Faculty of Science, Technology and Communication

Towards an Interactive and Explainable Ticketing Systems Leveraging Large Language Models

Thesis Submitted in Partial Fulfillment of the Requirements for the Degree of Master in Information and Computer Sciences

Author
Esada Licina

Student ID
019152920B

Supervisor
Prof. Christoph Schommer

Reviewer
Prof. Gilbert Fridgen

Advisor
Dr. Igor Tchappi

August 26, 2024

Acknowledgement

The successful completion of this thesis was reached by the support of various people. My sincere appreciation goes to my supervisor Christoph Schommer, who accepted my Master' Thesis and made it all possible. I also want to express my profound gratitude to my advisor Igor Tchappi for letting me work on such an interesting topic and encouraging me during the whole learning phase. He taught me a lot about science and research, giving me his trust in submitting two of my papers at different conferences. I'm sincerely thankful for having Gilbert Fridgen as my reviewer and thankful for his time to evaluate my work. Finally, I would like to thank all my friends and family for supporting me during the long nights and sunny days with hard times of work pressure, but the biggest thanks go to myself, for not giving up on the hard times and doing my best.

Statement on the Use of AI Tools

I would like to indicate that I used Grammarly and ChatGPT 3 during the work process of this thesis. Grammarly improved the grammar, clarity, and punctuation in different parts of the text and did not create any new ideas, knowledge, or facts. In most cases, Grammarly provided a reliable writing assistant and tutor which assisted in refining writing skills and giving a high quality to this thesis. ChatGPT, on the other hand, was used to understand the basics of different topics from the thesis, simplifying the research and analysis of different papers. I hereby certify that all research and analyses have been done, and every content and idea included in this thesis is purely mine or appropriately referenced to the sources.

A Poetry

This Thesis reached the end of its essence,
Started with Christoph Schommer's presence.

Igor Tchappi, my supervisor, his wisdom so vast,
Taught me learning that would surely last.

Another thanks goes to Gilbert Fridges' time,
Reviewing and Evaluating my works' climb.

AI, my virtual friend with great insights,
Always there through days and nights.

Grammarly polished every comma and point,
Though, no new ideas were brought in and joint.

ChatGPT, too, gave a digital hand,
Simplifying topics and knowledge expand.

Friends and family, you were the glue,
But the greatest thanks? Belongs to me, so true.

Abstract

A Helpdesk Ticketing System (TS) is used by organizations, to keep efficient support to their services for users. There are many types of helpdesk-related tickets, such as Account and Access Management, Inquiries and Information, Incident Reports, and Complaints that get handled by a support agent. Current state of art papers [28] [3] have discussed the advantages and gaps of automated ticket routing and resolving. Mainly focusing on automating the process with Machine Learning (ML) methods and transformers such as ChatGPT, neglecting the user experience through this modern solution.

This thesis provides a user-centric study that investigates the impact of transformer models on a helpdesk TS, aiming to enhance interactive user experiences. It employed a combination of different ticket classification methods, Explainable Artificial Intelligence (XAI), community engagement via Discord, chatbots, and user studies, to improve the helpdesk TS compared to a traditional helpdesk. Looking at the technologic-centric design, the study's results indicated that traditional ML models, such as Support Vector Machine (SVM) with Term Frequency-Inverse Document Frequency (TF-IDF), achieved higher performance than advanced models in classifying user finance complaints. On the user-centric design, the study showed that users tend to prefer an interactive TS, compared to a traditional one where the users have to manage the ticket creation and submission on their own.

Additionally, the thesis showed that while advanced Artificial Intelligence (AI) like Large Language Model (LLM) significantly enhances user satisfaction on different TS parts, simplicity remains crucial in user experience. These findings suggest the need for a broader audience for user studies across different organizations and the integration of advanced AI technology in various ticketing tasks, including automated ticket answering.

Keywords: Helpdesk Ticketing System; Automated Ticketing; Explainable AI

ACM Classification Keywords 5.5: Multi-Agent System; Natural Language Processing; Classification Methods; User Centric Design; User Studies

Contents

Abstract **4**

List of Acronyms **10**

Introduction **11**
Motivation . . . 11
Research Goals . . . 12
Structure of the Thesis . . . 13

1 State of the Art: Current Progress and Challenges **14**
1.1 Current Techniques in Text Representation . . . 14
1.2 Review of Machine Learning Models . . . 14
1.3 Review of Deep Learning Models . . . 15
1.4 Review of Large Language Models . . . 15
1.5 Explainable Artificial Intelligence . . . 15
1.6 Community-Driven Support . . . 16
1.7 Gaps in the Literature . . . 17

2 Foundation of Artificial Intelligence **18**
2.1 Neural Network Structure . . . 18
2.2 The Evolution and Layers of AI . . . 23
2.2.1 Machine Learning . . . 24
2.2.2 Deep Learning . . . 25
2.2.3 Transformers . . . 27
2.3 Comprehensive Guide to Natural Language Processing . . . 29
2.3.1 Preprocessing . . . 29
2.3.2 Text Representation . . . 32

3 Ticket Classification with Natural Language Processing **39**
3.1 Overview of Ticket Classification . . . 39
3.2 Customer Complaints Dataset . . . 40
3.2.1 Overview of the Original Customer Complaints Dataset . . . 40
3.2.2 Data Analysis and Cleaning . . . 41
3.3 Classification Approach for ML Models . . . 44
3.4 Classification Approach for DL Models . . . 47
3.5 Classification Approach for LL Models . . . 47
3.6 Experimental Environment and Model Performance . . . 48
3.6.1 Experiment Configuration . . . 48
3.6.2 Performance Evaluation of Machine Learning Models . . . 48
3.6.3 Performance Evaluation of Deep Learning Models . . . 51

3.6.4 Performance Evaluation of Large Language Models 53
3.7 Conclusions and Insights 54

4 Explainable Artificial Intelligence 56
4.1 Introduction to XAI 56
4.1.1 Key Benefits of XAI 57
4.1.2 Addressing the Complexities of XAI 58
4.1.3 Exploring Different Approaches to XAI 59
4.2 Application in the Ticketing System 60
4.2.1 Analyze Wrongly Classified Complaints with XAI 61
4.2.2 Conclusion 63

5 Interactive Ticketing System with Multiple Agents 65
5.1 Streamlined Ticketing Workflow 65
5.2 User-Agent Protocols 67
5.2.1 Discord Ticket-Bot Agent 67
5.2.2 ChatBot Agent 68
5.2.3 Helpdesk Agent 69
5.3 Study Design and Hypotheses 70
5.3.1 Experiment Configuration 70
5.3.2 User Interface 70
5.3.3 Support Agent Interface 72
5.4 Findings and Observations 73
5.4.1 User Test Outcomes 74
5.4.2 Support Agent Test Results 78
5.5 Discussion 78

Conclusion and Future Work 81

Bibliography 81

Appendices 85

ACM Classification 87

List of Figures

2.1 NN Architecture . . . 19
2.2 Confusion Matrix Representation . . . 21
2.3 Models' Prediction . . . 21
2.4 FN and FP Demonstration . . . 22
2.5 TN and TP Demonstration . . . 22
2.6 Layers of AI . . . 23
2.7 DTs Architecture . . . 25
2.8 CNN Architecture . . . 26
2.9 ML vs. DL . . . 26
2.10 Transformer Architecture . . . 27
2.11 Segmentation Method . . . 29
2.12 Tokenizing Method . . . 30
2.13 Stop Words Method . . . 30
2.14 Stemming Method . . . 30
2.15 Lemmatization Method . . . 31
2.16 Speech Tagging Method . . . 31
2.17 Named Entity Tagging Method . . . 32
2.18 Embedding Approach with NN . . . 37

3.1 Timely Ticket Resolution . . . 42
3.2 Ticket Submission Platforms . . . 42
3.3 Unique Complaint Counts by Product . . . 43
3.4 Unique Complaint Counts by Category . . . 44
3.5 Classification Pipeline for ML Models . . . 44
3.6 Balanced Dataset After SMOTE . . . 46
3.7 Classification Pipeline for DL Models . . . 47
3.8 Classification Pipeline for LL Models . . . 48
3.9 Confusion Matrix of SVM . . . 50
3.10 SVM Approach . . . 51
3.11 Training and Validation Loss Curves of CNN . . . 52
3.12 Training and Validation Loss Curves of BERT with 2 Epochs . . . 54
3.13 Training and Validation Loss Curves of BERT with 50 Epochs . . . 54

4.1 XAI Concept . . . 56
4.2 Interpretability of Models . . . 57
4.3 XAI Explanation 1 . . . 62
4.4 XAI Explanation 2 . . . 64

5.1 Ticket Workflow . . . 66
5.2 User-Agent Discord Protocol . . . 68

5.3 User-Agent Helpdesk Protocol 69
5.4 User Test Process of Traditional TS 71
5.5 User Test Process of Interactive TS 72
5.6 Support Agent Test Process of Traditional TS 73
5.7 Support Agent Test Process of Improved TS 73
5.8 Preference for Systems by Age Group 74
5.9 Straightforwardness of the Ticket Submission Process 75
5.10 Effectiveness of Category and Subcategory Selection 75
5.11 Overall Experience Rating 75
5.12 Distribution of XAI Effectiveness by Age Group 76
5.13 Distribution of ChatBot Effectiveness by Age Group 77
5.14 Distribution of Discord Effectiveness by Age Group 77
5.15 Usage of Discord 77
5.16 Actual Labels Classified as Predicted Label: Credit and Prepaid Cards . . 86
5.17 Actual Labels Classified as Predicted Label: Bank Account Service 86

List of Tables

2.1 Examples of Helpdesk Tickets . . . 33
2.2 Term Frequency Results . . . 34
2.3 Document Frequency Results . . . 34
2.4 Inverse Document Frequency Results . . . 34
2.5 TF-IDF Results . . . 35
2.6 Words with Highest TF-IDF Values . . . 35
2.7 TF-IDF Approach . . . 35

3.1 Use of the Dataset Features . . . 41
3.2 Unique Values of the Features tags and company_response . . . 42
3.3 Main Category and their Subcategories . . . 43
3.4 Example of a Sample After Preprocessing . . . 45
3.5 W2V Sample Example . . . 47
3.6 Python Packages . . . 48
3.7 Results of the ML classification models . . . 49
3.8 Results of the DL classification models . . . 52
3.9 Results of the best classification models . . . 53

4.1 Wrongly Classified Examples from the Dataset . . . 62
4.2 Second Wrongly Classified Examples from the Dataset . . . 63

5.1 Packages and Tools . . . 70
5.2 UX of User Test for Traditional TS . . . 71
5.3 UX of User Test for Interactive TS . . . 72
5.4 UX of User Test for TS Comparison . . . 72
5.5 Mann-Whitney test . . . 74
5.6 User Comments on Both Helpdesk TS's . . . 75
5.7 User Comments on XAI . . . 76
5.8 User Comments on Discord and Chatbot . . . 78
5.9 User Suggestions on Helpdesk TS . . . 79
5.10 Support Agent Test Key Findings of their Organisation's TS . . . 80
5.11 Complaints of the Finance Dataset . . . 85

List of Acronyms

ML Machine Learning

DL Deep Learning

LLM Large Language Model

NLP Natural Language Processing

TF-IDF Term Frequency-Inverse Document Frequency

W2V Word2Vec

SVM Support Vector Machine

RF Random Forest

DT Decision Tree

NB Naive Bayes

CNN Convolutional Neural Network

HNN Hierarchical Neural Networks

RNN Recurrent Neural Network

BERT Bidirectional Encoder Representations from Transformers

RoBerta Robustly Optimized BERT

XLNet eXtreme Learning Network

TS Ticketing System

XAI Explainable Artificial Intelligence

AI Artificial Intelligence

NN Neural Networks

HMI Human-Machine Interaction

Introduction

The growth of the digital world has led industries to leverage modern technologies to enhance productivity in helpdesk services. A helpdesk TS is a tool used to manage and track user support requests and issues. It serves therefore as an interface for communication between clients and organizations, managing user support and services. According to the 2020 Hubspot state of service survey, helpdesk systems have been proven to be the most preferred tool for user service, ranking above FAQs, live chats, emails, and the like [36]. The helpdesk system organizes user requests into 'tickets', which can be monitored from the time they are created until they are resolved. To this end, a helpdesk TS manages user and service requests by automatically creating and organizing tickets, assigning them to the right agents, tracking their status, facilitating team collaboration, and collecting user feedback.

The current helpdesk TS [25, 12] mainly focuses on ticket classification, topic modeling, prioritization, and automated ticket resolution to manage large amounts of data effectively. In fact, over the past years, helpdesk systems have advanced technologically, automating support ticket management and providing valuable data insights, scalability, and enabling integration with other tools [3]. However, much work remains to be done to improve user satisfaction, develop more interactive and user-friendly systems, optimize response times, and increase transparency by implementing XAI techniques [1].

The goal of this thesis is twofold. Firstly, it presents an explainable helpdesk TS aiming to enhance the transparency of the solution by providing explanations through XAI techniques. Secondly, this thesis aims to contribute to the shift from traditional TS towards a more social TS by enhancing the interactivity between the user and the system and improving user engagement. To this end, this thesis investigates how an LLM can enhance various aspects of a TS to create a more interactive workflow for real-world scenarios. The integration of LLMs, such as OpenAI's Robustly Optimized BERT (RoBerta) and Llama, can automate routine tasks like ticket classification and transformation, thereby reducing resolution times and workload for human agents. These advanced AI models are trained on extensive text data, enabling them to understand and generate human-like text based on context.

Motivation

The key to success in many organizations depends on their users, each new satisfied user can contribute to the growth of the business. The motivation lies in helping the organization and user to have the best possible relationship and decreasing the user's problems by keeping up with modern technology. Nikhil Dawer [1] analyzed the user satisfaction of various companies and created some statistics about it. Those results

[1] `https://www.zonkafeedback.com/blog/customer-satisfaction-stats`

show that American companies can lose more than $62 billion annually due to poor user service and 34% of the users would even switch the brand. The most shocking statistic is that 91% of unhappy users would leave the company without complaining or telling the company the reason. Those results show the importance of taking good care of the users and always staying updated with them.

However people are getting more and more familiar with modern technology and that increases the potential for improvement in the helpdesk TS by saving time and money, which are important aspects of an organization. That motivated the thesis to integrate and evaluate different techniques of advanced AI models in the system. The statistics also show that 51% of the users prefer to use advanced models like Chatbots to solve their problems, which can reduce more than half of the tickets that are sent to support agents. That will decrease the ticket resolution time and increase user satisfaction. One more motivation that played a big role in doing that project was my interest and experience in using the helpdesk TS at the university. The goal was to work with the university and improve their helpdesk system, by having a positive impact on the real world scenario. Unfortunately, the contribution failed because of privacy reasons of the dataset but through the experience of being a student, I discovered the advantage of community support. Many students use the functionalities of the Discord Community, to have a supportive hand through their studies. That brought me to the point of researching community-driven support in a TS and came to the idea of building a TS in Discord that can engage users to support each other.

Research Goals

The main goal of the thesis is to create and evaluate an interactive helpdesk TS through the integration of XAI, LLMs, and community-based support, providing efficient ticket routing. To achieve the goal, the following sub-objectives are defined:

O1: Gaining Knowledge about the Current State of the Art: The starting point of each new project is to look for similar works that have been done and try to solve their gaps. Searching and analyzing good papers made the next steps possible.

O2: Searching a Suitable Dataset: The most important sub-goal from the beginning was to find a good dataset, that is large enough, has a clear structure, and provides the most relevant features. After reaching that goal, it was possible to design the project's pipeline.

O3: Implementation of Classification Methods: The implementation of the classification models was the next step, by doing research and tests to find the best approaches for the system.

O4: Creating the Demo Website of the TS: The next step is to build a demo website for the user and support agent, such that it can be evaluated by user tests.

O5: Additional Agents for an Interactive User-Centric Design: Having done the main technologic-centric design, it was time to think about new ways to increase the user experience. The implementation of XAI, Chatbots, and the Discord Community were the next sub-goals on the list.

O6: Doing User Tests for System Evaluation: After finishing the entire TS, it was time to prepare the user test setup and find people who were interested in evaluating the proposed system.

Structure of the Thesis

This thesis starts with the Introduction, five main Chapters, and ends with a Conclusion. Chapter 1 introduces the state of the art in current research. Then it continues with Chapter 2, which offers a look into AI techniques for understanding Chapter 3, which talks about the automated ticket classification process followed by Chapter4 with XAI. Finally, Chapter 5 concludes with the framework of the proposed solution and user studies, followed by a discussion about the project outcomes, limitations, and future work. The outline of each chapter is presented below :

- **Introduction** starts by providing an overview of current helpdesk TS's and their gaps in the modern world. Then the motivation of the project is presented, followed by the research goals and the thesis structure.
- **Chapter 1 - State of the Art: Current Progress and Challenges** analysis the existing work in current research. The thesis focuses on five main topics, such as text representation, ML, Deep Learning (DL), LLM, XAI, and community-driven support, including the gaps in the research to have an overview of what could be improved.
- **Chapter 2 - Foundations of Artificial Intelligence** starts with a basic explanation of the meaning and use behind AI followed by an Natural Language Processing (NLP) guidance and model overview before going deeper into the thesis's proposed solution.
- **Chapter 3 - Ticket Classification with NLP** shows the dataset structure and preprocessing. Then the different classification approaches with ML, DL, and LLMs followed by their performance evaluation and discussion.
- **Chapter 4 - Explainable Artificial Intelligence** explains XAI with its benefits and challenges. It gives an overview of the XAI SHAP model used by the project and provides some examples of SHAP results. Those results are based on the wrongly classified tickets of the chosen classification model.
- **Chapter 5 - Interactive TS with Multiple Agents** demonstrates the final proposed solution, after having an understanding of the previous topics. The Chapter goes into the TS framework by explaining the integration of the previous Chapters with different agents for efficient ticket management. It ends with some user study tests followed by the evaluation and discussion of the proposed TS.
- **Conclusion** is the final Chapter that discusses the key contributions of our work, the limitations of the proposed methods, and the future directions of our work.

Chapter 1

State of the Art: Current Progress and Challenges

Several approaches [38] [28] [3] have been proposed in the literature to address the challenges associated with a helpdesk TS. The evolution of these systems reflects the ongoing efforts to enhance efficiency, accuracy, and user satisfaction in managing and resolving tickets. These approaches range from traditional methods that rely on manual ticket management to advanced, automated systems powered by AI. In light of the previous, this section will first briefly present some papers on AI and community-driven TS, and finally discuss the gaps in the existing literature. This section meets the requirements of sub-objective O1.

1.1 Current Techniques in Text Representation

One of the fundamentals of NLP is text representation, by developing methods to convert text into machine-interpretable formats. Papers [25] [12] that focus on AI-based TS, have shown that Word2Vec (W2V), TF-IDF, and Feature-based text representation promise outstanding outcomes in combination with specific classification models. The best results are shown from the feature-based method but it's also the most complex one [28]. It transforms raw text into structured data by extracting linguistic and statistical features. There are other papers [38] [30] that focus on TF-IDF that balance term frequency with inverse document frequency to highlight significant words in documents and get similar or even better results. W2V goes into the category of feature-based methods which learn distributed word representations and capture semantic similarities. This method was used for DL models on raw data to get the most out of the text. DL models also have some embedding methods that can be used and that have shown similar results as W2V.

1.2 Review of Machine Learning Models

Many experiments [34] [15] have been conducted in the domain of ML for helpdesk TS, particularly in text classification and labeling. It's one of the simplest and fastest methods that predicts well on small datasets using already existing libraries of TensorFlow and sci-kit-learn. It is an easily implemented process and requires less training time and computational power. Fuchs et al. [12] compared over 40 research papers on ML-based TS for classification tasks and showed an accuracy between 80-90% for Random Forest (RF), Naive Bayes (NB), and SVM models. The results have been achieved through text preprocessing (removing stopwords, and special characters, using lemmatization

and lower casing) and text representation with the TF-IDF. The traditional ML models perform better with the TF-IDF technique than with W2V or more complex ones, because TF-IDF provides straightforward, interpretable, and high-dimensional features that align well with the assumptions and capabilities. According to Revina et al. [28], there is no need to introduce unnecessary complexity into a well-functioning system, simpler methods often yield better results.

1.3 Review of Deep Learning Models

Further research [38] has been done into more advanced models like Convolutional Neural Network (CNN) and Recurrent Neural Network (RNN), which mimic the structure of the human brain. These models perform well on large and complex datasets but require significant computational resources and time for training. There is also no need to do feature extraction because DL has its embedding methods which represent words in a dense vector space, capturing semantic relationships between words. Paramesh and Shreedhara [25] have implemented a single-layer CNN to automatically extract relevant features from ticket descriptions using real-world IT service desk ticket data. The paper shows that CNN offers a more robust and accurate solution for categorizing tickets, leading to improved service quality and outstanding outcomes, with accuracy ranging between 85-95%.

1.4 Review of Large Language Models

Researchers [26] [13] [21] have been exploring and developing new approaches in helpdesk TS to adapt to the state of the art in AI. In the last few years, LLMs like Bidirectional Encoder Representations from Transformers (BERT) and ChatGPT have been used in many fields to automate human tasks. They are based on transformer architectures that allow for wide contextual awareness of text through enhanced NLP. Zangari et al. [38] present a multi-level ticket classification method with embedding strategies by comparing pre-trained Transformer-based language models (ML-LM, SupportedLM, DoubleHeadLM, DeepTriage, and eXtreme Learning Network (XLNet)) with traditional models (SVM with TF-IDF and FastText). The evaluation is based on each model's performance in the first level (main ticket category) and second level label (ticket subcategory) by using standard classification metrics. The integration of hierarchical information showed that the advanced models (ML-LM, SupportedLM, and DoubleHeadLM) outperform the baseline. The new approach increased the F1-score (performance) by 28% and 5.4% in accuracy on a public financial dataset of customer complaints. Arici et al. [3] defined an automated ticket assignment process, based on pre-trained LLMs like GPT-3.5 and GPT-4 in a real-world scenario with minimal data and effort. They evaluate the models by their performance through the impact of learning approaches like zero-shot, few-shot, and ensemble learning. The outcome shows that GPT-4 has the best accuracy and that zero-shot approaches can achieve similar results to fine-tuned BERT models which on the other hand need a huge amount of training data.

1.5 Explainable Artificial Intelligence

User-centered design is a significant part of AI-based systems, by having a clear, interactive, and friendly interface. Working with AI tools can decrease user satisfaction, because of insufficient knowledge about the model's decisions and actions. This can

lead to user trust issues and comfort with modern technology. Researchers Ali et al. [1] recognized the need for a transparent system and experimented with XAI methods to refine the system's decision by visualization and documentation. Many XAI methods exist to provide specific types of explanations for various questions about AI models. There are four main levels of explanation described by Ali et al. [1] (Scoop-based Explainers, Complexity-based Explainers, Model-based Explainers, and Methodology-based Explainers).

Liao et al. [22] talk about AI models in general application. They define a Question-Driven Design Process (QDDP) based on fundamental concepts in user-centered design by emphasizing user feedback and questions. It is a human-computer interaction (HCI) approach to understanding user needs and creating an XAI Question Bank with a mapping guide to connect user inquiries with appropriate XAI techniques. Sun et al. [32] used the QDDP approach in their paper to enhance the explainability of AI models in domain-specific applications. The emphasis lies on Generative AI for code through Scenario-based Design, to support the needs of software engineers. The experiments had two steps. Firstly, they create realistic scenarios to present what AI is capable of doing in real-life situations in the coding field. Three use cases are used, code translation, code auto-completion, and natural language to code transformation, to show a simple example that was easy to understand by users. Secondly, they organized a workshop with 42 software engineers, who needed to have a look over the scenarios and ask questions about it. As in the previous paper, this QDDP helps the researchers to find out what users want to know about the AI model workflow. Finally, they use pooling-clustering-voting to filter out the most important question for the XAI method. In this thesis, the XAI is tested on the user interface, to explain the ticket classification decision of the model to gather the trust of the user.

1.6 Community-Driven Support

The above-mentioned authors focus mainly on automating ticket classification, which facilitates ticket routing, but don't focus on the users and how to reduce the number of tickets for the support teams. There are organizations like universities that have started taking advantage of community platforms to connect students to easily share insights and support each other.

Heinrich et al. [16] discuss in their paper that platforms like Moodle, Teams, and Discord can support collaboration between teachers and students, facilitating online learning, and providing social and emotional support. This approach started growing during the COVID-19 pandemic, underscoring the need for effective communication tools in education. The evaluation of these approaches is tested by students in the field of science by doing semi-structured interviews to gather insights into their experiences with Moodle, Discord, and Teams. The findings underline the potential of the different platform features such as Discord's informal and user-friendly interface, fostering student-led communication channels and Teams was preferred for its formality and integration with Microsoft Office tools. Those tools enable interactive support, and effective and quick information exchange, while Discord was preferred by students because of its interface. Lauricella et al. [20] talk about Discord's positive characteristics such as the user-friendly and flexible interface, making it particularly effective for fostering a sense of community, especially for distance learners. Discord improves online learning by allowing students and teachers to text, talk, and video chat with each other. In addition, it is less formal compared to other community platforms and makes it easier for students to participate in

various conversations to enhance a supportive and engaging environment. Teachers can share updates, and reminders and make announcements through this platform, keeping their students informed on any updates.

