M.Sc. Mathematics

Prince

Farheen Azaz

2022

SENTIMENT CLASSIFICATION AND ANALYSIS ON DISTANCE LEARNING

A DISSERTATION

SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE AWARD OF THE DEGREE OF

MASTER OF SCIENCE IN

MATHEMATICS

Submitted by:

Prince

2K20/MSCMAT/22

Farheen Ajaz

2K20/MSCMAT/40

Under the supervision of

Ms. Sumedha Seniaray

DEPARTMENT OF APPLIED MATHEMATICS

DELHI TECHNOLOGICAL UNIVERSITY

(Formerly Delhi College of

Engineering) Bawana Road, Delhi-

110042

MAY, 2022

DEPARTMENT OF APPLIED MATHEMATICS
DELHI TECHNOLOGICAL UNIVERSITY
(Formerly Delhi College of
Engineering) Bawana Road, Delhi-
110042

DECLARATION

We, Prince, 2K20/MSCMAT/22 & Farheen Ajaz, 2K20/MSCMAT/40 students of M.Sc. Mathematics, hereby declare that the project Dissertation titled " Sentiment Classification and Analysis on Distance Learning " which is submitted by me to the Department of Applied Mathematics Delhi Technological University, Delhi in partial fulfillment of the requirement for the award of the degree of Master of Science, is original and not copied from any source without proper citation. This work has not previously formed the basis for the award of any Degree, Diploma Associateship, Fellowship or other similar title or recognition.

Place: New Delhi Prince

Date: Farheen Ajaz

DEPARTMENT OF APPLIED MATHEMATICS
DELHI TECHNOLOGICAL UNIVERSITY
(Formerly Delhi College of
Engineering) Bawana Road, Delhi-
110042

CERTIFICATE

We hereby certify that the Project Dissertation titled " SENTIMENT CLASSIFICATION AND ANALYSIS ON DISTANCE LEARNING " which is submitted by Prince, Roll No. 2K20/MSCMAT/22 & Farheen Ajaz, Roll No. 2K20/MSCMAT/40 [Department of Applied Mathematics], Delhi Technological University, Delhi in partial fulfillment of the requirement for the award of the degree of Master of Science, is a record of the project work carried out by the students under my supervision. To the best of my knowledge this work has not been submitted in part or full for any Degree or Diploma to this University or elsewhere.

Place: New Delhi **Ms. Sumedha Seniaray**

Date: **SUPERVISOR**

DELHI TECHNOLOGICAL UNIVERSITY
(Formerly Delhi College of
Engineering) Bawana Road, Delhi-
110042

ACKNOWLEDGEMENT

I would like to express my profound gratitude to Dr. S. Sivaprasad Kumar HOD, of Department of Applied Mathematics. I would like to express my special thanks to our mentor Ms. Sumedha Seniaray for his time and efforts he provided throughout the year. Your useful advice and suggestions were really helpful to me during the project’s completion. In this aspect, I am eternally grateful to you.

I would like to acknowledge that this project was completed entirely by me and not by someone else.

Signature

PRINCE

FARHEEN AJAZ

DELHI TECHNOLOGICAL UNIVERSITY
(Formerly Delhi College of
Engineering) Bawana Road, Delhi-
110042

ABSTRACT

The aim of this report is to study the reaction of people of the world towards distance education or distance learning. Females, vocational high school graduates and full time working students agree with some of the statements more than others do because for them it is convenient to do courses online. It is thought that students are satisfied with this education system which provides great convenience in time and cost.
In these changing times, globalization hales the education systems to have different alternatives of education, and one of them is Distance Education. Today distance education is an interesting option for those people who lack the opportunity to attend the traditional system of education -face to-face - due to constraints in time, space, and money that the students might have. What and how the students learn is the main concern which can take us to establish what is to be evaluated and how to evaluate considering that DE could be for everybody no matter the background, self-confidence and intellectual preferences of the interested students.
Online education comes in shades of grey. We analyze public view on the pandemic in regards with mental stress, interrupted power supply, affordability and access to internet, flexibility of schedule, reduction of long-distance commute, risk of covid and other such consequences of online learning and Government's take on the need for inclusive education policies. In this research, we qualitatively inspect the consequences of COVID-19 pandemic on education of the students. This study primarily focuses on the response of students of all age groups, educators, college professors, schoolteachers and also parents of young students towards the approach of distance learning or Online education in the past two years. We have taken two datasets, first being the Twitter dataset comprising of tweets from around the whole world and second, dataset which is specific to tweets from India. The data has been extracted from twitter with the aid of twitter API and then two sentiment analysis approaches have been implemented, first Machine learning classifiers namely, Naïve Bayes, SVM,

Random Forest, Logistic Regression, KNN, XG-Boost and secondly, Lexicon Based algorithms, VADER and TEXTBLOB. Upon performing the said approaches, the maximum accuracy achieved is 94%.

This study seeks to examine the extent to which the community accepts distance learning as a precaution by employing the sentiment analysis of Twitter's tweets as one of the most popular social media. This method is considered effective due to its ability to access the community's tweets quickly and at a low cost.

TABLE OF CONTENTS

CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 Distance Education

Recently, a growth in distance education programs can be seen because of the time and space restriction of face to-face learning system. It is also economically advantageous and preferable by working students [2]. One of the definitions of United States Distance Learning organizational framework and process of providing instruction at a distance.[3]
Distance education takes place when a teacher and student(s) are physically separated. Web based distance education systems should have some characteristics such as user identification and user management, preparation of course contents, course management, starting student specific programs, setting/delivery of homework and project, preparation and holding examination and test, monitoring and analysing student behavior, determination of student success status, establishment and management of interactive communication environment [2]
Distance education provides an avenue for students not close to a university center to complete degree or courses in a timely fashion[5]
Murphy & Crosser (2010, p. 19) mention the retention in online class by stating that about 40% of undergraduate students complete the courses with grades of C or better, when compared with on-campus students; twice as many distance students fail their courses.
In this work, we present a novel application of sentiment analysis on the response of people all around the world towards distance learning using data from Twitter. We compare sentiments of people who either support or condemn the online /distance education.

1.2 Objective

After performing extraction of tweets, we shall perform sentiment analysis using VADER and Textblob and also machine learning algorithms to obtain most effective results.

1.3 About Twitter

Twitter is a popular micro blogging service where users create status messages (called "tweets"). These tweets contain much of human expressions liking, disliking, and their contributions towards different topics. Sentiment Analysis builds systems that try to identify and extract opinions within text.[4]

CHAPTER 2

LITERATURE REVIEW

2.1 Related Work

Malak Aljabri et al. **[3]** proposed a model to analyze the distance learning in Saudi Arabia with the help of Twitter Dataset during COVID-19 pandemic period. All the tweets collected are written in Arabic language and geographic location is set to Saudi Arabia. The tweets are classified into the educational stages i.e., primary school and kindergarten, intermediate and high schools, or university. Then 6 Machine Learning Algorithms (LR, NB, KNN, XGB, SVM, RF) done on the dataset. The Logistic Regression ML technique has achieved the best accuracy of 89.9%.Filiz Angay Kutluk et al. **[4]** has proposed a model to analyze the student’s satisfaction on distance education.In the model, Dataset is collected with the help of first degree data collection method from the two universities of Turkey, and with the help of T-test and One Way Anova the analysis is performed on it.
Usame Omer OSMANOGLU et al. **[5]** has proposed a model to measure the satisfaction for distance education course materials of Anadolu University. Overall, 6059 feedbacks were received, scaling was processed with the help of the triple Likert method and finally, Machine learning techniques were performed on the dataset . In which, 77.5% accuracy was achieved using Logistic Regression algorithm
Nimasha Arambepola**[6]** has proposed a model to analyse the effectiveness of the distance learning. Twitter Dataset is collected consisted 202,645 tweets during the lockdown period during the pandemic. The model gives out the result of 54% sentiments as positive, while 30% tweets are negative and 16% tweets comes out to be neutral.
Imatitikua D. Aiyanyo et al. **[7]** has proposed a model to analyze the impact of COVID-19 pandemic on the educational sector of South

Korea. Twitter dataset is collected during the COVID-19 period with the geographical boundaries set to South Korea
The method followed by [3] to achieve the main goal of the study included several steps, starting from downloading related tweets within 48 h of the announcement of each prevention measure, then pre-processing the collected tweets. Unlike the manual annotation technique applied in this study, the researchers used sentiment labels based on emoji lexicons to label the tweets and restricted their sentiment study to only positive or negative categories. Then [3] applied the N-gram for feature extractions and Naïve Bayes model

for sentiment analysis.
If we talk about the works of [6] they hope to evaluate and use natural language processing methods and techniques by exploring the data. They have done sentiment analysis using NLTK 2.0.4 powered text classification process. In social media data, user types in multiple punctuation marks, acronyms and an emoticon to express their sentiments so they have made use of NLTK's VADER analyzer, which computationally identifies and categorizes text into three sentiments: positive, negative, or neutral.

CHAPTER 3

IMPLEMENTATION

3.1 Data Collection

The data has been collected in the form of tweets using the twitter API and the tweepy library in python.
A total of approximately 50k instances were collected to perform the analysis.
In order to analyse the sentiments of students and educators, a sampling of the population was taken from Twitter's publicly available tweets.
After data collection and filtering, the collected tweets were analyzed for expressions of emotions
The tweets have been collected in two phases
A) By hashtag search
B) By keyword search

A twitter developer account was created to access the tweets using the keys.

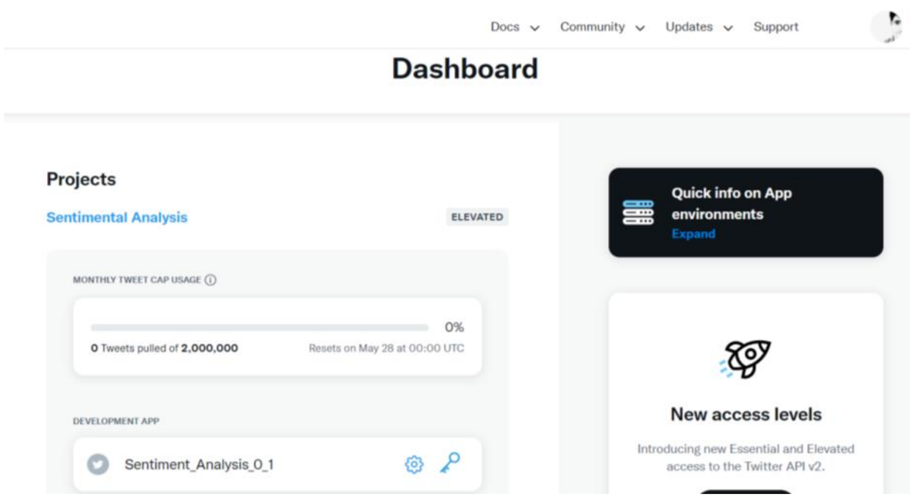
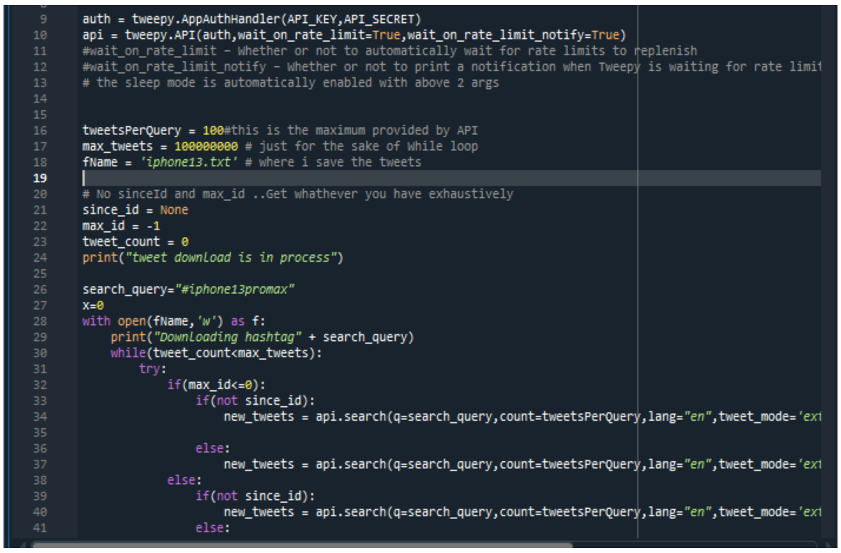


Fig1. Twitter developer account

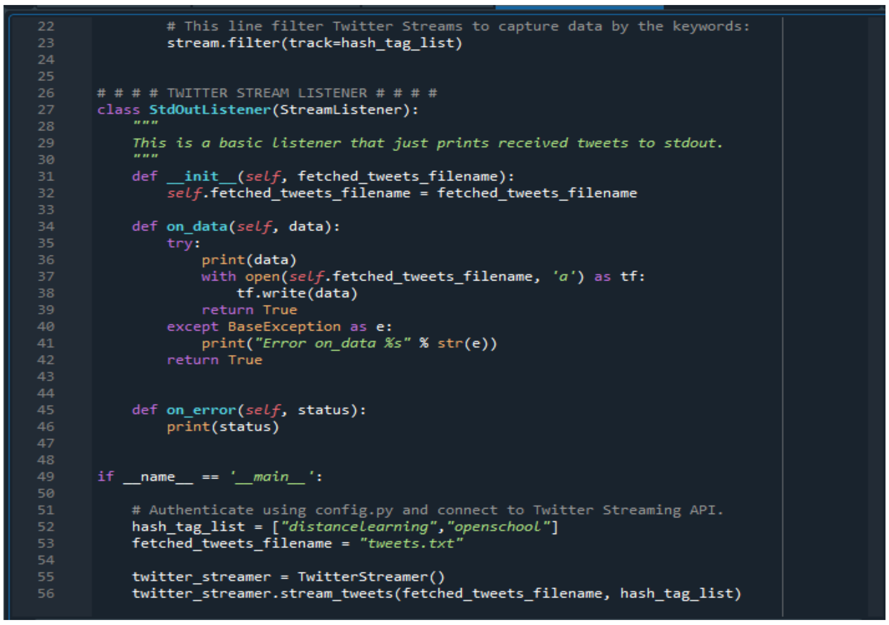
The tweets have been saved in a text file and appear as follows



