Improving Machine Learning Models Through Enrichment of Training Data

Yuanzhe Jin

St. Anne's College

University of Oxford

A thesis submitted for the degree of

Doctor of Philosophy

Michaelmas 2025

Acknowledgements

I would like to express my gratitude to my supervisor, Professor Min Chen. His wise guidance and support over the past few years have been instrumental in my journey of learning and discovery in the field of visualization. It was through his assistance that I had the opportunity to participate in the RISE project organized by the European Network of Data Scientists (NeEDS), led by Dolores Romero Morales from Copenhagen Business School, supported by the European Union. This project enabled me to undertake two secondments in the Netherlands and Spain, which were important for the development of three key chapters of this thesis.

I am also deeply grateful to the staff of the Department of Engineering Science and the Oxford e-Research Centre for their unwavering support and assistance. The backing provided by the department and the university allowed me to collaborate with outstanding scientists in Europe during my secondments without any concerns, and to share my experiences upon returning to Oxford.

During my secondment at Statistics Netherlands, I am particularly grateful to Tim J. A. de Jong and Martijn Tennekes for their support. Their warm and welcoming research and work environment left a lasting impression on me during my first visit to the Netherlands and inspired the work in two chapters of this thesis.

During my secondment at Inetum in Spain, I deeply appreciate Adrián Carrasco-Revilla for his assistance during my four months in Madrid. I am also thankful to Jesús Otero, the department head at Inetum, whose support allowed me, as a first-time visitor to Spain, to experience the warmth and inclusivity of Spanish culture. My time at Inetum was incredibly rewarding, offering me valuable insights into the differences between academia and industry, which have greatly informed my future career choices.

I have also learned a lot from my lab collaborators, Chenhao Xue, Zelin Ye, Zukang Liao, and Yu Zhang, whose contributions have been valuable to my growth. Additionally, I extend my gratitude to Professor Wendy Mackay and Professor Theophanis Tsandilas at Université Paris-Saclay for our enlightening discussions, which deepened my understanding of human-computer interaction principles.

Finally, I owe my deepest thanks to my friends and family, who have been a constant source of motivation. Thanks to your unwavering support, I was able to pursue my research while balancing other aspirations in life. You have been like a lighthouse on the shore, illuminating the darkness along my journey.

Abstract

There has been a rapid development of machine learning (ML) technology over the past decade. However, trained ML models frequently encounter performance degradation when deployed in real-world applications. One major cause of this phenomenon is the inconsistencies in terms of the information captured in different data spaces. Such inconsistencies are often manifested as discrepancies between the development data space and the deployment data space, or between the application-specific data space and the artificially designed adversarial data space. Existing approaches typically address these challenges by focusing on experiment-based optimization of model architectures, training parameters, or data preprocessing techniques, but such methods do not address the fundamental inconsistencies in terms of data spaces and often find it difficult to achieve the desired model improvement after extensive effort in experiments for model optimization. Therefore, it is desirable to formulate and support ML workflows that address the aforementioned inconsistencies in terms of data spaces. In particular, a workflow capable of analyzing and improving ML model performance from the perspective of training data quality has become an urgent need for both ML developers and users.

This thesis examines three applications, where ML models were developed for classifying text, image, and time series data, respectively. They feature two types of inconsistencies, i.e., (a) the data space encountered after model deployment becomes noticeably different from that used for training and testing, and (b) adversarial attacks that use data in a distorted data space differing from that used for training and testing. To address such inconsistencies, model developers need to focus on the data preparation stage of ML workflows and employ a variety of supporting techniques to visualize different datasets and their high-dimensional feature spaces, analyze different forms of data space inconsistencies, generate synthetic data for testing and improving models, and so on.

In order to support such data-focused efforts for improving ML models, a new design concept, namely "Four Views Design", is proposed in this thesis. It is a novel contribution to the area of using visualization and visual analytics (VIS) to support ML development (VIS4ML). Unlike most existing designs of VIS4ML software, it introduces a specific phase (View) of Synthetic Data Generation into the traditional "Data-Training-Testing" loop, while providing a variety of VIS capabilities to support

all four phases (Four Views) in an iterative ML workflow. Three case studies were presented in this thesis. The iGAiVA system focuses on a real-world application of text classification, addressing the inconsistency between the data spaces used for development and deployment. While LLMs are used to synthesize data to alleviate the deficiencies in a training data space, VIS techniques were used to target data augmentation to specific areas in a data space, such that model improvement is likely attained. The TA3 system focuses on a classic problem of adversarial attacks on an image classification model. VIS techniques were used to observe a huge volume of testing data resulting from repeated attempts of attacks using different synthetic images. The DS4ML system focuses on another real-world application of time series classification, where the training data is not captured in a manner consistent with real-world scenarios due to the practical constraints in data collection processes. A number of VIS techniques were used to compare different data augmentation methods. As part of the development of the DS4ML system, a novel visualization design was formulated, namely Radial Icicle Tree (RIT), which resolves the critical problems in two existing visual designs, the icicle tree and sunburst, including inconsistent area encoding and poor node separation. Together, the three case studies demonstrate the general applicability of the Four Views Design concept.

Contents

List of Figures

List of Abbreviations

. . . . . **AI** – Artificial Intelligence.

. . . . . **CNN** – Convolutional Neural Network.

. . . . . **DE** – Differential Evolution.

. . . . . **GAI** – Generative AI.

. . . . . **GBDT** – Gradient Boosted Decision Trees.

. . . . . **HCAI** – Human-centered Artificial Intelligence.

. . . . . **HCI** – Human-computer Interaction.

. . . . . **HCML** – Human-centered Machine Learning.

. . . . . **HITL** – Human-in-the-loop.

. . . . . **k-NN** – k-Nearest Neighbors.

. . . . . **LLM** – Large Language Model. LLMs play a significant part in many text-based GAI systems, such as ChatGPT.

. . . . . **ML** – Machine Learning.

. . . . . **PCA** – Principal Component Analysis.

. . . . . **RBF** – Radial Basis Function.

. . . . . **SVM** – Support Vector Machines.

. . . . . **t-SNE** – t-distributed Stochastic Neighbor Embedding.

. . . . . **VA** – Visual Analytics, which is a branch of VIS. A typical VA approach, method, technique, tool, or workflow involves the combined use of human-centric processes (e.g., visualization and human-computer interaction) and machine-centric processes (e.g., statistics, algorithms, and ML models).

. . . . . **VIS** – Visualization and Visual Analytics. The abbreviation was used by the VIS community to encompass all VIS research. Before the three conferences were merged, VIS stands for VA (or VAST), InfoVis, and SciVis.

. . . . . **VIS4ML** – Visualization and Visual Analytics for Machine Learning. The abbreviation was first introduced by Sacha et al. [1].

1 Introduction

1.1 Motivation

Over the past decade, machine learning (ML) has achieved remarkable advances in domains such as natural language processing (NLP), image classification, and time series data analysis. However, with the widespread deployment of models in real-world scenarios, researchers have gradually recognized that relying solely on model architecture and parameter optimization is often insufficient to ensure model robustness and reliability in practical applications. Growing research evidence indicates that inconsistencies across different data spaces are one of the core causes of model performance degradation. These inconsistencies include differences that often exist between training data space and deployment data space, where a model may perform excellently in laboratory environments but experience performance drops in deployment environments.

In addition to the inconsistency between training and deployment data space, a further challenge arises from adversarial attacks, which introduce a second type of inconsistency between the normal application data space and the artificially designed adversarial data space. In many application domains, maliciously crafted inputs are designed to exploit model weaknesses, leading to incorrect predictions while remaining nearly indistinguishable to human observers. These attacks highlight

that even when training and deployment distributions align, ML models can remain highly vulnerable to targeted perturbations that shift inputs into an adversarial data space. Enhancing robustness against such threats requires methods that reveal and address this type of inconsistency.

These inconsistency problems exhibit different specific manifestations in practical applications, further complicating the process for ML users and developers to analyze models.

In this context, how to analyze, diagnose, and improve ML models from the perspective of training data quality has become a research question that demands answers. Existing research emphasizes experiment-driven optimization of model architectures, training parameters, or data preprocessing techniques, yet such efforts often overlook the inconsistencies across different data spaces. Some studies develop tools targeting specific problems, but often do not provide a workflow for analyzing data spaces. This limitation constrains in-depth research and restricts practical applications in industrial settings. More importantly, as ML application domains continue to expand, constructing an ML workflow that can effectively handle data space inconsistency problems across different data types (text, images, time series, etc.) and diverse application scenarios has become a challenge that needs to be addressed. These demands drive us to consider: is it possible to re-examine and improve existing ML workflows from the perspective of data space consistency analysis, thereby advancing the development of more robust, interpretable, and trustworthy ML systems?

Based on these motivations, this thesis explores methods to analyze, diagnose, and mitigate inconsistencies between different data spaces for the ML model. It further explores visualization-based solutions to support this analysis, providing both conceptual design and practical tools for building ML systems that are more robust and reliable in diverse application scenarios.

1.2 Research Questions and Objectives

This thesis focuses on diagnosing and mitigating such inconsistencies through a workflow-driven, visualization-supported approach.

To address the inconsistencies across different data spaces that frequently lead to model performance degradation, this thesis is structured around the following four research questions:

Q1. How can we formulate an ML workflow that explicitly addresses inconsistencies across different data spaces, thereby providing a method to analyze and understand the causes of model performance degradation?

Q2. How can we analyze and diagnose inconsistencies between the development data space and the deployment data space, in order to identify the factors leading to poor model performance in real-world applications?

Q3. How can we analyze and diagnose inconsistencies between the application-specific data space and the artificially designed adversarial data space, so as to enhance model robustness against malicious adversarial attacks?

Q4. Is the proposed ML workflow effective across different data types and tasks?

These research questions summarize the challenge of this thesis: how to identify, interpret, and address inconsistencies across different data spaces through an ML workflow and visualization supporting methods, thereby improving ML models' quality for their users. Addressing these questions requires both conceptual innovation and practical evaluation. Building on these research questions, the thesis defines five specific research objectives to formulate the problems into actionable research pathways.

O1. Propose a new ML workflow that can analyze and diagnose inconsistencies across different data spaces and help ML model users improve their ML models.

O2. Develop methods and tools to analyze and diagnose inconsistencies between the development data space and the deployment data space, and help ML model users alleviate their model performance degradation in real-world applications.

O3. Develop methods and tools to analyze and diagnose inconsistencies between the application-specific data space and the artificially designed adversarial

data space, and help ML model users to improve their model robustness against adversarial attacks.

O4. Demonstrate the effectiveness of the proposed ML workflow across multiple data modalities and tasks, showing its potential to improve ML models in these domains.

O5. Explore potential visualization support methods to analyze and interpret inconsistencies across different data spaces, thereby assisting ML model training and evaluation.

These research objectives establish the conceptual and methodological foundation of this thesis. They specify the concrete research contributions made to address data inconsistencies across different data spaces and highlight the practical value of the proposed approach across different tasks and data modalities. To ensure exposition of these objectives, this thesis is organized into a series of chapters that correspond to each objective. The next section outlines the overall structure of the thesis and presents how the subsequent chapters address each objective.

1.3 Thesis Structure

Building on these research objectives, the subsequent chapters progressively develop and evaluate the proposed ML workflow, with a focus on addressing inconsistencies across different data spaces. Each chapter is organized around one or more research objectives (**O1**–**O5**), collectively advancing the analysis and improvement of ML models under different data space conditions. The thesis first introduces the methodological framework for constructing the workflow (**O1**). It then develops methods and tools for analyzing inconsistencies between the development data space and the deployment data space, as well as between the application-specific data space and the artificially designed adversarial data space (**O2, O3**). The workflow is further demonstrated across multiple data types and task scenarios (**O4**). Meanwhile, in the process of tool development, visualization support is also provided to facilitate inconsistency analysis and human–computer interaction (**O5**). Through this structural arrangement, the thesis shows conceptual development,

methodological innovation, and practical evaluation, ensuring that each chapter addresses specific problems with concrete solutions and contributes to the goal of mitigating data space inconsistencies in ML model performance.

Chapter 2 reviews prior work from three interconnected perspectives: ML workflows, visualization techniques, and visualization support for ML workflows. It provides the background necessary for the subsequent chapters and outlines the context in which this thesis is situated.

Chapter 3 introduces the "Four Views Design", a VIS4ML workflow for testing, analyzing, and improving ML models. It outlines the core concepts and structural components of the workflow and contrasts it with traditional ML workflows. The chapter also describes the example of the design, providing the conceptual basis for the case studies presented in the subsequent chapters.

Chapter 4 presents the iGAiVA, developed as a real-world text classification system. The chapter describes how the system incorporates radial basis functions (RBF) and a novel text visualization method (Tag-treemap) to support the generation and selection of training data with the aid of large language models (LLMs). It also explains the design of the interactive visualization interface and the role of Four Views Design in a text classification workflow to address the inconsistency between the development data space and the deployment data space.

Chapter 5 presents the TA3 system, which applies the Four Views Design concept to adversarial attack testing in image classification. The chapter describes how the workflow is adapted to this context to address the inconsistency between the application-specific data space and the artificially designed adversarial data space, outlines the visualization support provided for evaluating model robustness under adversarial perturbations, and introduces the novel evaluation methods used in the study.

Chapter 6 introduces the DS4ML system, which was developed to investigate the use of data synthesis methods in time series classification. The chapter explains how the system applies the Four Views Design workflow, integrates visualization techniques for comparing different synthetic time-series datasets, and demonstrates

its use in a classification setting to address the inconsistency problem between the development data space and the deployment data space.

Chapter 7 introduces the Radial Icicle Tree (RIT), a visual design developed to overcome limitations of traditional icicle trees and sunburst plots related to node separation and area constancy. The chapter describes the design principles and demonstrates its application to time series data, using wearable device datasets as an example.

Chapter 8 concludes the thesis by summarizing the work presented in the previous chapters and reflecting on how they fit together within the thesis research objectives. It also discusses future research directions, including methods for filtering and selecting synthetic training data, the study of human behavior in ML workflows, and the design of user-centered support. Additional opportunities include extending visualization techniques, enhancing human-in-the-loop processes, and applying the workflow to broader domains. Potential future work will also extend adversarial testing by analyzing successful attack samples to reveal weak regions in training data and guide targeted enrichment to improve model robustness.

1.4 Related Papers

Chapter 4 contains a paper:

Yuanzhe Jin, Adrian Carrasco-Revilla, and Min Chen. "iGAiVA: Integrated Generative AI and Visual Analytics in a Machine Learning Workflow for Text Classification." arXiv preprint arXiv:2409.15848 (2024).

Chapter 5 contains a paper:

Yuanzhe Jin, and Min Chen. "TA3: Testing Against Adversarial Attacks on Machine Learning Models." arXiv preprint arXiv:2410.05334 (2024).

Chapter 7 contains a paper:

Yuanzhe Jin, Tim J. A. de Jong, Martijn Tennekes, and Min Chen. 2024. Radial Icicle Tree (RIT): Node Separation and Area Constancy. IEEE Transactions on Visualization and Computer Graphics 30, 1 (Jan. 2024), 251–261. https://doi.org/10.1109/TVCG.2023.3327178.

2
Literature Review

This chapter reviews the relevant literature from three interrelated perspectives: ML workflows, visualization methods, and VIS4ML workflows to situate the work within the existing research landscape. VIS stands for visualization and visual analytics. The abbreviation was used by the VIS community to encompass all VIS research. VIS4ML stands for Visualization and Visual Analytics for Machine Learning. The abbreviation was first introduced by Sacha et al. [1].

2.1 Traditional ML Workflows and Challenges

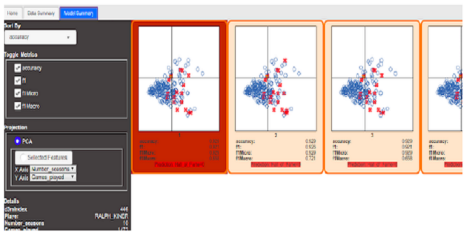
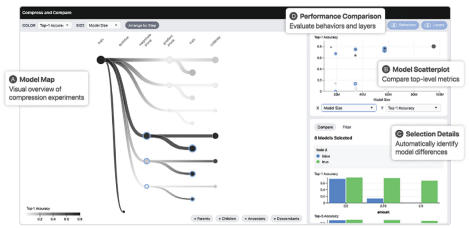
ML workflow refers to the entire process from raw data to ML model deployment [2]. This process builds, trains, and evaluates models to solve practical problems.

We searched major venues in visualization and machine learning (e.g., IEEE VIS, TVCG, CHI, and related ML conferences) using keywords such as "interactive machine learning", "visual analytics for ML", and "ML workflow". Our selection of papers is based on the following three criteria: first, the paper belongs to the ML application domain; second, the paper needs to have a user interface; and third, the paper contains an ML workflow. The final 15 papers were selected as they collectively cover diverse ML models (e.g., decision trees, DNNs, GNNs), different

application domains (e.g., healthcare, industry, education), and various user roles (e.g., ML developers, domain experts, non-expert users)

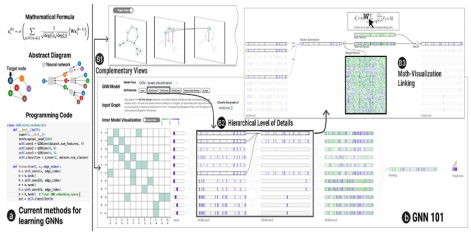
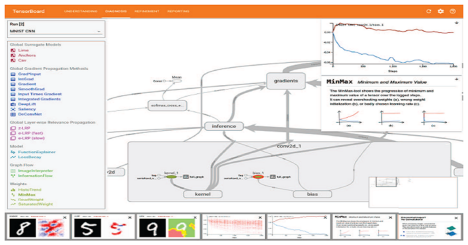
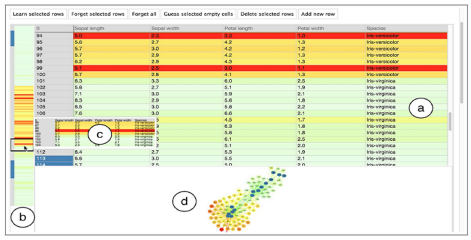
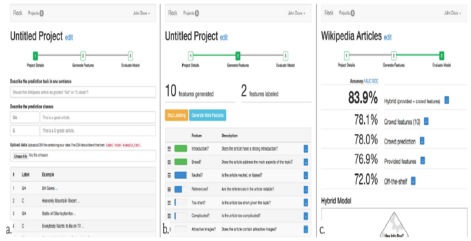
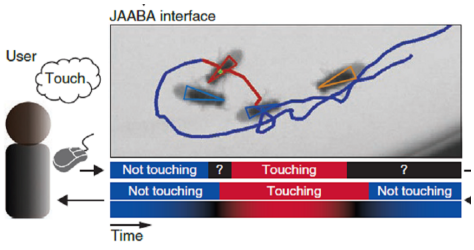
In Table 2.1, we list these ML papers together with their corresponding user interfaces. Among them, P-1 [3], P-5 [4], P-9 [5], and P-12 [6] emphasized the use of interactive visual analytics to support model exploration, hypothesis testing, and model validation, while P-2 [7] and P-10 [8] focused on interactive interface design for automated machine learning or model comparison. Some papers highlighted interpretability and debugging processes for specific models, such as visual learning for GNNs [9], interactive analysis for decision trees [10], and visual explanations for GAM models [11]. In domain-specific applications, several studies designed task-oriented workflows, such as breast cancer screening [12], industrial-scale deep neural network visualization [13], and animal behavior annotation [14]. Other works proposed different modes of human–machine collaboration, such as interactive machine learning in spreadsheet environments [15] and hybrid crowdsourcing approaches [16].

Table 2.1: UI and Keywords of different ML tasks papers. We have organized the keywords for each paper in the table, covering the following five aspects: research domain, model type, data type, application scenario, and user group. These keywords are listed sequentially in the keywords column.

Paper ID	UI	Keywords
P-1 [3]		Visual Analytics Multiple(SVM, k-NN, etc.) Multimodal Data Academia Data Scientist and Analyst
P-2 [7]		Visual Analytics DNN Image Classification, Text Industry ML Scientist

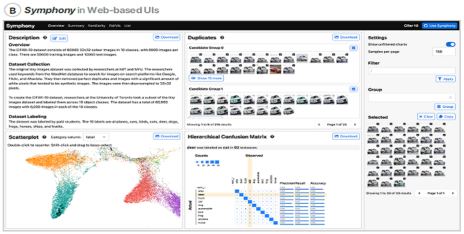
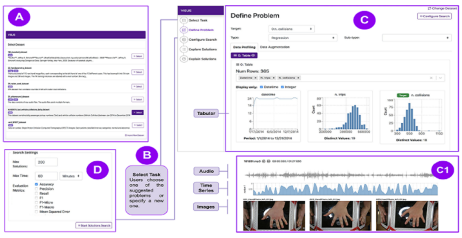
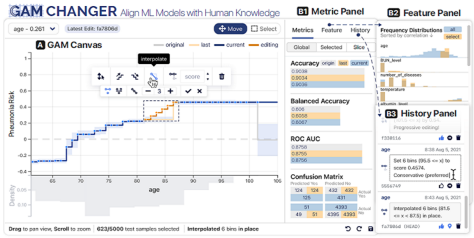
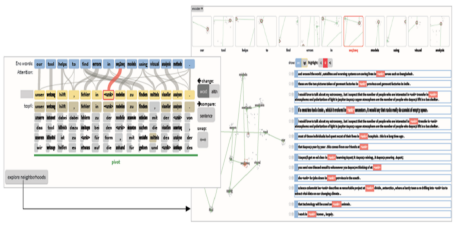
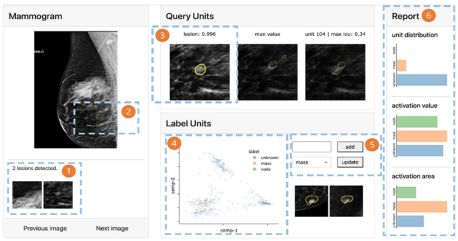
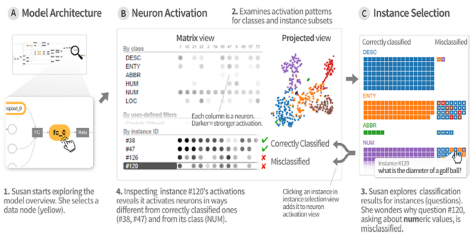
Continued on next page

Table 2.1 – continued from previous page

Paper ID	UI	Keywords
P-3 [9]		HCI GNN Graph Data Academia GNN Learner, Teacher
P-4 [17]		Visual Analytics LLMs Image Data Industry ML Model Developer
P-5 [4]		Explainable AI DNN, CNN Image Academia ML Developer & User
P-6 [15]		HCI k-NN Tabular Data Academia Non-professional ML User
P-7 [16]		Crowdsourced ML Random Forest Text, Image, Webpage Academia ML Model User
P-8 [14]		Animal Behavior Analysis Boosting (Tree-based) Video Data Academia Biologist

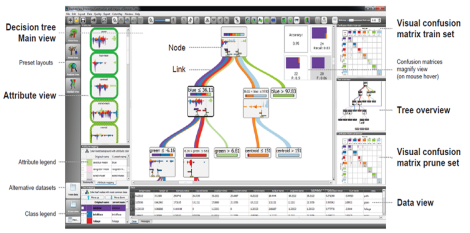
Continued on next page

Table 2.1 – continued from previous page

Paper ID	UI	Keywords
P-9 [5]		ML System HCI DNN Image, Video Industry ML Model User
P-10 [8]		Visual Analytics, AutoML Auto-sklearn, AlphaD3M Tabular, Time Series, Image Academia NYC Transportation Expert
P-11 [11]		Explainable AI, HCI GAM Tabular Data Academia Data Scientist, Doctor
P-12 [6]		ML Model Visualization Seq2seq Model Text Academia Model Developer
P-13 [12]		Medical Data Explainability VGG-16(DNN) Image Academia Radiologist
P-14 [13]		DNN Model Visualization CNN Text, Image Academia ML Engineer, Data Scientist

Continued on next page

Table 2.1 – continued from previous page

Paper ID	UI	Keywords
P-15 [10]		Interactive ML Visualization Decision Tree Tabular Data Academia Medical Diagnostician, Data Analyst

It can also be observed that the interface designs in these studies adhere to the traditional ML workflow and demonstrate differentiated interactions and innovations at the level of specific tasks and models. Some works emphasize model interpretability and debugging support, while others highlight task-specific requirements or modes of human–AI collaboration.

The majority of these papers' UI design did not include a synthetic data generation stage, but instead relied mainly on existing benchmark datasets or domain-specific real-world datasets [5]. Only a small number of studies explored aspects related to data generation or augmentation, such as semantic error analysis and feature generation [17]. This review suggests that current research has been primarily concentrated on model training, interpretation, and interaction, while the introduction and integration of synthetic data remains relatively underexplored.

The other challenges identified in current ML workflows and the ways in which this thesis contributes to addressing them will be discussed in detail in Chapter 3.

Despite this limited exploration of synthetic data, the importance of data enrichment through synthesis is growing rapidly. Synthetic data can provide a flexible and scalable means of expanding datasets, enabling more comprehensive model evaluation and supporting scenarios where real data are scarce or unavailable. To better understand these opportunities and challenges, we review representative data synthesis methods that have been proposed for training enrichment.

2.1.1 Data Enrichment Methods for Training

Although synthetic data generation is not typically included as a formal stage in traditional ML workflows, it has been studied as a standalone research direction

for data enrichment. In this section, we review some data synthesis methods to provide background for the new workflow proposed in this thesis.

Data Synthesis Methods

Broadly speaking, data enrichment includes the preparation of original data and synthetic data. Since the main research content of this thesis is the enrichment of training data, more attention is paid to the content of synthetic data.

Data synthesis has become an increasingly important area of research in recent years, with various methods and approaches being developed. There are methods for synthesizing data based on existing time series. For one-dimensional data, these methods can be categorized into two main classes: statistical methods and ML-based methods. Within statistical methods, they can be further divided into jittering [18], scaling [19], interpolation [20, 21], warping [22, 23], and permutation [24]. In fact, within statistical methods, each approach has several variants; we have only listed some of the relevant methods here. Using traditional methods allows for relatively quick data synthesis of existing training data. Another rapidly developing approach in recent years is ML-based data synthesis. This category mainly consists of GAN-based synthesis methods, with TimeGAN [25] being a primary representative, along with its variants [26, 27].

Similarly, diffusion models based on machine learning have been proposed [28]. These models generate high-quality synthetic time series data by gradually adding noise and learning the reverse denoising process. Diffusion-TS [29] integrates the Transformer architecture to achieve high-quality multivariate time series generation. TS-Diffusion [30] is capable of handling complex time series data with irregular sampling, missing values, and high-dimensional features. TimeLDM [31] encodes time series into a latent space and applies the diffusion process within that space for data generation. BioDiffusion [32] focuses on the generation of biomedical signals by synthesizing data using diffusion models. These methods demonstrate the versatility of diffusion models in time series data synthesis.

Building upon various methods, researchers have further developed some synthetic data frameworks. Lu et al. [33] comprehensively reviewed ML techniques for synthetic data generation, highlighting the rapid progress in this field. Restat et al. [34] introduced GouDa, a framework for generating universal datasets, indicating the growing demand for versatile data synthesis tools. Ichim [35] proposed a quantile-based bootstrapping method for generating continuous synthetic data, offering a new approach to creating realistic datasets. Mannino et al. [36] developed Synner, a system aimed at generating realistic synthetic data, further emphasizing the importance of data fidelity in the synthesis process. In specific applications, Rhys et al. [37] explored the use of automated search techniques to generate test data for MATLAB, demonstrating the potential of data synthesis in software testing and validation. Skander et al. [38] applied data synthesis concepts to manufacturing processes, showcasing the wide applicability of these techniques across different domains. Hittmeir et al. [39] studied the utility of synthetic data, providing insights into its effectiveness and limitations. Both Nikolenko [40] and Chen [41] authored comprehensive works on the use of synthetic data in deep learning and machine learning, respectively, emphasizing the growing importance of data synthesis systems in advancing AI technologies.

Data Synthesis Applications

Data synthesis has extensive applications in various domains, including image, audio, video, and even specialized fields like radiation pattern analysis [42]. The utilization of data synthesis in these domains is widespread and diverse. In the realm of image processing and computer vision, synthetic images can be found both in training and evaluating algorithms. Tsirikoglou et al. [43] provide a comprehensive survey of image synthesis methods for visual ML, highlighting the importance of synthetic data in this field. Synthetic videos find applications in tasks such as video analysis, motion capture, and video enhancement. They enable the simulation of diverse actions, scenes, and perspectives. Ilisescu et al. [44] introduced responsive action-based video synthesis, while Whiting et al. [45] explored methods for generating

synthetic video datasets. These studies demonstrate the potential of synthetic data in creating realistic and diverse content for various applications. In the audio domain, Simbelis et al. [46] explored synthesis in audiovisual contexts, while Schwarz et al. [47] developed a system for data-driven concatenative sound synthesis. Beyond traditional media, data synthesis has found applications in specialized fields. Kim et al. [42] proposed a deep learning-based approach for radiation pattern synthesis of array antennas, showcasing the potential of data synthesis in engineering and telecommunications. Wang et al. [48] introduced Falx, a synthesis-powered visualization authoring tool, demonstrating the application of data synthesis in creating data visualizations. Additionally, Whiting et al. [49] explored the creation of realistic, scenario-based synthetic data, emphasizing the importance of data synthesis in various research and application contexts.

As data synthesis tools become more prevalent, the design of user interfaces for these tools has gained importance. Peng et al. [50] developed a tool for generating synthetic data, focusing on usability and accessibility. Hwang et al. [51] introduced ElderSim, a synthetic data generation platform specifically designed for human action recognition in eldercare applications, demonstrating the potential for domain-specific synthesis tools. Latif et al. [52] presented Kori, an interactive system for synthesizing text and charts in data documents, highlighting the importance of user interaction in the synthesis process. Mannino et al. [53] explored the challenge of generating synthetic data that looks real, emphasizing the importance of visual fidelity in synthetic data generation tools.

Upon reviewing the existing research, it can be observed that previous studies primarily aim to enhance the underlying algorithms for data synthesis. However, there is a notable lack of exploration and application of comparative analysis among synthesized datasets. To further advance the field, it is necessary to delve into and apply comparative analysis among synthesized datasets. This approach could provide valuable insights into the strengths and weaknesses of different synthesis methods, as well as inform the development of more effective data synthesis techniques. Although many synthesis methods exist, there is little support for

comparative analysis and visualization of different synthetic datasets. This gap limits the ability of ML developers to select or refine suitable synthesis strategies. We show a solution to this question in Chapter 6 by introducing ideas of visualization and human-computer interaction into the comparison of synthetic data.

2.1.2 Representative ML Models for Visualization Supporting

ML models serve as the core computational engines in various analytical workflows, each offering distinct characteristics in terms of interpretability, performance, and applicability. Understanding these models is necessary for developing effective visualization support systems, as different model architectures present unique challenges and opportunities for visualization design. The selection of appropriate models directly influences both the analytical outcomes and the requirements for visual interpretation tools.

ML Models Overview

ML models can be broadly categorized based on their learning paradigms, structural complexity, and interpretability characteristics [54, 55]. Supervised learning models form the foundation of many predictive systems, including linear models (linear regression, logistic regression) [56], tree-based methods (decision trees, random forests) [57], ensemble methods (gradient boosting, AdaBoost) [58, 59], and neural networks [60]. These models learn from labeled training data to make predictions on new instances.

Unsupervised learning models focus on discovering hidden patterns in data without explicit target labels [61]. Principal Component Analysis (PCA) and clustering algorithms (k-means, hierarchical clustering) are commonly used for dimensionality reduction and data exploration [62, 63]. These models are valuable in exploratory data analysis and feature engineering phases of ML workflows [64]. Instance-based learning models, such as k-Nearest Neighbors (k-NN) and Support Vector Machines (SVM), make predictions based on similarity measures or geometric relationships in the feature space [65, 66]. These models often provide intuitive explanations

through their reliance on specific training instances or decision boundaries [67]. The interpretability spectrum of ML models ranges from highly transparent models like linear regression and decision trees to complex black-box models like deep neural networks [68, 69]. This spectrum significantly impacts the design requirements for visualization support systems [70]. Transparent models may require simpler visualization techniques focused on parameter interpretation, while black-box models necessitate sophisticated explanation and exploration tools [71]. Ensemble methods combine multiple base models to improve predictive performance, but this comes at the cost of increased complexity in interpretation [72, 73]. Gradient boosting machines and voting classifiers fall into this category [74].

Different types of ML models exhibit differences in terms of structural complexity, support for visual interaction, and sensitivity to training data quality [69]. These differences largely determine their suitability for integration into visual analytics systems. For example, interpretable models are often better suited for human-in-the-loop workflows, enabling rapid feedback and iterative refinement [75], whereas more complex models typically require dedicated visualization and explanation techniques to support understanding.

Decision Tree Models

As one of the widely used ML models, a decision tree is easy to understand [76]. Users without much ML experience can appreciate different decision flows by observing nodes and edges in the decision tree. In ML testing, ML developers can evaluate a decision model by observing the data flow generated when a model processes the input data.

The earliest decision tree algorithm is Hunt's Algorithm [77]. Built on the basic concept of Hunt's Algorithm, several software systems were developed for training decision trees, including ID3 [76], C4.5 [78], and CART [79]. Some methods, such as GBDT [80], LightGBM [81], and XGBoost tree [74], have been introduced to help decision tree building processes. They use different gradient-boosting techniques to improve efficiency and accuracy in ML. In the context of security and trustworthy,

Calzavar et al. [82] studied the problem of potential discrimination due to some sensitive features, and proposed an explainable approach for verifying the global fairness of tree-based classifiers.

In Chapter 5, the CART algorithm in scikit-learn [83] was used to train our ML models that are to be tested with TA3. We assume that ML developers would test their models in a white-box manner. Training decision tree models using scikit-learn allows us to obtain the internal structures of the ML models, which can be visualized during the testing process. In Chapter 6, we also used similar models to validate the results on synthetic data. In Chapter 4, we used CatBoost [84], an open-source ML library developed by Yandex. CatBoost is based on the Gradient Boosted Decision Trees (GBDT) algorithm [85]. We chose CatBoost because it supports categorical features without the need for manual encoding. The collaborator with us in Inetum Spain chose this model also because its open-source nature helps engineers analyze it.

2.1.3 ML Model Challenges and Testing

Adversarial Attacks Methods and Defenses

When ML models are deployed in applications, some small perturbations in input data that humans cannot see may have the ability to affect the accuracy of an ML model [86]. This phenomenon was discovered by the Fast Gradient Signed Method (FGSM) proposed by Goodfellow et al. in 2014 [87]. This method of perturbing input data indicates that the ML models are vulnerable to adversarial attacks.

Following that, researchers proposed several derivative algorithms based on FGSM, such as RFGSM [88], I-FGSM [89], DI2-FGSM [90], and so on, to make FGSM more efficient in black-box attacks or improve the attack success rate in white-box attacks. For example, by adding randomness or momentum, FGSM can evade some model defenses. In addition to FGSM, methods such as universal adversarial perturbations [91], foolbox [92], and poison frogs [93] can also interfere with the expected working results of ML models. These algorithms are referred to as white-box attacks, as they introduce perturbations to input data based on the known model structures, parameters, or both. There are also adversarial attack

algorithms that can introduce perturbations without the knowledge of the ML models concerned (black box) [94].

"One Pixel attack" is a group of commonly studied adversarial attack algorithms. In each attack, such an algorithm selectively modifies a single pixel in an input image, often successfully causing an ML model to fail. Su et al. proposed one such algorithm [95], which optimizes the selection of a pixel using differential evolution (DE) [96]. Kondratyuk provided open-source code for the algorithm [97], which is used in this work.

Recent studies have highlighted various aspects of adversarial attacks in ML models. For instance, Garcia et al. [98] demonstrated that dimension reduction techniques can identify on-manifold adversarial examples. Hong et al. [99] emphasized that while optimizing models for on-device efficiency, the increase in adversarial vulnerability cannot be overlooked. Apruzzese et al. [100] examined the gap between theoretical research and practical adversarial ML and pointed out that many real-world attackers do not rely on gradient computation, necessitating the need for more practical defense mechanisms. Croce et al. [101] introduced prompt-agnostic adversarial attacks on segmentation models. Debenedetti et al. [102] presented techniques for evading black-box classifiers, demonstrating that effective attacks can be crafted without significant computational resources.

These findings underscore the importance of the evaluation and testing of ML models against adversarial attacks to ensure their robustness in both theoretical and practical applications.

In recent years, more research effort has been placed on defending models against adversarial attacks. For example, Joshi et al. [103] studied the vulnerability of x-Vector-based speech recognition systems under various adversarial attacks and evaluated multiple preprocessing defense methods that do not require adversarial sample training. Zheng et al. [104] proposed a boundary differential privacy method, which prevents model parameter leakage by adding noise to the prediction responses, protecting the model from query-based attacks.

In testing adversarial attacks, Li et al. [105] proposed a training framework for enhancing the robustness of malware detection by using a mixture of adversarial attacks to provide training instances. Ali et al. [106] studied the vulnerability of deep forest models when facing adversarial attacks and discovered that such models are susceptible to cross-technology transfer attacks. Ding et al. [107] used several visualization plots to analyze the image characteristics of clean and adversarial imagery data. Jin et al. [108] conducted a comparative study on adversarial attacks and backdoor attacks and used the analytical findings to improve the detection of backdoor attacks.

The conventional workflow primarily focuses on two aspects: finding successful attack samples and calculating statistical measures [89]. While these objectives are important, they present several limitations.

Traditionally, the testing workflow of ML models has always followed a relatively linear and statistically focused approach [109, 110]. These traditional methods typically focus on generating adversarial examples and measuring their effectiveness through statistical indicators [109]. In addition, most current research aims to automate the testing process [111]. However, due to the lack of comprehensive consideration of various complex factors in the testing process, automated testing often overlooks the underlying reasons behind model behavior. Moreover, when dealing with adversarial samples, pure automation makes it difficult to simulate the diverse attack strategies of human attackers [112], resulting in limitations in the test results. Previous tools usually lack comprehensive HITL support and visualization functions, and thus cannot enhance the testing process.

Traditional approaches often lack systematic parameter exploration capabilities [113]. ML developers typically modify attack parameters through trial and error, without structured support for parameter space exploration [87]. This can lead to inefficient testing processes and potentially missed vulnerabilities [114]. The selection of target data objects in conventional workflows is often arbitrary or based on simple random sampling [115]. This approach may fail to identify particularly vulnerable data types or patterns that could be more susceptible to adversarial

attacks [116]. Traditional workflows typically lack comprehensive visualization support for model behavior analysis [117]. While basic accuracy metrics and loss curves are common, they often fail to provide insights into how different parts of the model respond to adversarial inputs [118].

Previous studies have shown that ML models are often not robust enough against adversarial attacks. Most existing research has focused on algorithmic approaches or automated methods to enhance and assess model security, often overlooking the role of human involvement in the training and deployment processes. Therefore, ML model users need to find a new way to evaluate the vulnerability of their models. In Chapter 5, we focus on the design and development of a software tool that enables such testing. This tool integrates statistical analysis, algorithmic modules, and human-computer interaction features, aiming to incorporate the tester's domain knowledge into the evaluation process. By doing so, it enhances the model's robustness against adversarial attacks throughout the entire ML pipeline—from training to evaluation—in a more comprehensive and human-centered manner.

Adversarial Attacks and Perturbations in Invariance Testing

This section is part of a literature review I collaborated on with Dr. Zukang Liao during my doctoral studies, focusing on the role of adversarial attacks and perturbations like noise in invariance testing. Invariance testing leverages adversarial attacks to evaluate the extent to which models violate certain prior invariance assumptions. This approach helps researchers gain a more comprehensive understanding of a model's generalization capabilities and potential vulnerabilities, thereby providing guidance for improving the robustness and reliability of models. Moreover, it can reveal potential weaknesses beyond the test data distribution, driving deeper insights into the behavior of deep learning models.

The following content will show some common adversarial attack methods and perturbations used in invariance testing.

Gaussian noise. In nature, Gaussian noise is a kind of random and regular data noise, which can be used to simulate the uncertainty and noise of data in practical

applications [119, 120]. Gaussian noise invariance testing evaluates the sensitivity of the model to random disturbances in the data [121].

Adversarial examples/attacks. An adversarial attack (example) refers to a deliberately synthesized/generated input that can mislead a trained model. Adversarial training injects these adversarial examples during the training process to boost the robustness of the trained ML models. Similarly to the Gaussian noise, the generated adversarial noise can be treated as a variable as a whole, and invariance testing can be conducted.

The adversarial attack has different implementation principles [122–124], most of which are to add noise that is not easily observed by people to the original data to affect the prediction results of the machine learning model. Adversarial attacks can be divided into gradient-based attacks and non-gradient attacks from the attack method [125]. From the perspective of invariance testing, both attack methods change the original data to a certain extent by adding noise or interference, which makes the prediction of the ML model deviate.

In 2014, the Fast Gradient Sign Method (FGSM) was first proposed by Goodfellow et al., which explained the basic principles of adversarial examples and proved that the state of high-dimensional networks is linear. Research shows that the high-dimensional nonlinearity of networks is responsible for adversarial attacks in networks [87]. After their work, other researchers successively proposed many derivative algorithms based on FGSM, such as RFGSM [88], NI-FGSM [126], M-DI2-FGSM [90], etc., making FGSM carry out more efficient black-box attacks and improving the attack success rate of its white-box attacks. In addition, Projected Gradient Descent (PGD) [114] is an iterative optimization algorithm that generates adversarial examples by continuously optimizing the loss function. Carlini and Wagner (C&W) Attack [127] is an effective adversarial example generation method, which generates adversarial examples by solving an optimization problem. DeepFool [128] is a method of using linear programming to construct adversarial examples, which can quickly generate adversarial examples. For different adversarial

attacks, Yao et al. compared the performance of various adversarial attacks on different ML models [129].

Adversarial training for adversarial attacks can improve the pre-resistance of ML models against adversarial attacks to a certain extent without additional cost [130]. The paper [131] reviews recent progress in adversarial robust training using novel taxonomies. One thing to note is that adversarial training does not always have a positive effect on the improvement of model invariance testing. In adversarial attacks, perturbation-based adversarial examples are not robust enough to achieve general robustness in some cases. A more precise definition is needed for general robustness [132]. In the paper [133], some ML models are improved in rotation invariance by gradually increasing the training enhancement of rotation, but while doing so, their adversarial robustness does not improve, or worse, it can even drop significantly on datasets like MNIST. In his article, Bühlmann analyzes the relationship between invariance, causality, and robustness, using statistical methods for more robust and better "causality-oriented" explanations [134].

The relationship between robustness against adversarial attacks and other types of invariance qualities also drew some researchers' attention. Gu et al. [135] investigated the trade-off between adversarial robustness and accuracy in neural networks by adding additive noises to either the input layer or all hidden layers. They studied the recovery of adversarial examples and misclassification of clean examples as the amount of noise varied (i.e., invariance testing), to understand the model's robustness [135]. Similarly, research [136] demonstrates fundamental trade-offs between robustness levels of two different types of adversarial examples. They find that defending against sensitivity-based attacks actively compromised model accuracy against invariance-based attacks. Yu et al. [137] found that affine transformation invariance qualities are closely related to adversarial robustness [138, 139]. However, being invariant to affine transformations does not guarantee robustness against all types of adversarial attacks [137], as there are other types of adversarial attacks that do not correspond to affine transformations.

Watermark attacks (noise/cropping). Watermarks are divided into traditional physical watermarks and digital watermarks [140]. A digital watermark is a marker covertly embedded into a noise-resistant signal such as audio, video, or image data [141]. It is often used to identify copyright ownership of such data. R.G. van Schyndel proposed digital watermarking in December 1994 [142]. Like traditional physical watermarking, digital watermarking is usually only perceptible under certain conditions, and since the digital copy of data is identical to the original data, digital watermarking is a passive protection tool [143]. In the Internet era, people use software watermarks to embed some secret information in the software as a sign of the copyright owner of the software to protect the software [144]. For an ML model or algorithm that is designed to encode/decode the watermarks for copyright protection, its performance should not be affected by any sort of attacks, e.g., noise or cropping. For example, when an image with a watermark is cropped or added with some noise, the ML algorithm/model should still be able to identify that it is a watermarked image. For example, Chou et al. [145] introduced a transform-invariant public fragile watermarking scheme for 3D object authentication. The proposed scheme embeds the watermark into 3D objects, making the 3D objects invariant to translation, rotation, and uniform scaling operations.

For generative ML models, generated outputs (images/videos/3D objects) can also be watermarked. Given a generative ML model to be protected, an invisible watermark was added to its outputs. When another ML model tries learning from its input-output pairs, the watermarks will be learned instead of the real semantic contents. For these generative ML models, invariance testing is important to guarantee the generated outputs' watermarks will still be effective under different types of attacks, e.g., affine transformation/noise/cropping. For example, Wu et al. [146] used a combined loss function so that the trained ML models are able to generate watermarked images.

In addition to traditional watermark attacks, e.g., cropping and noise, other more advanced techniques can be used to test the invariance qualities of ML models that involve encoding and decoding watermarks. For example, Wdnet [147] introduced

a watermark-decomposition network for visible watermark removal, as well as its anti-attack techniques to boost the invariance qualities in watermark attacks. They evaluated the proposed method on the Large-scale Visible Watermark Dataset (LVW) [148], which contained 60K watermarked images with 80 different types of added watermarks. The authors also introduced a newly collected dataset, CLWD (Colored Large-scale Watermark Dataset), which includes 60K watermarked images with 160 colored watermarks for training and 10K watermarked images with 40 colored watermarks for testing. Similarly, REFIT [149] introduced a unified watermark removal framework for deep learning systems using only a limited amount of data. The authors used an elastic weight consolidation (EWC) algorithm adaptation and unlabeled data augmentation to improve effectiveness. Evaluation of REFIT against diverse watermark embedding schemes. They used their watermark removal technique to test the watermark attack invariance qualities of existing algorithms, including pattern-based techniques [150–152], embedding samples drawn from other data sources as the watermarks [153], exponential weighting [154], and adversarial frontier stitching [155]. They found that all of the ML algorithms were vulnerable to watermark attacks.

2.2 Visualization Methods for Supporting ML Models

While algorithmic advances have increased the availability of synthetic data generation tools, they often fail to enable deeper analytical exploration. As synthetic datasets become increasingly diverse and complex, there is a growing need for visualization mechanisms that allow users to meaningfully interpret, compare, and interact with the generated data. This requires shifting the focus from mere data synthesis to how to communicate, perceive, and evaluate this data through visualization.

2.2.1 Human-Centered AI and UI Design

Humans are increasingly exposed to artificial intelligence (AI) and machine learning (ML) techniques. Human-centered AI is an approach for making AI part of a larger system participated in and managed by humans [156]. It features a high level of human control, while computer automation improves human performance [157]. In ML, although the machine may seem to do much of the work, there is a lot of human effort. Ramos et al. argued for more human-computer interactions (HCI) in ML processes [158]. The establishment of the HCAI (Human-centered Artificial Intelligence) enables a pathway of AI toward human goals with more human participation in AI [159]. On this basis, the concept of HCML (human-centered Machine Learning) has been developed [160]. While HCI refers to the broader discipline of designing interactive systems, HCAI emphasizes the role of human values and control in AI systems, and HCML can be viewed as a more specific instantiation of these principles within ML workflows.

In designing TA3, we drew inspiration from existing tools supporting HCAI and HCML. For instance, Tam et al. analyzed human contributions via interactive visualization in ML workflows [161], while McNutt and Chugh proposed pre-design templates for modular editing [162]. Interfaces for "fair" ML were explored by Yan et al. and Ge et al., focusing on user-driven input modifications [163, 164]. Zhang et al. contributed intelligent user interfaces for active learning and quality assurance in ML predictions [165, 166]. Additionally, Meng et al. developed VADAF for visualizing training dynamics [167], and Segura et al. supported collaborative sensemaking through visual narratives [168]. Research by Chen et al. and Li et al. emphasized customizing interfaces and enhancing interactivity to improve user engagement [169, 170]. Furthermore, studies by Yan et al. on user traits [171] and Nakao on bias in interfaces [172] underscore the importance of personalization and inclusivity in interface design. Additionally, we find studies that examine the correlation between decision-making style [173] and user interface interactions, exploring how an individual's decision-making approach can impact their interaction preferences and experiences within a given interface. Through this comprehensive exploration of

user interface design and its intricate connections with human interactions, we aim to contribute to the advancement and optimization of user-centered applications, ultimately enriching user experiences.

Researchers made several attempts at the design process. VINS [174] takes UI images as input to retrieve visually similar design examples, allowing interface designers to better understand the needs of the design. Several interactive tools have been developed to support different tasks, such as processing medical images [175], transferring text data into a graph [176], and time-series data [177]. There are also user interface (UI) designs for supporting model developers and users, e.g., in natural language processing (NLP) [178], face recognition [179], computer vision [180, 181], and real-time games [182]. Some designs consider not only the tasks in ML workflows but also the people who use interactive visualization facilities (e.g., [183]). VIS4ML is an ontology that highlights various steps in ML workflows where visualization has been provided or can be provided to assist ML developers [184]. Many researchers have proposed methods for visualizing various ML models [13, 184–189]. The most frequently mentioned ML models are deep neural networks [190–192]. As one of the most widely used developing tools, TensorBoard in TensorFlow is an interactive tool [193] for visualizing and editing neural network structures. Through interactive visualization, users can observe the data flowing running through the model.

Since many ML models [194] are often considered difficult to understand, there are still many challenges for humans to apprehend the behavior and performance of ML models. In Chapter 5, we focus on the family of decision tree models. In the field of visualization, researchers have proposed many tree visualization methods. For example, GoTree outlined a grammar [195] for visualizing tree structures. In response to the modification of decision trees, researchers propose a scalable decision tree visualization optimization method [10]. TreePOD [196] and another research [197] explored some of the decision tree generation processes that can be interactively managed by users. For visualizing the training process, BOOSTVis uses the width of the edge to encode the amount of data passing through a decision path [198]. BOOSTVis helps users analyze the training process visually to improve

training efficiency. We adopted their design of decision tree visualization in our work. RIT in Chapter 7, the design of which was motivated by an ML application, provides a mathematically rigorous visual representation for depicting statistical values associated with tree nodes [199].

2.2.2 Visual Designs for ML Applications

Machine learning applications employ a wide variety of visual design strategies. Surveys of VIS4ML papers (e.g., [200, 201]) show that visualization designs can be broadly grouped into categories. Table 2.2 provides a brief summary of representative visualization types used in ML research and applications.

Table 2.2: Representative Visualization Methods in ML Applications

Category	Use Scenarios	Example Plots
High-dimensional Data	Feature Exploration, Dimensionality Reduction	Scatter Plot, Parallel Coordinate
Tree and Hierarchical Structure	Decision Tree, Hierarchical Clustering, Taxonomy Visualization	Node-link Diagram, Treemap, Icicle Plot, Sunburst
Network and Graph	Neural Network, Knowledge Graph, Model Relationship	Adjacency Matrix, Arc Diagram
Sequential and Temporal Pattern	Time Series Analysis, Sequence Modeling, Training Dynamic	Line Chart, Stream Graph, Horizon Graph, Timeline Plot
Matrix and Tabular	Confusion Matrix, Attention Weight, Correlation Analysis	Heatmap, Matrix Plot, Sortable Table
Statistical and Distribution	Data Distribution, Uncertainty, Model Performance	Histogram, Box Plot, Violin Plot, Error Bar

By reviewing the broader design space, we further focus on two representative aspects in visualization research. The following discussion highlights tree visualizations as a typical form of hierarchical design, and graphics integrity as a key issue that spans multiple types of visualization.

Tree Visualization is a subarea of graph/network visualization, but it has engendered a large collection of visual designs, many of which are very different from visual designs for graph and network data [202–205]. A web platform, treevis.net, provides an extensive gallery of tree-structured data visualization methods (TSDV) [206]. The TSDV methods have been classified as implicit, explicit, or hybrid representations, according to how different components of a

tree structure are shown [207]. In explicit TSDVs, a tree-structured dataset is represented by a node-link diagram, e.g., [208–217].

Both *icicle* and *sunburst* trees are implicit TSDV methods because the edges between nodes are not shown explicitly. Schulz et al. provided a comprehensive survey on implicit TSDV methods [207].

The development of icicle tree visualization can be traced back to Kruskal and Landwehr [218] (1983), and Kleiner and Hartigan's castle tree may also be considered as an earlier variant [219]. Its noticeable variants include triangular aggregate treemap [220], cushioned icicle plot [221], and space-reclaiming icicle plots [222]. In particular, van de Wetering introduced a deformed icicle tree layout to reclaim some space freed by subtrees that do not reach certain depths. This improves the efficiency of space usage, especially at deeper hierarchical levels.

The development of sunburst tree visualization can be traced back to Paul Otlet's depiction of universal decimal classification [223] (1901). The hierarchical sector chart by American Society of Mechanical Engineers [224] (1939), Johnson's polar treemap [225] (1993), and many subsequent variants may also have influenced the design of sunburst trees, though they do not restrict the placement of each node strictly within a radius range according to the depth of the node. The number of "cascading" design variants that introduced such a restriction, including those by Andrews and Heidegger [226], Chuah [220], and Stasko and Zhang [227], led to the adoption of sunburst trees as a popular method. Other noticeable variants include disk tree [228], 3D multivariate sunburst plot [229], hyperbolic wheel [230], and sundown chart [231]. Yang et al. developed a visualization tool based on sunburst tree visualization for reconfiguration and manipulating hierarchical data [232].

Through empirical studies, Cawthon and Moere, and Muramalla et al. found that the user performance with the icicle tree is better than the sunburst [233, 234]. For multivariate variants of both designs, Zheng and Sadlo found that the sunburst performed better [235], presumably due to user acceptance. Users also found sunburst aesthetically more pleasing [233, 234].

Graphics Integrity is a concept proposed and articulated by Edwards Tufte [236], and it has been widely supported by VIS researchers and practitioners. For example, in IEEE VIS conferences, there have been regular events called "VisLies", where participants exhibit visualization imagery or techniques that depict data in misleading or confusing ways. Kindlmann and Scheidegger presented an algebraic framework to aid the formalization of this concept mathematically and defined three principles for visual encoding, namely "representation invariance", "unambiguous data depiction", and "visual-data correspondence" [237]. Correll et al. evidenced the negative impact of "unfaithful" visual representations [238].

In psychology, a number of empirical studies have been conducted to study the perception of different geometric attributes of shapes (e.g., [239–242]). In visualization, Skau and Kosara studied different visual cues used to read pie and donut charts and their impact on the accuracy in perceiving the encoded data values [243, 244]. Qi and Jing examined the accuracy of length and area perception in shape-based data encoding [245].

Meanwhile, a number of empirical studies have shown that faithfully encoded visual representations may lead to an incorrect perception of the data being encoded. For example, Alberts Szafir discovered that the same color could be perceived differently when the host objects change sizes [246]. Schloss et al. discovered that different task performances resulted from colormaps consisting of the same set of colors but mapping concepts to colors differently [247]. These suggest that graphics integrity at the encoding stage does not assure correct visualization at the decoding stage. Dasgupta et al. discussed a number of factors (in addition to encoding precision) that may affect the correct interpretation in visualization [248]. Kanjanabose et al. showed that for data retrieval tasks, visualization did not outperform data tables in both accuracy and response time [249]. Wang et al. showed that humans are not visually sensitive to small perturbations of the data [250]. These all led to the question of what the critical encoding precision is for graphics integrity in visualization processes.

As summarized by Chen and Edwards [251], there are two schools of thought in the field of VIS, namely *Isomorphism*, which asserts that "do not introduce any distortion that is inconsistent with the source data", and *Polymorphism*, which maintains that "distortion can be featured in visualization and can bring benefit." Chen et al. noticed that the strong argument for graphics integrity (i.e., isomorphism) cannot easily be enforced and some commonly-used visualization techniques (e.g., volume visualization and metro maps) exhibit a fair amount of infringement of graphics integrity [252, 253], and they reasoned that the viewers' knowledge may help alleviate the potential distortion that may be caused by such unfaithful visual encoding.

Chen and Golan noticed that many-to-one mapping is ubiquitous in visualization as well as in other data intelligence processes (e.g., statistics, algorithms, and human-computer interaction) [254]. Hence, retrieving data values from visualization typically involves a lot of one-to-many mappings, and potential distortion is inevitable. Their cost-benefit ratio suggests a trade-off among *alphabet compression* (a mathematical measure for abstraction, filtering, sorting, etc.), *potential distortion* (a mathematical measure for errors, misinterpretation, biases, etc.), and cost. Using the cost-benefit analysis, we can ascertain issues in icicle and sunburst trees as high cost in decoding and issues in sunburst trees as distorted abstraction in encoding, and potential distortion in decoding. As a trade-off argument, this is not to say that icicles and sunburst trees should not be used at all, but to instigate a question of whether there is a visual representation that offers a better trade-off.

2.3 VIS4ML Workflows

While previous section focuses on visualization methods for supporting ML models, this section reviews VIS4ML workflow-oriented approaches that integrate visualization methods into the overall ML development lifecycle.

Theoretical investigations into visualization for machine learning (VIS4ML) have gradually matured, with a growing body of work proposing frameworks, surveys, and models to guide the design, evaluation, and understanding of such systems.

Chen et al. [255] proposed a systematic ontology for identifying and reasoning about technical issues in visual analytics workflows, enabling structured exploration of design decisions and evaluation strategies. In a critical meta-review, Hohman et al. [70] highlighted the generalizability gap in VIS4ML research, emphasizing the need for broader user studies and more representative scenarios to ground claims. Yang et al. [256] examined the intersection of foundation models and visualization, outlining mutual opportunities and challenges in model explainability and visualization augmentation. Sacha et al. [184] reflected on over a decade of experience integrating ML and VIS, summarizing mutual benefits and highlighting persistent research challenges in explainability and automation.

Chatzimparmpas et al. [257] provided a state-of-the-art report categorizing visualization techniques that aim to foster user trust in ML systems, offering analytical insights into research trends and evaluation practices. In the industrial context, Okonkwo et al. [258] proposed a conceptual framework to guide the application of explainable visual analytics for human-centered AI deployment in high-stakes environments. Huang et al. [259] conducted a comprehensive survey on the use of embedding representations in visual analytics, identifying methodological trends and future research avenues. Wang et al. [260] developed a domain ontology to formalize VA-assisted ML workflows, offering a reusable conceptual framework grounded in semantic technologies.

2.3.1 VIS in Model Development

Tam et al. [161] analyzed humans' role in ML workflows and estimated their contributions quantitatively using information theory. Sacha et al. [184] outlined a VIS4ML ontology, pointing out that many processes in ML workflows can benefit from VIS. Wang et al. [250] introduced HypoML, a VIS-enabled method for testing different hypotheses about whether ML models have learned particular features. Zhang et al. [261] and Chen et al. [262] used VIS for quality analysis of the data used in training and testing.

VIS has been utilized to support a variety of ML workflows featuring techniques such as CNN [250, 263], RNN [264, 265], decision trees [198, 266], random forest [267], reinforcement learning [268–270], ensemble learning [189, 271], and so on. These VIS4ML workflows have been used to develop ML models for different applications, e.g., music analysis [189, 271], system control [268, 269], weather prediction [264, 272], image classification [250, 263], and so on.

Although existing research has made some progress in specific visualization design, the design ideas of many systems can only be applied to specific tasks, and the design ideas lack universality. In addition, the current system pays little attention to some problems existing in the training data, and there is a lack of corresponding research on the improvement of problematic training data. In Chapter 4, we use VIS to support example selection for LLM-based data augmentation, which in turn supports the development of text classifiers. In other words, it is VIS4ML4ML.

2.3.2 VIS for Domain-Specific ML Applications

VIS in Text Analysis Alharbi et al. [273, 274] conducted a survey on text visualization, categorizing existing surveys into five groups and offering recommendations to researchers. Liu et al. [275] conducted a task-driven survey, categorizing text visualization techniques according to user tasks, such as topic understanding, sentiment analysis, and discourse analysis. Fischer et al. [276] focused their survey on the analysis of communication data, highlighting current practices, challenges, and emerging research directions in the area.

There are several commonly seen plots for text visualization. TagCloud [277] and Word Cloud [278] summarize the statistics of word usage using different character sizes. Texts are often combined with trees and graphs to depict the relationships among words and phrases (e.g., [279, 280]). There are also several visual designs for social media data, such as maps for topics [281] and flows for changing texts [282–284].

Since texts are themselves "raw visual representations", almost all text visualization plots feature processed data resulting from statistics and algorithms. Much

of VIS research in text analysis focused on techniques and tools where statistics, algorithms, visualization, and interaction were integrated. In recent years, ML models (i.e., ML-learned algorithms) play more noticeable roles in text analysis. Luo et al. [285] developed a tool (EventRiver) where event-based text analysis and visualization were integrated to reveal the events that inspired some texts and stories. Park et al. [286] introduced a temporal model for representing narrative clinical text in chronological order. Brehmer et al. [287] designed a system (Overview) for investigative journalists to perform document clustering, visualization, and labeling. Liu et al. [288] combined a Bayesian network model with flow-based text visualization to analyze real-time text streams. Duo and Liu [289] combined topic modeling with interactive visualization to analyze temporal text data. Wu et al. [284] combined a self-organizing map (SOM) with glyph-based timeline visualization to explore social media data. Abdul-Rahman et al. [290] developed a VIS tool (ViTA) for text similarity analysis, which allowed users to define text comparison functions using block diagrams.

El-Assady et al. used an integrated approach to analyze conversational text [291, 292], explore topic space views [293], and examine named entity relationships [294]. Schorr et al. [295] developed a web-based tool for detecting repetitive patterns in a text corpus. Knittel et al. [296] proposed to use a parallel clustering approach to handle high-volume text data dynamically. Sevastjanova et al. [297] developed a system to enable personalized insight reports in visual discourse analysis. Fischer et al. [298] described a technique for communication analysis through interactive modeling of dynamic data.

ML in Text Analysis and Generation ML models have become an indispensable part of various workflows for processing texts. Many text analysis tasks fall into the category of text classification [299], which includes topic modeling (e.g., [300]) and sentiment analysis (e.g., [301, 302]). Other text processing tasks include relationship extraction (e.g., [303]), machine translation (e.g., [304]), text summarization (e.g., [305]), text generation (e.g., [306]), spelling and grammar checking (e.g., [307]), and so on.

The introduction of pre-trained models like BERT [308] enhanced text analysis capabilities. The emergence of LLMs, e.g., GPT-3 [309], has introduced important new resources to text analysis and visualization. Schetinger et al. discussed some challenges associated with these models [310], while El-Assady et al. [311] and Sevastjanova et al. [312, 313] explored potential opportunities. In our work, we use LLMs to generate synthetic training data to address the challenges in collecting labeled data in some real-world applications.

Data augmentation (e.g., [314]) and *data imputation* (e.g., [315]) both refer to techniques for generating synthetic data based on statistical characteristics of known data. The former usually focuses more on synthetic data used for training ML models, and the latter focuses on filling in the blanks of missing data. More recently, ML researchers began to utilize LLMs in ML development workflows [316], and in particular for data augmentation for text classification (e.g., movie, restaurant, and electronic product reviews [317]), SMS intent recognition [318], sentiment analysis in movie reviews [319], and spam detection [320]. Our work was first reported in the same period [321]. Unlike the common approach of randomly generating synthetic data, we used visual analytics to target data augmentation at specific areas of the data space with data deficiency.

VIS4ML in Text Analysis Workflows for developing ML models for text analysis often include different processes where human knowledge is used to assist in the development. For example, Heimerl et al. [322] proposed VIS methods for training and evaluating text classifiers. Sperrle et al. [323] developed a tool called VIANA for assisting annotation of argumentation with the aid of interactive visualization. El-Assady et al. proposed VIS techniques for topic model optimization, through visualizing parameters [311], progressive learning parameters [311], and manipulating speculative execution [324]. El-Assady et al. [325] developed a VIS4ML tool, with which ML developers can explore semantic concept spaces to guide topic model refinement. Sperrle et al. [326] developed a VIS tool for refining topic models with the aid of adaptive guidance generated using mixed-initiative. Wang et

al. [260] developed M2Lens, a VIS tool for better understanding and diagnosing multi-modality models for sentiment analysis.