1.7 Gaps in the Literature

Despite the extensive research, several notable gaps persist that require attention to advance in the field of helpdesk TS.

G1: The limited availability of publicly accessible datasets poses a significant challenge to model development and benchmarking. This scarcity is primarily due to privacy concerns and the sensitive nature of the data involved. There is a pressing need for more comprehensive and anonymized data resources that can support robust and comparative research efforts.

G2: The insufficient integration of user-centered design approaches restricts the potential for system enhancement and reduces user satisfaction. Incorporating feedback from end-users is critical for developing more intuitive and effective TS's.

G3: There is a need for interactive user interfaces that can improve user engagement and satisfaction. Such interfaces should be designed to facilitate easy navigation and interaction, enhancing the overall user experience.

G4: There is a need for XAI systems to ensure that users and stakeholders can understand and trust the AI's decision-making processes to foster trust and confidence in the system's operations.

G5: There is no direct use of community platforms in companies. They could combine the above-mentioned automated helpdesk TS with a community platform that benefits from the technological and user-centric design efficiently.

Addressing these gaps is essential for the continued development and optimization of a TS, ensuring they meet the evolving needs of users and stakeholders.

Chapter 2

Foundation of Artificial Intelligence

The objective of this Chapter is to have a basic understanding of AI and the different model types that can be used for the classification of tickets. This knowledge is important for further comprehension of the project. It will cover an overview of AI, then it will go into the different types like ML and DL followed by LLM. Finally, the fundamentals of NLP will be present, which is an important point when working with text, in our case tickets. The concept of AI is to enable machines to perform tasks typically done by humans such as speech recognition, language translation, visual perception, decision-making, and problem-solving. AI achieves this by simulating human intelligence, effectively defining the brain of a machine. The first project done in this field was in 1921 by Czech playwright Karel Capek who released a science fiction play 'Rossum's Universal Robots'. The name robot was first time used in this project by introducing the idea of 'artificial people'. There are three key foundations of AI to support the imitation of a human brain:

- **Data:** A model is like a little kid, it needs to be taught how things work and that's why we need a lot of information such that the model gets enough knowledge about the specific topic or task. This information is stored in a dataset and can include several types such as text, images, audio, and video. The data provides the AI with the information needed to train models.
- **Algorithms:** After having the data, the model needs some guidance on how to use it and what exactly it needs to do. The data will be the input of an algorithm which is a set of instructions on how to process the data to get the desired output.
- **Models:** They are mathematical representations built by algorithms to make decisions (predictions) with the given data.

2.1 Neural Network Structure

Combining the above-mentioned key foundations gives us a basic machine-learning system of AI that illustrates how a machine's brain called a neural network learns through a dataset with neurons as brain cells. Imagine the network will be trained to identify if an email is spam or not spam. Then the dataset will have two columns that contain the emails and the labels (spam or not). This column with the emails will be preprocessed by NLP methods which will be explained in the next sections. It transforms the data into a machine-readable format which is mostly a vector representation of the data because a machine can not understand raw text data. It works with numerical values because

it is a mathematical representation. The labels will also be encoded, which means they will be presented as numbers, for example, 'spam' is changed to 1, and 'not spam' is 0. The new data will be split randomly into training (mostly 80% of each label), validation (10% of the training set), and test sets (20% of each label). The training set is then passed to the network and it will get the inputs (vectors) and the outputs (labels 1 and 0). The outputs will help the network learn from its errors. It is like a teacher who corrects the child to improve. The network will produce random results in the beginning because it doesn't know anything about the data and after each iteration, it will learn from its errors by comparing its output (prediction) with the actual output and correct them by going back to the beginning. This process is also called a binary classification because there are just two labels. The validation set will be put in after each training iteration to check the model's performance on unseen data, so we can see if the data is overfitting (performs well on training data but poorly on unseen data) or underfitting (fails to correctly extract patterns in the data). If one of both is happening during the training then the model can stop earlier with training and the model can be trained with other hyperparameter values. Those parameters are the configurations you define before starting the training process of an ML model (ex. number of hidden layers, number of times the model sees the entire training set). Imagine the hyperparameters as the setting function in your phone or computer, that can be adjusted to make your device best fit to you. The test set is applied to the model at the end when it reaches a good performance by training. A basic neural network architecture example is shown in Figure 2.1.

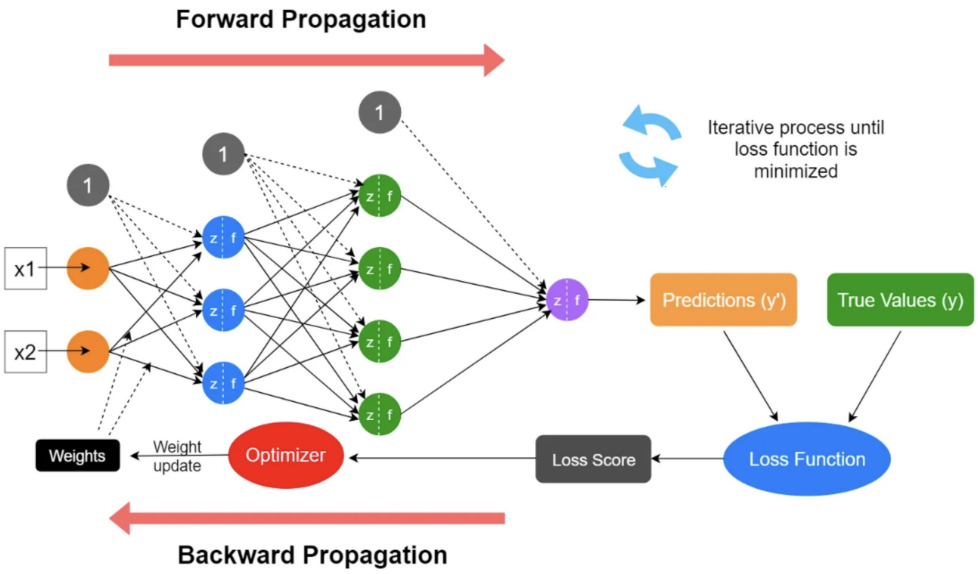

Figure 2.1: NN Architecture [1]

1. **Input Layer (x1 and x2):** The network starts with an input vector (x1 and x2 are the values of the vector) which can present any type of data. The values of the vector present different features or patterns of the given data. Each sample of the data is defined by its vector.

2. **Hidden Layers:** Forward Propagation: The network's hyperparameters include two hidden layers (green and blue), which can be adjusted to either increase or

[1] https://denizmogulkoc.medium.com/the-neural-network-club-deep-learning-cabe7013b691

decrease the network's depth. They are fully connected hidden layers that contain the neurons. All the input values (x1 and x2) are passed into each neuron because is fully connected and each neuron is composed of two main components to simulate a brain cell:

- **Weighted Sum:** The neuron will compute a weighted sum z of its inputs which is built out of weights (w), the input vector values (x1 and x2), and the bias (b). In the beginning, the w values (simulates the connection lines between the input and hidden layer on the figure) will be random, and with each iteration/learning, they will be updated/adapted to improve the output such that it gets closer to the actual output. The bias b is an additional parameter, which allows the activation function more flexibility and helps fit the data better.

$$z_1 = w_{11}x_1 + w_{12}x_2 + b_1 \tag{2.1}$$

- **Activation Function:** An activation function ensures that the model is non-linear. When the network is linear, each layer will be equivalent and this can lead to poor data learning. In a real-world scenario, the data is mainly non-linear because the patterns in the data are too complex.

  Example: Imagine we have pixel inputs of an image, and the pixel relations are linear, which means if we apply a change (increasing brightness by 10), each pixel will be increased by the same amount. If there is non-linearity, then each pixel will be treated fairly and according to the change such that it can handle complex visuals and image transformations. One example would be to play with the contrast and add shadows in some parts of the image and not everywhere in the image. There are four common functions f that can be applied to the weighted sum z.

$$ReLU(Rectified\,Linear\,Unit) : \int(z) = max(0, z) \tag{2.2}$$

$$Sigmoid : \int(z) = \frac{1}{1 + e^{-z}} \tag{2.3}$$

$$Tanh : \int(z) = \tanh(z) \tag{2.4}$$

$$Leaky\ ReLU : \int(z) = max(\alpha z, z) \tag{2.5}$$

3. **Output Layer:** The network will produce its first outputs/predictions which are mainly wrong (Predictions (y')).

4. **Loss Function:** The predictions (y') will be compared to the true values (y). This comparison gives a loss score which indicates how far the predictions are from the actual values.

5. **Backward Propagation:** The network will start again by the input layers but this time the weights will be adjusted/updated by the calculated loss score to minimize the error. The adjustment/update of the weights is done by an optimizer to get the best possible values.

6. **Iteration:** Each iteration of forward and backward propagation leads to better model performance, which means that the network will make more accurate predictions because the loss score decreases.

7. **Model Evaluation:** Models are mostly evaluated using their confusion matrices, providing a comprehensive measure of the model's overall performance. Figure 2.2 demonstrates the confusion matrix of binary classification, comparing the actual target values with those predicted by the model. Imagine the model classifies transactions into 'fraud' and 'not fraud', then the comparison provides information if the model's prediction is True or False. The Positive and Negative terms define the target categories which are 'fraud' and 'not fraud'.

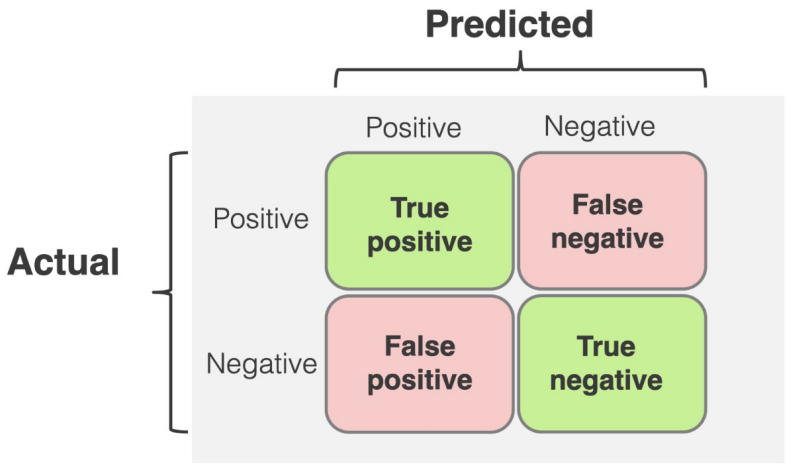

Figure 2.2: Confusion Matrix Representation [2]

Now we go deeper to understand the values True Positive (TP), True Negative (TN), False Positive (FP), and False Negative (FN) for binary classification.

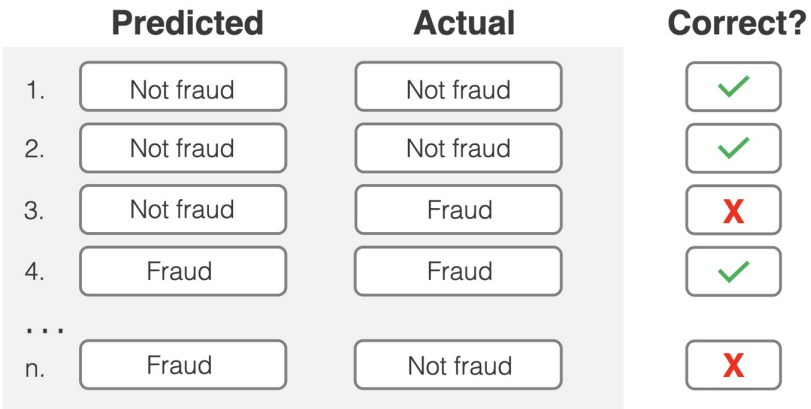

Figure 2.3: Models' Prediction [2]

Figure 2.3 demonstrates the prediction of a classification model that classifies transactions into two categories (fraud and not fraud). It predicted 3 samples correctly and 2 wrongly when comparing the predicted outcome and the actual outcomes.

[2] https://www.evidentlyai.com/classification-metrics/confusion-matrix

Figure 2.4: FN and FP Demonstration [2]

Figure 2.4 shows the FN and FP values presented by the Positive (fraud) and Negative (not fraud) terms that have been classified as false.

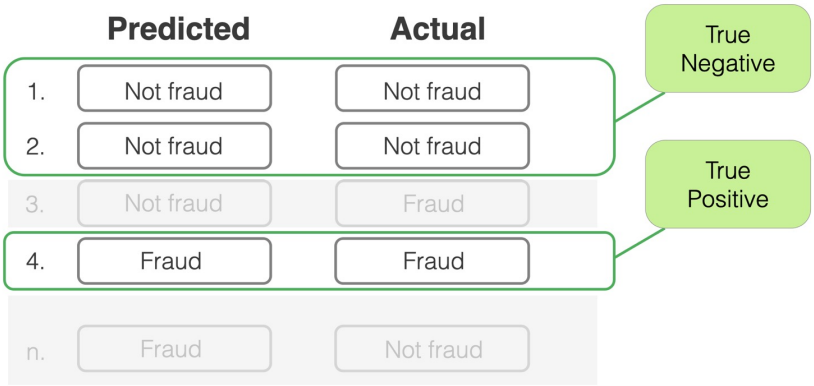

Figure 2.5: TN and TP Demonstration [2]

The TN and TP are the values where the Positive (fraud) and Negative (not fraud) have been classified correctly, shown in Figure 2.5. Those values can be used to calculate some evaluation metrics to evaluate the model by its predictions. Those metrics values are described below [35].

- **Precision:** The proportion of positive samples (fraud) that are correctly classified out of all positive predicted samples.

$$Precision = \frac{TP}{TP + FP} \tag{2.6}$$

- **Recall:** The proportion of positive samples (fraud) that are correctly classified out of all actual positive samples.

$$Recall = \frac{TP}{TP + FN} \tag{2.7}$$

- **Accuracy:** The proportion of samples (fraud and not fraud) that are correctly classified.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{2.8}$$

- **F-score:** It provides a single metric that balances the trade-offs between precision and recall. It is not possible to have both metrics maximized at the same time, that's why it balanced the trade between them to give up one metric in return for gaining another.

$$F1\ Score = 2\frac{Precision\ Recall}{Precision + Recall} \tag{2.9}$$

- **ROC-AUC:** Comparing models based on their ability to distinguish between classes. An AUC of 1 indicates a perfect model and an AUC close to 0 indicates a model that performs no better than random classification.

The mentioned approach was for a binary classification, to have a basic understanding of the evaluation method. A multiclass classification (our approach) is slightly different from getting the confusion matrix parameters. Imagine having four animal categories (cat, dog, horse, mouse). Each of them will be evaluated separately, which means that each of them is defined once by the term Positive and the rest will be Negative at the moment. When getting all the matrix parameters of each category, then they can be combined to calculate the overall performance.

2.2 The Evolution and Layers of AI

Figure 2.6 shows the hierarchy of different AI learning concepts that are used to perform different tasks or the same with better performance. This thesis focuses on a multi-level text classification task.

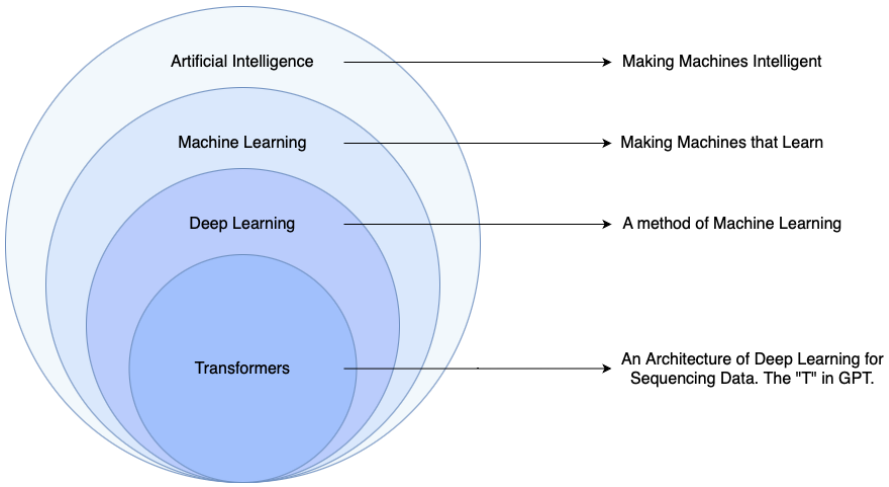

Figure 2.6: Layers of AI [3]

There are different techniques to perform/solve human tasks from basic rule-based systems like Neural Networks (NN) to more advanced ML models. NN are the fundamental building blocks of AI to imitate human intelligence. It was important to understand the basics before going to the advanced models. The concepts of NN are still considered

[3] https://www.notablehealth.com/blog/large-language-models-healthcare-revolution

in the more advanced AI methods, but their complexity and architecture can change based on the type of problem that needs to be solved. These advanced models are made to consider the limitations of basic ML models like NNs, which are:

- **Simple Architectural Structure:** The architecture of the model is too simple (containing few layers) for handling and understanding complex patterns in the input data.
- **Poor Generalization:** During the training, the model performs well but when it gets new unseen data, the performance can decrease. This can again be because of the poor architecture.
- **Feature Engineering Dependency:** Every model is built differently and so is their input data structure. The data structure (ex. vectors) in basic NN needs to be chosen and created manually, which can be time-consuming.
- **Dataset Scalability:** The model gets weak by handling complex and large datasets because of the increasing amount of features.
- **Interpretability/Transparency:** The model is like a black box, it is difficult to know how they combine these weights and activations values to predict the outputs.
- **Managing Unstructured Data:** The model is not performing well on text, image, or even audio inputs because of the low number of layers which can not capture all important patterns to learn from it.

2.2.1 Machine Learning

ML presents various techniques/algorithms that enable systems to learn from data and make predictions. NN is one type of ML algorithm, but there exist more advanced ones like DT, SVM, and RF. Then there also exist different ways of training/teaching a model. In real life, humans don't always learn from other humans, sometimes they learn from their own experience. This capability a machine also has because, in a real-world scenario, there is a lack of labeled data. That means there are no teachers (labels) who are waiting at the end of the process to tell the model if their prediction is wrong. There are different ways to teach/guide the models which are classified into 3 different ML model types [33]:

- **Supervised Learning:** The model learns from labeled data, where the output (labels) are known.
- **Unsupervised Learning:** The model learns from the data itself by checking on the patterns in the input. The output is unknown because the data is not labeled. Imagine you take many pictures of different flowers and you want to know which types of flowers there are. The model will try to analyze these pictures by looking at the patterns/features and classifying the pictures that have the same ones.
- **Reinforcement Learning:** The model learns from the environment by receiving feedback in the form of rewards (positive feedback) or penalties (negative feedback). Imagine the model is a chatbot that learns from user feedback. The chatbot has a sample of answers that it can provide to the user. If the user asks a question and the response of the chatbot does not satisfy the user then he/she will ask the same question. The chatbot will then get a penalty, meaning he learned that this was

the wrong answer. The feedback of the user can also be fetched in another way. Sometimes when the user writes, for example with the known chatbot ChatGPT, then in the end it always asks if the answer was good or bad. This is a direct way of getting feedback.

Considering the interpretability and limitation of basic NN

Decision Tree (DT) do not have such limitations, they are interpretable models due to their straightforwardness, as shown in Figure 2.7

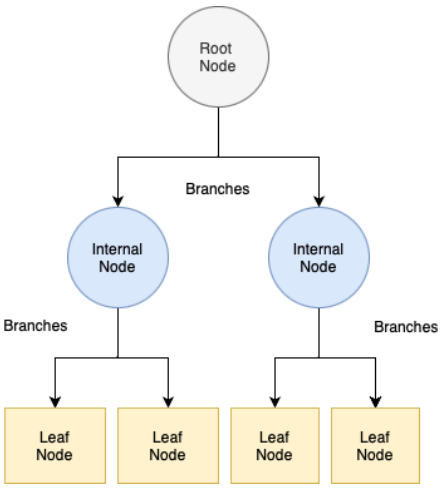

Figure 2.7: DTs Architecture

The root node contains the input which gets split into branches and each branch contains feature values of the input. Each path from the root to a leaf node represents a sequence of decisions based on feature values. Each split on the DT is a decision that was made by a simple rule like 'If feature X = 2, then branch left; otherwise, branch right'). Humans can easily understand why the model made the prediction (leaf node), it is straightforward. There is also a possibility to support the Interpretability by visualization because of the straightforward structure.

Limitations

Even if the models perform better than the basic NN, there are still the limitations of NN except the interpretability, but with some hypertunning (playing with the parameters of the models ex. the number of training iterations/epochs) they can get a really good performance.

2.2.2 Deep Learning

DL is a subset of ML that defines models that use deep (multi-layer) NN, enabling more complex data processing including feature engineering [33]. DL models are mainly used for image/object recognition (ex. In self-driving cars the object detection function) and speech recognition. In Figure 2.8, you can see a CNN which is an advanced neural network for image classification. It uses different types of layers that handle feature extraction to learn the different pixels of the images and the multi-level classification part. In the feature learning part, you can see that the model is taking one part of the images and is

going deeper and deeper into that part, to learn the different pixels in detail and extract the important features to create a vector representation for the classification.

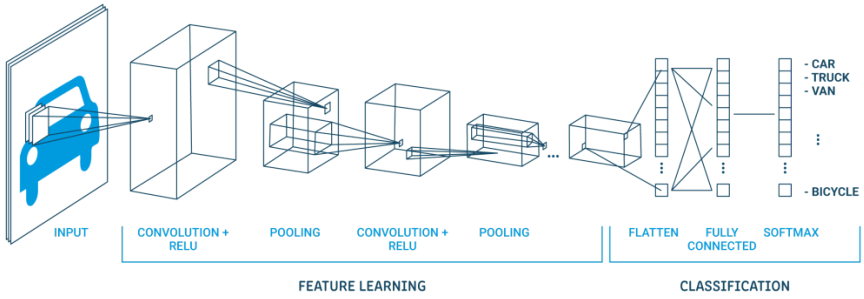

Figure 2.8: CNN Architecture [4]

Considering the limitations of ML models

- **Feature Engineering Dependency:** DL models can extract automatically relevant features from raw data, such as pixels in images or words in the text. Those models are End-to-End systems which means that all the preprocessing and model training is done in one process without splitting the tasks, and because of that, they can easily adapt to new tasks, as shown in Figure 2.9.

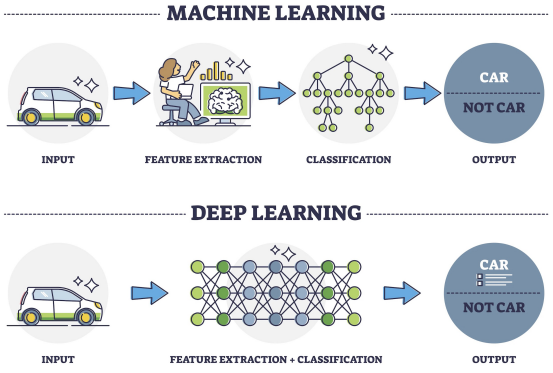

Figure 2.9: ML vs. DL [5]

- **Simple Architectural Structure:** DL models have a more complex and deep structure with different layers that perform different tasks. They can learn well from complex and unstructured data, for example, CNNs are specifically designed to handle images.

- **Dataset Scalability:** DL models are well-suited to scale with large datasets because of their architecture. When more layers are included then it is also simpler to handle more features.

[4] https://www.run.ai/guides/deep-learning-for-computer-vision/deep-convolutional-neural-networks

[5] https://www.telliant.com/consider-adding-deep-learning-to-your-next-software-project

- **Poor Generalization:** DL models have additional layers (dropout layers) that can handle overfitting (perform well on training data but poorly on unseen data) and underfitting (fail to correctly extract patterns in the data).

Limitations

The a need for high computational resources like GPU because of the large amount of data and training time. Data dependency is also a limitation because large complete datasets are difficult to find or create. Then the training can be long, which takes a lot of time away and also needs computational resources to handle it. DL models also don't perform well in language processing because it does not understand the complex context of the language.

2.2.3 Transformers

LLMs are advanced DL models that are optimized for language processing by understanding and generating human language [8]. The most known LLM is ChatGPT which can do various text transformation tasks like translation, summarization, and having a conversation with the user. Those models use ML techniques to learn from a very large amount of data which is mostly extracted from the internet because it's one of the biggest datasets. A simple workflow structure of an LLM is shown in Figure 2.10.

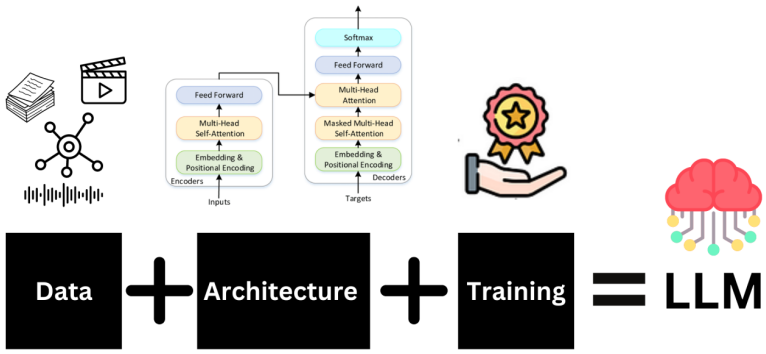

Figure 2.10: Transformer Architecture [6]

The first step is to extract a large amount of data from different outside sources. After collecting enough data, it gets processed and trained with an encoder and decoder.

