trials and unit testing for justification mining. If one of the goals of justification mining is to capture or represent the sentiment of a full document, it would prove useful to compare the sentiment scores obtained through mined justification versus those from full documents (Roberts, 2020; Simpson, 2020). The research investigation was already making predictions with a trained classifier on both mined justification and full documents (as part of the correlation to future returns analysis), thus, the data necessary to perform this evaluation was already available. This second evaluation was chosen mainly as a means to answer research question 2, along with sub-questions 2-1 and 2-2. Research question 2 is arguably the most important question and goal of this research and so providing well-structured forms of evaluation here is of great importance. This is because this research question and subsequent evaluation could have wide-reaching implications outside of finance. Capturing the majority of sentiment in a document has many uses, such as in the field of AI Explainability or sentiment summarisation (which is discussed further in Sections 9 and 11). Successful sentiment summarisation methodology would allow justification mining to become a means to present understandable and concise tools for explaining machine learning strategies to investors or act as a starting point for providing information to help make more ethically informed decisions in finance (which address research sub-question 2-3). Section 11 discusses this further. For now, I begin by outlining the process for both these forms of evaluation.

7.7.1 Correlation to Future Stock Returns

Since both the sentiment scores from all the variations of predictions made earlier, as well as the stock returns over multiple time windows, had already been calculated and stored, the next step was to determine a suitable correlation method. Correlation coefficients are tools for measuring the strength of association between two variables. For this thesis, the Pearson correlation method (Pearson, 1895) was chosen. It is a popular method for use in correlation analyses within the field of finance (Kenett et al., 2014; Rajan and Victor, 2014; Ranco et al., 2015; Guo et al., 2018; Wang et al., 2018).

Pearson correlation is used to gauge the power of two variables' linear relationship. The main coefficient of this test is named *r* and takes values from -1 to 1, with 0 signalling no correlation between the variables. A correlation between 0 and 1 implies the two variables increase or decrease together, whereas an *r* value between 0 and -1 implies the opposite; namely, as one variable increases, the other decreases. Values closer to 1 and -1 indicate a stronger correlation between the chosen variables and values close to 0 imply a weaker relationship (Kenett et al., 2014; Wang et al., 2018). The formula for calculating the Pearson correlation coefficient, as given in the scipy documentation, is as follows (scipy.stats.pearsonr, 2020):

$$r = \frac{\sum(x - m_x)(y - m_y)}{\sqrt{\sum(x - m_x)^2(y - m_y)^2}}$$

where *x* and *y* are the vectors containing data for the two variables being compared, and *m* is the mean of the two vectors respectively (scipy.stats.pearsonr, 2020).

Utilising Pearson correlation to compare sentiment to future returns is not a novel concept (Rajan and Victor, 2014; Ranco et al., 2015). Calculating correlation between a signal and future stock returns allows for a simple and fast gauge of the accuracy of its predictions. Compared to a backtest, a Pearson correlation might be more suitable when only a single number is required for evaluation. Pearson correlation, though, does have limitations compared to backtesting, for example not considering factors such as transaction costs (Rajan and Victor, 2014; Ranco et al., 2015). However, as mentioned previously, given the time constraints due to manual parsing, getting enough data to create a portfolio and perform proper backtesting was not an option within the scope of this thesis. Thus, Pearson correlation was a suitable choice and even had some advantages over portfolio optimisation.

In order to determine if the results from the correlation analysis are statistically significant, a p-value must be calculated and a benchmark set. The null hypothesis for this test is that the two variables have no correlation. P-values fall in the range of 0 to 1 and denote the probability of the data occurring in the case of the null hypothesis being true. Thus, a low p-value allows the rejection of the null hypothesis. Typically, in economic analyses like this, p-values of 0.05 or less are determined to be statistically significant. Statistical significance will also assess whether enough data was used in this research investigation (Kenett et al., 2014; Guo et al., 2018; Wang et al., 2018).

In this study, the two variables being tested are the sentiment scores from various predictions and the corresponding future returns in several different time windows. To calculate the *r* and *p-values* with a Pearson correlation test, a nested for loop was employed. Recall that several lists were made during the research experimentation. These lists included: one with all the column names for columns containing the stock returns over the time windows utilised, one with all the column names for columns containing the predictions for all sentiment analyses performed, one with the column names for columns containing sentiment analyses on only mined justifications, and finally one with the column names for columns containing sentiment analyses on only full MD&A Overview documents.

With these lists, nested for loops were the easiest option to do evaluations on all possible combinations (as they would perform the correlation analysis between all sets of sentiment predictions and a given time window of stock returns before moving on to the next time window).

To elaborate, the first set of calculations would be to perform the Pearson correlation analysis on the stock returns from all time windows with sentiment scores from all sentiment analyses performed. The first for loop would iterate through the returns from all time windows (one week, one month, three months, six months and one year returns following the 10-K publication date), and the nested for loop below that would iterate through all the sentiment score predictions. These column names would then iteratively be fed into the *pearsonr* function from *scipy.stats* and the *r* and *p* values would be returned and printed. This outputted the entire list of all possible combinations, however, and was too much data to draw any meaningful conclusions. For the full list of outputs, please see Appendix 2 (Section 14.2).

Thus, a new nested for loop was utilised, but this time printing only those lines whose p-value was less than or equal to 0.05. This new loop would, thus, print only those combinations where there was statistical significance. It would also be good to save those combinations into a data structure for future reference and organisation. Several options for doing so were available, such as a dictionary, tuples or a list of lists. As dictionaries are generally faster (optimised for key searching), they were tried first. A *count* was also coded into the for loop as a unit test to make sure the length of the dictionary matched the *count*. The nested loop was written in the same way, but this time before the print statement, an if statement was added to limit the results to those where $p \leq 0.05$. Besides printing as before, the information was also saved to this instantiated empty dictionary as key/value pairs, with the keys being the sentiment analysis variables information and the values being the time window chosen and *r* and *p* values calculated. However, while the *count* was 70, the length of the dictionary was only 25. The dictionary was saving multiple sets of values (i.e. calculations) to the same key because dictionaries could only save one unique key. Instead, an empty list was instantiated. A nested for loop was written as before, but this time the information was saved simply as a list which was then appended to the empty list instantiated previously. Included in this list were the sentiment analysis variables, the time window and the *r* and *p* values. This time the count matched the length of the list of lists, and the data was still saved and organised successfully. Table 25 contains the output from this new loop.

Table 25: Table containing all the combinations within the Pearson correlation analysis where $p \leq 0.05$. Detailed explanation of key presented first and table afterwards.

In the following table, the first column contains notation for the variables utilised in the sentiment analysis for that row. The following key explains what the various notations mean (these notations are used in all tables within Section 7 and corresponding Appendices):

First value:

rf = random forest classifier model used

nb = Naïve Bayes classifier model used

svm = Support Vector Machines classifier model used

Second Value:

SBERT = transformer models/Sentence-BERT were used for feature engineering and numericisation of text data for input into machine learning models

bow = Bag of Words method used for numericising text data

Third Value:

bal = training class labels were balanced with the *class_weight* hyperparameter so the effects of discrepancies between the number of any given label would be minimised

unbal = training class labels were not balanced

Fourth Value:

wiki = the Wikipedia pre-trained model was utilised for encoding Sentence-BERT embeddings for both encoding text into semantically meaningful sentence embeddings during justification mining as well as numericisation of text for input into classifier models

nli = the NLI pre-trained model instead was utilised for encoding Sentence-BERT embeddings for both tasks as explained above

Fifth Value:

mined = the sentiment analysis was performed on the mined justifications

full = the sentiment analysis was performed on the full MD&A Overview subsections

Sixth Value:

mean = mean aggregation utilised

mode = mode aggregation utilised

meanr = rounded mean aggregation utilised

Seventh Value (only applicable when using mined justifications):

agglo = Agglomerative model used for clustering during justification mining

kmeans = K-means model used instead

dbscan = DBSCAN model used instead

meanshift = MeanShift model used instead

To walk through an example, the first combination is *rf_SBERT_unbal_wiki_mined_mode_dbscan*. Here, rf means the random forests classifier was employed, SBERT refers to using transformer models and Sentence-BERT for numericisation of text, unbal is utilising unbalanced training class labels, wiki refers to employing the Wikipedia pre-trained model for Sentence-BERT, mined means sentiment analysis was performed on the mined justifications and not the full MD&A Overview documents, mode refers to using mode aggregation to obtain the final text sentiment score, and, finally, dbscan means utilising the DBSCAN clustering model during justification mining.

The second column in the table simply denotes which time window was used to calculate stock returns for that row (one week, one month, three months, six months, or one year). The table is sorted

by this second column. The third column contains the calculated Pearson coefficient, *r*, for that row and the fourth and final column contains the calculated p-value for that row.

Table 25: Actual table starts here.

Sentiment Analysis Variables	Returns Window	Pearson coefficient	p-values
rf_SBERT_unbal_wiki_mined_mode_dbscan	pct_change_1_year	r=-0.12756625154343026	p=0.04933637974513266
rf_SBERT_unbal_wiki_full_mean	pct_change_1_year	r=-0.12746288509330575	p=0.04952229551400624
rf_SBERT_bal_nli_mined_mean_agglo	pct_change_1_year	r=-0.20664250471102646	p=0.0013468156664763686
rf_SBERT_bal_nli_mined_mean_dbscan	pct_change_1_year	r=-0.13339623373452777	p=0.0397535050629828
rf_SBERT_bal_nli_mined_meanr_dbscan	pct_change_1_year	r=-0.13339623373452777	p=0.0397535050629828
rf_SBERT_bal_nli_mined_mode_dbscan	pct_change_1_year	r=-0.13339623373452777	p=0.0397535050629828
rf_SBERT_bal_nli_mined_mean_kmeans	pct_change_1_year	r=-0.1409664258189341	p=0.02969443385555039
rf_SBERT_bal_nli_mined_mean_meanshift	pct_change_1_year	r=-0.15363108490311284	p=0.017705314445628294
rf_SBERT_bal_nli_full_mean	pct_change_1_year	r=-0.1906106530887367	p=0.00315486848458724
svm_SBERT_bal_nli_full_mean	pct_change_1_year	r=-0.15290713497332828	p=0.018254544605497046
svm_SBERT_bal_nli_full_mode	pct_change_1_year	r=-0.15107342942741173	p=0.019712483787280955
rf_SBERT_bal_wiki_mined_mean_kmeans	pct_change_3_months	r=-0.14224912064206147	p=0.028226180098434608
rf_SBERT_bal_wiki_full_mean	pct_change_3_months	r=-0.16079402769397458	p=0.01300136313692643
rf_SBERT_unbal_wiki_mined_mean_agglo	pct_change_3_months	r=-0.14728946234591553	p=0.023043210181581647
rf_SBERT_unbal_wiki_mined_mean_dbscan	pct_change_3_months	r=-0.13570680820981137	p=0.03641626021573769
rf_SBERT_unbal_wiki_mined_meanr_dbscan	pct_change_3_months	r=-0.1466623612661396	p=0.023639702532141753
rf_SBERT_unbal_wiki_mined_mode_dbscan	pct_change_3_months	r=-0.13153725681557832	p=0.04262238760582463
rf_SBERT_unbal_wiki_mined_mean_kmeans	pct_change_3_months	r=-0.15855815276704024	p=0.014335358190428615
rf_SBERT_unbal_wiki_full_mean	pct_change_3_months	r=-0.1659714393889948	p=0.010323071438399539
rf_SBERT_bal_nli_mined_mean_agglo	pct_change_3_months	r=-0.18582181445221208	p=0.004018277427982355
rf_SBERT_bal_nli_mined_mean_dbscan	pct_change_3_months	r=-0.13224798173170968	p=0.04150565426558646
rf_SBERT_bal_nli_mined_meanr_dbscan	pct_change_3_months	r=-0.13224798173170968	p=0.04150565426558646
rf_SBERT_bal_nli_mined_mode_dbscan	pct_change_3_months	r=-0.13224798173170968	p=0.04150565426558646
rf_SBERT_bal_nli_mined_mean_kmeans	pct_change_3_months	r=-0.1399542095930891	p=0.03089869431098952
rf_SBERT_bal_nli_mined_mean_meanshift	pct_change_3_months	r=-0.1974753785088498	p=0.0022085084108392516
rf_SBERT_bal_nli_full_mean	pct_change_3_months	r=-0.19721525703277817	p=0.0022390330031892645
svm_SBERT_bal_nli_mined_mean_meanshift	pct_change_3_months	r=-0.1666858416230928	p=0.009994717341568097
svm_SBERT_bal_nli_full_mean	pct_change_3_months	r=-0.1866136900330011	p=0.003862210479074706
svm_SBERT_bal_nli_full_mode	pct_change_3_months	r=-0.22021219752588173	p=0.000622814609893812
rf_SBERT_bal_wiki_full_mean	pct_change_6_months	r=-0.15994546656821837	p=0.013494217768917304
rf_SBERT_unbal_wiki_mined_mean_dbscan	pct_change_6_months	r=-0.1376702159498808	p=0.03376986153133213
rf_SBERT_unbal_wiki_mined_meanr_dbscan	pct_change_6_months	r=-0.16979885392646893	p=0.008669336428914473
rf_SBERT_unbal_wiki_mined_mode_dbscan	pct_change_6_months	r=-0.16264182705261565	p=0.011982536633425108
rf_SBERT_unbal_wiki_mined_mean_kmeans	pct_change_6_months	r=-0.1602572503737491	p=0.013311255737247223
rf_SBERT_unbal_wiki_full_mean	pct_change_6_months	r=-0.1739187697862103	p=0.007156141018647447
rf_SBERT_bal_nli_mined_mean_agglo	pct_change_6_months	r=-0.1845129498324647	p=0.004288743206778107

rf_SBERT_bal_nli_mined_mean_kmeans	pct_change_6_months	r=-0.14804608507422795	p=0.022340846615377226
rf_SBERT_bal_nli_mined_mean_meanshift	pct_change_6_months	r=-0.15565420834067262	p=0.01624606605461402
rf_SBERT_bal_nli_full_mean	pct_change_6_months	r=-0.1900058803398099	p=0.003253746961023152
svm_SBERT_bal_nli_full_mean	pct_change_6_months	r=-0.16603160290229121	p=0.010295055755102856
rf_SBERT_bal_wiki_mined_mean_agglo	pct_change_1_month	r=-0.12727303404038254	p=0.049865284402722726
rf_SBERT_bal_wiki_mined_mean_kmeans	pct_change_1_month	r=-0.18714263492173033	p=0.0037610334535998474
rf_SBERT_bal_wiki_full_mean	pct_change_1_month	r=-0.1652693073456413	p=0.010655044421736395
rf_SBERT_unbal_wiki_mined_mean_agglo	pct_change_1_month	r=-0.14441891695220363	p=0.025883908052659013
rf_SBERT_unbal_wiki_mined_mean_kmeans	pct_change_1_month	r=-0.15004771909482076	p=0.02057126652608078
rf_SBERT_unbal_wiki_full_mean	pct_change_1_month	r=-0.1564469869924304	p=0.015703551284572285
rf_SBERT_bal_nli_mined_mean_agglo	pct_change_1_month	r=-0.14062787682406375	p=0.030092672416217044
rf_SBERT_bal_nli_mined_mean_dbscan	pct_change_1_month	r=-0.17421202335832384	p=0.00705800649762465
rf_SBERT_bal_nli_mined_meanr_dbscan	pct_change_1_month	r=-0.17421202335832384	p=0.00705800649762465
rf_SBERT_bal_nli_mined_mode_dbscan	pct_change_1_month	r=-0.17421202335832384	p=0.00705800649762465
rf_SBERT_bal_nli_mined_mean_kmeans	pct_change_1_month	r=-0.13922805502043445	p=0.03178806774464829
rf_SBERT_bal_nli_mined_mean_meanshift	pct_change_1_month	r=-0.24914139302513652	p=0.0001024291066417152
rf_SBERT_bal_nli_full_mean	pct_change_1_month	r=-0.20610191447918774	p=0.0013874885171142937
svm_SBERT_bal_nli_mined_mean_dbscan	pct_change_1_month	r=-0.21747774701843273	p=0.0007303095449552845
svm_SBERT_bal_nli_mined_meanr_dbscan	pct_change_1_month	r=-0.21747774701843273	p=0.0007303095449552845
svm_SBERT_bal_nli_mined_mode_dbscan	pct_change_1_month	r=-0.21747774701843273	p=0.0007303095449552845
svm_SBERT_bal_nli_mined_mean_meanshift	pct_change_1_month	r=-0.25338429745264657	p=7.71298815221731e-05
svm_SBERT_bal_nli_full_mean	pct_change_1_month	r=-0.2309768938682398	p=0.00032653089329424636
svm_SBERT_bal_nli_full_mode	pct_change_1_month	r=-0.14463875863912198	p=0.025656186466183815
nb_bow_wiki_full_mean	pct_change_1_week	r=0.15023977341800443	p=0.020408044175353513
rf_SBERT_bal_wiki_mined_meanr_agglo	pct_change_1_week	r=-0.14625724614651628	p=0.024032081961429325

From this first set of evaluation metrics, there are already several patterns emerging. First, it is safe to say that enough combinations had statistically significant p-values to demonstrate that a sufficient amount of data was utilised for this research investigation. Several other patterns emerge when analysing the first column. Most of the combinations occurred when the random forests classifier was used for training and predicting. Even more noteworthy is that in all but one of the combinations, transformer models and Sentence-BERT were used for the feature engineering and numericisation of text step. Recall that Sentence-BERT uses pre-trained models to encode text into semantically meaningful sentence embeddings. Unlike with Bag of Words, using Sentence-BERT for numericisation removes the need to truncate or pad sentences to be the same length and the need to remove any words that appear in the test set that were not present in the training set. This could explain why the overwhelming majority of combinations utilised Sentence-BERT based numericisation, as no data was lost in the process of creating the sentence embeddings, which might allow for these embeddings to better inform future stock returns. This helps begin to answer research sub-question 3-1 and is expanded upon later in this subsection.

Another important pattern that emerges is that the majority of combinations also derived the aggregated sentiment scores via performing sentiment analysis on mined justifications rather than full

documents. This observation illustrates the added benefit a process like justification mining could offer, which directly relates to answering research question 3. It does appear that justification mining provides advantage over using full documents for sentiment analysis in apprising future returns. This could be, in part, due to mined justifications summarising the sentiment from the full document and possibly eliminating the noise in the sentiment signal. Investigating this possibility would be a good direction for continued or future work.

On the other hand, there were also several areas where the above results proved inconclusive. No patterns could be detected in which pre-trained model was used for Sentence-BERT encoding, in which aggregation method was used for the final sentiment scores, in which clustering model was used for justification mining, nor in which time window was used to calculate future stock returns. One thing to note, however, is that most of the Pearson coefficient, *r*, values here are very small, signalling weak correlation between the aggregated sentiment scores and future stock returns for that company. Because of this it was necessary to re-filter the data.

The final filter would be to take the full list and return only those combinations where the Pearson correlation coefficient was >0.2 or <-0.2 in order to check which combinations had larger correlation scores. Again, a nested for loop was utilised in the same manner as before and the data was saved into a list of lists for future use. Table 26 below contains these combinations along with the corresponding *r* and *p* values. Note that one final filter could have been applied, returning combinations where both r was >0.2 or <-0.2 and p was ≤0.05 (effectively combining both filters) but, as can be seen in the table below, all the p-values were already ≤0.05, so this final filter was unnecessary. Refer to the key given above in Table 25 to identify the sentiment analysis variable notations used.

Table 26: Table containing all the combinations within the Pearson correlation analysis where r>0.2 or r<-0.2.

Sentiment Analysis Variables	**Returns Window**	**Pearson coefficient**	**p-values**
rf_SBERT_bal_nli_mined_mean_agglo	pct_change_1_year	r=-0.20664250471102646	p=0.0013468156664763686
svm_SBERT_bal_nli_full_mode	pct_change_3_months	r=-0.22021219752588173	p=0.000622814609893812
rf_SBERT_bal_nli_mined_mean_meanshift	pct_change_1_month	r=-0.24914139302513652	p=0.0001024291066417152
rf_SBERT_bal_nli_full_mean	pct_change_1_month	r=-0.20610191447918774	p=0.0013874885171142937
svm_SBERT_bal_nli_mined_mean_dbscan	pct_change_1_month	r=-0.21747774701843273	p=0.0007303095449552845
svm_SBERT_bal_nli_mined_meanr_dbscan	pct_change_1_month	r=-0.21747774701843273	p=0.0007303095449552845
svm_SBERT_bal_nli_mined_mode_dbscan	pct_change_1_month	r=-0.21747774701843273	p=0.0007303095449552845
svm_SBERT_bal_nli_mined_mean_meanshift	pct_change_1_month	r=-0.25338429745264657	p=7.71298815221731e-05
svm_SBERT_bal_nli_full_mean	pct_change_1_month	r=-0.2309768938682398	p=0.0003265308932942463 6

In narrowing down the combinations, several more distinct patterns emerge. Surprisingly, the first one is the prevalence of the Support Vector Machines classifier model. Six of the nine combinations in Table 26 used this model for training and predicting. This is particularly interesting as this was also the model that Malo et al. (2013), the source of the training data, utilised most successfully in their original research. The second pattern is that this time all of the combinations employed Sentence-

BERT and transformer models for numericising text. This again speaks to its advantage over the more traditional Bag of Words model in apprising future returns, which was the query proposed in research sub-question 3-1. Moreover, six out of the nine combinations were performed on mined justifications instead of full MD&A Overview documents. This is further evidence of the efficacy of justification mining in informing future returns better than the full document (research question 3). As stated previously, being able to summarise the sentiment of a full document in a few sentences could help eliminate noise in the data. In finance and data analysis in general, signal-to-noise ratios and eliminating noise are often important in making predictions (Xiong et al., 2006).

Another pattern of note is that all combinations were performed with balancing of class labels. While this does not directly help to answer any of the research questions, it is something important to keep in mind for future work in this area. Having examined the training data, as stated in Section 6, there were more neutral labels than the other two labels, so this is likely why unbalanced models did not perform as well. A further pattern is that all combinations employed the NLI pre-trained model in Sentence-BERT for both numericising text for input into ML classifiers as well as for encoding text into sentence embeddings during justification mining. This is somewhat surprising as the Wikipedia pre-trained model is known to be better for downstream clustering tasks (Reimers and Gurevych, 2019). However, it might be that the NLI dataset is better suited for the task of numericisation of text for input into machine learning models. Moreover, there is evidence from the literature (Conneau et al., 2017; Cer et al., 2018) that for training sentence embeddings the NLI and SNLI datasets work well. It will be interesting to see which pre-trained model for SBERT is favoured in the next subsection where mined justification sentiment scores and full document sentiment scores are compared. For the aggregation of sentiment methods, mean aggregation worked best, present in six of the nine combinations. This differed from the results in Section 6, where mode aggregation outperformed mean aggregation. However, there are several differences between this section and Section 6, namely that in Section 6 only the Bag of Words model for numericising text was utilised and supervised learning models were being evaluated, whereas in this section the majority of the work is done with unsupervised learning (hence the need for a downstream objective task for evaluation). Recall also that mean aggregation is the only one that added a weight factor to the sentiment scores, unlike mode and rounded mean aggregation. I surmised earlier that adding a weight factor (where the scores could take on float values between -1 and 1) could add an extra layer of depth to the predicted sentiment scores and allow them to better inform future returns. It seems likely this hypothesis could have some merit based on these results.

Also note that out of the six combinations that used justification mining, five of them used models that did not require specifying the k number of clusters beforehand (three used DBSCAN and two used MeanShift while only one used Agglomerative clustering). This could indicate that the number set for k (five) was not ideal, or that having a variable number of clusters and, therefore, a variable number of justifications returned was preferable. This would make sense, as one of the advantages justification mining might have is eliminating noise. Allowing the clustering algorithm to select the optimal number of clusters and consequently number of justifications returned (while also allowing this number to vary between combinations) might be optimal for the task of eliminating noise in the data. Moreover, if clustering is performed with a set number of clusters, k, clusters may not be reasonable because it could force the assignment of certain sentences to inappropriate clusters (Sarkar, 2009).

One final pattern that emerges is that seven of the nine combinations occurred using stock returns calculated one month after the 10-K filing was published. This could mean that the sentiment from the 10-K filings is more informative for short term returns but still has a period of time before investors act upon the information. This would make sense as 10-K filings are long and complex and even their annotation requires expert-level annotators with deep domain knowledge (Ren et al., 2013; Brew et al., 2010). So it would be natural for it to require a period of time before the information contained within them was utilised by investors to inform their investment decision-making. This is likely why there were no r-values of $|r|$>0.2 when using the one week returns, and even amongst all the statistically significant returns in Table 25, the one week returns had the smallest number of combinations (only two). It would also make sense that, after more than one month, perhaps the information contained within 10-K filings became less relevant as newer sources of news and information for investment decisions became available. Both of these factors could explain why the largest number of statistically significant Pearson correlation coefficients of $|r|$>0.2 occurred when calculated using one-month stock returns.

Something else to keep in mind is that many of these options that are predominantly showing up in this subsection are also options that were less utilised in the research calculations. That is to say, options such as Support Vector machines as the training model, variable number of cluster clustering models like DBSCAN and MeanShift for justification mining, and the NLI pre-trained model for Sentence-BERT, were used in fewer combinations than other options when looking at the full list of variable combinations (see Tables 17 to 24 in Section 7.6.4). This negates the notion that these options only presented more r-values of $|r|$>0.2 because they were more prevalent in the calculations already.

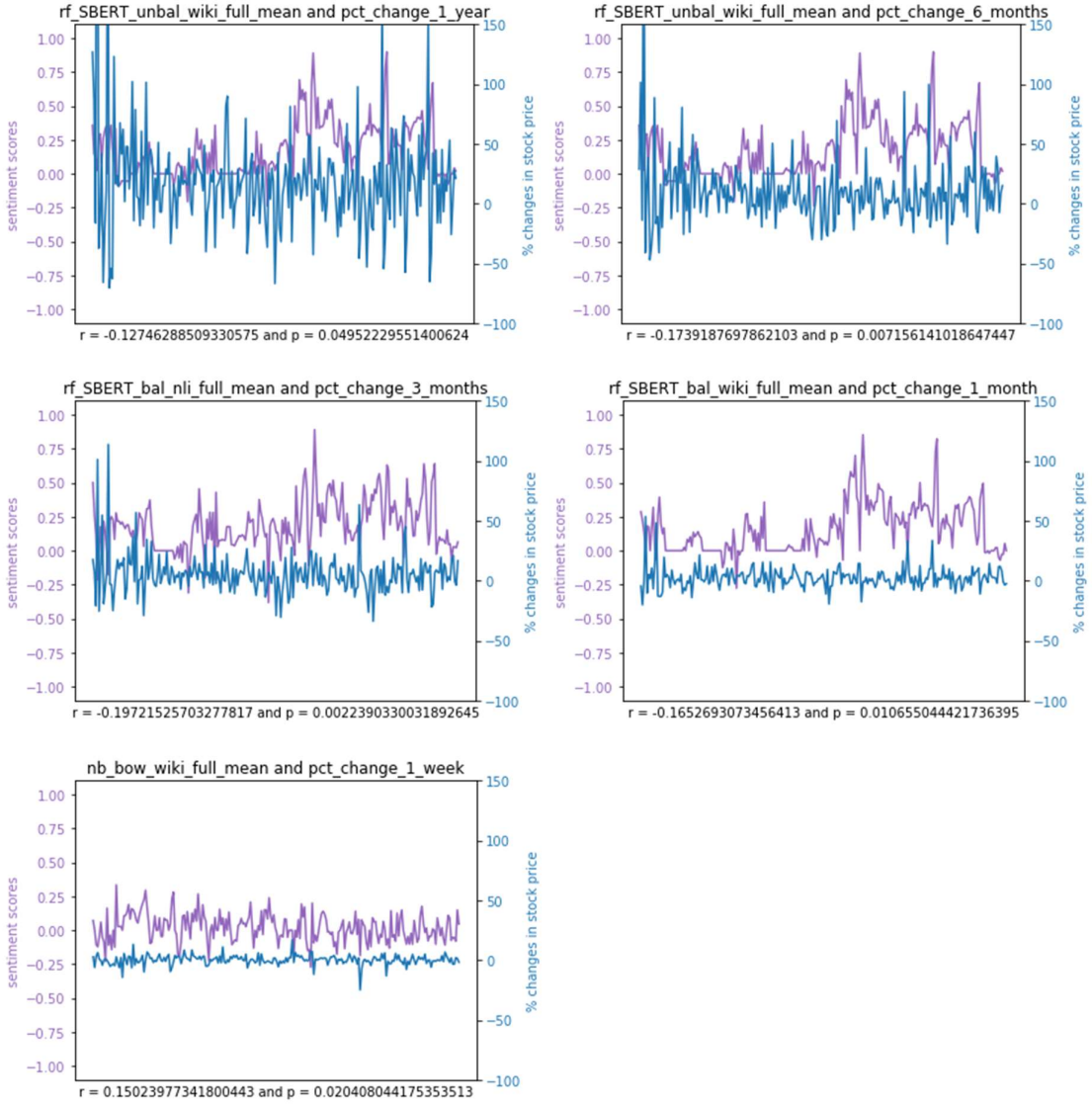
It should be mentioned, however, that although these combinations have low p-values, the r-values are quite small, with the largest being -0.2534. Although these correlations are somewhat weak, they are still statistically significant, and could be combined with traditional financial signals (such as momentum-based indicators like moving average, trend indicators like average directional movement or statistical indicators like entropy or variance) to enhance those signals. This idea is expanded upon later in Section 11. One of the reasons why the r-values were low but still statistically significant might be because large sample sizes allow for the detection of weak population correlations (Kenett et al., 2014; Rajan and Victor, 2014; Ranco et al., 2015). Also note that all the *r* values were negative in Table 26 above, implying a negative correlation between these aggregated sentiment scores and future returns. While this does not directly pertain to the research questions for this thesis, it is still an interesting pattern to mention.

To better visualise the results of this section, one other function was created to help plot the data. This function would take as parameters the sentiment scores and stock returns for any combination and make a double axis plot. Python's *matplotlib.pyplot* module was utilised for all plotting. The x-axis would simply be the index number of all companies in the data (total length being 238) and the two graphs plotted would be one for the sentiment scores of any given variables combination and the other for the future stock returns of any given time window. The first y-axis was set to the left and coloured purple. It would contain the sentiment scores and the axis would span from -1 to 1, in other words the bounds of possible sentiment scores. The *twinx()* function was employed to instantiate a second axis. This second y-axis was placed on the right and coloured blue. It would contain the stock return values for all companies in the data for any given time window. The limits of this axis were set based on two standard deviations away from the mean (in most cases this was around -100 to 150 and, thus, this

seemed like a good, standardised value to select). The *tight_layout()* function was also utilised to prevent clipping of the y-axis label on the right side. Finally, the title was set to be a combination of the variables employed within the chosen sentiment analysis for that plot, and the word '*and'* followed by the time windows utilised for the stock returns. Again, nested for loops were used to iterate through first the stock returns time windows and then the sentiment analysis tests. The nested loop would simply input each combination into the plot function written earlier and print out each plot. The function also had three optional parameters for setting the Pearson coefficient *r* and the p-value (both default to 0 as a placeholder) and a Boolean parameter to set a title for the x-axis. If this Boolean parameter is set to *true*, the x-axis would be labelled '*r = {} and p = {}*' where the brackets would be replaced by the corresponding Pearson coefficient and p-value calculated for that combination. This was done so that the information was readily available when needed for evaluating the plots, but optional so that it did not clutter the overall printing of plots if one simply wanted to see the graphs alone to assess general patterns.

As expected, the number of plots when going through the full list was simply too large and did not offer any insights into meaningful patterns or evaluation. As it would take up too much space, the full set of plots is not included in the Appendix. Instead, I employ the lists I created earlier to filter the plots. Recall that when the output for those combinations where $p \leq 0.05$ (statistically significant) were being printed, the sentiment analysis variables, stock returns time window, calculated *r* score and the p-values were all saved as a list within a larger list so that each combination could be saved in order. Similarly, when printing the output for those combinations where $r > 0.2$ or $r < -0.2$, the same information was saved as another list of lists. So now I could simply loop over any list of lists, iterating over each individual list inside. Starting with the statistically significant combinations ($p \leq 0.05$), each list could be fed directly into the plotting function (whose parameters match the order of the list: sentiment analysis variables, stock returns time windows, r, p-value), and set the x-axis labelling Boolean parameter to *True.* Plot 2 contains a few examples of some of the plots; one was included for each stock returns time window. The rest are included in Appendix 3 (Section 14.3). The key for the sentiment analysis variables remains the same as before (see Table 25 for the key). Due to a lack of space and redundancy, although it was originally planned to create different types of plots as well (scatter plots for instance), it was decided that these double axis plots were enough to convey the information necessary for evaluation. It would be an interesting area of future work to check how the values might be represented in different plots; however, for this research investigation, I stuck to double axis plots.

Plot 2: Plots of some example combinations between sentiment analysis predicted scores/labels and stock returns for combinations with statistically significant p-values (p≤0.05). The key for the sentiment variables notation is the same as Table 25. In the title of each plot, the sentiment analysis variables are listed first followed by the time window used to calculate the stock returns. Sentiment scores are on the left y-axis in purple and stock returns are on the right y-axis in blue. The Pearson coefficient (r) and p-values are listed on the x-axis.

