VISVESVARAYA TECHNOLOGICAL UNIVERSITY,
BELGAUM
"Jnana Sangama", Belgaum-590018

Minor Project (IS6C06)
titled

Schmaltz Surveyor - Sentimental Analysis of Social Media

Submitted in partial fulfillment of the requirement for the completion of 6th semester of

BACHELOR OF ENGINEERING
IN
INFORMATION SCIENCE & ENGINEERING

by

Nithyashree Arunachalam	**4NI19IS058**
Pradyoth P	**4NI19IS062**
Shashank BU	**4NI19IS086**
Tejasvini SJ	**4NI19IS106**

Under the Guidance of

Mrs. Nandini BM
Assistant Professor
Department of ISE, NIE Mysuru

The National Institute of Engineering
Mysuru - 570008

Department of Information Science and Engineering
NIE, Mysuru - 570008

2021-2022

THE NATIONAL INSTITUTE OF ENGINEERING
MYSURU - 570008

Department of Information Science and Engineering

CERTIFICATE

Certifies that the project work titled "**Schmaltz Surveyor**" is a bonafide work carried out by

Nithyashree Arunachalam (4NI19IS058)

Pradyoth P (4NI19IS062)

Shashank BU (4NI19IS086)

Tejasvini SJ (4NI19IS106)

in partial fulfillment for the requirements of the sixth semester BE in Information Science & Engineering prescribed by The National Institute of Engineering, Autonomous Institution under Visvesvaraya Technological University, Belagavi. It is certified that all corrections/suggestions indicated for Internal Assessment have been incorporated. The Project report has been approved as it satisfies the academic requirements concerning the project work prescribed for the sixth semester in Minor Project (IS6C06).

Signature of Guide	Signature of HOD	Signature of Principal
(Mrs. Nandini BM)	(Dr. S Kuzhalvaimozhi)	(Dr. Rohini Nagapadma)

Name of the Examiner: Signature:

1.

2.

ABSTRACT OF THE PROJECT

Social media has gained immense popularity and has become a major global platform to stay connected as well as express opinions. A huge amount of content is created on various topics and comments are posted on these platforms daily. The feedback received on a certain piece of content can be either negative or positive. Other than this, receiving negative feedback, on various occasions might affect the mental health of the content creators and, in some cases, might also lead to cyberbullying. Social media is a necessity in today's time to stay connected, informed, and relevant and therefore such issues must be tackled.

Sentimental Analysis (also known as Opinion Mining in this case) reads people's sentiments or emotions towards particular things or topics. Sentiment analysis is a machine learning tool that will help analyze and categorize the texts (written content and comments) as positive or negative. Sentiments cannot be directly predicted by analyzing the text and therefore the texts are vectorized or converted into numeric values. The text is converted using vectorization methods in NLP (Natural language Processing) such as Bag of Words or TF- IDF(Term Frequency–Inverse Document Frequency). This data is then put through machine learning algorithms such as Support Vector Machine, Logistic Regression, Random Forest, and K Nearest Neighbours. Based on the result after using these methods, the most accurate machine learning model is selected. Python would be the programming language across the whole project.

The number of positive or negative texts and comments can be used to analyze people's reception or opinion on what particular piece of content is getting. Further, it can also be used by social media platforms to detect negative comments and delete them or classify content based on the number of positive or negative comments.

ACKNOWLEDGEMENT

The success and the outcome of this project required a lot of guidance and assistance from many people and we are incredibly fortunate to have got this all along with the completion of project work.

We express our profound thanks to **Dr. Rohini Nagapadma,** Principal, NIE, Mysuru for his much-needed moral support and encouragement.

We are grateful to **Dr. S Kuzhalvaimozhi,** Professor & HOD, Information Science and Engineering, NIE for her support and encouragement in facilitating the progress of this work.

We sincerely extend our thanks to our Project Coordinator and Project Guide **Mrs. Nandini BM,** Assistant Professor in the Department of ISE, for her guidance, technical expertise, encouragement, and timely help in making this project a reality.

We would like to extend our sincere regards to all the teaching and non-teaching staff of the Department of ISE, NIE Mysuru for their assistance and timely support.

We would also like to give credit to the authors of the various resources which were made available on the Internet for our reference.

Nithyashree Arunachalam (4NI19IS058)
Pradyoth P (4NI19IS062)
Shashank BU (4NI19IS086)
Tejasvini SJ (4NI19IS106)

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1:

INTRODUCTION

Social Media has become a necessity to remain relevant. It gives everyone a platform to express their opinions and views on a larger scale. Since opinions are always going to be conflicting, much content on the internet can be segregated into positive and negative. Thousands and millions of data are produced regularly by these social media sites which can help us to look at the overall effect i.e we can calculate the amount of negative and positive content present, be it the opinions of a person towards a particular topic or towards a video.

The impact of social media is very intense. Severely negative comments have the potential to hurt and ruin the mental health of a person. As much as it is essential to express our individual opinions to keep a conversation going, there is no need to express and give out negative comments to random content creators and other people who are doing nothing but entertaining billions of people around the globe anonymously.

Machine learning is an emerging scientific field in data science dealing with the ways in which machines learn from experience. Machine learning (ML) is a type of artificial intelligence that allows software applications to become more accurate at predicting outcomes without being explicitly programmed to do so. Machine learning algorithms use historical data as input to predict new output values. Sentimental analysis, also known as opinion mining, is a method of analyzing and classifying data on the basis of the emotion they express with the help of machine learning models and algorithms present. The main aim of this project is to use machine learning and be able to predict a positive and negative comment or post in the future and create software of sorts that can be employed that blocks a user from typing negative comments. An already classified dataset is taken which is used to train the model and then the model will be further will able to predict the sentiment. Schmaltz means excessive sentimentality. Therefore, this project is called ‘Schmaltz Surveyor’ which on its own suggests that the sentiments are going to be surveyed here.

1.1 Our Objective

The main objective of this project is to be able to predict comments as positive or negative. This is done using an ML model which has been trained on a dataset that already has comments labeled based on positive or negative. This model will be applied at the backend while the frontend will have a functionality wherein the user will not be able to post a negative tweet/content. The user's text will go through the model at the backend that will predict its sentiment and based on it the user will be able to post it.

In the first half of the project, a set of 4 classifiers namely:

1. Logistic Regression
2. Support Vector Machine
3. K Nearest Neighbor
4. Random Forest

which are predominantly used to categorize will be used to train the dataset. The models then predict a set of tweets which have already been labeled and then compare its prediction against the actual results to calculate the accuracy and create a confusion matrix. The model with the highest accuracy score is taken as the main model to predict the sentiment.

1.2 Purpose

The purpose of this project is to analyze the amount of positive and negative influence present on the social media platform(Twitter in this case) and further be able to predict the sentiment of tweets/posts made by the users in future.This sentiment analysis will help us in developing an easy to use software which will prevent the user from putting out negative tweets out there. By crunching large volumes of data, machine learning technology can help to reduce the amount of negativity present on such social media platforms and make it a healthy and more positive environment for the users.

1.3 Existing System

Online social media platforms provide users with the ability to express their opinions anonymously which gives them the power to post and comment on anything that they want to. While there are positive conversations that are happening, there is a lot of chaos that is created by people using extremely harsh words which affects people. There is no check done internally by the platform that is done before a user is allowed to post or tweet. In a world where even the slightest ignorance of people while airing an opinion results in a controversy, it is necessary to filter content that people are allowed to put in to not create unnecessary drama and targeting of people.

Currently, the only functionality available on platforms such as Twitter is that of reporting offensive tweets/comments or completely blocking the users creating chaos. After the tweet/comment is reported, it is in most cases manually reviewed and then the decision is taken if it should be removed or not.

There is also a problem with this, sometimes to bring down other people a group might target an individual. In this case, the group might report the target individuals' accounts again and again. According to most policies levied by these platforms, an account that is reported repeatedly gets deleted on its own even if they were spreading positive content.

1.4 Proposed System

The proposed system is to develop an easy-to-use software in which when a user is about to post something, that piece of text goes through the trained model in the backend, and its sentiment is predicted. If the sentiment is negative the user is barred from posting that piece of content.

In the frontend, the main page has a dialogue box where the user can input keywords surrounding a particular tweet. The backend will then extract tweets from Twitter using tweepy which will fetch tweets containing those keywords. Then, the tweet goes through preprocessing of data and vectorization which then goes through the chosen classifier and the sentiment is predicted.

CHAPTER 2:
LITERATURE SURVEY

Xing Fang, and Justin Zhan[1], worked on Sentiment Analysis on Amazon Online Products Review by collecting data from amazon.com. They figured out the issues of categorical sentiment polarity using Machine Learning Algorithms. They used many libraries that hold Naive Bayesian, SVM, and Random Forest.

For this paper, reviews have been taken about amazon products from amazon.com. The reviews have been labeled and segregated into positive and negative reviews using machine learning algorithms like Naive Bayesian, SVM, and Random Forest. The accuracy in each case has been calculated by the author and then the prediction is done on the test set.

Faizan[2] built a model for the analysis of feeling using the KNN algorithm with unigram, bigram, and n-gram features. They then performed training and testing of their model on the US Airlines data set, for which they attained an accuracy of 65.33%.

For this paper, twitter is chosen as a social networking site. The posts in this social networking site are known as tweets. In this paper, the methods of preprocessing and extraction of twitter data using python and then train as well as test this data against a classifier in order to derive the sentiments behind tweets is done The classifier used here in KNN algorithm. In this extensive analysis of data has been done using visualization and then by the creation of the confusion matrix.

Chirag Kariya and Priti Khodke's[3] paper explains various steps involved in the analysis of Twitter sentiments along with the various tools that are used to perform Twitter sentiment analysis. Amongst the various algorithms available, the KNN algorithm is used to increase the efficiency of sentiment analysis whereas Naive Bayes for simple and efficient sentiment analysis by classifying the tweets as either positive, negative, or zero.