Encoder Architecture

The encoder part processes the collected dataset, which includes:

1. **Embedding & Positional Encoding:** In this part, the data goes through an embedding mechanism which is explained in the next sections. Where the data is tokenized into words and converted into vectors (embeddings) with contextual information about the words. Additionally, positional encoding is added to know where each word is located in the vector space.

[6] https://www.linkedin.com/pulse/understanding-building-llm-applications-pavan-belagatti-lakvc/

2. **Multi-Head Self-Attention:** This mechanism takes the previously generated embeddings and extracts different relations between words by focusing on different parts of the input sequence, which is done simultaneously (multi-head).

3. **Feed Forward Layer:** It is a fully connected layer like in the NN that takes the captured information from the self-attention mechanism and through the neurons in the layer it learns more complex patterns.

Decoder Architecture

The decoder generates the output sequence and also trains the model with the target output shifted to the right, which includes:

1. **Target Output**

   (a) **Training preparation:** Imagine the input in the encoder part was the sentence 'Lions are big cats'. Now we want to train the model to learn the word contexts and the relation to predict the next word in the sentence by giving a word. This will be achieved by using the target output but shifted to the right which will be ' [empty] Lions are big cats' to train the model. The process will be explained more in the next steps.

   (b) **Embedding & Positional Encoding:** Like the encoder, embeddings, and positional encoding are used on the target output which is shifted to the right.

   (c) **Masked Multi-Head Self-Attention:** This step is similar to the encoder part, just that the target output will be masked/hidden such that the model does not have access to future words in the sequence, only the previous one, to ensure a fair prediction. The target masked sentence ('[empty]', because that's the first word) will be passed into the next step for prediction and there is no word before [empty].

2. **Output of the Decoder**

   (a) **Multi-Head Attention:** The mechanism gets as input the masked target output [empty] to predict the next word with the help of the output from the encoder, such that it can use its multi-head attention to focus on relevant parts of the sequence for prediction. When the model predicts the next word correctly which is 'Lions'.

   (b) **Feed Forward Layer:** The layer will get the information about the relation and context of the predicted word and process it to generate the corresponding vector output. This is done for every word to learn the meaning and context of the word in a sentence.

3. **Iteration:**These steps are done until each word of the sentence is predicted. If the prediction was right then the next masked sentence will be '[empty] Lions'. If the prediction is wrong, the masked sentence will give access to the next word such that the model learns from its error.

Considering the limitations of ML and DL models

- **Language Processing Limitation:** The self-attention mechanism in transformers (ChatGPT) can handle and generate complex contextual relationships in language. They perform well in tasks like text generation, translation, and comprehension.

- **Diverse Input:** They can handle diverse inputs like images or structured data like text from articles or audio, because of their complex architecture.

Limitations

These models are complex because of the high-level humans task they perform and hyperparameter tuning would be hard and time-consuming. They also need more computational resources and longer training time for the DL models.

2.3 Comprehensive Guide to Natural Language Processing

Machines use a different language than humans for communication and comprehension, they don't understand for example raw text data (ticket description in our projects). In that case, NLP methods [27] are utilized to transform the human language into a machine-understandable format such that they can work with each other. It is the same concept when two people speak different languages and they need to use a translator, body language, or another tool for comprehension. In this section, we will explain the NLP approach which is split into two parts.

2.3.1 Preprocessing

The first part is the cleaning and preparation of the raw text data before it is used in more advanced processing tasks. It means removing any noise (not relevant information) from the text such that it keeps the most relevant information and gets simplified. Several methods can be used, but not all of them have to be used, and some are dependent on each other. It depends on the type of the machine's model because the model will deal with the text afterward for classification. Some advanced models can understand the meaning behind the text by keeping the original content without preprocessing. However, 7 different kinds of preprocessing methods are done by the system/machine symbolized by the robot in the pictures.

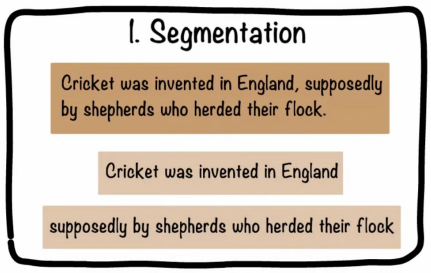

Figure 2.11: Segmentation Method [7]

- **Segmentation Explanation 2.11:** It takes long sentences of the text and splits it into smaller sentences. The system can do this by checking if there is any comma that combines sentences and that can be split.
- **Segmentation Purpose:** It makes the text more clear and simpler for further processing.

[7] https://www.youtube.com/watch?v=CMrHM8a3hqw&list=PLcPC3EBrWkrXe988s1dA0izIulqI_xYmA&index=5

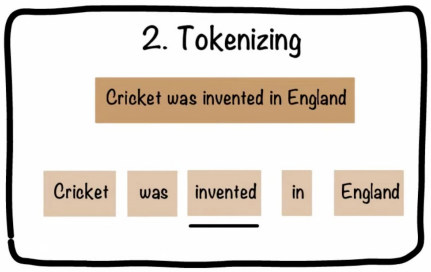

Figure 2.12: Tokenizing Method [7]

- **Tokenizing Explanation 2.12:** This approach takes each sentence and splits it into individual words.
- **Tokenizing Purpose:** It helps the system to check and analyze each word of the text to see which ones can be removed or changed.

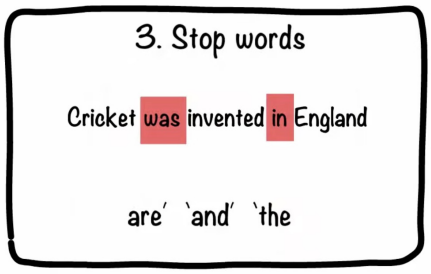

Figure 2.13: Stop Words Method [7]

- **Stop Words Explanation 2.13:** The removing part is happening in this method. Some words are called stopwords, they are common words that don't add much meaning to a sentence.
- **Stop Words Purpose:** When we think about text classification, the model wants to analyze the text efficiently by focusing on more informative and relevant words. It can extract the unique words for each text, which makes it easier to check which texts are similar and which are different. If there are stopwords, each text would have a lot of the same words which makes it harder to find the uniqueness and to classify. The methods can make the classification process faster and more accurate.

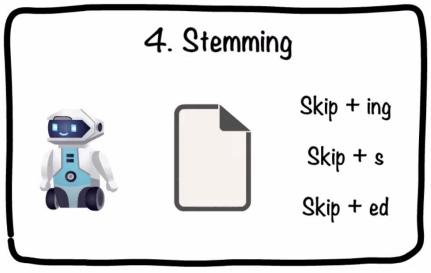

Figure 2.14: Stemming Method [7]

- **Stemming Explanation 2.14:** It is a normalization method that converts text to a standard format. It removes all prefixes or suffixes of the word (mainly verbs) and finds a base form for the same words. In some cases the base form can be an unreal word ex. 'running' could be stemmed (transformed) to 'runn'.
- **Stemming Purpose:** It helps the model to recognize the same word in different forms as one unique word. This method is mainly for traditional models which don't understand the meaning between words but just analyze the frequency of unique words. The model will see that 'skip' appears 3 times in a text.

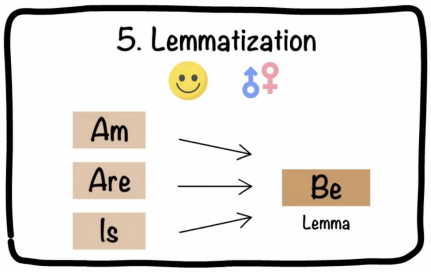

Figure 2.15: Lemmatization Method [7]

- **Lemmatization Explanation 2.15:** This approach is similar to the previous one and is a part of normalization. The difference is that there are no unreal words because it uses a dictionary to understand the part of speech. Then it reduces the words to their base form (lemma) by considering the context.
- **Lemmatization Purpose:** The goal is the same as for stemming, only that this method is slower because of its complexity by considering the word's meaning and context. This method is for advanced models that are built to understand the meaning behind the text and are more accurate.

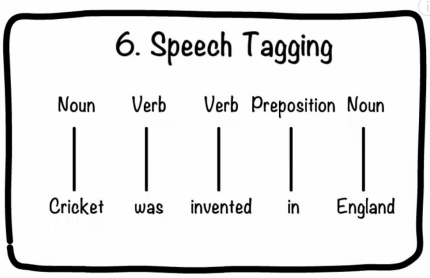

Figure 2.16: Speech Tagging Method [7]

- **Speech Tagging Explanation 2.16:** It assigned the POS (part of speech) to each word. POS are like roles that each word has in a sentence (ex. the POS 'Verb' has the role of describing an action of something like 'running')
- **Speech Tagging Purpose:** This approach is also more advanced because they use POS tagging to analyze the structure of sentences. It helps them to understand the grammar and meaning of the text in a better way.

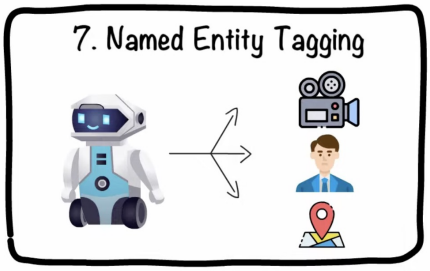

Figure 2.17: Named Entity Tagging Method [7]

- **Named Entity Tagging Explanation 2.17:** This method is similar to speech tagging, only here are the words labeled by 4 different kinds of named entities (Person (PER), Organization (ORG), Location (LOC), and Miscellaneous (MISC)).
- **Named Entity Tagging Purpose:** Advanced models use ENT (named entity tagging) to identify and categorize these names/categories in sentences. It helps them to understand which words have an important role in the text. When the model wants to classify the text into different countries (ex. Luxemburg, France, and Germany) and the NET method is applied in the preprocessing step then the model will focus on the word ('Paris') in the text (ex. Mike lives in Paris for six years). It will classify it to France just by looking at the ENT. This method makes the classification more accurate and faster, without the model analyzing the whole text.

2.3.2 Text Representation

The second part of the NLP process is the text representation [27] [9]. The previous step was preprocessing, it cleaned and prepared the text such that the next step can handle the text in a simpler and faster way. However, the text is not yet ready to be understood/read by machines because it's still human language just simplified and to make it readable by machines/models, there are text representation methods. It converts the text into numerical values, to be more precise it converts text to vectors because machines only work with numbers. The underlying architecture and design of a machine are based on binary arithmetic and machines are doing all the human tasks as classification by mathematical operations. There are three main methods that this thesis will use because there are three different model types for classification in this project. It goes from traditional to the most advanced models, which will be explained in the next section. The classification part can also be seen as a part of the NLP process because the model is trying to understand the meaning behind the numerical text to classify them. NLP defines all the steps and methods that are used to process, analyze, and understand human language.

Traditional Method: TF-IDF

TF-IDF [9] is a statistical method for evaluating the importance of a word to a set of documents (set of ticket descriptions in this thesis) or to one of the documents in a corpus (set of documents). The importance of a word increases proportionally with the number of times it appears in a document but decreases inversely with the frequency of its appearance in a corpus. In that way, the cleaned text will be converted into a vector with the corresponding values and this method is a traditional one that does not focus

on the meaning or context of the text. There is no need to apply speech or named entity tagging for the preprocessing step. TF-IDF is defined into TF and IDF, and the two concepts are described separately below.

- **Term Frequency:** TF denotes the frequency of occurrence of a term (word) in the document (text) by normalizing it with the number of total words.

$$TF = \frac{number\ of\ times\ the\ term\ appears\ in\ the\ document}{total\ number\ of\ terms\ in\ the\ document} \tag{2.10}$$

- **Inverse Document Frequency:** It denotes the frequency of occurrence of a term in the corpus. Terms that appear more common in a corpus receive lower importance than terms that are unique or have a small percentage.

$$IDF = log\frac{number\ of\ the\ documents\ in\ the\ corpus}{number\ of\ documents\ in\ the\ corpus\ contains\ the\ term} \tag{2.11}$$

- **TF-IDF Value:** The result shows how important that word is for each document, such that the models know which words to pay attention to when doing the classification. Lower scores indicate that it is not relevant and higher values indicate that it's important, there is no upper limit here and the minimum is 0.

$$TF\ -IDF = TF \cdot IDF \tag{2.12}$$

Example Scenario: Analyzing helpdesk tickets without preprocessing for a better understanding of the classification part

Table 2.1 shows some helpdesk tickets:

Table 2.1: Examples of Helpdesk Tickets

ID	Category	Ticket
1	Bank Service	'I need help with my bank account statement. The statement shows an incorrect balance. Last week, it wasn't incorrect, I don't know what happened'
2	Loan Service	'I'm having trouble with my loan repayment schedule. The payment amounts are incorrect. I'm not sure why they are incorrect.'
3	Bank Service	'How can I change my bank account password? I forgot my current password.'
4	Loan Service	'I want to know the status of my loan application, I think there is something incorrect. How long will it take to process?'

1. **Term Frequency (TF):** All the words from each ticket will be displayed on Table 2.2 as features/columns and the ticket as rows. Each cell will contain the frequency of the words appearing in the specific ticket. The rows indicate the vector of the ticket for the model. This example will just take some words to demonstrate how it works. It will also show why this method needs TF and the DFT functions and not just one of them. The Table 2.2 shows how many times a term (word) appears in each document (ticket), just for simplification but in the brackets you can see the real values of TF for each word. If we look at the values, we can tell that tickets 1 and 3 are probably from the same category because they have the same words (Bank, Account), and the same for tickets 2 and 4 (Loan). However, if we look deeper, we can get confused by the numbers for the word "My". It shows that Tickets 1,2 and 3 have the same amount of the word 'My' but we know that

Tickets 1 and 2 are not from the same category. We can also see that 'My' is a stopword and that's why it is important to do preprocessing to avoid this situation of confusion, same for the word 'With'. There is a similar situation with the word 'Incorrect', which is not a stopword. Here, the second function gets into work to avoid tickets getting classified into the wrong category.

Table 2.2: Term Frequency Results

	Bank	Account	Statement	My	Incorrect	With	Loan
Ticket 1	1 (0.042)	1 (0.042)	2 (0.083)	1 (0.042)	2 (0.083)	1 (0.042)	0
Ticket 2	0	0	0	1 (0.045)	2 (0.090)	1 (0.045)	1 (0.045)
Ticket 3	1 (0.077)	1 (0.077)	0	1 (0.077)	0	0	0
Ticket 4	0	0	0	0	1 (0.043)	0	1 (0.043)

2. **Document Frequency (DF):** The first step is to count how many tickets contain the word and let's remove the stopwords, as shown in Table 2.3

Table 2.3: Document Frequency Results

Bank	Account	Statement	My	Incorrect	With	Loan
2	2	1	3	3	2	2

3. **Inverse Document Frequency (IDF):** Then the IDF tells us how unique or rare a word is. We use the formula to calculate the IDF for the word 'Bank' as an example.

$$DFT : log(\frac{number\ of\ documents}{number\ of\ documents\ containing\ the\ word}) = log(\frac{4}{2}) = 0.301 \quad (2.13)$$

Table 2.4: Inverse Document Frequency Results

Bank	Account	Statement	My	Incorrect	With	Loan
0.301	0.301	0.602	0.125	0.125	0.301	0.301

The numbers in Table 2.4 can tell that 'My' and 'Incorrect' are appearing in most tickets which means that they are not so important (words that can appear in every category without giving a real meaning of the category) in most cases, that is why they have a lower score.

4. **TF-IDF Calculation:** TF-IDF combines TF and IDF. The higher the TF-IDF score, the more important the word is in that ticket. The Table 2.5 will ignore the stopwords and focus on the values that have a high score. The highest numbers tell us which words are the most important for each ticket. If the classification model gets the vectors in Table 2.5 of each ticket, then it can better compare the tickets and see which tickets have similar important words. Tickets 1 and 3 have 'Bank' and 'Account' as similar important words as shown in Table 2.6, which means they are both classified in the same category which is 'Bank Service'. Tickets 2 and 4 have 'Loan' as the same word importance, both get classified into the 'Loan Service' category. If new tickets arrive, they will go into the same process and then

the model will compare the new ticket vectors with the existing ones to see in which category they belong. This method has certain limitations which will be explained in the next section.

Table 2.5: TF-IDF Results

	Bank	Account	Statement	My	Incorrect	With	Loan
Ticket 1	0.013	0.013	0.050	0.005	0.010	0.013	0
Ticket 2	0	0	0	0.006	0.011	0.014	0.014
Ticket 3	0.023	0.023	0	0.010	0	0	0
Ticket 4	0	0	0	0	0.005	0	0.013

Table 2.6: Words with Highest TF-IDF Values

Ticket 1	Bank, Account and Statements
Ticket 2	Incorrect and Loan
Ticket 3	Bank and Account
Ticket 4	Loan

Advanced Methods: W2V and LLM embedding

The previous TF-IDF approach has certain limitations, which are listed below, and can be handled by more advanced methods like W2V or LLM embeddings [9].

- **Vectors Size:** The vectors can get long because they take every word of the corpus (set of texts [Ticket 1, 2, 3, and 4]). In the example above, the vector size of each ticket is 82 and if we work with a larger corpus it can get long, such that it needs a lot of computational resources and memory.

- **Sparse Representation:** The method is a sparse representation which means most of the numbers in the vectors will be 0. If each ticket vector contains every word of the corpus then not each ticket has these words.

- **Text context:** Imagine you have two new tickets [Ticket 5: 'I need help'] and [Ticket 6: 'I need assistance']:

Table 2.7: TF-IDF Approach

	assistance	need	I	help
Ticket 5	0	1	1	1
Ticket 6	1	1	1	0

In Table 2.7, the words 'assistance' and 'help' are not defined as the same but tickets 5 and 6 should have the same vector because they have the same meaning. There is a lack of context understanding.

Considering those limitations, a W2V approach is used, which is a neural network-based model that learns from word embeddings. Word embedding is a NLP technique that captures the semantic meaning of words and it presents each word into a continuous vector space. Words with similar meanings have similar vector representations. The section will show an example to understand the process better. There are two different ways to do it:

- **Continuous Bag Of Words (CBOW):** It predicts the target word (Which can be any word of the sentence, all words will be once a target one) by inputting the context words (these are the surrounding words of the target word)
- **Skip Gram:** It predicts the context words by inputting the target word.

Example: Step-by-Step Explanation of W2V using Skip-gram

1. **Collect and Preprocess Data:** Imagine we have a small dataset of 4 tickets, as shown below.

ID	Ticket
1	'My internet connection is down.'
2	'I cannot log in to my account.'
3	'The website is loading very slowly.'
4	'I forgot my password and can't reset it.'

   After preprocessing the data with the mentioned steps in the previous section, it will look like this:

ID	Preprocessed Ticket
1	['internet', 'connection', 'down']
2	['cannot', 'login', 'account']
3	['website', 'loading', 'slowly']
4	['forgot', 'password', 'reset']

2. **Create a Vocabulary:** A vocabulary of all unique words in the dataset will be created for the process.
   - 'internet', 'connection', 'down', 'cannot', 'login', 'account', 'website', 'loading', 'slowly', 'forgot', 'password', 'reset'
3. **Generate Training Data:** The training data will be generated by the context window size which determines how many words before and after the target word are considered. Let's use a context window size of 1 for simplicity. For the sentence 'Internet connection down', the training pairs would be:
   - (target: 'internet', context: 'connection')
   - (target: 'connection', context: 'internet')
   - (target: 'connection', context: 'down')
   - (target: 'down', context: 'connection')
4. **Convert Words to One-Hot Vectors:** Each word in the vocabulary is converted into a one-hot vector. It means that each vector value is 0 or 1. If it is 1 then it is the corresponding word which shows the position of the word in the vocabulary. If it is 0 then it shows the position of the other words.
   - 'internet' -> [1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]

- 'connection' -> [0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]
- 'down' -> [0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0]

5. **Train the Neural Network:** A simple neural network 2.18 with one hidden layer will be used to learn word embeddings. Here's how the training process works for the target word 'internet' with the context word 'connection'.

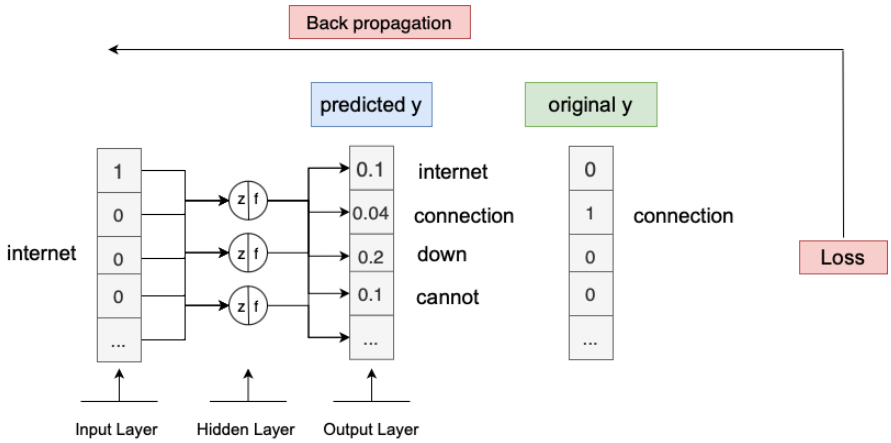

Figure 2.18: Embedding Approach with NN

(a) **Input Layer:** The one-hot vector of the target word is the input.

(b) **Hidden Layer:** This layer determines the dimensionality of the word embeddings (e.g., 10 dimensions) which is 4 in our example, it can be freely chosen. The layer contains the weight matrix where the word embeddings are created and learned. These embeddings are used to capture the contextual relationships between words.

(c) **Output Layer:** This is a softmax layer that predicts the probability distribution over the vocabulary for each context word. The first round will produce random values because it doesn't know anything about the word embeddings.

(d) **Loss function:** The loss value will be calculated by the cross-entropy function which calculates the loss between the predicted distribution y and the actual distribution (one-hot vector of the target word). The loss indicated how far the predictions were from the actual values.

$$Loss = -\frac{1}{C}\sum_{c=1}^{C}\sum_{i=1}^{V} t_{i,c} log(y_{i,c}) \tag{2.14}$$

- C is the number of context words.
- V is the size of the vocabulary
- t i,c is the on-hot encoded context word C.
- y i,c is the predicted probability for context word C.

(e) **Backpropagation:** The model will use the loss value with the help of the gradient descent to adjust the weights [w1, w2, w3] to increase the probability of 'connection' and lower the rest.

This process is done for each word of the training sample. The final generated embedding matrix that contains each word's vector embedding, is then used for new words

that are not in the training sample. The Model can understand with the help of the learned matrix the new word's context. The Continuous Bag Of Words (CBOW) model has a similar process which is the inverse of this one. When going back to the mentioned limitations of TF-IDF, this method's word vector size is determined by the model and has no fixed length. It can vary from small to large. Then there is no sparse vector and it focuses on learning the context of the words. LLM embedding has a similar way of representing text, it is just more complex and advanced. This embedding method is inside the Transformer model (ex. BERT) and the key difference from W2V lies in the vector representation of a word. Transformers can understand the meaning, context, and relation of a word in a sentence. That's why those word vectors can change depending on the other words of the sentence. If we have two sentences with the same word but different contexts such as 'river bank' and 'financial bank.', then the word 'bank' will have different embeddings.

Chapter 3

Ticket Classification with Natural Language Processing

The objective of this Chapter is to provide an understanding of ticket classification with the help of NLP. It highlights the importance of automated ticket routing in the current state of the art. Key points will include the classification pipeline, the challenges of data preprocessing, and the evaluation of different classification models with their metric values.

3.1 Overview of Ticket Classification

The fundamental of various industries, such as user support, IT service management, and other service-oriented fields, is the ticket classification process [24]. It organizes incoming ticket descriptions (ex. 'I have an issue with my bank account, I can't log in') into predefined categories (ex. 'Bank Account') to streamline their management and resolution. The classification [4] is mainly done by humans, which has its disadvantages such as a high-resolution time, overload of tickets, and risk of errors. The project will consider the challenges and focus on automating this process with machines for a more effective TS. It can be used in many parts of the ticketing handling process by automating the priority assignment of tickets, routing to the appropriate support team, feedback analysis, and automated responses. This thesis focuses only on the classification of tickets into their categories and the assignment to their corresponding teams. This approach offers several advantages and enhancements.

- **Efficiency:** The classification happens automatically by the system without any break or human action, this makes the ticket routing faster. In a situation, where administrative people are lacking because of any reason, the support agent needs to do his/her work and focus on assigning tickets rather than solving them. In that case, the proposed approach would help the support agent to focus more on their main task, which is to solve the ticket.

- **Improved Accuracy:** Humans are not machines, which makes them less effective in specific tasks. In manual ticket classification, it can happen that people are getting too much workload, fatigue, and subjectivity which can lead to inconsistent or incorrect categorization. A machine does not have such weaknesses and it can work without stopping by achieving high performance depending on how well the machine was trained for this task.

- **Scalability:** It's a challenging situation when it comes to handling manually a large amount of tickets at an appropriate time. The human can lose the overview of the tickets and even risk making errors with the classification. An automated system can handle many tickets simultaneously without getting overloaded and without compromising on speed or accuracy, ensuring that user issues are addressed promptly even as the demand increases.