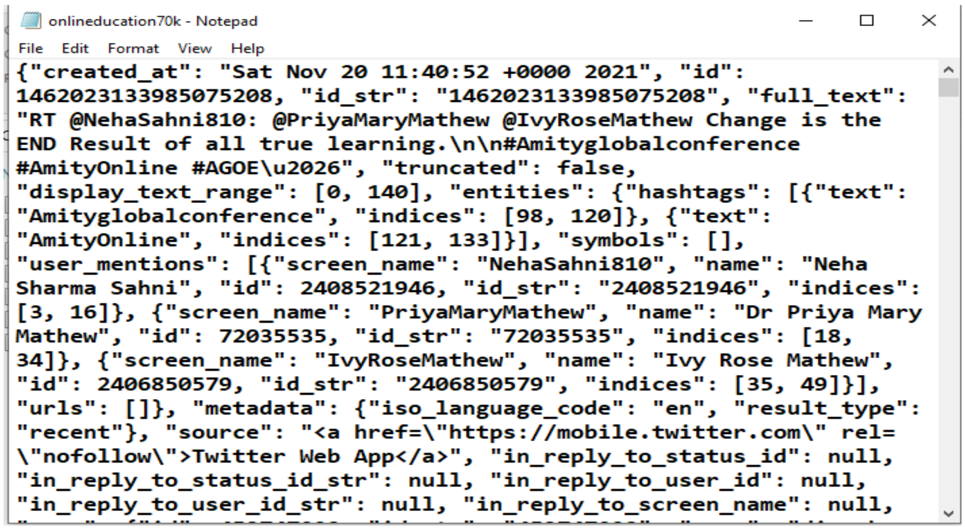
```
auth = tweepy.AppAuthHandler(API_KEY,API_SECRET)
api = tweepy.API(auth,wait_on_rate_limit=True,wait_on_rate_limit_notify=True)
#wait_on_rate_limit - Whether or not to automatically wait for rate limits to replenish
#wait_on_rate_limit_notify - Whether or not to print a notification when Tweepy is waiting for rate limi
# the sleep mode is automatically enabled with above 2 args


tweetsPerQuery = 100#this is the maximum provided by API
max_tweets = 100000000 # just for the sake of While loop
fName = 'iphone13.txt' # where i save the tweets

# No sinceId and max_id ..Get whathever you have exhaustively
since_id = None
max_id = -1
tweet_count = 0
print("tweet download is in process")

search_query="#iphone13promax"
x=0
with open(fName,'w') as f:
    print("Downloading hashtag" + search_query)
    while(tweet_count<max_tweets):
        try:
            if(max_id<=0):
                if(not since_id):
                    new_tweets = api.search(q=search_query,count=tweetsPerQuery,lang="en",tweet_mode='ext

                else:
                    new_tweets = api.search(q=search_query,count=tweetsPerQuery,lang="en",tweet_mode='ext
            else:
                if(not since_id):
                    new_tweets = api.search(q=search_query,count=tweetsPerQuery,lang="en",tweet_mode='ext
                else:
```

Fig2 : Code for tweet extraction by hashtags

Then the tweets were extracted using hashtags



```python
        # This line filter Twitter Streams to capture data by the keywords:
        stream.filter(track=hash_tag_list)


# # # # TWITTER STREAM LISTENER # # # #
class StdOutListener(StreamListener):
    """
    This is a basic listener that just prints received tweets to stdout.
    """
    def __init__(self, fetched_tweets_filename):
        self.fetched_tweets_filename = fetched_tweets_filename

    def on_data(self, data):
        try:
            print(data)
            with open(self.fetched_tweets_filename, 'a') as tf:
                tf.write(data)
            return True
        except BaseException as e:
            print("Error on_data %s" % str(e))
        return True


    def on_error(self, status):
        print(status)


if __name__ == '__main__':

    # Authenticate using config.py and connect to Twitter Streaming API.
    hash_tag_list = ["distancelearning","openschool"]
    fetched_tweets_filename = "tweets.txt"

    twitter_streamer = TwitterStreamer()
    twitter_streamer.stream_tweets(fetched_tweets_filename, hash_tag_list)
```



```
{"created_at": "Sat Nov 20 11:40:52 +0000 2021", "id":
1462023133985075208, "id_str": "1462023133985075208", "full_text":
"RT @NehaSahni810: @PriyaMaryMathew @IvyRoseMathew Change is the
END Result of all true learning.\n\n#Amityglobalconference
#AmityOnline #AGOE\u2026", "truncated": false,
"display_text_range": [0, 140], "entities": {"hashtags": [{"text":
"Amityglobalconference", "indices": [98, 120]}, {"text":
"AmityOnline", "indices": [121, 133]}], "symbols": [],
"user_mentions": [{"screen_name": "NehaSahni810", "name": "Neha
Sharma Sahni", "id": 2408521946, "id_str": "2408521946", "indices":
[3, 16]}, {"screen_name": "PriyaMaryMathew", "name": "Dr Priya Mary
Mathew", "id": 72035535, "id_str": "72035535", "indices": [18,
34]}, {"screen_name": "IvyRoseMathew", "name": "Ivy Rose Mathew",
"id": 2406850579, "id_str": "2406850579", "indices": [35, 49]}],
"urls": []}, "metadata": {"iso_language_code": "en", "result_type":
"recent"}, "source": "<a href=\"https://mobile.twitter.com\" rel=
\"nofollow\">Twitter Web App</a>", "in_reply_to_status_id": null,
"in_reply_to_status_id_str": null, "in_reply_to_user_id": null,
"in_reply_to_user_id_str": null, "in_reply_to_screen_name": null,
```

Fig3. Extracted tweets

3.2 Data Preprocessing

As we can see the tweets contain lot of unwanted information which may cause problems while performing sentiment analysis. In the next

phase the tweets have been cleaned so as to remove special characters,urls, unwanted information, duplicate tweets etc. and then they have been converted to a csv file and further into a data-frame to perform sentiment analysis.
Here, created at: determines the date and time of the tweet creation. Data obtained from twitter usually contains a lot of HTML entities, for example &lt; &gt; &amp; which get embedded in the original data. As a result, it is vital to eliminate these entities. This process is collectively known as Data cleaning or Data preprocessing. And we only keep the following attributes for further work.

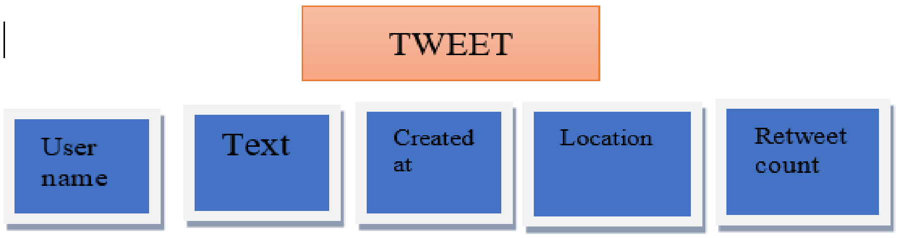
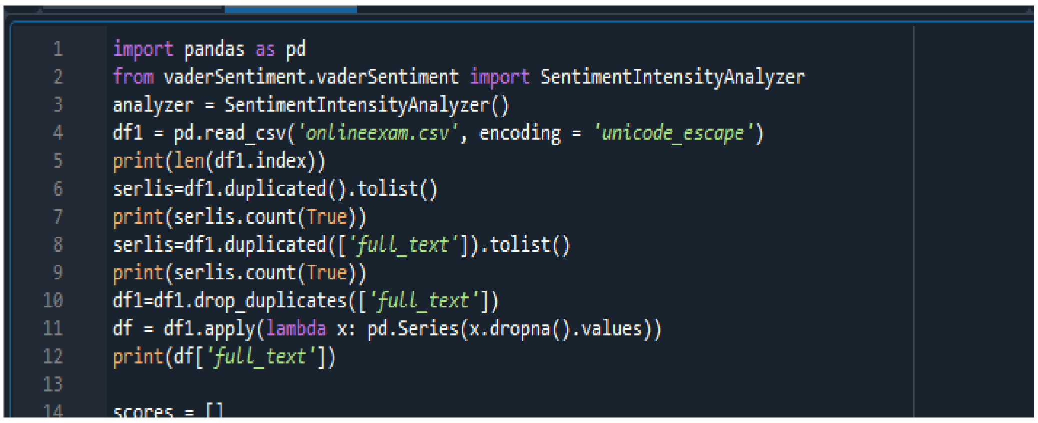


Fig4. attributes of a tweet

The tweets are further converted in a csv file and then in a dataframe. After the conversion the tweets are cleaned by the following code.

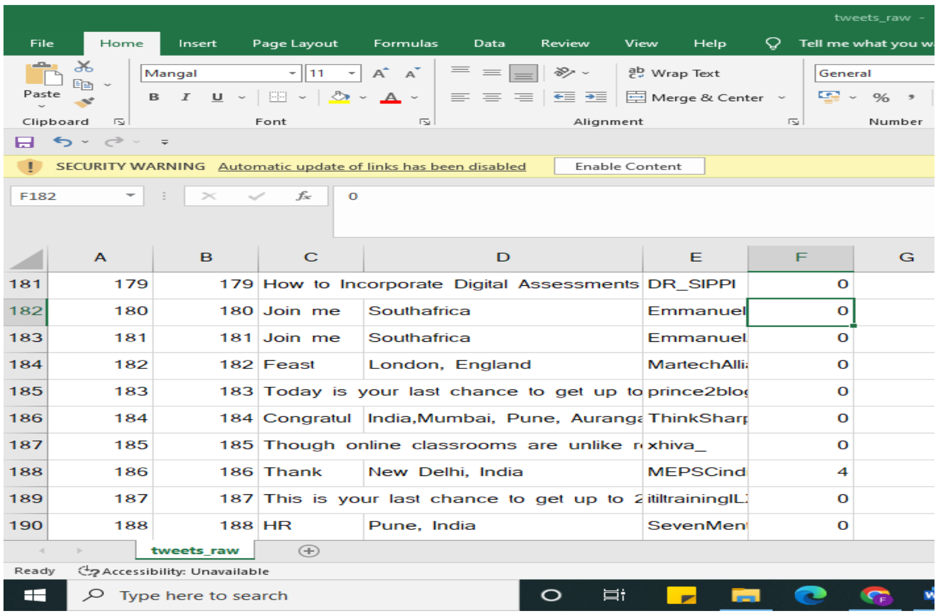


```
import pandas as pd
from vaderSentiment.vaderSentiment import SentimentIntensityAnalyzer
analyzer = SentimentIntensityAnalyzer()
df1 = pd.read_csv('onlineexam.csv', encoding = 'unicode_escape')
print(len(df1.index))
serlis=df1.duplicated().tolist()
print(serlis.count(True))
serlis=df1.duplicated(['full_text']).tolist()
print(serlis.count(True))
df1=df1.drop_duplicates(['full_text'])
df = df1.apply(lambda x: pd.Series(x.dropna().values))
print(df['full_text'])

scores = []
```

Fig5: code for cleaning the dataframe

3.3 Segregation of Indian Tweets from the dataset

In this proposed approach, we have focused on the sentiment analysis of twitter data from India. Since most of the tweets have location mentioned in the description, we have made use of the location to filter the Indian tweets from the world tweets dataset.
Now in order to separate Indian tweets, first we look at all the tweets after extraction and pre-processing, converted in dataframe.

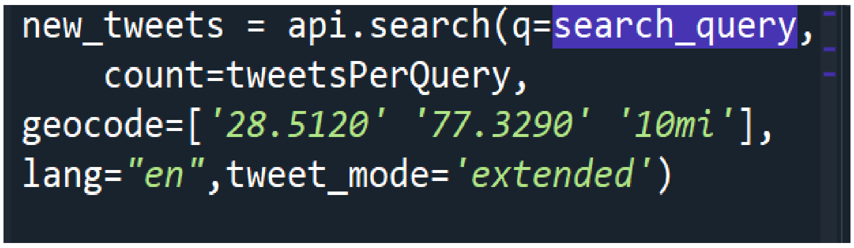


	A	B	C	D	E	F	G
181	179	179	How to Incorporate Digital Assessments		DR_SIPPI	0	
182	180	180	Join me	Southafrica	Emmanuel	0	
183	181	181	Join me	Southafrica	Emmanuel	0	
184	182	182	Feast	London, England	MartechAlli	0	
185	183	183	Today is your last chance to get up to		prince2blo	0	
186	184	184	Congratul	India,Mumbai, Pune, Auranga	ThinkSharp	0	
187	185	185	Though online classrooms are unlike r		xhiva_	0	
188	186	186	Thank	New Delhi, India	MEPSCind	4	
189	187	187	This is your last chance to get up to 2		itiltrainingIL	0	
190	188	188	HR	Pune, India	SevenMen	0	

Fig6: Tweets in csv

We can see in the D column we have the location of the tweets mentioned.