This paper presents an idea of extracting sentiments out of the tweet and an approach towards classifying the tweets as positive,negative and neutral. Generally the tweets being unstructured in format, first of all the tweet needs to be converted into the structured format. In this paper, tweets are resolved using pre-processing phase and access to tweets has been accomplished via libraries using Twitter API. This data is put through KNN algorithm and Naive Bayes to divide the sentiments into positive and negative.

Akshay Amolik et al.[4] proposed sentiment analysis and they accurately classified tweets by using Feature Vector and classifiers like Naïve-Bayesian and SVM. Exception of lower recall and accuracy, Naïve Bayesian has better precision as a comparison to SVM. SVM gives better results when it comes to accuracy. With the increase of training data, the accuracy of classification will also increase.

This paper presents an idea of extracting sentiments out of the tweet and an approach towards classifying the tweets as positive, negative, and neutral. Generally, the tweets being unstructured in format, first of all the tweet needs to be converted into a structured format. In this paper, tweets are resolved using the pre-processing phase, and tweets have been accessed via libraries using Twitter API. This data is put through SVM algorithm and Naive Bayes to divide the sentiments into positive and negative. The accuracy in each case has been calculated by the author and then the prediction is made on the test set.

CHAPTER 3:

SYSTEM REQUIREMENTS

Hardware Requirements:

- Processor- 5th gen Intel Core i3 equivalent or greater
- RAM- 8GB or above
- Keyboard and a Mouse
- Monitor

Software Requirements:

- Windows 11, macOS, or GNU/Linux.
- Visual Studio Code
- HTML, CSS, and Javascript
- Python 3.10
- Flask
- Chrome
- Jupyter Notebook
- Scikit-learn
- Natural Language Toolkit (NLTK)
- Modules used were Pickle, numPy, pandas, re, Tweepy, matplotlib, WordCloud.

CHAPTER 4:

SYSTEM DESIGN

The project has been mainly divided into two parts.

1. Analysing the best model available to predict sentiments.
2. Live sentiment Analysis Web Application.

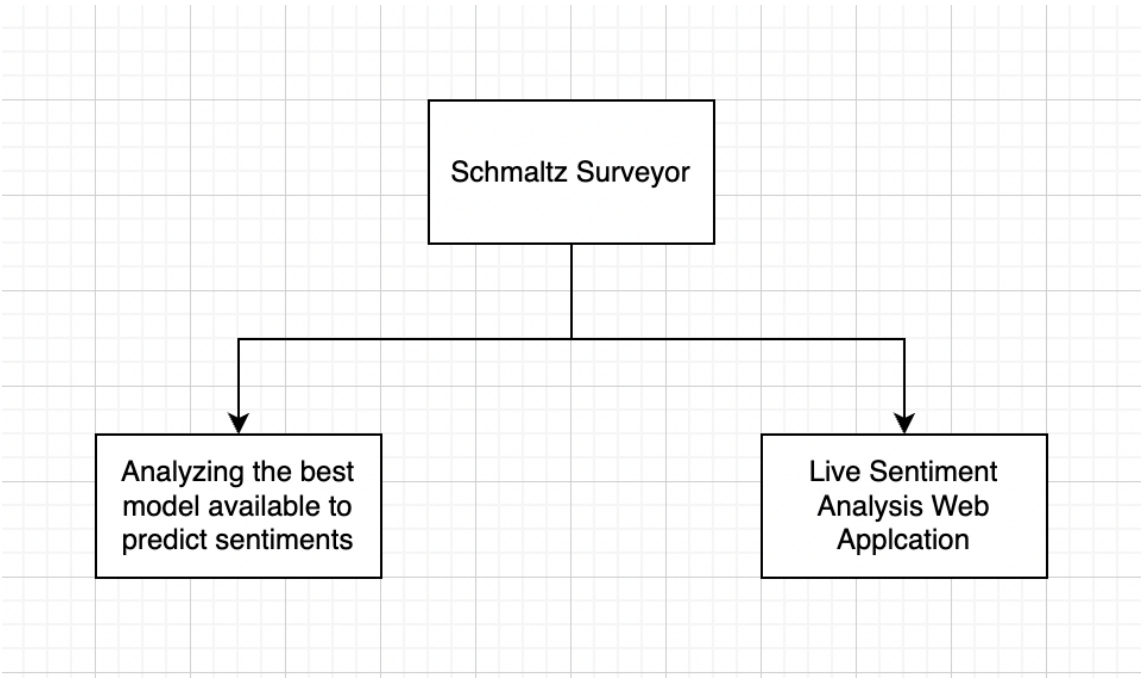


Fig 4.1 Division of project

The first part mainly deals with analyzing the efficiency of the methods through which Sentiment Analysis has been performed like we've seen in the Literature Survey and combining the results to find the best model for our use case.

The second part mainly deals with utilizing the information we got in the first part, deploying it at the backend, and making the model we chose to predict sentiments as and when we ask for from the User Interface provided through the Frontend, basically a Live Sentiment Analysis Web Application. The application provides an output that shows the sentiment the user is trying to display through that particular message.

System Design of any Machine Learning Application:

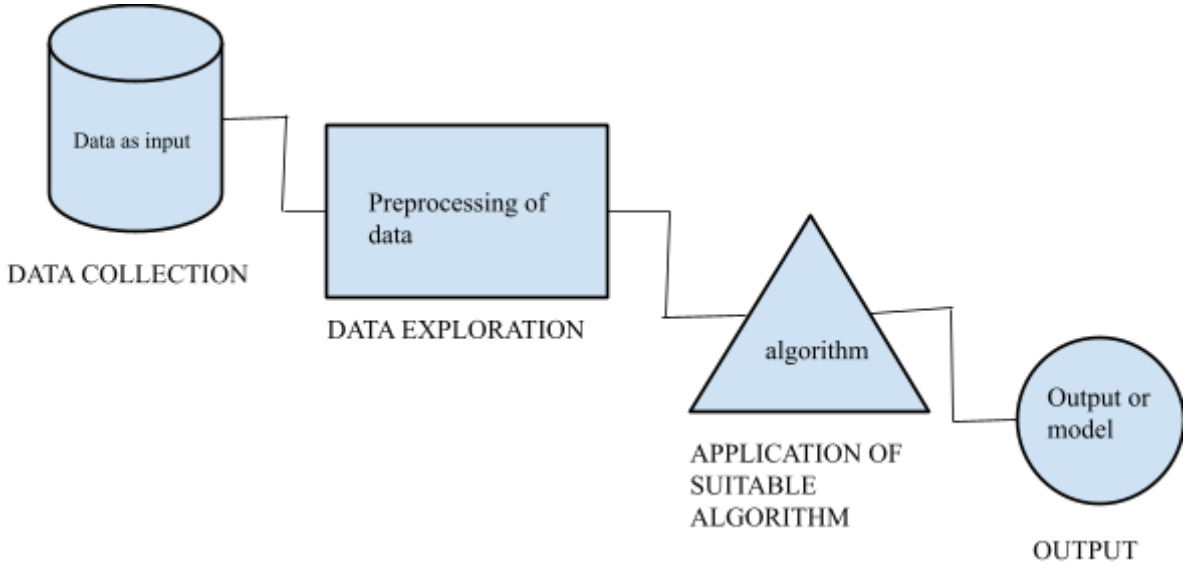


Fig 4.2 System Design for ML applications

The above figure shows the typical design of any Machine Learning Application in a generalized manner. Based on the same, we've created our Project Flow as shown.

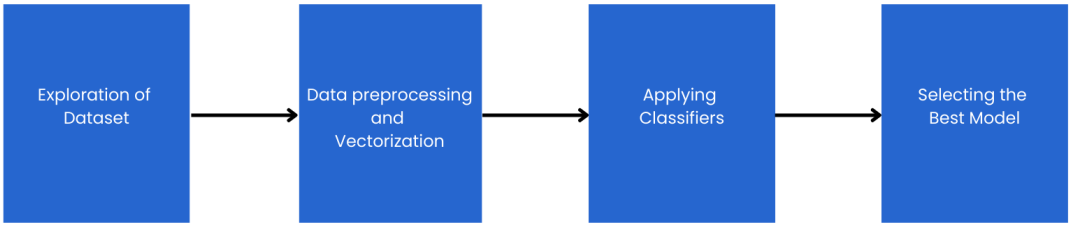


Fig 4.3 Project Flow

1. Exploration of Dataset
2. Data preprocessing and Vectorization
3. Applying Classifiers
4. Selecting the best Model.

CHAPTER 5:

SYSTEM IMPLEMENTATION

In this project, we have extensively used the Pandas module for creation and manipulation of dataframes. Here is a brief description about its function:

Pandas

pandas.DataFrame.sample

DataFrame.sample(n=None,frac=None,replace=False,weights=None,

random_state=None, axis=None, ignore_index=False)[source].

You can use random_state for reproducibility.

Parameters

n : int, optional

Number of items from axis to return. Cannot be used with frac. Default = 1 if frac = None. fracfloat, optional

Fraction of axis items to return. Cannot be used with n.

replacebool, default False

Allow or disallow sampling of the same row more than once.

The data set was initially taken from Kaggle and it contains three columns:

1. ID: Id number of the tweet
2. Tweet
3. Label: Positive sentiment is represented by 1 while negative sentiment is represented by 0.

```
3 print(df)

[5]

...          id  label                                              tweet
0             1      1   @user when a father is dysfunctional and is s...
1             2      1  @user @user thanks for #lyft credit i can't us...
2             3      1                                bihday your majesty
3             4      1  #model   i love u take with u all the time in ...
4             5      1             factsguide: society now    #motivation
...         ...    ...                                                ...
31957     31958      1  ate @user isz that youuu?ð□□□ð□□□ð□□□ð□□□ð□□□ð...
31958     31959      1    to see nina turner on the airwaves trying to...
31959     31960      1  listening to sad songs on a monday morning otw...
31960     31961      0  @user #sikh #temple vandalised in in #calgary,...
31961     31962      1                  thank you @user for you follow

[31962 rows x 3 columns]
```

Fig 5.1 Snippet of dataset used

Exploration of Dataset

This step involved finding a dataset suitable for all our needs and requirements while keeping a large amount of data, so we could train the model better.