- **Data-Driven Insights:** The data that contains all the ticket information with their description, user feedback, ticket creation time, and responses, can be used to gain valuable insights into the company's performance. A machine can analyze the data to classify them into specific topics to have an idea of the current state of the services. If the company wants to check how satisfied its users are and what gave them the feeling, the system can classify/separate the user feedback into different sentiments for an effective and clear analysis.

- **User Satisfaction:** Besides the classification process, the key point of a TS and this thesis lies in the happiness of the users. If the users are not satisfied with the services they get, the companies will lose value and it will hinder long-term success. The above-listed improvements of the system will automatically have a positive impact on the user because the response time will be reduced and the user's problem will be handled by the right team.

The above-mentioned benefits can be maintained by different AI models which are trained to do specific tasks by supporting the machine with the help of NLP methods.

3.2 Customer Complaints Dataset

Choosing the dataset is one of the most significant steps before starting with the design of the project because we can only build something with the information we get from the dataset. This section presents the dataset which has been used for the classification procedure. Each analyzing and cleaning step will be shown from the beginning. It is crucial to understand the data structure and the information it can provide. It gives us an idea of how to design the classification pipeline, the TS interface, and if there are any other text processing possibilities like building a chatbot or knowledge base that delivers ticket solutions. The advantages and disadvantages of the dataset will be discussed in the following section and what the final dataset looks like.

3.2.1 Overview of the Original Customer Complaints Dataset

The search was focused on finding a dataset that had at least the ticket description and category columns because the main idea was to do ticket classification on both features. After an extensive search and investigation, a suitable public customer complaint dataset [1] was found in Kaggle, successfully achieving sub-objective O2. The dataset is from a company named Consumer Financial Protection Bureau (CFPB), it manages and collects complaints from users and each complaint will be directed to the corresponding company such that they get faster support. CFPB has many different public datasets (considering the literature gap G1) and also a new version of the complaints dataset which was discovered late in the process. The new version has more samples from 2017-now and it contains some new features. The project uses the old version from 2017-2021 which is in

[1] `https://www.consumerfinance.gov/data-research/public-data-inventory/`

JSON format, contains about 78,313 samples, and has 22 features. Some information on the dataset features is presented in Table 5.11 in Appendix 5.5.

3.2.2 Data Analysis and Cleaning

The data analyzing and cleaning procedure was done in a Jupyter Notebook for better visualization by following an exploratory data analysis (EDA) approach to summarize the main characteristics. Looking at these characteristics, the thesis could find out which information can be used to make the TS efficient and automated. The first step was to remove unnecessary columns. Table 5.11 in Appendix 5.5 presents the content of the dataset with examples. Few columns can be removed because they don't give any useful information to test the performance of the TS. Those features can not be used for classification or even to build a knowledge base for a chatbot.

```
import pandas

df.drop(columns=[ 'index', 'type', 'score', 'zip\_code', 'complaint\_id',
                                    'date\_received', 'consumer\
                                    _disputed', 'state', 'company\
                                    _response','company','date\_sent\_to
                                    \_company', 'consumer\_consent\
                                    _provided', ])
```

After removing the unnecessary columns with the code above, the rest of the features can be analyzed in detail with EDA to see if they provide any information to work on. Table 3.1 describes why those columns are kept and how these can be used in the TS.

Table 3.1: Use of the Dataset Features

Features/columns	Could be used for ...
tags	Tags are also helpful for ticket classification. It could again help to see which support agent fits the best or even just to have a simple overview of the ticket without reading everything when seeing the tickets.
issue	This can be used to have a small summary for the ticket description. It can help the support agent to have an overview of the ticket before accepting it. An LLM can be trained with this feature to generate a summary for new tickets.
product	It can be used for classifying the tickets into categories that are assigned to specific support teams.
company_public_response	A knowledge base can be built out of the company's responses, such that the user can already look for similar problems and solve them on their own. The knowledge base can also be shared with the companies such that they solve the user's problems faster. This knowledge base can be used in a chatbot that is trained with LLM and can give an automatic solution when a similar ticket is already solved.
submitted_via	This feature can give us an overview of which tools are often used by users when submitting a ticket. It can help to know which tools to focus more on when working on the user-centric design.
sub_product	It can be used for classifying the tickets into subcategories that are assigned to specific support teams. When looking at the CFPB website, each category has subcategories.
complaints	This is the most important feature. Those are the ticket descriptions.
timely	It provides an overview of the performance of the CFPB TS. The column indicates whether the ticket was resolved within an appropriate time frame.
sub_issue	This can be used to have a small title for the ticket description. It can help the support agent to have an overview of the ticket before accepting it. An LLM can be trained with this feature to generate a title for new tickets.

The simplest way is to check how many unique values each column has before looking at the relation between them. It gives an overview of the content without looking too deep inside. The columns 'issue' and 'sub_issue' will not be used in this project because

of the time limit, instead, we will use pre-trained LLMs to generate a ticket title and summary.

Features	Unique Values
tags	3
product	17
submitted_via	6
company_public_response	3
sub_product	72
timely	2
complaints	20931

The Table above shows that company_response and tags don't have an appropriate number of unique values, such that we need to check the values.

Table 3.2: Unique Values of the Features tags and company_response

Features	Unique Values
tags	Servicemember / Older American / Older American and Servicemember
company_response	Company believes complaint relates to a discontinued policy or procedure / Company has responded to the consumer and the CFPB and chooses not to provide a public response / Company chooses not to provide a public response

Table 3.2 shows us that the values don't give any useful information about the column, such that they will be removed. Now the only columns that can be used are 'product' and 'complaints' for ticket classification. The 'sub_product' column has too many unique values to use in classification and by analyzing the CFPB website, they don't have so many 'sub_product'. They use the 'product' column for their subcategories. The only columns that are useful for this project are the ones for ticket classification and the 'timely' and 'submitted_via' for statistics.

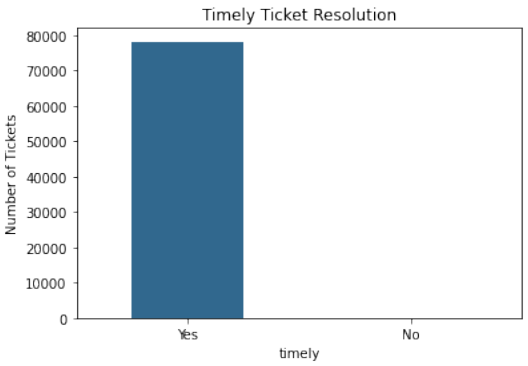

Figure 3.1: Timely Ticket Resolution

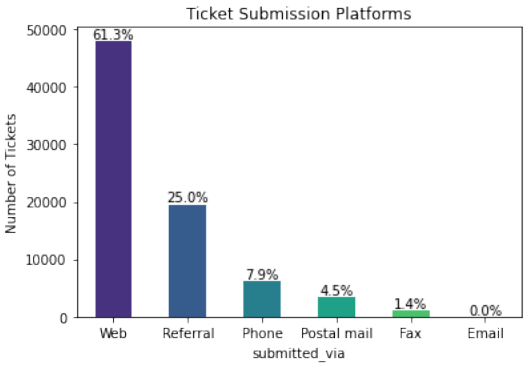

Figure 3.2: Ticket Submission Platforms

Figure 3.1 shows that all tickets were solved at an appropriate time, which is not 100% trustful because it's a big company and they work with other companies to solve user complaints about their service. They get billions of complaints and requests, such that this statistic will be ignored without knowing the exact meaning of it. Figure 3.2 shows that 61.3% of the users are submitting their complaints via the website, which gives us a hint that creating a website will enhance user satisfaction.

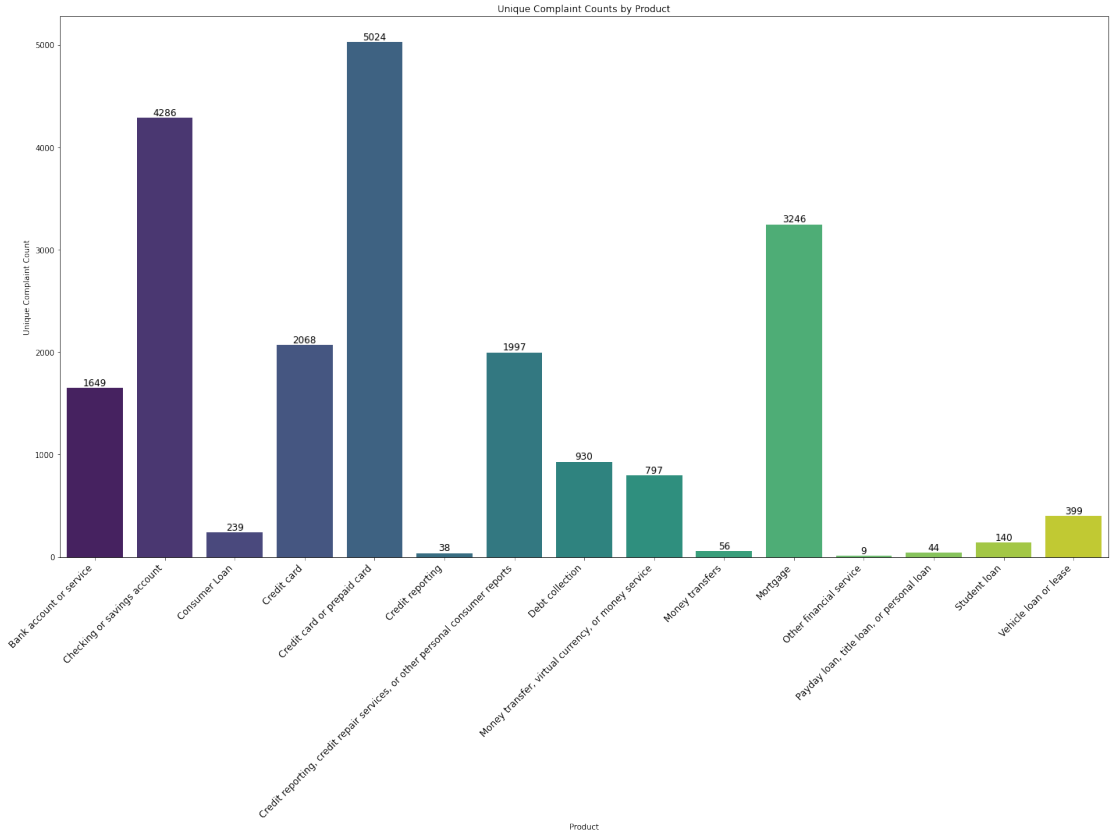

Figure 3.3: Unique Complaint Counts by Product

Figure 3.3 presents the distribution of the 'product' column over the 'complaints' column without nan values. The 'product' column has a high imbalance of data, which makes the classification harder. The unique values will be analyzed with the website and checked which product can be merged into main categories. Unfortunately, the CFPB website didn't help much because they don't have the same sub-products and it was hard to know their main categories. It is important to know which main categories they have such that the support teams can be created to solve the correct type of tickets and to have a clear structure in the system. Five different category types can be created out of the products, shown in Table 3.3.

Table 3.3: Main Category and their Subcategories

Main Category	Products
Bank Account Service	Checking or saving account / Bank account or service
Loans	Consumer Loan / Mortgage / Payday loan, title loan, or personal loan / Student loan / Vehicle loan or lease
Credit and Prepaid Cards	Credit card / Credit card or prepaid card
Credit Reporting and Debt Collection	Credit reporting / Credit reporting, credit repair services, or other personal consumer reports / Debt collection
Money Transfer and Financial Services	Money transfer, virtual currency, or money service / Money transfers / Other financial service

The imbalance is getting less by combining the products into main categories shown in Figure 3.4 without nan values. There is still some imbalance but it is a good practice to handle such data because, in real life, the data is mostly not balanced.

The 'sub_category' will be kept in the project such that the user has the choice to choose a more detailed reason for the complaint, same as the CFPB company, but there is no classification done on it.

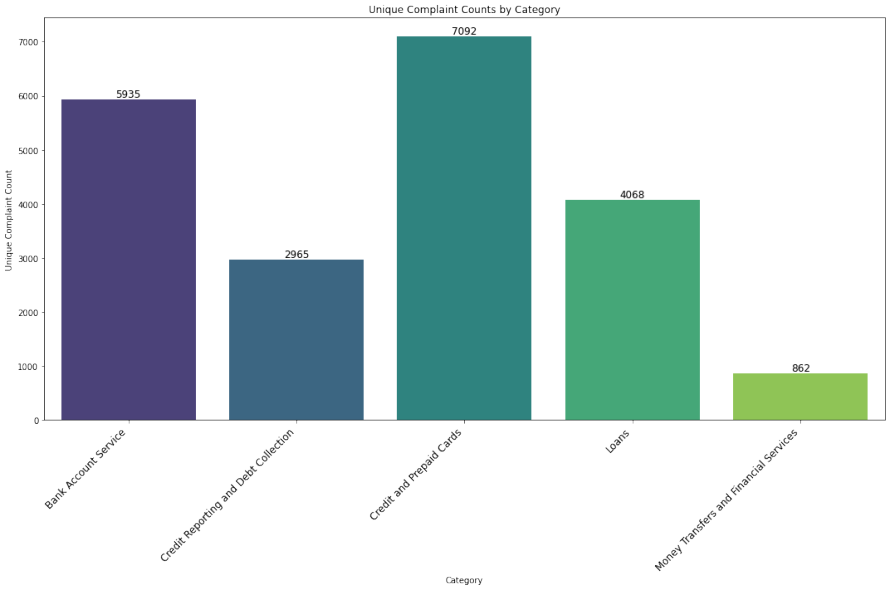

Figure 3.4: Unique Complaint Counts by Category

3.3 Classification Approach for ML Models

The following section demonstrates the pipelines of the classification process. Figure 3.5 will describe the ticket classification with traditional ML models [35], defining the baseline of the project.

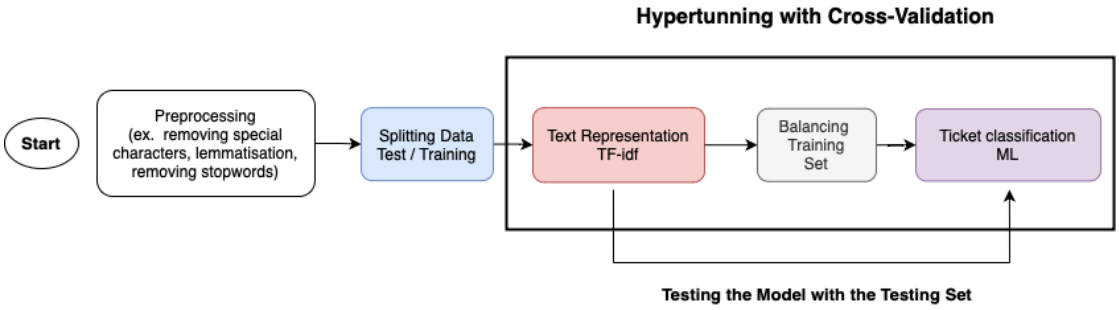

Figure 3.5: Classification Pipeline for ML Models

Data Preprocessing

The cleaned dataset will be preprocessed with the NLP methods that we mentioned in Chapter 2. The 'category' column will at first be encoded into numbers from 0-4 such that the ML models can understand the column. Preprocessing steps are applied to the 'complaints' column such as segmentation 2.11, tokenizing 2.12, lemmatization 2.15, POS tagging 2.16, lower casing, removal of stop words 2.13, special characters, single letters,

and punctuation. The sequence of the preprocessing steps is important because some are dependent on each other (ex. lemmatization can not be done before tokenization). The complaints should contain as little as possible noise data which means words that don't provide any relevant information. Stop words are one example, if most of the complaints contain 'the', the model can not use this word to separate the complaints from each other. The same with words that are not normalization (lemmatization, lowercasing stemming..), because if two complaints are from the same category and they have words that have the same meaning but are written differently for example 'better' and 'best', it will be counted as two different words. This noise will make the comparison harder for traditional ML models because they don't understand the meaning of words, that's why it is important to do the NLP steps. Table 3.4 shows a sample example of this step.

Table 3.4: Example of a Sample After Preprocessing

Category	Encoded Category	Complaint	Preprocessed Complaint
Credit Reporting and Debt Collection	0	Good morning my name is XXXX XXXX and I appreciate it if you could help me put a stop to Chase Bank cardmember services. In 2018 I wrote to Chase asking for debt verification and what they sent me a statement which is not acceptable ...	morning apreciate help stop chase bank cardmember service write chase ask debt verification send statement acceptable ...

Hypertunning with Cross-validation

Hypertunning is a method that creates a pipeline that streamlines the workflow of data preprocessing and modeling tasks. It is used to train the model with different hyperparameter combinations [6]. The model goes over each possible parameter to check which combinations give the best performance for the model. The pipeline has three steps, the first is text representation, then balancing the training set and finally training the model with the chosen hyperparameters. Before starting the hypertunning, the pipeline will use cross-validation that automatically splits the training set into a 10% validation set and partition the validation set into multiple folds (subsets). Those folds have the same size and are used to validate the model by each training epoch (number of times the entire training set passed through the model), improving the performance and avoiding under and overfitting. The steps below are included in the pipeline workflow.

1. **Text Representation Techniques:** TF-IDF text presentation method will be used for transforming the preprocessed complaints into vectors that quantify relevant words of each sample amongst the whole data (see Chapter 2). This step is done on the entire complaint, including training, validation, and test set. The table below shows a sample example of this step, with ticket (a).

   (a) **Preprocessed Complaint:** morning apreciate help stop chase bank cardmember service write chase ask debt verification send statement acceptable ...

Features	Values
debt	0.455128
cardmember	0.268102
aceptable	0.253602
validate	0.234095
help	0.231174
...	...
profesionism	0.000000
proficient	0.000000
procede	0.000000
profile	0.000000

2. **Data Balancing:** The dataset is still a bit imbalanced, and the model must have a balanced training set such that it can learn from each category the same amount of data to gather enough knowledge. Imagine the model needs to classify dogs and cats, and it gets more samples from dogs than from cats. The model would not get enough information for the cats such that it can not 100% separate it from the dogs, and as we know both animals have similar features. It is important to have enough information from both sides to distinguish them. In this case, the thesis will use the Synthetic Minority Oversampling Technique (SMOTE), which generates new samples for the minor categories as can be seen in Figure 3.6. SMOTE is looking for the samples of the minor class and using a nearest neighbors model, it will identify which samples are closest to each other. Then it uses the samples that are close to each other and generates a new one with the values between those samples (do not forget that the text representation is a vector). There is one important note, the balancing method is only used on the training set because the validation and test set need to simulate real-world samples.

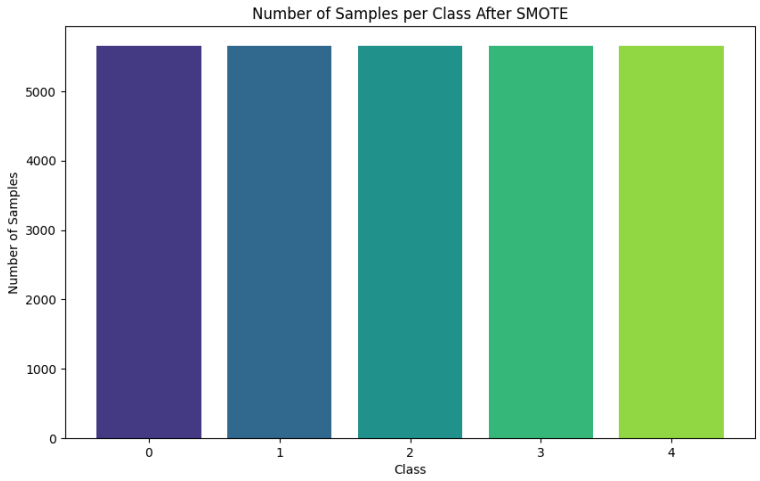

Figure 3.6: Balanced Dataset After SMOTE

3. **Traditional Ticket Classification Models:** The models are chosen based on related works (see Chapter 1). The chosen models are SVM, NB, RF, and DT. Those models will be evaluated using precision, f-score, recall, and accuracy.

3.4 Classification Approach for DL Models

Figure 3.7 will describe the ticket classification with DL models such as CNN, RNN, and Hierarchical Neural Networks (HNN), defining the baseline of the project.

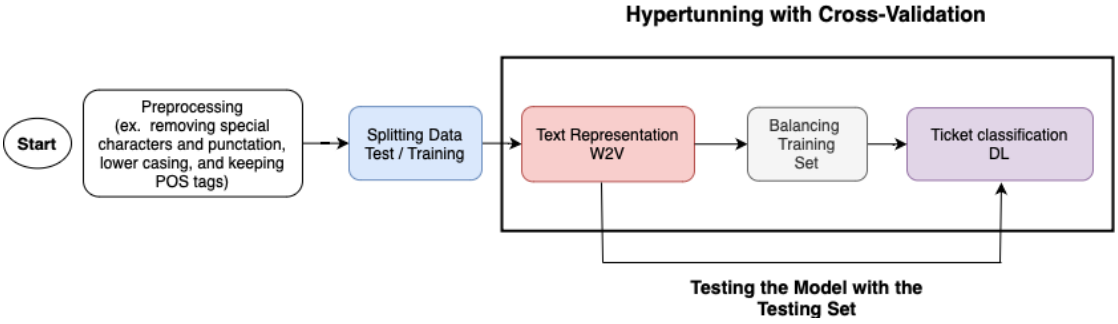

Figure 3.7: Classification Pipeline for DL Models

The thesis tried different preprocessing ways for DL models because they understand the words' context more than traditional ones (see Chapter 2). The best preprocessing way is similar to the one in the previous section, replacing lemmatization and stopword with POS tagging, keeping some original words, and still removing noise. The text representation was done with W2V (Table 3.5) for the models, even if they have their embedding methods (see Chapter 2). The classification process is similar to the one with traditional ML models, only that the hyperparameter tuning was done manually without a pipeline. The implementation workflow also included an earlier stopping function, that ends the training when it reaches a moment that looks like overfitting/underfitting. The evaluation was again analyzed with the metrics values.

Table 3.5: W2V Sample Example

Preprocessed Complaint	W2V - Text Representation of the Complaint
morning my name apreciate it you help me put stop chase bank cardmember services wrote chase asking debt verification what they sent me statement which aceptable ...	[-0.25739285 -0.24394263 -0.03696452 -0.13320279 -0.14487608 0.0213066 0.01972826 -0.3405301 -0.06535025 -0.14069006 0.1411449 0.42011803 0.02821266 -0.01000879 -0.15277132 ... 0.14448597 -0.43096408 -0.02419296 0.18687485 0.21825941 0.18185242 -0.19953424 -0.38465813 0.19869968 0.03397692]

3.5 Classification Approach for LL Models

Figure 3.8 will describe the ticket classification with LLMs [9] [11] [7] such as BERT, RoBerta, and XLNet, defining the proposed solution of the project. Different preprocessing ways were tried for LLMs because they wanted to understand the context and meaning of each word related to the whole sentence (see Chapter 2). The best preprocessing way is similar to the previous section but keeps even more of the original text. It converts words into lowercase, removing repeated characters in words and single letters. The text representation was done with their embedding methods (see Chapter 2) which is similar to W2V. The classification process of LLM is straightforward because the thesis

uses pre-trained models from the organization HuggingFace [2] by fine-tuning it with our dataset. The data balancing was done by giving more weight to the minor classes. The workflow also included an earlier stopping to avoid over and underfitting when looking at the validation and training losses.

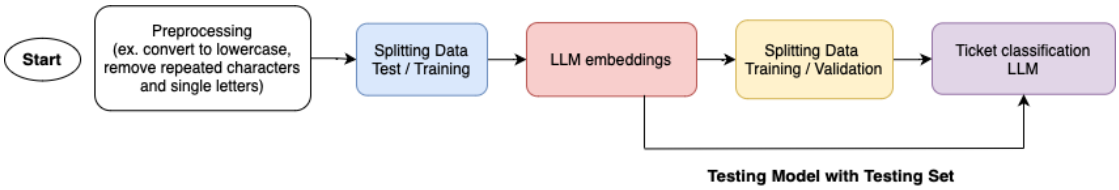

Figure 3.8: Classification Pipeline for LL Models

3.6 Experimental Environment and Model Performance

This section provides an overview of the technical tools, the model's performance, a comparison of the models with related works, and a discussion of the project decisions.

3.6.1 Experiment Configuration

The thesis was developed into the programming language Python 3.6 with the help of Visual Studio Code for efficient workflow with HPC (Hyper Performance Computers). The hardware setup comprised a machine with an Apple M1, 8 GB RAM, and the resources of HPC. The project can be found on the GitHub repository with an explanation of how to use and download the project. Table 3.6 presents some important libraries of the programming language Python that are used through preprocessing until model training.

