A Music Virtual Assistant Based on Machine Learning

by

Shadan Shameli Derakhshan

A Thesis Submitted in Partial Fulfillment of the
Requirements for the Degree of

MASTER OF SCIENCE

in the Department of Computer Science

A Music Virtual Assistant Based on Machine Learning

by

Shadan Shameli Derakhshan

Supervisory Committee

Dr. George Tzanetakis, Supervisor
(Department of Computer Science)

Dr. Alex Thomo, Departmental Member
(Department of Computer Science)

Dr. Kirk McNally, Outside Member
(School of Music)

Supervisory Committee

Dr. George Tzanetakis, Supervisor
(Department of Computer Science)

Dr. Alex Thomo, Departmental Member
(Department of Computer Science)

Dr. Kirk McNally, Outside Member
(School of Music)

ABSTRACT

This study focuses on the development of a music chatbot designed to address a gap in the digital music services market. The chatbot provides users with a seamless and interactive experience, enabling them to engage in music-related interactions. The enhancement that differentiates this chatbot from other standard UIs is while standard UIs of popular music streaming platforms like Spotify, Tidal, and Apple Music offer access to a vast library of songs and user-friendly interfaces, they may lack personalized and engaging interactions. That is, unlike traditional UIs where users navigate through menus and search boxes, the chatbot engages users in a conversation, allowing them to interact naturally using text which can later be improved to voice input. This creates a more human-like and enjoyable experience.

The Music4all [38] dataset is chosen in this research to train, develop, and evaluate the chatbot. This dataset contains data from 15,602 anonymous users, their listening histories, and 109,269 songs represented by their audio clips, lyrics, and 16 other metadata/attributes. Various techniques, including pattern-matching approaches, TF-IDF combined with machine learning algorithms, Word2vec embeddings, and the BERT model, were explored to determine the most effective methods for creating engaging and responsive music chatbots.

To achieve this, the study initially involved data preparation and the creation of a JSON file containing patterns as features and artists as classes. Numerical values representing 60 classes and TF-IDF sparse matrices of 8027 songs were then fed into various machine learning algorithms, including decision trees, random forests, KNN, and SVM. This is followed by a comprehensive comparison of the metrics obtained from these algorithms. The experimental results indicate that the combination of TF-IDF and SVM yielded the best results for designing the chatbot, achieving a classification accuracy of 91%. However, advanced methods such as the BERT model and Word2vec were found to be less useful due to over-fitting issues and since it is a classification problem for labeled data.

Finally, the classification of artists was integrated into a Flask app, which provided the song name and ID based on user-requested tags. This study contributes valuable insights into the development of a music chatbot and highlights the most effective methods for classification and response generation using the given dataset.

Contents

List of Tables

List of Figures

ACKNOWLEDGEMENTS

I would like to express my sincere gratitude and appreciation to the following individuals and organizations who have played a crucial role in the completion of this thesis.

First and foremost, I would like to thank my supervisor, ***Dr. George Tzanetakis****, for his invaluable guidance, unwavering support, and insightful feedback throughout the entire research process. His expertise and dedication have been instrumental in shaping this work.*

I extend my gratitude to ***Ergonomyx*** *for their financial support, which has made this research possible. Their investment in my academic pursuits has enabled me to delve deeper into the subject matter. I am genuinely thankful for their belief in my abilities and their commitment to advancing the field.*

I also want to express my deep appreciation to my ***family*** *for their emotional support and encouragement in every step of my educational journey.*

Shadan Shameli

Chapter 1

Introduction

1.1 Motivation

The primary motivation behind this research is to develop a sophisticated and interactive music chatbot that offers a seamless and personalized music experience for users. Music possesses a unique ability to evoke emotions and activate various regions in the brain's reward centers[47]. The goal of this study is to leverage the power of artificial intelligence and machine learning to create a chatbot that goes beyond traditional music streaming services. Instead of merely providing access to an extensive music library, our chatbot aims to understand and cater to individual user preferences. By analyzing user interactions, the chatbot will adapt its recommendations, creating a more personalized and engaging musical journey for each user. Through natural language processing the chatbot will interpret user queries, allowing users to express their emotions and musical preferences in a conversational manner. Additionally, the integration of the Music4All dataset will ensure access to a diverse range of songs. Ultimately, the chatbot aims to serve as a digital music companion, offering valuable insights into the world of music while providing comfort, entertainment, and inspiration to users. By understanding the reasons people listen to music whether for relaxation, motivation, emotional expression, or cultural connection - the chatbot will deliver a more fulfilling and enjoyable music exploration.

Chatbots or dialogue systems have gained significant attention in recent years. These software programs are designed to engage in natural conversations with humans, utilizing techniques such as natural language processing and machine learning to understand and respond to user input, be it written text or speech [22]. ELIZA is

considered the first chatbot and was developed in 1966. It aimed to emulate a therapist [46]. After that pioneering work, much research has been undertaken to improve the performance of chatbots. Nowadays, many dialogue systems are implemented in various areas such as education, health, finance, etc. The latest chatbot system that was released in 2023 and had a groundbreaking impact on the industry is ChatGPT. The research described in this thesis was undertaken prior to the release of ChatGPT.

However, despite the integration of chatbots in various industries, there has been limited research undertaken to develop chatbots specifically tailored for music datasets. Surprisingly, the digital music services commonly used, such as Spotify, SoundCloud, YouTube Music, and Amazon Music, lack chatbot or music recommendation functionalities with a dialogue box. This significant gap in the music industry and the absence of a dedicated music chatbot have motivated us to embark on this project. The baseline model that is used uses pattern matching to find a song. The main approach for this chatbot is using a machine learning algorithm to find a song related to user input. The chatbot matches the input to the tags of the songs and finally offers a song with the same tag.

1.2 Context of the Study

The limited research in music-specific chatbots was the main motivation behind this research. In this thesis, I explore making a chatbot by using different techniques. At first, the baseline model is implemented by using pattern-matching techniques and then various machine-learning techniques are explored, The context of the study can be seen as follows:

- **Development of a music chatbot**: The aim of the study is to create a chatbot specifically designed for music-related interactions, filling the gap in the digital music services market. This chatbot will offer users a seamless and interactive experience, allowing them to discover, explore, and engage with music in a conversational manner.

- **Exploration of different techniques**: To develop the chatbot, various techniques are explored, including pattern-matching approaches and a combination of TF-IDF and machine learning algorithms. The comparison and evaluation of these techniques with alternatives such as wave2vec embeddings and the BERT model are investigated. The key objective of the thesis is to identify the most

effective methods for creating a music chatbot. This exploration contributes to advancing state-of-the-art music chatbot technology.

- **Evaluation and analysis**: The developed chatbot using the Music4All dataset is thoroughly evaluated and analyzed.This evaluation involves comparing the performance of different machine-learning algorithms and assessing the chatbot's effectiveness and efficiency in understanding and responding to music-related queries. The chatbot uses the tags of the songs as input classifies them to their artists and finally gives us a song from the artist. In other words, the chatbot gives us a song by considering what they have seen in the input. that has The analysis of these results provides insights into the strengths and weaknesses of different approaches, aiding in the refinement and improvement of music chatbot systems.

The study is significant since it is the first time that the dataset is used for making a chatbot and it aims to conduct valuable experiments using the music4all dataset. The developed chatbot can be easily converted to a speech recognition virtual assistant for future projects when coupled with a speech recognition and text-to-speech synthesis system. Also, using chatbots has many benefits such as reducing the search time in applications. It can contribute valuable insights for future works to develop more complex chatbots for music datasets.

1.3 Dataset

To create this chatbot, the music4all dataset is used. The dataset is fully explained in chapter 4-1. The dataset was requested from the corresponding online website and we fulfilled the disclosure and confidentiality agreement. I signed a Disclosure and Confidentiality Agreement that the aim to use the music4all dataset is limited to research purposes, and any commercial use or activity involving this database requires written permission from the owners and is explicitly prohibited. Also in the same Disclosure, I confirm that appropriate reference will be made to the database and its paper [37].

1.4 Structure

This thesis is structured as follows:

Chapter 1: This chapter provides an introduction to the research topic, highlighting the potency of music and the significance of chatbots. The motivation behind this project, emphasizing the need for a music chatbot in the industry is also discussed.

Chapter 2: A comprehensive literature review is presented, with a focus on existing chatbots and their motivations within the context of music virtual assistants. The first chatbot to the current state-of-the-art dialog systems is analyzed and discussed to understand how they operate and which ones among them are relevant for creating the music chatbot using our dataset.

Chapter 3: This chapter covers the techniques used for developing chatbots, including natural language processing, machine learning, and rule-based approaches. It delves into the theoretical foundations required to understand the subsequent chapters

Chapter 4: In this chapter, the methodology that is employed to develop the dialogue system is outlined. First, the baseline model is introduced which is implemented using the pattern-matching method without utilizing any machine learning methods. Additionally, an alternative method is presented, leveraging machine learning (ML) and natural language processing (NLP) techniques to develop the chatbot.

Chapter 5: In the final chapter, contains the results obtained and compares the different machine-learning algorithms employed in the chatbot. It provides insights into the strengths and weaknesses of each algorithm, highlighting their performance in the context of the developed music chatbot

Chapter 6: This chapter concludes which model is better than others and can be used in this dialog system. The strengths and weaknesses of each algorithm that is used are discussed and suggest what can be done for future works.

Chapter 2

Literature review

This chapter provides a literature review of different topics that have informed the work described in this thesis. At first, we talk about the importance of music and what motivated us to develop the chatbot. Then, we trace the development and history of chatbots, starting from Eliza, the computer program considered to be the first chatbot, and ending with ChatGPT, which was recently introduced and has shown remarkable abilities compared to previous chatbots. We should note that the research and development of the music chatbot described in this thesis took place before ChatGPT became widely available. In the final section of this chapter, we describe existing work in music chatbots.

2.1 Music Information Retrieval

Music is an essential aspect of our daily lives, with individuals dedicating significant amounts of time listening to it and spending billions of dollars purchasing it. Despite its prevalence, mainstream social-personality psychology has not devoted much attention to this ubiquitous social phenomenon. Studies conducted in this field demonstrate that music can have significant impacts on cognition, emotion, and behavior. The research also suggests that individuals use music to fulfill diverse purposes, ranging from managing their emotions to expressing themselves to fostering social connections. The findings in this developing area reveal how social-personality psychology can enhance our comprehension of music, thereby emphasizing the practical importance of established theories and research [36].

Over the past two decades, music information retrieval (MIR), a branch of mul-

timedia information retrieval, has experienced rapid growth as a research field. The majority of MIR research to date has centered on extracting music-related information primarily from the audio using signal processing techniques [11].

The advances made in content-based music retrieval have generated interest in the research field of intelligent interfaces for music retrieval [11]. In 2019, Jin and others conducted a study to test two types of critiquing settings, UC and HC, used in a conversational agent for suggesting music. The study was online, and based on several UX metrics, the recommendations produced by both UC and HC were generally viewed equally by users. [19]

In 2020, Knees and others undertook an investigation that considered the advancement of intelligent user interfaces for music exploration and discovery-driven by MIR in the last twenty years [24]. They claimed that three key advancements have impacted and molded user interfaces during this period, each associated with a stage of novel listening practices. During Phase 1, there was the emergence of content-based music retrieval interfaces that relied on audio processing and content description algorithms to automate the organization of music databases and enable the identification of music based on sound qualities. These interfaces are mainly linked to individual music collections or small-scale commercial catalogs. The development of query by humming systems is a significant milestone in this phase[21]. In Phase 2, user interfaces integrated collaborative and automatic semantic descriptions of music, utilizing the information obtained from user-generated metadata. These interfaces are associated with collaborative web platforms. In Phase 3, the focus shifted towards recommender systems that rely on the large-scale collection of online music interaction data. These interfaces are connected to streaming services [24]. The emergence of chatbots in various application domains has made them a desirable platform to construct recommender systems [19].

2.2 Development of chatbots

The idea of an intelligent machine engaging in human interactions was first theorized by Alan Turing in 1950 with the famous Turing test thought experiment [43]. In 1966, the first chatbot called ELIZA was created by Joseph Weizenbaum as a psychotherapist chatbot. ELIZA operated on a program using MAC time-sharing at MIT and utilized basic decomposition rules that were activated by specific keywords in the input text to analyze sentences. ELIZA is available online [46].

Another famous chatbot, named PARRY, was created by Kenneth Colby in 1972. PARRY employs a similar response pattern organization as ELIZA but had a more sophisticated control system. PARRY is utilized for disease simulation. This Chatbot system also has language-understanding capabilities. PARRY has effective variables like anger, fear, and mistrust. PARRY was put through the wringer in the early 1970s by a group of 33 human psychologists using a variation of the "Turing Test", and it succeeded in fooling its human examiners 52% of the time [50].

Richard Wallace created another chatbot called ALICE (Artificial Linguistic Internet Computer Entity) in 1995. The chatbot was inspired by ELIZA. ALICE is an open-source natural language handling Chatbot program that uses some pattern-matching rules. ALICE is situated in an XML knowledge base. It coordinates the client's contribution against a predefined set of reactions. As it has a predefined set of QA reactions, it can't answer every one of the inquiries satisfactorily but it is feasible for ALICE bots to grow their insight bases through XML. Utilizing this, an ALICE bot can be intended to be a specialist in any space-specific data [5]. there was a theory circulating in applied AI called "Case-Based Reasoning" or CBR that maps well onto the ALICE algorithm. Another term, borrowed from pattern recognition, is "nearest-neighbor classification [45].

Enterprise chatbots are widely utilized in information technology (IT) service management, which is a crucial application area. Within organizations and companies, the IT service desk is an essential department responsible for ensuring work continuity and resolving technical problems faced by employees and clients. Due to the dynamic nature of this department, manual intervention and supervision are necessary, which can affect process execution speed and quality. In response to competitive pressure, IT service providers are seeking to enhance service quality and reduce operating costs through automation. Therefore, they are adopting chatbots to accelerate work processes and ensure their quality [42].

In 2001, the development of SmarterChild marked an important advancement in chatbot technology. SmarterChild was available through messenger applications and was able to assist people with simple tasks using information retrieved from databases [29]. In 2003, SIR International coordinated the development of CALO (Cognitive Assistant that Learns and Organizes), which was funded by the Defense Advanced Research Projects Agency (DARPA). It took 5 years to develop a cognitive assistant through the CALO project that could learn from experience and complete repetitive tasks for users. This project was a significant milestone in chatbot history, as it incor-

porated various fields of artificial intelligence and improved software systems' ability to comprehend human intentions. The CALO project also had several offshoots, with Apple's Siri being the most notable [31].

Apple Siri, created in 2010, was the first virtual personal assistant. This was soon followed by similar virtual assistants such as IBM Watson [1], Microsoft Cortana [3], Amazon Alexa [2], and Google Assistant[2]. Integrated into smartphones and smart speakers, these virtual assistants were able to comprehend human speech, deliver vocal responses, and execute more sophisticated tasks. With their internet connections, virtual personal assistants offer swift response times [16]. However, due to their limitations in comprehending colloquial language and considering the context of conversations, misunderstandings are not uncommon [6]. Most existing chatbots currently retrieve information from structured documents to build their knowledge base, since structured documents have labeled utterance-response (or Q-R) pairs the chatbot engine can store [49].

In recent times, the healthcare industry has witnessed significant advancements with the integration of information technology and AI. The authors of [12] have introduced a mobile healthcare application that utilizes a chatbot to provide prompt treatment in the event of accidents that occur during everyday life. Additionally, this application addresses sudden health changes that may endanger patientsĺives. The use of chatbots in businesses is becoming increasingly popular in the form of customer service agents, which assist in addressing customer issues and streamlining the sales process. With the emergence of digital marketing and artificial intelligence, companies are expanding their reach globally and moving towards online platforms to improve customer experiences during purchases and provide novel technical support for post-sales issues. Furthermore, fashion brands including Burberry, Louis Vuitton, Tommy Hilfiger, Leviś, HM, and eBay are contributing to the growing popularity of e-service agents [13].

In 2019, openAI demonstrated that language models can begin learning task-specific datasets without explicit supervision when trained on a new dataset called WebText, comprising millions of webpages. By conditioning the language model on a document and associated questions, it generates accurate answers. The performance matches or exceeds three out of four baseline systems, despite not using the 127,000+ training examples. The largest model, introduced was GPT-2, with 1.5 billion parameters. When a large-scale language model is trained on a diverse and extensive dataset, it exhibits strong performance across various domains and datasets. GPT-

2 achieves state-of-the-art results on seven out of eight tested language modeling datasets without any fine-tuning, demonstrating its ability to generalize to different tasks[34].

In 2020, OpenAI demonstrated that increasing the size of language models significantly enhances their ability to perform well across various tasks with minimal fine-tuning. To be specific, they trained a language model called GPT-3, which consists of an autoregressive architecture with a massive 175 billion parameters. This parameter count is ten times larger than any previous non-sparse language model. They evaluated the performance of GPT-3 in a few-shot scenario, where the model is tested with limited examples, to assess its generalization capabilities. GPT-3 demonstrates its versatility across various tasks without the need for gradient updates or fine-tuning. Tasks and demonstrations are conveyed to the model solely through text interactions. The performance of GPT-3 is robust on numerous NLP datasets, encompassing translation, question-answering, cloze tasks, and tasks involving on-the-fly reasoning or domain adaptation, such as word unscrambling, using novel words in sentences, or performing 3-digit arithmetic. However, there are specific datasets where GPT-3 struggles with few-shot learning, as well as certain methodological challenges stemming from its training on extensive web corpora. Interestingly, GPT-3 has the ability to generate news article samples that perplex human evaluators who find it difficult to differentiate them from articles authored by humans [10].

In 2023, OpenAI introduced GPT-4, an advanced model capable of processing both image and text inputs and generating text outputs. Although it falls short of human-level competence in various real-world scenarios, GPT-4 demonstrates performance comparable to humans on professional and academic benchmarks. GPT-4 utilizes a Transformer-based architecture and undergoes pre-training to predict the subsequent token in a document. Through a post-training alignment process, its performance improves in terms of factual accuracy and adherence to the desired behavior. A key focus of this project involved developing infrastructure and optimization techniques that exhibit consistent behavior across a wide range of scales. This enabled us to make accurate predictions about certain aspects of GPT-4's performance based on models trained with significantly less computational resources, as little as 1/1,000th the compute power of GPT-4 [25].

Since the early 2000s, MIR research has suggested numerous options to make music discovery more accessible. A path that has already been pursued is the distribution of music through smart speakers such as Amazon Echo, Google Home, or Apple

HomePod, which can be operated through personal assistants like Alexa, Google Assistant, or Siri, respectively, by using voice commands.

2.3 Different types of chatbots

We have different categories for chatbots. Each category has been defined based on a simple criterion and a chatbot can belong to more than one category at a time. The knowledge domain is another aspect that can be used to characterize chatbots. Chatbots that can answer any user question from whichever domain are called Generic chatbots. Chorus(Chorus—A Crowd-Powered Conversational Agent on Google Hangouts, 2020) is an example of a generic chatbot. On the other hand, Domain-Specific chatbots can respond only to questions concerning a particular knowledge domain [26].

Another aspect that can be used to classify chatbots is what is their goal. We have chat-based/Conversational and Task-based chatbots. Informative chatbots are when users communicate with a chatbot to get specific information stored in a dataset while Chat-based/Conversational chatbots hold a natural conversation with the user like a human.[6]

The Response Generation method separates chatbots into Rule-based, Retrieval based, and Generative based chatbots. Depending on the Permissions provided by the development platforms, chatbots can be divided into Open-source or Commercial. In the end, another classification depends on the Communication channel that chatbots use, which can be text, voice, image, or all of them. The latest chatbots can now react to pictures, and aside from recognizing objects in the images, they can also comment on them and express their emotion [41].

