PREMIER UNIVERSITY, CHATTOGRAM
Department of Computer Science & Engineering

Final Year Thesis Report

On

Sentiment Analysis of Youtube Comments Using Machine Learning Algorithms

SUBMITTED BY

Name: Swarna Das

ID: 2104010202172

Name: Rahul Singha

ID: 1903610201798

Name: Kapil Das

ID: 1903610201793

In partial fulfillment for the degree of
Bachelor of Science in Computer Science & Engineering
under the Supervision of

Md. Hasan
Lecturer
Department of Computer Science & Engineering
Premier University, Chattogram

January, 2024

Author's Declaration of Originality

We hereby declare that the thesis work entitled **"Sentiment Analysis of Youtube Comments Using Machine Learning Algorithms"** submitted to the Premier University, is a record of an original work done by us under the guidance of Mr.Md. Hasan, Lecturer, Department of Computer Science & Engineering, Premier University, Chattogram and this work is submitted for fulfillment of the degree of Bachelor Science in Computer Science & Engineering. We can assure that the result of this thesis has not been submitted to any other university.

Swarna Das
ID: 2104010202172

Rahul Singha
ID: 1903610201798

Kapil Das
ID: 1903610201793

CERTIFICATION

The thesis entitled **"Sentiment Analysis of Youtube Comments Using Machine Learning Algorithms"** submitted by, Swarna Das, ID: 2104010202172, Rahul Singha, ID: 1903610201798, Kapil Das, ID: 1903610201793 has been accepted as satisfactory in partial fulfillment of the degree of Bachelor Science in Computer Science & Engineering (CSE) to be awarded by Premier University, Chattogram.

_____

Dr. Shahid Md. Asif Iqbal
Chairman
Department of Computer Science & Engineering
Premier University, Chattogram

_____

Md. Hasan
Lecturer
Department of Computer Science & Engineering
Premier University, Chattogram

It is my genuine gratefulness and warmest regard that
I dedicate this work to my beloved
Father and Mother

Table of Contents

List of Figures

List of Tables

Abstract

The extensive user engagement on YouTube leads to a flood of comments, creating challenges for content creators who aim to understand audience sentiment. Previous studies have mainly concentrated on distinguishing between positive and negative sentiments in comments, occasionally incorporating a neutral category. This study involves extracting comments from YouTube and classifying them into positive, negative, interrogative, corrective, imperative, and neutral categories. To handle the inherent noise in textual data, including punctuation marks, stopwords, and spelling errors, we implement a preprocessing phase. In this phase, unnecessary elements are eliminated, and crucial features are selected using the TF-IDF vectorizer. In employing six machine learning algorithms—Naive Bayes (NB), Logistic Regression (LR), Decision Tree (DT), Random Forest (RF), K-Nearest Neighbors (KNN), and Support Vector Machine (SVM)—the SVM model demonstrated outstanding accuracy, reaching 94%. This outcome underscores the effectiveness of SVM in precisely classifying sentiments within YouTube comments.

Keywords: Sentiment Analysis, Machine Learning, Support Vector Machine (SVM).

CHAPTER 1

INTRODUCTION

1.1 Introduction

YouTube is a popular social media platform where thousands, or even millions, of people interact by uploading videos, liking, sharing, and commenting on them. By analyzing the comments, the content creator can gain valuable insights into the sentiment of their audience. However, it is often impractical to manually review all the comments, which may include questions, corrections, or irrelevant remarks. To address this issue, we have created a dataset of YouTube comments and categorized them based on sentiment, including positive, negative, interrogative, corrective, imperative, and neutral sentiment. Additionally, we preprocess the corpus to rectify issues such as spelling errors, emojis, and unformatted text. In our analysis, we employed supervised machine learning algorithms such as Naive Bayes, Random Forest, Decision Tree, Logistic Regression, K-Nearest Neighbors, and Support Vector Machine (SVM) to predict the sentiment of the comments. Subsequently, we comprehensively evaluate the performance of these algorithms using a range of performance measurement matrices which includes accuracy, precision, recall, F1-score, ROC curve, and confusion matrix.

1.2 Motivation

The exponential growth of user interactions on YouTube and the overwhelming volume of comments poses a significant challenge for content creators. The manual process of reviewing and understanding audience sentiment, addressing queries, and incorporating suggestions to enhance content becomes increasingly difficult. Sometimes, the overwhelming number of comments results in the oversight of certain queries and comments. This has motivated us to develop tools that can analyze sentiments in YouTube comments, providing content creators with the ability to categorize and comprehend the expressed sentiment.

1.3 Application

This research provides content creators with valuable insights into audience sentiments, queries, and corrective feedback. Its aim is to assist creators in crafting higher-quality content and improving audience engagement by comprehending audience sentiments and addressing their requirements.

CHAPTER 2

LITERATURE REVIEW

In recent years, several research has been done in the field of sentiment analysis of YouTube comments. Ritika Singh and Ayushka Tiwari presented a model for "YouTube Comments Sentiment Analysis" [1]. The research focuses on sentiment analysis of YouTube comments related to popular topics using machine learning algorithms. The study demonstrates that analyzing sentiments helps identify trends, seasonality, and forecasts, providing insights into the influence of real-world events on public sentiments. Six machine learning algorithms, including Naïve-Bayes, Support Vector Machine, Logistic Regression, Decision Tree, K-Nearest Neighbor, and Random Forest, were implemented and evaluated on a dataset. Results indicate that SVM performs the best, followed by Naïve Bayes, and the use of uni-grams, bi-grams, and tri-grams features significantly contributes to achieving high accuracy scores. The study emphasizes the importance of sentiment analysis in identifying research papers and presents improvements in classification results compared to the base system.

Abdel-Karim Al-Tamimi, Ali Shatnawi, and Esraa Bani-Issa presented a model for "Arabic Sentiment Analysis of YouTube Comments" [2]. The research focuses on Arabic sentiment analysis using 5,986 manually annotated Arabic YouTube comments from top-viewed videos in the Arabic world. Several supervised classification experiments were conducted, comparing SVM-RBF, Bernoulli NB, and KNN classifiers. SVM-RBF achieved the highest f-measure of 88.8% with an unbalanced 2-class (positive and negative) normalized dataset, containing both related and unrelated comments. The study highlights the positive impact of preprocessing and normalization steps on classification outcomes, particularly with larger datasets.

Nafis Irtiza Tripto and Mohammed Eunus Ali presented a model for "Detecting Multilabel Sentiment and Emotions from Bangla YouTube Comments" [3]. The paper introduces techniques for sentiment identification and emotion extraction from Bangla texts. Deep learning models are employed for three-class (positive, negative, neutral) and five-class (strongly positive, positive, neutral, negative, strongly negative) sentiment labeling, as well as extracting one of the six basic emotions (anger, disgust, fear, joy, sadness, surprise) from Bangla sentences. Evaluation on a new dataset of Bangla, English, and Romanized Bangla comments from various YouTube videos demonstrates the model's performance, achieving 65.97% and 54.24% accuracy in three and five labels sentiment, respectively. Additionally, NB and SVM were effective in extracting emotions, showcasing superior performance for domain and language-specific texts.

Rhitabrat Pokharel and Dixit Bhatta presented a model for "Classifying YouTube Comments Based on Sentiment and Type of Sentence" [4]. In their research, the objective is to aid YouTubers in interacting with relevant comments by extracting and classifying raw comments based on both sentiment and sentence types. They assessed various statistical measures and machine learning models, employing a dataset of 10,000 comments sourced from various tutorial videos through the scraping of YouTube comments. The comments were manually categorized into six classes, and from their experiments, Logistic Regression emerged as the most efficient model.

Risky Novendri, Annisa Syafarani Callista, Danny Naufal Pratama, Chika Enggar Puspita presented a model for "Sentiment Analysis of YouTube Movie Trailer Comments Using Naïve Bayes" [5]. The study focuses on analyzing sentiments in YouTube comments for Money Heist Season 4, using a dataset of 998 comments obtained through crawling. The authors employed TF-IDF for feature extraction and applied the Naive Bayes algorithm, achieving an accuracy of 81%, precision of 74.83%, and recall of 75.22%.

Tooba Tehreem and Hira Tahir presented a model for "Sentiment Analysis for YouTube Comments in Roman Urdu" [6]. The study focuses on YouTube comments related to Pakistani dramas and TV shows, using a dataset with multi-class classification (positive, negative, neutral). Comparative analysis of five supervised learning algorithms, including linear regression, SVM, KNN, Multi-layer Perceptron, and Naive-Bayes, was conducted. SVM demonstrated the highest accuracy at 64%, outperforming the other algorithms.

Mohammad Aufar, Rachmadita Andreswari, and Dita Pramesti presented a model for "Sentiment Analysis on YouTube Social Media Using Decision Tree and Random Forest Algorithm: A Case Study" [7]. The research employs VADER for sentiment labeling on 2000 comments from the YouTube Nokia's Mobile Channel. Despite challenges like

mispelled and slang comments, Decision Tree outperforms Random Forest with 89.4% and 88.2% accuracy, respectively. Both algorithms use Gini Index for splitting criteria, showing stability in handling noisy data and providing reliable predictions.

Shoaib Nawaz, Muhammad Rizwan, Samina Yasin, Mehtab Ahmed, and Umar Farooq presented a model for "Multi-Class Classification of YouTube Comments using Machine Learning" [8]. Their study aims to assist subscribers in finding the best-experienced ideas by classifying comments into categories like Experience-positive, Experience-negative, Warning, Suggestions, Questions, and Praise. The proposed SVM-based model achieved an 89.18% accuracy and 0.89 F1 score, demonstrating its effectiveness in qualitative comment analysis for both users and content creators.

Fiktor Imanuel Tanesab, Irwan Sembiring, and Hindriyanto Dwi Purnomo introduced a Sentiment Analysis Model focusing on YouTube comments related to the performance of Ahok as a governor [9]. Using Support Vector Machine (SVM), the research categorized opinions into positive, neutral, and negative classes, based on 1000 recorded data samples. The process involved data preprocessing, tokenization, cleansing, and filtering, with Lexicon-Based method determining sentiment class percentages. Results, including Lexicon-Based and Confusion Matrix, demonstrated an accuracy of 84%, precision of 91%, recall of 80%, TP rate of 91.1%, and TN rate of 44.8%.

Muhammad Alkaff, Andreyan Rizky Baskara, and Yohanes Hendro Wicaksono developed a model for sentiment analysis on Indonesian movie trailers' YouTube comments [10]. They categorized comments into action, romance, comedy, and horror genres, employing the Delta TF-IDF method and various classification approaches. Results indicated Logistic Regression and Naïve Bayes excelling in specific genres, while SVM demonstrated good performance across various genres, as evaluated through Stratified K-Fold cross-validation (K=10).

CHAPTER 3

METHODOLOGY

In this chapter, we will briefly explain our approach to sentiment analysis of YouTube comments using machine learning algorithms. The block diagram of the overall methodology is depicted in Figure 3.1.

At first, we created a dataset consisting of YouTube comments and manually labeled our dataset. The application of Python programming was instrumental in executing our methodology. The labeled dataset allowed us to implement machine learning algorithms for sentiment analysis.

After importing the dataset, we performed preprocessing using the NLTK package to clean and prepare the data for analysis. The NLTK (Natural Language Toolkit) provided tools for tasks such as tokenization, removing stop words, and stemming or lemmatization. Subsequently, label encoding was applied to categorize the data for machine learning algorithms. For feature extraction, we employed the TF-IDF (Term Frequency-Inverse Document Frequency) vectorizer. Following data preparation, the dataset was divided into a 60-20-20 ratio, where 60% of the data was allocated for training, 20% for testing, and 20% for validation purposes. After that, we implemented all machine learning algorithms and measured their performance by using different performance measurement matrices.