Table 3.6: Python Packages

	Python Libraries
Preprocessing	Plotting graphs, data visualization (matplotlib == 3.3.4) / NLP processing (spacy = 3.6.1) / Progress bar, task progress visualization (tqdm = 4.64.1)
Text Representation	Text processing, tokenization (nltk == 3.6.7) / Word vectors, document similarity (gensim == 4.2.0) / ML, feature extraction, vectorization, data splitting (scikit-learn == 0.24.2) / Imbalanced data handling, over-sampling, under-sampling (imbalance-learn == 0.8.1)
ML Pipeline	ML, feature extraction, vectorization, data splitting, model evaluation (scikit-learn == 0.24.2) / Imbalanced data handling, over-sampling, under-sampling (imbalance-learn == 0.8.1)
DL Pipeline	Feature extraction, vectorization, data splitting, model evaluation (scikit-learn == 0.24.2) / DL, NN, model training (tensorflow == 2.6.2)
LLM Pipeline	Preprocessing, feature engineering, model evaluation (scikit-learn == 0.24.2) / Model hosting, access to pre-trained models (huggingface-hub == 0.4.0) / Model transformers, BERT, GPT, language models (transformers == 4.18.0) / Tensor computation, model training (torch == 1.10.2)

3.6.2 Performance Evaluation of Machine Learning Models

The performance of different ML models (SVM, RF, DT, LR, NB) with their text representation techniques (TF-IDF) is indicated in Table 3.7. The section evaluates each

[2] `https://huggingface.co`

model type, by analyzing their metrics such as accuracy, precision, recall, F-score, training time, and specific hyperparameters. The training time is calculated for the entire hypertunning of each model, which means that the 77 minutes by SVM was for training with each possible hyperparameter combination and not with a single one. DT had only one hyperparameter and that's why the training time is really low in this model. One more reason for the low training duration is that the hypertuning was launched with 2 GPUs and 12 CPUs, speeding up the process but can be trained with fewer computational resources.

Table 3.7: Results of the ML classification models

Models	Text Representation	Accuracy	Precision	Recall	F-score	Training Time (min)	Best Hyperparameter values
SVM	TF-IDF	0.828	0.826	0.828	0.826	77	C=1, gamma=1, kernel=rbf
RF	TF-IDF	0.815	0.818	0.815	0.816	9	clf_max_depth = 30, clf_min_samples_leaf = 5, clf_n_estimators = 700
DT	TF-IDF	0.696	0.708	0.695	0.701	0,586833	clf_max_depth = 20
LR	TF-IDF	0.807	0.817	0.807	0.811	14	clf_C = 1, clf_penalty = l1, clf_solver = liblinear
NB	TF-IDF	0.786	0.800	0.786	0.791	0,169666	clf_alpha = 0.5

- In terms of performance, SVM is the best model among others with TF-IDF representation whereas its training time is relatively very high.
- It can be said that RF performs well and has balanced measures as well as less training time when compared to SVM, which makes it a suitable choice because of its efficiency and robustness.
- DT has the lowest performance metrics of all models, it compensates with a very low training time making it appropriate for situations where quick results are needed with limited importance on accuracy.
- With moderate efficiency and a reasonable training period, LR will strike the balance between these two aspects to obtain good performance.
- There is moderate performance in the NB model but it is highly efficient since its training period is too short, which makes it suitable for real-time applications.

The scores in almost all models are consistent, showing a stable performance. When comparing the project model outcomes with related work, it can be seen that the paper's outcomes are a bit better than ours with an accuracy between 80-90% for RF, NB, and SVM models. The reason for a low performance can lie in the dataset and the missing

hyperparameter information from the related works. The project dataset didn't have predefined categories, they needed to be defined manually and they are imbalanced, which can lead to wrong training. If the project had any idea about the related work model pipelines then it could be easier to create the same model and maybe get a better performance. However, RF with TF-IDF offers a good trade-off between performance and training time, with low computational resources but TSs need the best accuracy.

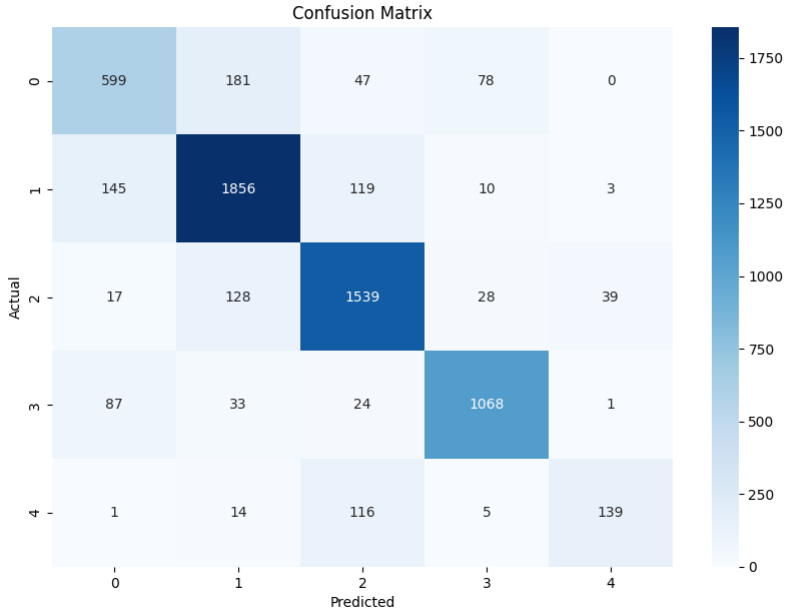

Figure 3.9: Confusion Matrix of SVM

0 - Credit Reporting and Debt Collection

1 - Credit Cards and Prepaid Cards

2 - Bank Account Service

3 - Loans

4 - Money Transfers and Financial Services

Figure 3.9 shows the confusion matrix of the SVM model, providing the overview of the predictions. When analyzing Figure 3.9, it can be seen that most complaints are classified correctly (diagonal), but there are still a lot of wrongly classified complaints like in the second column and first row with 181 wrongly classified complaints. It is important to check the wrongly classified ones, to see why the model classified them in this category. The explanation of the model's prediction will be discussed in Chapter 4). Let's now briefly explain the decision-making process of the SVM model. The goal of an SVM is to find an optimal hyperplane that divides the classes in a dataset, as shown in Figure 3.10. The distance (margin) between the classes should be large such that the classes are separated. The margin is defined by the nearest data points to the hyperplane which are called support vectors. The dataset is defined by numbers and vectors, so it can be distributed into an x-y hyperplane space. There is also a stack variable that presents

the wrongly classified point when the SVM does not separate the classes perfectly. After having the optimal hyperplane, SVM can classify new data points based on their side of the hyperplane. If the dataset is non-linear (see Chapter 2), then the SVM employs a kernel function, finding the curved boundary of the original space's projection which may pass as a hyper-plane. In the case of multiclass classification, SVM will create more hyperplanes that separate the classes.

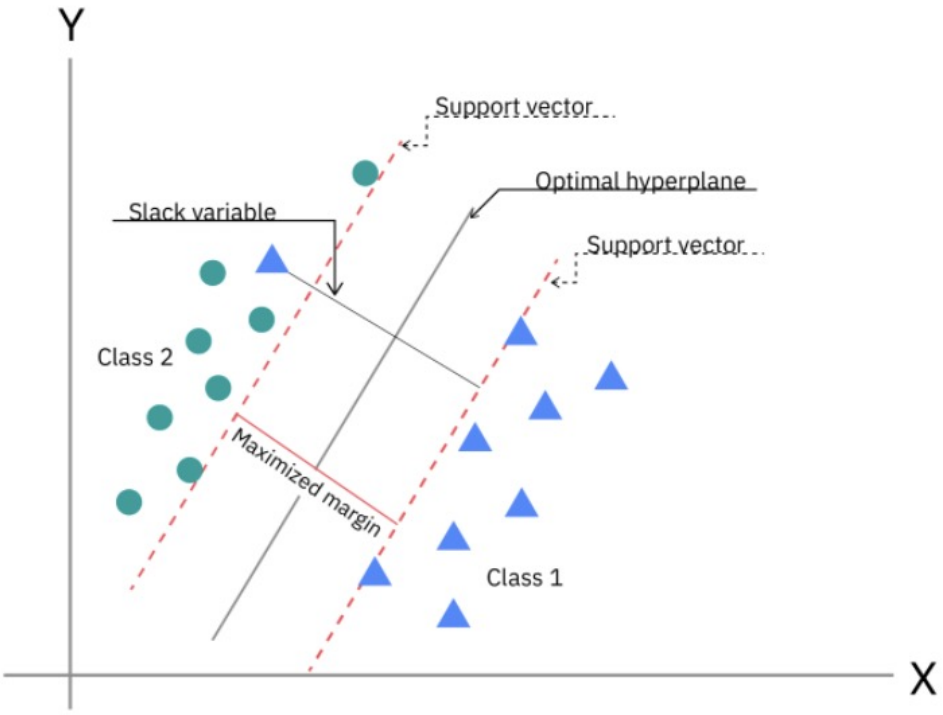

Figure 3.10: SVM Approach [3]

3.6.3 Performance Evaluation of Deep Learning Models

The performance of different ML models (CNN, RNN, HNN) with their text representation techniques (W2V) is indicated in Table 3.8. The section evaluates each model type, by analyzing their metrics such as accuracy, precision, recall, F-score, training time, and specific hyperparameters. The models are trained with 2 GPUs and 12 CPUs, speeding up the process.

- CNN has the best performance among the DL models within a shorter time of training.
- The performance of RNN is lower as compared to CNN, and it requires a large amount of training time, making it less efficient.
- HNN has the poorest performance and it requires the most training time, becoming inefficient over other models.

[3] https://developer.ibm.com/tutorials/awb-classifying-data-svm-algorithm-r/

Table 3.8: Results of the DL classification models

Models	Text Representation	Accuracy	Precision	Recall	F-score	Training Time (min)	Best Hyperparameter values
CNN	W2V	0.7924	0.8010	0.7924	0.7944	5	epochs=35, min-lr=1e-4, batch size=128, kernel=rbf
RNN	W2V	0.7658	0.7754	0.7658	0.7690	77	epochs=30, min-lr=1e-3, batch size=64
HNN	W2V	0.7644	0.7662	0.7644	0.7652	145	epochs=49, min-lr=1e-4, batch size=64

The scores are also consistent, showing a stable performance. When comparing the project model outcomes with related work, it can be seen that the paper's outcomes are better than ours with an accuracy ranging between 85-95%. The reason for a low performance may be due to the low size, structure, and imbalance of the dataset and the missing hyperparameter information from the related works. However, CNN with W2V offers a good trade-off between performance and training time compared to the other two DL models.

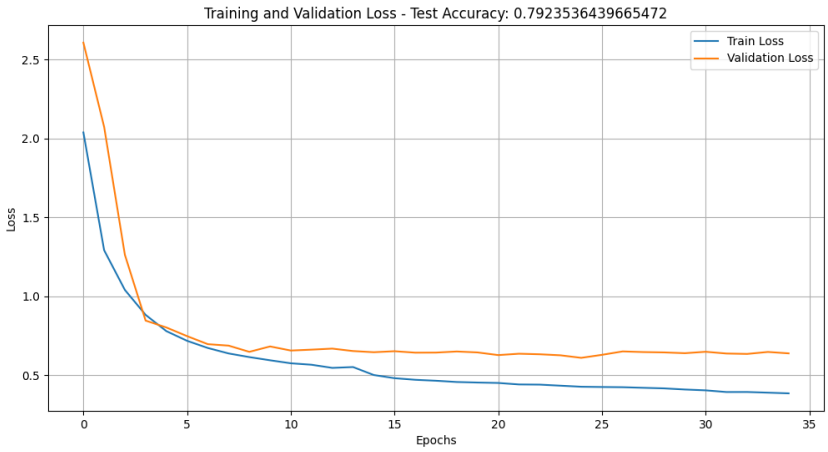

Figure 3.11: Training and Validation Loss Curves of CNN

Figure 3.11 illustrates how the training and validation loss changes for a CNN model during 35 epochs. The loss indicates how well the model learns from the data, if the loss is lower then the performance is better. The training data is well represented by the CNN model using steadily decreasing training loss starting from 2.1. On the other hand, there is a possibility of overfitting, defined by the static validation loss at point 0.7 beginning on epoch 10 until 35. The difference between the training and validation losses can also cause overfitting. The training epoch was set by 50 but the model stopped earlier

by recognizing overfitting. The performance of 79.24% is a good test accuracy (unseen data) but it can be improved with further tuning of the model to make it generalizable such as regularization techniques or adjusting the model's structure with more layers or dropouts.

3.6.4 Performance Evaluation of Large Language Models

The performance of different ML models (BERT, RoBerta, XLNet) with their text representation techniques (LLM embedding method) are indicated in Table 3.9. The section evaluates each model type, by analyzing their metrics such as accuracy, precision, recall, F-score, training time, and specific hyperparameters. The models are trained with 4 GPUs and 12 CPUs, speeding up the process.

Table 3.9: Results of the best classification models

Models	Text Representation	Accuracy	Precision	Recall	F-score	Training Time (min)	Best Hyperparameter values
BERT	Embedding	0.8225	0.8233	0.8225	0.8221	94	batch size=32, epochs=3, lr=2e-5
RoBerta	Embedding	0.8251	0.8246	0.8251	0.8244	192	batch size=32, epochs=6, lr=2e-5
XLNet	Embedding	0.8243	0.8263	0.8243	0.8234	221	batch size=32, epochs=3, lr=2e-5

- The accuracy levels of BERT are high, and its precision, recall, and F-score are well-balanced. However, the model's strong performance comes with long periods of training but compared to the others, it is shorter.
- Accuracy and F-score are a little bit higher in RoBerta, it has the highest performance but at the same time, it has slightly higher training time suggesting high computational costs like those of XLNet.

The scores are also consistent and BERT with embedding offers a good trade-off between performance and training time compared to the other two LLMs. If the dataset were larger and balanced, then the performance could be better with more fine-tuning on the pre-trained models.

Figure 3.12 shows how the training and validation loss varied over two epochs, which reflects how much the model has learned. At first, the training loss starts at around 0.78 and steadily declines to about 0.38 whereas the validation loss begins at 0.55 and declines to approximately 0.47. The fact that both losses keep decreasing consistently is an indication that it is learning from both its training data and generalizing on its validation set without showing any overfitting. On the whole, these results indicate progress in modeling while further training with a larger dataset might be needed for better performance.

Figure 3.13 shows high overfitting with the same graph but with 50 epochs. The overfitting could be because the dataset is not large enough for the model. LLMs need

a large dataset to gain a good performance. This is one disadvantage because it's hard to find large datasets but one possibility would be to generate new data samples with LLMs.

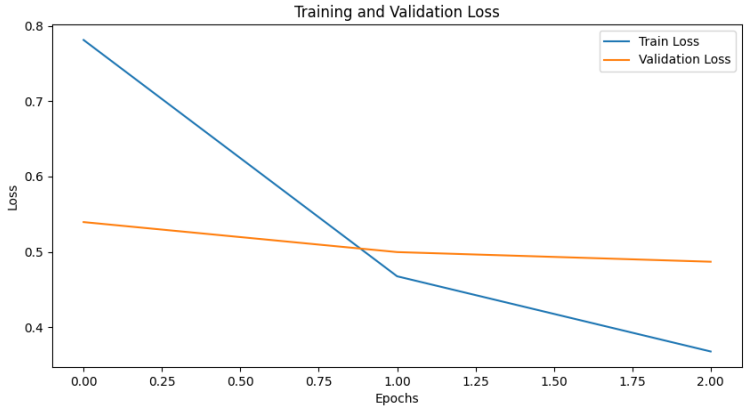

Figure 3.12: Training and Validation Loss Curves of BERT with 2 Epochs

Figure 3.13: Training and Validation Loss Curves of BERT with 50 Epochs

3.7 Conclusions and Insights

The thesis presents different models with a broad spectrum of performance metrics, training times, and computational resources, addressing and fulfilling sub-objective O3. We note that LLMs such as BERT-based models as well as those that use RoBerta or XLNet embeddings generally outperform conventional ones (except for SVM) when accuracy, recall, precision, and F-score are considered. Future work needs to be done on hypertuning LLMs and with different types of datasets.

Choosing an appropriate model mostly depends on the specific task. For instance, when accuracy is crucial and enough computation time and resources are available, advanced models such as BERT would be ideal. Others like CNN and RF provide a balanced approach with fair performance at lower computational costs. In situations where the highest possible accuracy might not be necessary but efficiency with quick solutions NB or DT could be a good fit. However, the mentioned duration of the models is for the

training but not for the testing process. The testing duration ranges from 0.33 - 34.62 seconds for the traditional model and 1-13 minutes for the advanced models.

This analysis highlights the need to consider performance metrics as well as computational efficiency in selecting models for text representation itself and classification tasks. Nonetheless, it is still possible to outperform others with traditional models because they can also be optimized in terms of their efficiency without any significant resources. The thesis decided to use SVM with the highest accuracy compared to all models, a low testing time, and low computational resources. SVM is also a traditional model, which is why they don't need a large dataset, which is beneficial for a lot of organisations. It is important to have high accuracy when classifying the tickets to avoid assigning the ticket to the wrong team and increasing the resolution time. Besides that, users must trust the system and have a good user experience. Revina et al. [28] were right in their paper when they mentioned that simple and traditional models are good enough for classification tasks, no need to make the systems more complex. This opinion was also reproduced by the user in the user studies (see Chapter 5).

Chapter 4

Explainable Artificial Intelligence

Chapter 4 goes deeper into the prediction of AI models, presenting a new approach to explain the decisions of a model. At first, it describes the meaning of XAI and its urgency in the current technological world, then it demonstrates an XAI model with examples from the SVM model outcomes.

4.1 Introduction to XAI

AI increased modern technology in many significant fields of our lives such as finance, health, and education, making it harder to imagine a world without them. It enabled the reduction of human work and delegated it to machines with a low failure rate. However, as humans integrate AI more into their daily lives and machines become more human-like, the need to understand their behavior and decision-making process is getting even more crucial. Humans rely on development books, seminary, psychology, and philosophy to gain insights into their and others' behavior to improve their decision-making, and so do machines with XAI, offering transparency and comprehension into their operations.

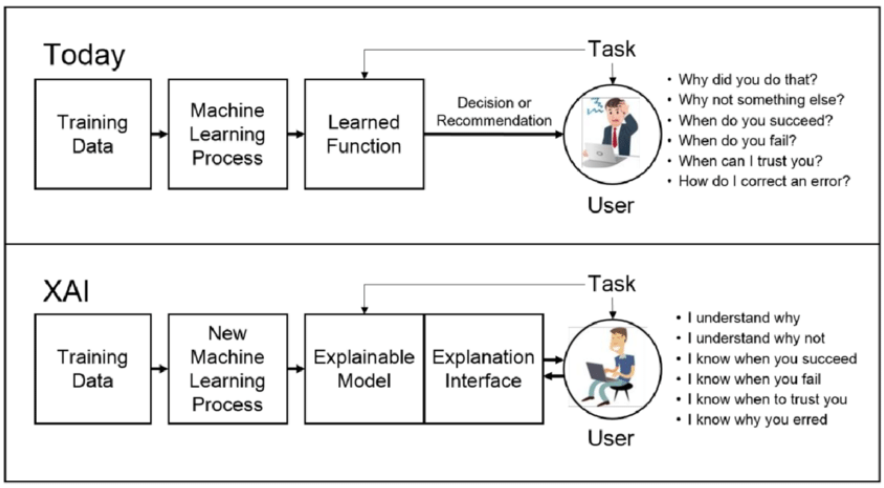

Figure 4.1: XAI Concept [1]

Figure 4.1 demonstrated the difference between simple AI and XAI. While AI approaches imitate human behavior and try to solve daily problems, XAI is the guide for

[1] https://medium.com/nanotrends/beyond-black-box-ai-237ecb6adb1a

users, developers, and stakeholders to follow the system's decision process. Looking at Figure 4.1 in the second row, the User has an interactive interaction with the system by using XAI methods, offering a direct connection between them. This Human-Machine Interaction (HMI) fosters accountability, trust, and continuous improvement to minimize the risk of failure, which can be compared to educational tools that help humans grow and make better decisions through psychological logic.

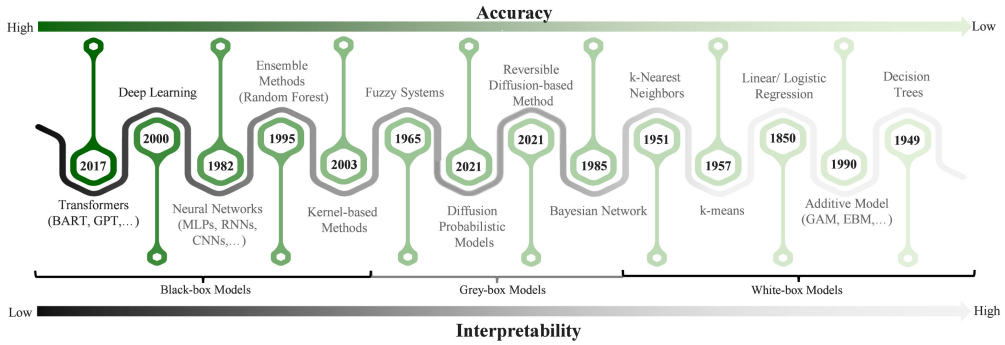

Figure 4.2: Interpretability of Models [2]

The higher the model's interpretability, the more transparency humans get to understand how they work and make decisions. The models can be categorized into three main types of interpretability, as illustrated in Figure 4.2 [2].

- **White-box Models:** Starting on the right side of Figure 4.2, we categorize the models like DT as white-box models. This category defines the models that have a high interpretability but low accuracy. They have a less complex and clear structure which is simpler to understand by humans, but because of the simplicity (see Chapter 2), the model can have low accuracy when doing human tasks like ticket classification. The results of our DT model prove the low accuracy with a value of 0.696.
- **Grey-box Models:** Those models have a trade-off between good accuracy and enough transparency, such as the NB model with an accuracy of 0.786.
- **Black-box Models:** Going closer to the modern models such as RF, SVM (kernel-based method), and LLMs with low interpretability and complex structures, delivering high accuracy around 0.82. As already explained in Chapter 2, the architecture of such models, especially of transformers, can be resource-consuming and complex depending on the task [37].

4.1.1 Key Benefits of XAI

Through XAI, humans gather various benefits [17] [14], and to understand the importance of XAI, it will be explained with a real-world comparison. The example will use the work life of an employee in a company.

- **Leader:** This will be the person who uses or works with the AI model to solve problems and complete the tasks for him.
- **Employee:** This will be the AI model and don't forget that AI tries to simulate human behavior so it can be interpreted as a human.

[2] https://www.sciencedirect.com/science/article/pii/S1566253523001148

- **Meetings:** This will illustrate an XAI approach because meetings have similar goals by connecting the leader and employee to share information about the work such that the leader can ask questions about the process, evaluate, and give feedback to the employee. Just as in XAI models, there exist different approaches for different kinds of questions the leader wants to know. If the leader wants to know about the employee's capabilities then the leader would do an interview and not a meeting with her/him, which presents an XAI approach but just for those specific questions. Further in this section, the different XAI approaches will be explained because they are different explanations and methods if the user wants to know something about the dataset the model uses or its workflow.

Let's now focus on the different advantages of an XAI model

- **Transparency:** If AI models become more transparent, then it will be clearer and simpler for humans to understand their decision-making process. Just like in job interviews, leaders will not hire a person to become an employee who doesn't convince them of his/her capabilities. The person must be open and honest, such that the leader has trust in the person and control over the process to make further decisions which brings us to the next point.

- **Trust and Control:** Having trust in the machine is as important as having trust in their employees. If the leader has the necessary information about the employee and his/her work, through interviews, meetings, or reports, then the leader has control over the process and trust in the employee that the job will be done in a good way. It is the same with AI models, they can be interpreted as our employees and the XAI is similar to the meetings or any other approach to get the information such that leaders can ask their questions about the work process and results.

- **Continuous Improvements:** Having regular meetings during the work process, is essential for getting updates, checking for any risks of failure, and improving for further development. Just like with XAI, humans can check whenever they want the decision-making process and change stuff if they are not happy with the model's result. Through regular checking with XAI methods, the model will always get new feedback such that it can improve and maximize its accuracy, same as with the meetings that are held to take further decisions or improve the existing ones.

- **Increase Adoption of AI systems:** If leaders realize that they have good communication and relations with their employees then they are more engaged to hire other employees for different jobs. XAI is this relation that can be defined in different ways (meetings, interviews, or even daily reports) that brings trust to the people and will motivate them to use AI models in their daily lives without struggling.

4.1.2 Addressing the Complexities of XAI

Besides the benefits, there are still some challenges to overcome to obtain a clear and accurate XAI [29].

- **Complexity:** The XAI methods of the models can be very difficult to understand depending on the model's complexity (ex. LLMs). Just as meetings or reports of the employees, they can get long and hard to follow when the work of the employee gets complex.

