Abstract

This project addresses the problem of sentiment analysis in twitter; that is classifying tweets according to the sentiment expressed in them: positive, negative or neutral. Twitter is an online micro-blogging and social-networking platform which allows users to write short status updates of maximum length 140 characters. It is a rapidly expanding service with over 200 million registered users - out of which 100 million are active users and half of them log on twitter on a daily basis - generating nearly 250 million tweets per day. Due to this large amount of usage we hope to achieve a reflection of public sentiment by analysing the sentiments expressed in the tweets. Analysing the public sentiment is important for many applications such as firms trying to find out the response of their products in the market, predicting political elections and predicting socioeconomic phenomena like stock exchange. The aim of this project is to develop a functional classifier for accurate and automatic sentiment classification of an unknown tweet stream.

Keywords : NLP-"Natural language Processing", RT-"re-tweet", HTML-"Hypertext Markup Language"

TABLE OF CONTENTS

List of Figures

List of Tables

Chapter 1

Introduction

Sentiment analysis is a set of algorithms and techniques used to detect the sentiment (positive, negative, or neutral) of a given text. This is a very powerful application of NLP. Here are some other examples of use cases of sentiment analysis:

1. A stock investor scanning news about a company to assess overall market sentiment
2. An individual scanning tweets about the launch of a new phone to decide the prevailing sentiment
3. A political party analyzing social media feeds to assess the sentiment regarding their candidate

1.1 Problem Statement

The proposed Project is Twitter Sentiment Analysis using Natural Language Processing (NLP) . The goal of the project is to classify a given tweet/message to whether the message is of positive, negative, or neutral sentiment.

1.1.1 WELCOME

For messages conveying both a positive and negative sentiment, whichever is the stronger sentiment should be chosen. Twitter sentiment analysis allows you to keep track of what's being said about your product or service on social media, and can help you detect angry customers or negative mentions before they turn into a major crisis.

1.2 Thesis Outline

The thesis is organized as follows:

Chapter 1 provides problem statement and a general introduction to the thesis.

Chapter 2 provides the motivation for doing this project and also introduces the necessary background knowledge.

Chapter 3 talks about various work done in past few years on twitter sentiment analysis.

Chapter 4 contains some difficulties and solutions provided to solve them.

Chapter 5 explains methodology or our proposed scheme.

Chapter 6 provides the experimental results of the proposed scheme and it analyses the time complexity of the scheme for various values of the parameters.

Chapter 7 concludes our work and gives the future work which can be done to improve this scheme.

Chapter 2

Motivation

In the past decade, new forms of communication, such as microblogging and text messaging have emerged and become ubiquitous. While there is no limit to the range of information conveyed by tweets and texts, often these short messages are used to share opinions and sentiments that people have about what is going on in the world around them.

2.1 Shortening of Tweets and Texts

Tweets and texts are short: A sentence or a headline rather than a document. The language used is very informal, with creative spelling and punctuation, misspellings, slang, new words, URLs, and genre-specific terminology and abbreviations, such as, RT for "re-tweet" and hashtags, which are a type of tagging for Twitter messages.

2.2 Rich Structured Information

Another aspect of social media data such as Twitter messages is that it includes rich structured information about the individuals involved in the communication. For example, Twitter maintains information of who follows whom and re-tweets and tags inside of tweets provide discourse information.

Chapter 3

Literature Review

3.1 Limitations of Prior Art

Sentiment analysis of in the domain of micro-blogging is a relatively new research topic so there is still a lot of room for further research in this area. Decent amount of related prior work has been done on sentiment analysis of user reviews, documents, web blogs/articles and general phrase level sentiment analysis. These differ from twitter mainly because of the limit of 140 characters per tweet which forces the user to express opinion compressed in very short text. The best results reached in sentiment classification use supervised learning techniques such as Naive Bayes and Support Vector Machines, but the manual labelling required for the supervised approach is very expensive. Some work has been done on unsupervised and semi-supervised approaches, and there is a lot of room of improvement. Various researchers testing new features and classification techniques often just compare their results to base-line performance. There is a need of proper and formal comparisons between these results arrived through different features and classification techniques in order to select the best features and most efficient classification techniques for particular applications.

