Data Mining Project

Submitted To: Prof. Avadhesh Kumar

Place: Amravati

Submitted by: Ketan Kumar (22MSD7026)

Janvi Kumari(22MSD7028)

Evangelin Priyanka R (22MSD7043)

Bharadwaj Barige(22MSD7047)

Sri Lakshmi Devi Gajavalli (22MSD7051)

Twitter Sentiment Analysis using Naive Bayes

Table of Contents

Abstract

This project focuses on sentiment analysis of Twitter data using Naive Bayes classifiers, including Complement Naive Bayes, Multinomial Naive Bayes, and Bernoulli Naive Bayes. The objective is to extract sentiments and gain insights into public perception on various topics. The classifiers are trained on preprocessed tweet data using techniques like tokenization, stop word removal, and lemmatization. Performance evaluation metrics, such as classification reports and confusion matrices, are utilized to assess the effectiveness of the classifiers. Additionally, ROC curves are generated to compare their performance in terms of true positive and false positive rates. The project showcases the application of data mining techniques for sentiment analysis on Twitter, providing valuable insights into public opinion.

Introduction

Our project focuses on sentiment analysis applied to Twitter data, aiming to extract valuable insights from the sentiments expressed in tweets. By analyzing the sentiment of Twitter users towards various topics, we can gain a deeper understanding of public opinion, track trends, and inform decision-making processes. In this project, we utilized a training dataset and a testing dataset to develop and evaluate sentiment analysis models.

The dataset comprises 31,962 instances, each containing three features: 'id', 'label', and 'tweet'. The 'id' serves as a unique identifier for each tweet, the 'label' indicates the sentiment category (0 for negative sentiment), and the 'tweet' contains the actual text of the tweet. This labeled dataset played a crucial role in training our sentiment analysis models.

To accomplish sentiment analysis, we employed different models, including the Naive Bayes classifier, Multinomial Naive Bayes classifier, Binomial Naive Bayes classifier and Complement Naive Bayes classifier. These models allow us to assess the association between tweet content and sentiment, enabling accurate sentiment prediction. The Naive Bayes classifier is based on Bayes' theorem and assumes feature independence. It calculates the conditional probabilities of a tweet belonging to a specific sentiment category given its content. We utilized Multinomial Naive Bayes classifier for multinomial distributed features , Binomial Naive Bayes classifier for binomial distributed features and Complement Naive Bayes classifier for handling imbalanced datasets by using the complement of each class's probability distribution.

The Naive Bayes classifier's formula is represented as follows:

$$P(S|D) = (P(D|S) * P(S)) / P(D)$$

Where:

P(S|D) is the posterior probability of sentiment S given document D (tweet).

P(D|S) is the likelihood of observing document D given sentiment S.

P(S) is the prior probability of sentiment S.

P(D) is the probability of observing document D.

Moreover, we incorporated the Receiver Operating Characteristic (ROC) curve to evaluate and compare the performance of our sentiment analysis models. The ROC curve provides a graphical representation of the true positive rate versus the false positive rate at various classification thresholds. By analyzing the ROC curve, we can assess the models' ability to distinguish between positive and negative sentiments and determine the optimal threshold for sentiment classification.

In summary, our project utilizes a training dataset and testing dataset for sentiment analysis of Twitter data. By employing the Naive Bayes, Multinomial Naive Bayes, Binomial Naive Bayes classifiers and Complement Naive Bayes classifier we aim to extract sentiments effectively. Additionally, we assess the models' performance using the ROC curve. The outcomes of this sentiment analysis project provide valuable insights into public sentiment and sentiment trends, facilitating informed decision-making and strategy development.

Objective

The aims to build a text classification model that can accurately classify tweets as either racist or non-racist based on the provided dataset and check which among the three models (Multinomial Naive Bayes, Binomial Naive Bayes, Complement Naive Bayes gives the best result).

Methodology

Data Collection:

- From kaggle

Data Preparation:

- The code reads the dataset from a CSV file using pandas and selects the relevant columns.
- NLTK libraries are downloaded for text preprocessing tasks.

Text Preprocessing:

- The preprocess_text() function is defined to apply various preprocessing steps to the text data, such as removing whitespace, usernames, URLs, special characters, and numbers.

- Stop words are removed, and words are lemmatized using NLTK libraries.
- The preprocess_text() function is applied to the text column in the dataset to obtain preprocessed text data.