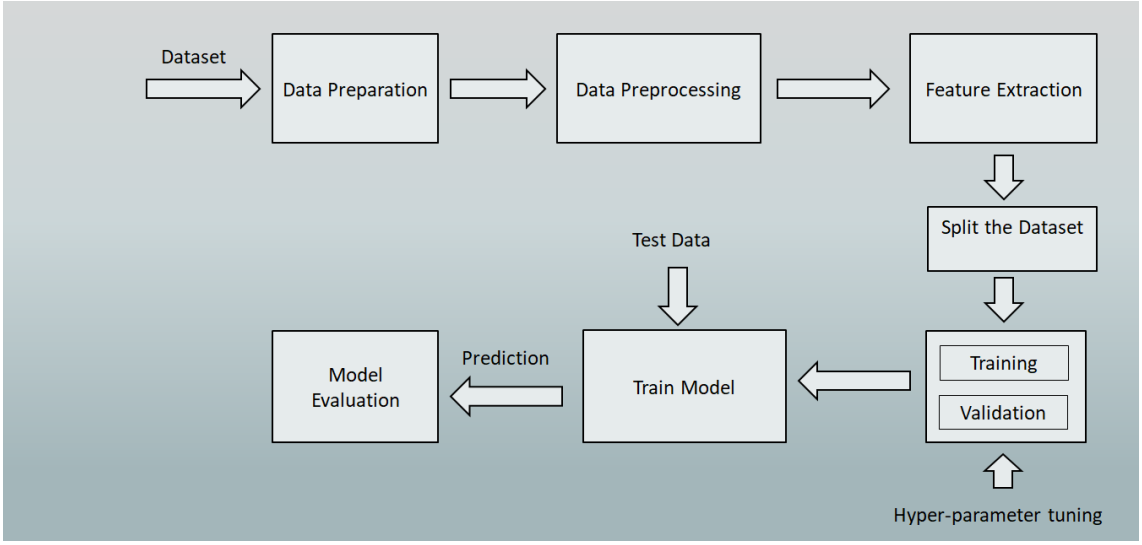

Figure 3.1. Workflow of the proposed sentiment analysis of youtube comments

3.1 Data Collection

The dataset consists of 2000 YouTube comments gathered from various tutorials. These comments cover a diverse range of sentiments, and they have been categorized into different sentiment classes:

1. Positive Sentiment (Appraisal): Comments expressing positive feedback and appreciation.

2. Negative Sentiment: Comments indicating challenges or difficulties in understanding the content.

3. Interrogative Sentiment (Query): Comments posing questions seeking clarification.

4. Corrective Sentiment: Comments suggesting improvements or pointing out mistakes.

5. Imperative Sentiment: Comments conveying requests, commands, or expectations.

6. Neutral Sentiment: Comments that do not exhibit a specific sentiment or are considered irrelevant.

Below are the number of comments in each class:

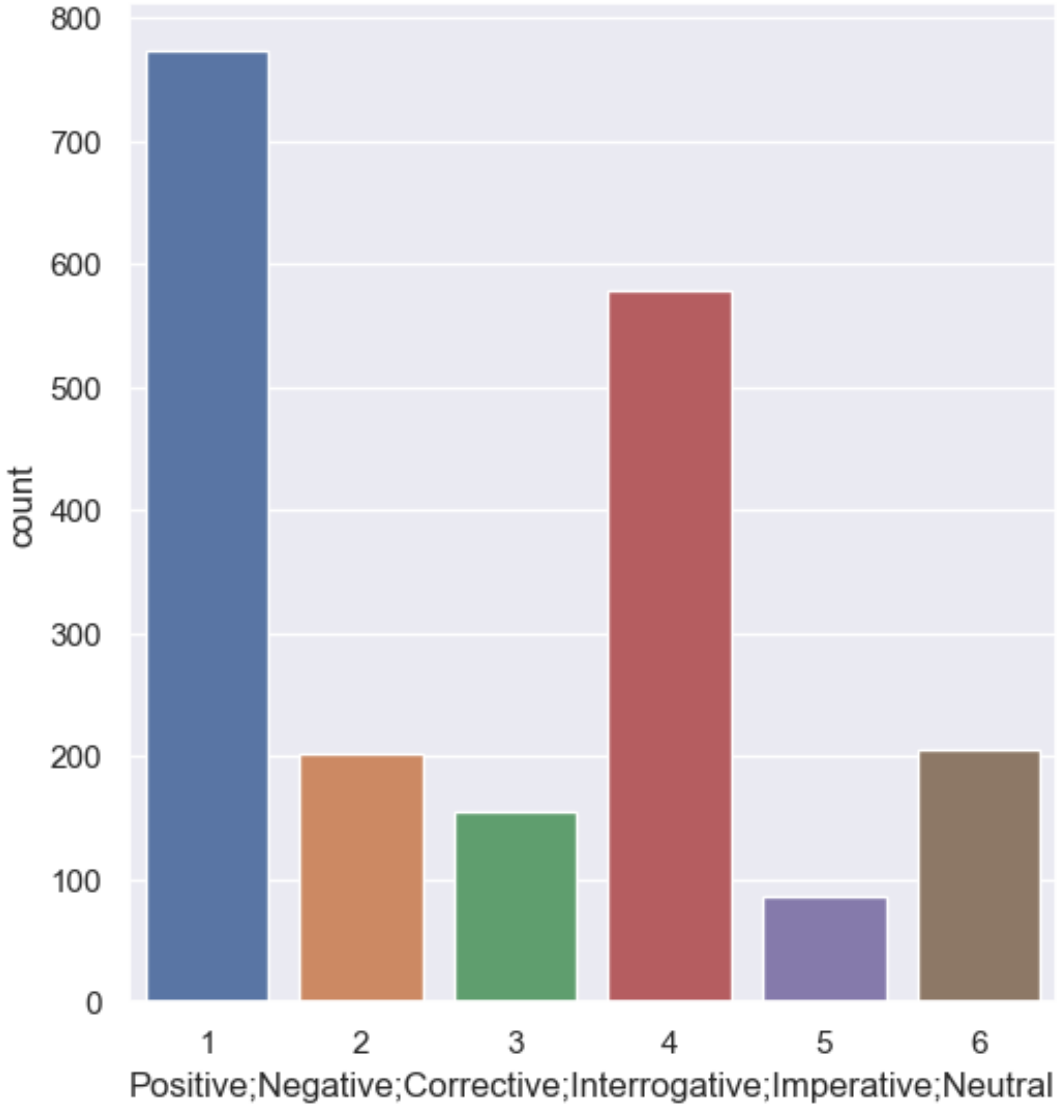

Figure 3.2. Number of comments in each class

3.2 Preprocessing

After importing the dataset, cleaning the dataset is crucial as it influences the performance of the classifiers. Below are the steps that we performed in the preprocessing phase:

1) **Lowercasing:** Initially, we converted the text within the dataset to lowercase.

2) **Stopwords Removal:** As stopwords are the most frequently used words and they do not carry any significant meaning, so we removed stopwords from our dataset. However, some stopwords, such as "not", "how", were retained as they contribute to the expression

of sentiment in comments. Below are the stopwords that were retained for each class:

Table 3.1. Retained Stopwords in Different Classes

Class	**Retained Stopwords**
Negative	no, not
Interrogative	how, what, which, who, whom, why, do, is, does, are, was, were, will, am, could, would, should, can, did, had, have
Imperative	could,would,should,can

3) **Removing Punctuation:** To refine the dataset and enhance accuracy, punctuation marks were removed, except the question mark ("?") which is instrumental in identifying interrogative sentiment.

4) **Removing Multiple Space:** Occasionally, when commenting on social media platforms, multiple spaces are inserted. These multiple spaces are unnecessary so we removed them for better data consistency.

5) **Tokenization, Lemmatization and POS Tagger:** To standardize and simplify words, we tokenized the text, identified the parts of speech for each word, and then applied lemmatization based on the identified parts of speech which makes it easier for algorithm to intrepret text in the context of sentiment analysis.

6) **Removing Emoji:** All emojis were replaced with the empty string.

7) **Removing URLs:** As URLs are not so important in terms of sentiment analysis, they were also removed from the dataset.

3.3 Feature Extraction

TF-IDF is used for extracting features in our dataset. It reflects the significance of words in a documents by measuring how frequently they are appearing in a document. When a word is frequently present in a specific document, its importance or significance within that document is higher. However, if the same word is found across multiple documents, its significance becomes comparatively lower. TF-IDF is a combination of TF (Term Frequncy) and IDF (Inverse Document Frequency).

TF tells us how often a specific word appears in a single document. It's calculated by dividing the number of times the word shows up in that document by the total number

of words in that document. This helps us understand the word's importance within that document. Each document has its unique Term Frequency.

$$\text{TF}(t, d) = \frac{\text{Total number of terms in document } d}{\text{Number of times term } t \text{ appears in document } d}$$

IDF measures how unique or rare a word is across all the documents in the dataset. If a word is common and appears in many documents, its IDF value will be lower. On the other hand, if a word is rare and appears in fewer documents, its IDF value will be higher. This helps to emphasize the importance of terms that are distinctive and carry more information when they appear.

$$\text{IDF(t)} = \log(\frac{\text{N}}{\text{DF(t)}})$$

Where:

- N is the total number of documents in the collection.
- DF(t) is the number of documents containing term t.

After calculating the Term Frequency (TF) and Inverse Document Frequency (IDF) scores, both are multiplied to determine the TF-IDF score.

$$\text{TF-IDF}(t, d) = \text{TF}(t, d) \times \text{IDF}(t)$$

3.4 CLassification Model

Machine learning involves a two-step process: the learning phase followed by the prediction phase. In the classification model, the initial step is training the model with labeled data, which is part of the learning phase. Then, the trained model is utilized to predict the labels of the test or unseen data, which is part of the prediction phase. Below are the descriptions of the classification models that are being used:

3.4.1 Naive Bayes

Naive Bayes, a classification algorithm based on Bayes' Theorem, excels in categorizing data into predefined classes. Its strength lies in assuming independence among features given the class, simplifying probability calculations and making it surprisingly effective in various applications. As part of generative learning algorithms, naive bayes constructs models of input distribution for each class. Naive Bayes finds widespread use in tasks like text classification, spam filtering, and sentiment analysis, where the independence assumption holds reasonably well. Notably, it efficiently handles large datasets, making it suitable for real-time applications, and can be enhanced with kernel density estimation for complex data distributions.Despite its advantages, naive bayes comes with certain considerations.The algorithm is sensitive to irrelevant features, and accuracy may be impacted by correlated attributes.

The Bayes Theorem is:

$$P(A|B) = \frac{P(B|A).P(A)}{P(B)}$$

Where,

- **P(A|B) is the posterior probability** ,expressing the probability of event A occurring given that event B has already occurred.
- **P(B|A) is the likelihood probability** expressing the probability of event B occurring given that event A has already occurred.
- **P(A) is the prior probability**, expressing the likelihood of event A happening independently.
- **P(B) is the marginal probability** that indicates the probability of event B occurring without any specific conditions.

3.4.2 Logistic Regression

Logistic regression is a supervised machine learning algorithm which decides the probability of an event belonging to one of two classes (yes or no, 0 or 1, true or false). It uses the logistic function, which is also known as sigmoid function, is an S-shaped curve which is used for transforming a linear combination of input features into a probability score between 0 and 1. To separate classes a threshold value is defined, values greater than the predefined threshold value are categorized as class 1 and the rest are categorized as class 0. One of the practical example of it is detecting spam email, considering the threshold value 0.5, the output of sigmoid function greater than 0.5 indicates the mail is spam and if less than 0.5 then the mail is not spam. The sigmoid function is referred to as an activation

function for logistic regression and is defined as:

$$f(x) = \frac{1}{1 + e^{-x}}$$

where,

- e = base of natural logarithms.