3.2 Related Work

The bag-of-words model is one of the most widely used feature model for almost all text classification tasks due to its simplicity coupled with good performance. The model represents the text to be classified as a bag or collection of individual words with no link or dependence of one word with the other, i.e. it completely disregards grammar and order of words within the text. This model is also very popular in sentiment analysis and has been used by various researchers. Generally speaking n-grams is a contiguous sequence of “n” words in our text, which is completely independent of any other words or grams in the text. This is a very simplifying assumption but it has been shown to provide rather good performance.

Chapter 4

Analysis And Design

We have analyse the difficulties and have tried to design the solutions for it.

Some of the difficulties we have analysed are as followed.

1.Cleaning the tweets: The tweets given in the data are usually not clean or in a desired structure. So the main difficulty is to clean the data .

2.Word Ambiguity: Word ambiguity is another pitfall we are facing working on the sentiment analysis problem.The problem of word ambiguity is the impossibility to define polarity in advance because the polarity for some words is strongly dependent on the sentence context.

So we will be discussing our solutions for solving such difficulties through some important steps:

4.1 Pre-processing Tweets

This is one of the essential steps in any natural language processing (NLP) task. Following processes were used to clean the tweets. Letter casing: Converting all letters to either upper case or lower case. Tokenizing: Turning the tweets into tokens. Tokens are words separated by spaces in a text. Noise removal: Eliminating unwanted characters, such as HTML tags, punctuation marks, special characters, white spaces etc. Stopword removal: Some words do not contribute much to the machine learning model, so it's good to remove them. A list of stopwords can be defined by the nltk library, or it can be business-specific.

4.2 Stemming

It may be defined as the process to remove the inflectional forms of a word and bring them to a base form called the stem. The chopped-off pieces are referred to as affixes. The two most common algorithms/methods employed for stemming include the: Porter Stemmer Snowball Stemmer

4.3 Lemmatization

It is a process wherein the context is used to convert a word to its meaningful base form. It helps in grouping together words that have a common base form and so can be identified as a single item. The base form is referred to as the lemma of the word and is also sometimes known as the dictionary form. The most commonly used lemmatizers are the WordNet Lemmatizer Spacy Lemmatizer TextBlob Lemmatizer We will be using WordNet Lemmatizer in our process.

4.4 Vectorization

Processing natural language text and extract useful information from the given word or a sentence using machine learning and deep learning techniques requires the string/text needs to be converted into a set of real numbers (a vector) — Word Embeddings. Word Embeddings or Word vectorization is a methodology in NLP to map words or phrases from vocabulary to a corresponding vector of real numbers which used to find word predictions, word similarities/semantics. The process of converting words into numbers are called Vectorization.

Chapter 5

Proposed Work

Here we have applied two machine learning algorithms.

1. Naive Bayes Classification

2. Logistic Regression Model.

5.1 Naive Bayes Classification

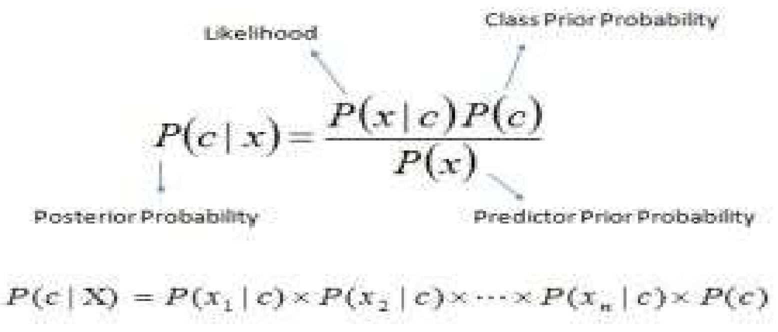
It is a classification technique based on Bayes' Theorem with an assumption of independence among predictors. In simple terms, a Naive Bayes classifier assumes that the presence of a particular feature in a class is unrelated to the presence of any other feature.

For example, a fruit may be considered to be an apple if it is red, round, and about 3 inches in diameter. Even if these features depend on each other or upon the existence of the other features, all of these properties independently contribute to the probability that this fruit is an apple and that is why it is known as 'Naive'.

So here we will be classifying our tweet into either positive, negative, or neutral by implementing our no. of predicted classes equal to three.

