



# **A.D. PATEL INSTITUTE OF TECHNOLOGY**

**(A Constituent College of CVM University)**

**New V. V. Nagar**



## **DEPARTMENT OF INFORMATION TECHNOLOGY**

### **Mini Project Report on Consumer Sentiment Analyzer**

**Submitted By –**

**Smit Patel (12002080601107)**

**Varni Bavarva (12002080601115)**

**Guided By –**

**Prof. Sherin Mariam Jijo**

**MINI PROJECT (102040601)**

**A.Y. 2022-23 EVEN TERM**



## **CERTIFICATE**

This is to certify that the Mini Project Report submitted entitled “**Consumer Sentiment Analyzer**” has been carried out by **Smit Patel** (12002080601107) under guidance in partial fulfillment for the Degree of Bachelor of Engineering Information Technology , 6<sup>th</sup> Semester of A. D. PATEL INSTITUTE OF TECHNOLOGY , CVM University , New Vallabh Vidyanagar during the academic year 2022- 23.

Prof. Sherin Mariam Jijo

Internal Guide

Dr. Narendrasinh Chauhan

Head of Department



## **CERTIFICATE**

This is to certify that the Mini Project Report submitted entitled “**Consumer Sentiment Analyzer**” has been carried out by **Varni Bavarva** (12002080601115 ) under guidance in partial fulfillment for the Degree of Bachelor of Engineering in Information Technology, 6<sup>th</sup> Semester of A. D. Patel Institute of Technology , CVM University , New Vallabh Vidyanagar during the academic year 2022-23.

Prof. Sherin Mariam Jijo

Internal Guide

Dr. Narendrasinh Chauhan

Head of Department

# Acknowledgement

We would like to express our sincere gratitude to all those who have contributed to the successful completion of our college project.

First and foremost, We would like to thank our project supervisor, Prof. Sherin Mariam Jijo for their guidance, support, and valuable feedback throughout the project. Their expertise and insights have been invaluable in shaping the direction and focus of our research.

We would also like to extend our thanks to the faculty members including lab-teacher, Prof. Sherin Mariam Jijo, who have provided us with their valuable time, insights, and resources. Their willingness to share their expertise and knowledge has been instrumental in shaping our understanding of the subject matter.

We would like to express our appreciation to our friends and family who have provided us with the emotional support and encouragement We needed to complete this project. Their unwavering support has been a source of motivation and inspiration for us.

Finally, We would like to acknowledge the participants who have taken part in our study. Their time, effort, and cooperation have been vital in collecting the necessary data and information needed for this project.

Once again, thank you to everyone who has contributed to the successful completion of our college project.

# **Abstract**

Sentiment Analysis also known as Opinion Mining refers to the use of natural language processing, text analysis to systematically identify, extract, quantify, and study affective states and subjective information. Sentiment analysis is widely applied to reviews and survey responses, online and social media, and healthcare materials for applications that range from marketing to customer service to clinical medicine.

In this project, we aim to perform Sentiment Analysis of product based reviews. Data used in this project are online product reviews collected from “amazon.com”. We expect to do review-level categorization of review data with promising outcomes.

# INDEX

<b>1. INTRODUCTION.....</b>	<b>1</b>
<b>2 REVIEW OF LITERATURE.....</b>	<b>3</b>
<b>3 METHODOLOGY .....</b>	<b>4</b>
3.1 HARDWARE AND SOFTWARE REQUIREMENTS.....	4
3.2 DATASET .....	4
3.3 DATA PREPROCESSING.....	5
3.4 SENTIMENT CLASSIFICATION ALGORITHMS .....	5
3.4.1 NAIVE BAYESIAN CLASSIFIER.....	5
3.4.2 RANDOM FOREST.....	6
3.4.3 SUPPORT VECTOR MACHINE .....	7
3.4.4 LOGISTIC REGRESSION.....	8
3.4.5 MODEL EVALUATION OF CLASSIFICATION ALGORITHM .....	9
3.5 SENTIMENT ANALYSIS USING NLP .....	11
3.5.1 VADER MODEL .....	13
3.5.2 ROBERTA MODEL.....	13
3.5.3 TRANSFORMERS PIPELINE.....	14
<b>4 RESULTS .....</b>	<b>16</b>
4.1 PERFORMANCE OF THE MODELS.....	16
4.2 CONFUSION MATRIX OF MODELS .....	18
4.3 DISCUSSION OF THE RESULTS.....	25
<b>5 CONCLUSION .....</b>	<b>26</b>
<b>6 REFERENCES.....</b>	<b>34</b>

# 1. INTRODUCTION

E-commerce platform sentiment analysis has turn out to be a distinguished area of study and experimentation in current years. Most research has been confined to classify tweets into positive, negative categories ignoring sarcasm. Human emotions are extremely diverse and cannot be restricted to certain metrics alone. Polarity analysis gives limited information on the actual intent of message delivered by author and just positive or negative classes are not sufficient to understand nuances of underlying tone of a sentence. This brings the need to take one step above sentiment analysis leading to emotion analysis. In this paper we throw light on methods we have used to derive sentiment analysis considering sarcasm and how we have accomplished emotion analysis of user opinions.

A supervised learning technique provides labels to classifier to make it understand the insights among various features. Once the classifier gets familiarized with train data it can perform classification on unseen test data. We have chosen Naive Bayes and Support Vector Machine classification algorithms to carry out sentiment and emotional analysis respectively.

Performing SA (sentiment analysis) and EA (emotion analysis) will help organizations or companies to improve services, track products and obtain customer feedback in a normalized form. Gaining insights from large volumes of data is a mountain of a task for humans hence using an automated process will easily drill down into different customer feedback segments mentioned on social media or elsewhere. Effective business strategies can be built from results of sentiment and emotion analysis. Identifying clear emotions will establish a transparent meaning of text which potentially develops customer relationships, motivation and extends consumer expectations towards a brand or service.

Generally, people discuss a lot of things daily, but it is difficult to get insights just by reading through each of their opinions so there should be a way that helps us to get insights of users opinions in an unbiased manner, So this model helps in drawing out Sentiment, Emotions of users, classify them and finally present them to us. Sentiment analysis is the prediction of emotions in a word, sentence, or corpus of documents.

It is intended to serve as an application to understand the attitudes, opinions and emotions expressed within an online mention. The intention is to gain an overview of the wider public opinion behind certain topics.

Businesses primarily run over customers satisfaction, customer reviews about their products. Shifts in sentiment on social media have been shown to correlate with shifts in stock markets. Identifying customer grievances thereby resolving them leads to customer satisfaction as well as trustworthiness of an organization. Hence there is a necessity of an unbiased automated system to classify customer reviews regarding any problem.

In today's environment where we're justifiably suffering from data overload (although this does not mean better or deeper insights), companies might have mountains of customer feedback collected; but for mere humans, it's still impossible to analyze it manually without any sort of error or bias.

Oftentimes, companies with the best intentions find themselves in an insights vacuum. You know you need insights to inform your decision making and you know that you're lacking them, but don't know how best to get them.

Sentiment analysis provides some answers into what the most important issues are, from the perspective of customers, at least. Because sentiment analysis can be automated, decisions can be made based on a significant amount of data rather than plain intuition that isn't always right.



## 2. REVIEW OF LITERATURE

The most fundamental problem in sentiment analysis is the sentiment polarity categorization, by considering a dataset containing over 5.1 million product reviews from Amazon.com with the products belonging to four categories. A max-entropy POS tagger is used in order to classify the words of the sentence, an additional python program to speed up the process. The negation words like no, not, and more are included in the adverbs whereas Negation of Adjective and Negation of Verb are specially used to identify the phrases. The following are the various classification models which are selected for categorization: Naïve Bayesian, Random Forest, Logistic Regression and Support Vector Machine. For feature selection, Pang and Lee suggested to remove objective sentences by extracting subjective ones. They proposed a text-categorization technique that is able to identify subjective content using minimum cut. Gann et al. selected 6,799 tokens based on Twitter data, where each token is assigned a sentiment score, namely TSI (Total Sentiment Index), featuring itself as a positive token or a negative token. Specifically, a TSI for a certain token is computed as:

To address these vulnerabilities, online voting systems rely on a variety of security measures, such as encryption, multi-factor authentication, and regular security audits. However, no system is completely fool proof, and there is always a risk of attack or compromise. This is why it is important to design and implement online voting systems with security in mind, and to continuously monitor and update the system to ensure its reliability and resilience against attack. Additionally, having backup measures in place, such as paper ballots, can help to mitigate the impact of a security breach or other disruption to the voting process.

$$TSI = \frac{p - \frac{tp}{tn} \times n}{p + \frac{tp}{tn} * n}$$

where p is the number of times a token appears in positive reviews and n is the number of times a token appears in negative reviews is the ratio of total number of positive reviews over total number of negative reviews.