2.4 Natural language processing

Natural Language Processing (NLP)[23] is an Artificial Intelligence field that examines how computer systems can interpret and control the natural language regarding text or speech. Information is collected on the comprehension and usage of human language to create the appropriate techniques for computer systems to manage human language and carry out a variety of tasks [20]. Most NLP techniques rely on machine learning. They consist of Natural Language Understanding, which evolves the mis-

sion to understand a text, and Natural Language Generation[27] which introduces the responsibility to generate the text commonly conducted by ANNs.

NLP technology enables people to interact with computers using natural language. To understand natural language, the process can be divided into two parts: syntactic and semantic analysis. Syntactic analysis involves organizing words in a sentence in a grammatically correct way and arranging them into structures that demonstrate their relationships. In contrast, the semantic analysis focuses on the meaning of words and sentences. By analyzing the logical structure of sentences, semantic analysis can determine the true meaning and identify similarities between words to comprehend the topic being discussed in the sentences [39].

2.5 Existing Music chatbots

Although there is not a lot of activity in this area, there are some existing music-related chatbots. Most of them rely on other social media such as Facebook, Discord, and Telegram. In this section, we are talking about a few different chatbots that stand alone. Shivanand and their colleagues developed a chatbot in 2020 that recommends music and movies based on a user's mood. Their application aims to determine the user's mood and then play songs or display a list of movies on a website accordingly. The application is designed to be run on a desktop computer and accurately identify the user's mood. Human-computer interaction (HCI) is essential in modern times, and emotion recognition from facial images is a popular concept in HCI. Their system utilizes the Haar Cascade Algorithm to detect the user's face in a live webcam feed and the CNN Algorithm to identify the emotions expressed by the user through facial features, such as the arrangement of the mouth and eyes. This process relies on frontal views of facial images to determine the user's mood [40].

In 2021, Nair and others introduced an emotion-based recommendation system that allows users to listen to music that matches their mood. The systems utilize CNN technology to analyze audio signals and collaborative filtering to suggest songs based on the user's listening history [30]. In 2021, Rishit Jain and colleagues explored the development of a music chatbot using the GTZAN dataset[44]. Their method involved genre classification, utilizing CNN to distinguish between audio files based on their timbral features' visual representations. They achieved an accuracy rate of 68.9% with their proposed model, which they believed to be superior to other classification models they had previously investigated [17].

Chapter 3

Chatbot Background

3.1 Chatbot Techniques

In order to develop a chatbot, it is important to be familiar with different techniques such as parsing, pattern matching, AIML, Chat Script, and Language tricks. In this section, we are going to discuss different techniques that can be used for designing a powerful chatbot.

3.2 Parsing

This technique gives us the ability to analyze the input and manipulate it by using NLP functions. In text parsing, the original text will be converted to a set of words with features to determine the grammatical structure of the word. This process is called lexical parsing. Additionally, the lexical structure can then be evaluated to ensure it forms a valid expression.

3.3 Pattern matching

The typical approach to creating a chatbot involves using rule-based methods, which involve using pattern matching algorithms to compare the user's input to a set of predetermined rules. Based on this comparison, the chatbot selects a predefined response from its available options. The context of the conversation can also influence the selection of the appropriate rule and determine the format of the response [28]. ELIZA and its successor ALICE were the first chatbots to use pattern matching. At

the same time, PARRY, PC Therapist III, Chatterbot in "TinyMUD", TIPS, FRED, CONVERSE, HEX, Albert, and Jabberwocky used this technique [6]. Typically, rule-based systems do not generate new responses because the knowledge they use is pre-written by the developer as conversational patterns [35]. The larger the database of rules, the more capable a rule-based chatbot becomes in answering user questions. However, it takes thousands of rules to make this type of chatbot function correctly, making it challenging to handle grammatical and syntactic errors in the user's inputs. Most rule-based chatbots for single-turn communication only consider the latest response when selecting an answer. Despite the large number of rules required for this type of chatbot to work correctly, it can still struggle with handling grammatical and syntactic errors in the user's inputs. Examples of rule-based Chatbots include Cleverbot, Chatfuel, and Watson [35]. This method has been used to develop our baseline model. Also, the combination of this method with TF-IDF and machine learning is used to make classification algorithms which are discussed in chapters four and five.

Most rule-based chatbots for one-time conversations only consider the latest response when selecting an answer. However, human-like chatbots use a multi-turn approach, where every response is used to guide the selection of an answer that is both normal and fitting for the entire conversation context[48]. The disadvantage of using pattern matching is that the responses are automated, repetitive, and lack the creativity and naturalness of human responses [35]. However, the advantage is that it provides a quick response, as it does not undergo a detailed analysis of the input text's syntax or meaning[18].

3.4 AIML

Artificial Intelligence Mark-up Language is one of the core techniques that have been used in common chatbots. AIML is a derivative of XML. It represents the knowledge put into the chatbot based on the software technology developed for A.L.I.C.E. AIML describes a class of data objects called AIML objects. Each object in AIML contains two units called topics and categories, that contain either parsed or unparsed data. The category is the basic unit of knowledge in AIML. Each category consists of an input question, and an output answer The question, or stimulus, is called the pattern. The answer, or response, is called the template. The two types of optional context are called "that" and "topic."

<category >
<pattern >MOTHER <pattern >
<template >Tell me more about your family. </template>
</category>

3.5 Chat script

ChatScript can be useful when there are no accurate matches in AIML. It focuses on providing the best syntax for constructing a meaningful default response. It offers various features such as variables, concepts, facts, and logical and/or operations. ChatScript is a successor to the AIML language, offering improved syntax and ease of maintenance. It addresses issues with zero word matching and introduces additional capabilities such as concepts, continuations, logical and/or operations, variables, fact triples, and functions. With these features, it attempts to fulfill the need for ontologies within the script itself. An example of a script defining a concept of meat with one pattern can be seen in Figure 3.1[9].

```
concept: ~meat ( bacon ham beef meat flesh veal lamb
chicken pork steak cow pig )
s: ( I love ~meat ) Do you really? I am a vegan.
```

Figure 3.1: An example of a script defining a concept of meat with one pattern

3.6 SQL and relational database

This method ensures consistency and precision in the conversation, as it allows the dialogue system to access and utilize past information. Relational databases are structured collections of data organized in tables. Each row represents a record or entry, while each column represents a specific attribute or piece of information.

When it comes to creating a chatbot, utilizing a relational database with SQL can be beneficial in several ways. To implement a chatbot using SQL and a relational database, creating a database schema to define the structure of the tables and the relationships between them is the method. Then, SQL queries to insert, update, retrieve, and delete data from the tables can be used. The chatbot application would

interact with the database by executing SQL queries and processing the results to generate appropriate responses. Overall, leveraging SQL and a relational database in chatbot development facilitates the storage, retrieval, and utilization of past conversational data. This helps in maintaining consistency and precision by enabling the chatbot to access historical information and provide more contextually relevant responses to users.

3.7 Markov Chain Models

Markov Chain Models are based on the concept that every instance of a word or letter in a given text data set appears with a specific probability. The model's "order" refers to the number of consecutive occurrences that are taken into consideration. For instance, if we consider the input text "agggcagcgggcg," a Markov model with an order of 0 would predict that the letter "a" has a probability of occurring 2/13 times. On the other hand, a model with an order of 1 would indicate that the occurrence probability of each letter remains fixed, but it is dependent on the preceding letter.

3.8 Machine learning Approach

Chatbots using Machine Learning techniques, instead of Pattern Matching, extract meaning from user input through Natural Language Processing (NLP) and have the capacity to learn from interactions. These chatbots take into account the entire conversational context, not just the current turn, and do not require predetermined responses for every potential user input. However, they typically require a large training dataset, which may be a major challenge as existing datasets may not be sufficient. For instance, movie script collections may be too general, while IT support data may be too specific. The machine learning algorithm is the best approach to make the dialog box for our dataset since we can define the problem in the classification algorithm and the dataset does not contain a complex corpus and because the dataset does not contain a complex corpus.

3.9 Different Layers of Chatbots

Creating a Chatbot software program involves identifying its different components. The Chatbot can be categorized into three primary parts, namely Responder, Classifier, and Graphmaster (as illustrated in Figure 3.2). Figure 3.2 is a figure based on [4] paper. Chatbot components can be described as follows.

- **Responder**: The responder acts as a bridge between the user and the classifier, facilitating the transfer of information and controlling both input and output. In simpler terms, it serves as the interface between the user and the chatbot. In the music dialog system that we made the responder is implemented by the Flask platform. The reason behind choosing Flask is that it is cross-platform and it is easy to use for the user

- **Classifier**: This layer can filter and normalize the input, segment the input into logical components, transfer the normalized sentences to the Graphmaster, process the output from the graph master, and handle the instructions of the database syntax (e.g. AIML). To implement the classifier we have built the classifier module that utilizes the TF-IDF and classification algorithm. Train the classifier on a labeled dataset to learn patterns and classify user input into appropriate categories or intents. This module should take the preprocessed input from the Responder, apply the TF-IDF technique, and use the trained classification algorithm to predict the intent of the user's input

- **Graphmaster**: pattern matching is held in this section of chatbots. Graphmaster organizes the brain's contents storage and holding the pattern matching algorithms[4].

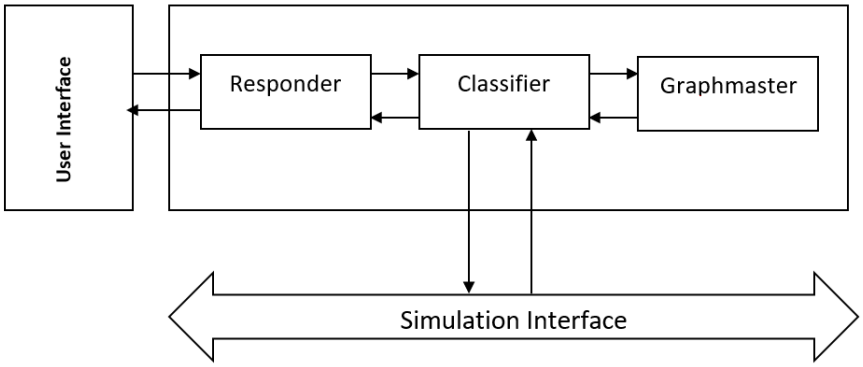


Figure 3.2: Chatbot component

Chapter 4

Methodology

In this chapter, the explanation of the Music4all dataset and the reason that this dataset is chosen for this experiment is discussed this dataset. Then, the baseline model will be introduced. Finally, the discussion of the machine-learning method and the stages that are used to develop the machine-learning chatbot.

4.1 Music4all Dataset

The music information retrieval community aims to develop new techniques and devise advanced systems that can competently and proficiently recommend and retrieve songs from vast collections of music content. There are many music datasets available to use for this project. After investigating different datasets, I decided to use music4all [37] as our main dataset. Table 4.1 indicates the comparison of different available music datasets. As it can be seen in the table 4.1, what makes music4all stand out is that it contains many available metadata and 109,269 instances along with authorize requests which makes this dataset one of the most comprehensive datasets of music. Music4all was created as a valuable database that could be utilized across various research fields. In order to accomplish this, it was necessary to collect a wide range of information pertaining to the songs. The development of Music4all was divided into two stages, namely the user phase and the music phase. During the user phase, information about the listening habits of users was collected, while in the song phase, data about the songs themselves were gathered. Figure 4.1 shows different stages of music4all development. Filtering steps were taken to filter out songs with missing information to ensure that only accurate data was retained in the database. The

first phase, which was focused on the users, began by selecting a random user from last.fm. Using the last.fm API, the friends of this user were obtained and continued to recursively gather the friends of those friends, excluding duplicates, ultimately collecting a total of 40,940 users. For each of these users, their listening history spanning from January 1st, 2019 00:00 to March 20th, 2019 23:59 was collected. This time range was chosen to obtain data on popular and well-known songs from recent times. The final phase of gathering information was eliminating outliers. A threshold was used to filter out users who had very few songs in their listening histories. Only users with at least 1,000 plays were kept in the database, while others were discarded. This resulted in a total of 15,602 users and 320,073 songs by the end of the user phase. During the song phase, various types of data were collected for each song, including tags, lyrics, genres, audio, and metadata. This information was sourced from multiple locations and integrated into the database. Songs with incomplete data were excluded from the database. Tags for each song were obtained through a last.fm crawler that accessed each song's page. A crawler was chosen because it can provide more tags than the API. Out of a total of 320,073 songs, only 306,010 had at least one tag. Genre information for each song was obtained by using the genre names available at Every Noise At Once, an application that is designed to classify musical genres based on the distinctions made by Spotify. Songs that had no genre names were filtered out from the 306,010 songs that had associated tags, resulting in 266,837 songs with genres. As a final step, songs that were discarded at some point in the song phase were filtered out from the users' listening histories. Hence, the final version of the database comprises 15,602 anonymous users, their listening histories, and 109,269 songs represented by their audio clips, lyrics, and 16 other metadata/attributes. Table 4.2 shows the metadata of music4all dataset [37].

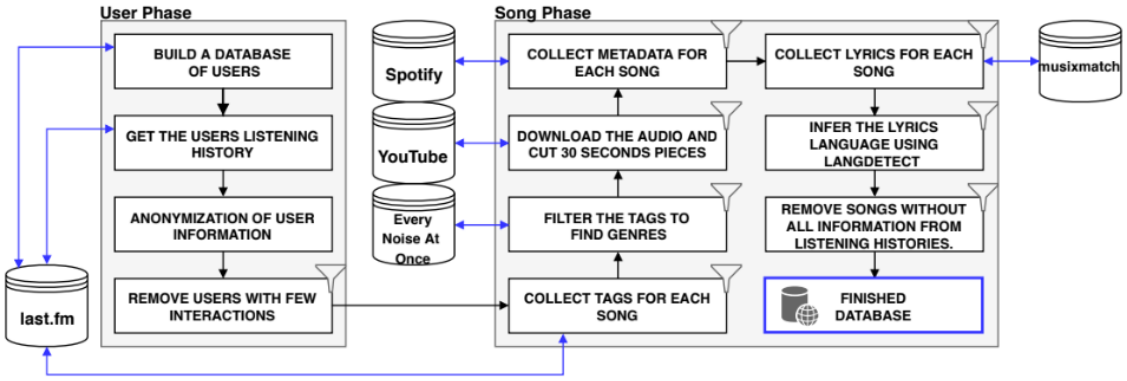


Figure 4.1: Music4all development stages

Database	Number of Instances	Number of Classes	Available Content	Authoriz Requested	Songs Lenght
RWC	100	10	Songs	Yes	Full
GTZAN	1000	10	Songs	No	30 sec
USPOP 2002	8752	10	Songs	Yes	Full
ISMIR 2004	1458	6	Songs	No	Full
Homburg	1886	9	Songs	No	10 sec
Ballroom	698	8	Songs	No	30 sec
CAL500	500	174	Songs	No	Ful
Holzapfel	N/A	6	Songs	Yes	Full
1517-artists	3180	19	Songs	Yes	Full
LMD	3227	10	Features	No	Full
Magnatagatune	21642	188	Songs	No	30 sec
MSD	1000000	Variable	Features	No	Full
Music4All	109269	Variable	Many	Yes	30 sec

Table 4.1: Comparison of different available music datasets

The music4all dataset is a novel music database that includes vital information such as metadata, tags, genre details, audio snippets, lyrics, and more. These data can assist in the development of innovative digital music systems. Figure 4.2 indicates the distribution of genre and tags per song.

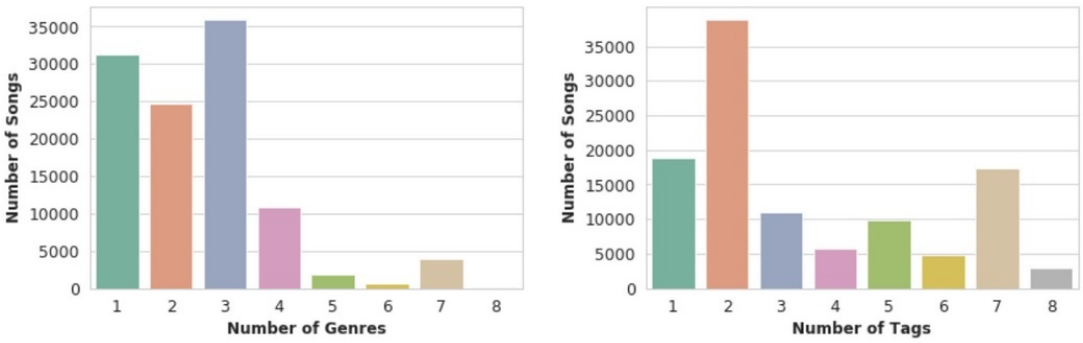


Figure 4.2: Distribution of genre and tags per song

In order to improve the organization of the database, the related features are grouped into distinct files. These files have a column for the song's ID and other columns for its corresponding attributes. Table 4.3 outlines the names of these files and the specific attributes they contain.

Attribute	Description
id	Unique 16 characters identifier for the song in the database.
Artist	Name of the artist that published the song in last.fm. There are 16,269 unique artists in the database.
song	Name of the song.
lang	Language assigned to the lyrics by the tool lang detect. There are 46 unique languages in the database.
spotify id	Song identifier in the Spotify application.
popularity	Integer values ranging from 0 to 100 represent how popular a song is on Spotify. This value is based on the total number of plays the song has and how recent those plays are.
album name	Name of the album that the song is in. There are 38,363 different albums in the database.
release	release Year in which the song was released.
danceability	Real value ranging from 0.0 to 1.0 representing how suitable the song is for dancing. This value is based on a combination of musical elements, provided by the Spotify API.
energy	Real value ranging from 0.0 to 1.0 provided by the Spotify API that is a perceptual measure of intensity and activity.
Key	Overall key of the song, using standard Pitch Class notation, provided by the Spotify API.
mode	Binary value provided by the Spotify API corresponding to the modality of the song, where major is represented by 1 and minor by 0.
Valence	Overall key of the song, using standard Pitch Class notation, provided by the Spotify API.
Tempo	Speed or pace of the song, measured in beats per minute(BPM), provided by the Spotify API.
genres	List of genres tags associated with the song. There are 853 unique genre tags in the database.
tags	User-given tags from the last.fm application, with 19,541 unique tags.

Table 4.2: the metadata or attributes of the Music4All database

Filename	Description
id genres.csv	File containing the id of the song and its list of genres, separated by a comma.
id information.csv	File containing the id of the song along with basic information about it, namely artist, song name, and album name.
id lang.csv	File containing the id of the song and its lyrics language.
id metadata.csv	File containing the id of the song, its attributes obtained through the Spotify official API, including its Spotify id.
id tags.csv	File containing the id of the song and the tags associated with it, separated by a comma.
listening history.csv	File containing the listening history for each user in the database.

Table 4.3: Meta data provided for the music4all dataset

4.2 Exploratory Data Analysis

Prior to developing the chatbot I conducted some analysis to better understand the information it contains. To perform a statistical analysis EDA on the Music4All dataset, I started by exploring the distribution of various numerical features and their correlations with each other. Figure 4.3 checks the distribution of three numerical features (popularity, duration, and tempo) using histograms, and checks the correlation between these features using a heatmap.

