SRI RAMACHANDRA
INSTITUTE OF HIGHER EDUCATION AND RESEARCH
(Category - I Deemed to be University) Porur, Chennai
SRI RAMACHANDRA FACULTY OF ENGINEERING AND TECHNOLOGY

STREAMING HIGHWAY TRAFFIC ALERTS USING TWITTER API

INTERNSHIP REPORT

Quarter IV (Year 1)

Submitted by

AYUSHMAAN DAS E0121037

In partial fulfilment for the award of the degree of

BACHELOR OF TECHNOLOGY

In

COMPUTER SCIENCE AND ENGINEERING

(Artificial Intelligence and Machine Learning)

Sri Ramachandra Faculty of Engineering and Technology

Sri Ramachandra Institute of Higher Education and Research, Porur, Chennai - 600116

JULY, 2022

SRI RAMACHANDRA
INSTITUTE OF HIGHER EDUCATION AND RESEARCH
(Category - I Deemed to be University) Porur, Chennai
SRI RAMACHANDRA FACULTY OF ENGINEERING AND TECHNOLOGY

BONAFIDE CERTIFICATE

Certified that this project report **"Streaming Highway Traffic Alerts using Twitter API"** is the bonafide work of **Ayushmaan Das** Reg No. **E0121037** who carried out the internship work under my supervision.

Signature of Project Mentors

Dr. Jayanthi G

Assistant Professor,

&

Prof. Ramya M

Lecturer,

Sri Ramachandra Faculty of Engineering and Technology

Porur, Chennai-600116

Signature of Vice-Principal

Prof. Prema

Vice-Principal

Sri Ramachandra Faculty of Engineering and Technology

Porur, Chennai-600116

EVALUATION DATE:

SRI RAMACHANDRA
INSTITUTE OF HIGHER EDUCATION AND RESEARCH
(Category - I Deemed to be University) Porur, Chennai
SRI RAMACHANDRA FACULTY OF ENGINEERING AND TECHNOLOGY

ACKNOWLEDGEMENT

I would like to express my sincere gratitude to our Chancellor, Vice-Chancellor, our Provost **Dr. V Raju** and our Vice-Principal **Prof. Prema** for providing me the opportunity to work for this internship and for providing all the facilities required for the successful completion of the project.

I sincerely want to thank my faculty mentors, **Dr. Jayanthi G** and **Prof. Ramya M** (Department of Computer Science, SRET, SRIHER – Chennai) for allowing me to work under them and for extending their help and cooperation throughout the work period. It was impossible to achieve success without their guidance and proper counsel. The resources provided by them helped me to expand my views and enhance my knowledge to a great extent.

I would also like to thank the entire Department of Computer Science of Sri Ramachandra Engineering and Technology, the staffs and all other faculty members who were always available for extending help and support. All of them contributed to overcoming the obstacles in the study.

Lastly, I would like to express my gratitude to my friends for their suggestions and cooperation. Also, I thank my family members for their moral support and care.

TABLE OF CONTENTS

LIST OF TABLES AND FIGURES

Tables

Figures

1. INTRODUCTION

1.1 Twitter and Twitter API

Twitter is an American company, which was created by Jack Dorsey, Noah Glass, Biz Stone and Evan Williams in March 2006. Twitter widely serves as a platform for users to share their thoughts and ideas publicly and privately. Twitter is a social networking service which enables the users to interact with messages in the form of tweets.

However, all these messages can widely serve as a big storehouse of information. Such type of data has been termed as *Big Data*. The tweets can be used to access information and day-to-day happenings about a particular place, individual, etc.

In this project, the tweets have been fetched and a model was built to only obtain the tweets related to “traffic” in a city. On reading the tweets, the users will be able to understand about the conditions of roads around them and accordingly, avoid taking the congested or blocked routes.

API stands for *Application Programming Interfaces*. Twitter API allows a programmer to access twitter in a unique way. One can use the functionalities without actually opening the application. This is widely used for Information Systems, Machine-Learning purposes, etc.

1.2 Pre-Processing

This involves removal of unnecessary characters, symbols or block of texts from a file. Pre-Processing of tweets are very essential as it makes the further steps in classification and model building more efficient and accurate. Tweets usually contain emojis, hashtags, mentions, etc. which are not of any use. Thus, they are needed to be discarded.

1.3 Model Building

A model is created using *Python* as the programming language. There are many libraries, primarily *Scikit-Learn* which provide us with various libraries and tools for building a model and training it with some pre-classified dataset.

After appropriate training, prediction is carried out for a particular tweet and they are classified as either ‘traffic’ or ‘non-traffic’. Various models are used for the purpose in order to cross-check their accuracy and see which one is giving greater accuracy.

The *ROC Curves* for various models are plotted and interpreted. Greater the area of the curve, greater is the accuracy.

1.4 Web Application

A user-friendly web application has been developed which enables users to stream and filter traffic-related tweets of the city of their choice. This has been achieved using *StreamLit*.

StreamLit is an open-source framework which has been developed in Python Language. It is widely used to develop web apps for projects related to data science or machine learning.

2. OBJECTIVES

- Collecting tweets from Twitter API
- Pre-Processing of tweets followed by Lemmatization
- Vectorization and model building
- Design of data pipeline for preparation of tweets and machine learning workflow
- Deployment of the framework into a web-app

The primary objective of this project is to build an application which would accept the city name from the user, then generate all the traffic-related tweets in the city within a period of seven days from the date of search. This would then alert the user to avoid those particular routes where there is a blockage, accident or any other obstacle.

The tweets are fetched from twitter API using *TweePy*. These tweets are later need to be classified as ‘traffic’ or ‘non-traffic’ based on the model.

The tweets also need to be pre-processed in order to improve the efficiency of the model. The tweets are also *Lemmatized* in order to improve the scope of search and accuracy.

The most essential part is the building of model which is preceded by *Vectorization*. This involves representation of tweets as an array of numbers for the machine to understand. The model is trained with a pre-classified dataset, and then is used to classify the tweets which were fetched earlier. The aim is to obtain maximum accuracy and hence, multiple methods are used.

Ultimately, the development of web-app comes into action. *StreamLit* serves as one of the best open-source frameworks for the purpose. The aim was to make the app interactive as well versatile, allowing the users to execute a search based on their choice.