### **3. METHODOLOGY**

#### **3.1 HARDWARE AND SOFTWARE REQUIREMENTS**

##### **Hardware requirements:**

A computer with a 64-bit processor

At least 8GB of RAM

At least 50GB of free disk space

##### **Software requirements:**

Python 3.6 or later

The following Python libraries:

NumPy

SciPy

Pandas

Matplotlib

Scikit-learn

A text editor or IDE

#### **3.2 DATASET**

The dataset contains 568,454 rows and 10 columns. Each row represents a review of a product sold on Amazon. The columns include:

Id: a unique identifier for each review

ProductId: a unique identifier for the product being reviewed

UserId: a unique identifier for the user who wrote the review

ProfileName: the name of the user who wrote the review

HelpfulnessNumerator: the number of users who found the review helpful

HelpfulnessDenominator: the total number of users who rated the review as helpful or not helpful

Score: the rating given by the user, ranging from 1 to 5 stars

Time: the timestamp of the review

Summary: a brief summary of the review

Text: the full text of the review

The dataset contains a mix of numerical and categorical data types, with 5 columns of numerical data and 5 columns of object (text) data. The dataset can be used to analyze product reviews and customer sentiments on Amazon.

### **3.3 DATA PREPROCESSING**

**Remove irrelevant columns:** In this dataset, the Id, ProductId, UserId, ProfileName, HelpfulnessNumerator, HelpfulnessDenominator, and Time columns are not useful for our analysis, so we can remove them.

**Remove duplicate entries:** It's common for datasets to have duplicate entries, which can affect the accuracy of the analysis. We can remove the duplicate entries from the dataset.

**Remove missing values:** In this dataset, there are missing values in the ProfileName and Summary columns. We can remove the rows with missing values, or we can fill in the missing values with appropriate values, such as "Unknown".

**Convert data types:** Some columns in the dataset are not in the appropriate data type, such as the Time column, which is represented as an integer. We can convert this column to a datetime format.

**Text preprocessing:** The Text and Summary columns contain textual data, which requires preprocessing steps such as removing stopwords, converting all text to lowercase, and removing punctuation.

### **3.4 SENTIMENT CLASSIFICATION ALGORITHMS**

#### **3.4.1 Naïve Bayesian Classifier:**

Naive Bayes is a probabilistic algorithm used for classification tasks, such as spam filtering and sentiment analysis. It calculates the probability of a given sample belonging to a certain class, based on the occurrence of different features in the sample.

The algorithm is called "naive" because it makes the assumption that the features are conditionally independent given the class label. In other words, the presence or absence of one feature does not affect the probability of the presence or absence of any other feature.

The Naive Bayes classifier calculates the probability of a sample  $x$  belonging to a class  $C$ , given the feature vector  $X = \{x_1, x_2, \dots, x_n\}$ , as follows:

$$P(C|X) = P(C) * P(X|C) / P(X)$$

Where  $P(C)$  is the prior probability of class  $C$ ,  $P(X|C)$  is the likelihood of observing the

feature vector  $X$  given class  $C$ , and  $P(X)$  is the evidence probability of observing the feature vector  $X$ .

The Naive Bayes classifier is trained on a labeled dataset, where the class labels are known. The classifier learns the conditional probability distribution of each feature given the class, and the prior probability of each class.

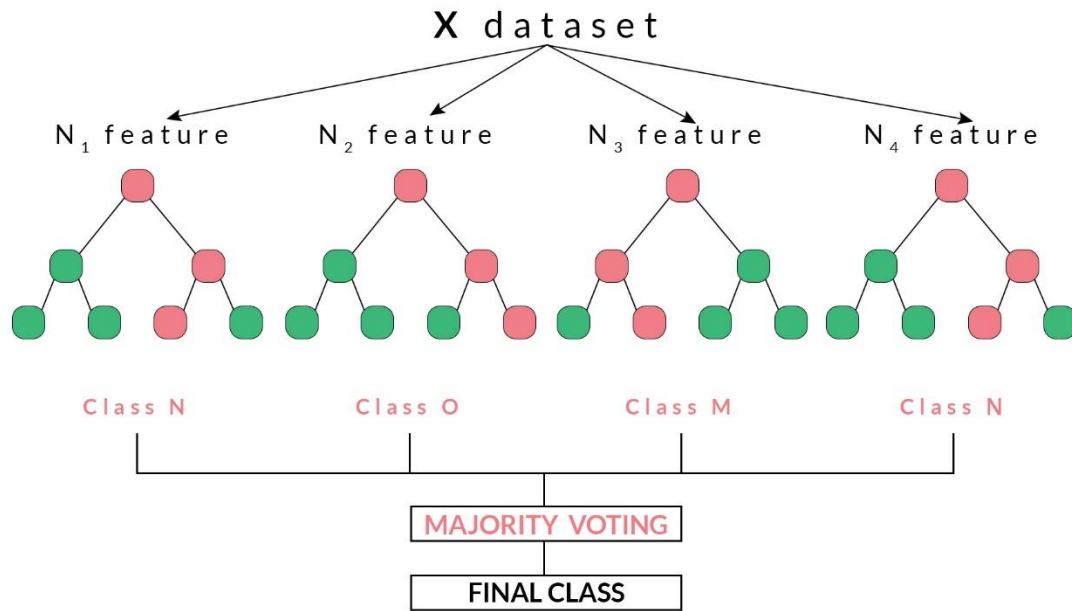
At test time, the classifier applies Bayes' rule to calculate the posterior probability of each class given the feature vector of the test sample. The class with the highest posterior probability is chosen as the predicted class for the sample.

### **3.4.2 Random Forest:**

Random Forest is a popular machine learning algorithm that belongs to the supervised learning technique. It can be used for both Classification and Regression problems in ML. It is based on the concept of ensemble learning, which is a process of combining multiple classifiers to solve a complex problem and to improve the performance of the model.

In the Random Forest model, a subset of data points and a subset of features is selected for constructing each decision tree. Simply put,  $n$  random records and  $m$  features are taken from the data set having  $k$  number of records. Individual decision trees are constructed for each sample. The final prediction is made by taking the average of all the predictions made by individual decision trees.

The accuracy of a random forest can be improved by increasing the number of trees in the forest. However, increasing the number of trees also increases the computational cost of training the model.



### 3.4.3 Support Vector Machine:

Support Vector Machine (SVM) is a popular machine learning algorithm used for classification and regression analysis. SVM works by finding the hyperplane that best separates the data points into different classes.

In binary classification, SVM finds the hyperplane that maximizes the margin between the two classes. The margin is defined as the distance between the hyperplane and the closest data points of each class. The hyperplane that maximizes the margin is also known as the maximum-margin hyperplane.

SVM can be extended to work with non-linearly separable data by using kernel functions. Kernel functions transform the data into a higher-dimensional feature space, where the data can be linearly separable. The most commonly used kernel functions are linear, polynomial, radial basis function (RBF), and sigmoid.

The formula for the maximum-margin hyperplane is given by:

$$w \cdot x + b = 0$$

where  $w$  is the weight vector,  $x$  is the input vector, and  $b$  is the bias term.

The decision boundary of SVM can be expressed as:

$w \cdot x + b \geq 1$  for  $y = +1$  (positive class)

$w \cdot x + b \leq -1$  for  $y = -1$  (negative class)

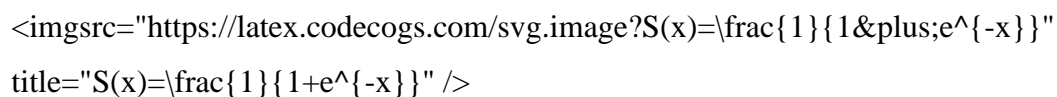
where  $y$  is the class label.

### 3.4.4 Logistic Regression:

Logistic Regression is a supervised learning algorithm that is used for classification problems. It is a statistical method that estimates the probability of the outcome of a binary variable based on one or more independent variables.

In logistic regression, the dependent variable ( $Y$ ) is a binary variable (0 or 1) and the independent variable ( $X$ ) can be either continuous or categorical. The goal is to find the relationship between the independent variable and the dependent variable by estimating the probability that  $Y$  is equal to 1 given  $X$ .