The Dataset referred to for training the models is a labeled dataset of 31,962 tweets. The dataset is provided in the form of a CSV file with each line storing a tweet id, its label, and the tweet.

Exploratory Data Analysis was also performed, where we got the following outputs.

1. Common Frequent Word Count Visualization

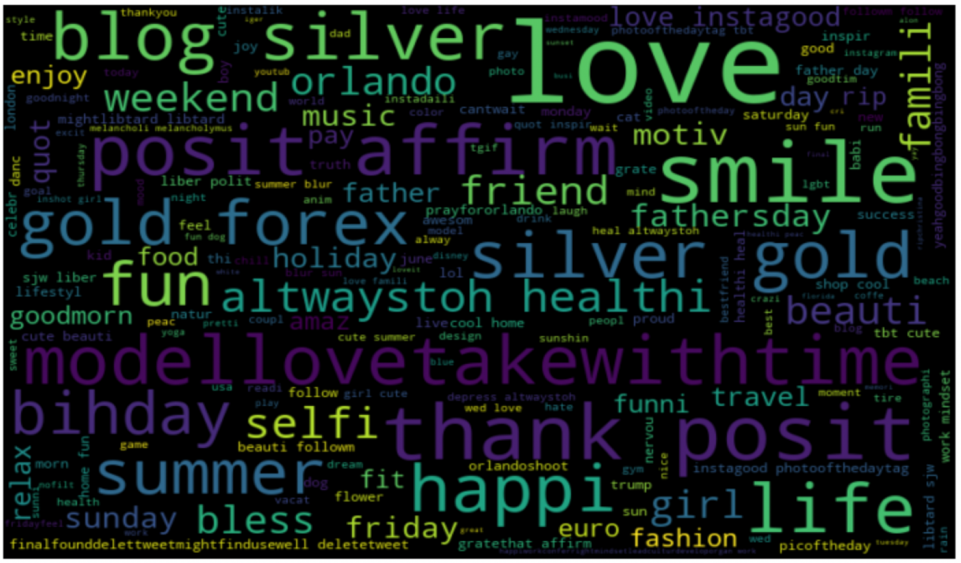


Fig 5.2 Frequent Word visualization

In the above visualization, the words that have been frequently used in the document have frequently appeared are enlarged. The frequency of the word in the document is proportional to size of the word in the figure. The more common ones are large whereas less common ones are small.

2. Frequent Word Visualization for positive Labeled tweets

Fig 5.3 Frequent Word visualization for Positive Tweets

In the above visualization, the words that have been frequently used in the document have frequently appeared are enlarged. The frequency of the word in the document is proportional to size of the word in the figure but in this we are doing it only for the tweets that have been labeled as positive (i.e have 1 in the label field). The more common ones are large whereas less common ones are small.

3. Frequent Word Visualization for negative Labeled tweets

Fig 5.4 Frequent Word visualization for Negative Tweets

In the above visualization, the words that have been frequently used in the document have frequently appeared are enlarged. The frequency of the word in the document is proportional to size of the word in the figure but in this we are doing it only for the tweets that have been labeled as positive (i.e have 1 in the label field). The more common ones are large whereas less common ones are small.

Now, we could also get the information about the top 10 used hashtags as shown in the graph below.

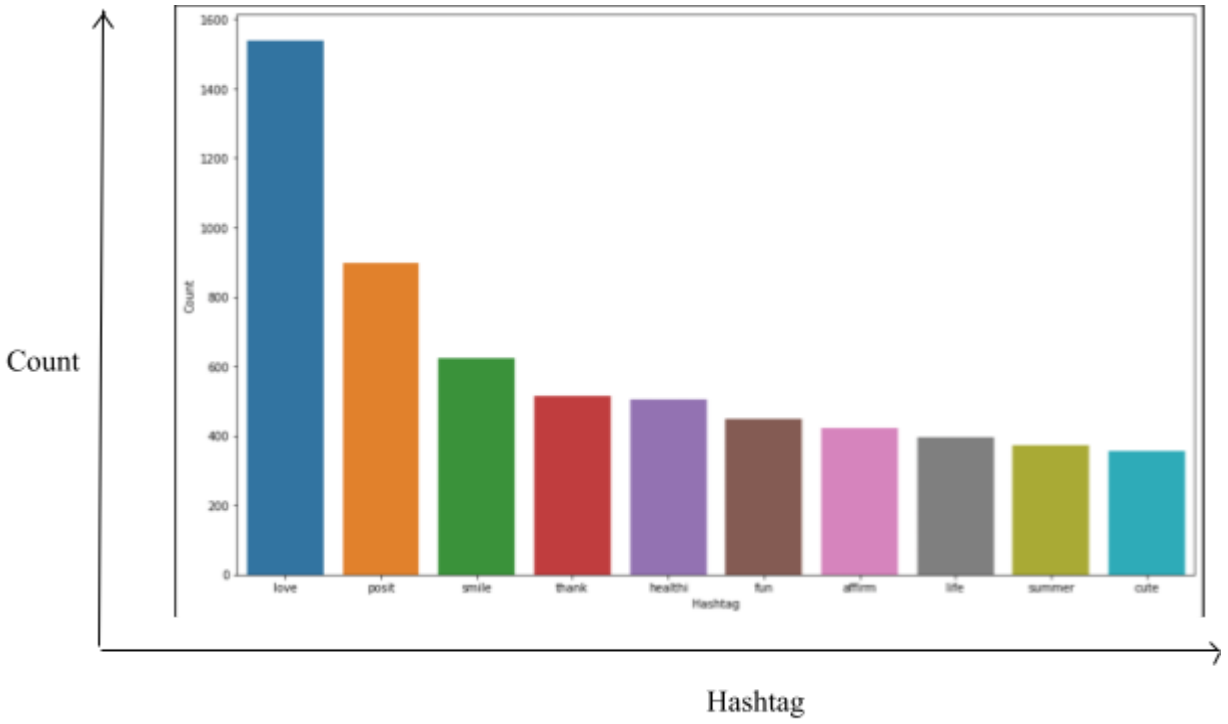


Fig 5.5 Bar graph for Hashtags

This bar graph represents how frequently a hashtag is used in the dataset chosen by us. The different hashtags have been plotted against the Count, which indicates the frequency of usage of those hashtags in the dataset.

We've also split our Input dataset into Training Data and Test Data as we need the same for processing and applying classifiers later.

Input Split

```
# feature extraction
from sklearn.feature_extraction.text import CountVectorizer
bow_vectorizer = CountVectorizer(max_df=0.90, min_df=2, max_features=1000, stop_words='english')
bow = bow_vectorizer.fit_transform(df['clean_tweet'])
```

Python

```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(bow, df['label'], random_state=42, test_size=0.25)
```

Python

Fig 5.6 Code to divide into train and test split

The test set already contains the actual values. Therefore after prediction, the predicted values will be measured against the actual values and the accuracy will be calculated.

Data Preprocessing and Vectorization:

Step 1: Checking for missing values.

Step 2: Text Normalization.

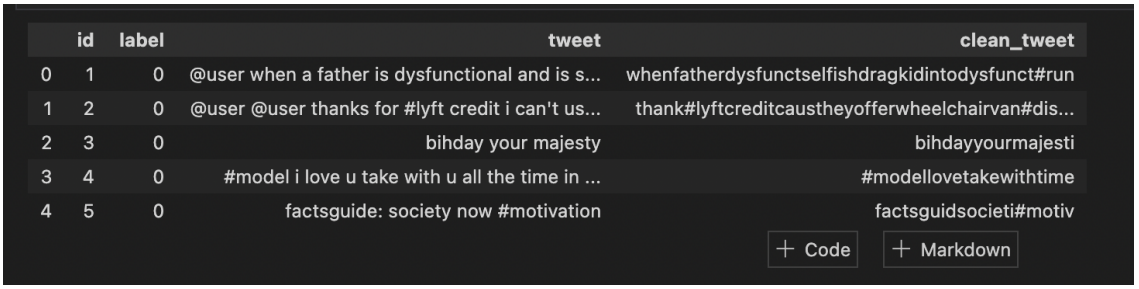
Normalize the text data as texts from such online platforms usually contain inconsistent language and the use of special characters in place of letters. To tackle such inconsistencies in data, Regex.

Step 3: Lemmatization

Lemmatization is the process of grouping together the different forms of a word so they can be analyzed as a single item. For example, we do not want the Machine Learning algorithm to treat eating, eats, and eat as three separate words because they convey the same message. Lemmatization helps reduce the words to their root form.

Step 4: Removal of stop words

Stop words are a set of commonly used words in the language. Examples of stop words in English are "a", "the", "is", "are" and etc. Stop words are commonly used in Text Mining and Natural Language Processing (NLP) to eliminate words that are so commonly used that they carry very little useful information.



	id	label	tweet	clean_tweet
0	1	0	@user when a father is dysfunctional and is s...	whenfatherdysfunctselfishdragkidintodysfunct#run
1	2	0	@user @user thanks for #lyft credit i can't us...	thank#lyftcreditcaustheyofferwheelchairvan#dis...
2	3	0	bihday your majesty	bihdayyourmajesti
3	4	0	#model i love u take with u all the time in ...	#modellovetakewithtime
4	5	0	factsguide: society now #motivation	factsguidsocieti#motiv

Fig 5.7 Pre-processed data

This is the data frame head, after Steps 1-4. This stands common for all classifiers and hence, we do it beforehand. As can be seen, the clean_tweet column contains tweets without stop words, hashtags, and they have also been lemmatized.

Step 5: Converting to Numerical Form

The dataset needs to be converted into the numeric form so that it can be put through classifiers.