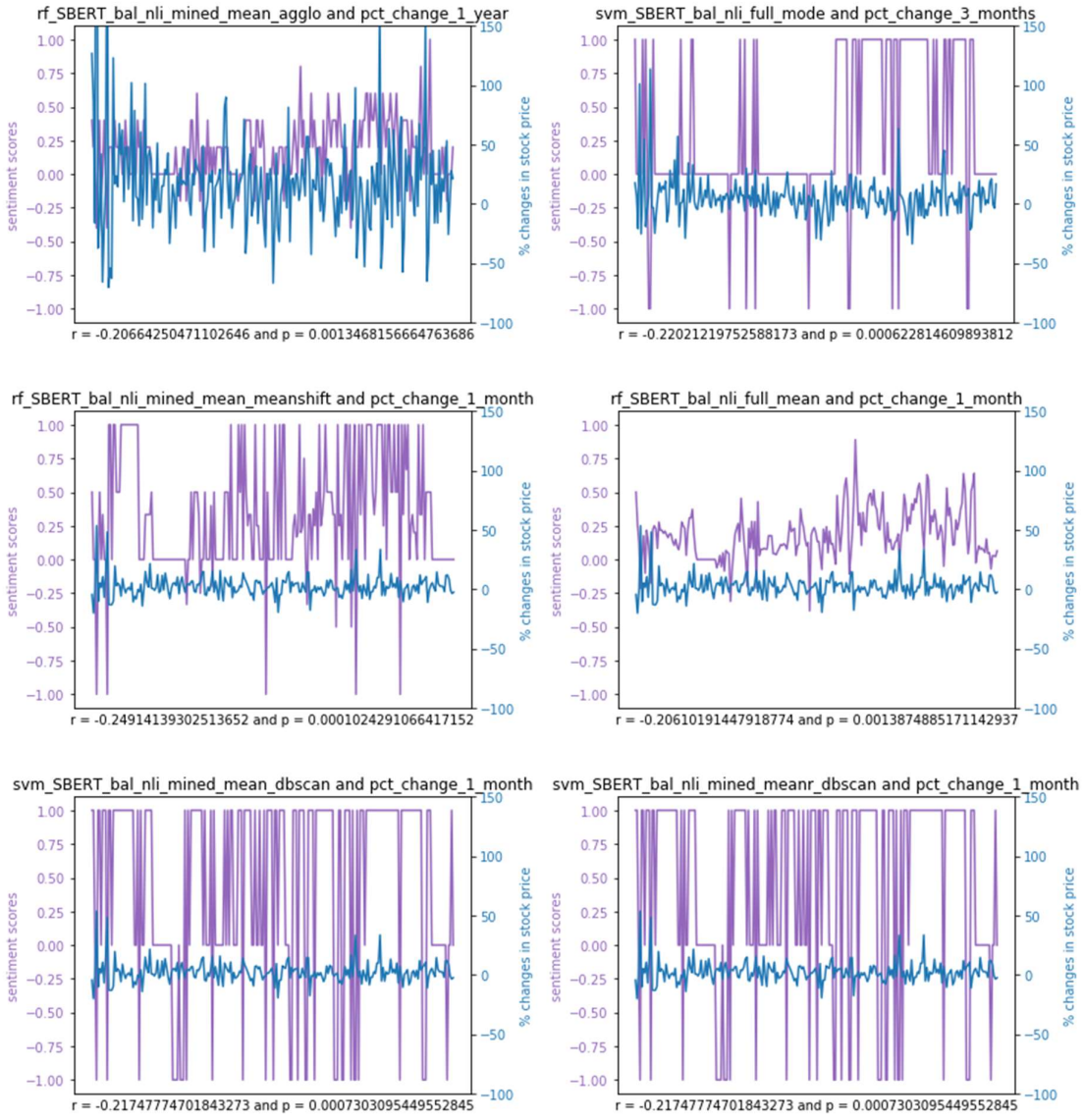


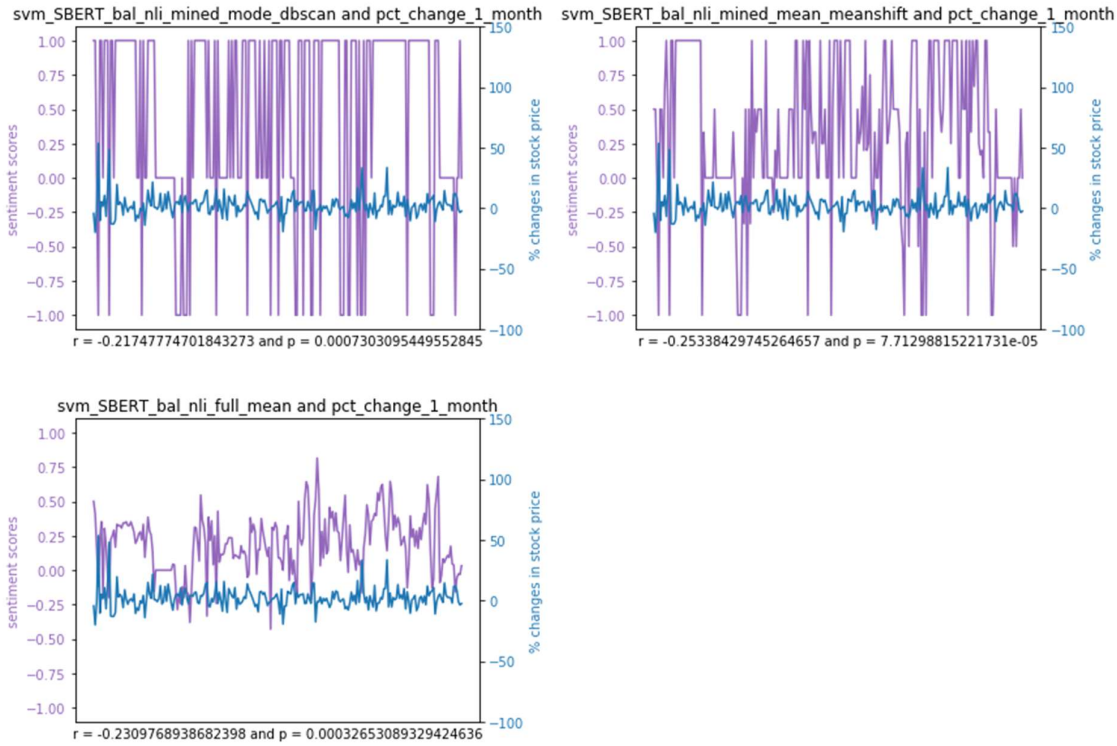
As expected, the above plots do not contribute much to evaluation of these models. One notable pattern, however, is the variance. The variance in stock returns, unsurprisingly, increases as the time window used increases. One week returns have a much lower variance than one-year returns, which is exactly what one would anticipate. This alone, however, does not really provide any meaningful insights. The change in the aggregated sentiment scores for the one-week plot is likely due to the different model that is used, rather than anything else. The random forests model is used in the other

four plots (with similar other variables chosen), which is why their sentiment scores plot (purple) is similar when compared to the one-week plot which used the Naïve Bayes classifier.

Instead, I can filter the data once again as before, and only plot those combinations with correlation coefficients (r) of >0.2 or <-0.2. Recall that all of these also had statistically significant p-values, $p \leq 0.05$. They were print using the same method, a *for* loop, but this time iterating over the Pearson correlations of $|r|>0.2$ list created earlier. As there are fewer combinations, all these plots are included here, see Plot 3 below, rather than in the Appendix. Again, refer to the key above (see Table 25) for the sentiment analysis variables notation.

Plot 3: Plots of all combinations between sentiment analysis predicted scores/labels and stock returns for combinations with Pearson coefficients, r, of >0.2 or <-0.2. See Plot 2 for details of plot layout.





Here, I can begin to note a few interesting observations. Although there might be instances where a negative correlation can be seen, it is still a weak negative correlation. As such, these plots do not offer much insight there. However, another intriguing aspect is the aggregated sentiment plots that use mode and rounded mean aggregation. As expected, these values are discrete and their plots reflect this (as opposed to the mean aggregation plots). This is because the mode and rounded mean aggregations took the weighting aspect out of the sentiment scores and only the mean aggregation had an aspect of weight to it. However, what is interesting is that it does not appear to impact the correlation too much. Both the unweighted and weighted scores had correlations around the same range (-0.20 to -0.25). Still, most of the informative patterns in this subsection are observed from the data itself rather than these plots. The plots in the next subsection - for similarity tests between mined justification sentiment scores and full document sentiment scores - might be more informative.

7.7.2 Similarity between Mined Justification Sentiment and Full Document Sentiment

With the correlation to future stock returns form of evaluation completed, it was time to turn to the next evaluation method discussed previously. This evaluation method focuses primarily on addressing research question 2 and its sub-questions. Research question 2 is arguably the most important research question of this thesis and so this section is of particular importance in assessing justification mining. I, along with input from professors (Roberts, 2020; Simpson, 2020), devised a method to evaluate the efficacy of justification mining in capturing the majority of the sentiment in a document.

To compare the sentiment scores obtained through mined justifications versus those from full documents, a variety of methods could be employed. For this study, cosine similarity and Pearson correlation were chosen, as both allow a comparison of the sentiment score vectors between mined

justifications and full documents (Roberts, 2020; Simpson, 2020). Cosine similarity and Pearson correlation have similar formulae (see below; Al-Hagery, 2016; scipy.stats.pearsonr, 2020).

Cosine similarity:

$$similarity = \cos\theta = \frac{X \cdot Y}{|X||Y|}$$

Pearson correlation:

$$Pearson\ coefficient = r = \frac{\sum(x - m_x)(y - m_y)}{\sqrt{\sum(x - m_x)^2(y - m_y)^2}}$$

Looking at the formula for each, it can be noticed that they are identical if the mean is 0 for both variables (x and y). Neither option is impacted by multiplying each element in the formula by a nonzero constant. However, only Pearson correlation is also not impacted by adding a constant as well. This is because in the calculation of the Pearson correlation coefficient, the mean is subtracted in each element (Dongen and Enright, 2012; Chunjie et al., 2017). Because of this small but important difference, both were used in the evaluation of the relationship between mined justification sentiment scores and full document sentiment scores (obtained through the aggregation of sentiment scores predicted by the previously trained classifiers).

If the aggregated sentiment scores between mined justifications and full documents prove to be similar, then justification mining would be an effective means of capturing most of the sentiment in a document with only a few sentences and could be used as a method for sentiment summarisation. As discussed previously, this could imply many things. First, it could provide less noisy sentiment signals (which may have been observed in the previous subsection). Next, it could provide utility in areas such as AI explainability, where such sentiment summarisations might prove useful in explaining to and convincing investors of various machine learning strategies. In finance, many investors who are not tech-savvy, and even those that are, might still view machine learning strategies as black boxes and not have confidence in them. This is where tools like justification mining might prove valuable (see Sections 9 and 9.1). These mined justifications might also prove beneficial in helping make more ethically informed decisions in finance (research sub-question 2-3). This topic is expanded upon in Sections 11 and 11.1 (discussing future work) as they are beyond the scope of this research investigation.

To start this second evaluation process, as before, Pearson correlation was performed using the *pearsonr* function in *scipy.stats* whereas cosine similarity was calculated using the *cosine_similarity* function in *sklearn.metrics.pairwise*. As with the stock returns evaluation, a nested for loop was employed to calculate both scores in every possible combination. Since I saved the column names for both the instances where sentiment analysis was performed on mined justifications and the instances where it was performed on full MD&A Overview documents in separate lists, each *for* loop had to simply iterate through one list after the other. As a result, one value column/vector from the mined justifications side and one from the full documents side (taken from the main Pandas dataframe) was inputted into the cosine similarity function. As with before, for cosine similarity, both vectors had to be reshaped since they were single samples using *reshape(1,-1)* and the numerical value had to be

extracted from the output using indexing (*[0][0]*). Afterwards, The Pearson coefficient was calculated using a similar method within the same loop. Finally, the loop printed the values by printing the variables for the mined justification sentiment analysis and the full document sentiment analysis along with the cosine similarity score, Pearson coefficient and p-value for the Pearson correlation. As you can imagine, the full list of combinations between these two was large. For the sake of space, it is not included within the Appendices. The output for the full list of combinations contained too much noise and too many variations to render it meaningful for evaluation, and so it proved best to filter it as before. One thing to note, however, is that in many of the cases where the cosine similarity score and Pearson coefficient were close to or equal to 0, different classifier training models were being compared or a model that used Bag of Words for numericisation was being compared to one that used transformer models and Sentence-BERT embeddings for numericisation. This might indicate that these differences produced vastly different sentiment predictions between mined justifications and full documents.

An alternative approach is to print only those combinations where the cosine similarity score or Pearson coefficient is large. To do so, *if* conditional statements were placed into the nested loops so that only cases where the cosine similarity score or Pearson coefficient was >0.7 or <-0.7 were printed. Cosine similarity using BoW (which assigns each word a nonnegative numeric index value) cannot be negative since the numeric indices assigned to words themselves cannot be negative. In other words, cosine similarity cannot be negative when the inputted vectors are nonnegative. However, sentence embeddings can have negative values and thus, potentially, negative cosine similarity values. Therefore, it was important to factor this in when printing the output. Absolute values could be checked instead to determine the strength of the similarity, but negative values indicate vectors that are strongly opposite (Dongen and Enright, 2012; Chunjie et al., 2017). Since I am trying to determine similarity and not an opposing relationship, only those values greater than 0 matter in this case and taking absolute values did not make sense. However, I still printed both values >0.7 and <-0.7 to see what results would be produced.

The first table that follows, Table 27, contains the cases where the cosine similarity score was >0.7 or <-0.7. The scores were calculated and extracted as above, with the only difference being the added *if* statement. Also, as with the stock returns evaluation, the output was saved into a list and a count was added to the *for* loop to make sure the length of the list and the count matched (this sanity check was included to make sure no cases with high scores were being omitted or not printed). Recall that all the notations in the tables below used for the variables of the sentiment analyses are the same as those used in the stock returns evaluation, see Table 25 in Section 7.7.1 for the key. It should also be noted that most of the combinations listed below generally had Pearson coefficients of 0.55 or higher (typically above 0.60), although a few had much weaker scores (the Pearson coefficients are not listed in Table 27 in the interest of space, but the largest coefficients are presented later in Table 28).

Table 27: Table containing all the combinations with cosine similarity scores >0.7 or <-0.7. See Table 25 for sentiment analysis variable notations key.

Variables used in Sentiment Analysis for Full Documents	Variables used in Sentiment Analysis for Mined Justifications	Cosine Similarity Score
rf_SBERT_bal_wiki_full_mean	rf_SBERT_bal_wiki_mined_mean_agglo	cosine=0.804426115614954
rf_SBERT_bal_wiki_full_mean	rf_SBERT_bal_wiki_mined_mean_dbscan	cosine=0.7168612127589199
rf_SBERT_bal_wiki_full_mean	rf_SBERT_bal_wiki_mined_mean_kmeans	cosine=0.8099058420539312
rf_SBERT_bal_wiki_full_mean	rf_SBERT_unbal_wiki_mined_mean_agglo	cosine=0.7904287808318623
rf_SBERT_bal_wiki_full_mean	rf_SBERT_unbal_wiki_mined_mean_dbscan	cosine=0.759560378780189
rf_SBERT_bal_wiki_full_mean	rf_SBERT_unbal_wiki_mined_meanr_dbscan	cosine=0.7350385811770725
rf_SBERT_bal_wiki_full_mean	rf_SBERT_unbal_wiki_mined_mean_kmeans	cosine=0.7750311549640602
rf_SBERT_unbal_wiki_full_mean	rf_SBERT_bal_wiki_mined_mean_agglo	cosine=0.7850786110028484
rf_SBERT_unbal_wiki_full_mean	rf_SBERT_bal_wiki_mined_mean_dbscan	cosine=0.7105032857795262
rf_SBERT_unbal_wiki_full_mean	rf_SBERT_bal_wiki_mined_mean_kmeans	cosine=0.805024080070877
rf_SBERT_unbal_wiki_full_mean	rf_SBERT_unbal_wiki_mined_mean_agglo	cosine=0.8242802559372079
rf_SBERT_unbal_wiki_full_mean	rf_SBERT_unbal_wiki_mined_mean_dbscan	cosine=0.7746303487124903
rf_SBERT_unbal_wiki_full_mean	rf_SBERT_unbal_wiki_mined_meanr_dbscan	cosine=0.7498327277962114
rf_SBERT_unbal_wiki_full_mean	rf_SBERT_unbal_wiki_mined_mean_kmeans	cosine=0.8178689269518066
rf_SBERT_bal_nli_full_mean	rf_SBERT_bal_wiki_mined_mean_agglo	cosine=0.7567452438209511
rf_SBERT_bal_nli_full_mean	rf_SBERT_bal_wiki_mined_mean_kmeans	cosine=0.7368905891836846
rf_SBERT_bal_nli_full_mean	rf_SBERT_unbal_wiki_mined_mean_agglo	cosine=0.7342853616339593
rf_SBERT_bal_nli_full_mean	rf_SBERT_unbal_wiki_mined_mean_kmeans	cosine=0.7247368679343799
rf_SBERT_bal_nli_full_mean	rf_SBERT_bal_nli_mined_mean_agglo	cosine=0.8285888253107828
rf_SBERT_bal_nli_full_mean	rf_SBERT_bal_nli_mined_mean_dbscan	cosine=0.7653045335397194
rf_SBERT_bal_nli_full_mean	rf_SBERT_bal_nli_mined_meanr_dbscan	cosine=0.7653045335397194
rf_SBERT_bal_nli_full_mean	rf_SBERT_bal_nli_mined_mode_dbscan	cosine=0.7653045335397194
rf_SBERT_bal_nli_full_mean	rf_SBERT_bal_nli_mined_mean_kmeans	cosine=0.7995499721776714
rf_SBERT_bal_nli_full_mean	svm_SBERT_bal_wiki_mined_mean_agglo	cosine=0.7145157522199707
rf_SBERT_bal_nli_full_mean	svm_SBERT_bal_wiki_mined_meanr_agglo	cosine=0.7102136150394366
rf_SBERT_bal_nli_full_mean	svm_SBERT_bal_wiki_mined_mode_agglo	cosine=0.7102136150394366
rf_SBERT_bal_nli_full_mean	svm_SBERT_bal_wiki_mined_mean_dbscan	cosine=0.7102136150394366
rf_SBERT_bal_nli_full_mean	svm_SBERT_bal_wiki_mined_meanr_dbscan	cosine=0.7102136150394366
rf_SBERT_bal_nli_full_mean	svm_SBERT_bal_wiki_mined_mode_dbscan	cosine=0.7102136150394366
rf_SBERT_bal_nli_full_mean	svm_SBERT_bal_wiki_mined_mean_kmeans	cosine=0.7161691548976721
rf_SBERT_bal_nli_full_mean	svm_SBERT_bal_wiki_mined_mean_meanshift	cosine=0.722459814161369
rf_SBERT_bal_nli_full_mean	svm_SBERT_bal_nli_mined_mean_agglo	cosine=0.733681627310457
svm_SBERT_bal_wiki_full_mean	svm_SBERT_bal_wiki_mined_mean_agglo	cosine=0.9987898522197902
svm_SBERT_bal_wiki_full_mean	svm_SBERT_bal_wiki_mined_meanr_agglo	cosine=0.9996896805662667
svm_SBERT_bal_wiki_full_mean	svm_SBERT_bal_wiki_mined_mode_agglo	cosine=0.9996896805662667
svm_SBERT_bal_wiki_full_mean	svm_SBERT_bal_wiki_mined_mean_dbscan	cosine=0.9996896805662667
svm_SBERT_bal_wiki_full_mean	svm_SBERT_bal_wiki_mined_meanr_dbscan	cosine=0.9996896805662667
svm_SBERT_bal_wiki_full_mean	svm_SBERT_bal_wiki_mined_mode_dbscan	cosine=0.9996896805662667
svm_SBERT_bal_wiki_full_mean	svm_SBERT_bal_wiki_mined_mean_kmeans	cosine=0.9989076483114812
svm_SBERT_bal_wiki_full_mean	svm_SBERT_bal_wiki_mined_mean_meanshift	cosine=0.9922818784512301
svm_SBERT_bal_wiki_full_meanr	svm_SBERT_bal_wiki_mined_mean_agglo	cosine=0.9979720809427626

svm_SBERT_bal_wiki_full_meanr	svm_SBERT_bal_wiki_mined_meanr_agglo	cosine=0.9999999999999997
svm_SBERT_bal_wiki_full_meanr	svm_SBERT_bal_wiki_mined_mode_agglo	cosine=0.9999999999999997
svm_SBERT_bal_wiki_full_meanr	svm_SBERT_bal_wiki_mined_mean_dbscan	cosine=0.9999999999999997
svm_SBERT_bal_wiki_full_meanr	svm_SBERT_bal_wiki_mined_meanr_dbscan	cosine=0.9999999999999997
svm_SBERT_bal_wiki_full_meanr	svm_SBERT_bal_wiki_mined_mode_dbscan	cosine=0.9999999999999997
svm_SBERT_bal_wiki_full_meanr	svm_SBERT_bal_wiki_mined_mean_kmeans	cosine=0.9983368343655237
svm_SBERT_bal_wiki_full_meanr	svm_SBERT_bal_wiki_mined_mean_meanshift	cosine=0.9899849184186656
svm_SBERT_bal_wiki_full_mode	svm_SBERT_bal_wiki_mined_mean_agglo	cosine=0.9979720809427626
svm_SBERT_bal_wiki_full_mode	svm_SBERT_bal_wiki_mined_meanr_agglo	cosine=0.9999999999999997
svm_SBERT_bal_wiki_full_mode	svm_SBERT_bal_wiki_mined_mode_agglo	cosine=0.9999999999999997
svm_SBERT_bal_wiki_full_mode	svm_SBERT_bal_wiki_mined_mean_dbscan	cosine=0.9999999999999997
svm_SBERT_bal_wiki_full_mode	svm_SBERT_bal_wiki_mined_meanr_dbscan	cosine=0.9999999999999997
svm_SBERT_bal_wiki_full_mode	svm_SBERT_bal_wiki_mined_mode_dbscan	cosine=0.9999999999999997
svm_SBERT_bal_wiki_full_mode	svm_SBERT_bal_wiki_mined_mean_kmeans	cosine=0.9983368343655237
svm_SBERT_bal_wiki_full_mode	svm_SBERT_bal_wiki_mined_mean_meanshift	cosine=0.9899849184186656
svm_SBERT_bal_nli_full_mean	rf_SBERT_bal_nli_mined_mean_agglo	cosine=0.7897961426591971
svm_SBERT_bal_nli_full_mean	rf_SBERT_bal_nli_mined_mean_dbscan	cosine=0.7358192593117305
svm_SBERT_bal_nli_full_mean	rf_SBERT_bal_nli_mined_meanr_dbscan	cosine=0.7358192593117305
svm_SBERT_bal_nli_full_mean	rf_SBERT_bal_nli_mined_mode_dbscan	cosine=0.7358192593117305
svm_SBERT_bal_nli_full_mean	rf_SBERT_bal_nli_mined_mean_kmeans	cosine=0.7692485832268792
svm_SBERT_bal_nli_full_mean	svm_SBERT_bal_nli_mined_mean_agglo	cosine=0.7906812705438784
svm_SBERT_bal_nli_full_mean	svm_SBERT_bal_nli_mined_mean_dbscan	cosine=0.7406596029523967
svm_SBERT_bal_nli_full_mean	svm_SBERT_bal_nli_mined_meanr_dbscan	cosine=0.7406596029523967
svm_SBERT_bal_nli_full_mean	svm_SBERT_bal_nli_mined_mode_dbscan	cosine=0.7406596029523967
svm_SBERT_bal_nli_full_mean	svm_SBERT_bal_nli_mined_mean_kmeans	cosine=0.742369563038398

The first thing to note is that there were no similarity scores that were <-0.7, implying that none of the combinations demonstrated a strong opposing relationship. This boded well for the argument that justification mining was capable of capturing and summarising the sentiment in a document. Other than that, the biggest and arguably most important pattern that emerges is that all combinations utilised Sentence-BERT to encode sentence embeddings for numericisation of text. This relates to answering research sub-question 2-2 and implies that transformer models and Sentence-BERT have a clear advantage over an approach like Bag of Words here. As stated before, this could be because information is not lost when encoding sentence embeddings like it is when utilising Bag of Words. Furthermore, due to its increased speed over models like BERT and RoBERTa (Reimers and Gurevych, 2019), Sentence-BERT could prove more readily accessible for many studies or applications. Finding applications such as this, where it outperforms a traditional approach, would support wider use of transformer models in this area, as well as potentially lead to an expansion of methods like Sentence-BERT. For instance, more pre-trained models could be created for Sentence-BERT, perhaps using financial datasets, which could prove very valuable in the field of finance.

Another thing to note is that many combinations in Table 27 used the same classifier model for making predictions on both mined justifications and full documents. For instance, if SVMs were used as the model on the mined justifications side, they were often also the model on the full documents side (at least in those cases with high similarity scores). This makes sense as two same models are likely to have more similar predictions over differing models. Both the mined justifications and full

document sentiment analyses in Table 27 also appear to favour the use of balanced class labels. As discussed earlier, the training data from Malo et al. (2013) did have more neutral labels, so balancing them might have led to better similarity between mined justifications and full document sentiment predictions.

Mean aggregation of sentiment scores also seems to be favoured. Recall that mean aggregation adds a weight aspect to the scores, allowing them to be floats between -1 and 1, whereas mode and rounded mean remove this weight attribute. As with the stock returns evaluation, it is likely having this weight component adds another layer of depth to the sentiment score, which once again has proven valuable in capturing more information within that score. At first, it was hypothesised that mode or rounded mean values might correlate better or be more similar, since they are discrete and can only take on values of -1, 0 or 1 (whereas mean values can be any float between -1 and 1). Thus, it is of particular interest that there are many mean aggregated scores in Table 27, demonstrating that the similarity discovered was not simply due to the small number of options for the scores (three options in the case of mode and rounded mean aggregation).

Areas in which there did not appear to be any pattern or favoured variable included the clustering model used for justification mining and the pre-trained model used for Sentence-BERT (Wikipedia or NLI).

There were, though, numerous cosine similarity scores very close to 1, implying nearly matching sentiment score predictions between mined justifications and full documents. In these cases, justification mining was able to capture practically all of the sentiment of the full MD&A Overview subsection with a few sentences. All of these cases, interestingly, used Support Vector Machines as the classifier model. Moreover, they also all used the Wikipedia pre-trained model for Sentence-BERT. Recall that in the stock returns evaluation, all of the combinations with Pearson coefficients of $|r|>0.2$ used the NLI pre-trained model. Taking correlation to stock returns out of the equation and just looking at justification mining as a tool for sentiment summarisation, it is the Wikipedia pre-trained model that is favoured in the highest similarity scores. This makes sense, since justification mining primarily involves a downstream clustering task and the Wikipedia pre-trained model is optimised for these kinds of tasks (Reimers and Gurevych, 2019).

Lastly, all the instances where the Pearson coefficient was >0.7 or <-0.7 are presented. The same method of nested loops and *if* statements was used for filtering here as well. Table 28 lists all the combinations where these conditions were met. Although the p-values are not included in the interest of space, it should be noted that all of them were less than 0.05, indicating statistical significance. Moreover, all of the combinations below in Table 28 also had high cosine similarity scores (although they are not listed again in the interest of space) but not all of the combinations previously in Table 27 had Pearson coefficients of $|r|>0.7$.

Table 28: Table containing all the combinations with Pearson coefficients >0.7 or <-0.7. See Table 25 for sentiment analysis variable notations key.

Variables used in Sentiment Analysis for Full Documents	Variables used in Sentiment Analysis for Mined Justifications	Pearson Correlation Coefficient
rf_SBERT_bal_wiki_full_mean	rf_SBERT_bal_wiki_mined_mean_kmeans	pearson=0.7005127204830015
rf_SBERT_unbal_wiki_full_mean	rf_SBERT_unbal_wiki_mined_mean_agglo	pearson=0.7306588067971225
rf_SBERT_unbal_wiki_full_mean	rf_SBERT_unbal_wiki_mined_mean_kmeans	pearson=0.7371809974388792
rf_SBERT_bal_nli_full_mean	rf_SBERT_bal_nli_mined_mean_agglo	pearson=0.7168222372437307
svm_SBERT_bal_wiki_full_mean	svm_SBERT_bal_wiki_mined_mean_agglo	pearson=0.7110430235908418
svm_SBERT_bal_wiki_full_mean	svm_SBERT_bal_wiki_mined_mean_meanshift	pearson=0.7405112640845319

As with cosine similarity, there were no Pearson coefficients that were less than -0.7, which again was good news. Before discussing other patterns, something else very interesting can be observed in the table above. In all six combinations, every single variable used in sentiment analysis is the same between full documents and mined justifications (other than the clustering algorithm as this only exists for mined justifications). This already is a large difference between the Pearson correlation and cosine similarity tests and could explain why there are fewer Pearson coefficients of $|r|$>0.7. It is possible that obtaining a Pearson coefficient of $|r|$>0.7 is more difficult than obtaining a cosine similarity score of around the same value.

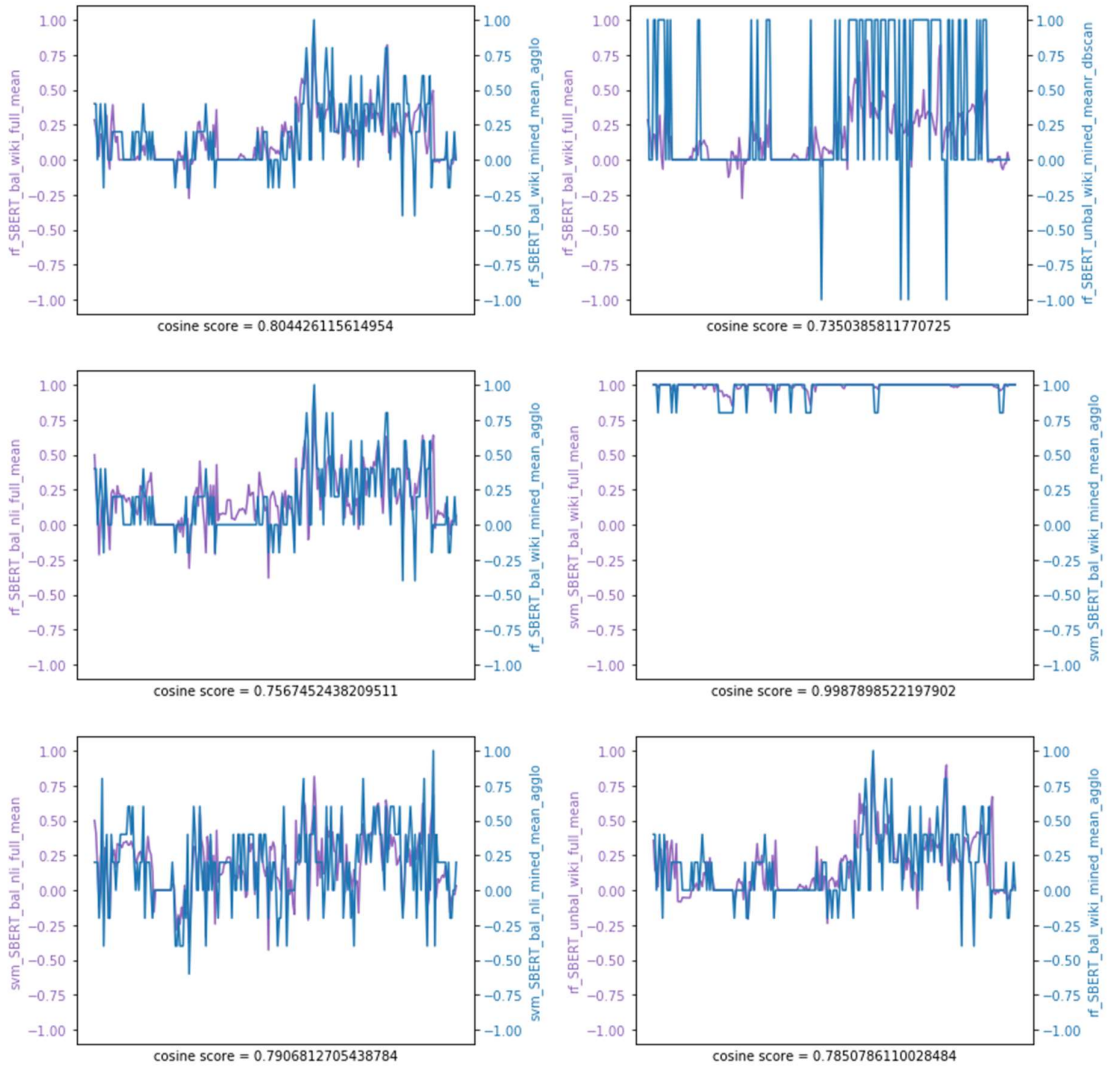
Sentence-BERT was the only option for numericisation returned, once again pointing to the advantage of this method over Bag of Words, likely for similar reasons of no information loss. Again, the Wikipedia pre-trained model for Sentence-BERT was favoured, with all but one combination using it. Four of the six combinations also used the random forests classifier model for training and prediction. As before, mean aggregation was favoured, with all combinations utilising it. This further supports the benefits of weighted sentiment over the discrete options. Finally, one pattern that did change from the previous list was that all but one combination used K-Means and Agglomerative clustering models for justification mining. Both of these models require specifying the number of clusters in advance (in this case five was chosen), which did not seem to work in their favour earlier when DBSCAN and MeanShift were more prevalent in the combinations with correlations to stock returns of $|r|$>0.2 (see Table 26). However, in this case, having the number of clusters fixed appears to have helped in increasing the correlation coefficient. Figuring out why would require further research investigation and might be an interesting direction of future work.

As for the stock returns evaluation, a plotting function was also created for this subsection using the *matplotlib.pyplot* module. This function was structured similarly to the plotting function in the previous subsection with a few key differences. As parameters it first took in the column names for one mined justification sentiment analysis column and one full document sentiment analysis column from the main dataframe. There were also three optional parameters. Two of these were Boolean parameters that could be set to *True* to set the x-axis label to the actual score/coefficient obtained (one Boolean for cosine similarity scores and one for Pearson correlation coefficients). The third optional parameter simply held an empty spot for that score (default 0), to be populated when either of the other optional parameters were set to *True*. The x-axis again was set to the index, ranging from 1 to the total number of filings. The left axis this time, again coloured purple, was set to the aggregated sentiment scores from sentiment analyses on full documents. The corresponding line/graph was also coloured purple and the limits of the left y-axis set from -1 to 1, the bounds of possible aggregated scores. Again, *twinx()* was used to create the double axis and the right axis was set to the aggregated

sentiment scores from sentiment analyses on mined justifications. The graph colour was set to blue and the limits of this axis were again -1 to 1. The *tight_layout()* function had to be used once more in order to avoid clipping of the right axis. As with the stock returns correlation evaluation, only double axis plots are provided in the interest of space, and the key for the sentiment analysis variables corresponds to the previous key from Table 25.

To begin, several examples of plots where there was a large cosine similarity score (greater than 0.7) are provided on the following page (see Plot 4) and the rest can be found in Appendix 4 (Section 14.4). Recall that none of the cosine similarity scores or Pearson coefficients were negative, implying only directly proportional relationships. Note, whenever plots are referred to by number, the numbering is assigned by going from left to right then downwards.

Plot 4: Example plots of some combinations of sentiment analysis predicted scores/labels from mined justifications versus those from full documents with large cosine similarities (>0.7). Sentiment scores from full document sentiment analyses are on the left y-axis in purple and sentiment scores from mined justifications sentiment analyses are on the right y-axis in blue. The y-axes labels are the sentiment analysis variables notation, the key for which was given previously in Table 25. The cosine score is listed on the x-axis.

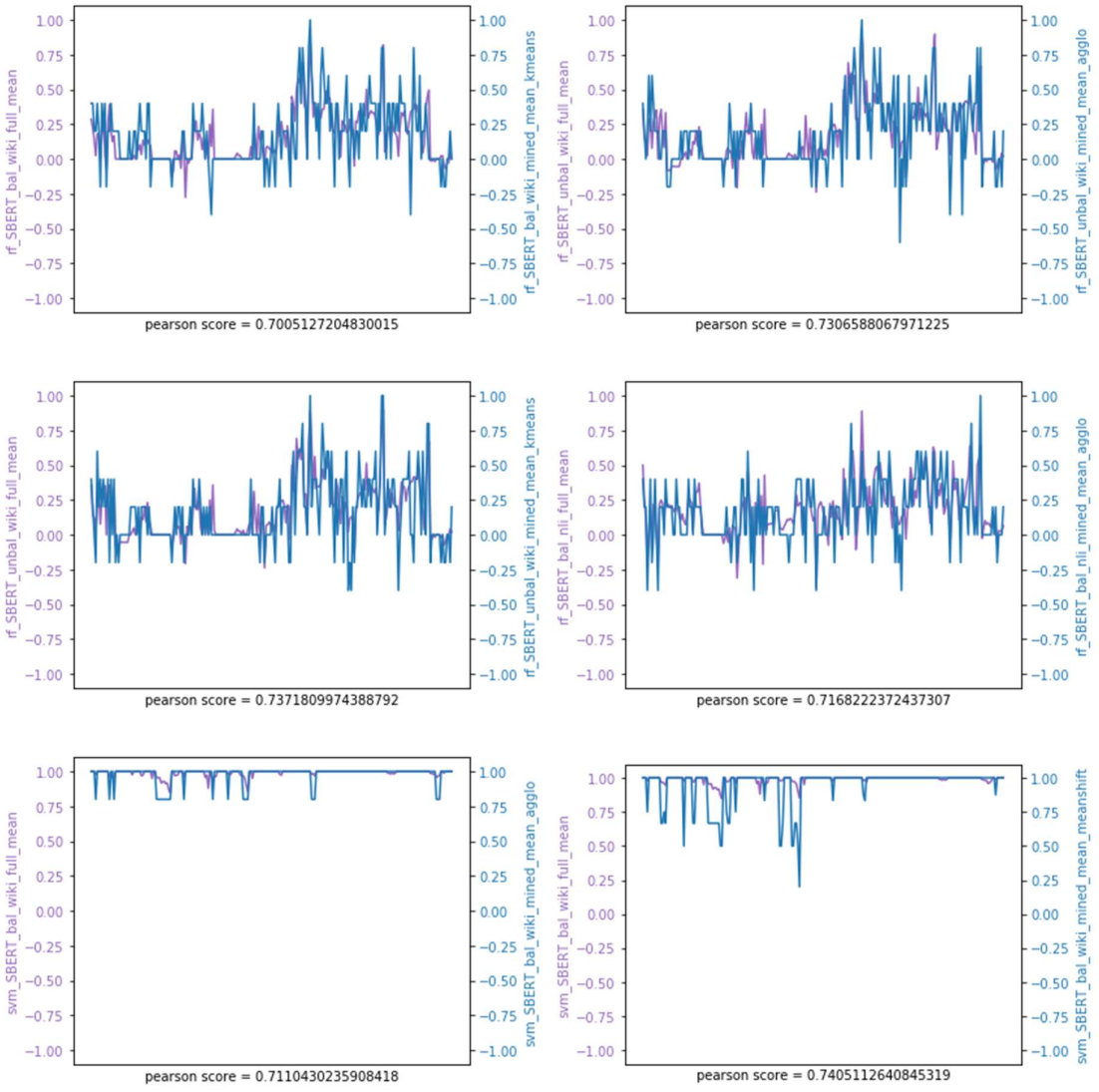


Unlike in the stock returns correlation analysis plots, here there is some definite similarity that can be observed in the plots themselves. The two lines in each plot appear to be similar in predicted sentiment scores in a variety of situations. This is most obvious when using mean aggregation (weighted) for both sides but, as can be seen in the second plot, it holds true even when one side uses one of the unweighted aggregation methods (mode and rounded mean). In this case, even though the mined justifications sentiment analysis employed rounded mean aggregation and had discrete aggregated sentiment scores, the similarity to the mean aggregated full document sentiment scores is easily observed. Moreover, in the fourth plot, even when the predicted scores have much less variance, mined justifications still presented high degrees of similarity to full documents. In fact, the

plots with low variance typically had the highest cosine similarity scores. As stated earlier, these results support the conclusion that justification mining is an effective tool for capturing the majority of the sentiment in a full document. This, in turn, helps begin to answer research question 2.