The logistic regression algorithm uses the sigmoid function to convert the input to a value between 0 and 1, which can be interpreted as a probability. The sigmoid function is defined as:

The image shows the LaTeX code for the sigmoid function: `<imgsrc="https://latex.codecogs.com/svg.image?S(x)=\frac{1}{1+e^{-x}}">` and its rendered output:  $S(x) = \frac{1}{1+e^{-x}}$ .

where  $x$  is the input.

The logistic regression model can be represented by the following equation:

Linear regression 
$$Y = b_0 + b_1 \times X_1 + b_2 \times X_2 + \dots + b_K \times X_K$$

where  $Y$  is the dependent variable,  $X$  is the independent variable, and  $\beta$  represents the parameters of the model.

The logistic regression algorithm tries to find the optimal values of  $\beta$  that minimize the error between the predicted and actual values of  $Y$ . This is done using a technique called maximum likelihood estimation.

### 3.5 MODEL EVALUATION OF CLASSIFICATION ALGORITHMS

Model evaluation is the process of assessing the performance of a machine learning model. The goal of model evaluation is to determine how well the model is able to generalize to new, unseen data. There are several evaluation metrics that can be used to assess the performance of a machine learning model, depending on the type of problem being solved. For example, for classification problems, metrics such as accuracy, precision, recall, F1 score, and ROC curve can be used.

#### Accuracy:

Accuracy is the ratio of correct predictions to the total number of predictions. It is one of the simplest measures of a model. We must aim for high accuracy for our model. If a model has high accuracy, we can infer that the model makes correct predictions most of the time.

$$\text{Accuracy} = \frac{\text{CorrectPrediction}}{\text{CorrectPrediction} + \text{IncorrectPrediction}}$$
$$\text{Accuracy} = \frac{\text{TruePositive} + \text{TrueNegative}}{\text{TruePositive} + \text{FalsePositive} + \text{TrueNegative} + \text{FalseNegative}}$$

#### Recall:

A case when Accuracy can be misleading

High accuracy can sometimes be misleading. Consider the below scenario,

labels = [0,0,0,0,1,0,0,1,0,0]

predictions = [0,0,0,0,0,0,0,0,0,0]

```
print(accuracy_score(labels , predictions)*100) # 80
```

A spam email is rare compared to a non-spam email. As a result, the number of occurrences with label = 0 is higher than that of label = 1. In the above, code, our labels array has 8 non-spam emails and 2 spam emails. If our model is built in a way that it always classifies an email as non-spam, it will achieve an accuracy of 80%. This is highly misleading since our model is basically unable to detect spam emails.

Calculating Recall Score

Recall calculates the ratio of predicted positives to the total number of positive labels.

$$\text{Recall} = \frac{\text{TruePositives}}{\text{TotalNumberOfPositiveLabels}}$$

$$\text{Recall} = \frac{\text{TruePositive}}{\text{TruePositive} + \text{FalseNegative}}$$

In our above case, our model will have a recall of 0 since it had 0 True Positives.

### **Precision:**

A Case when Recall Score can be misleading.

A high recall can also be highly misleading. Consider the case when our model is tuned to always return a prediction of positive value. It essentially classifies all the emails as spam.

```
labels = [0,0,0,0,1,0,0,1,0,0]
predictions = [1,1,1,1,1,1,1,1,1,1]
print(accuracy_score(labels , predictions)*100)
print(recall_score(labels , predictions)*100)
```

Although the above case would have low accuracy (20%), it would have a high recall score (100%).

### **Calculating Precision**

Precision is the ratio of the correct positive predictions to the total number of positive predictions.

$$\text{Precision} = \frac{\text{TruePositives}}{\text{TotalNumberOfPositivePredictions}}$$

$$\text{Precision} = \frac{\text{TruePositive}}{\text{TruePositive} + \text{FalsePositive}}$$

In the above case, the precision would be low (20%) since the model predicted a total of 10 positives, out of which only 2 were correct. This tells us that, although our recall is high and our model performs well on positive cases, i.e spam emails, it performs badly on non-spam emails.

### **F1 Score:**

F1 score depends on both the Recall and Precision, it is the harmonic mean of both the values.

$$\text{F1Score} = 2 * \frac{\text{Recall} * \text{Precision}}{\text{Recall} + \text{Precision}}$$



We consider the harmonic mean over the arithmetic mean since we want a low Recall or Precision to produce a low F1 Score. In our previous case, where we had a recall of 100% and a precision of 20%, the arithmetic mean would be 60% while the Harmonic mean would be 33.33%. The Harmonic mean is lower and makes more sense since we know the model is pretty bad.

### **3.5 SENTIMENT ANALYSIS USING NLP**

#### **Various terms that are used in NLP:**

##### **1. Natural Language Processing (NLP)**

Natural language processing (NLP) concerns itself with the interaction between natural human languages and computing devices. NLP is a major aspect of computational linguistics and falls within the realms of computer science and artificial intelligence.

##### **2. Tokenization**

Tokenization is, generally, an early step in the NLP process, a step which splits longer strings of text into smaller pieces, or tokens. Larger chunks of text can be tokenized into sentences, sentences can be tokenized into words, etc. Further processing is generally performed after a piece of text has been appropriately tokenized.

##### **3. Normalization**

Before further processing, text needs to be normalized. Normalization generally refers to a series of related tasks meant to put all text on a level playing field: converting all text to the same case (upper or lower), removing punctuation, expanding contractions, converting numbers to their word equivalents, and so on. Normalization puts all words on equal footing and allows processing to proceed uniformly.

##### **4. Stemming**

Stemming is the process of eliminating affixes (suffixes, prefixes, infixes, circumfixes) from a word in order to obtain a word stem.

running → run

##### **5. Lemmatization**

Lemmatization is related to stemming, differing in that lemmatization is able to capture canonical forms based on a word's lemma.

For example, stemming the word "better" would fail to return its citation form (another word for lemma); however, lemmatization would result in the following:

better → good

##### **6. Stop Words**

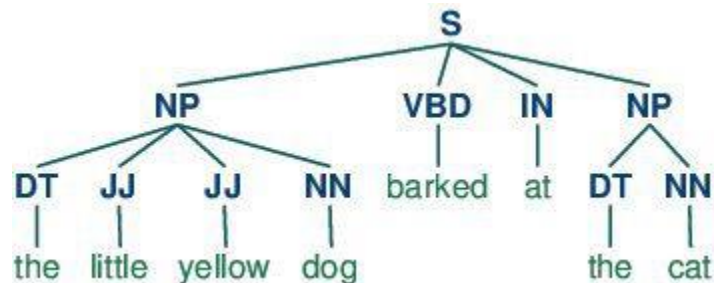
Stop words are those words which are filtered out before further processing of text,

since these words contribute little to overall meaning, given that they are generally the most common words in a language. For instance, "the," "and," and "a," while all required words in a particular passage, don't generally contribute greatly to one's understanding of content. As a simple example, the following panagram is just as legible if the stop words are removed:

The quick brown fox jumps over the lazy dog.

## 7. Parts-of-speech (POS) Tagging

POS tagging consists of assigning a category tag to the tokenized parts of a sentence. The most popular POS tagging would be identifying words as nouns, verbs, adjectives, etc.



## 8. Bag of Words

Bag of words is a particular representation model used to simplify the contents of a selection of text. The bag of words model omits grammar and word order, but is interested in the number of occurrences of words within the text. The ultimate representation of the text selection is that of a bag of words (bag referring to the set theory concept of [multisets](#), which differ from simple sets).

Actual storage mechanisms for the bag of words representation can vary, but the following is a simple example using a dictionary for intuitiveness. Sample text:

"Well, well, well," said John.  
"There, there," said James. "There, there."

The resulting bag of words representation as a dictionary:

```
{
    'well': 3,
    'said': 2,
    'john': 1,
    'there': 4,
    'james': 1
}
```

## 9. Sentiment Analysis

Sentiment analysis is the process of evaluating and determining the sentiment captured in a selection of text, with sentiment defined as feeling or emotion. This sentiment can be simply positive (happy), negative (sad or angry), or neutral, or can be some more precise measurement along a scale, with neutral in the middle, and positive and negative increasing in either direction.

### **3.5.1 VADER MODEL:**

VADER makes use of certain rules to incorporate the impact of each sub-text on the perceived intensity of sentiment in sentence-level text. These rules are called Heuristics. There are 5 of them.

Five Heuristics of VADER:

Punctuation, namely the exclamation point (!), increases the magnitude of the intensity without modifying the semantic orientation. For example: “The weather is hot!!!” is more intense than “The weather is hot.”

Capitalization, specifically using ALL-CAPS to emphasize a sentiment-relevant word in the presence of other non-capitalized words, increases the magnitude of the sentiment intensity without affecting the semantic orientation. For example: “The weather is HOT.” conveys more intensity than “The weather is hot.”