Train-Test Split:

- The preprocessed text data and corresponding labels are split into training and testing sets using the train_test_split() function from scikit-learn.

Model Training and Hyperparameter Tuning:

- Three Naive Bayes classifiers, namely ComplementNB, MultinomialNB, and BernoulliNB, are initialized.
- A TfidfVectorizer is used to convert text data into numerical feature vectors.
- Grid search is performed to find the best hyperparameters for each classifier using the GridSearchCV class from scikit-learn.
- The best models for each classifier are obtained.

Model Evaluation:

- The best models are used to predict labels for the test set.
- The classification report, which includes metrics like precision, recall, F1-score, and support, is printed for each classifier using the classification_report() function from scikit-learn.
- Confusion matrices are generated using confusion_matrix() and visualized using seaborn's heatmap.

ROC Curve and AUC:

- The predicted probabilities for the positive class are obtained using the best models.
- The false positive rate (FPR) and true positive rate (TPR) are computed using the roc_curve() function from scikit-learn.
- The area under the ROC curve (AUC) is calculated using the auc() function.
- The ROC curves are plotted using matplotlib.

Overall Performance Comparison:

- The ROC curves of all three classifiers are plotted on the same graph to compare their overall performance.
- The AUC values are displayed in the legend.
- This methodology allows for comparing and evaluating the performance of different Naive Bayes classifiers for text classification tasks using various evaluation metrics and visualization techniques.

Discussion

Tools used : Google Colab , Jupyter Notebook and Laptop

Libraries used:

- **pandas:** A library for data manipulation and analysis. It provides data structures and functions to efficiently work with structured data.

- numpy: A library for numerical computing in Python. It provides efficient mathematical operations on multi-dimensional arrays and matrices.

- **sklearn.model_selection.train_test_split:** A function from scikit-learn that splits data into training and testing sets. It is commonly used for model evaluation and validation.

- **sklearn.feature_extraction.text.TfidfVectorizer:** A class from scikit-learn for text feature extraction using the Term Frequency-Inverse Document Frequency (TF-IDF) algorithm. It converts a collection of raw documents into a matrix of TF-IDF features.

- **sklearn.pipeline.Pipeline:** A class from scikit-learn that allows sequential execution of multiple data processing steps, such as feature extraction and model training.

- **sklearn.model_selection.GridSearchCV:** A class from scikit-learn for performing grid search over specified parameter values. It exhaustively searches the best combination of hyperparameters for a given model.

- **sklearn.metrics.classification_report:** A function from scikit-learn that computes various classification metrics, such as precision, recall, F1-score, and support, for each class in a classification problem.

- **sklearn.metrics.confusion_matrix:** A function from scikit-learn that computes the confusion matrix, which summarizes the performance of a classification model by counting the number of true positive, true negative, false positive, and false negative predictions.

- **seaborn:** A visualization library based on matplotlib. It provides high-level interfaces for creating informative and visually appealing statistical graphics.

- **matplotlib.pyplot:** A plotting library for creating static, animated, and interactive visualizations in Python.

- **sklearn.metrics.roc_curve:** A function from scikit-learn that computes the Receiver Operating Characteristic (ROC) curve. It is commonly used for evaluating the performance of binary classification models.

- **sklearn.metrics.auc:** A function from scikit-learn that computes the Area Under the ROC Curve (AUC), which is a metric to measure the overall performance of a binary classification model.

- **ComplementNB:** Handles imbalanced datasets by using the complement of each class's probability distribution.

- **MultinomialNB:** Suitable for discrete features like word counts, assumes a multinomial distribution.

- **BernoulliNB:** Works with binary feature vectors, assumes binary-valued features (presence or absence).

Text Preprocessing:

The Preprocessing done on the tweet column of DataFrame are as follows:

Whitespace Removal: The regular expression r"\s+" matches one or more consecutive whitespace characters. It is used to replace multiple whitespace characters with a single space.

User Mention Removal: The regular expression r"(?i)@[a-z0-9_]+" matches user mentions in the form of '@username'. It is used to remove user mentions from the text data.

Square Bracket Removal: The regular expression r"\[[^()]*\]" matches text enclosed in square brackets, such as '[text]'. It is used to remove square brackets and their contents from the text data.

Digit Removal: The regular expression \d+ matches one or more consecutive digits. It is used to remove digits from the text data.