3. PROBLEM STATEMENT

Description

"Design of a Highway Traffic Alert System by mining tweets from twitter API"

The project has been divided into five different modules for easier understanding and implementation:

Table 1. Tools and Technologies

Modules	**Technologies Used**
Module 1: Tweet Extraction	TweePy, Pandas
Module 2: Pre-Processing of Tweets	Regular Expressions (Re), NLTK
Module 3: Vectorization, Model Building and Classification	Scikit-Learn
Module 4: Data Visualisation (ROC Plots) and Accuracy testing	Matplotlib, Seaborn, Scikit-Learn
Module 5: Deployment	StreamLit

4. LITERATURE SURVEY

Table 2. Survey of existing literature

YEAR	AUTHOR	PRODUCT	REVIEW
2016	Faculty CSE, SG Polytechnic, Atigre, IN	Ongoing Detection of Traffic from Twitter Stream Analysis	**Benefits:** Classification model used works with great accuracy
			Application: Proper classification of traffic related tweets
			Limitation: No data visualization and no frontend
			Challenges: Developing a proper UI for user interaction
2017	Pouza Rezai	Detection classification and location identification of traffic congestion from Twitter stream analysis	**Benefits:** Proper data visualization and accuracy
			Application: Proper data visualization and accuracy
			Limitation: Lack of UI implementation for user to interact
			Challenges: Developing a user-interface and improve the model
2018	Sivagurunathan V	Traffic Detection from Twitter using Spark	**Benefits:** Use of Kafka and ElastiSearch
			Application: Obtaining a Traffic HeatMap
			Limitation: No data visualization, tweets are old
			Challenges: Using data visualisation and recent Tweets
2015	IEEE Members	Real-Time Detection of Traffic from Twitter Stream Analysis	**Benefits:** A complete package with user-alerts system
			Application: E-mailing the users about the traffic alerts
			Limitation: Requires continuous internet connectivity
			Challenges: Decreasing the number of false-alerts

5. METHODOLOGY

5.1 Pipeline (Flowchart) for mining tweets

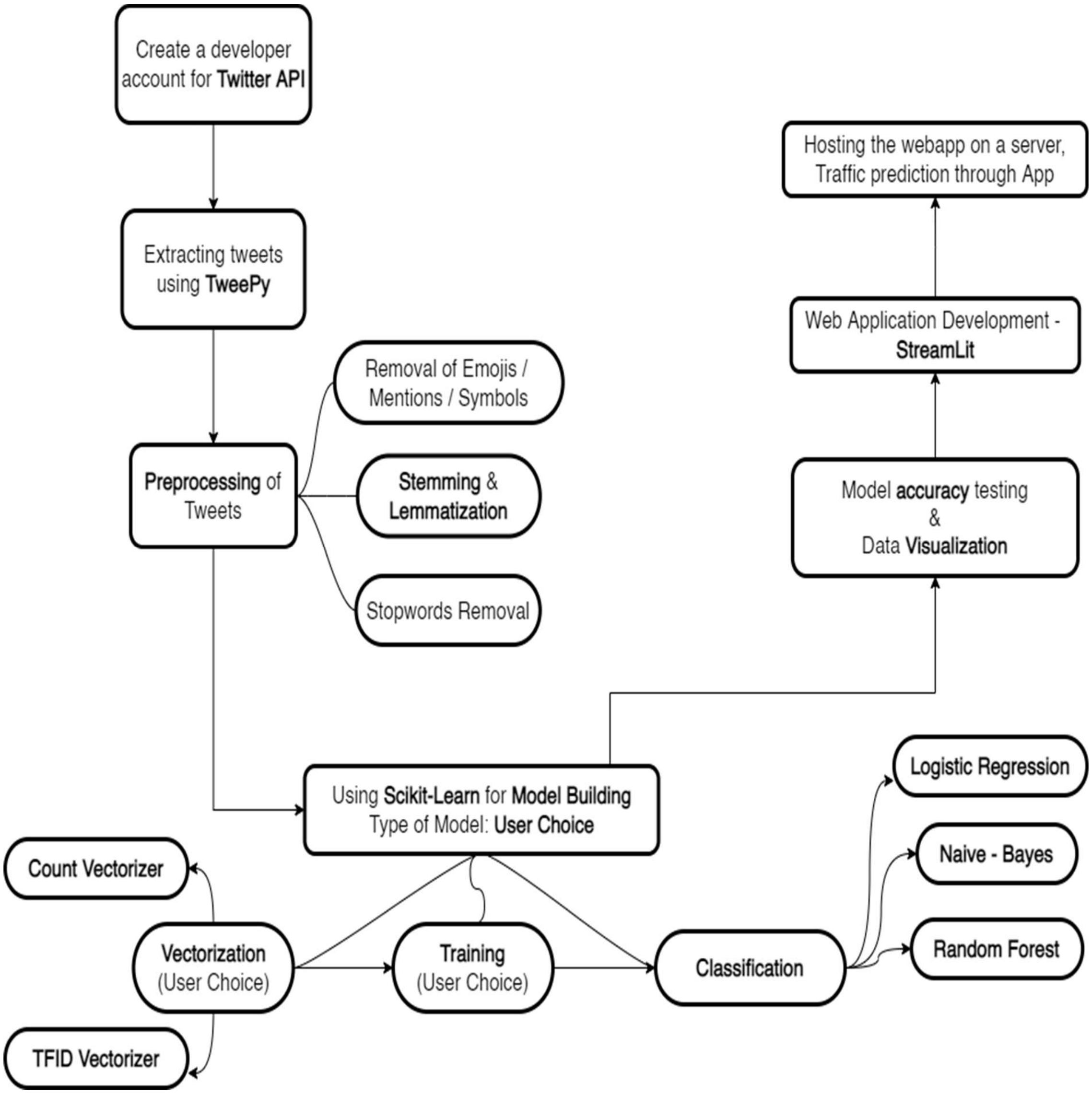

Create a developer account for Twitter API
Extracting tweets using TweePy
Preprocessing of Tweets
Removal of Emojis / Mentions / Symbols
Stemming & Lemmatization
Stopwords Removal
Using Scikit-Learn for Model Building Type of Model: User Choice
Count Vectorizer
Vectorization (User Choice)
TFID Vectorizer
Training (User Choice)
Classification
Logistic Regression
Naive - Bayes
Random Forest
Model accuracy testing & Data Visualization
Web Application Development - StreamLit
Hosting the webapp on a server, Traffic prediction through App

Figure 1. Data Pipeline: Pre-processing tweets and Machine learning workflow for classification

The sequential work methodology of the project involves fetching of tweets followed by pre-processing, model building, training – classification and data visualisation. Ultimately, the concepts are integrated and deployed in the form of a web-application using Streamlit.