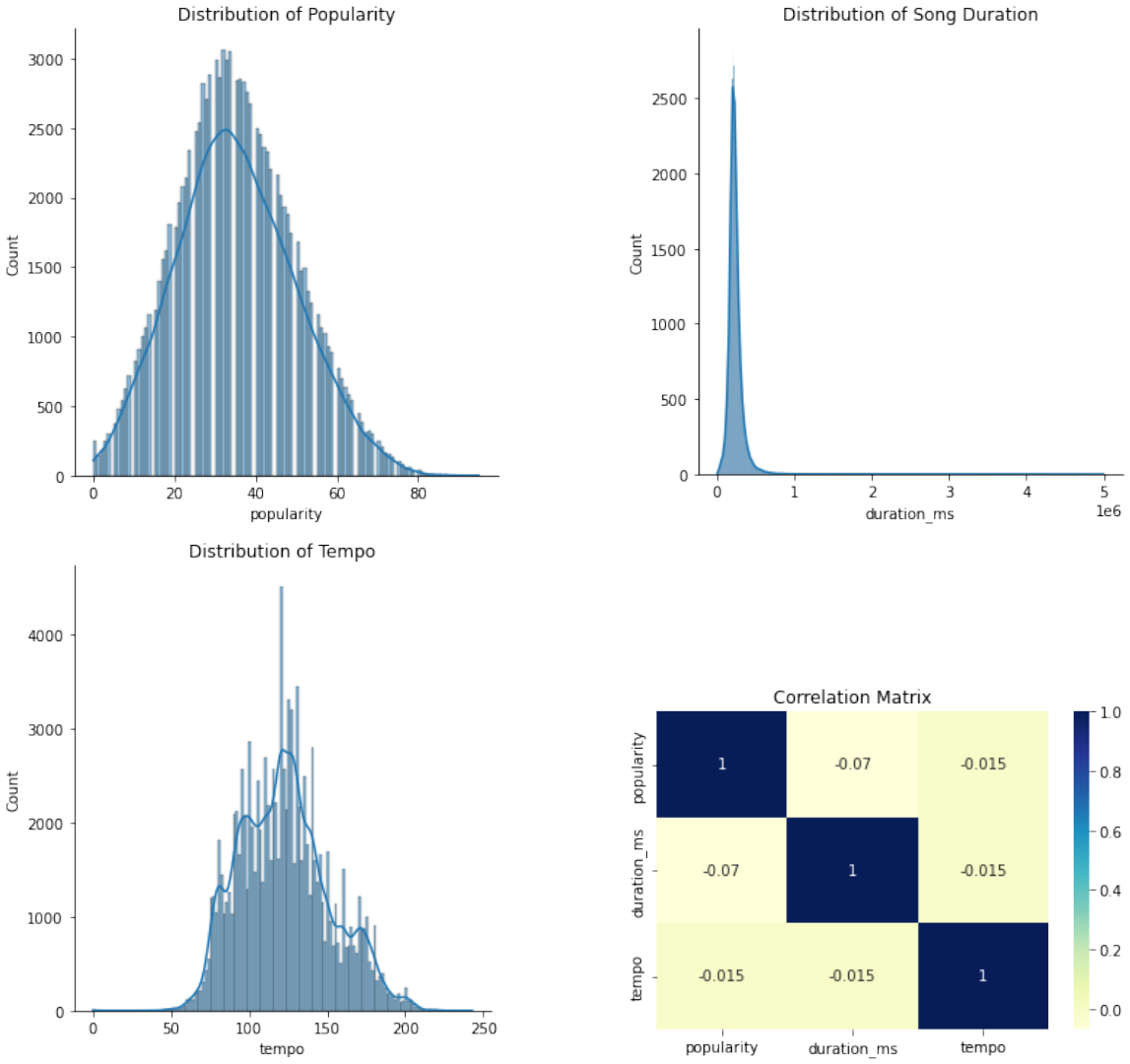


Figure 4.3: The distribution of numerical values in database

For the next step, I also performed hypothesis testing to investigate if there are any significant differences between certain groups or variables in the dataset.

Table 4.4 shows the results of a pairwise t-test comparison of the tempo for the top 10 genres in the Music4All dataset. Each row of the table corresponds to a comparison between two genres. The columns show the names of the two genres being compared, the p-value for the t-test, and the interpretation of the results.

The p-value is a measure of the probability of observing a difference as large as the one observed in the sample data, assuming that there is no true difference between the populations from which the samples were drawn. If the p-value is less than 0.05 (i.e., less than the conventional threshold for statistical significance), then we say that

Table 4.4: The results of a pairwise t-test comparison of the tempo for the top 10 genres

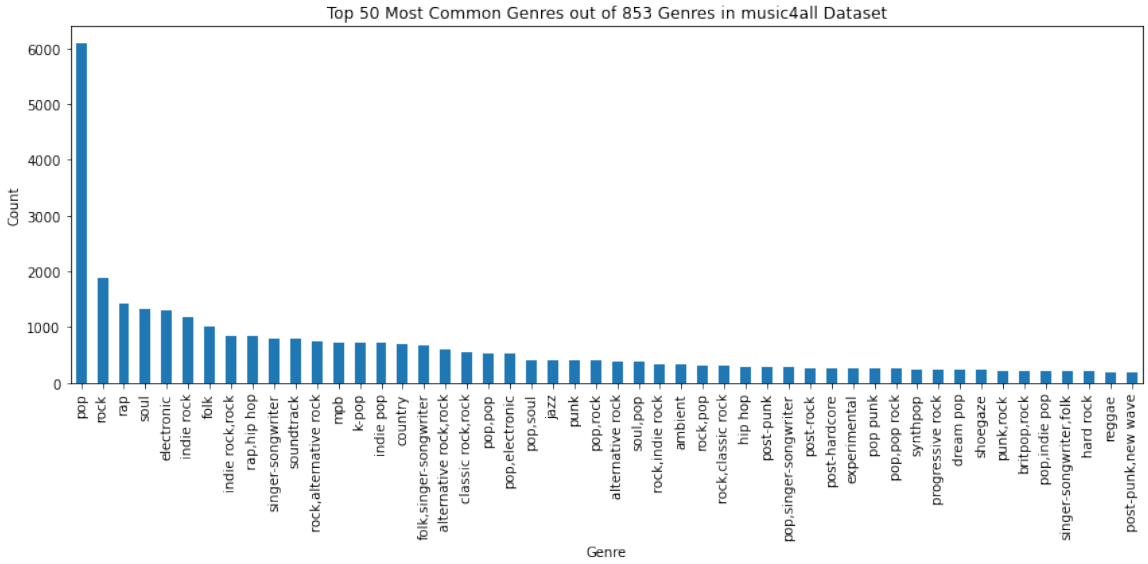
Genre 1	**Genre 2**	**t-value**	**p-value**	**Is there significant difference**
pop	rock	-5.75	0.0	Yes
pop	rap	-1.94	0.052993	No
pop	soul	7.37	0.0	Yes
pop	electronic	-2.49	0.012823	Yes
pop	indie rock	-8.17	0.0	Yes
pop	folk	1.68	0.092398	No
pop	indie rock-rock	-9.61	0.0	Yes
pop	rap-hip hop	1.58	0.115258	No
pop	singer-songwriter	0.72	0.468817	No
rock	rap	2.52	0.011812	Yes
rock	soul	10.27	0.0	Yes
rock	electronic	2.35	0.018842	Yes
rock	indie rock	-2.68	0.007345	Yes
rock	folk	5.3	0.0	Yes
rock	indie rock-rock	-4.78	2e-06	Yes
rock	rap-hip hop	4.95	1e-06	Yes
rock	singer-songwriter	4.1	4.3e-05	Yes
rap	soul	7.17	0.0	Yes
rap	electronic	-0.29	0.773852	No
rap	indie rock	-4.79	2e-06	Yes
rap	folk	2.74	0.006116	Yes
rap	indie rock-rock	-6.55	0.0	Yes
rap	rap-hip hop	2.61	0.009087	Yes
rap	singer-songwriter	1.88	0.060194	No
soul	electronic	-7.8	0.0	Yes
soul	indie rock	-11.92	0.0	Yes
soul	folk	-4.04	5.6e-05	Yes
soul	indie rock-rock	-12.93	0.0	Yes
soul	rap-hip hop	-3.73	0.000194	Yes
soul	singer-songwriter	-4.29	1.9e-05	Yes
electronic	indie rock	-4.73	2e-06	Yes
electronic	folk	3.13	0.001775	Yes
electronic	indie rock-rock	-6.55	0.0	Yes
electronic	rap-hip hop	2.96	0.00314	Yes
electronic	singer-songwriter	2.19	0.02873	Yes
indie rock	folk	7.25	0.0	Yes
indie rock	indie rock-rock	-2.27	0.02361	Yes
indie rock	rap-hip hop	6.83	0.0	Yes
indie rock	singer-songwriter	5.99	0.0	Yes
folk	indie rock-rock	-8.72	0.0	Yes
folk	rap-hip hop	0.04	0.965657	No
folk	singer-songwriter	-0.59	0.554891	No
indie rock-rock	rap-hip hop	8.29	0.0	Yes
indie rock-rock	singer-songwriter	7.5	0.0	Yes
rap-hip hop	singer-songwriter	-0.6	0.548036	No

there is a significant difference between the two genres in terms of their tempo.

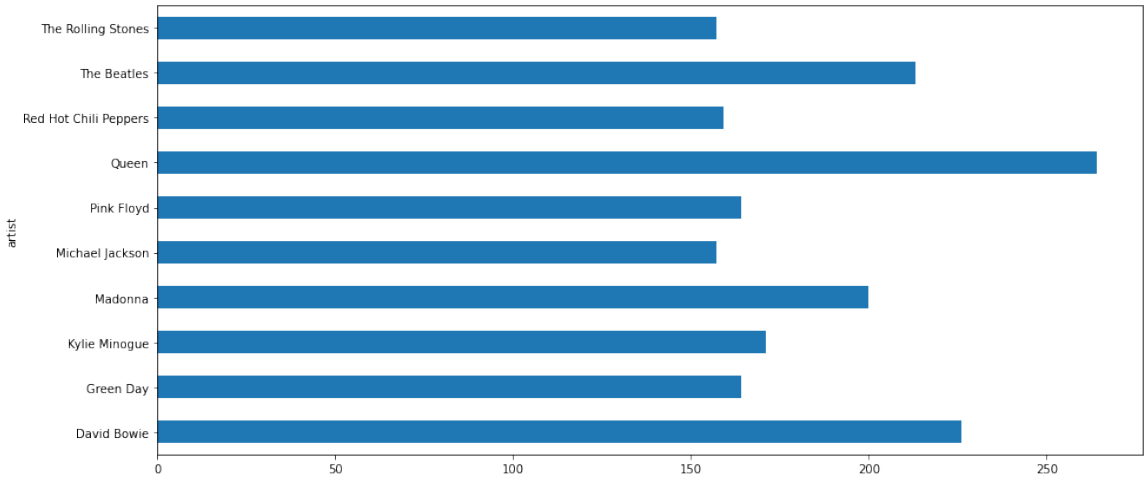
In this table, we can see that there are several pairs of genres that show a significant difference in tempo, such as "pop vs. rock", "pop vs. soul", "rock vs. rap-hip hop". On the other hand, some pairs of genres do not show a significant difference in tempo, such as "electronic vs. rap", "pop vs. indie rock ", and "rap-hip hop vs. singer-songwriter". Overall, this table provides a useful summary of the pairwise tempo comparisons between the top genres in the Music4All dataset.

4.3 Data Visualization

The next step for our project is visualizing data. Visualization allows us to identify patterns and trends in the data that might not be immediately apparent in tabular or textual form. Figure 4.4 shows the visualization of the most common genre and artists in the dataset.



(a) The occurrence of the genre in the dataset



(b) The most common artists with the number of occurrences

Figure 4.4: The most common genre and artists in the dataset.

In order to know the most common Genres in different eras, the interactive pie chart in figure 4.5 is made. As can be seen in figure 4.6 by the passing time the number of music genres that people are listening to increased. So if I ask our chatbot about music from the 50s, I don't expect any other genre but Folk, Rock, and Pop. I used popularity and genre columns to make this visualization.

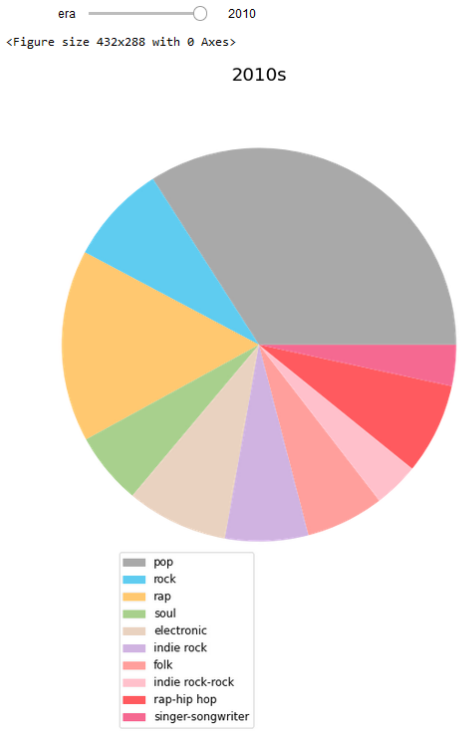


Figure 4.5: The most common genres in different eras

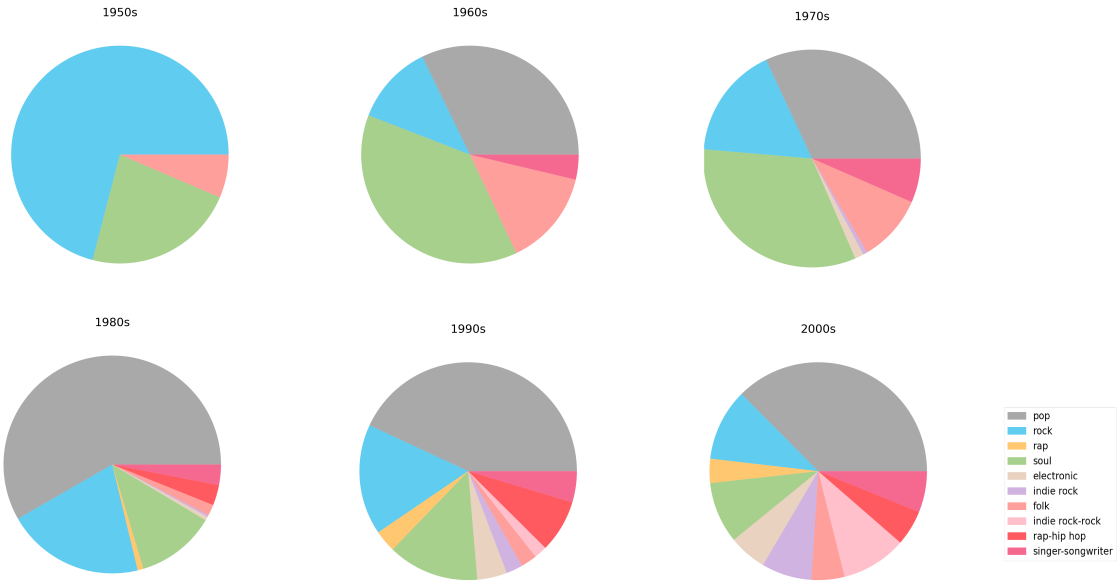


Figure 4.6: Pie charts of top 10 common genres from 1950 to 2000

4.4 Data Preparation

The first step of designing our chatbot is data preparation. I have used metadata of the music4all dataset to develop these chatbots. The metadata that I used is four csv files called id_lang.csv, id_metadata.csv, id_information.csv, and id_tags.csv. more information about the files can be seen in the table 4.3 in the previous section.

In order to use the metadata for our chatbot it is better to have them all in one file. At first, I made a Pandas library data frame and merged all the information in the CSV files in it. Then in order to do pattern matching, a JSON file is built with the names of the artists as a response. The patterns that are used are tags of songs. The main idea is that users will frequently use tags in their queries to the chat-bot and that the songs corresponding to a particular artist are similar in terms of tags used. Therefore, I can cast the problem of query/response as a problem of classification of a set of tags to a particular artist. By using a machine learning approach the exact syntax of the query does not need to be fixed like in rule-based approaches as long as the query contains relevant tags. For example the query "Play me a slow jazz song with female vocals" or "I would like to hear a female jazz singer" or "I like slow jazz" could also end up returning to an artist like Billie Holiday without explicitly having to specify patterns for all possible syntactic variations of a query. By simplifying the problem this way, I can use traditional machine-learning approaches by treating the tags as features and the artists as classes. I make the simplifying assumption that any of the songs corresponding to an artist can be returned as a response to a query. As can be observed in the table 4.2 I have 19,541 unique tags. if the chatbot sees any of these tags in the user input, it should find the output which is the requested song by the user. The tag can be the name of the song, the album's name whether the artist is male or female, genres, or any other tags. the output of the chatbot in the JSON file is the id of the songs. In figure 4.7 I can see the structure of the JSON file for one artist. As mentioned in section 4.1, the dataset has no missing value, and the outliers are removed, so I do not need the step of taking care of missing values and outliers.

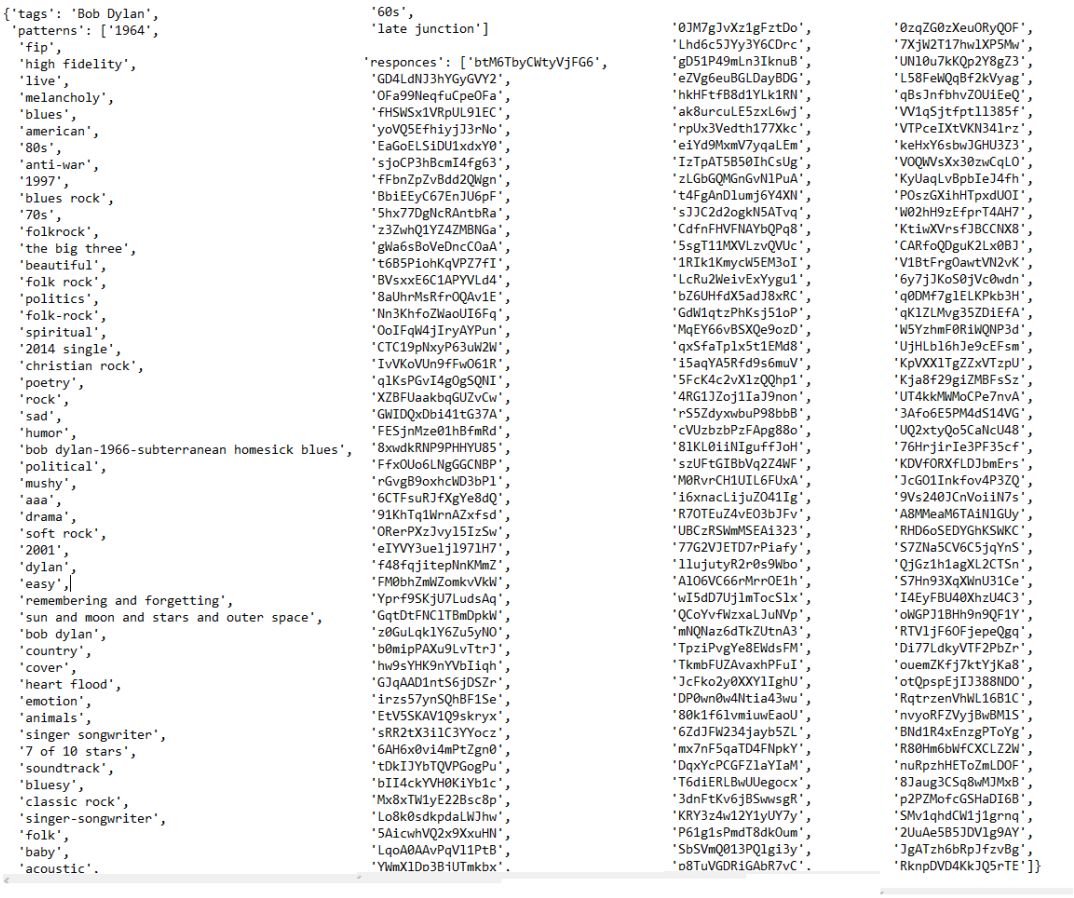


Figure 4.7: Structure of the JSON file for one artist

4.5 Baseline model

This section presents a baseline pattern-matching model for a chatbot system. The objective of the model is to enable the chatbot to identify and respond to specific patterns. A pattern-matching model in a chatbot refers to the utilization of predefined patterns and corresponding responses to handle user queries. When the chatbot receives a user input, it compares the input against these predefined patterns and selects the corresponding response from the dataset. Pattern-matching chatbots are typically less complex than other types of chatbots, such as those based on machine learning or natural language processing. However, they can still be useful for handling simple tasks. One advantage of pattern-matching chatbots is that they are typically easier and faster to develop than other types of chatbots. However, their responses are limited to the pre-defined patterns and rules that have been programmed into

them, which can make them less flexible and adaptable than other types of chatbots. The chatbot that is developed in this section is able to find any tags from the dataset. It is unable to understand complex sentences and it gets input and searches in the tags to find common tags between the query and the songs in the dataset. It then suggests a song based on the input.