Finally, the plots for combinations with Pearson coefficients greater than 0.7 are presented on the following page (see Plot 5). This plot was likewise created using a *for* loop, but this time iterating over the list of lists with the large Pearson correlation coefficient combinations saved in it as well as setting the function parameter to include the Pearson score on the x-axis to *True*. Recall that in all of these combinations, the p-value was less than or equal to 0.05 and that there were no combinations with negative correlations. All of the combinations are presented below instead of in an Appendix. The '*pearson score*' given in the x-axis refers to the Pearson coefficient for that combination.

Plot 5: Plots of all combinations of sentiment analysis predicted scores/labels from mined justifications versus those from full documents with Pearson coefficients of >0.7. Note all combinations satisfied p≤0.05. Refer to previous Plot 4 for plot layout description, the only change being the x-axis now lists the Pearson coefficient for each combination.



As expected, the Pearson correlation test is stricter than the cosine similarities test. The correlation between mined justifications sentiment and full document sentiment is more apparent in these plots than some of the plots presented in Plot 4. Cases of high variance can be observed in the first four plots and lower variance in the last two plots. Unlike with cosine similarity, the plots with low variance did not always have larger Pearson correlation coefficients than those with higher variance. These plots further support the efficacy of justification mining in capturing most of the sentiment from a full 10-K MD&A Overview subsection.

Once all calculations and plots had been completed, the relevant list of lists that had been used in this research investigation were saved to pickle files for future use and organisation purposes. Some final

things to note, as previously stated, CUDA and GPU processing were utilised whenever they were available. The number of total sentiment analysis variable combinations was 88, with 64 variable combinations for sentiment analysis on mined justifications and 24 combinations for sentiment analysis on full documents. This was done to limit the total number of combinations in order to keep both the time required for calculations low and to remain within the scope of a DPhil thesis. Since evaluating justification mining was the goal of this research investigation and the research questions, there were many more combinations created for mined justifications.

7.7.3 Evaluations Summary

Although many evaluations and metrics have been covered in the previous two subsections, there are four general conclusions that are important for answering the research questions in this thesis. The first two pertain to correlation to future stock returns while the next two pertain to similarity tests between mined justifications and full documents.

First, for correlation to future stock returns, mined justifications were favoured in many of the correlations that were both statistically significant and had Pearson coefficients of $|r|$>0.2. This directly relates to answering research question 3. Second, in all the statistically significant combinations with Pearson correlations of $|r|$>0.2, Sentence-BERT and transformer models were utilised for numericising text for input into the classifier models, supporting their advantage over Bag of Words in providing statistically significant correlation. This helps in answering research sub-question 3-1.

Third, high degrees of similarity and strong correlation were found between the aggregated sentiment scores from mined justifications and those from full MD&A Overview subsections, supporting the efficacy of justification mining in capturing most of the sentiment from the full document with only a few sentences. This connects back to research question 2, as well as sub-questions 2-1, 2-3 and 2-4. More on answering these sub-questions will be covered in subsequent sections. Finally, fourth, Sentence-BERT and transformer models were also decidedly favoured in the cases where there were high degrees of similarity and strong correlation (in fact, all trials with either large similarity or strong correlations used Sentence-BERT for numericisation), again supporting their value over traditional approaches like Bag of Words. This relates back to research sub-question 2-2.

One last thing to note, although not as important as the previous four conclusions, is that Naïve Bayes only shows up in one combination out of all the combinations reported for both the correlation to future returns and the aggregated sentiment similarity analysis between mined justifications and full documents. Recall that three classifier models were used for the various combinations (Naïve Bayes, random forests and Support Vector Machines). This result might be because linear classifiers such as Naïve Bayes are only capable of learning simple features. Support Vector Machines and random forests, however, are able to map more complex, non-linear relationships in the data (Simpson, 2020). While this does not directly relate to answering any of the research questions for this thesis, it is still an interesting observation that could be useful for future work.

Some of these four conclusions may be explained by a variety of factors. Justification mining might be efficient in eliminating noise in the sentiment, which could clarify why it was favoured in correlation to future stock returns. Sentence-BERT, unlike Bag of Words, does not eliminate any information from the text when encoding sentence embeddings. This might support why it was favoured as a numericisation technique for text during feature engineering in both evaluation methods. Recall, Cer et al. (2018) also found that transfer learning using sentence embeddings typically performs better than word-level transfer. Their study had good results in transfer learning with sentence embeddings even with small amounts of annotated training data (Cer et al., 2018).

7.8 Closing Statements – Justification Mining

In Section 7, I have answered research questions 2 and 3, presenting and evaluating the methodology for justification mining, as well as addressed many of their sub-questions (some of which will be further expanded upon in Sections 8 to 11). The data parsing and cleaning process was outlined. The steps involved in justification mining were presented along with the early trials of this method. Several classifier models were trained and employed to make predictions on mined justifications and full MD&A Overview documents. These aggregated sentiment scores were correlated to future stock returns and evaluated. Sentiment scores from mined justifications were also compared to those from full documents and evaluated with similarity indices. Finally, performance metrics were reviewed and summarised.

The remaining sections (Sections 8 to 11) outline the benefits, contributions as well as the challenges of this study, and present possible future directions for this type of research. These sections, in part, relate to answering research sub-questions 1-1, 2-3 and 2-4.

8 Benefits of this Research and Implications of Potential Success

Many benefits and implications of potential success have already been touched on in the previous sections. In this section, these points are brought together and summarised.

The success of justification mining itself, both during the correlation to future returns as well as in capturing most of the sentiment from the full documents, has many important implications. With regards to predicting future returns, although the correlation was weak using just aggregated sentiment from mined justifications, this data could be used to enhance other financial signals, a popular process in the field of finance (Banga and Brorsen, 2019). In this way, it might provide valuable information towards finding signals that work. Furthermore, evidence supports justification mining as an effective tool for sentiment summarisation and capturing the majority of sentiment from a full document. This could also have many important benefits as it might provide a means for obtaining important sentiment information from a document while eliminating noise. In these ways, justification mining could provide value to numerous fields, including NLP, ML, finance and AI explainability.

One of the other benefits of this research investigation is expanding research into the advantages of utilising transformer models like Sentence-BERT for a variety of tasks. These newer models are much faster than older variations such as BERT (Reimers and Gurevych, 2019), and, thus, more readily accessible and applicable for numerous tasks. Justification mining benefits from this increased speed and took on average only two seconds for each document mined. This allows it to be used without tools such as GPU clusters. Moreover, encoding sentence embeddings with methods like Sentence-BERT and transformer models does not require removing any information from text, a problem that hampers the Bag of Words method. Creating a numericisation technique that does not lose any potentially valuable sentiment information could provide numerous potential benefits in Natural Language Processing research, not just in the field of finance but in many other fields as well. Applications like this where transformer models demonstrate an advantage might also lead to wider use of these methods in NLP and Machine Learning research. This could also have potential implications in the field of finance if, for instance, financial datasets were utilised for creating pre-trained models in methods like Sentence-BERT. A pre-trained model using 10-K data, for example, might have led to even better sentence embeddings in this research.

Another benefit of this research arises from the slower timeframes for these types of financial strategies. Since speed is not a huge factor in financial strategies based around sentiment and semantic analysis (unlike in fields such as high-frequency trading), there is time to do things such as look at the motivations and justifications behind sentiment. Financial decisions do not need to be reached in fractions of seconds in order to be viable or lucrative. Unlike high-frequency trading, where computational and network speed are the primary competitive advantage, sentiment and semantic analysis strategies follow more traditional financial strategies where information is the competitive advantage, and justification mining would create more information.

The simplicity of the output of justification mining (research sub-question 2-1) also has several implications. There are some reasons why simpler/explainable strategies might be favourable in finance. Financiers might not have enough confidence in complex machine learning algorithms to rely

on them all the time. Simpler strategies might also be easier to sell to investors. Justification mining provides a tool that might allow for strategies to become more explainable. Being able to capture document sentiment with only a few sentences could have benefits in areas like AI explainability and machine learning model interpretability. Unlike other methods, such as the LIME framework (Ribeiro et al., 2016) or the Skater Project (Skater developers and contributors (MIT License), 2017) (see Section 9.1 for further discussion on these methods and AI explainability), justification mining does not output complex interpretations of a model made for other machine learning experts. Its output (i.e. sentences) provides benefit in its simplicity, allowing it to be a tool for explaining machine learning strategies to investors in the area of finance, as well as potentially increasing their confidence in such strategies. Moreover, besides investors, it could also provide important and easily interpretable information to fund managers. This information could, for instance, allow such managers to take into account important factors from sentiment analyses and apply them to their decision-making process. One potential benefit of doing so might be to drive more ethical or responsible decision making. Such a contribution has many implications in a field such as Responsible Research and Innovation (this is expanded upon in Section 11.1). Finally, interpreting models in this way might also help fund managers to better troubleshoot sentiment analysis-based financial strategies that worked at first but begin to falter over time. Providing information about what an ML model is doing could allow more people to contribute to its troubleshooting not only in the realm of finance but potentially any field where there are currently few machine learning experts.

9 Research Contribution and Gaps in Literature

One of the first contributions of this research is towards aggregation and transfer learning methods in machine learning. To the best of my knowledge, very few research papers attempt to apply transfer learning from lower-level data (sentences in this case) to entire documents. This study attempts to achieve successful methods in doing so. The benefits of such methods could be vast, as annotation of complex and lengthy data like 10-K filings is costly, time-consuming, difficult to crowd-source, and requires expert-level domain knowledge. This is likely the reason why there are no studies, to the best of my knowledge, up until now that utilise 10-K filings annotated at a document-level. Therefore, this research also provides contributions in the evaluation of novel ways to perform sentiment analysis and other NLP and machine learning tasks on 10-K data. The value of information contained in 10-K forms is well-known within the financial industry, with many from Berkshire Hathaway's Warren Buffet to Fidelity Investment's Christopher Bartel having praised the annual filings (MacBride, 2014). Moreover, academic research has also had some focus on discovering ways to evaluate and mine data from 10-Ks (Loughran and McDonald, 2011; Bodnaruk et al., 2015; Loughran and McDonald, 2016; Ren et al., 2013; Ball et al., 2012; McDonald, 2019a).

In the NLP domain, how best to extract and aggregate justifications for sentiment is an open question. A principal contribution of this thesis is therefore to identify suitable machine learning methods to do so, in other words, a practical means to mine justifications behind sentiment analyses (research question 2). Justification mining also provides contributions to the area of sentiment summarisation and capturing the sentiment of an entire document with a few sentences. As justification mining is a new method, it is difficult to find comparisons in current literature. There are fields of study like opinion mining or argument mining but after completing the research investigation for justification mining, I realised these fields are not actually very similar in methodology or goal to justification mining. Being a novel technique in and of itself allows justification mining to provide a valuable methodological contribution to the literature in areas like machine learning and natural language processing (particularly sentiment analysis). It also has potential to contribute to many other fields, including AI explainability, model interpretability, finance, Responsible Research and Innovation and ethics.

As stated in Section 6.3.1, machine learning and statistical methods are not often utilised in financial sentiment analysis (Malo et al., 2013). This thesis strives to contribute to both reasons Malo et al. (2013) attribute to this: the lack of practical applications and the lack of good quality training data, which is often reserved only for proprietary use (O'Hare et al., 2009).

Being able to mine justifications for sentiment analysis has three main practical applications. Justification mining could allow incorporation of ethical considerations into the financial decision-making process (expanded upon in Section 11.1). While there has been some work in creating tools surrounding ethical decision-making and frameworks for moral decision-making in machine learning (Noothigattu et al., 2017; Conitzer et al., 2017), it has not involved word/sentence embeddings or justification mining. The strategy proposed here, if successful, could be generalised beyond the scope of ethics. For instance, end-users (such as managers at investment firms) could be interested in certain markets and target justifications focused or based on these markets. It can also help troubleshoot financial strategies that worked at first but began to falter as time progresses by providing more information on the decisions of the often black-box machine learning methods utilised in finance.

Justification mining might provide insight as to why this happens or what has changed. Finally, justification mining might have many practical applications in the areas of AI explainability and model interpretability (expanded upon in the following subsection, Section 9.1).

To address the second reason provided by Malo et al. (2013), one of the simpler but potentially tremendous contributions of this thesis is the creation of a new financial dataset for training in tasks related to computational finance and NLP. 10-K data is a large and understudied source of financial information (See Sections 6.3.1 and 6.3.2). In Section 6, a corpus of parsed and annotated 10-K filings was created. Within that data, the Management Discussion and Analysis subsection is argued as being the most informative towards evaluating tone and sentiment (Loughran and McDonald 2011; Loughran and McDonald, 2016; Ren et al., 2013; MacBride, 2014; Ball et al., 2012). Section 7 created a new dataset of MD&A Overview subsections. Parsing this subsection is difficult if not impossible to code and must be done manually (McDonald, 2019b). While this subsection (MD&A) has been utilised in previous financial NLP research (Loughran and McDonald, 2011; Bodnaruk et al., 2015; Loughran and McDonald, 2016; Ball et al., 2012; Ren et al., 2013), it is not often used for machine learning methods such as sentiment analysis (Ren et al., 2013). Thus, creating a corpus of the MD&A Overview subsections from numerous 10-K filings would provide a good contribution to the field of computational finance and allow for more document-level NLP and machine learning tasks related to financial text as well as more evaluation of the benefits of these tasks in financial alpha generation. Unlike datasets of entire 10-K filings, this MD&A Overview dataset provides a much cleaner, NLP-friendly, dataset that contains very future-looking text. NLP-friendly refers to having no charts, graphs, html tags or other factors that could potentially impede performance of Natural Language Processing or machine learning tasks on the data. While this was not the focus of the research investigation, it is nevertheless an important contribution of this thesis in areas like computational finance research.

Another contribution to the field of finance is through finding statistically significant correlation of mined justifications sentiment with one-month stock returns. This has important implications towards the benefits of the information contained within 10-K filings and the time windows in which that information might be actionable. It also supports the idea that justification mining might provide value to the interpretation of sentiment in many other financial documents.

Furthermore, the research in this thesis found new and interesting ways to utilise novel state-of-the-art transformer models like Sentence-BERT. Models like these are still quite new and finding novel ways to use them (such as for numericisation of text or justification mining) provides methodological contributions to fields like natural language processing and machine learning.

Understanding the driving forces behind sentiment in a document, particularly in finance, has many applications, including, as discussed previously, helping managers make more ethically informed decisions (see Section 11.1), as well as convincing investors by making machine learning-based strategies more explainable. This could potentially be one of the most important contributions of this research, particularly to the fields of AI explainability and model interpretability.

9.1 AI Explainability

Other than sentiment summarisation, correlation of mined justifications sentiment to one-month stock returns and utilising transformer models for feature engineering or numericisation, perhaps one of the less practical but nevertheless important applications of this research is in the domains of AI explainability and model interpretability.

Two of the more well-known examples of AI explainability in the literature are the LIME framework and the Skater Project. One of the biggest limitations to explaining machine learning based models to investors is often the black box nature of certain models. Understanding this black box nature locally is one of the goals of the LIME framework (Ribeiro et al., 2016).

The Skater Project is a tool that strives to deliver model agnostic analyses or understandings for predictive machine learning classifiers. This tool probes machine learning models in order to understand the decision-making process on which they are trained. In this way, it too strives to comprehend the black box nature of AI models (Skater developers and contributors (MIT License), 2017).

There are, however, several issues with these methods. The Skater Project's model explanation is only feasible for smaller sized documents, otherwise it would become incredibly messy. This is because it analyses machine learning model features and their prediction probabilities by doing things like highlighting features with their class prediction probabilities in different colours on the whole text being evaluated or by making prediction probability graphs that include only the top ten features that contribute to the prediction (Skater developers and contributors (MIT License), 2017). As you can imagine, listing the whole document is not useful for larger documents and including only the top ten features might not make sense if the features are single words or even sentences in an extremely large document. This method also does not seem to work unless making predictions on entire documents and not when aggregating predictions made on individual sentences within a document.

Justification mining is somewhat simpler in its output and suitable for larger documents as well. The methodology of justification mining is also different, as the Skater project utilises prediction probabilities to ascertain important features and justification mining uses sentence embeddings, clustering models and cosine similarities to mine representative sentences. Another important distinction is that tools like the LIME framework and Skater Project are more geared towards providing model interpretability for other experts or at least for those with more than superficial understanding of machine learning. Justification mining, on the other hand, has the potential to provide model interpretability for non-experts as well (relating back to research sub-question 2-1).

10 Research Challenges and Limitations of Research

10.1 Transfer Learning Challenges

The challenges and limitations of performing sentiment analysis and transfer learning using long, document-level 10-K data have been discussed throughout Section 6. Here, some of these along with new challenges are outlined and summarised to further address research sub-question 1-1.

Availability of data is a key research challenge, particularly of good-quality annotated financial data (See Sections 6.3.1 and 6.3.2). While policymakers, politicians and the firms themselves are the ones who could decide how much data and resources are available to researchers overall (a process that could take large amounts of time and be met with severe pushback by many of the other actors in the area such as investors and businesses), there are other routes that can be taken. For example, increasing the access to data between the international academic communities could be a good early activity (Foresight: The Future of Computer Trading in Financial Markets, 2012). Others have discussed bringing together the various actors involved (policymakers, academics, investors and general public) to better decide how to proceed with the analytical work, leading to wider acceptance of the results of such work (Foresight: The Future of Computer Trading in Financial Markets, 2012). One of the specific challenges in this thesis was the requirements of a dataset suitable to three tasks (sentiment analysis, justification mining and correlation to future stock returns). Even when one was located, 10-K filings, it presented new challenges in the form of annotating such large and complex text. This was the primary reason for deciding to pursue new transfer learning methods to accomplish classification of these filings (with available annotated datasets from the literature) as well as devising novel means of annotating such lengthy documents. Even after deciding to employ transfer learning, however, the availability of good quality, opensource annotated financial data to use for training models was another challenge. The lack of availability of large annotated training datasets prevented use of many newer classification models, particularly in the area of deep learning (see Section 6.4.2.1).

The reliance of machine learning techniques on training data also creates two issues. The first is that it is inherently biased towards the contexts that are able to be sampled and the ways researchers annotate. Some groups will be underrepresented and some over (Schnoebelen, 2017). The second issue is that any categories that are created are usually fallible. They can be meaningful and significant but should not be considered innate or indisputable (Schnoebelen, 2017). Categorisation always valorises some points and negates others (Bowker and Star, 1999).

10.2 Justification Mining Challenges

There were numerous challenges and limitations that arose throughout the justification mining research investigation. One of the first was choosing an appropriate method for the evaluation of justification mining. The evaluation of unsupervised learning models is still an open and debated question in the field of machine learning (Roberts, 2020). There were no ‘best’ or ‘most ideal’ methods for evaluating justification mining. In fact, to figure out suitable methods, I had to consult

with various experts/professors, many of whom had differing opinions (Roberts, 2020; Amel-Zadeh, 2020; Simpson, 2020).

As with Transfer Learning in Section 6, another challenge came from obtaining the data itself. Automatic parsing of the Management Discussion and Analysis Overview subsections did not prove to be possible, at least not without too many errors that would need to be individually addressed in the code. Parsing 10-K data has proven difficult throughout the research investigation in Section 6, and Section 7 was no different. In the end, the MD&A text had to be manually parsed and organised into a *.csv* file, a long and extensive process. In addition, throughout the research in Section 7, there were instances where smaller parts of the data had to be manually annotated in one way or another to provide preliminary and exploratory testing of a method before committing to that method. These also proved to be a time-consuming tasks that would not be feasible on a much larger scale.

There are also challenges associated with sentiment and semantic analysis itself. The first involves the limitations of absolute classification in binary terms (Kessler, 2014), which is why three-class labelling was prioritised. Furthermore, another limit of current sentiment analysis techniques utilised in finance is that they do not consider the justifications/motivations behind the sentiment. Thus, they leave themselves open to not only pursuing strategies with problematic justification (Pope and Vasquez, 2016) but also to black box strategies that are difficult to fix when things go wrong. However, the solution to this, one problem this thesis is trying to address, is a difficult one.

On top of this, another challenge presented itself through the many choices that had to be made during the research investigation in Section 7. These included the variables or combinations of factors for models (such as sentiment analysis models). Numerous different combinations of variables were utilised during the sentiment analysis training and prediction steps (see Tables 17 to 24 in Section 7.6.4). While many important conclusions were found with these combinations, there is always the possibility that other combinations might have proven more valuable or fruitful. This is likely one of the limitations of a smaller study like a DPhil thesis (as opposed to larger corporate research studies). The same principle also applied to choosing what methods to utilise for evaluation of justification mining. Selecting statistical tests based on the data is not an exact science, so there is also some limitation there if other choices might have been better. For instance, a full portfolio optimisation analysis with backtesting might have led to consideration of other factors like transaction costs; however, due to the much larger data requirement - typically, it would take around 200 to 300 firms over at least a 12-year range (Amel-Zadeh, 2020) - it was not a practical option within the scope of this thesis.

One of the methodological limitations of justification mining is the restrictions on the input text. The input text has to be formatted properly, since the method utilises the *sent_tokenize* function from the nltk library. If text has html tags or tables and charts, it will not properly work when inputted into the justification mining algorithm. Because of this, text might require numerous preprocessing steps, such as with tools like the Regex library in Python, before it can be used. This could prove to be very frustrating and limiting. As was shown in both Sections 6 and 7, it is not always possible to automatically parse and clean text, and manually parsing and cleaning text so that it can be read by the justification mining algorithm can prove very time-consuming.

Finally, another limitation of this research was the time these models take to work. Because both justification mining and Sentence-BERT utilised deep learning transformer models, they were slower than traditional approaches like Bag of Words or sentiment analysis on full documents (recall, even when using SBERT for numericisation, sentiment analysis on full documents was faster than sentiment analysis on mined justifications because of the extra time it took to mine the actual justifications). Due to this, I also had to attempt many different approaches when writing the code to try finding faster methods to make up for the extra time taken by deep learning transformer models. However, as has been shown through the various evaluations performed, justification mining and Sentence-BERT did prove advantageous in numerous aspects, potentially making up for this increased computing time. However, it would still be beneficial to try finding even more ways to speed up the process.

11 Potential Future Directions for this Research

There are several directions of future work for the transfer learning part of the thesis (Section 6). The first could be trying to find a larger financial training dataset for transfer learning. This would allow re-evaluation of the deep learning models (LSTMs and bi-directional LSTMs) in a proper context. For these deep learning models, I could also try initialising the embeddings with transformer models like Sentence-BERT instead (elaborated upon later in this subsection when discussing transformer models), or with *word2vec/sent2vec* from the Gensim library trained on the entire 10-K text corpus of every company since 1994 available on Professor McDonald's repository (McDonald, 2019a).

Another avenue to try might be to annotate random sentences from various 10-K filings in the aforementioned repository (McDonald, 2019a) and then utilise these instead as the training data to make predictions on document-level 10-K filings. This might have the benefit of creating a training dataset specific to 10-K filings, which would be an invaluable asset in financial firms if it worked and generalised well (as 10-K filings are readily available, required by law to be truthful, and figuring out how best to utilise them as big data is an open question in computational finance research). Using training data from the same source (10-K filings) as the testing data might also lead to better performance metrics over using training data from another source (such as Malo et al., 2013).

In addition to these, there are numerous future directions of work with justification mining and transformer models (Section 7) that could prove valuable. One obvious direction would be to create more variable combinations for sentiment analysis on mined justifications (see Tables 17 to 24 in Section 7.6.4) based on what worked well in this study. For instance, not many combinations utilised the NLI pre-trained model for Sentence-BERT, although this model ended up performing very well in a variety of the evaluations. Thus, in future research, more combinations could be trialled with this model instead of the Wikipedia pre-trained model.

As discussed in previous sections, another option for future work would be to continue investigation into how mined justifications could potentially be eliminating noise in the sentiment signal from a financial document. One way this could be tested would be to check those cases where the aggregated sentiment predicted from the full MD&A Overview document differed to the aggregated sentiment predicted from mined justifications. For instance, if full document aggregated sentiment is neutral but the aggregated sentiment from mined justifications is negative, I can check to see if perhaps all the neutral sentiment sentences from the full document are being grouped into only a few clusters and the negative sentences grouped into more clusters and thus returning more justifications, outweighing the overall neutral sentiment (in other words, smaller but more numerous negative clusters than neutral clusters). To elaborate, in this example, the full document has many more neutral sentences, leading to a neutral full document aggregated sentiment, whereas there are more negative clusters, leading to more negative mined justifications and a negative mined justifications aggregated sentiment (assuming mined justifications have the same sentiment as most of the other sentences in a cluster, which was the case in the initial trials). This could represent some neutral sentiment noise that justification mining is potentially eliminating. Sentiment analysis on mined justifications is, in essence, only interested in sentiment per cluster, which theoretically each represent a common theme, and so this sentiment is possibly a representation of the themes of a document rather than the entire document. To begin researching into this, the code can be rewritten to return the number of sentences in each cluster along with the sentiment prediction for each mined justification or even each cluster

rather than just the overall aggregated sentiment score. These values can then be analysed in cases where the aggregated sentiment prediction differed between full documents and mined justifications.

Another interesting direction for future work related to signals versus noise would be to check if certain sentiment labels have stronger correlation to future returns than other labels. In order to test this, the data could be segmented into the three labels used and the correlation analysis could be performed again. Appendix 3 (Section 14.3) shows that, in some combinations or plots, there were large amounts of neutral labels predicted by the ML models, while in others almost no neutral labels were predicted. So, it might be interesting to check what the correlation would be without neutral labels present and whether that could lead to a stronger correlation. Likewise, any of the other two labels could be removed to check if that leads to larger Pearson coefficients. The question would be if it is possible to get a stronger correlation by listening to the signal only sometimes; in other words, trying to pick out the times when it is more likely to be wrong or right (Simpson, 2020).

Also mentioned in earlier sections was the possibility of combining the results from justification mining with more traditional financial signals. This was because the Pearson coefficients, while statistically significant, were still rather low, indicating a weak correlation. So, combining them with factors like momentum-based indicators (such as moving average), trend indicators (such as average directional movement) or statistical indicators (like entropy or variance) could prove successful in enhancing those signals. To get these signals, a Python library such as *pandas_ta* could be employed. This open-source library can take price information from stocks and use it to create a large variety of financial indicators, such as the ones mentioned above. Of course, with the price data, these indicators could also be calculated manually for more flexibility, however, the *pandas_ta* library provides a quick resource for obtaining many financial indicators that have the potential to be enhanced with aggregated sentiment scores from justification mining. One method for doing so could be through portfolio optimisation.

Potentially the most interesting application for justification mining and a direction for future work involves portfolio optimisation. It entails creating alpha features/factors from the results of Section 6 (transfer learning classification on 10-Ks) and Section 7 (justification mining sentiment analysis) to be utilised alone, together or in combination with traditional financial alpha factors (like mean reversion, momentum, etc.) in order to evaluate their efficacy as alpha signals (a process known as portfolio optimisation). Alpha, in the financial domain, is a portfolio's excess return above and beyond a chosen market benchmark or index (Jensen, 1968; Jensen, 1969; Metts, 2008). The portfolio, in this case, would consist of the firms utilised in Sections 6 and 7 of this study along with approximately another 300 with filings going back at least ten years, as portfolio optimisation requires a much larger number of firms and data (Amel-Zadeh, 2020). This process can be performed with Python's alphalens library, which can be utilised to calculate factor returns on these created alpha signals, as well as Python's zipline library. Alphalens, which is developed by Quantopian, is used for analysing performance of alpha factors. It allows for the creation of a workflow that involves aspects such as: selecting a universe of stocks, single alpha factor modelling and alpha combination, risk modelling, and portfolio construction. Quantopian's zipline package (which is a quantitative trading simulator that is open source) can be used to construct a pipeline to create additional, more traditional alpha factors to combine with the ones created from justification mining (Quantopian, 2016; Quantopian, 2020). This portfolio optimisation could provide some interesting results and represents another practical application of this research.

Although there was no time to do any work for portfolio optimisation in the scope of this thesis, during my research for justification mining, I come to a realisation regarding ways to use mined justifications and weighted sentiment scores as alpha features/factors that could contribute to the literature. Often, when sentiment from sentiment analysis is used as an alpha feature, it is given alpha scores of 1, -1 and sometimes 0 (to represent favourable, unfavourable, or neutral sentiment respectively). With justification mining and sentiment aggregation methods, however, it is possible to expand this by utilising weighted sentiment scores, as discussed in previous sections. There are two ways to do so. The first involves using the mean aggregation method employed in Sections 6 and 7. The mean aggregated sentiment score itself would be used as the alpha feature. The second is to count the number of justifications returned with sentiment that matches the overall sentiment of the document. So, for instance, in the example from the Nordstrom 2017 MD&A Overview data presented earlier (during the initial trials of justification mining in Sections 7.3, 7.4 and 7.5) this number would be three since there were three clusters that had favourable sentiment tone (see Table 14 in Section 7.5.1). This number could then be weighted through dividing by the total number of justifications and utilised as an alpha feature, with the sign changed to negative if the sentiment were negative, thus providing a kind of scaled sentiment analysis (similar to mean aggregated sentiment scores but ignoring the other two labels as discussed previously). Both of these methods would add another layer of depth over traditional sentiment scores, and it could be interesting to check how informative this added depth is for future returns. Their performance can be evaluated using a single alpha factor model with alphalens or they could be combined with other traditional alpha factors using zipline and alphalens. Another interesting application involving portfolio optimisation might be to utilise what has been learnt about sentence embeddings and cosine similarities or Pearson correlation in other interesting ways. For instance, cosine similarity and Pearson correlation could be performed on embeddings from 10-K filings in consecutive years to detect large changes in sentiment between years. Since 10-K filings do not typically change drastically from year to year (Ren et al., 2013; Loughran and McDonald, 2011; Loughran and McDonald, 2016), a large change in sentiment could imply issues within the company and might potentially be a powerful alpha signal.

Another direction of future research involves utilising embeddings from transformer models to initialise the weights for the embedding layer of neural networks. This would entail training deep learning classification tasks in combination with the novel transformer models (like Sentence-BERT) discussed previously, allowing the option to not only utilise these transformer models but also to trial their effectiveness with deep learning models given a small training dataset (recall, deep learning models did not work well with small training datasets in Section 6). Since Sentence-BERT does not require a large dataset in the downstream task (Reimers and Gurevych, 2019), using it to initialise the weights for the embedding layer of neural networks might be a way to overcome the small dataset problem that occurred in Section 6.

The weights for the embedding layer of these recurrent neural networks, such as LSTMs, could be initialised with the sentence embeddings extracted using either the pretrained datasets made available by Reimers and Gurevych (2019) for Sentence-BERT or by training a Sentence-BERT embedder model using the entire 10-K text corpus (available on Professor McDonald's website: McDonald, 2019a). However, as discussed previously, creating such a large new pre-trained model for Sentence-BERT would require a massive crowd-sourcing annotation effort that would take a lot of time and resources to complete. These deep learning classifier models (whose embedding layer weights were initialised with Sentence-BERT embeddings) could then be trained on a small selection of sentence-level annotated MD&A subsections. Finally, the trained models could be employed to make predictions on entire MD&A subsections utilising the aggregation methods from Sections 6 and 7. This sentiment could also be directly used as an alpha factor in portfolio optimisation. Doing so would

not only be interesting but also novel, as these types of transformer models (Sentence-BERT) have not been utilised before in combination with deep learning models for portfolio optimisation. Transformer models like Sentence-BERT are still new and testing their effectiveness in this manner and domain would provide a valuable and meaningful contribution in the realms of finance, NLP and machine learning.

11.1 Making More Ethically Informed Decisions with Justification Mining

Another important direction of future research could focus on the second part of research sub-question 2-3. Although there was no time to address using justification mining to help make more ethically informed decisions in finance within the scope of this thesis, there are some ways this could be addressed in future work. To do so, mined justifications could be presented to end-users (such as managers at financial institutions) to allow them to annotate the justifications. This annotation could cover any number of possibilities, but here they would annotate based on whether the end-user believes the motivation behind the mined justifications' sentiment is ethical or not (as opposed to the training data used throughout this thesis where sentences were annotated based on whether they indicated a rise, fall or no change in stock prices). This would require a large and lengthy crowd-sourced annotation effort, which is why it was not possible to do for this study. Once the justifications have been annotated, a classifier model could be trained on them in order to allow it to recognise ethical or unethical justifications from new data. If this classifier could successfully identify justifications which end-users would find to be ethical or neutral, it could provide valuable information to decision-makers to assist making ethically informed decisions in their investment strategies. If the justifications behind sentiment in a financial document are unethical, for instance, decision-makers might choose not to act on such information. Therefore, knowing the driving factors behind the sentiment of a document could provide valuable insight into whether or not to utilise that information in financial decision-making. In this way, justification mining could relate back to the previous discussions on Responsible Research and Innovation as well as ethics.

Responsible Research and Innovation (RRI) is a field in which academics, policymakers and businesses are increasingly concerned/interested. Sabine Louet (2016) argues an interesting point with regards to utilising RRI to tackle Grand Challenges. She states that science, technology and innovation are already incorporated into society and vice versa. In this sense she classifies RRI as a wake-up call (Louet, 2016). Certainly, it seems evident that scientific and technological innovation and societal goals have always gone hand in hand with one another. However, it is only now, maybe due to the emergence of these Grand Challenges, that society can begin to see the necessity of responsibility and ethics. There are several definitions of RRI, all of which include discussions of ethics (von Schomberg, 2011; The Directorate-General for Research and Innovation of the European Commission, 2012; Stilgoe et al., 2013; Sutcliffe, 2011; Grunwald, 2007; Grunwald, 2014; Coenen and Grunwald, 2017; van den Hoven et al., 2013). Regardless of which definition of RRI is favoured, all the definitions have a top-level in common, a process and outcome dimension (Iatridis and Schroeder, 2016). Process refers to responsibility being a collective effort and RRI having a holistic nature. The outcome dimension refers to RRI going beyond what is not wanted and focusing instead on what is wanted and what creates public benefit and value (Iatridis and Schroeder, 2016). Most of the definitions also focus on responsibility being a meta-level task, a level above individual responsibility (Stahl, 2013). In this potential future direction of research, the focus is on a specific RRI concern – ethics – as the factor that influences what is favourable and what is not in financial decision-making. Ethics provides not only a general and well-studied framework for making these decisions but is often a concern in the domain of finance, particularly after the last major financial crash. However, in some computational finance strategies, the use of black box techniques might limit

the consideration of ethics. One of the main impediments in automatic ethical decision-making is the absence of a formal specification of ground-truth ethical principles. When these are absent, an "*approximation as agreed upon by society*" could be an alternative (Dwork et al., 2012, p. 214). In order to even begin to create an accordance, however, the relevant information is required. This is where justification mining might play a part; it can allow insight into the motivation behind model sentiment predictions. While it may not be feasible for machine learning methods to reveal all ethical issues in investment decision-making, they can begin to underscore issues of which decision-makers may not have been aware or sufficiently taken into account. The importance and implications of this question have ramifications for not only machine learning but also finance and RRI. The idea here, however, is not to debate theoretical frameworks for ethics or ethical decision-making in finance. Instead, the focus is on techniques to obtain the relevant data so end-users might have the tools to begin considering ethics in their decision-making. Again, end-users refer to investment managers or employees at applicable financial firms who are involved in the decision-making for investments.

In summary, two main ethical dimensions are associated with this approach. The first is whether the justifications mined from the input data (the financial text corpus) are themselves ethical in nature. The second is the ethical implications of utilising machine learning to make certain decisions. Is it ethical to provide a strategy as a black box that makes mistakes from time to time? Would it be more ethical for the system to provide reasons for what it is doing? In doing so, the methods described for justification mining (which strive to provide those reasons or justifications) have an inherent Responsible Innovation aspect (Geoghegan-Quinn, 2012; The Directorate-General for Research and Innovation of the European Commission, 2012). For example, if an investment bank or private equity firm is deciding to invest in an energy company, a sentiment analysis of relevant data might suggest it is a good investment, but the justification for that sentiment might simply be that it will make a lot of money. Perhaps, however, the investors have other concerns such as renewable energy or want to avoid companies that engage in activities such as fracking. Another example might be negative sentiment classification towards investing in a particular company, but perhaps the justification for that sentiment is because the CEO is a woman. Such a gender bias might be not only unethical and unlawful but also generally off-putting as an investment strategy. Of course, different people in different situations might have different requirements regarding which justifications are acceptable. The strategies and methodology described here allow for a very flexible approach that simply provides end-users with tools and information they can utilise to make better-informed decisions. However, such flexibility could also mean such a tool might allow for an end-user to be less neutral and more biased, since they could dismiss justifications based on reasons they do not like. This could be another important ethical dimension, particularly in cases where there are no ethical or legal obligations involved. Bias in those instances could simply be a result of personal preference.