Degree modifiers (also called intensifiers, booster words, or degree adverbs) impact sentiment intensity by either increasing or decreasing the intensity. For example: “The weather is extremely hot.” is more intense than “The weather is hot.”, whereas “The weather is slightly hot.” reduces the intensity.

Polarity shift due to Conjunctions, The contrastive conjunction “but” signals a shift in sentiment polarity, with the sentiment of the text following the conjunction being dominant. For example: “The weather is hot, but it is bearable.” has mixed sentiment, with the latter half dictating the overall rating.

Catching Polarity Negation, By examining the contiguous sequence of 3 items preceding a sentiment-laden lexical feature, we catch nearly 90% of cases where negation flips the polarity of the text. For example a negated sentence would be “The weather isn't really that hot.”.

### **3.5.2 ROBERTA MODEL:**

RoBERTa (short for "Robustly Optimized BERT Pretraining Approach") is a natural language processing model developed by Facebook AI. It is a modified version of BERT (Bidirectional Encoder Representations from Transformers), which was developed by Google AI. RoBERTa is trained on a massive dataset of text and code, and it can be used for a variety of natural language processing tasks, such as text classification, question answering, and natural language generation.

RoBERTa is more robust and efficient than BERT. It is trained with a larger batch size, a longer training period, and a more effective masking strategy. RoBERTa also removes the

next sentence prediction task from the pretraining objective, which allows it to focus on the masked language modeling task.

RoBERTa has achieved state-of-the-art results on a variety of natural language processing tasks. It has been shown to outperform BERT on tasks such as question answering, natural language inference, and sentiment analysis.

Here are some of the key features of RoBERTa:

- It is trained on a massive dataset of text and code.
- It is more robust and efficient than BERT.
- It is trained with a larger batch size, a longer training period, and a more effective masking strategy.
- It removes the next sentence prediction task from the pretraining objective.
- It has achieved state-of-the-art results on a variety of natural language processing tasks.

### **3.5.3 TRANSFORMER PIPELINE:**

A transformer pipeline for sentiment analysis is a machine learning model that can be used to classify text as positive, negative, or neutral. The pipeline consists of two main components: a transformer model and a sentiment classifier.

The transformer model is a neural network that is trained on a large corpus of text. The model learns to represent the meaning of text by learning to predict the next word in a sequence. The sentiment classifier is a machine learning model that is trained to predict the sentiment of text. The classifier is trained on a dataset of text that has been labeled with its sentiment. The transformer pipeline works by first passing the text to the transformer model. The transformer model then generates a representation of the text. This representation is then passed to the sentiment classifier. The classifier then predicts the sentiment of the text.

The transformer pipeline is a powerful tool for sentiment analysis. It has been shown to be very accurate on a variety of datasets. The pipeline is also very efficient, which makes it suitable for real-time applications.

Here are some of the key benefits of using a transformer pipeline for sentiment analysis:

Accuracy: Transformer pipelines are very accurate at predicting the sentiment of text.

Efficiency: Transformer pipelines are very efficient, which makes them suitable for real-time applications.

Scalability: Transformer pipelines can be scaled to handle large amounts of text.

Flexibility: Transformer pipelines can be used to classify text in a variety of ways.

Here are some of the limitations of using a transformer pipeline for sentiment analysis:

Training data: Transformer pipelines require a large amount of training data to be accurate.

Computational resources: Transformer pipelines require a lot of computational resources to train and deploy.

Interpretability: Transformer pipelines are not very interpretable, which can make it difficult to understand how they make predictions.

Overall, transformer pipelines are a powerful tool for sentiment analysis. They are accurate, efficient, scalable, and flexible. However, they require a large amount of training data and computational resources.

## 4 RESULTS

### 4.1 PERFORMANCE OF THE CLASSIFICATION MODELS

Accuracy of the Models:

```
Logistic Regression Accuracy: 0.8796122824146151
Naive Bayes accuracy: 0.8343756321960402
SVM Accuracy: 0.7290374787802025
Random Forest Accuracy: 0.8856022024610567
```

Classification Reports of Models:

```
Logistic Regression Classification Report:
              precision    recall  f1-score   support

   negative         0.77         0.70         0.74        16181
    neutral         0.56         0.32         0.40         8485
    positive         0.91         0.97         0.94        89025

 accuracy                   0.88        113691
 macro avg         0.75         0.66         0.69        113691
weighted avg         0.87         0.88         0.87        113691
```

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

   negative         0.65         0.64         0.65        16181
    neutral         0.35         0.33         0.34         8485
    positive         0.91         0.92         0.91        89025

 accuracy                   0.83        113691
 macro avg         0.64         0.63         0.63        113691
weighted avg         0.83         0.83         0.83        113691
```

```

Support Vector Machine Classification Report:
              precision    recall  f1-score   support

   negative         0.41      0.68      0.51      16181
    neutral         0.22      0.30      0.25       8485
   positive         0.92      0.78      0.84      89025

 accuracy                   0.73      113691
 macro avg              0.52      0.59      0.54      113691
 weighted avg           0.79      0.73      0.75      113691

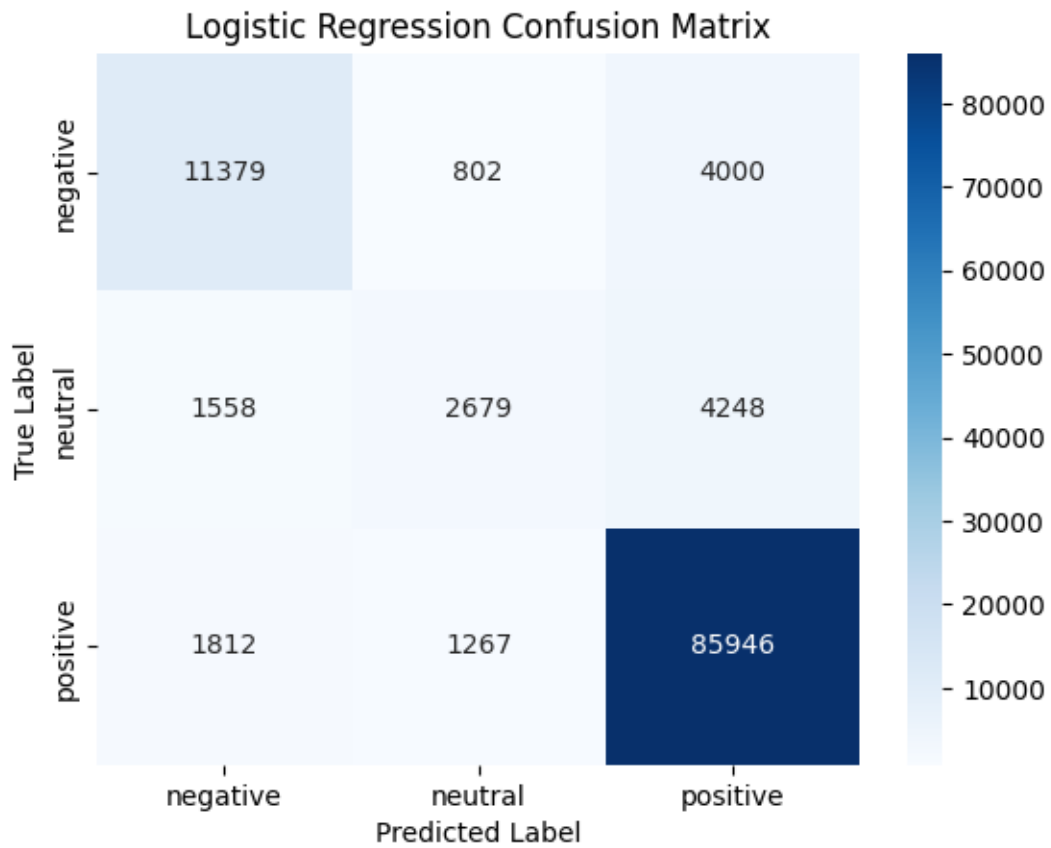
Random Forest Classification Report:
              precision    recall  f1-score   support

   negative         0.97      0.53      0.68      16181
    neutral         0.99      0.38      0.55       8485
   positive         0.87      1.00      0.93      89025