- **Accuracy:** It is hard to verify the correctness or completeness of the explanations from XAI models and if they even deliver the most relevant information about the model. This can happen when trying to follow back the decision-making path by models such as LLMs that repeat their processes and refine the data each time. Just like with meetings, it is hard for the leader to know if the insight gained by the meeting is the truth or even complete, maybe the employee forgot what exactly he/she did to reach the goal and just gives some information.
- **Computational Resources:** If the model already requires high computational resources like LLMs then adding an XAI model can increase the resources because XAI is an additional process (like generating and analyzing explanations), which can be demanding. Again if we switch to the term meeting, if an employee already has a lot of work and there is not much space for putting a meeting in between because the leader doesn't provide enough working hours or higher payment or even better tools to finish the work faster. Then adding a meeting will be tiring for an employee and that can increase the risk of errors and poor performance.
- **Generalization:** As mentioned before, it is hard to create one XAI model that can provide explanations to various user questions or situations. Mainly different kinds of XAI approaches are used for specific user questions and situations. It always depends on the AI complexity and the interest of the user. Every model is different so they also need different XAI approaches. Similarly, with a company, a leader can not combine a meeting with an interview, those both approaches are made for different reasons. A meeting is for employees that have already been hired and are working on the projects but the person in an interview doesn't get the same questions or tasks to do.
- **Integration:** The integration of XAI approaches into an existing model process can be challenging. It requires changes in the workflow of the model that can lead to a decrease in accuracy by benefiting from model interpretability. When building the model with the XAI together then the integration would be simpler and the performance would not suffer too much. Imagine the leader has a well-working employee that delivers amazing results and then the leader suddenly wants daily reports of his/her workflow and some presentations of the work. The additional work of the employee will maybe hinder him/her from focusing fully with all the energy on the project. That means that the performance of the employee will decrease and it would take more time to finish the main project because the employee needs to change his workflow. Furthermore, the leader needs to provide additional resources (money, extra work hours, support) to the employee.

4.1.3 Exploring Different Approaches to XAI

Many XAI methods [2] exist to provide specific types of explanations for various questions about AI models. There are four main levels of explanation.

- **Scoop-based Explainers:** They can provide information about the entire dataset prediction (all tickets) from the model or about specific individuals (one ticket).
- **Complexity-based Explainers:** These explainers use approaches that range from linear to non-linear interactions between features.
- **Model-based Explainers:** They focus on specific AI model types, leveraging their unique architectures.

- **Methodology-based Explainers:** Those explainers break complex models/systems into simpler and clearer steps to support humans by understanding how they work.

4.2 Application in the Ticketing System

The Thesis will focus on the Scoop-based Explainers because XAI will be integrated for the users, to understand why their ticket got classified into the specific category (considering the literature gap G4). A method called Shapley Additive Explanations (SHAP) will be used, which checks each feature (word) of the ticket and understands the impact of individual features on the model's prediction. SHAP fits the most to this project because of its benefits that are mentioned below [1].

- **Theoretical Foundation:** They use solid Shapley values from game theory (field of mathematics) to figure out how much each feature contributes to a model's prediction. The user could find out which parts of the ticket description have a positive impact on the category selection, such that they can focus on or understand why the category choice of the model is the best fit.
- **Model-Agnostic:** SHAP can be used in various models. This project compares different models and it would be more effective to implement one XAI approach for most of the classification models than for each model another XAI technique.
- **Global and Local Interpretations:** It can provide information about global and local feature importance. The local one will be used in this project because it focuses on individual users with their tickets and not on all users.
- **Feature Interaction:** It can give insight into the feature (word) interaction and how they impact together the prediction.
- **Visualizations:** SHAP can visualize its Shapley values into a diagram that indicates the feature's importance and makes it more comprehensive for users.

The XAI will be integrated with the SVM Model, automating ticket classification and decision explanation. The goal of XAI integration in the thesis proposed solution is to have an interactive user interface. The user should have the possibility to check the explanation of the model prediction. When the user isn't convinced by the explanation or the prediction then the user can manually classify his/her ticket. The interface and process will be explained in Chapter 5.

```
import shap
import numpy as np

def explain_texts(texts):
    X_tfidf = tfidf_vectorizer.transform(texts)
    background = np.zeros(X_tfidf.shape)  # Using a background dataset of
                                          zeros
    explainer = shap.KernelExplainer(loaded_model.predict_proba,
                                     background)
    shap_values = explainer.shap_values(X_tfidf)
    return shap_values
```

The code shows the implementation of the XAI model SHAP (Python Package Shap = 0.46.0) integrated into the SVM model. The explain_texts function takes the user's

tickets and lets them go through the trained TF-IDF text representation model that was used for the SVM. The new ticket gets the same presentation and the learned word vectors get applied in the new one. Then the processed ticket goes into the kernel explainer with the trained SVM predicted probabilities (how much percentages the ticket belongs to each category) of the ticket. Then the Shapley values will be calculated for each word in the ticket.

```
import matplotlib.pyplot as plt

# Simplified SHAP explanation plot
def plot_shap_values(texts, class_index=0):
    shap_values = explain_texts(texts)

    # Extract shap_values for the specified class
    shap_values_class = shap_values[0][:, class_index]

    # Filter SHAP values to include only those greater than 0
    positive_mask = shap_values_class > 0
    positive_shap_values = shap_values_class[positive_mask].reshape(1, -1
                                            )
    positive_feature_names = tfidf_vectorizer.get_feature_names_out()[
                                            positive_mask]

    plt.figure(figsize=(10, 5))
    plt.title(f'Words Impact on Classification')

    shap.summary_plot(positive_shap_values, feature_names=
                                            positive_feature_names)
    plt.gcf().patch.set_linewidth(2)
    plt.gcf().patch.set_edgecolor('black')
    st.pyplot(plt)
```

These values will be used in the plot_shap_values to create a diagram that shows the SHAP values for each word in the ticket. The diagram will only show the words that had a positive impact (positive SHAP value) on the prediction and only for the predicted category. The explainer can also show which words had an impact on the other categories and how high the impact was but that part can be done in the future to analyze the wrongly classified values deeper or even tell the user why the ticket does not fit to the other categories.

4.2.1 Analyze Wrongly Classified Complaints with XAI

Many situations benefit from XAI approaches such as simplifying the developing process for software engineers, users who are using AI in their daily lives, stakeholders who want to understand if the models fit the best for them, and also for data analysis like in this section. Chapter 3 showed the confusion matrix of the SVM model, showing the predictions it made. In this section, we will analyze two categories that had a high misclassification of complaints. The analysis will be done with the support of the SHAP model, which tells us why the SVM model predicted complaints wrongly. The SHAP model outcome will be presented as a diagram that shows the SHAP values, giving us an idea of which words had the most impact on the decision.

Figure 5.16 in Appendix B shows how many complaints of each category got wrongly classified to the category 'Credit Cards and Prepaid Cards'. The y-axis presents the number of complaints and the x-axis shows the complaint's actual category. Looking at Figure 5.16, 181 complaints from the category 'Credit Reporting and Debt Collection'

have been classified wrongly, and that's why we will have a look at one example of this category, to see if we can understand why it got wrongly predicted.

Table 4.1: Wrongly Classified Examples from the Dataset

Preprocessed Text	Raw Text	Predicted Label/Category	True Label/Category
concern unable apply bank acount typeof financial service credit score low dude account unknown use fel acounte report help	To whom it may concern i am unable to apply for bank accounts are any typeof financial service to to my credit score being lower dude to accounts unknown are not to my use i feel like its unfairly accounted and i did not do anything on my report i need help	Credit Cards and Prepaid Cards	Credit Reporting and Debt Collection

Looking at the raw text in Table 4.1, humans would also classify the complaint in the 'Credit Reporting and Debt Collection ' category because the text uses words like 'report' and 'credit'. It doesn't talk about the cards and their use. Let's find out why the model classified it wrongly.

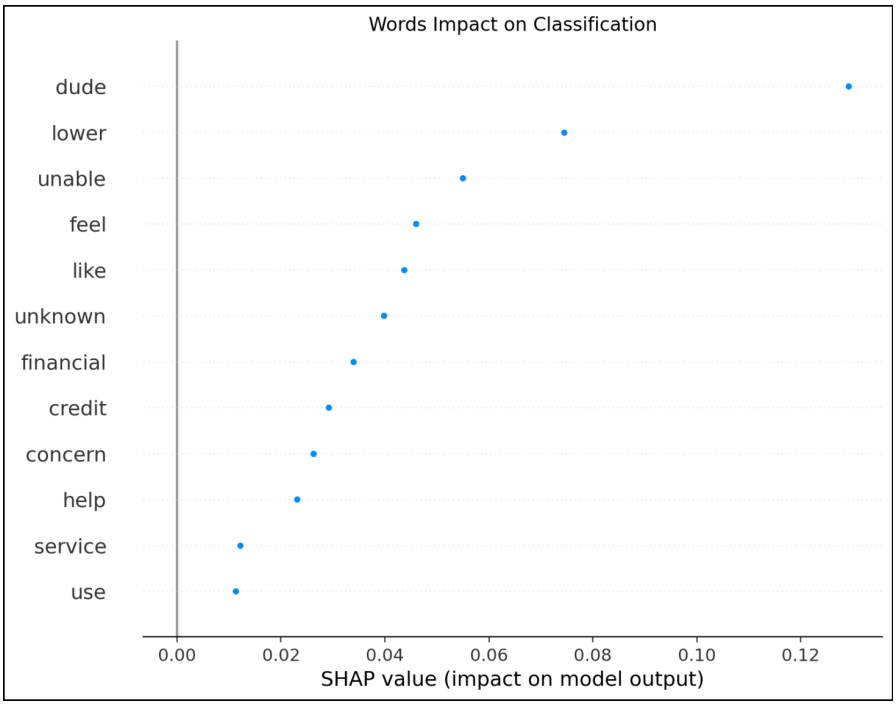

Figure 4.3: XAI Explanation 1

Figure 4.3 in Appendix C shows that words such as 'dude', 'lower', and 'unable' had the highest impact on the decision with a value between 0.05 and 0.13. This could be because most of the 'Credit Cards and Prepaid Cards' complaints have those words and this complaint has more similar words from this category than from the actual one. The diagram was created to display only words that had a positive impact on the model's decision, which means words such as 'report' and 'acounte' have a negative/no impact on the decision, which is good because 'report' should not be a common word on the 'Credit Cards and Prepaid Cards' category. Another reason could be that those two

categories should be combined as one category. They have too many similar words and the classification would be simpler. This can be done in further work when having an idea about the company's real categories.

Table 4.2: Second Wrongly Classified Examples from the Dataset

Preprocessed Text	Raw Text	Predicted Label/Category	True Label/-Category
bank fes stop payment problem cancel agrement month acepte charge couple time date date talk time helpful try conversation prof scare charge charge customer bad waste time ask questione end talk time use acount net mounth charge charge ne stop payment chase bank account want stop insuficent dolar want block payment solution replacement cart waste time money fraude chase stop suporte case bank acount number email	Chase Bank fees and stop XXXX payment I have a problem with XXXX.I cancelled my XXXX agremment two month ago and they accepted it.But they charged me a couple time (Date:XX/XX/2019 $12.00 / Date:XX/XX/2019 $15.00) .I ve talked XXXX more than ten times but they are not helpful, I tried to get my conversation more than last one as a proof but i am scaring to charged me again, They are charging me again again and again their customer servis very bad.They have wasted my time asking questiones.End of the talk, they are saying at this time you use account but next mounth we wont charge it but charged it again.So i need to stop this XXXX payments on my Chase Bank Acount ...	Bank Account Service	Credit Cards and Prepaid Cards

Same here as before, we see that in Figure 5.17 it shows that 119 complaints of 'Credit Cards and Prepaid Cards' got classified wrong into the category 'Bank Account Service'. Looking at the raw text in Table 4.2, humans would also classify it in the predicted category 'Bank Account Service ' because the text uses words like 'bank' and 'account'. It doesn't mention anything about prepaid and credit cards. Let's find out why the model classified it into this category. Figure 4.4 shows that words such as 'bank', 'acount', and 'stop' had the highest impact on the decision with a value between 0.12 and 0.23. The reason here is a bit more understandable because the complaint fits more to the predicted category than to the actual one. This can happen when a user chooses the wrong category and it isn't corrected by the helpdesk team.

4.2.2 Conclusion

AI and humans can take advantage of XAI, which simplifies the process on both sides. Humans can use it to gain an understanding of modern technology, analyze data, develop new models, and simplify their daily lives by using AI with trust for different tasks. The AI itself can improve through the feedback it gets from the users when checking the XAI of the model. They can gain the trust of the user with XAI and be a big part of the world. The thesis took the benefit of AI to analyze the wrongly classified data for further processing and improving the user experience with the TS. It helped to realize that some categories can be put together as one category because they had too much similarity through SHAP values. That recommendation and improvement will be implemented and tested in future work because of the time limit.

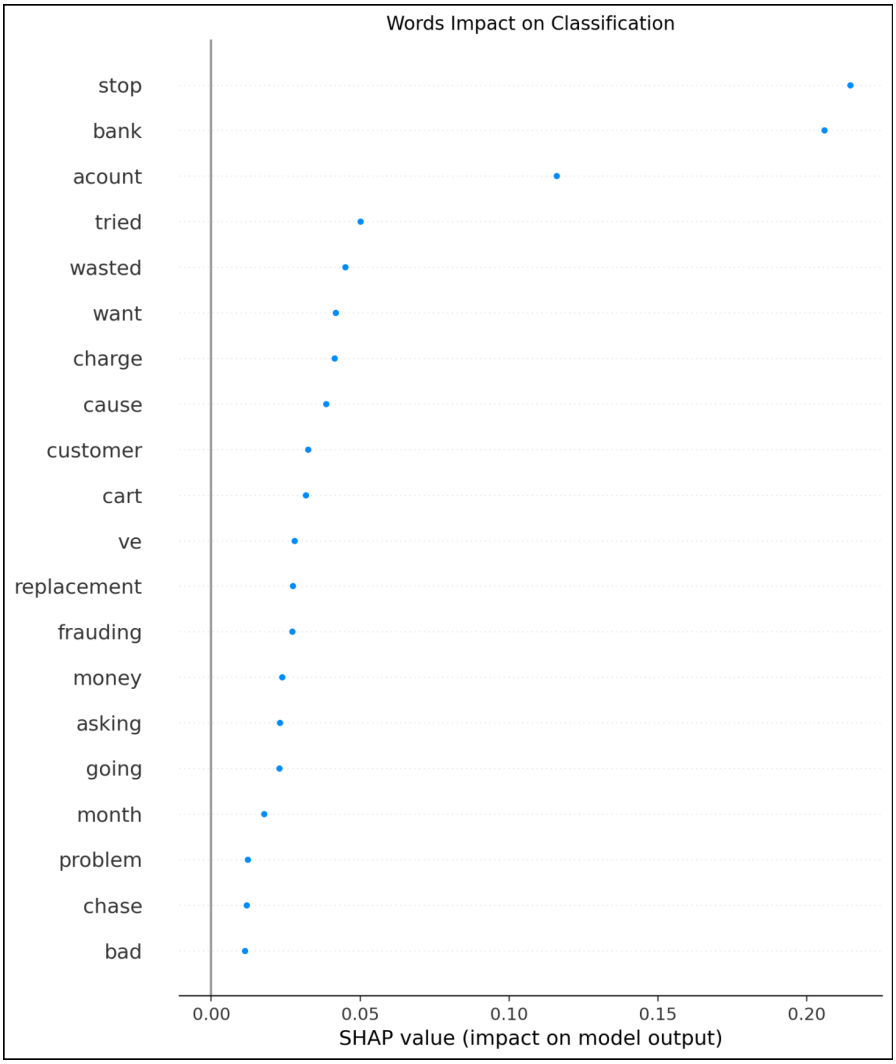

Figure 4.4: XAI Explanation 2

Chapter 5

Interactive Ticketing System with Multiple Agents

The final Chapter will discuss the proposed TS, combined with the knowledge and solutions of the previous Chapters 2 3 4. It presents the overall TS with the integration of different agents, providing optimal helpdesk support for the users. The agents are defined as different systems that manage the routing or solve the user's problems and complaints. Demo websites were developed to realize the idea and to evaluate the system with user tests.

5.1 Streamlined Ticketing Workflow

Figure 5.1 shows the overview of the proposed TS for the user. Note that this proposed solution is a demo version, it's not completely functional for a real-world situation, it needs some additional improvement for use in large organizations but it provides a good start package for small organizations. The full version with visual demonstration can be watched on this link [1] which fulfills the criteria outlined in sub-objective O4.

User Interface

The user needs to register first such that he/she can get a confirmation email that allows the user to log into his/her account.

1. **Login:** When login is successful, the user can enter into the home page where he/she gets different options to seek help.

2. **Discord and Chatbot Agents:** In the best case, the user does not go directly to the helpdesk to ask for help from the support agent but first tries to find help with the Chatbot or Discord Community. Both ways are added to reduce the amount of tickets for the support teams. The Discord community offers an integrated TS and a large number of people who may have a similar issue. The chatbot is a traditional one that a lot of people are familiar with. The user-agent communication of both tools will be explained in the next sections.

3. **Helpdesk Agent:** This is the main Agent that allows you to submit a ticket and get support from a real expert. It is not like the classical TS where the user needs to select the ticket category and fill everything on his/her own. This one offers

[1] `https://drive.google.com/drive/folders/18Mi7GyLE53tOybAQJ4XKKdokiMGyTnrU?usp=sharing`

an interactive experience with the AI and uses AI to classify, assign, and generate titles and tags for the ticket. This process and communication between the user, agent, and support agent will be explained in the next sections.

4. **User Ticket Information:** When the user's ticket is assigned and accepted by a support agent then the user will get a notification email to let him/her know about the state of the ticket. There is a page where the user can see all his/her tickets that were submitted.

5. **Ticket Resolution:** On the same page as the ticket information, there is a chat conversation with the support agent to solve the issue and finally give feedback after closing the ticket.

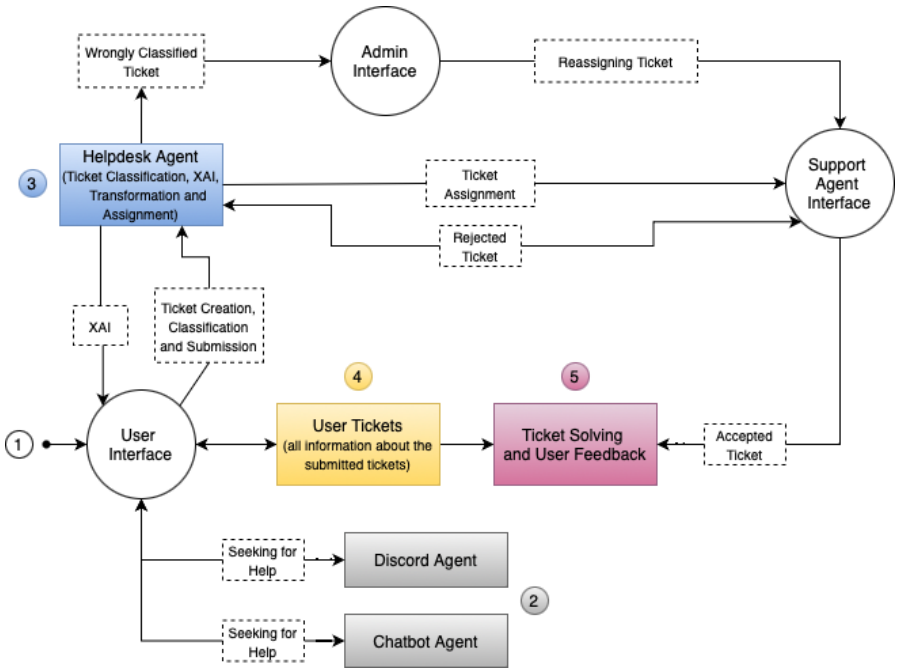

Figure 5.1: Ticket Workflow

Support Agent Interface

The process of the support agent is not demonstrated in Figure 5.1, such that it doesn't get too complex. However, when the ticket gets assigned to the right team, the support agent members can see all the tickets with the ticket description, generated ticket title, and tags. The goal of generating additional features (title and tags) to the ticket, is to simplify the overview of the tickets, such that the support agent can look at the additional features to check if the ticket can be handled fast or who can handle it better without reading the whole ticket description. Then the agent can accept the ticket or mark the ticket as wrongly classified.

- **Accept Ticket:** When the ticket is accepted by a support agent, then he/she can go to his/her accepted tickets page and solve it.

- **Mark as wrongly classified:** The team members have a group chat, where they can discuss everything and if they agree that a ticket is not assigned in the correct

category, then they can mark the ticket as wrongly classified. This ticket will be sent back, such that the admin of the helpdesk can manually classify it to the right category.

The support agent has additional tools like a translator that translates English into French or German (the company has international clients) and a user feedback page for analysis of user satisfaction.

Admin Interface

The process of the admin is not demonstrated in Figure 5.1, such that it doesn't get too complex. The admin of the helpdesk has control over everything such that there is no chaos and for security reasons. He/she can add and remove support agents, check and analyze all user feedback and tickets, can also chat with the support agents, and reclassify tickets. The admin has also control over the dataset and can always update the classification model with the new dataset or let it automatically update on its own after some months (not in the project because of time limits)

5.2 User-Agent Protocols

After having an overview of the proposed TS, we can have a deeper look into the different agents that support the user in handling his/her problem, which completes the sub-objective O5. The following sub-section will discuss the communication between the agent, the user, and the support agent (Resolver of the ticket).

5.2.1 Discord Ticket-Bot Agent

When a user has a request, he can seek assistance within the Discord community before creating a ticket, which is a communication platform for people who share similar interests (considering the literature gap G5). Additionally, Discord has a function called ticket-bot, it manages the user's question/ticket routing inside the Discord channels and this ticket-bot is considered the Discord ticket-bot agent. The discord community may help to resolve the user requests in a shorter time than through a Helpdesk Agent and to reduce the amount of tickets for the Helpdesk Agent. For instance, if a user has an urgent request over the weekend, he can interact with the Discord ticket-bot agent to potentially receive immediate assistance from the other users. To this end, we propose a user-discord ticket-bot agent interaction protocol defined in Figure Figure 5.2 for the discord community. The user has two options to seek help.

- **Public:** The user navigates to the corresponding discord category for his/her issue. If the user is not sure about the category then he/she can ask for help in the main group. In this case, each member of the discord community can see the user's issue and write on the corresponding text channel to resolve the issue together.

- **Private:** If the user doesn't want to share his/her problem with the whole community, then she/he can create a ticket with the ticketing bot Agent. The Agent will create a private channel that only the user and the member who accepted the ticket can see and write into it. If the ticket is resolved and closed then the Agent will remove the private channel and the ticket

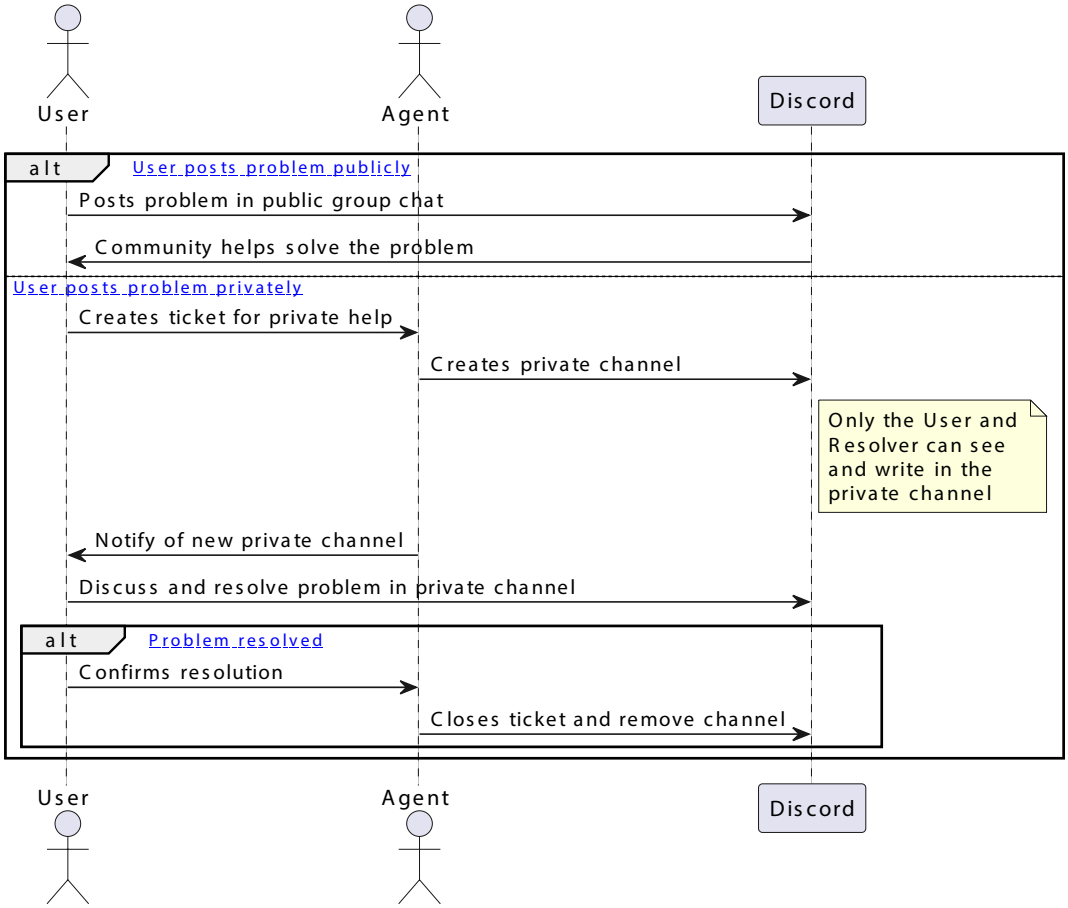

Figure 5.2: User-Agent Discord Protocol

Moreover, as the answers received by the users from the discord community are dependent on the people's voluntary participation, the proposition also includes an interaction with chatbot agents to mitigate the over-reliance on the voluntary participation of people. There is also a variety of users that don't use Discord, in that case, they can communicate with the chatbot

5.2.2 ChatBot Agent

In addition to the Discord Ticket-Bot Agent, this thesis introduces a ChatBot Agent that enables users to interact with a chatbot powered by LLMs. The motivation is to create more natural and intuitive conversations between the user and the machine. If the user doesn't have a discord account or doesn't want to use it, then the user has the chatbot option to get a solution quickly to his/her issue. The integrated chatbot aims to support the user for faster resolution and decrease the ticket amount for the Helpdesk. To enable the User-chatbot interaction protocol, this thesis focuses on the classical user-chatbot interaction widely defined in literature by several authors such as Jenneboer et al. [18], Basha [5]. Moreover, as the answer received from the chatbot may not be certified and fully accurate, the proposition, therefore, also includes an interaction with the Helpdesk Agent.