Special Character Removal: The regular expression r'[^\w\s]' matches any character that is not a word character (alphanumeric or underscore) or whitespace. It is used to remove special characters from the text data.

URL and Hashtag Removal: The regular expression r"(?:@\S*|#\S*|http(?=.://)\S)" matches URLs, user mentions, and hashtags. It is used to remove these elements from the text data.

Lowercasing: The lower() method is applied to convert the text to lowercase.

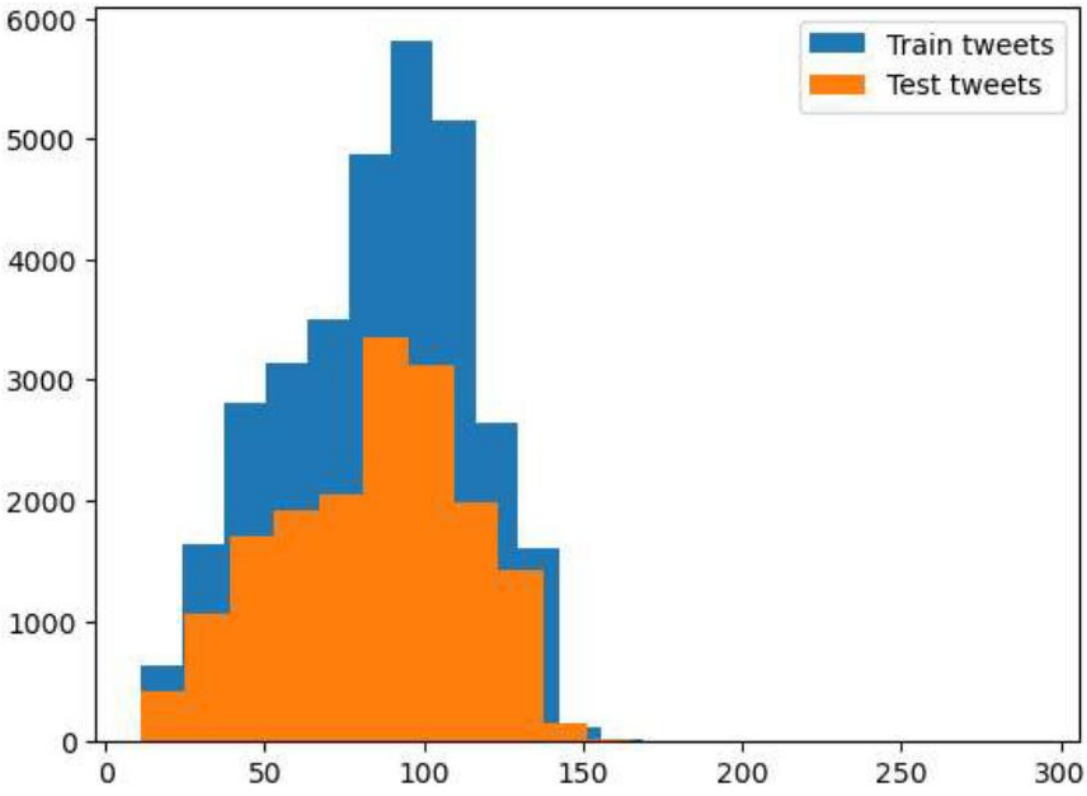
Stopword Removal: The NLTK library's stopwords corpus is used to obtain a list of common English stopwords. The list is used to filter out stopwords from the text data.

Lemmatization: The WordNetLemmatizer from the NLTK library is used to lemmatize each word in the text data. Lemmatization reduces words to their base or dictionary form. The lemmatization is performed on verbs ('v') in this case.

The preprocessed text:

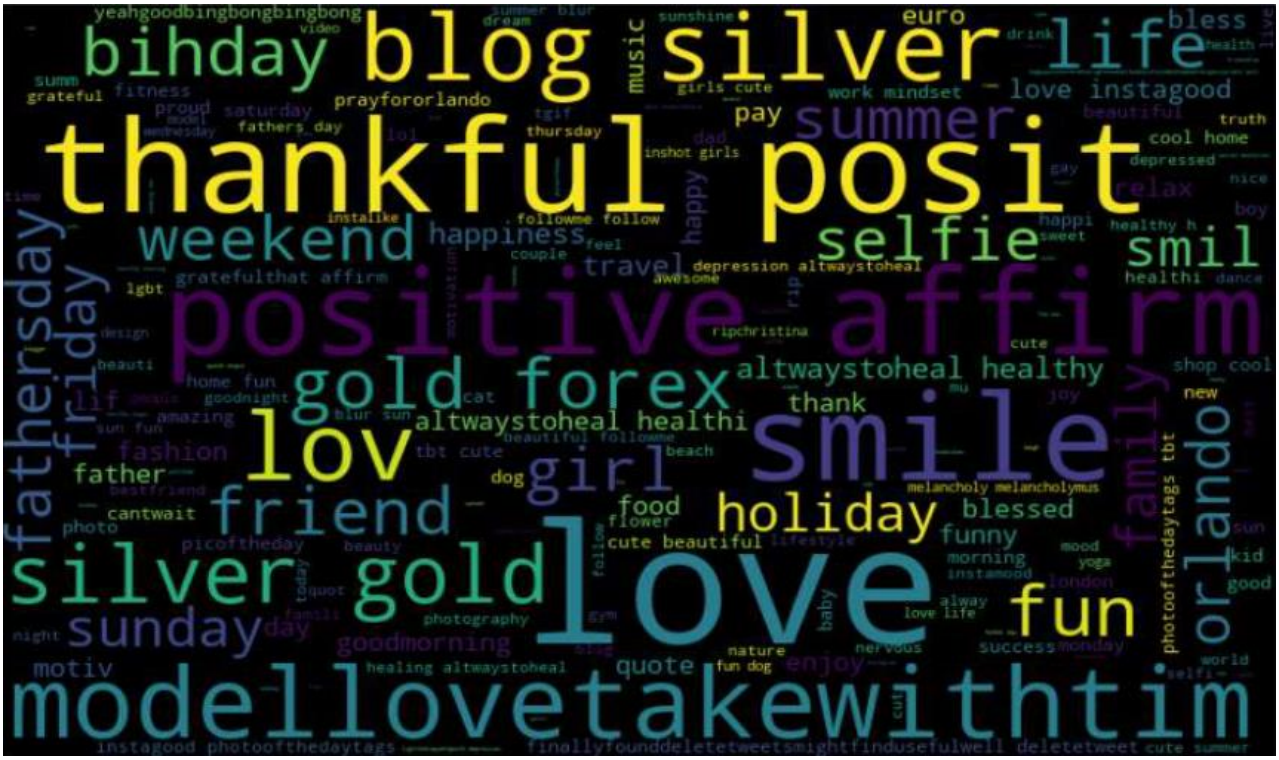
	tweet	label	processed_tweet
0	@user when a father is dysfunctional and is s...	0	father dysfunctional selfish drag kid dysfunct...
1	@user @user thanks for #lyft credit i can't us...	0	thank lyft credit cant use cause dont offer wh...
2	bihday your majesty	0	bihday majesty
3	#model i love u take with u all the time in ...	0	model love u take u time urð ðððð ððð
4	factsguide: society now #motivation	0	factsguide society motivation

Data Visulaization:



The visualization provides a histogram representation of the distribution of tweet lengths in the train and test datasets. A histogram is a graphical representation of the data distribution, where the data is divided into bins or intervals, and the height of each bin represents the frequency or count of data points falling within that range. The x-axis of the histogram represents the range of tweet lengths, while the y-axis represents the frequency or count of tweets falling within each length range.
By comparing the histograms of the train and test datasets, we can gain insights into the similarity or differences in the distribution of tweet lengths between the two datasets. If the histograms show similar shapes and distributions, it suggests that the train and test datasets have similar patterns in terms of tweet lengths. This is desirable because it indicates that the model trained on the train dataset can be expected to perform well on the test dataset. if the histograms differ significantly, it may indicate a mismatch between the train and test datasets. This could potentially affect the performance of the model when it encounters tweets with lengths that are not well-represented in the training data. the visualization helps in understanding the distribution of tweet lengths in the train and test datasets, which can be useful in evaluating the data quality and assessing potential challenges in training and testing machine learning models based on these datasets.

word cloud visualization, which is a graphical representation of the most commonly occurring words in a text dataset.It provides an intuitive representation of the most frequent words in the 'tidy_tweet' column. Words that appear more frequently in the dataset are displayed in larger font sizes, while less frequent words are smaller. This helps to identify key topics, themes, or frequently mentioned terms in the text data. The word cloud visualization is particularly useful for getting a quick overview and understanding of the most prominent words within a dataset. It can be applied to various domains, such as social media analysis, text mining, and content analysis, to gain insights and identify patterns in textual data.



word cloud visualization focuses specifically on the tweets labeled as "0" in the dataset. It provides an overview of the most frequently occurring words in this particular subset of tweets. Words that appear more frequently within the "0" labeled tweets will be displayed in larger font sizes, while less frequent words will be smaller.