4.6 Machine learning Approach

While rule-based chatbots can be effective for certain use cases, they are limited in their ability to handle complex or large inputs. To create a more versatile chatbot, it is better to consider using a machine learning approach instead. Overall, Machine-learning chatbots are generally considered to be better than rule-based chatbots because they can handle a wider range of inputs and provide more natural and personalized responses. Here are some reasons why machine-learning chatbots are better than rule-based chatbots:

- Ability to handle complex inputs: Machine learning chatbots can learn from a large dataset of examples and can handle a wide range of inputs, even if they are not explicitly programmed into the system. This allows them to provide more accurate and helpful responses, even when dealing with complex or unpredictable inputs.

- Personalized responses: Machine learning chatbots can learn from user interactions over time, allowing them to provide more personalized responses based on the user's preferences and history. This can improve the user experience and make the chatbot feel more like a natural conversation

- Scalability: Machine learning chatbots can be trained on large datasets and can handle a high volume of user interactions without requiring significant manual programming. This makes them more scalable than rule-based chatbots, which can become unwieldy and difficult to maintain as the number of rules increases.

- Adaptability: Machine learning chatbots can adapt to new inputs and changes in user behavior over time, making them more flexible and adaptable than rule-based chatbots.

Overall, while rule-based chatbots can be effective for certain use cases, machine-learning chatbots offer greater flexibility, scalability, and accuracy, making them a

better choice for many applications. The architecture of the designed machine learning dialog box can be seen in the 4.8.

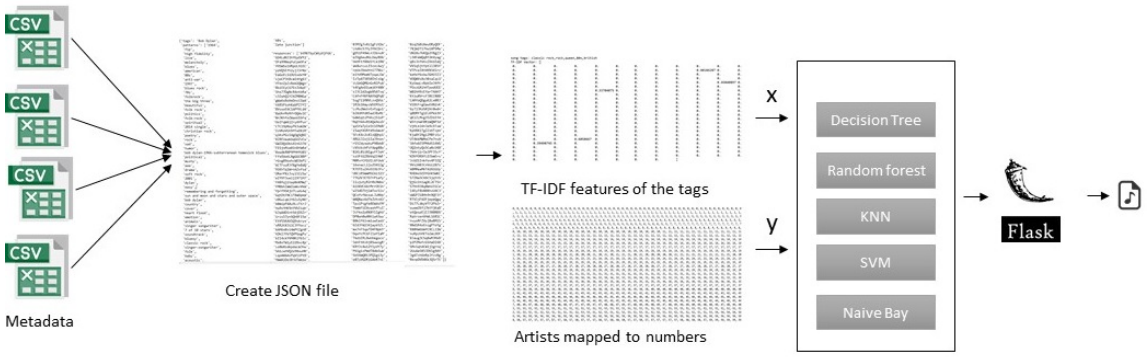


Figure 4.8: Architecture of chatbot.

4.6.1 Data Pre-processing

The first step of using the machine learning method is that I will do some data pre-processing. In this stage, in addition to what I have done in section 4.2, I have to do some extra steps. Our goal with using the JSON file is to generate a set of likely user messages and associate them with suitable responses. Each dictionary in the file is labeled with a tag indicating the category of the corresponding message. I will use this data to train a classifier to classify a sentence into one of the tags in our file. Once the classification is made, I can retrieve a response that is the id of the song from the associated group and present it to the user. The effectiveness and complexity of the chatbot will increase with the number of tags, responses, and patterns provided to it. The next step is to extract the information from our JSON file. To address that, I require all of the patterns and their corresponding class/tag information. I also need a list of all of the unique words in our patterns.

4.6.2 Natural Language Processing

In this step, I utilized the NLTK package first I do tokenization on the list of patterns and classes. Text is the input in natural language processing. The text needs to be cleaned and processed before using the data for analysis through some natural language processing techniques. The first step of our natural language processing is tokenization. Tokenization is a process of dividing a sequence of text into smaller

units, called tokens. In the context of natural language processing (NLP), tokenization is often the first step in preparing text data for analysis. Tokenization can be performed at different levels of granularity, depending on the specific application and the desired level of analysis. For example, tokenization can be done at the level of words, sentences, or even sub-words. The word_tokenize function has been used to extract tokens from the pattern. I need to use the tokenized words as input for the chatbots. I tokenized both the patterns that I have and also the classes. The number of unique classes is 19,541 which is the number of unique tags. In table 4.5, the lists generated from the JSON file to develop the chatbots can be seen. The table contains a column called "size of files for stratified k fold cross validation". That is, in order to have balanced data to use in the stratified cross-validation, artists with less than 100 instances are eliminated, and the size of our files as can be seen reduced in this column. Table 4.6 shows the classes that are fed to the models with the number of their songs. The next step is stemming or lemmatization. Stemming is a technique used to transform inflected words to their base or root form. For instance, if the words "playing", "plays", and "played" is taken, stemming would result in a common word "play". This process can help to simplify the vocabulary in our model and uncover the underlying meaning of sentences. Another similar technique is lemmatization, which differs from stemming in that it produces valid words that can be found in a dictionary, whereas stemming can generate nonexistent words. Therefore, the root form produced by stemming cannot be directly searched in a dictionary, while a lemma can be. In other words, lemmatization is a related technique to stemming, which also aims to reduce words to their base or root form. However, unlike stemming, lemmatization considers the context and part of speech of a word in order to determine its lemma, which is an actual word found in a dictionary. This makes lemmatization a more accurate technique than stemming, as it produces valid words that can be understood by humans. In this chatbot, the lemmatization technique is used.

Name	Description	Size	size of files for stratified k fold cross validation
Words	The words that we have in the patterns	204,700	5304
Classes	The classes that we have in the dataset	19,541	60
doc_x	List of tokenized words	137,640	3246
doc_y	List of tokenized classes	137,640	3246

Table 4.5: Generated lists as the machine learning inputs

Artist	Number of songs	Artist	Number of songs
Queen	264	Eminem	127
David Bowie	226	Rush	123
The Beatles	213	Muse	122
Madonna	200	Radiohead	121
Kylie Minogue	171	Taylor Swift	121
Green Day	164	Dream Theater	120
Pink Floyd	164	Prince	120
Red Hot Chili Peppers	159	Björk	119
The Rolling Stones	157	Johnny Cash	118
Michael Jackson	157	In Flames	118
Bob Dylan	156	Tori Amos	116
Iron Maiden	155	Sonic Youth	115
Metallica	153	Coldplay	114
Britney Spears	149	Chico Buarque	114
Depeche Mode	148	Sufjan Stevens	112
The Cure	147	R.E.M.	112
U2	145	Moby	112
Pet Shop Boys	143	Black Sabbath	110
Tom Waits	142	Korn	110
Glee Cast	135	Blur	110
Cher	134	Motörhead	107
Megadeth	134	Autechre	105
Céline Dion	133	Shakira	105
Nine Inch Nails	132	Linkin Park	105
Mariah Carey	132	ABBA	104
Marilyn Manson	131	Katatonia	103
Bad Religion	130	Foo Fighters	102
Elton John	129	Gal Costa	101
Pearl Jam	129	Frank Zappa	101
Christina Aguilera	128	The Kinks	100

Table 4.6: Chatbot classes with the number of each artist's songs

4.6.3 Bag of words

After loading our data and generating a vocabulary consisting of stemmed words, the concept of a "bag of words" can be discussed. Machine learning algorithms need numerical inputs. Therefore, to represent sentences using numerical values, a "bag of words" method as a first method is employed. Our approach involves representing each sentence as a list with a length equivalent to the number of words present in our model's vocabulary. Each index in the list corresponds to a specific word in the vocabulary. In addition to formatting our input, the output must also be formatted in a manner that the machine learning model can comprehend. To achieve this, the output lists are generated that are as long as the number of labels/tags present in our dataset, following a similar approach to that of the "bag of words" method. Because of the size of the dataset, this method was not successful and ran out of memory. I

tried to use part of the dataset that is the 100 most frequent artists but still using the bag of words was not an appropriate solution. That is why I used another method called TF-IDF.

4.6.4 Using Count Vectorizer

In order to feed the text into the machine learning model a collection of text documents need to be converted to a matrix of token counts. The count vectorizer is used to make vectors from our patterns and tags and then tf-idf transformers is used and then the data is fed to the machine learning models. In other words, first, the raw count matrix using a count vectorizer is computed and then fit TF-IDF transformers to transform them into tf-idf weighted vectors.

4.6.5 Term frequency-inverse document frequency

A problem with the Bag of Words approach is that highly frequent words start to dominate in the document (e.g. larger score), but may not contain as much "informational content". Also, it will give more weight to longer documents than shorter documents. This approach is not the right approach when the dataset is large. The same method is used for a smaller JSON file to make a chatbot and it was working but after using music for all datasets we had a memory issue and we decided to move on to the second method which is called TF-IDF. The term weighting approach known as "Term frequency–inverse document frequency" (tf–idf) is widely employed in modern information retrieval systems [7]. The concept behind TF-IDF is derived from the language modeling theory, which suggests that terms within a document can be classified into two groups: words that have importance and those that do not [51]. The TF-IDF approach is to rescale the frequency of words by how often they appear in all documents [15] so that the scores for frequent words like "the" that are also frequent across all documents are penalized. This approach to scoring is called Term Frequency-Inverse Document Frequency, or TF-IDF for short, where: Term Frequency: is a scoring of the frequency of the word in the current document. There are several ways of calculating this frequency, with the simplest being a raw count of instances a word appears in a document. Then, there are ways to adjust the frequency, by the length of a document, or by the raw frequency of the most frequent word in a document. The inverse document frequency of the word across a set of documents. This means, how common or rare a word is in the entire document

set. The closer it is to 0, the more common a word is. This metric can be calculated by taking the total number of documents, dividing it by the number of documents that contain a word, and calculating the logarithm. So, if the word is very common and appears in many documents, this number will approach 0. Otherwise, it will approach 1. Multiplying these two numbers results in the TF-IDF score of a word in a document. The higher the score, the more relevant that word is in the document. To put it in more formal mathematical terms, the TF-IDF score for the word t in document d from the document set D is calculated as in equation 4.1:

$$\begin{aligned} \text{tf-idf}(t, d, D) &= \text{tf}(t, d) \cdot \text{idf}(t, D) \\ &\text{where:} \\ \text{tf}(t, d) &= \log(1 + \text{freq}(t, d)) \\ \text{idf}(t, D) &= \log\left(\frac{n}{\text{count}(d \subseteq D : t \subseteq d)}\right) \end{aligned} \tag{4.1}$$

Figure 4.9 shows an example of an input feature of the machine learning models. First, the CountVectorizer object is created. Then, the transform is fitted to the patterns to obtain the count vectorizer representation. The shape of the tf-idf fitted transformer is (8027, 284). That is the shape of the features is a sparse matrix of 284 values for the 8027 inputs or songs.

```
song tags: classic rock,rock,queen,80s,british
TF-IDF Vector: [
 0.         0.         0.         0.         0.         0.         0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.         0.         0.         0.         0.40184287 0.         0.
 0.         0.         0.         0.         0.         0.         0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.         0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.         0.         0.         0.         0.         0.45040897 0.
 0.         0.         0.         0.         0.         0.         0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.33704075 0.         0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.         0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.         0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.         0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.         0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.         0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.         0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.         0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.         0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.         0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.         0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.         0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.6050667  0.         0.         0.         0.         0.         0.         0.         0.
 0.         0.39490745 0.         0.         0.         0.         0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.         0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.         0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.         0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.         0.         0.         ]
```

Figure 4.9: Instance of input feature of machine learning model

4.6.6 Machine Learning Models

Now it is time to use some machine learning algorithms to classify which pattern belongs to which artist. The number of classes for our classifiers is the number of artists which is 16,269. For using the machine learning approach, the next step is to split the dataset into a training dataset and a test dataset. In order to do that Stratified cross-validation is used to feed the data to machine learning models. The dataset is not balanced since many artists only have one song or a small number of songs. In order to have a balanced dataset, artists with more than 100 songs are used. In that case, the number of artists is 60 and 8027 instances. The number of k for k fold cross-validation is 10. The classifiers use tf-idf feature of songs as inputs and give us which artists they belong to then later in the Flask app I map the artists and get the id of the songs. the chatbot later gives us the song id and name of the song. The classifiers used here are MultinomialNB, Decision Tree, K nearest neighbor(KNN), support vector machine (SVM), and random forest classifiers. The evaluation of different models is in the next chapter. To evaluate the performance of any machine learning models 10 fold cross-validation technique is performed. Here is an example of a machine learning-based approach using a machine learning classifier. The classifier is trained to predict the artist of a song based on its tags. Consider the tags for "ABBA" are "pop,80s, abba, Swedish, pop,80s, disco, abba,swedis." then if the tag "pop" and "female vocalists" is present in the request, predict could be the artist "ABBA." All of the classes have the same probability here. So the probability of "ABBA" and "Maddona" is the same for choosing a pop song. The chatbot randomly chooses one of the pop artists. If more than one tag is in the request, for example, "pop" and "80s", the chatbot finds the artists with those tags and again randomly returns one of the classes. Since stratified k-fold cross-validation is utilized here, all of the classes ("Artists") have a high frequency (artists with more than 100 songs) with the same probability.

4.7 Implementation Using Flask Platform

After training the classifiers and performing a model evaluation, which is extensively covered in detail in the subsequent chapter, the next step involves integrating the classifier into a Flask web application, serving as the chatbot interface. The choice made was to set up a Flask web application for this purpose. Flask, being a widely

adopted web framework in Python, is utilized to develop web applications. The decision to use Flask for deploying machine learning models is well-founded, considering several reasons:

- Easy to use: Flask is lightweight and easy to use. It provides a simple and intuitive interface that allows developers to get started quickly.
- Integration: Flask integrates well with other Python libraries and frameworks commonly used in machine learning, such as NumPy, Pandas, and Scikit-learn. This makes it easy to create a complete machine-learning pipeline using Python.
- Scalability: Flask can handle multiple requests simultaneously, making it suitable for applications that require scalability.
- Deployment: Flask is designed to be lightweight and easy to deploy, which is ideal for machine learning models. It can be deployed on a server, or a cloud platform.

The flask can be used to organize the code such that the web interface design and the underlying mathematical logic are decoupled. This logic involves handling user inputs, managing saved data, and executing mathematical simulations. HTML5, CSS, and JavaScript can be utilized to design a responsive and engaging graphical interface on the web for the user [8]. Overall, Flask is a great choice for deploying machine learning models because of its ease of use, flexibility, integration with other Python libraries, scalability, and ease of deployment. The steps that the Flask app can be taken and how it is connected to the chatbot are as follows:

- **Flask Web Application Setup:** A Flask web application has been set up to serve as the chatbot interface. The application is created using the Flask class, and you define routes to handle user interactions.
- **Endpoint for User Input:** The endpoint (/get) is defined in the Flask application to handle user input. When a user sends a POST request to this endpoint with their input message, the chatbot responds with a song recommendation from the predicted artist.
- **Preprocessing User Input:** In the /get endpoint, a function is defined to process the user's input message. This function cleans up the user input by removing punctuation, tokenizing, and converting it to lowercase.

- **Using the Classifier to Make Predictions:** The preprocessed user input is then passed to another function, which uses the trained machine learning classifier to predict the artist based on the input.
- **Song Retrieval:** Next, the id of a song is fetched from the dataset associated with that artist. The function used here finds a random song belonging to the predicted artist and returns the song's details, such as the song ID, song name, and artist

To conclude, The "Integration with Chatbot" step involves combining the trained classifier and song retrieval logic with a Flask web application to create a functional chatbot. This allows users to interact with the chatbot, provide their input, and receive song recommendations based on the predicted artist from the classifier. Figure 4.10 shows the picture of the chatbot with that is the combination of tf-idf and SVM classifier.

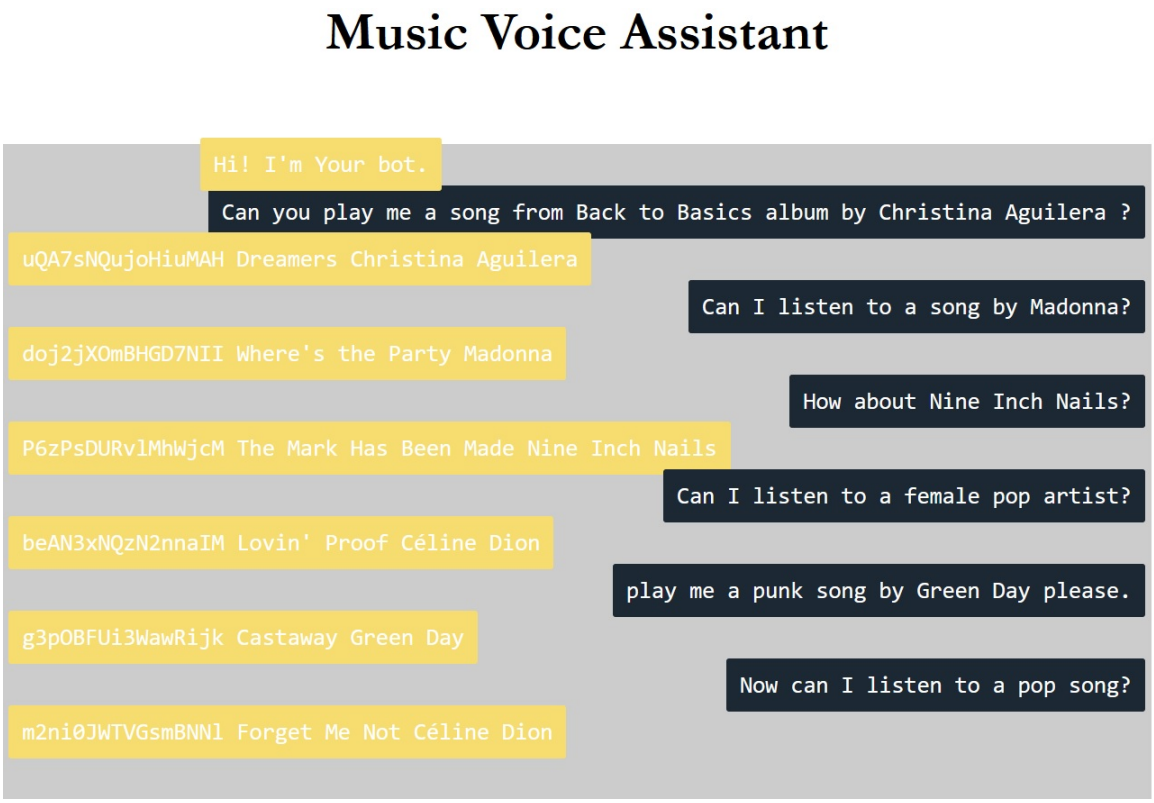


Figure 4.10: Picture of the chatbot

Chapter 5

Evaluation, Analysis, and Comparisons

5.1 Evaluation of baseline model

The baseline model is not able to understand complex statements. We didn't train the model to understand what exactly the user wants, it takes one of the patterns and returns us the ID and the name of the song that has the same pattern. We estimated the accuracy of this model to be 50% which can be improved by using machine learning models. The accuracy of the model is estimated by first finding songs that have the input pattern and then among all the artists that have songs with the input pattern, finding the one that the pattern is the most repeated. For example, if the query is slow, jazz and only two artists have songs that have both the slow and jazz tags, then the returned artist would be the one that has more songs with these two tags. This approach is quite limited as it requires exact matches and does not handle variation in the syntax of the input query.