Bayes theorem provides a way of calculating posterior probability P(c—x) from P(c), P(x) and P(x—c). Look at the equation below:

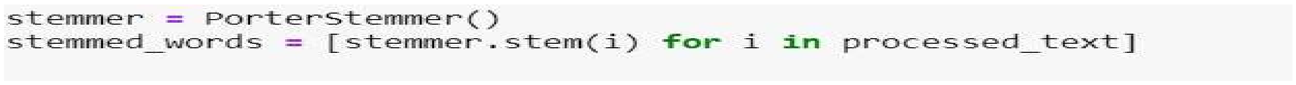
So here is our code which we have implemented.



$$P(c \mid x) = \frac{P(x \mid c)P(c)}{P(x)}$$

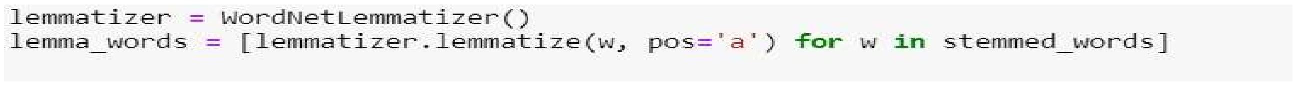
$$P(c \mid X) = P(x_1 \mid c) \times P(x_2 \mid c) \times \cdots \times P(x_n \mid c) \times P(c)$$

Figure 5.1: A Formula.



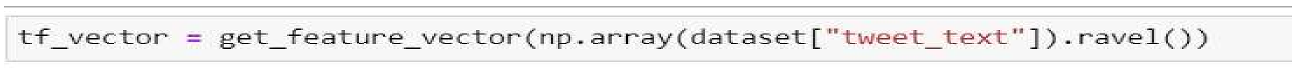
```
stemmer = PorterStemmer()
stemmed_words = [stemmer.stem(i) for i in processed_text]
```

Figure 5.2: Stemming.



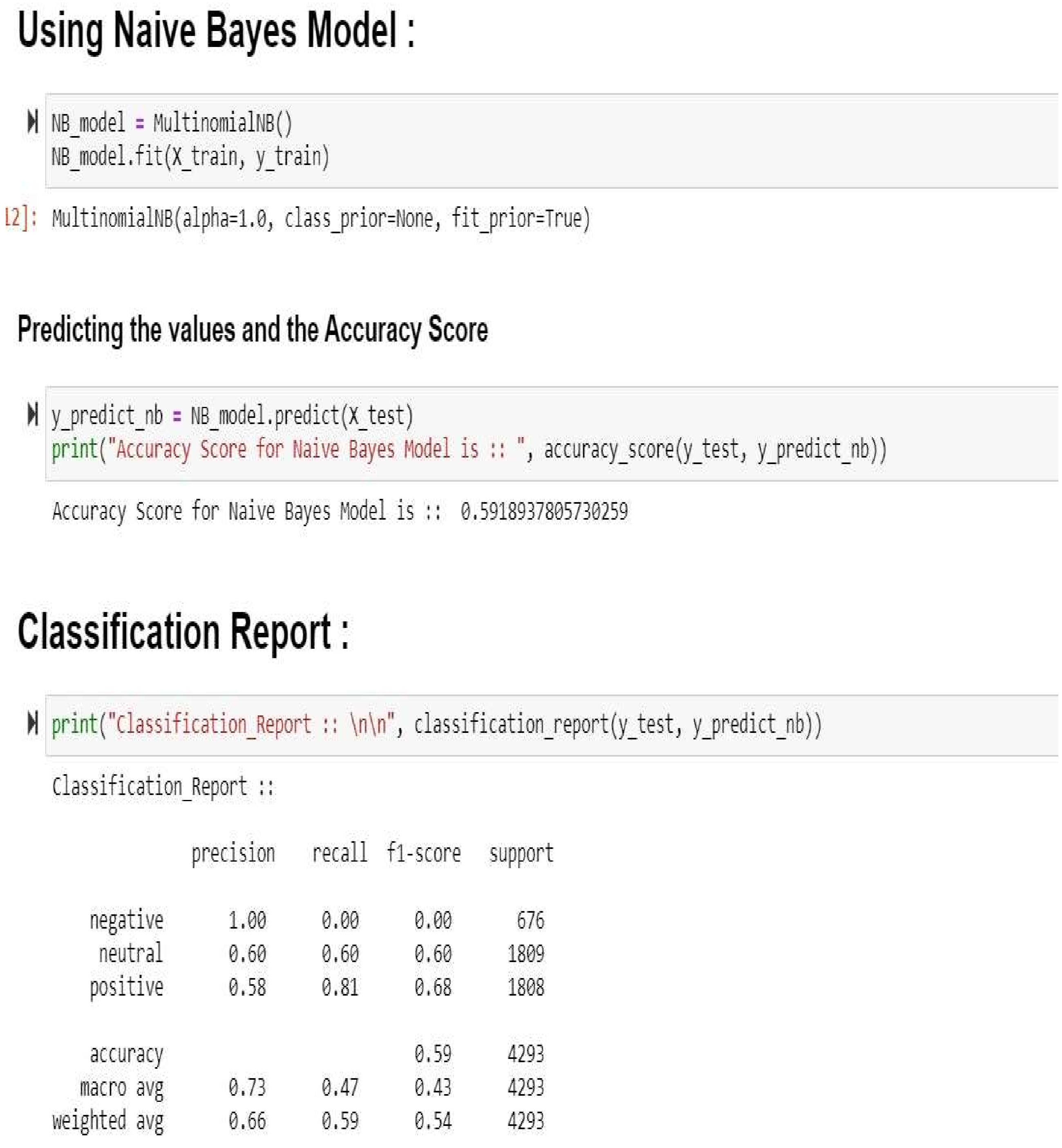
```
lemmatizer = WordNetLemmatizer()
lemma_words = [lemmatizer.lemmatize(w, pos='a') for w in stemmed_words]
```