5.2 Developers Account in Twitter

A developers account must be created in order to access the Twitter API. On successful creation of the account, an app must be created within a project. The app must be given an authentication of version *OAuth1.0* and this will enable us to stream tweets from twitter API. The elevated level of access can be applied for which is granted after a short period of time. The following features of the app must be stored safely for future references:

- API Key
- API Key Secret
- Bearer Token
- Access Token
- Access Token Secret

5.3 Fetching Tweets from Twitter API

Twitter API is accessed using the aforementioned *Keys* and *Tokens*. After successful authentication, tweets are fetched using *.search_tweets()* method of *TweePy*. The queries like languages, removing retweets and city name, etc. are created and passed as a parameter to the method. Accordingly, tweets are fetched which can be stored in a CSV or Excel file for future reference.

Table 3. Fetching tweets for testing

Current Date	From Date	To Date	No. of tweets collected
10/07/2022	1/07/2022	9/07/2022	984
22/02/2022	11/07/2022	22/07/2022	236
28/07/2022	23/07/2022	27/07/2022	1153

5.4 Text Pre-Processing and Lemmatization

The tweets contained unwanted symbols, emojis, characters or words which should be removed for efficient model building. This is known as pre-processing of tweets. This is followed my *Lemmatization*. Stemming and Lemmatization involves grouping together the inflected form of a particular word, so that they can be analysed as single item. It is required for increasing the efficiency of searching of tweets and vectorization.

5.5 Vectorization

Vectorization involves representation of a piece of text as a combination of numbers for the machine to understand. In Machine Learning, vectorization is a stage in highlight extraction. The thought is to get a few distinct elements out of the text for the model to prepare on, by changing text over completely to mathematical vectors.

Two types of vectorizers have been used for the purpose:

- *Count Vectorizer*: Works on the Bag of Words model
- *TFIDF Vectorizer*: Better than count vectorizer as it considers the overall document weightage.

5.6 Model Building, Training and Classification

Scikit-Learn(sklearn) is one of the most widely used Python library that provides all the basic tools required for data analysis. After vectorization, the model is trained using a pre-classified dataset. The following approaches have been used:

- *Logistic Regression*
- *Naïve – Bayes Model*
- *Random – Forrest Classifier*

After training, the model is now used to classify the earlier fetched tweets either as ‘traffic’ or ‘non-traffic’. Various models have various accuracy scores. The *ROC Curves* for various models are visualised using *Matplotlib* and *Seaborn*. The curve which has greater area will have greater accuracy.

5.7 Creation of a Web-Application

The web-application has been developed using *StreamLit*, one of the most commonly used packages for building applications related to machine learning purposes. The application has multiple pages, namely:

- Home: default page that opens on running the application
- Custom Search: user can search for traffic related tweets from a particular city. Also, the model, approach, test-split size can be chosen by the user for complete versatility.
- Quick Search: for searching traffic-tweets in some common cities on the click of a button
- About: information about the app

5.8 Deployment of the App on a server

StreamLit provides a free cloud service that enables us to deploy the develop app on its server using *GitHub*. The service has been termed as *StreamLit Cloud*. On its deployment, the app can be used from anywhere through the link generated after deployment.

6. TECHNOLOGIES USED

6.1 Python

Python is a high-level and one of the most popular programming languages, first designed by Guido van Rossum which was first released in 1991. It has a widescale application and can be used for various tasks like data analysis, machine learning, building websites, automate tasks, etc.

Python is considered as one of the best languages for Machine Learning purposes because of the following:

- simple and consistent
- flexible as well as platform independent
- has a number of Libraries for machine learning purposes
- wide community of developers

6.2 TweePy

TweePy is a python package which is open-source and is used by the programmers to access the Twitter API. It comprises of various types of classes and methods that represents Twitter's Models and API endpoints. It can give us almost all kinds of functionalities provided by Twitter API.

Before fetching tweets, one must use '*ConfigParser'* to read the keys and tokens of our Twitter API app stored in some file, generally 'config.ini'.

6.3 Pandas

Pandas is an open-source python package. It is built on top of *NumPy*, which is another python package. It is widely used for Machine Learning tasks. It serves the purpose of Data Cleansing, Normalisation, Merges, Joins, Reading and Writing Data, etc. Various types of files can be read in the form of dataframes which make it easy for data manipulation and analysis.

In this project, pandas have been used to read tweets from stored CSVs which are later used for displaying information and data visualisation. Some pandas functionalities also support limited data visualisation.

6.4 Scikit-Learn

Scikit-Learn is a library in Python used for machine learning and data analysis purposes. It provides all sorts of tools required for model building that include classification, regression, clustering, score matrices, vectorization, etc. It has been built on top of *NumPy*, *SciPy* and *Matplotlib*. It is open-source and accessible to everybody.

In this project *Logistic Regression*, *Random Forest Classifier* and *Naïve-Bayes Model* algorithms have been used from the library. Also for vectorization, scikit-learn's *Count Vectorizer* and *TFIDF Vectorizer* have been implemented.

6.5 Natural Language Toolkit (NLTK)

NLTK is a python toolkit used for Natural Language Processing (NLP) in python. NLTK supports various functionalities like parsing, tokenization, stemming, lemmatization, etc. Lemmatization involves conversion of the word into its base form. The extracted word is termed as *Lemma*. The lemmas of words can be obtained using NLTK's *WordNetLemmatizer.*

6.6 Matplotlib and Pyplot

Matplotlib is used for plotting graphs and curves in python. It is used in combination with *NumPy* and *Pandas* for data visualisation in the form of graphs, plots and charts.

Pyplot is a module within Matplotlib that provides *MATLAB-like* interface. Pyplot has various types of graphs like bar, pie, histograms, etc. Values can be obtained from any sort of datasets and graphs can accordingly be plotted.