5.2 Decision Tree

After defining the intents in the previous section, now it is time to create a decision tree for each one that maps user input to a corresponding response. We have 60 classes and each node in the decision tree would represent a possible input from the user, and each leaf node would represent a pre-defined response that is the id of the song. Stratified-KFold cross-validation is used to feed the tags into the decision tree

classifier. Then 80 percent of the songs of each artist were used for training and 20 percent of them were used for testing. The accuracy of the best model in 10-fold cross-validation is 89 percent. In the figure 5.1 the accuracy for 10-fold cross-validation can be seen. Figure 5.2 shows accuracy, precision, recall, and f1 score after performing Stratified-KFold cross-validation.

K value	Accuracy
1	0.8381795195954488
2	0.8445006321112516
3	0.0.8457648546144121
4	0.0.857142857142857
5	0.8835443037974684
6	0.8607594936708861
7	0.8468354430379746
8	0.8582278481012658
9	0.8518987341772152
10	0.850632911392405

Table 5.1: K-fold cross-validation accuracy

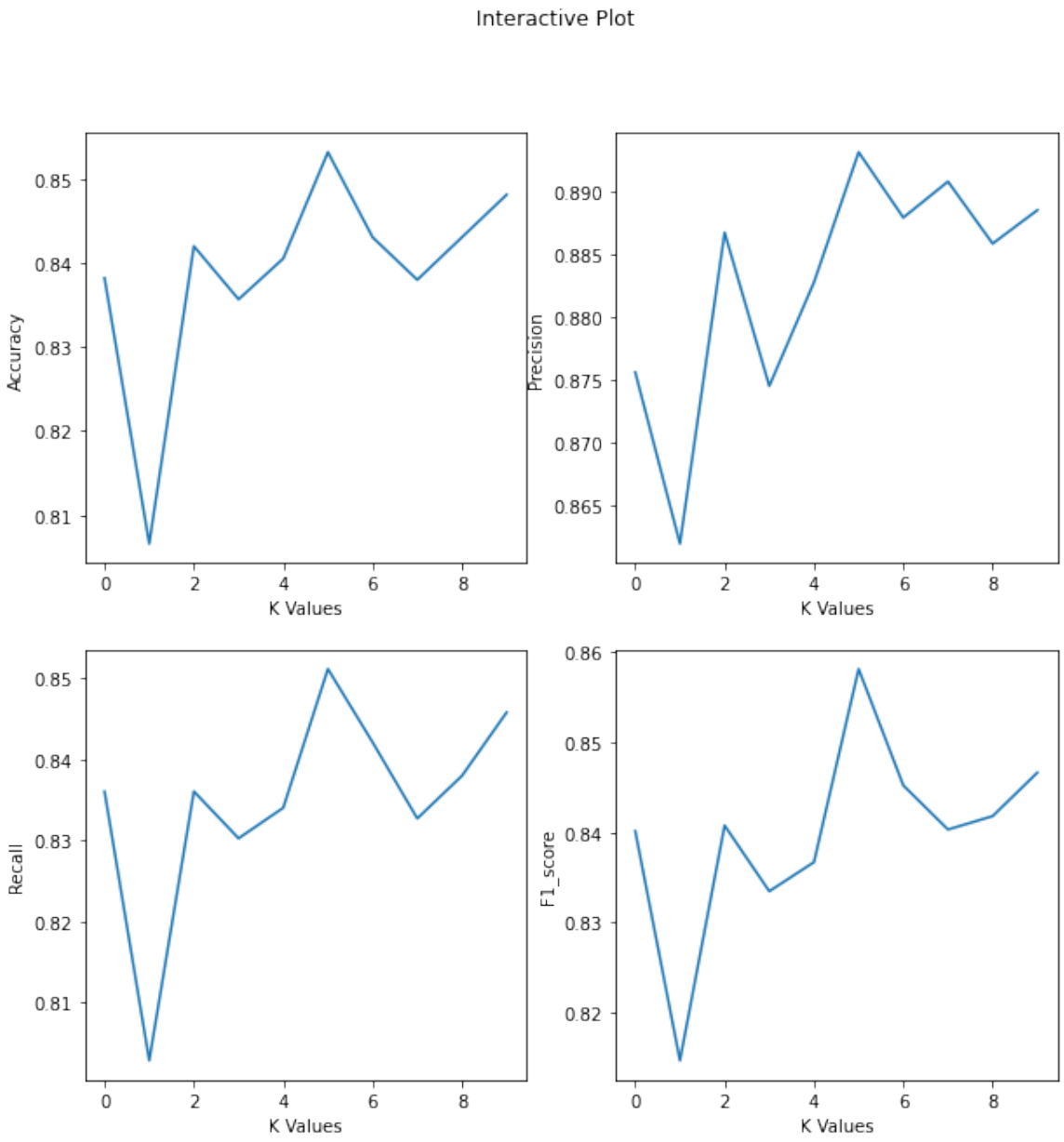


Figure 5.1: Decision tree evaluation metrics

In figure 5.2 the decision tree with the max-depth of two can be seen. This is a decision tree structure, where each line represents a decision based on a feature and a threshold value. The tree starts with the root node and iteratively splits the data based on the feature and threshold with the best information gained at each step until all leaves are pure or stopping criteria are met. The root node is based on feature 264 with a threshold of 0.12. If a data point has a feature value less than or equal to 0.12, it follows the left branch, otherwise, it follows the right branch. 5.3 shows a

part of the text representation of the decision tree with a depth of 12. As it can be observed, the tree then splits again based on feature 215 and its threshold of 0.48, then on feature 197 and its threshold of 0.14, and so on, until the final decision is reached. Each leaf node represents a class label or a decision based on the data that arrived at that point.

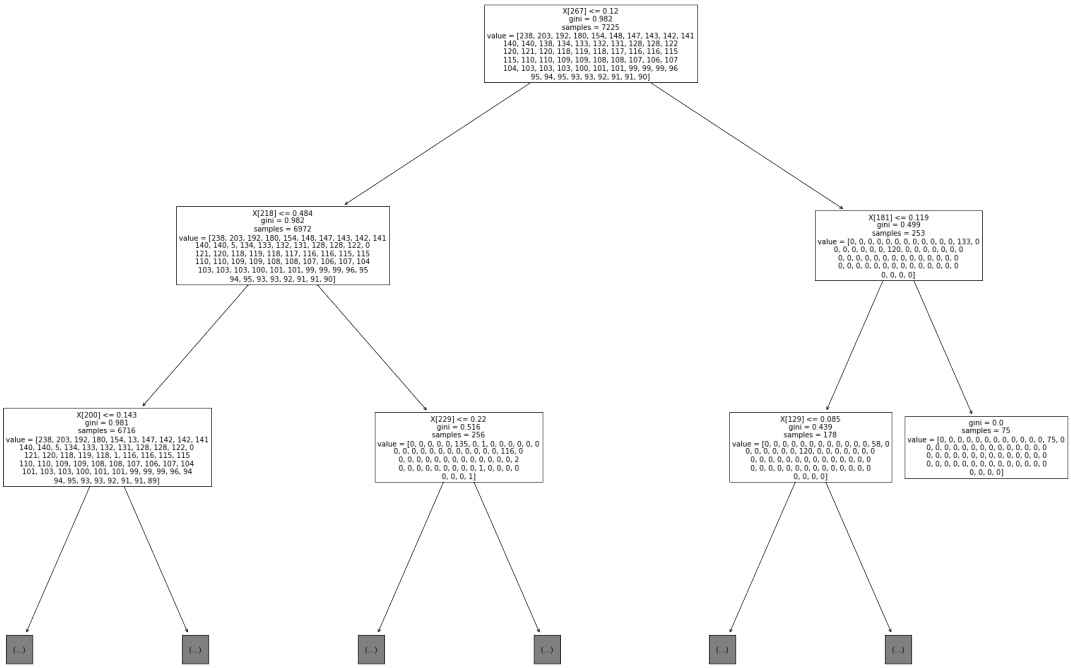


Figure 5.2: Decision tree structure

```
|--- feature_264 <= 0.12
|   |--- feature_215 <= 0.48
|   |   |--- feature_197 <= 0.14
|   |   |   |--- feature_216 <= 0.33
|   |   |   |   |--- feature_147 <= 0.09
|   |   |   |   |   |--- feature_140 <= 0.36
|   |   |   |   |   |   |--- feature_258 <= 0.22
|   |   |   |   |   |   |   |--- feature_219 <= 0.11
|   |   |   |   |   |   |   |   |--- feature_276 <= 0.24
|   |   |   |   |   |   |   |   |   |--- feature_272 <= 0.21
|   |   |   |   |   |   |   |   |   |   |--- feature_177 <= 0.13
|   |   |   |   |   |   |   |   |   |   |   |--- truncated branch of depth 74
|   |   |   |   |   |   |   |   |   |   |--- feature_177 >  0.13
|   |   |   |   |   |   |   |   |   |   |   |--- class: 37
|   |   |   |   |   |   |   |   |   |--- feature_272 >  0.21
|   |   |   |   |   |   |   |   |   |   |--- class: 16
|   |   |   |   |   |   |   |   |--- feature_276 >  0.24
|   |   |   |   |   |   |   |   |   |--- feature_211 <= 0.32
|   |   |   |   |   |   |   |   |   |   |--- feature_193 <= 0.25
|   |   |   |   |   |   |   |   |   |   |   |--- truncated branch of depth 10
|   |   |   |   |   |   |   |   |   |   |--- feature_193 >  0.25
|   |   |   |   |   |   |   |   |   |   |   |--- class: 40
|   |   |   |   |   |   |   |   |   |--- feature_211 >  0.32
|   |   |   |   |   |   |   |   |   |   |--- feature_191 <= 0.13
|   |   |   |   |   |   |   |   |   |   |   |--- class: 40
|   |   |   |   |   |   |   |   |   |   |--- feature_191 >  0.13
|   |   |   |   |   |   |   |   |   |   |   |--- class: 15
|   |   |   |   |   |   |   |   |--- feature_219 >  0.11
|   |   |   |   |   |   |   |   |   |--- feature_178 <= 0.11
|   |   |   |   |   |   |   |   |   |   |--- feature_7 <= 0.35
|   |   |   |   |   |   |   |   |   |   |   |--- class: 30
|   |   |   |   |   |   |   |   |   |   |--- feature_7 >  0.35
|   |   |   |   |   |   |   |   |   |   |   |--- class: 35
|   |   |   |   |   |   |   |   |   |--- feature_178 >  0.11
|   |   |   |   |   |   |   |   |   |   |--- feature_219 <= 0.46
|   |   |   |   |   |   |   |   |   |   |   |--- class: 50
|   |   |   |   |   |   |   |   |   |   |--- feature_219 >  0.46
```

Figure 5.3: text representation of the decision tree

The hyperparameters that are used in the decision tree can be seen in table 5.2 which are hyperparameters in the scikit-learn implementation of the Decision Tree classifier. The confusion matrix of the decision tree model can be seen in figure 5.4. The decision trees of other classification methods can be found in the appendix.

Hyper Parameter	Value
ccp_alpha	0.0
class_weight	None.
criterion	gini
max_depth	None
splitter	best
min_samples_leaf	1
min_samples_split	2

Table 5.2: Hyper parameters of decision tree

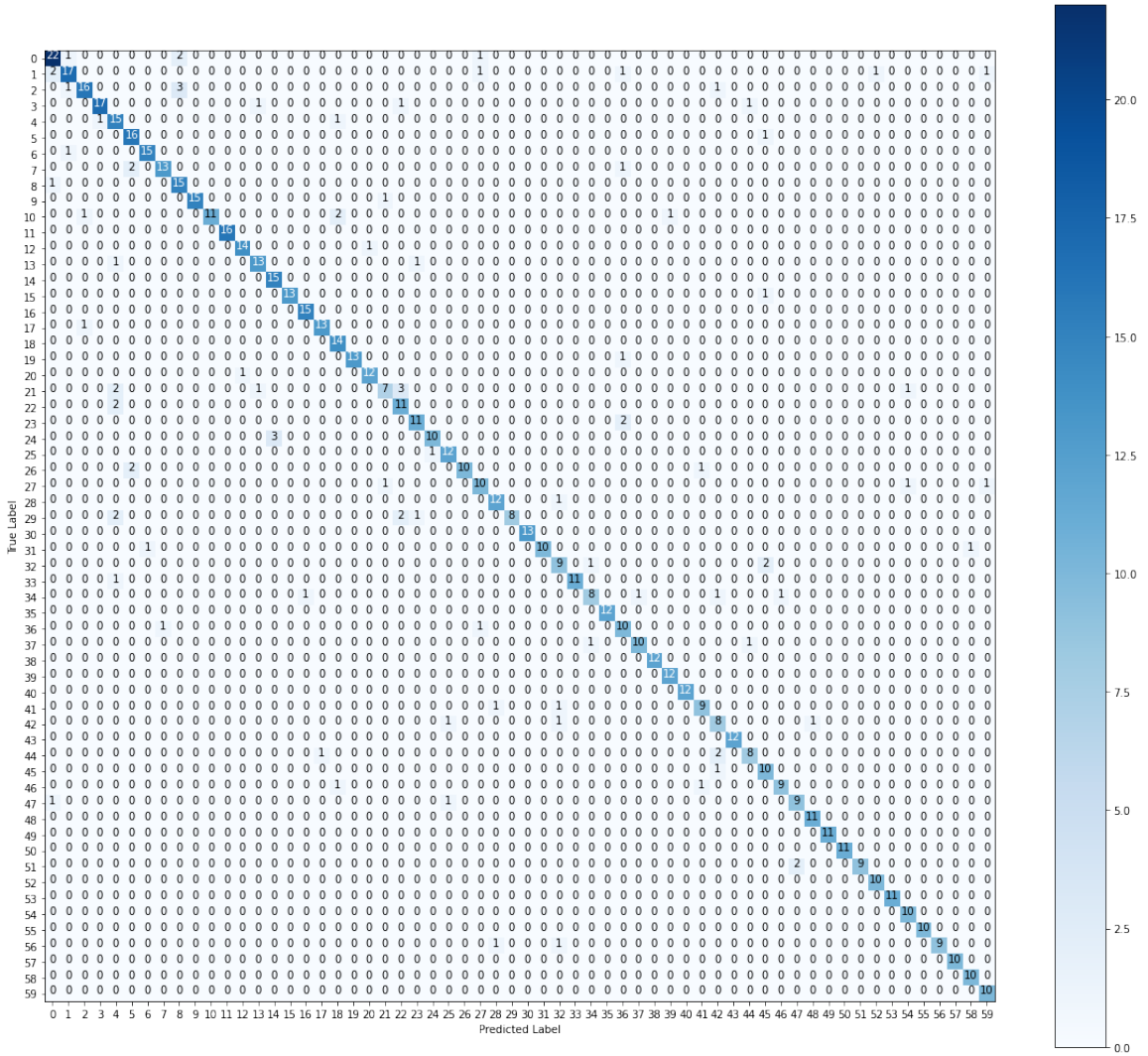


Figure 5.4: The confusion matrix of decision tree model

5.3 Naive Bayes

Multinomial Naive Bayes (MNB) is a probabilistic machine learning algorithm that's commonly used for text classification problems. It's a variant of the Naive Bayes algorithm that's designed for classification problems such as sentiment analysis, spam detection, and document classification. MNB works by using Bayes' theorem to compute the probability of a document belonging to a particular class, given the observed word counts in the document. The algorithm assumes that the feature values are conditionally independent, which means that the presence or absence of one feature does not affect the probability of any other feature. To train an MNB model, the algorithm first calculates the prior probabilities of each class based on the frequency of the classes in the training data. Then, for each class, it calculates the likelihood probabilities of each word given that class, based on the frequency of the words in the training documents for that class. During prediction, the algorithm takes the input text and computes the conditional probability of each class given the observed word frequencies in the document, using Bayes' theorem. The class with the highest probability is then predicted as the output class for the document. It's based on the Bayes theorem of probability theory and assumes that the occurrence of a feature is independent of the occurrence of any other feature. To use MNB for classification, at first, we split data into training and testing sets. After vectorizing the data using TF-IDF, which assigns weights to words based on their frequency in the document. Figure 5.5 shows the evaluation metrics after using MNB model. Table 5.3 indicates accuracy in 10-fold cross-validation.

K value	Accuracy
1	0.8381795195954488
2	0.8065739570164349
3	0.8419721871049305
4	0.8356510745891277
5	0.8405063291139241
6	0.8531645569620253
7	0.8430379746835444
8	0.8379746835443038
9	0.8430379746835444
10	0.8481012658227848

Table 5.3: K-fold cross-validation accuracy for MultinomialNB model

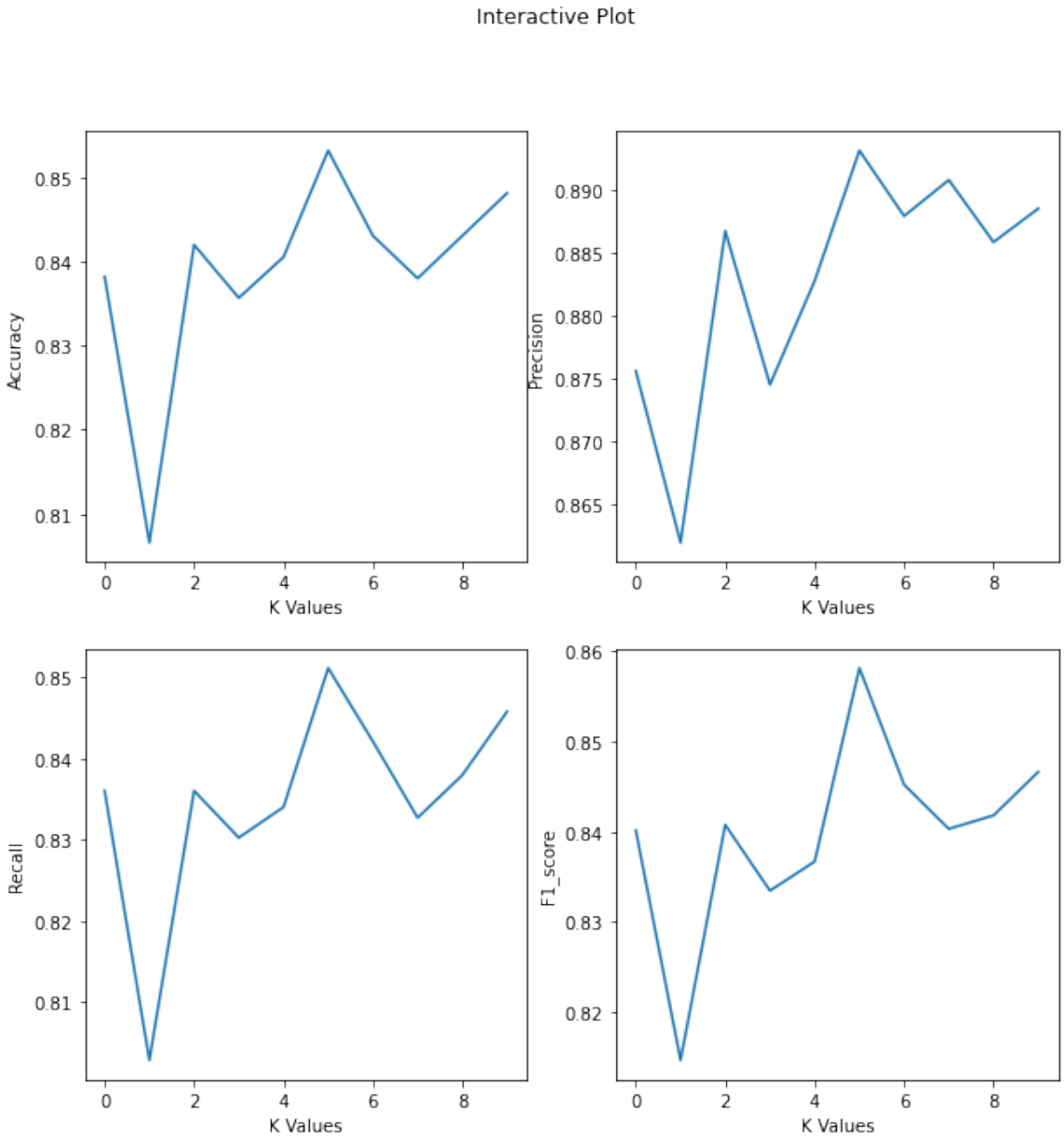


Figure 5.5: MultinomialNB evaluation metrics

5.4 K-nearest neighbors

K-nearest neighbors (KNN) is a type of supervised learning algorithm used for classification and regression. In KNN, the output for a given test instance is computed based on the k closest training instances in the feature space. Figure 5.6 shows different k values ranging from 1 to 10 among Stratified-KFold cross-validation accuracy. As can

be observed the higher the k value gets the accuracy of the model improves. After performing cross-validation the accuracy, recall, precision, and f1 score of the model can be seen in Figure 5.7. Table 5.4 shows the accuracy in k-fold cross-validation.