The following equation represents logistic regression:

$$y = \frac{e^{(b_0 + b_1 X)}}{1 + e^{(b_0 + b_1 X)}}$$

here,

- x = input value
- y = predicted output
- b0 = bias or intercept term
- b1 = coefficient for input (x)

Like linear regression, it uses weight or coefficient values to predict an output. However, the key difference between linear regression and logistic regression is linear regression predicts continuous value whereas logistic regression predicts value between 0 and 1. The concept is explained in the below figure:

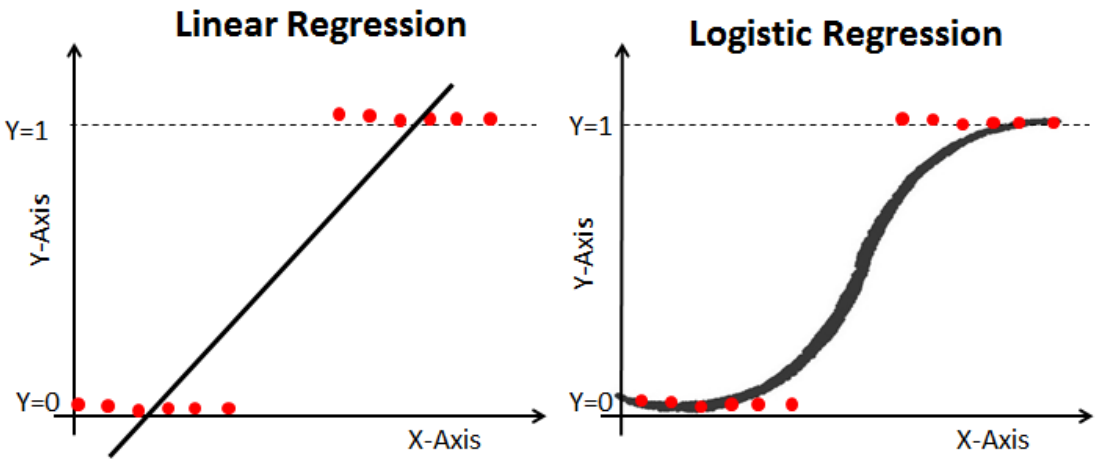

Figure 3.3. Linear Regression vs. Logistic Regression Output

3.4.3 Decision Tree

3.4.3.1 Introduction

A decision tree is a versatile supervised machine learning algorithm employed for both classification and regression tasks. Its primary purpose is to categorize data into distinct classes in the case of classification or predict continuous variables for regression. The decision tree operates by partitioning the dataset into subsets based on various attributes, with the selection of the optimal attribute for each split determined through metrics like information gain. This tree-like structure is constructed using algorithms such as CART (Classification and Regression Tree), where nodes represent decisions based on specific data features, ultimately leading to conclusive outcomes at the leaf nodes. Decision trees are valued for their interpretability, ease of implementation, and ability to handle both categorical and numerical data. However, they are susceptible to overfitting, especially with complex datasets, and may not perform optimally when faced with highly correlated features. The diagram below illustrates the general structure of a decision tree:

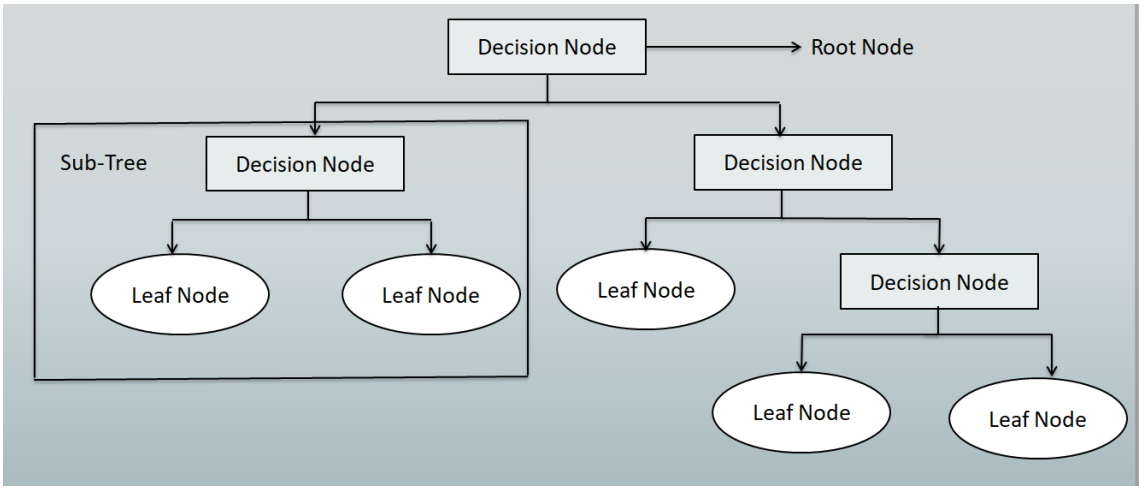

Figure 3.4. Structure of Decision Tree

3.4.3.2 Terminologies

Below are the key terminologies that are associated with decision tree:

- **Root Node:** The root node is the starting point of the decision tree, representing the entire dataset. It's the initial decision point from which the tree branches out based on specific conditions or features.
- **Decision Node/Internal Node:** In the decision tree, decision nodes, or internal

nodes, play a crucial role as points where the algorithm evaluates specific features or attributes to make decisions. These nodes serve as forks in the tree, guiding the path to different branches based on the conditions set by the features.

- **Leaf Node/Terminal Node:** The leaf nodes mark the endpoints of the decision tree where no additional splitting occurs. These nodes signify the ultimate outcomes or decisions based on the assessed conditions and features throughout the tree.
- **Branch:** A path connecting nodes in the tree. It represents the outcome of a decision.
- **Splitting:** The process of dividing a node into two or more sub-nodes based on a certain condition.
- **Parent Node:** A node that is divided into child nodes during the splitting process.
- **Child Node:** Nodes resulting from the splitting of a parent node.
- **Attribute/Feature:** The characteristics or variables used to make decisions at decision nodes.
- **Pruning:** Involves eliminating branches from the decision tree that contribute minimal predictive value, enhancing the tree's efficiency.
- **Entropy:** Entropy is a metric used to quantify disorder or impurity within a dataset. In the context of information gain, it provides a measure of uncertainty. The formula for calculating entropy is:

$$Entropy(S) = -P(yes)log_2P(yes) - P(no)log_2P(no)$$

  where,
  - S is the total sample space.
  - P(yes) is a probability of yes.
  - P(no) is a probability of no.
- **Information Gain:** Information Gain, a crucial aspect in a decision tree, measures the entropy change when a node divides training instances. It indicates the extent to which a feature provides valuable information about a class. The decision tree algorithm prioritizes nodes with greater information gain during the splitting process, aiming to enhance predictive capabilities by maximizing the effectiveness of each node. Mathematically, information gain (IG) is calculated as follows:

$$InformationGain(IG) = Entropy(S) - [WeightedAvg \times Entropy(eachfeature)]$$

- **Gini Index:** The Gini index, employed in the CART (Classification and Regression Tree) algorithm, serves as a metric for impurity or purity when constructing decision trees. A lower Gini index is favored over a higher one, and it is particularly used for creating binary splits in the decision tree. The Gini index is calculated using the

formula:

$$GiniIndex = 1 - \sum_{j} P_j^2$$

3.4.4 Random Forest

3.4.4.1 Introduction

Random Forest is a versatile and robust machine learning algorithm employed for both classification and regression tasks. Its strength lies in its ability to enhance predictive accuracy while mitigating the shortcomings of individual decision trees. The algorithm operates by constructing an ensemble of multiple decision trees, each trained on a distinct subset of the dataset. Through the random selection of subsets, random forest achieves resilience and guards against overfitting, making it particularly effective for complex data. In classification tasks, the final prediction is determined by majority voting among the individual trees, while in regression tasks, it involves averaging their outputs. Widely favored for its capacity to handle high-dimensional data and improve overall generalization, random forest does come with the trade-off of increased complexity, resulting in longer training times. Nevertheless, its substantial performance enhancement makes it a popular and reliable choice in various machine learning applications. The below diagram explains the working of the random forest algorithm:

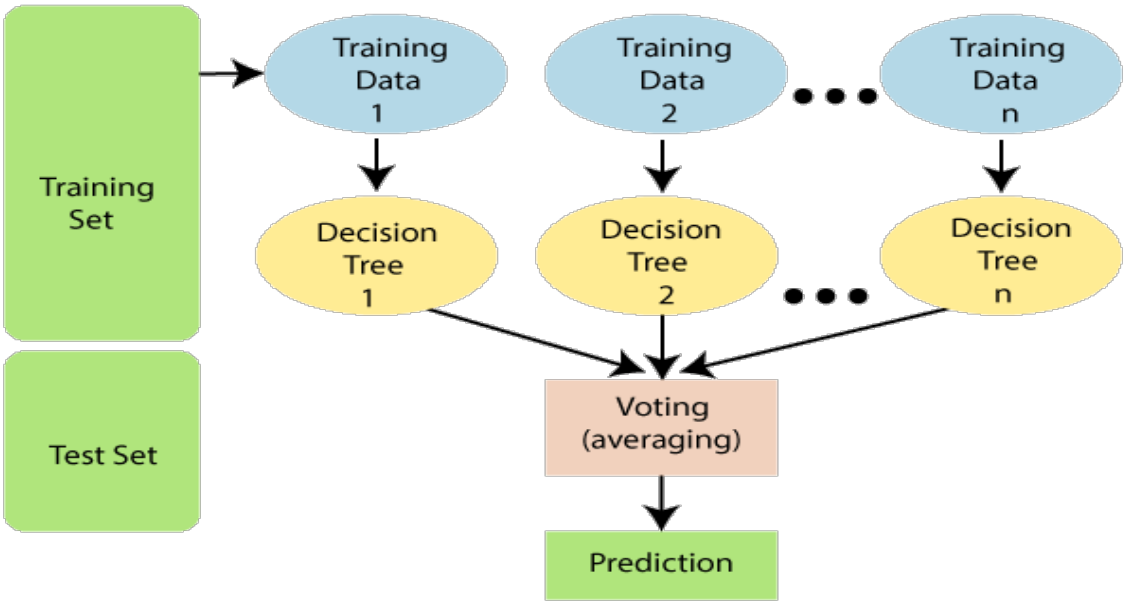

Figure 3.5. Visual representation of how Random Forest works

3.4.4.2 Hyperparameters

There are several hyperparameters that can be used for improving the model's performance. Here are some key hyperparameters:

- **n_estimators:** Determines the number of decision trees in the forest. Increasing the number of trees generally improve the performance.
- **max_depth:** It controls the maximum depth of each decision in the forest. Allowing greater depth enables capturing complex details but there is a risk of overfitting. On the other hand, limiting depth guards against overfitting but halts capturing complex details. So finding the right balance is the key.
- **min_samples_split:** Determines the minimum number of samples needed to split an internal node. Setting higher values helps to avoid splitting nodes with too few samples, contributing to preventing overfitting.
- **min_samples_leaf:** It sets the minimum samples required for a leaf node. It helps to control overfitting by defining the minimum sample size in each leaf.
- **max_features:** Defines the maximum number of features considered when splitting a node.
- **Criterion:** It is used for splitting nodes. "gini" is the default and is used for Gini impurity. "entropy" can be used for information gain.

3.4.5 K-Nearest Neighbors

KNN is one of the most popular supervised machine learning algorithm as it is simple and easier to implement, robust to the noisy training data and can be used for both classification and regression problem. It is also known as a lazy learner algorithm as it stores the entire dataset and calculates the Euclidean distance between the input data point and all the training example. The formula for calculating Euclidean distance is:

$$d(x, y) = \sqrt{\sum_{i=1}^{N}(y_i - x_i)^2}$$

Where,

x,y = two points in Euclidean n-space.

y_i, x_i = Euclidean vectors, starting from the origin of the space (initial point).

n = n-space.