TF-IDF means Term Frequency - Inverse Document Frequency. This is a statistic that is based on the frequency of a word but it also provides a numerical representation of how important a word is for statistical analysis.

Term frequency works by looking at the frequency of a particular term you are concerned with relative to the document. This is preferred over CountVectoriser method for vectorisation as TF-IDF has the ability to measure the weight or in layman terms the importance of the word in the document

The document frequency of word i represents the number of documents in the corpus with word i in them. Let us represent document frequency for word i by . With N as the number of documents in the corpus, the tf-idf weight for word i in document j is computed by the following formula:

$$w_{ij} = tf_{ij} \times (1 + \log \frac{1+N}{1+df_{ij}})$$

Fig 5.8 Formula to calculate weight in TF-IDF

Applying Classifiers:

We've used our dataset on four classifiers, namely

Logistic Regression

Logistic regression is a statistical analysis method to predict a binary outcome, such as 0 or 1, based on prior observations of a data set.

A logistic regression model predicts a dependent data variable by analyzing the relationship between one or more existing independent variables.

The regression line divides them into two classes, instead of fitting a regression line, we fit an "S" shaped logistic function, which predicts two maximum values (0 or 1). Logistic Regression is a significant machine learning algorithm because it has the ability to provide probabilities and classify new data using continuous and discrete datasets.

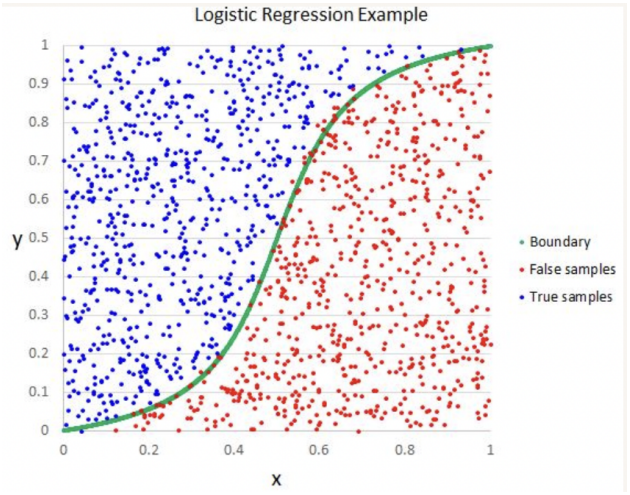


Fig 5.9 Sample graph for Logistic Regression

Arguments to decide threshold:-

1. Low Precision/High Recall: In applications where we want to reduce the number of false negatives without necessarily reducing the number of false positives, we choose a decision value that has a low value of Precision or a high value of Recall.

2. High Precision/Low Recall: In applications where we want to reduce the number false positives without necessarily reducing the number of false negatives, we choose a decision value that has a high value of Precision or a low value of Recall.

Steps to implement Logistic Regression

- Data Pre-processing
- Fitting Logistic Regression to the Training set
- Predicting the test result
- Test accuracy of the result(Creation of Confusion matrix)
- Visualizing the test set result

Now, we apply Logistic Regression to our dataset:

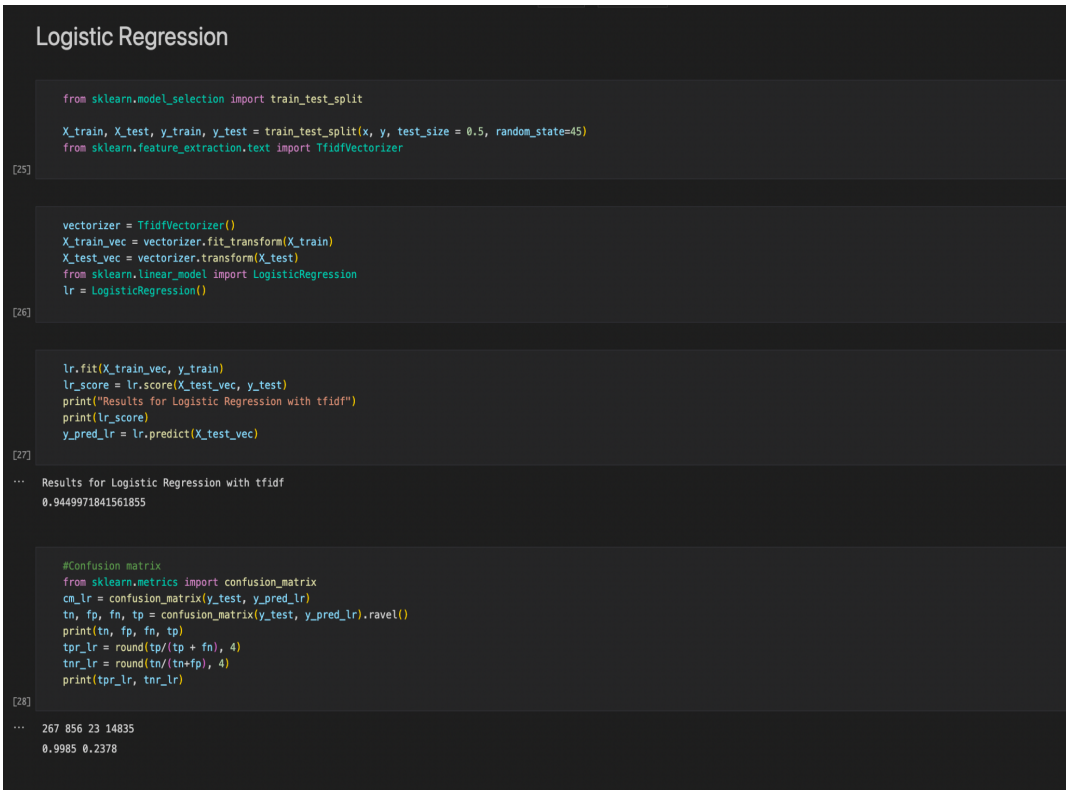


Fig 5.10 code snippet for logistic regression applied on dataset

As can be seen, we have a result of accuracy percentage of ~94.5% with Logistic Regression.

Linear Support Vector Classifier or Support Vector Machine

SVM performs classification by finding the hyper-plane that differentiates the classes we plotted in n-dimensional space.

The objective of the SVM algorithm is to find the hyperplane in an N-dimensional space that distinctly classifies the data points. The dimension of the hyperplane depends upon the number of features.

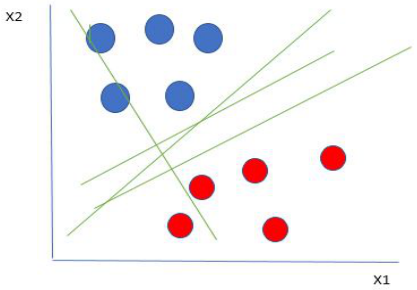


Fig 5.11 Linearly separable data points

From the above figure we can see that there are multiple lines that segregate our data points into red and blue circles.

Selecting the best hyperplane:

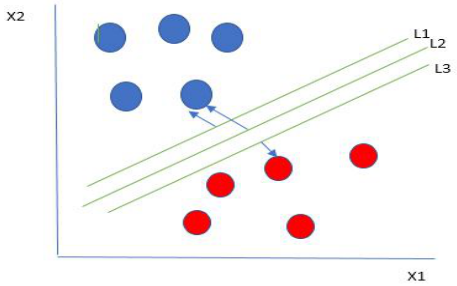


Fig 5.12 Linearly separable data points

The best hyperplane is the one that represents the largest separation between the two classes.From the above figure, we choose L2.The extreme cases (L1 and L2) are called support vectors.

Types of SVM:

1. Linear SVM : When a particular data can be divided easily into two sets or two categories using a straight line then it is known as linear SVM.
2. Non LInear SVM: Non-Linear SVM is used for non-linearly separated data, which means if a dataset cannot be classified by using a straight line, then such data is termed as non-linear data and classifier used is called as Non-linear SVM classifier.

Largest margin is found in order to avoid overfitting. The optimal hyperplane is at the maximum distance from the positive and negative example

Now, we apply SVM to our data

Using SVM

```
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer,CountVectorizer
from sklearn.metrics import confusion_matrix
```

```
x=df['clean_tweet']
y=df['label']
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(x,y,test_size = 0.5, random_state=55)
vectorizer = TfidfVectorizer()
X_train_vec = vectorizer.fit_transform(X_train)
X_test_vec = vectorizer.transform(X_test)
```

```
from sklearn import svm
#params = {'kernel':('linear', 'rbf'), 'C':[1, 10, 100]}
svcl = svm.SVC(kernel = 'rbf')
#clf_sv = GridSearchCV(svcl, params)
svcl.fit(X_train_vec, y_train)
svcl_score = svcl.score(X_test_vec, y_test)
print("Results for Support Vector Machine with tfidf")
print(svcl_score)
```

```
Results for Support Vector Machine with tfidf
0.9537575871347225
```

```
y_pred_sv = svcl.predict(X_test_vec)
#Confusion matrix
from sklearn.metrics import confusion_matrix
cm_sv = confusion_matrix(y_test, y_pred_sv)
tn, fp, fn, tp = confusion_matrix(y_test, y_pred_sv).ravel()
print(tn, fp, fn, tp)
tpr_sv = round(tp/(tp + fn), 4)
tnr_sv = round(tn/(tn+fp), 4)
print(tpr_sv, tnr_sv)
```

```
413 721 18 14829
0.9988 0.3642
```

Fig 5.13 Code snippet for SVM applied on dataset

As can be seen, we get an accuracy of ~95.38% with SVM.

K Nearest Neighbors

KNN algorithm assumes the similarity between the new data and available datasets and put the new data into the category that is most similar to the available categories. It can be used for Regression as well as for Classification but mostly it is used for Classification problems.

It is a non-parametric algorithm, which means it does not make any assumptions on underlying data.

It is also called a lazy learner algorithm because it does not learn from the training set immediately instead it stores the dataset and at the time of classification, it performs an action on the dataset.