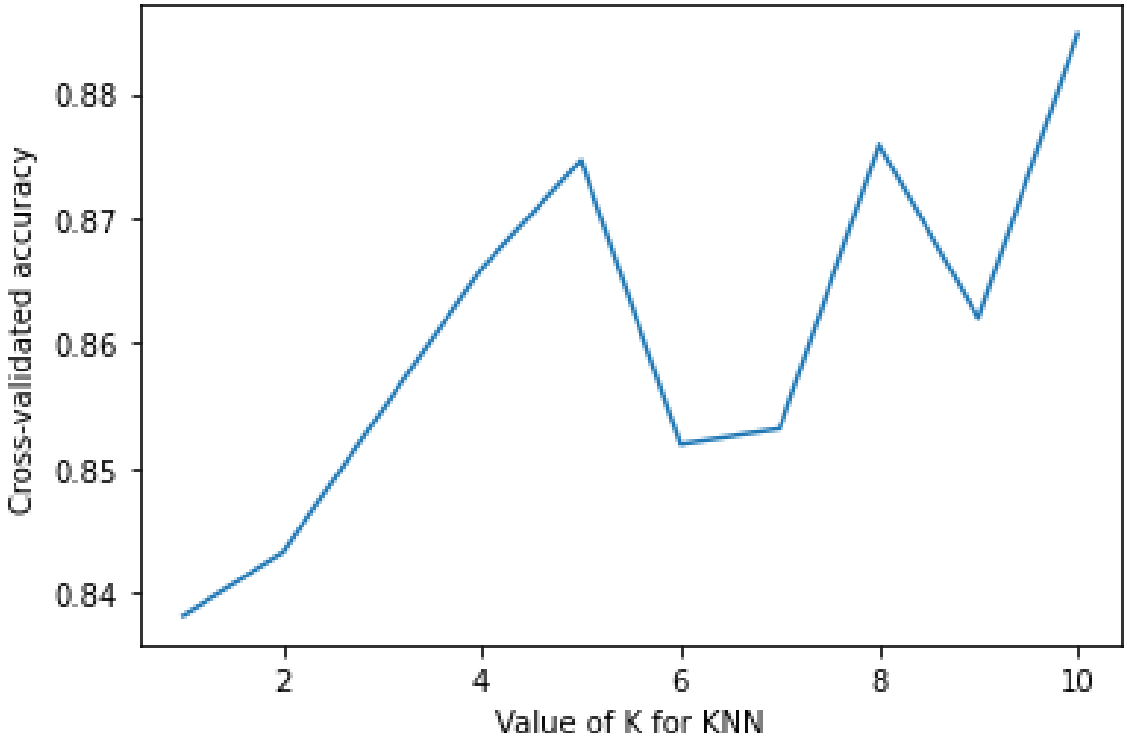


Figure 5.6: Different k values among Stratified-KFold cross-validation accuracy

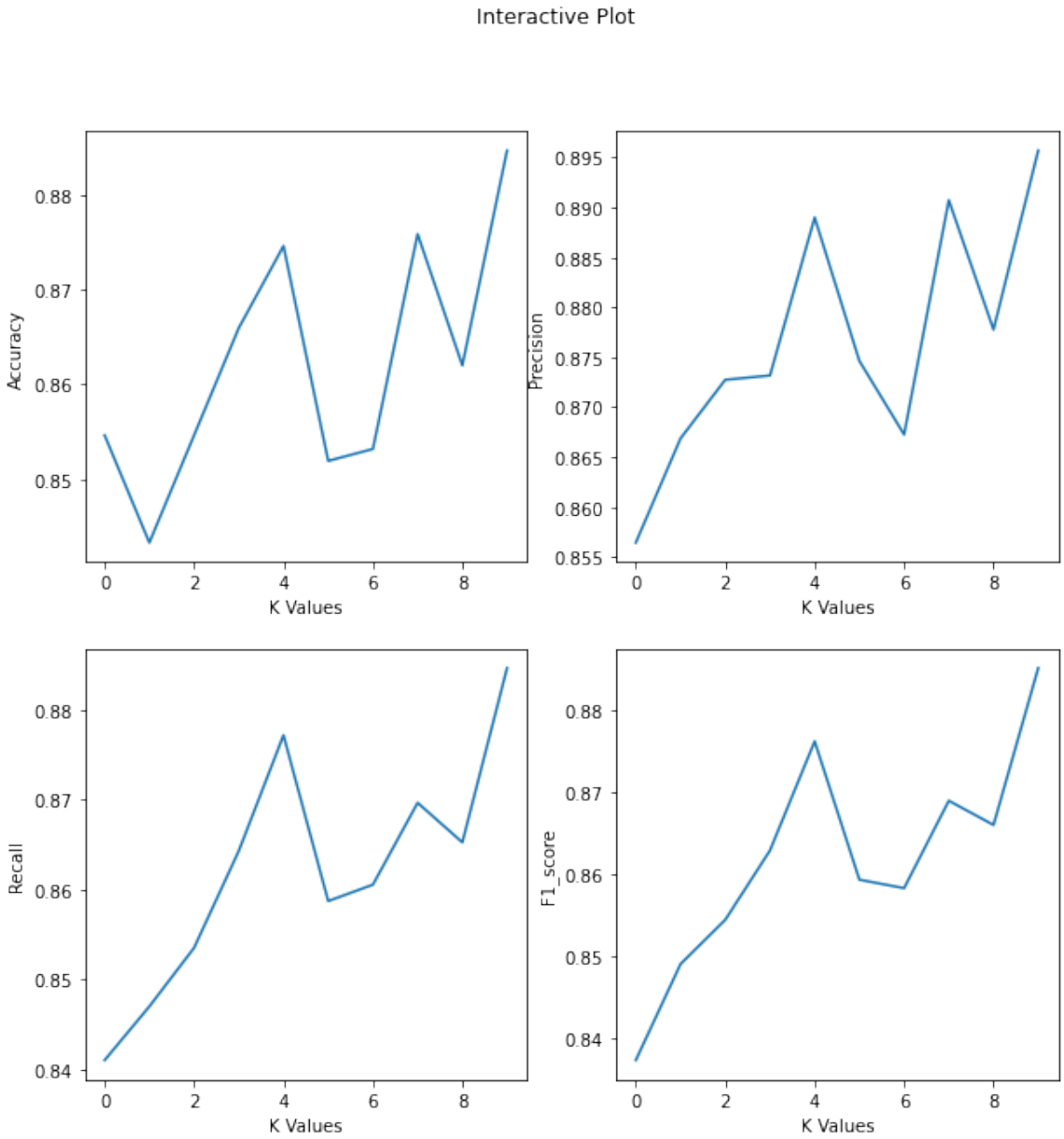


Figure 5.7: K-nearest neighbor evaluation metrics

K value	Accuracy
1	0.854614412136536
2	0.843236409608091
3	0.854614412136536
4	0.865992414664981
5	0.8746835443037975
6	0.8518987341772152
7	0.8531645569620253
8	0.8759493670886076
9	0.8620253164556962
10	0.8848101265822785

Table 5.4: K-fold cross-validation accuracy for SVM classifier

5.5 Support Vector Machine

Support Vector Machines (SVMs) are a class of machine learning models used for classifications. SVMs work by constructing a hyperplane in a high-dimensional space that separates different classes by maximizing the margin between them. SVMs are powerful models that can work well in high-dimensional spaces and can handle non-linear decision boundaries through the use of kernel functions. They can also handle data with noise and outliers by adjusting the hyperparameters of the model. The training of SVM involves the optimization of a cost function that takes into account the distance between the data points and the decision boundary, as well as any misclassifications. the SVM model hyperparameters can be seen in the table 5.5. The accuracy of the model after performing k-fold cross-validation can be seen in the table 5.6

Hyperparameter	Value
C	0.1
cache_size	200
class_weight	None
decision_function_shape	ovr
degree	3
gamma	scale
kernel	rbf
max_iter	-1
shrinking	True
tol	0.001

Table 5.5: Get the hyperparameters of the model

K value	Accuracy
1	0.8735777496839444
2	0.8584070796460177
3	0.8900126422250316
4	0.8900126422250316
5	0.8949367088607595
6	0.8810126582278481
7	0.8746835443037975
8	0.8949367088607595
9	0.8784810126582279
10	0.8886075949367088

Table 5.6: K-fold cross-validation accuracy for SVM classifier

To find the best hyperparameter. In table different kernels in SVM algorithm have been used 5.7, and table 5.9 shows the best C parameter. Table 5.8 shows alternative SVM classifiers.

Kernel	Accuracy	Precision	Recall	F1 Score
linear	0.868171	0.899029	0.878344	0.879771
poly	0.663086	0.890680	0.681552	0.742105
rbf	0.820222	0.865917	0.824159	0.832793
sigmoid	0.840337	0.875807	0.844749	0.845695

Table 5.7: Comparison of mean metrics with different kernels using StratifiedKFold with n_splits of 10

Model	Accuracy	Precision	Recall	F1 Score
SVC	0.882467	0.901077	0.882441	0.885649
NuSVC	0.837681	0.903136	0.833649	0.853063
LinearSVC	0.882215	0.892196	0.884255	0.882905

Table 5.8: Alternative SVM classifiers

C	Accuracy
0.1	0.8223714573764983
1	0.8824668341628128
10	0.8823415321096514

Table 5.9: Finding the best "C" parameter in grid search

5.6 Random Forest

Random Forest is a popular machine learning algorithm used for classification, regression, and other tasks. It is an ensemble method that combines multiple decision trees to make a prediction. In a Random Forest, each tree is trained on a randomly selected subset of the data, and at each node in the tree, a random subset of features is considered for splitting the data. This helps to reduce overfitting and improve the generalization performance of the model. The key steps in training a Random Forest are as follows:

- *D*ata Sampling: Randomly select a subset of the data for each tree in the forest.
- *F*eature Sampling: Randomly select a subset of features for each split in each tree.

- *T*ree Training: Train a decision tree on the selected date and features using a split criterion such as Gini impurity or entropy.
- *E*nsemble: Combine the predictions of all the trees in the forest to make a final prediction. This is typically done by majority vote for classification problems or by averaging for regression problems.

Random Forest has several advantages over other machine-learning algorithms:

- *R*obustness: Random Forest is less prone to overfitting than single decision trees, and it can handle noisy or missing data.
- *A*ccuracy: Random Forest has been shown to perform well on a wide range of classification and regression problems, and it can achieve high accuracy with relatively few hyperparameters to tune
- *I*nterpretability: Random Forest can provide insight into feature importance and the decision-making process of the model, which can be useful for understanding the problem and making informed decisions.

the hyperparameters that are used in the random forest to tune the model and get the optimized hyperparameters can be observed in 5.10 Also the accuracy of the k fold cross validation before using the tunned hyperparameters can be seen in table 5.11.

Hyperparameter	Value
max_features	['auto', 'sqrt', 'log2', None]
max_depth	[10,20,30]
criterion	['gini', 'entropy']
bootstrap	[True, False]
n_estimators	[20, 50, 100, 150, 200]
gamma	scale
kernel	rbf
shrinking	True
tol	0.001

Table 5.10: Hyperparameter used in GridSearchCV to find the optimized hyperparameters for random forest model

K	Accuracy
1	0.8647281921618205
2	0.8558786346396966
3	0.8571428571428571
4	0.8710493046776233
5	0.8949367088607595
6	0.8746835443037975
7	0.8759493670886076
8	0.8886075949367088
9	0.8708860759493671
10	0.889873417721519

Table 5.11: Accuracy of the random forest model before hyperparameter optimization

the accuracy improved by tuning the hyperparameter and selecting the best parameters for the random forest by using the GridSearchCV class from the scikit-learn library for hyperparameter tuning. The accuracy increased in table 5.12 and the best hyperparameters can be seen in table 5.13. Figure 5.8 compares the accuracy before and after using the optimized parameters in the random forest model.

Hyperparameter	Value
bootstrap	False
criterion	entropy
max_depth	30
max_features	log2
n_estimators	200

Table 5.12: Optimized hyperparameters for the random forest model

K	Accuracy
1	0.8798988621997471
2	0.8697850821744627
3	0.8849557522123894
4	0.8988621997471555
5	0.9050632911392406
6	0.8911392405063291
7	0.8911392405063291
8	0.9
9	0.8873417721518987
10	0.9025316455696203

Table 5.13: Accuracy of the random forest model after hyperparameter optimization

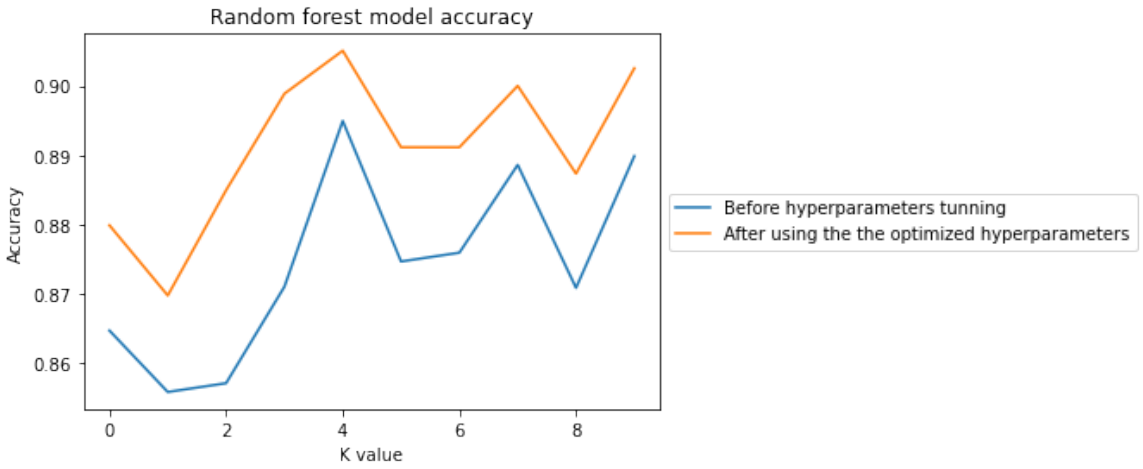


Figure 5.8: K fold cross validation accuracy improvement after hyperparameter tunning

5.7 Comparison of different machine learning models

Table 5.14: Comparison of Evaluation Metrics for Different Methods

Method	Accuracy	Precision	Recall	F1 Score
Decision Tree	0.850633	0.868867	0.855475	0.85574
KNN	0.862099	0.877417	0.863855	0.863682
Naive Bayes	0.83882	0.882782	0.834899	0.83976
Random Forest	0.873105	0.88752	0.874453	0.875653
SVM	0.882467	0.901077	0.882441	0.885649

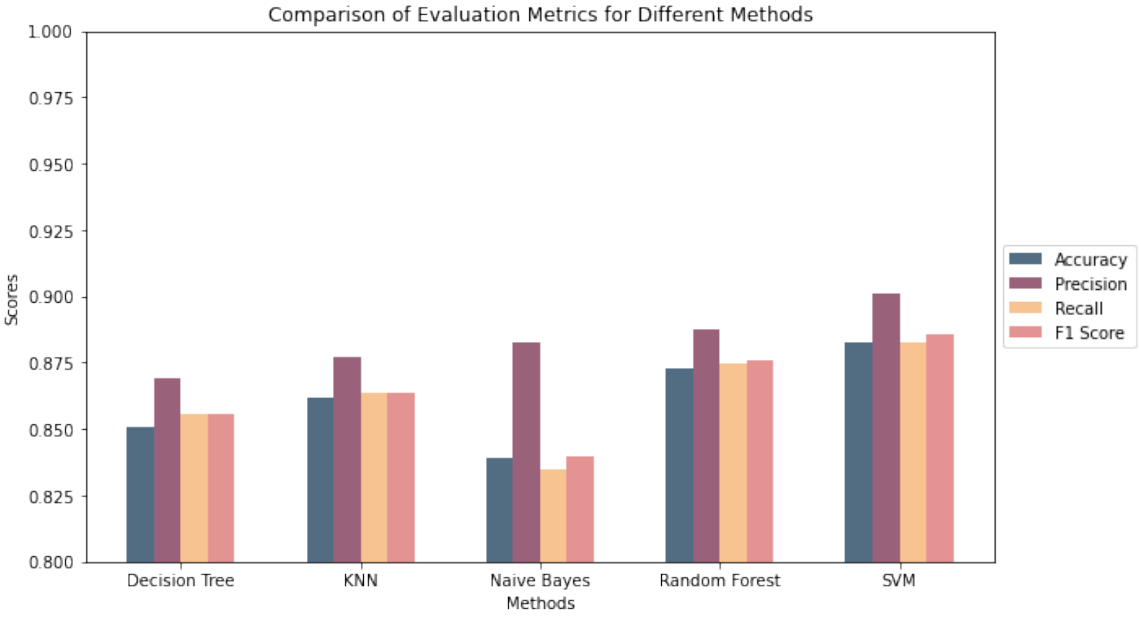


Figure 5.9: Comparison of mean different metrics of machine learning models

5.8 Word2vec

Another method that can be used for the chatbot development is word2vec. Word2vec is a method that converts individual words into vectors. These vectors capture the essential characteristics of words, such as their semantic meaning and contextual usage. It allows us to represent words in a mathematical form, enabling machines to process and understand textual data more effectively. By utilizing word embeddings,

words are no longer treated as isolated symbols but as points in a high-dimensional space, where the distances and relationships between vectors reflect the relationships between words in the underlying language.