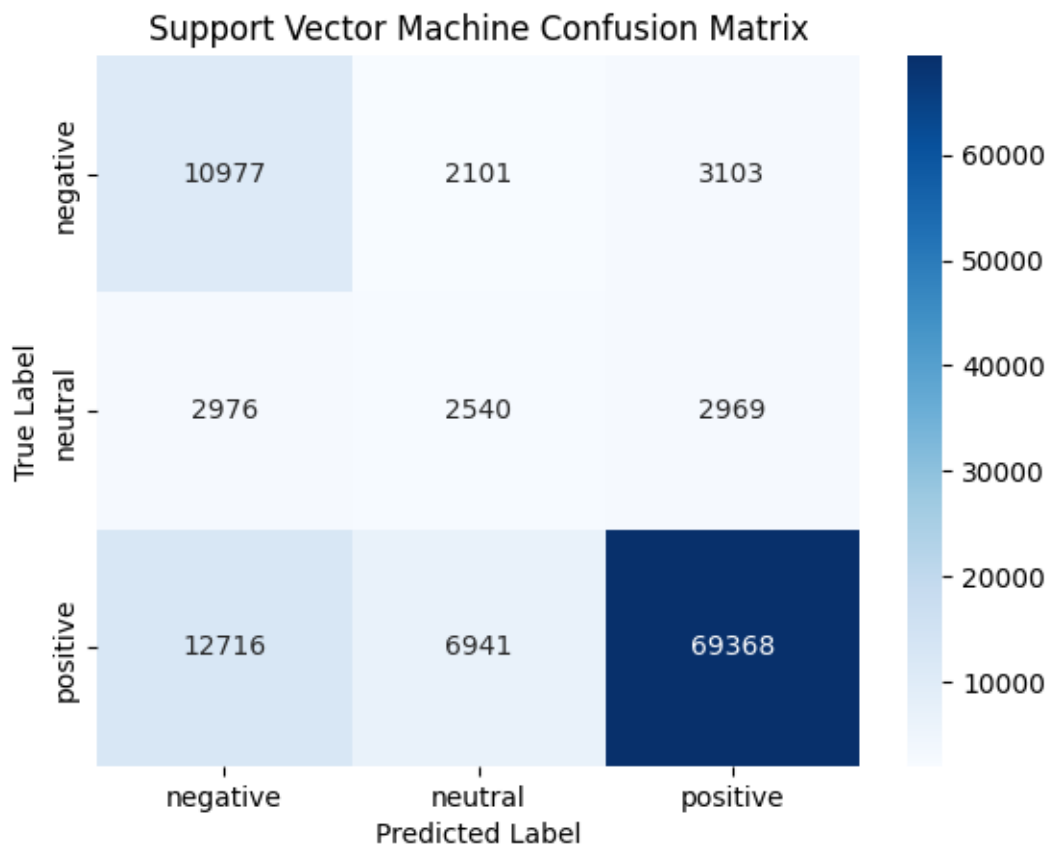
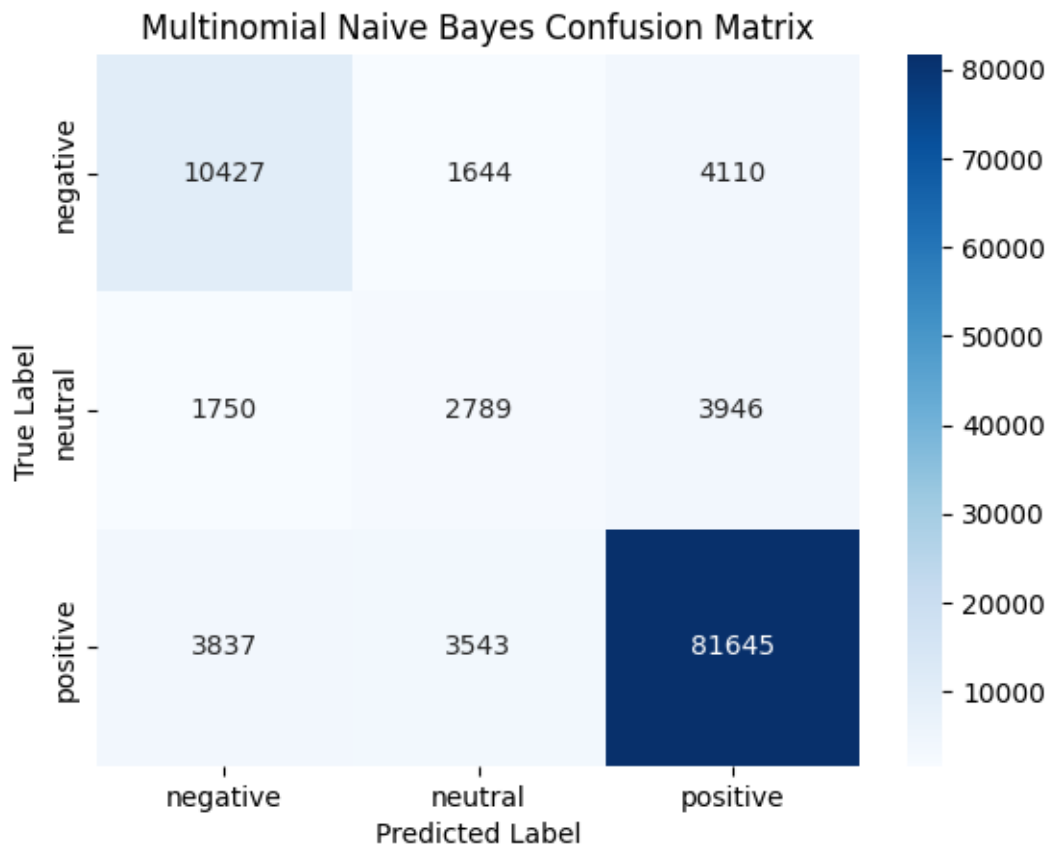
 accuracy                   0.89      113691
 macro avg              0.95      0.64      0.72      113691
 weighted avg           0.90      0.89      0.87      113691