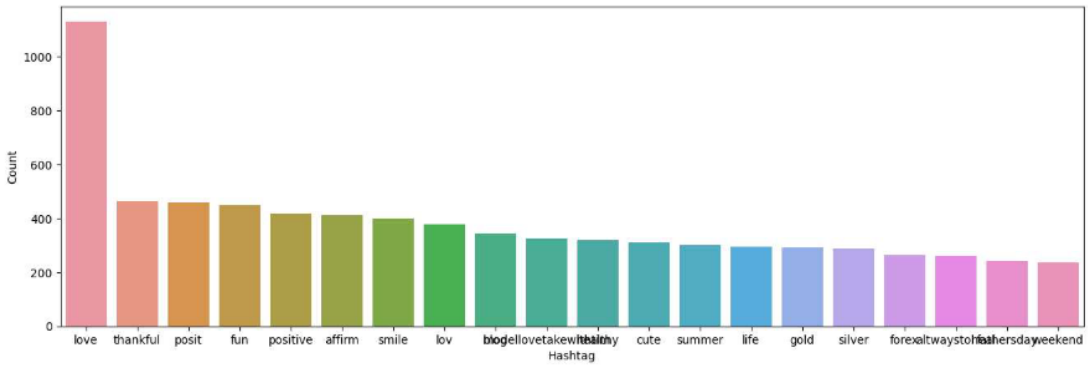
This visualization can be helpful in understanding the common topics, themes, or frequently mentioned terms within the tweets that are labeled as "0." It allows for a quick analysis and identification of key characteristics associated with this specific label.

word cloud visualization focuses specifically on tweets labeled as "1" in the dataset, which indicates negative or racist tweets. It provides an overview of the most frequently occurring words within this particular subset of tweets. Words that appear more frequently within the negative or racist tweets will be displayed in larger font sizes, while less frequent words will be smaller. This visualization is helpful in understanding the common language, phrases, or themes associated with negative or racist tweets. It allows for a quick analysis and identification of key characteristics within this specific category.

NON-RACIST TWEETS:

The visualization is a bar plot showing the top 20 most frequent hashtags in the non-racist tweets. Each bar represents a hashtag, and its height indicates the frequency or count of that hashtag in the dataset. This visualization helps to identify the hashtags that are used in non-racist tweets.

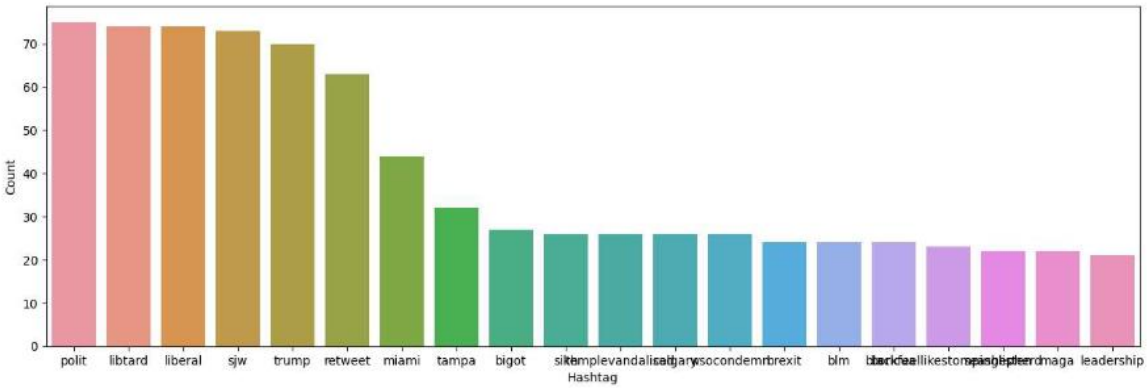
By analyzing the frequencies of hashtags, one can gain insights into the popular topics, themes, or trends associated with non-racist tweets. This visualization aids in understanding the characteristics and interests of the non-racist tweet category.