Suppose there are two categories, Category A and Category B, and we have a new data point x1, so this data point will lie in which of these categories. To solve this type of problem, we need a K-NN algorithm. With the help of K-NN, we can easily identify the category or class of a particular dataset.

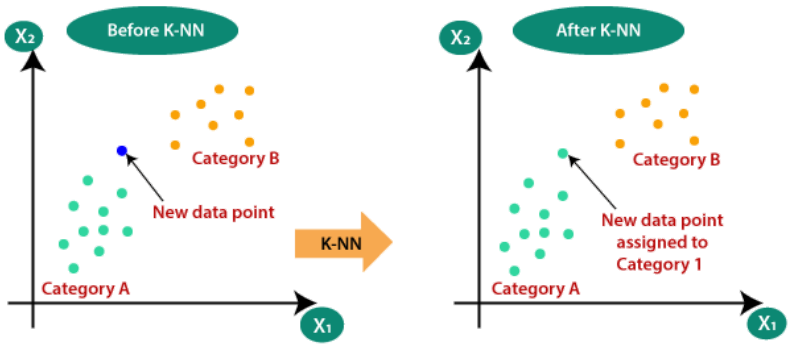


Fig 5.14 Graph depicting the working of KNN

Working of KNN Algorithm

Step-1: Select the number K of the neighbors.

Step-2: Calculate the Euclidean distance of K number of neighbors.

Step-3: Take the K nearest neighbors as per the calculated Euclidean distance.

Step-4: Among these k neighbors, count the number of the data points in each category

Step-5: Assign the new data points to that category for which the number of the neighbours is maximum.

Now, we apply kNN to our dataset.

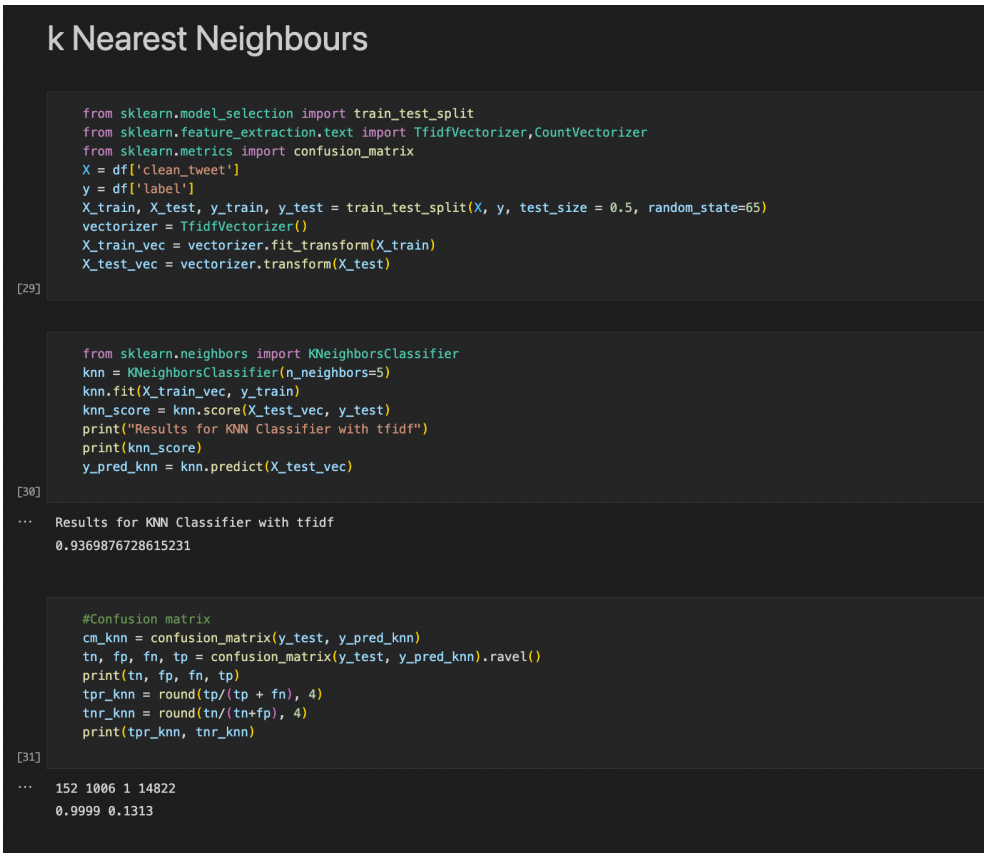


Fig 5.15 Code snippet for KNN applied on dataset

As it can be seen, kNN gives an accuracy of ~93.7%.

Random Forest

Random Forest is a classifier that contains a number of decision trees on various subsets of the given dataset and takes the average to improve the predictive accuracy of that dataset.

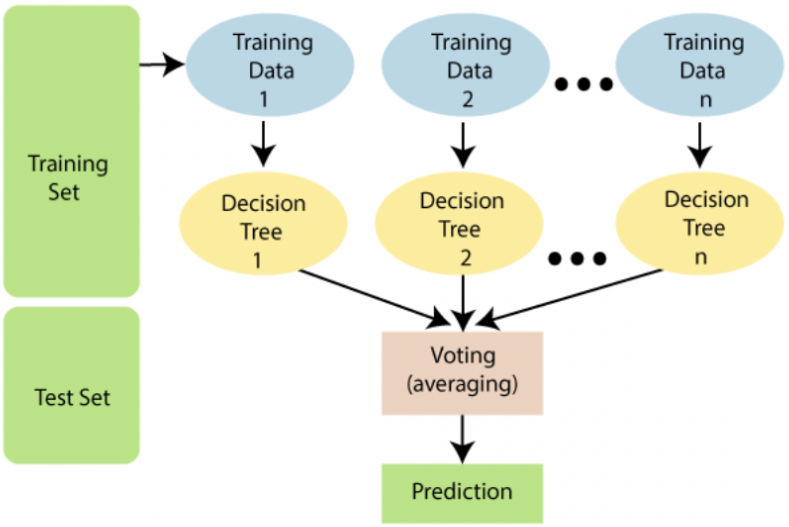


Fig 5.16 Depicting working of Random Forest

Instead of relying on one decision tree, the random forest takes the prediction from each tree, and based on the majority votes of predictions, it predicts the final output. The greater number of trees in the forest leads to higher accuracy and prevents the problem of overfitting.

Random Forest works in two-phase first is to create the random forest by combining N decision tree, and second is to make predictions for each tree created in the first phase.

Step-1: Select random K data points from the training set.

Step-2: Build the decision trees associated with the selected data points (Subsets).

Step-3: Choose the number N for decision trees that you want to build.

Step-4: For new data points, find the predictions of each decision tree, and assign the new data points to the category that wins the majority votes.

Advantages of Random Forest over other classifiers

It takes less training time as compared to other algorithms.
It predicts output with high accuracy, even for the large dataset it runs efficiently.
It can also maintain accuracy when a large proportion of data is missing.
Random Forest is capable of performing both Classification and Regression tasks.
It is capable of handling large datasets with high dimensionality.
It enhances the accuracy of the model and prevents the overfitting issue.

By the fact it prevents overfitting and relies on each tree, and majority of the trees votes, we can assume a high accuracy rate, however, we will check the same with our code on our dataset which can be seen in the figure in the next page.

Now, we apply Random Forest to our dataset.

Random Forest Model

```
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer,CountVectorizer
from sklearn.metrics import confusion_matrix
X = df['clean_tweet']
y = df['label']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.5, random_state=65)
vectorizer = TfidfVectorizer()
X_train_vec = vectorizer.fit_transform(X_train)
X_test_vec = vectorizer.transform(X_test)
```
[32]

```
from sklearn.ensemble import RandomForestClassifier
test_classifier= RandomForestClassifier(n_estimators=200,random_state=0)
test_classifier.fit(X_train_vec,y_train)
predictions = test_classifier.predict(X_test_vec)
```
[33]

```
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score
print(confusion_matrix(y_test, predictions))
print(classification_report(y_test, predictions))
print("Results for Random Forest with tfidf")
rf_score = accuracy_score(y_test, predictions)
print(rf_score)
test_classifier
```
[35]

```
[[  512   646]
 [   57 14766]]
              precision    recall  f1-score   support

           0       0.90      0.44      0.59      1158
           1       0.96      1.00      0.98     14823

    accuracy                           0.96     15981
   macro avg       0.93      0.72      0.78     15981
weighted avg       0.95      0.96      0.95     15981

Results for Random Forest with tfidf
0.9560102621863463

RandomForestClassifier(n_estimators=200, random_state=0)
```

```
#Confusion matrix
text_classifier = confusion_matrix(y_test, predictions)
tn, fp, fn, tp = confusion_matrix(y_test, predictions).ravel()
print(tn, fp, fn, tp)
tpr_knn = round(tp/(tp + fn), 4)
tnr_knn = round(tn/(tn+fp), 4)
print(tpr_knn, tnr_knn)
```
[36]

```
512 646 57 14766
0.9962 0.4421
```

Fig 5.17 Code snippet for Random Forest applied on dataset

As can be seen, we have a result of accuracy percentage of ~95.6%

Summary of the Outputs:

The Model used Classifier	Accuracy Score (in %)
Support Vector Machines	95.38
Logistic Regression	94.50
k Nearest Neighbors	93.69
Random Forest	**95.60**

From the above data comparison, it is clear that Random Forest Model, is the way to go with the highest accuracy of approx. 95.60%

Evaluation of the Models Used:

Since this is classification, there is a lot of confusion, especially when there is a mismatch in prediction, giving rise to True and False predictions.

When there is sample data and we use modeling through which we can predict its class/labels. The “true” represents the records that the model was able to identify its class, whereas “false” represents the records that the model was not able to identify.
There are also “Positive” and “Negative” classes and depending on the objective of the study, we take one as a positive and the other as a negative.
To understand this better, let us look at an example, with the Actual Values and Predicted Values.