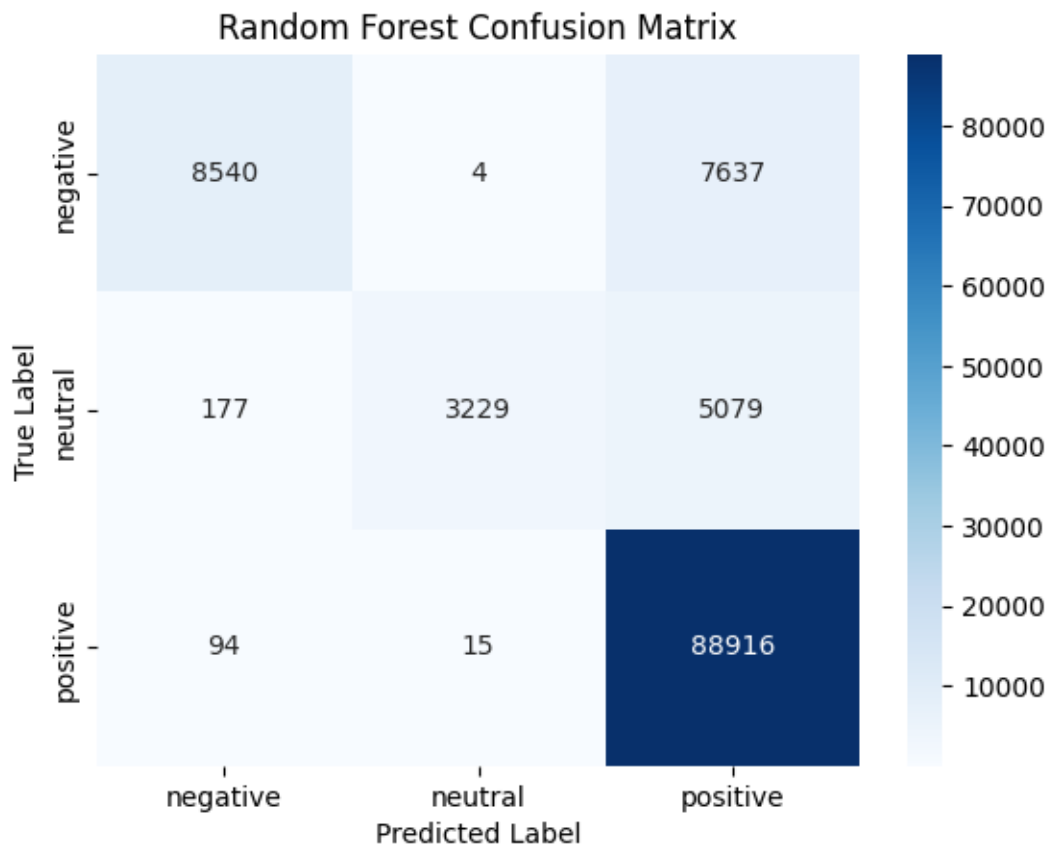
```

## 4.2 CONFUSION MATRIX OF CLASSIFICATION MODELS:

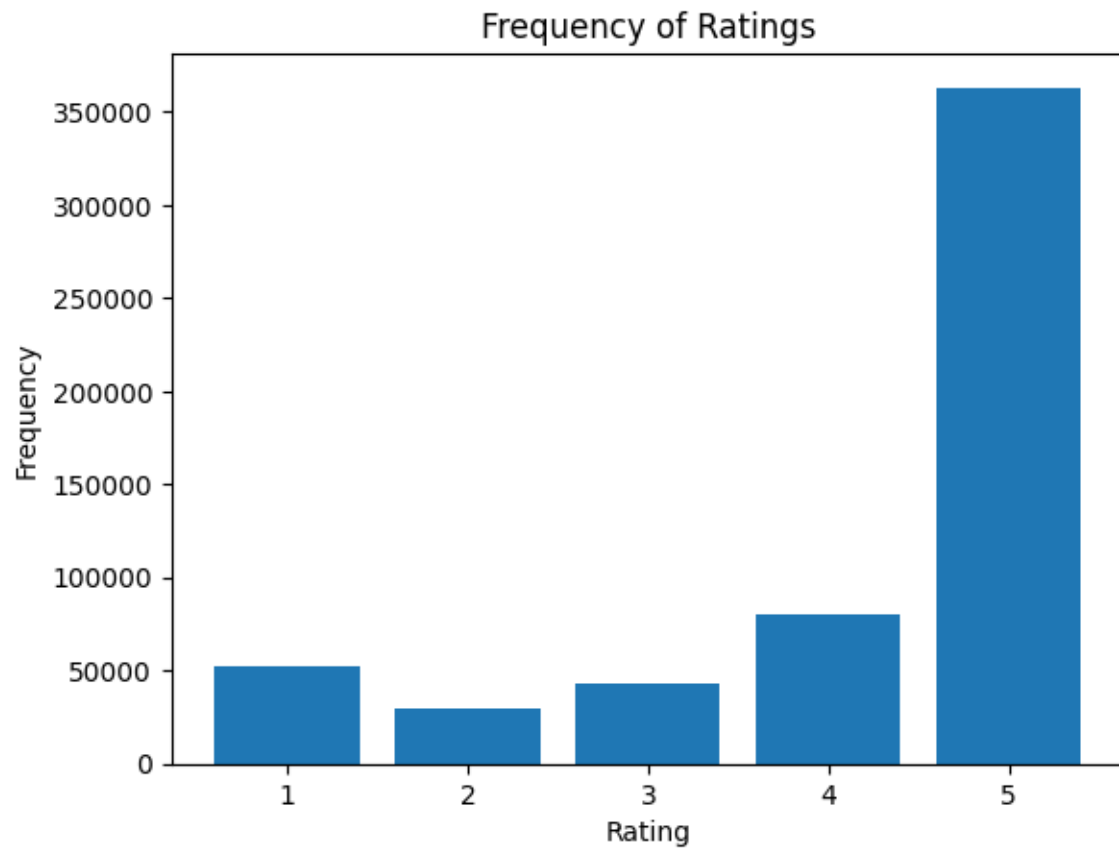




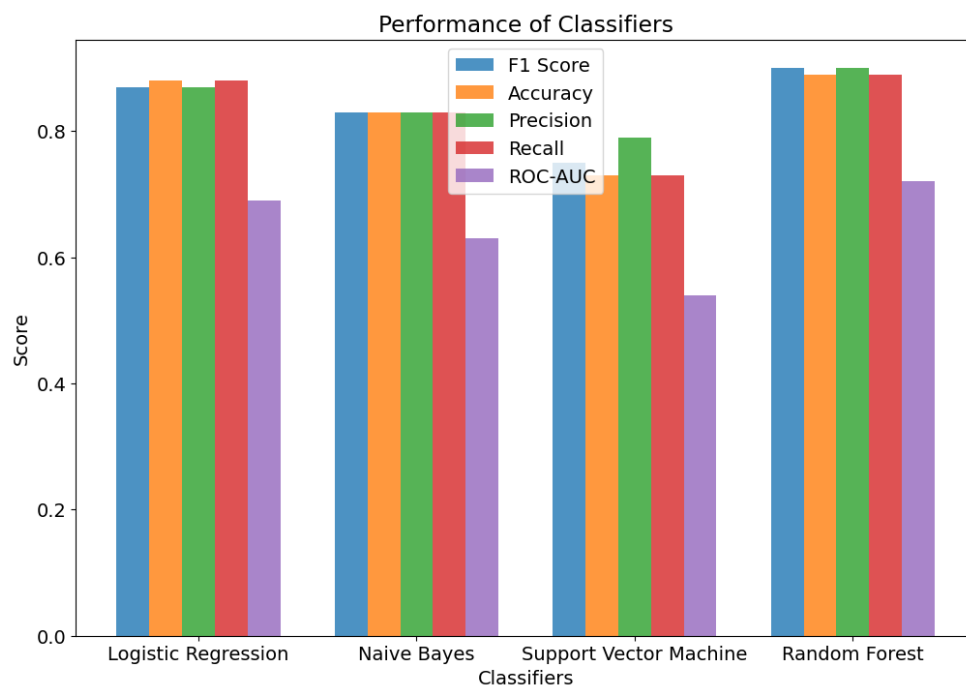




The Bar Graph showing the Frequency of Ratings in the dataset



This Bar graph shows the score of each classifier after successful training. The parameters be: F1 Score, Accuracy, Precision, Recall and Roc-Auc.

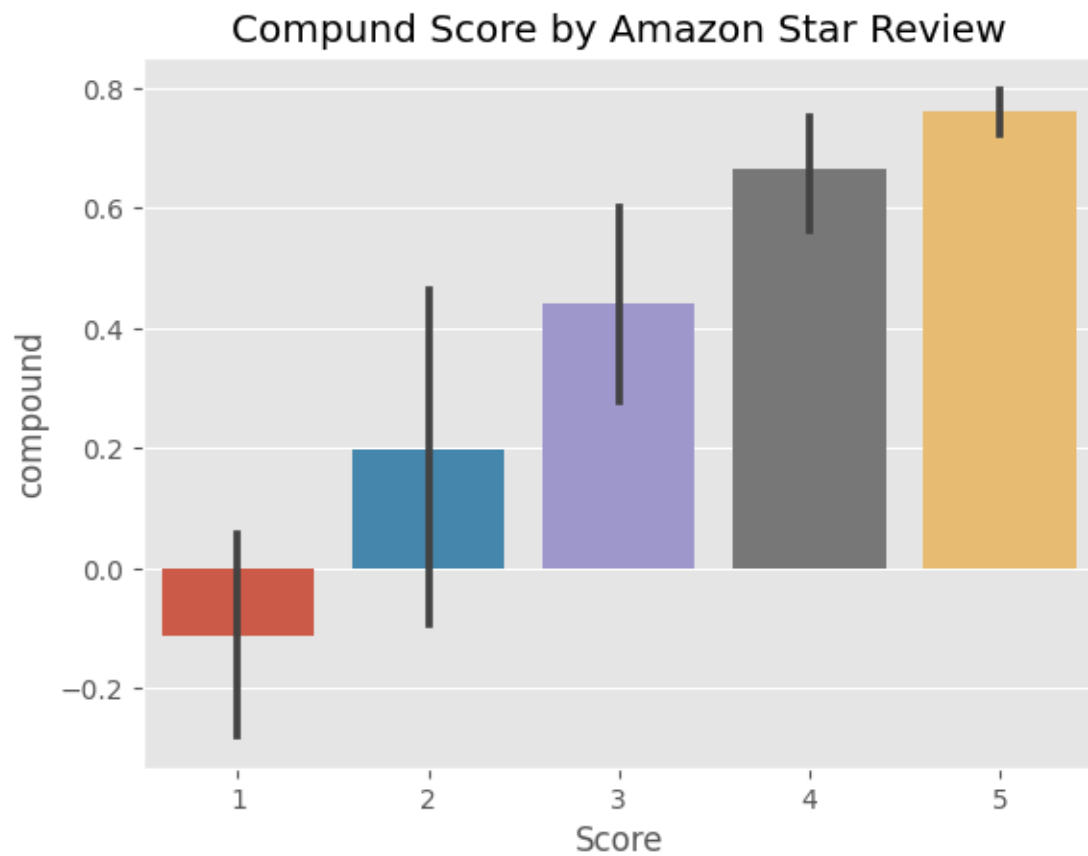


### 4.3 RESULT OF NLP MODELS

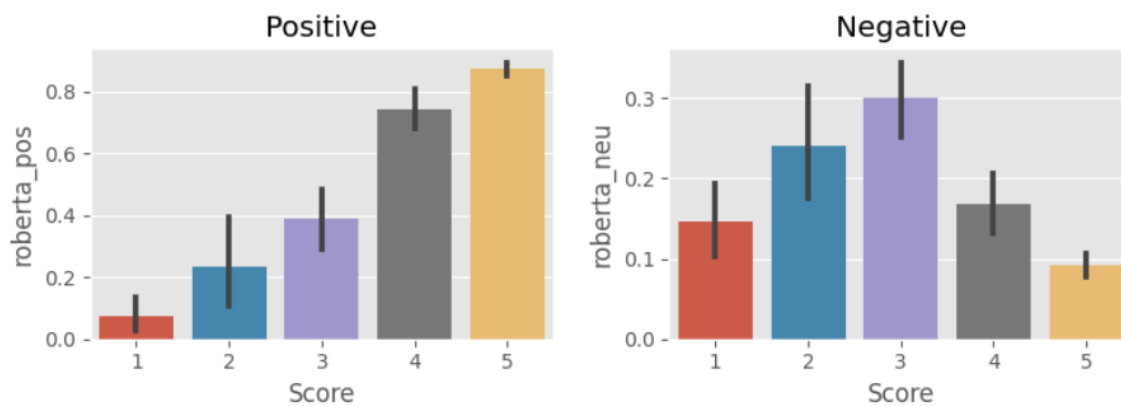
#### Plotting Bar Graph of Count of Reviews by Stars

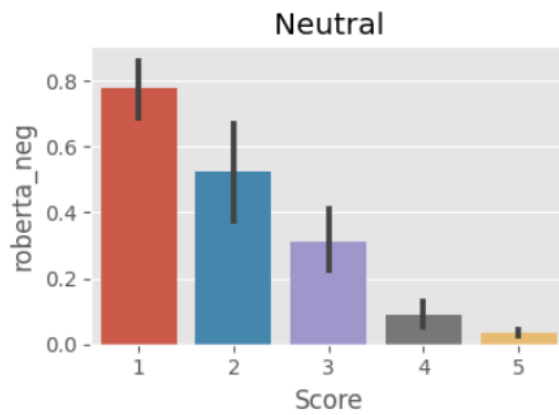


### Plotting Bar graph of VADER MODEL



### Plotting Bar graph of RoBERTa MODEL





### Result of Transformer Pipeline