Using this location we can filter tweets from India in python.
Apart from this method, Geolocation has also been incorporated to directly extract the tweets specific to India. The geolocation aids in sieving tweets as per the wish of the user.
It works by adding the ‘geocode’ parameter in the search query. The geocode parameter expects three values namely the latitude, longitude and the radius (which can either be in kilometers or miles).
For example, if a user wants tweets from Delhi within 10 miles the code is illustrated below.



```
new_tweets = api.search(q=search_query,
    count=tweetsPerQuery,
geocode=['28.5120' '77.3290' '10mi'],
lang="en",tweet_mode='extended')
```

Fig7: Gelocation example

After separation of Indian tweets, we convert the dataset into a dataframe for further analysis.

The figure below illustrates the methodology for this research work.

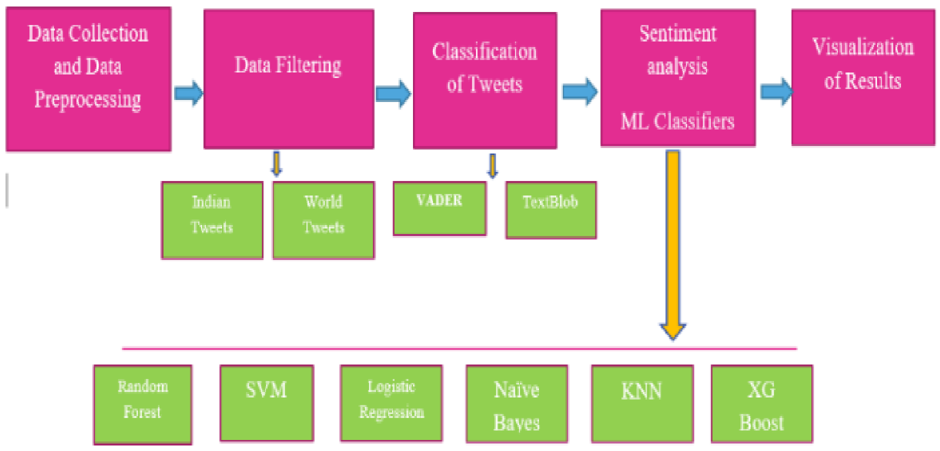


Fig8: Illustration of Methodology

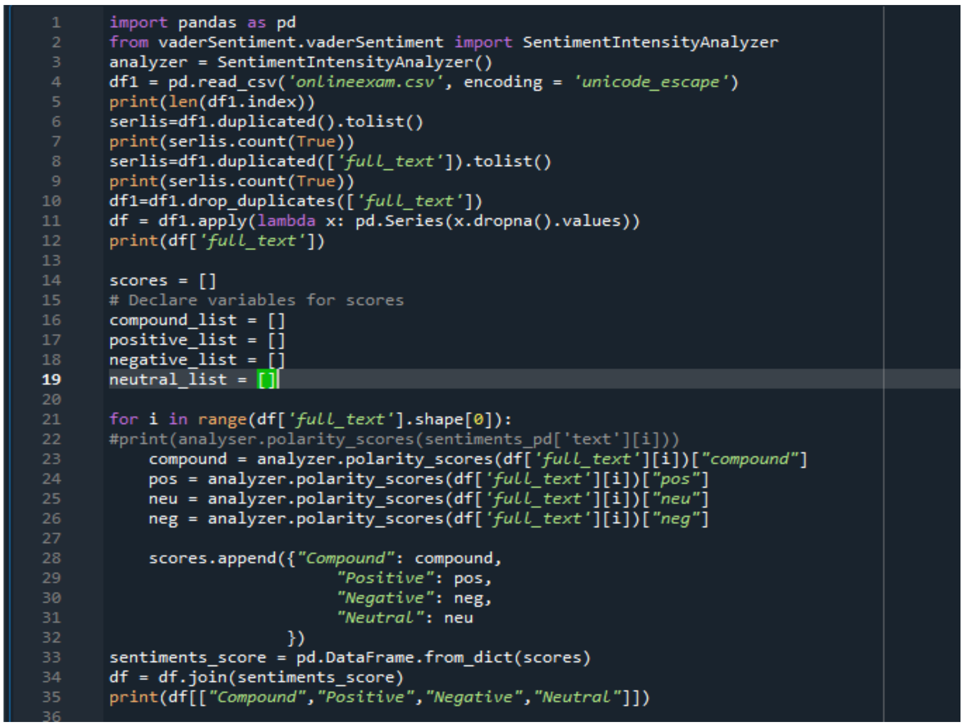
CHAPTER 4

SENTIMENT CLASSIFICATION

4.1 Sentiment classification by Vader

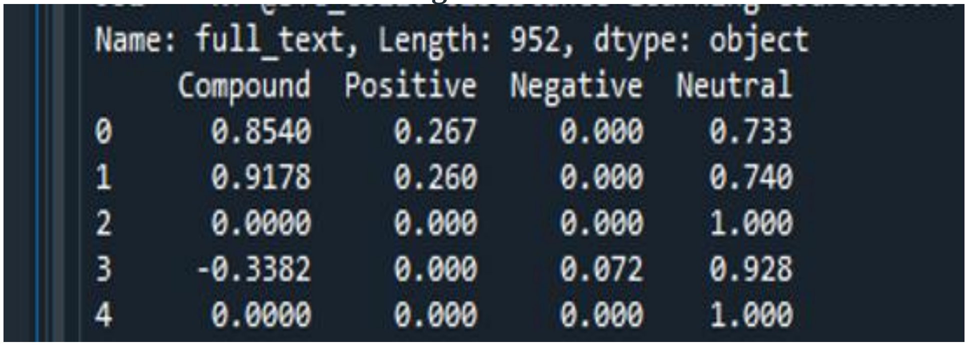
The analysis was performed using VADER in python.
VADER (Valence Aware Dictionary and Sentiment Reasoner) is a lexicon and rule-based sentiment analysis tool that is specifically **attuned to sentiments expressed in social media**.
VADER finds out the polarity of the tweets by finding positive, negative and compound score.
VADER not only tells about the Positivity and Negativity score but also tells us about how positive or negative a sentiment is
The Compound score is a metric that calculates the sum of all the lexicon ratings which have been normalized between -1(most extreme negative) and +1 (most extreme positive).
positive sentiment: (compound score >= 0.05)
neutral sentiment: (compound score > -0.05) and (compound score < 0.05)
negative sentiment : (compound score <= -0.05)

The tweets dataframe will be joined by score dataframe which will hold the values corresponding to each tweet.



```python
import pandas as pd
from vaderSentiment.vaderSentiment import SentimentIntensityAnalyzer
analyzer = SentimentIntensityAnalyzer()
df1 = pd.read_csv('onlineexam.csv', encoding = 'unicode_escape')
print(len(df1.index))
serlis=df1.duplicated().tolist()
print(serlis.count(True))
serlis=df1.duplicated(['full_text']).tolist()
print(serlis.count(True))
df1=df1.drop_duplicates(['full_text'])
df = df1.apply(lambda x: pd.Series(x.dropna().values))
print(df['full_text'])

scores = []
# Declare variables for scores
compound_list = []
positive_list = []
negative_list = []
neutral_list = []

for i in range(df['full_text'].shape[0]):
#print(analyser.polarity_scores(sentiments_pd['text'][i]))
    compound = analyzer.polarity_scores(df['full_text'][i])["compound"]
    pos = analyzer.polarity_scores(df['full_text'][i])["pos"]
    neu = analyzer.polarity_scores(df['full_text'][i])["neu"]
    neg = analyzer.polarity_scores(df['full_text'][i])["neg"]

    scores.append({"Compound": compound,
                   "Positive": pos,
                   "Negative": neg,
                   "Neutral": neu
                  })
sentiments_score = pd.DataFrame.from_dict(scores)
df = df.join(sentiments_score)
print(df[["Compound","Positive","Negative","Neutral"]])
```

Fig9: Code for Vader



```
Name: full_text, Length: 952, dtype: object
   Compound  Positive  Negative  Neutral
0    0.8540     0.267     0.000    0.733
1    0.9178     0.260     0.000    0.740
2    0.0000     0.000     0.000    1.000
3   -0.3382     0.000     0.072    0.928
4    0.0000     0.000     0.000    1.000
```

Fig9(A): Output

Each tweet’s score has been printed corresponding to its number along with the compound score, positive, negative and neutral score as well.

Plotting Wordcloud

Wordcloud displays the maximum or most prevalent words that are evident in the tweets

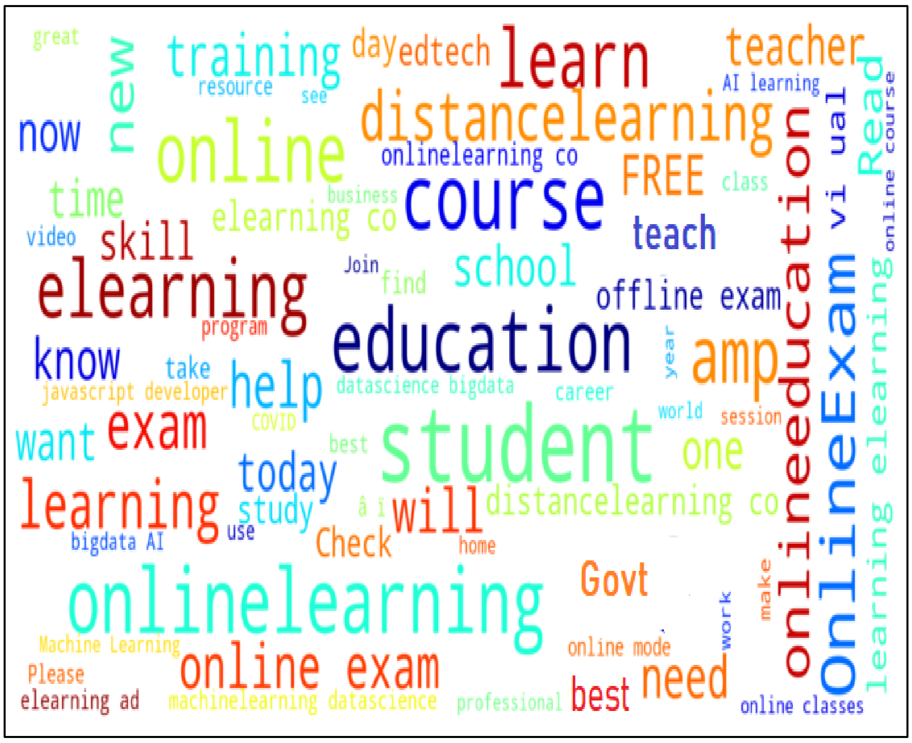


Fig10 : Wordcloud of most prominent words in dataset

4.1.a Visualisation of Results

The following pie chart has been created to visualize the data. A for loop was run over compound values in the dataframe and the percentages of positive, neutral and negative tweets were calculated and then plotted using matplotlib.
We can compare these results obtained for india dataset with the result for world dataset obtained in dissertation1.

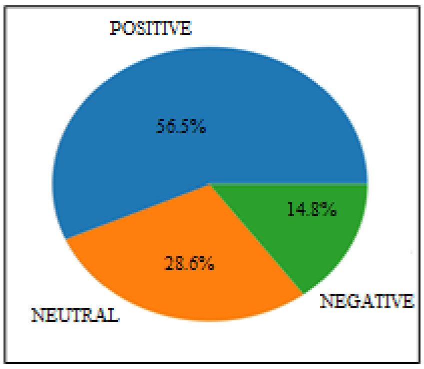


Fig11(A): Vader Stats for India

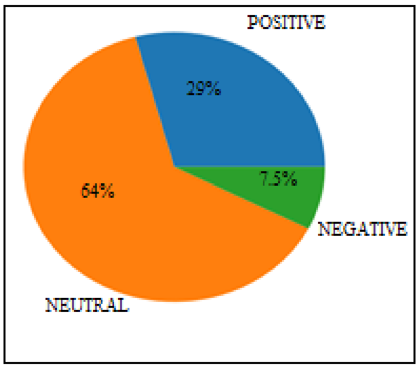
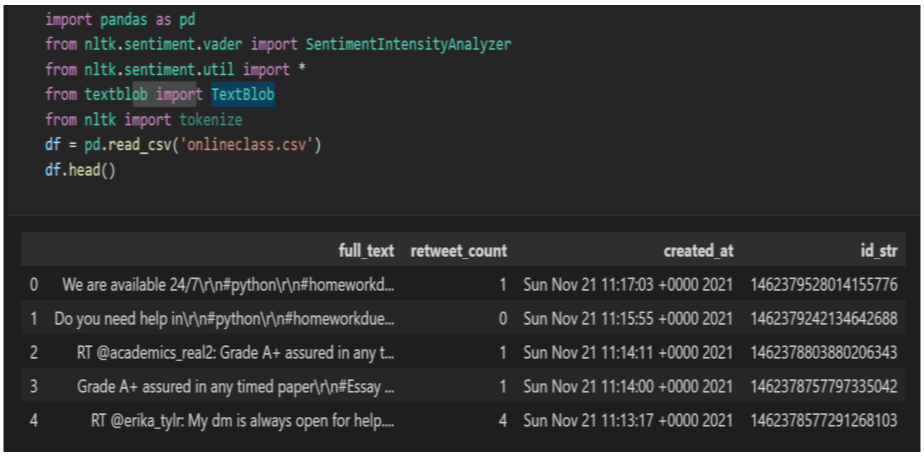