6.7 Seaborn

Seaborn can be used as an alternative to matplotlib. It uses Matplotlib underneath for data visualisation. It provides some additional plots and also, provides a prettier output than the basic plots. Plots like Violin Plots, Swarm Plots, etc. can be used through seaborn.

6.8 StreamLit

StreamLit is a python tool which is generally used for the development of web applications specifically for Machine Learning and Data Visualisation purposes. The app can be written in the same way and syntax in which we write a python code. StreamLit provides a wide range of features in the form of *widgets*. There are various types of inputs, graphs and even markdowns that allow us to write a chunk of code in the form of HTML. Thus, it is versatile and supports a wide range of Python libraries including scikit-learn.

StreamLit Cloud is an app hosting cloud-based service provided by streamlit for free. The app needs to be upload into a GitHub repository and then, needs to be deployed into the cloud. Once deployed, it can be accessed through the link provided by the cloud.

7. PRE-PROCESSING TWEETS: VECTORIZATION

7.1 Vectorization

Word vectorization is a part of *Natural Language Processing (NLP)* and is one of the mandatory steps in machine learning. This is required because our interpreters cannot understand words by itself. Vectorizers are used to map words or phrases in the vocabulary to a corresponding vector of real number.

Two different types of vectorizers have been used in this project:

- CountVectorizer
- TF-IDF Vectorizer

7.2 Terminologies

- Corpus: collection of documents
- Vocabulary: collection of all the unique words within the corpus
- Document: every individual block of text (in this project, single tweet)
- Word: every individual word within a document

7.3 CountVectorizer

CountVectorizer is a part of the scikit-learn library of Python. It is based on the *Bag of Words* model. It can be used to convert a given word into a vector based on the frequency of each word in the entire block of text. It has a number of disadvantages, some of them are:

- The words which are abundantly present in the vocabulary are simply considered as significant.
- Discrimination between more important words and less important words is lacking in this model.
- It is unable to study the similarity between words or phrases.

7.4 TF-IDF Vectorizer

TF-IDF stands for *Term Frequency – Inverse Document Frequency*. TF is a measure of the frequency of a term in a document:

$$tf = \frac{n}{Number\ of\ terms\ in\ the\ document} \tag{1}$$

Here, ‘n’ is the number of times term ‘t’ occurs in the document ‘d’. Every document has a separate TF value. Thus, this approach also provides the importance of a word in a document.

TF-IDF is considered better than CountVectorizer because successfully provides the significance of a word. Thus, non-important words can be removed for more efficient model building and lesser input size.

8. MACHINE LEARNING ALGORITHMS

8.1 Logistic Regression

Logistic regression is a supervised Machine Learning Algorithm. It can be used for predicting binary outcomes, i.e., 0 or 1. In this project, the outcome is binary – ‘traffic’ or ‘non-traffic’.

Logistic Regression estimates the probability of a given event, based on an input variable. The model is trained with a pre-classified dataset, and then predicts the outcomes for the data inputted.

The *Sigmoid* function is used in logistic regression in order to convert the result into a categorical value. Sigmoid function (logistic function) is a function that can convert a real value into a value between 0 and 1.

$$P = \frac{1}{1+e^{-z}} \qquad (2)$$

$$Z = \log(odds) \qquad (3)$$

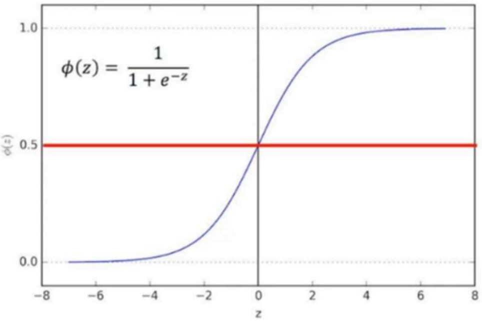


Figure 2. Sigmoid Curve

8.2 Naïve-Bayes Model

This is also a supervised machine learning algorithm, which is based on the *Bayes Theorem*. It completely works on the basis of Probability of an object, mainly Conditional Probability. This algorithm is generally used in case of high-dimensional training dataset.

This theorem is based on the Bayes Theorem of mathematics, which deals with conditional probability. It means the probability of an even to occur is calculated based on the probability of another event which has already occurred.

$$P(A|B) = \frac{P(B|A).P(A)}{P(B)} \qquad (4)$$

Here,

A is an event dependent on B

P(A|B) is the probability of A after event B has taken place

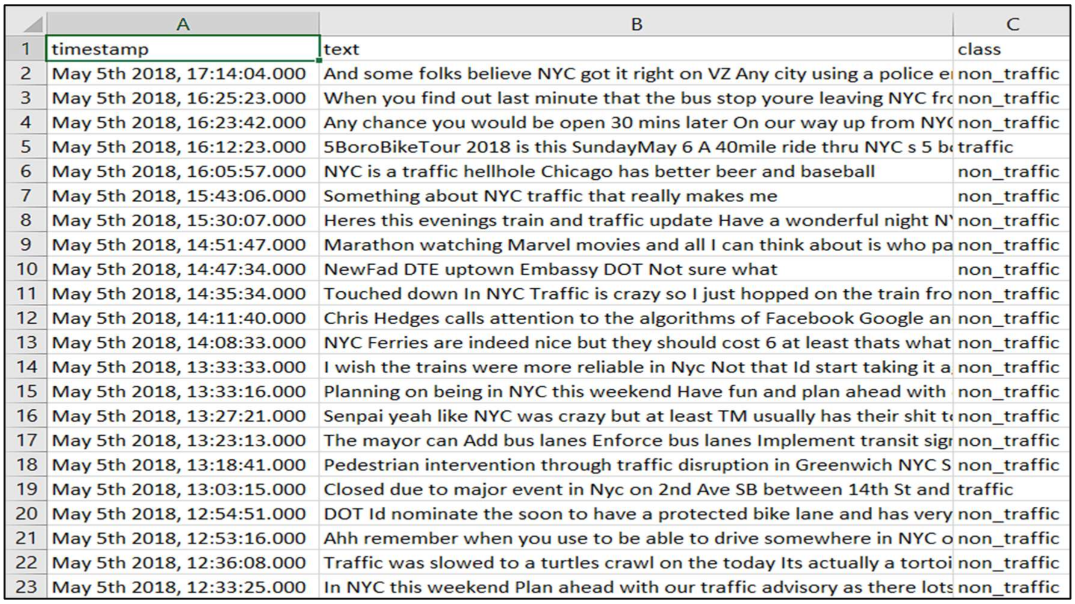
8.3 Random Forest Classifier

A type of supervised machine learning, Random Forest Classifier can be used for classification, regression and other purposes using decision trees. A set of decision trees are created from the training set. Votes from various decision trees are collected to give out the final prediction.