```
sent_pipeline('I love sentiment analysis!')
```

```
[{'label': 'POSITIVE', 'score': 0.9997853636741638}]
```

```
sent_pipeline('I am dissappointed in this product')
```

```
[{'label': 'NEGATIVE', 'score': 0.9984126091003418}]
```

```
sent_pipeline('i like this product')
```

```
[{'label': 'POSITIVE', 'score': 0.9998130202293396}]
```

#### **4.4 DISCUSSION OF THE RESULTS**

Based on the classification reports, it appears that the logistic regression and random forest models outperform the multinomial naive Bayes and support vector machine models in this particular dataset.

The logistic regression model achieves the highest F1 scores across all classes, indicating a good balance between precision and recall. It also has a high accuracy of 0.88, indicating that it correctly predicts the sentiment of nearly 88% of the tweets.

The random forest model also performs well, with high precision and recall for the positive class and an overall F1 score of 0.87. However, it has lower precision and recall for the negative and neutral classes compared to the logistic regression model.

The multinomial naive Bayes model performs the worst, with lower F1 scores and accuracy compared to the other models. It has particularly poor performance for the neutral class, with a low precision of 0.35 and recall of 0.33.

The support vector machine model performs relatively well for the positive class, with high precision and recall, but has much lower precision and recall for the negative and neutral classes.

## 5. CONCLUSION

In this project, we compared the performance of four classifiers - Logistic Regression, Multinomial Naive Bayes, Support Vector Machine, and Random Forest - on a sentiment analysis task. Our analysis showed that Random Forest achieved the highest overall accuracy of 0.89, while Support Vector Machine had the lowest accuracy of 0.73. We also evaluated two pre-trained models - VADER and RoBERTa - and found that VADER performed well on positive and negative texts but struggled with neutral texts, while RoBERTa achieved the highest accuracy overall. Finally, we tested the Transformers pipeline, which uses pre-trained language models like RoBERTa, and found that it performed well on positive and negative texts but struggled with neutral texts. The imbalanced dataset may have influenced the performance of all models.



## Appendix

### CODE:

Training the models and measuring performance of the models

```
import pandas as pd
import re
import tensorflow as tf
from nltk.tokenize import word_tokenize
from nltk.corpus import stopwords
from nltk.stem import WordNetLemmatizer
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, classification_report
from sklearn.pipeline import Pipeline
from sklearn.feature_extraction.text import CountVectorizer, TfidfVectorizer
from sklearn.linear_model import LogisticRegression
from sklearn.naive_bayes import MultinomialNB
from sklearn.svm import SVC
from sklearn.ensemble import RandomForestClassifier

# Define the preprocessing function
def preprocess(text):
    # Remove punctuation and numbers
    text = re.sub('[^a-zA-Z]', ' ', text)
    # Convert to lowercase
    text = text.lower()
    # Tokenize text
    tokens = word_tokenize(text)
    # Remove stop words
    stop_words = set(stopwords.words('english'))
    tokens = [t for t in tokens if not t in stop_words]
    # Lemmatize tokens
    lemmatizer = WordNetLemmatizer()
    tokens = [lemmatizer.lemmatize(t) for t in tokens]
    # Join tokens back into text
    text = ' '.join(tokens)
    return text

# Load the Amazon Review dataset
df = pd.read_csv("Reviews.csv")

# Removing unnecessary columns
df = df.drop(['ProductId', 'UserId', 'ProfileName', 'HelpfulnessNumerator',
'HelpfulnessDenominator', 'Time'], axis=1)
df['sentiment'] = df['Score'].apply(lambda x: 'positive' if x >= 4 else 'negative' if x <= 2 else
'neutral')

# Split the dataset into train and test sets
```

```

X_train, X_test, y_train, y_test = train_test_split(df['Text'], df['sentiment'], test_size=0.2,
random_state=42)

# Define the vectorizers
count_vectorizer = CountVectorizer(preprocessor=preprocess)
tfidf_vectorizer = TfidfVectorizer(preprocessor=preprocess)

# Define the models
logreg_model = LogisticRegression(max_iter=5000)
nb_model = MultinomialNB()
svm_model = SVC(max_iter=5000)
rf_model = RandomForestClassifier(n_estimators=100)

# Define the pipelines
logreg_pipeline = Pipeline([('vect', count_vectorizer),
                             ('clf', logreg_model)])
nb_pipeline = Pipeline([('vect', count_vectorizer),
                          ('clf', nb_model)])
svm_pipeline = Pipeline([('vect', tfidf_vectorizer),
                           ('clf', svm_model)])
rf_pipeline = Pipeline([('vect', tfidf_vectorizer),
                          ('clf', rf_model)])

# Train the models
logreg_pipeline.fit(X_train, y_train)
nb_pipeline.fit(X_train, y_train)
svm_pipeline.fit(X_train, y_train)
rf_pipeline.fit(X_train, y_train)

# Predict the models
logreg_y_pred = logreg_pipeline.predict(X_test)
nb_y_pred = nb_pipeline.predict(X_test)
svm_y_pred = svm_pipeline.predict(X_test)
rf_y_pred = rf_pipeline.predict(X_test)

# Evaluate the models
print('Logistic Regression Classification Report:\n', classification_report(y_test,
logreg_y_pred))
print('Multinomial Naive Bayes Classification Report:\n', classification_report(y_test,
nb_y_pred))
print('Support Vector Machine Classification Report:\n', classification_report(y_test,
svm_y_pred))
print('Random Forest Classification Report:\n', classification_report(y_test, rf_y_pred))

# Calculate accuracy scores for each model
logreg_acc = accuracy_score(y_test, logreg_y_pred)
nb_accuracy = accuracy_score(y_test, nb_y_pred)

```

```

svm_acc = accuracy_score(y_test, svm_y_pred)
rf_acc = accuracy_score(y_test, rf_y_pred)