VIS can also provide means to enhance the explainability and interpretability of ML models. Liu et al. [190] highlighted how visual analytics enhances the interpretation of ML models in text analysis. Spinner et al. [4] developed explAIner, a VIS framework for interactive and explainable ML. Sevastjanova et al. [327] used VIS to aid the explanation of language model contextualization. Sevastjanova et al. [328] proposed a gamified approach for explaining language phenomena through interactive labeling. Li et al. [216] developed a VIS tool (DeepNLPVis) to facilitate a better understanding of NLP models for text classification.

Through the review and analysis of visualization methods for text data in existing VIS4ML workflows, it becomes evident that current techniques for visualizing model results make limited use of methods such as radial basis function (RBF)-based visual analysis for exploring results in sparse or unobserved regions. Furthermore, more novel visualization techniques, such as tag treemaps, have rarely been proposed or applied in prior work. In Chapter 4, we address these gaps by applying radial basis functions and tag treemap visualizations to analyze both the training data generated by LLMs and the resulting model outputs, thereby contributing new perspectives and methods to the visualization support of ML workflows on text classification tasks.

In summary, this chapter reviewed related research from three perspectives. Although progress has been made in model training and visualization, there remains a lack of approaches to address *inconsistencies between different data spaces*, such as those arising between development and deployment or between real and adversarial data spaces. While data synthesis methods have advanced rapidly in recent years, most studies remain focused on algorithmic improvements, and the integration of synthetic data generation as a stage of ML workflows is still uncommon. And there is limited comparative analysis and visualization support for different synthetic datasets, which restricts developers' ability to make effective and targeted use of data augmentation methods.

These limitations motivate a new ML workflow design that incorporates synthetic data generation into traditional ML workflows and leverages visualization supports to analyze and improve training data quality, thereby providing more effective means of addressing inconsistencies between different data spaces. The next chapter introduces the concept of *Four Views Design* and demonstrates its role in tackling the aforementioned challenges.

3

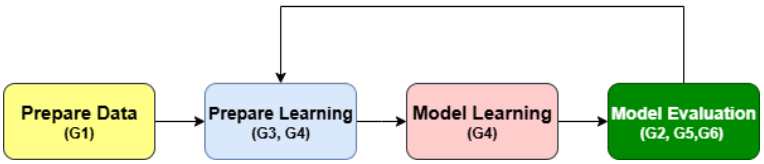
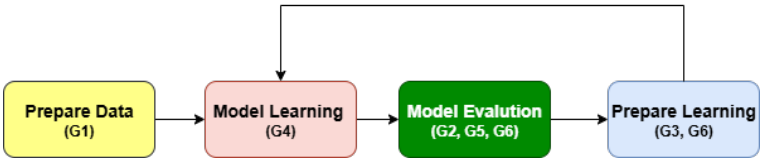
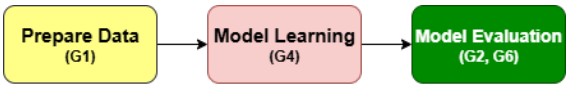
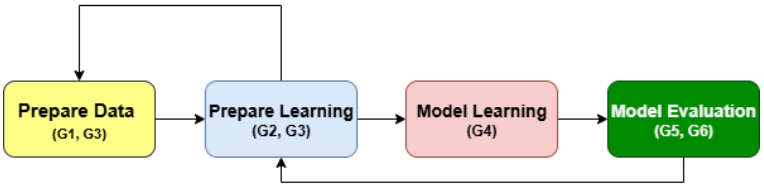
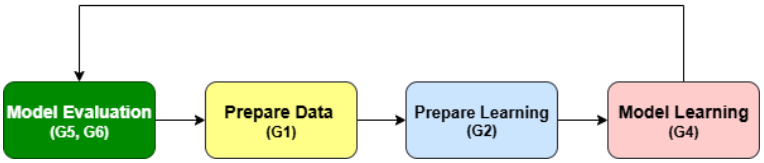
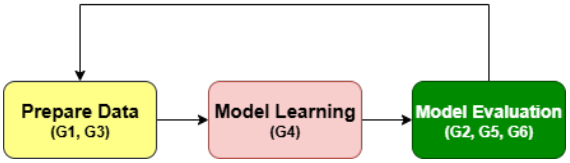
Four Views Design for Machine Learning Workflows

3.1 Comparison with Traditional ML Workflows

After reviewing the literature in Chapter 2, we identified that the inconsistencies across different data spaces represent a challenge in ML workflows that existing approaches inadequately address. These inconsistencies, manifested as discrepancies between development and deployment data spaces, or between application-specific and adversarial data spaces, motivated our examination of traditional ML workflows to identify their limitations in handling such challenges.

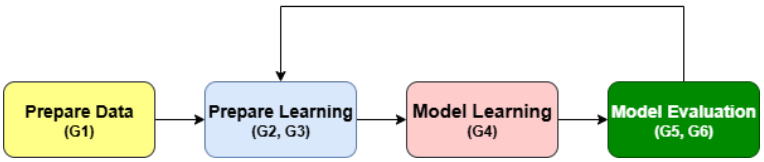
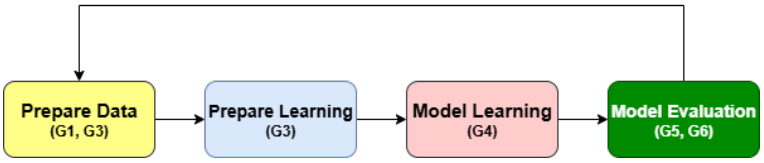
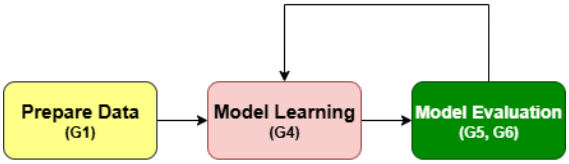
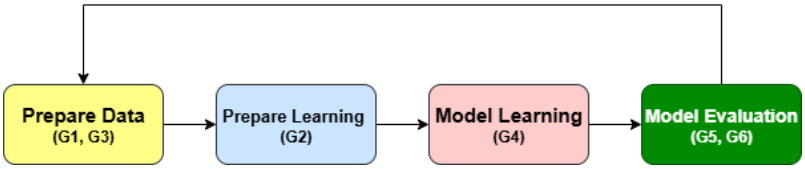
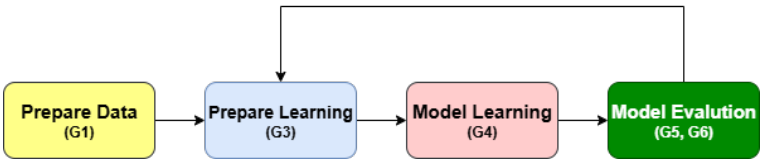
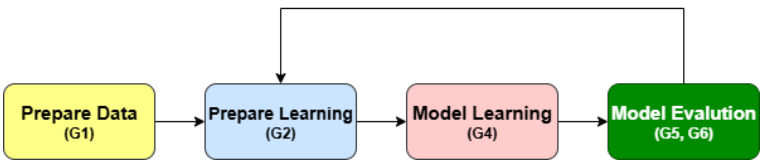
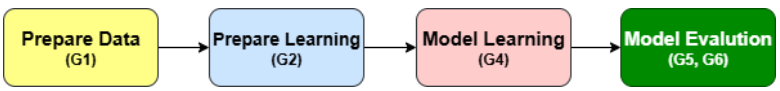
In Table 2.1 of Chapter 2, we collected and summarized 15 representative papers in the field of machine learning. For each paper, we organized the ML workflow involved, as shown in Table 3.1, where the paper indices correspond one-to-one with those in Chapter 2. The terminology we use for each ML workflow stage is drawn from the four specific phases described in the VIS4ML paper by Sacha et al. [184], which are represented with four distinct colors. For the six different VIS4ML goals mentioned in their paper, we have also marked the specific goals to be solved in each phase of the workflow in smaller fonts.

Table 3.1: Traditional ML workflow from different ML tasks. To facilitate comparison with the new design, we set Prepare Data-related to yellow, Prepare Learning-related to blue, Model Learning-related to pinkish color, and Model Evaluation-related to green.The meanings of G1 to G6 are: G1:Prepare Data; G2: Understand Model; G3: Feature Analysis; G4: Learning Process; G5: Result Analysis; G6: Comparative Analysis.

Paper ID	Workflow
P-1 [3]	
P-2 [7]	
P-3 [9]	
P-4 [17]	
P-5 [4]	
P-6 [15]	

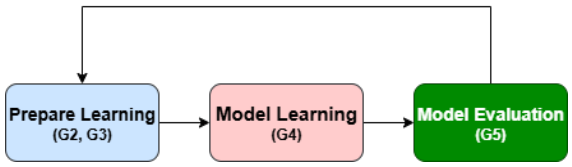
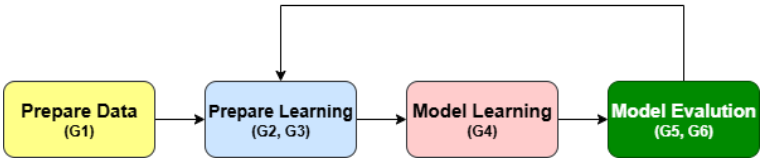
Continued on next page

Table 3.1 – continued from previous page

Paper ID	Workflow
P-7 [16]	
P-8 [14]	
P-9 [5]	
P-10 [8]	
P-11 [11]	
P-12 [6]	
P-13 [12]	

Continued on next page

Table 3.1 – continued from previous page

Paper ID	Workflow
P-14 [13]	
P-15 [10]	

We can observe from Table 3.1 that most workflows follow the standard sequential process of "Prepare Data–Prepare Learning–Model Learning–Model Evaluation" often accompanied by loops.

These loops in traditional ML workflows are designed to enable iterative refinement, allowing model users to revisit earlier stages when model performance is unsatisfactory or when new insights emerge during the development process. The loops typically facilitate feedback between model evaluation and earlier phases, enabling data preprocessing adjustments, feature engineering modifications, or hyperparameter tuning based on evaluation results.

Notably, some workflows in Table 3.1 (such as P-3 [9] and P-13 [12]) follow a purely sequential approach without explicit feedback loops. These linear workflows are typically designed for straightforward tasks. While such approaches can be efficient for specific scenarios, they may lack the flexibility needed for complex or exploratory ML tasks where iterative refinement is necessary. The ML workflows proposed in different papers also show differences in terms of module coverage: some workflow designs are relatively complete, encompassing multiple phases such as data exploration, model selection, debugging, and export, whereas others remain only at partial phases with limited loop feedback. Researchers often tailor the workflows individually according to specific task requirements or research backgrounds.

By closely examining the feedback mechanisms of these workflows, we find that most loops are mainly concentrated on aspects such as model parameter adjustment,

feature selection optimization, or algorithm switching, while they only occasionally involve feedback to the initial Prepare Data phase. Such a design is reasonable and often sufficient for many traditional ML tasks when the dataset is relatively stable, of high quality, and the task objectives are well-defined. In these contexts, the focus on model parameters, features, or algorithm adjustments can effectively address most performance issues without requiring changes to the training data. In these scenarios, the Prepare Data phase is often regarded as a one-time preprocessing step without the need for subsequent feedback from model results.

With the increasing complexity of ML application scenarios and the diversification of practical environments, this traditional feedback pattern has begun to reveal certain shortcomings. When a model performs poorly on specific categories, it may be necessary to reconsider the data sampling strategy; when the bias in feature distributions is detected, adjustments to data preprocessing may be required; and when facing data sparsity or imbalance problems, relying solely on modifications of existing data may not fundamentally resolve the issue. These challenges motivate the exploration of additional data sources to enrich the training data used in the model learning process, thereby better supporting ML workflows and enhancing model performance.

In analyzing existing workflows, another observation is the absence of *synthetic data generation* and integration. Through an examination of the ML workflows in Table 3.1, we find that the majority focus on processing, modeling, and evaluating existing data, while paying little attention to the generation and utilization of synthetic data. This phenomenon is understandable in the era of traditional machine learning, as tasks primarily relied on improving models, and data augmentation techniques were relatively limited. However, with the development of machine learning and the advances in generative models, the potential value of synthetic data in ML workflows has become increasingly prominent, serving as a practical tool to address the problem of inconsistencies across different data spaces.

The integration of synthetic data can bring many benefits to ML workflows. Synthetic data can effectively address the problem of data sparsity in places, such

as enterprises, where strict privacy requirements are imposed; by generating data with similar statistical properties but without involving real individual information, it is possible to expand the training dataset while preserving privacy. In addition, synthetic data provides a new solution for handling data imbalance: by selectively generating minority class samples, it can improve model performance on imbalanced datasets. Synthetic data can also support adversarial training and robustness testing by generating boundary cases or anomalous samples.

And the process of synthetic data generation itself is an iterative optimization procedure that requires feedback with data quality assessment, model training, and result analysis. This creates an important opportunity for building more comprehensive ML workflows. Based on these considerations, the integration of the synthetic data generation process into traditional ML workflows has become a key factor in the design of new ML workflows.

Motivated by these observations, we propose the "Four Views Design", a visual analysis workflow that integrates the different components of ML development into a coherent structure. By organizing the process into four interconnected and iteratively accessible analytical perspectives, this design emphasizes feedback, adaptability, and human-machine interaction support. The "Four Views Design" consists of four components: Dataset View, Model View, Results View, and Synthesis View. The following demonstrates the typical functions of these four Views:

- **Dataset View**: Provides visual summaries of training data characteristics and potential quality issues, guiding informed dataset selection and evaluation.
- **Model View**: Provides visual analysis of model architecture and parameter settings, thereby improving interpretability and supporting the training process.
- **Results View**: Evaluates model performance, prediction results, and output behavior using different visualization methods.
- **Synthesis View**: Generates, compares, and selects synthetic data generated through different generation methods using visualization techniques.

Table 3.2: Comparison of Traditional ML Workflow vs. Four Views Design

Aspect	**Traditional ML Workflow**	**Four Views Design**
Parameter Control	Manual Trial	Structured Exploration
Target Selection	Random Sampling	Systematic Selection
Visualization	Basic Visual Plots	Comprehensive Analysis
Iteration Support	Limited	Fully Integrated
HITL Integration	Minimal	Extensive

It should be noted that the names of each view are not rigidly fixed and can be adapted in different implementations or applications.

The Four Views Design's functions show that they can be extended or refined according to specific application scenarios. It provides a flexible workflow that enables ML researchers to tailor or extend its components according to specific application scenarios. Unlike traditional workflows, the Four Views Design emphasizes flexible switching and iterative integration among the four views, enabling developers to dynamically adjust data processing, model optimization, or result interpretation based on analytical insights, thereby transforming the development process from a mode dependent on repeated trial-and-error into a more insightful and controllable process. Besides, it introduces an approach to address the problem of the inconsistencies across different data spaces that traditional experiment-based optimization cannot resolve. By improving training data quality, the Four Views Design helps ML developers reveal differences between different data spaces, create targeted synthetic samples to probe and repair these differences, and use visualization support evaluation to measure the synthetic samples' quality. Specific practical examples of the interaction processes between the four "Views" can be found in Figure 3.1 in the following Section 3.2 or in Figure 4.1 in Section 4.2.

Table 3.2 illustrates the differences between the traditional ML workflow and the proposed Four Views Design. The key differentiator is that the Four Views Design makes iteration view-driven and multi-directional. Any finding in the Four Views can trigger a new loop, rather than relying mainly on feedback loops in traditional workflows. The comparison highlights several key areas of improvement: parameter control, target selection (including both model and training data), visualization

techniques, support for multiple iterative loops, and the integration of human-in-the-loop (HITL). Unlike the traditional process, the Four Views Design reconceptualizes model development as an analysis process that can be conducted from four relatively independent yet interrelated perspectives: model, data, generation, and results. This shift enables ML developers to more effectively evaluate and improve model quality.

After understanding the functions of the four views in the Four Views Design workflow, we can compare them with existing workflows in Table 3.1 and notice that some papers [13] lack Dataset View content as visual analysis of training data, and mostly conclude with direct data input. This creates limitations for users in understanding data distribution, identifying potential biases, and detecting anomalous samples. The lack of Dataset View support means that researchers typically cannot intuitively control data quality before modeling, making it difficult to timely discover issues such as class imbalance, noise contamination, or feature redundancy. This "black box" data input process undermines the interpretability of data throughout the entire workflow and reduces the targeting and efficiency of subsequent model optimization.

Regarding the Model View, some papers provide detailed descriptions and visualizations of model structures and hyperparameter configurations, such as papers [10] that show comparisons of different model architectures or explain intermediate results of the training process [7]. However, some papers do not conduct a specific analysis of the model itself, but treat it as a "black box" [9], focusing only on the input-output mapping relationship. This difference reflects the current imbalance in ML workflows regarding the transparency of internal model mechanisms, requiring visualization methods to help users understand model decision logic and performance differences, thereby providing more targeted support in optimization and iteration.

The Synthesis View is what we can observe from the table that the vast majority of ML workflows have not addressed. In existing research, datasets are typically treated as given input conditions, and researchers rely more on data cleaning, feature engineering [17], or sampling methods to improve model performance,

rarely considering actively repairing or expanding datasets through generating new synthetic data. The lack of attention to synthetic data makes existing workflows inadequate when dealing with issues such as class imbalance, sample scarcity, or distribution shift. In contrast, the introduction of Synthesis View provides users with a new approach: directly improving data quality and providing richer learning materials for models through dynamically generated and iteratively optimized data, thereby enhancing the adaptability and robustness of the overall system at the data level.

Besides, some papers' ML workflows [9, 12] only serve as one-time processes, lacking iterative processes. Even in the few works that involve iteration, the iteration only remains at the level of model parameter adjustment or repeated training, making it difficult to achieve full feedback between data, model parameters, and model performance. This limitation causes workflows to often exhibit "static optimization" when facing complex tasks, meaning researchers typically only make single or limited improvements within fixed datasets and preset model frameworks, with optimization objectives limited to improving performance metrics for a particular round of experiments. Such processes lack continuous responsiveness to data distribution changes or insufficient result interpretation, making it difficult for systems to dynamically adjust based on intermediate feedback.

Another phenomenon from the table is that almost all ML workflows place "Model Evaluation" as the core component after model training. Although this performance metric-oriented design ensures that experiments can quickly obtain intuitive feedback, researchers often rely on final evaluation results to judge experimental effectiveness, but lack in-depth attention to data quality, model structure transparency, and intermediate process diagnosis. The result is that traditional workflows easily evolve into improving accuracy or recall of single-round experiments, while lacking proactive identification of potential problems and improvement. This makes them difficult to support continuous optimization and dynamic adjustment requirements under complex tasks.

Against this backdrop, the proposal of the Four Views Design workflow is precisely aimed at providing a comprehensive response to the aforementioned deficiencies: it supplements the three parts of Dataset, Model, and Synthesis, and introduces iterative mechanisms, thereby achieving reconstruction and enhancement of traditional ML workflows.

In other words, the main innovations of the Four Views Design workflow lie in the following two aspects:

First, a major difference between Four Views Design and traditional testing lies in the introduction of the concept of synthetic data. While most existing workflows limit model improvement to parameter tuning, data cleaning, or feature engineering, Four Views Design introduces a dedicated synthesis view that provides diverse generation strategies to address issues such as class imbalance, data sparsity, or insufficient data. More importantly, synthetic data is not treated as a one-off addition but as a continuously managed resource: its quality can be iteratively updated and optimized. In this way, synthetic data becomes an adjustable factor on par with model parameters, enabling more data-driven model development.

Second, Four Views Design introduces an iteration mechanism, which is another of its innovations that distinguishes it from traditional ML workflows. Traditional experiments often treat training and testing as separate phases, with evaluation results rarely feeding back into data refinement. Even when iterations occur, they typically involve parameter adjustments followed by retraining, leaving data quality unchanged. In contrast, Four Views Design tightly integrates testing, visualization, and training into an iterative loop that continuously diagnoses and addresses problems. The system provides timely feedback when performance declines or biases emerge, and it guides users to retrain models with improved synthetic data. This mechanism allows models to evolve dynamically rather than remain static, enhancing the adaptability and robustness of the workflow. Iterations in the Four Views Design can occur between any two of the four views, whereas traditional ML workflow iterations typically occur primarily between model evaluation and prepare learning phases, with limited iterative processes between other stages.

This limitation causes the phases, such as data collection, feature construction, or model selection, to often remain fixed on previous decisions, making it difficult to promptly adapt to new discoveries or feedback. The Four Views Design establishes multi-directional iterations that enable users to proactively initiate corrections and explorations at any stage. This iterative approach breaks the linear constraints of conventional workflows and provides greater scope for human-machine collaboration, allowing model development to evolve dynamically and better address complex and variable application scenarios.

In the following sections, we show how "Four Views Design" leverages this iterative capability, combining visual analysis, synthetic data generation, and model evaluation.

3.2 Four Views Design Workflow in Model Testing against Adversarial Attacks

In the literature, almost all workflows for testing adversarial attack algorithms can be seen as an extension of the conventional ML testing workflows. As illustrated in Figure 3.1(a), in such a workflow, selected testing data is perturbed by an adversarial attack algorithm, and the resultant attack data is then fed into an ML model. The ML developers typically evaluate the adversarial attack algorithm by observing some statistical measures about the attacks, as well as some successful instances of the attack data. It is common for such a workflow to include many iterations, in which ML developers may select different test data to perturb or modify the parameters of the adversarial attack algorithm. Although the accumulated amount of human effort made in such iterative processes is non-trivial, it is usually not documented in the literature, and it is unsupported by any purposely built user interface to make the human effort more efficient and effective.

To a certain extent, ML developers have been accustomed to such iterative processes with limited software support for HITL activities. The objectives of testing adversarial attack algorithms are defined as (i) finding instances of effective adversarial attacks, and (ii) obtaining the statistical measures for evaluating

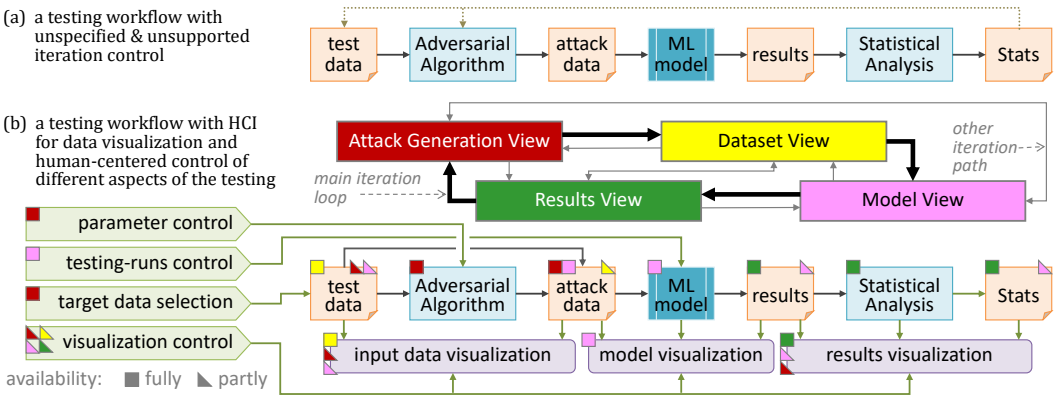


Figure 3.1: (a) Conventional workflows for testing adversarial attack algorithms focus on the statistics and instances that confirm the successes of an attack algorithm. Although iterative testing is common, such processes are typically not mentioned in the literature and are not supported by a purposely designed user interface. (b) An ideal workflow for testing against adversarial attacks should focus on the analysis of the behaviors of the model under attack and may feature many iterative loops where ML developers need to observe input data, generate attack data, carry out testing runs, and observe results. A software tool designed to support HITL activities can provide such iterative loops with effective and efficient user controls of the testing processes and visualization of the data, models, and results in the processes.

adversarial attack algorithms. However, if we focus the testing on evaluating the ability of a model to defend against adversarial attacks, we will need more than a few instances or a few statistical measures. Likely, the ML developers would like to obtain and observe information about the following questions:

Q_a. How easy for an ML model to be effectively attacked by an adversarial attack algorithm with some randomly selected parameters?

Q_b. How easy for an ML model to be effectively attacked by an adversarial attack algorithm with some randomly selected target data objects?

Q_c. Are some types of target data objects more vulnerable than other types?

Q_d. Are some parts of the ML model more vulnerable than other parts?

These requirements suggest the necessity of effective and efficient HITL support for several user tasks performed frequently in iterative testing processes, including:

1. **Parameter Control** (cf. Q_a) — with which an ML developer can assign some randomly selected parameters to an adversarial attack algorithm to observe the performance of the ML model under random attacks. The ML developer may also try to set parameters manually to see how easy to identify parameters that are "harmfully" effective for the attack algorithm.

2. **Target Data Selection** (cf. $Q_{b,c}$) — with which an ML developer can activate a random selection of target data objects to observe the performance of the ML model under random. The ML developer may also try to select target data objects manually to see how easy to identify some specific types of data objects as targets.

3. **Testing-runs Control** (cf. $Q_{a,b,c,d}$) — with which an ML developer can determine how many original data objects and how many perturbed data objects to be fed into the ML model.

4. **Visualization Control** (cf. $Q_{a,b,c,d}$) — with which an ML developer can interact with the visualization facilities to determine what data to be plotted, what levels of details are included, and what feature to be highlighted, enabling exploratory observation and analysis.

5. **Input Data Visualization** (cf. $Q_{a,b,c}$) — with which an ML developer can observe and analyze how a specific parameter setting affects the perturbation of the target data objects, what types of target data objects are selected, and which type may be more effectually attacked than other types.

6. **Model Visualization** (cf. $Q_{c,d}$) — with which an ML developer can observe and analyze how various parts of a model may behave differently in response to the original and perturbed data. If effectual attacks make certain parts of the model behave differently, these parts may be more vulnerable. The ML developer can also observe how different types of data affect different parts of the model differently.

7. **Results Visualization** (cf. $Q_{a,b,c,d}$) — with which an ML developer can observe complex results from the testing runs of the model under different conditions. For example, ML developers may wish to observe the temporal patterns of effectual attacks among a sequence of attacks or compare multiple sequences of attacks by observing a variety of statistical measures associated with these sequences.

Hence, we need to support the testing workflow in Figure 3.1(a) by introducing a purposely built user interface, where these HITL-supporting facilities can be made available to ML developers as illustrated in the lower part of Figure 3.1(b). Because there are many types of HITL-supporting facilities, some of which are shared by different tasks, we divide the testing workflow coarsely into four stages. As shown in the upper part of Figure 3.1(b), the four stages are *Attack Generation*, *Data Observation*, *Model Run & Analysis*, and *Results Analysis*.

Typically, when an ML developer starts a testing workflow, the first iteration will likely commence at the *Data Observation* stage, and the ML developer may conduct a test with the original data objects without any perturbation (i.e., *Model Run & Analysis*). Having observed the "normal" behavior of the model (i.e., *Results Analysis*), the ML developer may initiate a new iteration by generating some attack data at the *Attack Generation* stage. The process will likely be repeated through multiple iterations with different algorithmic parameters and target data objects. Hence, we use thick arrow-directed lines to indicate the main iteration loop. In practice, an ML developer may follow other paths as indicated by the thin gray lines in the diagram. Therefore, the switching among the four stages does not need to follow a particular order. This high-level design of the workflow for testing against adversarial attacks provides the design and development of TA3 in Chapter 5 with the basic conceptual framework as shown in Figure 3.1(b).

3.3 Four Views Design in Two Other Machine Learning Tasks

The Four Views Design workflow was initially proposed in the context of adversarial attack testing, but its application is not limited to this. This thesis explores the performance of the Four Views Design in two important applications: text classification tasks and human physical activity classification from wearable devices.

3.3.1 Application in Text Classification

One of the major applications extended the Four Views Design to text classification tasks called "iGAiVA" in Chapter 4, implemented in collaboration with Inetum Spain for their internal ticket classification system. This adaptation demonstrated the workflow's flexibility in handling different data types and model architectures. The workflow was modified to address text classification challenges as follows:

- **Data View**: A new tag-treemap visualization method was proposed, which is integrated to analyze text data distributions and identify imbalances in training data across different categories.
- **Model View**: The Model View was adapted to show the model's structure with different metrics.
- **Results View**: The Results View was adapted to analyze the model results, which helps to search and locate suitable data as targets for synthetic data input.
- **Synthesis View**: Visualization methods were applied to compare the similarity between generated and real text data, helping ensure the quality of synthetic training data examples.

Through visual analysis of LLM generation results, the Four Views Design can help developers quickly identify differences and potential quality issues between generated data and real data. Developers can determine whether LLM-generated

text matches the target category in terms of content and style, thereby filtering out high-quality synthetic samples for training. Furthermore, the multi-round iterative visual analysis process allows for the introduction of human feedback in generation strategies, enabling timely adjustment of generation parameters or prompts to reduce bias and inconsistencies. Through this workflow that integrates data visualization, model analysis, result evaluation, and data generation, the Four Views Design effectively improves the utility of synthetic data and the overall classification performance of models in text classification tasks. Other details of the iGAiVA can be found in Chapter 4.

3.3.2 Application in Human Physical Activity Classification for Wearable Devices

The other application of the Four Views Design workflow was in the domain of human physical activity classification called "DS4ML" in Chapter 6, conducted in collaboration with Statistics Netherlands. The workflow was adapted to address the challenge of inconsistent hidden patterns between training data and deployment data in physical activity recognition models. In this application, the four views were utilized as follows:

- **Dataset View**: The visualization method was used to effectively compare state changes in time series data, enabling detailed analysis of movement patterns in both training and deployment datasets.

- **Model View**: The Model View facilitated the assessment of model behavior across different movement patterns, helping identify where the model's performance diverged between laboratory and deployment conditions.

- **Results View**: Through visualization of model outputs, researchers could pinpoint specific movement sequences where the model's performance degraded.

- **Synthesis View**: The Synthesis View guided the generation of synthetic training data that better represented deployment environment movement patterns, improving model robustness.

This application improved model performance, with the enhanced training data enabling better recognition of complex motion patterns in deployment scenarios. Furthermore, the introduction of the Four Views Design in this application allows researchers to analyze human activity classification tasks from four interconnected perspectives: data, model, results, and data generation. This structured analytical approach provided by the Four Views Design helps guide subsequent data augmentation strategies and model optimization processes, thereby improving practical deployment robustness while maintaining the accuracy of laboratory model training.

3.4 Conclusion for the Four Views Design Workflow

The successful application of the Four Views Design across these different domains presented sequentially in the thesis, from iGAiVA to TA3 to DS4ML, demonstrates its ability to become a useful workflow that enhances ML model performance by improving training data quality, enriching training data, and testing methods. These applications demonstrate key contributions of the Four Views Design, including:

1. **Adaptability**: The workflow successfully accommodated different data types (text data, image, and time series data) and various model architectures while maintaining its core workflow.

2. **Effectiveness**: From the smaller MNIST and Fashion-MNIST datasets to the massive Oxford-Capture-24 and Titanic datasets, the workflow can effectively scale to datasets of varying sizes and complexities.

3. **Integration Capability**: The workflow effectively integrated new visualization techniques (new visual design, new VIS4ML strategy) and model technologies (generative AI) while maintaining its fundamental workflow.

The successful cross-domain application of the "Four Views Design" workflow demonstrates that whether applied to enterprise-level text classification tasks, image classification tasks for adversarial attack testing, or time series-based human physical activity classification from wearable devices, these seemingly different application scenarios all follow the same underlying logic: the iterative process of data preparation, model analysis, result evaluation, and data synthesis. These applications demonstrate the effectiveness of the "Four Views Design" workflow, providing an analytical workflow for the ML model development process into four interconnected perspectives.

The Four Views Design establishes an important workflow for VIS4ML. From an industrial perspective, the successful application of the Four Views Design demonstrates its value in solving practical business problems. Whether in Inetum Spain's text classification system or Statistics Netherlands' wearable device data analysis project, the workflow has shown effects in improving model performance, reducing development costs, and enhancing system reliability. This provides strong empirical support for the large-scale adoption of such methods in industry. These enhancements enable ML developers to conduct more thorough and efficient testing of their models against adversarial attacks while gaining deeper insights into model behavior and vulnerabilities.

Based on the "Four Views Design" workflow introduced above, the following chapters of the thesis will elaborate on how this workflow is specifically applied in practical ML tasks. These examples provide evidence of the practical utility of the "Four Views Design" across various ML environments.

4

iGAiVA: Integrated Generative AI and Visual Analytics for Text Classification

In developing machine learning (ML) models for text classification, one common challenge is that the collected data is often not ideally distributed, especially when new classes are introduced in response to changes of data and tasks. In this chapter, we present a solution for using visual analytics (VA) to guide the generation of synthetic data using large language models. As VA enables model developers to identify data-related deficiency, data synthesis can be targeted to address such deficiency. We discuss different types of data deficiency, describe different VA techniques for supporting their identification, and demonstrate the effectiveness of targeted data synthesis in improving model accuracy. In addition, we present a software tool, iGAiVA, which maps four groups of ML tasks into four VA views, integrating generative AI and VA into an ML workflow for developing and improving text classification models.

The design philosophy of iGAiVA follows the Four Views Design approach presented in Chapter 3, and represents an application based on the Four Views Design workflow. This reflects a form of data space inconsistency, where the training data space fails to capture sufficient coverage of the deployment data space. In the thesis, the term "system" is used to describe all three developed

prototypes (iGAiVA, TA3, and DS4ML), even though they were initially referred to as "software" or "tool" in different contexts.

4.1 Introduction

Text analysis and visualization have been studied extensively in the field of visualization and visual analytics (VIS for short). VIS4ML [1], i.e., using VIS to support ML workflows, has also been a research topic attracting much attention in the past decade.

While large language models (LLMs) and generative AI have become a disruptive technology with global impact, developing *organization-specific ML models* for analyzing texts in many contexts is still a non-trivial undertaking.

Although such ML models can capture organization-specific semantics and cost less to deploy, the performance of these ML models is commonly impaired by the limitation of data collection, because (i) the information space (i.e., all possible variations) of texts is huge and adequate sampling would demand both time and resources, (ii) often the amount of text data available to an ML workflow in an application is limited and the distribution of the sampled data is stewed, or (iii) organizational and societal changes sometimes undermine existing ML models while their retraining is hindered by limited availability of new data.

This work is concerned with using VIS to support an industrial ML workflow in the domain of text analysis. In particular, we use LLMs to address the limited availability of real-world data in training ML models for analyzing text data automatically and dynamically in computerized ticketing systems. Because the search space for selecting text examples for LLMs is huge, we use VIS to enable ML developers to target the uses of LLMs at the gaps and shortcomings in the training data. In other words, through visual analytics, ML developers can use their knowledge and reasoning effectively to reduce the search space significantly. More details about the industrial context and requirements analysis can be found in Section 4.2 and Section 4.3. The contributions of this work include:

- Proposing a novel VIS4ML approach in which VIS techniques are used to guide the processes of data synthesis using LLMs.
- Developing a software tool, **iGAiVA**, which integrates generative AI and visual analytics into a unified ML workflow.
- Demonstrating the effectiveness of targeted data synthesis in improving ML model accuracy through visual analytics.

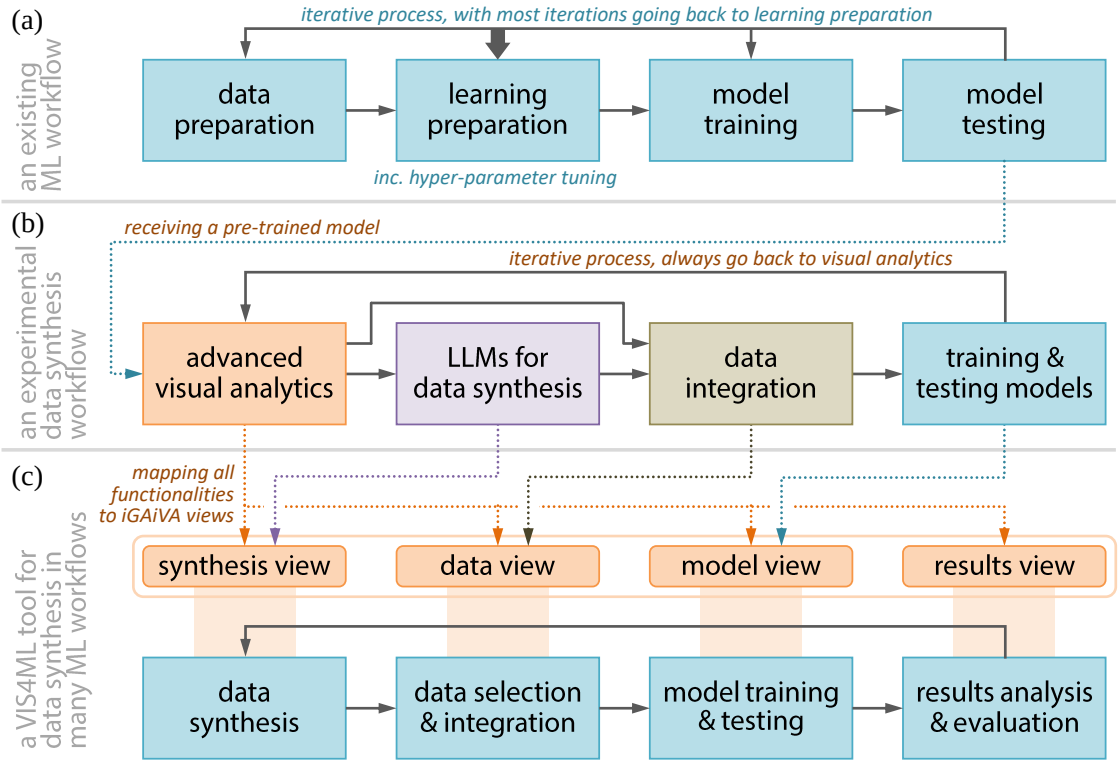


Figure 4.1: The evolution from a conventional ML workflow to an experimental workflow involving the use of VIS techniques and LLMs for data synthesis, and then to an iterative workflow supported by a VIS4ML tool where VIS and LLMs techniques are integrated.

4.2 Background, Motivation, and Process Overview

Inetum (https://www.inetum.com/en/Inetum) is an agile IT services company that provides digital services and solutions, and a global group that helps companies and institutions to get the most out of digital flow. The FabLab team in Inetum

(Spain) has been delivering ML models for classifying text messages received from computerized ticketing systems, facilitating automated task distribution, and rapid ticket responses. For different ticketing systems (e.g., finance, IT, estate, etc.), the messages can be different (e.g., in terms of contexts, lengths, keywords, acronyms, etc.). Different hosts of ticketing systems may have different organizational structures for handling these tickets, leading to different classification schemes. The FabLab team has found that developing models for individual ticketing systems can deliver better solutions to the customers.

In conventional ML workflows, as shown in Figure 4.1(a), ML developers perform numerous iterations from data preparation to learning preparation, training, and testing [187]. As demonstrated by many existing VIS4ML techniques (Section 2.3), VIS techniques can help reduce the number of iterations and speed up the human-centric processes in these iterations. When I started my 2-month placement in Inetum, the company collaborator and I quickly identified the first requirement based on our observation of the existing ML workflows and our knowledge and experience of VIS4ML, i.e., **R1: Using more VIS techniques to help identify possible causes of errors**.

During the first month of the placement, we introduced several VIS techniques for identifying possible causes of errors (**R1**). In particular, we noticed two types of causes: for some classes, (a) there may not be enough training data, and (b) the training data may be of a skewed distribution. Similar to many industrial ML problems, it is not always feasible to collect a sufficient amount of real-world data, especially when some organizational changes trigger semantic modifications to the classification scheme (e.g., adding a new class, merging two classes, or changing some class definitions). In order for the ML workflow not to wait for the relatively slow process to collect a large amount of new data, we identified the second requirement, i.e., **R2: Using large language models (LLMs) to generate synthetic data for training and using VIS techniques to guide the data synthesis process**.

During the second month of the placement, we carried out experiments to address **R2**. We received positive results indicating that (i) synthetic data could improve

ML models, and (ii) VIS techniques could help target data synthesis to the possible causes of errors and evaluate the effectiveness of each data synthesis action. These results encouraged us to consider a substantial industrial requirement. Because the FabLab team in Inetum (Spain) has to develop different ML models for different ticketing systems, ideally, there is a software tool that can support multiple ML workflows. This led to the third requirement, i.e., **R3: Designing and developing a VIS4ML tool for supporting ML workflows involving data synthesis**.

Using the concept discussed in Chapter 2, these three requirements can be interpreted as concrete ways of operationalizing the Four Views Design workflow. **R1** addresses the analysis of the inconsistency between training and deployment data by making them observable and discussable through visualization. **R2** explicitly enriches the training data space in targeted subregions to reduce the training–deployment data inconsistency. **R3** implements the workflow by making synthetic data generation an integral phase, aligning with the Four Views Design introduced in Chapter 3.

The three main requirements for this work were identified in an agile manner, which is consistent with the nested model approach [329]. During the 2-month placement, we formulated a VIS4ML workflow as shown in Figure 4.1(b). We will detail these VIS techniques in Section 4.4, and VIS-guided data synthesis in Section 4.5. Following the placement, we designed and prototyped a VIS4ML tool, called iGAiVA (*integrated Generative AI and Visual Analytics*), for enabling VIS-guided data synthesis in iterative ML workflows as illustrated in Figure 4.1(c). We will detail the design of iGAiVA and its prototype in Section 4.6. There were two further placements, where the system was evaluated and improved in an agile manner. We will report this process in Section 4.7.

The gradual introduction of visualization techniques, then the use of LLMs for data synthesis, and later the decision to develop iGAiVA were iterations guided by the information-theoretic method for improving visual analytics workflows systematically [255]. Our analysis of symptoms, causes, remedies, and side effects is reported in Section 4.3.

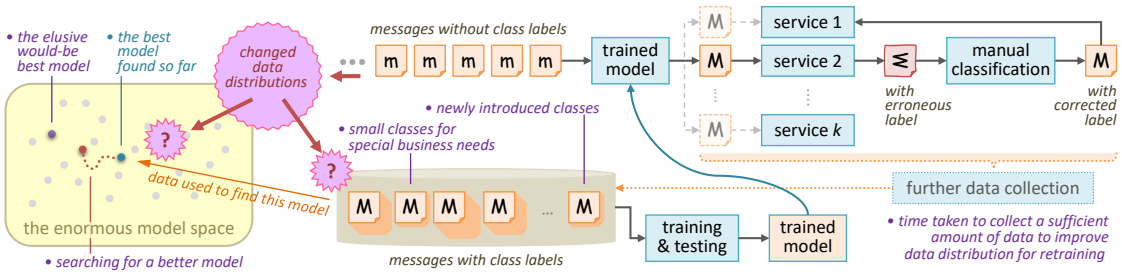


Figure 4.2: A ticketing system (top) operating in an organization, and a traditional ML workflow for developing an organization-specific model for the ticketing system. After development, the performance of the trained model often deteriorates due to environmental changes. The ML developers need an effective and efficient way to improve the model without waiting for collecting sufficiently more data.

4.3 Workflow Analysis and Optimization

Although this work focused on an industrial application, the process for making the transformations from the traditional ML workflow operating in the industrial partner of the project to the iGAiVA workflow, as shown in Figure 4.1, was guided by the systematic design method for visual analytics workflows proposed by Chen and Ebert [255]. The method, which is underpinned by information-theoretic cost-benefit analysis [254], has been used in dozens of case studies, including optimizing visual designs in ML workflows [189, 199]. Since the cost-benefit analysis can also be applied to non-visualization processes in an ML workflow [330], we utilized this method more extensively beyond visualization designs [189, 199]. We describe below the analysis of symptoms, causes, remedies, and side-effects (the terms used in [255]) during the process of formulating the three requirements in Section 4.2.

Initial Analysis. Figure 4.2 illustrates a ticketing system operating in an organization, and a traditional ML workflow for developing a text classification model to be embedded in a ticketing system. The FabLab team in Inetum (Spain), led by the company collaborator, identified the need to find efficient methods for improving ML models in different ticketing systems developed for different organizations many years ago. The need was discussed with the university partner when the university partner visited Inetum in October 2023. The symptoms and possible causes can be summarized as:

- **Symptom 1**: The performance of ML models is affected by the organizational or environmental changes in places where ticketing systems are deployed. The possible cause is that the data statistics used to train these models may become dated, and therefore the models become dated.
  ⇢ Information-theoretically, *potential distortion* is too high (see [255] for detailed explanation).

  One intuitive **remedy** is to collect more data to retrain the models with updated data statistics.

- **Symptom 2**: Data collection for improving an ML model is not always feasible. This hinders the potential **remedy** just mentioned. With the knowledge of the ticketing systems, we reasoned the cause as the *time* delay in collecting sufficient data to produce meaningful statistics and the *cost* for a client organization to collect such data and remove sensitive information.
  ⇢ Information-theoretically, *cost* is too high.

Visual Analytics. When I started my two-month placement in the FabLab team in Inetum in November 2023, I was first given one example dataset (i.e., Table 4.1). I generated multiple t-SNE plots (one example is shown in Figure 4.4), and compiled various performance statistics about different classes together with a variety of simple plots (e.g., Figure 4.3). Although the ML developers computed performance statistics routinely, they were intrigued by the use of many visualization plots. This indirectly indicated a symptom:

- **Symptom 3**: Visualization was not widely used in the traditional ML workflow operating at that time. As statistics often compress information too quickly while analyzing data in detail without visualization may be costly, one promising **remedy** would be to use visual analytics where visualization and algorithms work together. For example, a t-SNE plot is an integration of a dimensionality reduction algorithm and a scatter plot.
  ⇢ Information-theoretically, too much *alphabet compression*.

While a t-SNE plot was useful for observing distribution patterns of the 15 clusters, it has some side effects, which are also symptoms in further iterations:

- **Symptom 4a**: It is not easy to introduce further colors for encoding training vs. testing data points and correct vs. incorrect data points. The **cause** is due to the humans' limited ability to distinguish many colors, which is known to almost all VIS practitioners.
  ⇢ Information-theoretically, *potential distortion* is too high.
  ⇢ Information-theoretically, not enough *alphabet compression*.

- **Symptom 4b**: It is not easy to have a stable and consistent feature space for all subsets of data, e.g., per class, testing data only, etc. The **cause** is that the available t-SNE software does not allow detailed control of the feature space.
  ⇢ Information-theoretically, too much *alphabet compression*.

- **Symptom 4c**: If one did create t-SNE plots for different subsets of data. One could not relate one plot to others easily.
  ⇢ Information-theoretically, cognitive *cost* is too high.

We therefore introduced the coupling of PCA (Principal Component Analysis) and scatter plot as a **remedy**. PCA can be computed once for the whole dataset, and the formulae of different components (feature dimensions) can be applied to different subsets of data consistently. This allowed us to produce many class-based plots similar to Figure 4.5(a). When visualizing these class-based plots, we observed many different patterns of testing errors. Some patterns indicated that erroneous data points were clustered together, but the training data of the same class were relatively sparse in the area where the erroneous cluster appeared. A few other patterns were also discussed in Section 4.5. Sparseness of training data is one of the main causes of unreliable statistics. This partly confirmed the cause of **Symptom 1**.

Major Breakthrough. The FabLab team in Inetum was researching the potential uses of large language models (LLMs) in text analysis. They concluded that

deploying generic LLMs in individual ticketing systems would not be cost-effective, nor would training many organization-specific LLMs. Because the visual analytics approach, especially the coupling of PCA and scatter plot, allowed ML developers to identify erroneous clusters, it became feasible to address the underlying problem (i.e., sparseness of training data) locally. In other words, the **remedy** is:

1. Visualization helps identify small areas in the feature space, where the error rate is high and the training data is sparse.

2. Using LLMs to generate synthetic data based on examples of training data to address the sparseness of training data in such small areas.

This remedy addressed the cause of **Symptom 1** by improving data statistics only in areas of concern, and at the same time addressed the cause of **Symptom 2** by enabling a model to be improved frequently without the need to collect a large amount of real-world data. The initial experiments during the placement indicated the usability and feasibility of this remedy. Further experiments after the placement provided conclusive confirmation.

iGAiVA Software Design. Despite the successful experiments, the visual analytics method could not be deployed easily into the current workflow, because:

- **Symptom 5**: ML developers were not used to data visualization, and without suitable software, they would not adopt the visual analytics approach naturally. ⇢ Information-theoretically, *cost* is too high.

This led to the decision to develop iGAiVA as a *remedy* to reduce the cost of adopting a visual analytics approach. During the design of iGAiVA, we also identified a few minor side-effects of the coupling PCA and scatter plot (i.e., symptoms for the next iteration), and the introduction of RBF and Tag-Treemap was to provide remedies for these minor side-effects.

Table 4.1: As an example, a real-world dataset was collected from a computerized ticketing system. 39,100 labeled messages are divided into 15 classes, and the column "Size" indicates the number of messages. 80% of the data was used for training, and 20% for testing. The column "Recall" shows the results of testing the best CatBoost model developed with the traditional ML workflow in Figure 4.1(a).

T[i]	Topic Name	Size	Recall
T1	IT support and assistance	8529	0.976
T2	Account activation and access issues	11350	0.943
T3	Password and device security	4719	0.892
T4	Printer issues and troubleshooting	1387	0.769
T5	HP Dock connectivity issues	2755	0.732
T6	Employee documentation and errors	1888	0.528
T7	Access and login issues	1963	0.714
T8	Opening and managing files/devices	1028	0.508
T9	Mobile email and VPN setup	1466	0.626
T10	IT support and communication	1699	0.427
T11	Error handling in RPG programming	471	0.926
T12	Email security and attachments	358	0.375
T13	Humanitarian aid for Ukraine	180	0.178
T14	Internet connectivity issues in offices	764	0.526
T15	Improving integration with Infojobs	543	0.376
	All data objects	39100	0.821

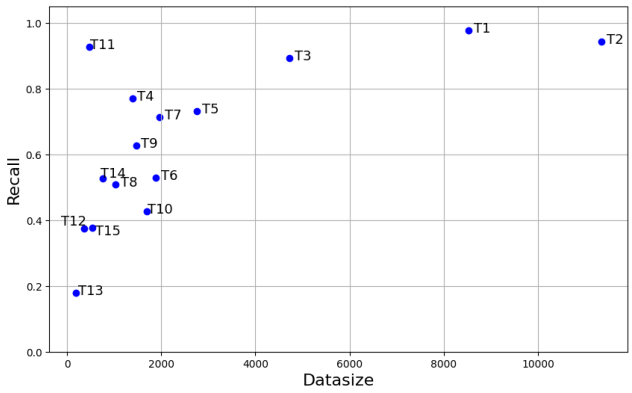


Figure 4.3: Source data size vs. Recall

4.4 Visual Analytics for Problem Identification

While the technical work reported in this chapter is applicable to most computerized ticketing systems, as an example, we focus on one specific system in this Chapter. This ticketing system deals with IT-related problems. As illustrated in Table 4.1, according to the organizational structure, incoming messages are to be classified into 15 task categories and assigned to the relevant IT specialists.

Before advanced VIS techniques were deployed, the team of ML developers worked with the traditional ML workflow as shown in Figure 4.1(a) for many months in conjunction with a dataset consisting of 39,100 labeled data objects (messages). The team used two ML methods, namely gradient-boosted decision trees (using CatBoost) and a convolutional neural network (CNN, using TensorFlow), to train the models in conjunction with 80% randomly selected data objects, and tested with the remaining 20% data objects. A large number of iterations were focused on hyperparameters of the training process. The recall column in Table 4.1 shows the results of testing the best CatBoost model obtained using the traditional ML workflow. Note that each class T_i has p_i labelled positive messages and $n_i = \text{Total} - p_i$ labelled negative messages. Hence, $n_i \gg p_i$, especially for some small classes. In this work, we focus on recall that is not affected by $n_i \gg p_i$, though we also used other performance metrics (e.g., accuracy, F1-score, and so on).

As shown in Figure 4.3, one of our initial visualizations revealed some noticeable correlation between the number of data objects in a class and the recall of the class. We suspected that there were some data sampling issues and therefore focused our attention on data distribution. While visualizing summary statistics (e.g., Figure 4.3) can suggest aspects to be investigated, it is necessary to conduct a more detailed investigation into the potential causes of the high error rates in some classes.

As there are ≥ 180 data objects in each class, visualization alone could not prioritize what to visualize, and interactively reading every message is excessively costly in terms of time and cognitive effort. Meanwhile, texts are rich in semantics and their meanings are context-sensitive; commonly available algorithms cannot encode an adequate amount of human knowledge for dealing with messages in a specific ticketing system. This is exactly the scenario where statistics, algorithms, visualization, and interaction cannot work alone effectively but have to be co-deployed in a workflow [331], which is the principal motivation of visual analytics (VA).

For the requirement **R1** discussed in Section 4.2, we use four types of VA techniques for organizing data objects using analytical algorithms and observing different patterns visually. From the visual patterns, we hypothesize potential

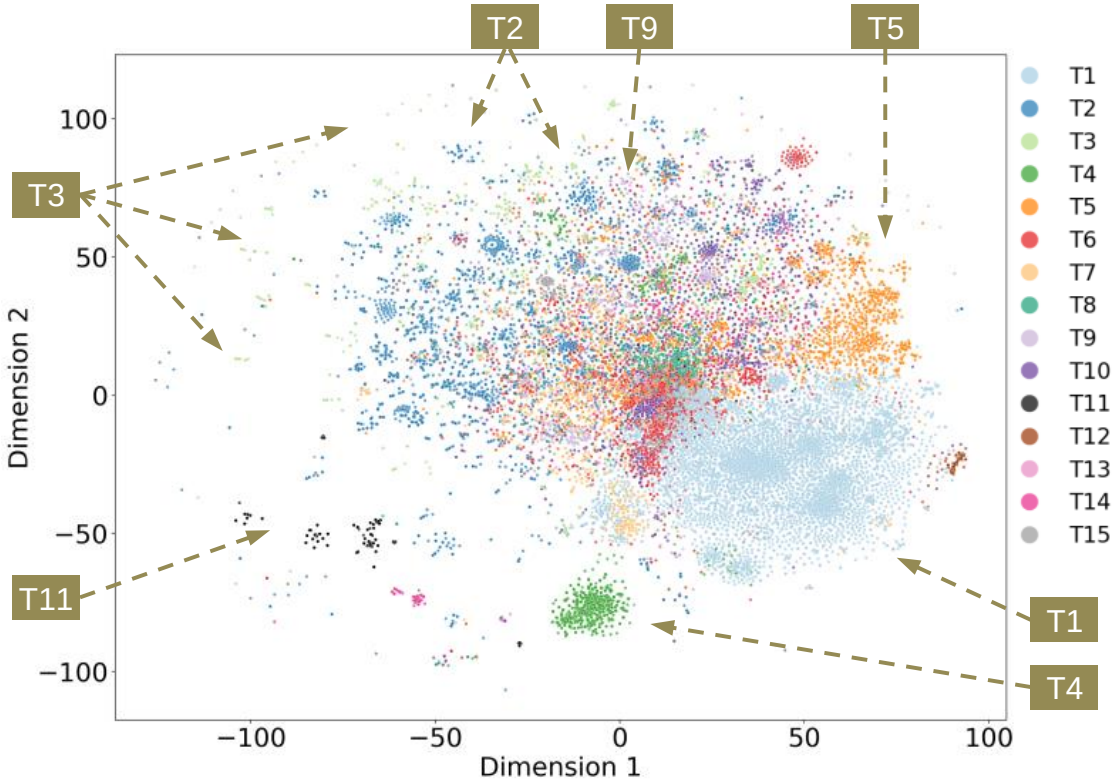


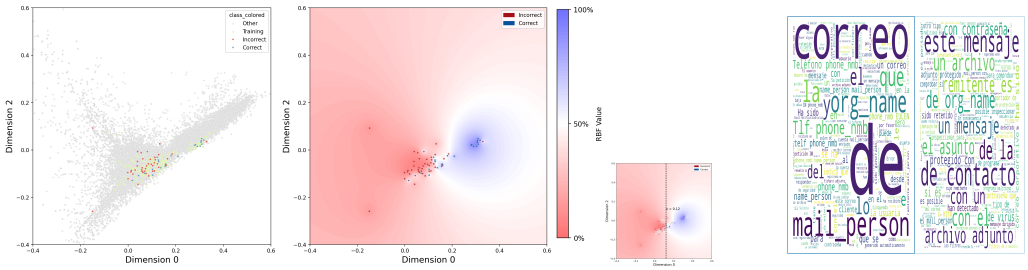
Figure 4.4: Visual patterns in a t-SNE scatter plot can offer some hints about the data-related causes of accurate or erroneous classification.

causes of errors and use statistics to evaluate such hypotheses. In the following four subsections, we describe how each type of VA enables pattern discovery and hypothesis generation.

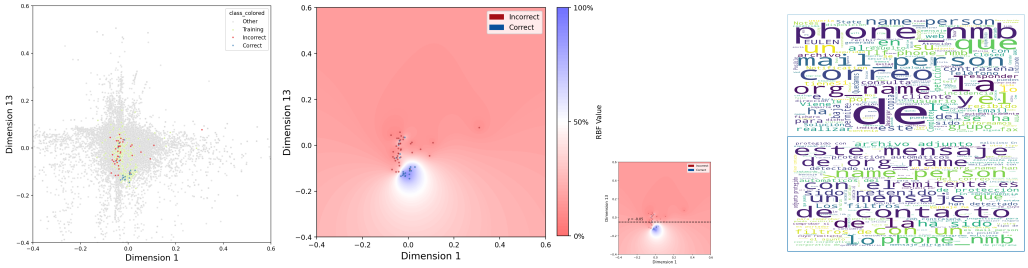
4.4.1 t-SNE(t-distributed Stochastic Neighbor Embedding)

t-SNE is an unsupervised non-linear dimensionality reduction that constructs a k-D data model based on a n-D dataset (typically $n \gg k$). When $k = 2$, the data model is commonly visualized using a scatter plot. Figure 4.4 shows the application of t-SNE to the aforementioned dataset captured from an IT ticketing system. From the figure, we can observe two groups (or kinds) of patterns, and each group has three subgroups:

P1: Data objects in a class (**P1a**) are clumped together, e.g., classes T1, T4; (**P1b**) form many small clusters, e.g., T2; or (**P1c**) are very much scattered around, e.g., T3.



(a) Class T12 PCA dimensions 0 (7.98%) and 2 (4.06%), from left to right: PCA scatter plot, RBF heatmap, division line, tag-treemap



(b) Class T12 PCA dimensions 1 (5.61%) and 13 (0.74%), from left to right: PCA scatter plot, RBF heatmap, division line, tag-treemap

Figure 4.5: Two examples of detailed visual analysis for investigating class T12. The two PCA scatter plots on the left show that Dimension 0 in (a) and Dimension 13 in (b) can separate the data objects into two regions, and data objects in one region have higher recall, while the overall class recall is only 37.5%. Each RBF plot in the second column makes the boundary between the high-recall and low-recall regions clearer, enabling the selection of a division line to study the summary statistics of the messages in the two regions using a tag-treemap on the right.

P2: Data objects in a class are (**P2a**) relatively isolated from other classes, e.g., T11; (**P2b**) partly isolated and partly mingled, e.g., T5; (**P2c**) mostly mingled with other classes, e.g., T9.

In a t-SNE scatter plot, when data objects in a class are clumped together or isolated from other classes, it indicates that the two feature dimensions formulated by the t-SNE dimensionality reduction algorithm can be used to distinguish this class from others relatively easily. Likely, a reasonably good ML model can learn similar features and hence classify this class accurately. When data objects in a class are mingled with others or scattered around, it indicates that the two feature dimensions used for plotting the data objects are not ideal for distinguishing this class from others. It suggests that an ML model may have some difficulties in classifying this class. However, since the ML model will likely learn many other

features, some of which could still help achieve good performance. For example, in Figure 4.4, the visual patterns for classes T2 and T3 are not ideal, but their classification results are fairly good (Table 4.1). The ML model must have learned some useful features that are not characterized by the t-SNE plot.

4.4.2 PCA (Principal Component Analysis)

In order to visualize the data distribution of each class in relation to a good number of features, we use PCA to extract K features (we set $K = 20$ for the data in this Chapter). As shown in Figure 4.5, we use a scatter plot to visualize the data points in each class with the x and y axes corresponding to two dimensions d_i, d_j such that $d_i, d_j \in [0, K-1]$, and $d_i \neq d_j$. In particular, we color the data objects that we correctly classified in blue and otherwise in red. As the visual context, data objects in other classes are shown in grey.

We start with viewing the scatter plots for those classes with relatively low recall (Table 4.1). For each of such classes, we first examine $K-1$ scatter plots with consecutive dimensions (d_0, d_1), $(d_2, d_3), \ldots, (d_{18}, d_{19})$, ensuring that all dimensions are observed. When we identify more than one interesting dimension (e.g., d_0 and d_6), we also examine additional scatter plots (e.g., (d_0, d_6)). When we examine these scatter plots, we look for the following patterns:

P3: An area has more red dots than blue dots.

P4: An area has only a few dots, some of which are red.

P5: An area has red dots scattered among grey dots.

In Figure 4.5, there are two examples of PCA scatter plots, which show patterns of **P3** and **P5**. It focuses on class T12, within PCA dimensions 0 and 2 in (a) and dimensions 1 and 13 in (b). The data objects used for training are shown in yellow-green, testing data objects are shown in blue (if correct) or red (if incorrect), and the data objects in other classes are shown in grey to provide a holistic context.

In (a), the colored dots on the left are largely red (incorrect), scattered among grey dots. In (b), the colored dots in the upper part exhibit a similar pattern.

In the context of the two feature dimensions plotted, **P3** indicates that a group of misclassified messages shares similar features. This leads to hypothesis **H1:** *Is it possible that the ML model relies heavily on features similar to these two PCA features?*

P4 indicates that there are a few messages in the same class that may differ from others in the same class (in terms of the two dimensions concerned). If this pattern is repeated for most dimensions, especially among the top dimensions (the first a few ones) corresponding to the principal components that encode more information about the data, this leads to hypothesis **H2:** *Is it possible that the ML model has not learned suitable features for classifying these messages?*

P5 indicates that some messages in the class may appear quite similar to those in other classes. Similar to **P4**, if the pattern is repeated among most dimensions, including the top dimensions, this leads to hypothesis **H3:** *Is it possible that the ML model relies on these features for classifying messages in some other classes, but they may not be suitable for classifying messages in this class?*

ML model developers usually have some knowledge and skills to investigate the hypotheses **H1**, **H2**, and **H3** further numerically.

4.4.3 RBF (Radial Basis Function)

From the PCA scatter plots in Figure 4.5, one cannot always judge easily whether there are more red or blue dots if the numbers of correct and incorrect results are not significantly different, because counting demands costly cognitive effort. Furthermore, ML developers are often curious about areas between sampled data objects in a class, since PCA assumes that each feature dimension is continuous, and potentially, there are other messages that could have feature values between the known values of two sampled messages. We therefore use a radial basis function (RBF) to estimate the recall error rate across the 2D feature space depicted in a 2D PCA scatter plot. In Figure 4.5, the second column shows two heatmaps after

applying an RBF to the two PCA scatter plots on the left. We superimpose the testing data objects, as dots, on top of the RBF heatmap. This allows viewers to judge if a pixel in the heatmap is further away or closer to the tested data objects, providing intuitive and implicit uncertainty information about the colors in the RBF heatmap. From an RBF plot, one can observe the following patterns:

P6: An area that has a high or low recall error rate (when there are many data objects nearby).

P7: An area that may **potentially** have a high or low recall error rate (when not many data objects are nearby).