Figure 5.3: Lemmatization.



```
tf_vector = get_feature_vector(np.array(dataset["tweet_text"]).ravel())
```

Figure 5.4: Vectorization.



Using Naive Bayes Model :

```
NB_model = MultinomialNB()
NB_model.fit(X_train, y_train)
```

```
12]: MultinomialNB(alpha=1.0, class_prior=None, fit_prior=True)
```

Predicting the values and the Accuracy Score

```
y_predict_nb = NB_model.predict(X_test)
print("Accuracy Score for Naive Bayes Model is :: ", accuracy_score(y_test, y_predict_nb))
```

```
Accuracy Score for Naive Bayes Model is ::  0.5918937805730259
```

Classification Report :

```
print("Classification_Report :: \n\n", classification_report(y_test, y_predict_nb))
```

```
Classification_Report :: 

              precision    recall  f1-score   support

    negative       1.00      0.00      0.00       676
     neutral       0.60      0.60      0.60      1809
    positive       0.58      0.81      0.68      1808

    accuracy                           0.59      4293
   macro avg       0.73      0.47      0.43      4293
weighted avg       0.66      0.59      0.54      4293
```

Figure 5.5: Naive Bayes Model.

5.2 Logistic Regression Model

In statistics, the logistic model (or logit model) is used to model the probability of a certain class or event existing such as pass/fail, win/lose. This can be extended to model several classes of events such as determining whether an image contains a cat, dog, lion, etc. So here we have used the concept of multinomial logistic regression to pridict whether a tweet is neutral, positive or negative. Types of Logistic Regression:

1. Binary Logistic Regression

   The categorical response has only two 2 possible outcomes. Example: Spam or Not.