Decision Trees can be used for both classification and regression. Trees answer some sequence of questions and is we move down the tree, we get our answers accordingly. These trees work on the concept of “if this than that”, which penultimately gives us a final result.

9. RESULTS AND DISCUSSION

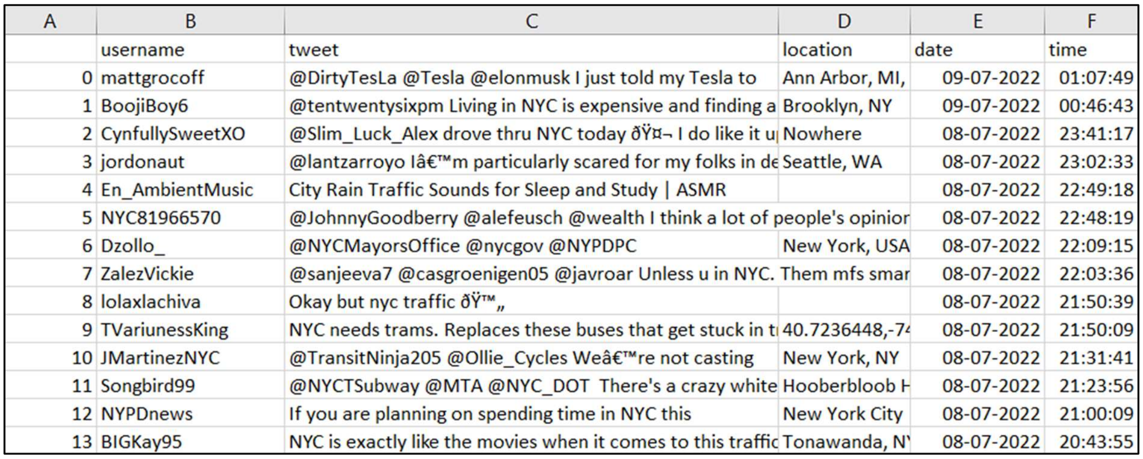
9.1 Datasets



	A	B	C
1	timestamp	text	class
2	May 5th 2018, 17:14:04.000	And some folks believe NYC got it right on VZ Any city using a police e	non_traffic
3	May 5th 2018, 16:25:23.000	When you find out last minute that the bus stop youre leaving NYC fr	non_traffic
4	May 5th 2018, 16:23:42.000	Any chance you would be open 30 mins later On our way up from NY(	non_traffic
5	May 5th 2018, 16:12:23.000	5BoroBikeTour 2018 is this SundayMay 6 A 40mile ride thru NYC s 5 b	traffic
6	May 5th 2018, 16:05:57.000	NYC is a traffic hellhole Chicago has better beer and baseball	non_traffic
7	May 5th 2018, 15:43:06.000	Something about NYC traffic that really makes me	non_traffic
8	May 5th 2018, 15:30:07.000	Heres this evenings train and traffic update Have a wonderful night N	non_traffic
9	May 5th 2018, 14:51:47.000	Marathon watching Marvel movies and all I can think about is who pa	non_traffic
10	May 5th 2018, 14:47:34.000	NewFad DTE uptown Embassy DOT Not sure what	non_traffic
11	May 5th 2018, 14:35:34.000	Touched down In NYC Traffic is crazy so I just hopped on the train fro	non_traffic
12	May 5th 2018, 14:11:40.000	Chris Hedges calls attention to the algorithms of Facebook Google an	non_traffic
13	May 5th 2018, 14:08:33.000	NYC Ferries are indeed nice but they should cost 6 at least thats what	non_traffic
14	May 5th 2018, 13:33:33.000	I wish the trains were more reliable in Nyc Not that Id start taking it a	non_traffic
15	May 5th 2018, 13:33:16.000	Planning on being in NYC this weekend Have fun and plan ahead with	non_traffic
16	May 5th 2018, 13:27:21.000	Senpai yeah like NYC was crazy but at least TM usually has their shit t	non_traffic
17	May 5th 2018, 13:23:13.000	The mayor can Add bus lanes Enforce bus lanes Implement transit sig	non_traffic
18	May 5th 2018, 13:18:41.000	Pedestrian intervention through traffic disruption in Greenwich NYC S	non_traffic
19	May 5th 2018, 13:03:15.000	Closed due to major event in Nyc on 2nd Ave SB between 14th St and	traffic
20	May 5th 2018, 12:54:51.000	DOT Id nominate the soon to have a protected bike lane and has very	non_traffic
21	May 5th 2018, 12:53:16.000	Ahh remember when you use to be able to drive somewhere in NYC o	non_traffic
22	May 5th 2018, 12:36:08.000	Traffic was slowed to a turtles crawl on the today Its actually a tortoi	non_traffic
23	May 5th 2018, 12:33:25.000	In NYC this weekend Plan ahead with our traffic advisory as there lots	non_traffic

Figure 3. CSV File: Training dataset Source: https://github.com/SivagurunathanV/Traffic-Detection-from-Twitter-using-Spark/blob/master/src/twitter_traffic_data_static.csv