When one is viewing an RBF heatmap, usually one cannot help but hypothesize **H4:** *Will there be dots between the two known dots?* Technically, the hypothesis is: *Are there messages whose feature values will fall between the values of known messages?*

4.4.4 Tag-Treemap: Combining Tag Cloud and Treemap

In a PCA scatter plot, when two dots are closely located, it indicates that, in terms of the two principal components (feature dimensions) depicted by the scatter plot, the two text messages (data objects) are similar. However, viewers may not be able to interpret what the similarity means in terms of the texts concerned, e.g., uses of words, sentence structure, and so on. While one can always read the actual messages to consolidate the interpretation of distances in a PCA scatter plot, this can be time-consuming. We thus combine two visual representations, tag cloud and treemap (referred to as *tag-treemap*), to depict k tag clouds associated with k subsets of texts that belong to the same parent set.

For example, we can define all testing data objects in a class as a parent set. We can consider two subsets, for the data objects classified correctly and those incorrectly. We can also divide testing data objects based on their features. In Figure 4.5, the two RBF heatmaps can be divided into two regions, and we can enrich the relatively abstract patterns in the PCA scatter plots and RBF heatmaps by juxtaposing the tag clouds for these regions, as shown on the right of Figure 4.5.

Each of these two tag-treemaps reveals the characteristic difference between the two regions. In (a), the tag clouds on the left and right convey the keyword statistics of 38 and 26 text messages, respectively. We can observe that the right (i.e., the more accurate part) has a much more even distribution of keywords than the left. Likely, the ML model has learned some features similar to PCA Dimension 0 and may treat these two parts differently. In (b), the tag clouds above (33 messages) and below (31 messages) show similar divergence between the two parts, though the divergence is not as substantial as in (a). We also use similar tag-treemaps to check whether the testing data has the same keyword statistics as the training data and all data objects in the class. In this case, the checking confirms the similarity. If there were noticeable divergences, the ML developers could resample the testing and training data to ensure that the two datasets could represent the whole class.

Using tag-treemaps, one can observe:

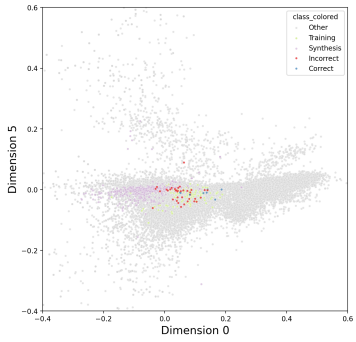
P8: The summary statistical patterns about the keywords in two or more groups of data objects.

For example, one can compare (**P8a**) a group of closely clustered data objects with other data objects in the same class, (**P8b**) red dots vs. blue dots, (**P8c**) different classes, and so on.

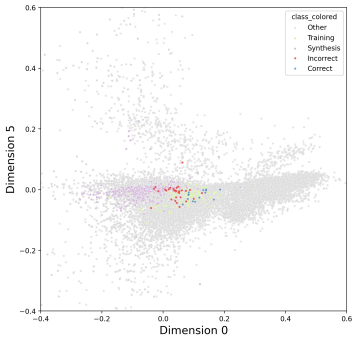
In the next section, we will discuss that the hypotheses **H1**∼**H4** lead to the approach of data synthesis.

4.5 LLMs for Synthesizing Training Data

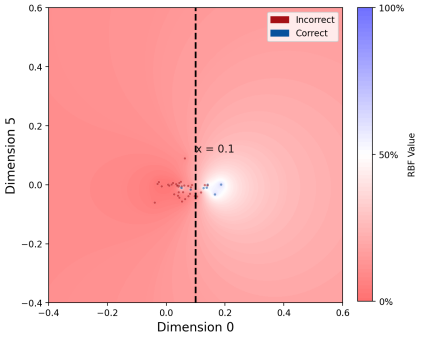
As illustrated in the second workflow in Figure 4.1, the VIS process helped ML developers formulate hypotheses about data-related causes of erroneous results, allowing ML developers to shift their focus from fine-tuning hyperparameters during the previous months (in the first workflow) to improving training data. As it was not feasible to collect more training data, we decided to experiment with synthetic data generated using large language models (LLMs), i.e., requirement **R2** (Section 4.2).



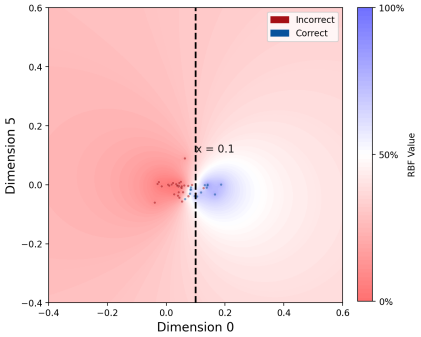
(a) correctness distribution (before)



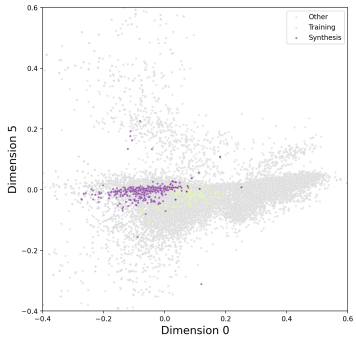
(d) correctness distribution (after)



(b) two area divided by dimension 0



(e) noticeable improvement in RBF



(c) distribution of synthesized data

(f) zoomed PCA scatter plots

Figure 4.6: The class T13 has the lowest recall among all classes. The scatter plot in (a) indicates more classification errors (red dots) when the data objects are associated with lower values in PCA feature dimension 0. The RBF heatmap in (b) confirms this pattern and enables data synthesis to be targeted at an erroneous cluster on the left, as shown in (c). The model retrained with additional LLM-synthesized data is improved in (d). The RBF heatmap for the new testing results in (e) and the zoomed-in scatter plots confirm the improvement.

Although some low-performing classes may appear to suffer from class imbalance, our objective differs from conventional imbalance-handling techniques such as loss reweighting or global oversampling. Those methods primarily adjust the optimization objective or overall class frequency without modifying the underlying

Table 4.2: Example testing results with synthetic data. The **test** column shows the number of data objects in the testing dataset, which is used to obtain all testing results in the table. The **train** columns show the number of data objects for training. In the five cases of synthetic data, a +num indicates the number of synthetic text messages that were added to the specific class in the training data, while the blank cell indicates that no synthetic data was added to a class. The five **Δ-recall** columns show the difference between the original recall (column 4) and recall values obtained from testing the five models trained with the corresponding training data (i.e., original training data + synthetic data). More testing results can be found in Appendix .1, including favorable comparison with tests based on randomly selected examples.

Topic class	Original Data			T11-s1		T12-s1		T13-s1		T14-s1		T15-s1	
	test	train	recall	train	Δ-recall	train	Δ-recall	train	Δ-recall	train	Δ-recall	train	Δ-recall
Overall	782	031280	0.821		▲0.003		▲0.009		▲0.009		▲0.001		▲0.004
T1	1748	6781	0.976		▼0.002		▲0.003		▼0.001		▲0.005		▲0.005
T2	2259	9091	0.943		▲0.002		▲0.004		▼0.003		▼0.003		▲0.001
T3	927	3792	0.892		▼0.009		▼0.005		▼0.004		▼0.019		▼0.012
T4	281	1106	0.769		▲0.039		▲0.060		▲0.028		▲0.010		▲0.050
T5	533	2222	0.732		▲0.020		▲0.005		▲0.009		▲0.015		▼0.023
T6	373	1515	0.528		▲0.030		▲0.059		▲0.038		▲0.035		0.000
T7	395	1568	0.714		▲0.013		▲0.040		▲0.025		▼0.020		▲0.018
T8	236	792	0.508		▼0.055		▼0.046		▲0.077		▼0.046		▼0.067
T9	278	1188	0.626		▲0.011		▲0.054		▼0.018		▼0.032		▲0.032
T10	335	1364	0.427		▲0.006		▲0.018		▲0.095		▲0.033		▲0.048
T11	94	377	0.926	+110	▲0.021		▲0.010		0.000		0.000		▲0.010
T12	64	294	0.375		0.000	+325	▲0.031		▼0.016		0.000		▼0.078
T13	45	135	0.178		▲0.155		▼0.022	+525	▲0.133		▲0.066		▲0.066
T14	135	629	0.526		▼0.037		▲0.007		▼0.022	+1265	▲0.148		▲0.015
T15	117	426	0.376		▼0.034		▼0.060		▼0.034		▼0.034	+1485	▲0.094

feature distribution, whereas our approach enriches the training data by generating new samples in localized regions identified through visual analytics.

A Generative Pre-trained Transformer (GPT) is a type of LLM. GPT-3, which was developed by OpenAI, includes an "attention" mechanism that is able to predict which segments of input text are most relevant, enabling LLMs to focus on these segments [332]. GPT-3.5 is a subclass of GPT-3 Models. Based on the GPT-3.5 architecture, Inetum developed a customized LLM, which was optimized for specific tasks and scenarios. The 2023-03-15-preview version of the GPT-3.5 API was used in the development. This customized LLM can take some text examples and generate new realistic texts in a range of topics, including the 15 topics in Table 4.1.

For example, consider a message in the class T12,

> "*Telefono:* <phone number> *Solicita que todas las carpetas de drive de las cuales era propietario mail person (ha causado baja en la empresa) pasen a su usuario mail person*"

which is one of the red dots in the PCA scatter plot in Figure 4.5(a), was misclassified by an existing ML model. We can use this as a piece of input text to the Inetum LLM. With appropriate parameters such as "temperature = 0.7", "max tokens = 550", "top p = 0.5", "frequency penalty = 0.3", "presence penalty = 0.0", and so on, the LLM model generates k similar messages as the output:

1. *Teléfono: 555-5618 Solicita que se actualice la informacion de contacto en el sistema con su nueva direccion de correo electrónico: new email@company.com*
2. *Teléfono: 555-4312 Solicita que se elimine la cuenta de correo electrónico asociada al usuario "maria perez" y se transfieran todos los correos a la cuenta de "juan rodriguez*
3. ... Other $k - 2$ synthesized messages

Without VIS techniques described in Section 4.4, we would naturally target all classes with relatively poor recall, e.g., T6, T8-T10, T12-T15. We would have to select example text messages (i.e., data objects) randomly from all data objects in a class. With the VIS techniques, we can target data synthesis to more specific subareas in many ways, e.g.,

A. In a t-SNE scatter plot, there is a relatively isolated class, but its classification results are not satisfactory. Alternatively, in a PCA scatter plot, there is a relatively isolated, small, red clump of data objects. Both scenarios suggest that there are distinct features to enable correct classification, but the ML model may not have learned such features, possibly because there is not enough training data in this subarea of the plot. **Possible Action:** We can select example text messages from this subarea and synthesize more data for training.

B. In a PCA scatter plot, when there is a red clump of data objects mingled with grey data objects, it suggests that the ML model may confuse these text messages with those in other classes, possibly because in this subarea, there

are substantially more text messages from other classes (shown in grey) than the class concerned (shown in red or blue). **Possible Action:** We can select example text messages from this subarea and synthesize more to make the training data more balanced in this subarea. We can anticipate that it is possible that this action may improve the testing results for this class, but may impair those of other classes. We hope that the trade-off is in favor of the improvement.

C. In the RBF plot, there are a few scattered red data objects that seem to make a large area red. After a close examination of these text messages, directly or using a tag cloud, one notices that these messages are not uncommon in the real world. Likely, the data collection was not comprehensive enough, making them appear to be outliers. **Possible Action:** We can select these text messages as examples for synthesizing more training data. Because we are still using collected real-world data for testing, the data objects in this subarea remain to be sparse. Hence, the improvement to the testing results may appear to be small. As long as there is improvement, we should consider that the ML model becomes more robust in this subarea.

D. In a tag-treemap, when one compares the keywords of two sets of text messages, one may notice that the one with poor testing results may have less expected ordering of keywords. This suggests a possibility that the collected data in this area may be skewed in favor of certain keywords, causing the ML model to have overlooked some other keywords that can better differentiate this class from other classes. **Possible Action:** We can select text messages with possibly overlooked keywords to synthesize more training data. Similar to scenario B, we need to observe the trade-off between improvement and impairment.

Figure 4.5 suggests that T12 can potentially be divided into two parts based on PCA Dimension 0 or Dimension 13.

As shown in Table 4.1, Class T13 has the lowest recall. One of the PCA scatter plots, as shown in Figure 4.6(a), reveals that the data objects in the class also exhibit two parts, the left part is less accurate than the right. The RBF heatmap in Figure 4.6(b) helps us determine a separation line. We then select example messages from the training data on the left part and generate 525 synthetic messages as additional training data. Figure 4.6(c) shows these synthetic data objects in purple. After retraining the model, the testing results show that the recall of the class improved from 18% to 31%. Figure 4.6(d) indicates noticeable changes in the blue region and the reduction of the shade of red. Figure 4.6(e) juxtaposes two zoomed-in PCA scatter plots, and we can observe the changes of some red dots to blue dots.

The number of synthetic samples was determined based on the subregion after threshold-based separation. Synthetic data were generated proportionally to the number of original samples in that region.

As shown in column T13-s1 in Table 4.2, the synthetic data helps improve the recall of T13 by 13.3%. Adding synthetic data to T13 training data has a positive impact on the recall of the other six classes, including T4 by 2.8%, T5 by 0.9%, T6 by 3.8%, T7 by 2.5%, T8 by 7.7%, and T10 by 9.5%. Although the recall of seven topics was reduced, the overall recall improved by nearly 1%. More experiment results can be found in Appendix .1.

In order to maintain the testing consistently, we define a testing dataset with 20% of data objects from each class. The other 80% of the data is used for training. Because we have some very small classes, e.g., T11-T15, such a training-testing division is unavoidable. As described in Section 4.4, we use tag-treemap to compare the training and testing datasets for each class to ensure that they both represent the class in a similar way. When we generate synthetic data using the LLM, we always select example messages from the training data, as we are aware that selecting examples from testing data could introduce biases in favor of testing.

There are many other scenarios, including those involving the use of multiple plots. For instance, an ML developer may first observe two classes overlapping in a t-SNE plot, and then examine closely the PCA plots for both classes in different

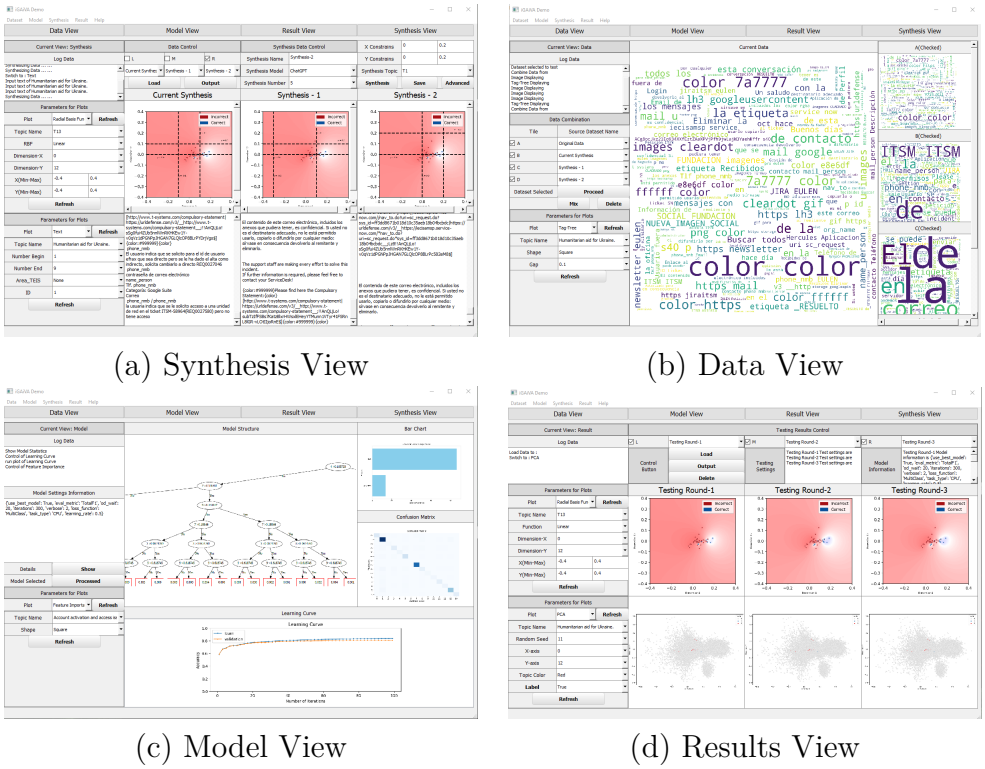
combinations of dimensions. The ML developer may notice that one of the two classes has poor testing results in the subarea of the scatter plots for two specific dimensions. The ML developer may drill down into the details by visualizing the tag-treemap for this subarea, with two tag clouds for the two classes, respectively, leading to a scenario of D. There are many different lines of investigations, and hence the above scenarios A∼D represent only a small subset of possible scenarios.

Likely, the more ML developers have visualized different patterns and conducted experiments to target data synthesis at different pattern-stimulated hypotheses, the more comprehensive knowledge they will gain about different visual patterns, their association with different hypotheses, and their connection with different parameter settings for data synthesis, and their correlation with the performance of the ML models retrained with additional synthetic data.

4.6 iGAiVA: A Tool for Data Synthesis in ML

In the previous two sections, we reported the successful use of VIS and LLM techniques for improving the performance of ML models for text classification. As mentioned in Section 4.2, this successful experience led to the requirement for designing and prototyping a VIS4ML tool where VIS and LLM techniques would be integrated (**R3**). When we were working on the VIS and LLM techniques in the experimental workflow, we used VIS techniques, including PCA scatter plots, RBF heatmaps, and tag-treemaps frequently. On average, we would view at least 10 plots (e.g., Figure 4.5) before a run of LLMs to generate some synthetic data. After the synthetic data was generated, one would visualize the synthetic data (e.g., Figure 4.6(c)) before retraining the model. After the retrained model is tested, we would visualize the testing results (e.g., Figure 4.6(d,e)). We concluded that the VIS techniques should ideally be available in almost every stage of the workflow.

As illustrated in Figure 4.1, we divided the workflow into four major stages: (A) data synthesis, (B) data selection and integration, (C) model training and testing, and (D) results evaluation. Although an ML developer is most likely to commence at stage (B) for selecting the given training and testing data without any



(a) Synthesis View (b) Data View

(c) Model View (d) Results View

Figure 4.7: Four views of the iGAiVA tool. The user can switch between views using the top menu bar. (a) The Synthesis View is for supporting mainly the tasks for identifying suitable example data objects as inputs to LLMs for generating synthetic data. (b) The Data View is for selecting a subset of synthetic datasets and combining them with the original training data. (c) The Model View is for monitoring the process of retraining a model and running the retrained model against one or more predefined testing datasets. (d) The Results View is for analyzing and evaluating the results of the model's performance.

synthetic data, after the first iteration involving (B → C → D), the ML developer will follow the sequence of (A → B → C → D) in most of the subsequent iterations, which typically take more than a few days.

We also notice that the high-level visualization tasks in these four stages are quite different, though some VIS techniques may be required by multiple stages. For example, RBF heatmaps are useful at stage (A) for enabling a user to determine a dividing line as in Figure 4.6(b), and they are also useful at Stage (D) for comparing the results before and after retraining, e.g., comparing Figure 4.6(b) and (e). We therefore identify a set of VIS techniques for each major stage and make these VIS techniques available through a user interface that is designed to suit the tasks at each stage. This results in a design of the VIS4ML tool with four "views"

corresponding to the four major stages of the workflow. We name this VIS4ML tool as *iGAiVA*, which stands for *Integrated Generative AI and Visual Analytics*. Below, we describe the functions of each view of iGAiVA.

Synthesis View for Data Synthesis. In iGAiVA, this view plays a distinct role in supporting an ML workflow involving data synthesis, while the ML tasks supported by the other three views are comparatively common in conventional ML workflows. At this stage, the primary task of an ML developer is to identify a set of example messages that can be used as inputs to LLMs for generating synthetic data. As demonstrated in Sections 4.4 and 4.5, this task can benefit from VIS techniques extensively, with which the ML developer may use a t-SNE scatter plot to select a class to work on, use PCA scatter plots to observe patterns in different combinations of PCA dimensions; use RBF heatmaps to visualize subareas in a more quantitative and predictive manner; use tag-treemaps to compare the keywords statistics of different subareas; and use PCA scatter plots or RBF heatmaps to determine a subarea for selecting examples (from the training data) as inputs to LLMs.

As there are usually a few dozen of the PCA dimensions to be considered, the ML developer is expected to spend a fair amount of time performing interactive visualization. We, therefore, designed this view as a 2×3 matrix as shown in Figure 4.7(a). The interaction panel above the 2×3 matrix is used to control what data to be visualized, such as the whole dataset or a class, which class, how a class is subdivided, previous testing results, training data, or synthesized data; and so on.

The interaction panel on the left of the 2×3 matrix is used to select what VIS techniques to use and set the corresponding parameters. Two VIS techniques can be used simultaneously and selected from a set of visual representations, including t-SNE, PCA scatter plots, RBF heatmap, tag-treemap, and a number of simple plots, as well as viewing text messages directly. In this way, the ML developer can compare up to three pieces of data using two types of VIS techniques at any moment. There are interaction widgets for setting parameters of different VIS techniques, and commands for refreshing two rows of visualization.

There are additional commands and pop-up windows for controlling the data synthesis process, e.g., for randomly selecting examples under specific constraints (e.g., regions defined by PCA dimensions), loading data from the file system, selecting data from the iGAiVA cache area, setting parameters for LLMs, saving generated data to the iGAiVA cache area (to be accessed by all views) and to the file system, and deleting synthesized data that is no longer needed.

Data View for Data Selection and Integration. The primary task at this stage is to have the data ready for training an ML model or retraining the model after adding synthesized data. After the first iteration to train an ML model using only the collected real-world data (referred to as the main dataset), there are usually many iterations for retraining. Normally, one or more new synthetic datasets are generated for a specific class in each iteration. After a number of iterations, there are many synthetic datasets. iGAiVA maintains an internal cache area for storing about 20 datasets. (The actual number depends on the system parameter for the cache area as well as the sizes of the main dataset and the synthetic datasets.) The Data View provides commands and pop-up windows for managing the cache area, e.g., loading (saving) a dataset from (to) the file system, deleting a dataset from the cache, and merging datasets in the cache.

An ML developer can select one or more synthetic datasets from the cache area, and place them on the small tiles on the right of the screen as shown in Figure 4.7(b). These selected datasets are combined with the main dataset (usually the original training data) and are visualized in the large canvas in the middle. For example, in Figure 4.7(b), the smaller tag-treemap on the top-right shows the keyword statistics of the original training data in a class (i.e., the main dataset). The two smaller tag-treemaps below show the keyword statistics of two synthetic datasets. The larger tag-treemap in the middle shows the keyword statistics when the three datasets are combined. In this way, the ML developer can assess the impact of the synthetic data, e.g., to observe whether the level of skewness increases or decreases.

The VIS techniques commonly used in the Data View are PCA scatter plot, RBF heatmap, tag-treemap, and bar charts for keyword statistics. There are tiles

for all selected datasets, including the main dataset. The same VIS technique is applied to the main canvas and all tiles to facilitate consistent comparison. The two interaction panels on the left are for controlling data selection and integration, selecting a VIS technique, and setting its parameters.

Model View for Model Training and Testing. The primary tasks to be performed in the Model View are (i) train or retrain a model and (ii) test the model using a dataset selected or multiple datasets integrated with the Data View. In the Model View, the testing data is normally loaded from the file system, as we normally use the same testing dataset consistently for multiple iterations. There are VIS techniques for monitoring the progress of training and testing.

Usually, training and testing may take some time, and the Model View is designed to focus on unhurried monitoring rather than intensive analysis. As shown in Figure 4.7(c), the VIS techniques used include a line chart for progress monitoring, tree visualization for decision trees or random forest models, and log data display. The Model View supports the monitoring of intermediate testing results, and there are numerical and visual displays showing performance metrics, e.g., accuracy and confusion matrix. It was intentional not to provide more complicated VIS techniques to examine the testing results in detail. As the main tasks in Model View are machine-centric, while the tasks of results analysis and evaluation are human-centric, we purposely designed the Results View for the human-centric tasks.

Results View for Results Analysis and Evaluation. The primary task at this stage is to analyze the testing results from the Model View in order to make some high-level decisions about the next iteration, e.g., go to the Synthesis View to generate another synthetic dataset or go to the Data View to configure a different integration. A major part of the analysis is to compare the latest results with those of the previous models. This is one important reason why the Model View cannot support results analysis easily. As shown in Figure 4.7(d), it has a similar 2×3 matrix layout as the Synthesis View. The main difference is that the interaction panel above the 2×3 matrix is for selecting results data, which is combined with the data objects in the testing data. For example, each data object in the testing

data is automatically annotated with correct and incorrect labels, and the testing dataset can be visualized using a PCA scatter plot or an RBF heatmap. There are also commonly used plots and various statistics about the results.

Using iGAiVA to Conduct Further Experiments. The four views corresponding to the four major steps in Figure 4.1(c). With iGAiVA, the processes described in Sections 4.4 and 4.5 can be carried out iteratively and systematically.

4.7 Evaluation

The evaluation of this work focuses on the three requirements described in Section 4.2. We can rephrase the requirements **R1**, **R2**, and **R3** as evaluation questions: **Q1**: *Can using VIS techniques help identify possible causes of errors?* **Q2**: *Can using LLMs to generate synthetic data for training and using VIS techniques to guide the data synthesis process?* **Q3**: *Can iGAiVA – a VIS4ML tool for supporting ML workflows involving data synthesis - help industrial ML developers?*

Q1 was largely answered by the subsequent developments in this work. Firstly, readers with VIS and ML knowledge can easily evaluate our reasoning of data-related causes at the high-level through plots in Figures 4.3 and 4.4, and at the low-level through plots in 4.5 (i.e., class level in specific feature dimensions). Readers familiar with Chen and Ebert's method [255] can also audit the reasoning about the causes and remedies in Section 4.3. Importantly, it was the success in addressing **R1** that led to the identification of **R2**. The positive answer below to **Q2** also provides a concrete confirmation to **Q1**.

As the workflow shown in Figure 4.1(b) resulted from **R2**, we used extensive testing to answer **Q1**. In addition to the five experiments in Table 4.2, further 29 experiments are reported in Appendix .1. In particular, we compared VIS-assisted experiments with "random" experiments where examples for generating synthetic data were randomly selected. The results of iteration 1 (Tables 4.2 and 1 vs. Tables 5 and 6) showed that the VIS-assisted approach resulted in more "effective synthetic data" for improving ML models:

Improvement	VIS-assisted	Random
Target classes (successes)	10 of out 10	7 of out 10
Target classes (changes)	$[2.1\%, 14.8\%]$	$[-1.6\%, 7.9\%]$
Overall (successes)	7 of out 10	4 of out 10
Overall (changes)	$[-0.2\%, 0.9\%]$	$[-0.3\%, 0.3\%]$
Both at the same time	7 out of 10	2 out of 10

Although some numerical improvements may appear modest at the overall level, the gains were consistent across iterations and more substantial for previously underperforming classes (e.g., T13). Our objective was to demonstrate reproducible performance improvements enabled by VIS-guided targeted synthesis. The comparison with randomly selected synthetic data further indicates that the observed improvements are unlikely to be attributable to random fluctuation.

For the second and third iterations, where gaining improvement became harder, the VIS-assisted approach continued to yield positive improvements (Tables 2, 3, and 4), while it is probabilistically rare for random experiments to yield positive improvements.

In addition, we applied the VIS-assisted approach to a different ML application, i.e., time series classification and statistical data augmentation methods. The application background and experiment results were reported in Section 4.8, which evidences the general applicability.

A comprehensive answer to **Q3** will involve the transformation of iGAiVA to a piece of industrial software and a significant change to the current industrial workflow. In this work, we engaged with potential users of the iGAiVA by following the *user evaluation* protocol outlined by [333]. Because of the space constraint, a detailed report on user evaluation can be found in Appendix .2. Here we briefly summarize the evaluation process, outcomes, and further actions.

Process. In Phase 1 (2 months), the three requirements were identified in multiple steps during my placement in the FabLab of Inetum, Spain. There were eight in-person meetings (two formal and six informal, 2∼15 people) during this phase, and weekly online meetings involving the university partner. I wrote two reports and gave two presentations on the VIS and LLM solutions as part of the Phase 1 evaluation.

Table 4.3: Four sets of testing results for jittering augmentation. The first seven rows are the data within the improvement workflow, including internal testing results. The bottom two rows are external testing results.

Activity class	Original Data test	train	recall	Jitt-vis-t1 trainΔ-recall	Jitt-rand-t1 trainΔ-recall	Jitt-vis-t2 trainΔ-recall	Jitt-rand-t2 trainΔ-recall
Overall (int)	181,907	545,719	0.770	0.000	0.000	0.000	0.000
Bicycling (int)	1762	5287	0.615	+124 ▼0.001	+124 ▲0.002	+638 ▲0.017	+638 ▲0.012
Mixed (int)	23117	69352	0.606	▲0.013	0.000	▲0.014	▼0.003
Sit-stand (int)	70688	212061	0.829	▼0.005	0.000	▼0.006	0.000
Sleep (int)	68016	204046	0.904	0.000	0.000	0.000	0.000
Vehicle (int)	6876	20628	0.381	▲0.008	▼0.001	▲0.009	0.000
Walking (int)	11448	34345	0.199	0.000	0.000	0.000	▲0.002
Overall (ext)			0.770	0.000	0.000	0.000	0.000
Bicycling (ext)			0.615	0.000	▲0.003	▲0.011	▲0.011

In Phase 2 (4 months), the design and development of iGAiVA were carried out at Oxford, UK. There were weekly in-person meetings and several online meetings involving the company collaborator, who had a 1-month visit to the first, and the university partner at Oxford. The evaluation was focused on the testing results and the design of iGAiVA.

In Phase 3 (2 months), I had a further 1-month placement at Inetum, where the iGAiVA tool was evaluated by ML researchers and developers through several meetings (up to 15 people) and individual interviews (5 people). This was followed by efforts for software improvement and technical documentation.

Outcomes. While the quantitative testing results (Section 4.5 and Appendix .1) allowed domain experts to draw a convincing conclusion that the VA+LLM-based workflow, i.e., Figure 4.1(b), is effective and efficient, the qualitative evaluation of the iGAiVA enabled domain experts to plan for developing iGAiVA into an industrial software product. The evaluation of three requirements described in Section 4.2 is detailed in Appendix .2.

Further Actions. In addition, the work on iGAiVA inspired further technical development in aspects of machine learning. It is anticipated that the new advancement in machine learning will be integrated into the future development of iGAiVA.

Table 4.4: Four sets of testing results for permutation augmentation. The first seven rows are the data within the improvement workflow, including internal testing results. The bottom two rows are external testing results.

Activity class	Original Data test	train	recall	Perm-vis-t1 trainΔ-recall	Perm-rand-t1 train Δ-recall	Perm-vis-t2 trainΔ-recall	Perm-rand-t2 train Δ-recall
Overall (int)	181,907	545,719	0.770	0.000	0.000	▲0.001	0.000
Bicycling (int)	1762	5287	0.615	+124 ▲0.001	+124 ▲0.003	+638 ▲0.012	+638 ▼0.001
Mixed (int)	23117	69352	0.606	0.000	0.000	▼0.001	▲0.012
Sit-stand (int)	70688	212061	0.829	▼0.001	0.000	▼0.001	▼0.005
Sleep (int)	68016	204046	0.904	▲0.001	0.000	▲0.001	0.000
Vehicle (int)	6876	20628	0.381	▲0.002	▼0.001	▲0.003	▲0.006
Walking (int)	11448	34345	0.199	0.000	0.000	0.000	▲0.001
Overall (ext)			0.770	0.000	0.000	0.000	0.000
Bicycling (ext)			0.615	0.000	▲0.004	▲0.010	▼0.004

Table 4.5: Four sets of testing results for scaling augmentation. The first seven rows are the data within the improvement workflow, including internal testing results. The bottom two rows are external testing results.

Activity class	Original Data test	train	recall	Scal-vis-t1 trainΔ-recall	Scal-rand-t1 trainΔ-recall	Scal-vis-t2 trainΔ-recall	Scal-rand-t2 trainΔ-recall
Overall (int)	181,907	545,719	0.770	0.000	0.000	0.000	▼0.001
Bicycling (int)	1762	5287	0.615	+124 0.000	+124 0.000	+638 ▲0.003	+638 ▲0.004
Mixed (int)	23117	69352	0.606	▲0.013	0.000	▲0.004	▲0.006
Sit-stand (int)	70688	212061	0.829	▼0.005	0.000	▼0.002	▼0.005
Sleep (int)	68016	204046	0.904	0.000	0.000	▼0.001	0.000
Vehicle (int)	6876	20628	0.381	▲0.006	0.000	▲0.015	▲0.005
Walking (int)	11448	34345	0.199	0.000	0.000	▲0.004	▲0.001
Overall (ext)			0.770	0.000	0.000	0.000	▼0.001
Bicycling (ext)			0.615	0.000	0.000	▲0.004	▲0.002

4.8 Second Application: Time Series Classification

In this section, we report another application where the VIS was used to assist in the generation of synthetic data in order to improve ML models for time series classification. In this case, synthetic data was generated using statistical data augmentation methods. The experiments reported here focused on the variations from the experiments reported in the main body of the Chapter, i.e., from text messages to time series, from text classification to time series classification, and from LLMs to statistical data augmentation. In this way, we demonstrate the general applicability of the VIS-assisted approach presented in the Chapter.

4.8.1 Background and Dataset Description

The Oxford Capture-24 dataset is an open dataset that comprises movement data collected from 151 participants in the Oxfordshire area between 2014 and 2016. The

data was captured using the Axivity AX3 wrist-worn activity tracker. Participants wore the device continuously for approximately 24 hours, resulting in nearly 4,000 hours of tri-axial accelerometer sensor data. Ground truth activity labels were obtained using Vicon Autograph wearable cameras and Whitehall II sleep diaries, leading to more than 2,500 hours of annotated data. The labels of the activities are *bicycling*, *mixed activities*, *sit-stand*, *sleep*, *vehicle*, and *walking* [334]. More details about the data and its collection process can be found at the project's GitHub repository at `https://github.com/OxWearables/capture24`.

4.8.2 ML Process

The dataset was used to develop a classification model using ML. In our experiments, we used the decision tree method and used scikit-learn to train our classifiers. We utilized a set of 40 features extracted from tri-axial accelerometer data provided by the Capture-24 dataset, including:

- 23 statistical features
- 1 temporal feature
- 9 angular features
- 5 spectral features
- 2 peak features

Because this is a relatively large dataset and each class has at least 8,811 data objects, we conducted our experiments using a 60:20:20 data division, i.e., with two testing datasets for internal and external testing.

We focused on recall as our primary evaluation metric, in alignment with the metric for evaluating text classification models discussed in this Chapter. Our workflow is:

1. We trained our baseline model (M0) using the training dataset (60% data).

2. M0 was tested using the internal testing data (20% data). It was assigned as the **current model**.

3. We used a VIS-assisted approach and random approach to generate synthetic data in conjunction with three data augmentation methods, namely *jittering*, *permutation*, and *scaling*. The VIS approach involved the visualization of the internal testing results, mainly with training data as the context. Examples were selected only from the training data.

4. For each set of synthetic data D[i], we incorporated it into the training data and retrained the model, resulting in a new model M[i]. It was then tested using the internal testing dataset.

5. Given many sets of synthetic data generated using different data augmentation methods based on different examples selected using the VIS-assisted approach or randomly, the best model was selected according to their internal testing results.

6. If the best model improves upon the **current model** (Step 2), the best model is then replaced by the **current model** for the next iteration with Steps 3∼5.

7. After a number of iterations, the improvement became difficult to achieve, and the improvement process was terminated.

8. Finally, we tested all models using the external testing dataset (20%).

4.8.3 Synthetic Methods

In this section, we report our experiments with three data augmentation methods: *jittering*, *permutation*, and *scaling*. The parameters for each method were fixed to ensure a consistent augmentation process across all experiments.

Jittering: The jittering method adds noise to an example time series to create synthetic data objects. In our implementation, we use Gaussian noise. For each axis (x, y, and z), random noise with a mean of 0 and a standard deviation of 0.05 is generated and added element-wise to the data. This process

simulates the inherent sensor noise and helps the model become more robust to slight variations in the input signals. The standard deviation parameter ($\sigma = 0.05$) was chosen based on preliminary experiments to balance noise injection without overly distorting the original signal.

Permutation: The permutation method rearranges segments of an example time series to create synthetic data objects. In our implementation, we divide an example time series into four segments (with `nPerm=4`), while ensuring that each segment has a minimum length of 10 data points (`minSegLength=10`). These segments are then randomly shuffled and concatenated to form new data objects. The permutation method preserves the local characteristics of the example time series while altering the global.

Scaling: The scaling method simulates the change of signal strength while maintaining its temporal structure and other local characteristics. In our implementation, we first normalize an example time series and then apply different scaling factors to the normalized time series to generate synthetic data objects.

Scaling is implemented by normalizing the time series data. The scaling function computes the mean of each series and then adjusts the data by subtracting this mean and dividing by a scaling parameter. In our implementation, we set the scaling parameter to $\sigma = 0.2$.

4.8.4 Testing Results and Discussions

As examples, Table 4.3 shows the results of four experiments of jittering, two VIS-assisted and two random. Table 4.4 shows the results of four experiments of permutation, two VIS-assisted and two random. Table 4.5 shows the results of four experiments of scaling, two VIS-assisted and two random. These 12 experiments all targeted the *bicycling* class. The performance of the benchmark model trained with 60% of the data achieves 0.770 overall recall. The performance of the *bicycling* class is 0.615 recall. The internal and external testing results were consistent. These experiments were designed to compare three different data augmentation methods.

We first conducted a number of experiments with relatively small synthetic datasets, as many cases of improvement in text classification involved small synthetic datasets, which also cost less to complete a cycle from data synthesis to ML training and testing. However, from the examples in Tables 4.3, 4.4, and 4.5, we can observe that when the synthetic dataset had 124 data objects (two middle columns), the VIS-assisted approach performed similarly to the random approach, and occasionally, it performed worse. We reasoned that the primary cause is that all synthetic datasets are rather small in comparison with the amount of training data, and their impact on the training was small.

When we increased the size of the synthetic datasets, we noticed that the VIS-assisted approach began to gain advantages. The right two columns of Tables 4.3, 4.4, and Table 4.5 show the cases where synthetic datasets had 638 data objects.

We also noticed that when the size of the synthetic datasets increased, the three methods also began to show different levels of performance, i.e. **jittering performs better than permutation, which is better than scaling.**

Because of the large synthetic data, the more the computational cost for data synthesis and ML training, and testing, and we are still conducting some costly experiments where synthetic datasets contain thousands and tens of thousands of data objects. Note that with the text classification problem, we often generated more synthetic data objects than training data objects for a small class, especially in iterations after the first iteration. From the experiments conducted so far, we found that the VIS-assisted approach was hugely useful in the following ways:

- It is somehow more critical to use VIS to analyze the distribution patterns of correct and incorrect testing results, and to guide the selection of examples from the training data. Due to the high computational cost of the time series classification problem, the inefficiency of the naive trial-and-error approach is much more serious and noticeable than with the text classification problem.

- Randomly-generated synthetic data can sometimes result in positive improvement of the model. This happened more often with the time series classification

problem than with the classification problems. We are using VIS to analyze the distribution patterns of these "helpful" randomly-generated synthetic datasets.

- The VIS-assisted approach also prompted us the need to develop more effective visual representations. For example, we have not found a visual representation like the tag-treemap for text data. Inventing such a visual representation in the future will be hugely helpful for analyzing time series data in ML workflows.

4.9 Visual Analytics for AutoML

Building on iGAiVA, I further collaborated with Chenhao Xue to complete work on an automated text synthesis data analysis system. The following content will briefly introduce this work and clarify my contributions to it.

In addressing the challenges of class imbalance and data scarcity in text classification, this research introduced AutoGeTS (Automated Generation of Text Synthetics), a novel framework designed to enhance model performance through the automated generation of synthetic text data. These challenges are particularly pronounced in industrial applications, such as IT ticketing systems, where small or underrepresented classes often hinder the development of robust models. By leveraging the capabilities of large language models (LLMs), specifically GPT-3.5, AutoGeTS systematically generates synthetic data to augment the training process and improve classification metrics, especially for rare or infrequent classes.

The workflow begins with training an initial baseline model and identifying specific classes that require improvement, typically those with low balanced accuracy, recall, or F1-scores. To address these performance gaps, the framework selects optimal subsets of examples from the target class using advanced data selection strategies. The selected examples are then processed by the LLM to generate synthetic text samples, which are subsequently validated and added to the training dataset. This iterative process ensures continuous improvement of the model as

new synthetic data is incorporated. The selection of input examples for synthetic data generation is a critical step, as it directly impacts the quality and effectiveness of the synthetic samples. To this end, AutoGeTS employs three primary strategies for selecting data subsets: Sliding Window (SW), which exhaustively evaluates all potential subsets; Hierarchical Sliding Window (HSW), which narrows the search space through a hierarchical approach; and Genetic Algorithm (GA), an evolutionary optimization method that balances exploration and exploitation in identifying impactful subsets.

A key aspect of the research lies in optimizing the objectives of data generation and model retraining. Depending on the specific needs of the task, AutoGeTS can target improvements in class-specific metrics, such as recall or balanced accuracy, or prioritize global metrics like overall balanced accuracy and F1-score. For highly imbalanced real-world datasets, the ensemble algorithm proposed in this study enables tailored application of the most effective strategies and objectives for each class. By iteratively applying the framework across multiple underperforming classes, AutoGeTS balances local (class-specific) and global (overall) performance improvements, ensuring minimal trade-offs between classes.

My contribution focused on the synthetic data generation process, which played a necessary role in the overall framework. Specifically, I provided the synthetic text samples using GPT-3.5. The generation process involved configuring the LLM with certain parameters which align with iGAiVA's settings, such as a temperature of 0.7 and a sample size of five synthetic examples per original input. Input prompts were crafted to guide the LLM in producing contextually appropriate text while maintaining variability in content. For example, prompts were formatted to include clear instructions for generating structured ticketing system entries while introducing subtle variations. After generation, automated validation processes were applied to ensure the quality of the samples, checking for empty responses, formatting errors, and adherence to specified structures. If any issues were detected, the process was repeated with adjusted prompts to achieve the desired outputs.

My work ensured the generation of synthetic data that not only preserved the contextual relevance of the original examples but also provided sufficient diversity to enhance model generalization. This contribution was integral to the success of AutoGeTS in improving classification metrics across all class sizes, particularly for those most affected by data imbalance. The framework's ability to automatically optimize synthetic data generation with minimal human intervention represents a significant step forward in the development of scalable and effective ML systems for text classification.

4.10 Conclusions

In this Chapter, we have presented a visual analytics solution for enabling targeted data synthesis in ML workflows. We used VIS techniques that closely coupled analysis and visualization, such as dimensionality reduction and scatter plot (t-SNE), PCA and scatter plot, RBF and heatmap, and keyword statistics coupled with tag cloud and treemap. The code for RBF has been open-sourced on GitHub https://github.com/MattJin19/RBF. The close coupling of VIS and LLMs processes in a workflow represents a high-level integration, demonstrating the successful deployment of VA concepts.

Through this Chapter, we have demonstrated that with VIS techniques, it has become feasible to target data synthesis to "areas", which ML developers can observe visually and can use their knowledge to reason about the possible causes of poor performance and hypothesize potential improvement if synthetic data were introduced in these "areas". We have demonstrated that discovering such areas can effectively lead to the improvement of the ML models concerned.

To our best knowledge, this approach has not been reported previously in the literature. We are continuing this work mainly in the form of developing a piece of industrial software based on this approach. Meanwhile, we anticipate that there must be many other visual patterns that could suggest some types of causes of poor performance, and there must be some other iterative pathways from improving the performance of a single class to improving the overall performance.

We believe that the more ML developers can use VIS techniques, the more effective pathways will be discovered.

Further discussions on *scalability*, *potential challenges with LLMs*, *lessons learned*, and *limitations* can be found in Appendix .3.

iGAiVA addresses the inconsistency between development and deployment data spaces for text classification. By coupling LLM-based data synthesis with interactive visual analytics, iGAiVA enables ML developers to identify sparse or skewed regions of the development training space and selectively enrich them with synthetic samples. Through the integration of Synthesis, Data, Model, and Results Views, iGAiVA operationalizes the Four Views Design to align training data statistics with deployment requirements. Experiments in industrial ticketing systems show that such VIS-guided data synthesis improves classification accuracy beyond experiment-based optimization alone.

5

TA3: Testing Against Adversarial Attacks

Adversarial attacks are major threats to the deployment of ML models in many applications. Testing ML models against such attacks is becoming an essential step for evaluating and improving ML models. In this chapter, we report the design and development of an interactive system for aiding the workflow of Testing Against Adversarial Attacks (TA3). In particular, with TA3, human-in-the-loop (HITL) enables human-steered *attack simulation* and visualization-assisted *attack impact evaluation*. While the current version of TA3 focuses on testing decision tree models against adversarial attacks based on the One Pixel Attack Method, it demonstrates the importance of HITL in ML testing and the potential application of HITL to the ML testing workflows for other types of ML models and other types of adversarial attacks.

From the perspective of Chapter 2, adversarial examples instantiate a data space inconsistency between the application-specific data space and the artificially designed adversarial data space. Traditional experiment-based optimization approaches primarily focus on model parameter adjustment and algorithmic improvements, but these methods treat data space inconsistencies as fixed constraints rather than addressable problems. This limitation causes traditional approaches to struggle with understanding the underlying causes of model vulnerabilities when dealing with adversarial attacks. In TA3, we adopt the Four Views Design introduced in Chapter 3

and tailor it to adversarial testing. It addresses the inconsistency problem by decomposing adversarial testing into four interconnected analytical perspectives that enable a comprehensive examination of the relationships and discrepancies across the two different data spaces to enhance model robustness against adversarial attacks.

5.1 Background of the Project

The rapid development of ML technology has started to bring forth the deployment of ML models in our daily lives, e.g., face recognition, traffic management, unmanned supermarkets, and autonomous vehicles. At the same time, the quality of ML models has received increasing attention. Research on *adversarial attacks* has discovered a variety of vulnerabilities in ML models, ranging from getting a model to mistake a panda for a gibbon [87] to getting Tesla's self-driving car to misjudge the speed limit sign [335]. In the field of ML, the current research effort on adversarial attacks places the main emphasis on discovering new adversarial attack methods and improving ML models' resistance against specific types of adversarial attacks. The former has resulted in a large collection of attacking methods [125], while the latter typically focuses on the training processes, e.g., data distillation [336] and adversarial training [131]. In this work, we focus on ML testing processes against adversarial attacks.

There are differences as well as similarities between testing ML models and testing conventional software. The main differences include (i) deployable ML models are expected to make mistakes while deployable conventional software is expected to be bug-free; (ii) the inner workings of an ML model are expected to be difficult to comprehend, while a handcrafted program is expected to be fully understood. Because of these differences, the testing of ML models has relied primarily on statistical measures resulting from automated test runs, while the testing workflows for conventional software have relied extensively on HITL. Fundamentally, an ML model is also a program. Similar to a handcrafted program, the search space for a potential error in these programs is huge, in that relying on a fully automated process to search for errors is mostly intractable computationally.

Hence, in conventional software testing, HITL allows human experts to inject their knowledge to focus on highly probable parts of the search space. There is no reason why HITL cannot help human experts in testing ML models. This reasoning motivates us to develop an interactive software system to support ML testing workflows against adversarial attacks.

There is a large collection of attacking methods, many of which apply to models trained with a specific algorithmic framework and specific types of training data [337]. There are also several strategies for defending against adversarial attacks, such as *threat modeling*, *attack simulation*, *attack impact evaluation*, *countermeasure design*, *attack detection*, and *defensive training* [338]. These strategies may be implemented in individual techniques operating at different phases of ML workflows, e.g., for *preparing data*, *preparing learning*, *training models*, and *evaluating models* [184]. Hence, developing HITL-support techniques for supporting ML developers against adversarial attacks will be a long-term endeavor. In this chapter, we focus on the phase of *evaluating models* and the strategies of *attack simulation* and *attack impact evaluation*. The software developed in this chapter currently features components for simulating adversarial attacks created by the One Pixel attack and components for visualizing decision-tree-based models. With this chapter, we wish to make the following contributions:

- We introduce a conceptual workflow for testing ML models against adversarial attacks, where, through HITL-support techniques, ML experts can steer the simulation of potential attacks on an ML model and visualize the simulation results to evaluate the attack impact as well as the weak aspects of the model at the level of details that statistical measures cannot provide.

- We develop a prototype system, called TA3 (*Testing Against Adversarial Attacks*), as an implementation of the conceptual workflow for testing decision tree models against One Pixel attacks, as well as for demonstrating the feasibility and the potential of the conceptual workflow.

- We propose a set of evaluation measures for quantifying the vulnerability of ML models, and demonstrate their uses as a statistical overview in conjunction with visual analysis.

5.2 Four Views Design User Interface

While the conceptual workflow in Figure 3.1(b) and the required HITL support for the workflow are generic to most processes for testing ML models against adversarial attacks, the design of some HITL-supporting facilities will need to be optimized for specific categories of ML models. The variations of the design may typically depend on (i) the type of input data to be processed by the ML model being tested since the data observation stage will require different visual representations for different types of data, (ii) the type of the ML model being tested since it may affect how the model is invoked and how the internal structure of the model is to be visualized, and (iii) the type of adversarial attack algorithm used in the testing since different types will likely have different sets of parameters. In this chapter, we focus on a software system, TA3 (*Testing Against Adversarial Attacks*), that demonstrates the feasibility of the conceptual workflow in Figure 3.1(b), while supporting the testing processes against a commonly-studied family of adversarial attacks, i.e., pixel-based attacks on image classification models [339, 340].

Pixel-based attacks introduce perturbations to imagery data by modifying one or more pixels in an image to be processed by an ML model. Since Szegedy et al. first reported such attacks in 2013 [118], many attacking algorithms have been reported in the literature. The TA3 incorporates the One Pixel attack algorithm [95], which uses the computation method of differential evolution to find the pixel that maximizes the probability of an erroneous classification result. While the traditional testing workflow in Figure 3.1(a) may be satisfied by finding the One Pixel, the testing against such attacks would like to find how an ML model would react to other similar attacks in addition to that One Pixel. Hence, we adjust the One Pixel attack algorithm to generate multiple attacks and provide statistical and visual analysis of the performance of the ML model under such attacks.

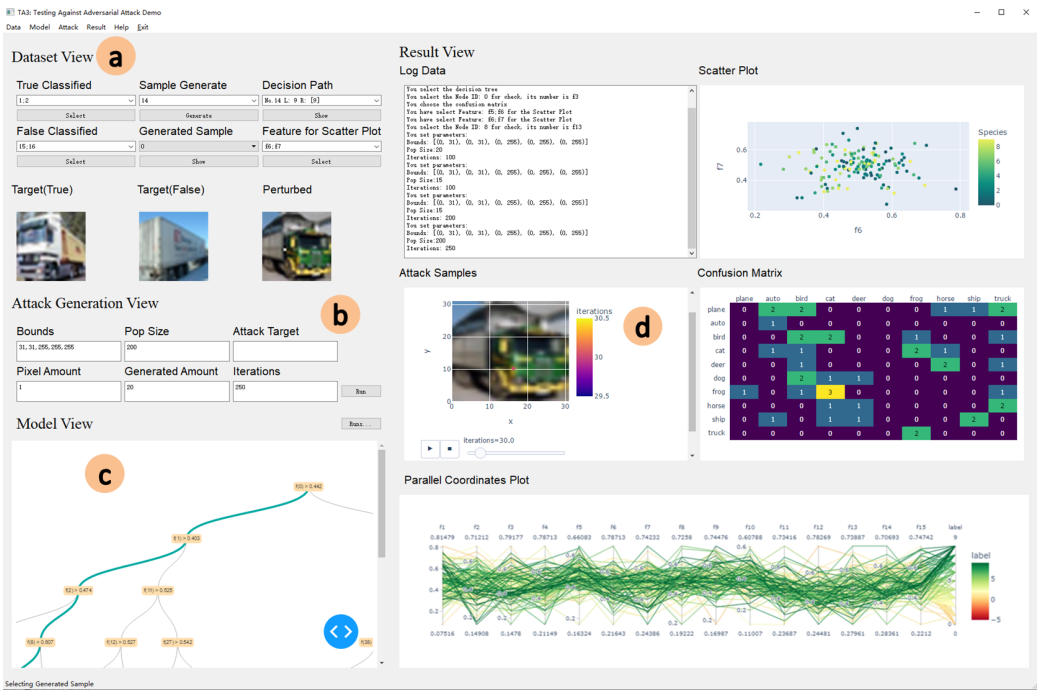


Figure 5.1: The main user interface of TA3, which is an HITL-supporting tool for testing ML models against adversarial attacks. The screen is roughly divided into four areas. (a) Below the menu bar, the *Dataset View* area supports HITL activities related to the data used for testing ML models. (b) The *Attack Generation View* area focuses on the HITL activities for controlling the adversarial attack algorithm and observing its input and output data. (c) The *Model View* area enables the visualization of the testing data flowing through the model structure. (d) The *Results View* area provides HITL-supporting facilities for observing and interacting with different visualization plots.

TA3 was designed based on the conceptual workflow in Figure 3.1(b). As shown in Figure 5.1, the main screen is organized into four parts, referred to as *Dataset View* (top-left), *Attack Generation View* (middle left), *Model View* (bottom-left), and *Results View* (right). These views host the most frequently used HITL-supporting facilities and frequently viewed information in the iterative testing processes. The users can access such facilities and information through the commands in the menu bar, as well as a few shortcut command buttons in individual views. The four views correspond to the four stages in the upper part of Figure 3.1(b). The *Dataset View* was intentionally placed at the top-left part of the screen after some experimentation. The rationale of this design decision will be discussed below in the descriptions of *Dataset View* and *Attack Generation View*.

The *Dataset View* provides HITL-supporting facilities for (d_1) loading a specific

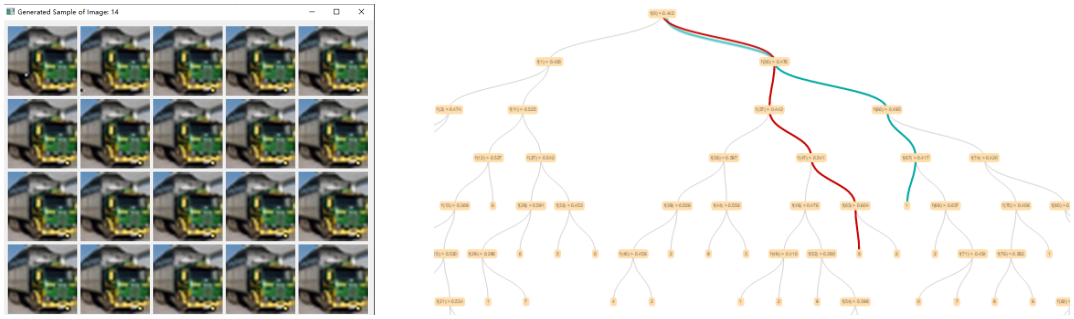


Figure 5.2: A pop-up window showing all perturbed data objects generated by the adversarial attack algorithm. Left: A pop-up window showing all perturbed data objects. Right: A decision tree with true- and false-positive data flows.

testing dataset, (d_2) viewing the summary information about the dataset, (d_3) selecting one or more target data objects in the dataset to be perturbed by the adversarial attack algorithm, (d_4) viewing the data objects in the dataset, (d_5) viewing the selected target data objects, (d_6), viewing the perturbed data objects, and (d_7) selecting the data objects (e.g., all or some data objects (original and/or perturbed)). The function (d_1) needs to be performed before performing any other tasks in the testing process, and partly because of this reason, *Dataset View* was placed at the top-left part of the screen, which is naturally a place to start HITL actions.

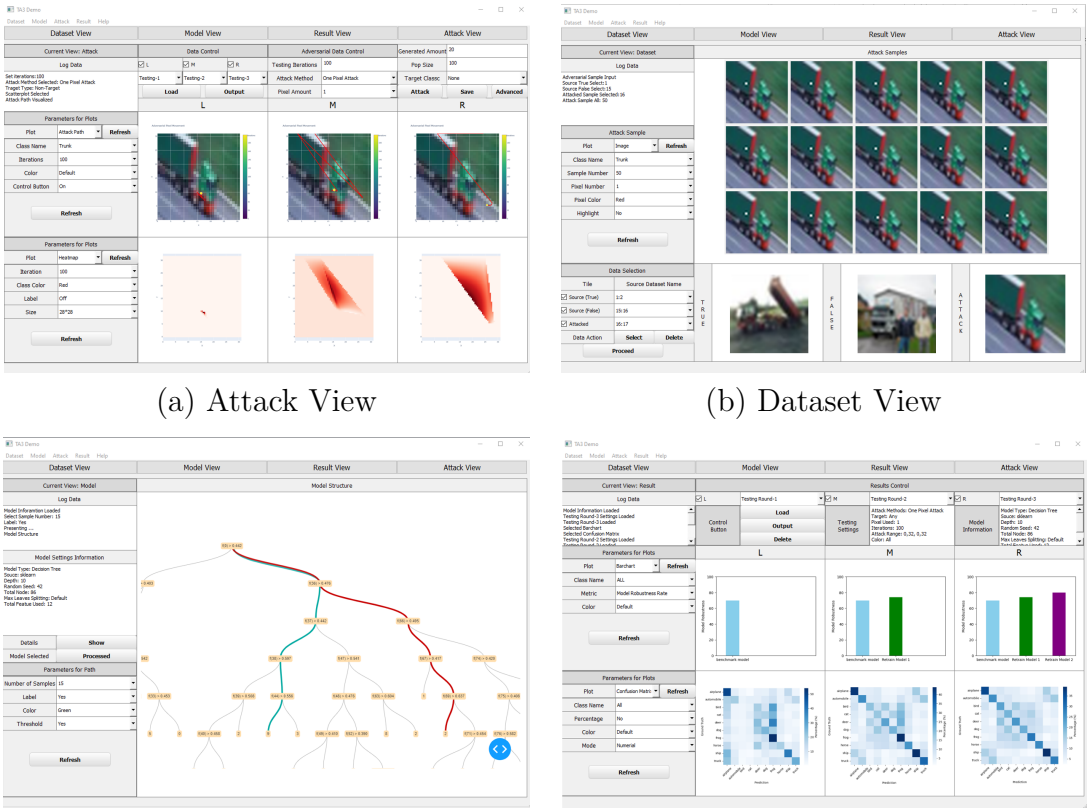
Attack Generation View provides HITL-supporting facilities for (g_1) setting and modifying the parameters of the adversarial algorithm, (g_2) viewing the input data to the algorithm (i.e., the target data), and (g_3) viewing the perturbed data (e.g., the results of applying the algorithm to the target data). The functions (g_2) and (g_3) are more or less the same as (d_4) and (d_5), and therefore, the visual representations of these data objects are shared between the *Dataset View* and the *Attack Generation View*. As the adversarial attack algorithm usually generates many perturbed samples from the same target data object, TA3 provides three visual representations for displaying the perturbed data, including (i) a static representation of a perturbed data object that was generated first by the adversarial attack algorithm, (ii) an animation of all perturbed data objects, and (iii) a static grid view of all perturbed data objects. The first two representations are available

on the main screen, while the third representation is available on demand through a pop-up window, as shown in Figure 5.2(left).

Model View provides HITL-supporting facilities for (m_1) controlling the execution runs of the ML model being tested and (m_2) investigating how the model predictions are affected by the input data objects and the internal structure of the model with the aid of model visualization. The design for (m_2) depends on two main factors, namely whether the ML model is available to the testing processes as a black-box model or white-box model, and if it is a white-box model, which model type (e.g., decision tree, CNN, etc.). The TA3 system was designed for ML developers to evaluate their models in the decision tree family. We thus consider only white-box decision-tree models. The internal structure of the model and its data flow from input to output can be observed using a tree visualization. For example, as illustrated in Figure 5.2(right), users can visualize the data flow for true- and false-positives to see how adversarial attacks redirect some part of the true-positive data flow into wrong paths.

Results View provides HITL-supporting facilities for (r_1) viewing the testing results and summary statistics through a variety of visualizations. For example, as shown in Figure 5.1, there is a scatter plot for displaying the features used in the decision tree, a matrix plot for displaying a confusion matrix derived from the testing results, and a parallel coordinates plot for showing the testing data objects and their feature values. In Section 5.3, we will further discuss the visualization plots and summary statistics available in TA3 in detail.

For TA3's UI, based on the user feedback, we further improved it based on Four Views Design, as shown in Figure 5.3. The improved design strengthened the interaction between the four views and incorporated good aspects from the iGAiVA design, enabling researchers to conduct model robustness testing in a more controllable manner. In Figure 5.3 (a), the *Attack View* provides more intuitive attack parameter settings and perturbation visualization, allowing researchers to adjust parameters while simultaneously observing and comparing the effects of different attack strategies on input data. The figure visualizes different attack



(a) Attack View (b) Dataset View

(c) Model View (d) Results View

Figure 5.3: The updated UI presents each view on a separate screen (a–d). This separation improves clarity, strengthens interaction within each view, and provides a more effective environment for testing an ML model's robustness against adversarial attacks.

paths and analyzes the attacked regions in the image using heatmaps, facilitating researchers' discovery of potential problems in the model's judgment of specific regions. The three attacks shown in the figure are determined by different initial positions of the One Pixel Attack. The leftmost image shows that when the initial position is close to the target point, the algorithm can quickly find the attack location, while if the starting position is farther away, as in the middle or rightmost images, it requires longer attempts to find the attack position, which is also reflected in the heatmap as more red attack regions.

Figure 5.3(b) *Dataset View* supports comparative browsing of original data and generated adversarial samples, helping to identify vulnerable sample types and potential pattern differences. The figure shows comparisons between correctly classified categories, incorrectly classified categories, and adversarial samples. Mul-

tiple possible adversarial samples are generated to help with model training. In the incorrectly classified category, there are people in front of the truck and other interference, which may be the cause of the model's classification error, while the adversarial sample's added white dot is located at the connection between the truck head and cargo compartment, possibly indicating that the model relies on features of variable parts when performing classification. This information will continuously accumulate as researchers conduct multiple rounds of iterative testing under different parameter settings and sample selections, gradually revealing the model's sensitivity patterns to specific perturbations and providing data support for subsequent model improvements.

Figure 5.3(c) *Model View* visualizes the decision-making process of the decision tree model, enabling users to locate vulnerable parts within the model under adversarial inputs. The figure shows the decision path changes between a correctly classified path and an incorrect category path after being attacked. Researchers can quickly locate problematic nodes from the decision tree's decision path, and then further analyze issues with the node, such as threshold size or whether the selected features are reasonable.

Figure 5.3 (d) *Results View* integrates various statistical indicators and confusion matrices, helping researchers conduct performance analysis under different attack and data conditions. The figure shows that across three tests, although the model's robustness rate score steadily increases, the confusion matrices below reveal that the model's actual classification performance has not consistently improved. The confusion matrix in the middle image shows higher recognition of correct categories. This result indicates that there is some deviation between adversarial sample generation and the model's learning process. The model may have achieved higher robustness metric scores by learning attack generation patterns, or its generalization ability in real scenarios has not been correspondingly improved. Through the multi-dimensional result display of Results View, researchers can discover this potential inconsistency, thereby avoiding the problem of one-sided evaluation of model robustness using single metrics.

Through the Four Views Design concept, TA3 overcomes the problems of insufficient parameter exploration and visualization support in the original workflow, and achieves a complete closed loop from attack generation, data observation, model analysis, to result evaluation. Through multi-perspective collaborative visualization, it helps researchers identify model limitations hidden behind single quantitative indicators and provides a reference for subsequent improvement of data generation methods and robustness evaluation approaches.

5.3 Adversarial Attack Methods Implementation and Metrics for Evaluation

Through the user interface, as shown in Figure 5.1, TA3 provides ML developers with access to algorithms, interaction (interactive mechanisms), statistics, and visualization. In this section, we detail these four technical aspects of TA3.