While there was no time during this research investigation to examine whether justification mining could be employed in this manner, to help make more ethically informed decisions in finance (see research sub-question 2-3), the potential research contributions can be discussed along with the gaps in literature addressed by this method. There is an increasing amount of work being done in technical fields (such as quantum computing and nanotechnology) focused on ethical concerns (Floridi, 2010; Jirotka and Inglesant, 2016). However, there is not much work in the area of RRI and intelligent algorithms, particularly little around sentiment analysis and justification mining. The closest research I could locate to this is McBride and Stahl's paper on robotics (McBride and Stahl, 2014). McBride and Stahl note that little work has been done in creating a tailored methodology that embeds RRI into the engineering and research of robotics itself (McBride and Stahl, 2014). However, as mentioned in Section 5.6, their approach focuses on robotics where a physical product is manufactured and is not

directly applicable to this thesis. Moreover, McBride and Stahl (2014) primarily discussed responsibility of the engineers.

This future direction of work also potentially provides novelty in the realm of finance. Finance is an area that is often the subject of much ethics discussion but in practice is typically not concerned with anything other than financial gain and return on investment, referred to as alpha in finance. Being able to create ML based strategies with RRI concerns like ethical investment in mind allows for a unique opportunity to automatically factor these concepts into decisions, something fund managers might not typically do. Lastly, there are benefits to the areas of machine learning and NLP. The notion of whether machine learning algorithms can inform moral decisions is not a new one. This approach could consider a novel means for creating a machine learning program that is capable of providing information related to ethical factors.

To summarise, this tool would function by attempting to embed end-user specifications into the machine learning process by learning what the end-users consider ethical and potentially automatically presenting only new information to them that is based on ethical justifications. Of course, there is the issue of "ethical" being subjective, but using accordance to determine what is ethical is a much larger discussion. Besides the inherent challenges in creating good and accurate machine learning based techniques here, there will always be subjectivity when discussing which justifications are ethical, moral, etc. The issue is how to make such a decision. In the field of machine learning and artificial intelligence, ethical decision-making has been a grand challenge for a long time (Wallach and Allen, 2008). "*The design of intelligent machines that can make ethical decisions is, arguably, one of the hardest challenges in [machine learning]*" (Noothigattu et al., 2017, p. 7). This is influenced by numerous factors from culture and religion to geographic location and personal bias. However, due to the robust and flexible nature of the research design, the methodology presented in this subsection could begin to address these limitations through justification mining. Regardless, such a tool can provide immense implications for intelligent algorithms tackling issues like incorporating ethical concerns into financial decision-making. The creation of a novel tool for addressing Responsible Research and Innovation (RRI) concerns utilising machine learning, sentiment and semantic analysis with applications in finance and investment strategies could have much value in numerous fields. It would not only be beneficial to the fields of machine learning and NLP but also in the areas of RRI and finance, where such methods have not been applied before.

12 Conclusions

The research questions were presented. Initial literature that contributed to the development of justification mining was reviewed. Research question 1 was addressed through investigation and assessment of transfer learning methods. Research questions 2 and 3 were addressed through the development and evaluation of justification mining. The benefits, novel contributions of this research and gaps in relevant literature were explored. The research challenges and difficulties of this kind of study were discussed. Finally, directions for future work were outlined.

The methods and processing steps utilised for transfer learning achieved performance metrics comparable to those obtained by cross-validation on the training dataset alone, substantiating the effectiveness of the transfer learning methodology presented in this thesis. Simple ML models worked best for transfer learning, with Naïve Bayes obtaining the highest accuracy and F1 score on the 10-K testing dataset.

Notable findings of this research also include a statistically significant correlation between sentiment of mined justifications from 10-K filings and the following one-month stock returns (which could be important for event-driven trading strategies in finance) and the effectiveness of transformer models over more traditional approaches like Bag of Words in creating features / numericising textual data for input into machine learning algorithms (which could have many general implications for quantitative research involving Natural Language Processing and, in particular, sentiment analysis).

The novel means developed for mining justifications from documents with unsupervised clustering and transformer models revealed high degrees of similarity between mined/summarized and full document sentiment scores, supporting the efficacy of this approach in capturing most of the important sentiment in these documents with only a few sentences. These results have important implications for machine learning, AI explainability and model interpretability, as well as practical implications in providing a tool for explaining machine learning / sentiment analysis output in a concise and understandable way to various groups in the financial domain that are not as well-versed in machine learning methods (such as investors, quant traders or other, non-technical teams). Based on the trials performed earlier in Section 7, utilising the Nordstrom 2017 MD&A Overview data (see Sections 7.2, 7.3, 7.4 and 7.5), justification mining does have the potential to summarise well the main sentiment driving points of a document. Mined justifications are tangible and easily presentable pieces of information that can be used in numerous applications, from convincing investors of what sentiment analysis algorithms are doing and their merit, to knowing the motivating factors behind sentiment in a document to assist in, for instance, more ethically informed decision-making.

The successful strategies outlined in this thesis could lead to a wider consideration of justification mining, sentiment summarisation, and transformer models, as well as RRI concepts and tools in finance and machine learning. In striving to provide the justifications behind sentiment analysis algorithm decisions, this research has an inherent AI explainability and RRI aspect (Geoghegan-Quinn, 2012; The Directorate-General for Research and Innovation of the European Commission, 2012) and so it not only has relevance for the fields of machine learning and NLP but also for AI explainability, model interpretability, RRI and finance.

13 References

Al-Hagery, M., 2016. Google Search Filter Using Cosine Similarity Measure to Find All Relevant Documents of a Specific Research Topic. International Journal Of Education And Information Technologies 10, 1-15.

Amel-Zadeh, A., 2020. Personal Communication. [email and phone conference]

Anderson, M., Anderson S., 2011. Machine Ethics. Cambridge University Press, 1-7.

Angelaki, M., 2016. An Introduction to Responsible Research and Innovation. Pasteur 40A, Available at: http://www.pasteur4oa.eu/sites/pasteur4oa/files/resource/RRI_POLICY%20BRIEF.pdf [Accessed: 30 June 2017].

Armstrong, M., Cornut, G., Delacôte, S., Lenglet, M., Millo, Y., Muniesa, F., Pointier, A., Tadjeddine, Y., 2012. Towards a Practical Approach to Responsible Innovation in Finance: New Product Committees Revisited. Journal of Financial Regulation and Compliance 20, 147-168.

Asimov, I., 1950. I, Robot. Gnome Press, 1-253.

Bahdanau, D., Cho, K., Bengio, Y., 2015. Neural Machine Translation by Jointly Learning to Align and Translate. International Conference on Learning Representations, 1-15.

Ball, C., Hoberg, G., Maksimovic, V., 2012. Redefining Financial Constraints: A Text-Based Analysis. Forthcoming in Review of Financial Studies, 1-55.

Banga, J., Brorsen, B., 2019. Profitability of alternative methods of combining the signals from technical trading systems. Intelligent Systems in Accounting, Finance and Management Volume 26, Issue 1, 32-45.

Batmanghelich, K., Saeedi, A., Narasimhan, K., Gershman, S., 2016. Nonparametric Spherical Topic Modeling with Word Embeddings. Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, 537–542.

Baugh, J., 2016. Linguistic Profiling and Discrimination. The Oxford Handbook of Language and Society, 349–368.

Bender, E., 2012. 100 Things You Always Wanted to Know about Linguistics But Were Afraid to Ask*. Tutorial Abstracts at the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, 1-2.

Bengio, Y., Ducharme, R., Vincent, P., Jauvin, C., 2003. A Neural Probabilistic Language Model. Journal of Machine Learning Research 3, 1137–1155.

Bodnaruk, A., Loughran, T., McDonald, B., 2015, Using 10-K Text to Gauge Financial Constraints. Journal of Financial and Quantitative Analysis, 50(4), 1-24.

Bollen, J., Mao, H., Zeng, X., 2010. Twitter mood predicts the stock market. arXiv:1010.3003v1 [cs.CE], 1-8.

Bowker, G. C., Star, S. L., 1999. Sorting things out: classification and its consequences. MIT Press, Cambridge, MA.

Brandseye, 2017. What is opinion mining? *Brandseye*, [Online] 25 May. Available at: https://www.brandseye.com/news/what-is-opinion-mining-next-level-sentiment-analytics/ [Accessed: 2 Dec 2017].

Brew, A., Greene, D., Cunningham, P., 2010. Using crowdsourcing and active learning to track sentiment in online media. In: Coelho, H., Studer, R., Wooldridge, M. (eds) Frontiers in Artificial Intelligence and Applications. IOS Press. ECAI 2010 19th European Conference on Artificial Intelligence vol 215, 1-11.

Brownlee, J., 2017. Master Machine Learning Algorithms. Edition 1.9, 1-163.

Brownlee, J., 2017b. What Are Word Embeddings for Text? [Online]. Available at: https://machinelearningmastery.com/what-are-word-embeddings/ [Accessed: 20 October 2017].

Burton, C., 2016. Ethics in Machine Learning – Summary, viewed 25 June 2017, < http://www.kdnuggets.com/2016/06/ethics-machine-learning-mlconf.html>.

Burton, C., 2016b. Ethics in Machine Learning: What We Learned from Tay Chatbot Fiasco? Viewed 25 June 2017, <http://www.kdnuggets.com/2016/03/ethics-machine-learning-tay-chatbot-fiasco.html>.

Callon, M., Lascoumes, P., Barthe, Y., 2009. Acting in an Uncertain World: An Essay on Technical Democracy. MIT Press.

Cer, D., Yang, Y., Kong, S. , Hua, N., Limtiaco, N., St. John, R., Constant, N., Guajardo-Cespedes, M., Yuan, S., Tar, C., Sung, Y., Strope, B., Kurzweil, R., 2018. Universal Sentence Encoder. arXiv:1803.11175v2 [cs.CL], 1-7.

Chunjie, L., Jianfeng, Z., Lei, W., Qiang, Y., 2017. Cosine Normalization: Using Cosine Similarity Instead of Dot Product in Neural Networks. arXiv:1702.05870v5 [cs.LG], 1-9.

Cliche, M., 2017. Twitter Sentiment Analysis with CNNs and LSTMs. In *SemEval 2017: Task 4, 1-8.

Coenen, C., Grunwald, A., 2017. Responsible Research and Innovation (RRI) in Quantum Technology. Ethics and Information Technology 19(4), 277–294.

Cohan, A., Dernoncourt, F., Kim, D., Bui, T., Kim, S., Chang, W., Goharian, N., 2018. A Discourse-Aware Attention Model for Abstractive Summarization of Long Documents. Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers), 615-621.

Conitzer, V., Sinnott-Armstrong, W., Borg, J., Deng, Y., Kramer, M., 2017. Moral Decision Making Frameworks for Artificial Intelligence. Association for the Advancement of Artificial Intelligence, 1-5.

Conneau, A., Kiela, D., Schwenk, H., Barrault, L., Bordes, A., 2017. Supervised Learning of Universal Sentence Representations from Natural Language Inference Data. In Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, Association for Computational Linguistics, 670–680.

Cortis, K., Freitas, A., Daudert, T., Hurlimann, M., Zarrouk, M., Handschuh, S., Davis, B., 2017. Fine-Grained Sentiment Analysis on Financial Microblogs and News. In *SemEval 2017: Task 5, 519-535.

Costa-jussa, M., Nuez, A., Segura, C., 2018. Experimental Research on Encoder-Decoder Architectures with Attention for Chatbots. Computación y Sistemas, Vol. 22, No. 4, 1233–1239.

Crawford, K., 2016. Artificial Intelligence's White Guy Problem. *New York Times*, [Online] 25 June. Available at: https://www.nytimes.com/2016/06/26/opinion/sunday/artificial-intelligences-white-guy-problem.html [Accessed: 22 October 2017].

Devlin, J., Chang, M., Lee, K., Toutanova, K., 2018. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. arXiv preprint arXiv:1810.04805[cs.CL], 1-16.

Dongen, S., Enright, A., 2012. Metric Distances Derived from Cosine Similarity and Pearson and Spearman Correlations. arXiv:1208.3145v1 [stat.ME], 1-5.

Dor, L., Mass, Y., Halfon, A., Venezian, E., Shnayderman, I., Aharanov, R., Slonim, N., 2018. Learning Thematic Similarity Metric from Article Sections Using Triplet Networks. Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers), 49-54.

Dwork, C., Hardt, M., Pitassi, T., Reingold, O., Zemel, R., 2012. Fairness through awareness. In Proceedings of the 3rd Innovations in Theoretical Computer Science Conference (ITCS), 214–226.

Eger, S., Daxenberger, J., Gurevych, I., 2017. Neural End-to-End Learning for Computational Argumentation Mining. ACL, 1-12.

ESRC Framework for Research Ethics, 2015. 1-51.

Ester, M., Kriegel, H., Sander, J., Xu. X., 1996. A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise. In: Proceedings of the 2nd International Conference on Knowledge Discovery and Data Mining, Portland, OR, AAAI Press, 226-231.

Fan, R., Chang, K., Hsieh, C., Wang, X., Lin, C., 2008. Liblinear: A library for large linear classification. Journal of Machine Learning Research 9, 1871-1874.

Farha, I., Magdy, W., 2019. Mazajak: An Online Arabic Sentiment Analyser. Proceedings of the Fourth Arabic Natural Language Processing Workshop, 192–198.

Floridi, L. Ed., 2010. The Cambridge Handbook of Information and Computer Ethics. Cambridge University Press.

Foltz, P., 1996. Latent semantic analysis for text-based research. Behavior Research Methods, Instruments, & Computers 28(2), 197-202.

Foresight: The Future of Computer Trading in Financial Markets, 2012. Final Project Report. The Government Office for Science, 1-175.

García, D., Zuazua, E., Perat, B., López, I., 2016. A Practical Guide to Responsible Research and Innovation: Key Lessons from RRI Tools. Milimétrica Producciones S.L, 1-60.

Gaskell, G., Stares, S., Allansdottir, A., Allum. N., Castro, P., Esmer, Y., Fischler, C., Jackson, J., Kronberger, N., Hampel, J., Mejlgaard, N., Quintanilha, A., Rammer, A., Revuelta, G., Stoneman, P., Torgersen, H., Wagner, W., 2010. Europeans and biotechnology in 2010: Winds of change? A report to the European Commission's Directorate-General for Research. Directorate-General for Research, Science in Society and Food, Agriculture & Fisheries, & Biotechnology, 1-176.

Geoghegan-Quinn, M., 2012. Science in Dialogue - Towards a European Model for Responsible Research and Innovation.

Gidiotis, A., Tsoumakas, G., 2020. A Divide-and-Conquer Approach to the Summarization of Long Documents. IEEE/ACM Transactions On Audio, Speech, And Language Processing, 1-14.

Goldberg, Y., 2017. Neural Network Methods in Natural Language Processing. Morgan & Claypool Publishers.

Grimpe, B., Hartswood, M., Jirotka, M., 2014. Towards a Closer Dialogue between Policy and Practice: Responsible Design in HCI. Proceedings of the 32nd Annual ACM Conference on Human Factors in Computing Systems, Toronto, Canada, 26 April-01 May. ACM Press, New York, 2965-2974.

Grove-White, R., Macnaghten, P., Wynne, B., 2000. Wising Up: The Public and New Technology. Lancaster, UK: CSEC, Lancaster University.

Grunwald, A., 2007. Converging technologies: Visions, increased contingencies of the conditio humana, and search for orientation. Science Direct Futures 39, 380-392.

Grunwald, A., 2014. The hermeneutic side of responsible research and innovation. Journal of Responsible Innovation 1(3), 274-291.

Guo, X., Zhang, H., Tian, T., 2018. Development of stock correlation networks using mutual information and financial big data. PLoSONE 13(4):e0195941, 1-16.

Guston, D., Sarewitz, D., 2002. Real-Time Technology Assessment. Technology in Society 24, 93-109.

Guston. D., 2004. Responsible Innovation in the Commercialised University. In: Stein, D. ed. *Buying in or Selling Out: The Commercialisation of the American Research University*. New Brunswick: Rutgers University Press, 161-174.

Guston, D., 2012. The Pumpkin or the Tiger? Michael Polanyi, Frederick Soddy, and Anticipating Emerging Technologies. Minerva 50, 363-379.

Heuer, H., 2015. Text comparison using word vector representations and dimensionality reduction. Proceedings of the 8th European Conference on Python in Science, Euroscipy 2015, 13-16.

Humphreys, A., Humphreys, J., 2013. Reading Difficulty Levels of Selected Articles in the Journal of Research in Music Education and Journal of Historical Research in Music Education. Music Education Research International 6, 15-23.

Iatridis, K., Schroeder, D., 2016. Responsible Research and Innovation in Industry. SpringerBriefs in Research and Innovation Governance.

Ilya Sutskever, I., Vinyals, O., Le, Q., 2014. Sequence to Sequence Learning with Neural Networks. arXiv:1409.3215v3 [cs.CL], 1-9.

Innova Integra: A Multi-disciplinary Research and Innovation Company 2017, viewed 20 March 2017, <http://www.innovaintegra.com/responsible-research-innovation/>.

Irsoy, O., Cardie, C., 2014. Opinion Mining with Deep Recurrent Neural Networks. Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), Association for Computational Linguistics, 720–728.

Jarvis Thomson, J., 1985. The Trolley Problem. The Yale Law Journal 94(6), 1395-1415.

Jensen, M., 1968. The Performance of Mutual Funds 1954-64. Journal of Finance 23, 389-416.

Jensen, M., 1969. Risk, the Pricing of Capital Assets, and the Evaluation of Investment Portfolios. Journal of Business 42, 167-247.

Jirotka, M., Inglesant, P., 2016. Thinking Ahead to a World with Quantum Computing, viewed 20 March 2017, < http://www.nqit.ox.ac.uk/content/responsible-research-innovation>.

Jirotka, M., Grimpe, B., Stahl, B., Eden, G., Hartswood, M., 2017. Responsible Research and Innovation in the Digital Age. Communications of the ACM 60(5), 62-68.

Joachims T., 1998. Text categorization with Support Vector Machines: Learning with many relevant features. In: Nédellec C., Rouveirol C. (eds) Machine Learning. ECML-98. ECML 1998. Lecture Notes in Computer Science (Lecture Notes in Artificial Intelligence), vol 1398. Springer, 137-142.

Karinen, R., Guston, D., 2010. Towards Anticipatory Governance. The Experience with Nanotechnology. In: Kaiser, M. ed. *Governing Future Technologies. Nanotechnology and the Rise of an Assessment Regime*, Springer, 217-232.

Katiyar, A., Cardie, C., 2016. Investigating LSTMs for Joint Extraction of Opinion Entities and Relations. Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, 919–929.

Kaushik, M., Mathur, B., 2014. Comparative Study of K-Means and Hierarchical Clustering Techniques. International Journal of Software and Hardware Research in Engineering, ISSN No: 2347-4890, Volume 2, Issue 6, 93-98.

Keller, D., 1935. The Living Machine. *Wonder Stories*, May 1935, 227-355.

Kenett, D., Huang, X., Vodenska, I., Havlin, S., Stanley, H., 2014. Partial Correlation Analysis: Applications for Financial Markets. arXiv:1402.1405v1 [q-fin.ST], 1-15.

Kenter, T., Borisov, A., Rijke, M., 2016. Siamese CBOW: Optimizing Word Embeddings for Sentence Representations. Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, Association for Computational Linguistics, 941–951.

Keskar, N., Mudigere, D., Nocedal, J., Smelyanskiy, M., Tang, P., 2017. On Large-Batch Training for Deep Learning: Generalization Gap and Sharp Minima. ICLR 2017, 1-16.

Kessler, S., 2014. The Problem with Sentiment Analysis. Fast Company [Online]. Available at: https://www.fastcompany.com/3037915/the-problem-with-sentiment-analysis [Accessed: 12 June 2017].

Kotlerman, L., Dagan, I., Gorodetsky, M., Daya, E., 2012. Sentence Clustering via Projection over Term Clusters. First Joint Conference on Lexical and Computational Semantics (*SEM), 38–43.

Landauer, T., Foltz, P., Laham, D., 1998. An Introduction to Latent Semantic Analysis. Discourse Processes 25, 259-284.

Lee, R., 2012. Look at Mother Nature on the Run in the 21st Century: Responsibility, Research and Innovation. Transnational Environmental Law 1, 105-117.

Levenberg, A., Pulman, S., Moilanen, K., Simpson, E., Roberts, S., 2014. Predicting Economic Indicators from Web Text Using Sentiment Composition. Oxford University, 1-10.

Li, M., Gao, Y., Wen, H., Du, Y., Liu, H., Wang, H., 2017. Joint RNN Model for Argument Component Boundary Detection. 2017 IEEE International Conference on Systems, Man, and Cybernetics (SMC), 1-6.

Louet, S., 2016. *RRI as a Wake-Up Call*, Euroscientist, viewed 19 March 2017, <http://www.euroscientist.com/towards-responsible-research-innovation/>.

Loughran, T., McDonald, B., 2011. When Is a Liability Not a Liability? Textual Analysis, Dictionaries, and 10-Ks. The Journal of Finance 66(1):35–65.

Loughran, T., McDonald, B., 2016, Textual Analysis in Accounting and Finance: A Survey. Journal of Accounting Research, 54(4), 1187-1230.

Lui, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., Stoyanov, V., 2019. RoBERTa: A Robustly Optimized BERT Pretraining Approach. arXiv preprint arXiv:1907. 11692v1 [cs.CL], 1-13.

Lusetti, M., Ruzsics, T., Göhring, A., Samardžić, T., Stark, E., 2018. Encoder-decoder methods for text normalization. In: Proceedings of the Fifth Workshop on NLP for Similar Languages, Varieties and Dialects, 18-28.

MacBride, E., 2014. How to read a 10-K like Warren Buffett. *CNBC*, [Online] 27 Jan. Available at: https://www.cnbc.com/amp/2014/01/27/how-to-read-a-10-k-like-warren-buffet.html [Accessed: 7 April 2019].

Malo, P., Sinha, A., Takala, P., Korhonen, P. and Wallenius, J., 2013. Good debt or bad debt: Detecting semantic orientations in economic texts. Journal of the American Society for Information Science and Technology (in Press), 1-15.

Manning, C., Raghavan, P., Schütze, H., 2009. An Introduction to Information Retrieval. Cambridge, England: Cambridge University Press.

Mäntylä, M., Graziotin, D., Kuutila, M., 2018. The Evolution of Sentiment Analysis - A Review of Research Topics, Venues, and Top Cited Papers. Computer Science Review 27, 16-32.

McBride, N., Stahl, B., 2014. Developing Responsible Research and Innovation for Robotics. IEEE, International Symposium on Ethics in Science, Technology and Engineering, 1-10.

McDonald, B., 2019a. Software Repository for Accounting and Finance [Online]. Available at: http://sraf.nd.edu/ [Accessed: 19 February 2019].

McDonald, B., 2019b. Personal Communication. [email]

Metts, G., 2008. Alpha Investing: Exploring Investment Portfolio Heuristics Using Real Data Introduction. MBAA International Conference, Chicago, 1-12.

Mikolov, T., Yih, W., Zweig, G., 2013. Linguistic Regularities in Continuous Space Word Representations. Proceedings of NAACL-HLT, Association for Computational Linguistics, 746–751.

Mittelstadt, B., Allo, P., Taddeo, M., Wachter, S., Floridi, L., 2016. The Ethics of Algorithms: Mapping the Debate. Big Data and Society July-December, 1-21.

Moor, J., 1985. What is computer ethics? Metaphilosophy 16(4), 266–275.

Munikar, M., Shakya, S., Shrestha, A., 2019. Fine-grained Sentiment Classification using BERT. arXiv:1910.03474v1 [cs.CL], 1-4.

Naderi N., Hirst G., 2016. Argumentation Mining in Parliamentary Discourse. In: Baldoni M. et al. (eds) Principles and Practice of Multi-Agent Systems. IWEC 2014, CMNA 2015, IWEC 2015. Lecture Notes in Computer Science, vol 9935, 16-25.

Neto, J., Freitas, A., Kaestner, C., 2002. Automatic Text Summarization Using a Machine Learning Approach. Advances in Artificial Intelligence, 16th Brazilian Symposium on Artificial Intelligence, SBIA 2002, Porto de Galinhas/Recife, Brazil, Proceedings, 205-215.

Networked Quantum Information Technologies (n.d.), viewed 20 March 2017, <http://www.nqit.ox.ac.uk/content/about>.

Noothigattu, R., Gaikwad, S., Awad, E., Dsouza, S., Rahwan, I., Ravikumar, P., Procaccia, A., 2017. A Voting-Based System for Ethical Decision Making. arXiv:1709.06692v1 [cs.AI], 1-13.

Obama, B., 2016. Interview by S. Dadich and J. Ito for MIT and WIRED, 24 August, Available at: https://www.wired.com/2016/10/president-obama-mit-joi-ito-interview/ [Accessed: 3 July 2017].

O'Hare, N., Davy, M., Bermingham, A., Ferguson, P., Sheridan, P., Gurrin, C., Smeaton, A., 2009. Topic-Dependent Sentiment Analysis of Financial Blogs. Proceedings of the 1st international cikm workshop on topic-sentiment analysis for mass opinion, 9-16.

Owen, R., Baxter, D., Maynard, T., Depledge, M., 2009. Beyond Regulation: Risk Pricing and Responsible Innovation. Environmental Science and Technology 43, 5171-5175.

Owen, R., Goldberg, N., 2010. Responsible Innovation: A Pilot Study with the U.K. Engineering and Physical Sciences Research Council. Risk Analysis 30, 1699-1707.

Owen, R., Macnaghten, P., Stilgoe, J., 2012. Responsible research and innovation: From science in society to science for society, with society. Science and Public Policy 39, 751-760.

Owen, R., Stilgoe, J., Macnaghten, P., Gorman, M., Fisher, E., Guston, D., 2013. A Framework for Responsible Innovation. In: Owen, R., Bessant, J., Heintz, M. ed. *Responsible innovation: Managing the responsible emergence of science and innovation in society.* John Wiley, 27-50.

Owen, R., 2014. Responsible Research and Innovation: Options for Research and Innovation Policy in the EU. Exeter Business School, 1-33.

Palacio-Nino, J., Berzal, F., 2019. Evaluation Metrics for Unsupervised LearningAlgorithms. arXiv:1905.05667v2, 1-9.

Pang, B., Lee, L., 2008. Opinion Mining and Sentiment Analysis. Foundations and Trends in Information Retrieval 2(1-2), 1-135.

Partow-Navid, P., Ardekani, M., 1991. Measuring Readability of Management Information System Journals. Journal of Information Technology Management 2(1), 43-46.

Pearson, K., 1895. Contributions to the mathematical theory of evolution. II. Skew variation in homogeneousmaterial. Philosophical Transactions of the Royal Society of London, 343-414.

Pennington, J., Socher, R., Manning, C., 2014. GloVe: Global Vectors for Word Representation. Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), Association for Computational Linguistics, 1532–1543.

Pope, K., Vasquez, M., 2016. Ethics in psychotherapy and counseling: A practical guide. John Wiley & Sons.

Prentice, R., 2007. Ethical Decision Making: More Needed Than Good Intentions. Financial Analysts Journal 63(6), 17-30.

Quantopian, 2016, Alphalens, viewed 9 January 2020, <https://quantopian.github.io/alphalens/>.

Quantopian, 2020, Zipline, viewed 9 January 2020, <https://www.zipline.io/beginner-tutorial>.

Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., Liu, P., 2019. Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. arXiv:1910.10683v2 [cs.LG], 1-53.

Rajan, A., Victor, S., 2014. Web Sentiment Analysis: Comparison of Sentiments with Stock Prices Using Correlation and Regression. Journal of Computing Technologies (2278 – 3814) # 9 Volume 3 Issue 5, 9-14.

Ranco, G., Aleksovski, D., Caldarelli, G., Grčar, M., Mozetič, I., 2015. The Effects of Twitter Sentiment on Stock Price Returns. PLoS ONE 10(9): e0138441.doi:10.1371/journal.pone.0138441, 1-21.

Reddy, M., Vivekananda, M., Satish, R., 2017. Divisive Hierarchical Clustering with K-means andAgglomerative Hierarchical Clustering. International Journal of Computer Science Trends and Technology (IJCST) - Volume 5 Issue 5, 6-11.

Reimers, N., Gurevych, I., 2019. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing, Association for Computational Linguistics, 1-11.

Reimers, N., 2020. Personal Communication. [email]

Ren, J., Huizhong, G., Xiaoyu, W., Wang, G., Wang, W., Liao, S., 2013. Effective Sentiment Analysis Of Corporate Financial Reports. Thirty Fourth International Conference on Information Systems, 1-9.

Ribeiro, M., Singh, S., Guestrin, C., 2016. Why Should I Trust You?": Explaining the Predictions of Any Classifier. Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics, 1-10.

Rickford, J., King, S., 2016. Language and linguistics on trial: Hearing Rachel Jeantel (and other vernacular speakers) in the courtroom and beyond. Language 92(4), 948–988.

Rip, A., Schot, J., Misa, T., 1995. Managing Technology in Society: The Approach of Constructive Technology Assessment. Thomson.

Rip, A., Parandian, A., Kulve, H., 2012. Dual Dynamics of Promises, and Waiting Games Around Emerging Nanotechnologies. Technology Analysis & Strategic Management Vol. 24, No. 6, 565–58.

Roberts, S., 2020. Personal Communication. [email and video conference]

Rodriguez, M., Comin, C., Casanova, D., Bruno, O., Amancio, D., Costa, L., Rodrigues, F., 2019. Clustering algorithms: A comparative approach. PLoS ONE 14(1): e0210236, 1-34.

Root, A., 2019. How to Read a 10-K Annual Report and Suss Out a Stock in 10 Minutes. *Barron's*, [Online] 9 Mar. Available at: https://www.barrons.com/amp/articles/understanding-10-k-annual-report-51552084536 [Accessed: 7 April 2019].

Sarkar, K., 2009. Sentence Clustering-based Summarization of Multiple Text Documents. TECHNIA – International Journal of Computing Science and Communication Technologies, 2(1), 325-335.

Schnoebelen, T., 2017. Goal-Oriented Design for Ethical Machine Learning and NLP. Proceedings of the First Workshop on Ethics in Natural Language Processing, Association for Computational Linguistics, 88–93.

Schnoebelen, T., 2017b. Ethics in machine-learning, natural language processing, and AI, viewed on 22 June 2017, <https://medium.com/@TSchnoebelen/ethics-in-machine-learning-natural-language-processing-and-ai-609277a66c01>.

Schot, J., Rip, A., 1996. The Past and Future of Constructive Technology Assessment. Technological Forecasting and Social Change 54, 251-268.

Schroff, F., Kalenichenko, D., Philbin, J., 2015. FaceNet: A Unified Embedding for Face Recognition and Clustering. arXiv preprint arXiv:1503.03832v3, 1-10.

Schuurbiers, D., 2014. NANOCODE – Implementing the European Commission Code of Conduct for Responsible Nanotechnologies, viewed 19 June, 2017, <http://www.nanodiode.eu/publication/nanocode-implementing-european-commission-code-conduct-responsible-nanotechnologies/>.

Science with and for Society, 2014. Work Programme, European Commission, 1-52.

scipy.stats.pearsonr 2020, viewed 19 February 2020, <https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.pearsonr.html>.

Simpson, E. (supervisor), 2020. Personal Communication. [email and phone conference]

Skater developers and contributors (MIT License), 2017, Skater, viewed 7 March 2020, <https://oracle.github.io/Skater/overview.html>.

sklearn.cluster.AgglomerativeClustering 2020, viewed 2 February 2020, <https://scikit-learn.org/stable/modules/generated/sklearn.cluster.AgglomerativeClustering.html>.

sklearn.cluster.KMeans 2020, viewed 2 February 2020, <https://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html>.

Stahl, B., 2013. Responsible Research and Innovation: The Role of Privacy in an Emerging Framework. Science and Public Policy 40 (6), 708-716.

Stahl, B., Eden, G., Jirotka, M., Coeckelbergh, M., 2014. From computer ethics to responsible research and innovation in ICT: The transition of reference discourses informing ethics-related research in information systems. Information & Management 51(6), 810-818.

Stilgoe, J., Owen, R., Macnaghten, P., 2013. Developing a Framework for Responsible Innovation. Research Policy 42 (9), 1568-1580.

Stilgoe, J., 2017. *Machine Learning, Social Learning and Responsible Innovation in Self-driving Cars*. [Colloquium at UC Berkeley, Website Description]. Available at: http://cstms.berkeley.edu/current-events/machine-learning-social-learning-and-responsible-innovation-in-self-driving-cars/ [Accessed: 15 March 2017].

Stilgoe, J., 2017b. Machine learning, social learning and the governance of self-driving cars. Social Studies of Science, SAGE publishing, 1-32.

Stirling, A., 2007. A General Framework for Analysing Diversity in Science, Technology and Society. Journal of the Royal Society Interface 4, 707-719.

Stirling, A., 2008. "Opening Up" and "Closing Down": Power, Participation, and Pluralism in the Social Appraisal of Technology. Science, Technology & Human Values 33, 262-294.

Sun, F., Belatreche, A., Coleman, S., McGinnity, T.M., Li, Y., 2014. Pre-processing Online Financial Text for Sentiment Classification: A Natural Language Processing Approach. IEEE Computational Intelligence for Financial Engineering and Economics 2014, At London, UK, 1-9.

Sutcliffe, H., 2011. A report on Responsible Research and Innovation. MATTER, 1-34.

The Directorate-General for Research and Innovation of the European Commission, 2012. Responsible Research and Innovation – Europe's Ability to Respond to Societal Challenges. Available at: https://ec.europa.eu/research/swafs/pdf/pub_public_engagement/responsible-research-and-innovation-leaflet_en.pdf [Accessed: 30 June 2017].

The Royal Society, 2017, Machine learning requires careful stewardship says Royal Society, viewed on 28 April 2017, <https://royalsociety.org/news/2017/04/machine-learning-requires-careful-stewardship-says-royal-society/>.

The SPICE Project, 2017, viewed 20 June 2017, < http://www.spice.ac.uk/>.

Tucker, J., Zilinskas, R., 2006. The Promises and Perils of Synthetic Biology. The New Atlantis 12, 25-45.

UK National Quantum Technologies Programme 2013, viewed 3 July 2017, <http://uknqt.epsrc.ac.uk/about/>.

United States. 2002. Sarbanes-Oxley Act of 2002: conference report (to accompany H.R. 3763). [Washington, D.C.], [U.S. G.P.O.].

United States Federal Government, 2019. U.S. Securities and Exchange Commission [Online]. Available at: https://www.sec.gov/ [Accessed: 15 February 2019].

Van den Hoven, J., Jacob, K., Nielsen, L., Roure, F., Rudze, L., Stilgoe, J., Blind, K., Guske, A., Riera, C., 2013. Options for Strengthening Responsible Research and Innovation: Report of the Expert Group on the State of Art in Europe on Responsible Research and Innovation. Directorate-General for Research and Innovation Science in Society.

Van de Poel, I., Fahlquist, J., Doorn, N., Zwart, S., Royakkers, L., 2012. The Problem of Many Hands: Climate Change as an Example. Sci Eng Ethics 18(1), 49-67.

Van de Poel, I., Royakkers, L., Zwart, S., 2015. Moral Responsibility and the Problem of Many Hands. Routledge, 1-226.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A., Kaiser, Ł., Polosukhin, I., 2017. Attention is All You Need. 31st Conference on Neural Information Processing Systems (NIPS 2017), Long Beach, CA, USA, 1-15.

Von Schomberg, R., 2011. Prospects for Technology Assessment in a Framework of Responsible Research and Innovation. Technikfolgen abschätzen lehren: Bildungspotenziale transdisziplinärer Methode, 39-61.

Von Schomberg, R., 2011b. Towards Responsible Research and Innovation in the Information and Communication Technologies and Security Technologies Fields. Brussels: European Commission, <http://ec.europa.eu/research/science-society/document_library/pdf_06/mep-rapport2011_en.pdf>, accessed 02 July 2017.

Wallach, W., Allen C., 2008. Moral Machines: Teaching Robots Right from Wrong. USA: Oxford University Press.

Walter, C., 2016. The financial Logos: The framing of financial decision-making by mathematical modelling. Research in International Business and Finance 37, 597-604.

Wang, G., Xie, C., Stanley, H., 2018. Correlation Structure and Evolution of World Stock Markets: Evidence from Pearson and Partial Correlation-Based Networks. Computational Economics 51(3), 607-635.

Wiebe, J., Wilson, T., Cardie, C., 2005. Annotating expressions of opinions and emotions in language. Language Resources and Evaluation 39, 165–210.