After calculating the distance, the algorithm checks the nearest neighbors of the input data based on the value of K (K nearest neighbors). For classification, KNN identifies the predominant class labels within the K nearest neighbors of a data point. In regression, it predicts the value of the data point by averaging the target values of its K nearest neighbors. One of the major drawback of this algorithm is to select the value of K, which maybe complex sometimes. The most preferable value for K is K=5. Low values for K can be noisy and lead to the outliers in the model and large value for K can cause under-fitting.

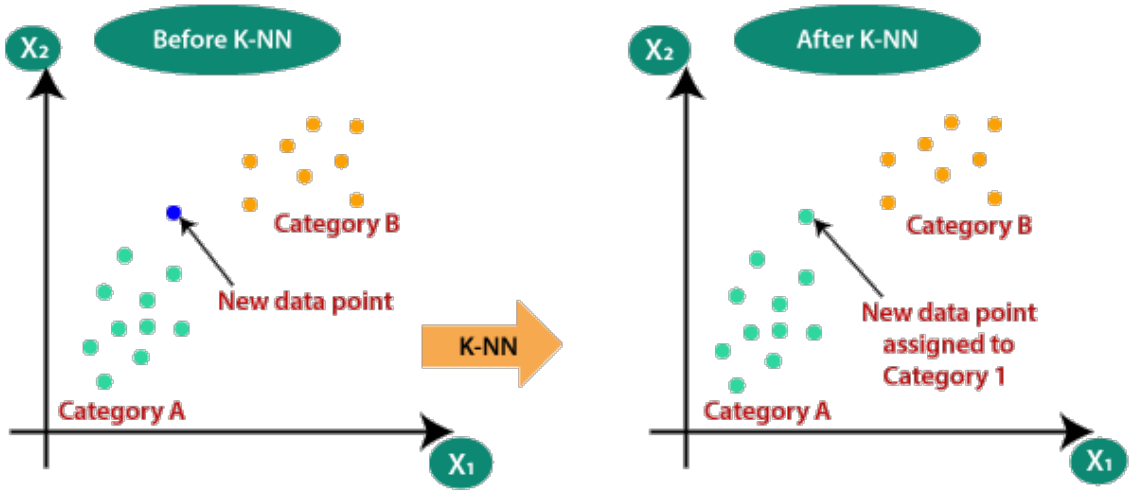

Figure 3.6. Assignment of new data point to a category using KNN

3.4.6 Support Vector Machine

SVM stands out as a widely-used supervised machine learning algorithm which can be used for both classification and regression tasks. It encompasses two categories: Linear SVM and Non-linear SVM. The former assumes a linear relationship between features and classes, while the latter utilizes kernel functions to handle non-linear relationships. Much like logistic regression, SVM's primary goal is to identify the optimal hyperplane that effectively separates data points of different classes in feature space. In SVM, this hyperplane functions as the decision boundary. The algorithm identifies support vectors, the points closest to the hyperplane from each class, and measures the distance between these support vectors and the hyperplane, termed as the margin. SVM's objective is to maximize this margin, leading to the determination of the optimal hyperplane. In contrast to logistic regression, which deploys a sigmoid function to create an S-shaped decision boundary and predict probabilities, SVM concentrates on maximizing the separation

between classes through the strategic placement of the hyperplane and margin.

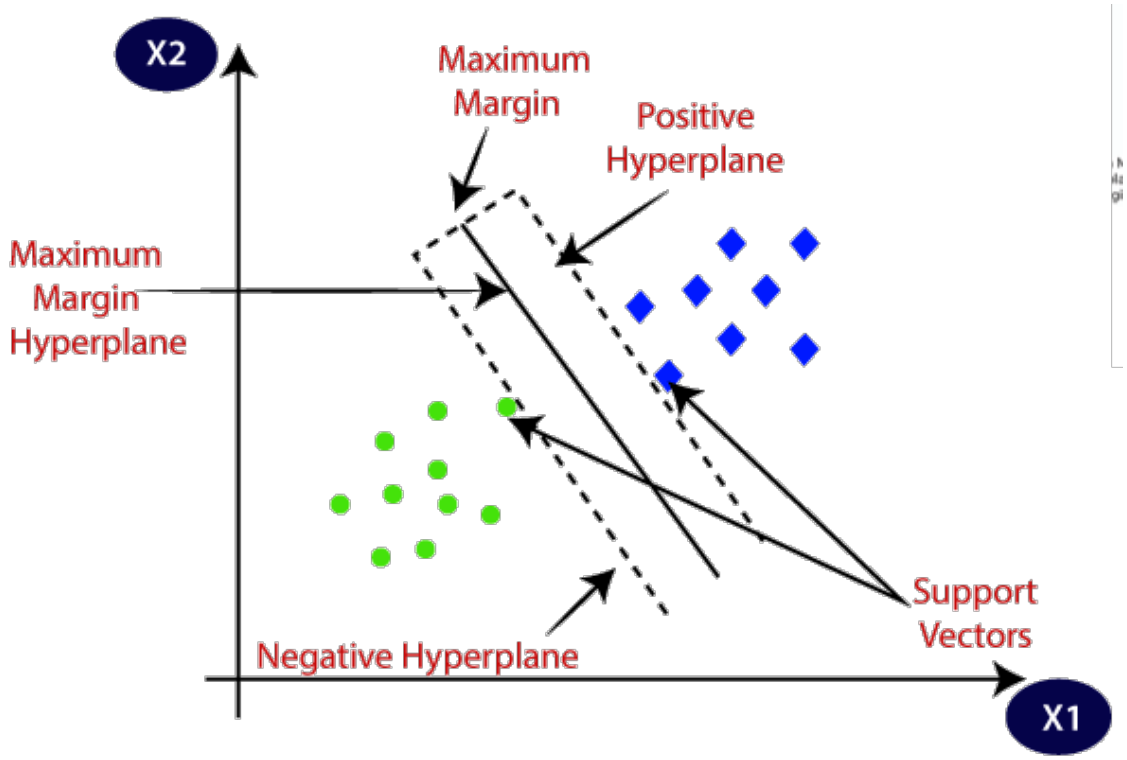

Figure 3.7. Support Vector Machine

3.5 Performance Evaluation

Performance evaluation involves measuring the effectiveness and accuracy of a model in making predictions. It helps to understand how well a model generalizes to new, unseen data. There are several metrics used for performance evaluation, below are the explanation for those:

Confusion matrix: The confusion matrix provides detailed evaluation of a classification-based machine learning model. It breaks down the model's predictions into accurate classifications (true positives and true negatives) and where it got confused (false positives and false negatives). This in-depth analysis offers valuable insights into how well the model is performing overall and highlights particular areas that may require enhancements.

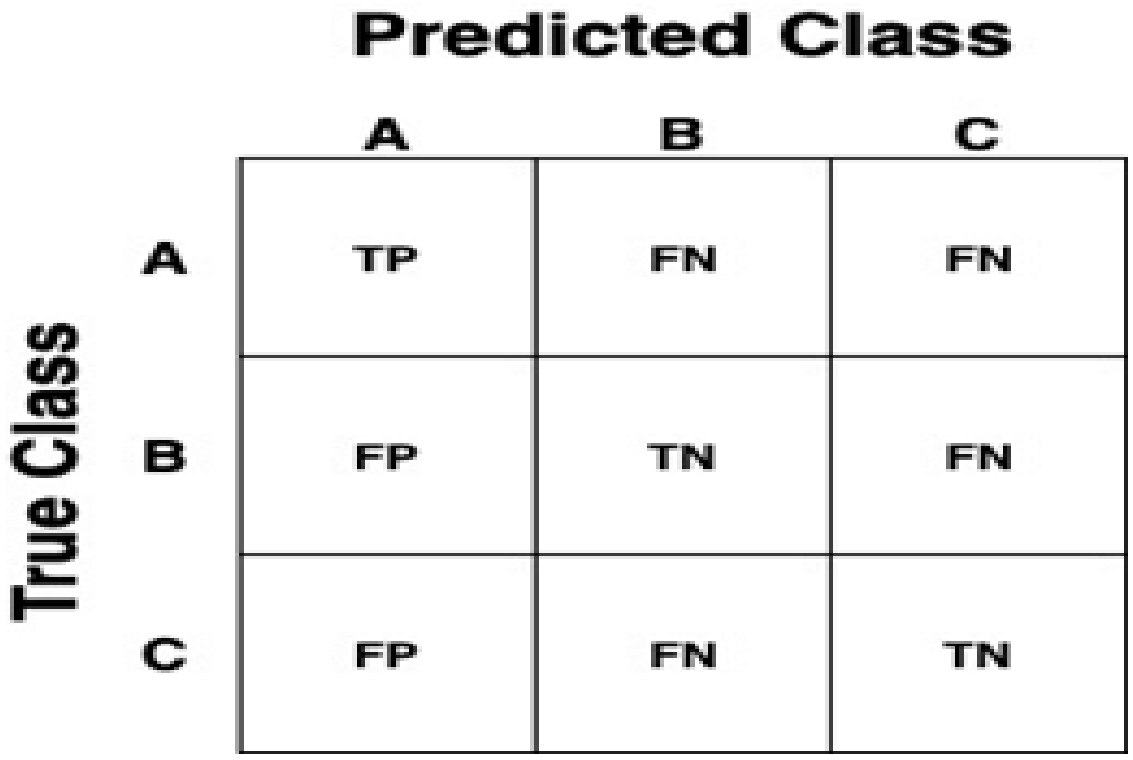

Figure 3.8. Confusion Matrix

- **True Positive(TP):** Model correctly identifies a positive instance as positive.
- **False Positive(FP):** Model wrongly identifies a negative instance as positive.
- **True Negative(TN):** Model correctly identifies a negative instance as negative.
- **False Negative(FN):** Model wrongly identifies a positive instance as negative.

Accuracy: Accuracy is a metric that measures the correctness of predictions made by a model, encompassing both positive and negative instances. It is calculated as the ratio of correctly predicted instances (both true positives and true negatives) to the total number of instances.

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

Precision: Precision focuses on the accuracy of positive predictions. It is calculated as the number of positive predictions divided by the total number of positive predictions. Precision is valuable when the cost of false positive is high, as it emphasizes the accuracy of the model's positive predictions.

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

Recall: Recall, also known as sensitivity or true positive rate, assesses a model's capability to identify all the relevant instances within a dataset. Mathematically, it is computed as

the number of correctly predicted positive instances divided by the total number of actual positive instances. It is crucial when the cost of false negative is high, as it emphasizes to correctly capture all relevant cases.

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

F1-score: The F1-score is the harmonic average of precision and recall, offering a comprehensive evaluation of a model's performance. It reaches its highest value at 1, indicating perfect precision and recall, and its lowest at 0, signifying poor performance. The F1-score is particularly valuable in situations with imbalanced data, where both false positives and false negatives are crucial. Unlike accuracy, which is suitable for balanced data, the F1-score provides a more accurate result in scenarios where the distribution of positive and negative instances is uneven.

$$\text{F1-score} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

ROC-curve: The ROC curve is a visual representation of how good a model is to differentiate between different classes. It plots false positive rate on the X-axis and true positive rate on the Y-axis.The Area Under the ROC Curve (AUC-ROC) is a summary score that encapsulates this performance into a single number. Higher AUC-ROC values signal better performance in separating the different groups.

CHAPTER 4

EXPERIMENTAL RESULTS AND DISCUSSION

Before splitting the dataset into train, test and validation, the Synthetic Minority Oversampling Technique (SMOTE) is applied to address class imbalance, to improve algorithm's ability to handle minority classes. Following the model training and testing phases, the classification report is generated to provide a comprehensive overview of the model's performance across different classes.