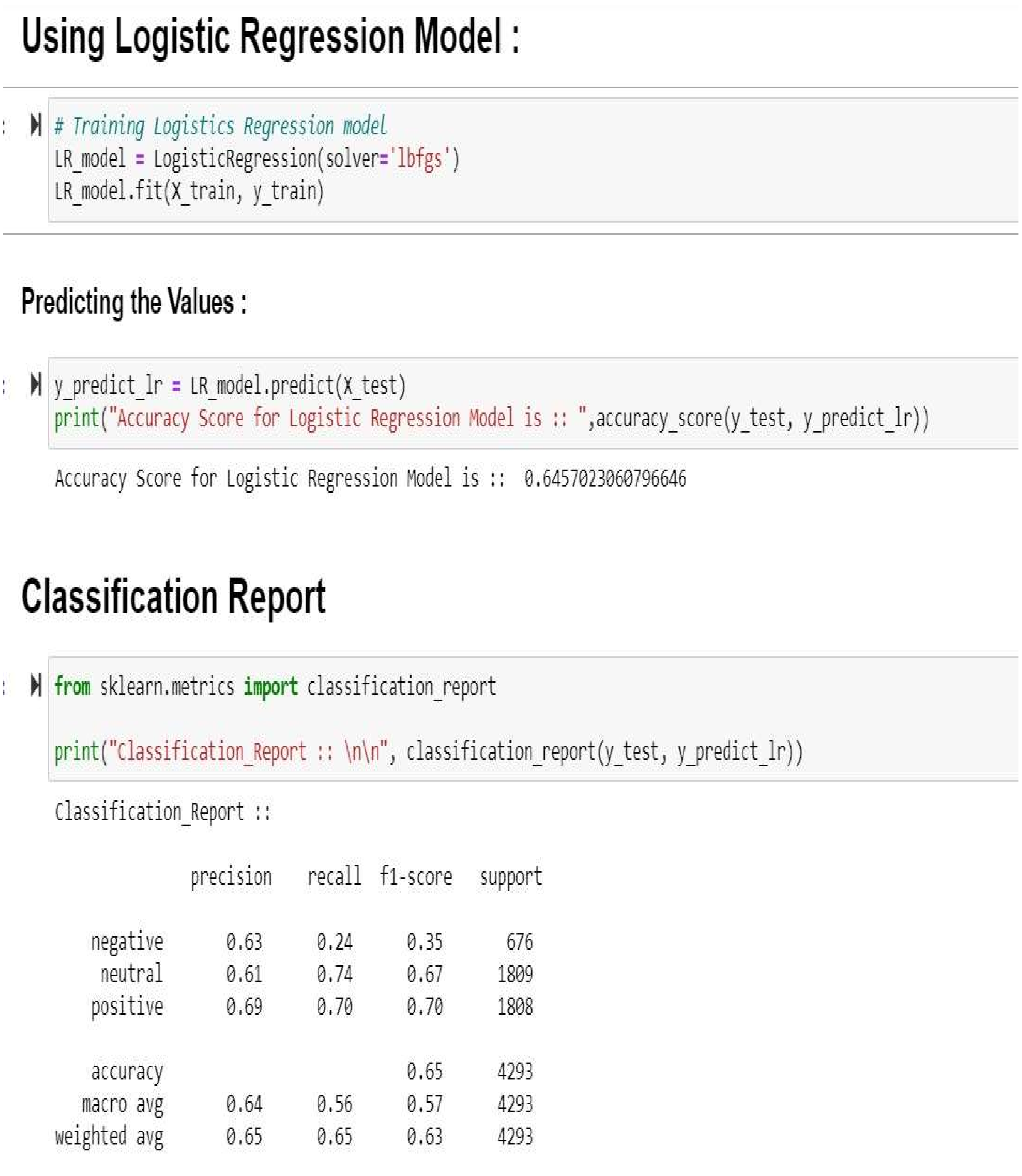
2. Multinomial Logistic Regression

   Three or more categories without ordering. Example: Predicting which food is preferred more (Veg, Non-Veg, Vegan).

3. Ordinal Logistic Regression

   Three or more categories with ordering. Example: Movie rating from 1 to 5.

Decision Boundary is to predict which class a data belongs, a threshold can be set. Based upon this threshold, the obtained estimated probability is classified into classes. Say, if predicted value 0.5, then classify email as spam else as not spam. Decision boundary can be linear or non-linear. Polynomial order can be increased to get complex decision boundary. So here we are using Multinomial Logistic Regression.



Using Logistic Regression Model :

```
# Training Logistics Regression model
LR_model = LogisticRegression(solver='lbfgs')
LR_model.fit(X_train, y_train)
```

Predicting the Values :

```
y_predict_lr = LR_model.predict(X_test)
print("Accuracy Score for Logistic Regression Model is :: ",accuracy_score(y_test, y_predict_lr))
```

```
Accuracy Score for Logistic Regression Model is ::  0.6457023060796646
```

Classification Report

```
from sklearn.metrics import classification_report

print("Classification_Report :: \n\n", classification_report(y_test, y_predict_lr))
```