5.3.1 Algorithms: Machine Learning and Attack Simulation

In TA3 and indeed any software of this nature, there are many algorithms. Here, we focus on two families of algorithms, namely machine learning (ML) and attack simulation. Because this work relies on such algorithms available in the public domain, we describe them briefly here.

In the context of ML, there are two main types of algorithms. A trained ML model is an algorithm that is partly designed by human developers (e.g., the structure of a neural network or a set of features of a decision tree) and partly learned from the training data. Meanwhile, an ML software platform for training an ML model hosts the core algorithm that embodies an ML technique, which may be unblended or hybrid and may also be referred to as an ML method or an ML framework.

In this work, we focus on decision tree models, which are particularly suitable for white-box testing. We trained our ML models using the CART algorithm in the scikit-learn library [83]. Given a set of pre-processed feature data, CART constructs a binary tree iteratively by selecting the most information-rich feature

in the iteration concerned and determining the optimal threshold for splitting the data flow into two branches.

For white-box testing, TA3 displays the decision tree structure as well as the relationships between a list of features and data objects (i.e., images in this case) in a parallel coordinates plot. TA3 can accommodate any number or type of features. As an example, the models shown in this Chapter are trained with feature data obtained by using principal component analysis (PCA) [341, 342]. For each image in a training dataset, we use fifteen principal components as features for training each model. These models can easily be replaced by decision tree models that are trained with other types of features, such as HOG [343] or SIFT [344] features. Users can also apply their methods. The datasets used for ML in this work are Cifar-10 [345], Fashion-MNIST [346], and MNIST [347], while TA3 currently can handle any decision tree model for image classification and any image dataset for testing such a model.

Algorithms for adversarial attacks are mostly hand-crafted by experts, though most algorithms contain optimization processes that may be regarded as a kind of ML. There are many adversarial attack algorithms in the literature, and most algorithms have user-adjustable parameters. Different algorithms and different parameters result in different performances, making the testing processes important as well as challenging.

In this work, we focus on one particular attacking algorithm, i.e., the One Pixel attack. We used the open-source implementation provided by Kondratyuk (from Google Research), who made the code available at GitHub [97]. The implementation was based on the original algorithm proposed by Su et al. [95]. Because it is an open-source implementation, we can easily make a small modification to extract some interim data for visualization and to generate multiple attacks for testing ML models.

5.3.2 Interaction: Testing Steering

As shown in Figure 3.1(b), the HITL approach is essential for target data selection, parameter control (for the adversarial attack algorithm), testing runs control,

and visualization control. In general, human-computer interaction (HCI) is an elementary component of HITL. Without HCI, ML developers will not be able to steer the testing process based on their knowledge of the ML models, datasets, and adversarial attack algorithms that are being tested, and their observation and interpretation of the statistics and visualization.

Some of the widgets for these HCI tasks are accessible on the main screen shown in Figure 5.1, while others are accessible in the corresponding pop-up windows. These HCI tasks are:

Target Data Selection

- Dataset Selection – for selecting a dataset to be loaded into the system.
- Candidate Data Object Selection – for selecting two original data objects (images in this work), one is correctly classified by the ML model, and another is incorrectly classified by the model. The correctly classified data object is usually the target of an adversarial attack algorithm. The incorrectly classified data object is a source of risk, perhaps with a more serious one, since it is likely easier to perturb.
- Target Data Object Selection – for selecting an image to be perturbed.

Parameter Control

- Bound – for restricting the 5D search space for a pixel in an RGB image, i.e., x, y, r, g, b.
- Pop Size – for defining the population size of the differential evolution method.
- Target Data Object Selection – for selecting an image to be perturbed.
- Attack Target – for selecting the target to be perturbed. If it is left blank, it is an untargeted attack.
- Num. Pixels – for controlling the number of pixels to be perturbed per attack.

- Num. Attacks – for controlling the number of adversarial images to be generated.
- Iterations – for controlling the number of iterations for finding the pixels to be perturbed.

Testing Runs Control

- Data Flow Selection – for selecting the data flow of a set of images to be tested, with options for various combinations of the target images (0, 1, or 2), generated attack images (0 $\sim$ all), and a subset of the loaded testing data (0 $\sim$ all). This also defines what is to be visualized in the decision tree visualization.
- Run – for activating a simple testing run based on the Data Flow Selection
- Runs ... – for specifying more complex testing involving multiple target images.

Visualization Control

- Features for the scatter plot – for selecting two features to be visualized in the scatter plot.
- Axes movement – for moving the axes within the parallel coordinate plot to enable the easy observation of feature relations.
- Brushing – for selecting and highlighting a subset of data objects (i.e., lines).

Although visualization control seems to be for viewing the testing results rather than steering the testing, visualizing the testing results in one testing iteration will influence the users' decisions about the next iteration and subsequent iterations. Hence, visualization control is a form of indirect testing steering.

Table 5.1: Six new measures designed for testing a model's ability to deal with adversarial attacks.

Measure	General	Positive-specific	Negative-specific
model-robustness rate:	$\frac{\boxdot\text{PPP}+\boxdot\text{PNN}+\boxdot\text{NNN}+\boxdot\text{NPP}}{\boxdot\text{P}+\boxdot\text{N}}$	$\frac{\boxdot\text{PPP}+\boxdot\text{PNN}}{\boxdot\text{P}}$	$\frac{\boxdot\text{NNN}+\boxdot\text{NPP}}{\boxdot\text{N}}$
attack-breach rate:	$\frac{\boxdot\text{PPN}+\boxdot\text{PNP}+\boxdot\text{NNP}+\boxdot\text{NPN}}{\boxdot\text{P}+\boxdot\text{N}}$	$\frac{\boxdot\text{PPN}+\boxdot\text{PNP}}{\boxdot\text{P}}$	$\frac{\boxdot\text{NNP}+\boxdot\text{NPN}}{\boxdot\text{N}}$
adversarial-impact rate:	$\frac{\boxdot\text{PPN}+\boxdot\text{NNP}}{\boxdot\text{P}+\boxdot\text{N}}$	$\frac{\boxdot\text{PPN}}{\boxdot\text{P}}$	$\frac{\boxdot\text{NNP}}{\boxdot\text{N}}$
attack-failure rate:	$\frac{(\boxdot\text{P}-\boxdot\text{PPN})+(\boxdot\text{N}-\boxdot\text{NNP})}{\boxdot\text{P}+\boxdot\text{N}}$	$\frac{\boxdot\text{P}-\boxdot\text{PPN}}{\boxdot\text{P}}$	$\frac{\boxdot\text{N}-\boxdot\text{NNP}}{\boxdot\text{N}}$
intended-perturbation rate:	$\frac{\boxdot\text{PPN}+\boxdot\text{NNP}}{\boxdot\text{PPN}+\boxdot\text{PNP}+\boxdot\text{NNP}+\boxdot\text{NPN}}$	$\frac{\boxdot\text{PPN}}{\boxdot\text{PPN}+\boxdot\text{PNP}}$	$\frac{\boxdot\text{NNP}}{\boxdot\text{NNP}+\boxdot\text{NPN}}$
unintended-perturbation rate:	$\frac{\boxdot\text{PNP}+\boxdot\text{NPN}}{\boxdot\text{PPN}+\boxdot\text{PNP}+\boxdot\text{NNP}+\boxdot\text{NPN}}$	$\frac{\boxdot\text{PNP}}{\boxdot\text{PPN}+\boxdot\text{PNP}}$	$\frac{\boxdot\text{NPN}}{\boxdot\text{NNP}+\boxdot\text{NPN}}$

5.3.3 Statistics: Simulation Summarization

Research papers on adversarial attacks typically evaluate an attack algorithm by measuring the statistics about the successful attacks (e.g., [348]) and the difference between the accuracy of original testing data and the attacked testing data (e.g., [128]). Although such statistical measures support the basic evaluation, they are much too coarse in comparison with commonly used statistical measures in computer vision (e.g., precision, recall, F1-score, and so on). In this work, we define several new statistical measures to enable ML developers to conduct a finer evaluation.

Consider a binary classification model **M** and a set of testing data objects $o_1, o_2, \ldots, o_n$, each of which has a ground truth label "P" (positive) or "N" (negative). When the model **M** is tested against an original data object, o_i, it may return "P" or "N". Similarly, when **M** is tested against an attacked data object $\alpha(o_i)$, it also returns either "P" or "N".

Hence, for each of the n data objects, its testing results can be represented by three letters, representing $\langle$ground truth$\rangle\langle$test against $o_i\rangle\langle$test against $\alpha(o_i)\rangle$ respectively. For example, "PPN" indicates that a data object has a ground truth label "P", **M** classifies it correct as "P" (true positive), and **M** classifies the attacked object as "N" (false negative). There are eight possible cases, "PPP", "PPN", "PNP", "PNN", "NPP", "NPN", "NNP", and "NNN".

Let us add a symbol "$\boxdot$" or "$\boxdot_n$" in front of these to denote the number of each case in testing a total of n data objects. Let us first define two simplified notations:

$$\boxdot\text{P} = \boxdot\text{PPP} + \boxdot\text{PPN} + \boxdot\text{PNP} + \boxdot\text{PNN}$$

$$\boxdot\mathrm{N} = \boxdot\mathrm{NPP} + \boxdot\mathrm{NPN} + \boxdot\mathrm{NNP} + \boxdot\mathrm{NNN}$$

We can easily write down some commonly-used statistical measures for testing $\mathbf{M}$ against the original data and the attacked data as:

■ **Original Data Statistics**

$$\textbf{accuracy:}\quad \frac{\boxdot\mathrm{PPP} + \boxdot\mathrm{PPN} + \boxdot\mathrm{NNP} + \boxdot\mathrm{NNN}}{\boxdot\mathrm{P} + \boxdot\mathrm{N}}$$

$$\textbf{precision:}\quad \frac{\boxdot\mathrm{PPP} + \boxdot\mathrm{PPN}}{\boxdot\mathrm{PPP} + \boxdot\mathrm{PPN} + \boxdot\mathrm{NPP} + \boxdot\mathrm{NPN}}$$

$$\textbf{recall:}\quad \frac{\boxdot\mathrm{PPP} + \boxdot\mathrm{PPN}}{\boxdot\mathrm{P}}$$

$$\textbf{F1 score:}\quad \frac{2 \cdot \boxdot\mathrm{PPP} + 2 \cdot \boxdot\mathrm{PPN}}{\boxdot\mathrm{P} + \boxdot\mathrm{PPP} + \boxdot\mathrm{PPN} + \boxdot\mathrm{NPP} + \boxdot\mathrm{NPN}}$$

■ **Attacking Data Statistics**

$$\textbf{accuracy:}\quad \frac{\boxdot\mathrm{PPP} + \boxdot\mathrm{PNP} + \boxdot\mathrm{NPN} + \boxdot\mathrm{NNN}}{\boxdot\mathrm{P} + \boxdot\mathrm{N}}$$

$$\textbf{precision:}\quad \frac{\boxdot\mathrm{PPP} + \boxdot\mathrm{PNP}}{\boxdot\mathrm{PPP} + \boxdot\mathrm{PNP} + \boxdot\mathrm{NPP} + \boxdot\mathrm{NNP}}$$

$$\textbf{recall:}\quad \frac{\boxdot\mathrm{PPP} + \boxdot\mathrm{PNP}}{\boxdot\mathrm{P}}$$

$$\textbf{F1 score:}\quad \frac{2 \cdot \boxdot\mathrm{PPP} + 2 \cdot \boxdot\mathrm{PNP}}{\boxdot\mathrm{P} + \boxdot\mathrm{PPP} + \boxdot\mathrm{PNP} + \boxdot\mathrm{NPP} + \boxdot\mathrm{NNP}}$$

Essentially, the original statistics ignore the last letter by considering, e.g., true positives as $\boxdot\mathrm{PPP} + \boxdot\mathrm{PPN}$, since the last letter is about the predictions $\mathbf{M}$ for the attacked objects. Meanwhile, the attacking statistics ignore the middle letter (e.g., true positives = $\boxdot\mathrm{PPP} + \boxdot\mathrm{PNP}$), since the middle letter is about the predictions $\mathbf{M}$ for the original objects.

In addition, we can introduce new statistical measures to summarize the successful and failed attacks, as well as intended and unintended perturbations in the testing results. The formulae of these new measures are given in Table 5.1.

The measure **model-robustness rate** summarizes the ability of $\mathbf{M}$ to resist the adversarial attack algorithm by considering those predictions of $\mathbf{M}$ that remain the same before and after the attacks. The measure **attack-breach rate** summarizes

the level of success of the adversarial attack algorithm in altering the predictions of **M**. We have **attack-breach rate** = 1− **model-robustness rate**. Among such perturbations, some cause **M** to change correct decisions to incorrect decisions, and others vice versa. The measure **adversarial-impact rate** summarizes the actual adversarial damages by excluding those attacks that have unintentionally corrected the errors made by a model when applying it to the unperturbed data, i.e., ⊡PNP and ⊡NPN. Meanwhile, the measure **attack-failure rate** summarizes the failed attacks, such that **attack-failure rate** = 1− **adversarial-impact rate**. The measure **intended-perturbation rate** summarizes the damaging attacks among the successful attacks, while **unintended-perturbation rate** summarizes those successful attacks that unintentionally cause **M** to change incorrect decisions to correct decisions. We have **unintended-perturbation rate** = 1− **intended-perturbation rate**.

The above measures are all defined in the same context of commonly-used statistical measures in machine learning by assuming each data object o_i is attacked only once (cf. only tested once). As discussed in previous subsections, in practice, adversarial attacks will likely be applied to the same data object multiple times, and TA3 facilitates the testing against such multi-attacks. We can define a similar set of single-object and multi-attack measures by adapting the above multi-object and single-attack measures.

Let us first consider a single data object o being attacked multiple times; different attacks attenuate the data object differently. When o is attacked for k times, there will be k modified data objects. In many ways, this is similar to having a testing dataset that contains k copies of the same data objects. To distinguish the difference between testing n different data objects with a single attack on each of them and testing the same data object with k attacks, we introduce a new symbol ⊛ (instead of ⊡) to indicate the number qualities in the statistical measures. For example, consider an object with the ground truth label "P", and it would be correctly classified as "P" by **M** normally. ⊛PPN indicates among k attacks, the number of the attacks that made **M** returning the label "N".

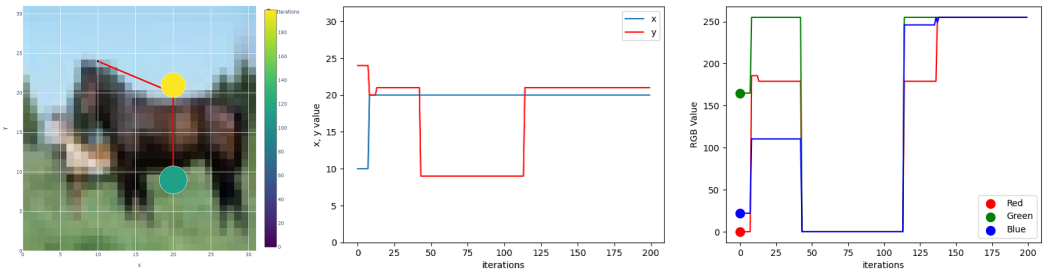
It is necessary to note that the measures for single-object and multi-attacks depend only on the number of quantities related to the specific ground truth label of the data object. For example, if the ground truth label is "P", then $\circledast$P $= k$ and $\circledast$N $= \circledast$NPP $= \circledast$NPN $= \circledast$NNP $= \circledast$NNN $= 0$. The negative-specific measures do not apply to the test of this data object. If the **M** could classify the original object correctly and s ($s \leq k$) attacks were successful, we would have $\circledast$PPN $= s$, $\circledast$PPP $= k - s$, and $\circledast$PNP $= \circledast$PNN $= 0$. We have **attack-breach rate**($\circledast_k$) = **adversarial-impact rate**($\circledast_k$) $= s/k$, and the **model-robustness rate**($\circledast_k$) = **attack-failure rate**($\circledast_k$) $= 1 - s/k$.

Note that the square in $\boxdot$ symbolizes a dataset, while the circle in $\circledast$ symbolizes a data object. The dot indicates a single attack per object, while the $*$ indicates multiple attacks. By extrapolation, we can also use $\circledast$ to symbolize statistical measures obtained by testing n data objects, each of which receives k attacks. In many situations, the computer screen may display a measure without a detailed formula. Therefore, we always indicate the context of the test, with at least one of the three symbols $\boxdot$, $\circledast$, and $\circledast$. If there is sufficient display space, we will indicate the number of objects and attacks, e.g., **attack-impact rate**($\circledast_{1000,10}$) = 0.52 or **attack-impact rate**(1000 objects, 10 attacks) = 0.52.

Although these new statistical measures offer more information than the traditional measures, such as success rate, model developers may wish to observe some details about multi-attacks. Visualization can deliver more information to complement statistics that may abstract data too quickly [331]. This will be discussed in the next subsection.

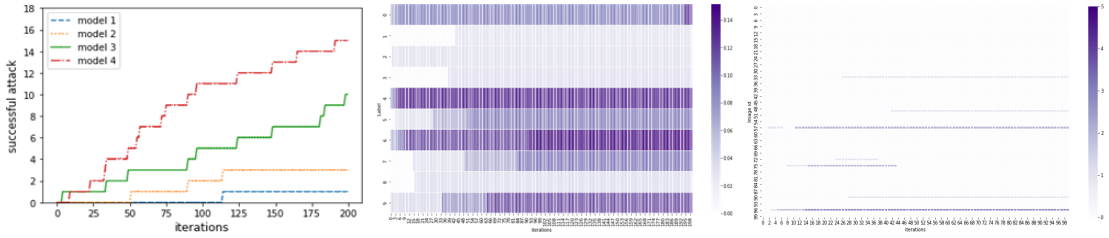
5.3.4 Visualization: Simulation Progression and Testing Results

As shown in Figure 5.1, visualization plots enable ML developers to observe a variety of visualization plots, which display the data captured during model testing as well as the summary statistics about the model performance. Broadly, the data that can be visualized in TA3 falls into the following categories:



(a) path on image (b) x, y-coordinate (c) RGB values

Figure 5.4: Two ways to visualize the movement of the simulated attack points during a test. (a) Intuitive view in the context of the image (Image No. 75) being attacked. (b & c) Detailed progression of the attacking points during the simulation, with color-mapped pixels representing the amount of increasing and decreasing pixel brightness and a yellow background indicating a successful attack.

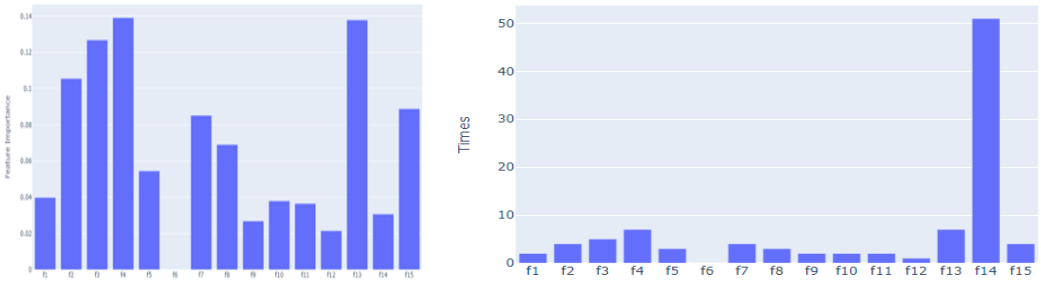


(a) four models (b) 10 classes (c) 100 images

Figure 5.5: Three visualization plots for comparing (a) four models in terms of the number of cumulative successes vs. iterations, (b) compare 10 classes in terms of success rate vs. the number of iterations, and (c) compare 100 images in terms of success rate vs. the number of iterations.

- **Simulation progression: attacking patterns** — During simulation, the perturbed data objects can be visualized directly as shown on the left of Figure 5.4. The movement of the attacking pixels can also be visualized as a path superimposed on an enlarged image and as two time series for x- and y- coordinates. The former enables the observation of attacking pixels in conjunction with the imagery context, while the latter enables the observation of whether there are repeated attacks in a small area.

- **Simulation progression: evolving statistics** — During simulation, various statistical measures are computed for each iteration. As exemplified in Figure 5.5, we can observe (a) the number of successful attacks on the same image during testing different models, (b) the vulnerability of images in



(a) feature importance (b) feature usage in nodes

Figure 5.6: Two example visualizations of model statistics.

different classes, and (c) the vulnerability of individual images. We will show more examples of such plots deployed in our case studies in Section 5.4.

- **Testing results: model statistics** — after simulation, various statistical measures are made available through visualization plots as well as numerical values. This includes some commonly-used ML statistics such as accuracy, precision, recall, F1-scores, and confusion matrix, as well as the statistical measures introduced in Section 5.3.3 for summarizing adversarial attacks. For example, a confusion matrix is shown in the middle-right of Figure 5.1, while two bar charts are shown in Figure 5.6, one for summarizing the importance of features used by an ML model and another to show features used in the nodes.

- **Testing results: analytical visualization** — Some visualization plots are designed to support detailed analysis of ML models being tested. As shown in Figure 5.1, there are scatter plots for observing the possible correlational or independent relations between two features (top right), parallel coordinates plots for examining such relations across many features (bottom right), and data flows in a decision tree (bottom left).

The types of plots currently available in TA3 include:

- bar chart — for visualizing the overall statistics for different categories of testing data. Figure 5.6 shows two such examples. Other uses of this form

of visualization include average success rates for different classes, models, datasets, and so on.

- line plot — for visualizing the evolution of various statistical measures during simulation, such as the number of successful and/or unsuccessful attackers (y-axis) at each iteration, as shown in Figure 5.5(a). Multiple lines can be used to represent different conditions controlled by categorical variables, such as multiple models, multiple datasets, and classes, as well as a small number of discrete instances of a numerical control parameter.

- scatter plot — for visualizing the relationships between two features (e.g., top-right of Figure 5.1 and (b) in Figure 5.4).

- pixel-based plot — for visualizing the confusion matrix of the testing results (e.g., middle-right of Figure 5.1), numerical indicators about the testing results in terms of different categories of data objects (e.g., (b) and (c) in Figure 5.5).

- node-link tree plot — for visualizing the testing data that flows through a decision tree model (e.g., bottom-left of Figure 5.1).

- parallel coordinates plot — for visualizing the testing data objects and features used by the decision models (e.g., bottom-left of Figure 5.1).

5.4 Evaluation, Metrics, and Results

Case Studies

We evaluated TA3 software by testing classification models trained using three open ML datasets, i.e., the Cifar-10 dataset for 6 classes of animal images and 4 classes of transport images [345] as shown in Figure 5.1, the MNIST dataset for 10 classes of handwritten digits [347] as shown in Figure 5.7(a), and the Fashion-MNIST dataset for 10 classes of clothing [346] as shown in Figure 5.7(b).

While the Cifar-10 dataset consists of color images, the MNIST dataset and Fashion-MNIST dataset consist of gray-scale images. Hence, TA3 includes a modified version of the One Pixel attack algorithm, which perturbs gray-scale pixels instead

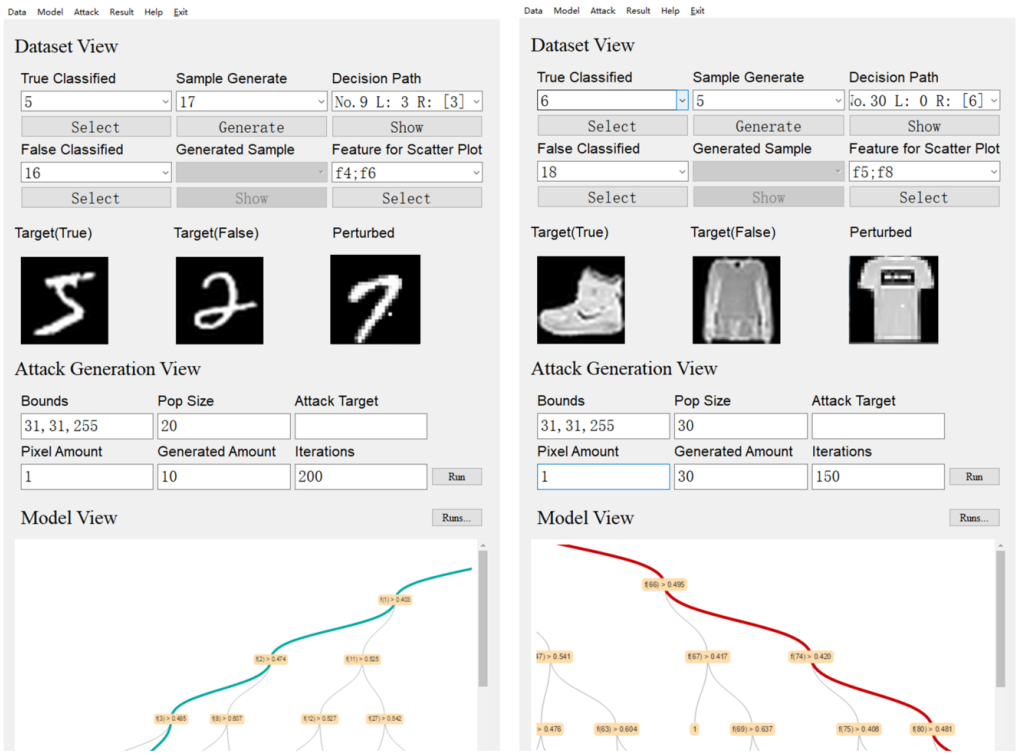


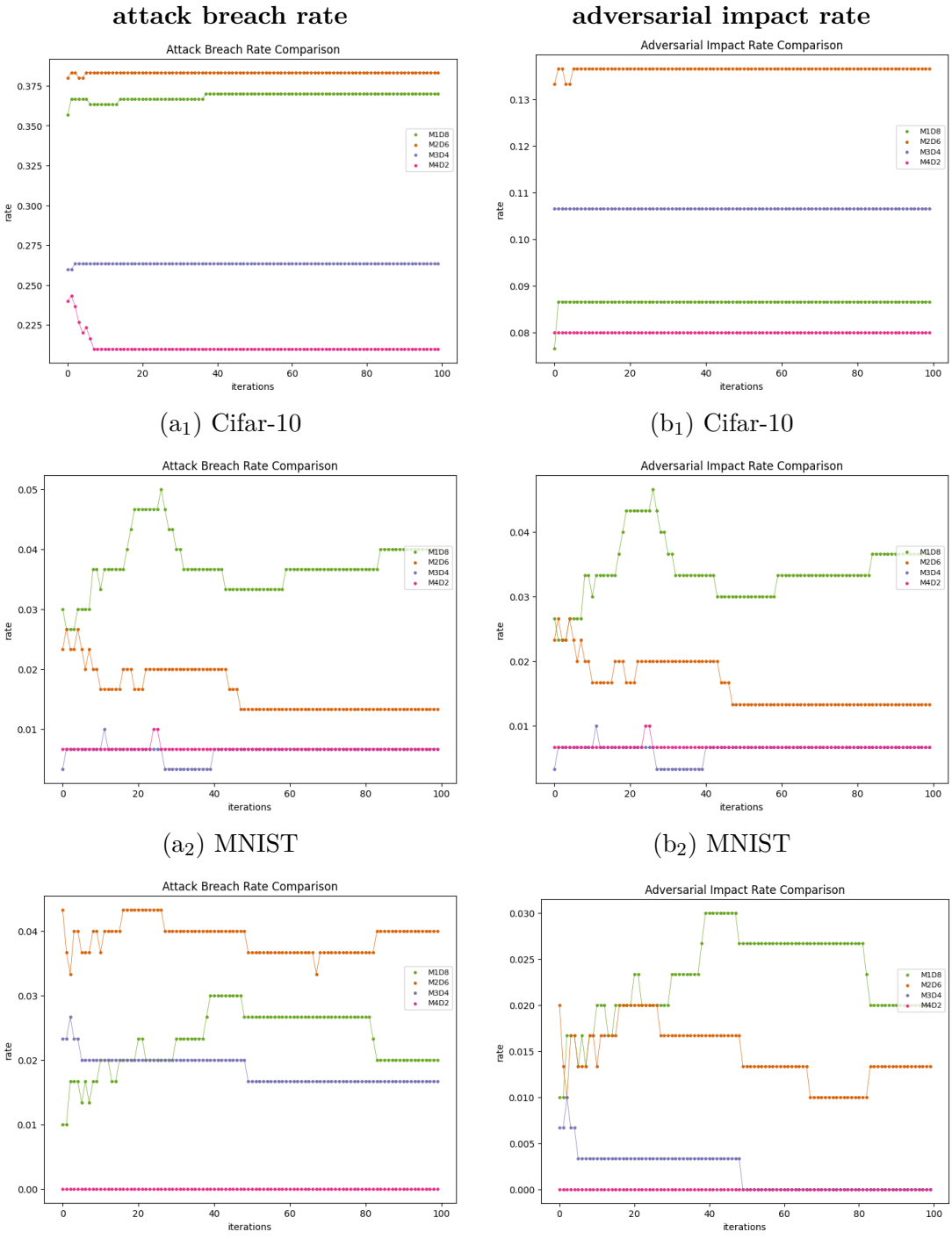
Figure 5.7: In addition to the Cifar-10 dataset featured in Figure 5.1, we also used left (a) the MNIST dataset and right (b) the Fashion-MNIST dataset for training ML models to be evaluated by TA3.

of RGB pixels. This demonstrates the flexibility of TA3 in testing ML models trained with images in different color spaces.

Among these datasets, we trained several models to test various hypotheses about whether hypo-parameters used for the training processes would affect the models' resistance to adversarial attacks. When TA3 is used to test these models, it automatically generates a variety of plots as described in Section 5.3.4.

Hypothesis 1: Does Depth Control Affect the Resistance? For each of the three datasets, we trained four models that are of different tree depths, i.e., depth = 2, 4, 6, and 8, respectively. Figure 5.8 shows two statistical measures, **attack breach rates** and **adversarial impact rates**, when these 4×3 models are tested iteratively, allowing us to observe and compare the performance of different models easily.

It is common knowledge that the depth of a decision tree affects its accuracy. When a tree is too shallow, the model may suffer from under-fitting, and when it is too deep, it may be over-fitting. Consider 12 models, four trained with



(a_1) Cifar-10 (b_1) Cifar-10

(a_2) MNIST (b_2) MNIST

(a_3) Fashion-MNIST (b_3) Fashion-MNIST

Figure 5.8: Testing hypothesis 1 about depth control in training. Four models are of different tree depths, i.e., depth = 2 (M4D2, dark pink), 4 (M3D4, purple), 6 (M2D6, orange), and 8 (M1D8, green), respectively. (a) Comparing the **attack breach rates** of four models trained using each of the three datasets. (b) Comparing the **adversarial impact rates** of four models trained using each of the three datasets.

the Cifar-10 dataset (cM1D8, cM2D6, cM3D4, cM4D2), four trained with the MNIST dataset (mM1D8, mM2D6, mM3D4, mM4D2), and four trained with the Fashion-MNIST dataset (fM1D8, fM2D6, fM3D4, fM4D2). Here, D2, D4, D6, and D8 stand for tree depths of 2, 4, 6, and 8, respectively. Note that we can train models with other depth control parameters, such as at depth 7, 10, 11, and so on, or dynamic depth control. In terms of accuracy, these models are ordered as (F-MNIST is short for Fashion-MNIST):

$$\begin{aligned}
\text{Cifar-10:} &\quad Acc(cM2D6) > Acc(cM1D8) > Acc(cM4D2) > Acc(cM3D4) \\
\text{MNIST:} &\quad Acc(mM1D8) > Acc(mM2D6) > Acc(mM3D4) > Acc(mM4D1) \\
\text{F-MNIST:} &\quad Acc(fM1D8) > Acc(fM2D6) > Acc(fM3D4) > Acc(fM4D1)
\end{aligned}$$

In this context, it would be interesting to find out whether depth control affects the models' performance in terms of defending against adversarial attacks. From the plots in Figure 5.8, we can easily observe that models of different depths have different qualities in both **attack breach rates** and **adversarial impact rates**. However, their ordering may not always correspond to the accuracy ordering.

From Figure 5.8(a), we can observe that the models trained with the Cifar-10 and MNIST datasets apparently exhibit the same ordering in terms of **attack breach rates** and accuracy. In other words, the more accurate the model is, the easier it seems for attacks to be successful. Meanwhile, the **attack breach rates** for the models trained with the Fashion-MNIST dataset have a slightly different ordering from that of accuracy. This indicates that the two measures are not closely dependent.

As described in Section 5.3.3, an adversarial attack can sometimes "correct" an error made by a model (i.e., in the case of PNP and NPN). The measure **adversarial impact rates** excludes such events. From Figure 5.8(b), we can observe that the ordering of the four models changes for the models trained with the Cifar-10 and Fashion-MNIST datasets. In particular, comparing Figure 5.8(b_1) with Figure 5.8(a_1), the green line for M1D8 drops noticeable, indicating some

successful attacks shown in a_1 are actually "unintentional corrections" of model errors. Meanwhile, comparing Figure 5.8(b_3) with Figure 5.8(a_3), we can observe that the orange line for M2D6 exhibits a similar phenomenon. We can also observe that Figure 5.8(a_2) and Figure 5.8(b_2) are identical, indicating that no "unintentional correction" was found in testing models trained with the MNIST dataset.

These observations suggest that tree depth influences the model's robustness characteristics, and that robustness is not solely aligned with baseline classification accuracy.

Hypothesis 2: Does Minimum Sample Split Affect the Resistance? Minimum sample split is a parameter for controlling the minimum number of samples required to split an internal node [83]. We trained several models with the three datasets. Here we focused on four models (fM1S2, fM2S5, fM3S10, fM4S20) that were trained with the fashion-MNIST dataset, with the parameter for "minimum sample split" set as 2, 5, 10, and 20, respectively. The accuracies of the four models are (in Fashion-MNIST):

$$Acc(\text{fM1S2}) = 0.693 > Acc(\text{fM2S5}) = 0.674 >$$
$$Acc(\text{fM3S10}) = 0.659 > Acc(\text{fM4S20}) = 0.642$$

Figure 5.9(a) provides evidence to support the hypothesis that changing this parameter can affect the models' ability against adversarial attacks. The three statistical indicators show patterns that are consistent with the order of their accuracy values. As shown in Figure 5.9(a_3), while attacks on models fM1S2, fM2S5, and fM4S20 do not result in much unintended perturbation, fM3S10 exhibits a rather different pattern at the later steps of the iteration. One possible explanation is that these "positive attacks" may happen in a certain area of the testing image, where fM3S10 tends to make errors.

Hypothesis 3: Does Maximum Features Affect the Resistance? The maximum number of features considered by a training process is known to impact the quality of decision tree models [83]. We also trained several models with

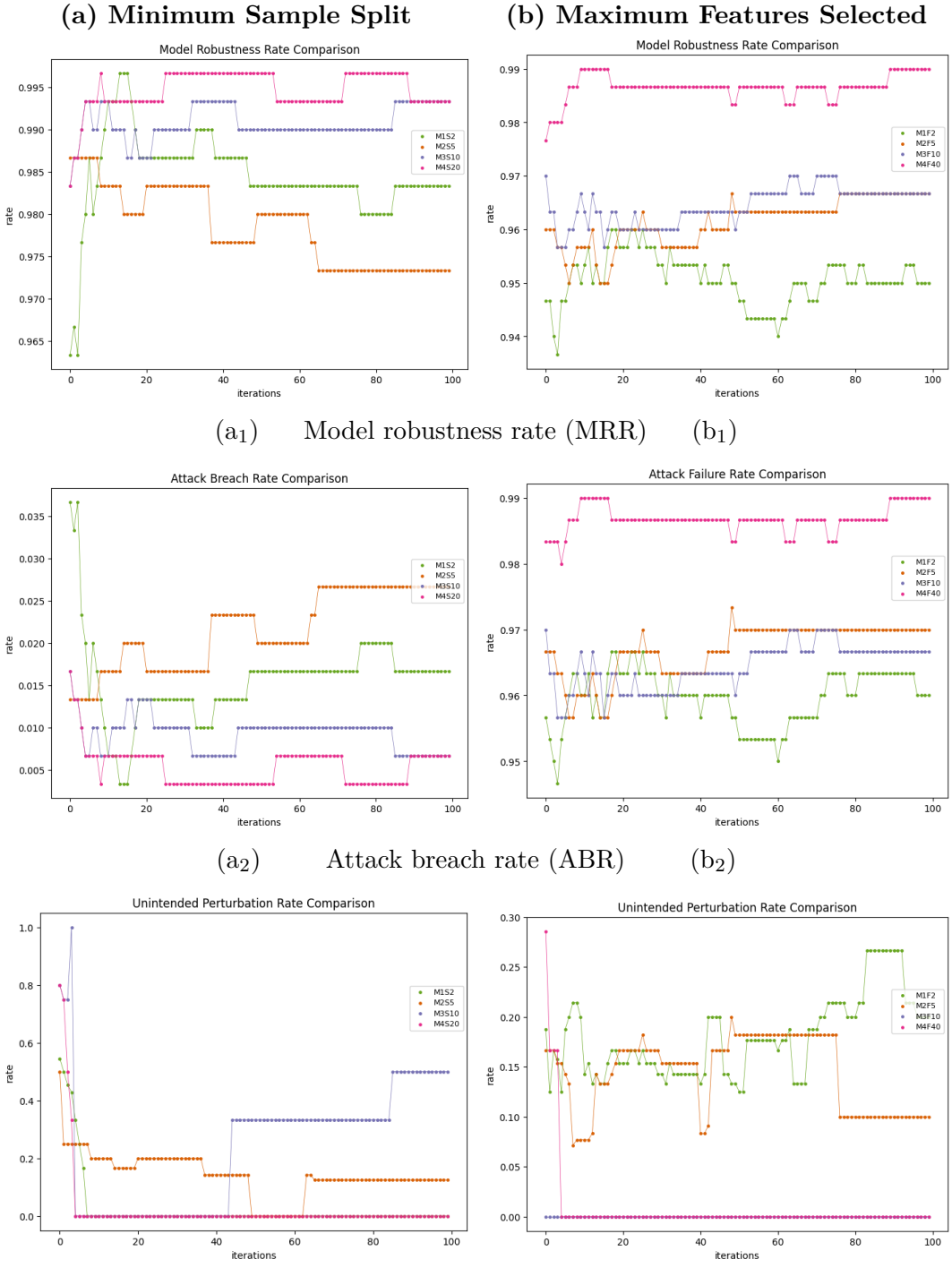


Figure 5.9: Testing hypotheses 2 and 3: (a) about minimum sample split, and (b) about maximum features selected. All models are trained with the Fashion-MNIST dataset. In (a), four models trained with the parameter of minimum sample split are set to 2, 5, 10, and 20. In (b), four models trained with the parameter of maximum features selected are set to 2, 5, 10, and 40.

the three datasets. Here we focused on four models (fM1F2, fM2F5, fM3F10, fM4F40) that were trained with the fashion-MNIST dataset, with the parameter for "maximum features selected" set as 2, 5, 10, and 40, respectively. The accuracies of the four models are (in fashion-MNIST):

$$Acc(\text{fM1F2}) = 0.539 < Acc(\text{fM2F5}) = 0.604$$
$$< Acc(\text{fM3F10}) = 0.625 < Acc(\text{fM4F40}) = 0.693$$

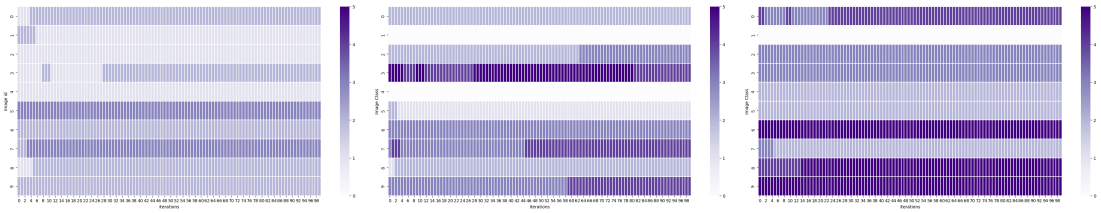
Figure 5.9(b) provides evidence to support the hypothesis that changing this parameter can affect the models' ability against adversarial attacks. However, the three statistical indicators show patterns that are inconsistent with the order of their accuracy values. The **model robustness rate** (b_1) indicates $MRR(\text{fM1F2}) < MRR(\text{fM2F5}) < MRR(\text{fM3F10}) < MRR(\text{fM4F40})$. In other words, this is consistent with the accuracy ordering. Meanwhile, the **attack failure rate** (b_2) shows that the positions of $AFR(\text{fM2F5})$ and $AFR(\text{fM3F10})$ are swapped. On the other hand, the statistical measure of **unintended perturbation rate** (b_3) indicates that a good amount of attacks on fM1F2 and fM2F5 had "positive" outcomes.

Hypothesis 4: Do Image Classes Have Different Vulnerability? Given a classification model, we can compare the vulnerability of different classes. Figure 5.10 shows the results of testing three models trained with the Cifar-10, MNIST, and Fashion-MNIST datasets respectively, The 10 classes of Cifar-10 are 0: airplane, 1: automobile, 2: bird, 3: cat, 4: deer, 5: dog, 6: frog, 7: horse, 8: ship, 9: truck. The 10 classes of MNIST are ten digits, i.e., 1, 2, 3, 4, 5, 6, 7, 8, 9, 0. The 10 classes of fashion-MNIST are 0: T-shirt, 1: trouser, 2: pullover, 3: dress, 4: coat, 5: sandal, 6: shirt, 7: sneaker, 8: bag, 9: ankle boot. In each plot in Figure 5.10, the k^{th} row shows the number of successful attacks to the images in the k^{th} class when the testing progresses iteratively.

For each class, 100 images were randomly selected to be attacked. The number of successful attacks is color-coded, i.e., the darker the color is, the more attacks were successful and therefore the class concerned is potentially more vulnerable.

Figure 5.10 shows the results of testing each class in terms of the number of images (out of 100) that were breached in each iteration during the testing.

From Figure 5.10, we can observe that color variation is more noticeable in (a_2) and (a_3), indicating that there is evidence for supporting the hypothesis.



(a_1) Cifar-10 (a_2) MNIST (a_3) Fashion-MNIST

Figure 5.10: Testing hypothesis 4 about image classes. In each plot, a row corresponds to a class, and the sequence of color-coded shapes in the row represents the varying number of successful attacks during the multi-iteration testing process.

5.5 Discussions and Conclusion

Discussions. We recognize that testing against adversarial attacks is not an easy undertaking. This work shows that there is a non-trivial amount and a variety of HITL activities needed for supporting the testing against One Pixel attacks on decision tree models. This justifies the need for a purposely built user interface.

While some HITL activities are common to different ML models being tested or different adversarial attack algorithms, many will be specific to individual ML techniques and attack algorithms. The strategy for developing a testing software system thus requires further research to answer questions such as whether one system should accommodate different ML techniques and attack algorithms. Based on our experience in designing TA3, we suggest that for white-box testing, the best strategy might be to design multiple systems, one for each family of ML techniques that share the same form of internal structures. Meanwhile, such a system should include as many attack algorithms as possible that could be applied to the family of ML techniques. Meanwhile, for black-box testing, one may have a more generic user interface that includes many attack algorithms.

Technically, much more research and development effort will be needed to develop interactive software systems for different families of ML models, some of which will be challenging to visualize and interact with. There will also be a need to design and develop an API to improve the efficiency of developing such interactive software systems.

Conclusions. In this Chapter, we present TA3, a novel HITL-support tool for testing ML models in terms of their ability against adversarial attacks. As illustrated in Figure 3.1, the testing workflow (Figure 3.1(b)) is much more complex than the one for testing an adversarial attack algorithm (Figure 3.1(a)). This confirms the necessity of providing more efficient and effective HITL support. The development of TA3 confirms the feasibility of this approach. The design of the TA3 enables the close integration of algorithms, interaction, visualization, and statistics.

By using visualization in adversarial attacks with testing of ML models, testers can have a new view of the quality assurance for the model. A better test of an ML model also means an improvement in the model development, which is of great significance in practical applications. We also give some metrics to measure a model's quality. It can be used as a basis for evaluating the performance of different models against adversarial attacks. In addition to testing the model's robustness to adversarial attacks, TA3 helps users to have an understanding of the model's quality from multiple aspects. The future of TA3 can perform a visual analysis for more kinds of ML models and more adversarial methods. It will help people achieve more efficient testing of ML models and assist in much wider use.

From application functions, TA3 shows a false-negative classification result directly, but it is not easy for users to search for a false-positive result in the model classification. Users need to find the false positive data by checking the true label and the classification result one by one. The TA3 only conducts a visualization study of the type of decision tree in image classification. Some ML tasks (such as stock forecasts, weather forecasts, etc.) require continuous input data for analysis, and it is difficult to implement when using TA3 in the testing process.

Future Work. We plan to research the HITL support for testing other families of ML models, such as neural networks, Bayesian networks, and hidden Markov models, which feature different internal structures and hence need different support for interaction and visualization.

We hope to visualize and analyze more kinds of ML models (CNN, LSTM, Transformer, GAN) in the future to improve the quality of these models against adversarial attacks. As a kind of ML model, the decision tree has the characteristics of easy analysis. In the visualization of a neural network model, the trajectory of the image in the model will be more complicated compared with the decision tree, where one image has only one unique decision path. In a neural network, each part of the image may pass through multiple nodes in the network, and some nodes will be passed through repeatedly. Testing on neural networks will be the focus of our future research. Through visual analysis, the transparency of the neural network model is improved to better help the model resist adversarial attacks or natural noise in the environment. We hope that testers of the ML model can use TA3 to solve other types of testing problems and improve the quality of the application of the ML model in reality.

Using the concept discussed in Chapter 2, TA3 addresses the inconsistency between the application-specific data space and the artificially constructed adversarial data space. By adopting the Four Views Design and tailoring it to adversarial testing, TA3 enables human-in-the-loop attack simulation and visualization-assisted impact evaluation. This design allows ML model testers to explore different parameter settings, visualize model decision paths, and detect discrepancies that quantitative metrics alone may conceal. Experiments on decision tree models trained with Cifar-10, MNIST, and Fashion-MNIST datasets show that such visualization-supported, data-centered testing provides deeper insights into model vulnerabilities and leads to robustness improvements beyond traditional experiment-based optimization.

6

DS4ML: Data Synthesis for Machine Learning Models Training

As machine learning (ML) models are widely applied to time series data classification tasks, the accuracy and reliability of these models have become important. Existing models often encounter discrepancies between laboratory training data and real-world deployment data during the training process, leading to poor performance in testing. Using synthetic data can address this issue to some extent; however, effectively analyzing and comparing synthetic data has long been a challenging problem. This chapter introduces DS4ML (Data Synthesis for Machine Learning), an innovative visualization support system that aims to support data synthesis and improve the model accuracy in the deployment scenarios through interactive visualization techniques. We conducted an expert evaluation of DS4ML, applying it to public datasets. The experimental results demonstrate that it can improve model performance in real-world deployment conditions. Additionally, through evaluations with ML and visualization experts, we confirmed that the system has positive implications for ML model developers in the data synthesis process.

From the perspective of Chapter 2, DS4ML addresses a data space inconsistency between lab-collected training data and real-world deployment data in time-series classification. Traditional experiment-based optimization methods are often ineffec-

tive because they do not alter the underlying training data space and thus fail to capture the state change patterns observed in deployment. DS4ML applies the Four Views Design introduced in Chapter 3 as a practical solution: by embedding data synthesis and visual analysis directly into the workflow, it enables ML developers to iteratively repair the development training data space, bringing it closer to the deployment data space and thereby improving model performance.

6.1 Introduction

The discrepancy between lab-collected training data and deployment data often troubles ML users. The reasons for this situation are complex [349]. Training data is typically collected in controlled laboratory environments, while actual data comes from unpredictable real-world scenarios. These environmental differences can lead to variations in data distribution, noise levels, and feature representation, thereby affecting the performance and reliability of ML models in practical applications. Although ideally, these differences would only have a minor impact on the model's actual performance [350], in reality, there is a significant cost difference between obtaining actual data and training data. Considering the use of limited training data, simulating actual data as closely as possible becomes a cost-effective option worth considering [351].

In time series classification tasks, how to effectively use experimental data to simulate actual data often troubles ML model developers. To address these challenges, we propose DS4ML (Data Synthesis for ML), an innovative ML model training data generation and analysis application. The main idea of DS4ML is to provide a comprehensive and intuitive platform for ML model training through visualizations, helping ML model developers effectively analyze the data synthesis process. DS4ML helps users simulate various possible data distributions by generating synthetic datasets, thus producing more targeted synthetic training data. Its interactive design allows users to quickly iterate and optimize models, thereby improving the efficiency of model analysis.

The main contributions of this chapter include the following three parts:

- Developed a data synthesis and visualization platform, DS4ML, providing ML researchers and users with a more intuitive and interactive model analysis platform.
- Experimentally validated the effectiveness of synthetic data with the aid of visual analysis in improving model performance.
- Helps users more accurately identify and solve problems in their models, especially when dealing with differences between training and testing data.

DS4ML provides a powerful tool for ML researchers and practitioners, which is expected to promote the development and application of high-quality and reliable ML models and improve the efficiency of ML models in processing big data.

6.2 Background

Statistics Netherlands collects data on Dutch citizens' exercise habits to understand their physical activity patterns. They incorporate objective indicators of sports and exercise into health and lifestyle surveys. This allows them to provide targeted policy recommendations to the government for future infrastructure development and other areas. The data collection is primarily divided into two parts: a lab-collected training part and a real-world collected deployment part.

In a laboratory environment, 40 participants wore sensors (ActivPAL on the thigh) to perform 8 specific activities (sitting, standing, stepping at two intensities, cycling at two intensities, climbing stairs, and jumping). Characteristic signals for each specific activity were recorded to help train ML models for real-world applications and identification. The data labeling was determined by the prescribed physical activities.

For the real-world component, 40 participants wore sensors for one week. They also kept diaries to record their physical activities, work, and travel times. ActivPAL is a physical activity monitoring device developed by PAL Technologies [352]. It is a small, lightweight wearable device typically attached to the front of the thigh.

ActivPAL is primarily used to accurately measure and record an individual's body posture and activity patterns. Researchers and healthcare professionals often use ActivPAL to evaluate the effectiveness of interventions, monitor patients' rehabilitation progress, or study the impact of lifestyle changes on health. A major advantage of this device is its ability to continuously record data for up to 7 days, and its waterproof design allows wearers to bathe and swim normally.

The data collected from the IMU (Inertial Measurement Unit) in ActivPAL and similar wearable devices is known as Human Activity Recognition (HAR) data. HAR data typically includes detailed information about human movements, postures, and behaviors, which can be used to identify, monitor, and analyze individual activity patterns and habits. HAR data has various applications across different fields, including health monitoring [353], sports analysis [354], and others. HAR data collection is usually achieved through various sensors. These sensors capture subtle changes in human movement and convert them into digital signals, thus providing an objective record of human activity. With the proliferation of portable smart devices, HAR data collection has become more convenient and real-time [355]. Common challenges in processing HAR data include data noise, inconsistent sampling rates, and obtaining data labels [356]. To overcome these challenges, researchers often use ML [357], deep learning [358], and other methods to process and analyze HAR data. These methods help extract useful features and perform activity analysis.

Statistics Netherlands previously focused on researching the method of using unsupervised ML techniques to analyze and segment accelerometer data collected from daily activities. They used accelerometer data to classify physical activity intensity without relying on predefined thresholds or calibration studies commonly used in traditional methods [359]. However, when the ML models trained on the lab-collected training data were tested on real-world deployment data, they were found to have low matching rates with daily activity records. After relevant research and analysis [199], a key reason was identified: there are differences in state change patterns between lab-collected training data and real-world deployment data.

Further inspection revealed that the state change patterns observed in deployment differed notably from those encoded in the training data. Certain state changes, like "jumping-bicycling", occurred frequently in the training data but were underrepresented in the deployment data, while other state changes common in deployment are even absent in the training data. These differences in data state change structures provide evidence of data space inconsistencies between the training and deployment phases. Such inconsistencies are not always obvious when inspecting raw time-series signals with line charts, since subtle differences may be obscured. Visualizations for the data state changes can reveal these discrepancies more clearly. In the Chapter 7, we present the Radial Icicle Tree (RIT), a dedicated visualization design that makes these state change differences explicit. The new visualization support methods help ML model users to obtain concrete evidence of the data space inconsistency between training and deployment at the core of this chapter.

These observations confirm that the lab-collected training data cannot faithfully represent the deployment data space, leading to the inconsistency that degrades the model's performance in the deployment environment. Ideally, one would resolve this issue by collecting large-scale deployment data. However, in practice, this is often infeasible due to cost, participant burden, and labeling inaccuracies. Because the cost of collecting real-world deployment data by having volunteers wear devices for long periods is high, and volunteers' records of their own physical activity are sometimes inaccurate. To address this issue, after discussions with Statistics Netherlands, we decided to enhance the performance of ML models by using synthetic data methods based on the lab-collected training data. This leads to the next three questions:

- **Q1**: How to generate synthetic data?
- **Q2**: How do we ensure that the synthetic data meets our expectations?
- **Q3**: How can ML users compare and select the synthetic data they want?

By answering these three questions, we hope to help ML model developers better improve their models' performance in situations where the training data differs from real-world deployment data, yet still train a model that performs well.

6.3 DS4ML: Algorithm, Interaction, and System

Following Section 6.2, we need to elaborate on two important concepts: **lab training data** and **real-world deployment data**. Typically, in most ML tasks, these two types of data are considered to be from the same source. However, in this Chapter, we recognize that training data collected in the laboratory is not similar to deployment data collected in real-world environments. The differences between these two types of data may stem from various factors. We aim to make the lab training data as close as possible to real-world deployment data by employing generative data algorithms.

Another important concept is **state change**. In the context of human physical activity, state change is defined as the transition between different types of activities, such as from running to climbing stairs to walking. We use the term "**rank**" to denote the number of these state changes. Since there are transitions between three states (running-climbing stairs-walking), we use "rank-3" to represent this. For "rank-2", it looks like two states (running-walking) or some transitions similar.

6.3.1 Synthesis Algorithm

Due to findings in previous research, the difference between lab-collected training data and real-world testing data lies in the different patterns of state changes. In the data synthesis section, we used two algorithms to augment the training data, aiming to make the data collected in the laboratory as close as possible to real-world deployment data.

The first method for synthesizing data is called the Brutal Force Algorithm (BFA). The main idea is to directly use the patterns of motion state changes observed in real-world scenarios as input. From the existing lab-collected training data, we select data with the same motion states to compose new training data, in order to generate data that fits the patterns of motion state data in real-world situations. This algorithm is implemented by traversing all possible combinations of state changes, comparing them with the target combination, and outputting the result if they are the same.

Algorithm 1: Brutal Force Algorithm (BFA)

Require: *ele_1*: element numbers, *ele_2*: state changes
Ensure: Training data synthesized from real-world states
Initialize training set $T = \emptyset$
for each possible combination C of *ele_1* **do**
  Generate motion sequence S from C
  if S matches *ele_2* exactly **then**
    Append S to T
  end if
end for
if $T \neq \emptyset$ **then**
  return T as training data
else
  return No matching combination found
end if

The pseudocode of the brutal force algorithm is shown in Algorithm 1. The advantage of this method is that it ensures the generated data are exactly consistent with the motion state patterns observed in reality, thus theoretically preserving the authenticity of the data to the greatest extent. However, since it requires traversing all possible combinations, the computational complexity is high. This becomes especially problematic when the number of states increases, leading to excessive computational costs and making the method unsuitable for large-scale data synthesis tasks. Therefore, this algorithm is more appropriate as a baseline method to verify the effectiveness and rationality of subsequent improved algorithms.

The second algorithm, which we named the Pattern Similarity Algorithm (PSA). Compared to the brutal force algorithm, it considers flexibility and allows for some variation. The pseudocode implementation of the algorithm can be found in Algorithm 2.

In the algorithm, we take rank-2 as an example. In practical applications, situations higher than rank-2 can be achieved by recursively performing the corresponding parts of the algorithm. The main idea is to first input two certain lists, ele_1 and ele_2, where ele_1 represents the number of various labels collected in the entire experimental data, and ele_2 represents the state changes in real-world deployment data. The input is the rank-2 state change pattern, which represents the

Algorithm 2: Pattern Similarity Algorithm (PSA)

Require: *ele*_1: element numbers, *ele*_2: state changes
Ensure: Adjusted laboratory data for training
Initialize chains list $C = \emptyset$
while *ele*_1 is not empty **do**
  Randomly select E_i from *ele*_1
  Randomly select E_j from *ele*_1 where $E_j \neq E_i$ {Avoid repeated states}
  Create chain $R = \{E_i, E_j\}$
  Append R to C
  Remove E_i, E_j from *ele*_1
end while
Connect chains in C to form motion sequence S
Count pattern occurrences in S
Compute dissimilarity d between S and *ele*_2
if $d \leq$ threshold **then**
  return S as training data
else
  return Retry with different random selections
end if

case of one state change in real-world deployment data. After these two input lists, we randomly select an element from ele_1, then select another element without replacement to form a chain R. This is done to facilitate comparison with the number of state changes in ele_2. It's important to note that the second element selected cannot be the same as the previous one to avoid situations where there appears to be a repeated state when, in fact, the state hasn't changed. We repeat this process multiple times until all elements in ele_1 have been selected. We connect these chains to form a sequence of motion state changes over a period of time, count their occurrences, and compare them with ele_2. If the difference is within an acceptable dissimilarity range, we output the adjusted lab-collected training data as training data.

The main difference between PSA and the brutal force algorithm lies in the fact that the brutal force algorithm may not be able to completely mimic the exact real-world collected training data, and might have to fill in by repeatedly using the same type of movement. In contrast, the PSA algorithm, due to its dissimilar tolerance setting, can avoid the issue of repeatedly using the same type of movement

data, thereby preventing the loss of data balance.

Through the demonstration of the Synthesis Algorithm section, we have answered question **Q1**. It should be noted that various approaches can be used for data synthesis. We only present two algorithms that are designed for human physical activity data as illustrative examples. The motivation of this chapter is not to propose better synthesis algorithms, but rather to demonstrate how the DS4ML system, under the Four Views Design concept, can assist ML researchers in analyzing models and data through the use of synthesis algorithms. We will use the aforementioned methods to perform data synthesis and collect experimental results.

6.3.2 Dataset Details

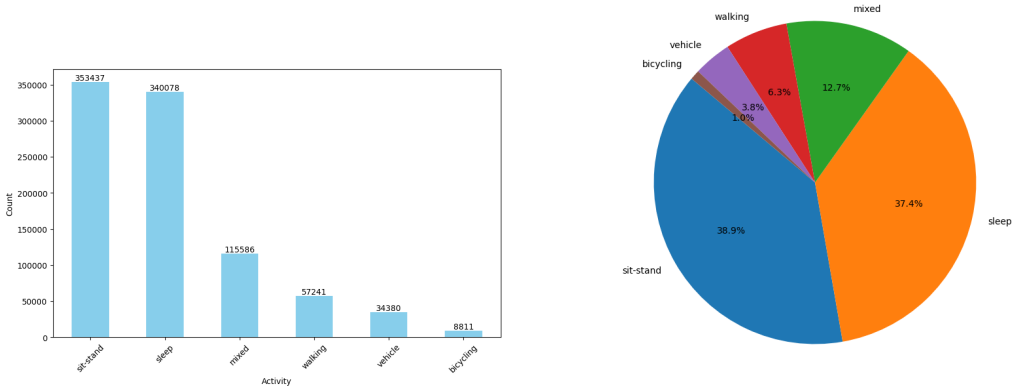


Figure 6.1: Oxford Capture-24 dataset details. We notice that the distribution of the label is not average; some activities were much more than others.

Previous research [360] indicates that decision tree models or random forest models trained under laboratory conditions do not perform well when directly applied to real-world training data. The reason for this phenomenon is that physical activities in real-world conditions exhibit more randomness. It is difficult for lab-collected training data to simulate this randomness accurately without knowing specific movement habits.

In order to protect the privacy of relevant users from the Dutch government, the dataset we used is the open-source Oxford Capture-24 data [334]. This dataset includes data from Activity AX3 wrist-worn activity trackers, collected from 151

participants in the Oxfordshire region between 2014 and 2016. Participants were required to wear the device for approximately 24 hours in their daily lives, totaling nearly 4,000 hours of data collected. The Vicon Autograph wearable camera and Whitehall II sleep diary were used to capture real-life activities (such as sitting and watching TV, washing dishes, and sleeping) during this period, resulting in over 2,500 hours of labeled data [360].

The data details consist mainly of six different activity types: sleep, sit-stand, mixed, vehicle, walking, and bicycling, totaling 909,533 preprocessed data points. The data format consists of timestamps, xyz coordinates, and labels. Detailed information about the dataset and the related preprocessing code can be downloaded from https://github.com/OxWearables/capture24.

By analyzing the dataset, we can obtain Figure 6.1. From the figure, we can intuitively observe that the characteristics of the Oxford Capture-24 dataset, which is the distribution of different activity types, are highly uneven. Based on previous research, uneven data distribution of the training data is a major reason for the poor performance of ML models [361].

6.3.3 Experiments and Results

Our experiment is mainly divided into two parts. In Experiment-I, we set a distribution sequence found in real data as the input of the PSA algorithm (ele_2). Then we use the Oxford Capture data as the data collected in the laboratory, extract the number of corresponding labels, and use it as the second input of the PSA algorithm (ele_1). The training set, test set, and validation set are divided according to the 60:20:20 principle [362]. Synthetic data is added to the training set, and the test data is taken out separately at the beginning of the split. After adjusting according to the corresponding distribution sequence in reality, no changes are made.