Wikipedia Contributor, 2019a. Special relativity. [online] Wikipedia. Available at: https://en.wikipedia.org/wiki/Special_relativity [Accessed 2 Mar 2019].

Wikipedia Contributor, 2019b. Statistical mechanics. [online] Wikipedia. Available at: https://en.wikipedia.org/wiki/Statistical_mechanics [Accessed 2 Mar 2019].

Wikipedia Contributor, 2019c. Stochastic calculus. [online] Wikipedia. Available at: https://en.wikipedia.org/wiki/Stochastic_calculus [Accessed 2 Mar 2019].

Wikipedia Contributor, 2019d. Theoretical physics. [online] Wikipedia. Available at: https://en.wikipedia.org/wiki/Theoretical_physics [Accessed 2 Mar 2019].

Wilsdon, J., Willis, R., 2004. See-Through Science. Demos.

Xie, Q., Dai1, Z., Hovy, E., Luong, M., Le, Q., 2019. Unsupervised Data Augmentation For Consistency Training. arXiv:1904.12848v4 [cs.LG], 1-19.

Xiong, H., Pandey, G., Steinbach, M., Kumar, V., 2006. Enhancing data analysis with noise removal. IEEE Transactions on Knowledge and Data Engineering, 18(3), 304–319.

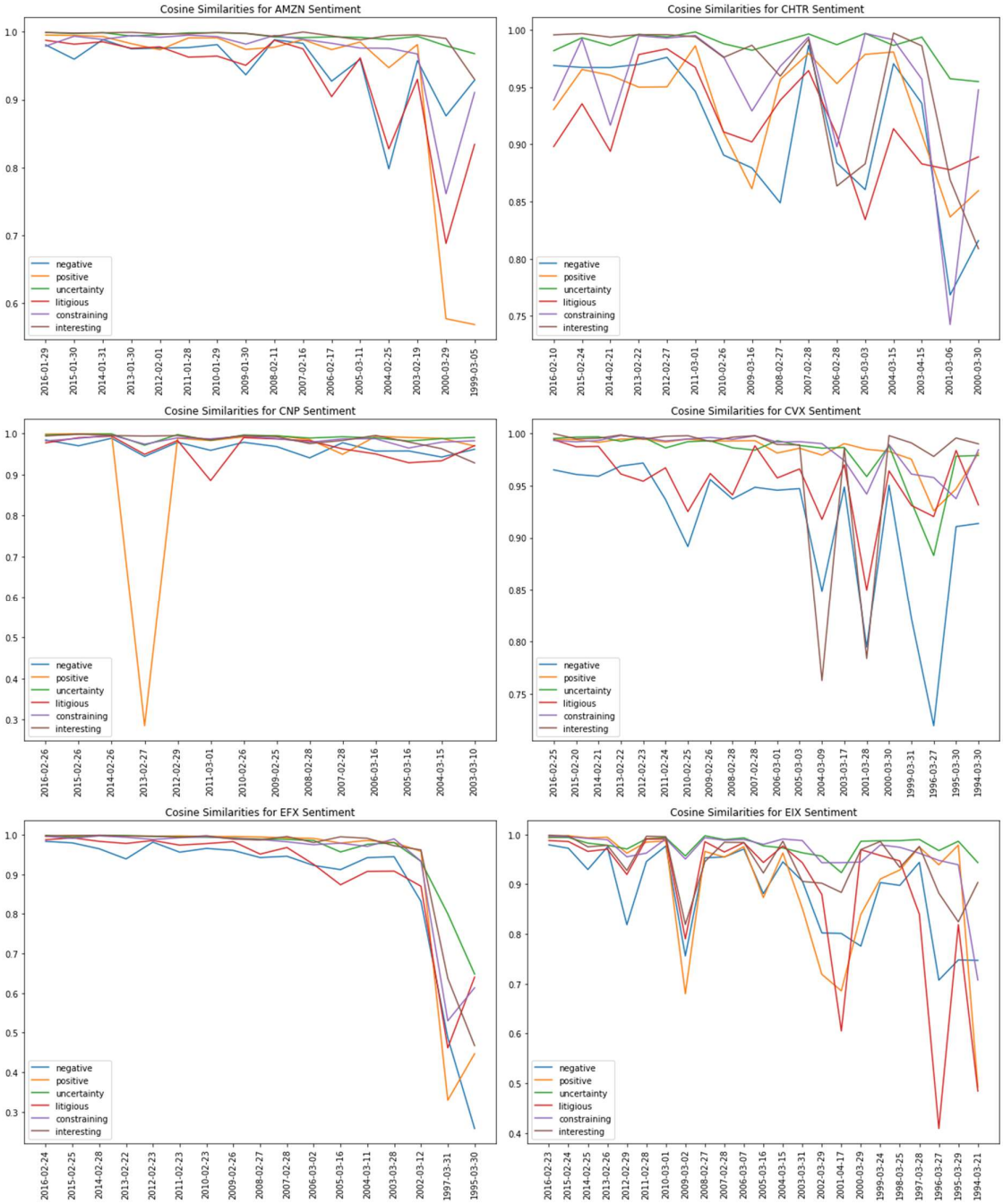
Xu, D., Tian., Y., 2015. A Comprehensive Survey of Clustering Algorithms. Annals of Data Science volume 2, 165–193.

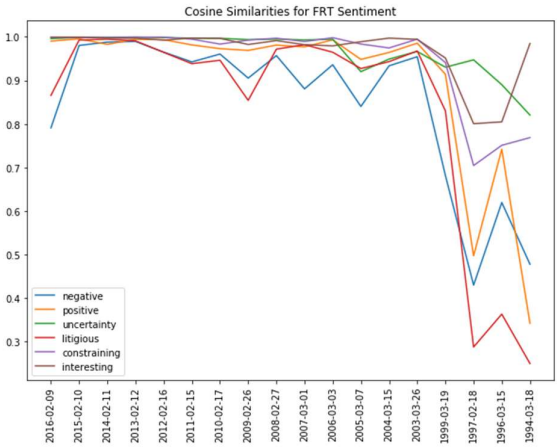
Zhang, M., Liang, P., Fu, G., 2019. Enhancing Opinion Role Labeling with Semantic-Aware Word Representations from Semantic Role Labeling. Proceedings of NAACL-HLT 2019, 641–646.

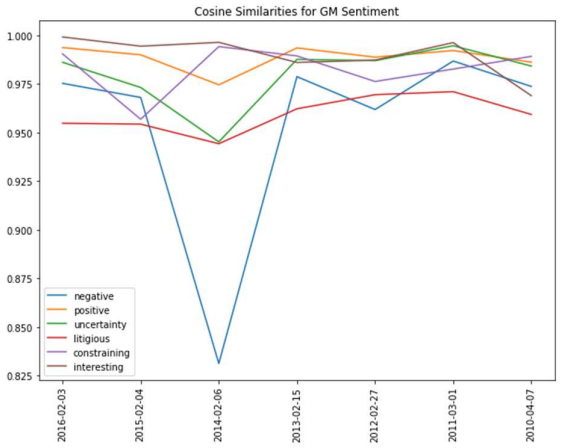
Zhou, S., Jeong, H., Green, P., 2017. How Consistent Are the Best-Known Readability Equations in Estimating the Readability of Design Standards? IEEE Transactions On Professional Communication 60(1), 97-111.

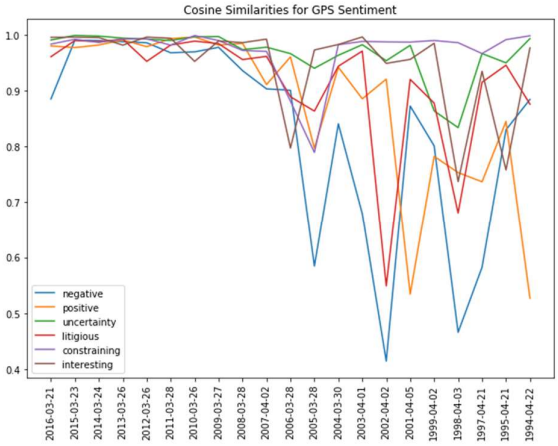
14 Appendices

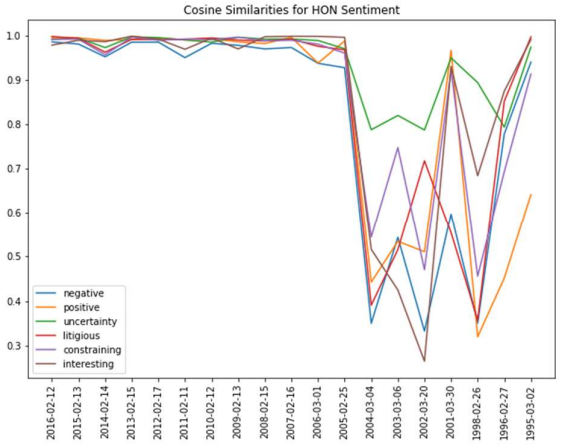
14.1 Appendix 1: Cosine similarity graphs for remaining 15 companies from Section 6

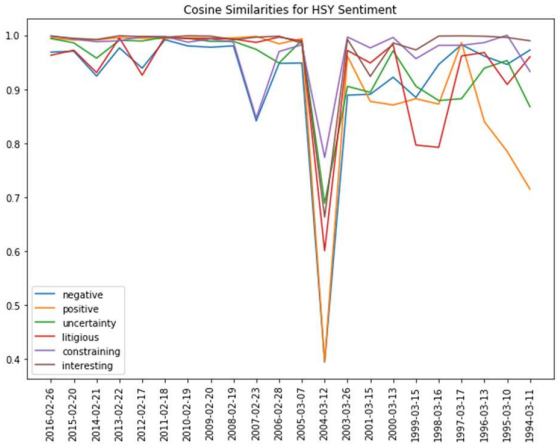


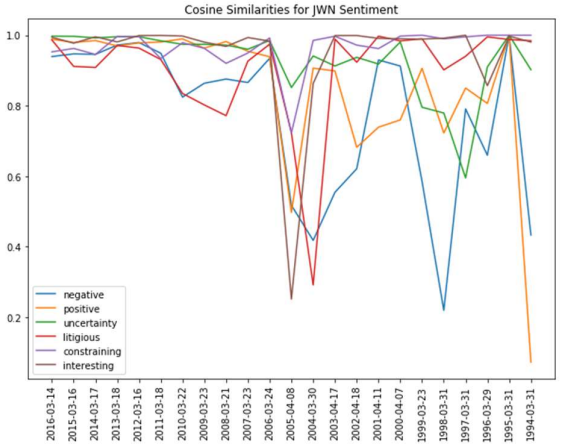

Cosine Similarities for FRT Sentiment
negative
positive
uncertainty
litigious
constraining
interesting

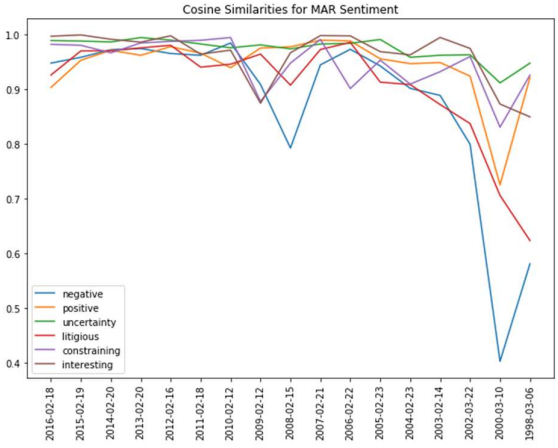

Cosine Similarities for GM Sentiment
negative
positive
uncertainty
litigious
constraining
interesting

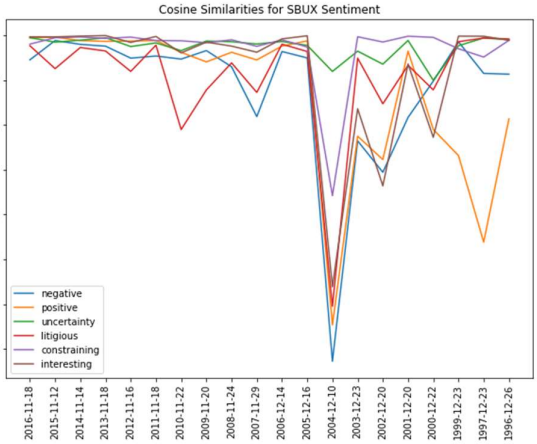

Cosine Similarities for GPS Sentiment
negative
positive
uncertainty
litigious
constraining
interesting

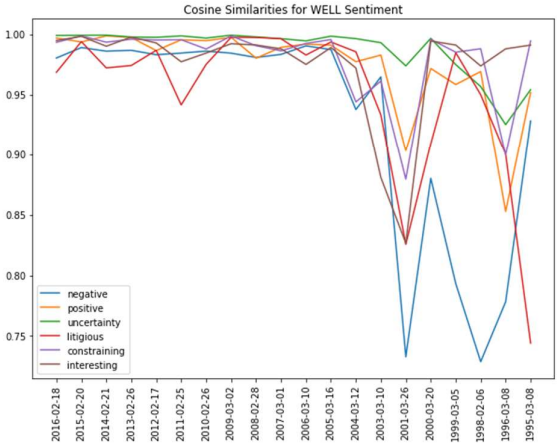

Cosine Similarities for HON Sentiment
negative
positive
uncertainty
litigious
constraining
interesting


Cosine Similarities for HSY Sentiment
negative
positive
uncertainty
litigious
constraining
interesting


Cosine Similarities for JWN Sentiment
negative
positive
uncertainty
litigious
constraining
interesting


Cosine Similarities for MAR Sentiment
negative
positive
uncertainty
litigious
constraining
interesting


Cosine Similarities for SBUX Sentiment
negative
positive
uncertainty
litigious
constraining
interesting


Cosine Similarities for WELL Sentiment
negative
positive
uncertainty
litigious
constraining
interesting

14.2 Appendix 2: List of outputs for the full Pearson correlation analysis between all possible time windows and predictions from sentiment analyses

In the following table for this appendix, the first column contains notation for the variables utilised in the sentiment analysis for that row. The following key explains what the various notations mean (these notations are used in all tables for Appendix 2 and plots in Appendices 3 and 4):

First value:

rf = random forest classifier model used

nb = Naïve Bayes classifier model used

svm = Support Vector Machines classifier model used

Second Value:

SBERT = transformer models/Sentence-BERT were used for feature engineering and numericisation of text data for input into machine learning models

bow = Bag of Words method used for numericising text data

Third Value:

bal = training class labels were balanced with the *class_weight* hyperparameter so the effects of discrepancies between the number of any given label would be minimised

unbal = training class labels were not balanced

Fourth Value:

wiki = the Wikipedia pre-trained model was utilised for encoding Sentence-BERT embeddings for both encoding text into semantically meaningful sentence embeddings during justification mining as well as numericisation of text for input into classifier models

nli = the NLI pre-trained model instead was utilised for encoding Sentence-BERT embeddings for both tasks as explained above

Fifth Value:

mined = the sentiment analysis was performed on the mined justifications

full = the sentiment analysis was performed on the full MD&A Overview subsections

Sixth Value:

mean = mean aggregation utilised

mode = mode aggregation utilised

meanr = rounded mean aggregation utilised

Seventh Value (only applicable when using mined justifications):

agglo = Agglomerative model used for clustering during justification mining

kmeans = K-means model used instead

dbscan = DBSCAN model used instead

meanshift = MeanShift model used instead

To walk through an example, one combination is *rf_SBERT_unbal_wiki_mined_mode_dbscan*. Here, rf means the random forests classifier was employed, SBERT refers to using transformer models and Sentence-BERT for numericisation of text, unbal is utilising unbalanced training class labels, wiki refers to employing the Wikipedia pre-trained model for Sentence-BERT, mined means sentiment analysis was performed on the mined justifications and not the full MD&A Overview documents, mode refers to using mode aggregation to obtain the final text sentiment score, and, finally, dbscan means utilising the DBSCAN clustering model during justification mining.

The second column in the table simply denotes which time window was used to calculate stock returns for that row (one week, one month, three months, six months, or one year). The table is sorted by this second column. The third column contains the calculated Pearson coefficient, *r*, for that row and the fourth and final column contains the calculated p-value for that row.

Sentiment Analysis Variables	**Returns Window**	**Pearson coefficient**	**p-values**
rf_bow_wiki_mined_mean_agglo	pct_change_1_year	r=-0.07600696946795044	p=0.2427682615449973
rf_bow_wiki_mined_meanr_agglo	pct_change_1_year	r=0.0009320324164758988	p=0.9885882483667537
rf_bow_wiki_mined_mode_agglo	pct_change_1_year	r=0.0009320324164758988	p=0.9885882483667537
rf_bow_wiki_mined_mean_dbscan	pct_change_1_year	r=-0.08886303275296445	p=0.17181371690078892
rf_bow_wiki_mined_meanr_dbscan	pct_change_1_year	r=-0.06753338940827433	p=0.2994776537482387
rf_bow_wiki_mined_mode_dbscan	pct_change_1_year	r=-0.06753338940827433	p=0.2994776537482387
rf_bow_wiki_mined_mean_kmeans	pct_change_1_year	r=0.008186913773564139	p=0.900017925799984
rf_bow_wiki_mined_mean_meanshift	pct_change_1_year	r=-0.02204850626355893	p=0.7350631983949105
rf_bow_wiki_full_mean	pct_change_1_year	r=-0.07934593374126601	p=0.22262847261717808
rf_bow_wiki_full_meanr	pct_change_1_year	r=nan	p=nan
rf_bow_wiki_full_mode	pct_change_1_year	r=nan	p=nan
rf_bow_nli_mined_mean_agglo	pct_change_1_year	r=0.011092117428894011	p=0.8648330605557074
rf_bow_nli_mined_meanr_agglo	pct_change_1_year	r=nan	p=nan
rf_bow_nli_mined_mode_agglo	pct_change_1_year	r=nan	p=nan
rf_bow_nli_mined_mean_dbscan	pct_change_1_year	r=0.05080131854576299	p=0.4353306877653214
rf_bow_nli_mined_meanr_dbscan	pct_change_1_year	r=0.05080131854576299	p=0.4353306877653214
rf_bow_nli_mined_mode_dbscan	pct_change_1_year	r=0.05080131854576299	p=0.4353306877653214
rf_bow_nli_mined_mean_kmeans	pct_change_1_year	r=-0.013610803782033212	p=0.8345407978800649
rf_bow_nli_mined_mean_meanshift	pct_change_1_year	r=0.07392188652340292	p=0.25597242047849605
rf_bow_nli_full_mean	pct_change_1_year	r=-0.07934593374126601	p=0.22262847261717808
rf_bow_nli_full_meanr	pct_change_1_year	r=nan	p=nan
rf_bow_nli_full_mode	pct_change_1_year	r=nan	p=nan
nb_bow_wiki_mined_mean_agglo	pct_change_1_year	r=0.011789112124575755	p=0.8564285519555465
nb_bow_wiki_mined_meanr_agglo	pct_change_1_year	r=0.025656935270870304	p=0.6937310286485724
nb_bow_wiki_mined_mode_agglo	pct_change_1_year	r=-0.00541721675682667	p=0.9337454983774129
nb_bow_wiki_mined_mean_dbscan	pct_change_1_year	r=-0.0035734664036005492	p=0.9562669802457634
nb_bow_wiki_mined_meanr_dbscan	pct_change_1_year	r=-0.002929034763659282	p=0.9641478470036268

nb_bow_wiki_mined_mode_dbscan	pct_change_1_year	r=-0.009452688455107926	p=0.8846599863640376
nb_bow_wiki_mined_mean_kmeans	pct_change_1_year	r=0.08508774908714314	p=0.190828402301848
nb_bow_wiki_mined_mean_meanshift	pct_change_1_year	r=0.016045188224450148	p=0.8054920099058598
nb_bow_wiki_full_mean	pct_change_1_year	r=-0.0521784458861401	p=0.42297082232315925
nb_bow_wiki_full_meanr	pct_change_1_year	r=nan	p=nan
nb_bow_wiki_full_mode	pct_change_1_year	r=-0.008397198257059677	p=0.8974637595745425
rf_SBERT_bal_wiki_mined_mean_agglo	pct_change_1_year	r=-0.031675725976347525	p=0.6268122045510127
rf_SBERT_bal_wiki_mined_meanr_agglo	pct_change_1_year	r=-0.08780726897690645	p=0.17698331810783072
rf_SBERT_bal_wiki_mined_mode_agglo	pct_change_1_year	r=-0.08734806888387621	p=0.179267405064399
rf_SBERT_bal_wiki_mined_mean_dbscan	pct_change_1_year	r=-0.02121393729925068	p=0.7447373836458876
rf_SBERT_bal_wiki_mined_meanr_dbscan	pct_change_1_year	r=-0.06184112909600614	p=0.3421502859572431
rf_SBERT_bal_wiki_mined_mode_dbscan	pct_change_1_year	r=-0.0907960940555633	p=0.16264076705674133
rf_SBERT_bal_wiki_mined_mean_kmeans	pct_change_1_year	r=-0.05576330620987211	p=0.39177005047987334
rf_SBERT_bal_wiki_mined_mean_meanshift	pct_change_1_year	r=-0.0625866349718515	p=0.33635156478545886
rf_SBERT_bal_wiki_full_mean	pct_change_1_year	r=-0.1254901031892129	p=0.05318396287394917
rf_SBERT_bal_wiki_full_meanr	pct_change_1_year	r=-0.03790638184229683	p=0.5606195284982675
rf_SBERT_bal_wiki_full_mode	pct_change_1_year	r=-0.05873325472686777	p=0.36700315565147784
rf_SBERT_unbal_wiki_mined_mean_agglo	pct_change_1_year	r=-0.05734407092931593	p=0.3784647216480792
rf_SBERT_unbal_wiki_mined_meanr_agglo	pct_change_1_year	r=-0.0842032056091736	p=0.19549803781885036
rf_SBERT_unbal_wiki_mined_mode_agglo	pct_change_1_year	r=-0.0900383708753639	p=0.16619164415347437
rf_SBERT_unbal_wiki_mined_mean_dbscan	pct_change_1_year	r=-0.0724449401090909	p=0.265620252302651
rf_SBERT_unbal_wiki_mined_meanr_dbscan	pct_change_1_year	r=-0.11669029249143188	p=0.07235533674871673
rf_SBERT_unbal_wiki_mined_mode_dbscan	pct_change_1_year	r=-0.12756625154343026	p=0.04933637974513266
rf_SBERT_unbal_wiki_mined_mean_kmeans	pct_change_1_year	r=-0.11469229504119575	p=0.07741215416642017
rf_SBERT_unbal_wiki_mined_mean_meanshift	pct_change_1_year	r=-0.026151666593740765	p=0.6881309223620093
rf_SBERT_unbal_wiki_full_mean	pct_change_1_year	r=-0.12746288509330575	p=0.04952229551400624
rf_SBERT_unbal_wiki_full_meanr	pct_change_1_year	r=-0.03458962557525876	p=0.5954380137338375
rf_SBERT_unbal_wiki_full_mode	pct_change_1_year	r=-0.04414865933984172	p=0.49787498701646554
rf_SBERT_bal_nli_mined_mean_agglo	pct_change_1_year	r=-0.20664250471102646	p=0.0013468156664763686
rf_SBERT_bal_nli_mined_meanr_agglo	pct_change_1_year	r=-0.05464906879309885	p=0.40131587506294114
rf_SBERT_bal_nli_mined_mode_agglo	pct_change_1_year	r=-0.0037039861085976502	p=0.954671324353819
rf_SBERT_bal_nli_mined_mean_dbscan	pct_change_1_year	r=-0.13339623373452777	p=0.0397535050629828
rf_SBERT_bal_nli_mined_meanr_dbscan	pct_change_1_year	r=-0.13339623373452777	p=0.0397535050629828
rf_SBERT_bal_nli_mined_mode_dbscan	pct_change_1_year	r=-0.13339623373452777	p=0.0397535050629828
rf_SBERT_bal_nli_mined_mean_kmeans	pct_change_1_year	r=-0.1409664258189341	p=0.02969443385555039
rf_SBERT_bal_nli_mined_mean_meanshift	pct_change_1_year	r=-0.15363108490311284	p=0.017705314445628294
rf_SBERT_bal_nli_full_mean	pct_change_1_year	r=-0.1906106530887367	p=0.00315486848458724
rf_SBERT_bal_nli_full_meanr	pct_change_1_year	r=-0.07751210057532607	p=0.233537444104156
rf_SBERT_bal_nli_full_mode	pct_change_1_year	r=-0.06998142997872367	p=0.28226025340375266
svm_SBERT_bal_wiki_mined_mean_agglo	pct_change_1_year	r=-0.07196466166364682	p=0.2688104964371353
svm_SBERT_bal_wiki_mined_meanr_agglo	pct_change_1_year	r=nan	p=nan
svm_SBERT_bal_wiki_mined_mode_agglo	pct_change_1_year	r=nan	p=nan
svm_SBERT_bal_wiki_mined_mean_dbscan	pct_change_1_year	r=nan	p=nan
svm_SBERT_bal_wiki_mined_meanr_dbscan	pct_change_1_year	r=nan	p=nan

svm_SBERT_bal_wiki_mined_mode_dbscan	pct_change_1_year	r=nan	p=nan
svm_SBERT_bal_wiki_mined_mean_kmeans	pct_change_1_year	r=-0.10421598409077303	p=0.10878658389506617
svm_SBERT_bal_wiki_mined_mean_meanshift	pct_change_1_year	r=-0.0038431366527353046	p=0.9529703519363079
svm_SBERT_bal_wiki_full_mean	pct_change_1_year	r=0.04077911514637855	p=0.5312769806973203
svm_SBERT_bal_wiki_full_meanr	pct_change_1_year	r=nan	p=nan
svm_SBERT_bal_wiki_full_mode	pct_change_1_year	r=nan	p=nan
svm_SBERT_bal_nli_mined_mean_agglo	pct_change_1_year	r=-0.0914860740711053	p=0.15945709787218515
svm_SBERT_bal_nli_mined_meanr_agglo	pct_change_1_year	r=-0.03510281897624208	p=0.5899869747523128
svm_SBERT_bal_nli_mined_mode_agglo	pct_change_1_year	r=-0.08544971690157038	p=0.18894122921191261
svm_SBERT_bal_nli_mined_mean_dbscan	pct_change_1_year	r=-0.11625178815337546	p=0.0734415051157519
svm_SBERT_bal_nli_mined_meanr_dbscan	pct_change_1_year	r=-0.11625178815337546	p=0.0734415051157519
svm_SBERT_bal_nli_mined_mode_dbscan	pct_change_1_year	r=-0.11625178815337546	p=0.0734415051157519
svm_SBERT_bal_nli_mined_mean_kmeans	pct_change_1_year	r=-0.04601917995455773	p=0.47982378116608115
svm_SBERT_bal_nli_mined_mean_meanshift	pct_change_1_year	r=-0.12304840257566141	p=0.05802440554536363
svm_SBERT_bal_nli_full_mean	pct_change_1_year	r=-0.15290713497332828	p=0.018254544605497046
svm_SBERT_bal_nli_full_meanr	pct_change_1_year	r=-0.04656857848485962	p=0.47459008330335
svm_SBERT_bal_nli_full_mode	pct_change_1_year	r=-0.15107342942741173	p=0.019712483787280955
rf_bow_wiki_mined_mean_agglo	pct_change_3_months	r=-0.11401808140817772	p=0.07918175201426096
rf_bow_wiki_mined_meanr_agglo	pct_change_3_months	r=0.038622463272393534	p=0.5532329282733038
rf_bow_wiki_mined_mode_agglo	pct_change_3_months	r=0.038622463272393534	p=0.5532329282733038
rf_bow_wiki_mined_mean_dbscan	pct_change_3_months	r=-0.0962907600418362	p=0.138573447896483
rf_bow_wiki_mined_meanr_dbscan	pct_change_3_months	r=-0.05991853366098432	p=0.3573958309350196
rf_bow_wiki_mined_mode_dbscan	pct_change_3_months	r=-0.05991853366098432	p=0.3573958309350196
rf_bow_wiki_mined_mean_kmeans	pct_change_3_months	r=0.011909750962672057	p=0.8549754768429642
rf_bow_wiki_mined_mean_meanshift	pct_change_3_months	r=0.030393022727053038	p=0.6408433809427421
rf_bow_wiki_full_mean	pct_change_3_months	r=-0.1022269894693187	p=0.1157416291744725
rf_bow_wiki_full_meanr	pct_change_3_months	r=nan	p=nan
rf_bow_wiki_full_mode	pct_change_3_months	r=nan	p=nan
rf_bow_nli_mined_mean_agglo	pct_change_3_months	r=0.03182682226089998	p=0.6251680689335266
rf_bow_nli_mined_meanr_agglo	pct_change_3_months	r=nan	p=nan
rf_bow_nli_mined_mode_agglo	pct_change_3_months	r=nan	p=nan
rf_bow_nli_mined_mean_dbscan	pct_change_3_months	r=0.053754889354894204	p=0.4090758496738617
rf_bow_nli_mined_meanr_dbscan	pct_change_3_months	r=0.053754889354894204	p=0.4090758496738617
rf_bow_nli_mined_mode_dbscan	pct_change_3_months	r=0.053754889354894204	p=0.4090758496738617
rf_bow_nli_mined_mean_kmeans	pct_change_3_months	r=0.07499352476001865	p=0.24912542295002937
rf_bow_nli_mined_mean_meanshift	pct_change_3_months	r=0.07958483415122096	p=0.22123445640101563
rf_bow_nli_full_mean	pct_change_3_months	r=-0.1022269894693187	p=0.1157416291744725
rf_bow_nli_full_meanr	pct_change_3_months	r=nan	p=nan
rf_bow_nli_full_mode	pct_change_3_months	r=nan	p=nan
nb_bow_wiki_mined_mean_agglo	pct_change_3_months	r=0.03141613166008991	p=0.6296412504783742
nb_bow_wiki_mined_meanr_agglo	pct_change_3_months	r=-0.005136629896694729	p=0.9371699265819279
nb_bow_wiki_mined_mode_agglo	pct_change_3_months	r=0.026024837248829175	p=0.6895649619030159
nb_bow_wiki_mined_mean_dbscan	pct_change_3_months	r=-0.006261603751223203	p=0.9234477594297874

nb_bow_wiki_mined_meanr_dbscan	pct_change_3_months	r=-0.023277969031081813	p=0.7208876251227725
nb_bow_wiki_mined_mode_dbscan	pct_change_3_months	r=0.015013742118404037	p=0.8177698696021117
nb_bow_wiki_mined_mean_kmeans	pct_change_3_months	r=0.049536259173966556	p=0.4468650249360281
nb_bow_wiki_mined_mean_meanshift	pct_change_3_months	r=-0.039957256439047434	p=0.539591944387032
nb_bow_wiki_full_mean	pct_change_3_months	r=-0.028942190876519412	p=0.6568685587361196
nb_bow_wiki_full_meanr	pct_change_3_months	r=nan	p=nan
nb_bow_wiki_full_mode	pct_change_3_months	r=-0.0568244140456046	p=0.3828078637577502
rf_SBERT_bal_wiki_mined_mean_agglo	pct_change_3_months	r=-0.11951724718564136	p=0.06566287180752722
rf_SBERT_bal_wiki_mined_meanr_agglo	pct_change_3_months	r=-0.05261026954011901	p=0.4191376214631534
rf_SBERT_bal_wiki_mined_mode_agglo	pct_change_3_months	r=-0.10337609604546101	p=0.11168231193251381
rf_SBERT_bal_wiki_mined_mean_dbscan	pct_change_3_months	r=-0.06260480722328286	p=0.3362110056492343
rf_SBERT_bal_wiki_mined_meanr_dbscan	pct_change_3_months	r=-0.058704345852658164	p=0.3672394581563559
rf_SBERT_bal_wiki_mined_mode_dbscan	pct_change_3_months	r=-0.06661411720118579	p=0.30611913545341674
rf_SBERT_bal_wiki_mined_mean_kmeans	pct_change_3_months	r=-0.14224912064206147	p=0.028226180098434608
rf_SBERT_bal_wiki_mined_mean_meanshift	pct_change_3_months	r=-0.06364262872568038	p=0.3282459943993046
rf_SBERT_bal_wiki_full_mean	pct_change_3_months	r=-0.16079402769397458	p=0.01300136313692643
rf_SBERT_bal_wiki_full_meanr	pct_change_3_months	r=-0.03218754321266146	p=0.6212504250315418
rf_SBERT_bal_wiki_full_mode	pct_change_3_months	r=-0.03629622082665845	p=0.5774000702021208
rf_SBERT_unbal_wiki_mined_mean_agglo	pct_change_3_months	r=-0.14728946234591553	p=0.023043210181581647
rf_SBERT_unbal_wiki_mined_meanr_agglo	pct_change_3_months	r=-0.0910334052408178	p=0.16154044660886385
rf_SBERT_unbal_wiki_mined_mode_agglo	pct_change_3_months	r=-0.11049947411346377	p=0.08895416121445779
rf_SBERT_unbal_wiki_mined_mean_dbscan	pct_change_3_months	r=-0.13570680820981137	p=0.03641626021573769
rf_SBERT_unbal_wiki_mined_meanr_dbscan	pct_change_3_months	r=-0.1466623612661396	p=0.023639702532141753
rf_SBERT_unbal_wiki_mined_mode_dbscan	pct_change_3_months	r=-0.13153725681557832	p=0.04262238760582463
rf_SBERT_unbal_wiki_mined_mean_kmeans	pct_change_3_months	r=-0.15855815276704024	p=0.014335358190428615
rf_SBERT_unbal_wiki_mined_mean_meanshift	pct_change_3_months	r=-0.07990871129437316	p=0.21935453197874272
rf_SBERT_unbal_wiki_full_mean	pct_change_3_months	r=-0.1659714393889948	p=0.010323071438399539
rf_SBERT_unbal_wiki_full_meanr	pct_change_3_months	r=-0.034652109163344075	p=0.5947731059885075
rf_SBERT_unbal_wiki_full_mode	pct_change_3_months	r=-0.03645621836614573	p=0.5757221620644168
rf_SBERT_bal_nli_mined_mean_agglo	pct_change_3_months	r=-0.18582181445221208	p=0.004018277427982355
rf_SBERT_bal_nli_mined_meanr_agglo	pct_change_3_months	r=-0.009577967787396502	p=0.8831421864320518
rf_SBERT_bal_nli_mined_mode_agglo	pct_change_3_months	r=-0.011605061051126273	p=0.8586463474333544
rf_SBERT_bal_nli_mined_mean_dbscan	pct_change_3_months	r=-0.13224798173170968	p=0.04150565426558646
rf_SBERT_bal_nli_mined_meanr_dbscan	pct_change_3_months	r=-0.13224798173170968	p=0.04150565426558646
rf_SBERT_bal_nli_mined_mode_dbscan	pct_change_3_months	r=-0.13224798173170968	p=0.04150565426558646
rf_SBERT_bal_nli_mined_mean_kmeans	pct_change_3_months	r=-0.1399542095930891	p=0.03089869431098952
rf_SBERT_bal_nli_mined_mean_meanshift	pct_change_3_months	r=-0.1974753785088498	p=0.0022085084108392516
rf_SBERT_bal_nli_full_mean	pct_change_3_months	r=-0.19721525703277817	p=0.0022390330031892645
rf_SBERT_bal_nli_full_meanr	pct_change_3_months	r=-0.04102658788546104	p=0.5287859413625225
rf_SBERT_bal_nli_full_mode	pct_change_3_months	r=-0.06843597736875094	p=0.293050326006188
svm_SBERT_bal_wiki_mined_mean_agglo	pct_change_3_months	r=-0.09762982107273531	p=0.1331436598367586
svm_SBERT_bal_wiki_mined_meanr_agglo	pct_change_3_months	r=nan	p=nan
svm_SBERT_bal_wiki_mined_mode_agglo	pct_change_3_months	r=nan	p=nan
svm_SBERT_bal_wiki_mined_mean_dbscan	pct_change_3_months	r=nan	p=nan