5.2.3 Helpdesk Agent

If the above-mentioned approaches do not resolve the user's issue, then the user may proceed to create a ticket through the company's helpdesk. The user will have an interactive experience with the Helpdesk Agent which is defined in the Helpdesk Agent Protocol in Figure 5.3 by considering the literature gaps G2 and G3.

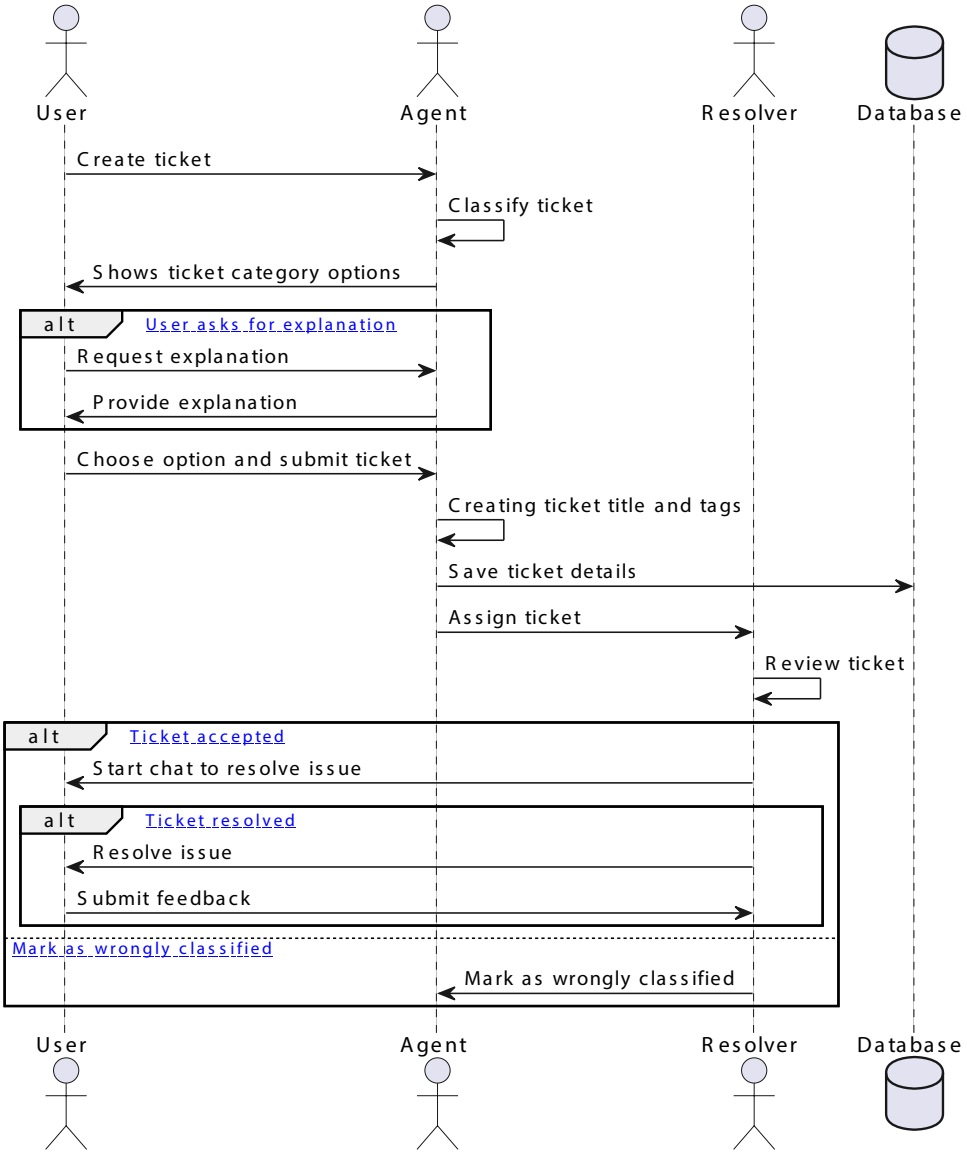

Figure 5.3: User-Agent Helpdesk Protocol

When using the Helpdesk Agent, the user can create a ticket related to his/her issue. Simultaneously the ticket description will be delivered to the Helpdesk Agent, where the Helpdesk Agent will process the ticket classification and return the category type (e.g., 'Money Transfers and Financial Services') to the user. Additionally, the user can get an explanation of the category decision from the Helpdesk Agent by an XAI method (which shows a diagram of the ticket world's impact on the decision). Then the user has the option to choose between the given category of the Helpdesk Agent or choose another category. After choosing the category, the user can finally submit the ticket to

the Helpdesk Agent for additional ticket transformation (adding tags and titles to the ticket). Then the Helpdesk Agent will save the received ticket to the privacy database and assign the ticket to the corresponding Resolver (Support Agent). Finally, the Resolver will review the ticket to check if he/she can resolve it. If the Resolver accepts the ticket, then he/she contacts the user to have a discussion about the ticket and resolve his/ her issue. When the ticket is resolved, the user can directly submit feedback to the Resolver. On the other hand, if the Resolver doesn't accept the ticket because it's been assigned to the wrong category, then the Resolver can mark the ticket as 'wrongly classified' and send it back to the Helpdesk Agent. The Agent will send the ticket to the Admin, who has control over all tickets and Resolvers (this part is not shown on the protocol because of simplicity). The Admin will look over the ticket and reclassify it to the correct category. Our interaction protocol is dynamic and after a specific time, the Helpdesk Agent can be updated its classification model by using the new data to enhance the Helpdesk Agent's performance.

5.3 Study Design and Hypotheses

The evaluation of the proposed solution is done with user tests on both sides (user and support agents), meeting the requirements of sub-objective O6. The setup of those user tests will be explained in this section, including the tools that are used to create the experimental environment.

5.3.1 Experiment Configuration

The websites for the project were developed in the programming language Python 3.6. In Table 5.1, you can see the different tools and Python packages that are used for the realization of the different components.

Table 5.1: Packages and Tools

	Tools
Website	Simple Web-App creation without HTML / Streamlit == 1.36.0 / Secure Database Management for the tickets and login informations / SQLAlchemy == 2.0.30
User Questionnaires (UX)	User Feedback Collection with simple structure and user-friendly interface (Anonymous surveys) - Google Forums
Ticket Transformation	NLP Models for generating title and tags - Transformers == 4.42.3 / Provides Pretrained Models - Huggingface-hub == 0.23.4
Discord Ticket-Bot	Ticket Management for Discord without code - ticketsbot.net
Chatbot	AI Automated Conversation without coding (only the company website link as a knowledge base) - FastBots.ai

5.3.2 User Interface

The user website of the proposed solution is tested by twenty-five participants with an average age of 27 years and a standard deviation of approximately 4.90 years. The user needed to do the test on their own at home and it's a quantitative user test. The objective is to evaluate the usability of an interactive TS Figure 5.5 compared to a traditional system Figure 5.4, to gather user feedback on new features designed to expedite ticket resolution. The test setup involves the two websites and user questionnaires administered via Google Forum, with a third questionnaire used for comparison. The experiment lasted about 20-30 minutes by completing each task and fulfilling the questionnaires.

Traditional Ticketing System

Figure 5.4 shows the user test process of the traditional TS. The user logs in with the given password and username. Then he/she goes to the helpdesk to create a ticket by copying the given ticket description and pasting it to the corresponding field. The user should read the ticket to be able to select the corresponding category, subcategory, priority, and title for the ticket. Afterward, he/she can submit the ticket and go to the ticket Information page, to have a look at the submitted ticket. The page already provides an example of a closed ticket, an assigned ticket, and a solved ticket. The user can now go to the ticket that was assigned and try out the chat function which is like classical chats. Finally, the user can provide feedback for closed tickets and log out to fill the UX in the Table. The first questionnaire 5.2 is about the Baseline and composed of 16ten questions that include Demographic Information, Ticket Submission Completion, Ticket Information, Chat Functions, and Overall Experience, which are mainly scale ratings.

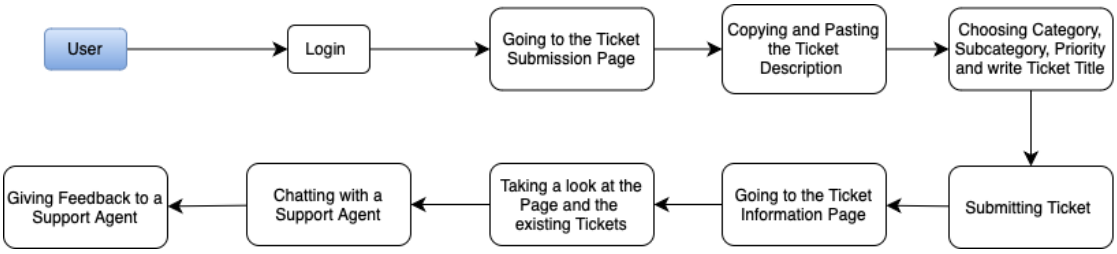

Figure 5.4: User Test Process of Traditional TS

Table 5.2: UX of User Test for Traditional TS

Category	Questions
General Information	Do you agree with the above requirements? / How do you feel at the moment? / How familiar are you with ticketing systems?
Demographic Information	Select your Age range / Select your Gender / What is your Profession?
Ticket Submission Completion	Was the ticket submission process straightforward? / How much did you like the process of selecting the appropriate category and subcategory for the ticket? / How much did you like the process of writing your title for the ticket? / Comments?
Ticket Information and Chat Functions	How effective was the chat function in allowing you to complete your tasks? / How much did you like the feedback function? / Comments?
Overall Experience	Overall, how would you rate your experience with the ticket creation process? / How likely are you to recommend the system to others? / Comments?

Interactive Ticketing System

Figure 5.5 shows the user test process of the interactive TS. The user logs in with the given password and username. Then he/she goes to the helpdesk to create a ticket by copying the given ticket description and pasting it to the corresponding field. The user should read the ticket to be able to select the corresponding category, subcategory, and priority. The system displays the category and the user should now check the explanation of the prediction which is visualized by the diagram. Afterward, he/she can select which category and subcategory suits the best and submit the ticket. Then the user can try out the Chatbot and Discord Community, some users may not have Discord. When they choose to try out the Discord community then they can test the TS inside Discord. Finally, the user can fill in the UX defined in the Table 5.3

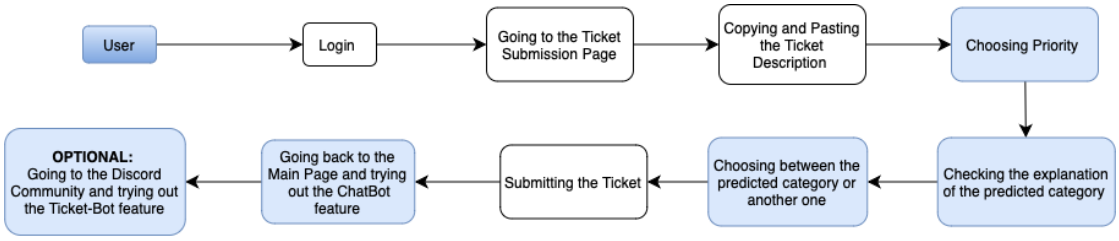

Figure 5.5: User Test Process of Interactive TS

The second questionnaire 5.3 is about the interactive website that consists of 18ten questions which are similar to the Baseline questionnaire but with some additional questions about the interactive side with the Agents.

Table 5.3: UX of User Test for Interactive TS

Category	Questions
General Information	Do you agree with the above requirements? / How do you feel at the moment? / How familiar are you with ticketing systems?
Demographic Information	Select your Age range / Select your Gender / What is your Profession?
Ticket Submission Completion	Was the ticket submission process straightforward? / How much did you like the process of selecting the appropriate category and subcategory for the ticket? / How useful was the 'Explainable AI' diagram in understanding the system's choice? / Comments?
Additional Features	How effective did you find the idea of the Chatbot? / How effective did you find the idea of the Discord Community? / How easy was it to navigate to the 'Credit card and Prepaid Card' category and create a ticket in the 'ticket' channel? (If you tried) / Would such extra features help you solve your ticket faster? / Comments?
Overall Experience	Overall, how would you rate your experience with the ticket creation process? / How likely are you to recommend the system to others? / Comments?

The last questionnaire 5.4 for the comparison has 12 questions about Demographic Information, Ticket Submission Completion, and Open-ended Feedback.

Table 5.4: UX of User Test for TS Comparison

Category	Questions
General Information	Do you agree with the above requirements?
Demographic Information	Select your Age range / Select your Gender / What is your Profession?
Ticket Submission Completion	Which ticketing system was more straightforward? / Which ticketing system was more intuitive in choosing the appropriate category and subcategory for the ticket? / Comments?
Open-Ended Feedback	What did you like the most about each system? / What improvements would you suggest for the systems? / Which one do you prefer? / Why do you prefer the selected one? / Any additional comments or suggestions?

5.3.3 Support Agent Interface

Another study was done on support agents, which was a qualitative test. The objective is to evaluate the effectiveness of new features for support agents and gather diverse perspectives on their experiences with their TS's. The test setup includes two websites, user questionnaires administered via Google Forums after each session, a third questionnaire for comparison, and a person-to-person interview which was semi-structured.

Two participants, aged 20-30 years, took part in the test which lasted 50 minutes to 1 hour. The first Website was about a traditional TS on the support agent side 5.6 and an improved one 5.7. The questionnaires were similar to the user test and the interview questions were semi-structured which means fixed questions with open-minded questions. The questions were structured to gather information about the user's experience with their organization's TS and their work process.

Traditional Ticketing System

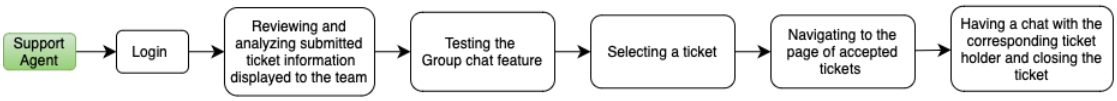

Figure 5.6: Support Agent Test Process of Traditional TS

Figure 5.6 shows the support agent test process of the traditional TS. The support agent logins with the given password and username. Then he/she can check the tickets that are assigned for his/her team and those tickets do not have any tags, just the title from the tickets. After looking over the ticket structure the support agent can test the group chat function and accept a ticket. When a ticket is accepted, the support agent can navigate to the ticket page where only the corresponding support agent can see the accepted tickets. He/she can now look over the page and play with the chat function, then close the ticket. Finally, the user can log out and fill in the UX.

Improved Ticketing System

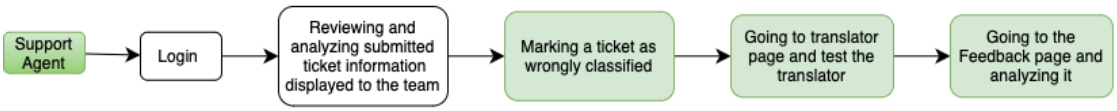

Figure 5.7: Support Agent Test Process of Improved TS

Figure 5.7 shows the support agent test process of the improved TS. The support agent logins with the given password and username. Then he/she can check the tickets that are assigned for his/her team and those tickets now have tags from the tickets. After looking over the ticket structure the support agent can mark a ticket wrongly classified and the wrong ticket can be recognized directly through the tags. Then the support agent can play with the translator and the user feedback page. Finally, the support agent can log out and fill in the UX.

5.4 Findings and Observations

The most important outcomes of the thesis are displayed in this section. It provides the results of the previously described user tests and the discussion about the user experience and future improvement.

5.4.1 User Test Outcomes

System Overview

The interactive system is preferred by the majority (72%) of users as shown in Figure 5.8, mainly due to its user-friendly interface and time-efficient functionalities. Users found it beneficial that the interactive system could automatically classify tickets as well as the option to maintain control over the system. The rest of the users found that the page was too overwhelmed with different options.

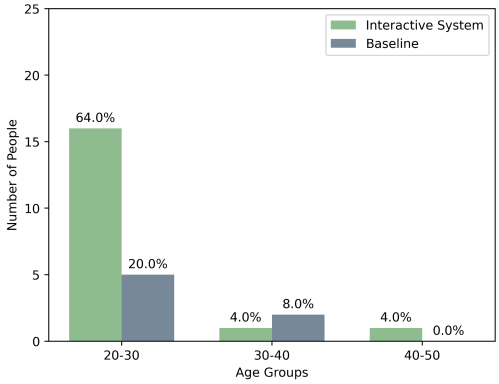

Figure 5.8: Preference for Systems by Age Group

A Mann-Whitney test was done to compare three parts of the systems shown in Table 5.5, to understand the reason behind the Figure 5.8. The Mann-Whitney test can help to indicate if there is a significant difference between the two websites on the specific parts. To this end, the U and p values need to be evaluated together to get an understanding outcome. The U-value tells us if there is a difference and the p-value tells us how significant the difference is.

Table 5.5: Mann-Whitney test

Comparison	U statistic	P-value
Straightforwardness of the Ticket Submission Process	402.0	0.125
Category and Subcategory Selection	429.5	0.033
Overall Experience Rating	433.0	0.032

When the p-value is larger than 0.05 (standard) then there is no significant difference between the websites for example for the straightforwardness of the ticket submission process in Figure 5.9. If the p-value is smaller than 0.05 there is a significant difference between the category and subcategory Selection in Figure 5.10 and the overall experience rating in Figure 5.11. That means that the rating on the category and subcategory selection and overall experience is higher in one of the systems. When looking at Figure 5.10 we can see that the interactive system has an interquartile range (IQR) between 4 - 5 which means that most users gave a high rate to the category and subcategory selection compared to the traditional one with an IQR between 3-4. The average rating on both systems is 4 out of 5. The overall experience Figure 5.11 is also higher by the interactive one with an IQR between 4-5 and a median rate of 5, which is the maximum rate. The traditional on the other side has a lower median rate of 4 and IQR between 3-5. Table 5.6 shows some positive comments from the user for both systems.

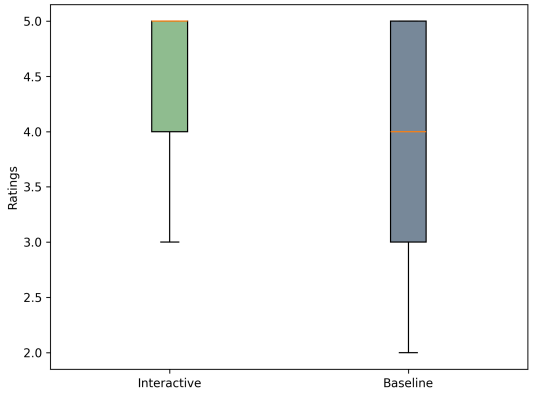

Figure 5.9: Straightforwardness of the Ticket Submission Process

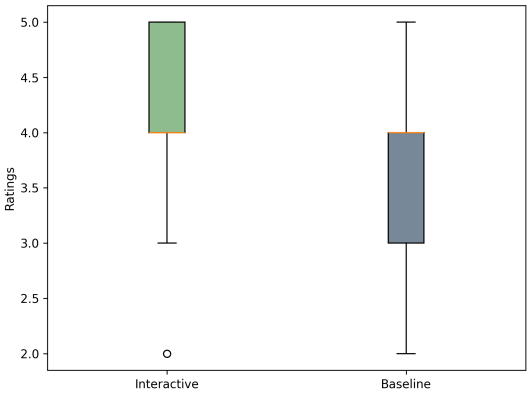

Figure 5.10: Effectiveness of Category and Subcategory Selection

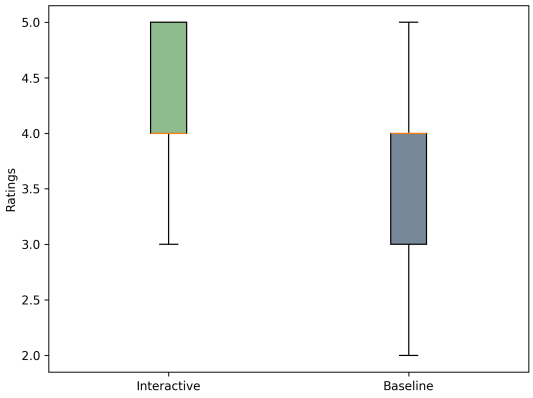

Figure 5.11: Overall Experience Rating

Table 5.6: User Comments on Both Helpdesk TS's

Interactive System - Positive Comments	Traditional System - Positive Comments
Interactive Ticketing System was more user friendly by automatically determining categories but also giving the flexibility to change in case of misrepresentation of the user's needs.	I think the choosing the category myself was more straightforward. Maybe due to the UI, the thing that was generated after my input took more effort to figure out what it was doing, then just clicking a category myself.
The interactive ticketing system was a lot just clicking a category myself. better because AI already had chosen the category for me, making it easier to select an appropriate subcategory. It facilitates my life and makes using the ticketing system less of a pain.	The structure was very clear and easy to use
Raising a ticket using interactive ticketing system was easy and time saving. No need to think about anything extra as it was done by the AI model and the best part was the explanation about choosing a particular category.	The first one was very straightforward like any ticket system.
The automatic category assignment facilitates the use of the ticketing system	The first was fast and easy

XAI Overview

Figure 5.12 indicates that users between 20-30 years found the XAI informative and helpful with a median rating around 4.0 and an interquartile range (IQR) between 3.0 to 5.0. The distribution on the other side is relatively spread out and shows a variety of opinions. In general, the median rate in all age groups is around 4.0 and the IQR is mainly placed in the middle around 3.0 and 4.0. Looking at further textual feedback, people without much IT knowledge found it confusing, not important, or felt that it did not contribute to their understanding of the process.

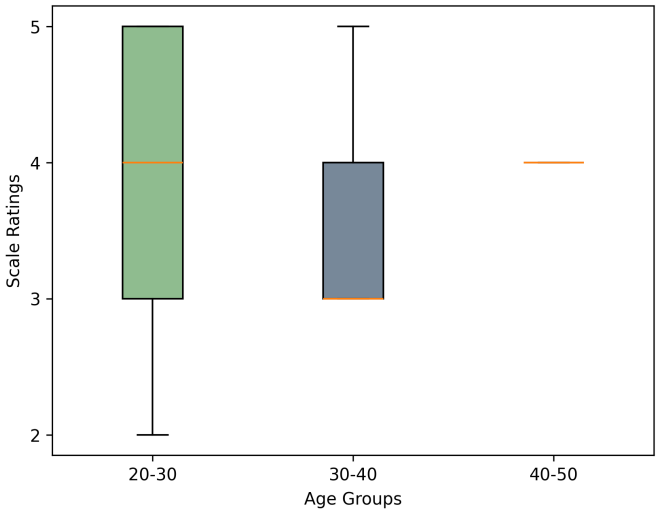

Figure 5.12: Distribution of XAI Effectiveness by Age Group

Table 5.7 shows positive and negative comments on the XAI function that the users had. In general, the user had difficulties understanding the XAI diagram or just was not interested in knowing it because of its additional work and they just wanted to solve the problem. One of the users was right to just separate the XAI from the ticket workflow.

Table 5.7: User Comments on XAI

XAI - Positive Comments	XAI - Negative Comments
If you are not inspired, then AI can be a good thing ahah... it is also very useful to see how the AI is accurate!	I find it hard to understand at first maybe because I don't have enough knowlegth in this kind of thing.
It is interesting for computer scientists to know what led the AI model to choose that specific category based on a word.	I believe most people will not understand or care about the diagram that explains how AI decided to choose that category. Maybe showing the impact of each word as a percentage using a bar diagram might make it more 'user friendly' for people that are not familiar with computers. Especially because most people won't know what SHAP value is and how to interpret is, but a percentage everyone knows what it is.
The Explainable AI diagram was pretty informative as it tells the user why the ticket falls in that particular category.	'Explainable AI' diagram might be useful for someone interested in the topic but I don't believe this would be advantageous for regular customers.

Discord and Chatbot

Users had a generally positive opinion about the chatbot and Discord community as additional features. The chatbot effectiveness in Figure 5.13 has an overall positive

distribution with a median value larger than or equal to 3. The IQR shows that the chatbot is preferred by young people with values between 4-5. In further questionnaire analysis, there was a suggestion to integrate the chatbot on every page. Regarding the Discord community, Figure 5.14 tells us that there is potential in the idea of a community with an IQR value between 3-5 and a median value between 3-4. People in the age range between 20-30, mostly students, understand the importance of working in groups and helping each other. However, there was an acknowledgment that Discord is primarily for gaming and development, which might limit its use for a broader audience. The chatbot has a better impact on the user because of the direct and easy access.