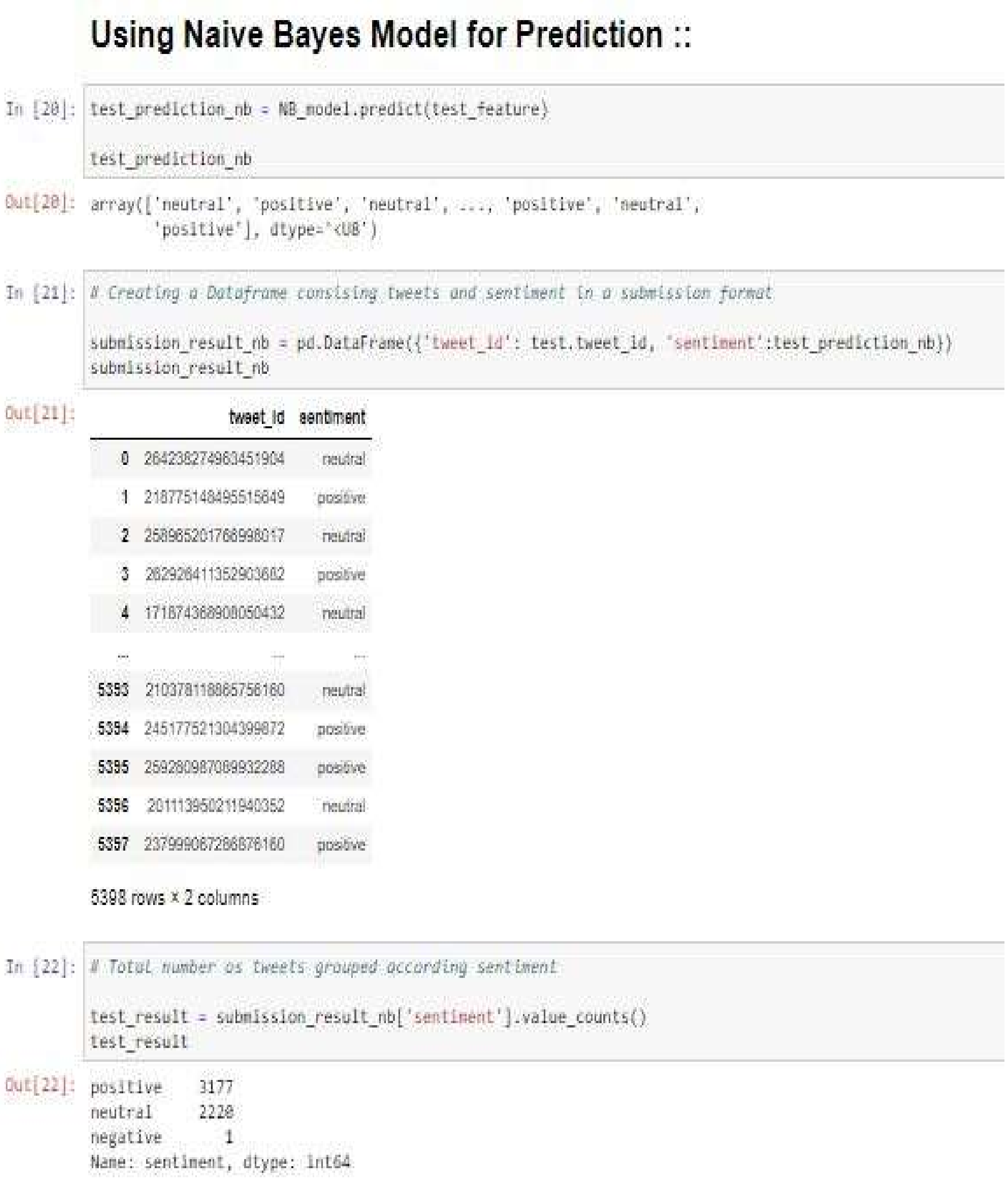
```
Classification_Report ::

              precision    recall  f1-score   support

    negative       0.63      0.24      0.35       676
     neutral       0.61      0.74      0.67      1809
    positive       0.69      0.70      0.70      1808

    accuracy                           0.65      4293
   macro avg       0.64      0.56      0.57      4293
weighted avg       0.65      0.65      0.63      4293
```

Figure 5.6: Logistic Regression Model.



Using Naive Bayes Model for Prediction ::

```
In [20]: test_prediction_nb = NB_model.predict(test_feature)

         test_prediction_nb

Out[20]: array(['neutral', 'positive', 'neutral', ..., 'positive', 'neutral',
                'positive'], dtype='<U8')
```

```
In [21]: # Creating a Dataframe consising tweets and sentiment in a submission format

         submission_result_nb = pd.DataFrame({'tweet_id': test.tweet_id, 'sentiment':test_prediction_nb})
         submission_result_nb

Out[21]:
```

	tweet_id	sentiment
0	264238274963451904	neutral
1	218775148495515649	positive
2	258965201766998017	neutral
3	262926411352903682	positive
4	171874368908050432	neutral
...	...	...
5393	210378118865756160	neutral
5394	245177521304399872	positive
5395	259280987089932288	positive
5396	201113950211940352	neutral
5397	237999067286876160	positive