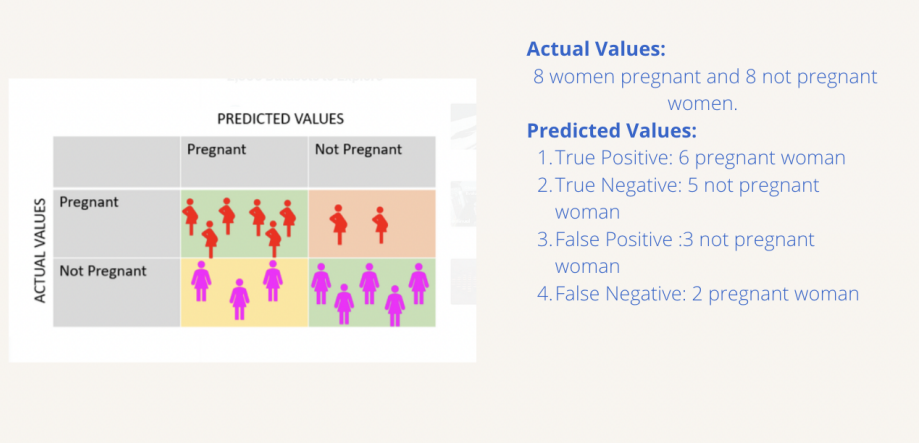


Fig 5.18 Example for confusion matrix

There is also the concept of Confusion Matrix, which helps us understand the problem better.

A confusion matrix is a table that represents the summary of the prediction results on a classification problem.

PREDICTED VALUES

ACTUAL VALUES	Positive	Negative
Positive	True Positive (TP)	False Negative (FN)
Negative	False Positive (FP)	True Negative (TN)

ACTUAL VALUES

PREDICTED VALUES	Positive	Negative
Positive	True Positive (TP)	False Positive (FP)
Negative	False Negative (FN)	True Negative (TN)

Fig 5.19 Confusion matrix

The position of the predicted values and actual values changes the position of False negative (FN) and False positive (FP) but True positive (TP) and True negative (TN) remains in the same place in the matrix placed diagonally to each other. But because of this, the situation becomes confusing.

True Positive(TP): Values that are actually positive and predicted positive.

False Positive(FP): Values that are actually negative but predicted to be positive.

False Negative(FN): Values that are actually positive but predicted to be negative.

True Negative (TN): Values that are actually negative and predicted to be negative.

Rate is a measuring factor in a confusion matrix.

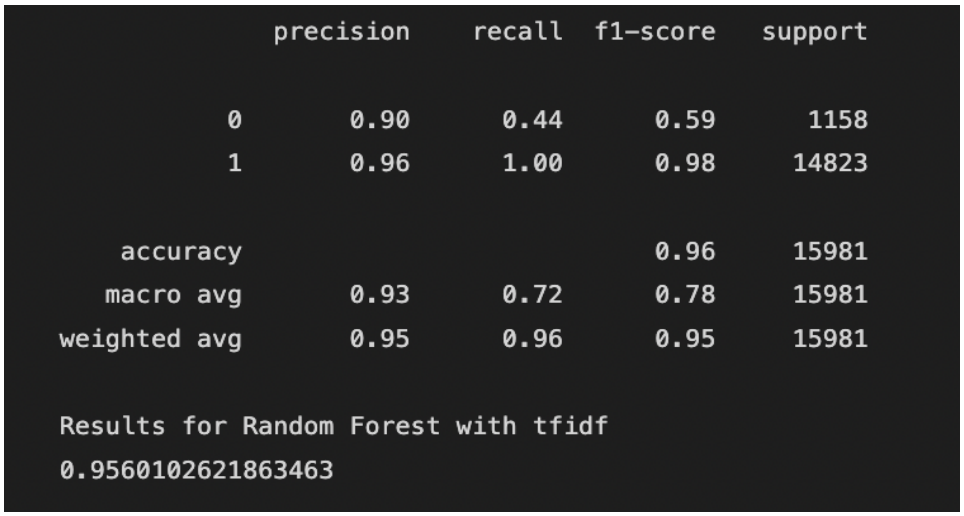
True Positive Rate(TPR): True Positive/(True Positive + False Negative)
True Negative Rate(FPR): True Negative /(True Negative + False Positive)

Let us analyze the results for the same we got from our models.

The Model used as a Classifier	**True Positive Rate**	**True Negative Rate**
Logistic Regression	0.9985	0.2378
Support Vector Machines	0.9988	0.3642
k Nearest Neighbors	0.9999	0.1313
Random Forest	0.9962	0.4421

Finding a balanced output and predicting True Positives and Negatives is important, and hence, out of the models we've shown here, Random Forest is the one we choose as the best one.

The classification report for Random Forest is



	precision	recall	f1-score	support
0	0.90	0.44	0.59	1158
1	0.96	1.00	0.98	14823
accuracy			0.96	15981
macro avg	0.93	0.72	0.78	15981
weighted avg	0.95	0.96	0.95	15981

Results for Random Forest with tfidf
0.9560102621863463

Fig 5.20 Classification Report for Random Forest

The next step was to deploy a backend that does the same analysis and prediction LIVE when a user enters a keyword about tweets.

LIVE Sentiment Analyser

The LIVE Sentiment Analyser was built using a Python framework Flask for the backend, HTML and CSS for frontend, and Google Cloud NLP. Python Modules like Tweepy, Google Cloud Languages etc helped implement the helper scripts for the backend. It has a dialog box that has the functionality to accept keywords. The drop down button next to it allows to us pull the real time tweets using the said keyword.

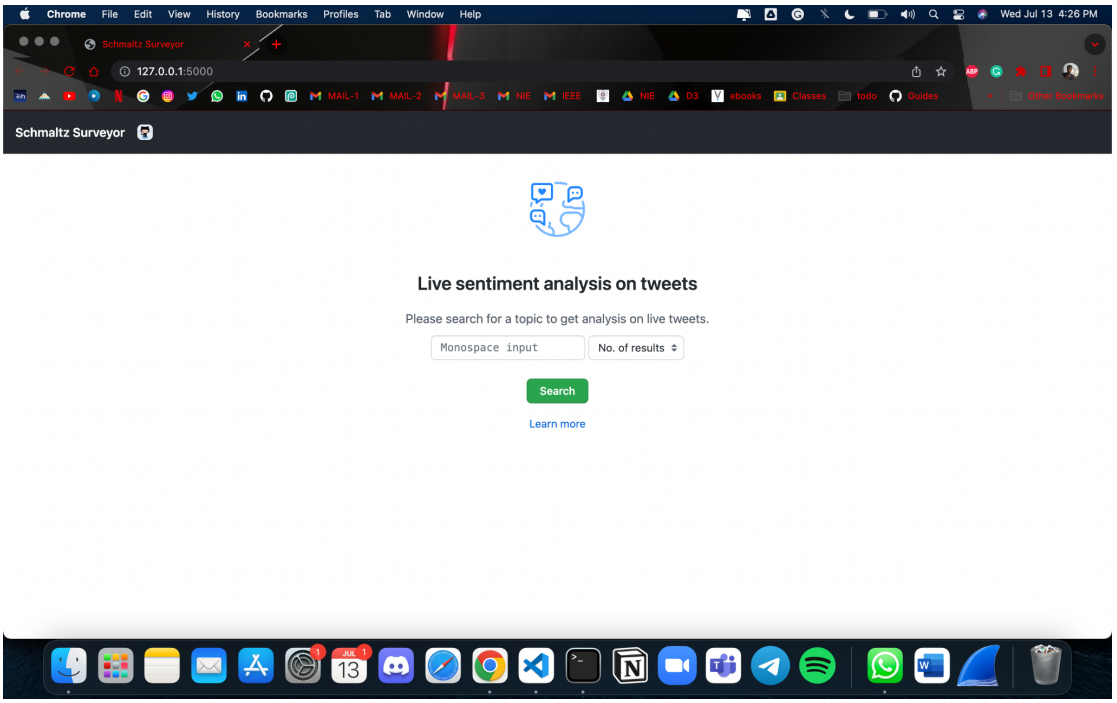


Fig 5.21 Main Home Page

The user lands on this page. Here the user can type the keyword and select the number of results i.e tweets from the drop-down. The drop-down has the values 10,100 and 1000. After clicking on the search button, the user is redirected to another page that will display the sentiments for the tweets acquired.