Fig11(B): Vader Stats for World

	Positive	*Negative*	*Neutral*
World	29%	7.5%	64%
India	56.5%	14.8%	28.6%

4.2 Sentiment Classification by Textblob

TextBlob is a Python (2 and 3) library for processing textual data. It provides a simple API for diving into common natural language processing (NLP) tasks such as part-of-speech tagging, noun phrase extraction, sentiment analysis, classification, translation, and more.



```python
import pandas as pd
from nltk.sentiment.vader import SentimentIntensityAnalyzer
from nltk.sentiment.util import *
from textblob import TextBlob
from nltk import tokenize
df = pd.read_csv('onlineclass.csv')
df.head()
```

```
                                        full_text  retweet_count                      created_at               id_str
0     We are available 24/7\r\n#python\r\n#homeworkd...              1  Sun Nov 21 11:17:03 +0000 2021  1462379528014155776
1   Do you need help in\r\n#python\r\n#homeworkdue...              0  Sun Nov 21 11:15:55 +0000 2021  1462379242134642688
2       RT @academics_real2: Grade A+ assured in any t...              1  Sun Nov 21 11:14:11 +0000 2021  1462378803880206343
3       Grade A+ assured in any timed paper\r\n#Essay ...              1  Sun Nov 21 11:14:00 +0000 2021  1462378757797335042
4          RT @erika_tylr: My dm is always open for help....              4  Sun Nov 21 11:13:17 +0000 2021  1462378577291268103
```

Fig12 : Code for Textblob

4.2.a Visualization of results

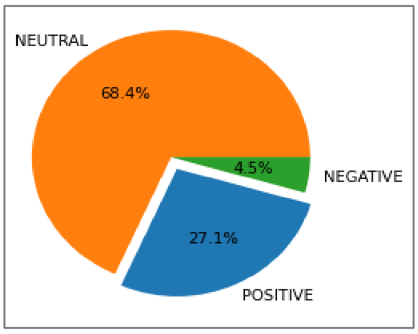


Fig12(A): Stats for World dataset textblob

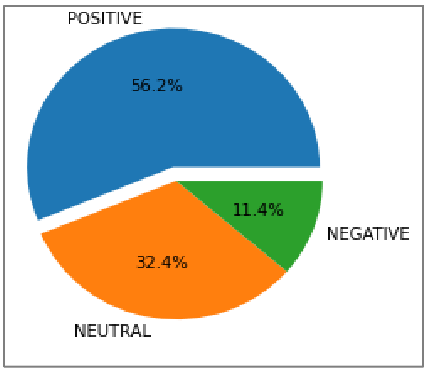
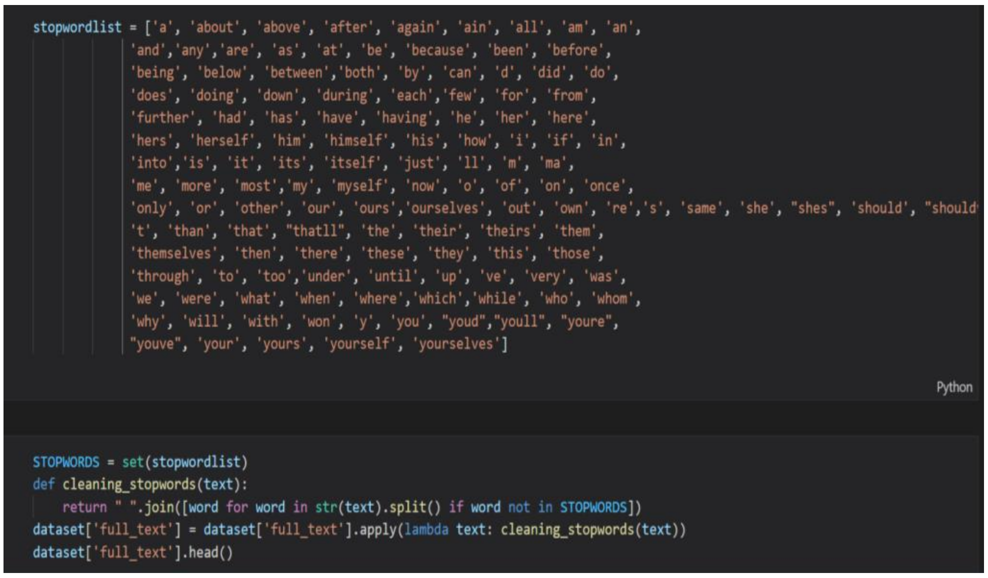


Fig12(B) : Stats for India Dataset textblob

	Positive	***Negative***	***Neutral***
World	27.1%	4.5%	68.4%
India	56.2%	11.4%	32.4%

Plotting Wordcloud for Textblob



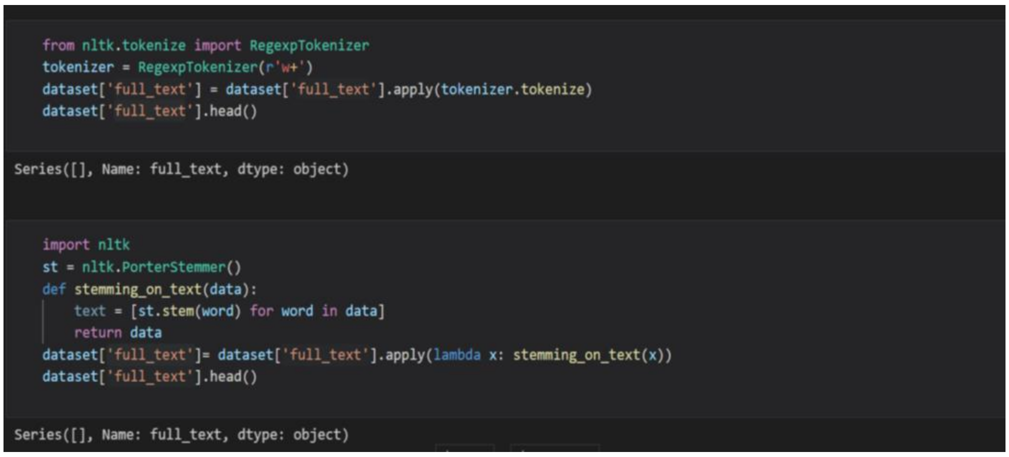
```
stopwordlist = ['a', 'about', 'above', 'after', 'again', 'ain', 'all', 'am', 'an',
             'and','any','are', 'as', 'at', 'be', 'because', 'been', 'before',
             'being', 'below', 'between','both', 'by', 'can', 'd', 'did', 'do',
             'does', 'doing', 'down', 'during', 'each','few', 'for', 'from',
             'further', 'had', 'has', 'have', 'having', 'he', 'her', 'here',
             'hers', 'herself', 'him', 'himself', 'his', 'how', 'i', 'if', 'in',
             'into','is', 'it', 'its', 'itself', 'just', 'll', 'm', 'ma',
             'me', 'more', 'most','my', 'myself', 'now', 'o', 'of', 'on', 'once',
             'only', 'or', 'other', 'our', 'ours','ourselves', 'out', 'own', 're','s', 'same', 'she', "shes", 'should', "should
             't', 'than', 'that', "thatll", 'the', 'their', 'theirs', 'them',
             'themselves', 'then', 'there', 'these', 'they', 'this', 'those',
             'through', 'to', 'too','under', 'until', 'up', 've', 'very', 'was',
             'we', 'were', 'what', 'when', 'where','which','while', 'who', 'whom',
             'why', 'will', 'with', 'won', 'y', 'you', "youd","youll", "youre",
             "youve", 'your', 'yours', 'yourself', 'yourselves']
```

Python

```
STOPWORDS = set(stopwordlist)
def cleaning_stopwords(text):
    return " ".join([word for word in str(text).split() if word not in STOPWORDS])
dataset['full_text'] = dataset['full_text'].apply(lambda text: cleaning_stopwords(text))
dataset['full_text'].head()
```

Fig13(A) : Removing stopwords

Tokenization and Lemmatizing the data to build Wordcloud for positive and negative tweets

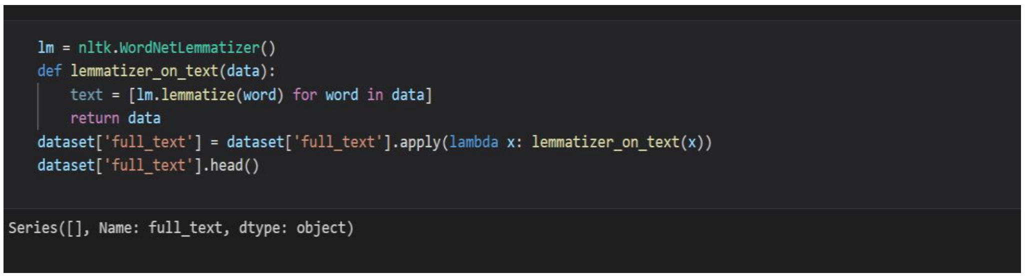


```
from nltk.tokenize import RegexpTokenizer
tokenizer = RegexpTokenizer(r'w+')
dataset['full_text'] = dataset['full_text'].apply(tokenizer.tokenize)
dataset['full_text'].head()
```

```
Series([], Name: full_text, dtype: object)
```

```
import nltk
st = nltk.PorterStemmer()
def stemming_on_text(data):
    text = [st.stem(word) for word in data]
    return data
dataset['full_text']= dataset['full_text'].apply(lambda x: stemming_on_text(x))
dataset['full_text'].head()
```

```
Series([], Name: full_text, dtype: object)
```

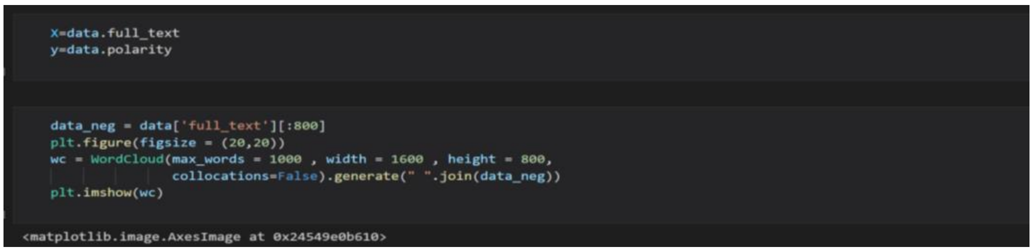


```
lm = nltk.WordNetLemmatizer()
def lemmatizer_on_text(data):
    text = [lm.lemmatize(word) for word in data]
    return data
dataset['full_text'] = dataset['full_text'].apply(lambda x: lemmatizer_on_text(x))
dataset['full_text'].head()
```

```
Series([], Name: full_text, dtype: object)
```

Fig13(B) : Tokenization and Lemmatizing

Then we have built the Wordcloud for positive and negative hashtags as well as displayed them in the form of bar charts.



```
X=data.full_text
y=data.polarity
```

```
data_neg = data['full_text'][:800]
plt.figure(figsize = (20,20))
wc = WordCloud(max_words = 1000 , width = 1600 , height = 800,
               collocations=False).generate(" ".join(data_neg))
plt.imshow(wc)
```

```
<matplotlib.image.AxesImage at 0x24549e0b610>
```

Fig13(C) : code for Wordcloud by textblob

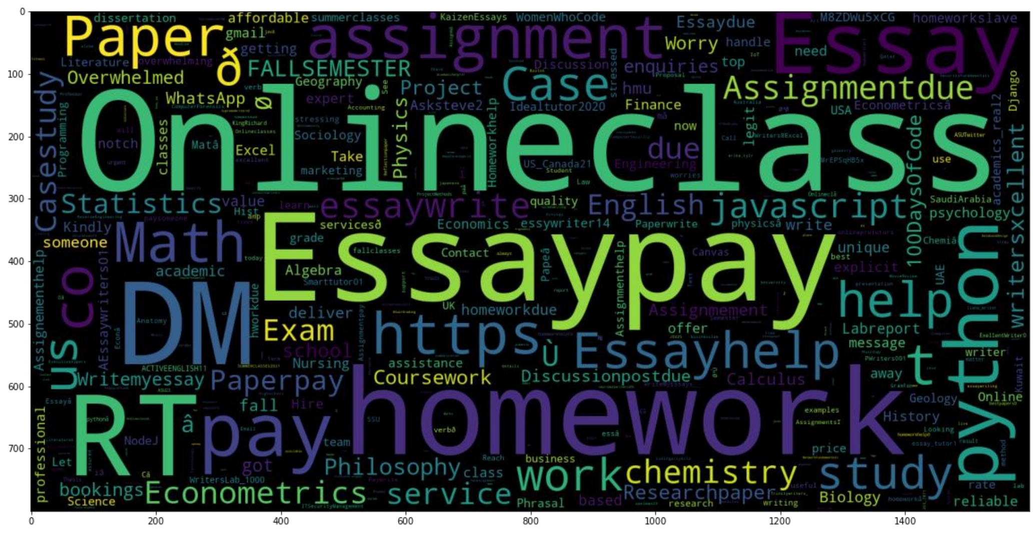
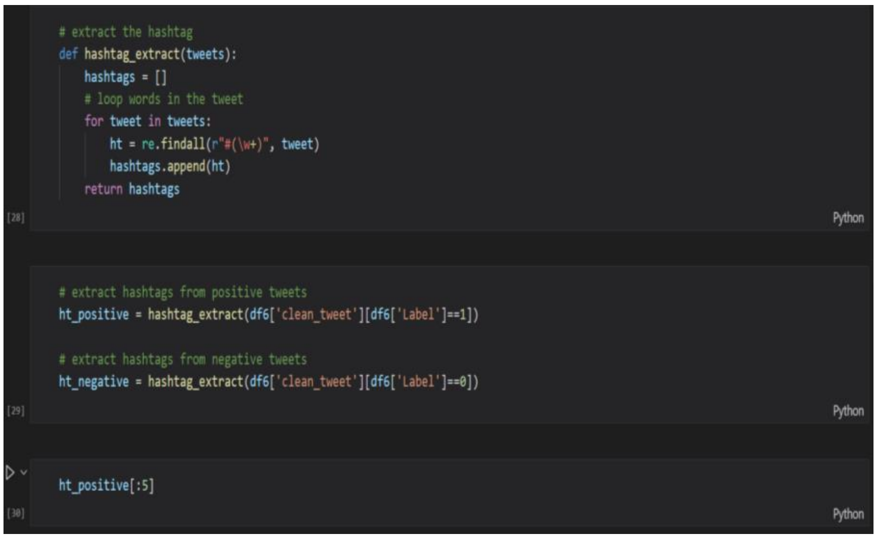