svm_SBERT_bal_wiki_mined_meanr_dbscan	pct_change_3_months	r=nan	p=nan
svm_SBERT_bal_wiki_mined_mode_dbscan	pct_change_3_months	r=nan	p=nan
svm_SBERT_bal_wiki_mined_mean_kmeans	pct_change_3_months	r=-0.059731058455240814	p=0.35890485274241235
svm_SBERT_bal_wiki_mined_mean_meanshift	pct_change_3_months	r=0.0012949877832876583	p=0.9841447434119038
svm_SBERT_bal_wiki_full_mean	pct_change_3_months	r=-0.010129626636262214	p=0.8764638031056289
svm_SBERT_bal_wiki_full_meanr	pct_change_3_months	r=nan	p=nan
svm_SBERT_bal_wiki_full_mode	pct_change_3_months	r=nan	p=nan
svm_SBERT_bal_nli_mined_mean_agglo	pct_change_3_months	r=-0.06003260194817348	p=0.3564796199216454
svm_SBERT_bal_nli_mined_meanr_agglo	pct_change_3_months	r=0.012741376159088049	p=0.8449722335860574
svm_SBERT_bal_nli_mined_mode_agglo	pct_change_3_months	r=-0.08178674384394083	p=0.20867803984859704
svm_SBERT_bal_nli_mined_mean_dbscan	pct_change_3_months	r=-0.12352542103846195	p=0.05705118381207981
svm_SBERT_bal_nli_mined_meanr_dbscan	pct_change_3_months	r=-0.12352542103846195	p=0.05705118381207981
svm_SBERT_bal_nli_mined_mode_dbscan	pct_change_3_months	r=-0.12352542103846195	p=0.05705118381207981
svm_SBERT_bal_nli_mined_mean_kmeans	pct_change_3_months	r=-0.04317811898015814	p=0.5073806279371417
svm_SBERT_bal_nli_mined_mean_meanshift	pct_change_3_months	r=-0.1666858416230928	p=0.009994717341568097
svm_SBERT_bal_nli_full_mean	pct_change_3_months	r=-0.1866136900330011	p=0.003862210479074706
svm_SBERT_bal_nli_full_meanr	pct_change_3_months	r=-0.03239066709408182	p=0.6190490559828681
svm_SBERT_bal_nli_full_mode	pct_change_3_months	r=-0.22021219752588173	p=0.000622814609893812
rf_bow_wiki_mined_mean_agglo	pct_change_6_months	r=-0.0904714887936309	p=0.16415492114709443
rf_bow_wiki_mined_meanr_agglo	pct_change_6_months	r=0.032456063379171694	p=0.6183410398886638
rf_bow_wiki_mined_mode_agglo	pct_change_6_months	r=0.032456063379171694	p=0.6183410398886638
rf_bow_wiki_mined_mean_dbscan	pct_change_6_months	r=-0.06538987215668986	p=0.3151134586171224
rf_bow_wiki_mined_meanr_dbscan	pct_change_6_months	r=-0.040496125437291856	p=0.5341327595083504
rf_bow_wiki_mined_mode_dbscan	pct_change_6_months	r=-0.040496125437291856	p=0.5341327595083504
rf_bow_wiki_mined_mean_kmeans	pct_change_6_months	r=-0.008217566844274129	p=0.8996455414865814
rf_bow_wiki_mined_mean_meanshift	pct_change_6_months	r=0.010667272971346307	p=0.8699634496929648
rf_bow_wiki_full_mean	pct_change_6_months	r=-0.07259325530255316	p=0.2646403272978077
rf_bow_wiki_full_meanr	pct_change_6_months	r=nan	p=nan
rf_bow_wiki_full_mode	pct_change_6_months	r=nan	p=nan
rf_bow_nli_mined_mean_agglo	pct_change_6_months	r=0.04455646021966282	p=0.49390919766903724
rf_bow_nli_mined_meanr_agglo	pct_change_6_months	r=nan	p=nan
rf_bow_nli_mined_mode_agglo	pct_change_6_months	r=nan	p=nan
rf_bow_nli_mined_mean_dbscan	pct_change_6_months	r=0.019558296444169045	p=0.7640468344222453
rf_bow_nli_mined_meanr_dbscan	pct_change_6_months	r=0.019558296444169045	p=0.7640468344222453
rf_bow_nli_mined_mode_dbscan	pct_change_6_months	r=0.019558296444169045	p=0.7640468344222453
rf_bow_nli_mined_mean_kmeans	pct_change_6_months	r=0.07225685954286719	p=0.26686647541562414
rf_bow_nli_mined_mean_meanshift	pct_change_6_months	r=0.02280011244597421	p=0.7263862545805152
rf_bow_nli_full_mean	pct_change_6_months	r=-0.07259325530255316	p=0.2646403272978077
rf_bow_nli_full_meanr	pct_change_6_months	r=nan	p=nan
rf_bow_nli_full_mode	pct_change_6_months	r=nan	p=nan
nb_bow_wiki_mined_mean_agglo	pct_change_6_months	r=-0.008136006497913666	p=0.9006364136185439
nb_bow_wiki_mined_meanr_agglo	pct_change_6_months	r=0.01166598298212854	p=0.8579121211567322
nb_bow_wiki_mined_mode_agglo	pct_change_6_months	r=0.019297615850320183	p=0.7671007842226766

nb_bow_wiki_mined_mean_dbscan	pct_change_6_months	r=-0.06598121534332997	p=0.3107476333346011
nb_bow_wiki_mined_meanr_dbscan	pct_change_6_months	r=-0.05992785948183034	p=0.3573208695021466
nb_bow_wiki_mined_mode_dbscan	pct_change_6_months	r=-0.051507685486494245	p=0.42896528193359357
nb_bow_wiki_mined_mean_kmeans	pct_change_6_months	r=0.009802861837705533	p=0.8804185944977567
nb_bow_wiki_mined_mean_meanshift	pct_change_6_months	r=-0.027600673105371162	p=0.6718273643812822
nb_bow_wiki_full_mean	pct_change_6_months	r=-0.05730939671694042	p=0.37875357664419745
nb_bow_wiki_full_meanr	pct_change_6_months	r=nan	p=nan
nb_bow_wiki_full_mode	pct_change_6_months	r=-0.03310441129862404	p=0.6113407952304725
rf_SBERT_bal_wiki_mined_mean_agglo	pct_change_6_months	r=-0.06912468580930171	p=0.28820836791383003
rf_SBERT_bal_wiki_mined_meanr_agglo	pct_change_6_months	r=-0.06051061996385508	p=0.35265613563756804
rf_SBERT_bal_wiki_mined_mode_agglo	pct_change_6_months	r=-0.07097328421771959	p=0.2754781060099966
rf_SBERT_bal_wiki_mined_mean_dbscan	pct_change_6_months	r=-0.07179087769622326	p=0.26997126889003936
rf_SBERT_bal_wiki_mined_meanr_dbscan	pct_change_6_months	r=-0.10378632323363488	p=0.11026048859124758
rf_SBERT_bal_wiki_mined_mode_dbscan	pct_change_6_months	r=-0.11258351583578666	p=0.08305582903787984
rf_SBERT_bal_wiki_mined_mean_kmeans	pct_change_6_months	r=-0.09889963771782327	p=0.12814724289202298
rf_SBERT_bal_wiki_mined_mean_meanshift	pct_change_6_months	r=-0.11160069864586457	p=0.0857969727865684
rf_SBERT_bal_wiki_full_mean	pct_change_6_months	r=-0.15994546656821837	p=0.013494217768917304
rf_SBERT_bal_wiki_full_meanr	pct_change_6_months	r=-0.0307632826728055	p=0.6367797971062162
rf_SBERT_bal_wiki_full_mode	pct_change_6_months	r=-0.06970643649937094	p=0.2841603682295479
rf_SBERT_unbal_wiki_mined_mean_agglo	pct_change_6_months	r=-0.12616882196939483	p=0.051899628927932324
rf_SBERT_unbal_wiki_mined_meanr_agglo	pct_change_6_months	r=-0.1001260305454964	p=0.12346040134976748
rf_SBERT_unbal_wiki_mined_mode_agglo	pct_change_6_months	r=-0.09647463923529172	p=0.13781794875914474
rf_SBERT_unbal_wiki_mined_mean_dbscan	pct_change_6_months	r=-0.1376702159498808	p=0.03376986153133213
rf_SBERT_unbal_wiki_mined_meanr_dbscan	pct_change_6_months	r=-0.16979885392646893	p=0.008669336428914473
rf_SBERT_unbal_wiki_mined_mode_dbscan	pct_change_6_months	r=-0.16264182705261565	p=0.011982536633425108
rf_SBERT_unbal_wiki_mined_mean_kmeans	pct_change_6_months	r=-0.1602572503737491	p=0.013311255737247223
rf_SBERT_unbal_wiki_mined_mean_meanshift	pct_change_6_months	r=-0.09286132276254815	p=0.15325156950248617
rf_SBERT_unbal_wiki_full_mean	pct_change_6_months	r=-0.1739187697862103	p=0.007156141018647447
rf_SBERT_unbal_wiki_full_meanr	pct_change_6_months	r=-0.044425686216651515	p=0.4951791186799095
rf_SBERT_unbal_wiki_full_mode	pct_change_6_months	r=-0.06125127853631137	p=0.34678302758804086
rf_SBERT_bal_nli_mined_mean_agglo	pct_change_6_months	r=-0.1845129498324647	p=0.004288743206778107
rf_SBERT_bal_nli_mined_meanr_agglo	pct_change_6_months	r=-0.016061674945834423	p=0.8052961373440541
rf_SBERT_bal_nli_mined_mode_agglo	pct_change_6_months	r=0.034076554342134004	p=0.6009104635728054
rf_SBERT_bal_nli_mined_mean_dbscan	pct_change_6_months	r=-0.10944347526982184	p=0.09206838831324461
rf_SBERT_bal_nli_mined_meanr_dbscan	pct_change_6_months	r=-0.10944347526982184	p=0.09206838831324461
rf_SBERT_bal_nli_mined_mode_dbscan	pct_change_6_months	r=-0.10944347526982184	p=0.09206838831324461
rf_SBERT_bal_nli_mined_mean_kmeans	pct_change_6_months	r=-0.14804608507422795	p=0.022340846615377226
rf_SBERT_bal_nli_mined_mean_meanshift	pct_change_6_months	r=-0.15565420834067262	p=0.01624606605461402
rf_SBERT_bal_nli_full_mean	pct_change_6_months	r=-0.1900058803398099	p=0.003253746961023152
rf_SBERT_bal_nli_full_meanr	pct_change_6_months	r=-0.05501607243316659	p=0.39815648821313354
rf_SBERT_bal_nli_full_mode	pct_change_6_months	r=-0.06812527107597605	p=0.29525240089029725
svm_SBERT_bal_wiki_mined_mean_agglo	pct_change_6_months	r=-0.07843718364384919	p=0.22798820736828995
svm_SBERT_bal_wiki_mined_meanr_agglo	pct_change_6_months	r=nan	p=nan
svm_SBERT_bal_wiki_mined_mode_agglo	pct_change_6_months	r=nan	p=nan

svm_SBERT_bal_wiki_mined_mean_dbscan	pct_change_6_months	r=nan	p=nan
svm_SBERT_bal_wiki_mined_meanr_dbscan	pct_change_6_months	r=nan	p=nan
svm_SBERT_bal_wiki_mined_mode_dbscan	pct_change_6_months	r=nan	p=nan
svm_SBERT_bal_wiki_mined_mean_kmeans	pct_change_6_months	r=-0.08426257946113086	p=0.1951820119380013
svm_SBERT_bal_wiki_mined_mean_meanshift	pct_change_6_months	r=-0.026653446230408707	p=0.6824683379863
svm_SBERT_bal_wiki_full_mean	pct_change_6_months	r=0.01940822832251895	p=0.7658044845220797
svm_SBERT_bal_wiki_full_meanr	pct_change_6_months	r=nan	p=nan
svm_SBERT_bal_wiki_full_mode	pct_change_6_months	r=nan	p=nan
svm_SBERT_bal_nli_mined_mean_agglo	pct_change_6_months	r=-0.09502091491721745	p=0.14387763123333
svm_SBERT_bal_nli_mined_meanr_agglo	pct_change_6_months	r=-0.03674863940498248	p=0.572661443735154
svm_SBERT_bal_nli_mined_mode_agglo	pct_change_6_months	r=-0.03378956290424441	p=0.6039813436332142
svm_SBERT_bal_nli_mined_mean_dbscan	pct_change_6_months	r=-0.10446423970906613	p=0.10794205560298921
svm_SBERT_bal_nli_mined_meanr_dbscan	pct_change_6_months	r=-0.10446423970906613	p=0.10794205560298921
svm_SBERT_bal_nli_mined_mode_dbscan	pct_change_6_months	r=-0.10446423970906613	p=0.10794205560298921
svm_SBERT_bal_nli_mined_mean_kmeans	pct_change_6_months	r=-0.03496702255183068	p=0.5914271536709597
svm_SBERT_bal_nli_mined_mean_meanshift	pct_change_6_months	r=-0.12290521887527385	p=0.05831919232842032
svm_SBERT_bal_nli_full_mean	pct_change_6_months	r=-0.16603160290229121	p=0.010295055755102856
svm_SBERT_bal_nli_full_meanr	pct_change_6_months	r=-0.0360212311760656	p=0.5802892445139252
svm_SBERT_bal_nli_full_mode	pct_change_6_months	r=-0.11249399265727715	p=0.08330255983452281
rf_bow_wiki_mined_mean_agglo	pct_change_1_month	r=-0.009766876782611122	p=0.880854299924937
rf_bow_wiki_mined_meanr_agglo	pct_change_1_month	r=0.010982808027654074	p=0.8661525406379024
rf_bow_wiki_mined_mode_agglo	pct_change_1_month	r=0.010982808027654074	p=0.8661525406379024
rf_bow_wiki_mined_mean_dbscan	pct_change_1_month	r=-0.06410615953156983	p=0.3247281321833619
rf_bow_wiki_mined_meanr_dbscan	pct_change_1_month	r=-0.053134005222856835	p=0.414515891989937
rf_bow_wiki_mined_mode_dbscan	pct_change_1_month	r=-0.053134005222856835	p=0.414515891989937
rf_bow_wiki_mined_mean_kmeans	pct_change_1_month	r=0.0958154290259193	p=0.1405411145531766
rf_bow_wiki_mined_mean_meanshift	pct_change_1_month	r=-0.01425237161532108	p=0.8268616360378129
rf_bow_wiki_full_mean	pct_change_1_month	r=0.014856404963204726	p=0.8196467296605773
rf_bow_wiki_full_meanr	pct_change_1_month	r=nan	p=nan
rf_bow_wiki_full_mode	pct_change_1_month	r=nan	p=nan
rf_bow_nli_mined_mean_agglo	pct_change_1_month	r=0.04584981625658282	p=0.4814434634708575
rf_bow_nli_mined_meanr_agglo	pct_change_1_month	r=nan	p=nan
rf_bow_nli_mined_mode_agglo	pct_change_1_month	r=nan	p=nan
rf_bow_nli_mined_mean_dbscan	pct_change_1_month	r=0.08776773669747549	p=0.1771791022684861
rf_bow_nli_mined_meanr_dbscan	pct_change_1_month	r=0.08776773669747549	p=0.1771791022684861
rf_bow_nli_mined_mode_dbscan	pct_change_1_month	r=0.08776773669747549	p=0.1771791022684861
rf_bow_nli_mined_mean_kmeans	pct_change_1_month	r=0.039297091193079184	p=0.5463175798268949
rf_bow_nli_mined_mean_meanshift	pct_change_1_month	r=0.10998248036421697	p=0.09046811578075303
rf_bow_nli_full_mean	pct_change_1_month	r=0.014856404963204726	p=0.8196467296605773
rf_bow_nli_full_meanr	pct_change_1_month	r=nan	p=nan
rf_bow_nli_full_mode	pct_change_1_month	r=nan	p=nan
nb_bow_wiki_mined_mean_agglo	pct_change_1_month	r=0.03984948392924346	p=0.5406870900525557
nb_bow_wiki_mined_meanr_agglo	pct_change_1_month	r=0.0930416012400037	p=0.15245181962563709

nb_bow_wiki_mined_mode_agglo	pct_change_1_month	r=0.03297639864162528	p=0.612720188866888
nb_bow_wiki_mined_mean_dbscan	pct_change_1_month	r=-0.060102665373160766	p=0.3559175914030431
nb_bow_wiki_mined_meanr_dbscan	pct_change_1_month	r=-0.04273188051549496	p=0.5117827313307247
nb_bow_wiki_mined_mode_dbscan	pct_change_1_month	r=-0.06556869225449499	p=0.3137890447913355
nb_bow_wiki_mined_mean_kmeans	pct_change_1_month	r=0.09737686992574579	p=0.13415662159126512
nb_bow_wiki_mined_mean_meanshift	pct_change_1_month	r=-0.07556143083984208	p=0.2455489481176061
nb_bow_wiki_full_mean	pct_change_1_month	r=-0.0018352506812235403	p=0.977531485732332
nb_bow_wiki_full_meanr	pct_change_1_month	r=nan	p=nan
nb_bow_wiki_full_mode	pct_change_1_month	r=-0.0547932819918432	p=0.40007262194015486
rf_SBERT_bal_wiki_mined_mean_agglo	pct_change_1_month	r=-0.12727303404038254	p=0.049865284402722726
rf_SBERT_bal_wiki_mined_meanr_agglo	pct_change_1_month	r=-0.12642188900263512	p=0.05142739842163143
rf_SBERT_bal_wiki_mined_mode_agglo	pct_change_1_month	r=-0.11385184469343418	p=0.07962305732344699
rf_SBERT_bal_wiki_mined_mean_dbscan	pct_change_1_month	r=-0.01123074624873282	p=0.863160202978911
rf_SBERT_bal_wiki_mined_meanr_dbscan	pct_change_1_month	r=-0.027633234647864297	p=0.6714627181549059
rf_SBERT_bal_wiki_mined_mode_dbscan	pct_change_1_month	r=-0.07586631959604026	p=0.24364369616225967
rf_SBERT_bal_wiki_mined_mean_kmeans	pct_change_1_month	r=-0.18714263492173033	p=0.0037610334535998474
rf_SBERT_bal_wiki_mined_mean_meanshift	pct_change_1_month	r=-0.019394759592462524	p=0.7659622937275034
rf_SBERT_bal_wiki_full_mean	pct_change_1_month	r=-0.1652693073456413	p=0.010655044421736395
rf_SBERT_bal_wiki_full_meanr	pct_change_1_month	r=-0.09699941779877572	p=0.13567915460501817
rf_SBERT_bal_wiki_full_mode	pct_change_1_month	r=-0.054786235592793074	p=0.4001333149993914
rf_SBERT_unbal_wiki_mined_mean_agglo	pct_change_1_month	r=-0.14441891695220363	p=0.025883908052659013
rf_SBERT_unbal_wiki_mined_meanr_agglo	pct_change_1_month	r=-0.11162152190898632	p=0.08573815123311292
rf_SBERT_unbal_wiki_mined_mode_agglo	pct_change_1_month	r=-0.10221717701183045	p=0.11577678221528008
rf_SBERT_unbal_wiki_mined_mean_dbscan	pct_change_1_month	r=-0.026089914457095593	p=0.6888290053109921
rf_SBERT_unbal_wiki_mined_meanr_dbscan	pct_change_1_month	r=-0.05572117768034809	p=0.39212846217611674
rf_SBERT_unbal_wiki_mined_mode_dbscan	pct_change_1_month	r=-0.0699515592922059	p=0.2824662346330154
rf_SBERT_unbal_wiki_mined_mean_kmeans	pct_change_1_month	r=-0.15004771909482076	p=0.02057126652608078
rf_SBERT_unbal_wiki_mined_mean_meanshift	pct_change_1_month	r=-0.008771223707935136	p=0.892923380233703
rf_SBERT_unbal_wiki_full_mean	pct_change_1_month	r=-0.1564469869924304	p=0.015703551284572285
rf_SBERT_unbal_wiki_full_meanr	pct_change_1_month	r=-0.0427072380494989	p=0.5120264028204247
rf_SBERT_unbal_wiki_full_mode	pct_change_1_month	r=-0.032584293976195514	p=0.6169537671211683
rf_SBERT_bal_nli_mined_mean_agglo	pct_change_1_month	r=-0.14062787682406375	p=0.030092672416217044
rf_SBERT_bal_nli_mined_meanr_agglo	pct_change_1_month	r=0.009877205156628488	p=0.8795185629943062
rf_SBERT_bal_nli_mined_mode_agglo	pct_change_1_month	r=0.025685003917087498	p=0.6934128572830884
rf_SBERT_bal_nli_mined_mean_dbscan	pct_change_1_month	r=-0.17421202335832384	p=0.00705800649762465
rf_SBERT_bal_nli_mined_meanr_dbscan	pct_change_1_month	r=-0.17421202335832384	p=0.00705800649762465
rf_SBERT_bal_nli_mined_mode_dbscan	pct_change_1_month	r=-0.17421202335832384	p=0.00705800649762465
rf_SBERT_bal_nli_mined_mean_kmeans	pct_change_1_month	r=-0.13922805502043445	p=0.03178806774464829
rf_SBERT_bal_nli_mined_mean_meanshift	pct_change_1_month	r=-0.24914139302513652	p=0.0001024291066417152
rf_SBERT_bal_nli_full_mean	pct_change_1_month	r=-0.20610191447918774	p=0.0013874885171142937
rf_SBERT_bal_nli_full_meanr	pct_change_1_month	r=-0.05089811647976649	p=0.43445520726499065
rf_SBERT_bal_nli_full_mode	pct_change_1_month	r=-0.0596711630738507	p=0.3593877991250642
svm_SBERT_bal_wiki_mined_mean_agglo	pct_change_1_month	r=-0.07832604809192664	p=0.22864989162048036
svm_SBERT_bal_wiki_mined_meanr_agglo	pct_change_1_month	r=nan	p=nan

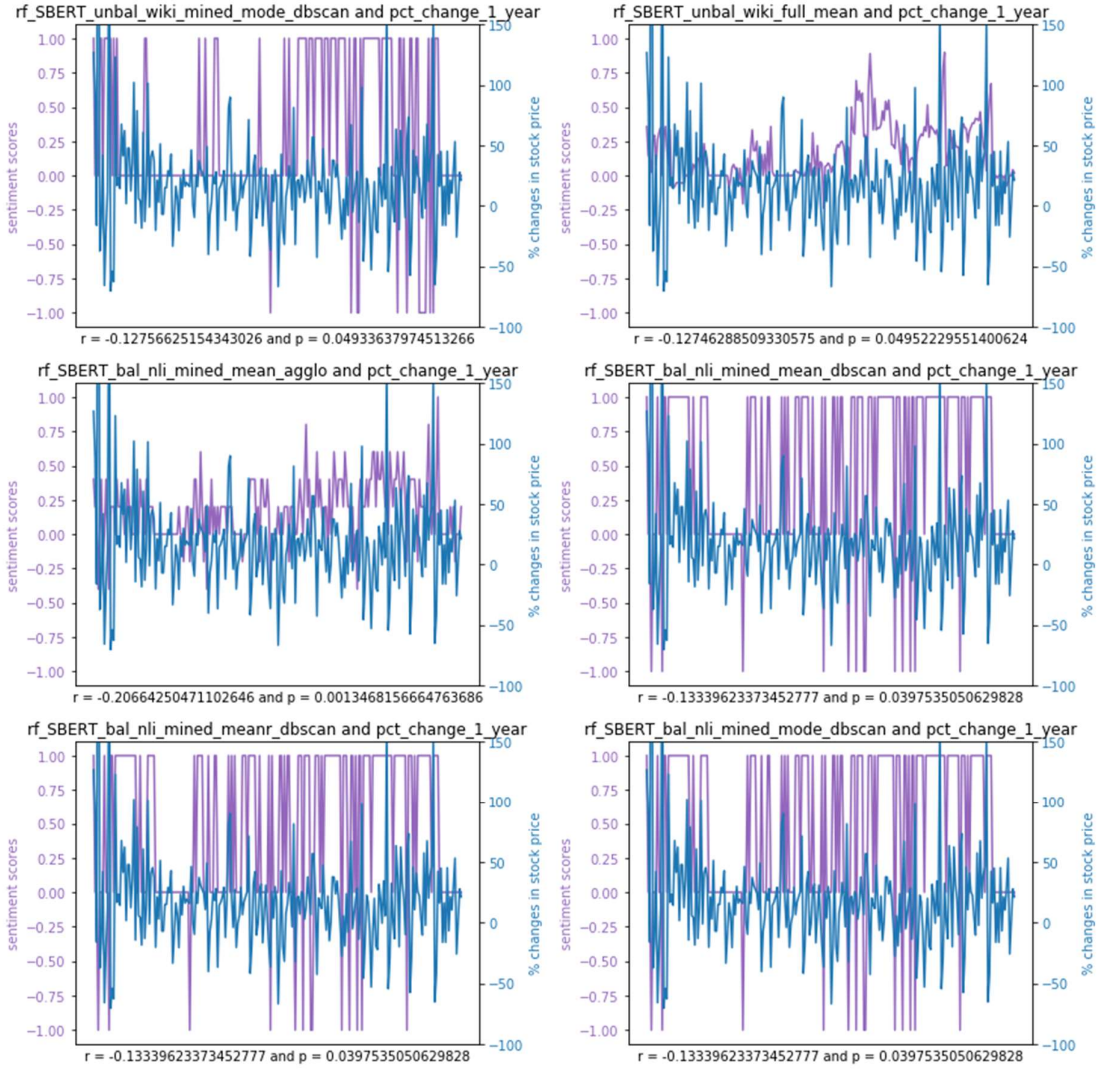
svm_SBERT_bal_wiki_mined_mode_agglo	pct_change_1_month	r=nan	p=nan
svm_SBERT_bal_wiki_mined_mean_dbscan	pct_change_1_month	r=nan	p=nan
svm_SBERT_bal_wiki_mined_meanr_dbscan	pct_change_1_month	r=nan	p=nan
svm_SBERT_bal_wiki_mined_mode_dbscan	pct_change_1_month	r=nan	p=nan
svm_SBERT_bal_wiki_mined_mean_kmeans	pct_change_1_month	r=-0.04146965897034969	p=0.5243408306504397
svm_SBERT_bal_wiki_mined_mean_meanshift	pct_change_1_month	r=-0.0004660364759925852 7	p=0.9942937310440597
svm_SBERT_bal_wiki_full_mean	pct_change_1_month	r=0.027146347422936416	p=0.676923184243841
svm_SBERT_bal_wiki_full_meanr	pct_change_1_month	r=nan	p=nan
svm_SBERT_bal_wiki_full_mode	pct_change_1_month	r=nan	p=nan
svm_SBERT_bal_nli_mined_mean_agglo	pct_change_1_month	r=-0.11797458252232126	p=0.0692491675322709
svm_SBERT_bal_nli_mined_meanr_agglo	pct_change_1_month	r=-0.031244795204114894	p=0.6315114321382295
svm_SBERT_bal_nli_mined_mode_agglo	pct_change_1_month	r=-0.07814073837506048	p=0.22975621694273726
svm_SBERT_bal_nli_mined_mean_dbscan	pct_change_1_month	r=-0.21747774701843273	p=0.0007303095449552845
svm_SBERT_bal_nli_mined_meanr_dbscan	pct_change_1_month	r=-0.21747774701843273	p=0.0007303095449552845
svm_SBERT_bal_nli_mined_mode_dbscan	pct_change_1_month	r=-0.21747774701843273	p=0.0007303095449552845
svm_SBERT_bal_nli_mined_mean_kmeans	pct_change_1_month	r=-0.08346010211933337	p=0.199484880351828
svm_SBERT_bal_nli_mined_mean_meanshift	pct_change_1_month	r=-0.25338429745264657	p=7.71298815221731e-05
svm_SBERT_bal_nli_full_mean	pct_change_1_month	r=-0.2309768938682398	p=0.00032653089329424636
svm_SBERT_bal_nli_full_meanr	pct_change_1_month	r=-0.0836045618172273	p=0.19870525049005022
svm_SBERT_bal_nli_full_mode	pct_change_1_month	r=-0.14463875863912198	p=0.025656186466183815
rf_bow_wiki_mined_mean_agglo	pct_change_1_week	r=-0.027177972804869065	p=0.6765679871661192
rf_bow_wiki_mined_meanr_agglo	pct_change_1_week	r=-0.04679267935788431	p=0.4724642256755475
rf_bow_wiki_mined_mode_agglo	pct_change_1_week	r=-0.04679267935788431	p=0.4724642256755475
rf_bow_wiki_mined_mean_dbscan	pct_change_1_week	r=0.016561149803619907	p=0.7993678727889117
rf_bow_wiki_mined_meanr_dbscan	pct_change_1_week	r=0.023281381445436	p=0.7208484100672813
rf_bow_wiki_mined_mode_dbscan	pct_change_1_week	r=0.023281381445436	p=0.7208484100672813
rf_bow_wiki_mined_mean_kmeans	pct_change_1_week	r=0.003405349763129635 2	p=0.9583225295369925
rf_bow_wiki_mined_mean_meanshift	pct_change_1_week	r=0.022255495003109832	p=0.7326702010929401
rf_bow_wiki_full_mean	pct_change_1_week	r=-0.005223060629472278	p=0.9361149548731241
rf_bow_wiki_full_meanr	pct_change_1_week	r=nan	p=nan
rf_bow_wiki_full_mode	pct_change_1_week	r=nan	p=nan
rf_bow_nli_mined_mean_agglo	pct_change_1_week	r=-0.032372930577784335	p=0.619241141846573
rf_bow_nli_mined_meanr_agglo	pct_change_1_week	r=nan	p=nan
rf_bow_nli_mined_mode_agglo	pct_change_1_week	r=nan	p=nan
rf_bow_nli_mined_mean_dbscan	pct_change_1_week	r=-0.06444297728508618	p=0.3221872839635725
rf_bow_nli_mined_meanr_dbscan	pct_change_1_week	r=-0.06444297728508618	p=0.3221872839635725
rf_bow_nli_mined_mode_dbscan	pct_change_1_week	r=-0.06444297728508618	p=0.3221872839635725
rf_bow_nli_mined_mean_kmeans	pct_change_1_week	r=0.05115484152103919	p=0.43213818009254557
rf_bow_nli_mined_mean_meanshift	pct_change_1_week	r=-0.07017687829996902	p=0.2809149855432767
rf_bow_nli_full_mean	pct_change_1_week	r=-0.005223060629472278	p=0.9361149548731241
rf_bow_nli_full_meanr	pct_change_1_week	r=nan	p=nan
rf_bow_nli_full_mode	pct_change_1_week	r=nan	p=nan
nb_bow_wiki_mined_mean_agglo	pct_change_1_week	r=0.03838425623028515	p=0.5556848389518863

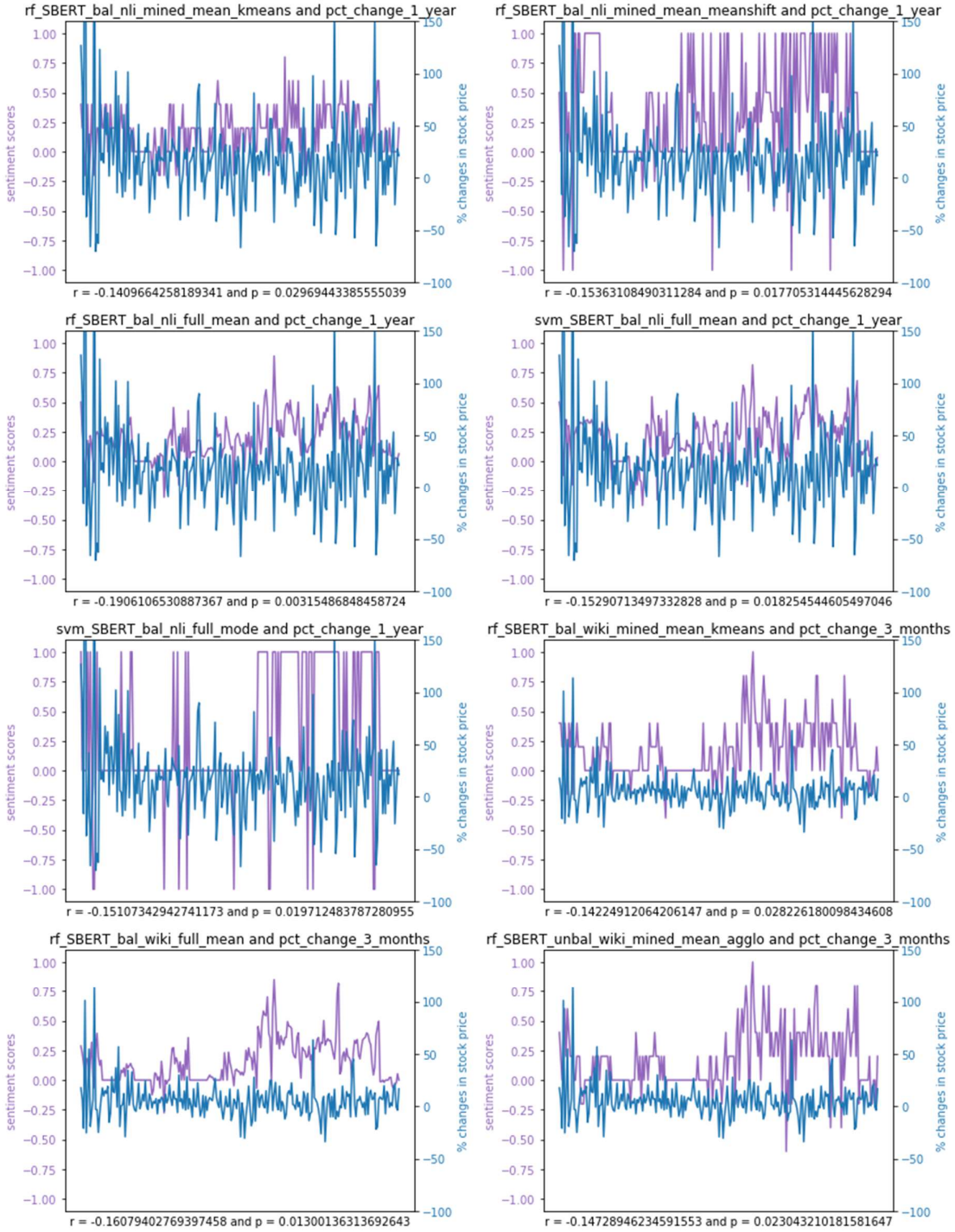
nb_bow_wiki_mined_meanr_agglo	pct_change_1_week	r=0.017333334257441685	p=0.7902255023955351
nb_bow_wiki_mined_mode_agglo	pct_change_1_week	r=-0.0007546105446531344	p=0.9907604837960359
nb_bow_wiki_mined_mean_dbscan	pct_change_1_week	r=-0.006141506212956588	p=0.924911674236036
nb_bow_wiki_mined_meanr_dbscan	pct_change_1_week	r=0.013189837140358108	p=0.8395880829010146
nb_bow_wiki_mined_mode_dbscan	pct_change_1_week	r=0.0033857486353990095	p=0.9585622092109751
nb_bow_wiki_mined_mean_kmeans	pct_change_1_week	r=0.0900388559618974	p=0.16618935253530223
nb_bow_wiki_mined_mean_meanshift	pct_change_1_week	r=-0.037627297904878176	p=0.5635111626796008
nb_bow_wiki_full_mean	pct_change_1_week	r=0.15023977341800443	p=0.020408044175353513
nb_bow_wiki_full_meanr	pct_change_1_week	r=nan	p=nan
nb_bow_wiki_full_mode	pct_change_1_week	r=0.010406405320073583	p=0.8731163786778836
rf_SBERT_bal_wiki_mined_mean_agglo	pct_change_1_week	r=-0.1153720146490583	p=0.07566064963439638
rf_SBERT_bal_wiki_mined_meanr_agglo	pct_change_1_week	r=-0.14625724614651628	p=0.024032081961429325
rf_SBERT_bal_wiki_mined_mode_agglo	pct_change_1_week	r=-0.12596582467874695	p=0.05228102818062038
rf_SBERT_bal_wiki_mined_mean_dbscan	pct_change_1_week	r=-0.014500882233228646	p=0.8238914932575394
rf_SBERT_bal_wiki_mined_meanr_dbscan	pct_change_1_week	r=-0.00033958306803183783	p=0.9958420421626717
rf_SBERT_bal_wiki_mined_mode_dbscan	pct_change_1_week	r=-0.06312042527830082	p=0.3322384477832026
rf_SBERT_bal_wiki_mined_mean_kmeans	pct_change_1_week	r=-0.06783195697115055	p=0.29734129172158574
rf_SBERT_bal_wiki_mined_mean_meanshift	pct_change_1_week	r=0.03502386527647298	p=0.590824117038542
rf_SBERT_bal_wiki_full_mean	pct_change_1_week	r=-0.004992430557508143	p=0.9389302637556011
rf_SBERT_bal_wiki_full_meanr	pct_change_1_week	r=-0.08753939282507323	p=0.17831311646885145
rf_SBERT_bal_wiki_full_mode	pct_change_1_week	r=0.006005266846899555	p=0.926572653607823
rf_SBERT_unbal_wiki_mined_mean_agglo	pct_change_1_week	r=-0.05748294726041545	p=0.37730915791094616
rf_SBERT_unbal_wiki_mined_meanr_agglo	pct_change_1_week	r=-0.11035983883678627	p=0.08936105310358819
rf_SBERT_unbal_wiki_mined_mode_agglo	pct_change_1_week	r=-0.0944032647347577	p=0.1465128499188618
rf_SBERT_unbal_wiki_mined_mean_dbscan	pct_change_1_week	r=-0.029081510425273023	p=0.6553227285483978
rf_SBERT_unbal_wiki_mined_meanr_dbscan	pct_change_1_week	r=-0.0427075804359484	p=0.5120230167969925
rf_SBERT_unbal_wiki_mined_mode_dbscan	pct_change_1_week	r=-0.041014657991692495	p=0.5289058909835056
rf_SBERT_unbal_wiki_mined_mean_kmeans	pct_change_1_week	r=-0.01715003221517929	p=0.7923931846745125
rf_SBERT_unbal_wiki_mined_mean_meanshift	pct_change_1_week	r=0.07686382583841225	p=0.2374824774172978
rf_SBERT_unbal_wiki_full_mean	pct_change_1_week	r=0.004411424143259326	p=0.9460259128714383
rf_SBERT_unbal_wiki_full_meanr	pct_change_1_week	r=-0.03118754347710005	p=0.6321368730562484
rf_SBERT_unbal_wiki_full_mode	pct_change_1_week	r=-0.008839577110115017	p=0.8920939951456724
rf_SBERT_bal_nli_mined_mean_agglo	pct_change_1_week	r=-0.020173358378326095	p=0.7568557191324847
rf_SBERT_bal_nli_mined_meanr_agglo	pct_change_1_week	r=-0.0132556052383181	p=0.8387991021153819
rf_SBERT_bal_nli_mined_mode_agglo	pct_change_1_week	r=0.013780052284895617	p=0.8325134456593158
rf_SBERT_bal_nli_mined_mean_dbscan	pct_change_1_week	r=-0.03640485696758545	p=0.5762605441109502
rf_SBERT_bal_nli_mined_meanr_dbscan	pct_change_1_week	r=-0.03640485696758545	p=0.5762605441109502
rf_SBERT_bal_nli_mined_mode_dbscan	pct_change_1_week	r=-0.03640485696758545	p=0.5762605441109502
rf_SBERT_bal_nli_mined_mean_kmeans	pct_change_1_week	r=-0.01390470752847713	p=0.8310209629348629
rf_SBERT_bal_nli_mined_mean_meanshift	pct_change_1_week	r=-0.022344643482176674	p=0.7316403487825045
rf_SBERT_bal_nli_full_mean	pct_change_1_week	r=0.0011099817729462857	p=0.9864096392102355
rf_SBERT_bal_nli_full_meanr	pct_change_1_week	r=-0.0123243130255112	p=0.8499858978479525
rf_SBERT_bal_nli_full_mode	pct_change_1_week	r=-0.0038874630569795303	p=0.9524285515257694
svm_SBERT_bal_wiki_mined_mean_agglo	pct_change_1_week	r=-0.012677080719358868	p=0.8457447505803833