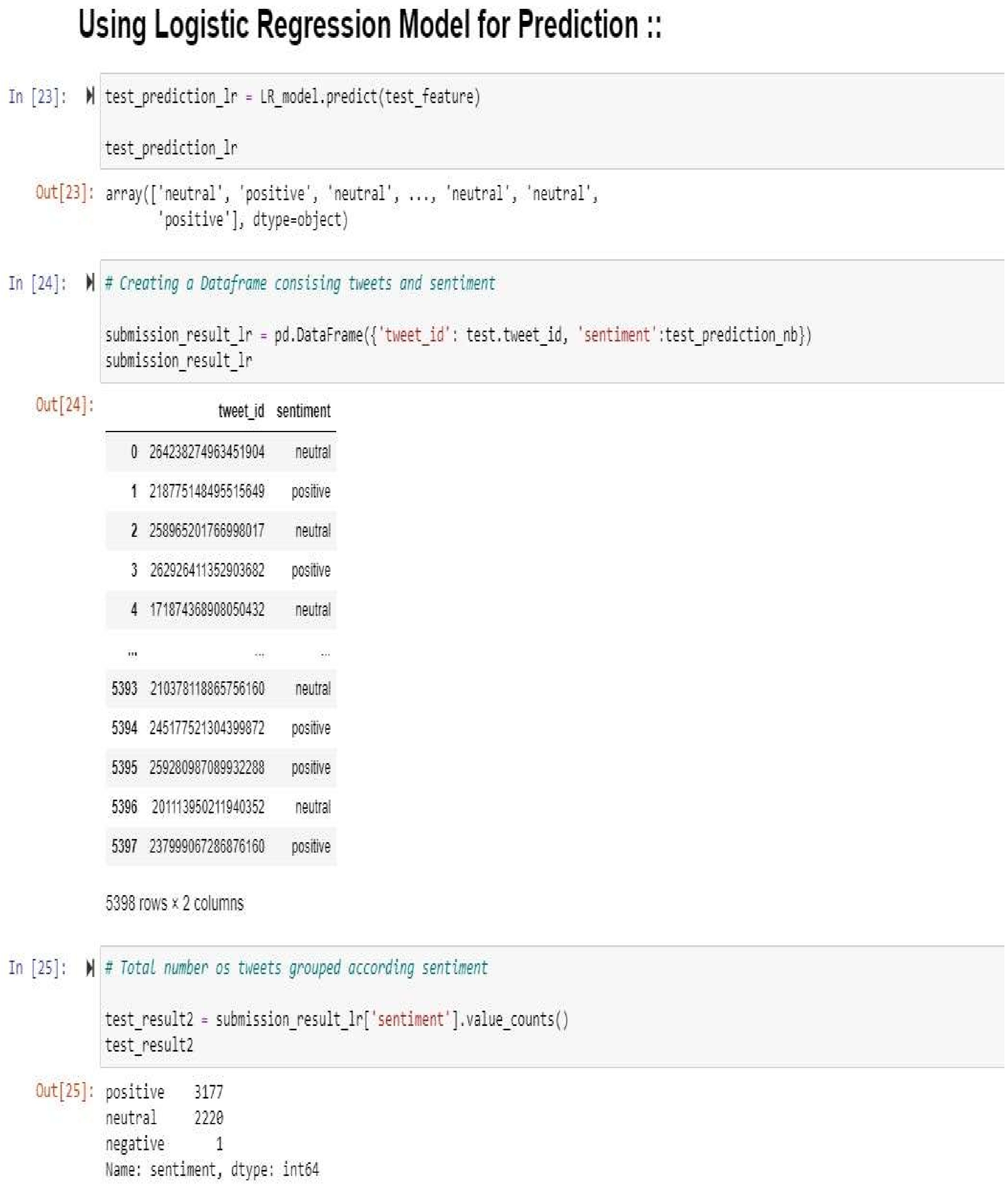
5398 rows × 2 columns

```
In [22]: # Total number os tweets grouped according sentiment

         test_result = submission_result_nb['sentiment'].value_counts()
         test_result

Out[22]: positive    3177
         neutral     2220
         negative       1
         Name: sentiment, dtype: int64
```

Figure 5.7: Naive Bayes for Prediction.