Fig13(D): Wordcloud by textblob

Now we print top ten positive and negative hashtags

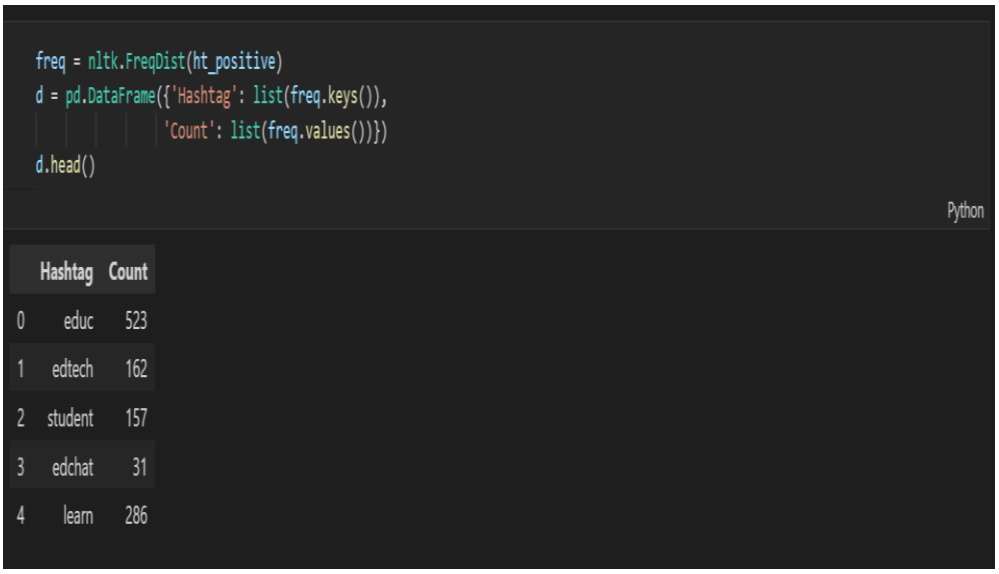


```python
# extract the hashtag
def hashtag_extract(tweets):
    hashtags = []
    # loop words in the tweet
    for tweet in tweets:
        ht = re.findall(r"#(\w+)", tweet)
        hashtags.append(ht)
    return hashtags
```

```python
# extract hashtags from positive tweets
ht_positive = hashtag_extract(df6['clean_tweet'][df6['Label']==1])

# extract hashtags from negative tweets
ht_negative = hashtag_extract(df6['clean_tweet'][df6['Label']==0])
```

```python
ht_positive[:5]
```

Fig13(E): Code for positive hashtags



```python
freq = nltk.FreqDist(ht_positive)
d = pd.DataFrame({'Hashtag': list(freq.keys()),
                  'Count': list(freq.values())})
d.head()
```

	Hashtag	Count
0	educ	523
1	edtech	162
2	student	157
3	edchat	31
4	learn	286

Fig13(F): Output for positive hashtags

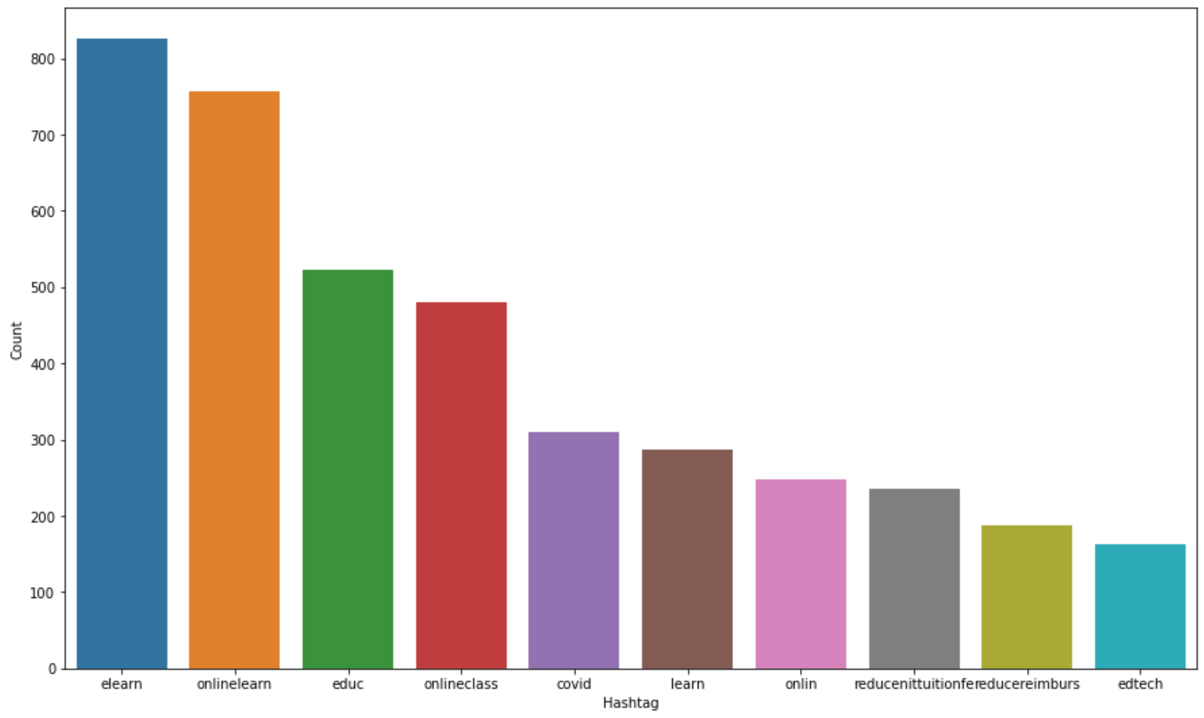
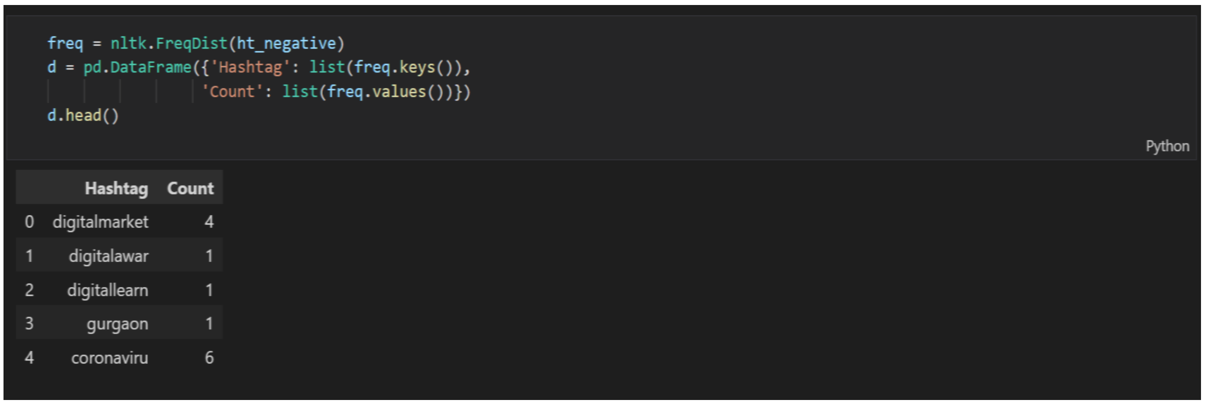


Fig13(G): Visualization of positive hashtags



```
freq = nltk.FreqDist(ht_negative)
d = pd.DataFrame({'Hashtag': list(freq.keys()),
                  'Count': list(freq.values())})
d.head()
```

	Hashtag	Count
0	digitalmarket	4
1	digitalawar	1
2	digitallearn	1
3	gurgaon	1
4	coronaviru	6

Fig13(H): Output for negative hashtags

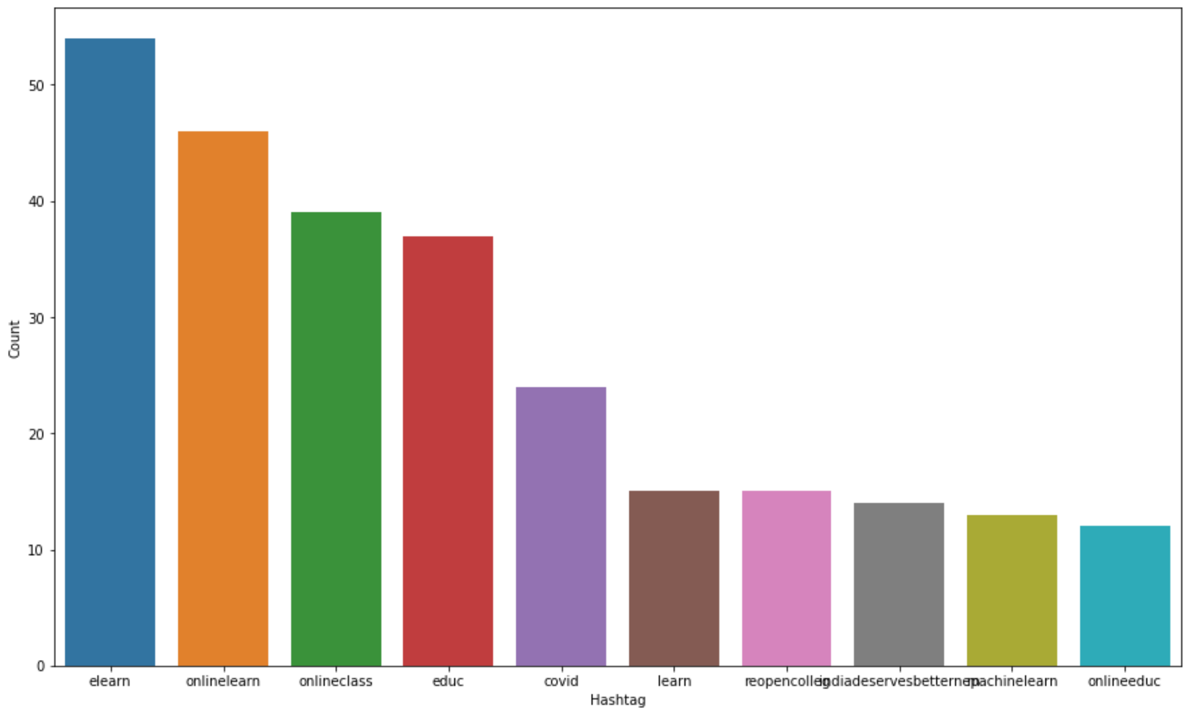


Fig 13(I): Visualization of negative hashtags

CHAPTER 5

SENTIMENT ANALYSIS USING ML CLASSIFIERS

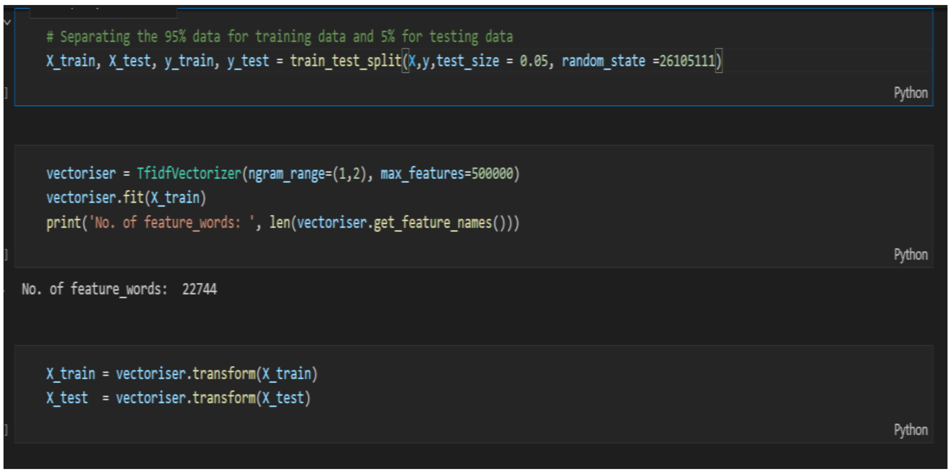
5.1 Brief Introduction

The results that we have obtained by Vader and TextBlob can be further studied and more concretely confirmed by the use of machine learning classifiers.