In the experiment, we used a decision tree and a random forest model. The decision tree model was constructed using the Python-based sklearn library [83]. The parameters used for the model were: “max depth" set to 12, “random state" set to 42, and “min samples split" set to 2, with the remaining parameters set to

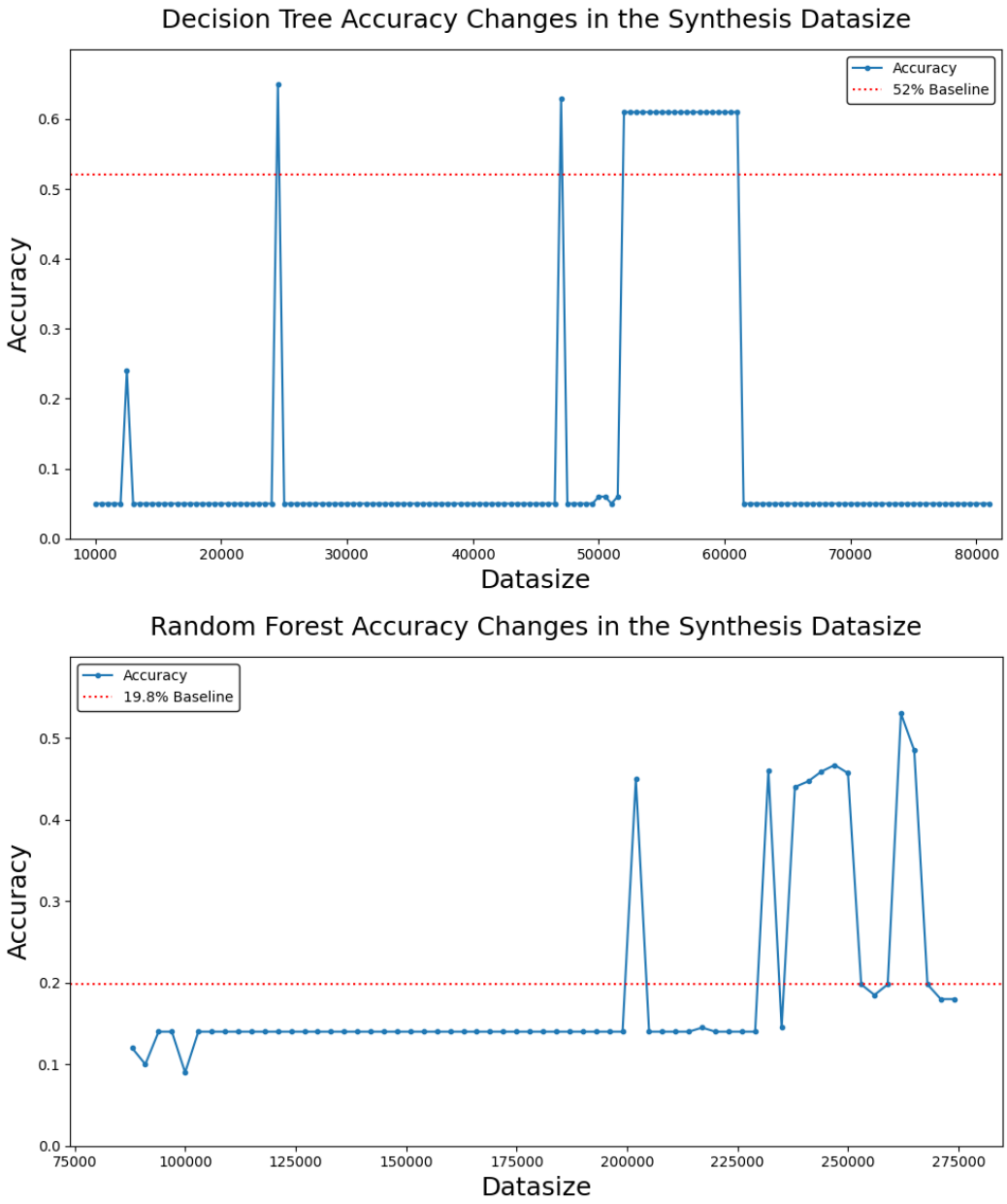


Figure 6.2: Experiment-I results of the ML model performances on the testing data after retraining on the synthetic data. (a) Decision Tree Model's performance on the top and (b) Random Forest Model's Performance on the bottom. The x-axis represents the amount of data generated.

default. The random forest model was also constructed using functions provided by sklearn, with parameters set to "n estimators" at 100, "max depth" at 12, and "min samples split" at 2, with other parameters set to default. The number of synthetic data generated by PSA gradually increased, aiming to reach the planned data size. The experiment was conducted three times, and the average prediction

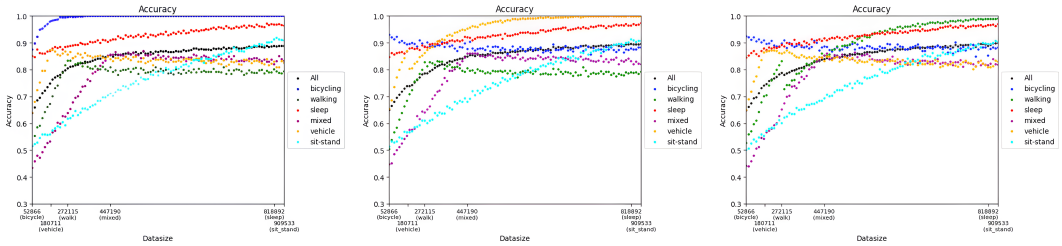


Figure 6.3: Results of the model performances on Oxford Capture-24 using synthetic data. From left to right: bicycling, vehicle, and walking. The dark blue dots represent bicycling, dark green dots represent walking, dark red dots represent sleeping, purple dots represent mixed, orange dots represent vehicle, light blue dots represent sit-stand, and the black dots represent overall performance. The numbers above the x-axis represent the amount of data in the corresponding category in the dataset. The amount of training data increases as the input training data increases. After the amount of training data exceeds the original amount, oversampling is performed by inputting the same time point.

accuracy of the models was taken.

The experimental results are shown in Figure 6.2. From the results, we can see that with the training data augmented by the PSA algorithm, the performance of both the decision tree and random forest models exceeds their performance without any augmentation (Baseline in red). This demonstrates the effectiveness of the PSA algorithm, showing that using synthetic data can improve the performance of ML models.

Experiment-II involves using only the oversampling method on the Oxford Capture-24 dataset for data augmentation. We tested the effectiveness of the synthesis algorithm on the ML model without using external data. The main purpose of doing this is to verify whether the increase in the amount of data leads to better performance than before. The experiment synthesized data from three categories in the Oxford Capture-24 dataset and gradually increased the amount of training data during model training until the amount of data reached the target value and matched the amount of data in other categories. We performed data augmentation on three categories (bicycling, vehicle, and walking) and tested the effect of data augmentation using the standard oversampling algorithm. In terms of the amount of synthetic data, the amount of synthetic data in the experiment increases proportionally with the amount of training data. After all the input data are entered, the amount of synthetic data is controlled by entering the same time point.

The ML model used in Experiment-II is a decision tree model, with the same parameters as in the first part of the experiment. Using the sklearn library, the model was generated with “max depth" set to 12, “random state" set to 42, and “min samples split" set to 2, with the remaining parameters set to default. General oversampling methods were applied to the synthetic data, where the data was randomly selected and increased to gradually reach the planned size of the target data. The experiment was conducted three times, and the average testing accuracy of the models was taken.

The experimental results are shown in Figure 6.3. By comparing the experimental results of three different categories in Figure 6.3, we can observe that the common phenomenon is that the use of non-specified synthesis algorithms can improve the performance of the model to a certain extent, but the extent of the improvement does not increase completely with the increase in the amount of data. For the synthetic target data, the recognition accuracy of this category is improved with the increase in the amount of data, but the improvement effect is not obvious for the class that does not use synthetic data, and may even decrease with the increase in the amount of synthetic data. It can be considered that the model is overfitting. The overall accuracy began to decline after a certain increase, which may also be due to overfitting. The experimental results show that the use of oversampling algorithms(random sampling) only partially improves the accuracy of the model for specific categories within a certain range of small amounts of generated data, but the improvement of the overall model is limited. And if only the amount of generated data is increased, the actual effect may even lead to a decrease in the prediction accuracy of the model.

In addition, by comparing the experimental results of three different categories in Figure 6.3, we can also observe that data with a larger sample size is less affected by synthetic data, which may be due to the sparsity of data distribution. When the amount of synthetic data exceeds the total amount of sample data, the prediction accuracy of the model decreases. This may be because the synthetic data fails to fully cover the distribution of real data in the high-dimensional feature space,

thereby introducing noise samples and interfering with the learning process of the model. When synthetic data occupies a dominant proportion of the training set, the model is more likely to learn pseudo-features or offset patterns in the synthetic data, resulting in a decrease in the generalization ability of the real distribution. Therefore, when using synthetic data for training, it is necessary to find a balance between the amount of data and the distribution to avoid a decrease in model performance. These observations are consistent with our expectation that moderate data augmentation improves class-specific recognition by reducing sparsity, whereas excessive randomly generated synthetic data introduces distributional distortion and overfitting.

The experimental results show an answer to **Q2**. Through experimentation and visualization methods, we can ensure that the synthesized data meets our expectations and is similar to real-world deployment data. While general synthesis algorithms have limited improvement on the model, synthesis data assisted by visualization methods can effectively enhance model performance. We should also note that better algorithms for augmenting training data using synthetic data certainly exist. The main purpose of the experiment is to address **Q2**, which is to verify the effectiveness of synthetic data and demonstrate that it can be used as a method to improve ML model performance. Additionally, this part of the experiment also serves as preparation for answering **Q3** later.

6.3.4 DS4ML Introduction

After obtaining the experimental results, we realized that having addressed **Q1** and **Q2**, there is a need to enable users to conveniently compare and select synthetic data. This leads us to the answer for **Q3**.

To facilitate convenient data interaction for ML researchers, we developed the DS4ML system. DS4ML stands for Data Synthesis for Machine Learning, an application for synthesizing training data and testing ML models. DS4ML helps ML model users synthesize the necessary training data to improve their model performance across different datasets. DS4ML provides a user-friendly interface, allowing users to load their datasets and select the data they need for retraining.

Through data visualization and visual analysis of model performance results, users can understand the quality of their models and then synthesize suitable training data to address identified issues effectively.

DS4ML currently supports decision tree-based models as tested models. Once users select their dataset and ML model, DS4ML can generate various visualizations and synthetic training datasets based on the input data, helping users evaluate model accuracy and identify potential issues, such as overfitting, dataset bias, or other data quality problems. Users interact with and control the synthetic data through DS4ML. DS4ML uses a three-level architecture to build the application. These three levels are designed to separate different parts of the application for better maintainability and scalability. The three-level architecture is as follows, from top to bottom:

- **Sample Level**: This level is responsible for interacting with users, controlling the presentation of the interface, and responding to user operations.
- **Class Level**: We use the “View" as a module responsible for implementing a specific functionality. This level is responsible for dispatching user requests to different modules for processing and returning the results to the controller/user interface.
- **Model Level**: This level is responsible for reading, storing, synthesizing, and visualizing data and models.

The three-layer structure is derived from the progression of data to categories to models [363], which also informs the DS4ML workflow design. Built on this three-level architecture, DS4ML focuses on analyzing model performance across different synthesized training datasets. Based on this foundation, we developed a user interface following the “Four Views Design” concept [364], as illustrated in Figure 6.4. The DS4ML interface consists of four interconnected views that support different tasks in ML data synthesis and testing, presented in sequential order: (a) Synthesis View, (b) Dataset View, (c) Model View, and (d) Result View.

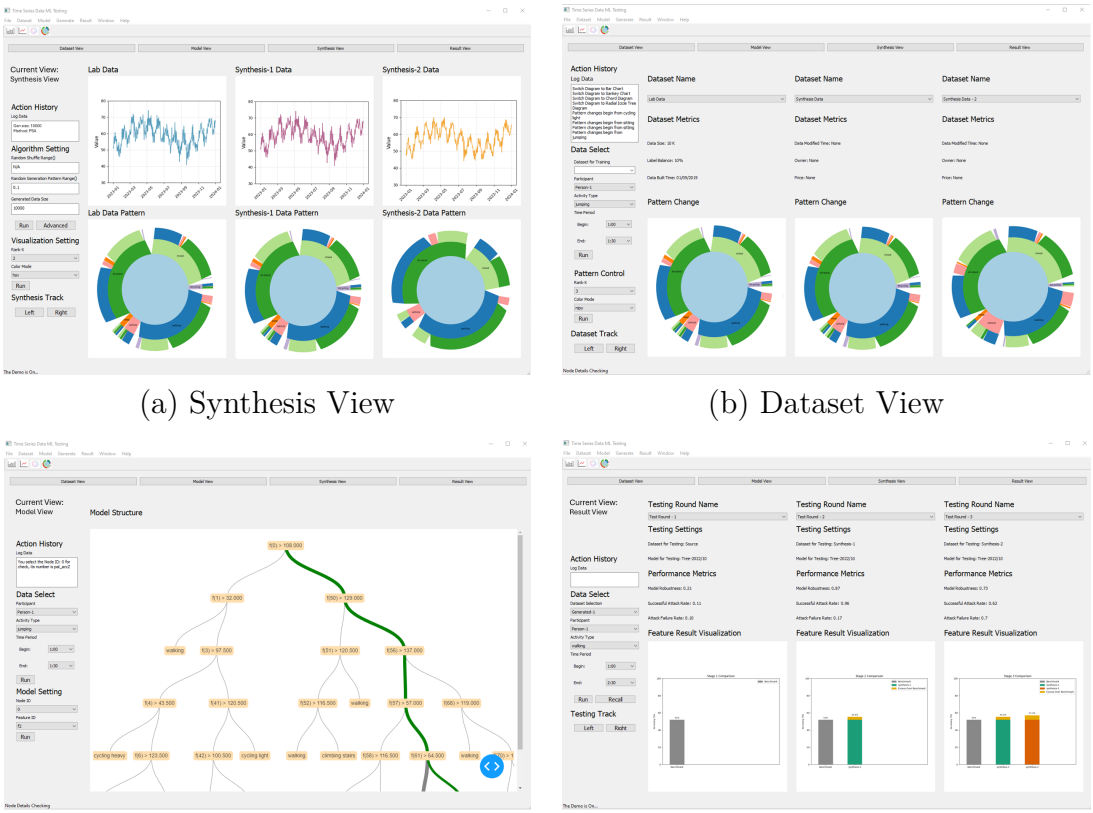
The Synthesis View can focus on displaying more data generation methods and parameter configurations, helping users flexibly explore different synthesis strategies and compare differences between different synthetic data as well as with original data. The Dataset View can expand more data management and distribution analysis functions, allowing users to compare different samples while observing the whole dataset's structure. The Model View supports more complex model structures on the original foundation, facilitating users' rapid experimentation. The Result View can accommodate more intuitive visualization displays and performance comparison analysis.

Visualization techniques have been widely applied in the analysis of models and data, including some of the latest works [199] as well as established methods such as chord diagrams, heatmaps, treemaps, and line charts. The incorporation of these visualization tools enables ML model developers to intuitively analyze model structures, results, and the inherent characteristics of datasets. This facilitates the improvement of model performance through a human-in-the-loop [365] approach.

It is important to notice that while ML developers are most likely to start from dataset view, selecting given training and testing data without any synthesis steps, after the first iteration involving (b) $\rightarrow$ (c) $\rightarrow$ (d), they will primarily follow the sequence (a) $\rightarrow$ (b) $\rightarrow$ (c) $\rightarrow$ (d) in subsequent iterations. This process, typically project-dependent, can last from days to weeks.

In specific views, ML developers observe the input data in the Dataset View and inspect the raw data. They organize the properly synthesized data. After data selection and combination, they proceed to the Model View to train the model. Subsequently, they enter the Result View to check the model quality and observe model performance. Based on the analysis of the model results, ML developers can choose to enter the Synthesis View to further augment the training data, thereby achieving further model optimization.

Human-computer interaction(HCI) can provide effective and efficient user control for such iterative loops, allowing control over training data generation and model testing processes. They will also benefit from combining the synthetic data with the



(a) Synthesis View (b) Dataset View

(c) Model View (d) Result View

Figure 6.4: We use the Four Views Design concept to design a UI for a certain stage of the testing and retraining of an ML model.

source data to better retrain the ML model. The goal of the "Four Views Design" software layout is that users can conveniently switch between the four views via different looping paths, thereby enhancing model testing efficiency. At any stage corresponding to the four views, if ML developers identify issues during testing, they can quickly switch to the target view for inspection and solution finding. DS4ML currently supports ML model testing on various time-series datasets. Compared to traditional model testing processes that require repeatedly running models, collecting, and analyzing data, the DS4ML process allows data exchange and human-centered control over different aspects of testing, thus avoiding time-consuming model runs and improving testing efficiency.

In Figure 6.4, we present a specific example to demonstrate how DS4ML, designed with the Four Views Design idea, specifically helps ML users and researchers analyze models.

In general scenarios, users are not satisfied with the model's current results after initial training and hope to enhance the model's performance through targeted reinforcement of training data. The user plans to strengthen data for all categories within a certain time period, so the user uses different synthesis methods in Figure 6.4 (a) to perform synthesis operations on the corresponding time series data. The leftmost column of results shows the original data itself (represented by line charts) and the state change diagram of the original data, the middle column shows data synthesized using the brutal force algorithm and the state change diagram of the synthesized data, and the rightmost column uses the previously mentioned PSA algorithm for state changes to perform synthesis and displays the synthesized state change diagram. For state changes in the data, we used a visualization method called Radial Icicle Tree (RIT) [199] to perform visual analysis of the synthesized time series data. The specific drawing strategy of RIT will be detailed in Chapter 7. In traditional visualization methods, such as line charts, comparing the similarity between two time series can often only be done by overlapping two line charts for comparison, while using RIT, differences in time series data can be discovered under side-by-side conditions.

In the figure, we can actually observe that the time series generated using the PSA algorithm is basically consistent with the brutal force algorithm and original data on the line chart, but actually has some differences in state changes. Some corresponding states, such as vehicle (the pink area in the figure), have increased in proportion, mainly due to the algorithm's certain tolerance for different state changes. Readers can focus on the differences in state-transition structures visualized by the RIT diagrams and the corresponding performance changes shown in the Result View. The key aspect to observe is not merely the visual similarity, but whether the synthesized state-change patterns closely align with deployment environments and whether this alignment leads to measurable performance improvement.

Through intuitive comparison of the visualized differences between the original data and the other two synthesis algorithms, after users select the needed synthetic data, they can move to Figure 6.4 (b) to compare the training data after adding

synthetic data with the original data and observe whether the original state changes have been affected. From the figure, we can see that the changes are very minimal, possibly because the amount of synthetic data itself is only 10,000, which is quite small compared to the data scale of the corresponding time period. In fact, even if the changes are very small, adding certain perturbations is helpful for model training [366]. In other words, the data synthesized using PSA has been shown in previous experiments to have positive significance for model performance results within a certain perturbation range.

Subsequently, the user proceeds with retraining and visualization of the model in Figure 6.4 (c). In Figure 6.4 (d), a specific comparison is made of the model's performance before and after adding synthetic data. From the bar chart, we can observe that the leftmost result is the first test result, which represents the training result without adding synthetic data. This can typically be considered as the benchmark for the current task. The middle result shows the model performance using the brutal force algorithm, which has certain improvements compared to the previous benchmark. The rightmost result shows the outcome after retraining with synthetic data generated using the PSA algorithm added to the training data. We can see that the model's performance has improved compared to the benchmark, and the improvement is greater than that achieved with the brutal force algorithm. Above the visualization results is a display of relevant data for each test result after retraining with synthetic data, such as test settings and some test parameter results, facilitating users' intuitive observation of differences between different test rounds.

The above demonstrates a typical testing workflow using DS4ML designed with the Four Views Design. Users synthesize several potential training data samples from the synthesis view, select appropriate ones, and then enter the dataset view to integrate with original training data or already enhanced training data, retrain in the model view, and finally analyze results in the result view. After completing one round of model testing, users can choose to retain good results and record corresponding parameter settings for future test comparisons, discard poor results to save space, or retain them for analysis. This process can be repeated

multiple times, and users can move between different views at any time for further analysis. In this process, users' personal knowledge also plays an important role. For instance, specifically regarding human physical activity data, some state changes can be judged as impossible in reality based on human common sense, such as transitioning directly from vehicle to bicycling or from sleep directly to vehicle. These situations that could only occur in laboratory settings are impossible in reality. When generating data, humans need to identify and judge potential problems when using synthesis algorithms. Users will choose to discard based on experience to avoid introducing noise or misleading model training.

Furthermore, some professional users are often more sensitive to changes in specific categories or states, so in synthetic data and visualization results, they pay special attention to subtle differences in the proportions or distributions of these categories. For example, in traffic scenario analysis, users might be more concerned with proportion changes in the "vehicle" state, thus prioritizing synthetic data that can better enhance that particular state. Users might also actively adjust the magnitude and scale of perturbations based on their understanding of the model's generalization ability, balancing performance improvements from data augmentation with overfitting risks. These decisions do not rely solely on visual analysis results provided by the system, but combine users' comprehensive understanding of task objectives, data structure, and experimental strategies, reflecting the importance of human-machine collaboration within the DS4ML.

6.4 Expert Advice

To obtain qualitative feedback on usability and practical relevance, we conducted interviews with five domain experts. The interviews focused on system usability, interpretability, and perceived impact on workflow efficiency. We interviewed five experts from different fields, including machine learning, data visualization, human-computer interaction, and application deployment, to evaluate the practicality and usability of DS4ML. Their backgrounds and insights are summarized as follows.

- **HCI Expert:** A senior researcher with more than five years of extensive experience in designing interactive systems for data analysis. The expert noted that DS4ML's "Four Views Design" architecture provides an interaction flow for iterative model development, enhancing user productivity. They emphasized the value of integrating visual feedback loops in ML workflows.

- **ML Developer:** An AI engineer focused on time-series classification said that the DS4ML suggests the train-test distribution gap. DS4ML provides a low-friction interface for quickly testing how synthesized data impacts model accuracy in real-world deployment.

- **Data Synthesis Researcher:** A postdoc researcher specializing in synthetic data generation techniques appreciated DS4ML's flexible and interactive framework for controlling synthetic data generation. The expert focused on how DS4ML enables fine-grained adjustment of data variability and distribution, which supports more realistic and diverse training datasets.

- **Data Scientist:** An analyst who has been working in the data analytics field for more than five years emphasized the role DS4ML plays in reducing the cost of data collection by enabling visual validation of synthetic datasets. The tool improves communication and iteration speed between data analysts and project managers.

- **Visual Analytics Researcher:** An expert who has been working in the field of visualization for more than five years said that the use of the latest radial icicle trees and "rank" based visualization techniques to interpret the structure of time series data. He pointed out that the system is able to reveal overfitting or label imbalance problems through intuitive visual patterns.

These qualitative observations provide preliminary evidence of the system's practical utility; however, a formal user study with quantitative measures remains future work.

The feedback indicates that DS4ML may support ML developers in identifying data-related issues and iterating on model refinement. Several experts reported that the tool helped them discover model blind spots, improve data quality, and more effectively iterate on feature selection and model tuning. Notably, the combination of visual analysis with synthesis control was considered an important innovation that improves both transparency and performance in ML development workflows.

However, we must also admit that the design of DS4ML is not perfect yet, and experts have also put forward some constructive suggestions for its existing problems. For example, some experts pointed out that the generalizability of the interface still needs to be improved when dealing with different types of ML models; some experts also suggested enhancing the flexible integration capabilities with external ML libraries and pipelines.

These insights have been instrumental in shaping the future development of DS4ML. Specifically, they point out the need for improved extensibility, scalability, and domain adaptation. Based on this feedback, the next phase of development will focus on modularizing key components, supporting plug-in architectures for different ML model visualization and synthesis techniques for multi-stakeholder workflows.

6.5 Conclusion

In this chapter, we propose a visual analytics solution for targeted time series data synthesis in ML workflows. Through this work, we demonstrate that using visualization techniques will help ML developers synthesize appropriate training data, thereby improving ML model performance. ML developers can use visualization techniques to visually compare the differences between synthesized training data and original data, infer potential issues in model performance using their knowledge, and address these issues by selecting suitable synthetic data. Through experiments and expert advice, we prove that using visualization techniques can effectively help achieve comparison and control of synthetic data, thus enhancing the benefits of synthetic data for ML model training.

Building on this foundation, we completed the prototype design and development of DS4ML. DS4ML is an ML model testing and data synthesis application designed to help users generate and select appropriate training data and test models. It provides a user-friendly interface that allows users to interact with test models and datasets. Through the visualization techniques, DS4ML achieves the goal of helping users better train their ML models using synthetic data. It uses a user interface and front-end and back-end processing programs developed based on Python. We welcome interested researchers to deploy it locally and provide valuable suggestions.

To our knowledge, this approach has not been previously reported in the literature. We will continue this work, further developing enterprise-level applications based on the existing program. The application currently supports decision tree-based models(XGBoost, random forest, etc.) as tested models and will continue to be developed to accommodate other types of ML models. At the same time, we anticipate that there are certainly many more synthetic data algorithms that can help improve the efficiency of ML models, and there are undoubtedly other iterative approaches to improve performance from individual categories to overall performance. We believe that the more ML developers are able to use visualization techniques for their model development, the more effective approaches will be discovered.

Using the concept discussed in Chapter 2, DS4ML addresses the inconsistency between development and deployment data spaces for time-series classification. By introducing a Synthesis View and coupling it with Dataset, Model, and Result Views, DS4ML enables targeted synthesis and visual validation of state-transition structures, aligning the development training data space with the deployment data space. Experiments on public HAR data show that such data-centered remediation improves robustness beyond experiment-based optimization alone.

During the development of DS4ML, due to the involvement of visualization research on state change processes in time series data, this research gave rise to our ideas for a new visualization design, which will be presented in the next chapter.

7

Radial Icicle Tree (RIT): Node Separation and Area Constancy

Icicles and sunbursts are two commonly-used visual representations of trees. While icicle trees can map data values faithfully to rectangles of different sizes, often some rectangles are too narrow to be noticed easily. When an icicle tree is transformed into a sunburst tree, the width of each rectangle becomes the length of an annular sector that is usually longer than the original width. While sunburst trees alleviate the problem of narrow rectangles in icicle trees, it no longer maintains the consistency of size encoding. At different tree depths, nodes of the same data values are displayed in annular sections of different sizes in a sunburst tree, though they are represented by rectangles of the same size in an icicle tree. Furthermore, two nodes from different subtrees could sometimes appear as a single node in both icicle trees and sunburst trees. In this Chapter, we propose a new visual representation, referred to as *radial icicle tree* (RIT), which transforms the rectangular bounding box of an icicle tree into a circle, circular sector, or annular sector while introducing gaps between nodes and maintaining area constancy for nodes of the same size. We applied the new visual design to several datasets. Both the analytical design process and user-centered evaluation have confirmed that this new design has improved the design of icicles and sunburst trees without introducing any relative demerit.

7.1 Problem in the Existing Visualization Methods

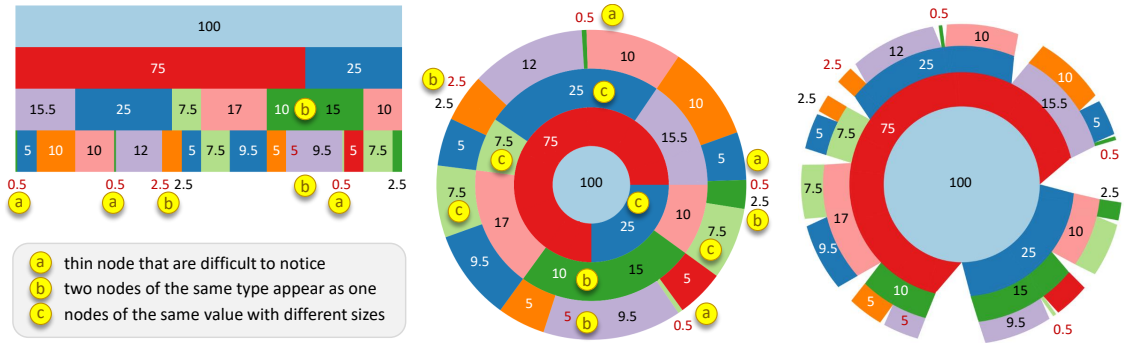


Figure 7.1: In a traditional icicle tree plot (left), thin nodes (i.e., nodes of small values) can be difficult to see, and two nodes of the same type (i.e., encoded using the same color) can appear as one node, though they belong to different subtrees. A traditional sunburst plot (middle) has a third issue, i.e., nodes of the same value can be mapped to different sizes. These three issues are labeled as (a), (b), (c) in the figure. As shown on the right, a new visual design, referred to as *Radial Icicle Tree* (RIT), addresses these three issues.

Visual clarity and consistency have always been among the desired qualities of data visualization. Many have made strong arguments for maintaining such qualities in most, if not all, visual representations. For example, Tufte made a powerful argument of "graphics integrity" and coined the term "lie factor" to indicate the level at which a visualization image deviated from its source data [236]. Kindlmann and Scheidegger defined three principles, namely "representation invariance", "unambiguous data depiction", and "visual-data correspondence", to formalize the notion of graphical integrity mathematically [237]. Although such graphics integrity or principles cannot always be maintained by some commonly-used visual representations (e.g., in volume visualization and metro maps [252, 253]), they are nevertheless desired qualities.

Icicle tree plots [218] and *sunburst tree plots* [223] are two commonly-used visual representations for tree visualization. As illustrated in Figure 7.1, some visual patterns depicted by these plots may exhibit undesirable qualities. As illustrated on the left of Figure 7.1, in an icicle tree, some thin nodes may be difficult to notice (marked as (a) in the figure), and two nodes that belong to different subtrees but are encoded using the same color (e.g., because of same categorical label or

semantic type) may visually appear as a single node (marked as (b) in the figure). As illustrated in the middle of Figure 7.1, a sunburst tree may feature nodes that are of the same data values but mapped to visual objects of different sizes (marked as (c) in the figure). Meanwhile, a sunburst tree may also exhibit issues (a) and (b).

For the aforementioned strong arguments of graphics integrity, issue (c) is a serious problem. Even with a more accommodating argument, such as the cost-benefit trade-off [254], issue (c) could lead to potential distortion and cognitive cost that should ideally be reduced. Although from the perspective of visual encoding, one could argue that issues (a) and (b) have not breached the aforementioned strong arguments by Tufte [236] and Kindlmann and Scheidegger [237] since the visual objects are encoded with a "correct" mapping, and viewers can zoom-in to address the issue (a) and can infer the separation of visually-merged nodes by tracing the separation lines from their parent nodes, grandparent nodes, and so on. For example, in Figure 7.1, presumably, one could infer the separation of the two green nodes (green-10 and green-15) in the sunburst tree (middle-bottom) based on the separation of their parent nodes (red-75 and blue-25). One could then infer the separation of their child nodes pale-purple-5 and pale-purple-9.5. However, with Chen and Golan's argument [254], the cost-beneficial ratio would be unfavorable as the effort for cognitive reasoning and interaction would be high.

In this Chapter, we propose a new visual representation that bears some resemblance to the designs of icicle trees and sunburst trees but addresses the aforementioned issues mathematically and algorithmically. In particular, we introduce visual gaps between neighboring nodes to improve the separation of sub-trees and the visibility of thin nodes, while maintaining the numerical consistency in mapping data values to sizes of visual objects (i.e., areas of annuli and annular sectors). This new visual representation is referred to as *Radial Icicle Tree* (RIT). We report our design process in Section 7.2, where we also present our design rationales by analyzing the symptoms, causes, remedies, and side-effects [255] of major design options considered in the process. We mathematically confirm the area constancy of our approach in Section 7.3, and provide a recursive algorithm for drawing an RIT

in Section 7.4. In Section 7.6, following several images for testing different layouts of RITs, we demonstrate the uses of this new visual representation by applying it to two public-domain datasets as well as utilizing it to study movement data in an application. In Section 7.7, we discuss the limitations of icicle and sunburst trees that RITs could not resolve completely, and a few possible variants of RITs that could be studied in the future. This is followed by our concluding remarks in Section 7.8.

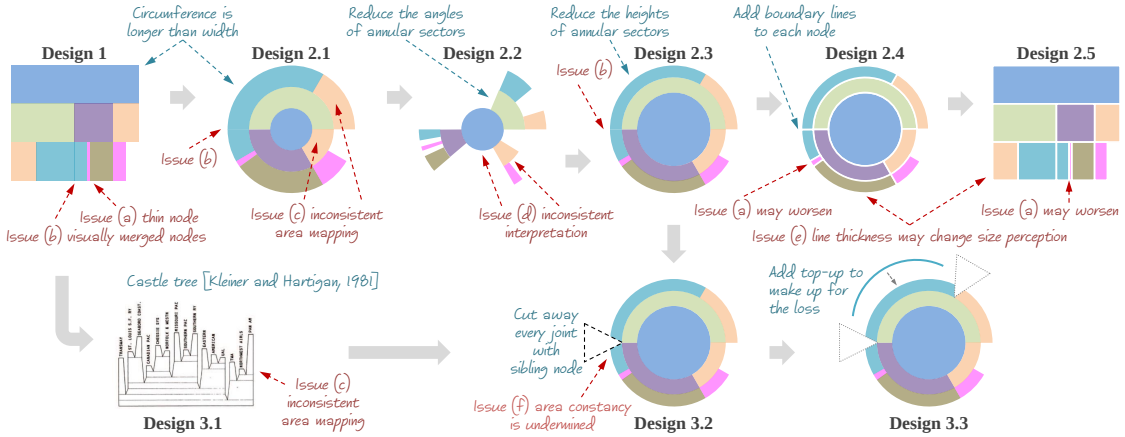


Figure 7.2: The major designs considered in our design process, where some designs would introduce new issues while addressing the existing issues. Although Design 3.2 was derived from Design 2.3, we might have unconsciously been influenced by Kleiner and Hartigan's castle tree [219].

7.2 Radial Icicle Tree: Design Rationales

As described in Section 7.1 and as illustrated in Figure 7.1, this work proposes a new visual design for addressing the two issues associated with both icicle and sunburst tree plots and another issue associated only with sunburst trees. In this section, we delineate our process for developing a new design, sketch out the main design options, and describe the design rationales considered in the process. In our design process, we adopted the systematic methodology for improving a visual design (or visual analytics workflow) proposed by Chen and Ebert [255] by using the icicle tree design as the starting point for analyzing symptoms, causes, remedies, and side effects about a visual representation. In Figure 7.2, we illustrate the design process starting from Design 1 on the top-left to Design 3.3 on the bottom-right.

Design 1: *Icicle Tree Plot*

- **Symptom:** (a) Hard-to-see thin nodes and (b) visually-merged nodes as illustrated in Figure 7.1.
- **Cause:** The width of the display space is limited, causing thin nodes and a lack of gaps between subtrees and nodes.
- **Remedy:** Transform it to polar coordinates (i.e., sunburst) to gain more space as the circumference is longer than the width. It can alleviate (a) as some thin nodes become more noticeable.
- **Side-effect:** It does not address (b) while introducing issue (c) of inconsistent size encoding.

We then considered the design of the sunburst tree as **Design 2.1** in Figure 7.2. The side-effect in **Design 1** became the symptom in **Design 2.1**.

Design 2.1 $\sim$ 2.5 *Sunburst Tree Plot and its Variants*

- **Symptoms:** Aforementioned issues (b) and (c). Issue (a) is not fully addressed.
- **Cause:** The size encoding based on rectangles is now distorted. The gaps between subtrees and nodes have been abstracted away.
- **Remedy:** One remedy may be to add larger gaps proportionally to ensure area constancy, resulting in Design 2.2 in Figure 7.2.
- **Side-effect:** A new issue, (d) arises, i.e., the interpretation of these gaps is inconsistent with the conventional interpretation about a portion of the outer edge that is not connected to any child node. Conventionally, it would mean that a parent node does pass all of its data values to its child nodes. This interpretation cannot be applied to the "new type" of gaps introduced for area constancy.

- **Remedy:** Instead of reducing the angle of each annular sector, another remedy may be to reduce the height of each annular sector to ensure that the corresponding full annulus has the same area as other full annuli (including the root node that may be depicted as a circle or a full annulus). This is shown as Design 2.3 in Figure 7.2.

- **Side-effect:** Issue (b) reappears, while the advantage gained by Design 2.1 for addressing the issue (a) starts dissipating because reducing the height of a small annular sector will also make the sector narrower in terms of its arc length.

- **Remedy:** We can add boundary lines between nodes to address (a) and (b) as shown in Design 2.4 in Figure 7.2.

- **Side-effect:** Boundary lines may take up some space that would worsen issue (a) for small and thin nodes. They may change the size perception, and such changes will affect small and thin nodes more. We consider this as Issue (e).

- **Remedy:** Boundary lines can also be applied to icicle tree plots as shown in Design 2.5 in Figure 7.2.

- **Side-effect:** Similar to Design 2.4, issue (a) may become worse for small and thin nodes, and issue (e) may occur as the size perception is also affected.

In practice, the design variants, such as adding boundary lines to icicles and sunburst trees, are relatively common. Nevertheless, the design process led us to a new approach for addressing these issues. The approach is to take the idea of creating bigger gaps from Design 2.2. Instead of removing a portion of an annular sector in the shape of a smaller annular sector (i.e., with a smaller angle), we may remove a fan-like shape such that the inner arc of an annular sector is not shortened, avoiding the creation of issue (d). Later, we discovered that the idea of cutting away a wedge was first reported by Kleiner and Hartigan [219] in the context of an icicle tree. Likely, our Design 3.2 might be influenced by their castle tree design, which is shown as Design 3.1 in Figure 7.2.

Design 3.2 and Design 3.3: *Radial Icicle Tree*

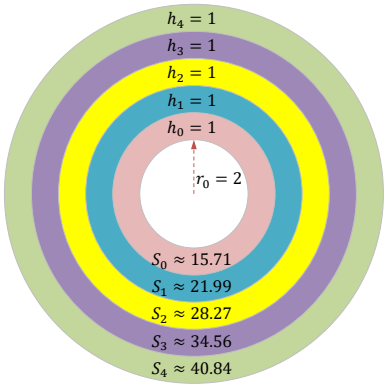
- **Symptoms:** Design 2.3 addressed issue (c), but not an issue (b). Meanwhile, issue (a) is not fully addressed.
- **Remedy:** Similar to Design 3.1 for an icicle tree, we can also cut away a triangle or fan-like shape at each end of each annular sector. Note that only one of such cuts is illustrated in Design 3.2 in Figure 7.2, though the cuts are to be applied to every annular sector, except a full circle or a full annulus. The cuts enable the separation of nodes and subtrees.
- **Side-effect:** Naturally, issue (f) arises since the cuts mean size loss, which would undermine our objective of area constancy.
- **Remedy:** Issue (f) can be addressed by adding a thin top-up annular sector to each annular sector that has lost a portion of its area due to the cuts. As long as we can ensure that the top-up sector is of the same size as the lost area, we can maintain area constancy. This is shown as Design 3.3 in Figure 7.2.

After we established that Design 3.3 could address the main symptoms and causes related to issues (a), (b), and (c) as illustrated in Figure 7.1 without incurring any significant side-effects as illustrated in Figure 7.2, we needed to consider the mathematical and algorithmic mechanisms for realizing the design. There were still unsolved technical problems, such as how to calculate the lost area that a top-up sector would need to make up for. In Section 7.3, we will discuss the mathematical properties of Design 3.3, and in Section 7.4, we will outline a recursive plotting algorithm for realizing this design.

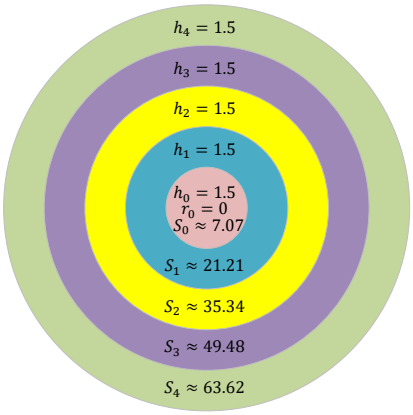
7.3 Radial Icicle Tree: Mathematical Concepts

7.3.1 Area Constancy

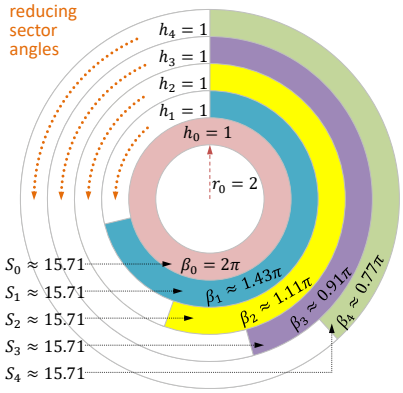
The traditional sunburst design does not maintain *area constancy* among different annuli. Figures 7.3(a, b) show two example sunburst plots depicting two simple



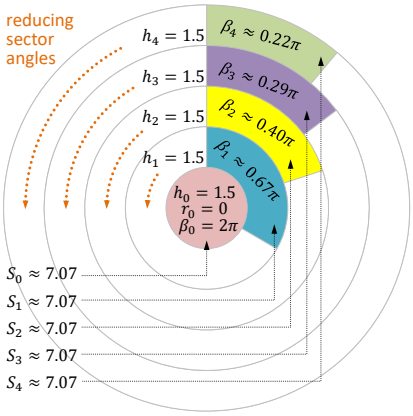
(a) $r_0 = 2;\ h_0 = h_1 = \ldots = 1$



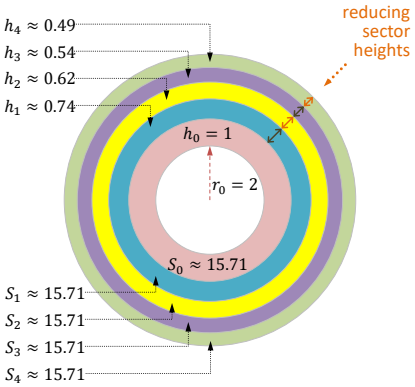
(b) $r_0 = 0;\ h_0 = h_1 = \ldots = 1.5$



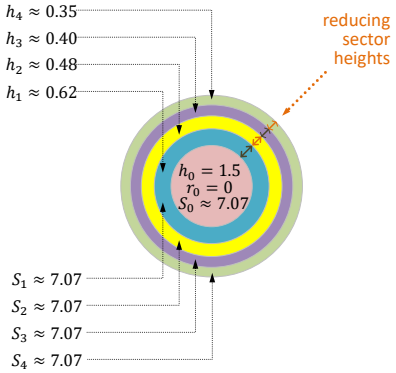
(c) observing size difference in (a)



(d) observing size difference in (b)



(e) reducing sector heights in (a)



(f) reducing sector heights in (b)

Figure 7.3: Two examples of size inconsistency are shown in (a, b), where the annuli in each sunburst plot have the same height, causing the increasing areas $S_0 < S_1 < \ldots < S_4$. The scale of inconsistency can be observed more easily in (c, b). One approach to maintain area constancy is to reduce the sector heights gradually, i.e., $h_0 > h_1 > \ldots > h_4$.

trees, respectively, each with five nodes. In each example, each parent node has exactly one child, and all nodes have the same data value of 1 (i.e., 100%). In (a), the root annulus (pale red) starts at inner radius $r_0 = 2$, and all nested annuli have the same height, i.e., $h_0 = h_1 = \ldots = h_4 = 1$. The area of each annulus S_i is given at the lower part of the figure, from which we can easily observe that different annuli are of different sizes. In (b), the root annulus is the circle in the middle (i.e., $r_0 = 0$), while $h_0 = h_1 = \ldots = h_4 = 1.5$. Similarly, area constancy is not maintained for this simple sunburst tree.

One way to observe the scale of the size difference is to maintain the root node annulus as a full annulus in (a) and a full circle in (b), and then change each annulus to an annular sector with an angle $\beta < 2\pi$ such that the area of the annular sector is the same as the area of the corresponding root. Figures 7.3(c, d) illustrate such changes. Of course, this will not be a suitable approach for ensuring area constancy because some portion of the outer arc of each node shape (i.e., circle, annulus, or annular sector) is not connected to anything, and it will be interpreted as if a parent node does not pass the full amount of its data value to all of its child nodes, i.e., the data value of the parent is less than the sum of the data values of all of its child nodes.

A more suitable approach is to reduce the heights of annular sectors gradually outwards from the root to its leaves in the tree, as illustrated in Figures 7.3(e, f). Note that there is only one leaf in each of the two simple examples in the figure. For each annulus, its area can be computed as:

$$S_i = \pi\Big((r_i + h_i)^2 - r_i^2\Big), \quad i = 0, 1, \ldots \tag{7.1}$$

where i indicates the level of an annulus that corresponds to the depth of a tree, such that $i = 0$ indicates a tree root and the innermost annulus. Furthermore, $r_{i+1} = r_i + h_i$ with r_0 is predefined. In Figures 7.3(a, b), the same annular height was used for all annuli in a plot, i.e., $h_0 = h_1 = \ldots = h_4$, hence causing the increasing annular areas outwards. In Figures 7.3(e, f), h_0 is predefined, while $h_1, h_2, \ldots$ are adjusted to ensure all annular areas are the same as S_0.

Theorem 1. Let the inner radius of an annulus be a known value r_i. When its area is set to a standard size S_{std}, the height of an annulus is:

$$h_i = -r_i + \sqrt{r_i^2 + S_{\text{std}}/\pi} \tag{7.2}$$

Proof. From Eq. 7.1, we obtain $h_i^2 + 2r_i h_i - S_{\text{std}}/\pi = 0$. Because h_i is expected to be ≥ 0 and since $r_i \geq 0, S_{\text{std}} \geq 0$, the quadratic equation has only one solution as given in Eq. 7.2. □

With this Theorem, we can derive appropriate values for $h_1, h_2, \ldots$ in Figure 7.3(e, f), where area S_0 is set as the standard size S_{std}.

Note that the area of an annular sector can be calculated using a formula similar to Eq. 7.1 by replacing π with half of the angle of the sector, i.e., for an annular sector with an arc angle $0 < \beta < 2\pi$, inner radius r, and height h, its area is:

$$S_{\text{a-sec}} = \frac{1}{2}\beta\Big((r+h)^2 - r^2\Big) \tag{7.3}$$

Given a predefined area $S_{\text{a-sec}}$ for the sector, we can derive its height using a formula similar to Eq. 7.2 by replacing S_{std} and π with $S_{\text{a-sec}}$ and $\frac{1}{2}\beta$ respectively. In fact, we can use Eq. 7.2 directly, since:

$$\beta S_{\text{std}} = 2\pi S_{\text{a-sec}}$$

7.3.2 Node Separation

As discussed in Section 7.2, the icicle design typically packs the nodes more tightly than the sunburst design, often making it hard to distinguish individual nodes. In a way, the sunburst design can be seen as a deformed icicle design, where all shapes are collectively transformed from Cartesian Coordinates to polar coordinates. As discussed in Section 7.2, this partly alleviates the hard-to-distinguish problem as nodes become wider since the angular axis is usually longer than the horizontal axis. However, unlike a node-link design, there is no space between nodes. Naturally, as illustrated in Figure 7.2, introducing a gap between each pair of neighboring nodes at the same level would address the issue (a) – the distinguishability of thin nodes, while preventing issue (b) – the phenomena of merged nodes.

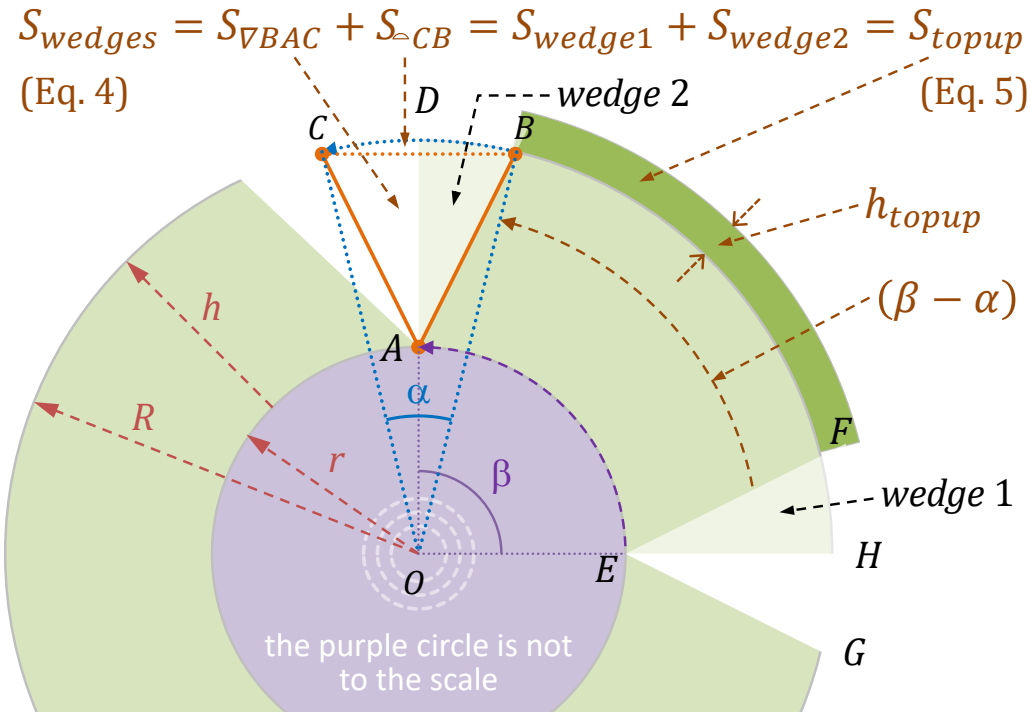


Figure 7.4: The two wedges to be removed and the top-up annular sector.

In a sunburst design, a node that has a data value of less than 1 (i.e., $< 100\%$) is represented by an annular sector. One approach is to cut off a wedge at each end of the sector, opening a gap with the neighboring sectors as shown in Figure 7.4. Let the wedge at one end be mirrored at the other end. For area calculation, we can consider the combined shape that consists of the triangle $\triangledown BAC$ and the circular segment $\frown CB$ as shown in Figure 7.4. The area of the two wedges is thus the sum of the area of $\triangledown BAC$ and that of $\frown CB$. $\frown CB, S_{\frown CB}$

Let the annular sector be defined with r (inner radius), h (height), and β (angle in radian). Let the segment $\frown CB$ has an angle $\alpha < \beta$. We can calculate the area of the two wedges to be removed as the sum of the areas of $\triangledown BAC$ and $\frown CB$, that is:

$$\begin{aligned} S_{\text{wedges}} &= S_{\frown CB} + S_{\triangledown BAC} = S_{\frown CB} + (S_{\triangledown BOC} - 2S_{\triangledown BOA}) \\ &= \frac{1}{2}(r+h)^2(\alpha - \sin\alpha) + \frac{1}{2}(r+h)^2 \sin\alpha - r(r+h)\sin\frac{\alpha}{2} \end{aligned} \tag{7.4}$$

Since the removal of the two wedges from the annular sector causes area loss, we need to expand the remaining part of the annular sector somehow to make up for the loss. To maintain the same look-and-feel, we simply add a *top-up* annular sector with the same color and with its area equal to the lost area, i.e., $S_{\text{wedges}} = S_{\text{topup}}$.

As the outer arc of the remaining part has an angle of $\beta - \alpha$ and its inner radius is $r + h$, we can derive the height of the top-up sector, h_{topup}, as:

$$S_{\text{wedges}} = S_{\text{topup}} = (\beta - \alpha)\Big((r + h + h_{\text{topup}})^2 - (r + h)^2\Big) \tag{7.5}$$

$$\implies h_{\text{topup}} = -(r + h) + \sqrt{(r + h)^2 + S_{\text{wedges}}/(\beta - \alpha)} \tag{7.6}$$

7.3.3 Maximum of Wedge Angle

From Figure 7.4, we can observe that the combined wedge angle α used to remove two wedges at the ends of the annular sector must be smaller than β. If $\alpha > \beta$, the two wedges, ABD and EFH would overlap with each other. If $\alpha = \beta$, the two wedges would join at the middle point along the arc between B and F, and it would not be possible to add a top-up annular sector since $\beta - \alpha = 0$ in Eq. 7.6.

In fact, it is not really desirable to have α anywhere near β because this would make the top-up sector have a very small angle, which is defined by $\beta - \alpha$. Therefore, we define a wedge angle ratio $ar = \alpha/\beta$. Although ar is mathematically constrained by $0 < ar < 1$, we recommend to maintain a stricter restriction such as $0 < ar < 0.5$. In an implementation, one typically sets ar to 0.1 (i.e., α is 10% of β) as a constant for all annular sectors except the root node that does not need wedge removal. One may also decrease ar gradually when the tree depth increases.

Another necessary constraint is illustrated in Figure 7.5. If one were to remove the wedge ABD, the wedge would include an area (indicated by a red arrow ⇓) that is not on the annular sector (shown in orange). The constraint is determined by the angle between the lines AB and AC, which cannot exceed 90 degrees (or $\frac{1}{2}\pi$ in radian). From the triangle $\triangledown AOC$, we can derive:

$$\cos(\frac{1}{2}\alpha_{\max}) = \frac{r}{R} \quad \implies \quad \alpha_{\max} = 2\cos^{-1}\Big(\frac{r}{R}\Big)$$

We, therefore, recommend the maximum rule for the wedge angle as:

$$0 < \alpha < \min\left(\frac{1}{2}\beta,\ 2\cos^{-1}\Big(\frac{r}{R}\Big)\right) \tag{7.7}$$

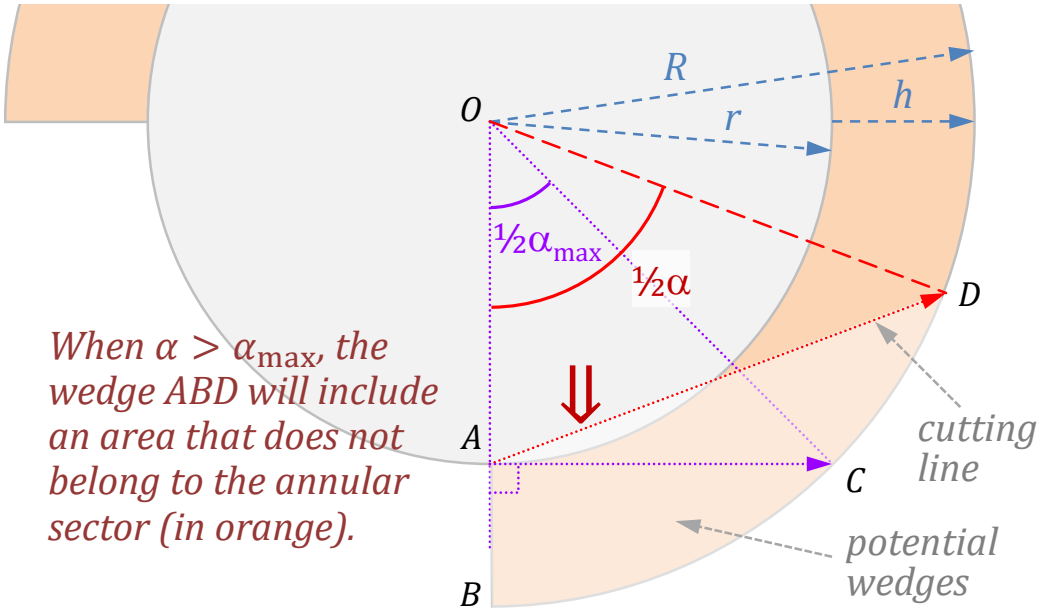


Figure 7.5: The double-wedge angle α is restricted by two constraints.

Theorem 2. Given an annular sector defined with r (inner radius), h (height), β (sector angle), and $\frac{\alpha}{2}$ (wedge angle), and given that α meets the condition of Eq. 7.7, we can remove two wedges at the ends of the sector and add a top-up sector defined with $r + h$ (inner radius), h_{topup} (height), and $\beta - \alpha$ (sector angle). The resulting shape enables node separation while maintaining area constancy.

Proof. In Eq. 7.7, $\alpha > 0$ assures that two wedges will be removed, hence, assuring node separation. $\alpha < \frac{1}{2}\beta$ assures that a top-up area can be added. $\alpha < \alpha_{\max}$ assures that the area lost by an annular sector is exactly the area of the two wedges S_{wedges}.

Theorem 1 assures area constancy before removing any wedge. Eqs. 7.4, 7.5, and 7.6 assure that the area of the two removed wedges S_{wedges} is the same as the area of the top-up sector S_{topup}. Hence, the area constancy is maintained after removing the two wedges and replacing them with a top-up sector according to Eqs. 7.4, 7.5, and 7.6. □

7.4 Radial Icicle Tree: Recursive Drawing Algorithm

Tree drawing algorithms often make use of a recursive procedure for drawing subtrees. We adopt the same approach in the design of an algorithm for drawing radial icicle trees (RITs). The algorithm starts with the main procedure `draw_rit()` that draws the root node as an annulus or an annular sector. It then recursively invokes the procedure `draw_subtree()` for drawing its child nodes. After drawing each child node, `draw_subtree()` recursively invokes itself for drawing the child nodes of the node that has just been drawn.

The algorithm assumes that the tree data structure consists of a list of nodes, and the data of each node is accessed through a handler n, which can be an index or a pointer depending on the implementation. The data record of each node contains several fields, including $n.Children$ (handlers of child nodes), $n.Data$ (data value), $n.Color$ (node color), $n.\theta$ (starting angle of an annular sector), $n.\beta$ (arc angle of an annular sector), and $n.\alpha$ (double-wedge angle of an annular sector). A more detailed description of these fields can be found in Section 7.5.1.

Algorithm 3 outlines the top level of the RIT algorithm, where n is the handler of the root node. θ_0, β_0, r_0, and h_0 specify the annular sector for the root node. When $\theta_0 = 0,\ \beta_0 = 2\pi$, it is a full annulus. When $\theta_0 = 0,\ \beta_0 = 2\pi,\ r_0 = 0$, it is a circle with radius h_0. C_{ar0} and C_{acr} are two constants for controlling the double-wedge angle of each annular sector except for the root node. C_{ar0} is the initial angle ratio α/β, which is used for removing wedges in `draw_subtree()`. $C_{ar0} > 0$ is the angle change rate (> 0) that defines how the angle ratio ar may change according to

Algorithm 3: The main RIT algorithm: `draw_rit()`

Data: $n,\ \theta_0,\ \beta_0,\ r_0,\ h_0,\ C_{ar0},\ C_{\text{acr}}$

1 draw_a_sector(n.Color, $\theta_0,\ \beta_0\ r_0,\ h_0$);

2 $A_{\text{std}} \leftarrow 0.5 \cdot \beta_0 \cdot \left((r_0 + h_0)^2 - r_0^2\right)$; `// standard area`

3 $r_{\text{new}} \leftarrow r_0 + h_0$;

4 $h_{\text{new}} \leftarrow$ calculate_normalised_height($r_{\text{new}}, A_{\text{std}}$);

5 draw_subtree($n,\ \theta_0,\ \beta_0,\ C_{ar0},\ r_{\text{new}},\ h_{\text{new}},\ A_{\text{std}},\ C_{\text{acr}}$);

the tree depth. When $C_{\text{acr}} = 1$, the angle ratio does not change, i.e., the initial value $C_{ar0} > 0$ will be used for all nodes except the root node. When $C_{\text{acr}} = 0.9$, the angle ratio decreases gradually from the first generation of child-nodes towards the leaf-nodes, i.e., C_{ar0} for children, $C_{ar0}C_{\text{acr}}$ for grandchildren, $C_{ar0}C_{\text{acr}}^2$ for great-grandchildren, and so on. C_{ar0} and C_{acr} are typically defined as global constants.

Algorithm 4 outlines a recursive procedure for processing a subtree. The procedure is invoked after the parent node n has already been drawn. The procedure `draw_subtree()` receives the following inputs from the proceeding calling procedure:

- n — a handler (e.g., an index or a pointer) of the parent node.
- θ, β — the starting angle and the arc angle of the fan-shaped sector for accommodating this subtree.
- ar — the angle ratio α/β for the two wedges to be removed.
- r — the outer radius of the parent node n, which is the inner radius of its child nodes' annular sectors.
- h — the tentative height (i.e., without any top-up sector) of these child nodes' annular sectors.
- A_{std} — standard area for a 100% annular sector, which is usually set based on the area of the root node of the whole tree.
- C_{acr} — angle change rate (> 0) that defines how the angle ratio ar may change according to the tree depth.

The procedure `draw_subtree()` makes use of several subroutines, which are detailed in Section 7.5.2.

7.5 Further Details of the RIT Drawing Algorithm

In this part, we provide further details of the RIT algorithm presented in Section 7.4 in the main body of this chapter.

Algorithm 4: Recursive procedure: `draw_subtree()`

Input: $n,\ \theta,\ \beta,\ ar,\ r,\ h,\ A_{\text{std}},\ C_{\text{acr}}$

`// DRAW all children's annular sectors`

$\theta_c \leftarrow \theta$;
for $c_i \in n.Children$ **do** `// for each child node`
  $c_i.\beta \leftarrow 2 \cdot \pi \cdot c_i$.Data;
  $c_i.\theta \leftarrow \theta_c$;
  draw_a_sector(c_i.Color, $c_i.\theta,\ c_i.\beta\ r,\ h$);
  $\theta_c \leftarrow \theta_c + c_i.\beta$;
end

`// REMOVE 2 wedges (`0.5α`) for each child`

for $c_i \in n.Children$ **do** `// for each child node`
  $c_i.\alpha \leftarrow$ set_wedge_angle($ar, c_i.\beta$);
  remove_wedge_start($c_i.\theta,\ c_i.\alpha,\ r,\ h$);
  remove_wedge_end($c_i.\theta + c_i.\beta,\ c_i.\alpha,\ r,\ h$);
end

`// ADD top-up sectors & RECURSION for subtrees`

for $c_i \in n.Children$ **do** `// for each child node`
  $\theta_t \leftarrow c_i.\theta + 0.5 \cdot c_i.\alpha$;
  $\beta_t \leftarrow c_i.\beta - c_i.\alpha$;
  $A_{\text{wedge}} \leftarrow$ calculate_wedge_area($r,\ h,\ c_i.\alpha$);
  $r_t \leftarrow r + h$;
  $h_t \leftarrow$ calculate_topup_height($r_t,\ c_i.\alpha,\ c_i.\beta,\ A_{\text{wedge}}$);
  draw_a_sector(c_i.Color, $\theta_t,\ \beta_t,\ c_i.\alpha,\ h_t$);
  $r_{\text{new}} \leftarrow r_t + h_t$;
  $h_{\text{new}} \leftarrow$ calculate_normalised_height($r_{\text{new}}, A_{\text{std}}$);
  draw_subtree($c_i,\ \theta_t,\ \beta_t,\ ar \cdot C_{\text{acr}},\ r_{\text{new}},\ h_{\text{new}},\ A_{\text{std}},\ C_{\text{acr}}$);
end

7.5.1 The Main Data Fields in a Node Record

Each node record consists of the following data fields:

- $n.Children$ — a set of node handlers linking to all child nodes of node n. These child nodes may be stored in an array or a linked list. We denote them simply as $c_1, c_2, \ldots, c_i, \ldots \in n.Children$.

- $n.Data$ — a normalized data value to be mapped to the size of a node in a tree plot. Here we assume that the value is in the range of [0, 1], and $n.Data = 1$ for the root node, indicating that it is mapped to 100% of the standard area A_{std}. Typically, we may draw a root node as a full annulus

defined by its inner radius r_0 and height h_0. We can use Eq. 7.1 to obtain its area S_0 and define $A_{\text{std}} = S_0$. Alternatively, we may draw the root node as an annular sector that starts at a polar angle θ and ends at an angle $\theta + \beta$. Here we specify angles in radian and confine β to $0 < \beta \leq 2\pi$. θ is normally specified in the range $[0, 2\pi]$, though it is not difficult for an algorithm to deal with any finite value of θ and convert it to an angle within the range $[0, 2\pi]$ whenever necessary.

- $n.Color$ — the color of node n. Here, we assume that the color of each node has already been assigned for simplicity and clarity in describing the algorithm. In a practical implementation, the tree data structure will likely store application-specific data (e.g., type, quality level, etc.), which is mapped to color during the drawing process.

- $n.\theta$, $n.\beta$, $n.\alpha$ — the starting angle, the arc angle, and the double-wedge angle of an annular sector corresponding to node n. These are computed dynamically by the algorithm, except that those for the root node are predefined before invoking `draw_rit()`. In fact, it is not necessary to store these in the data record of n, except that it is slightly safer and more convenient to use these fields than internal variables in a recursive algorithm.

7.5.2 The Main Subroutines in the RIT Algorithm

The main procedure `draw_subtree()` in Algorithm 4 (Section 7.4) makes use of several subroutines, including:

- `draw_a_sector()` — This calls low-level one or more graphics subroutines to draw an annular sector.

- `set_wedge_angle()` — This first determines α as a portion of β according to the angle ratio ar, i.e., $\alpha \leftarrow ar \cdot \beta$. It then checks to see if α violates the two constraints according to Section 7.3.3. If α does, it adjusts α to meet the maximum rule.

- `remove_wedge_start()` — This removes a wedge ($\frac{1}{2}\alpha$) at the starting end of an annular sector (see Section 7.3.2).

- `remove_wedge_end()` — This removes a wedge ($\frac{1}{2}\alpha$) at the terminating end of an annular sector (see Section 7.3.2).

- `calculate_wedge_area()` — This calculates the area of two wedges to be removed as described in Eq. 7.4 in Section 7.3.2.

- `calculate_topup_height()` — This calculates the height of a top-up annular sector as described in Eq. 7.6 in Section 7.3.2. Note that the term $(r + h)$ is replaced with r_t in `draw_subtree()`.

- `calculate_normalised_height()` — This calculates the height of an annular sector as described in Eq. 7.2 in Section 7.3.1. It does not take into account the area loss due to wedge removal, which will be compensated by the top-up annular sector.

7.6 Testing and Results

In this section, we report the testing of the visual design and the algorithm of RIT in conjunction with synthetic data featuring the three issues associated with icicle and sunburst trees, open data in the public domain, and application-specific data.