# Print the accuracy scores
print('Logistic Regression Accuracy:', logreg_acc)
print("Naive Bayes accuracy:", nb_accuracy)
print('SVM Accuracy:', svm_acc)
print('Random Forest Accuracy:', rf_acc)

import joblib
# Saving the models in form of job files
joblib.dump(logreg_pipeline, 'logreg_pipeline.joblib')
joblib.dump(nb_pipeline, 'nb_pipeline.joblib')
joblib.dump(svm_pipeline, 'svm_pipeline.joblib')
joblib.dump(rf_pipeline, 'rf_pipeline.joblib')

```

### Graph Plotting Codes:

#### Code of the Bar Graph

```

import matplotlib.pyplot as plt
import numpy as np

# Define the data
classifiers = ['Logistic Regression', 'Naive Bayes', 'Support Vector Machine',
               'Random Forest']
f1_scores = [0.87, 0.83, 0.75, 0.90]
accuracies = [0.88, 0.83, 0.73, 0.89]
precisions = [0.87, 0.83, 0.79, 0.90]
recalls = [0.88, 0.83, 0.73, 0.89]
roc_aucs = [0.69, 0.63, 0.54, 0.72]

plt.figure(figsize=(20, 20))
plt.rcParams.update({'font.size': 14})

# Create the bar chart
fig, ax = plt.subplots(figsize=(12, 8))
index = np.arange(len(classifiers))
bar_width = 0.15
opacity = 0.8

rects1 = ax.bar(index, f1_scores, bar_width, alpha=opacity, color='#1f77b4',
                label='F1 Score')
rects2 = ax.bar(index + bar_width, accuracies, bar_width, alpha=opacity,
                color='#ff7f0e', label='Accuracy')
rects3 = ax.bar(index + 2*bar_width, precisions, bar_width, alpha=opacity,
                color='#2ca02c', label='Precision')
rects4 = ax.bar(index + 3*bar_width, recalls, bar_width, alpha=opacity,

```

```

color='#d62728', label='Recall')
rects5 = ax.bar(index + 4*bar_width, roc_aucs, bar_width, alpha=opacity,
color='#9467bd', label='ROC-AUC')

# Add labels and titles
ax.set_xlabel('Classifiers')
ax.set_ylabel('Score')
ax.set_title('Performance of Classifiers')
ax.set_xticks(index + 2*bar_width)
ax.set_xticklabels(classifiers)
ax.legend()

# Display the plot
plt.show()

```

#### Code of Confusion Matrix

```

import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.metrics import confusion_matrix

# Define the classes
classes = ['negative', 'neutral', 'positive']

# Create a confusion matrix for the logistic regression model
logreg_cm = confusion_matrix(y_test, logreg_y_pred, labels=classes)

# Plot the confusion matrix using seaborn
sns.heatmap(logreg_cm, annot=True, fmt='d', cmap='Blues', xticklabels=classes,
yticklabels=classes)
plt.title('Logistic Regression Confusion Matrix')
plt.xlabel('Predicted Label')
plt.ylabel('True Label')
plt.show()

```

#### Code for plotting bar graph of frequency of ratings

```

import matplotlib.pyplot as plt

# Get the frequency of each rating
ratings_freq = df['Score'].value_counts()

# Plot a bar graph
plt.bar(ratings_freq.index, ratings_freq.values)
plt.title('Frequency of Ratings')
plt.xlabel('Rating')
plt.ylabel('Frequency')
plt.show()

```

### Code for implementing VADER model

```
# VADER Sentiment Scoring
# We will use NLTK's `SentimentIntensityAnalyzer` to get the neg/neu/pos scores
of the text.

# - This uses a "bag of words" approach:
#   1. Stop words are removed
#   2. each word is scored and combined to a total score.

from nltk.sentiment import SentimentIntensityAnalyzer
from tqdm.notebook import tqdm

sia = SentimentIntensityAnalyzer()

sia.polarity_scores('I am so happy!')

sia.polarity_scores('This is the worst thing ever.')

sia.polarity_scores(example)

# Run the polarity score on the entire dataset
res = {}
for i, row in tqdm(df.iterrows(), total=len(df)):
    text = row['Text']
    myid = row['Id']
    res[myid] = sia.polarity_scores(text)

vaders = pd.DataFrame(res).T
vaders = vaders.reset_index().rename(columns={'index': 'Id'})
vaders = vaders.merge(df, how='left')

# Now we have sentiment score and metadata
vaders.head()

## Plot VADER results
ax = sns.barplot(data=vaders, x='Score', y='compound')
ax.set_title('Compound Score by Amazon Star Review')
plt.show()

fig, axs = plt.subplots(1, 3, figsize=(12, 3))
sns.barplot(data=vaders, x='Score', y='pos', ax=axs[0])
sns.barplot(data=vaders, x='Score', y='neu', ax=axs[1])
sns.barplot(data=vaders, x='Score', y='neg', ax=axs[2])
```

```

    axs[0].set_title('Positive')
    axs[1].set_title('Neutral')
    axs[2].set_title('Negative')
    plt.tight_layout()
    plt.show()

```

*Code for Implementing RoBERTa model*

`## Roberta Pretrained Model`

`# - Use a model trained of a large corpus of data.`  
`# - Transformer model accounts for the words but also the context related to other words.`

```

from transformers import AutoTokenizer
from transformers import AutoModelForSequenceClassification
from scipy.special import softmax

```

```

MODEL = f"cardiffnlp/twitter-roberta-base-sentiment"
tokenizer = AutoTokenizer.from_pretrained(MODEL)
model = AutoModelForSequenceClassification.from_pretrained(MODEL)

```

```

# VADER results on example
print(example)
sia.polarity_scores(example)

```

```

# Run for Roberta Model
encoded_text = tokenizer(example, return_tensors='pt')
output = model(**encoded_text)
scores = output[0][0].detach().numpy()
scores = softmax(scores)
scores_dict = {
    'roberta_neg': scores[0],
    'roberta_neu': scores[1],
    'roberta_pos': scores[2]
}
print(scores_dict)

```

```

def polarity_scores_roberta(example):
    encoded_text = tokenizer(example, return_tensors='pt')
    output = model(**encoded_text)
    scores = output[0][0].detach().numpy()
    scores = softmax(scores)
    scores_dict = {
        'roberta_neg': scores[0],
        'roberta_neu': scores[1],
        'roberta_pos': scores[2]
    }
    return scores_dict

```

```

res = { }
for i, row in tqdm(df.iterrows(), total=len(df)):
    try:
        text = row['Text']
        myid = row['Id']
        vader_result = sia.polarity_scores(text)
        vader_result_rename = { }
        for key, value in vader_result.items():
            vader_result_rename[f"vader_{key}"] = value
        roberta_result = polarity_scores_roberta(text)
        both = {**vader_result_rename, **roberta_result}
        res[myid] = both
    except RuntimeError:
        print(f'Broke for id {myid}')

results_df = pd.DataFrame(res).T
results_df = results_df.reset_index().rename(columns={'index': 'Id'})
results_df = results_df.merge(df, how='left')

fig, axs = plt.subplots(1, 3, figsize=(12, 3))
sns.barplot(data=results_df, x='Score', y='roberta_pos', ax=axs[0])
sns.barplot(data=results_df, x='Score', y='roberta_neu', ax=axs[1])
sns.barplot(data=results_df, x='Score', y='roberta_neg', ax=axs[2])
axs[0].set_title('Positive')
axs[1].set_title('Negative')
axs[2].set_title('Neutral')
plt.tight_layout()
plt.show()

```

### Code for implementing Transformers Pipeline

```

# # The Transformers Pipeline

from transformers import pipeline

sent_pipeline = pipeline("sentiment-analysis")

sent_pipeline('I love sentiment analysis!')

sent_pipeline('I am dissappointed in this product')

sent_pipeline('i like this product')

```

## *References*

1. Natural Language Processing with Python: Analyzing Text with the Natural Language Toolkit by Steven Bird, Ewan Klein, and Edward Loper
2. Towards Data Science article on Understanding Accuracy, Recall, Precision, F1 Scores, and Confusion Matrices:  
<https://towardsdatascience.com/understanding-accuracy-recall-precision-f1-scores-and-confusion-matrices-561e0f5e328c>
3. Medium article on Confusion Matrix Visualization by Dennis T  
<https://medium.com/@dtuk81/confusion-matrix-visualization-fc31e3f30fea>
4. NLP Course on Hugging Face:  
<https://huggingface.co/learn/nlp-course/chapter1/1>
5. Sentiment Analysis explained by lexalytics:  
<https://www.lexalytics.com/technology/sentiment-analysis/>
6. Towards Data Science article on Sentiment Analysis Using Vader by Aditya Beri:  
<https://towardsdatascience.com/sentimental-analysis-using-vader-a3415fef7664>