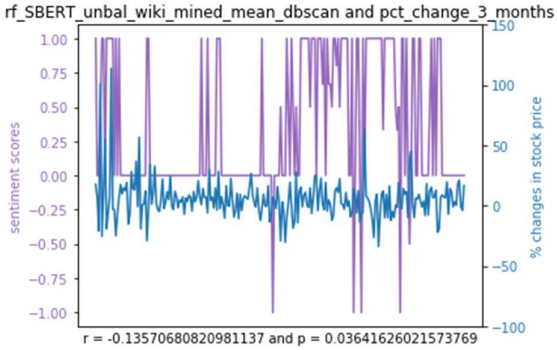
svm_SBERT_bal_wiki_mined_meanr_agglo	pct_change_1_week	r=nan	p=nan
svm_SBERT_bal_wiki_mined_mode_agglo	pct_change_1_week	r=nan	p=nan
svm_SBERT_bal_wiki_mined_mean_dbscan	pct_change_1_week	r=nan	p=nan
svm_SBERT_bal_wiki_mined_meanr_dbscan	pct_change_1_week	r=nan	p=nan
svm_SBERT_bal_wiki_mined_mode_dbscan	pct_change_1_week	r=nan	p=nan
svm_SBERT_bal_wiki_mined_mean_kmeans	pct_change_1_week	r=-0.033991733431442	p=0.6018173388637391
svm_SBERT_bal_wiki_mined_mean_meanshift	pct_change_1_week	r=0.04521211608701473	p=0.4875684422724911
svm_SBERT_bal_wiki_full_mean	pct_change_1_week	r=0.05064844034239992	p=0.4367154395159258
svm_SBERT_bal_wiki_full_meanr	pct_change_1_week	r=nan	p=nan
svm_SBERT_bal_wiki_full_mode	pct_change_1_week	r=nan	p=nan
svm_SBERT_bal_nli_mined_mean_agglo	pct_change_1_week	r=0.002168317504477249	p=0.9734552123862782
svm_SBERT_bal_nli_mined_meanr_agglo	pct_change_1_week	r=-0.0036853059250186544	p=0.954899686191675
svm_SBERT_bal_nli_mined_mode_agglo	pct_change_1_week	r=-0.056595460519077725	p=0.3847309831007757
svm_SBERT_bal_nli_mined_mean_dbscan	pct_change_1_week	r=-0.03966744035280933	p=0.5425394555610946
svm_SBERT_bal_nli_mined_meanr_dbscan	pct_change_1_week	r=-0.03966744035280933	p=0.5425394555610946
svm_SBERT_bal_nli_mined_mode_dbscan	pct_change_1_week	r=-0.03966744035280933	p=0.5425394555610946
svm_SBERT_bal_nli_mined_mean_kmeans	pct_change_1_week	r=0.02392014924822894	p=0.7135207166013017
svm_SBERT_bal_nli_mined_mean_meanshift	pct_change_1_week	r=-0.02737378248726356	p=0.6743703659135426
svm_SBERT_bal_nli_full_mean	pct_change_1_week	r=0.014400085998914442	p=0.8250958875443596
svm_SBERT_bal_nli_full_meanr	pct_change_1_week	r=-0.056394537905715615	p=0.3864234807498793
svm_SBERT_bal_nli_full_mode	pct_change_1_week	r=0.06196042837149885	p=0.34121810241728784

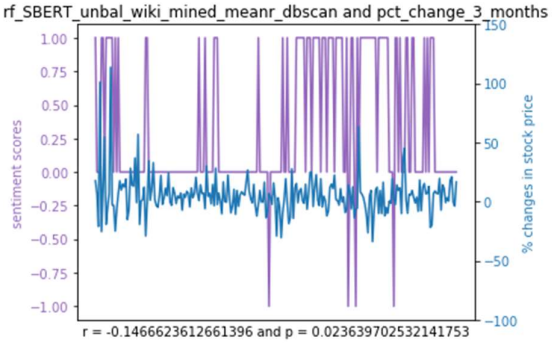
14.3 Appendix 3: Plots of all statistically significant combinations for correlation analyses between sentiment scores and future stock returns.

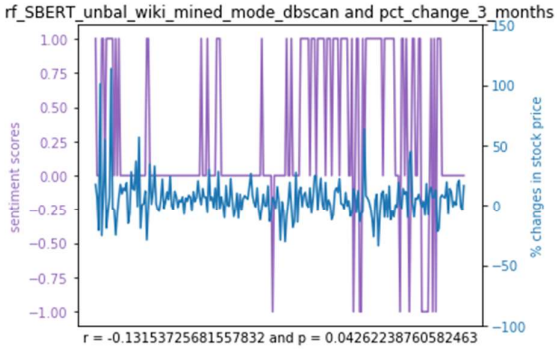
Full list of plots of all combinations between sentiment analysis predicted scores/labels and stock returns for combinations with statistically significant p-values ($p \leq 0.05$). The key for the sentiment variables notation is the same as above (see Appendix 2). In the title of each plot, the sentiment analysis variables are listed first followed by the time window used to calculate the stock returns. Sentiment scores are on the left y-axis in purple and stock returns are on the right y-axis in blue. The Pearson coefficient (r) and p-values are listed on the x-axis.

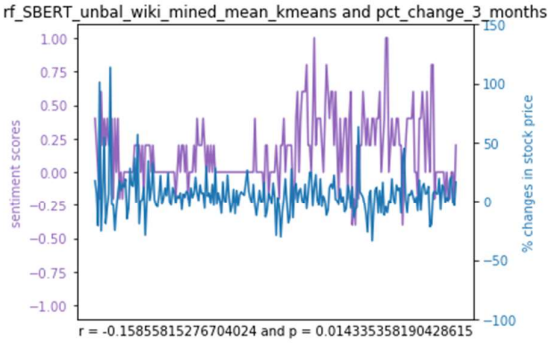


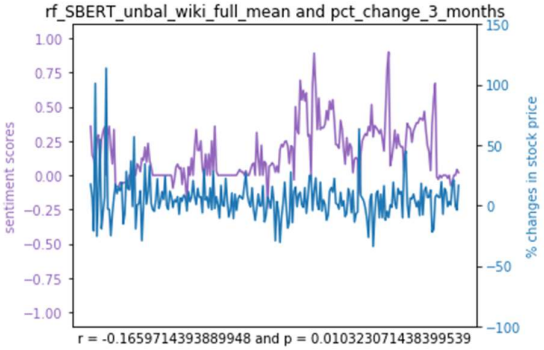

rf_SBERT_bal_nli_mined_mean_kmeans and pct_change_1_year
r = -0.1409664258189341 and p = 0.02969443385555039
rf_SBERT_bal_nli_mined_mean_meanshift and pct_change_1_year
r = -0.15363108490311284 and p = 0.017705314445628294
rf_SBERT_bal_nli_full_mean and pct_change_1_year
r = -0.1906106530887367 and p = 0.00315486848458724
svm_SBERT_bal_nli_full_mean and pct_change_1_year
r = -0.15290713497332828 and p = 0.018254544605497046
svm_SBERT_bal_nli_full_mode and pct_change_1_year
r = -0.15107342942741173 and p = 0.019712483787280955
rf_SBERT_bal_wiki_mined_mean_kmeans and pct_change_3_months
r = -0.14224912064206147 and p = 0.028226180098434608
rf_SBERT_bal_wiki_full_mean and pct_change_3_months
r = -0.16079402769397458 and p = 0.01300136313692643
rf_SBERT_unbal_wiki_mined_mean_agglo and pct_change_3_months
r = -0.14728946234591553 and p = 0.023043210181581647
sentiment scores
% changes in stock price

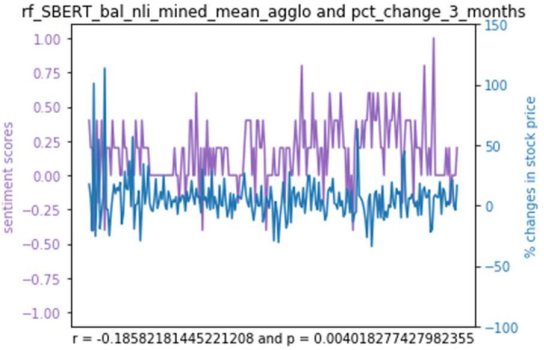

rf_SBERT_unbal_wiki_mined_mean_dbscan and pct_change_3_months
sentiment scores
% changes in stock price
r = -0.13570680820981137 and p = 0.03641626021573769

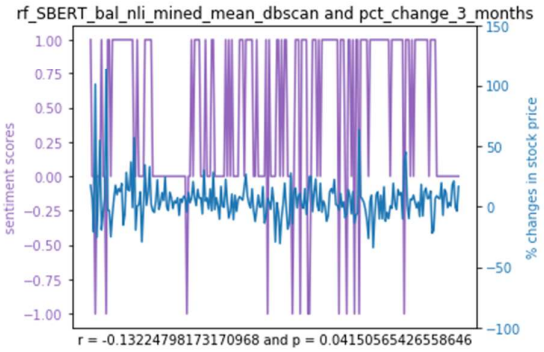

rf_SBERT_unbal_wiki_mined_meanr_dbscan and pct_change_3_months
sentiment scores
% changes in stock price
r = -0.1466623612661396 and p = 0.023639702532141753

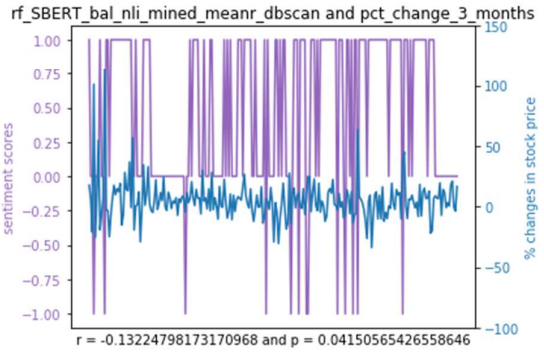

rf_SBERT_unbal_wiki_mined_mode_dbscan and pct_change_3_months
sentiment scores
% changes in stock price
r = -0.13153725681557832 and p = 0.04262238760582463

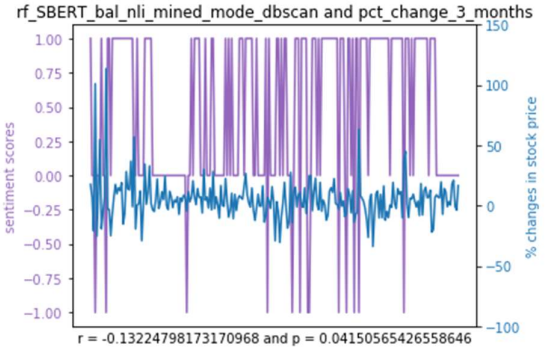

rf_SBERT_unbal_wiki_mined_mean_kmeans and pct_change_3_months
sentiment scores
% changes in stock price
r = -0.15855815276704024 and p = 0.014335358190428615

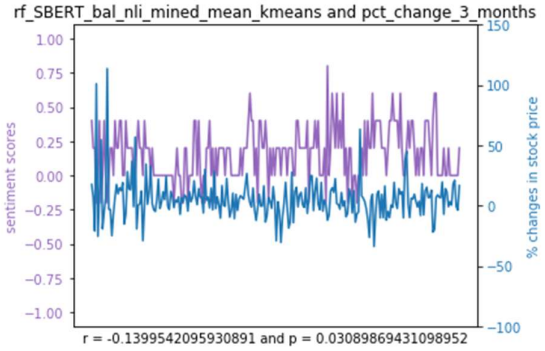

rf_SBERT_unbal_wiki_full_mean and pct_change_3_months
sentiment scores
% changes in stock price
r = -0.1659714393889948 and p = 0.010323071438399539

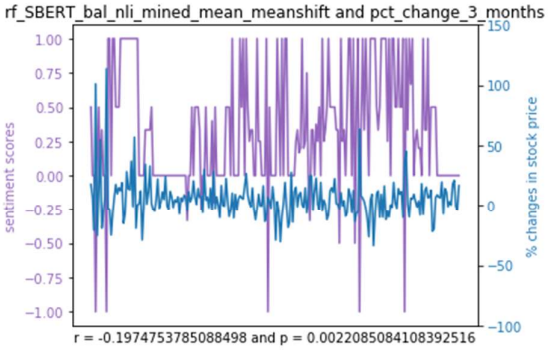

rf_SBERT_bal_nli_mined_mean_agglo and pct_change_3_months
sentiment scores
% changes in stock price
r = -0.18582181445221208 and p = 0.004018277427982355

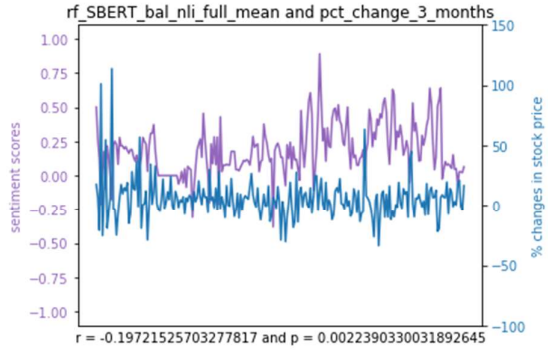

rf_SBERT_bal_nli_mined_mean_dbscan and pct_change_3_months
sentiment scores
% changes in stock price
r = -0.13224798173170968 and p = 0.04150565426558646

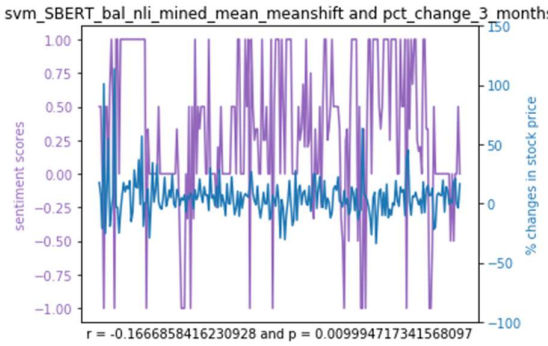

rf_SBERT_bal_nli_mined_meanr_dbscan and pct_change_3_months
sentiment scores
% changes in stock price
r = -0.13224798173170968 and p = 0.04150565426558646

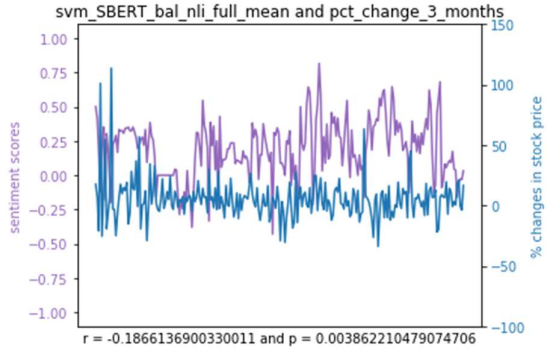

rf_SBERT_bal_nli_mined_mode_dbscan and pct_change_3_months
sentiment scores
% changes in stock price
r = -0.13224798173170968 and p = 0.04150565426558646

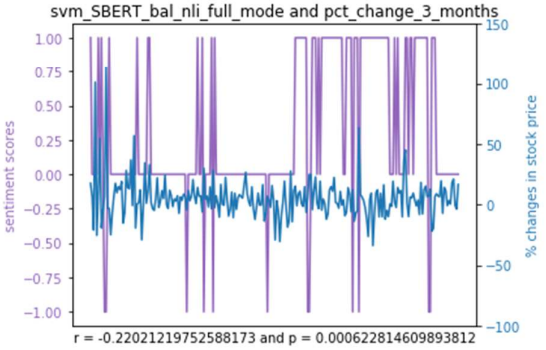

rf_SBERT_bal_nli_mined_mean_kmeans and pct_change_3_months
sentiment scores
% changes in stock price
r = -0.1399542095930891 and p = 0.03089869431098952

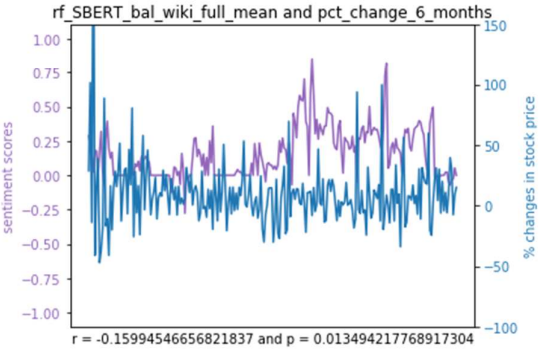

rf_SBERT_bal_nli_mined_mean_meanshift and pct_change_3_months
sentiment scores
% changes in stock price
r = -0.1974753785088498 and p = 0.0022085084108392516

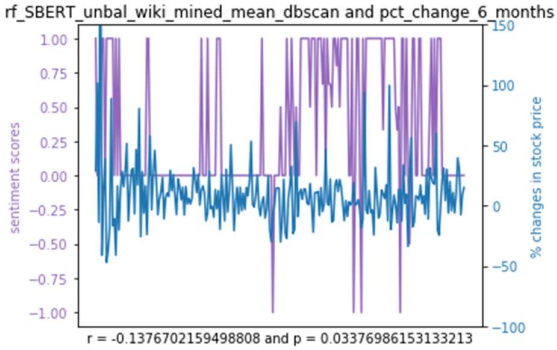

rf_SBERT_bal_nli_full_mean and pct_change_3_months
sentiment scores
% changes in stock price
r = -0.19721525703277817 and p = 0.0022390330031892645

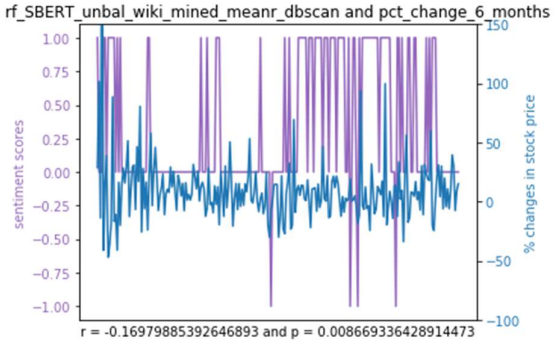

svm_SBERT_bal_nli_mined_mean_meanshift and pct_change_3_months
sentiment scores
% changes in stock price
r = -0.1666858416230928 and p = 0.009994717341568097

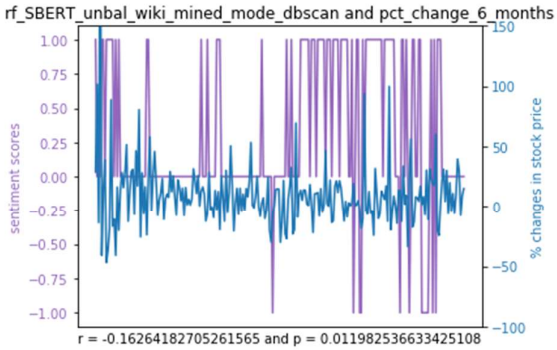

svm_SBERT_bal_nli_full_mean and pct_change_3_months
sentiment scores
% changes in stock price
r = -0.1866136900330011 and p = 0.003862210479074706

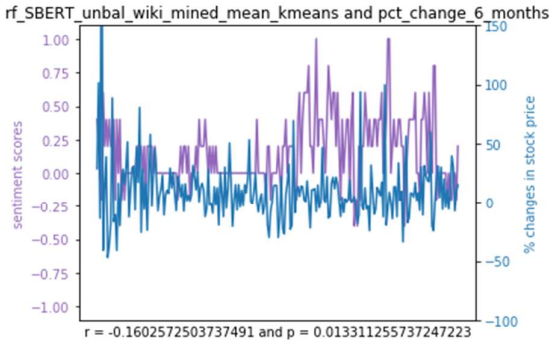

svm_SBERT_bal_nli_full_mode and pct_change_3_months
sentiment scores
% changes in stock price
r = -0.22021219752588173 and p = 0.000622814609893812

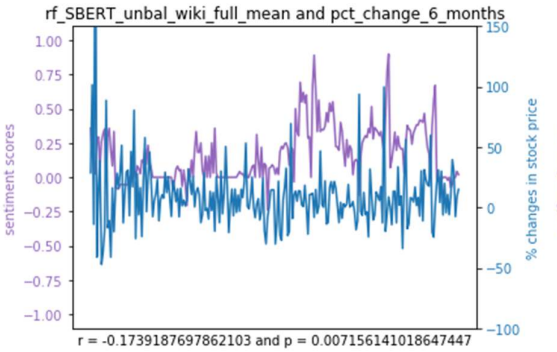

rf_SBERT_bal_wiki_full_mean and pct_change_6_months
sentiment scores
% changes in stock price
r = -0.15994546656821837 and p = 0.013494217768917304

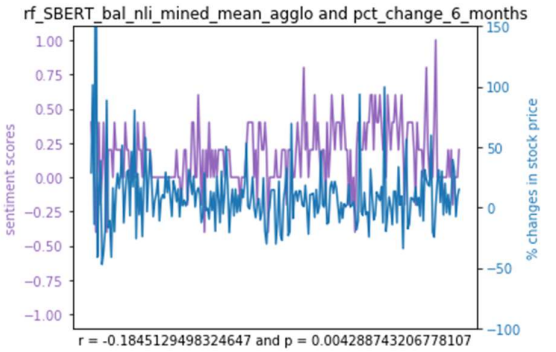

rf_SBERT_unbal_wiki_mined_mean_dbscan and pct_change_6_months
sentiment scores
% changes in stock price
r = -0.1376702159498808 and p = 0.03376986153133213

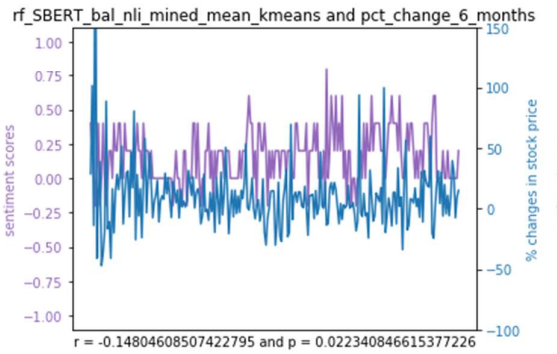

rf_SBERT_unbal_wiki_mined_meanr_dbscan and pct_change_6_months
sentiment scores
% changes in stock price
r = -0.16979885392646893 and p = 0.008669336428914473

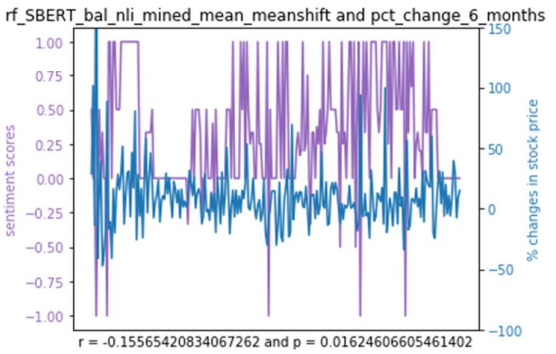

rf_SBERT_unbal_wiki_mined_mode_dbscan and pct_change_6_months
sentiment scores
% changes in stock price
r = -0.16264182705261565 and p = 0.011982536633425108

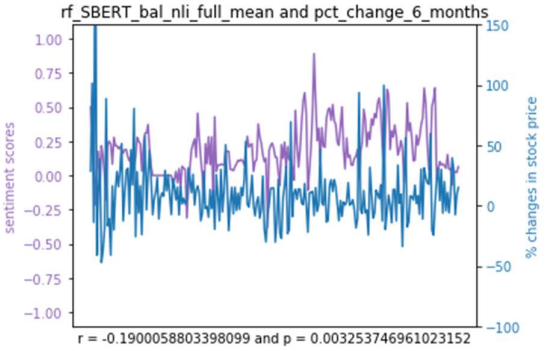

rf_SBERT_unbal_wiki_mined_mean_kmeans and pct_change_6_months
sentiment scores
% changes in stock price
r = -0.1602572503737491 and p = 0.013311255737247223

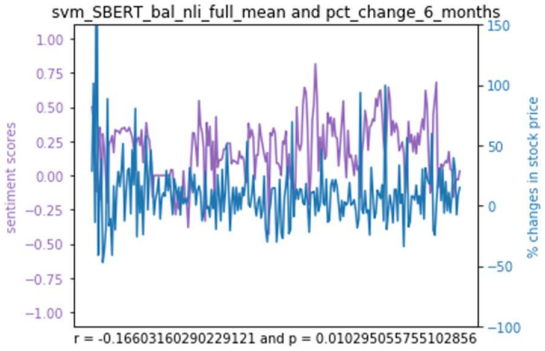

rf_SBERT_unbal_wiki_full_mean and pct_change_6_months
sentiment scores
% changes in stock price
r = -0.1739187697862103 and p = 0.007156141018647447

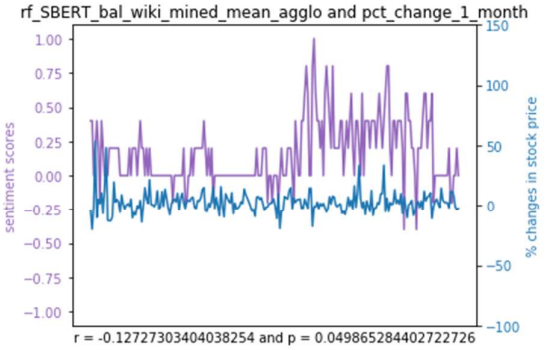

rf_SBERT_bal_nli_mined_mean_agglo and pct_change_6_months
sentiment scores
% changes in stock price
r = -0.1845129498324647 and p = 0.004288743206778107

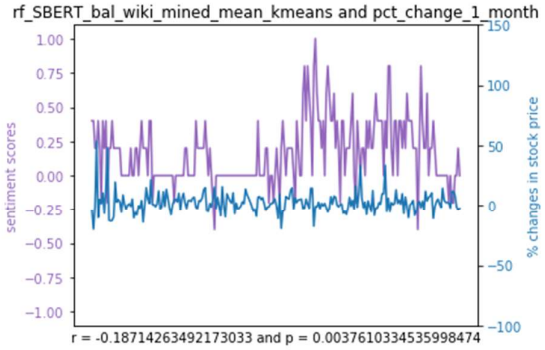

rf_SBERT_bal_nli_mined_mean_kmeans and pct_change_6_months
sentiment scores
% changes in stock price
r = -0.14804608507422795 and p = 0.022340846615377226

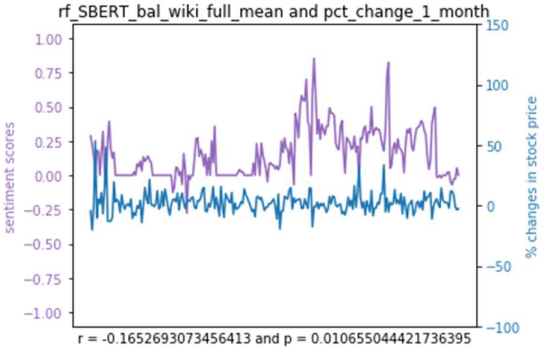

rf_SBERT_bal_nli_mined_mean_meanshift and pct_change_6_months
sentiment scores
% changes in stock price
r = -0.15565420834067262 and p = 0.01624606605461402

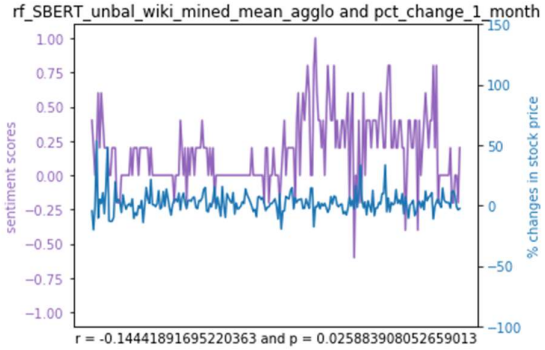

rf_SBERT_bal_nli_full_mean and pct_change_6_months
sentiment scores
% changes in stock price
r = -0.1900058803398099 and p = 0.003253746961023152

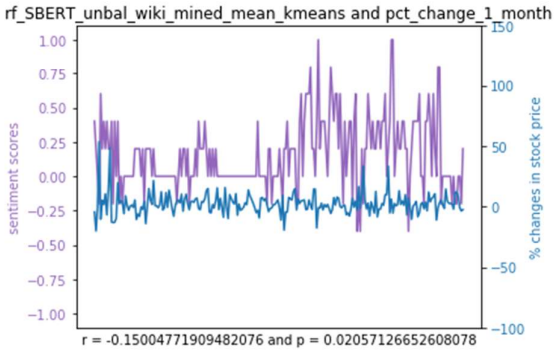

svm_SBERT_bal_nli_full_mean and pct_change_6_months
sentiment scores
% changes in stock price
r = -0.16603160290229121 and p = 0.010295055755102856

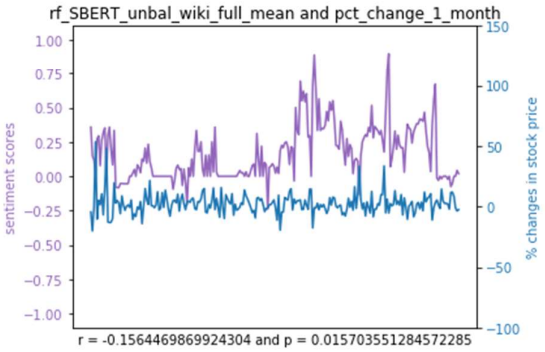

rf_SBERT_bal_wiki_mined_mean_agglo and pct_change_1_month
sentiment scores
% changes in stock price
r = -0.12727303404038254 and p = 0.049865284402722726

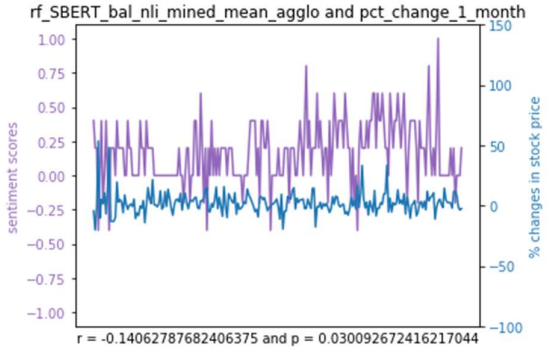

rf_SBERT_bal_wiki_mined_mean_kmeans and pct_change_1_month
sentiment scores
% changes in stock price
r = -0.18714263492173033 and p = 0.0037610334535998474

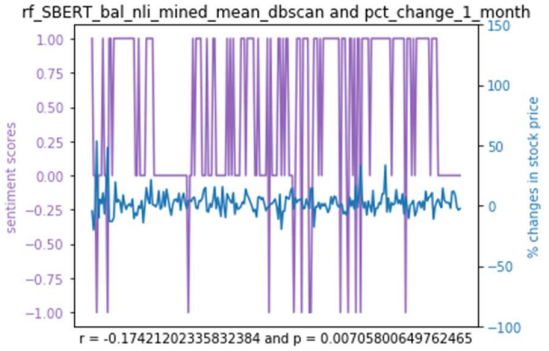

rf_SBERT_bal_wiki_full_mean and pct_change_1_month
sentiment scores
% changes in stock price
r = -0.1652693073456413 and p = 0.010655044421736395

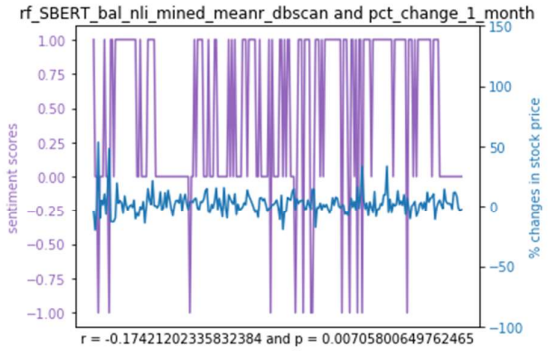

rf_SBERT_unbal_wiki_mined_mean_agglo and pct_change_1_month
sentiment scores
% changes in stock price
r = -0.14441891695220363 and p = 0.025883908052659013

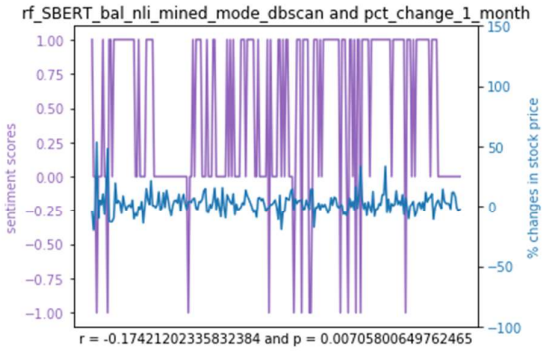

rf_SBERT_unbal_wiki_mined_mean_kmeans and pct_change_1_month
sentiment scores
% changes in stock price
r = -0.15004771909482076 and p = 0.02057126652608078

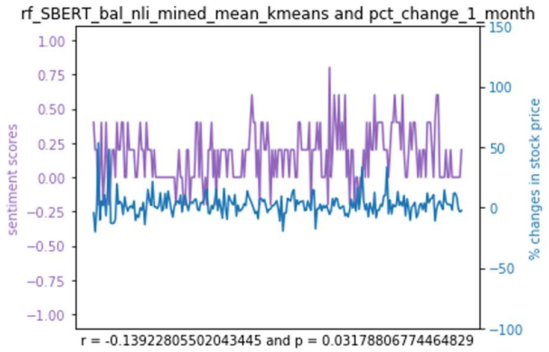

rf_SBERT_unbal_wiki_full_mean and pct_change_1_month
sentiment scores
% changes in stock price
r = -0.1564469869924304 and p = 0.015703551284572285

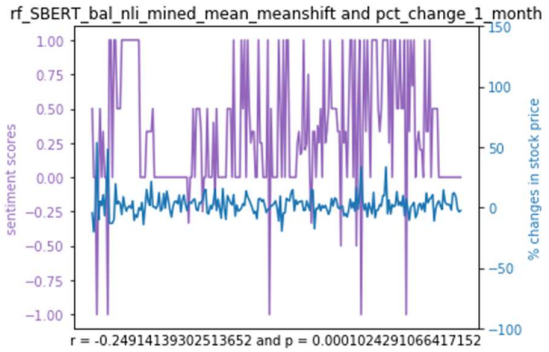

rf_SBERT_bal_nli_mined_mean_agglo and pct_change_1_month
sentiment scores
% changes in stock price
r = -0.14062787682406375 and p = 0.0300926724162170 44