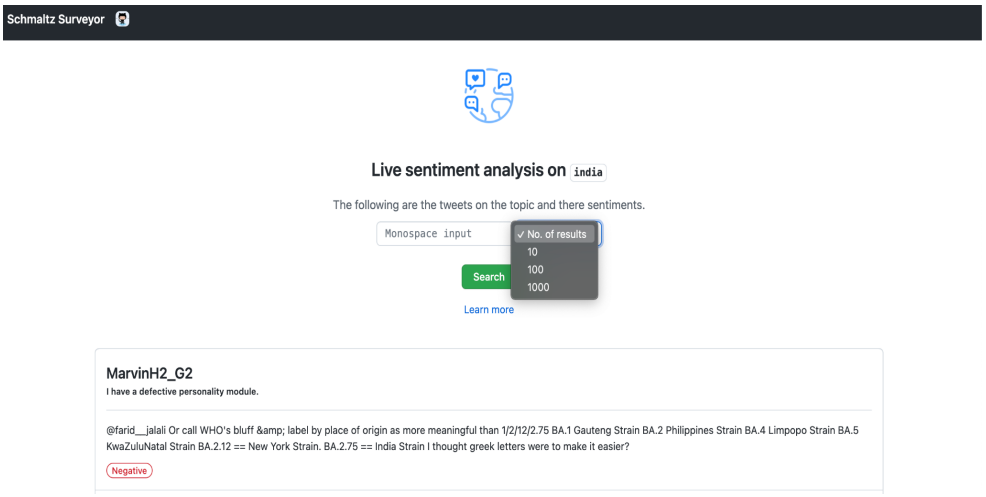


Fig 5.22 Main Home Page

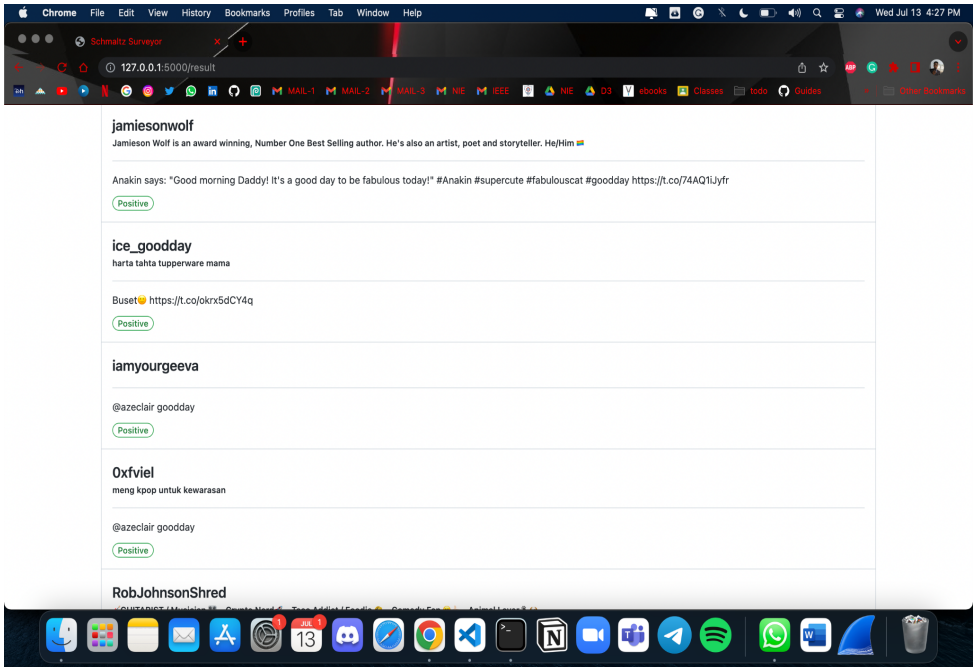


Fig 5.23 Page displaying sentiments

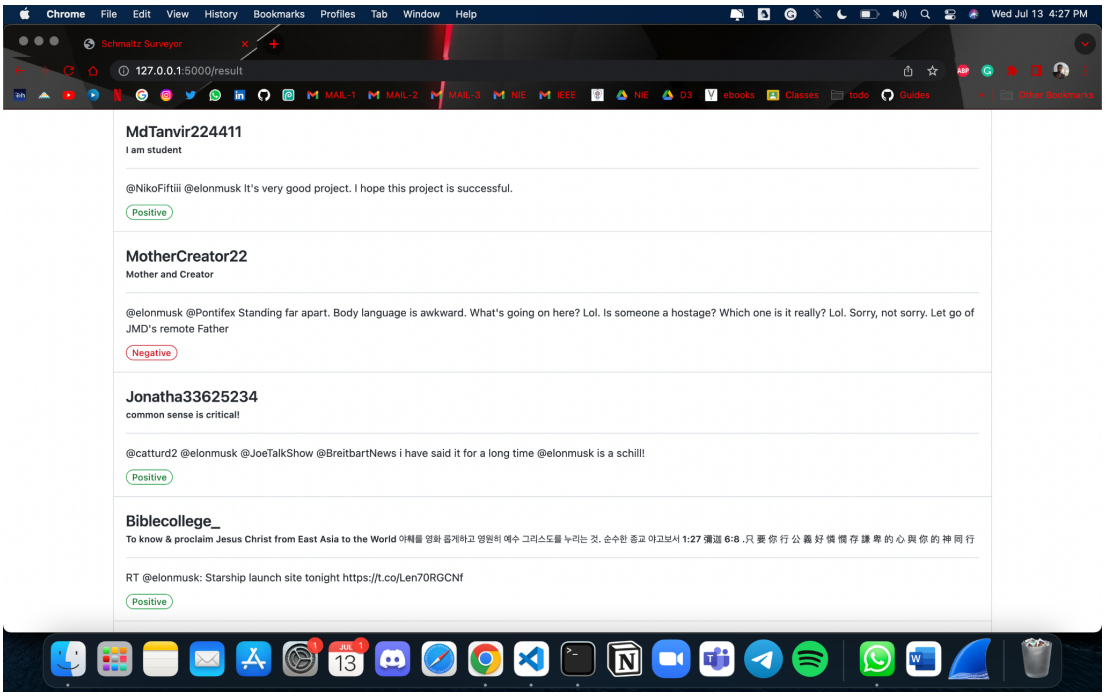


Fig 5.24 Page displaying sentiments

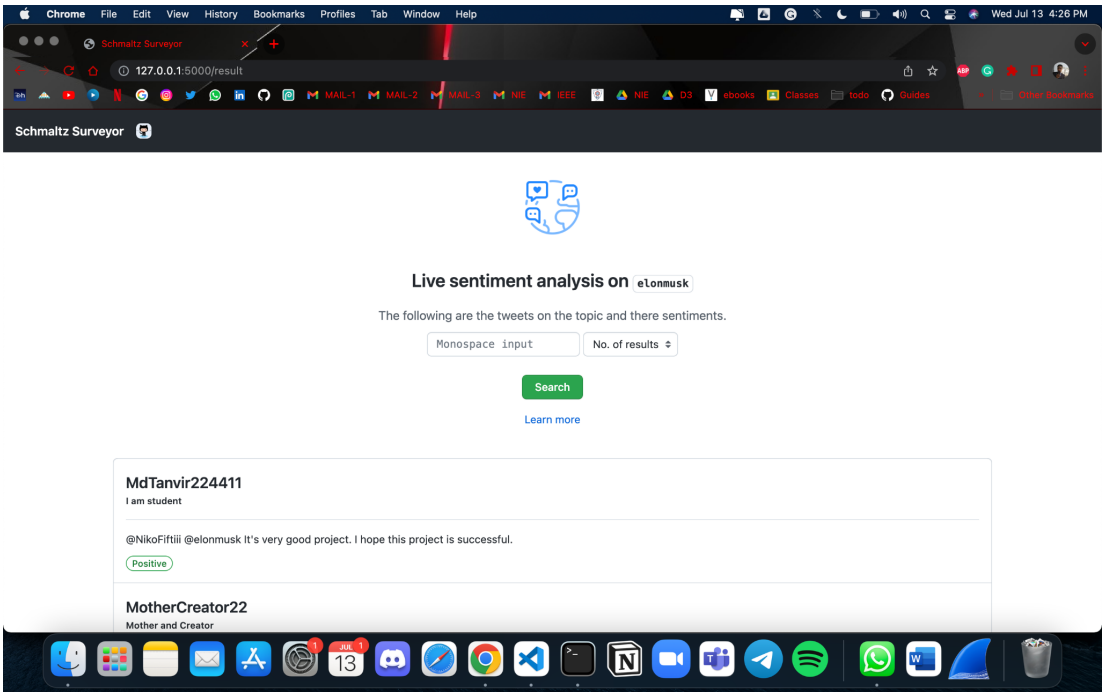


Fig 5.25 Page displaying sentiment

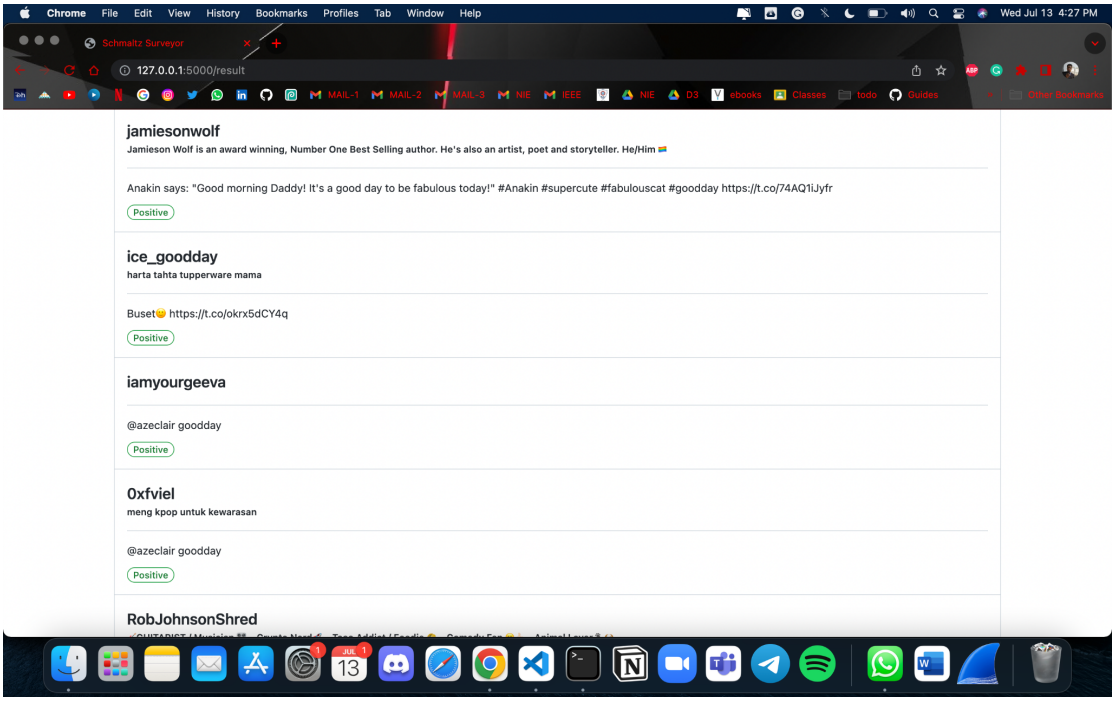


Fig 5.26 Page displaying sentiments

In the above figures, the second landing page displays all the real-time tweets containing the keyword. Here the tweet is displayed along with the positive or it negative sentiment that it is displaying. This is done using the helper scripts at the backend which provide us the sentiment score and based on the same, we output the sentiment as either positive or negative in the web application like we have shown here.