The first step to Implementing the chatbot with word2vec is to import the necessary libraries for the Word2Vec model training and data preprocessing. the Word2Vec class is imported from the Gensim library, which is used for training and working with Word2Vec models.the pad_sequences function is imported from TensorFlow Keras to pad sequences, and the word_tokenize function from NLTK to tokenize sentences into individual words. the 'tags' were converted into a list of sentences. Then, each sentence is tokenized using the word_tokenize function to obtain a list of tokenized sentences. Each sentence is represented as a list of words. Then, a Word2Vec model is created and trained on the tokenized sentences. Word2Vec class from the Gensim library is used and passed the tokenized sentences as input. The vector_size parameter specifies the dimensionality of the word embeddings, which determines the size of the vectors representing each word. The window parameter defines the maximum distance between the current word and the predicted word within a sentence, indicating the context window size. The min_count parameter sets the minimum frequency of a word to be included in the vocabulary. Table 5.15 shows different hyperparameters and the accuracy of the model. The next step is to define the architecture of the neural network model that utilizes the Word2Vec word embeddings. The architecture is defined using the Keras functional API. The architecture of the neural network is as follows:

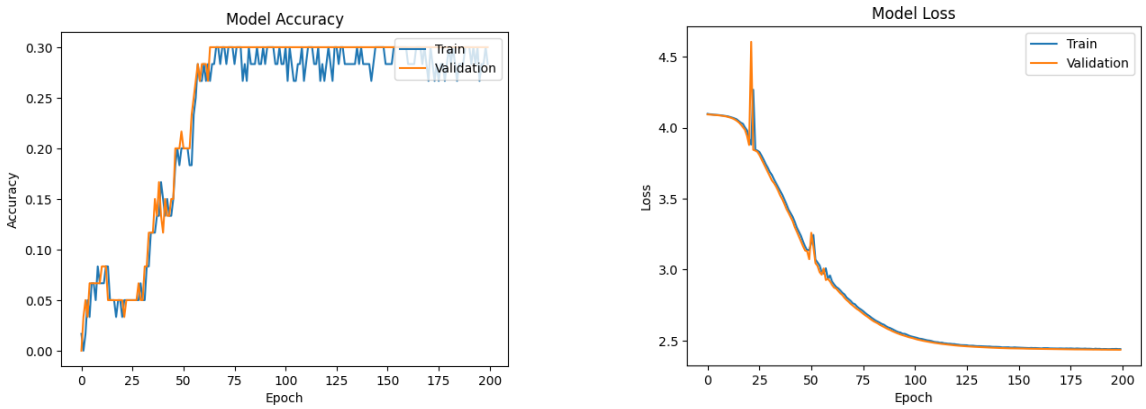
- An embedding layer that maps each word index to its corresponding word embedding vector.
- A bidirectional LSTM layer that captures the contextual information from both directions of the input sequences.
- A global max pooling layer that extracts the most important features from the LSTM output sequence.
- A dense layer with ReLU activation introduces non-linearity to the model.
- n output layer with softmax activation that produces the final class probabilities.

Vector Size	Window	Min Count	Loss	Accuracy
50	3	1	2.502648	0.316667
50	3	5	2.417916	0.316667
50	3	10	2.461397	0.316667
50	5	1	2.428306	0.316667
50	5	5	2.436245	0.316667
50	5	10	2.439840	0.316667
50	7	1	2.396932	0.316667
50	7	5	2.585840	0.266667
50	7	10	2.545551	0.316667
100	3	1	2.539012	0.316667
100	3	5	2.627056	0.283333
100	3	10	2.526539	0.316667
100	5	1	2.437509	0.316667
100	5	5	2.455441	0.316667
100	5	10	2.381080	0.316667
100	7	1	2.373605	0.316667
100	7	5	2.395071	0.316667
100	7	10	2.406547	0.316667
200	3	1	2.433117	0.316667
200	3	5	2.467819	0.316667
200	3	10	2.542110	0.316667
200	5	1	2.501073	0.300000
200	5	5	2.630915	0.300000
200	5	10	2.416222	0.316667
200	7	1	2.460558	0.316667
200	7	5	2.438662	0.316667
200	7	10	2.485887	0.300000

Table 5.15: Word2Vec Model Performance with different hyper_parameters

Finally figure 5.10 shows the performance of the model with vector_size of 100, window size of 5 and min_count of 1. As can be seen, the classification accuracy is significantly lower than the traditional machine learning models. there is not sufficient data in the training set to generate meaningful embeddings which results in the

word2vec features not capturing as well information as the tf-idf vectors. Further investigation is needed to better understand if that's the case and whether using a pre-trained word2vec model could be used in our context.



(a) Accuracy of wave2vec model

(b) Loss of wave2vec model

Figure 5.10: Learning curve of wave2vec model

5.9 Bidirectional Encoder Representations from Transformers

In 2019, [14] presented a novel language model called BERT, which stands for Bidirectional Encoder Representations from Transformers. In contrast to recent language models like [32] and [33], BERT is specifically designed to pre-train deep bidirectional representations using unlabeled text. The pre-trained Bert model for English language text processing is used. Table 5.16 shows the used Bert model.

Specification	**Value**
Transformer Layers	12
Hidden Size	768
Self-Attention Heads	12
Model Version	4

Table 5.16: Specifications of the BERT model

The first step for using BERT in this study involved data preparation, where numeric labels were required. Thus, integer values were assigned to the descriptions

of topics. The LabelEncoder from the sklearn library was utilized to convert each unique artist into a numerical value.

The subsequent step involved splitting the dataset into training and testing datasets. This was achieved by allocating 80 percent of each unique artist's data for training and reserving the remaining 20 percent of the tags for the test dataset. Once the dataset was divided, it was then fed into the BERT model for further processing and training.

Then, the input data for training the BERT model was prepared by tokenizing the text and converting it into a format compatible with BERT. This process included tokenization, converting the tokens to token IDs, and ensuring a fixed sequence length. To determine the appropriate maximum length for the tokens, the distribution of word lengths in the text was analyzed, as shown in Figure **??**. This analysis helped in choosing a suitable maximum sequence length to handle the text data effectively during tokenization and input preparation for the BERT model.

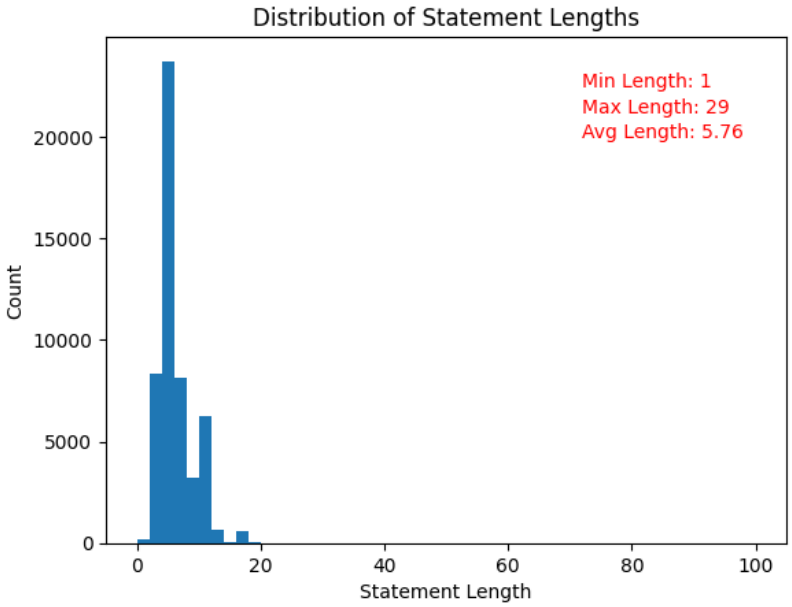


Figure 5.11: The distribution of words lengths in tags

The next step is to define the BERT model architecture using the Keras functional API as follows:

- It starts with an input layer that takes the input token IDs.

- The input is then passed through an embedding layer that maps each token ID to a dense vector representation of size 768.
- A bidirectional LSTM layer with 29 units processes the embedded sequences in both forward and backward directions.
- A global max pooling layer extracts the most important features from the LSTM outputs.
- A dense layer with 64 units and ReLU activation function further processes the pooled features.
- The final output layer with a softmax activation function predicts the class probabilities for the classification task. The number of units in this layer corresponds to the number of classes.

And finally, Compiling the model by specifying the loss function, optimizer, and evaluation metrics. The model uses sparse categorical cross-entropy as the loss function since the labels are provided as integers. The Adam optimizer is used to optimize the model parameters. The accuracy metric is used to evaluate the model performance during training. Finally, the model is trained using the training data for a specified number of epochs (1000 in this case), with a batch size of 32. It also validates the model's performance on the validation data. Evaluates the trained model on the testing data and print the test loss and accuracy. the model works architecture of the Bert model can be seen in the table 5.17. The model is trained in 200 and 1000 epochs. The learning curve of both can be seen in figure 5.12

Layer (type)	**Output Shape**	**Param #**
input_ids (InputLayer)	((None, 29)	0
embedding (Embedding)	(None, 29, 768)	23,441,664
bidirectional (Bidirectional)	(None, 29, 256)	918,528
global_max_pooling1d (GlobalMaxPooling1D)	(None, 256)	0
dense (Dense)	(None, 64)	16,448
output (Dense)	(None, 60)	3,900 × *num_classes*
Total params: 24,380,540		
Trainable params: 938,876		
Non-trainable params: 23,441,664		

Table 5.17: Bert model Architecture

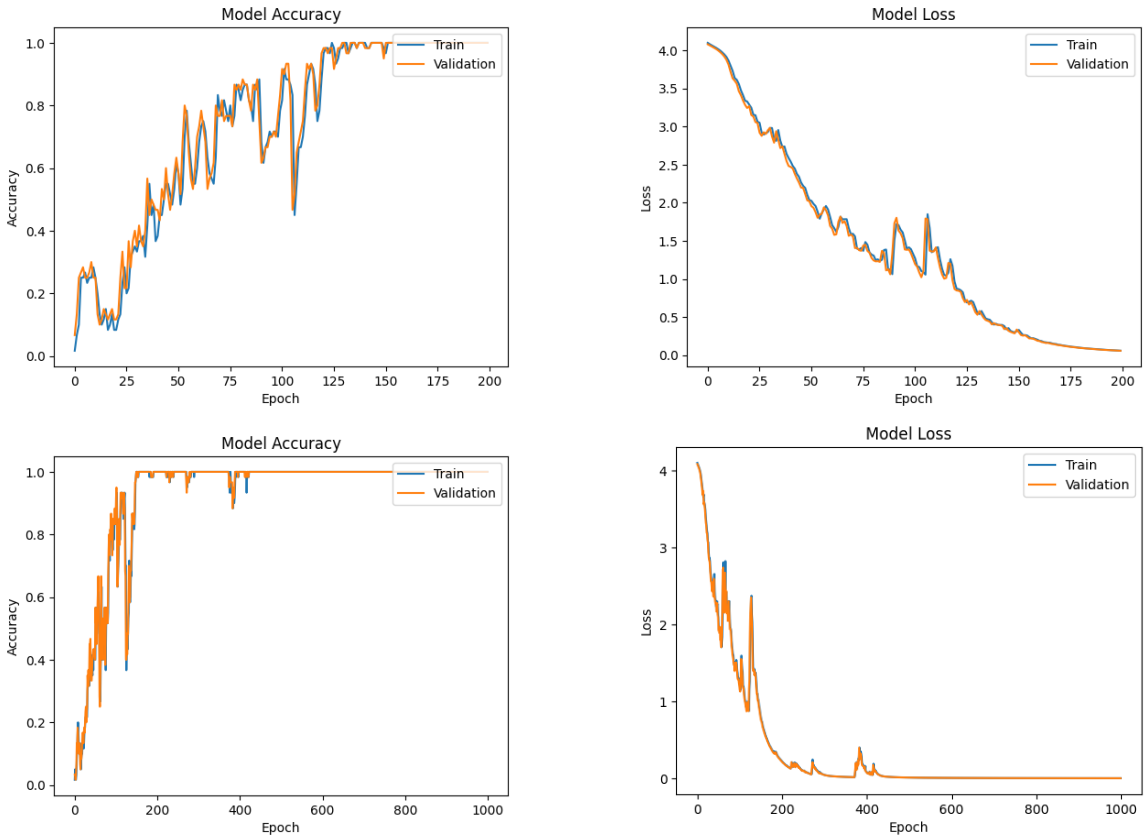


Figure 5.12: The learning curve of Bert model

As can be observed from the curve it reached 1 and there is a possibility that it is overfitted. But the Bert model could be considered Better if the dataset was bigger

with more sentences.

Chapter 6

Conclusions

In conclusion, this study aims to fill the gap in the digital music services market by developing a music chatbot specifically designed for music-related interactions. The chatbot offers a seamless and interactive experience, allowing users to discover, explore, and engage with music in a conversational manner. The research explores various techniques, including pattern-matching approaches, TF-IDF combined with machine learning algorithms, wave2vec embeddings, and the BERT model, to identify the most effective methods for creating engaging and responsive music chatbots. The developed chatbot is thoroughly evaluated and analyzed using the Music4All dataset, providing insights into the strengths and weaknesses of different approaches and aiding in the refinement and improvement of music chatbot systems. The first step was data preparation. Then the numerical value of 60 classes and TF-IDF sparse matrices of 8027 songs were fed into the machine learning algorithms. The algorithms that are used are decision tree, random forest, KNN, and SVM as classification algorithms and In section 5 the evaluation of these algorithms is discussed. The result of the experiment shows the best method to design a chatbot by using metadata is a combination of TF-IDF and SVM model which reached the highest accuracy of 91%. The more state-of-the-art methods such as BERT and Word2vec are not useful for this chatbot since the BERT model because of the small size of the dataset was overfitted. Word2vec model was also trained with different parameters but it did not exceed the accuracy of using TF-IDF and SVM. Finally after the classification of the artists, the result is given to the Flask app and then the Flask app gives us the name of the song and the id of the song with the requested tag. This study contributes valuable experiments using the Music4All dataset and has the potential to be converted into a speech recognition virtual assistant when coupled with a speech recognition

and text-to-speech synthesis system. Also, it is possible to generate a playlist. It is possible to have the classifier output probabilities for each class and then return the top K most likely artists or randomly select songs in proportion to the predicted probability. It is also worth to try expanding the size of the dataset and use lyrics in the chatbot. Another experiment for future works can be exploring in more detail with more data deep learning approaches including large language models(LLMs). The final improvement for this research is to design some user studies for evaluation. This method alongside the classification accuracy can help us better understand how well the chatbot works and give us insights about how it can be improved.

Bibliography

[1] IBM Watson is AI for business, url = https://www.ibm.com/watson.,.

[2] Meet your Google Assistant, url = https://assistant.google.com/,.

[3] Microsoft.

[4] Sameera A Abdul-Kader and John C Woods. Survey on chatbot design techniques in speech conversation systems. *International Journal of Advanced Computer Science and Applications*, 6(7), 2015.

[5] Bayan AbuShawar and Eric Atwell. Alice chatbot: Trials and outputs. *Computación y Sistemas*, 19(4):625–632, 2015.

[6] Eleni Adamopoulou and Lefteris Moussiades. Chatbots: History, technology, and applications. *Machine Learning with Applications*, 2:100006, 2020.

[7] Akiko Aizawa. An information-theoretic perspective of tf–idf measures. *Information Processing & Management*, 39(1):45–65, 2003.

[8] Matthew S Bonney, Marco De Angelis, Mattia Dal Borgo, Luis Andrade, Sandor Beregi, Nidhal Jamia, and David J Wagg. Development of a digital twin operational platform using python flask. *Data-Centric Engineering*, 3:e1, 2022.

[9] Luka Bradeško and Dunja Mladenić. A survey of chatbot systems through a loebner prize competition. In *Proceedings of Slovenian language technologies society eighth conference of language technologies*, volume 2, pages 34–37. sn, 2012.

[10] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

[11] Michael A Casey, Remco Veltkamp, Masataka Goto, Marc Leman, Christophe Rhodes, and Malcolm Slaney. Content-based music information retrieval: Current directions and future challenges. *Proceedings of the IEEE*, 96(4):668–696, 2008.

[12] Kyungyong Chung and Roy C Park. Chatbot-based heathcare service with a knowledge base for cloud computing. *Cluster Computing*, 22:1925–1937, 2019.

[13] Minjee Chung, Eunju Ko, Heerim Joung, and Sang Jin Kim. Chatbot e-service and customer satisfaction regarding luxury brands. *Journal of Business Research*, 117:587–595, 2020.

[14] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[15] Ari Aulia Hakim, Alva Erwin, Kho I Eng, Maulahikmah Galinium, and Wahyu Muliady. Automated document classification for news article in bahasa indonesia based on term frequency inverse document frequency (tf-idf) approach. In *2014 6th international conference on information technology and electrical engineering (ICITEE)*, pages 1–4. IEEE, 2014.

[16] Matthew B Hoy. Alexa, siri, cortana, and more: an introduction to voice assistants. *Medical reference services quarterly*, 37(1):81–88, 2018.

[17] Rishit Jain, Ritik Sharma, Preeti Nagrath, and Rachna Jain. Music genre classification chatbot. In *Proceedings of Second International Conference on Computing, Communications, and Cyber-Security: IC4S 2020*, pages 393–408. Springer, 2021.

[18] Jiyou Jia and Weichao Chen. Motivate the learners to practice english through playing with chatbot csiec. In *Technologies for E-Learning and Digital Entertainment: Third International Conference, Edutainment 2008 Nanjing, China, June 25-27, 2008 Proceedings 3*, pages 180–191. Springer, 2008.

[19] Yucheng Jin, Wanling Cai, Li Chen, Nyi Nyi Htun, and Katrien Verbert. Musicbot: Evaluating critiquing-based music recommenders with conversational interaction. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, pages 951–960, 2019.

[20] Namcheol Jung and Ghang Lee. Automated classification of building information modeling (bim) case studies by bim use based on natural language processing (nlp) and unsupervised learning. *Advanced Engineering Informatics*, 41:100917, 2019.

[21] Tetsuya Kageyama. Melody retrieval with humming. In *Proc. ICMC 1993*, 1993.

[22] Anirudh Khanna, Bishwajeet Pandey, Kushagra Vashishta, Kartik Kalia, Bhale Pradeepkumar, and Teerath Das. A study of today's ai through chatbots and rediscovery of machine intelligence. *International Journal of u-and e-Service, Science and Technology*, 8(7):277–284, 2015.

[23] Diksha Khurana, Aditya Koli, Kiran Khatter, and Sukhdev Singh. Natural language processing: State of the art, current trends and challenges. *Multimedia tools and applications*, pages 1–32, 2022.

[24] Peter Knees, Markus Schedl, and Masataka Goto. Intelligent user interfaces for music discovery. *Transactions of the International Society for Music Information Retrieval*, 3(1), 2020.

[25] Anis Koubaa. Gpt-4 vs. gpt-3.5: A concise showdown. 2023.

[26] Pavel Kucherbaev, Alessandro Bozzon, and Geert-Jan Houben. Human-aided bots. *IEEE Internet Computing*, 22(6):36–43, 2018.

[27] Brian Langner, Stephan Vogel, and Alan W Black. Evaluating a dialog language generation system: comparing the mountain system to other nlg approaches. In *Eleventh Annual Conference of the International Speech Communication Association*, 2010.

[28] Maria Marietto, Rafael Aguiar, Gislene Barbosa, Wagner Botelho, Edson Pimentel, Robson Franca, and Vera Silva. Artificial intelligence markup language: A brief tutorial. *International Journal of Computer Science and Engineering Survey*, 04, 07 2013.

[29] György Molnár and Zoltán Szüts. The role of chatbots in formal education. In *2018 IEEE 16th International Symposium on Intelligent Systems and Informatics (SISY)*, pages 000197–000202. IEEE, 2018.

[30] Amrita Nair, Smriti Pillai, Ganga S Nair, and Anjali T. Emotion based music playlist recommendation system using interactive chatbota. In *2021 6th International Conference on Communication and Electronics Systems (ICCES)*, pages 1767–1772, 2021.