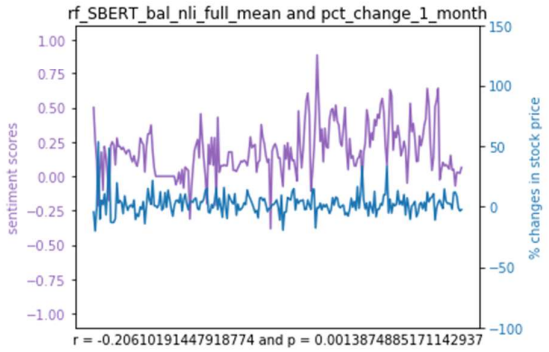

rf_SBERT_bal_nli_mined_mean_dbscan and pct_change_1_month
sentiment scores
% changes in stock price
r = -0.17421202335832384 and p = 0.00705800649762465

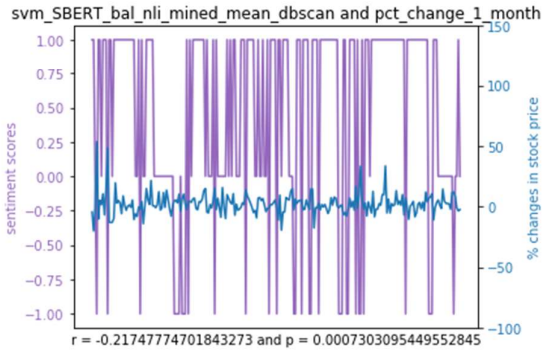

rf_SBERT_bal_nli_mined_meanr_dbscan and pct_change_1_month
sentiment scores
% changes in stock price
r = -0.17421202335832384 and p = 0.00705800649762465

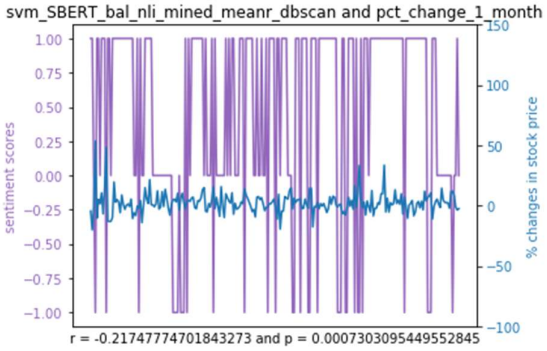

rf_SBERT_bal_nli_mined_mode_dbscan and pct_change_1_month
sentiment scores
% changes in stock price
r = -0.17421202335832384 and p = 0.00705800649762465

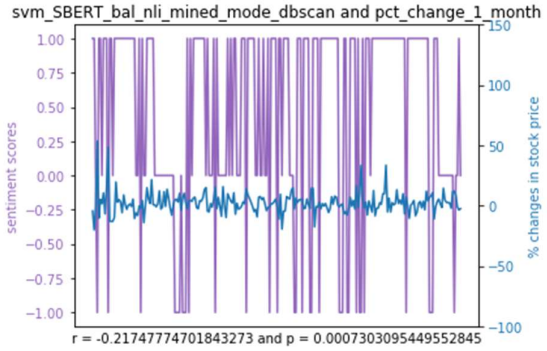

rf_SBERT_bal_nli_mined_mean_kmeans and pct_change_1_month
sentiment scores
% changes in stock price
r = -0.13922805502043445 and p = 0.03178806774464829

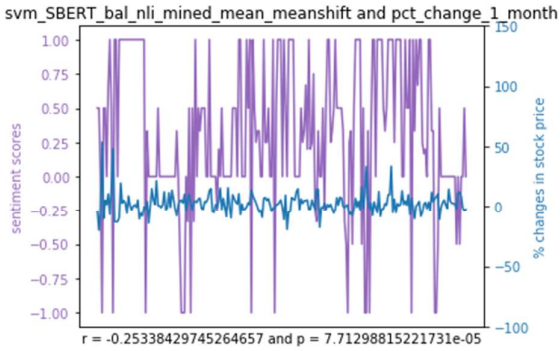

rf_SBERT_bal_nli_mined_mean_meanshift and pct_change_1_month
sentiment scores
% changes in stock price
r = -0.24914139302513652 and p = 0.0001024291066417152

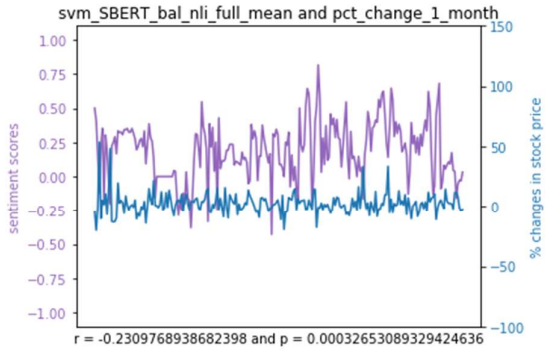

rf_SBERT_bal_nli_full_mean and pct_change_1_month
sentiment scores
% changes in stock price
r = -0.20610191447918774 and p = 0.0013874885171142937

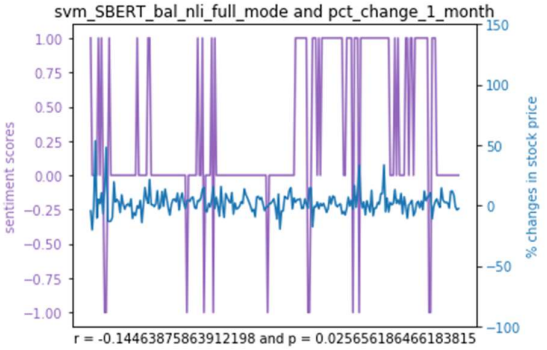

svm_SBERT_bal_nli_mined_mean_dbscan and pct_change_1_month
sentiment scores
% changes in stock price
r = -0.21747774701843273 and p = 0.0007303095449552845

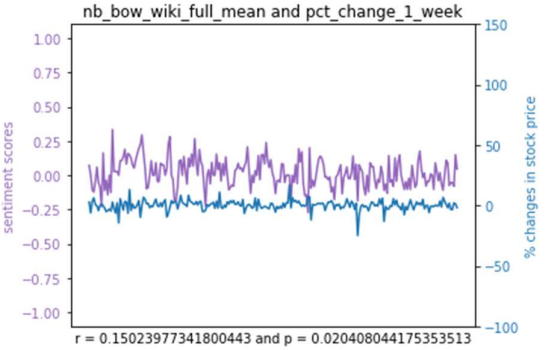

svm_SBERT_bal_nli_mined_meanr_dbscan and pct_change_1_month
sentiment scores
% changes in stock price
r = -0.21747774701843273 and p = 0.0007303095449552845

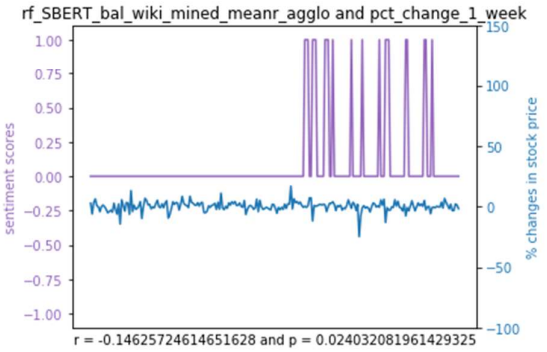

svm_SBERT_bal_nli_mined_mode_dbscan and pct_change_1_month
sentiment scores
% changes in stock price
r = -0.21747774701843273 and p = 0.0007303095449552845


svm_SBERT_bal_nli_mined_mean_meanshift and pct_change_1_month
sentiment scores
% changes in stock price
r = -0.25338429745264657 and p = 7.71298815221731e-05


svm_SBERT_bal_nli_full_mean and pct_change_1_month
sentiment scores
% changes in stock price
r = -0.2309768938682398 and p = 0.00032653089329424636

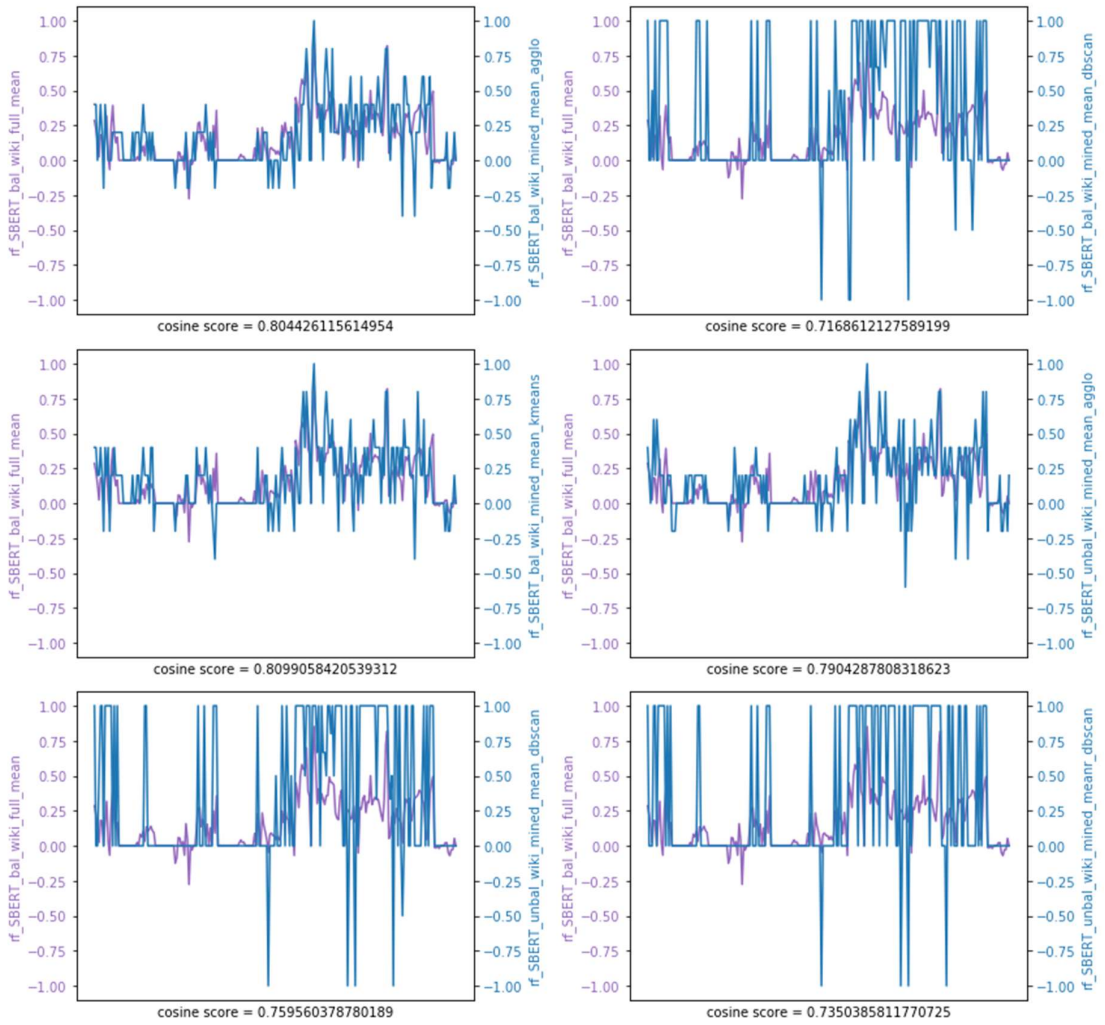

svm_SBERT_bal_nli_full_mode and pct_change_1_month
sentiment scores
% changes in stock price
r = -0.14463875863912198 and p = 0.025656186466183815

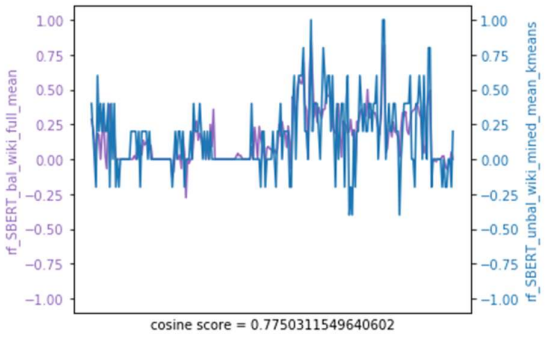

nb_bow_wiki_full_mean and pct_change_1_week
sentiment scores
% changes in stock price
r = 0.15023977341800443 and p = 0.020408044175353513

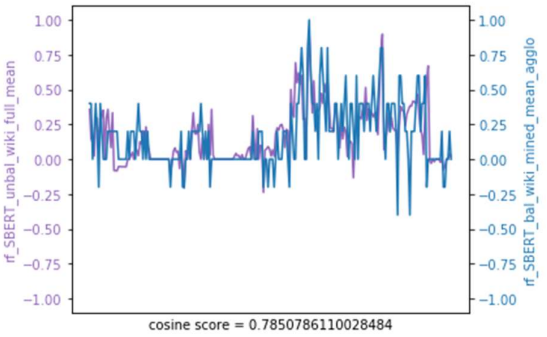

rf_SBERT_bal_wiki_mined_meanr_agglo and pct_change_1_week
sentiment scores
% changes in stock price
r = -0.14625724614651628 and p = 0.024032081961429325

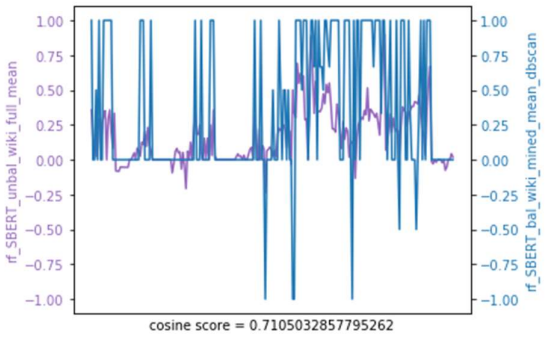
14.4 Appendix 4: Plots of all mined justifications versus full document aggregated sentiment scores with large cosine similarities

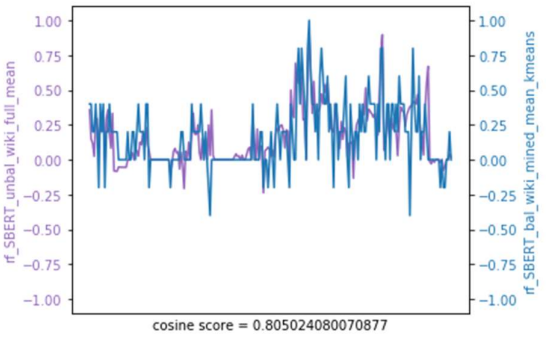
Plots of all combinations between sentiment analysis predicted scores/labels from mined justifications versus those from full documents with larger cosine similarities (>0.7). Sentiment scores from full document sentiment analyses are on the left y-axis in purple and sentiment scores from mined justifications sentiment analyses are on the right y-axis in blue. The y-axes labels are the sentiment analysis variables notation, the key for which was given previously in Appendix 2. The cosine score is listed on the x-axis.

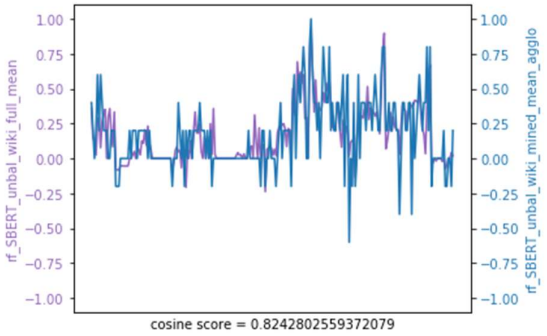


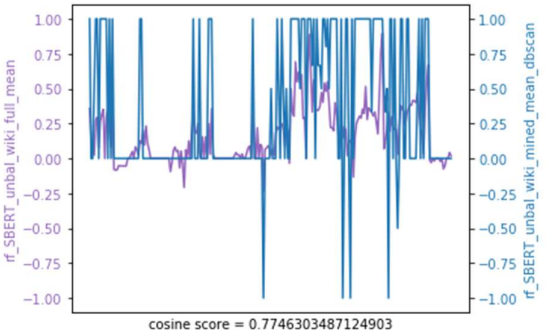

rf_SBERT_bal_wiki_full_mean
rf_SBERT_unbal_wiki_mined_mean_kmeans
cosine score = 0.7750311549640602

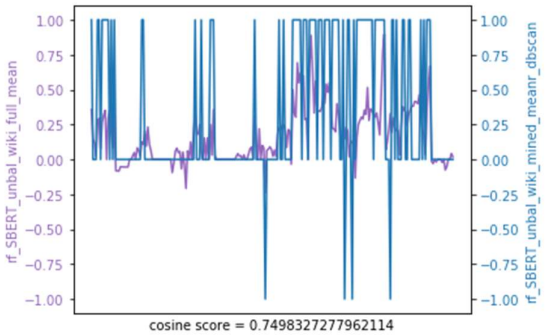

rf_SBERT_unbal_wiki_full_mean
rf_SBERT_bal_wiki_mined_mean_agglo
cosine score = 0.7850786110028484

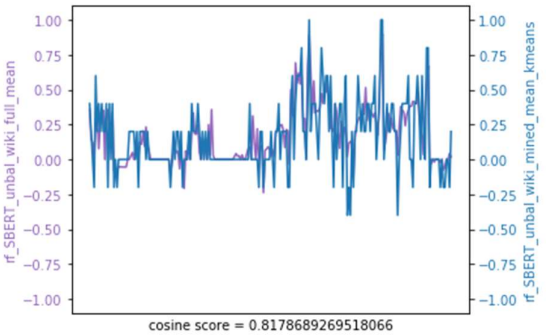

rf_SBERT_unbal_wiki_full_mean
rf_SBERT_bal_wiki_mined_mean_dbscan
cosine score = 0.7105032857795262

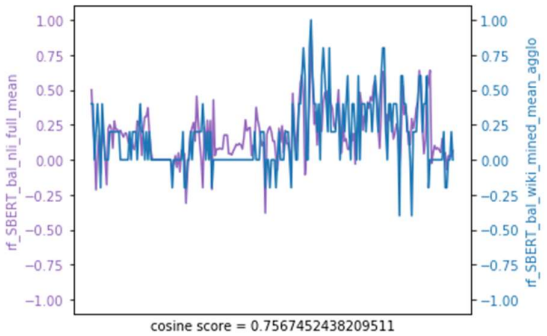

rf_SBERT_unbal_wiki_full_mean
rf_SBERT_bal_wiki_mined_mean_kmeans
cosine score = 0.805024080070877

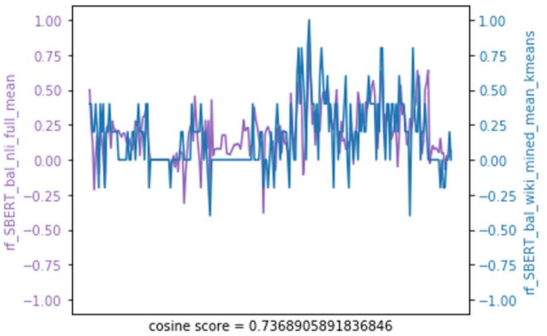

rf_SBERT_unbal_wiki_full_mean
rf_SBERT_unbal_wiki_mined_mean_agglo
cosine score = 0.8242802559372079

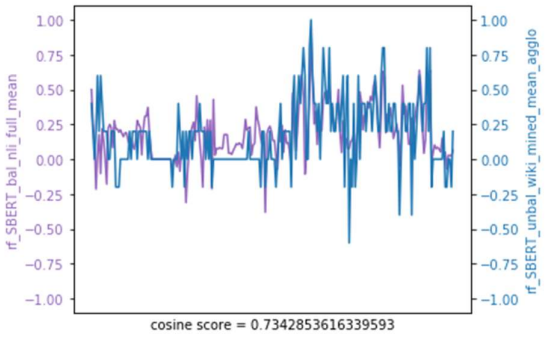

rf_SBERT_unbal_wiki_full_mean
rf_SBERT_unbal_wiki_mined_mean_dbscan
cosine score = 0.7746303487124903


rf_SBERT_unbal_wiki_full_mean
rf_SBERT_unbal_wiki_mined_meanr_dbscan
cosine score = 0.7498327277962114

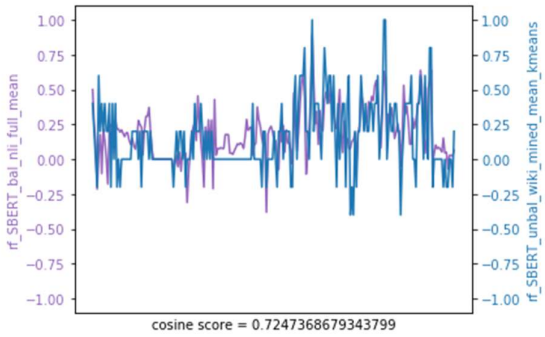

rf_SBERT_unbal_wiki_full_mean
rf_SBERT_unbal_wiki_mined_mean_kmeans
cosine score = 0.8178689269518066


rf_SBERT_bal_nli_full_mean
rf_SBERT_bal_wiki_mined_mean_agglo
cosine score = 0.7567452438209511

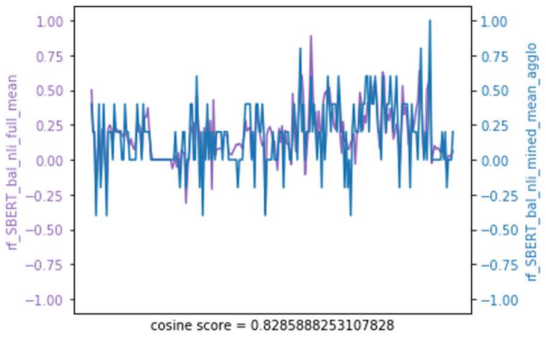

rf_SBERT_bal_nli_full_mean
rf_SBERT_bal_wiki_mined_mean_kmeans
cosine score = 0.7368905891836846



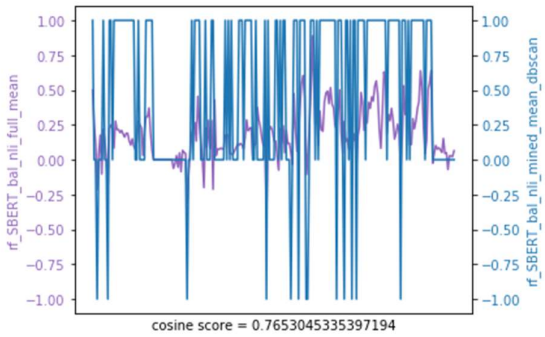
cosine score = 0.7342853616339593



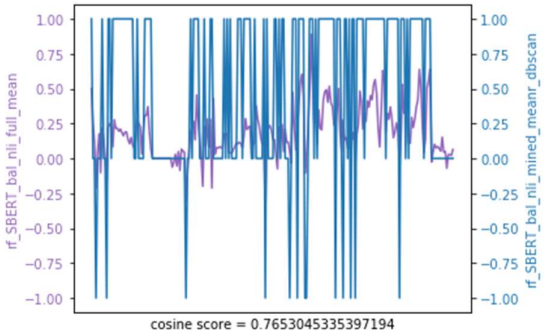
cosine score = 0.7247368679343799



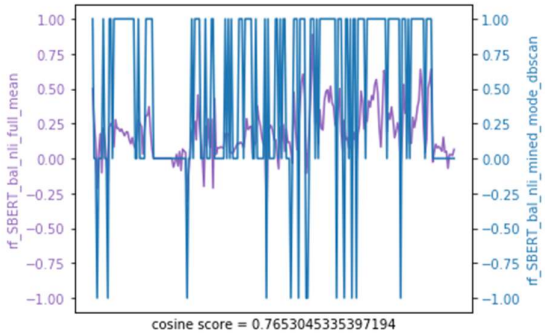
cosine score = 0.8285888253107828



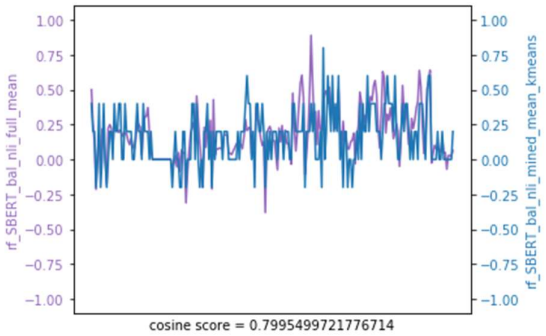
cosine score = 0.7653045335397194



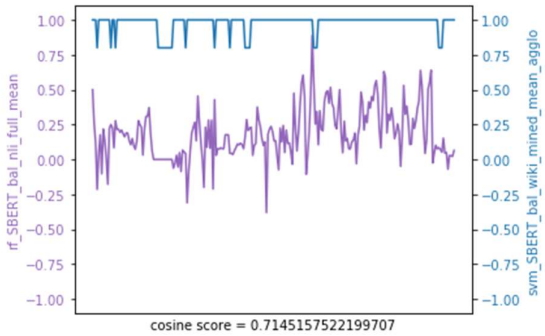
cosine score = 0.7653045335397194



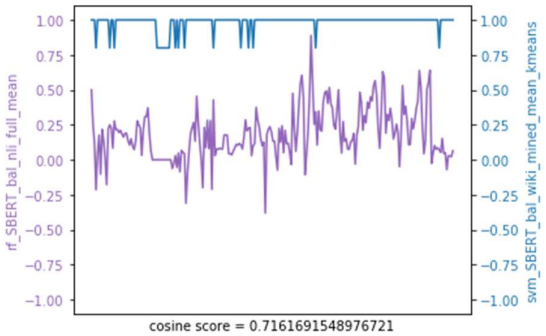
cosine score = 0.7653045335397194



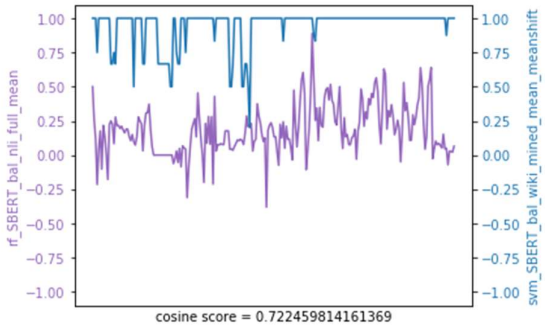
cosine score = 0.7995499721776714



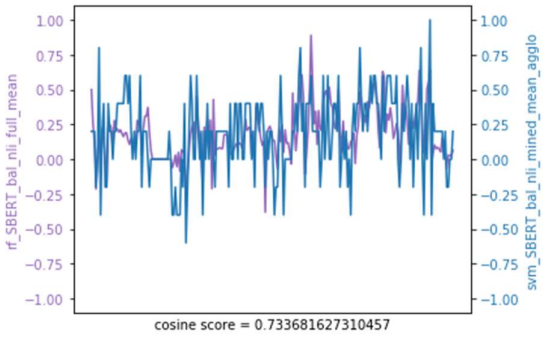
cosine score = 0.7145157522199707

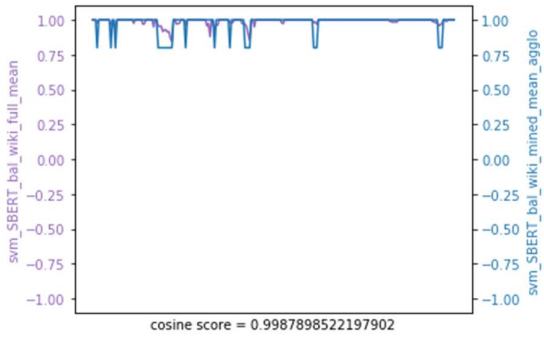


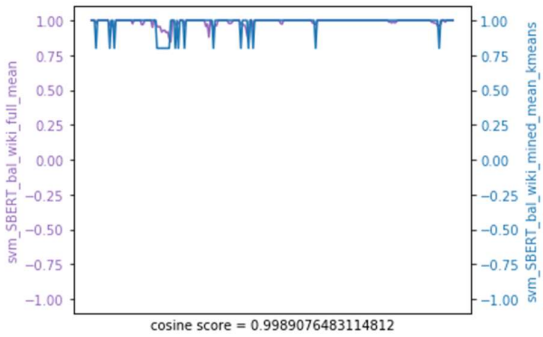
cosine score = 0.7161691548976721

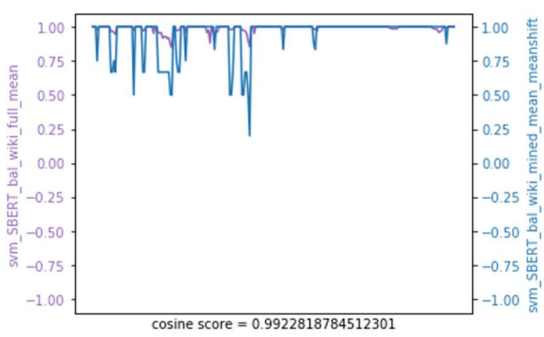


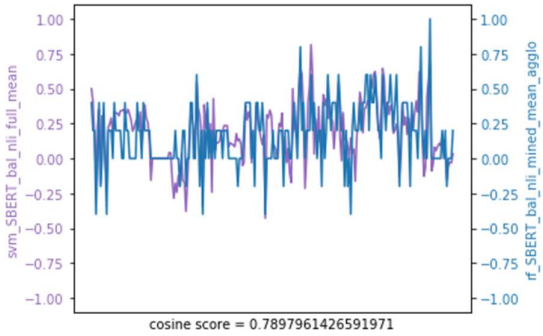
cosine score = 0.722459814161369

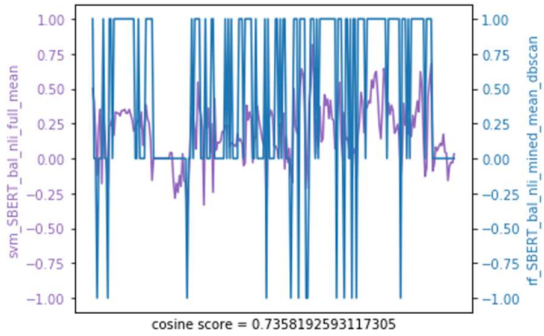

rf_SBERT_bal_nli_full_mean
svm_SBERT_bal_nli_mined_mean_agglo
cosine score = 0.733681627310457

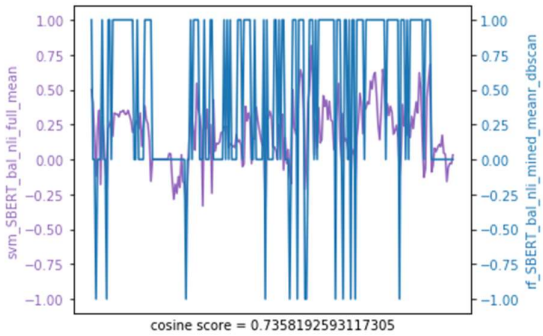

svm_SBERT_bal_wiki_full_mean
svm_SBERT_bal_wiki_mined_mean_agglo
cosine score = 0.9987898522197902

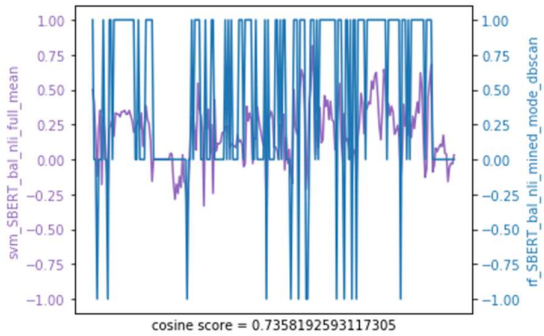

svm_SBERT_bal_wiki_full_mean
svm_SBERT_bal_wiki_mined_mean_kmeans
cosine score = 0.9989076483114812

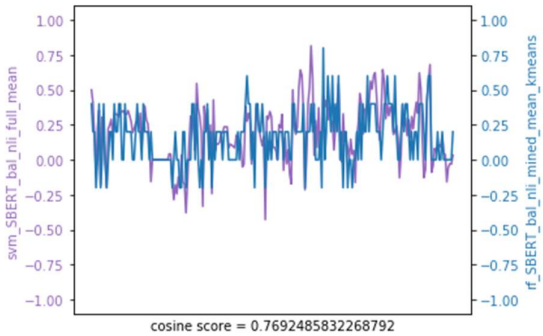

svm_SBERT_bal_wiki_full_mean
svm_SBERT_bal_wiki_mined_mean_meanshift
cosine score = 0.9922818784512301

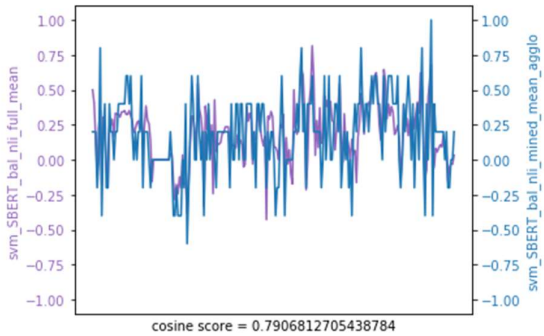

svm_SBERT_bal_nli_full_mean
rf_SBERT_bal_nli_mined_mean_agglo
cosine score = 0.7897961426591971

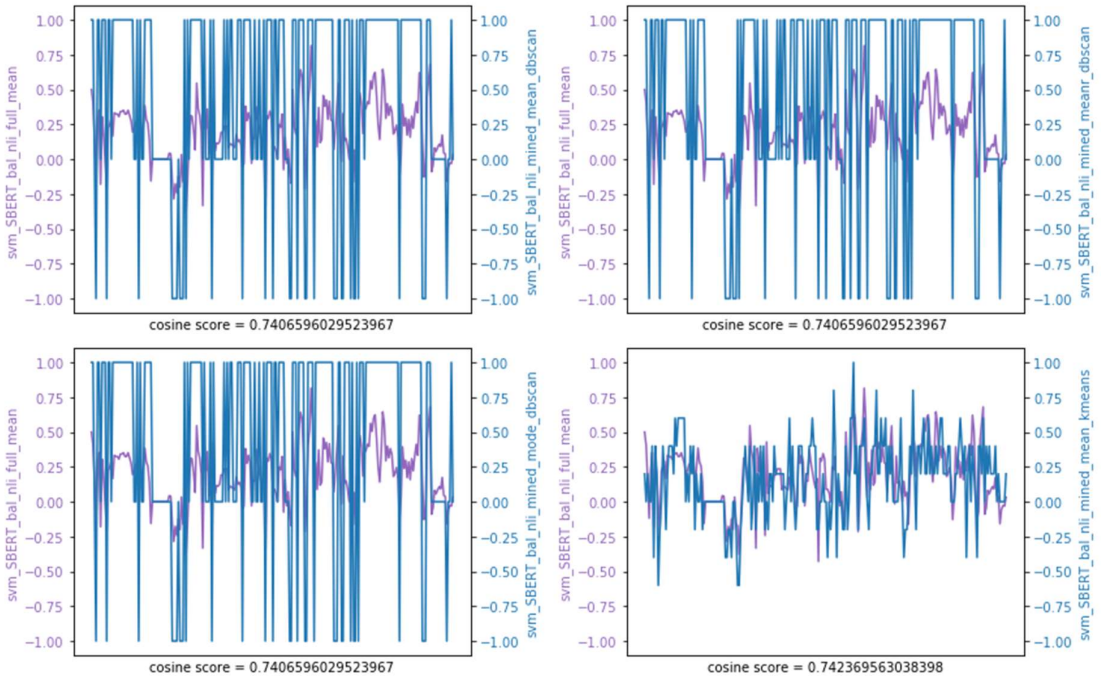

svm_SBERT_bal_nli_full_mean
rf_SBERT_bal_nli_mined_mean_dbscan
cosine score = 0.7358192593117305


svm_SBERT_bal_nli_full_mean
rf_SBERT_bal_nli_mined_meanr_dbscan
cosine score = 0.7358192593117305


svm_SBERT_bal_nli_full_mean
rf_SBERT_bal_nli_mined_mode_dbscan
cosine score = 0.7358192593117305


svm_SBERT_bal_nli_full_mean
rf_SBERT_bal_nli_mined_mean_kmeans
cosine score = 0.7692485832268792


svm_SBERT_bal_nli_full_mean
svm_SBERT_bal_nli_mined_mean_agglo
cosine score = 0.7906812705438784


svm_SBERT_bal_nli_full_mean
svm_SBERT_bal_nli_mined_mean_dbscan
cosine score = 0.7406596029523967
svm_SBERT_bal_nli_full_mean
svm_SBERT_bal_nli_mined_meanr_dbscan
cosine score = 0.7406596029523967
svm_SBERT_bal_nli_full_mean
svm_SBERT_bal_nli_mined_mode_dbscan
cosine score = 0.7406596029523967
svm_SBERT_bal_nli_full_mean
svm_SBERT_bal_nli_mined_mean_kmeans
cosine score = 0.742369563038398