CHAPTER 6:

TESTING

The main aspect of the testing involved for this project was to test it with different Vectorizers and Classifiers.

We had the option to choose two vectorizers, Count Vectorizer and TF-IDF Vectorizer, and based on the use case, both were tested and TF-IDF was chosen.

TF-IDF is better than Count Vectorizers because it not only focuses on the frequency of words present in the corpus but also provides the importance of the words. We can then remove the words that are less important for analysis, hence making the model building less complex by reducing the input dimensions.

The next was to test the data for different classifiers and the same has been illustrated in-depth in the System Implementation part.

The data were tested with 4 classifiers, namely

1. Random Forest
2. Support Vector Machines
3. k Nearest Neighbours
4. Logistic Regression

Finally, based on the same, we chose the Random Forest which gave the output we needed with the highest accuracy.

The table below shows the accuracy levels and the True positive rate and True Negative Rate we got from different classifiers.

The Model used Classifier	**Accuracy Score (in %)**
Support Vector Machines	95.38
Logistic Regression	94.50
k Nearest Neighbors	93.69
Random Forest	**95.60**

The Model used as a Classifier	True Positive Rate	True Negative Rate
Logistic Regression	0.9985	0.2378
Support Vector Machines	0.9988	0.3642
k Nearest Neighbors	0.9999	0.1313
Random Forest	0.9962	0.4421

Random Forest Classifier was then observed to be the more accurate one, as it clearly has the better Accuracy Score and a better balanced True Positive and True Negative Rates.

The second part of the Testing done was based on the number of inputs in the Web Application.

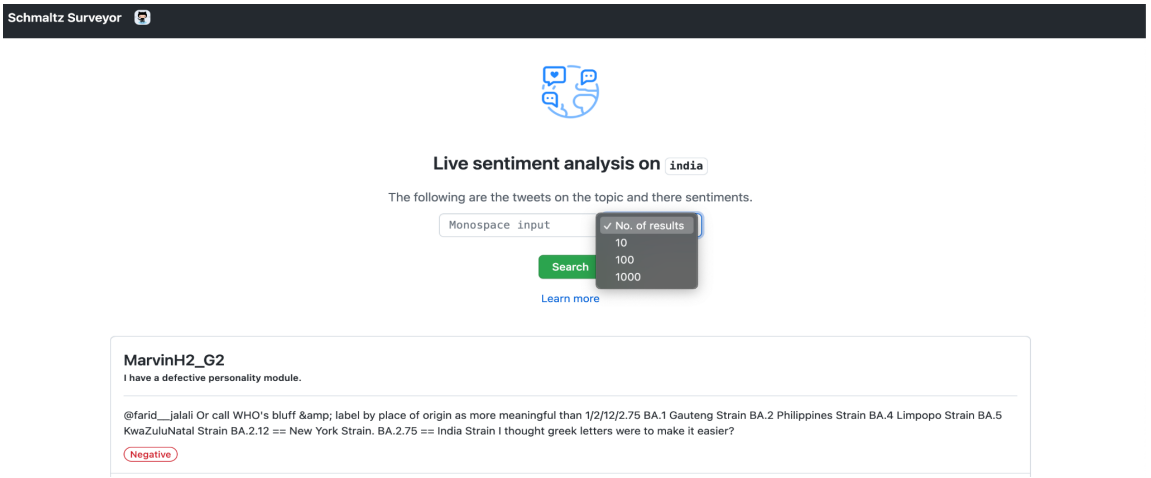


Fig 6.1 Testing with the different number of inputs

The webpage has the option to analyze sentiments for the number of tweets we need. Based on our needs, we can choose the number of tweets to analyse and those number of tweets will be fetched from Twitter and the webpage will indicate the sentiment expressed in each tweet.

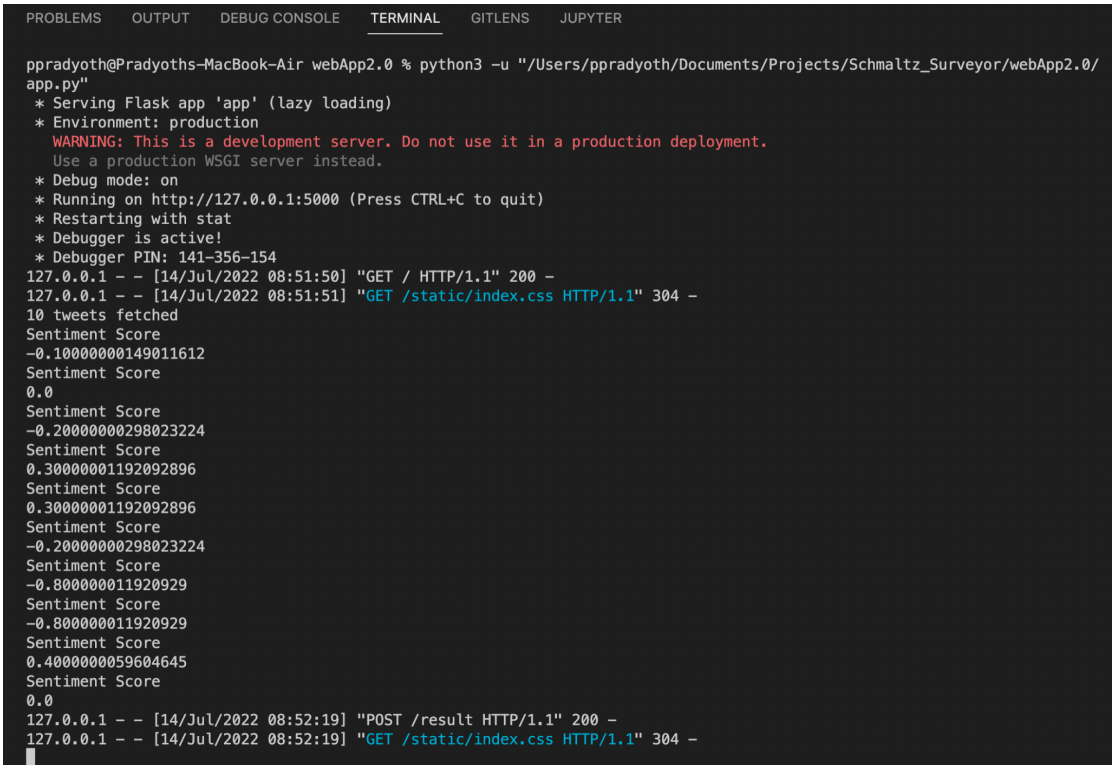


Fig 6.2 Sentiment Scores at the backend

The backend terminal, as shown here, retrieves the tweets and shows the sentiment score which is either positive or negative, indicating how much positive sentiment was expressed and how much negative sentiment was expressed in the given tweet. Thus, we know that the depth of sentiment expressed, and if the frontend showed the same accurately.

CHAPTER 7:

CONCLUSION

In conclusion, as said in the introduction and the abstract, we've clearly demonstrated the Sentiment Analysis being done LIVE, along with the Comparison between various ML Models, and choosing the right one for our test case.

We have used 4 classifiers and they gave the following accuracy :

1. Logistic Regression : 95.38%
2. Support Vector Machine : 94.50%
3. K Nearest Neighbour : 93.69%
4. Random Forest : 95.60%

From the above data comparison of the classifiers used, it is clear that Random Forest Model is the way to go with the highest accuracy of approx. 95.60%

Classifying the tweet into positive and negative would help us understand and analyze the extent of positivity and negativity on Twitter. Sentiment Analysis is a great way to analyze the response to a particular tweet. The model at a backend to a web application helps us analyze the sentiments, as and when needed, and helps us with a dashboard of the same as well.

CHAPTER 8:

FUTURE ENHANCEMENTS

Firstly, English is definitely one of the most largely used languages for communication on social media platforms. But there is a lot of content that is produced in regional languages as well.

Detecting sentiment is definitely hard for regional languages as most of the models built are for English. But this is not an impossible task. This software can be used as a starting point for creating models to detect sentiment in regional languages.

Secondly, such software can be used in a widespread manner and this functionality can be added by users who want to avoid unnecessary negative comments on their page and want to filter out negative content.

CHAPTER 9:

REFERENCES

[1]. Fang, Xing, and Justin Zhan. "Sentiment analysis using product review data." Journal of Big Data 2.1.

[2] Faizan. "Twitter Sentiment Analysis" International Journal of Innovative Science and Research Technology (2019)

[3] Chirag Kariya and Priti Khodke. "Twitter Sentiment Analysis" 2020 International Conference for Emerging Technology (INCET) Belgaum.

[4]Amolik, Akshay, et al. "Twitter sentiment analysis of movie reviews using machine learning techniques." International Journal of Engineering and Technology 7.6.

[5] Scikit Learn user guide https://scikit-learn.org/stable/user_guide.html

[6] Numpy Documentation https://numpy.org/doc/

[7] Pandas Documentation https://pandas.pydata.org/docs/

[8] Dataset
https://www.kaggle.com/datasets/dv1453/twitter-sentiment-analysis-analytics-vidya

[9] Flask Documentation https://flask.palletsprojects.com/en/2.1.x/

[10] NLTK Documentation https://www.nltk.org/

[11] Google Cloud NLP Documentation
https://cloud.google.com/natural-language/docs/reference/rest

[12] Tweepy Documentation https://docs.tweepy.org/en/stable/

[13]Jalilifard, A., Caridá, V.F., Mansano, A.F., Cristo, R.S., da Fonseca, F.P.C. (2021). Semantic Sensitive TF-IDF to Determine Word Relevance in Documents. In: Thampi, S.M., Gelenbe, E., Atiquzzaman, M., Chaudhary, V., Li, KC. (eds) Advances in Computing and Network Communications. Lecture Notes in Electrical Engineering, vol 736. Springer, Singapore.
https://doi.org/10.1007/978-981-33-6987-0_27