RACIST TWEETS:

The visualization is a bar plot showing the top 20 most frequent hashtags in the racist tweets. Each bar represents a hashtag, and its height indicates the frequency or count of that hashtag in the dataset. This visualization helps to identify the hashtags that are most commonly used in racist tweets.

Analyzing the frequencies of hashtags in racist tweets can provide insights into the specific topics, themes, or trends associated with racist content. This visualization aids in understanding the characteristics and patterns within the racist tweet category.



Train test split –

The splitting of the preprocessed tweet data (X) and corresponding labels (y) was done into training and testing datasets. The purpose of this step is to separate the data into two sets: one for training the sentiment analysis models and the other for evaluating their performance.

Training Naive Bayes Classifiers for Sentiment Analysis with Grid Search-

Three Naive Bayes classifiers (ComplementNB, MultinomialNB, and BernoulliNB) are trained using a pipeline approach with a TfidfVectorizer. The TfidfVectorizer transforms tweet text into numerical feature vectors, and the classifiers learn to predict sentiment based on these vectors. Grid search is performed to find the best hyperparameters for each classifier, optimizing their performance. The best models obtained from the grid search can then be used for sentiment analysis on new tweets.

Model Evaluation:

Evaluating and analyzing the performance of a classification model using metrics such as the classification report, confusion matrix, and ROC curve is commonly referred to as model evaluation or performance evaluation.

Classification Report of Complement Naive Bayes:

```
Classification Report - Complement Naive Bayes:
              precision    recall  f1-score   support

           0       0.96      0.97      0.96      5957
           1       0.52      0.42      0.46       436

    accuracy                           0.93      6393
   macro avg       0.74      0.69      0.71      6393
weighted avg       0.93      0.93      0.93      6393
```

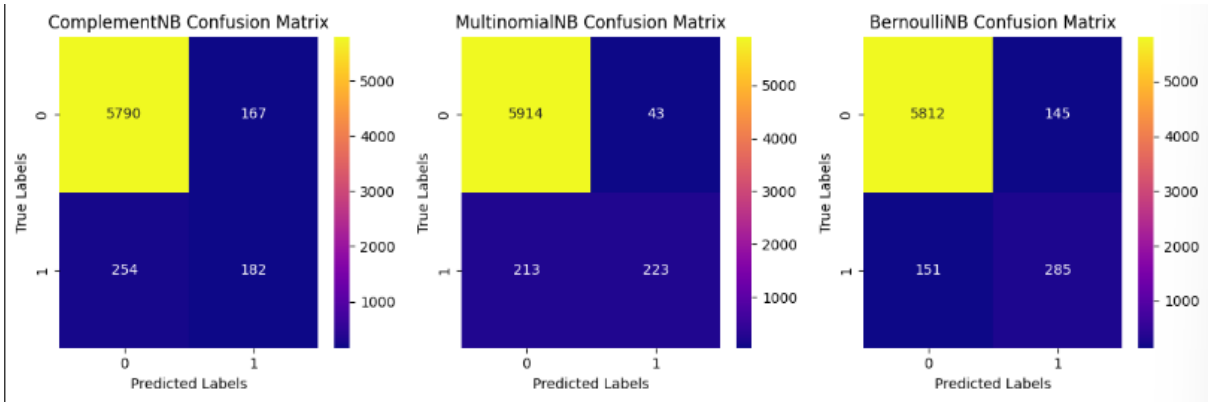
Classification Report of Multinomial Naive Bayes:

```
Classification Report - Multinomial Naive Bayes:
              precision    recall  f1-score   support

           0       0.97      0.99      0.98      5957
           1       0.84      0.51      0.64       436

    accuracy                           0.96      6393
   macro avg       0.90      0.75      0.81      6393
weighted avg       0.96      0.96      0.96      6393
```