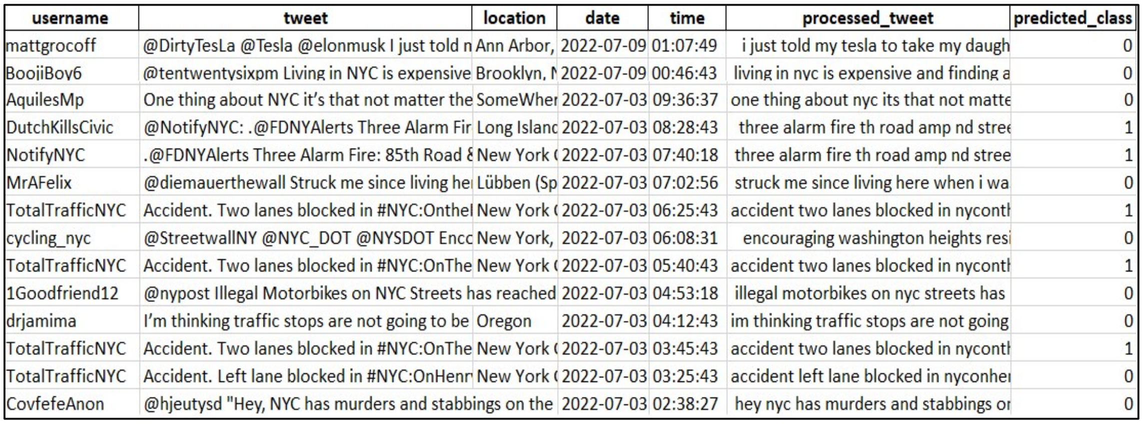
The above training dataset was read using Pandas and the 'class' column was converted into numerical column – 1 for traffic and 0 for non-traffic.



A	B	C	D	E	F
	username	tweet	location	date	time
0	mattgrocoff	@DirtyTesLa @Tesla @elonmusk I just told my Tesla to	Ann Arbor, MI,	09-07-2022	01:07:49
1	BoojiBoy6	@tentwentysixpm Living in NYC is expensive and finding a	Brooklyn, NY	09-07-2022	00:46:43
2	CynfullySweetXO	@Slim_Luck_Alex drove thru NYC today ðŸ¤¬ I do like it u	Nowhere	08-07-2022	23:41:17
3	jordonaut	@lantzarroyo Iâ€™m particularly scared for my folks in de	Seattle, WA	08-07-2022	23:02:33
4	En_AmbientMusic	City Rain Traffic Sounds for Sleep and Study | ASMR		08-07-2022	22:49:18
5	NYC81966570	@JohnnyGoodberry @alefeusch @wealth I think a lot of people's opinior		08-07-2022	22:48:19
6	Dzollo_	@NYCMayorsOffice @nycgov @NYPDPC	New York, USA	08-07-2022	22:09:15
7	ZalezVickie	@sanjeeva7 @casgroenigen05 @javroar Unless u in NYC. Them mfs smar		08-07-2022	22:03:36
8	lolaxlachiva	Okay but nyc traffic ðŸ™„		08-07-2022	21:50:39
9	TVariunessKing	NYC needs trams. Replaces these buses that get stuck in t	40.7236448,-7	08-07-2022	21:50:09
10	JMartinezNYC	@TransitNinja205 @Ollie_Cycles Weâ€™re not casting	New York, NY	08-07-2022	21:31:41
11	Songbird99	@NYCTSubway @MTA @NYC_DOT There's a crazy white	Hooberbloob H	08-07-2022	21:23:56
12	NYPDnews	If you are planning on spending time in NYC this	New York City	08-07-2022	21:00:09
13	BIGKay95	NYC is exactly like the movies when it comes to this traffic	Tonawanda, N	08-07-2022	20:43:55

Figure 4. CSV File: Fetched tweets

The tweets were fetched from Twitter API using TweePy and stored in the computer memory in the form of a CSV file.



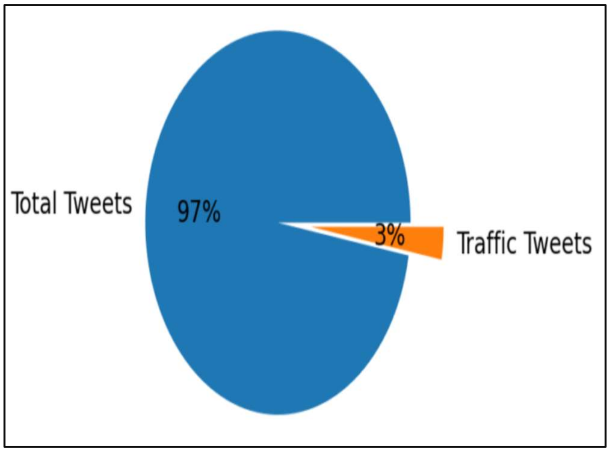
username	tweet	location	date	time	processed_tweet	predicted_class
mattgrocoff	@DirtyTesLa @Tesla @elonmusk I just told n	Ann Arbor,	2022-07-09	01:07:49	i just told my tesla to take my daugh	0
BooiiBov6	@tentwentysixpm Living in NYC is expensive	Brooklyn, N	2022-07-09	00:46:43	living in nyc is expensive and finding a	0
AquilesMp	One thing about NYC it's that not matter the	SomeWher	2022-07-03	09:36:37	one thing about nyc its that not matte	0
DutchKillsCivic	@NotifyNYC: .@FDNYAlerts Three Alarm Fir	Long Islanc	2022-07-03	08:28:43	three alarm fire th road amp nd stree	1
NotifyNYC	.@FDNYAlerts Three Alarm Fire: 85th Road &	New York (	2022-07-03	07:40:18	three alarm fire th road amp nd stree	1
MrAFelix	@diemauerthewall Struck me since living he	Lübben (Sp	2022-07-03	07:02:56	struck me since living here when i wa	0
TotalTrafficNYC	Accident. Two lanes blocked in #NYC:Onthel	New York (	2022-07-03	06:25:43	accident two lanes blocked in nyconth	1
cycling_nyc	@StreetwallNY @NYC_DOT @NYSDOT Encc	New York,	2022-07-03	06:08:31	encouraging washington heights resi	0
TotalTrafficNYC	Accident. Two lanes blocked in #NYC:OnThe	New York (	2022-07-03	05:40:43	accident two lanes blocked in nyconth	1
1Goodfriend12	@nypost Illegal Motorbikes on NYC Streets has reached		2022-07-03	04:53:18	illegal motorbikes on nyc streets has	0
drjamima	I'm thinking traffic stops are not going to be	Oregon	2022-07-03	04:12:43	im thinking traffic stops are not going	0
TotalTrafficNYC	Accident. Two lanes blocked in #NYC:OnThe	New York (	2022-07-03	03:45:43	accident two lanes blocked in nyconth	1
TotalTrafficNYC	Accident. Left lane blocked in #NYC:OnHenr	New York (	2022-07-03	03:25:43	accident left lane blocked in nyconher	0
CovfefeAnon	@hjeutysd "Hey, NYC has murders and stabbings on the		2022-07-03	02:38:27	hey nyc has murders and stabbings or	0

Figure 5. CSV File: Final result after pre-processing and classification

The final dataset after the fetched tweets were, pre-processed, lemmatized, vectorized and classified successfully by the model we designed.