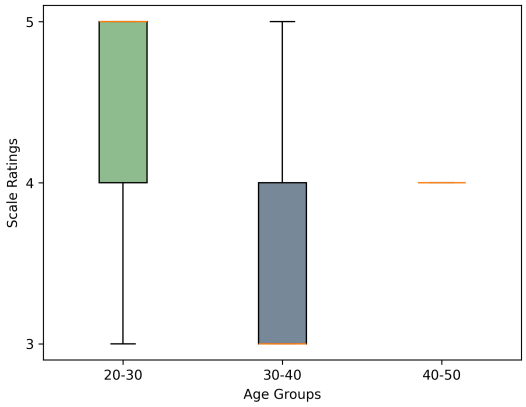

Figure 5.13: Distribution of ChatBot Effectiveness by Age Group

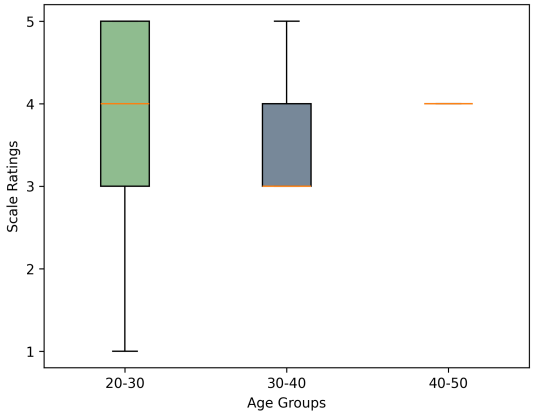

Figure 5.14: Distribution of Discord Effectiveness by Age Group

In general, the chatbot and Discord had more positive aspects 5.8 than bad ones, which shows that both have potential in a TS. Figure 5.15 shows us that 56% of users would like to use Discord and 36% as maybe.

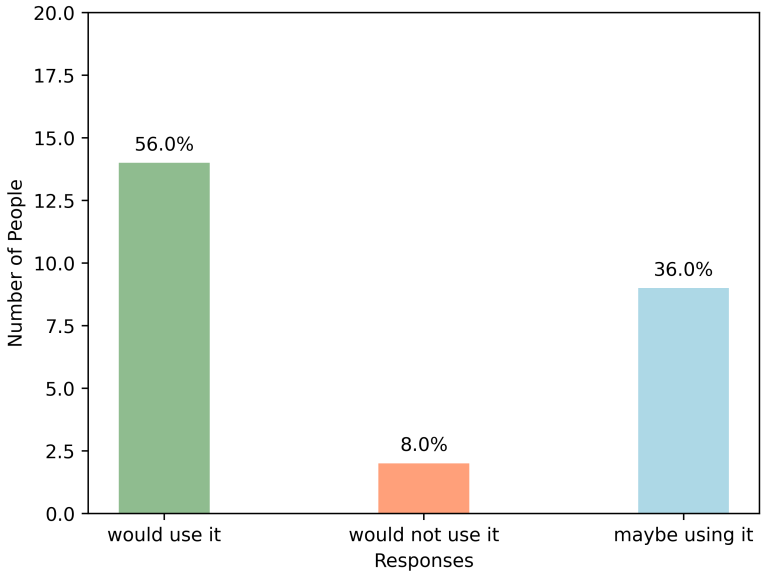

Figure 5.15: Usage of Discord

Table 5.8: User Comments on Discord and Chatbot

Discord and Chatbot - Comments
Discord is a very good idea
I really like communicating with others to help me solve my problem
Very few people outside of the gaming and development sectors have discord accounts.
Discord is key
Chatbot help a lot

5.4.2 Support Agent Test Results

The support agent test is a qualitative user test that is defined by the project's website and a semi-structured interview. Two support agents participated and gave us a view into their TS's. They found that the project's TS for support agents is user-friendly by clearly understanding the main content of the ticket by its title and tags. They find that the additional tool like the translator is a good idea but not necessary. The key findings from the interview of their organizations' TS's experience can be seen in Table 5.10. It shows two completely different organizations that have other needs and types of clients. The first Support Agent works in a hospital and offers its service to the hospital employees. The second one works for a banking software company named Sopra, which is focused on clients outside of the company who pay for the service they get. The support agent also has other ways of helping users, the first has most of the time direct contact with the users and the second does it over the computer. That's why both have a different opinion about user feedback, the first one mentioned that it's not necessary and the second one thinks it is one of the most important features. Both Support Agents use Jira as a ticketing platform that offers a wide range of good functions to handle tickets, such as the knowledgebase and the possibility to send the ticket as an email to the corresponding user. The final discussion about the results can be found in the next section.

5.5 Discussion

Simplicity is the key for users, such that they get what they want in a short, easy, and clear way. The user study results showed that implementing AI in TS can have benefits, but only in situations where it offers direct support such as chatbots (5.9 ID 1) and automated ticket classification. In situations where they get additional information that is not necessary for the user to finish its task such as the XAI(5.9 ID 4,6), are not well seen by the users. This method just takes space away and overfills the page which reduces the clearness. XAI [17] would be more useful for developers or people who need to understand the AI tool to continue working (5.9 ID 5). The users like a website that contains just the necessary information and is simply structured (5.9 ID 2, 3). In this case, Chatbots [19] are ideal because they can assist and produce all the additional information (XAI results) whenever the user asks for it. The chatbot function also meets the needs of 61% of users who prefer faster response times [2] and each page should have access to the chatbot to keep the simplicity. The discord community [23] also reached a lot of users, who helped them to support each other and have a feeling of not being the only one with the problem. The project chose Discord because of its implemented ticketing bot that offers a TS to solve issues in private, which other community tools don't have. Besides the mentioned outcomes and personal experience, Discord is a well-known platform that

[2] https://www.revechat.com/blog/online-customer-service/

is used worldwide with around 154 million users per month [10] [31].

Table 5.9: User Suggestions on Helpdesk TS

Comments ID	Improvements for the Interactive TS
1	The chatbot could be integrated more into the website. For example, having a sidebar on the right side where the user can always use the chatbot, no matter in which window he is at the moment.
2	Maybe the tickets that have been completed but feedback pending and active ticket can be put in different categories for easier navigation to open tickets
3	To improve how the explainable AI shows the information. Make it more appealing to people that are not familiar with CS and easier to understand (like the diagram etc)
4	Take advantage of the streamlining of the interactive system and reduce clutter on the page, have a separate section for deep diving into the ml results, separate from the main workflow.
5	The Diagramm is good just for a person that can really read a diagramm I find it hard to understand at first. Maybe if I put my mouse on the points they could give me one really easy sentence of explanation that would be really helpful for someone who can't read a diagramm
6	the second one, I think if the UI would be different, when I put my input and it shows me directly that it has chosen a category for me instead for me to read through all the little text what is happening. In other words, it took me more effort to figure out what the generated text after the input was than manually selecting the category like in the first system.

Now going to the support agent side, depending on the organization's needs, there are different ways to create effective TS's as analyzed in Table 5.10.

- **Organizations that offer services for their employees:** Mainly they don't care about employee feedback on the support service because the tickets get handled inside the company and the employees mostly don't interact with the system. The organization is focused on reducing the ticket resolution time such that the employees can continue to work. The organization should have features like automatic ticket solving through advanced AI and automatic ticket prioritization/classification to speed up the resolution time.

- **Organizations that offer services for their clients:** User feedback is highly recommended in such organizations, to analyze and make statistics about the client's experience such that they keep paying for the service. The focus lies on the clients, and how to keep them and attract more. According to the feedback, they can improve the TS. The improvement can be the same as for the above organization and even implement a community tool like Discord to reduce the number of tickets. The community can let the clients feel like they are a part of the organization and that they are not alone with the problem. There are always people who have gone through the same process and the clients will feel more comfortable.

Table 5.10: Support Agent Test Key Findings of their Organisation's TS

	First Support Agent	Second Support Agent
Position	CHL (Hospital) It Support	IT assistant in Sopra Banking Software
Type of Tickets	Incidents, hardware issues, desktop service	Incidents, software issues, installation issues
Working experience	7 years	2 years
TS	Jira	Jira
Users of their helpdesk	The employees inside the hospital	Banks and their employees
Ticket Creation Types	Per email, phone call, or website	Per email, phone call, or website
Ticket Handling	Mainly person to person, the support agent is in direct contact with the user	Mainly in their office, through their computer without having direct contact with users
Ticketing Routing	Best case: Customer calls service desk to create ticket because customers have not enough knowledge about the technology 1. The Service Desk assigns tickets to the right category/group 2. Group members have the same skills and take any ticket from the queue, of course by checking the priority. Note: With their working experience, they know how to manage the tickets, to solve them faster	Best case: Customers create their tickets on the website because the customer mainly knows what exactly the problem is and the technology. 1. The Service Desk assigns tickets to the right category/group 2. Group members have the same skills and take any ticket from the queue, of course by checking the priority Note: With their working experience, they know how to manage the tickets, to solve them faster
Overall Satisfaction	Satisfied	Neutral
User Feedback	User feedback is unnecessary, it should be used in companies where the users pay for the service they offer	Important, because they can do statistics and the banks pay to get a good service
Improvement	Automation of ticket solving for simple tasks such as out-of-servers	Automating ticketing classification and prioritization, better graphics
Favorite Feature	Sending Emails through the ticket with the current ticket information, if the customer/user doesn't answer per phone calls	Knowledge base to search for similar incidents

Conclusion and Future Work

This thesis provides an explainable and interactive helpdesk TS. The main objective is to engage users in the decision-making processes of the system and support each other through community-driven support. An interactive system enables users to interact directly with the functions of the system and personalize their experience by having full control over their actions. Whereas an explainable system ensures that transparency is there through explanations of how decisions have been made. These characteristics therefore make users powerful stakeholders by giving them increased authority and comprehension. The proposed system can be reached by focusing on recent research in the field of text classification, XAI, and LLMs. Specifically, we compare various classification models (traditional models, DL models, and LLMs) and define new agents for the TS. The first agent is the Discord community with an implemented TS for user-to-user support. The second one is an open-source chatbot for automatic ticket resolution. The helpdesk is the third agent that provides various functionalities for an interactive and explainable user experience, such as automated ticket classification, XAI explanation of the classification model, and LLMs for ticket title and tag generation. The user studies outcomes of the proposed solution showed that simplicity is the key for users. However, AI can have a big impact on ticketing routing and resolving by using traditional models such as SVM with TF-IDF with an accuracy of 0.83 for ticket classification, advanced LLM for ticket transformation, and chatbots for automated ticket answering.

However, the thesis has some limitations, mainly from the dataset. It makes it hard to build a knowledge base or chatbot from the data because the company does not provide any ticket solutions (privacy reasons). Then further simplification of the ticket routing by classifying and assigning the ticket to a specific support agent is not possible because there is no ID available for the corresponding ticket solver. Then there are some limitations with the AI tools. This thesis always uses the free version of the AI tools such as for the chatbot or even the Discord ticketing-bot, having a limited use of functions.

To this end, future work will focus on advanced AI technology such as implementing a conversational chatbot that supports the users in real-time per screen/camera by showing exact steps to solve the ticket. Google is working in this direction through the implementation of a chatbot agent named Astra [3]. The user can show anything with the camera and have a real-time conversion with the chatbot. It uses object recognition and answers real-time questions. Additionally, the user can navigate the chatbot with their hands or draw directly on the screen what exactly the chatbot should focus on. The mentioned idea is for a long-term project but we can already start by implementing a chatbot on the Discord Community, having a fully functional TS in Discord with automated ticket resolving. Furthermore, user tests need to be done with a broader audience and with different organizations to create a better user-oriented design.

[3] `https://etedge-insights.com/technology/artificial-intelligence/googles-project-astra-a-revolution-in-ai-assistants/`

Bibliography

[1] Ali, Abuhmed Sajid, El-Sappagh Tamer, Muhammad Shaker, Alonso-Moral Khan, Confalonieri Jose M, Guidotti Roberto, Del Ser Riccardo, Diaz-Rodriguez Javier, Herrera Natalia, and Francisco. Explainable artificial intelligence (xai): What we know and what is left to attain trustworthy artificial intelligence. *Information fusion*, 99:101805, 2023.

[2] Sajid Ali, Tamer Abuhmed, Shaker El-Sappagh, Khan Muhammad, Jose M Alonso-Moral, Roberto Confalonieri, Riccardo Guidotti, Javier Del Ser, Natalia Díaz-Rodríguez, and Francisco Herrera. Explainable artificial intelligence (xai): What we know and what is left to attain trustworthy artificial intelligence. *Information fusion*, 2023.

[3] Nicola Arici, Luca Putelli, Alfonso Emilio Gerevini, Luca Sigalini, Ivan Serina, et al. Llm-based approaches for automatic ticket assignment: A real-world italian application. In *NL4AI@ AI* IA*, 2023.

[4] Deepak Suresh Asudani, Naresh Kumar Nagwani, and Pradeep Singh. Impact of word embedding models on text analytics in deep learning environment: a review. *Artificial intelligence review*, 2023.

[5] Anas Shamsi Basha. Chat marketing influence on customer support satisfaction in the financial startups.

[6] Renato Bruni, Gianpiero Bianchi, and Pasquale Papa. Hyperparameter black-box optimization to improve the automatic classification of support tickets. *Algorithms*, 2023.

[7] Youngjin Chae and Thomas Davidson. Large language models for text classification: From zero-shot learning to fine-tuning. *Open Science Foundation*, 2023.

[8] Yupeng Chang, Xu Wang, Jindong Wang, Yuan Wu, Linyi Yang, Kaijie Zhu, Hao Chen, Xiaoyuan Yi, Cunxiang Wang, Yidong Wang, et al. A survey on evaluation of large language models. *ACM Transactions on Intelligent Systems and Technology*, 2024.

[9] Philipp Engelbrechtsmüller. Fine-tuning large language models for ticket classification at doka gmbh/submitted by philipp engelbrechtsmüller. 2024.

[10] Sally Espinosa. 10 discord stats to know if you're on the fence about discord influencer marketing. `https://www.theshelf.com/the-blog/discord-stats/#:~:text=Discord%20boasts%20about%20154%20million,Discord%20are%20not%20there%20passively.`, 2024.

[11] John Fields, Kevin Chovanec, and Praveen Madiraju. A survey of text classification with transformers: How wide? how large? how long? how accurate? how expensive? how safe? *IEEE Access*, 2024.

[12] Simon Fuchs, Clemens Drieschner, and Holger Wittges. Improving support ticket systems using machine learning: A literature review. 2022.

[13] Yanchu Guan, Dong Wang, Zhixuan Chu, Shiyu Wang, Feiyue Ni, Ruihua Song, Longfei Li, Jinjie Gu, and Chenyi Zhuang. Intelligent virtual assistants with llm-based process automation. *arXiv preprint arXiv:2312.06677*, 2023.

[14] AKM Bahalul Haque, AKM Najmul Islam, and Patrick Mikalef. Explainable artificial intelligence (xai) from a user perspective: A synthesis of prior literature and problematizing avenues for future research. *Technological Forecasting and Social Change*, 2023.

[15] Su Cheng Haw, Kyle Ong, Lit Jie Chew, Kok Why Ng, Palanichamy Naveen, and Elham Abdulwahab Anaam. Improving the prediction resolution time for customer support ticket system. *Journal of System and Management Sciences*, 2022.

[16] Eva Heinrich, Heather Thomas, and Ella R Kahu. An exploration of course and cohort communication spaces in discord, teams, and moodle. *Australasian Journal of Educational Technology*, 2022.

[17] Robert R Hoffman, Shane T Mueller, Gary Klein, and Jordan Litman. Measures for explainable ai: Explanation goodness, user satisfaction, mental models, curiosity, trust, and human-ai performance. *Frontiers in Computer Science*, 2023.

[18] Liss Jenneboer, Carolina Herrando, and Efthymios Constantinides. The impact of chatbots on customer loyalty: A systematic literature review. *Journal of theoretical and applied electronic commerce research*, 2022.

[19] VAMSI KATRAGADDA. Automating customer support: A study on the efficacy of machine learning-driven chatbots and virtual assistants. 2023.

[20] Sharon Lauricella, Chris Craig, and Robin Kay. Examining the benefits and challenges of using discord in online higher education classrooms. *Journal of Educational Informatics*, 2023.

[21] Yuanchun Li, Hao Wen, Weijun Wang, Xiangyu Li, Yizhen Yuan, Guohong Liu, Jiacheng Liu, Wenxing Xu, Xiang Wang, Yi Sun, et al. Personal llm agents: Insights and survey about the capability, efficiency and security. *arXiv preprint arXiv:2401.05459*, 2024.

[22] Q. Vera Liao, Milena PribiÄĞ, Jaesik Han, Sarah Miller, and Daby Sow. Question-driven design process for explainable ai user experiences, 2021.

[23] Alexander Lill, André N Meyer, and Thomas Fritz. On the helpfulness of answering developer questions on discord with similar conversations and posts from the past. 2024.

[24] Ashokkumar Palanivinayagam, Claude Ziad El-Bayeh, and Robertas Damaševičius. Twenty years of machine-learning-based text classification: A systematic review. *Algorithms*, 2023.

[25] SP Paramesh and KS Shreedhara. A deep learning based it service desk ticket classifier using cnn. *ICTACT Journal on Soft Computing*, 2022.

[26] Soya Park and Chinmay Kulkarni. Thinking assistants: Llm-based conversational assistants that help users think by asking rather than answering. *arXiv preprint arXiv:2312.06024*, 2023.

[27] Rajvardhan Patil, Sorio Boit, Venkat Gudivada, and Jagadeesh Nandigam. A survey of text representation and embedding techniques in nlp. *IEEE Access*, 2023.

[28] Aleksandra Revina, Krisztian Buza, and Vera G. Meister. It ticket classification: The simpler, the better. *IEEE Access*, 2020.

[29] Waddah Saeed and Christian Omlin. Explainable ai (xai): A systematic meta-survey of current challenges and future opportunities. *Knowledge-Based Systems*, 2023.

[30] M VENKATA Subbarao, KASUKURTY Venkatarao, and C Suresh. Automation of incident response and it ticket management by ml and nlp mechanisms. *Journal of Theoretical and Applied Information Technology*, 2022.

[31] Kate Sukhanova. The latest discord statistics trends for 2023. `https://techreport.com/statistics/software-web/discord-statistics/#:~:text=Discord%20Number%20of%20Users%20Statistics&text=The%20Discord%20user%20count%20grew,big%20contributors%20to%20that%20growth.`, 2023.

[32] Jiao Sun, Q Vera Liao, Michael Muller, Mayank Agarwal, Stephanie Houde, Kartik Talamadupula, and Justin D Weisz. Investigating explainability of generative ai for code through scenario-based design. 2022.

[33] Mohammad Mustafa Taye. Understanding of machine learning with deep learning: architectures, workflow, applications and future directions. *Computers*, 2023.

[34] Mario Truss and Stephan Boehm. Ai-based classification of customer support tickets: State of the art and implementation with automl. *arXiv preprint arXiv:2406.01789*, 2024.

[35] Mario Truss and Stephan Boehm. Ai-based classification of customer support tickets: State of the art and implementation with automl. *arXiv preprint arXiv:2406.01789*, 2024.

[36] Alvian Shanardi Wijaya and Tanty Oktavia. Machine learning approaches for helpdesk ticketing system: A systematic literature. *Journal of Theoretical and Applied Information Technology*, (5), 2024.

[37] Xuansheng Wu, Haiyan Zhao, Yaochen Zhu, Yucheng Shi, Fan Yang, Tianming Liu, Xiaoming Zhai, Wenlin Yao, Jundong Li, Mengnan Du, et al. Usable xai: 10 strategies towards exploiting explainability in the llm era. *arXiv preprint arXiv:2403.08946*, 2024.

[38] Alessandro Zangari, Matteo Marcuzzo, Michele Schiavinato, Andrea Gasparetto, and Andrea Albarelli. Ticket automation: An insight into current research with applications to multi-level classification scenarios. *Expert Systems with Applications*, 2023.

Appendices

Appendix A: Key-Findings about the support agent interviews

Table 5.11: Complaints of the Finance Dataset

Features	Non-Null Count	Data Type	Example
index	78313	object	complaint-public-v2
type	78313	object	complaint
id	78313	object	2562526
score	78313	float 64	0.0
tags	10900	object	Servicemember
zip_code	71556	object	347XX
complaint_id	78313	object	2562526
issue	78313	object	Managing an account
date_received	78313	object	2017-06-29T12:00:00-05:00
state	76322	object	FL
consumer_disputed	78313	object	No
product	78313	object	Checking or savings account
company_repsonse	78313	object	Closed with explanation
company	78313	object	JPMORGAN CHASE CO.
submitted_via	78313	object	Web
date_sent_to_company	78313	object	2017-06-29T12:00:00-05:00
company_public_response	4	object	The company chooses not to provide a public response
sub_product	67742	object	Checking account
timely	78313	object	Yes
complaints	78313	object	Hello, my name is XXXX XXXX. I just opened my XXXX account yesterdayXX/XX/XXXX. My account was verified through jp Morgan Chase and now my XXXX XXXX account States that my bank is not participating. I just request to cash out $40.00 onXX/XX/XXXX. Now I wish to know where will my money go and how can I rectify this situation asap. Thank you
sub_issue	32016	object	Deposits and withdrawals
consummer_consent_provided	77305	object	Consent provided

Appendix B: Wrongly Predicted Categories for Credit and Prepaid Card

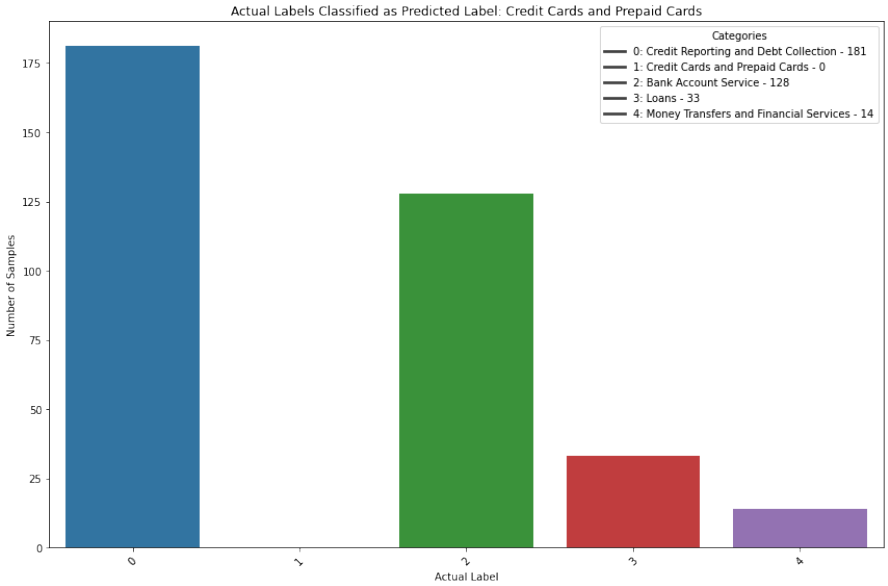

Figure 5.16: Actual Labels Classified as Predicted Label: Credit and Prepaid Cards

Appendix C: Wrongly Predicted Categories for Bank Account Service

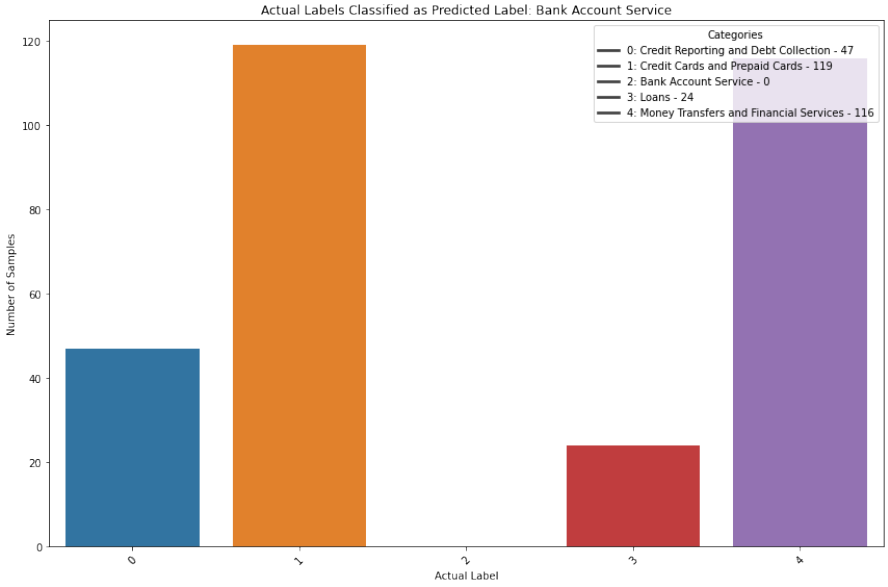

Figure 5.17: Actual Labels Classified as Predicted Label: Bank Account Service

ACM Classification

- **Human-centered computing ~ User-centered design**
  Concept ID: 10003120.10003123.10010860.10010859
  Significance: 500
- **Computing methodologies ~ Natural language processing**
  Concept ID: 10010147.10010178.10010179
  Significance: 500
- **Computing methodologies ~ Classification and regression trees**
  Concept ID: 10010147.10010257.10010293.10003660
  Significance: 500
- **Human-centered computing ~ User studies**
  Concept ID: 10003120.10003121.10003122.10003334
  Significance: 500
- **Computing methodologies ~ Multi-agent systems**
  Concept ID: 10010147.10010178.10010219.10010220
  Significance: 500