We have made use of Supervised machine learning to obtain the results.

There are 6 algorithms we have taken into consideration.

We have taken 95% data from the sentiment classification results for training the dataset and 5% data for testing our dataset and to obtain results.

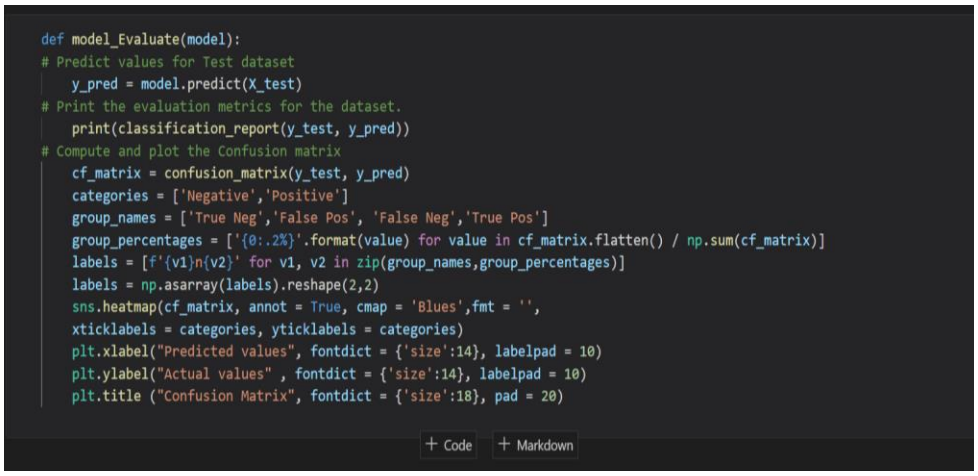


```python
# Separating the 95% data for training data and 5% for testing data
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size = 0.05, random_state =26105111)
```

```python
vectoriser = TfidfVectorizer(ngram_range=(1,2), max_features=500000)
vectoriser.fit(X_train)
print('No. of feature_words: ', len(vectoriser.get_feature_names()))
```

```
No. of feature_words:  22744
```

```python
X_train = vectoriser.transform(X_train)
X_test  = vectoriser.transform(X_test)
```



```python
def model_Evaluate(model):
# Predict values for Test dataset
    y_pred = model.predict(X_test)
# Print the evaluation metrics for the dataset.
    print(classification_report(y_test, y_pred))
# Compute and plot the Confusion matrix
    cf_matrix = confusion_matrix(y_test, y_pred)
    categories = ['Negative','Positive']
    group_names = ['True Neg','False Pos', 'False Neg','True Pos']
    group_percentages = ['{0:.2%}'.format(value) for value in cf_matrix.flatten() / np.sum(cf_matrix)]
    labels = [f'{v1}n{v2}' for v1, v2 in zip(group_names,group_percentages)]
    labels = np.asarray(labels).reshape(2,2)
    sns.heatmap(cf_matrix, annot = True, cmap = 'Blues',fmt = '',
    xticklabels = categories, yticklabels = categories)
    plt.xlabel("Predicted values", fontdict = {'size':14}, labelpad = 10)
    plt.ylabel("Actual values" , fontdict = {'size':14}, labelpad = 10)
    plt.title ("Confusion Matrix", fontdict = {'size':18}, pad = 20)
```

Fig 14: Code for ML algorithms

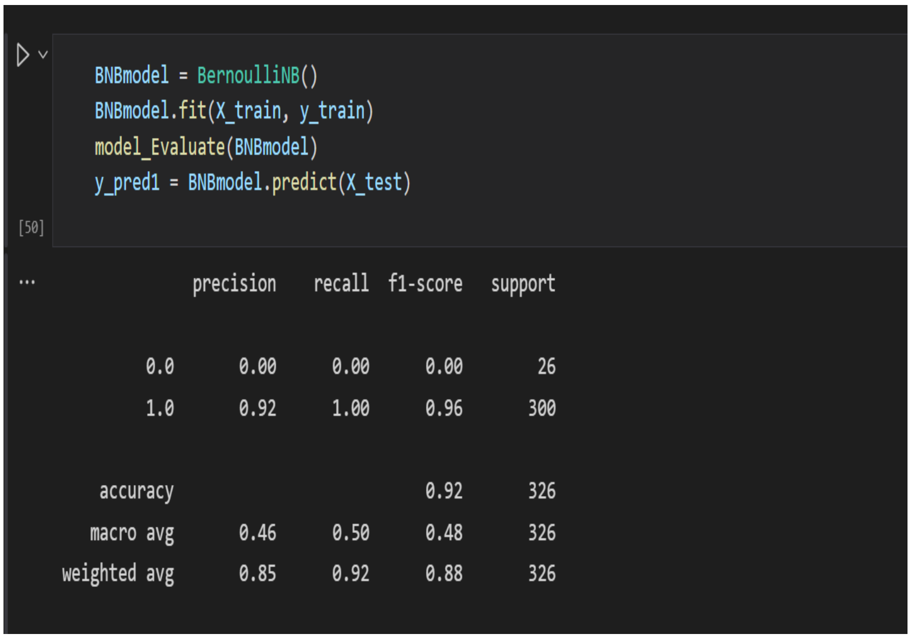
5.2 Naïve Bayes Algorithm

Naive Bayes is the simplest and fastest classification algorithm for a large chunk of data. In various applications such as spam filtering, text classification, sentiment analysis, and recommendation systems, Naive Bayes classifier is used successfully. It uses the Bayes probability theorem for unknown class prediction.

The Naive Bayes classification technique is a simple and powerful classification task in machine learning. The use of Bayes' theorem with a strong independence assumption between the features is the basis for naive Bayes classification. When used for textual data analysis, such as Natural Language Processing, the Naive Bayes classification yields good results.
Simple Bayes or independent Bayes models are other names for nave Bayes models. All of these terms refer to the classifier's decision rule using Bayes' theorem. In practice, the Bayes theorem is applied by the Naive Bayes classifier. The power of Bayes' theorem is brought to machine learning with this classifier.
Bayes' theorem states the following relationship, given class variable y and dependent feature vector x_1 through x_n:

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

Application of Naïve Bayes



```
BNBmodel = BernoulliNB()
BNBmodel.fit(X_train, y_train)
model_Evaluate(BNBmodel)
y_pred1 = BNBmodel.predict(X_test)
```

```
[50]
```

```
              precision    recall  f1-score   support

         0.0       0.00      0.00      0.00        26
         1.0       0.92      1.00      0.96       300

    accuracy                           0.92       326
   macro avg       0.46      0.50      0.48       326
weighted avg       0.85      0.92      0.88       326
```

Fig 14(A) : Output for Naïve Bayes

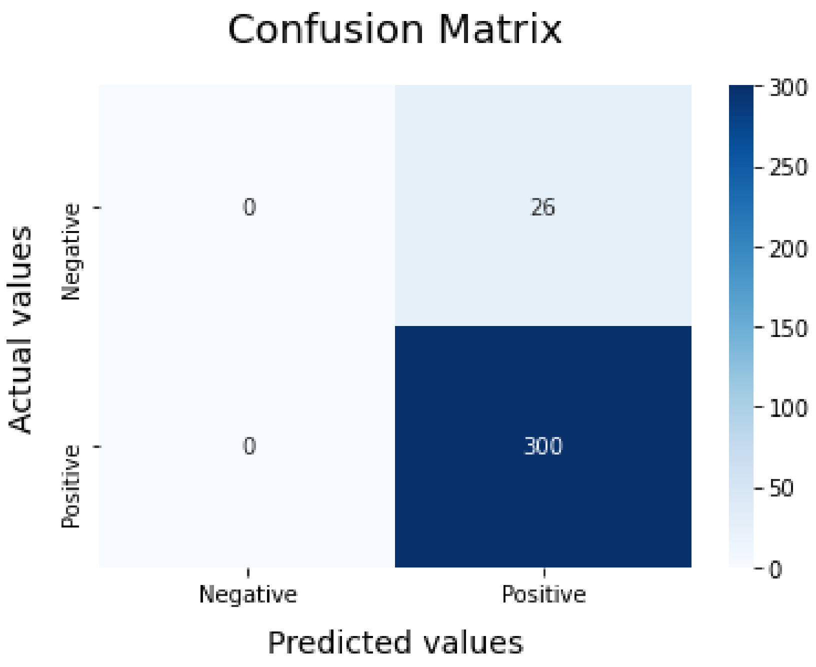
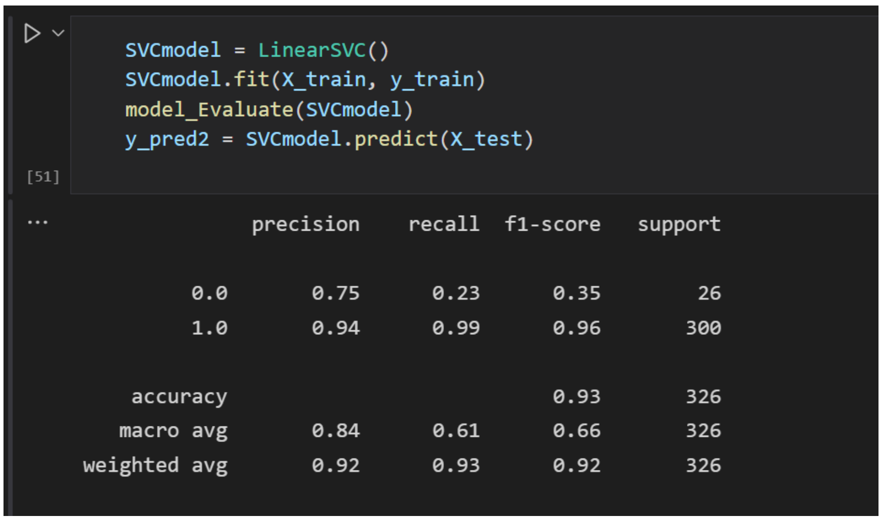


Fig 14(B) : Confusion Matrix for Naïve Bayes

5.3 Support Vector Machine

A support vector machine constructs a hyper-plane or set of hyper-planes in a high or infinite dimensional space, which can be used for classification, regression or other tasks. Intuitively, a good separation is achieved by the hyper-plane that has the largest distance to the nearest training data points of any class (so-called functional margin), since in general the larger the margin the lower the generalization error of the classifier.
A support vector machine is a supervised learning algorithm that sorts data into two categories. It is trained with a series of data already classified into two categories, building the model as it is initially trained. The task of an SVM algorithm is to determine which category a new data point belongs in. This makes SVM a kind of non-binary linear classifier.
An SVM algorithm should not only place objects into categories, but have the margins between them on a graph as wide as possible.



```
SVCmodel = LinearSVC()
SVCmodel.fit(X_train, y_train)
model_Evaluate(SVCmodel)
y_pred2 = SVCmodel.predict(X_test)
```
[51]

```
              precision    recall  f1-score   support

         0.0       0.75      0.23      0.35        26
         1.0       0.94      0.99      0.96       300

    accuracy                           0.93       326
   macro avg       0.84      0.61      0.66       326
weighted avg       0.92      0.93      0.92       326
```