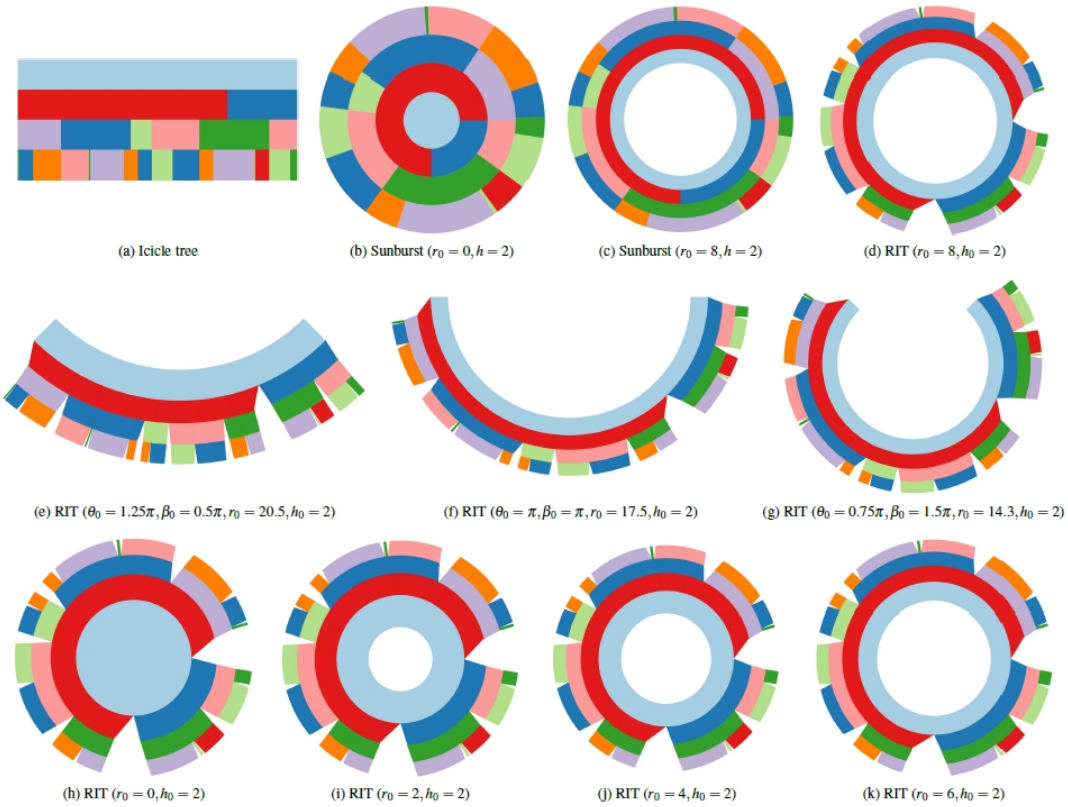


Figure 7.6: The dataset used to illustrate the three issues in Figure 7.1 was used to test RIT plots with different configuration parameters.

7.6.1 Problem-driven Testing

We constructed several testing datasets that exhibit issues (a), (b), and (c) discussed in Section 7.1. This allowed us to control the complexity of a tree structure, the number of problematic nodes, the severity of the problems, where in the tree the problems occur, and so on. Figure 7.1 shows one such synthetic dataset. As annotated in Figure 7.1, we assume that the color of each node encodes the categorical label of the node, and nodes with the same color are of the same category. The number in each node indicates its normalized data value such that the root is of the full amount of 100. The area of each node encodes the data value of the node. Figure 7.6 displays this synthetic dataset in different visual representations, i.e., (a) icicle tree, (b,c) sunburst trees, and (d∼k) RITs.

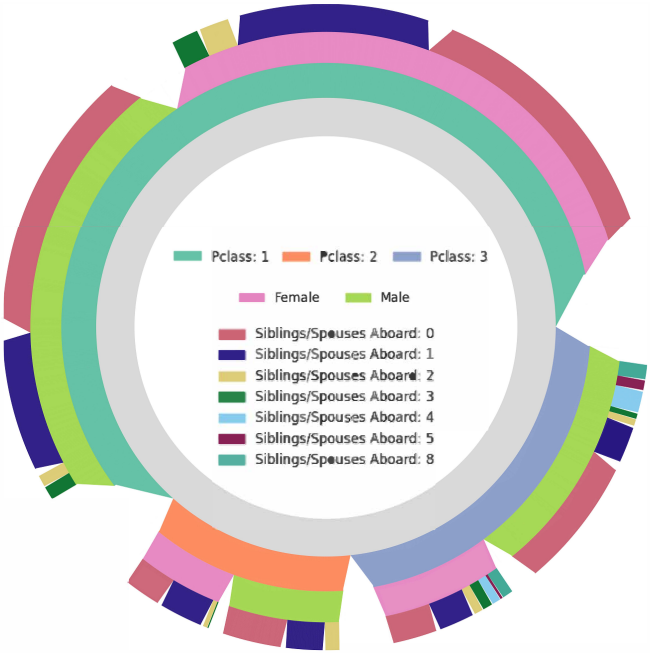
As we have already made observations about the three issues by comparing the three plots in Figures 7.6, here we focus on other observations. Firstly, Figure 7.6(e)

can be seen as a slightly-curved version of an icicle tree. Issue (b) – merged nodes – has been resolved completely, while it does not suffer from the issue (c) – inconsistent area mapping. Issue (a) – thin nodes – is significantly improved in comparison with Figure 7.6(a). The three thin leaf nodes (two in green and one in pale green) are visible but not quite noticeable. Their distinguishability improves when the angle of the root sector β_0 (in radian) increases from 0.5π in (e) to π in (f), then to 1.5π in (g), and 2π in (d). Comparing Figures 7.6(c,d), we can observe that the three thin nodes are more noticeable in (d) RIT than in (c) sunburst. Note that because of the need to ensure area constancy, we cannot enlarge thin nodes without enlarging other nodes. So nodes with small data values will always be relatively small unless one does not encode the data values using shape areas.

Similar to sunburst trees, RITs with a circular layout can adjust the inner radius of the root node, r_0, to change the size of the empty space at the center. Figures 7.6(h∼k,d) illustrate the change of $r_0 = 0, 2, 4, 6, 8$. With sunburst trees, r_0 is commonly set to a large value, so the issue (c) – inconsistent area mapping – is less obvious. Since RITs do not suffer from the issue (c), the change of r_0 may serve different purposes. For example, a user may wish to leave plenty of blank space in the center for some text or a legend. Alternatively, a user may wish for each annular sector to have a relatively large h_i for placing a text label.

7.6.2 Testing with Datasets in the Public Domain

We have also tested several open datasets in the public domain. Figure 7.7 shows two example RITs produced using two of these datasets, respectively. The Titanic passenger dataset contains some information about each passenger aboard the Titanic, including ticket class, gender, and a categorical label about how many accompanied persons. In Figure 7.7(a), the grey root annulus is the total number of passengers. At tree depth 2, passengers are clustered into three categories based on their ticket classes. At tree depth 3, passengers are then divided into male and female groups. At the leaf level, passengers are further divided into seven categories according to the number of accompanied persons. There are thin nodes in the tree,



(a) Titanic passenger dataset (data source: `https://web.stanford.edu/class/archive/cs/cs109/cs109.1166/stuff/titanic.csv`)

(b) The salesperson dataset (data source: `https://raw.githubusercontent.com/plotly/datasets/master/sales_success.csv`)

Figure 7.7: RIT plots for two open datasets in the public domain. The Titanic dataset features a good number of thin nodes, while the salesperson dataset has many nodes that need to be identified using text labels, as a color legend with many colors would not be appropriate.

indicating that some passengers with ticket classes 2 and 3 were traveling with many accompanied persons. The RIT plot is able to handle these thin nodes. The data indicate no passenger who was traveling with 6 or 7 accompanied persons. Passengers with first-class tickets have 3 or fewer accompanying persons.

Figure 7.7(b) shows a dataset with four regions, each region of which has several counties. Each leaf node is associated with an individual salesperson, and the size of the node encodes the amount of sales values by the salesperson. We designed a colormap that allocates four distinct hue ranges to the four regions, respectively. The colors of the counties and salespersons are algorithmically assigned such that the colors are of different HSV values, except that the hue value H is within a smaller hue-range defined by the color of the region.

The gaps between nodes are more important in this RIT as the node colors can be rather similar. As we can observe from Figure 7.7(b), all nodes are well separated.

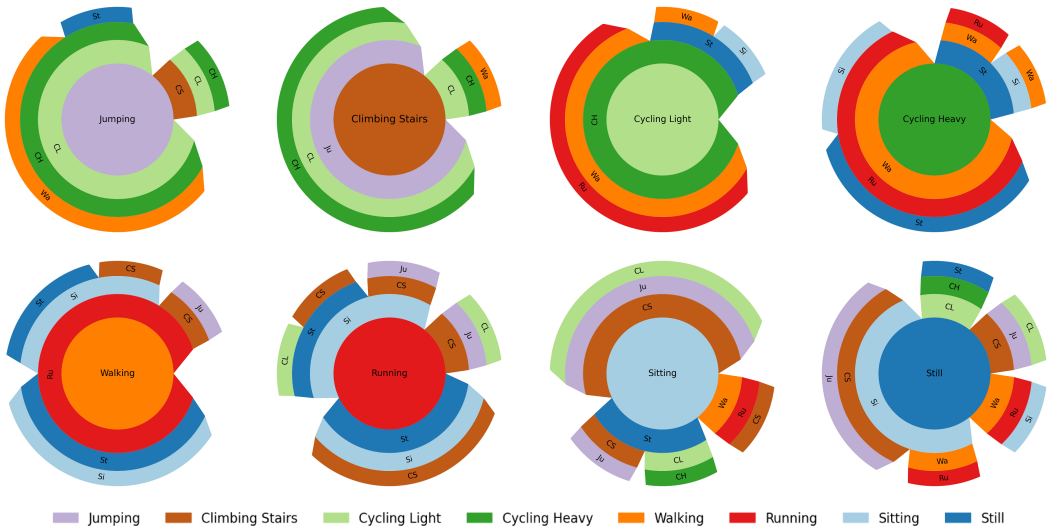


Figure 7.8: The statistics about three-step state transitions from each of the eight states to different states.

7.6.3 Testing with Data in a Specific Application

Statistics Netherlands collects, processes, and analyzes a variety of data that features tree or graph representations. There are two major categories of such data: (i) data represented as explicit tree structures (e.g., data pertaining to the geographical regions of a country, categories of land uses, performance indicators of economic

sectors, and classifications of goods), and (ii) tree structures extracted by processing other types of data. For example, some data sets are collected as temporal data with categorical and numerical values, such as employment data in socioeconomics and movement data in public transport. In the context of the latter, Statistics Netherlands is particularly interested in state transitions, e.g., unemployed to employed or changes in transport modality. To enable statistical analysis of the relations among different events in the data, the original temporal data is commonly processed to extract relational information into tree or graph representations.

To provide data about the general health and activity of the Dutch population, Statistics Netherlands regularly sends out surveys investigating the amount of physical activity in a representative sample of the Dutch population. A problem with these surveys is that the measured activity levels can be subjective. In general, self-reported activity levels are overestimated and introduce subjectivity in health statistics. In search of more objective measures, Statistics Netherlands, in cooperation with other public parties like the Dutch Health Authorities (RIVM), is investigating the use of accelerometers. Accelerometers measure the acceleration of a subject in three axes (x, y, z) and can be used to measure the intensity of physical activity, and moreover, be used to give an estimation of the type of activity performed. As such, accelerometer data can provide a more detailed view of physical activity during day-to-day life in the Dutch population. Using accelerometers, Statistics Netherlands hopes to obtain more detailed statistics about the amount of moderate to vigorous physical activity (MVPA), but also insights into sedentary behavior and sleep patterns in the general Dutch population.

Statistics Netherlands has been studying the statistics of different events in the data collected from wearable devices. One particular analytical need is to observe, compare, and reason the statistics about the transitions among different events. For example, we may consider eight types of movement events, including *climbing stairs*, *cycling heavy*, *cycling light*, *jumping*, *running*, *sitting*, *still*, and *walking*. We can use RIT plots, such as those in Figure 7.8, to observe the statistics about transitions from event x_1 to event x_2, then event x_3, and so on.

We first considered a number of visual representations for graphs, such as chord diagrams and Sankey diagrams. We found that it was difficult to compare different transition sequences in the same graph plot, e.g., to compare transitions:

running ⇒ sitting ⇒ climbing stairs
vs. running ⇒ climbing stairs ⇒ sitting
vs. walking ⇒ sitting ⇒ climbing stairs
vs. walking ⇒ climbing stairs ⇒ sitting

The main challenge seemed to be the costly cognitive effort required for multiple observation tasks (i.e., locate one transition sequence and remember it, and then locate and remember another) before a comparison task can be performed. This made us consider the use of multiple tree representations instead of a single graph representation.

When each type of event has its tree, depicting the statistics of different transitions starting from the specific type of event, makes the location of the first event much easier, while depicting the following events in a much less cluttered way. The individual tree plots also provide external memorization. This reasoning led us to consider various tree visualization techniques that could encode statistical data with node sizes, which include treemap, icicle tree, sunburst tree, and their variants. We found that one seemed to need more cognitive effort to perceive the root and internal nodes in a treemap, and it was easier to notice the root and internal nodes with an icicle or a sunburst tree. However, we also identified some issues as illustrated in Figure 7.1. This became part of the motivation for proposing a new visual design that could address these issues.

Figure 7.8 shows an example set of statistics for 3-transition sequences. The raw data were collected in a laboratory setting, and the main observational and analytical tasks were to compare the statistics of different transitional sequences (e.g., between cycling heavy and cycling light in Figure 7.8) to see if they meet the anticipated statistical properties of the data collection exercise, and to compare with data collected in real-life and other laboratory settings.

In one of the evaluation meetings, domain experts confirmed the merits of RIT plots in comparison with the icicle tree and sunburst tree. With the mathematical assurance of the RIT plots, one can now compare different visual patterns without

having to worry about the inconsistent size-encoding or mingled nodes. This can reduce the need for one to bring up numerical data after observing an interesting visual pattern. One domain expert, specialized in machine learning, also identified the uses of such visual comparison in other scenarios (e.g., comparing data collected in different regions, seasons, or times of day), and would like to see the design to be deployed in a software system for supporting the modeling of movement data using machine learning.

The evaluation meetings also resulted in a few suggestions for improvement, such as color mapping. The plots shown in Figure 7.8 are improved versions of the original plots discussed in two evaluation meetings. One domain expert, who is specialized in visualization but was not involved in the mathematical reasoning and algorithmic design, asked a series of technical questions as part of the evaluation. Some questions served as "rigorous conceptual tests" of the RIT design and algorithm, while other questions led to more in-depth discussions on several design decisions, including two questions in Section 7.7.

7.7 Discussions

In this section, we discuss several questions about a few minor design decisions, which we encountered during our design, implementation, testing, and evaluation processes. In particular, the last question was posed by reviewers of this work.

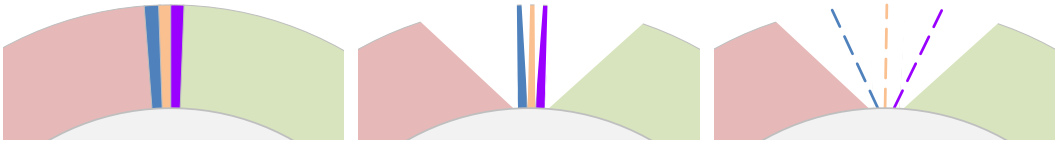
Setting a Shared Double-Wedge Angle? In our algorithm, the wedge to be removed has an angle $\frac{1}{2}\alpha$ that is proportional to the angle β of the corresponding annular sector. When the two neighboring nodes are of very different sizes, i.e., with very different β angles, the two removed wedges form a noticeably-asymmetric shape between the two sectors. This led to the consideration of a possible design option for removing two wedges of the same angle.

It was quickly realized that this "shared double-wedge angle" would have to be set according to the smaller neighboring node for each pair of sibling nodes. For a thin node (i.e., a very small β), the maximum rule discussed in Section 7.3.3 (i.e.,

Eq. 7.7) would ensure a very small α. On balance, we considered that it is more important to have sufficient gaps near thin nodes than symmetry, as the former affects perception, while the latter affects aesthetics. We, therefore, selected the option of defining the double-wedge angle α proportional to the annular sector angle β in our implementation.

Optimizing the Height of an Annular Sector? We also considered other options for determining the wedge angle and the size of the top-up sector. In Section 7.3, we outlined one option, with which the double wedge angle α is determined first, and the attributes of the top-up sector (i.e., sector angle $(\beta - \alpha)$ and height h_{topup}) are calculated accordingly. Mathematically, it is possible to determine h_{topup}, and then calculate α. However, since the algorithm would have to validate α according to the maximum rule, any adjustment to α would lead to a change of h_{topup}, causing a looped calculation step.

Another option is to have an adjustable height h for each annular sector, i.e., to adjust h (instead of adding a top-up) after removing the two wedges. Aesthetically, the advantage is to have a straight line for the removed area. One would need to determine α and h together, using an optimization technique, according to a set of predefined constraints for the size of the two wedges and the overall node size (i.e., the uncut annular sector − two wedges). As an optimization process may lead to noticeable computational cost and implementation complexity, we hope that some future work may explore this option further.



(a) three thin nodes (b) after wedge removal (c) space relaxation

Figure 7.9: One practical solution for dealing with small gaps between thin nodes (i.e., with very small data values) is to relax the mathematically-defined gaps by redistributing the thin nodes in a large space that contains some large gaps. Meanwhile, it is necessary to modify the visual representations of these redistributed nodes to indicate the uncertainty of the encoded sizes.

Dealing with Multiple Thin Nodes? As discussed previously, it is necessary to set α (double wedge angle) as a portion of β (annular sector angle). However, the gaps between nodes can sometimes be rather small if two or more thin nodes are positioned next to each other, as illustrated in Figure 7.9. If an RIT plot is rendered into a scalable vector-graphics representation (e.g., SVG), the gaps will always be visible when one zooms into a part of an RIT plot containing thin nodes.

In many situations, an RIT plot may be rendered into a raster image, or users do not need to figure out the sizes of these thin nodes precisely, though they do need to perceive the presence of these thin nodes. This may not strictly be a mathematical problem, but rather a common practical challenge that many different visual representations would face, including icicle trees and sunburst trees.

One practical solution is to relax the mathematically-defined gaps by redistributing each set of *tightly-packed nodes* (TPNs) in a large space that contains some large gaps. This can be done using the following algorithmic steps:

1. Identify each set of neighboring child nodes with small data values (i.e., less than a pre-defined threshold) as a TPN set, label these nodes in their records as nodes not to be displayed normally, and create a group identity for each TPN set such that the nodes in a TPN set can be processed together.

2. For each TPN set, identify the total space available for redistributing the nodes in the TPN set. As illustrated in Figure 7.9(b), typically a TPN set may have two neighboring nodes that are not TPNs and have left larger gaps. The total space thus includes these gaps as well as the space that would be occupied by the TPN set. In the case of Figure 7.9(b), it is the space between the pale red node and the pale green node.

   When a TPN set includes the first or last child node of a parent node, at least one side of a TPN set does not border a large gap left by a sibling node. In such a case, we can use the wedge angle ($\frac{1}{2}\alpha$) of the parent node, as we can infer that there is always a gap between the parent node and the parent's siblings.

3. Redistribute the nodes in the TPN set evenly in the available space as illustrated in Figure 7.9(c).

Meanwhile, it is necessary to modify the visual representations of these redistributed TPNs to indicate the uncertainty of the encoded sizes. We recommend using dashed lines, which are visually different from the ordinary nodes in an RIT, do not consume much space, and convey size uncertainty.

How does the algorithm scale? An analysis of the RIT drawing algorithm in Section 7.4 indicates that all nodes are visited three times (Algorithm 4). This suggests that the runtime complexity of the algorithm is $O(N_v)$, where N_v is the total number of nodes. In other words, the algorithm is linearly scalable in relation to the number of nodes. Figure 7.10 shows the results of a scalability test that evidences the scalability analysis. More results of the test are given in Appendix .5.

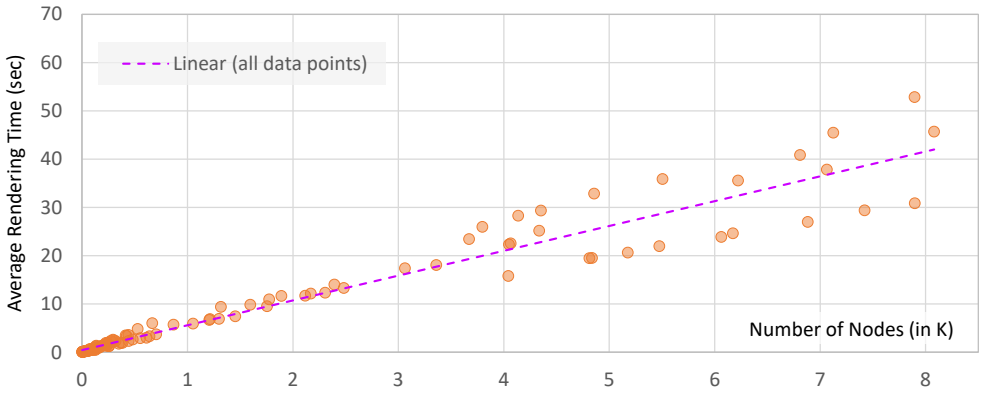


Figure 7.10: The results of a scalability test evidence that the RIT algorithm is linearly scalable in relation to the total number of nodes in a tree N_v.

7.8 Conclusion

In this Chapter, we have presented a new visual design, *Radial Icicle Tree* (RIT), which addresses three issues identified in Section 7.1 and illustrated in Figure 7.1. Using the design process reported in Section 7.2, we were able to explore and analyze a number of design options systematically and identify the most promising design to focus on in the following mathematical and algorithmic considerations. Through our mathematical reasoning, we were able to confirm the desired mathematical

properties of the design while working out a computational means for ensuring node separation and area constancy. We have presented a recursive algorithm for plotting RITs and validated the algorithm through an implementation in Python. We have tested the proposed visual design, its algorithm, and its implementation with synthetic data featuring all three issues concerned, open data in the public domain, and movement data that is being studied in Statistics Netherlands. The domain experts confirmed the merits of RIT in comparison with other visual designs that are currently being developed as part of a software system for supporting their analysis of movement data.

Through the processes of visual design, mathematical reasoning, algorithm design, implementation, and testing, we have ascertained that RIT does address three issues without incurring any relative demerit. Through our discussion in Section 7.7, we have identified a few limitations, which RIT has inherited from icicle and sunburst trees, but offered potential opportunities for making further improvements through research and development. We have made our Python-based implementation available as open-source software at `https://github.com/MattJin19/Radial-Icicle-Tree`. We also plan to develop an open-source implementation in D3.js.

8 Conclusion and Future Work

8.1 Summary of Contributions

In this thesis, I developed methods to improve ML model performance by enhancing training data quality, with a central focus on addressing inconsistencies across different data spaces. A core contribution is the proposal of a visualization-supported ML workflow, the Four Views Design, which introduces synthetic data generation and data space analysis as explicit and iterative components of the ML development process. This workflow goes beyond conventional experiment-based optimization and makes ML development a more visualization-supported process. Building on this workflow, I presented three representative systems, iGAiVA, TA3, and DS4ML, which demonstrate the workflow's effectiveness across text, image, and time-series domains. I also introduced a novel visualization technique, the Radial Icicle Tree (RIT), which extends the design space of hierarchical data visualization and supports the analysis of state-change structures in time-series data.

The following parts relate the contributions to the five research objectives (O1–O5) discussed in Section 1.2, showing how each research objective has been addressed through the systems and methods developed in this thesis.

In Chapter 3, I introduced the "Four Views Design", which integrates ML model testing, analysis, and training data improvement through four complementary

visualization perspectives. The workflow is operationalized through Dataset View, Model View, Results View, and Synthesis View, which provide user interfaces for data space exploration, model quality evaluation, model structure visualization, and data synthesis optimization. Compared with traditional ML workflows, the Four Views Design workflow enables the identification and interpretation of inconsistencies across different data spaces and provides actionable pathways for subsequent improvements. This chapter addressed research objective **O1**.

Importantly, the Four Views Design is not just a modular reorganization of existing practices, but a new workflow that elevates data quality from a preprocessing concern to a continuous optimization target throughout the ML model development process. By embedding synthetic data generation as a dedicated workflow stage and linking it tightly with visualization and iterative feedback, the workflow makes training data quality as tunable and improvable as model parameters in the ML development. It provides a potential methodological solution for a new generation of VIS4ML workflows.

Building upon this conceptual workflow, I presented iGAiVA in Chapter 4, a system for text classification that integrates generative AI with visual analytics to address training–deployment data space inconsistencies in real-world classification scenarios in a ticketing system. iGAiVA introduced new visualization methods, such as Radial Basis Function (RBF) heatmaps and Tag-Treemaps, which enable ML model practitioners to identify sparse or error-prone subregions of the training data space and to guide the use of large language models in generating synthetic text samples. By embedding targeted data synthesis into an interactive workflow, iGAiVA empowers ML users to iteratively improve training datasets and retrain models, leading to measurable improvements for underrepresented classes.

Beyond methodological innovation, iGAiVA was evaluated in collaboration with Inetum Spain, where it was deployed in an enterprise environment for practical issues. The user evaluation confirmed that visualization-assisted data synthesis reduced experiment workload, improved the ticketing system for rare classes, and provided a way to maintain model performance under dynamic changes in ticketing categories.

This work demonstrates that VIS4ML methods can transition from academic prototypes to practical industrial tools, contributing to both research objectives **O2** and **O4**, while also highlighting the industrial relevance of human–computer collaboration in ML workflows. This work also inspired other researchers to conduct subsequent research on AutoML, demonstrating its research value and potential.

I presented TA3 in Chapter 5, a system designed to test and analyze model robustness under adversarial attacks. TA3 operationalizes the Four Views Design for adversarial testing tasks, providing interactive facilities for human-in-the-loop attack simulation, visualization-assisted parameter steering, and model vulnerability analysis. The system introduced robustness metrics and visualization techniques to analyze adversarial sample generation and model decision processes, thereby identifying vulnerabilities that traditional accuracy or F1-based evaluations cannot reveal.

More broadly, TA3 advances adversarial ML research by demonstrating the value of interactive, visualization-supported workflows for testing model robustness. Existing adversarial testing workflows often treat testing as a black-box process, limited to generating successful adversarial examples. TA3 shows that by bringing human experts into the loop and providing visualization tools for exploring parameter spaces, target data selection, and model structure responses, ML researchers can uncover model weaknesses. TA3 demonstrates how visualization and human reasoning can complement algorithmic testing while extending the applicability of the proposed workflow to adversarial ML scenarios. These contributions address research objective **O3**, by diagnosing inconsistencies between application-specific and adversarial data spaces, and also contribute to research objective **O4**, by showing the Four Views Design in the domain of adversarial testing.

I presented DS4ML in Chapter 6, which provides interactive methods for generating, comparing, and integrating synthetic time-series datasets extending the Four Views Design workflow to time-series classification. This system was applied in collaboration with Statistics Netherlands to address inconsistencies between training and deployment data space for human activity recognition. DS4ML enables ML developers to evaluate the effectiveness of different data augmentation strategies

and to make informed decisions about which synthetic methods best improve model robustness in deployment scenarios.

Another important contribution of DS4ML is its capability to support comparative analysis of synthetic datasets, an aspect that has been overlooked in prior research. By embedding data synthesis into a visual, human-in-the-loop workflow, DS4ML enables ML practitioners not only to generate synthetic data and to evaluate its quality, but also to select appropriate samples for training. This capability is important in real-world applications, where synthetic samples may be produced while only a subset proves useful for model training. By addressing this challenge, DS4ML contributes to both **O2** and **O4**, and demonstrates how visualization can act as an enabler for data enrichment tasks.

The three systems together, iGAiVA for text, TA3 for adversarial image testing, and DS4ML for time-series classification, demonstrate the effectiveness of the Four Views Design workflow. Their combined contributions show that a visualization-supported workflow can improve ML model performance across diverse modalities, application domains, and types of data space inconsistency. This addresses research objective **O4** and establishes the Four Views Design as a practical VIS4ML workflow.

In addition to the workflow and systems, I presented the Radial Icicle Tree (RIT) in Chapter 7, a novel visualization method that addresses limitations of existing hierarchical visualization designs such as icicle trees and sunburst plots. RIT resolves two problems in the previous visualization: inconsistent area encoding and poor node separation. Its mathematically grounded design ensures area constancy while enhancing node readability. Applied to time-series state change analysis, RIT allowed clearer comparisons between training and deployment datasets, providing a new means for detecting structural differences in real-world data. This methodological advance represents a contribution to visualization research and supports research objective **O5**.

The RIT contribution demonstrates that advances in visualization design can directly support VIS4ML workflows. The contributions, both as a theoretical

extension of tree visualization design and as a practical analytical tool, illustrate the broader impact of visualization research when integrated into ML workflows.

Taken together, these contributions advance the understanding and application of visualization in human–machine collaboration for machine learning. This thesis provides a new perspective on addressing data space inconsistencies as a challenge in ML development, introduces an innovative workflow (Four Views Design) for embedding visualization and synthetic data generation into ML workflows, develops three systems that operationalize the workflow across multiple domains, and contributes a new visualization method that extends the design space of hierarchical data analysis.

In the current state-of-the-art (SOTA) landscape of machine learning, including large language models, data-centric AI, and automated machine learning pipelines, many approaches emphasize increasing model capacity, architectural innovation, and large-scale pretraining. In contrast, the work presented in this thesis highlights data space inconsistency, distribution alignment, and human-in-the-loop interpretability as critical factors for reliable real-world deployment. By demonstrating that identification and repair of different data space inconsistencies can yield measurable performance improvements, this thesis complements SOTA model-centric approaches with a human-centric and workflow-oriented perspective.

Besides technical innovations, these contributions demonstrate practical and industrial relevance through collaborations with Inetum Spain and Statistics Netherlands, where visualization-based solutions were shown to improve ML reliability in real deployments. The case studies illustrate how the proposed workflow can be instantiated across different data types and task domains, providing concrete examples of its effectiveness. These examples confirm that visualization and human–computer interaction can play an active role in interpreting ML models and in improving training data quality and model robustness.

The thesis also advances the broader understanding and application of visualization techniques in human–computer collaboration and machine learning. By introducing the Four Views Design as an advanced VIS4ML workflow and by

exploring interactive visualization tools to support it, this thesis offers innovative solutions for model testing, inconsistency analysis, and data enrichment. These contributions provide both theoretical and practical solutions for building more reliable ML systems, while opening new research directions for the VIS4ML community and offering a promising alternative to other optimization approaches.

8.2 Future Work

This thesis addresses the problem of inconsistencies across different data spaces through a set of research contributions, ranging from the proposal of a new VIS4ML workflow and the implementation of system prototypes to the innovation of visualization techniques. The Four Views Design workflow and its applications in three representative systems demonstrate that the visualization can effectively support ML workflows to improve model performance and reliability. The research contributions presented here open several promising ways for future investigation that can advance the field.

One research direction focuses on addressing the challenge of efficiently filtering and selecting high-quality synthetic data from large-scale generated datasets. In practice, practitioners often face the issue of producing thousands or even millions of synthetic samples, making manual quality assessment infeasible. Although there has been some exploration in the area of AutoML, developing systematic methods that combine interactive visualization to quickly identify and choose synthetic data samples most suitable for specific ML tasks remains an important direction for future work.

This research project would develop a system combining large-scale data visualization, intelligent filtering algorithms, and interactive selection tools through several components. The first component involves creating visualization techniques designed for exploring massive synthetic samples, including overview visualizations that can handle a lot of data points, visualization methods for detailed inspection, and comparative visualizations that highlight differences between synthetic and real data distributions. These visualizations would need to address the challenges

of large-scale synthetic data exploration, enabling ML users to quickly identify patterns, outliers, and quality variations across extensive generated datasets.

The project would also include the development of intelligent filtering algorithms that can automatically identify promising synthetic data candidates based on multiple quality criteria, including statistical similarity to target distributions or diversity measures. This involves creating a rapid evaluation method for synthetic samples without requiring expensive model training. The filtering system would provide multiple filtering strategies, allowing users to apply different criteria depending on their specific requirements and constraints.

Another innovation would be the design of an interactive selection user interface that enables users to efficiently explore filtered results and make informed decisions about which synthetic samples to include in their training datasets. This system would integrate the Four Views Design approach with large-scale data selection, providing coordinated visualizations that allow users to understand the synthetic data, assess coverage of different data regions, and identify potential gaps or biases in their selected samples. Users would be able to refine their selection criteria iteratively, seeing the impact of different filtering parameters on the resulting dataset composition.

Expected contributions include a system for large-scale synthetic data exploration and filtering, novel visualization techniques for massive synthetic dataset analysis, intelligent filtering algorithms that can rapidly identify high-quality synthetic samples, and interactive selection tools that enable efficient synthetic data selection. This project would establish visual analytics for synthetic data selection as a new research area, providing both theoretical and practical tools that enable ML practitioners to manage and utilize large-scale synthetic data generation in real-world ML applications.

The second major research direction addresses the challenge of understanding and optimizing human behavior patterns in ML workflows, focusing on discovering user habits and providing intelligent recommendations to improve collaboration effectiveness. While this thesis demonstrates effective human-in-the-loop workflows,

there remains a gap in understanding how different users approach ML tasks and how systems can learn from successful user strategies to provide personalized guidance. There are good opportunities for the ML models to learn from user behavior patterns and provide proactive recommendations that can improve both efficiency and outcomes.

This project would develop a novel framework combining behavioral pattern mining, machine learning, and intelligent recommendation systems through several integrated research components. The first component involves developing algorithms to automatically discover recurring patterns in human behavior during ML workflows, including data exploration sequences, parameter adjustment strategies, and decision-making patterns. This would involve extending existing sequence mining and pattern discovery techniques to handle the multi-dimensional nature of human-AI interactions, creating new methods that can identify meaningful behavioral patterns across different users, tasks, and domains.

The framework would include the design and implementation of novel visualization techniques that help users understand their own working patterns and see how their approaches compare to successful strategies used by others. This would extend the Four Views Design approach to include temporal and behavioral dimensions, enabling users to visualize their workflow patterns, identify potential inefficiencies, and discover alternative approaches that might be more effective. These visualizations would need to present complex behavioral data in intuitive ways that help users reflect on and improve their own practices.

Other innovations would include the development of an intelligent recommendation system that learns from successful user behaviors to provide personalized suggestions during ML workflows. This system would analyze patterns from high-performing users and tasks to identify best practices, then provide contextually appropriate recommendations to other users based on their current situation, expertise level, and working style. The recommendation system would operate at multiple levels, from suggesting specific visualization combinations to recommending

overall workflow strategies, and would continuously adapt based on user feedback and outcome effectiveness.

The project would also focus on developing methods to capture and model user expertise and preferences, creating detailed user profiles that enable the system to provide increasingly personalized recommendations over time. This involves developing new approaches to implicit feedback collection, expertise level assessment, and preference modeling that can operate within existing ML workflows without disrupting user activities.

Expected contributions include a framework for behavioral pattern analysis in human-AI collaborative ML workflows, novel visualization techniques for behavioral pattern discovery and comparison, intelligent recommendation algorithms that can provide contextually appropriate guidance, and an open dataset of anonymized user behavior traces with performance annotations. This project would establish behavioral analytics for ML workflows as a new research area, providing both theoretical understanding and practical tools for designing more effective human-AI collaborative systems that can learn from and guide human behavior in ML tasks.

Another potential research direction focuses on leveraging successful adversarial attack samples to improve model robustness. The analysis of successful attacks can reveal vulnerable regions in the training data space that are not well captured during development. Building on the TA3 system and the Four Views Design workflow, future research could extend the visualization and analysis of adversarial testing results to find the successful attack patterns. These insights can then guide the synthesis or targeted selection of new training data to strengthen weak regions of the training data space. It will provide a potential pathway for repairing vulnerabilities exposed by adversarial attacks and enhance the robustness of ML models that refine existing algorithmic defenses.

The long-term vision is to combine methods for synthetic data with intelligent systems that can learn from and support human behavior, enabling future VIS4ML systems to better tackle both the challenges of different data space inconsistencies and the complexities of human decision-making in ML development. This vision

points to a new collaborative ML system that is not only more resilient to distribution shifts but also more responsive to user expertise. In this way, the research agenda outlined in this thesis connects the future directions of synthetic data exploration and behavioral pattern analysis with its broader conclusions, reinforcing the goal of building reliable and efficient ML systems for diverse real-world scenarios.

Appendices

APPENDICES OF

Chapter 4: iGAiVA: Integrated Generative AI and Visual Analytics in a Machine Learning Workflow for Text Classification

The main body of the Chapter 4 is accompanied by several appendices:

.1 Further testing results obtained in a multi-iteration workflow supported by iGAiVA.

.2 Further reporting on user evaluation, while Chapter 4 provides a relatively brief summary.

.3 This appendix provides further discussions on several topics like scalability, potential challenges with LLMs, and limitations.

.4 This appendix provides high-resolution images, as the same images were included in the main body of the Chapter as smaller images.

.1 iGAiVA Further Testing Results Obtained in a Multi-Iteration Workflow

Table 1: Further results from the first iteration: retraining with different class-based synthetic data. The Δ-recall results were compared with the original model M0 (Major Column 2).

Topic class	Original Data test	train	recall	T11-s2 train	Δ-recall	T12-s2 train	Δ-recall	T13-s2 train	Δ-recall	T14-s2 train	Δ-recall	T15-s2 train	Δ-recall
Overall	782	31280	0.821		0.000		▲0.008		0.000		▼0.002		▲0.001
T1	1748	6781	0.976		▼0.001		▲0.003		▲0.003		▲0.003		▲0.004
T2	2259	9091	0.943		▲0.004		▲0.002		▼0.002		▼0.010		▼0.008
T3	927	3792	0.892		▼0.037		▼0.024		▼0.005		▼0.014		▼0.001
T4	281	1106	0.769		▲0.042		▲0.046		0.000		▼0.015		▲0.035
T5	533	2222	0.732		▲0.017		▲0.028		▼0.023		▲0.015		▲0.002
T6	373	1515	0.528		▲0.022		▲0.067		▲0.008		▲0.040		▲0.008
T7	395	1568	0.714		▲0.030		▲0.033		▲0.013		▼0.010		▲0.015
T8	236	792	0.508		▼0.055		▼0.012		▲0.034		▼0.008		▼0.072
T9	278	1188	0.626		▲0.025		▲0.050		▼0.014		▼0.061		▲0.039
T10	335	1364	0.427		▼0.030		▼0.009		▲0.021		▲0.027		▲0.009
T11	94	377	0.926	+110	▲0.021		0.000		0.000		▲0.021		▲0.010
T12	64	294	0.375		▼0.016	+325	▲0.125		▲0.016		▼0.016		▼0.047
T13	45	135	0.178		▲0.066		▲0.044	+525	▲0.022		▲0.066		▲0.022
T14	135	629	0.526		▼0.015		▼0.022		▼0.030	+1265	▲0.111		▼0.022
T15	117	426	0.376		▼0.026		▼0.026		▼0.026		▼0.043	+1485	▲0.086

In Section 4.4, we explained the VIS-assisted processes for testing different synthetic data in order to improve an existing text classification model M0. Table 4.2 provides some of the testing results. In this appendix, we provide more testing results, including those produced by further iterations in the workflow.

Iteration 1:

- **T11-s1**: Results in Table 4.2, Major Column 3.

  → T11 recall: +0.021, overall recall: +0.003.

- **T12-s1**: Results in Table 4.2, Major Column 4.

  → T12 recall: +0.031, overall recall: +0.009.

- **T13-s1**: Results in Table 4.2, Major Column 5.

  → T13 recall: +0.133, overall recall: +0.009.

- **T14-s1**: Results in Table 4.2, Major Column 6.

  → T14 recall: +0.148, overall recall: +0.001.

Table 2: Results of the second iteration: retraining with different class-based synthetic data in addition to **T11-s1** synthetic data. The Δ-recall results were compared with the original model M0 (Major Column 2), while the M1a results are given in Major Column 3.

Topic class	Original Data test	train	recall	T11-s1 train	Δ-recall	T11-s1+T12-s1 train	Δ-recall	T11-s1+T12-s2 train	Δ-recall	T11-s1+T13-s1 train	Δ-recall	T11-s1+T13-s2 train	Δ-recall
Overall	7820	31280	0.821		▲0.003		▲0.004		▼0.003		▼0.004		▼0.003
T1	1748	6781	0.976		▼0.002		▼0.006		▼0.012		▲0.003		0.000
T2	2259	9091	0.943		▲0.002		▲0.004		▼0.002		▼0.009		▼0.005
T3	927	3792	0.892		▼0.009		▼0.006		▼0.014		▼0.025		▼0.012
T4	281	1106	0.769		▲0.039		▲0.032		▲0.014		▲0.025		▲0.017
T5	533	2222	0.732		▲0.020		▲0.005		▼0.023		▲0.013		▼0.002
T6	373	1515	0.528		▲0.030		▲0.024		▲0.046		▲0.011		▲0.022
T7	395	1568	0.714		▲0.013		▲0.033		▲0.025		▼0.003		▲0.015
T8	236	792	0.508		▼0.055		▲0.056		▲0.047		▼0.029		▼0.059
T9	278	1188	0.626		▲0.011		▲0.007		▼0.047		▲0.007		▼0.011
T10	335	1364	0.427		▲0.006		▼0.012		▲0.018		▲0.036		▲0.027
T11	94	377	0.926	+110	▲0.021	+110	▲0.031	+110	▲0.021	+110	▲0.010	+110	▲0.010
T12	64	294	0.375		0.000	+615	▲0.047	+615	▼0.016		▼0.063		▼0.078
T13	45	135	0.178		▲0.155		▲0.066		▼0.022	+585	▲0.089	+585	▲0.111
T14	135	629	0.526		▼0.037		▼0.104		▼0.030		▼0.045		▼0.007
T15	117	426	0.376		▼0.034		▼0.008		▼0.026		▼0.077		▼0.077

- **T15-s1**: Results in Table 4.2, Major Column 7.

  → T15 recall: +0.094, overall recall: +0.004.

- **T11-s2**: Results in Table 1, Major Column 3.

  → T11 recall: +0.021, overall recall: ∼0.000.

- **T12-s2**: Results in Table 1, Major Column 4.

  → T12 recall: +0.125, overall recall: +0.008.

- **T13-s2**: Results in Table 1, Major Column 5.

  → T13 recall: +0.022, overall recall: ∼0.000.

- **T14-s2**: Results in Table 1, Major Column 6.

  → T14 recall: +0.111, overall recall: −0.002.

- **T15-s2**: Results in Table 1, Major Column 7.

  → T15 recall: +0.086, overall recall: +0.001.

Iteration 2A (based on T11-s1):

- **T11-s1 + T12-s1**: Results in Table 2, Major Column 4.

  → T12 recall: +0.047, overall recall: +0.004.

  → Comparing with M1a (T11-s1) overall recall: +0.001.

- **T11-s1 + T12-s2**: Results in Table 2, Major Column 5.
  → T12 recall: -0.016, overall recall: -0.003.
  → Comparing with M1a (T11-s1) overall recall: -0.006.

- **T11-s1 + T13-s1**: Results in Table 2, Major Column 6.
  → T13 recall: $+0.089$, overall recall: -0.004.
  → Comparing with M1a (T11-s1) overall recall: -0.007.

- **T11-s1 + T13-s2**: Results in Table 2, Major Column 7.
  → T13 recall: $+0.111$, overall recall: -0.003.
  → Comparing with M1a (T11-s1) overall recall: -0.006.

Iteration 2B (based on T13-s1):

- **T13-s1 + T11-s1**: Results in Table 3, Major Column 4.
  → T11 recall: $+0.021$, overall recall: ~ 0.000.
  → Comparing with M1b (T13-s1) overall recall: -0.009.

- **T13-s1 + T11-s2**: Results in Table 3, Major Column 5.
  → T11 recall: $+0.021$, overall recall: $+0.004$.
  → Comparing with M1b (T13-s1) overall recall: -0.005.

- **T13-s1 + T12-s1**: Results in Table 3, Major Column 6.
  → T12 recall: $+0.031$, overall recall: $+0.004$.
  → Comparing with M1b (T13-s1) overall recall: -0.005.

- **T13-s1 + T12-s2**: Results in Table 3, Major Column 7.
  → T12 recall: ~ 0.000, overall recall: ~ 0.000.
  → Comparing with M1b (T13-s1) overall recall: -0.009.

Iteration 3 (based on T11-s1 + T12-s1):

- **T11-s1 + T12-s1 + T13-s1**: Results in Table 4, Major Column 3.
  → T13 recall: $+0.200$, overall recall: ~ 0.000
  → Comparing with M2 (**T11-s1**+**T12-s1**) overall recall: -0.004.

Table 3: Results of the second iteration: retraining with different class-based synthetic data in addition to **T13-s1** synthetic data. The Δ-recall results were compared with the original model M0 (Major Column 2), while the M1b results are given in Major Column 3.

Topic class	Original Data			T13-s1		T13-s1+T11-s1		T13-s1+T11-s2		T13-s1+T12-s1		T13-s1+T12-s2	
	test	train	recall	train	Δ-recall	train	Δ-recall	train	Δ-recall	train	Δ-recall	train	Δ-recall
Overall	7820	31280	0.821		▲0.009		0.000		▲0.004		▲0.004		0.000
T1	1748	6781	0.976		▼0.001		▲0.006		▼0.007		▲0.003		▲0.001
T2	2259	9091	0.943		▼0.003		▲0.002		▲0.005		▼0.006		▼0.011
T3	927	3792	0.892		▼0.004		▼0.013		▼0.004		▼0.003		▼0.011
T4	281	1106	0.769		▲0.028		▲0.039		▲0.053		▲0.039		▲0.025
T5	533	2222	0.732		▲0.009		▼0.008		▲0.002		▲0.032		▲0.020
T6	373	1515	0.528		▲0.038		▲0.011		▲0.003		▲0.032		▲0.003
T7	395	1568	0.714		▲0.025		▲0.033		▲0.081		▼0.028		▲0.008
T8	236	792	0.508		▲0.077		▼0.084		▼0.025		▼0.004		▼0.004
T9	278	1188	0.626		▼0.018		▲0.018		▼0.004		▼0.018		▼0.014
T10	335	1364	0.427		▲0.095		▼0.018		▼0.006		▲0.074		▲0.066
T11	94	377	0.926		0.000	+120	▲0.021	+120	▲0.021		▲0.010		▲0.010
T12	64	294	0.375		▼0.016		0.000		0.000	+685	▲0.031	+685	0.000
T13	45	135	0.178	+525	▲0.133	+525	▲0.044	+525	▲0.089	+525	▲0.133	+525	▲0.178
T14	135	629	0.526		▼0.022		▼0.022		▼0.045		▼0.022		0.000
T15	117	426	0.376		▼0.034		▼0.034		▼0.043		▼0.051		▼0.094

- **T11-s1 + T12-s1 + T13-s2**: Results in Table 4, Major Column 4.

  → T13 recall: +0.155, overall recall: −0.007

  → Comparing with M2 (**T11-s1**+**T12-s1**) overall recall: −0.011.

- **T11-s1 + T12-s1 + T14-s1**: Results in Table 4, Major Column 5.

  → T14 recall: +0.030, overall recall: −0.004

  → Comparing with M2 (**T11-s1**+**T12-s1**) overall recall: −0.008.

- **T11-s1 + T12-s1 + T14-s2**: Results in Table 4, Major Column 6. → T14 recall: +0.009, overall recall: −0.008

  → Comparing with M2 (**T11-s1**+**T12-s1**) overall recall: −0.012.

After three iterations, we can see that if one wishes to improve only the overall recall, in the first iteration, **T12-s1** and **T13-s1** managed to deliver an improvement of the overall recall by 0.9%. This is followed by **T12-s2** that delivered 0.8% improvement of the overall recall.

If one wishes to improve some poor-performing classes, we may consider:

- For **T12**:

  T11-s1 + **T12-s1** + **T13-s1** → +9.4%,

Table 4: Results of the third iteration: retraining with different class-based synthetic data in addition to **T11-s1** and **T12-s1** synthetic data. The Δ-recall results were compared with the original model M0 (Major Column 2).

Topic class	Original Data test	train	recall	T11-T12-s1+T13-s1 train	Δ-recall	T11-T12-s1+T13-s2 train	Δ-recall	T11-T12-s1+T14-s1 train	Δ-recall	T11-T12-s1+T14-s2 train	Δ-recall
Overall	782	031280	0.821		0.000		▼0.007		▼0.004		▼0.008
T1	1748	6781	0.976		▼0.001		▲0.005		▲0.003		▼0.015
T2	2259	9091	0.943		▼0.005		▼0.011		▼0.008		▼0.006
T3	927	3792	0.892		▼0.009		▼0.023		▼0.015		▼0.018
T4	281	1106	0.769		▲0.039		▲0.032		▲0.028		▲0.039
T5	533	2222	0.732		▲0.020		▼0.006		▲0.030		▲0.013
T6	373	1515	0.528		0.000		▼0.008		▲0.040		0.000
T7	395	1568	0.714		▼0.008		▲0.010		▼0.015		▼0.005
T8	236	792	0.508		▼0.042		▼0.105		▼0.127		▼0.131
T9	278	1188	0.626		▲0.011		▲0.011		▼0.014		▼0.047
T10	335	1364	0.427		▲0.009		▲0.024		▼0.003		▲0.033
T11	94	377	0.926	+110	▲0.010	+110	▲0.010	+110	▲0.021	+110	▲0.010
T12	64	294	0.375	+615	▲0.094	+615	0.000	+615	0.000	+615	▼0.031
T13	45	135	0.178	+550	▲0.200	+550	▲0.155		▼0.045		▼0.022
T14	135	629	0.526		▼0.037		▼0.074	+595	▲0.030	+595	▲0.089
T15	117	426	0.376		▼0.034		▼0.077		▼0.051		▼0.008

Table 5: Synthetic data generated from random samples of T11, T12, T13, T14, and T15 class. The Δ-recall results were compared with the original model M0 (Major Column 2).

Topic class	Original Data test	train	recall	T11__random__s1 train	Δ-recall	T12__random__s1 train	Δ-recall	T13__random__s1 train	Δ-recall	T14__random__s1 train	Δ-recall	T15__random__s1 train	Δ-recall
Overall	782	031280	0.821		▼0.003		▲0.002		▲0.002		▼0.001		▼0.001
T1	1748	6781	0.976		▼0.005		0.000		▲0.003		▼0.011		▼0.010
T2	2259	9091	0.943		▼0.002		▼0.001		0.000		▼0.002		▲0.002
T3	927	3792	0.892		▼0.019		▼0.010		▲0.001		▼0.018		▼0.023
T4	281	1106	0.769		▲0.042		▲0.035		▲0.039		▲0.003		▲0.032
T5	533	2222	0.732		▼0.030		0.000		▼0.030		▲0.003		▼0.023
T6	373	1515	0.528		▲0.003		▲0.024		▲0.040		▲0.040		▲0.032
T7	395	1568	0.714		▲0.020		▲0.020		▼0.015		▲0.020		▲0.058
T8	236	792	0.508		▼0.025		▼0.012		▼0.012		▼0.008		▼0.055
T9	278	1188	0.626		0.000		▼0.040		▲0.007		▼0.025		▼0.007
T10	335	1364	0.427		▲0.042		▲0.066		▲0.042		▲0.051		▲0.036
T11	94	377	0.926	+250	▲0.031		0.000		0.000		0.000		0.000
T12	64	294	0.375		▼0.031	+250	▼0.016		0.000		0.000		▼0.031
T13	45	135	0.178		▲0.044		▲0.155	+250	▲0.066		▲0.133		▲0.044
T14	135	629	0.526		▼0.037		▼0.007		▼0.045	+250	▲0.030		▼0.015
T15	117	426	0.376		▼0.026		▼0.077		▼0.017		▼0.094	+250	▲0.034

T11-s1 + **T12-s1** → +3.1%,

T13-s1 + **T12-s1** → +3.1%

- For **T13**:

  T11-s1 + **T12-s1** + **T13-s1** → +20.0%,

  T13-s1 + **T12-s2** → +17.8%,

  T11-s1 → +15.5%

  T11-s1 + **T12-s1** + **T13-s2** → +15.5%,

  T13-s1 → +13.3%,

  T13-s1 + **T12-s1** → +13.3%

Randomly-Selected Examples. We also performed tests to see how the VA-assisted process for selecting examples for LLMs to generate synthetic data has any advantage over a random process for selecting examples. Tables 5 and 6

Table 6: Synthetic data generated from random samples of T5, T8, T9, T14, and a combination of random samples of T11 and T12 class. The Δ-recall results were compared with the original model M0 (Major Column 2).

Topic class	Original Data test	train	recall	T5_random_s1 train	Δ-recall	T8_random_s1 train	Δ-recall	T9_random_s1 train	Δ-recall	T14_random_s2 train	Δ-recall	T11-T12_random_s1 train	Δ-recall
Overall	782	31280	0.821		▲0.001		0.000		▼0.001		▲0.003		▼0.001
T1	1748	6781	0.976		▲0.001		▼0.006		▼0.005		▲0.001		0.000
T2	2259	9091	0.943		▼0.005		▼0.004		▼0.003		▲0.003		▼0.005
T3	927	3792	0.892		▼0.021		▼0.006		▼0.012		▼0.010		▼0.004
T4	281	1106	0.769		▲0.039		▲0.035		▲0.025		▲0.053		▲0.028
T5	533	2222	0.732	+150	▲0.017		▲0.005		▼0.030		▼0.017		▼0.002
T6	373	1515	0.528		▼0.005		▼0.013		▲0.048		▼0.037		▲0.024
T7	395	1568	0.714		▲0.030		0.000		▲0.008		▼0.003		▼0.013
T8	236	792	0.508		▲0.026	+400	▲0.051		▼0.038		▲0.022		▼0.016
T9	278	1188	0.626		▲0.050		▲0.036	+175	▲0.079		▲0.021		▲0.047
T10	335	1364	0.427		▼0.006		▲0.015		▼0.021		▲0.063		▲0.027
T11	94	377	0.926		▲0.010		▲0.010		0.000		▲0.010	+100	▲0.010
T12	64	294	0.375		0.000		▼0.047		▲0.016		▼0.047	+100	▼0.031
T13	45	135	0.178		0.000		▲0.044		▲0.111		▲0.111		0.000
T14	135	629	0.526		▼0.045		▼0.030		▼0.007	+125	▼0.015		▼0.015
T15	117	426	0.376		▼0.068		▼0.043		▼0.068		▼0.017		▼0.043

show 10 of such examples.

The ten experiments are:

- **T11-random-s1**: Results in Table 5, Major Column 3.

  → T11 recall: +0.031, overall recall: −0.003.

- **T12-random-s1**: Results in Table 5, Major Column 4.

  → T12 recall: −0.016, overall recall: +0.002.

- **T13-random-s1**: Results in Table 5, Major Column 5.

  → T13 recall: +0.066, overall recall: +0.002.

- **T14-random-s1**: Results in Table 5, Major Column 6.

  → T14 recall: +0.030, overall recall: −0.001.

- **T15-random-s1**: Results in Table 5, Major Column 7.

  → T15 recall: +0.034, overall recall: −0.001.

- **T5-random-s1**: Results in Table 6, Major Column 3.

  → T5 recall: +0.017, overall recall: +0.001.

- **T8-random-s1**: Results in Table 6, Major Column 4.

  → T8 recall: +0.051, overall recall: ∼ 0.000.

- **T9-random-s1**: Results in Table 6, Major Column 5.

  → T9 recall: +0.079, overall recall: −0.001.

- **T14-random-s2**: Results in Table 6, Major Column 6.
  → T14 recall: −0.015, overall recall: +0.001.

- **T11+T12-random-s1**: Results in Table 6, Major Column 7.
  → T11 recall: +0.010, T12 recall: −0.031, overall recall: −0.001.

For the five experiments in Table 5, five sets of synthetic data were generated using LLMs for classes **T11**, **T12**, **T13**, **T14**. and **T15**. Each set was created by randomly selecting 50 unique messages as examples from a class and then generating 5 synthetic samples per example. This process resulted in 250 synthetic messages per class. These messages were added to the training data, and the model was retrained. The performance of the retrained model is given in the corresponding column.

For the first four experiments in Table 6, examples were randomly selected from classes **T5**, **T8**, **T9**, and **T14** respectively. In these cases, we used different numbers of examples, i.e., 30 for **T5**, 80 for **T8**, 35 for **T9**, and 25 for **T14**. Again, LLMs were used to generate 5 synthetic messages per example.

The last column in Table 6 is an experiment where 100 examples were randomly selected from **T11** and another 100 examples were randomly selected from **T12**. This can be considered as a two-iteration experiment, with **T11** first followed by **T12**, or **T12** first followed by **T11**.

The experiments reported in this appendix are a subset of all experiments conducted in this work. In general, randomly-selected examples can lead to performance improvement or decline in the range $[-0.3\%, +0.3\%]$ in terms of overall recall. On the other hand, the VA-assisted approach for selecting examples has led to performance improvement or decline in the range $[-0.7\%, +0.9\%]$ in terms of overall recall. As shown in Table 4.2 in the main body of the Chapter and tables in this appendix, +0.9% improvement was achieved multiple times. Meanwhile, $\leq -0.4\%$ decline all occurred in the third iteration (Table 4), when model improvement becomes difficult.

As shown in Tables 5 and 6, for individually-targeted classes, randomly-selected examples can lead to performance improvement or decline in the range [−3.1%,

+7.9%] in terms of recall. VA-assisted approach for selecting examples has led to performance improvement or decline in the range $[-3.1\%, +17.8\%]$ in terms of recall. Our experience showed that the users can usually make a reasonably good prediction about a positive outcome for the targeted class when selecting examples through the visualization plots, as shown in Section 4.4. The overall performance is less predictable, especially when the targeted area in a PCA-based scatter plot overlaps extensively with other classes (i.e., the gray dots behind).

.2 iGAiVA Further Reporting on User Evaluation

As described in Section 4.2, the three requirements were identified in multiple steps during my 2-month placement in the FabLab of Inetum, Spain. The experimental workflow in Figure 4.1, including the VIS and LLM solutions described in Sections 4.4 and 4.5, was developed and evaluated during the placement as there were daily contacts between me (who is specialized in both VIS and ML) and the ML researchers and developers in the FabLab. There were two formal meetings and six informal meetings during the 2-month period. There were up to 15 people in the formal meetings. I wrote two reports and gave two presentations on the VIS and LLM solutions in the experimental workflow. In addition, there were weekly online technical meetings, involving the university partner, who visited the Inetum in Spain, before the placement.

During the development of iGAiVA, the university partner and I had weekly design meetings to analyze the requirements and evaluate interim designs as well as the prototype under development. There were several online meetings, involving the company collaborator, who paid a 1-month visit to the university partner and me at Oxford. This was followed by my 1-month placement at Inetum, Spain, where the iGAiVA tool was evaluated by ML researchers and developers through several meetings (up to 15 people), five interviews, and many informal discussions.

Through these frequent engagements and the development of VIS and LLM solutions, the three requirements emerged gradually. These requirements were evaluated continually when iGAiVA was developed. Below we summarize the

major feedback on the technical work presented in Sections 4.4, 4.5, and 4.6, which correspond to the three requirements. All texts in *italic* are from the minutes of meetings and transcripts of conversations.

R1: Using more VIS techniques to help identify possible causes of errors. The initial evaluation was welcoming but uncertain about whether VIS is really useful. The uncertainty was largely removed after VIS was used to help the LLM-based data synthesis process.

▶ When t-SNE scatter plots and PCA scatter plots were first used to analyze the training data and testing results, the ML developers were interested but unsure about what information they could gain. They commented: *They* [(i.e., these plots)] *are nice. We saw similar pictures in some research papers and presentations. We have not used these in our workflows* [for developing ML models.] *We like these pictures. Hopefully, when you* [(i.e., me)] *are here, we can find out how useful they are.*

▶ On RBF and tag-treemap, *We have not seen these before in ML. Very interesting. It may take some time for us to learn to use them.*

▶ After some experimental results for the requirement **R2** became available, ML developers are convinced that VIS techniques can be used to solve problems. They commented: *We can now see that these plots can show where to choose examples for data generation. Seeing these pictures is a bit like seeing targets in sports.*

▶ *Being able to see different parts of a class with different behaviors, we can use a stacked model to help some classification tasks.*

▶ *Visualization can be used during the interaction between ML developers and the clients.*

▶ *We can use visualization to monitor a model's behavior in a dynamic environment as the whole dataset "refreshes" periodically.*

▶ *I can see two benefits for the company to receive from visualization methods: cost saving and understanding the data.*

R2: Using large language models (LLMs) to generate synthetic data for training and using VIS techniques to guide the data synthesis process. The combined use of VIS and LLM helped improve the ML model for the specific

ticketing system in a relatively shorter period and more controllable manner (in comparison with hyper-parameter tuning.) The focus of the evaluation was quickly shifted to the long-term impact of the approach on the business processes.

▶ *The current results* [for the dataset used in this work] *are very promising. We can see* [the ML models for] *other ticketing systems can also be improved in this way.*

▶ *Clients are sometimes only interested in certain categories* [(i.e., classes),] *it is acceptable when these certain categories improve* [their accuracy,] *while other categories lose some accuracy.*

▶ *From a business point of view, targeted data generation will be very useful. I can remember one case,* [where] *a client wanted to have a class with only a few messages* [(i.e., training and testing data)]. *After training, the model only achieved about 10% accuracy. The client explained the reason: they knew that the messages for this class would increase quickly.*

▶ *Other business workflows can also potentially be changed. At the moment, after a ticketing system has been released, the client's feedback* [about the classification model(s)] *is difficult to deal with, because retraining models cannot target individual issues. Now we can. This is wonderful. It will be possible that we can receive feedback regularly from clients and update ML models, possibly monthly. Our software for these ticketing systems is very sophisticated. ML models can be updated easily.*

R3: Designing and developing a VIS4ML tool for supporting ML workflows involving data synthesis. The iGAiVA prototype is useful for informing future design and development in the industry. The company will invest more resources to transform it into a web-based prototype suitable as a "demo" in the company. The discussion on industrial software engineering is informative to academic researchers. A research prototype is not the same as an industrial prototype.

▶ *It is a good idea to build a software tool for supporting ML developers. The current design* [of iGAiVA] *is suitable for a piece of academic work. It can help shape the design of an industrial software tool, which will take many months and will require the company to allocate a fair amount of resources* [for the development].

▶ *The design idea of four views is sensible at the moment. However, the software engineering process in a large industrial company is very different from* [an] *academic* [development]. *The next step will be for us* [the company] *to allocate a small amount of resources to transform this* [i.e., iGAiVA] *to a web-based tool. It is still a prototype. We call this "demo"* [in the company]. *So the ML people in the company across different regions can get their hands on it. In this way, we can use this demo* [i.e., web-based prototype] *in different contexts and receive comprehensive feedback. I am sure that this will happen.*

▶ *From the* [comprehensive] *feedback about the web-based prototype, the company will make a long-term decision. It is about how to integrate the new VIS and LLM functions into the ML* [development] *workflows in the company. This could be a standalone tool, or we can add these functions to existing tools. It is too early to predict such a decision. But the combined VIS and LLM solution and the iGAiVA prototype are setting the ball rolling.*

.3 iGAiVA Further Discussions

Scalability. The term "scalability" is defined in relation to a scaling variable x. Given n messages in a training dataset, there are numerous ways to select k messages. The total number of possible combinations for a fixed k is:

$$C\binom{n}{k} = \frac{n!}{k!(n-k)!}$$

In our example, $n = 39,100 \times 80\% = 31,280$ (Table 4.1) and $20 < k < 300$ (Table 4.2, 5 synthetic messages per example message). When k is varying from 1 to n, the total possible combinations is:

$$\sum_{k=1}^{n} C\binom{n}{k}$$

Among these combinations, there is one or more combinations that can be used to generate the most effective set of synthetic data using LLMs for improving a model. In addition, finding the optimal set of example messages also depends on many other factors, e.g., the parameter settings for LLMs, choices of ML frameworks,

design options of the architectures of the text classification models, hyperparameters for training the models, and so on. Hence, the search space for an optimal set of messages in the training data is gigantic, and it is not feasible to search the whole space using a brute-force strategy, either manually or using a computer. In other words, when $x = n$, neither humans nor computers are scalable to the whole space.