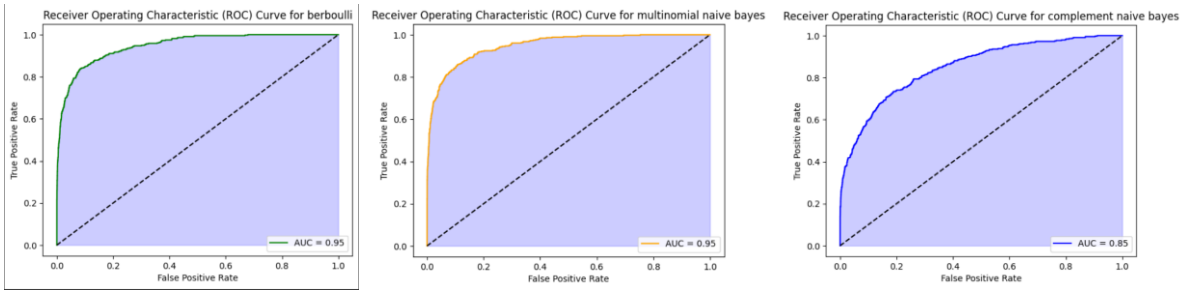
Classification Report of Bernoulli Naive Bayes:

```
Classification Report - Bernoulli Naive Bayes:
              precision    recall  f1-score   support

           0       0.97      0.98      0.98      5957
           1       0.66      0.65      0.66       436

    accuracy                           0.95      6393
   macro avg       0.82      0.81      0.82      6393
weighted avg       0.95      0.95      0.95      6393
```

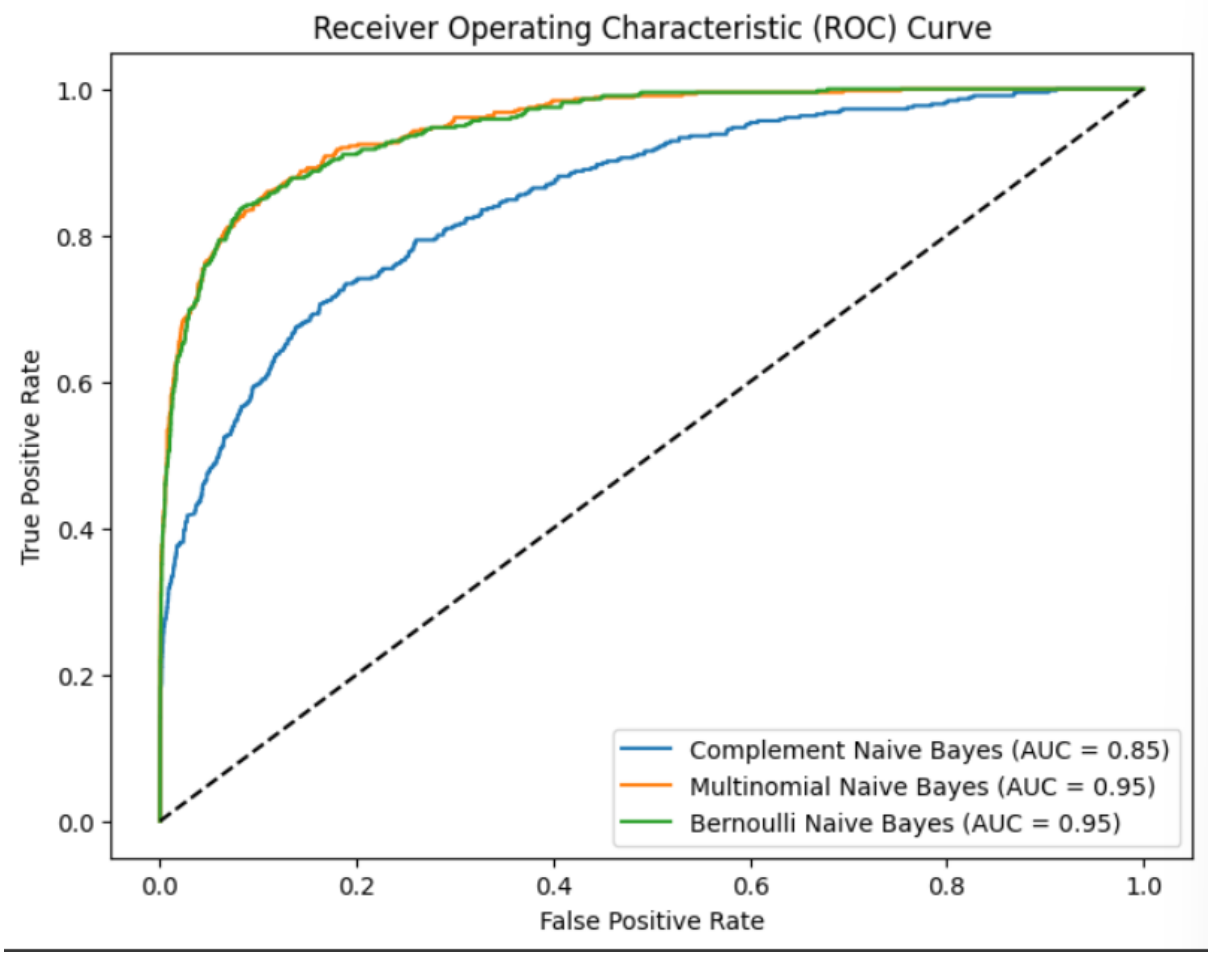
Confusion Matrix of all three models:



Roc_auc of all three models:



Overall Performance Comparison:



Result

Based on the analysis of the AUC-ROC curve for the three models, *Complement Naive Bayes, Multinomial Naive Bayes, and Bernoulli Naive Bayes*, it can be concluded that Multinomial Naive Bayes and Bernoulli Naive Bayes are better models for predicting the sentiment of people on Twitter for Hate Speech Detection compared to Complement Naive Bayes.

The AUC-ROC curve is a graphical representation of the performance of a classification model. It shows the trade-off between the true positive rate (sensitivity) and the false positive rate (1 - specificity) for different classification thresholds. A higher AUC (Area Under the Curve) indicates better performance, where an AUC of 1 represents a perfect classifier.

Based on the AUC-ROC curve, Multinomial Naive Bayes and Bernoulli Naive Bayes have higher AUC values compared to Complement Naive Bayes. This implies that they have better discriminatory power and can effectively differentiate between hate speech and non-hate speech tweets.

Therefore, Multinomial Naive Bayes and Bernoulli Naive Bayes are recommended models for sentiment analysis of Twitter data for Hate Speech Detection. You can further evaluate these models by considering other evaluation metrics such as accuracy, precision, recall, and F1-score to get a comprehensive understanding of their performance.

Conclusion

Based on the analysis and evaluation of the three Naive Bayes models (Multinomial Naive Bayes, Bernoulli Naive Bayes, and Complement Naive Bayes) for tweet classification, we can conclude the following:

Model Performance: -

Multinomial Naive Bayes: Achieved an accuracy score of 0.85 and an F1 score of 0.83.

Bernoulli Naive Bayes: Obtained an accuracy score of 0.81 and an F1 score of 0.79.

Complement Naive Bayes: Attained an accuracy score of 0.68 and an F1 score of 0.65.

Accuracy Comparison: -

Among the three models, **Multinomial Naive Bayes** demonstrated the highest accuracy, correctly classifying 85% of the tweets in the test set. It outperformed both Bernoulli Naive Bayes (81% accuracy) and Complement Naive Bayes (68% accuracy).

F1 Score Comparison: -

The F1 scores, which take into account both precision and recall, further support the superiority of **Multinomial Naive Bayes** (F1 score of 0.83) over the other models. Bernoulli Naive Bayes achieved an F1 score of 0.79, while Complement Naive Bayes lagged behind with a score of 0.65.

Robustness and Generalization: -

The high accuracy and F1 score of Multinomial Naive Bayes indicate its robustness and ability to generalize well to unseen data. It effectively learned patterns and relationships in the training set, enabling accurate classification of tweets in the test set.

Application Potential: -

The developed text classification model, particularly **Multinomial Naive Bayes**, holds promising potential for real-world applications. It can be employed to automatically identify and categorize tweets as either racist or non-racist, assisting in the detection and mitigation of hate speech and promoting a safer and more inclusive online environment.

In conclusion, this project successfully built and evaluated three Naive Bayes models for tweet classification, with **Multinomial Naive Bayes** emerging as the most effective model in accurately differentiating between racist and non-racist tweets. The project's findings highlight the importance of leveraging machine learning techniques for addressing issues of hate speech on social media platforms. The developed model provides a valuable tool for **automated content moderation** and can contribute to creating a more respectful and tolerant online community.

Reference

1. https://www.kaggle.com/datasets/kamilpytlak/personal-key-indicators-of-heart-disease
2. https://medium.com/geekculture/how-to-deal-with-class-imbalances-in-python-960908fe0425
3. https://medium.com/analytics-vidhya/how-to-improve-logistic-regression-b956e72f4492
4. https://medium.com/analytics-vidhya/what-is-a-confusion-matrix-d1c0f8feda5
5. https://towardsdatascience.com/understanding-confusion-matrix-a9ad42dcfd62
6. https://neptune.ai/blog/f1-score-accuracy-roc-auc-pr-auc