Fig 14 (C): Output of SVM

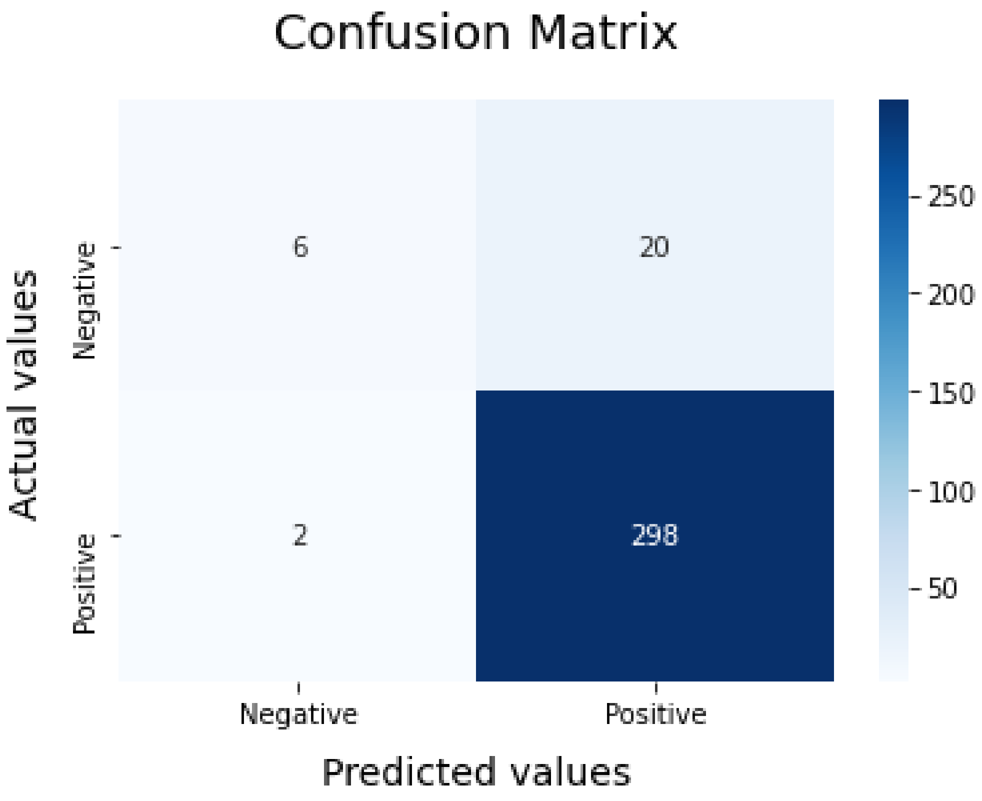
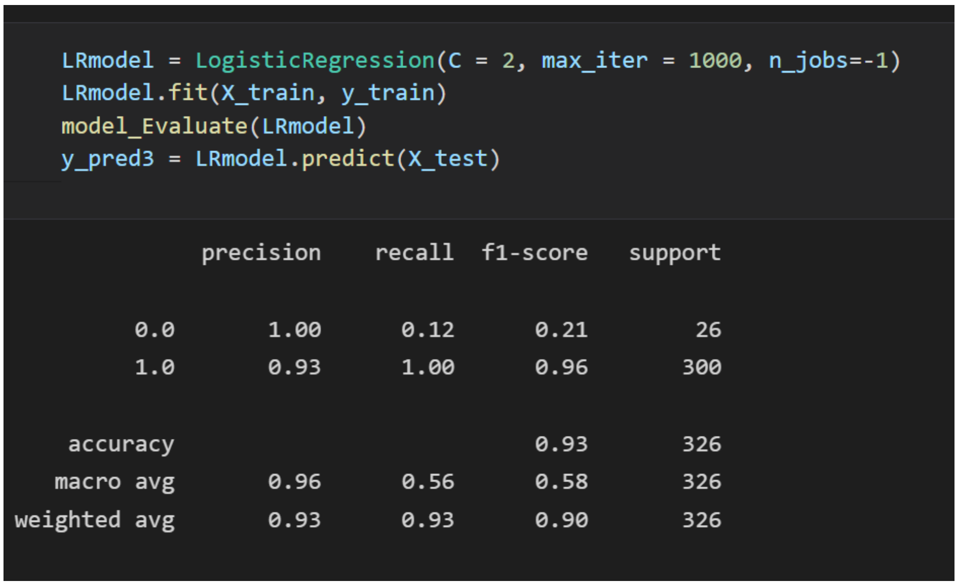


Fig14(D): Confusion matrix for SVM

5.4 Logistic Regression

Logistic regression is a supervised learning algorithm used in machine learning to predict the probability of a binary outcome. A binary outcome is limited to one of two possible outcomes. Logical regression is used in predictive modeling to analyze large datasets in which one or more independent variables can determine an outcome.

The outcome is expressed as a dichotomous variable that has one of two possible outcomes. Essentially, logistic regression works by estimating the mathematical probability that an instance belongs to a specified class -- or not.
Logistic regression uses something called the Sigmoid function to map predicted predictions and their probabilities. On a graph, if the estimated probability is greater than a pre-defined acceptance threshold, then the model will predict that the instance belongs to that class. If the estimated probability is less than the pre-defined threshold on the graph, then the model will predict the instance does not belong to the class.



```
LRmodel = LogisticRegression(C = 2, max_iter = 1000, n_jobs=-1)
LRmodel.fit(X_train, y_train)
model_Evaluate(LRmodel)
y_pred3 = LRmodel.predict(X_test)
```

```
              precision    recall  f1-score   support

         0.0       1.00      0.12      0.21        26
         1.0       0.93      1.00      0.96       300

    accuracy                           0.93       326
   macro avg       0.96      0.56      0.58       326
weighted avg       0.93      0.93      0.90       326
```

Fig14(E): Output for LR

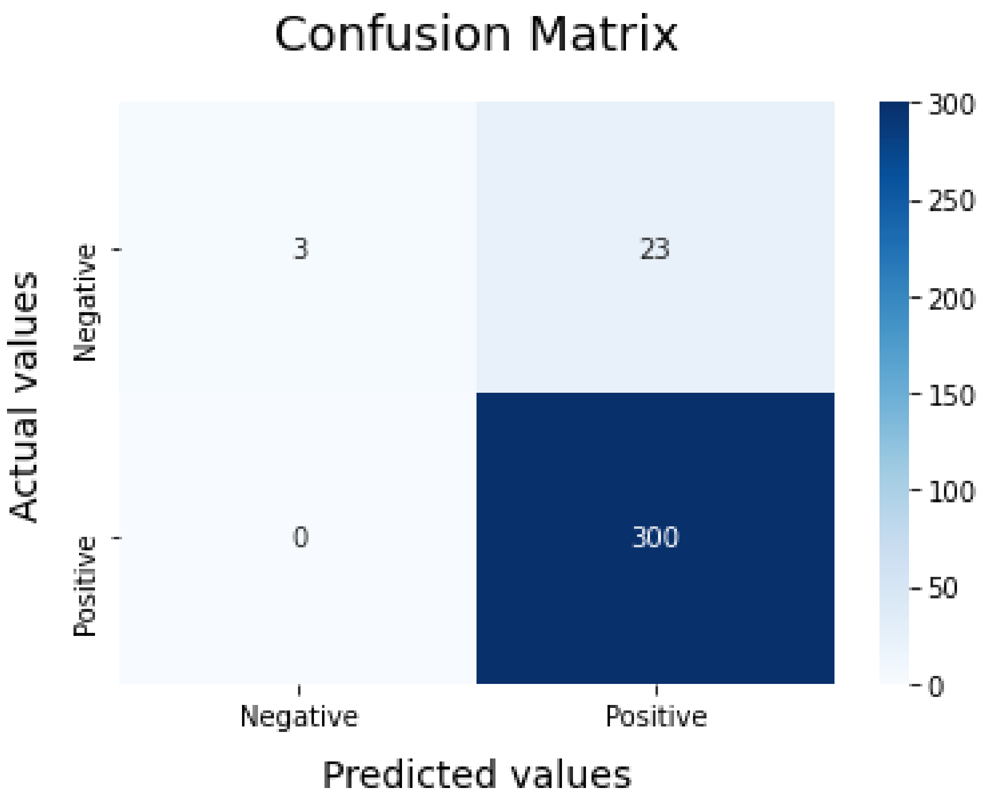
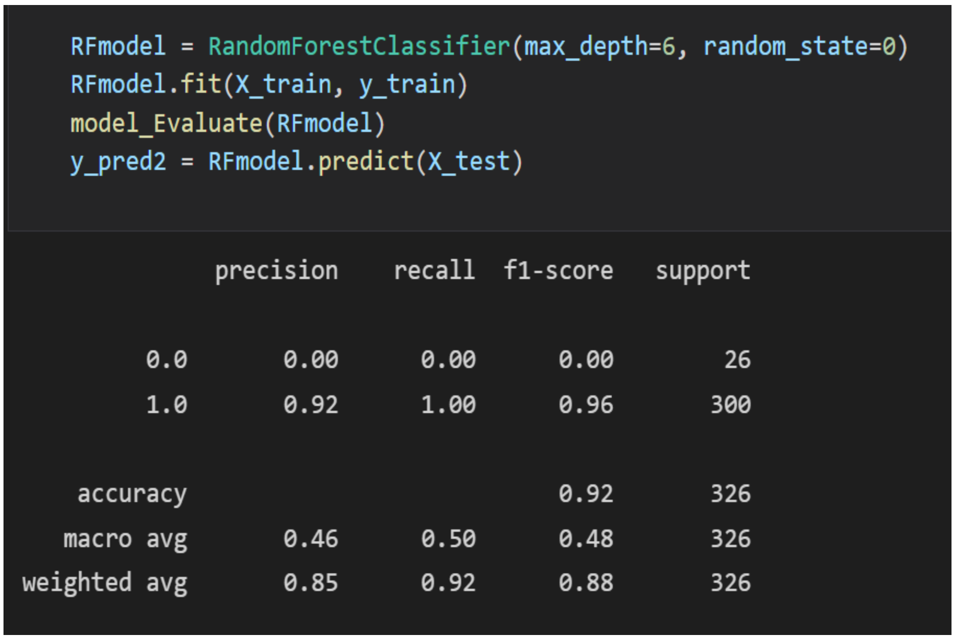


Fig 14(F): Confusion matrix for LR

5.5 Random Forest

Random forest is a consensus algorithm used in supervised machine learning (ML) to solve regression and classification problems. Each random forest is comprised of multiple decision trees that work together as an ensemble to produce one prediction.

A decision tree is a logical construct that resembles a flowchart and illustrates a series of if-else statements. An important purpose of using random forest is to compensate for the limitations of decision tree algorithms by mapping multiple trees and using the forest's average output (statistical mean).
Random forest algorithms can produce acceptable predictions even if individual trees in the forest have incomplete data. Statistically, increasing the number of trees in the ensemble will correspondingly increase the precision of the outcome.



```
RFmodel = RandomForestClassifier(max_depth=6, random_state=0)
RFmodel.fit(X_train, y_train)
model_Evaluate(RFmodel)
y_pred2 = RFmodel.predict(X_test)
```

```
              precision    recall  f1-score   support

         0.0       0.00      0.00      0.00        26
         1.0       0.92      1.00      0.96       300

    accuracy                           0.92       326
   macro avg       0.46      0.50      0.48       326
weighted avg       0.85      0.92      0.88       326
```

Fig 14(G) : Output for Random Forest

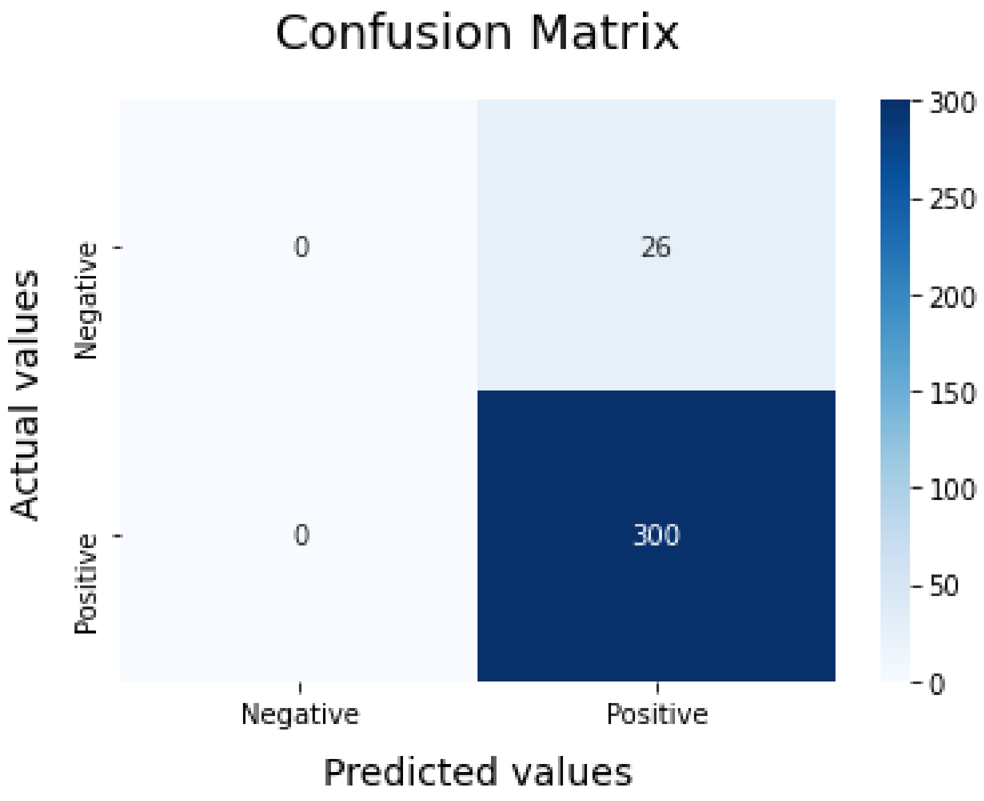
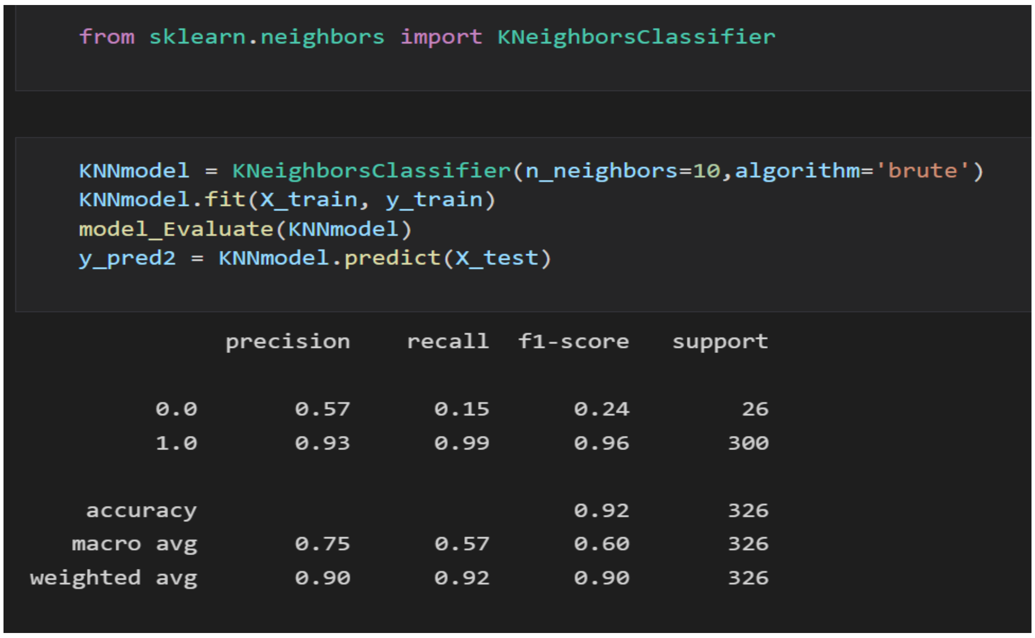