Using Logistic Regression Model for Prediction ::

```
In [23]:  test_prediction_lr = LR_model.predict(test_feature)

          test_prediction_lr

Out[23]:  array(['neutral', 'positive', 'neutral', ..., 'neutral', 'neutral',
                 'positive'], dtype=object)

In [24]:  # Creating a Dataframe consising tweets and sentiment

          submission_result_lr = pd.DataFrame({'tweet_id': test.tweet_id, 'sentiment':test_prediction_nb})
          submission_result_lr

Out[24]:
```

	tweet_id	sentiment
0	264238274963451904	neutral
1	218775148495515649	positive
2	258965201766998017	neutral
3	262926411352903682	positive
4	171874368908050432	neutral
...	...	...
5393	210378118865756160	neutral
5394	245177521304399872	positive
5395	259280987089932288	positive
5396	201113950211940352	neutral
5397	237999067286876160	positive

5398 rows × 2 columns

```
In [25]:  # Total number os tweets grouped according sentiment

          test_result2 = submission_result_lr['sentiment'].value_counts()
          test_result2

Out[25]:  positive    3177
          neutral     2220
          negative       1
          Name: sentiment, dtype: int64
```

Figure 5.8: Logistic Model for Prediction.

Chapter 6

Results and Discussion

So as we have used two algorithms. Here are the results.

1.Naive Bayes for Prediction.

Table 6.1: Convolution Matrix-Naive Bayes.

Parameters	**Precision**	**Recall**	**F1-Score**	**Support**
negative	1.00	0.00	0.00	676
neutral	0.60	0.60	0.60	1809
positive	0.58	0.81	0.68	1808
accuracy			0.59	4293
macro avg	0.73	0.47	0.43	4293
weighted avg	0.66	0.59	0.54	4293

Table 6.2: Naive Bayes Results.

Class	**Calculation**
positive	3177
neutral	2220
negative	1

2.Logistic Regression for Prediction.

Table 6.3: Convolution Logistic Regression.

Parameters	**Precision**	**Recall**	**F1-Score**	**Support**
negative	0.63	0.24	0.35	676
neutral	0.61	0.74	0.67	1809
positive	0.69	0.70	0.70	1808
accuracy			0.65	4293
macro avg	0.64	0.56	0.57	4293
weighted avg	0.65	0.65	0.63	4293

Table 6.4: Logistic Regression Results.

Class	**Calculation**
positive	3177
neutral	2220
negative	1

Chapter 7
Conclusion and Future Work

Conclusion:-

We conclude that using different NLTK classifier it is easier to classify that tweets and more we improve the training data set more we can get accurate results.

Future Work:-

We look forward to use bigger dataset to improve the accuracy considering the emoticons and internationalization. The task of sentiment analysis, especially in the domain of micro-bloging, is still in the developing stage and far from complete. So we propose a couple of ideas which we feel are worth exploring in the future and may result in further improved performance.

One potential problem with our research is that the sizes of the three classes are not equal. The problem with unequal classes is that the classifier tries to increase the overall accuracy of the system by increasing the accuracy of the majority class, even if that comes at the cost of decrease in accuracy of the minority classes.

Last but not the least, we can attempt to model human confidence in our system. For example if we have 5 human labellers labelling each tweet, we can plot the tweet in the 2-dimensional objectivity / subjectivity and positivity / negativity plane while differentiating between tweets in which all 5 labels agree, only 4 agree, only 3 agree or no majority vote is reached.

References

[1]Alexander Pak and Patrick Paroubek. Twitter as a Corpus for Sentiment Analysis and Opinion Mining. In Proceedings of international conference on Language Resources and Evaluation (LREC), 2010.

[2]Luciano Barbosa and Junlan Feng. Robust Sentiment Detection on Twitter from Biased and Noisy Data. In Proceedings of the international conference on Computational Linguistics (COLING), 2010.

[3]Peter D. Turney. Thumbs Up or Thumbs Down? Semantic Orientation Applied to Unsupervised Classification of Reviews. In Proceedings of the Annual Meeting of the Association of Computational Linguistics (ACL), 2002.

[4]Theresa Wilson, Janyce Wiebe and Paul Hoffmann. Recognizing Contextual Polarity in Phrase-Level Sentiment Analysis. In the Annual Meeting of Association of Computational Linguistics: Human Language Technologies (ACL-HLT), 2005.