In our work, we found that (i) targeting one class was usually better than targeting multiple classes simultaneously; and (ii) the success rate for VA-assisted selection was higher than random selection (see Appendix .1). With this finding, iGAiVA was designed for ML developers to observe PCA scatter plots and RBF heatmaps and to select examples in a class-by-class manner, though the iGAiVA users can merge synthetic data generated using examples chosen from multiple classes.

As iGAiVA represents a new approach. According to our observation, when a user works with a specific class, the user pays relatively little attention to other classes (i.e., the grey dots in PCA scatter plots). Hence, the effort for processing m classes is largely linearly related to m. Let us consider x is the number of classes of a text classification problem, e.g., $x = 15$ in our example. Since most text classification problems have a relatively small number of classes, it is not difficult to conclude that the VA approach is scalable in this context.

In many informal conversations, the term "scalability" is often used to convey a doubt, e.g., *can humans handle more data than machines, or can humans process data faster than machines?* One often hears the question of whether a human-centric process is scalable from colleagues who are interested in automatic techniques. Mathematically, the term "scalability" is not correctly used here. For a given problem with a well-defined algorithmic solution, of course, humans usually cannot handle more data than machines, and humans cannot process data faster than machines. However, one should not draw a general conclusion that VA is not good enough, or use the incorrect term, "VA is not scalable". Meanwhile, a well-defined algorithm that is faster than humans can still not be scalable. Most timetabling algorithms are of NP complexity, and they are not scalable. However, they are faster than humans, and that is why they are commonly deployed in universities.

Currently, the visual patterns discussed in Section 4.4 are not well-defined, and there is no adequate algorithm to identify such patterns. Hence, VA allows humans to perform tasks that machines cannot perform. In addition, from identified patterns, an experienced ML developer can formulate hypotheses as discussed in Section 4.4. There is no known algorithm that can transform such visual patterns into hypotheses. Hence, in this context, the algorithmic ability of machines does not exist, and we cannot take the machines' advantages in terms of processing more data and processing faster.

Potential Challenges with LLMs. There must be many potential challenges with LLMs. It will not be appropriate for this work to cover a broad spectrum of such potential challenges. From the perspective of this work, the current LLM-based GAI technology can help solve a major industrial problem, i.e., generating synthetic data for improving text classification models. During this work, we were involved in several discussions about the shortcomings of LLMs. For example, some questioned that LLMs might not generate realistic or correct text messages, assuming that the ML models must be trained with realistic and correct text messages. From an industrial perspective, the answer was rather simple. Many collected messages contain spelling and grammatical errors. The styles of some messages could easily lead to a conclusion as unrealistic if one did not know that it was collected. There are also mislabeled messages. Training ML models using imperfect data is the norm in the industry. As long as any less-ideal synthetic data can help improve a model, using such data is the correct approach.

Some suggested that the cost of using LLMs in an industrial setting could be an issue. However, this does not affect the application of this research. Because LLMs are not deployed in ticketing systems, the hosts of ticketing systems do not need to bear the cost of using LLMs. The text classification models are developed, improved, and maintained in the VA+LLM workflows residing at the industrial partner. When the developed models are deployed in different ticketing systems, LLMs are not used.

Lessons learned, Both academic and industrial partners gained a lot of experience in this collaboration. We had the opportunity to read a collection

of research papers. Academic researchers had the opportunity to work on real-world problems. Industrial partners had the opportunity to receive ideas about VA approaches that were unfamiliar to them before. Of course, there are many things that we could do differently, such as in some cases, meeting domain experts individually was more effective than a large meeting. In general, the project was conducted smoothly according to the plan.

Limitations. The main contributions of this work are (i) proposing a novel VIS4ML approach in which VIS techniques are used to guide the processes of data synthesis using LLMs. (ii) Developing a software tool, iGAiVA, which integrates generative AI and visual analytics into a unified ML workflow. (iii) Demonstrating the effectiveness of targeted data synthesis in improving ML model accuracy through visual analytics. In comparison with the existing workflow as shown in Figure 4.1(a), the new workflow has many advantages as discussed throughout this Chapter and Appendix .2. Perhaps the only perceived "problem" is that Inetum has to invest additional resources to transform iGAiVA to an industrial software product. This is a good problem to have than not so.

A limitation that cannot be improved is not really a limitation. In comparison with a piece of future technology that improves upon some aspects of iGAiVA, those aspects that will be improved are limitations from future perspectives. For example, if there were a better visual representation than the RBF heatmap, the current RBF heatmap has limitations. We can easily anticipate that there will be future technologies that will improve some aspects of the iGAiVA approach, such as improved LLMs, new and better visual designs, better user interface designs, and so on. Hence, from a future perspective, it is possible that every aspect of this work can be a limitation. That is why many published research works are now obsolete.

.4 iGAiVA High-resolution Images

In this appendix, we provide high-resolution versions of the images in Figures 4.5, 4.6, and 4.7.

Original Figure 4.5 Caption

Two examples of detailed visual analysis for investigating class T12. The two PCA scatter plots on the left show that Dimension 0 in (a) and Dimension 13 in (b) can separate the data objects into two regions, and data objects in one region have higher recall, while the overall class recall is only 37.5%. Each RBF plot in the second column makes the boundary between the high-recall and low-recall regions clearer, enabling the selection of a division line to study the summary statistics of the messages in the two regions using a tag-treemap on the right.

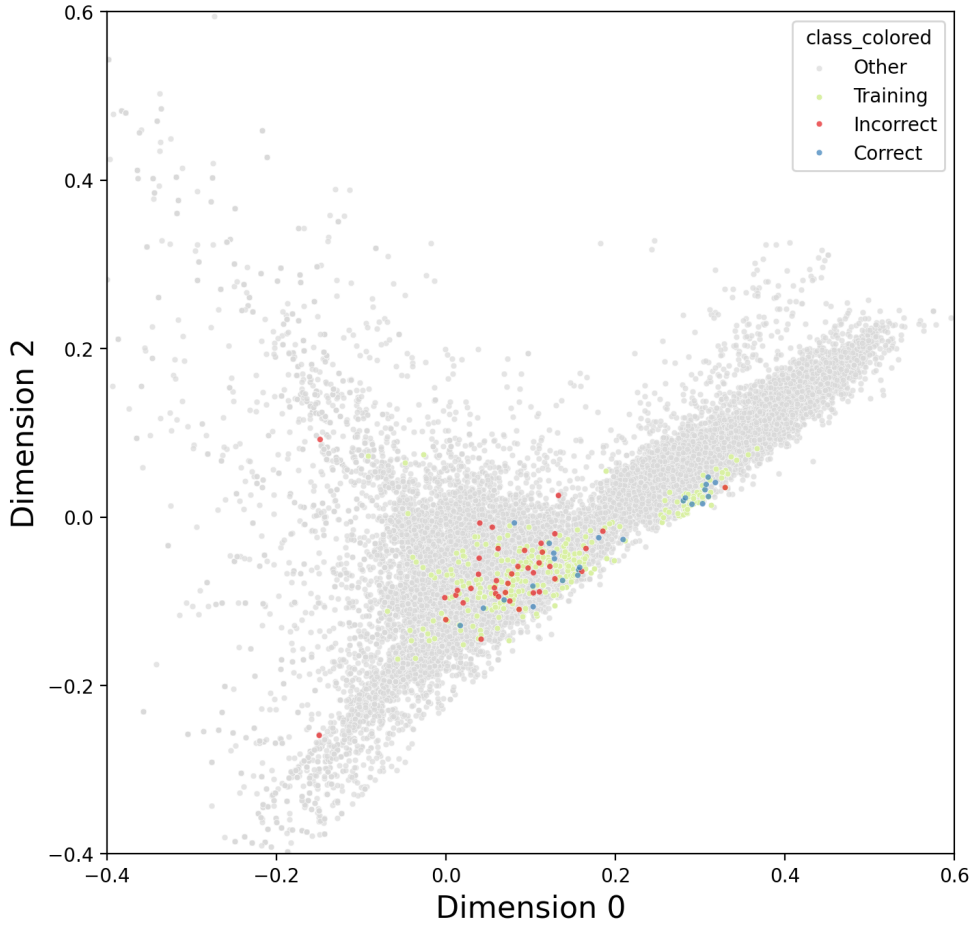


Figure 1: High resolution Figure 4.5(a): column 1.

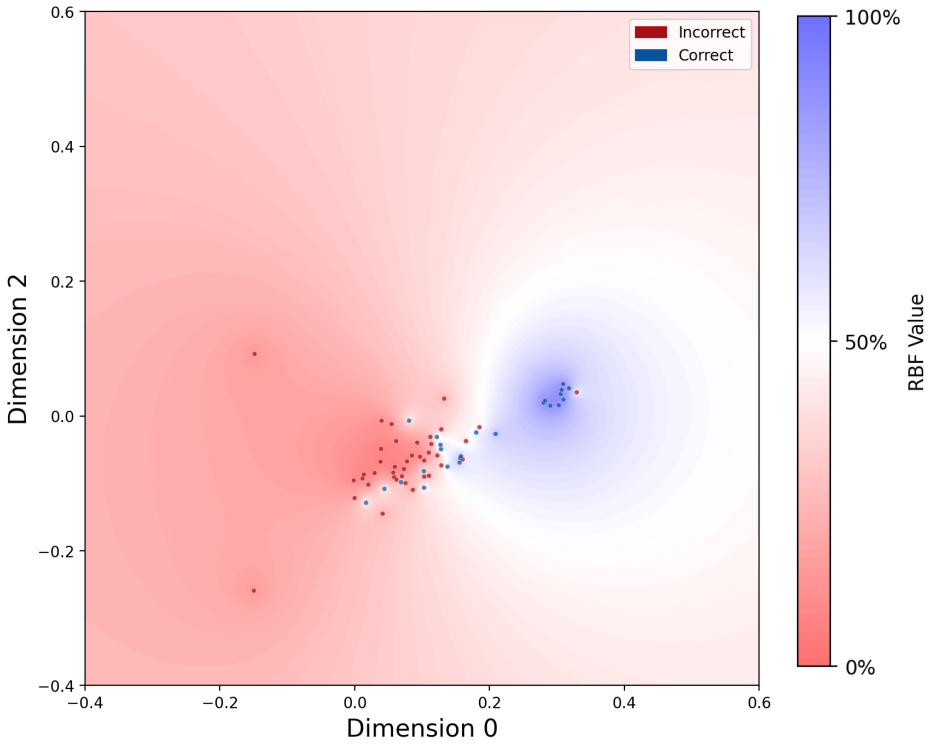


Figure 2: High resolution Figure 4.5(a): column 2.

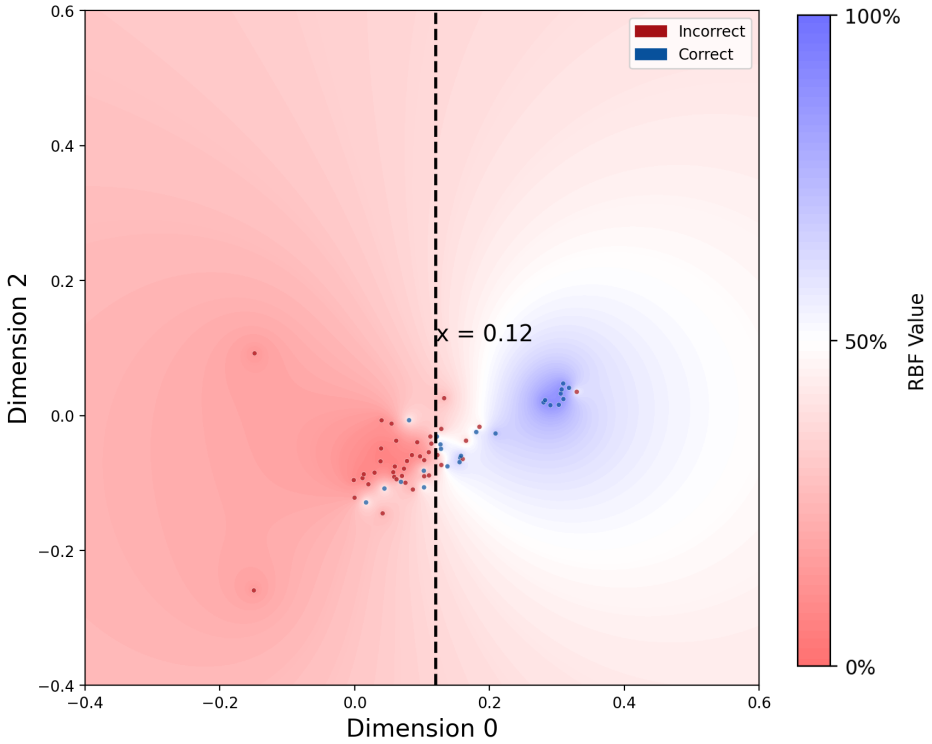


Figure 3: High resolution Figure 4.5(a): column 3.

Figure 4: High resolution Figure 4.5(a): column 4.

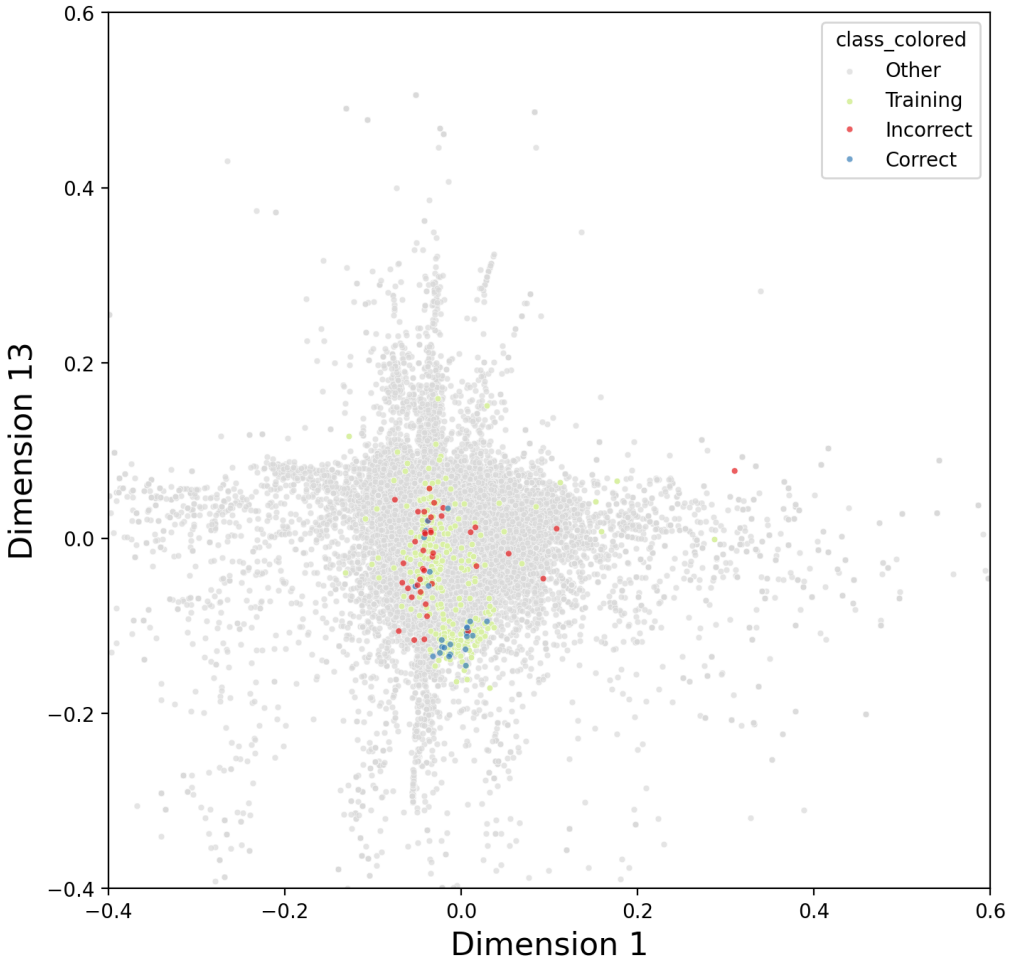


Figure 5: High resolution Figure 4.5(b): column 1.

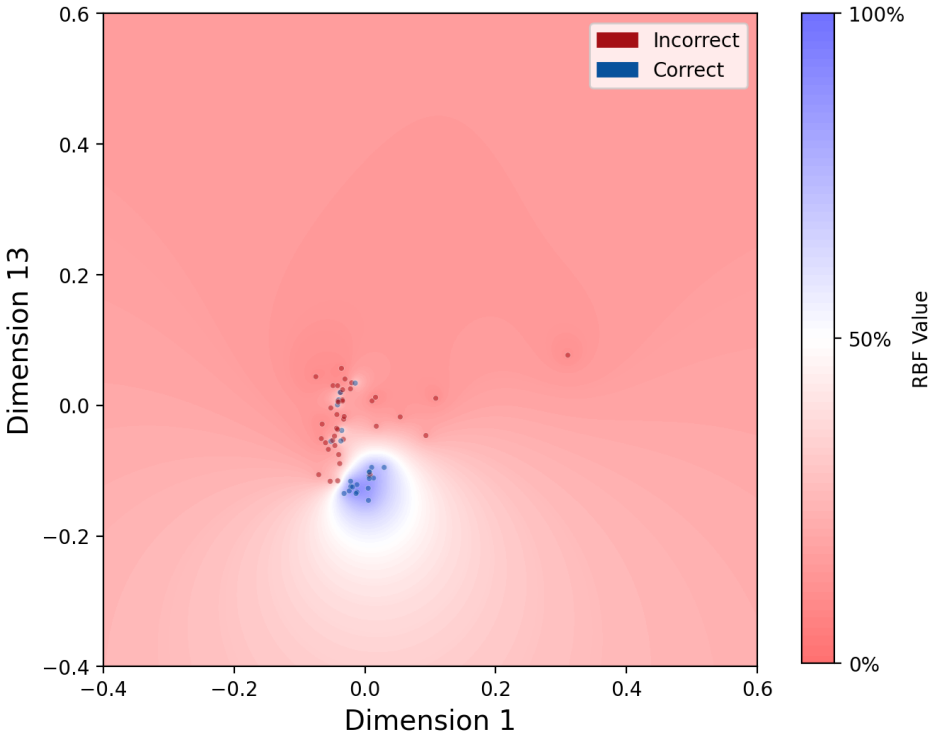


Figure 6: High resolution Figure 4.5(b): column 2.

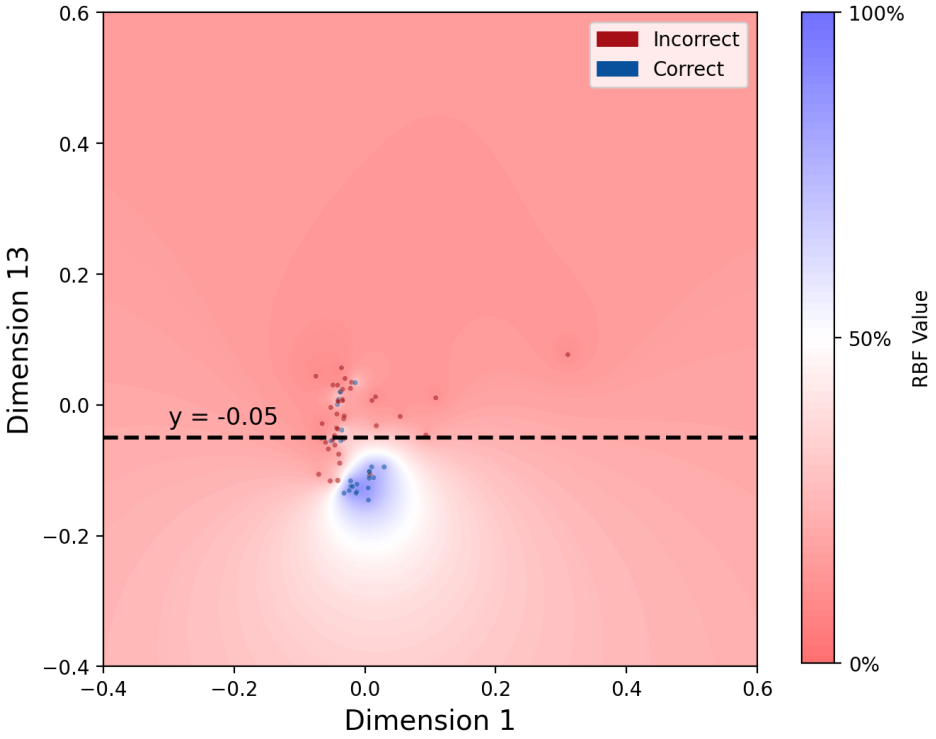


Figure 7: High resolution Figure 4.5(b): column 3.

Figure 8: High resolution Figure 4.5(b): column 4.

Original Figure 4.6 Caption

The class T13 has the lowest recall among all classes. The scatter plot in (a) indicates more classification errors (red dots) when the data objects are associated with lower values in PCA feature dimension 0. The RBF heatmap in (b) confirms this pattern and enables data synthesis to be targeted at an erroneous cluster on the left, as shown in (c). The model retrained with additional LLM-synthesized data is improved in (d). The RBF heatmap for the new testing results in (e) and the zoomed-in scatter plots confirm the improvement.

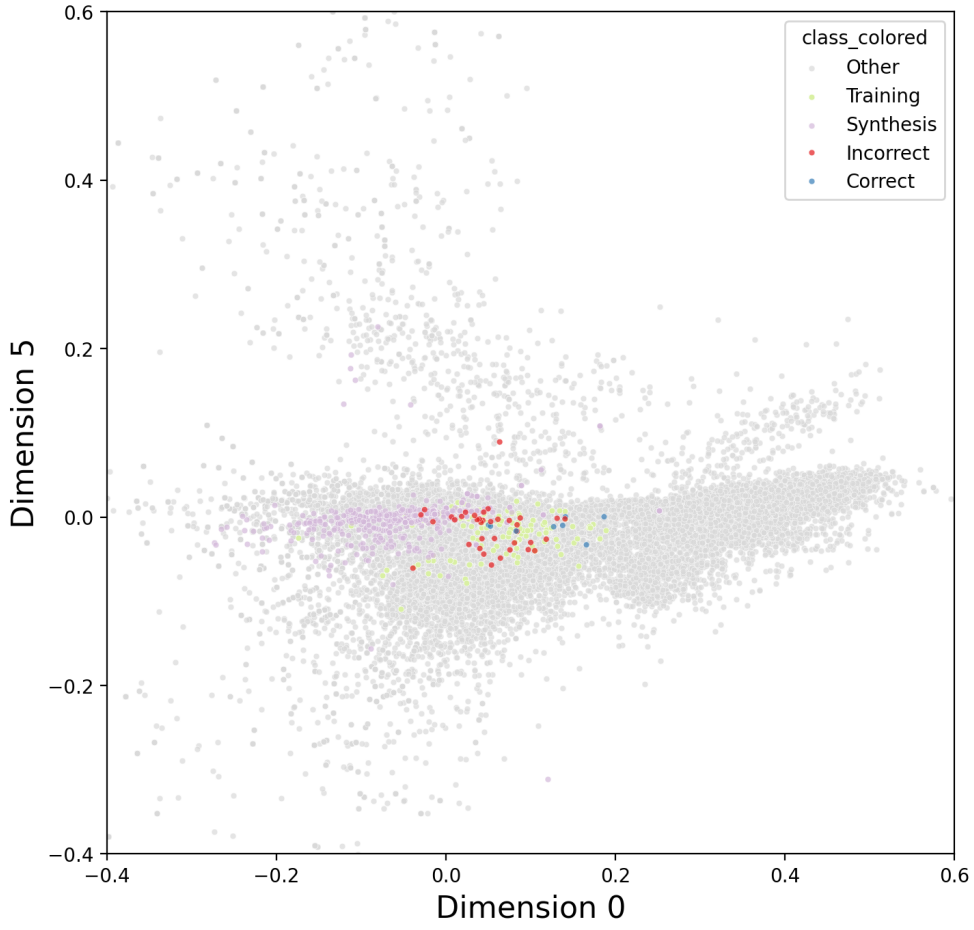


Figure 9: High resolution Figure 4.6(a): correctness distribution (before).

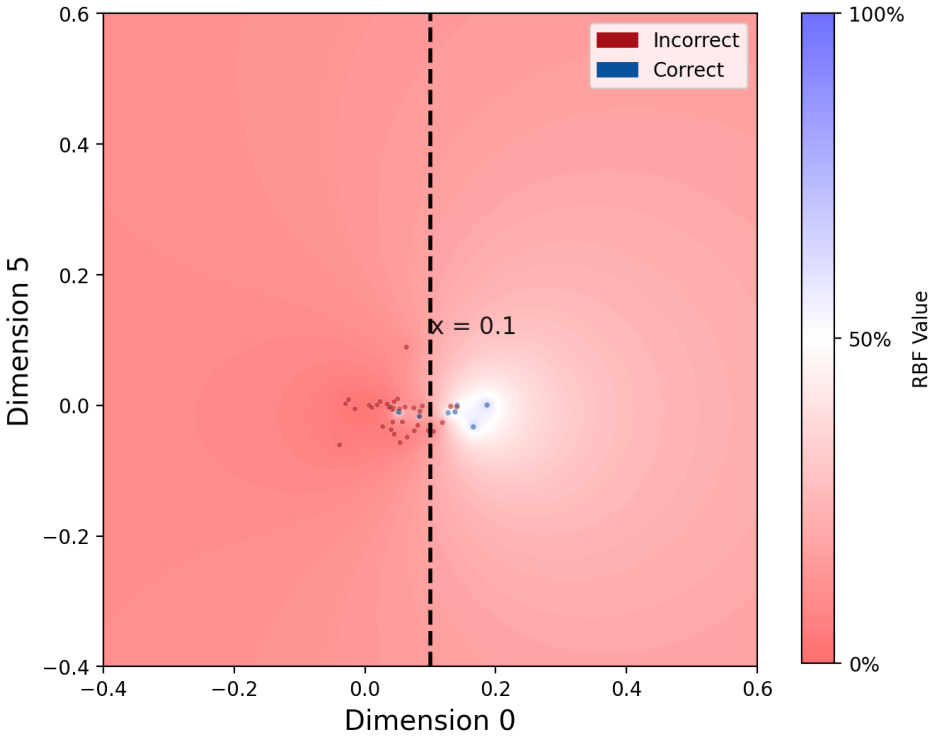


Figure 10: High resolution Figure 4.6(b): two area divided by dimension 0.

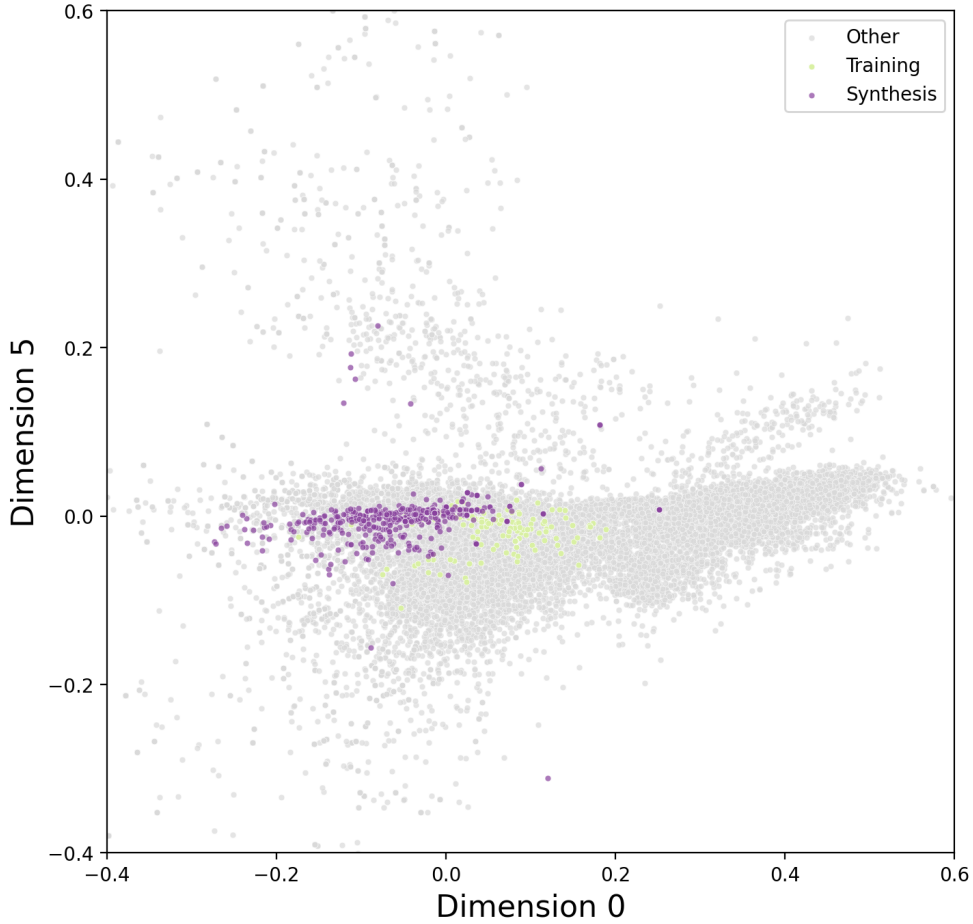


Figure 11: High resolution Figure 4.6(c): distribution of synthesized data.

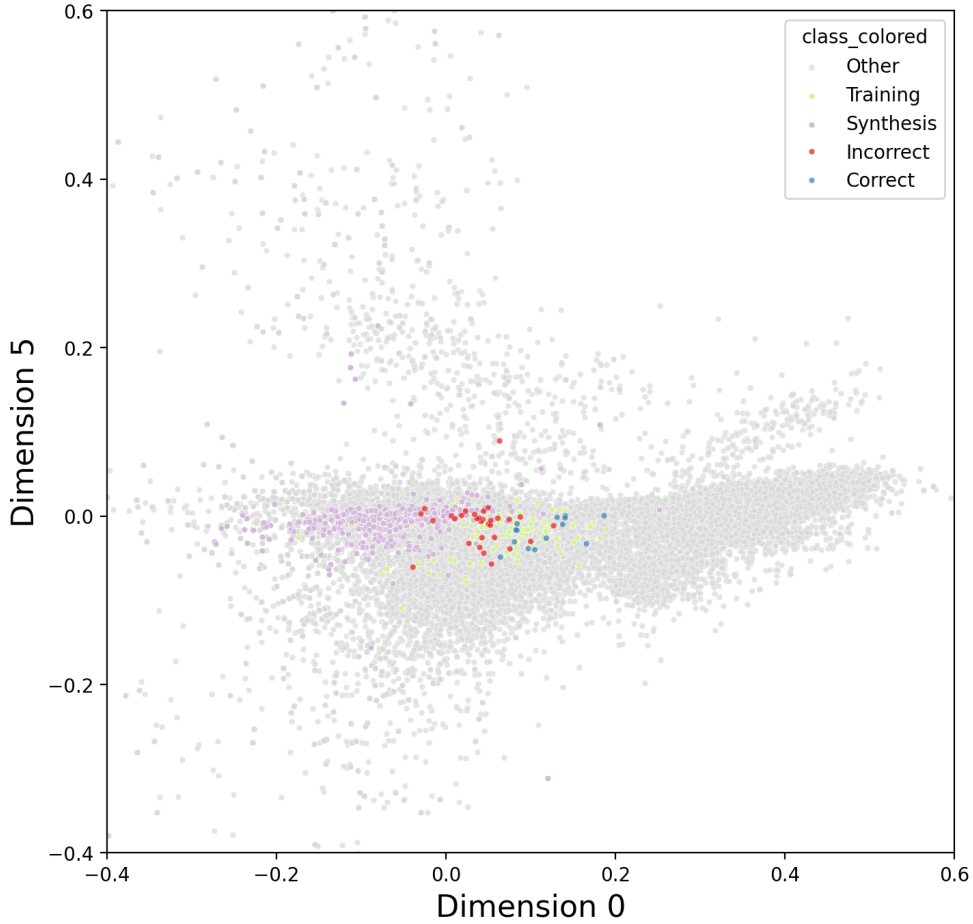


Figure 12: High resolution Figure 4.6(d): correctness distribution (after).

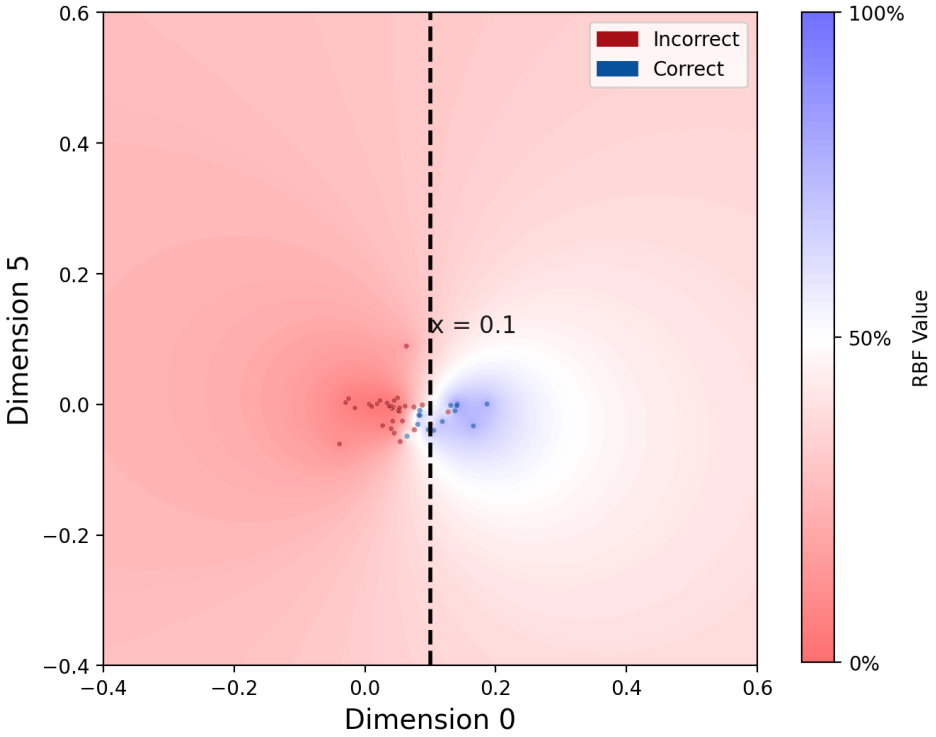


Figure 13: High resolution Figure 4.6(e): noticeable improvement in RBF.

Figure 14: High resolution Figure 4.6(f): zoomed PCA scatter plots.

Original Figure 4.7 Caption

Four views of the iGAiVA tool. The user can switch between views using the top menu bar. (a) The Synthesis View is for supporting mainly the tasks for identifying suitable example data objects as inputs to LLMs for generating synthetic data. (b) The Data View is for selecting a subset of synthetic datasets and combining them with the original training data. (c) The Model View is for monitoring the process of retraining a model and running the retrained model against one or more predefined testing datasets. (d) The Results View is for analyzing and evaluating the results of the model's performance.

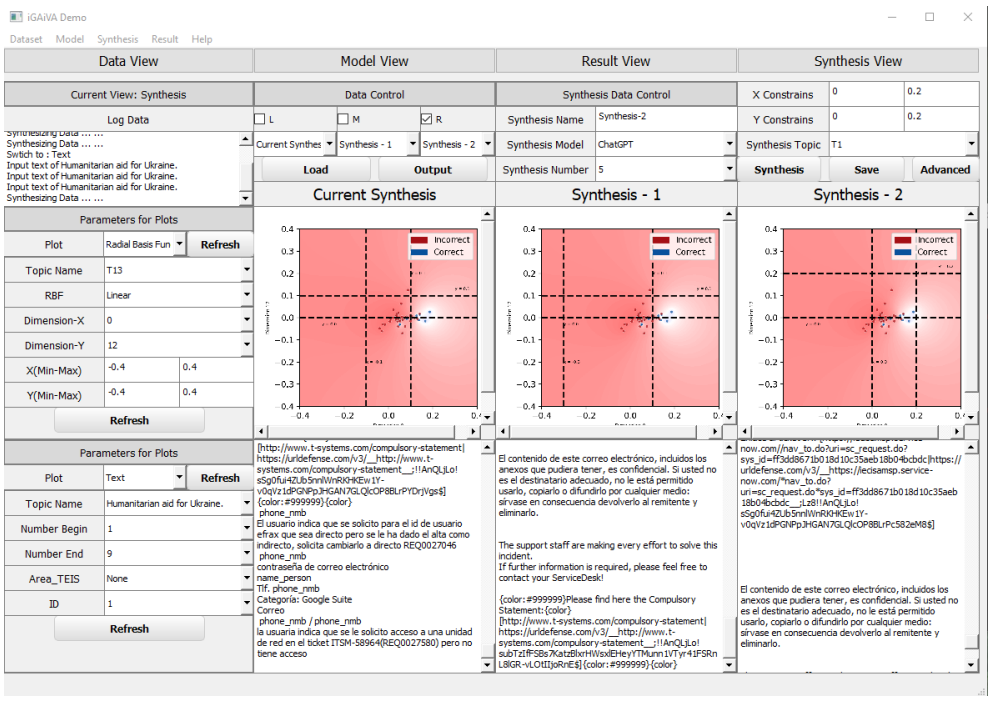


Figure 15: High resolution Figure 4.7(a): Synthesis View.

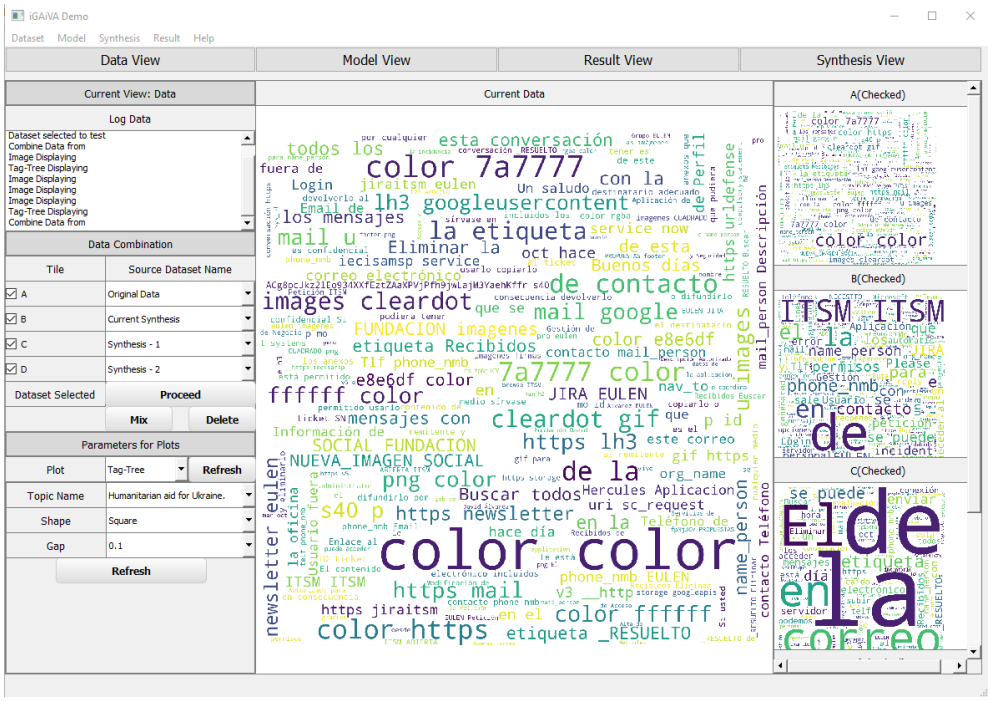


Figure 16: High resolution Figure 4.7(b): Data View.

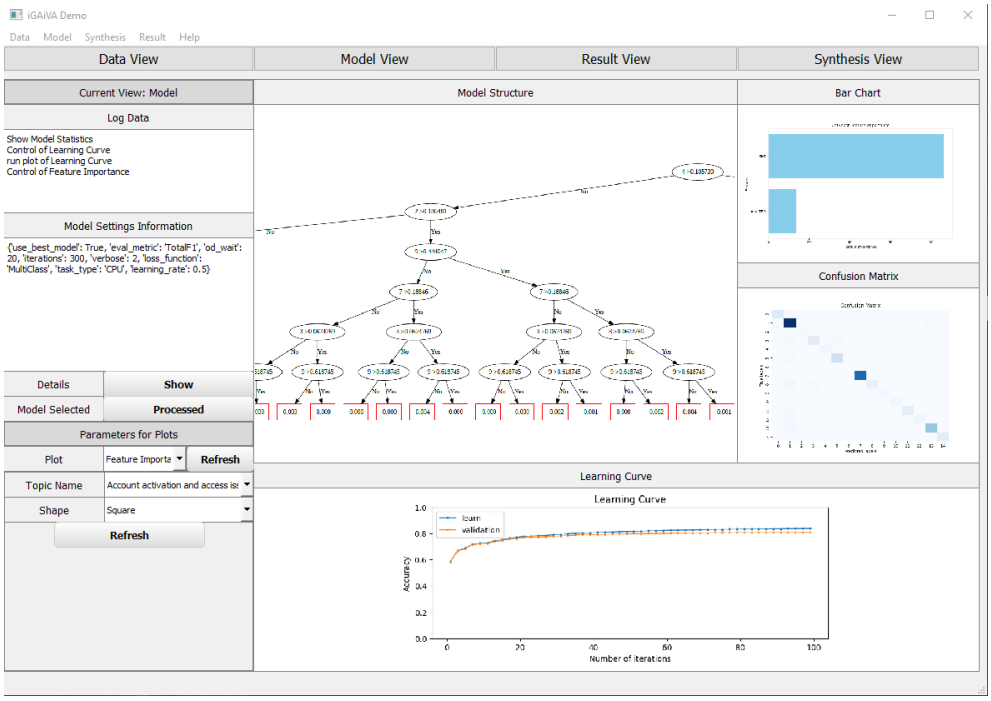


Figure 17: High resolution Figure 4.7(c): Model View.

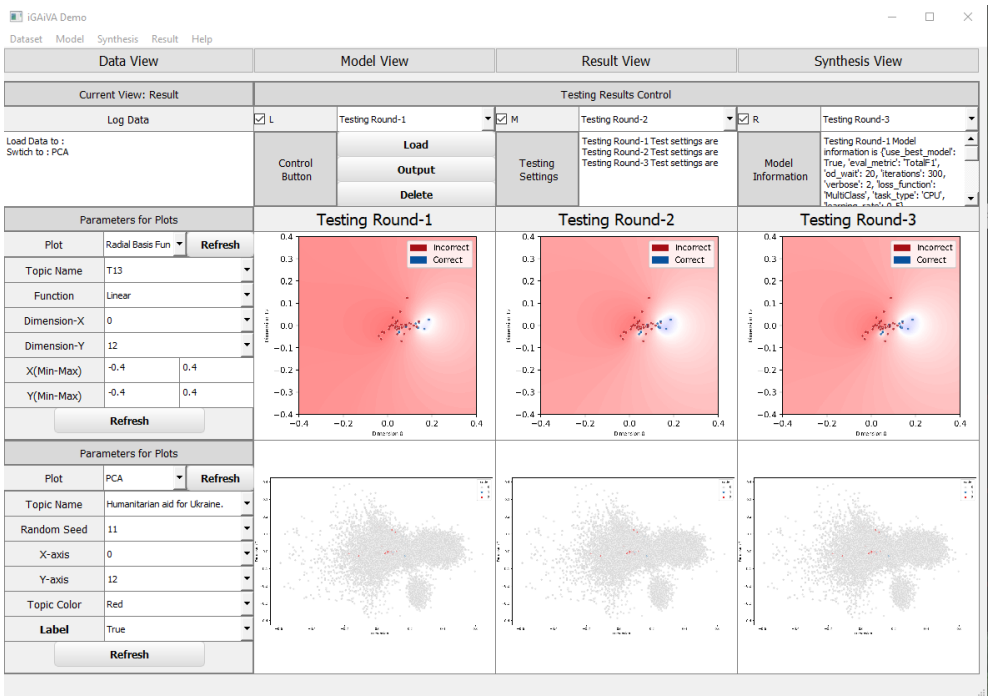


Figure 18: High resolution Figure 4.7(d): Results View.

APPENDICES OF

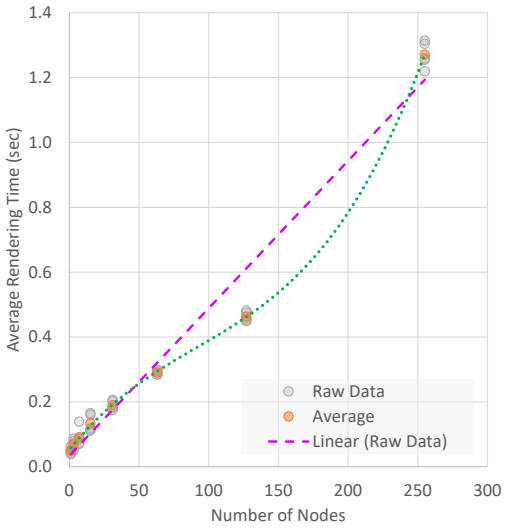
Chapter 7: Radial Icicle Tree (RIT): Node Separation and Area Constancy

.5 Radial Icicle Tree Scalability Test

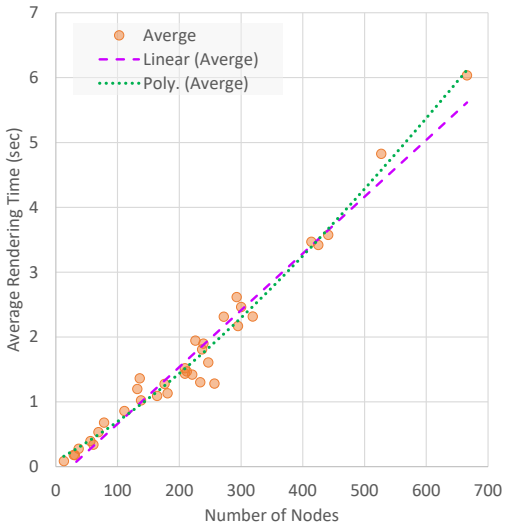
We have conducted a scalability test to evaluate the theoretical conclusion that the complexity of the RIT algorithm is linear, i.e., $O(N_v)$, where N_v is the total number of nodes in a tree. We have created 112 trees of depths between 1 and 8. Each parent node may have up to 12 child nodes. We used three types of tree-generation algorithms:

- **Fixed** — Let $C_{\max} > 1$ be the maximal number of child nodes per parent. This algorithm was used for relatively small $C_{\max}$ values. Otherwise, the total number of nodes would grow too quickly in relation to the tree depth.
- **Random** — For each parent node, the algorithm randomly selects a number $R_C \in [1, C_{\max}]$, and creates R_C child nodes. We noticed that for a slightly large value of $C_{\max}$, the total number of nodes can grow quickly in relation to the tree depth. This algorithm was used for trees that are not so deep or $C_{\max}$ is not so large.
- **Semi-random** — We assign different values of $C_{\max}$ at different tree depths with an overall decreasing trend from depth 2 to leaf nodes, though $C_{\max}$ may not necessarily decrease at each depth. For each parent node, the algorithm creates child nodes in the same way as the above **Random** algorithm.

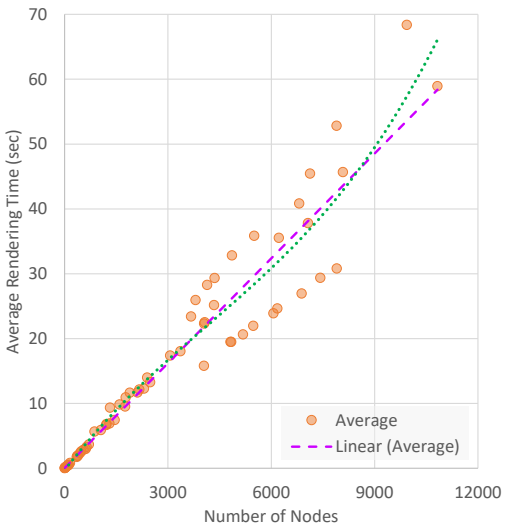
Applying the RIT algorithm to trees generated using the above three tree-generation algorithms exhibited similar scalability patterns, but not exactly the same. Figure 19(a) shows the scalability test of eight binary trees, which are of different depths. Each tree was tested five times, and its average rendering time (sec.) is shown as a single orange dot. The cubic polynomial trend line (green dotted) fits the average data points almost perfectly and differs from the linear trend line



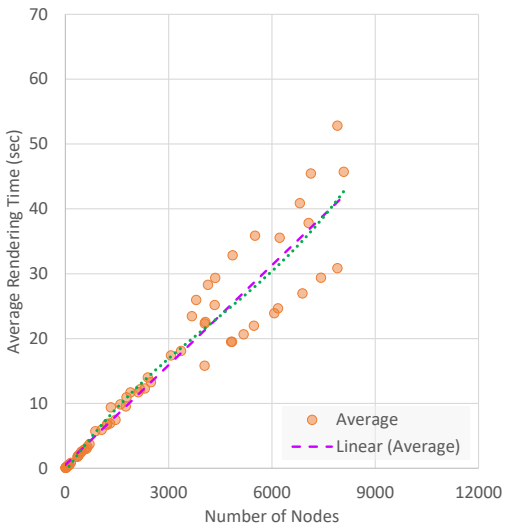
(a) Fixed: $C_{\max} = 2$



(b) Semi-random: $C_{\max} = 12$



(c) Random: $C_{\max} = 8$



(d) Remove two data points in (c)

Figure 19: Scalability tests for three different sets of trees.

(purple dashed) noticeably. For semi-randomly created trees with $C_{\max} = 12$, the difference between cubic and linear trend lines is reduced as shown in Figure 19(b). For randomly-generated trees with $C_{\max} = 8$, there seems to be a similar difference between cubic and linear trend lines as shown in Figure 19(c). However, a close look indicates that the sampling for trees with more than 9000 nodes was too sparse to be reliable. In Figure 19(d), we removed the two trees with 9934 and 10824 nodes respectively, the cubic and linear trend lines became very close to each other. Figure 7.10 in Section 7.7 shows the combined testing results but includes only trees with fewer than 9000 nodes. In other words, it includes all data points in

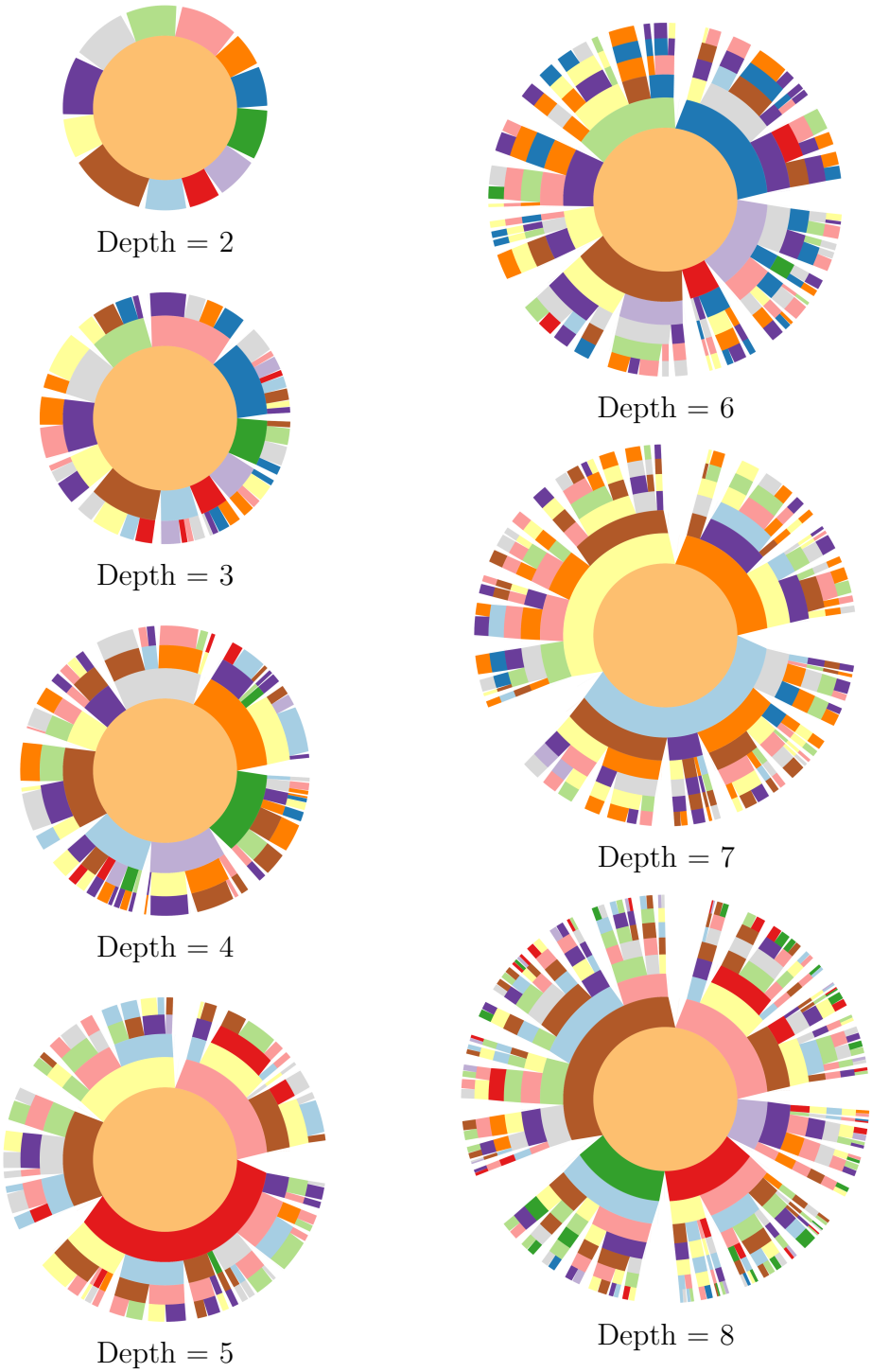


Figure 20: Examples of the trees that were tested.

Figures 19(a,b,d). Figure 20 shows seven examples of the trees tested.

References

[1] Dominik Sacha et al. “SOMFlow: Guided Exploratory Cluster Analysis with Self-Organizing Maps and Analytic Provenance”. In: *IEEE Transactions on Visualization and Computer Graphics* 24.1 (2018), pp. 120–130.

[2] Navdeep Gill et al. “A Responsible Machine Learning Workflow with Focus on Interpretable Models, Post-hoc Explanation, and Discrimination Testing”. In: *Information* 11.3 (2020), p. 137.

[3] Dylan Cashman et al. “A User-based Visual Analytics Workflow for Exploratory Model Analysis”. In: *Computer Graphics Forum*. Vol. 38. 3. Wiley Online Library. 2019, pp. 185–199.

[4] Thilo Spinner et al. “explAIner: A Visual Analytics Framework for Interactive and Explainable Machine Learning”. In: *IEEE Transactions on Visualization and Computer Graphics* 26.1 (2019), pp. 1064–1074.

[5] Alex Bäuerle et al. “Symphony: Composing Interactive Interfaces for Machine Learning”. In: *Proc. ACM Conference on Human Factors in Computing Systems*. 2022, pp. 1–14.

[6] Hendrik Strobelt et al. “Seq2Seq-Vis: A Visual Debugging Tool for Sequence-to-sequence Models”. In: *IEEE Transactions on Visualization and Computer Graphics* 25.1 (2018), pp. 353–363.

[7] Angie Boggust et al. “Compress and Compare: Interactively Evaluating Efficiency and Behavior Across ML Model Compression Experiments”. In: *IEEE Transactions on Visualization and Computer Graphics* (2024).

[8] Aécio Santos et al. “Visus: An Interactive System for Automatic Machine Learning Model Building and Curation”. In: *Proc. Workshop on Human-In-the-Loop Data Analytics*. 2019, pp. 1–7.

[9] Yilin Lu et al. *GNN 101: Visual Learning of Graph Neural Networks in Your Web Browser*. arXiv:2411.17849. 2024.

[10] Stef Van Den Elzen and Jarke J Van Wijk. “Baobabview: Interactive Construction and Analysis of Decision Trees”. In: *Proc. IEEE Conference on Visual Analytics Science and Technology*. IEEE. 2011, pp. 151–160.

[11] Zijie J Wang et al. *GAM Changer: Editing Generalized Additive Models with Interactive Visualization*. arXiv:2112.03245. 2021.

[12] Yuzhe Lu and Adam Perer. *An Interactive Interpretability System for Breast Cancer Screening with Deep Learning*. arXiv:2210.08979. 2022.

[13] Minsuk Kahng et al. “ActiVis: Visual Exploration of Industry-Scale Deep Neural Network Models”. In: *IEEE Transactions on Visualization and Computer Graphics* 24.1 (2018), pp. 88–97.

[14] Mayank Kabra et al. “JAABA: Interactive Machine Learning for Automatic Annotation of Animal Behavior”. In: *Nature Methods* 10.1 (2013), pp. 64–67.

[15] Advait Sarkar et al. “Interactive Visual Machine Learning in Spreadsheets”. In: *IEEE Symposium on Visual Languages and Human-Centric Computing*. IEEE. 2015, pp. 159–163.

[16] Justin Cheng and Michael S Bernstein. “Flock: Hybrid Crowd-machine Learning Classifiers”. In: *Proc. 18th ACM Conference on Computer Supported Cooperative Work & Social Computing*. 2015, pp. 600–611.

[17] Jun Yuan et al. “Vibe: A Visual Analytics Workflow for Semantic Error Analysis of CVML Models at Subgroup Level”. In: *Proc. 30th International Conference on Intelligent User Interfaces*. 2025, pp. 1529–1547.

[18] A. V. Balakrishnan. “On the Problem of Time Jitter in Sampling”. In: *IRE Transactions on Information Theory* 8.3 (1962), pp. 226–236.

[19] Frank Olken. “Random Sampling from Databases”. PhD thesis. Citeseer, 1993.

[20] Christian Habermann and Fabian Kindermann. “Multidimensional Spline Interpolation: Theory and Applications”. In: *Computational Economics* 30 (2007), pp. 153–169.

[21] Akram Aldroubi, Michael Unser, and Murray Eden. “Cardinal Spline Filters: Stability and Convergence to the Ideal Sinc Interpolator”. In: *Signal Processing* 28.2 (1992), pp. 127–138.

[22] Khandakar M Rashid and Joseph Louis. “Window-warping: A Time Series Data Augmentation of IMU Data for Construction Equipment Activity Identification”. In: *ISARC. Proc. International Symposium on Automation and Robotics in Construction*. Vol. 36. IAARC Publications. 2019, pp. 651–657.

[23] Brian Kenji Iwana and Seiichi Uchida. “An Empirical Survey of Data Augmentation for Time Series Classification with Neural Networks”. In: *Plos one* 16.7 (2021), e0254841.

[24] Brian Kenji Iwana and Seiichi Uchida. “Time Series Data Augmentation for Neural Networks by Time Warping with A Discriminative Teacher”. In: *25th International Conference on Pattern Recognition*. IEEE. 2021, pp. 3558–3565.

[25] Jinsung Yoon, Daniel Jarrett, and Mihaela Van der Schaar. “Time-series Generative Adversarial Networks”. In: *Advances in Neural Information Processing Systems* 32 (2019).

[26] Marta Fochesato et al. “On the Use of Conditional TimeGAN to Enhance the Robustness of A Reinforcement Learning Agent in the Building Domain”. In: *Proc. ACM International Conference on Systems for Energy-Efficient Buildings, Cities, and Transportation*. 2022, pp. 208–217.

[27] ZG Liu et al. “Conditional-TimeGAN for Realistic and High-quality Appliance Trajectories Generation and Data Augmentation in Non-intrusive Load Monitoring”. In: *IEEE Transactions on Instrumentation and Measurement* (2024).

[28] Lequan Lin et al. “Diffusion Models for Time-series Applications: A Survey”. In: *Frontiers of Information Technology & Electronic Engineering* 25.1 (2024), pp. 19–41.

[29] Xinyu Yuan and Yan Qiao. *Diffusion-TS: Interpretable Diffusion for General Time Series Generation*. arXiv:2403.01742. 2024.

[30] Yangming Li. *TS-Diffusion: Generating Highly Complex Time Series with Diffusion Models*. arXiv:2311.03303. 2023.

[31] Jian Qian et al. *TimeLDM: Latent Diffusion Model for Unconditional Time Series Generation*. arXiv:2407.04211. 2024.

[32] Xiaomin Li et al. "BioDiffusion: A Versatile Diffusion Model for Biomedical Signal Synthesis". In: *Bioengineering* 11.4 (2024), p. 299.

[33] Yingzhou Lu, Huazheng Wang, and Wenqi Wei. *Machine Learning for Synthetic Data Generation: A Review*. arXiv:2302.04062. 2023.

[34] Valerie Restat et al. "GouDa-generation of Universal Data Sets: Improving Analysis and Evaluation of Data Preparation Pipelines". In: *Proc. Workshop on Data Management for End-To-End Machine Learning*. 2022, pp. 1–6.

[35] Daniela Ichim. "Quantile-based Bootstrap Methods to Generate Continuous Synthetic Data". In: *Proc. EDBT/ICDT Workshops*. 2010, pp. 1–10.

[36] Miro Mannino and Azza Abouzied. "Synner: Generating Realistic Synthetic Data". In: *Proc. ACM SIGMOD International Conference on Management of Data*. 2020, pp. 2749–2752.

[37] Sion Ll Rhys, Simon Poulding, and John A Clark. "Using Automated Search to Generate Test Data for MATLAB". In: *Proc. Annual conference on Genetic and Evolutionary Computation*. 2009, pp. 1697–1704.

[38] Achraf Skander, Lionel Roucoules, and Jean Sébastien Klein Meyer. "Design and Manufacturing Interface Modelling for Manufacturing Processes Selection and Knowledge Synthesis in Design". In: *The International Journal of Advanced Manufacturing Technology* 37 (2008), pp. 443–454.

[39] Markus Hittmeir, Andreas Ekelhart, and Rudolf Mayer. "On the Utility of Synthetic Data: An Empirical Evaluation on Machine Learning Tasks". In: *Proc. International Conference on Availability, Reliability and Security*. 2019, pp. 1–6.

[40] Sergey I Nikolenko. *Synthetic Data for Deep Learning*. Vol. 174. Springer, 2021.

[41] Richard J Chen et al. "Synthetic Data in Machine Learning for Medicine and Healthcare". In: *Nature Biomedical Engineering* 5.6 (2021), pp. 493–497.

[42] Jae Hee Kim and Sang Won Choi. "A Deep Learning-based Approach for Radiation Pattern Synthesis of an Array Antenna". In: *IEEE Access* 8 (2020), pp. 226059–226063.

[43] Apostolia Tsirikoglou, Gabriel Eilertsen, and Jonas Unger. "A Survey of Image Synthesis Methods for Visual Machine Learning". In: *Computer Graphics Forum*. Vol. 39. Wiley Online Library. 2020, pp. 426–451.

[44] Corneliu Ilisescu et al. "Responsive Action-based Video Synthesis". In: *Proc. ACM Conference on Human Factors in Computing Systems*. 2017, pp. 6569–6580.

[45] Mark A. Whiting, Jereme Haack, and Carrie Varley. "Generating A Synthetic Video Dataset". In: *Proc. BELIV'10 Workshop: BEyond Time and Errors: Novel EvaLuation Methods for Information Visualization*. BELIV '10. Atlanta, Georgia: Association for Computing Machinery, 2010, pp. 43–48.

[46] Vygandas' Vegas' Šimbelis and Anders Lundström. "Synthesis in the Audiovisual". In: *Proc. ACM Conference Extended Abstracts on Human Factors in Computing Systems*. 2016, pp. 301–304.