[31] Erica Naone. Software That Learns from Users, year = 2007, url = https://www.technologyreview.com/2007/11/30/98176/software-that-learns-from-users/., urldate = 2007-11-30.

[32] Matthew E Peters, Waleed Ammar, Chandra Bhagavatula, and Russell Power. Semi-supervised sequence tagging with bidirectional language models. *arXiv preprint arXiv:1705.00108*, 2017.

[33] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding with unsupervised learning. 2018.

[34] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

[35] Kiran Ramesh, Surya Ravishankaran, Abhishek Joshi, and K Chandrasekaran. A survey of design techniques for conversational agents. In *Information, Communication and Computing Technology: Second International Conference, ICICCT 2017, New Delhi, India, May 13, 2017, Revised Selected Papers*, pages 336–350. Springer, 2017.

[36] Peter J Rentfrow. The role of music in everyday life: Current directions in the social psychology of music. *Social and personality psychology compass*, 6(5):402–416, 2012.

[37] Igor André Pegoraro Santana, Fabio Pinhelli, Juliano Donini, Leonardo Catharin, Rafael Biazus Mangolin, Valéria Delisandra Feltrim, Marcos Aurélio Domingues, et al. Music4all: A new music database and its applications. In *2020 International Conference on Systems, Signals and Image Processing (IWSSIP)*, pages 399–404. IEEE, 2020.

[38] Igor André Pegoraro Santana, Fabio Pinhelli, Juliano Donini, Leonardo Gabiato Catharin, Rafael B. Mangolin, Yandre M. G. Costa, Valéria Delisandra Feltrim,

and Marcos Aurélio Domingues. Music4all: A new music database and its applications. *2020 International Conference on Systems, Signals and Image Processing (IWSSIP)*, pages 399–404, 2020.

[39] Jesus Savage, David A Rosenblueth, Mauricio Matamoros, Marco Negrete, Luis Contreras, Julio Cruz, Reynaldo Martell, Hugo Estrada, and Hiroyuki Okada. Semantic reasoning in service robots using expert systems. *Robotics and Autonomous Systems*, 114:77–92, 2019.

[40] Shivani Shivanand, KS Pavan Kamini, Monika Bai, Ranjana Ramesh, and Sumathi HR. Chatbot with music and movie recommendation based on mood.

[41] Heung-Yeung Shum, Xiao-dong He, and Di Li. From eliza to xiaoice: challenges and opportunities with social chatbots. *Frontiers of Information Technology & Electronic Engineering*, 19:10–26, 2018.

[42] Pankaj R Telang, Anup K Kalia, Maja Vukovic, Rahul Pandita, and Munindar P Singh. A conceptual framework for engineering chatbots. *IEEE Internet Computing*, 22(6):54–59, 2018.

[43] Alan M Turing. *Computing machinery and intelligence*. Springer, 2009.

[44] George Tzanetakis and Perry Cook. Musical genre classification of audio signals. *IEEE Transactions on speech and audio processing*, 10(5):293–302, 2002.

[45] Richard Wallace. The elements of aiml style. *Alice AI Foundation*, 139, 2003.

[46] Joseph Weizenbaum. Eliza—a computer program for the study of natural language communication between man and machine. *Commun. ACM*, 9(1):36–45, jan 1966.

[47] Sarah Wilson. The benefits of music for the brain. 2013.

[48] Yu Wu, Wei Wu, Chen Xing, Ming Zhou, and Zhoujun Li. Sequential matching network: A new architecture for multi-turn response selection in retrieval-based chatbots. *arXiv preprint arXiv:1612.01627*, 2016.

[49] Zhao Yan, Nan Duan, Junwei Bao, Peng Chen, Ming Zhou, Zhoujun Li, and Jianshe Zhou. Docchat: An information retrieval approach for chatbot engines using unstructured documents. In *Proceedings of the 54th Annual Meeting of*

the Association for Computational Linguistics (Volume 1: Long Papers), pages 516–525, 2016.

[50] M Tmáš ZEMČÍK. A brief history of chatbots. *DEStech Transactions on Computer Science and Engineering*, 10, 2019.

[51] Wen Zhang, Taketoshi Yoshida, and Xijin Tang. A comparative study of tf* idf, lsi and multi-words for text classification. *Expert systems with applications*, 38(3):2758–2765, 2011.

Appendix A

Additional Information

This section includes the confusion matrices of SVM A.1, KNN A.2, Random forest A.3 and Naive bays A.4 classifiers in the study.

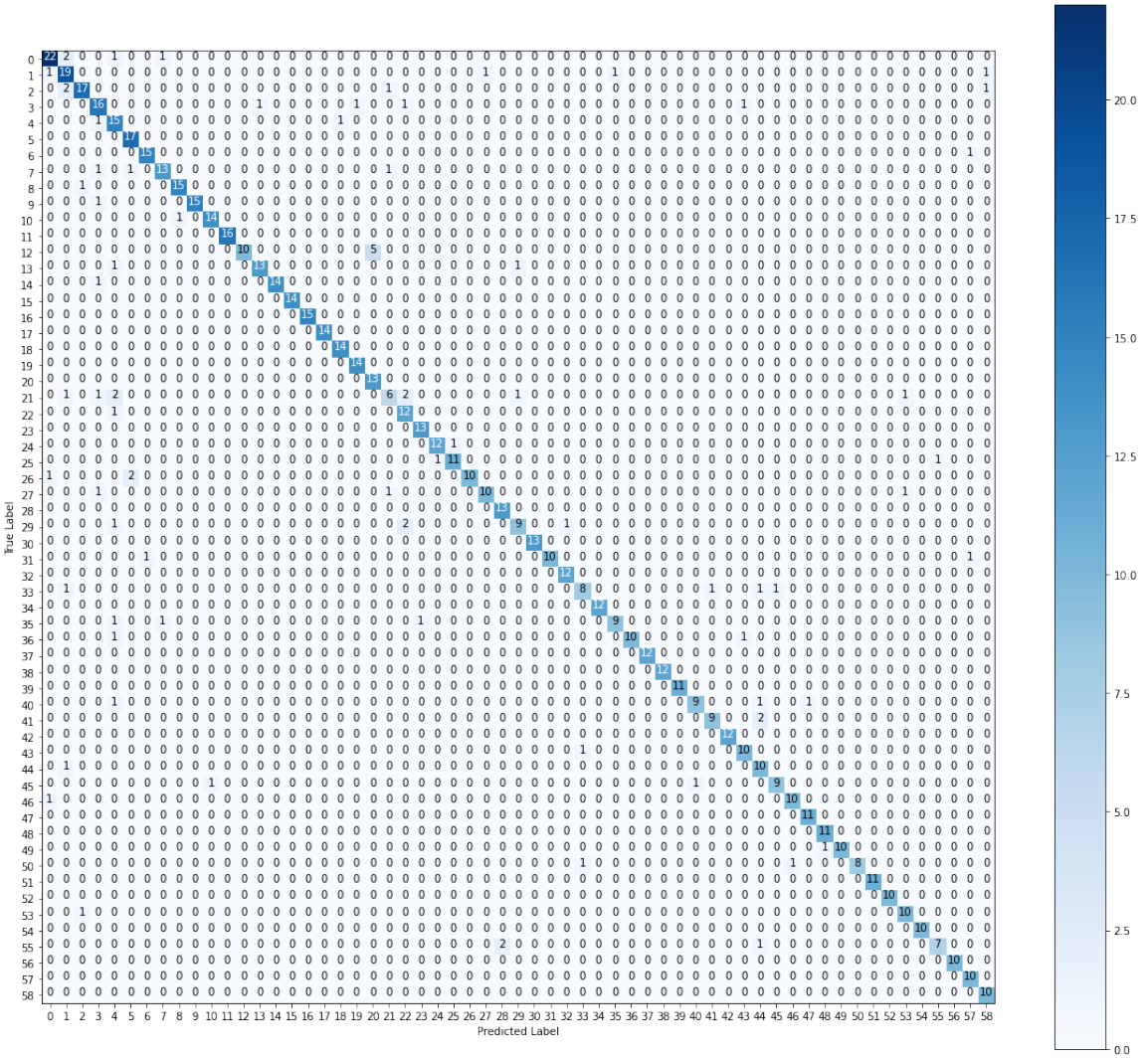


Figure A.1: The confusion matrix of SVM model

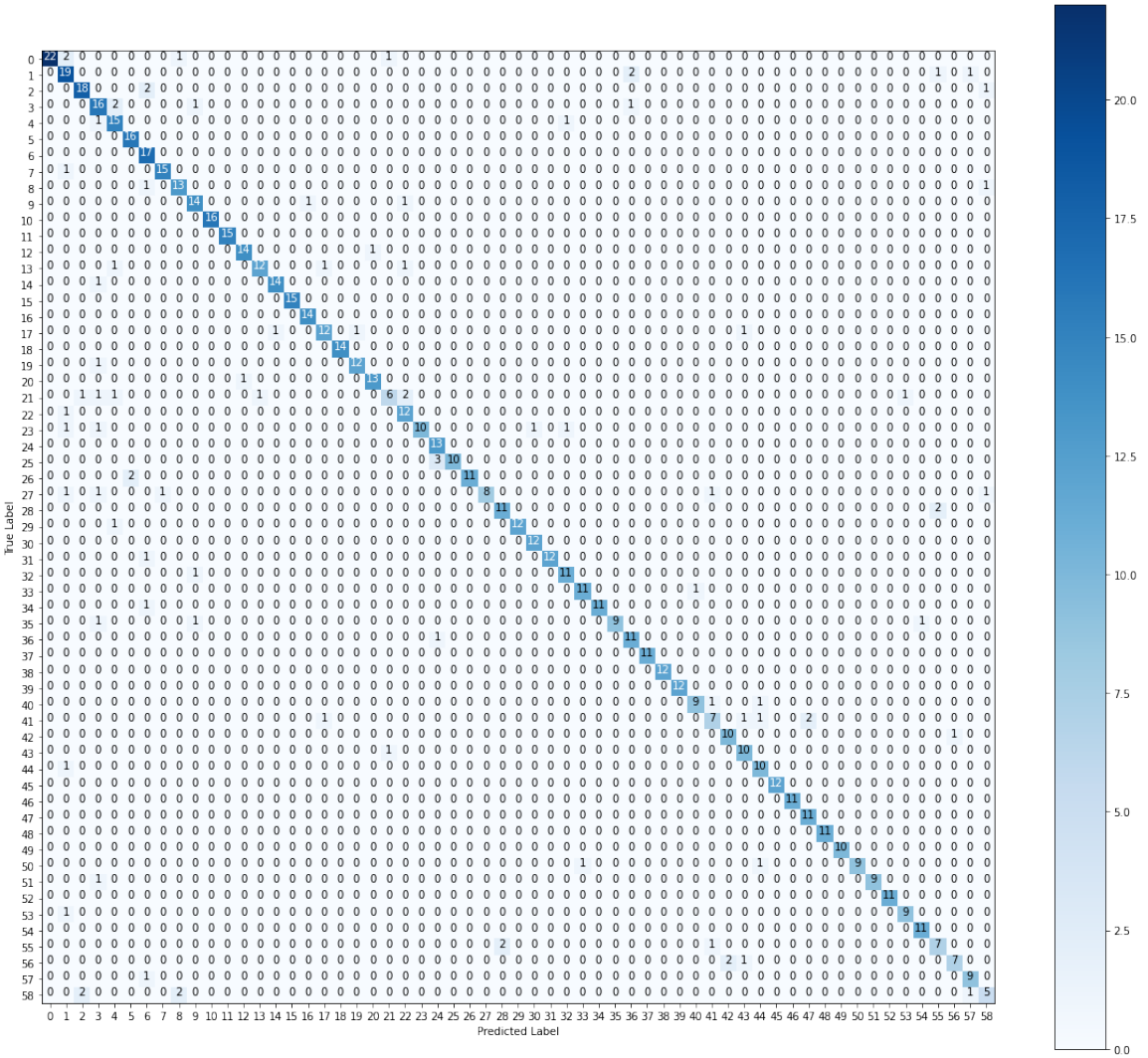


Figure A.2: The confusion matrix of KNN model

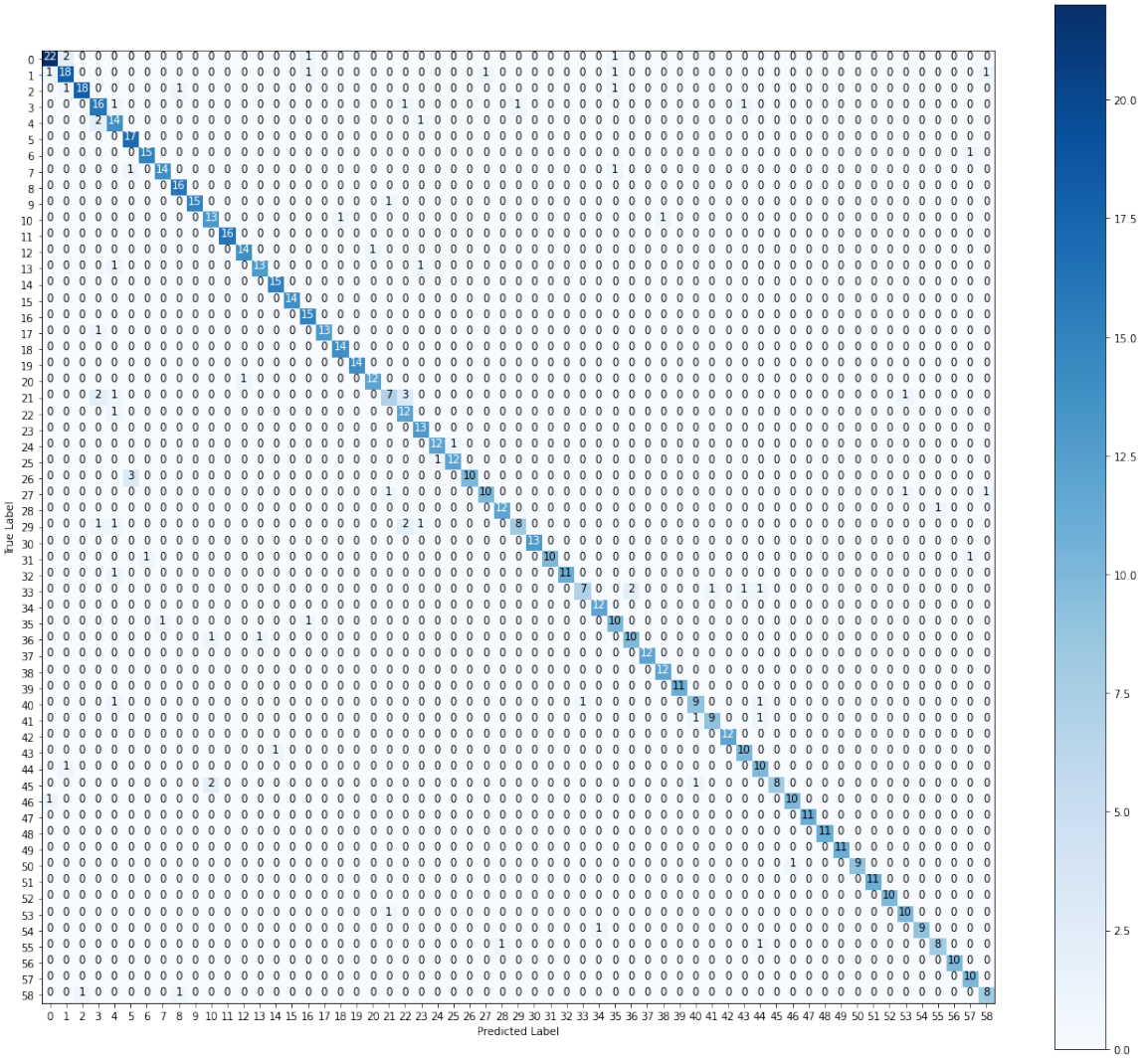


Figure A.3: The confusion matrix of random forest classifier

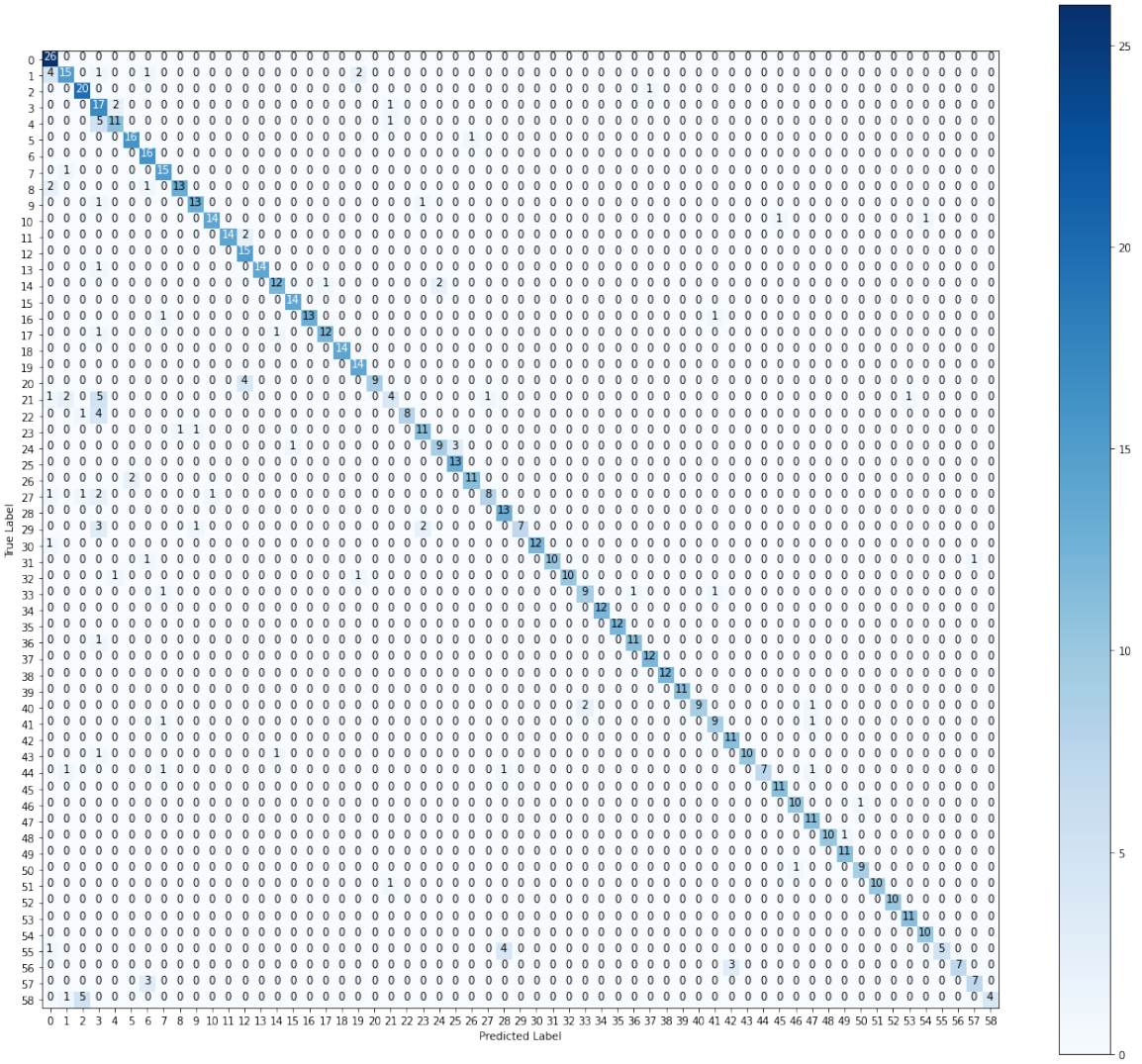


Figure A.4: The confusion matrix of Naive bay classifier