Fig 14(H): Confusion Matrix For Random Forest

5.6 K-Nearest Neighbor

A k-nearest-neighbor algorithm, often abbreviated k-nn, is an approach to data classification that estimates how likely a data point is to be a member of one group or the other depending on what group the data points nearest to it are in.
The k-nearest-neighbor is an example of a "lazy learner" algorithm, meaning that it does not build a model using the training set until a query of the data set is performed.
A k-nearest-neighbor is a data classification algorithm that attempts to determine what group a data point is in by looking at the data points around it.
An algorithm, looking at one point on a grid, trying to determine if a point is in group A or B, looks at the states of the points that are near it. The range is arbitrarily determined, but the point is to take a sample of the data. If the majority of the points are in group A, then it is likely that the data point in question will be A rather than B, and vice versa.
The k-nearest-neighbor is an example of a "lazy learner" algorithm because it does not generate a model of the data set beforehand. The only calculations it makes are when it is asked to poll the data point's neighbors. This makes k-nn very easy to implement for data mining.



```
from sklearn.neighbors import KNeighborsClassifier
```

```
KNNmodel = KNeighborsClassifier(n_neighbors=10,algorithm='brute')
KNNmodel.fit(X_train, y_train)
model_Evaluate(KNNmodel)
y_pred2 = KNNmodel.predict(X_test)
```

```
              precision    recall  f1-score   support

         0.0       0.57      0.15      0.24        26
         1.0       0.93      0.99      0.96       300

    accuracy                           0.92       326
   macro avg       0.75      0.57      0.60       326
weighted avg       0.90      0.92      0.90       326
```

Fig14(I):Output for KNN

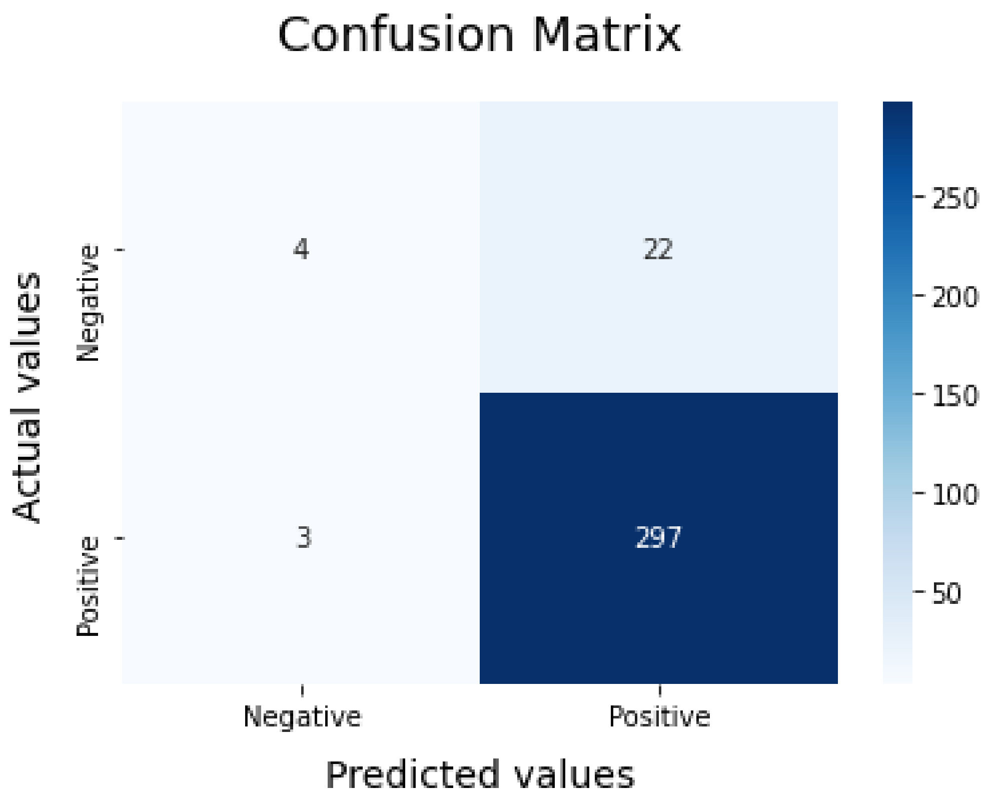
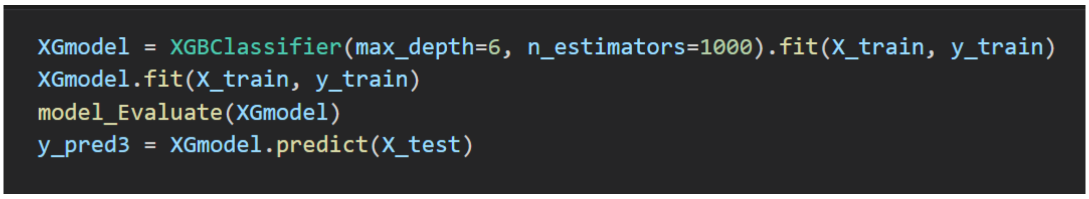


Fig 14(J): Confusion matrix for KNN

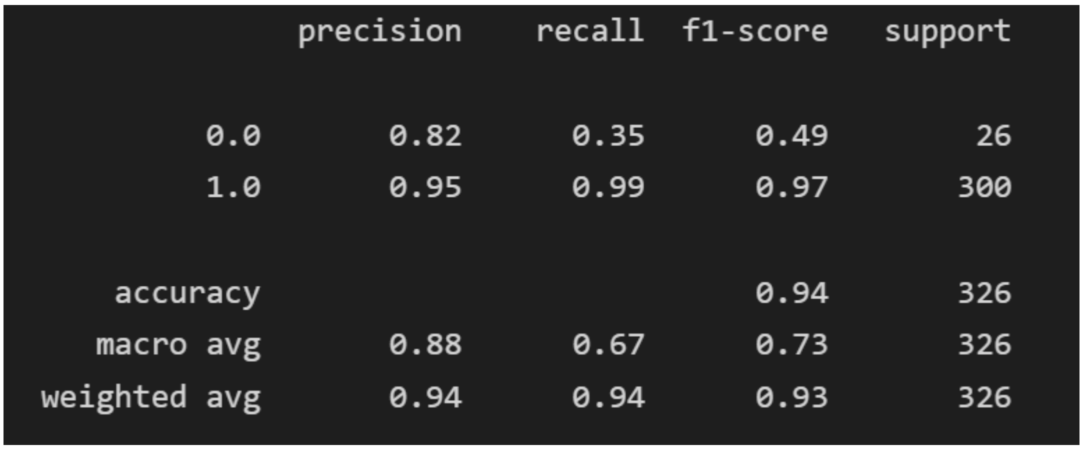
5.7 XGBoost

XGBoost is a popular and efficient open-source implementation of the gradient boosted trees algorithm. Gradient boosting is a supervised learning algorithm, which attempts to accurately predict a target variable by combining the estimates of a set of simpler, weaker models.
When using gradient boosting for regression, the weak learners are regression trees, and each regression tree maps an input data point to one of its leafs that contains a continuous score.

XGBoost minimizes a regularized (L1 and L2) objective function that combines a convex loss function (based on the difference between the predicted and target outputs) and a penalty term for model complexity (in other words, the regression tree functions). The training proceeds iteratively, adding new trees that predict the residuals or errors of prior trees that are then combined with previous trees to make the final prediction. It's called gradient boosting because it uses a gradient descent algorithm to minimize the loss when adding new models.



```
XGmodel = XGBClassifier(max_depth=6, n_estimators=1000).fit(X_train, y_train)
XGmodel.fit(X_train, y_train)
model_Evaluate(XGmodel)
y_pred3 = XGmodel.predict(X_test)
```



```
              precision    recall  f1-score   support

         0.0       0.82      0.35      0.49        26
         1.0       0.95      0.99      0.97       300

    accuracy                           0.94       326
   macro avg       0.88      0.67      0.73       326
weighted avg       0.94      0.94      0.93       326
```

Fig14(K): Output for XGBoost

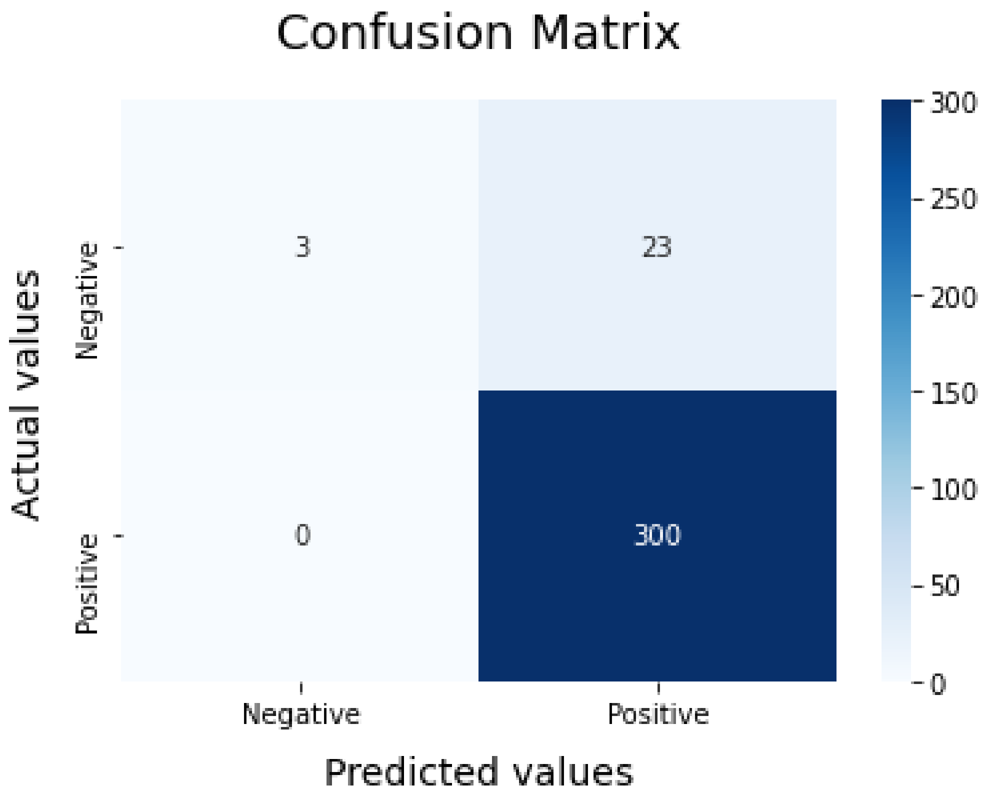


Fig 14(L): Confusion Matrix for XGBoost

CHAPTER 6

RESULTS

This section presents the details of the experiments conducted in this research, describing the setup of the experiments and the results of the models applied in all the experiments.

Experimental setup: All experiments were conducted using a ASUS ASPIRE 7 with a 2.7 GHz HEXA-Core AMD 5500 U and 8 GB 3200 MHz DDR4 memory.

Classifier	Accuracy	F-Score	Precision	Recall
NB	0.92	0.96	0.92	1.00
SVM	0.93	0.96	0.94	0.99
LR	0.93	0.96	0.93	1.00
XGB	0.94	0.97	0.95	0.99
RF	0.92	0.96	0.92	1.00
KNN	0.92	0.96	0.93	0.99

TABLE 1: Result of the sentiment analysis. Highest accuracy is achieved by XGBoost.

References

[1] AntonPak, Oyelola A.Adegboye,,"Economic consequences of the covid-19 outbreak: The need for epidemic preparedness"

[2] Yahya Almurtadha , Sentiment Analysis to Measure Public Response to Online Education During Coronavirus Pandemic

[3] Aljabri, M.; Chrouf, S.M.B.; Alzahrani, N.A.; Alghamdi, L.; Alfehaid, R.; Alqarawi, R.; Alhuthayfi, J.; Alduhailan, N. Sentiment Analysis of Arabic Tweets Regarding Distance Learning in Saudi Arabia during the COVID-19 Pandemic. Sensors 2021, 21, 5431. Doi: 10.3390/s21165431

[4] Filiz Angay Kutluk, Mustafa Gulmez, "A research about distance education students' satisfaction with education quality at an accounting program" Published by Elsevier Ltd. Selection and/or peer review under responsibility of Prof. Dr. Hüseyin Uzunboylu doi: 10.1016/j.sbspro.2012.05.556

[5] Osmanoglu,U.O., Atak,O.N., Caglar,K., Kayhan, H. &Can, T.C. (2020). Sentiment Analysis for Distance Education, Doi: 10.31681/jetol.663733

[6] Nimasha Arambepola, Analysing the Tweets about Distance Learning during COVID-19 Pandemic using Sentiment Analysis, ISSN 2756-9160 / November 2020.

[7] Aiyanyo, I.D.; Samuel, H.; Lim, H. Effects of the COVID-19 Pandemic on Classrooms: A Case Study on Foreigners in South Korea Using Applied Machine Learning. Sustainability 2021, 13, 4986. Doi:10.3390/su13094986