[47] Diemo Schwarz. "A System for Data-driven Concatenative Sound Synthesis". In: *Digital Audio Effects*. 2000, pp. 97–102.

[48] Chenglong Wang et al. "Falx: Synthesis-powered Visualization Authoring". In: *Proc. ACM Conference on Human Factors in Computing Systems*. 2021, pp. 1–15.

[49] Mark A Whiting, Jereme Haack, and Carrie Varley. "Creating Realistic, Scenario-based Synthetic Data for Test and Evaluation of Information Analytics Software". In: *Proc. Workshop on Beyond Time and Errors: Novel Evaluation Methods for Information Visualization*. 2008, pp. 1–9.

[50] Taoxin Peng and Alexander Telle. "A Tool for Generating Synthetic Data". In: *Proc. International Conference on Data Science, E-Learning and Information Systems*. 2018, pp. 1–6.

[51] Hochul Hwang et al. "Eldersim: A Synthetic Data Generation Platform for Human Action Recognition in Eldercare Care Applications". In: *IEEE Access* (2021).

[52] Shahid Latif et al. "Kori: Interactive Synthesis of Text and Charts in Data Documents". In: *IEEE Transactions on Visualization and Computer Graphics* 28.1 (2021), pp. 184–194.

[53] Miro Mannino and Azza Abouzied. "Is This Real? Generating Synthetic Data that Looks Real". In: *Proc. Annual ACM Symposium on User Interface Software and Technology*. 2019, pp. 549–561.

[54] Christopher M Bishop and Nasser M Nasrabadi. *Pattern Recognition and Machine Learning*. Vol. 4. 4. Springer, 2006.

[55] Kevin P Murphy. *Machine Learning: A Probabilistic Perspective*. MIT Press, 2012.

[56] Trevor Hastie et al. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Vol. 2. Springer, 2009.

[57] Leo Breiman. "Random Forests". In: *Machine Learning* 45 (2001), pp. 5–32.

[58] Jerome H Friedman. "Greedy Function Approximation: A Gradient Boosting Machine". In: *Annals of Statistics* (2001), pp. 1189–1232.

[59] Yoav Freund and Robert E Schapire. "A Decision-theoretic Generalization of On-line Learning and an Application to Boosting". In: *Journal of Computer and Systems Sciences* 55.1 (1997), pp. 119–139.

[60] Ian Goodfellow et al. *Deep Learning*. Vol. 1. 2. MIT Press Cambridge, 2016.

[61] Anil K Jain. "Data Clustering: 50 years beyond K-means". In: *Pattern Recognition Letters* 31.8 (2010), pp. 651–666.

[62] Hervé Abdi and Lynne J Williams. "Principal Component Analysis". In: *Wiley Interdisciplinary Reviews: Computational Statistics* 2.4 (2010), pp. 433–459.

[63] James MacQueen. "Some Methods for Classification and Analysis of Multivariate Observations". In: *Proc. 15th Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Statistics*. Vol. 5. University of California Press. 1967, pp. 281–298.

[64] Max Kuhn, Kjell Johnson, et al. *Applied Predictive Modeling*. Vol. 26. Springer, 2013.

[65] Thomas Cover and Peter Hart. "Nearest Neighbor Pattern Classification". In: *IEEE Transactions on Information Theory* 13.1 (1967), pp. 21–27.

[66] Corinna Cortes and Vladimir Vapnik. "Support-vector Networks". In: *Machine Learning* 20 (1995), pp. 273–297.

[67] David W Aha, Dennis Kibler, and Marc K Albert. "Instance-based Learning Algorithms". In: *Machine Learning* 6 (1991), pp. 37–66.

[68] Cynthia Rudin. "Stop Explaining Black Box Machine Learning Models for High Stakes Decisions and Use Interpretable Models Instead". In: *Nature Machine Intelligence* 1.5 (2019), pp. 206–215.

[69] Molnar Christoph. *Interpretable Machine Learning: A Guide for Making Black Box Models Explainable*. Leanpub, 2020.

[70] Fred Hohman et al. "Visual Analytics in Deep Learning: An Interrogative Survey for the Next Frontiers". In: *IEEE Transactions on Visualization and Computer Graphics* 25.8 (2018), pp. 2674–2693.

[71] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. "" Why Should I Trust You?" Explaining the Predictions of Any Classifier". In: *Proc. 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 2016, pp. 1135–1144.

[72] Thomas G Dietterich. "Ensemble Methods in Machine Learning". In: *International Workshop on Multiple Classifier Systems*. Springer. 2000, pp. 1–15.

[73] Zhi-Hua Zhou. *Ensemble Methods: Foundations and Algorithms*. CRC press, 2025.

[74] Tianqi Chen and Carlos Guestrin. "XGBoost: A Scalable Tree Boosting System". In: *Proc. 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 2016, pp. 785–794.

[75] Dominik Sacha et al. "What You See Is What You Can Change: Human-centered Machine Learning by Interactive Visualization". In: *Neurocomputing* 268 (2017), pp. 164–175.

[76] J. Ross Quinlan. "Induction of Decision Trees". In: *Machine Learning* 1.1 (1986), pp. 81–106.

[77] Leo Breiman et al. *Classification and Regression Trees Belmont*. CA: Wadsworth International Group, 1984.

[78] J Ross Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann Publishers Inc., 1993.

[79] Roger J Lewis. "An Introduction to Classification and Regression Tree (CART) Analysis". In: *Annual Meeting of the Society for Academic Emergency Medicine in San Francisco, California*. Vol. 14. Citeseer. 2000.

[80] Jerry Ye et al. "Stochastic Gradient Boosted Distributed Decision Trees". In: *Proc. 18th ACM Conference on Information and Knowledge Management*. CIKM '09. Hong Kong, China: ACM, 2009, pp. 2061–2064.

[81] Guolin Ke et al. “LightGBM: A Highly Efficient Gradient Boosting Decision Tree”. In: *Advances in Neural Information Processing Systems*. Vol. 30. Curran Associates, Inc., 2017.

[82] Stefano Calzavara et al. “Explainable Global Fairness Verification of Tree-Based Classifiers”. In: *Proc. IEEE Conference on Secure and Trustworthy Machine Learning*. IEEE. 2023, pp. 1–17.

[83] Fabian Pedregosa et al. “Scikit-learn: Machine Learning in Python”. In: *The Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[84] Liudmila Prokhorenkova et al. “CatBoost: Unbiased Boosting with Categorical Features”. In: *Advances in Neural Information Processing Systems* 31 (2018).

[85] Si Si et al. “Gradient Boosted Decision Trees for High Dimensional Sparse Output”. In: *International Conference on Machine Learning*. PMLR. 2017, pp. 3182–3190.

[86] Zixiao Kong et al. “A Survey on Adversarial Attack in the Age of Artificial Intelligence”. In: *Wireless Communications and Mobile Computing* 2021.1 (2021), p. 4907754.

[87] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. *Explaining and Harnessing Adversarial Examples*. arXiv:1412.6572. 2014.

[88] Florian Tramèr et al. *Ensemble Adversarial Training: Attacks and Defenses*. arXiv:1705.07204. 2017.

[89] Alexey Kurakin, Ian Goodfellow, and Samy Bengio. *Adversarial Machine Learning at Scale*. arXiv:1611.01236. 2016.

[90] Cihang Xie et al. “Improving Transferability of Adversarial Examples with Input Diversity”. In: *Proc. IEEE Conference on Computer Vision and Pattern Recognition*. June 2019.

[91] Seyed-Mohsen Moosavi-Dezfooli et al. “Universal Adversarial Perturbations”. In: *Proc. IEEE Conference on Computer Vision and Pattern Recognition*. July 2017.

[92] Jonas Rauber, Wieland Brendel, and Matthias Bethge. *Foolbox: A Python Toolbox to Benchmark the Robustness of Machine Learning Models*. arXiv:1707.04131. 2017.

[93] Ali Shafahi et al. “Poison Frogs! Targeted Clean-label Poisoning Attacks on Neural Networks”. In: *Advances in Neural Information Processing Systems* 31 (2018).

[94] Siddhant Bhambri et al. *A Survey of Black-box Adversarial Attacks on Computer Vision Models*. arXiv:1912.01667. 2019.

[95] Jiawei Su, Danilo Vasconcellos Vargas, and Kouichi Sakurai. “One Pixel Attack for Fooling Deep Neural Networks”. In: *IEEE Transactions on Evolutionary Computation* 23.5 (2019), pp. 828–841.

[96] Rainer Storn and Kenneth Price. “Differential Evolution – A Simple and Efficient Heuristic for Global Optimization over Continuous Spaces”. In: *Journal of Global Optimization* 11.4 (Dec. 1997), pp. 341–359.

[97] Dan Kondratyuk. *One-pixel-attack-keras*. [EB/OL]. `https://github.com/Hyperparticle/one-pixel-attack-keras` Accessed December 19, 2020.

[98] Washington Garcia et al. “Less Is More: Dimension Reduction Finds On-manifold Adversarial Examples in Hard-label Attacks”. In: *Proc. IEEE Conference on Secure and Trustworthy Machine Learning*. IEEE. 2023, pp. 254–270.

[99] Sanghyun Hong, Nicholas Carlini, and Alexey Kurakin. “Publishing Efficient On-device Models Increases Adversarial Vulnerability”. In: *Proc. IEEE Conference on Secure and Trustworthy Machine Learning*. IEEE. 2023, pp. 271–290.

[100] Giovanni Apruzzese et al. “Real Attackers Don’t Compute Gradients”: Bridging the Gap between Adversarial ML Research and Practice”. In: *Proc. IEEE Conference on Secure and Trustworthy Machine Learning*. IEEE. 2023, pp. 339–364.

[101] Francesco Croce and Matthias Hein. “Segment (Almost) Nothing: Prompt-Agnostic Adversarial Attacks on Segmentation Models”. In: *Proc. IEEE Conference on Secure and Trustworthy Machine Learning*. IEEE. 2024, pp. 425–442.

[102] Edoardo Debenedetti, Nicholas Carlini, and Florian Tramèr. “Evading Black-box Classifiers without Breaking Eggs”. In: *Proc. IEEE Conference on Secure and Trustworthy Machine Learning*. IEEE. 2024, pp. 408–424.

[103] Sonal Joshi et al. “Study of Pre-processing Defenses against Adversarial Attacks on State-of-the-art Speaker Recognition Systems”. In: *IEEE Transactions on Information Forensics and Security* 16 (2021), pp. 4811–4826.

[104] Huadi Zheng et al. “Protecting Decision Boundary of Machine Learning Model with Differentially Private Perturbation”. In: *IEEE Transactions on Dependable and Secure Computing* 19.3 (2020), pp. 2007–2022.

[105] Deqiang Li et al. “Pad: Towards Principled Adversarial Malware Detection against Evasion Attacks”. In: *IEEE Transactions on Dependable and Secure Computing* 21.2 (2023), pp. 920–936.

[106] Ziad Ali, Ameer Mohammed, and Imtiaz Ahmad. “Vulnerability of Deep Forest to Adversarial Attacks”. In: *IEEE Transactions on Information Forensics and Security* (2024).

[107] Yi Ding et al. “Interpreting Universal Adversarial Example Attacks on Image Classification Models”. In: *IEEE Transactions on Dependable and Secure Computing* 20.4 (2022), pp. 3392–3407.

[108] Kaidi Jin et al. “Can We Mitigate Backdoor Attack Using Adversarial Detection Methods?” In: *IEEE Transactions on Dependable and Secure Computing* 20.4 (2022), pp. 2867–2881.

[109] Jie M Zhang et al. “Machine Learning Testing: Survey, Landscapes and Horizons”. In: *IEEE Transactions on Software Engineering* 48.1 (2020), pp. 1–36.

[110] Dusica Marijan, Arnaud Gotlieb, and Mohit Kumar Ahuja. “Challenges of Testing Machine Learning Based Systems”. In: *IEEE International Conference on Artificial Intelligence Testing*. IEEE. 2019, pp. 101–102.

[111] Kexin Pei et al. “Deepxplore: Automated Whitebox Testing of Deep Learning Systems”. In: *Proc. 26th Symposium on Operating Systems Principles*. 2017, pp. 1–18.

[112] Qianyu Guo et al. “Audee: Automated Testing for Deep Learning Frameworks”. In: *Proc. 35th IEEE/ACM International Conference on Automated Software Engineering*. 2020, pp. 486–498.

[113] Arjun Nitin Bhagoji et al. *Exploring the Space of Black-box Attacks on Deep Neural Networks*. arXiv:1712.09491. 2017.

[114] Aleksander Madry et al. *Towards Deep Learning Models Resistant to Adversarial Attacks*. arXiv:1706.06083. 2017.

[115] Andrew Ilyas et al. “Adversarial Examples Are Not Bugs, They Are Features”. In: *Advances in Neural Information Processing Systems* 32 (2019).

[116] Florian Tramer et al. “On Adaptive Attacks to Adversarial Example Defenses”. In: *Advances in Neural Information Processing Systems* 33 (2020), pp. 1633–1645.

[117] Jason Yosinski et al. *Understanding Neural Networks through Deep Visualization*. arXiv:1506.06579. 2015.

[118] Christian Szegedy et al. *Intriguing Properties of Neural Networks*. arXiv:1312.6199. 2013.

[119] Daniel Zoran and Yair Weiss. “Scale Invariance and Noise in Natural Images”. In: *IEEE 12th International Conference on Computer Vision*. 2009, pp. 2209–2216.

[120] James R Voss, Luis Rademacher, and Mikhail Belkin. “Fast Algorithms for Gaussian Noise Invariant Independent Component Analysis”. In: *Advances in Neural Information Processing Systems* 26 (2013).

[121] L. Scharf and D. Lytle. “Signal Detection in Gaussian Noise of Unknown Level: An Invariance Application”. In: *IEEE Transactions on Information Theory* 17.4 (1971), pp. 404–411.

[122] Nicolas Papernot et al. “The Limitations of Deep Learning in Adversarial Settings”. In: *IEEE European Symposium on Security and Privacy (EuroS&P)*. IEEE. 2016, pp. 372–387.

[123] Pin-Yu Chen et al. “Zoo: Zeroth Order Optimization Based Black-box Attacks to Deep Neural Networks without Training Substitute Models”. In: *Proc. 10th ACM Workshop on Artificial Intelligence and Security* (Nov. 2017), pp. 15–26.

[124] Gamaleldin F. Elsayed, Ian Goodfellow, and Jascha Sohl-Dickstein. *Adversarial Reprogramming of Neural Networks*. arXiv:1806.11146. 2018.

[125] Naveed Akhtar and Ajmal Mian. “Threat of Adversarial Attacks on Deep Learning in Computer Vision: A Survey”. In: *IEEE Access* 6 (2018), pp. 14410–14430.

[126] Jiadong Lin et al. *Nesterov Accelerated Gradient and Scale Invariance for Adversarial Attacks*. arXiv:1908.06281. 2019.

[127] Nicholas Carlini and David Wagner. “Towards Evaluating the Robustness of Neural Networks”. In: *IEEE Symposium on Security and Privacy*. IEEE. 2017, pp. 39–57.

[128] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, and Pascal Frossard. “DeepFool: A Simple and Accurate Method to Fool Deep Neural Networks”. In: *Proc. IEEE Conference on Computer Vision and Pattern Recognition*. 2016.

[129] Yao Li et al. “A Review of Adversarial Attack and Defense for Classification Methods”. In: *The American Statistician* 76.4 (2022), pp. 329–345.

[130] Ali Shafahi et al. “Adversarial Training for Free!” In: *Advances in Neural Information Processing Systems*. Vol. 32. Curran Associates, Inc., 2019.

[131] Tao Bai et al. “Recent Advances in Adversarial Training for Adversarial Robustness”. In: *Proc. 30th International Joint Conference on Artificial Intelligence*. Survey Track. International Joint Conferences on Artificial Intelligence Organization, Aug. 2021, pp. 4312–4321.

[132] Jörn-Henrik Jacobsen et al. *Exploiting Excessive Invariance Caused by Norm-Bounded Adversarial Robustness*. arXiv:1903.10484. 2019.

[133] Sandesh Kamath, Amit Deshpande, and K V Subrahmanyam. *Invariance vs Robustness of Neural Networks*. 2020. URL: `https://openreview.net/forum?id=HJxp9kBFDS`.

[134] Peter Bühlmann. “Invariance, Causality and Robustness”. In: *Statistical Science* 35.3 (2020), pp. 404–426.

[135] Shixiang Gu and Luca Rigazio. *Towards Deep Neural Network Architectures Robust to Adversarial Examples*. arXiv:1412.5068. 2014.

[136] Florian Tramèr et al. “Fundamental Tradeoffs between Invariance and Sensitivity to Adversarial Perturbations”. In: *International Conference on Machine Learning*. PMLR. 2020, pp. 9561–9571.

[137] Guoshen Yu and Jean-Michel Morel. “ASIFT: An Algorithm for Fully Affine Invariant Comparison”. In: *Image Processing On Line* 1 (2011), pp. 11–38.

[138] Evgeni Begelfor and Michael Werman. “Affine Invariance Revisited”. In: *IEEE Conference on Computer Vision and Pattern Recognition*. Vol. 2. 2006, pp. 2087–2094.

[139] Xifeng Gao et al. “A Robust High-Capacity Affine-Transformation-Invariant Scheme for Watermarking 3D Geometric Models”. In: *ACM Transactions on Multimedia Computing, Communications, and Applications* 8.2S (Sept. 2012).

[140] Namita Agarwal, Amit Kumar Singh, and Pradeep Kumar Singh. “Survey of Robust and Imperceptible Watermarking”. In: *Multimedia Tools and Applications* 78.7 (2019), pp. 8603–8633.

[141] Vidyasagar M Potdar, Song Han, and Elizabeth Chang. “A Survey of Digital Image Watermarking Techniques”. In: *3rd IEEE International Conference on Industrial Informatics*. 2005, pp. 709–716.

[142] Ron G Van Schyndel, Andrew Z Tirkel, and Charles F Osborne. “A Digital Watermark”. In: *Proc. 1st International Conference on Image Processing*. Vol. 2. 1994, 86–90 vol.2.

[143] C.I. Podilchuk and E.J. Delp. “Digital Watermarking: Algorithms and Applications”. In: *IEEE Signal Processing Magazine* 18.4 (2001), pp. 33–46.

[144] William Zhu, Clark Thomborson, and Fei-Yue Wang. “A Survey of Software Watermarking”. In: *International Conference on Intelligence and Security Informatics*. Springer. 2005, pp. 454–458.

[145] Chang-Min Chou and Din-Chang Tseng. “Affine-Transformation-Invariant Public Fragile Watermarking for 3D Model Authentication”. In: *IEEE Computer Graphics and Applications* 29.2 (2009), pp. 72–79.

[146] Hanzhou Wu et al. “Watermarking Neural Networks with Watermarked Images”. In: *IEEE Transactions on Circuits and Systems for Video Technology* 31.7 (2020), pp. 2591–2601.

[147] Yang Liu, Zhen Zhu, and Xiang Bai. “Wdnet: Watermark-decomposition Network for Visible Watermark Removal”. In: *Proc. IEEE/CVF Winter Conference on Applications of Computer Vision*. 2021, pp. 3685–3693.

[148] Danni Cheng et al. “Large-scale Visible Watermark Detection and Removal with Deep Convolutional Networks”. In: *Pattern Recognition and Computer Vision: First Chinese Conference*. Springer. 2018, pp. 27–40.

[149] Xinyun Chen et al. “Refit: A Unified Watermark Removal Framework for Deep Learning Systems with Limited Data”. In: *Proc. ACM Asia Conference on Computer and Communications Security*. 2021, pp. 321–335.

[150] Xinyun Chen et al. *Targeted Backdoor Attacks on Deep Learning Systems Using Data Poisoning*. arXiv:1712.05526. 2017.

[151] Tianyu Gu, Brendan Dolan-Gavitt, and Siddharth Garg. *Badnets: Identifying Vulnerabilities in the Machine Learning Model Supply Chain*. arXiv:1708.06733. 2017.

[152] Jialong Zhang et al. “Protecting Intellectual Property of Deep Neural Networks with Watermarking”. In: *Proc. Asia Conference on Computer and Communications Security*. 2018, pp. 159–172.

[153] Yossi Adi et al. “Turning Your Weakness into A Strength: Watermarking Deep Neural Networks by Backdooring”. In: *27th USENIX Security Symposium*. 2018, pp. 1615–1631.

[154] Ryota Namba and Jun Sakuma. “Robust Watermarking of Neural Network with Exponential Weighting”. In: *Proc. ACM Asia Conference on Computer and Communications Security*. 2019, pp. 228–240.

[155] Erwan Le Merrer, Patrick Perez, and Gilles Trédan. “Adversarial Frontier Stitching for Remote Neural Network Watermarking”. In: *Neural Computing and Applications* 32.13 (2020), pp. 9233–9244.

[156] Jan Auernhammer. “Human-centered AI: The Role of Human-centered Design Research in the Development of AI”. In: *DRS International Conference*. Aug. 2020.

[157] Mark O. Riedl. “Human-centered Artificial Intelligence and Machine Learning”. In: *Human Behavior and Emerging Technologies* 1.1 (2019), pp. 33–36.

[158] Gonzalo Ramos et al. “Emerging Perspectives in Human-Centered Machine Learning”. In: *Proc. ACM Conference Extended Abstracts on Human Factors in Computing Systems*. CHI’19. Glasgow, Scotland Uk: ACM, 2019, pp. 1–8.

[159] Ben Shneiderman. *Human-centered AI*. Oxford University Press, 2022.

[160] Marco Gillies et al. “Human-Centered Machine Learning”. In: *Proc. ACM Conference Extended Abstracts on Human Factors in Computing Systems*. CHI’16. San Jose, California, USA: ACM, 2016, pp. 3558–3565.

[161] Gary K. L. Tam, Vivek Kothari, and Min Chen. “An Analysis of Machine- and Human-Analytics in Classification”. In: *IEEE Transactions on Visualization and Computer Graphics* 23.1 (2017), pp. 71–80.

[162] Andrew M McNutt and Ravi Chugh. “Integrated Visualization Editing via Parameterized Declarative Templates”. In: *Proc. ACM Conference on Human Factors in Computing Systems*. ACM, May 2021.

[163] Jing Nathan Yan et al. “Silva: Interactively Assessing Machine Learning Fairness Using Causality”. In: *Proc. ACM Conference on Human Factors in Computing Systems*. New York, NY, USA: ACM, 2020, pp. 1–13.

[164] Tong Ge, Bongshin Lee, and Yunhai Wang. “CAST: Authoring Data-Driven Chart Animations”. In: *Proc. ACM Conference on Human Factors in Computing Systems*. May 2021.

[165] Yu Zhang, Bob Coecke, and Min Chen. “MI3: Machine-initiated Intelligent Interaction for Interactive Classification and Data Reconstruction”. In: *ACM Transactions on Interactive Intelligent Systems* 11.3-4 (2021), pp. 1–34.

[166] Yu Zhang et al. “Simulation-based Optimization of User Interfaces for Quality-assuring Machine Learning Model Predictions”. In: *ACM Transactions on Interactive Intelligent Systems* 14.1 (2024), pp. 1–32.

[167] Linhao Meng et al. “VADAF: Visualization for Abnormal Client Detection and Analysis in Federated Learning”. In: *ACM Transactions on Interactive Intelligent Systems* 11.3-4 (2021), pp. 1–23.

[168] Vinicius Segura and Simone DJ Barbosa. “BONNIE: Building online Narratives from Noteworthy Interaction Events”. In: *ACM Transactions on Interactive Intelligent Systems* 11.3-4 (2021), pp. 1–31.

[169] Xiaoyu Chen, Nathan Lau, and Ran Jin. “PRIME: A Personalized Recommender System for Information Visualization Methods via Extended Matrix Completion”. In: *ACM Transactions on Interactive Intelligent Systems* 11.1 (2021), pp. 1–30.

[170] Xingjun Li et al. “EDAssistant: Supporting Exploratory Data Analysis in Computational Notebooks with In Situ Code Search and Recommendation”. In: *ACM Transactions on Interactive Intelligent Systems* 13.1 (Mar. 2023).

[171] Dongning Yan and Li Chen. “The Influence of Personality Traits on User Interaction with Recommendation Interfaces”. In: *ACM Transactions on Interactive Intelligent Systems* 13.1 (Mar. 2023).

[172] Yuri Nakao et al. “Toward Involving End-Users in Interactive Human-in-the-Loop AI Fairness”. In: *ACM Transactions on Interactive Intelligent Systems* 12.3 (2022), pp. 1–30.

[173] Diana C. Hernandez-Bocanegra and Jürgen Ziegler. “Explaining Recommendations through Conversations: Dialog Model and the Effects of Interface Type and Degree of Interactivity”. In: *ACM Transactions on Interactive Intelligent Systems* 13.2 (Apr. 2023).

[174] Sara Bunian et al. “VINS: Visual Search for Mobile User Interface Design”. In: *Proc. ACM Conference on Human Factors in Computing Systems*. Yokohama, Japan: ACM, 2021.

[175] Carrie J Cai et al. “Human-centered Tools for Coping with Imperfect Algorithms During Medical Decision-making”. In: *Proc. ACM Conference on Human Factors in Computing Systems*. 2019, pp. 1–14.

[176] John R Thompson, Zhicheng Liu, and John Stasko. “Data Animator: Authoring Expressive Animated Data Graphics”. In: *Proc. ACM Conference on Human Factors in Computing Systems*. CHI’21. Yokohama, Japan: ACM, 2021.

[177] Ke Xu et al. “MTSeer: Interactive Visual Exploration of Models on Multivariate Time-Series Forecast”. In: *Proc. ACM Conference on Human Factors in Computing Systems*. CHI’21. Yokohama, Japan: ACM, 2021.

[178] Youwen Kang et al. “MetaMap: Supporting Visual Metaphor Ideation through Multi-Dimensional Example-Based Exploration”. In: *Proc. ACM Conference on Human Factors in Computing Systems*. CHI’21. Yokohama, Japan: ACM, 2021.

[179] Leslie Wöhler et al. “Towards Understanding Perceptual Differences between Genuine and Face-Swapped Videos”. In: *Proc. ACM Conference on Human Factors in Computing Systems*. CHI’21. Yokohama, Japan: ACM, 2021.

[180] Agathe Balayn et al. “How Can Explainability Methods Be Used to Support Bug Identification in Computer Vision Models?” In: *Proc. ACM Conference on Human Factors in Computing Systems*. CHI’22. New Orleans, LA, USA: ACM, 2022.

[181] Wencan Zhang, Mariella Dimiccoli, and Brian Y Lim. “Debiased-CAM to Mitigate Image Perturbations with Faithful Visual Explanations of Machine Learning”. In: *Proc. ACM Conference on Human Factors in Computing Systems*. CHI’22. New Orleans, LA, USA: ACM, 2022.

[182] Jonathan Dodge et al. “How the Experts Do It: Assessing and Explaining Agent Behaviors in Real-Time Strategy Games”. In: *Proc. ACM Conference on Human Factors in Computing Systems*. CHI’18. Montreal QC, Canada: ACM, 2018, pp. 1–12.

[183] Alexis Hiniker et al. “Anchored Audio Sampling: A Seamless Method for Exploring Children’s Thoughts During Deployment Studies”. In: *Proc. ACM Conference on Human Factors in Computing Systems*. CHI’19. Glasgow, Scotland Uk: ACM, 2019, pp. 1–13.

[184] Dominik Sacha et al. “VIS4ML: An Ontology for Visual Analytics Assisted Machine Learning”. In: *IEEE Transactions on Visualization and Computer Graphics* 25.1 (2018), pp. 385–395.

[185] Donghao Ren et al. “Squares: Supporting Interactive Performance Analysis for Multiclass Classifiers”. In: *IEEE Transactions on Visualization and Computer Graphics* 23.1 (2017), pp. 61–70.

[186] Yao Ming et al. “Understanding Hidden Memories of Recurrent Neural Networks”. In: *Proc. IEEE Conference on Visual Analytics Science and Technology*. IEEE. 2017, pp. 13–24.

[187] Fred Hohman et al. “Understanding and Visualizing Data Iteration in Machine Learning”. In: *Proc. ACM Conference on Human Factors in Computing Systems*. New York, NY, USA: ACM, 2020, pp. 1–13.

[188] Josua Krause et al. “A Workflow for Visual Diagnostics of Binary Classifiers Using Instance-Level Explanations”. In: *Proc. IEEE Conference on Visual Analytics Science and Technology*. 2017, pp. 162–172.

[189] Zelin Ye and Min Chen. “Visualizing Ensemble Predictions of Music Mood”. In: *IEEE Transactions on Visualization and Computer Graphics* 29.1 (2022), pp. 864–874.

[190] Mengchen Liu et al. “Towards Better Analysis of Deep Convolutional Neural Networks”. In: *IEEE Transactions on Visualization and Computer Graphics* 23.1 (2017), pp. 91–100.

[191] Paulo E. Rauber et al. “Visualizing the Hidden Activity of Artificial Neural Networks”. In: *IEEE Transactions on Visualization and Computer Graphics* 23.1 (2017), pp. 101–110.

[192] Nicola Pezzotti et al. “DeepEyes: Progressive Visual Analytics for Designing Deep Neural Networks”. In: *IEEE Transactions on Visualization and Computer Graphics* 24.1 (2018), pp. 98–108.

[193] Kanit Wongsuphasawat et al. “Visualizing Dataflow Graphs of Deep Learning Models in TensorFlow”. In: *IEEE Transactions on Visualization and Computer Graphics* 24.1 (2018), pp. 1–12.

[194] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. “Deep Learning”. In: *Nature* 521.7553 (May 2015), pp. 436–444.

[195] Guozheng Li et al. “GoTree: A Grammar of Tree Visualizations”. In: *Proc. ACM Conference on Human Factors in Computing Systems*. New York, NY, USA: ACM, 2020, pp. 1–13.

[196] Thomas Mühlbacher et al. “Treepod: Sensitivity-aware Selection of Pareto-optimal Decision Trees”. In: *IEEE Transactions on Visualization and Computer Graphics* 24.1 (2017), pp. 174–183.

[197] Soon Tee Teoh and Kwan-Liu Ma. “PaintingClass: Interactive Construction, Visualization and Exploration of Decision Trees”. In: *Proc. 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. Washington, D.C.: ACM, 2003, pp. 667–672.

[198] Shixia Liu et al. “Visual Diagnosis of Tree Boosting Methods”. In: *IEEE Transactions on Visualization and Computer Graphics* 24.1 (2018), pp. 163–173.

[199] Yuanzhe Jin et al. “Radial Icicle Tree (RIT): Node Separation and Area Constancy”. In: *IEEE Transactions on Visualization and Computer Graphics* (2023).

[200] Junpeng Wang, Shixia Liu, and Wei Zhang. “Visual Analytics for Machine Learning: A Data Perspective Survey”. In: *IEEE Transactions on Visualization and Computer Graphics* 30.12 (2024), pp. 7637–7656.

[201] Jun Yuan et al. “A Survey of Visual Analytics Techniques for Machine Learning”. In: *Computational Visual Media* 7.1 (2021), pp. 3–36.

[202] Fabian Beck et al. “A Taxonomy and Survey of Dynamic Graph Visualization”. In: *Computer Graphics Forum* 36.1 (2017), pp. 133–159.

[203] Michael Behrisch et al. “Matrix Reordering Methods for Table and Network Visualization”. In: *Computer Graphics Forum* 35.3 (2016), pp. 693–716.

[204] Fintan McGee et al. “The State of the Art in Multilayer Network Visualization”. In: *Computer Graphics Forum* 38.6 (2019), pp. 125–149.

[205] Martijn Tennekes and Min Chen. “Design Space of Origin-destination Data Visualization”. In: *Computer Graphics Forum* 40.3 (2021), pp. 323–334.

[206] Hans-Jorg Schulz. “Treevis.net: A Tree Visualization Reference”. In: *IEEE Computer Graphics and Applications* 31.6 (2011), pp. 11–15.

[207] Hans-Jorg Schulz, Steffen Hadlak, and Heidrun Schumann. “The Design Space of Implicit Hierarchy Visualization: A Survey”. In: *IEEE Transactions on Visualization and Computer Graphics* 17.4 (2010), pp. 393–411.

[208] Quang Vinh Nguyen and Mao Lin Huang. “A Space-optimized Tree Visualization”. In: *Proc. IEEE Symposium on Information Visualization*. IEEE. 2002, pp. 85–92.

[209] Desney Tan et al. “AdaptiviTree: Adaptive Tree Visualization for Tournament-style Brackets”. In: *IEEE Transactions on Visualization and Computer Graphics* 13.6 (2007), pp. 1113–1120.

[210] Chris Culy and Verena Lyding. “Double Tree: An Advanced KWIC Visualization for Expert Users”. In: *Proc. 14th International Conference on Information Visualisation*. IEEE. 2010, pp. 98–103.

[211] Adrian Rusu et al. “PieVis: Interactive Graph Visualization Using A Rings-Based Tree Drawing Algorithm for Children and Crust Display for Parents”. In: *Proc. 15th International Conference on Information Visualisation*. IEEE. 2011, pp. 465–470.

[212] Liang Gou and Xiaolong Luke Zhang. “Treenetviz: Revealing Patterns of Networks Over Tree Structures”. In: *IEEE Transactions on Visualization and Computer Graphics* 17.12 (2011), pp. 2449–2458.

[213] Yukio Sadahiro and Tetsuo Kobayashi. “Exploratory Analysis of Time Series Data: Detection of Partial Similarities, Clustering, and Visualization”. In: *Computers, Environment and Urban Systems* 45 (2014), pp. 24–33.

[214] Sheng-Jie Luo et al. “Ambiguity-free Edge-bundling for Interactive Graph Visualization”. In: *IEEE Transactions on Visualization and Computer Graphics* 18.5 (2011), pp. 810–821.

[215] H Paul Zellweger. “Tree Visualizations in Structured Data Recursively Defined by the Aleph Data Relation”. In: *Proc. 20th International Conference Information Visualisation*. IEEE. 2016, pp. 21–26.

[216] Zhen Li et al. “A Unified Understanding of Deep NLP Models for Text Classification”. In: *IEEE Transactions on Visualization and Computer Graphics* 28.12 (2022), pp. 4980–4994.

[217] Guangchuang Yu. “Using Ggtree to Visualize Data on Tree-like Structures”. In: *Bioinformatics* 69.1 (2020), e96.

[218] Joseph B. Kruskal and James M. Landwehr. “Icicle Plots: Better Displays for Hierarchical Clustering”. In: *The American Statistician* 37.2 (1983), pp. 162–168.

[219] Beat Kleiner and John A Hartigan. “Representing Points in Many Dimensions by Trees and Castles”. In: *Journal of the American Statistical Association* 76.374 (1981), pp. 260–269.

[220] Mei C Chuah. “Dynamic Aggregation with Circular Visual Designs”. In: *Proc. IEEE Information Visualization Symposium*. 1998, pp. 35–43.

[221] Fanny Chevalier, David Auber, and Alexandru Telea. “Structural Analysis and Visualization of C++ Code Evolution Using Syntax Trees”. In: *Proc. International Workshop on Principles of Software Evolution*. 2007, pp. 90–97.

[222] Huub van de Wetering, Nico Klaassen, and Michael Burch. “Space-Reclaiming Icicle Plots”. In: *Proc. IEEE Pacific Visualization Symposium*. 2020, pp. 121–130.

[223] Steffen Ducheyne. ““To Treat of the World” Paul Otlet’s Ontology and Epistemology and the Circle of Knowledge”. In: *Journal of Documentation* 65.2 (2009), pp. 223–244.

[224] American Society of Mechanical Engineers. *Hierarchical Sector Chart*. Ed. by Willard Cope Brinton. Brinton Associates, 1939, p. 89.

[225] Brian Scott Johnson. “Treemaps: Visualizing Hierarchical and Categorical Data”. PhD thesis. University of Maryland, 1993.

[226] Keith Andrews and Helmut Heidegger. “Information Slices: Visualising and Exploring Large Hierarchies Using Cascading, Semicircular Discs”. In: *Proc. IEEE Symposium on Information Visualization (Late Breaking Hot Topics)*. 1998, pp. 9–12.

[227] John Stasko and Eugene Zhang. “Focus+context Display and Navigation Techniques for Enhancing Radial, Space-filling Hierarchy Visualizations”. In: *Proc. IEEE Symposium on Information Visualization*. 2000, pp. 57–65.

[228] Ed H Chi et al. “Visualizing the Evolution of Web Ecologies”. In: *Proc. SIGCHI Conference on Human Factors in Computing Systems*. 1998, pp. 400–407.

[229] Tatiana Tekusova and Tobias Schreck. “Visualizing Time-dependent Data in Multivariate Hierarchic Plots - Design and Evaluation of An Economic Application”. In: *Proc. International Conference on Information Visualization*. 2008, pp. 143–150.

[230] Ho-Ching Lam and Ivo D. Dinov. “Hyperbolic Wheel: A Novel Hyperbolic Space Graph Viewer for Hierarchical Information Content”. In: *International Scholarly Research Notices* (2012), 609234 (10 pages).

[231] Linda Woodburn, Yalong Yang, and Kim Marriott. “Interactive Visualisation of Hierarchical Quantitative Data: An Evaluation”. In: *Proc. IEEE Visualization Conference*. IEEE. 2019, pp. 96–100.

[232] Jing Yang, Matthew O Ward, and Elke A Rundensteiner. “Interring: An Interactive Tool for Visually Navigating and Manipulating Hierarchical Structures”. In: *Proc. IEEE Symposium on Information Visualization*. 2002, pp. 77–84.

[233] Nick Cawthon and Andrew Vande Moere. “The Effect of Aesthetic on the Usability of Data Visualization”. In: *Proc. 11th International Conference on Information Visualization*. IEEE. 2007, pp. 637–648.

[234] Sujay Muramalla et al. “Radial vs. Rectangular: Evaluating Visualization Layout Impact on User Task Performance of Hierarchical Data”. In: *International Journal of Computer Science and Information Systems* 12.2 (2017), pp. 17–31.

[235] Boyan Zheng and Filip Sadlo. “On the Visualization of Hierarchical Multivariate Data”. In: *Proc. IEEE Pacific Visualization Symposium*. IEEE. 2021, pp. 136–145.

[236] Edward R Tufte and Peter R Graves-Morris. *The Visual Display of Quantitative Information*. 2nd. Graphics Press, 2001.

[237] Gordon Kindlmann and Carlos Scheidegger. "An Algebraic Process for Visualization Design". In: *IEEE Transactions on Visualization and Computer Graphics* 20.12 (2014), pp. 2181–2190.

[238] Michael Correll et al. "Looks Good to Me: Visualizations as Sanity Checks". In: *IEEE Transactions on Visualization and Computer Graphics* 25.1 (2018), pp. 830–839.

[239] Vito Di Maio and Petr Lánskỳ. "Area Perception in Simple Geometrical Figures". In: *Perceptual and Motor Skills* 71.2 (1990), pp. 459–466.

[240] Michael J Morgan. "The Visual Computation of 2-D Area by Human Observers". In: *Vision research* 45.19 (2005), pp. 2564–2570.

[241] Jacob Nachmias. "Judging Spatial Properties of Simple Figures". In: *Vision Research* 48.11 (2008), pp. 1290–1296.

[242] Syed Saiden Abbas, Tjeerd MH Dijkstra, and Tom Heskes. "A Direct Comparison of Visual Discrimination of Shape and Size on A Large Range of Aspect Ratios". In: *Vision Research* 91.10 (2013), pp. 84–92.

[243] Drew Skau and Robert Kosara. "Arcs, Angles, or Areas: Individual Data Encodings in Pie and Donut Charts". In: *Computer Graphics Forum* 35.3 (2016), pp. 121–130.

[244] Robert Kosara. "Evidence for Area as the Primary Visual Cue in Pie Charts". In: *Proc. IEEE Visualization Conference Short Papers*. IEEE. 2019, pp. 101–105.

[245] Guo Qi and Zhang Jing. "The Quantitative Research on Length and Area Perception: A Guidance on Shape Encoding in Visual Interface". In: *Displays* 75 (2022), p. 102325.

[246] Danielle Albers Szafir. "Modeling Color Difference for Visualization Design". In: *IEEE Transactions on Visualization and Computer Graphics* 24.1 (2018), pp. 392–401.

[247] Karen B Schloss et al. "Mapping Color to Meaning in Colormap Data Visualizations". In: *IEEE Transactions on Visualization and Computer Graphics* 25.1 (2018), pp. 810–819.

[248] Aritra Dasgupta, Min Chen, and Robert Kosara. "Conceptualizing Visual Uncertainty in Parallel Coordinates". In: *Computer Graphics Forum* 31.3 (2012), pp. 1015–1024.

[249] Rassadarie Kanjanabose, Alfie Abdul-Rahman, and Min Chen. "A Multi-task Comparative Study on Scatter Plots and Parallel Coordinates Plots". In: *Computer Graphics Forum* 34.3 (2015), pp. 261–270.

[250] Qianwen Wang et al. "HypoML: Visual Analysis for Hypothesis-based Evaluation of Machine Learning Models". In: *IEEE Transactions on Visualization and Computer Graphics* 27.2 (2020), pp. 1417–1426.

[251] Min Chen and Darren J Edwards. ""Isms" in Visualization". In: *Foundations of Data Visualization*. Springer, 2020, pp. 225–241.

[252] Min Chen and Mateu Sbert. "A Bounded Measure for Estimating the Benefit of Visualization (Part I): Case Studies and Empirical Evaluation". In: *Entropy* 24.2 (2022), p. 228.

[253] Min Chen et al. "A Bounded Measure for Estimating the Benefit of Visualization (Part II): Case Studies and Empirical Evaluation". In: *Entropy* 24.2 (2022), p. 282.

[254] Min Chen and Amos Golan. "What May Visualization Processes Optimize?" In: *IEEE Transactions on Visualization and Computer Graphics* 22.12 (2016), pp. 2619–2632.

[255] Min Chen and David S Ebert. "An Ontological Framework for Supporting the Design and Evaluation of Visual Analytics Systems". In: *Computer Graphics Forum* 38.3 (2019), pp. 131–144.

[256] Weikai Yang et al. "Foundation Models Meet Visualizations: Challenges and Opportunities". In: *Computational Visual Media* 10.3 (2024), pp. 399–424.

[257] Angelos Chatzimparmpas, Kostiantyn Kucher, and Andreas Kerren. "Visualization for Trust in Machine Learning Revisited: The State of the Field in 2023". In: *IEEE Computer Graphics and Applications* 44.3 (2024), pp. 99–113.

[258] Roy Okonkwo et al. "Explainable Artificial Intelligence (AI) through human-AI collaborative Frameworks: Quantifying Trust and Interpretability in High-stakes Decisions". In: *Computer Science & IT Research Journal* 6 (June 2025), pp. 333–354.

[259] Zeyang Huang et al. "VA+ Embeddings STAR: A State-of-the-Art Report on the Use of Embeddings in Visual Analytics". In: *Computer Graphics Forum*. Vol. 42. 3. Wiley Online Library. 2023, pp. 539–571.

[260] Xingbo Wang et al. "M2lens: Visualizing and Explaining Multimodal Models for Sentiment Analysis". In: *IEEE Transactions on Visualization and Computer Graphics* 28.1 (2021), pp. 802–812.

[261] Xiaoyu Zhang et al. "SliceTeller: A Data Slice-driven Approach for Machine Learning Model Validation". In: *IEEE Transactions on Visualization and Computer Graphics* 29.1 (2022), pp. 842–852.

[262] Changjian Chen et al. "OoDAnalyzer: Interactive Analysis of Out-of-distribution Samples". In: *IEEE Transactions on Visualization and Computer Graphics* 27.7 (2020), pp. 3335–3349.

[263] Judy Borowski et al. *Exemplary Natural Images Explain CNN Activations Better than Feature Visualizations*. arXiv:2010.12606. 2020.

[264] Qiaomu Shen et al. "Visual Interpretation of Recurrent Neural Network on Multi-dimensional Time-series Forecast". In: *Proc. IEEE Pacific Visualization Symposium*. IEEE. 2020, pp. 61–70.

[265] Junpeng Wang et al. "Visual Analytics for RNN-based Deep Reinforcement Learning". In: *IEEE Transactions on Visualization and Computer Graphics* 28.12 (2021), pp. 4141–4155.

[266] Georgios Sarailidis, Thorsten Wagener, and Francesca Pianosi. "Integrating Scientific Knowledge into Machine Learning using Interactive Decision Trees". In: *Computers & Geosciences* 170 (2023), p. 105248.

[267] Angelos Chatzimparmpas, Rafael M Martins, and Andreas Kerren. “VisRuler: Visual Analytics for Extracting Decision Rules from Bagged and Boosted Decision Trees”. In: *Information Visualization* 22.2 (2023), pp. 115–139.

[268] Minjae Shin et al. “Development of An HVAC System Control Method Using Weather Forecasting Data with Deep Reinforcement Learning Algorithms”. In: *Building and Environment* 248 (2024), p. 111069.

[269] Yanjie Ze et al. “Visual Reinforcement Learning with Self-supervised 3D Representations”. In: *IEEE Robotics and Automation Letters* 8.5 (2023), pp. 2890–2897.

[270] Yannick Metz et al. “VISITOR: Visual Interactive State Sequence Exploration for Reinforcement Learning”. In: *Computer Graphics Forum* 42 (2023), pp. 397–408.

[271] Francisco Calvin Arnel Ferano, Amalia Zahra, and Gede Putra Kusuma. “Stacking Ensemble Learning for Optical Music Recognition”. In: *Bulletin of Electrical Engineering and Informatics* 12.5 (2023), pp. 3095–3104.

[272] Ken Lau. *Random Forest Ensemble Visualization*. `https://api.semanticscholar.org/CorpusID:15743377`. 2014.

[273] Mohammad Alharbi and Robert S Laramee. “SoS TextVis: A Survey of Surveys on Text Visualization”. In: *CGVC*. 2018, pp. 143–152.

[274] Mohammad Alharbi and Robert S Laramee. “SoS Textvis: An extended Survey of Surveys on Text Visualization”. In: *Computers* 8.1 (2019), p. 17.

[275] Shixia Liu et al. “Bridging Text Visualization and Mining: A Task-driven Survey”. In: *IEEE Transactions on Visualization and Computer Graphics* 25.7 (2018), pp. 2482–2504.

[276] Maximilian T Fischer et al. *Communication Analysis through Visual Analytics: Current Practices, Challenges, and New Frontiers*. IEEE, 2022.

[277] Owen Kaser and Daniel Lemire. *Tag-Cloud Drawing: Algorithms for Cloud Visualization*. arXiv:cs/0703109. 2007.

[278] Weiwei Cui et al. “Context Preserving Dynamic Word Cloud Visualization”. In: *Proc. IEEE Pacific Visualization Symposium*. IEEE. 2010, pp. 121–128.

[279] Quoc Le and Tomas Mikolov. “Distributed Representations of Sentences and Documents”. In: *International Conference on Machine Learning*. PMLR. 2014, pp. 1188–1196.

[280] Emden R Gansner, Yifan Hu, and Stephen North. “Visualizing Streaming Text Data with Dynamic Graphs and Maps”. In: *International Symposium on Graph Drawing*. Springer. 2012, pp. 439–450.

[281] Jack Park and Sam Hunting. *XML Topic Maps: Creating and Using Topic Maps for the Web*. Addison-Wesley Professional, 2003.

[282] Weiwei Cui et al. “TextFlow: Towards Better Understanding of Evolving Topics in Text”. In: *IEEE Transactions on Visualization and Computer Graphics* 17.12 (2011), pp. 2412–2421.

[283] Shixia Liu et al. “Tiara: Interactive, Topic-based Visual Text Summarization and Analysis”. In: *ACM Transactions on Intelligent Systems and Technology* 3.2 (2012), pp. 1–28.

[284] Yingcai Wu et al. “StreamExplorer: A Multi-stage System for Visually Exploring Events in Social Streams”. In: *IEEE Transactions on Visualization and Computer Graphics* 24.10 (2017), pp. 2758–2772.

[285] Dongning Luo et al. “Eventriver: Visually Exploring Text Collections with Temporal References”. In: *IEEE Transactions on Visualization and Computer Graphics* 18.1 (2010), pp. 93–105.

[286] Heekyong Park and Jinwook Choi. “V-model: A New Innovative Model to Chronologically Visualize Narrative Clinical Texts”. In: *Proc. SIGCHI Conference on Human Factors in Computing Systems*. 2012, pp. 453–462.

[287] Matthew Brehmer et al. “Overview: The Design, Adoption, and Analysis of A Visual Document Mining Tool for Investigative Journalists”. In: *IEEE Transactions on Visualization and Computer Graphics* 20.12 (2014), pp. 2271–2280.

[288] Shixia Liu et al. “Online Visual Analytics of Text Streams”. In: *IEEE Transactions on Visualization and Computer Graphics* 22.11 (2015), pp. 2451–2466.

[289] Wenwen Dou and Shixia Liu. “Topic-and Time-oriented Visual Text Analysis”. In: *IEEE Computer Graphics and Applications* 36.4 (2016), pp. 8–13.

[290] Alfie Abdul-Rahman et al. “Constructive Visual Analytics for Text Similarity Detection”. In: *Computer Graphics Forum* 36.1 (2017), pp. 237–248.

[291] Mennatallah El-Assady et al. “ThreadReconstructor: Modeling Reply-chains to Untangle Conversational Text through Visual Analytics”. In: *Computer Graphics Forum* 37 (2018), pp. 351–365.

[292] Mennatallah El-Assady et al. “Lingvis.io-A Linguistic Visual Analytics Framework”. In: *Proc. 57th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*. 2019, pp. 13–18.

[293] Mennatallah El-Assady et al. “ConToVi: Multi-party Conversation Exploration Using Topic-space views”. In: *Computer Graphics Forum* 35 (2016), pp. 431–440.

[294] Mennatallah El-Assady et al. “NEREx: Named-Entity Relationship Exploration in Multi-Party Conversations”. In: *Computer Graphics Forum* 36 (2017), pp. 213–225.

[295] Moshe Schorr et al. *ViS-\’A-ViS: Detecting Similar Patterns in Annotated Literary Text*. arXiv:2009.02063. 2020.

[296] Johannes Knittel et al. “Real-time Visual Analysis of High-volume Social Media Posts”. In: *IEEE Transactions on Visualization and Computer Graphics* 28.1 (2021), pp. 879–889.

[297] Rita Sevastjanova et al. “Visinreport: Complementing Visual Discourse Analytics through Personalized Insight Reports”. In: *IEEE Transactions on Visualization and Computer Graphics* 28.12 (2021), pp. 4757–4769.

[298] Maximilian T Fischer et al. “CommAID: Visual Analytics for Communication Analysis Through Interactive Dynamics Modeling”. In: *Computer Graphics Forum* 40 (2021), pp. 25–36.

[299] MK Aravindan et al. “Evaluation of Machine Learning Algorithms for Text Classification Tasks”. In: *3rd International Conference on Smart Generation Computing, Communication and Networking*. IEEE. 2023, pp. 1–7.

[300] Jipeng Qiang et al. “Short Text Topic Modeling Techniques, Applications, and Performance: A Survey”. In: *IEEE Transactions on Knowledge and Data Engineering* 34.3 (2020), pp. 1427–1445.

[301] Nilaa Raghunathan and Kandasamy Saravanakumar. “Challenges and Issues in Sentiment Analysis: A Comprehensive Survey”. In: *IEEE Access* 11 (2023), pp. 69626–69642.

[302] Yuming Li et al. “An Explanation Framework and Method for AI-based Text Emotion Analysis and Visualisation”. In: *Decision Support Systems* 178 (2024), p. 114121.

[303] Zehan Wang. “Information Extraction and Knowledge Map Construction based on Natural Language Processing”. In: *Frontiers in Computing and Intelligent Systems* 7 (2024), pp. 47–49.

[304] Massimo La Morgia et al. “Translated Texts under the Lens: From Machine Translation Detection to Source Language Identification”. In: *International Symposium on Intelligent Data Analysis*. Springer, 2023, pp. 222–235.

[305] Wafaa S El-Kassas et al. “Automatic Text Summarization: A Comprehensive Survey”. In: *Expert Systems with Applications* 165 (2021), p. 113679.

[306] Evan N Crothers, Nathalie Japkowicz, and Herna L Viktor. “Machine-generated Text: A Comprehensive Survey of Threat Models and Detection Methods”. In: *IEEE Access* 11 (2023), pp. 70977–71002.

[307] Pravallika Etoori, Manoj Chinnakotla, and Radhika Mamidi. “Automatic Spelling Correction for Resource-scarce Languages Using Deep Learning”. In: *Proc. ACL Student Research Workshop*. 2018, pp. 146–152.

[308] Jacob Devlin et al. “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding”. In: *Proc. the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. 2019, pp. 4171–4186.

[309] Robert Dale. “GPT-3: What’s It Good for?” In: *Natural Language Engineering* 27.1 (2021), pp. 113–118.

[310] Victor Schetinger et al. “Doom or Deliciousness: Challenges and Opportunities for Visualization in the Age of Generative Models”. In: *Computer Graphics Forum* 42 (2023), pp. 423–435.

[311] Mennatallah El-Assady et al. “Progressive Learning of Topic Modeling Parameters: A Visual Analytics Framework”. In: *IEEE Transactions on Visualization and Computer Graphics* 24.1 (2017), pp. 382–391.

[312] Rita Sevastjanova et al. “Visual Comparison of Language Model Adaptation”. In: *IEEE Transactions on Visualization and Computer Graphics* 29.1 (2022), pp. 1178–1188.

[313] Rita Sevastjanova et al. “Visual Comparison of Text Sequences Generated by Large Language Models”. In: *IEEE Visualization in Data Science*. IEEE. 2023, pp. 11–20.

[314] Markus Bayer, Marc-André Kaufhold, and Christian Reuter. “A Survey on Data Augmentation for Text Classification”. In: *ACM Computing Surveys* 55.7 (2022), pp. 1–39.

[315] Anna Richter et al. “A Survey on Multivariate Time Series Imputation using Adversarial Learning”. In: *IEEE Access* (2024).

[316] Lin Long et al. “On LLMs-driven Synthetic Data Generation, Curation, and Evaluation: A Survey”. In: *Findings of the Association for Computational Linguistics ACL 2024*. 2024, pp. 11065–11082.

[317] Jiacheng Ye et al. “ProGen: Progressive Zero-shot Dataset Generation via In-context Feedback”. In: *Findings of the Association for Computational Linguistics: EMNLP 2022*. 2022, pp. 3671–3683.

[318] Gaurav Sahu et al. “Data Augmentation for Intent Classification with Off-the-shelf Large Language Models”. In: *Proc. 4th Workshop on NLP for Conversational AI*. 2022, pp. 47–57.

[319] Jiahui Gao et al. “Self-guided Noise-free Data Generation for Efficient Zero-shot Learning”. In: *International Conference on Learning Representations*. 2023.

[320] Ryan Smith et al. “Language Models in the Loop: Incorporating Prompting into Weak Supervision”. In: *ACM/JMS Journal of Data Science* 1.2 (2024), pp. 1–30.

[321] Yuanzhe Jin, Adrian Carrasco-Revilla, and Min Chen. *iGAiVA: Integrated Generative AI and Visual Analytics in A Machine Learning Workflow for Text Classification*. 2024.

[322] Florian Heimerl et al. “Visual Classifier Training for Text Document Retrieval”. In: *IEEE Transactions on Visualization and Computer Graphics* 18.12 (2012), pp. 2839–2848.

[323] Fabian Sperrle et al. “VIANA: Visual Interactive Annotation of Argumentation”. In: *Proc. IEEE Conference on Visual Analytics Science and Technology*. IEEE. 2019, pp. 11–22.

[324] Mennatallah El-Assady et al. “Visual Analytics for Topic Model Optimization Based on User-steerable Speculative Execution”. In: *IEEE Transactions on Visualization and Computer Graphics* 25.1 (2018), pp. 374–384.

[325] Mennatallah El-Assady et al. “Semantic Concept Spaces: Guided Topic Model Refinement Using Word-embedding Projections”. In: *IEEE Transactions on Visualization and Computer Graphics* 26.1 (2019), pp. 1001–1011.

[326] Fabian Sperrle et al. “Learning Contextualized User Preferences for Co-Adaptive Guidance in Mixed-Initiative Topic Model Refinement”. In: *Computer Graphics Forum* 40 (2021), pp. 215–226.

[327] Rita Sevastjanova et al. “Explaining Contextualization in Language Models Using Visual Analytics”. In: *Proc. 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing*. 2021, pp. 464–476.

[328] Rita Sevastjanova et al. “Questioncomb: A Gamification Approach for the Visual Explanation of Linguistic Phenomena through Interactive Labeling”. In: *ACM Transactions on Interactive Intelligent Systems* 11.3-4 (2021), pp. 1–38.

[329] Gregg Henriques, Kimberly Kleinman, and Craig Asselin. “The Nested Model of Well-being: A Unified Approach”. In: *Review of General Psychology* 18.1 (2014), pp. 7–18.

[330] Min Chen. “Cost-Benefit Analysis of Data Intelligence – Its Broader Interpretations”. In: *Advances in Info-Metrics: Information and Information Processing across Disciplines*. Oxford University Press, 2020, pp. 433–463.

[331] Min Chen et al. “From Data Analysis and Visualization to Causality Discovery”. In: *Computer* 44.10 (2011), pp. 84–87.

[332] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. *Neural Machine Translation by Jointly Learning to Align and Translate*. arXiv:1409.0473. 2015.

[333] Michael Sedlmair, Miriah Meyer, and Tamara Munzner. “Design Study Methodology: Reflections from the Trenches and the Stacks”. In: *IEEE Transactions on Visualization and Computer Graphics* 18.12 (2012), pp. 2431–2440.

[334] Shing Chan et al. “CAPTURE-24: A Large Dataset of Wrist-worn Activity Tracker Data Collected in the Wild for Human Activity Recognition”. In: *Scientific Data* 11.1 (2024), p. 1135.

[335] Mi-Young Kim et al. “A multi-component Framework for the Analysis and Design of Explainable Artificial Intelligence”. In: *Machine Learning and Knowledge Extraction* 3.4 (2021), pp. 900–921.

[336] Tongzhou Wang et al. *Dataset Distillation*. arXiv:1811.10959. 2018.

[337] Kui Ren et al. “Adversarial Attacks and Defenses in Deep Learning”. In: *Engineering* 6.3 (2020), pp. 346–360.

[338] Anirban Chakraborty et al. *Adversarial Attacks and Defences: A Survey*. arXiv:1810.00069. 2018.

[339] Zhipeng Chen et al. “A Gradient-based Pixel-domain Attack against SVM Detection of Global Image Manipulations”. In: *Proc. IEEE Workshop on Information Forensics and Security*. 2017, pp. 1–6.

[340] Nitheesh Chandra Yaratapalli et al. “Pixel-based Attack on Odenet Classifiers”. In: *Proc. ICT Systems and Sustainability*. Springer, 2020, pp. 629–637.

[341] Andrzej Maćkiewicz and Waldemar Ratajczak. “Principal Components Analysis (PCA)”. In: *Computers Geosciences* 19.3 (1993), pp. 303–342.

[342] Jonathan Bodine and Dorit S. Hochbaum. “A Better Decision Tree: The Max-Cut Decision Tree with Modified PCA Improves Accuracy and Running Time”. In: *SN Computer Science* 3.4 (May 2022), p. 313.

[343] Navneet Dalal and Bill Triggs. “Histograms of Oriented Gradients for Human Detection”. In: *Proc. IEEE Conference on Computer Vision and Pattern Recognition*. Vol. 1. IEEE. 2005, pp. 886–893.

[344] David G. Lowe. “Distinctive Image Features from Scale-Invariant Keypoints”. In: *International Journal of Computer Vision* 60.2 (Nov. 2004), pp. 91–110.

[345] Alex Krizhevsky, Geoffrey Hinton, et al. “Learning Multiple Layers of Features from Tiny Images”. In: *online* (2009).

[346] Han Xiao, Kashif Rasul, and Roland Vollgraf. *Fashion-MNIST: A Novel Image Dataset for Benchmarking Machine Learning Algorithms*. arXiv:1708.07747. 2017.

[347] Yann LeCun et al. "Gradient-based Learning Applied to Document Recognition". In: *Proc. IEEE* 86.11 (2002), pp. 2278–2324.

[348] Sizhe Chen et al. "Universal Adversarial Attack on Attention and the Resulting Dataset DAmageNet". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 44.4 (2022), pp. 2188–2197.

[349] Hugo Touvron et al. "Fixing the Train-test Resolution Discrepancy". In: *Advances in Neural Information Processing Systems* 32 (2019).

[350] Congzheng Song, Thomas Ristenpart, and Vitaly Shmatikov. "Machine Learning Models that Remember Too Much". In: *Proc. ACM SIGSAC Conference on Computer and Communications Security*. 2017, pp. 587–601.

[351] Tero Karras et al. "Training Generative Adversarial Networks with Limited Data". In: *Advances in Neural Information Processing Systems* 33 (2020), pp. 12104–12114.

[352] Nancy F Butte, Ulf Ekelund, and Klaas R Westerterp. "Assessing Physical Activity Using Wearable Monitors: Measures of Physical Activity". In: *Medicine and Science in Sports and Exercise* 44.1 Suppl 1 (2012), S5–S12.

[353] Luigi Bibbò, Riccardo Carotenuto, and Francesco Della Corte. "An Overview of Indoor Localization System for Human Activity Recognition (HAR) in Healthcare". In: *Sensors* 22.21 (2022), p. 8119.

[354] Zijie Chen et al. "MP-HAR: A Novel Motion-powered Real-time Human Activity Recognition System". In: *IEEE Internet of Things Journal* (2023).

[355] Liufeng Fan et al. "Context-Aware Human Activity Recognition (CA-HAR) Using Smartphone Built-In Sensors". In: *International Conference on Advances in Mobile Computing and Multimedia Intelligence*. Springer. 2022, pp. 108–121.

[356] Chao-Lung Yang et al. "HAR-time: Human Action Recognition with Time Factor Analysis on Worker Operating Time". In: *International Journal of Computer Integrated Manufacturing* 36.8 (2023), pp. 1219–1237.

[357] Aleksej Logacjov et al. "HARTH: A Human Activity Recognition Dataset for Machine Learning". In: *Sensors* 21.23 (2021), p. 7853.

[358] Saurabh Gupta. "Deep Learning Based Human Activity Recognition (HAR) Using Wearable Sensor Data". In: *International Journal of Information Management Data Insights* 1.2 (2021), p. 100046.

[359] Dafne van Kuppevelt et al. "Segmenting Accelerometer Data from Daily Life with Unsupervised Machine Learning". In: *PloS one* 14.1 (2019), e0208692.

[360] Sigrid van Hoek et al. *Comparing Activity Trackers to Investigate Physical Activity: Balancing Quality, Costs, and Usability*. Statistics Netherlands, 2022.

[361] Le Wang et al. "Review of Classification Methods on Unbalanced Data Sets". In: *IEEE Access* 9 (2021), pp. 64606–64628.

[362] V. Roshan Joseph. "Optimal Ratio for Data Splitting". In: *Statistical Analysis and Data Mining: The ASA Data Science Journal* 15.4 (Apr. 2022), pp. 531–538.

[363] Norvig Peter and Russell Stuart Artificial Intelligence. *A Modern Approach*. Pearson Education, USA, 2021.

[364] Yuanzhe Jin and Min Chen. *TA3: Testing Against Adversarial Attacks on Machine Learning Models*. arXiv:2410.05334. 2024.

[365] Andreas Holzinger. "Interactive Machine Learning for Health Informatics: When Do We Need the Human-in-the-loop?" In: *Brain Informatics* 3.2 (2016), pp. 119–131.

[366] Dongxian Wu, Shu-Tao Xia, and Yisen Wang. "Adversarial Weight Perturbation Helps Robust Generalization". In: *Advances in Neural Information Processing Systems* 33 (2020), pp. 2958–2969.