9.2 Data Visualisation

Graphs depicting the ratio of tweets classified as traffic to the total volume of tweets fetched



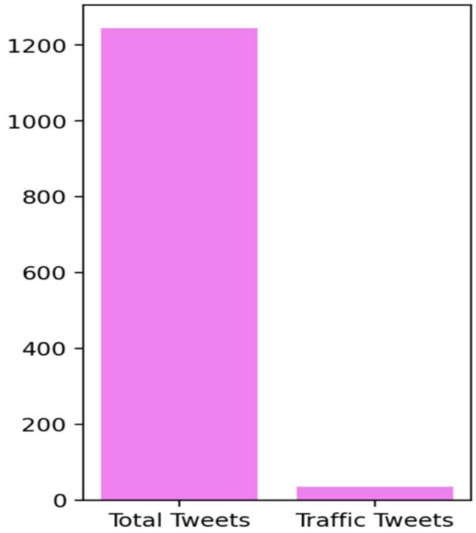


Figure 6. Pie-Chart

Figure 7. Bar Plot

The above graphs depict that although a large volume of tweets is fetched from Twitter API, all the tweets do not significantly ensure that they relay information about traffic conditions in the specified city.

ROC Curve and AUROC:

The ROC Plots are *Receiver Operating Characteristic Curves* which depict the performance of the classification models and various levels of threshold. They use two essential parameters:

- *True Positive Rate (TPR)*
- *False Positive Rate (FPR)*

$$TPR = \frac{TP}{TP + FN} \qquad (5) \qquad\qquad FPR = \frac{FP}{FP + TN} \qquad (6)$$

TP: True Positives; FP: False Positives; TN: True Negatives; FN: False Negatives

AUROC stands for *Area Under a ROC Curve*. The greater the are under the curve, greater is the accuracy of our model.

Sample ROC Curve obtained with AUROC Values:

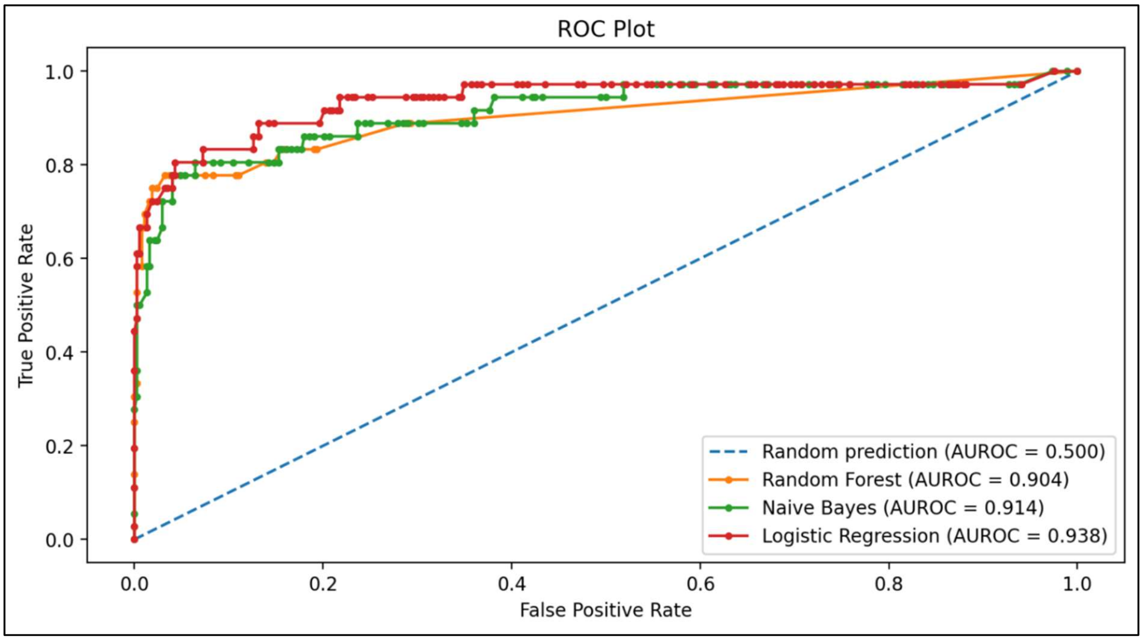
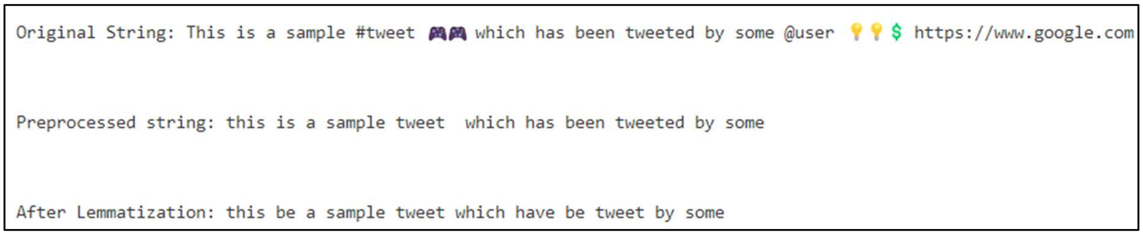


Figure 8. ROC Plot

9.3 Pre-Processing Outputs



```
Original String: This is a sample #tweet 🎮🎮 which has been tweeted by some @user 💡💡💲 https://www.google.com


Preprocessed string: this is a sample tweet  which has been tweeted by some


After Lemmatization: this be a sample tweet which have be tweet by some
```