4.1 Classification Report of Naive Bayes

Firstly, the Naive Bayes algorithm is applied. For Naive Bayes, the test and validation accuracy is 86%. Below is the classification report of naive bayes:

Table 4.1. Classification Report of Naive Bayes

	Precision	Recall	F1-score	Support
1	0.92	0.73	0.81	122
2	0.85	0.92	0.88	130
3	0.87	0.97	0.92	134
4	0.94	0.74	0.81	137
5	0.82	1.00	0.90	111
6	0.84	0.84	0.84	108
Accuracy			0.86	742
Macro avg	0.87	0.87	0.86	742
Weighted avg	0.87	0.86	0.86	742

The Naive Bayes model achieves its highest precision for the class 1, scoring an impressive

score of 0.92, while encountering a slightly lower precision of 0.82 for the class 5. In terms of recall and F1-score, Class 1 exhibits the lowest values among all classes, with recall at 0.73 and F1-score at 0.81. On the contrary, the model excels in capturing instances of the Class 5, achieving a perfect recall score of 1.00. Additionally, it demonstrates superior F1-score for the Class 3, reaching an impressive 0.92. The overall accuracy of the model is reported at 86%, indicating its ability to make correct predictions across all classes. The below confusion matrix and roc-curve further illustrate the model's performance:

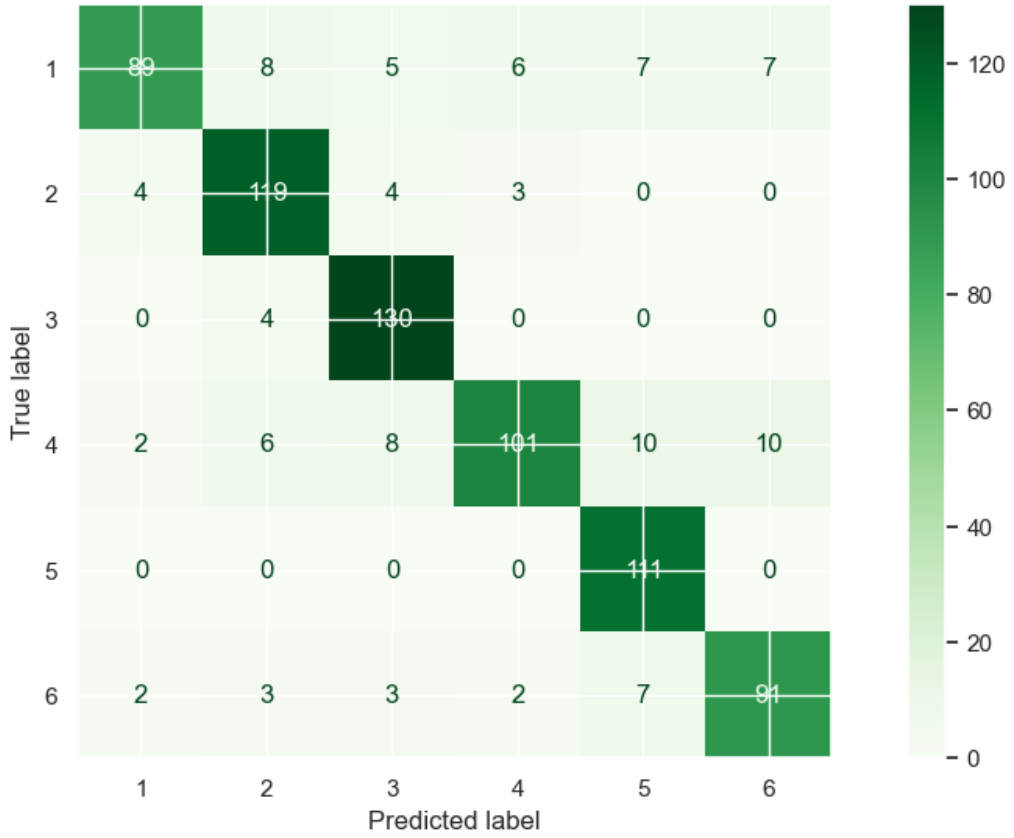

Figure 4.1. Confusion Matrix of Naive Bayes

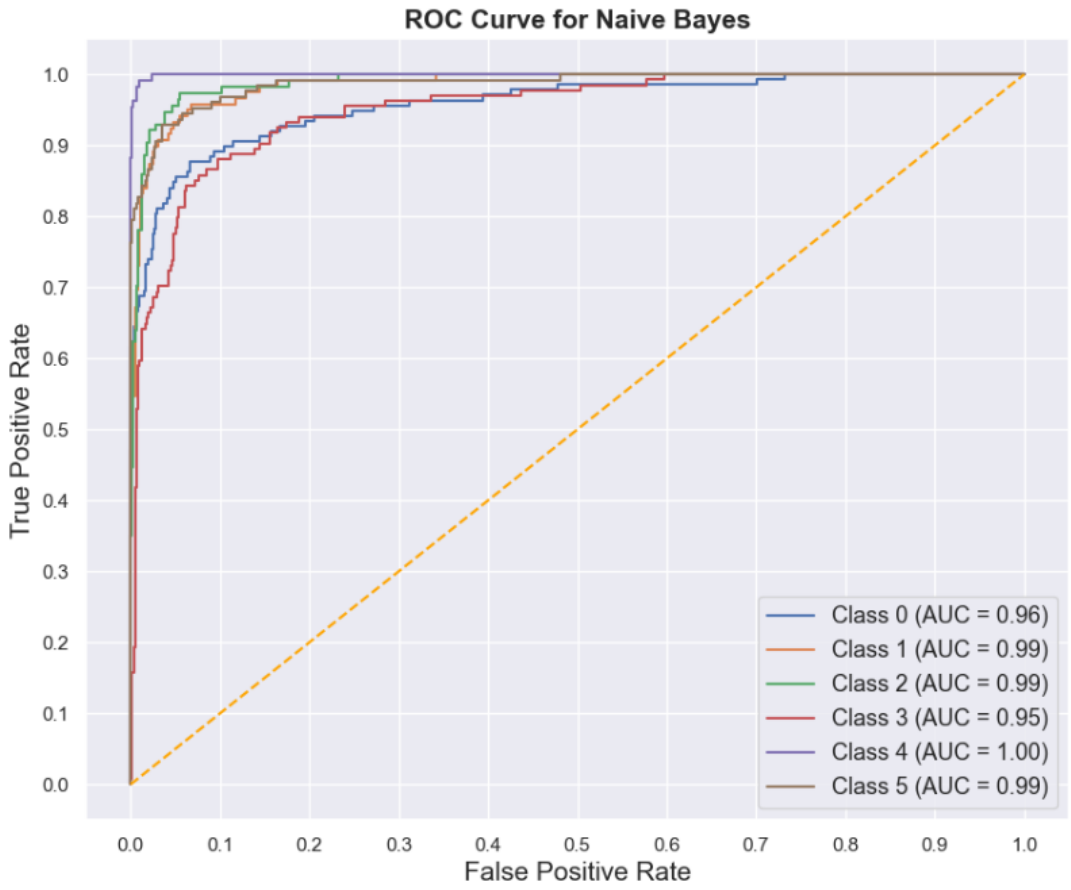

Figure 4.2. ROC Curve of Naive Bayes

4.2 Classification Report of Logistic Regression

For logistic regression, the test and validation accuracy is 88%. Below is the classification report of logistic regression:

Table 4.2. Classification Report of Logistic Regression

	Precision	Recall	F1-score	Support
1	0.86	0.78	0.82	122
2	0.88	0.88	0.88	130
3	0.90	0.96	0.93	134
4	0.90	0.82	0.85	137
5	0.92	0.99	0.96	111
6	0.82	0.87	0.84	108
Accuracy			0.88	742
Macro avg	0.88	0.88	0.88	742
Weighted avg	0.88	0.88	0.88	742

The Logistic Regression model demonstrates consistent and commendable performance

across various classes. It stands out with exceptional precision, recall, and F1-score for different classes, showing its versatility and effectiveness. For instance, the model exhibits strong performance in identifying Class 5, achieving high precision, recall, and F1-score, with values of 0.92, 0.99, and 0.96, respectively. Similarly, the model performs well for other classes, with precision values ranging from 0.82 to 0.90 and recall values ranging from 0.78 to 0.99. The overall accuracy of the model is reported at 88%, indicating its capability to make correct predictions across all classes. Both macro avg and weighted avg metrics further support the model's balanced and robust performance, with macro avg F1-score and weighted avg F1-score both at 0.88. The Logistic Regression model appears to be well-suited for the given classification task, delivering consistent and accurate predictions across various classes. The below confusion matrix and roc-curve further illustrate the model's performance:

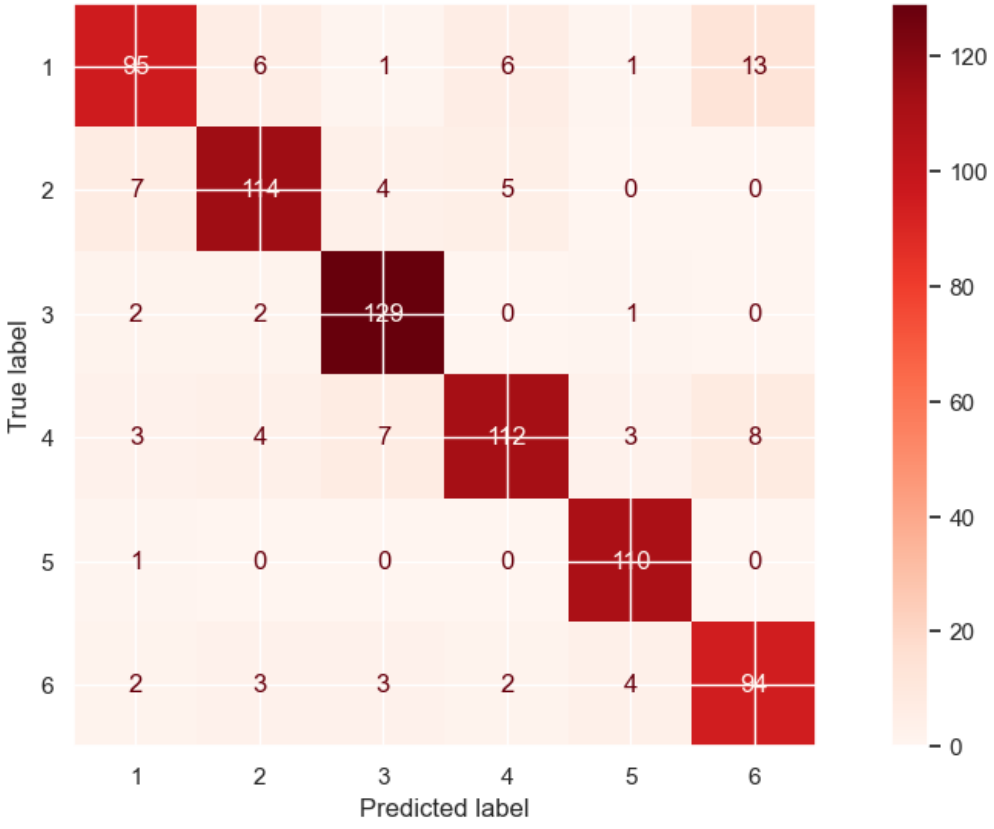

Figure 4.3. Confusion Matrix of Logistic Regression

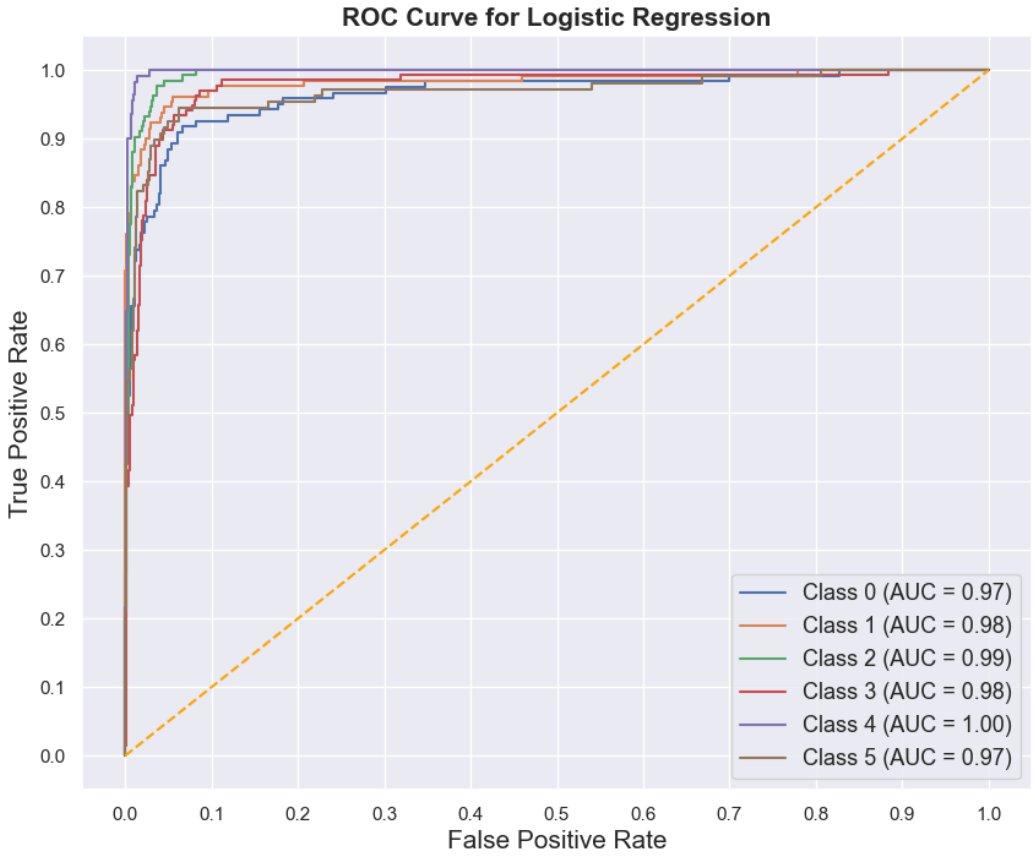

Figure 4.4. ROC Curve of Logistic Regression

4.3 Classification Report of Decision Tree

The Decision Tree algorithm has an test accuracy of 77% and the validation accuracy of 76%. Below is the classification report of decision tree:

Table 4.3. Classification Report of Decision Tree

	Precision	Recall	F1-score	Support
1	0.65	0.70	0.68	122
2	0.79	0.72	0.75	130
3	0.80	0.78	0.79	134
4	0.68	0.76	0.72	137
5	0.93	0.97	0.95	111
6	0.77	0.67	0.72	108
Accuracy			0.77	742
Macro avg	0.77	0.77	0.77	742
Weighted avg	0.77	0.77	0.77	742

The performance of the Decision Tree model varies across different classes, as highlighted in the classification report. The model excels in precision, recall, and F1-score for Class

5, achieving impressive values of 0.93, 0.97, and 0.95, respectively. However, challenges arise in accurately identifying instances of Class 1, where precision, recall, and F1-score are comparatively lower at 0.65, 0.70, and 0.68, respectively. The overall accuracy of the model is reported at 77%, indicating a moderate level of effectiveness in making correct predictions across all classes. Both macro avg and weighted avg metrics suggest a moderate performance, with macro avg F1-score and weighted avg F1-score both at 0.77 The below confusion matrix and roc-curve further illustrate the model's performance:

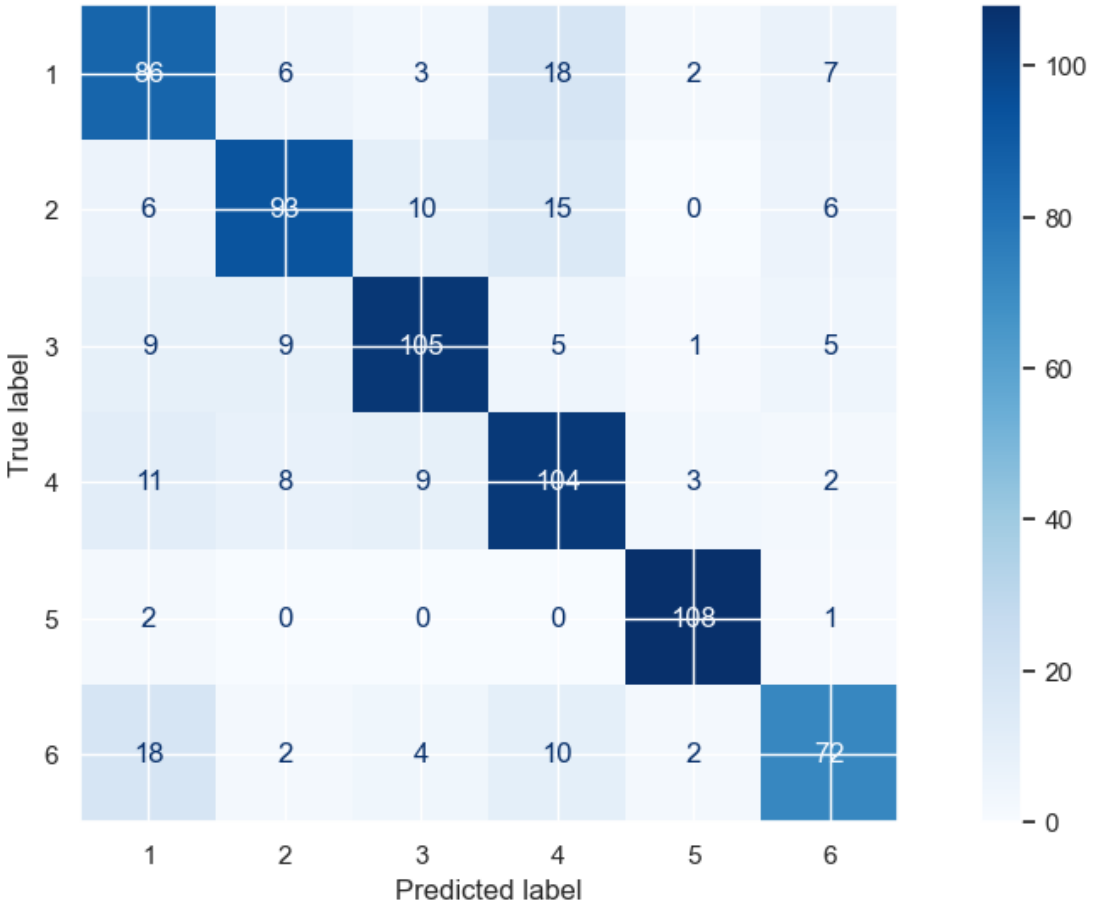

Figure 4.5. Confusion Matrix of Decision Tree

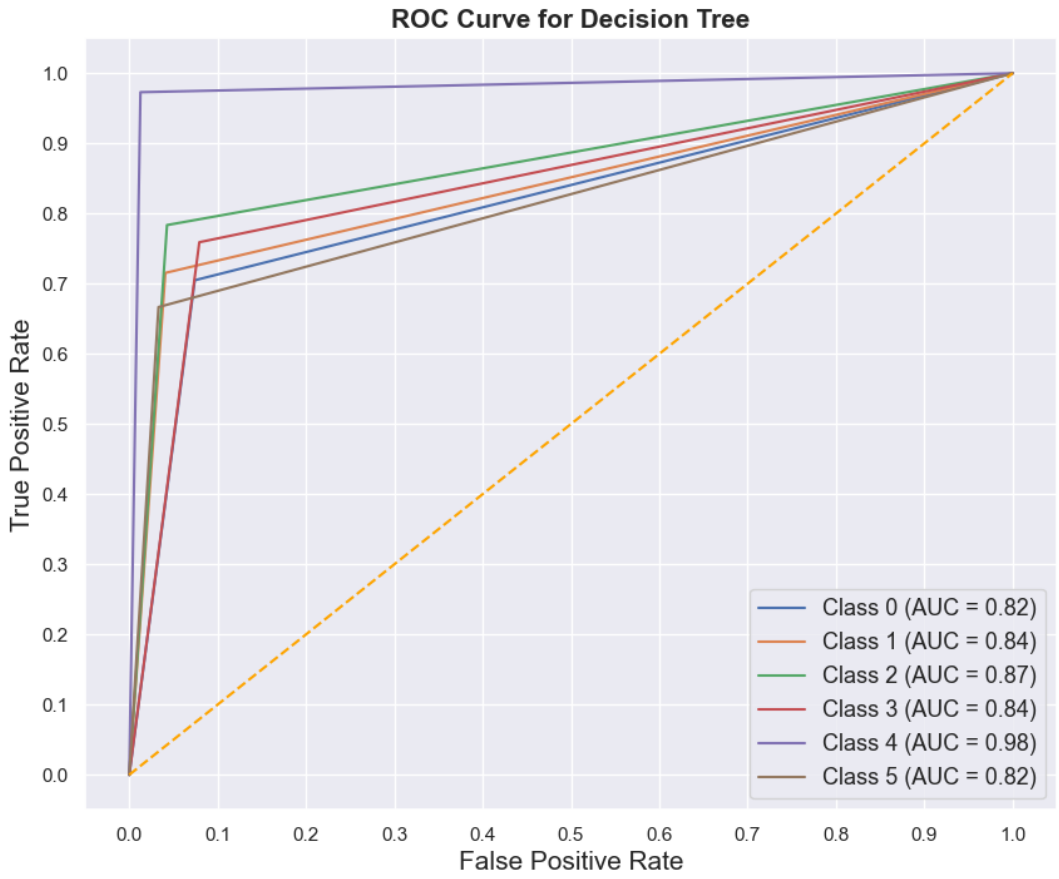

Figure 4.6. ROC Curve of Decision Tree

4.4 Classification Report of Random Forest

The Random Forest algorithm has an test accuracy of 85% and the validation accuracy of 84% when n_estimators=10. For other hyper-parameters the accuracy is lower. Below is the classification report of Random Forest:

Table 4.4. Classification Report of Random Forest

	Precision	Recall	F1-score	Support
1	0.68	0.80	0.74	122
2	0.83	0.89	0.86	130
3	0.96	0.87	0.91	134
4	0.80	0.79	0.79	137
5	0.96	0.98	0.97	111
6	0.93	0.77	0.84	108
Accuracy			0.85	742
Macro avg	0.86	0.85	0.85	742
Weighted avg	0.86	0.85	0.85	742

From the classification report, it is apparent that the Random Forest model delivers a balanced performance across various classes. For instance, the model demonstrates robust performance in identifying instances of Class 3, with precision, recall, and F1-score values of 0.96, 0.87, and 0.91, respectively. Similarly, the model excels in distinguishing instances of Class 5, achieving high precision, recall, and F1-score, with values of 0.96, 0.98, and 0.97, respectively. The overall accuracy of the model is reported at 85%, indicating its effectiveness in making correct predictions across all classes. Both macro avg and weighted avg metrics further support the model's balanced and reliable performance, with macro avg F1-score and weighted avg F1-score both at 0.85. The below confusion matrix and roc-curve further illustrate the model's performance:

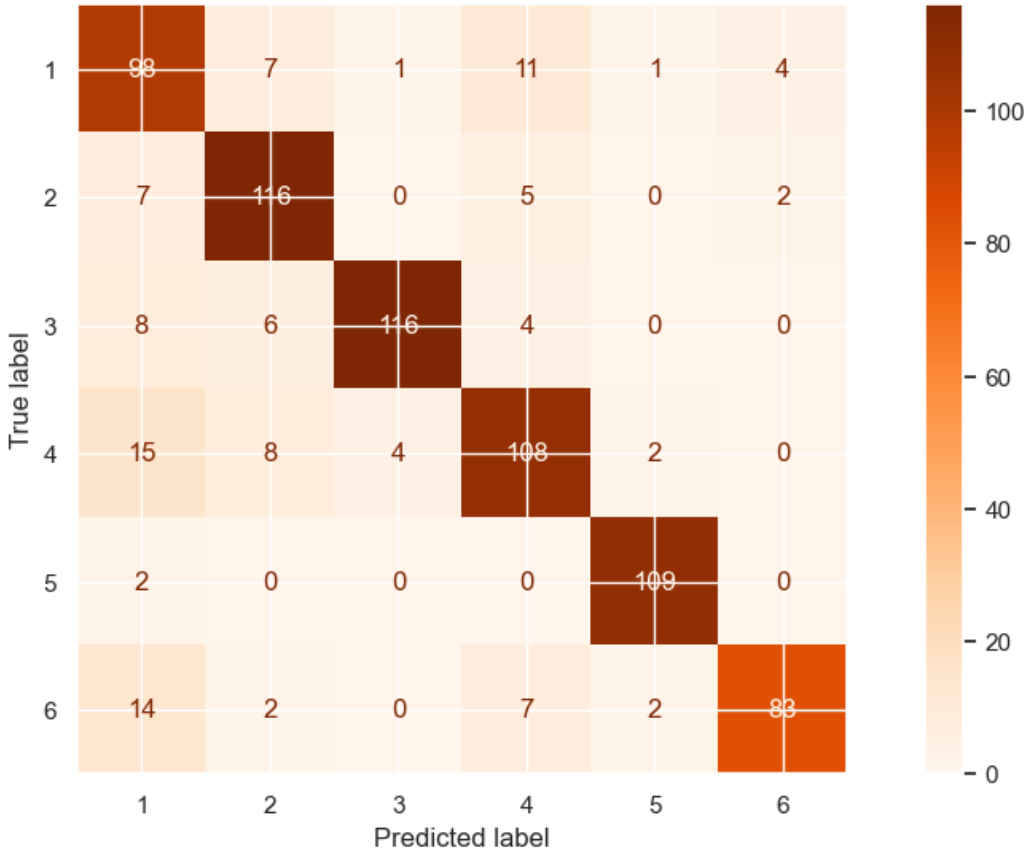

Figure 4.7. Confusion Matrix of Random Forest

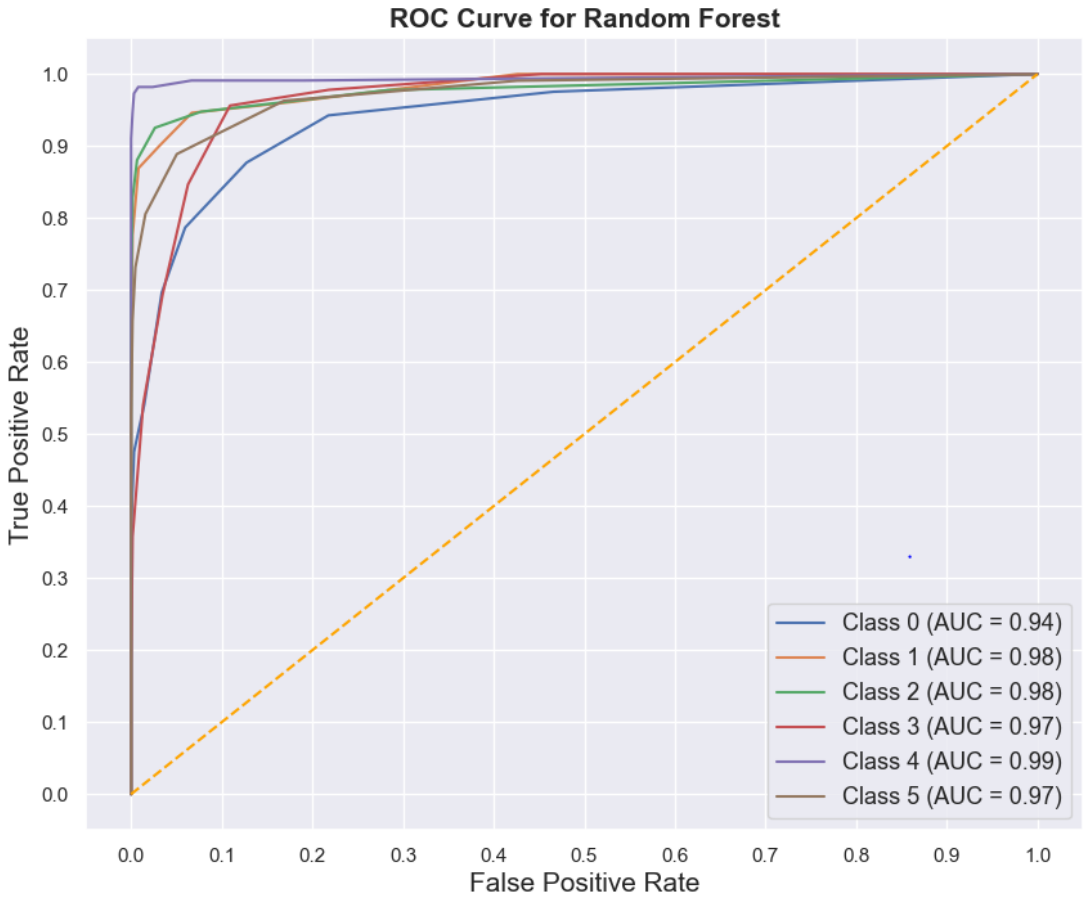

Figure 4.8. ROC Curve of Random Forest

4.5 Classification Report of K-Nearest Neighbors

For n_neighbors=10, The KNN algorithm achieves an test accuracy of 73% and validation accuracy of 72%. Below is the classification report of KNN:

Table 4.5. Classification Report of K-Nearest Neighbors

	Precision	Recall	F1-score	Support
1	0.97	0.31	0.47	122
2	0.77	0.93	0.84	130
3	0.69	0.97	0.81	134
4	0.98	0.31	0.47	137
5	0.74	1.00	0.85	111
6	0.62	0.94	0.75	108
Accuracy			0.73	742
Macro avg	0.79	0.74	0.70	742
Weighted avg	0.80	0.73	0.69	742

From the classification report, it is evident that the KNN model demonstrates a varying

degree of performance across different classes. The model excels in precision for Class 1, achieving a high value of 0.97, but struggles with recall and F1-score, indicating challenges in identifying instances of this class. On the other hand, the model performs well for Class 5, with a precision of 0.74, recall of 1.00, and an impressive F1-score of 0.85. Overall, the model exhibits mixed results across classes, and the reported accuracy of 73% suggests a moderate level of overall performance. The below confusion matrix and roc-curve further illustrate the model's performance:

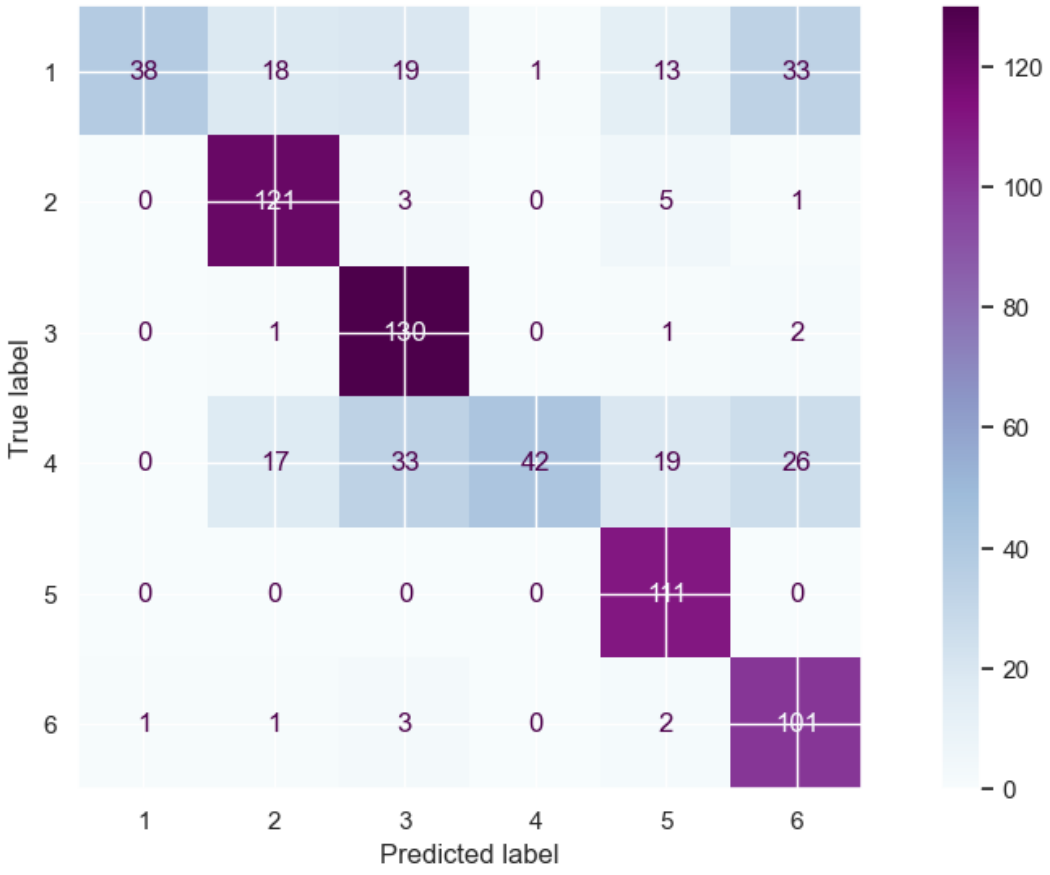

Figure 4.9. Confusion Matrix of K-Nearest Neighbors

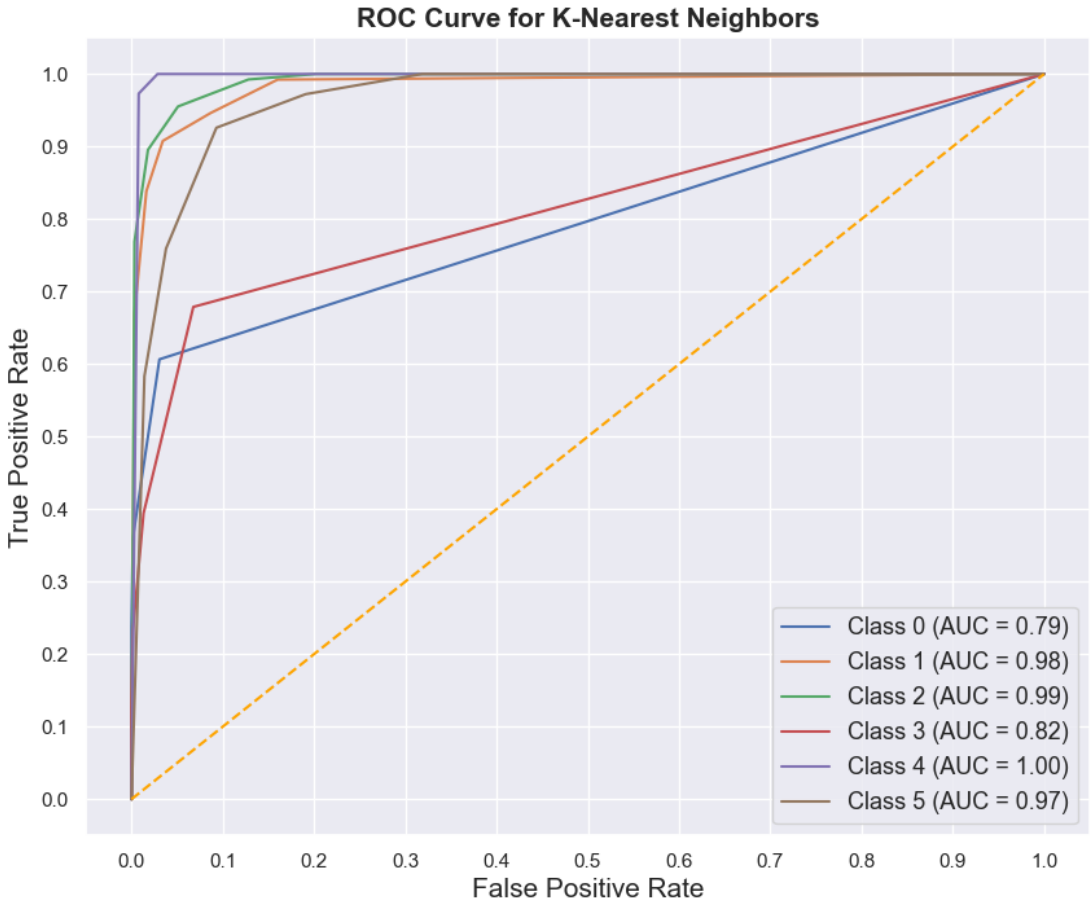

Figure 4.10. ROC Curve of K-Nearest Neighbors

4.6 Classification Report of Support Vector Machine

For SVM, the test accuracy and validation accuracy is 94% and 92% respectively. Below is the classification report of SVM:

Table 4.6. Classification Report of Support Vector Machine

	Precision	Recall	F1-score	Support
1	0.80	0.90	0.86	122
2	0.99	0.91	0.95	130
3	0.98	0.97	0.98	134
4	0.91	0.96	0.93	137
5	0.99	1.00	1.00	111
6	0.99	0.86	0.92	108
Accuracy			0.94	742
Macro avg	0.94	0.94	0.94	742
Weighted avg	0.94	0.94	0.94	742

The Support Vector Machine (SVM) model demonstrates remarkable precision, recall, and

F1-score across various classes, showing its versatility and effectiveness. For instance, it achieves outstanding precision for Class 2 with a remarkable value of 0.99 and exceptional recall for Class 5 with a perfect score of 1.00. The overall accuracy of the SVM model is reported at an impressive 94%, confirming its ability to make accurate predictions across all classes. Both macro avg and weighted avg metrics further underscore the model's consistent and robust performance, with macro avg F1-score and weighted avg F1-score standing at an impressive 0.94. The below confusion matrix and roc-curve further illustrate the model's performance:

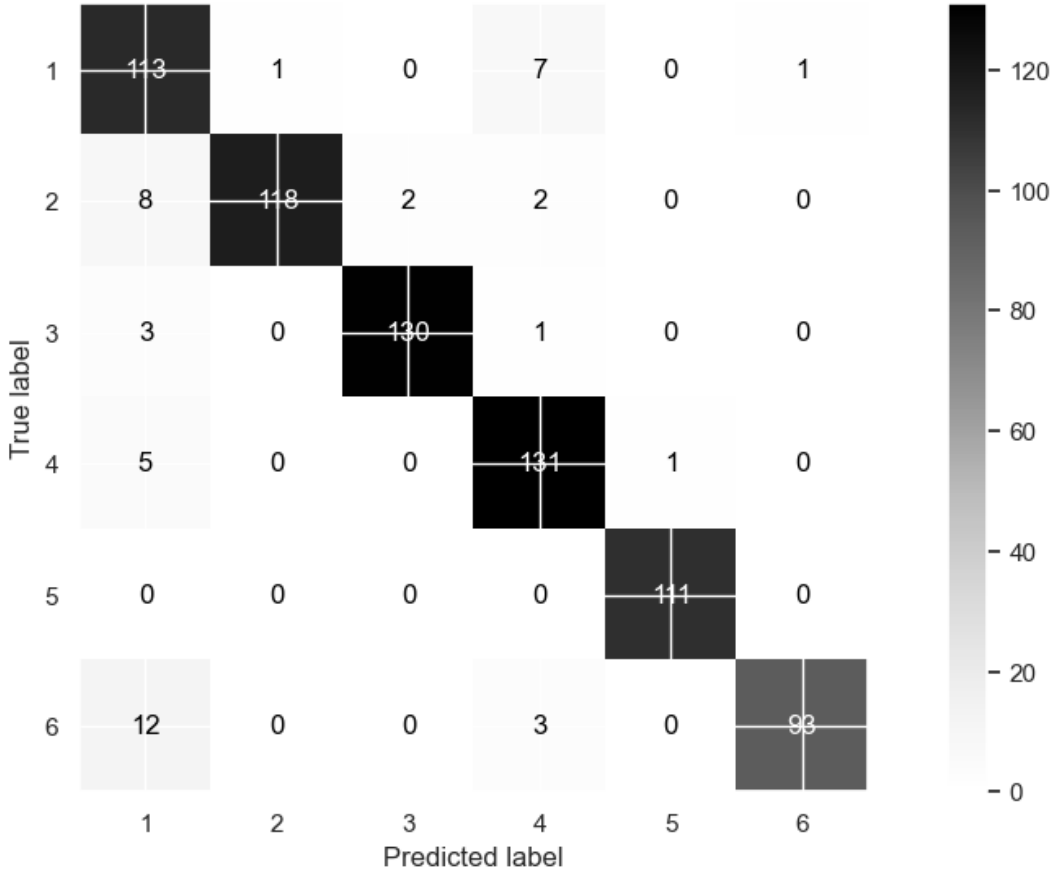

Figure 4.11. Confusion Matrix of SVM

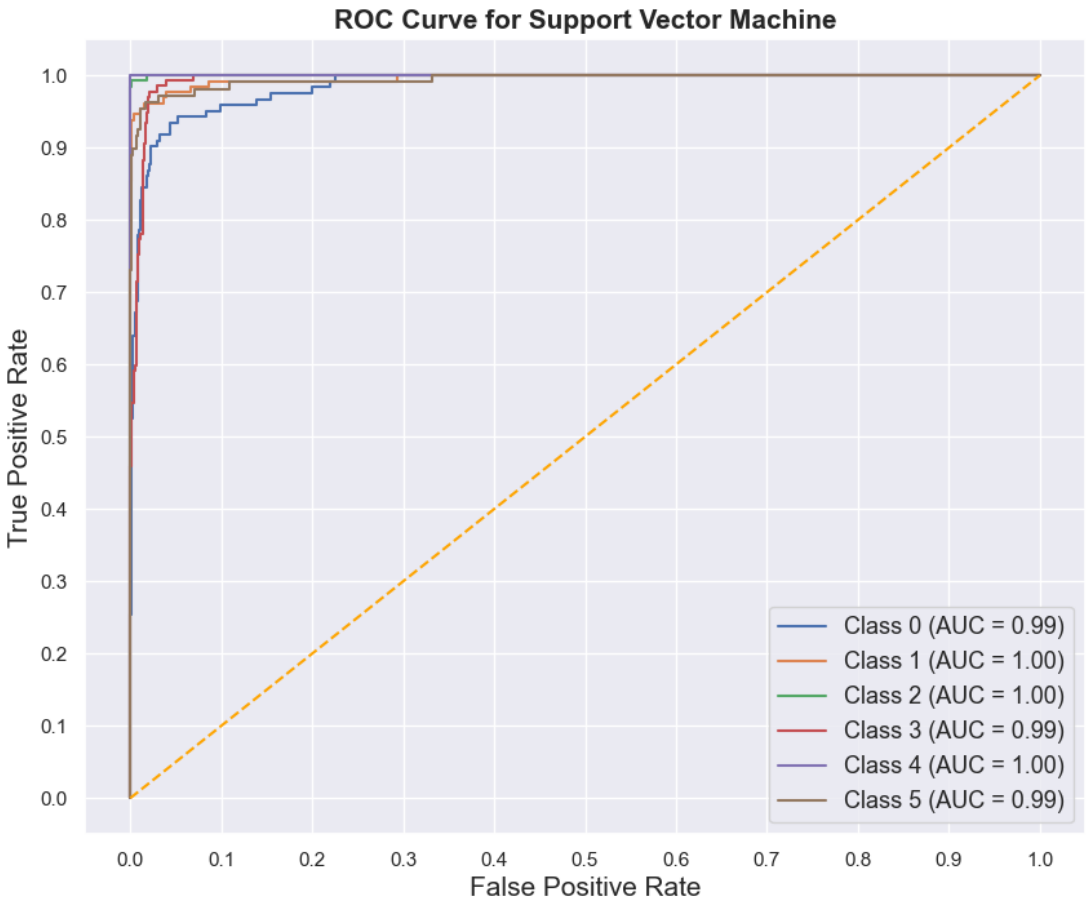

Figure 4.12. ROC Curve of SVM

CHAPTER 5

CONCLUSION AND FUTURE WORK

The article introduces a sentiment analysis model for YouTube comments, utilizing a dataset of 2000 comments collected from various tutorials to capture diverse sentiments. After preprocessing and TF-IDF vectorization, the dataset was divided into training, testing, and validation sets in a 60:20:20 ratio. Six machine learning algorithms were employed, and Support Vector Machine (SVM) emerged as the top performer. SVM demonstrated exceptional precision, recall, and F1-score across various classes, achieving an impressive accuracy of 94%. Other models, including Naive Bayes, Random Forest, and Logistic Regression, also performed well, maintaining consistent accuracy, precision, recall, and F1-score values at 86%, 85%, and 88%, respectively. Conversely, Decision Tree and K-Nearest Neighbors (KNN) exhibited lower effectiveness, with accuracies of 77% and 73%, respectively.

The limitation of the proposed model is that the dataset is relatively smaller in size, and the model is static as it cannot find the sentiment of a comment dynamically.

In future, we plan to increase the dataset size to potentially boost accuracy and detect the sentiment of comments in real time. Additionally, we want to use deep learning algorithm to improve the model's performance.

BIBLIOGRAPHY

[1] R. Singh and A. Tiwari, "Youtube comments sentiment analysis," *International Journal of Scientific Research in Engineering and Management*, vol. 5, no. 5, pp. 1–11, 2021.

[2] A. -K. Al-Tamimi, A. Shatnawi, and E. Bani-Issa, "Arabic sentiment analysis of youtube comments," in *2017 IEEE Jordan Conference on Applied Electrical Engineering and Computing Technologies (AEECT)*, Aqaba, Jordan, 2017, pp. 1–6. DOI: 10.1109/AEECT.2017.8257766.

[3] N. I. Tripto and M. E. Ali, "Detecting multilabel sentiment and emotions from bangla youtube comments," in *2018 International Conference on Bangla Speech and Language Processing (ICBSLP)*, Sylhet, Bangladesh, 2018, pp. 1–6. DOI: 10.1109/ICBSLP.2018.8554875.

[4] R. Pokharel and D. Bhatta, "Classifying youtube comments based on sentiment and type of sentence," *arXiv preprint arXiv:2111.01908*, 2021.

[5] R. Novendri, A. S. Callista, D. N. Pratama, and C. E. Puspita, "Sentiment analysis of youtube movie trailer comments using naïve bayes," *Bulletin of Computer Science and Electrical Engineering*, vol. 1, no. 1, pp. 26–32, 2020.

[6] T. Tehreem, "Sentiment analysis for youtube comments in roman urdu," *arXiv preprint arXiv:2102.10075*, 2021.

[7] M. Aufar, R. Andreswari, and D. Pramesti, "Sentiment analysis on youtube social media using decision tree and random forest algorithm: A case study," in *2020 International Conference on Data Science and Its Applications (ICoDSA)*, Bandung, Indonesia, 2020, pp. 1–7. DOI: 10.1109/ICoDSA50139.2020.9213078.

[8] S. Nawaz, M. Rizwan, S. Yasin, M. Ahmed, and U. Farooq, "Multi-class classification of the youtube comments using machine learning," *Pakistan Journal of Engineering and Technology*, vol. 3, no. 2, pp. 183–188, 2020.

[9] F. I. Tanesab, "Sentiment analysis model based on youtube comment using support vector machine," Ph.D. dissertation, Magister Sistem Informasi Program Pascasarjana FTI-UKSW, 2017.

[10] M. Alkaff, A. R. Baskara, and Y. H. Wicaksono, "Sentiment analysis of indonesian movie trailer on youtube using delta tf-idf and svm," in *2020 Fifth International Conference on Informatics and Computing (ICIC)*, Gorontalo, Indonesia, 2020, pp. 1–5. DOI: 10.1109/ICIC50835.2020.9288579.