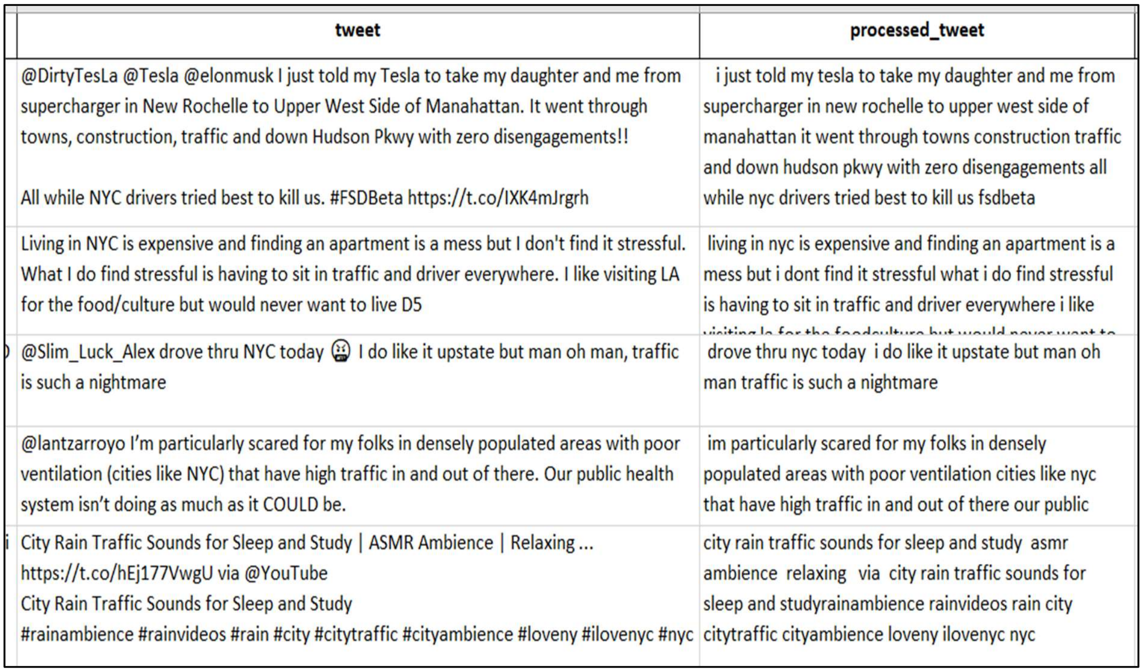
Figure 9. Sample output: Functionality of pre-processing tool

The following items were removed from the sample input:

- Symbols like Hashtags, etc.
- Emojis
- Mentions
- Links

Also, the output was later lemmatized, the words got converted to their lemma.

Sample Output showing difference between fetched tweets before and after pre-processing:



tweet	processed_tweet
@DirtyTesLa @Tesla @elonmusk I just told my Tesla to take my daughter and me from supercharger in New Rochelle to Upper West Side of Manahattan. It went through towns, construction, traffic and down Hudson Pkwy with zero disengagements!! All while NYC drivers tried best to kill us. #FSDBeta https://t.co/IXK4mJrgrh	i just told my tesla to take my daughter and me from supercharger in new rochelle to upper west side of manahattan it went through towns construction traffic and down hudson pkwy with zero disengagements all while nyc drivers tried best to kill us fsdbeta
Living in NYC is expensive and finding an apartment is a mess but I don't find it stressful. What I do find stressful is having to sit in traffic and driver everywhere. I like visiting LA for the food/culture but would never want to live D5	living in nyc is expensive and finding an apartment is a mess but i dont find it stressful what i do find stressful is having to sit in traffic and driver everywhere i like visiting la for the foodculture but would never want to
@Slim_Luck_Alex drove thru NYC today 🤬 I do like it upstate but man oh man, traffic is such a nightmare	drove thru nyc today i do like it upstate but man oh man traffic is such a nightmare
@lantzarroyo I'm particularly scared for my folks in densely populated areas with poor ventilation (cities like NYC) that have high traffic in and out of there. Our public health system isn't doing as much as it COULD be.	im particularly scared for my folks in densely populated areas with poor ventilation cities like nyc that have high traffic in and out of there our public
City Rain Traffic Sounds for Sleep and Study | ASMR Ambience | Relaxing ... https://t.co/hEj177VwgU via @YouTube City Rain Traffic Sounds for Sleep and Study #rainambience #rainvideos #rain #city #citytraffic #cityambience #loveny #ilovenyc #nyc	city rain traffic sounds for sleep and study asmr ambience relaxing via city rain traffic sounds for sleep and studyrainambience rainvideos rain city citytraffic cityambience loveny ilovenyc nyc

Figure 10. Difference between the tweets before and after pre-processing

10. GRAPHICAL USER INTERFACE (GUI)

10.1 Technologies

- StreamLit
- Pandas
- Matplotlib and Seaborn
- HTML and CSS (Cascading Style Sheets)
- NumPy

10.2 Code Snippets

Sample code for importing StreamLit and setting the Page Layout:

```
from pyparsing import col
import streamlit as st

st.set_page_config(
   page_title="Twitter Traffic Analysis",
   page_icon=" 🚦 ",
   layout="wide"
)

st.title("TRAFFIC ANALYSIS FROM TWITTER STREAMS")
```

Sample code for accepting Inputs (Text Boxes, Sliders, Dropdowns):

```
city = st.text_input("TYPE IN THE NAME OF THE CITY", placeholder="Example : Seattle")
split_size = st.slider("DATASET SPLITTING SIZE (Training : Testing)", min_value=0.2,
max_value=0.8, step=0.1, value=0.3)

col_opt1, col_opt2 = st.columns([1,1])
vector_type = col_opt1.selectbox("CHOOSE THE VECTORIZER", options=['COUNT
VECTORIZER','TFIDF VECTORIZER'])

model_type = col_opt2.selectbox("CHOOSE THE MODEL TO BE USED",
options=['LOGISTIC REGRESSION','NAIVE - BAYES MODEL', 'RANDOM FOREST
CLASSIFIER'])

city_but = st.button("SEARCH FOR TWEETS")
```

Sample code for displaying the classified tweets and metrics:

```
st.title("RESULTS")
st.header("Traffic - Related tweets in {0}".format(city.capitalize()))
st.dataframe(df)

col_metric1, col_metric2, col_metric3 = st.columns([1,1,1])
col_metric1.metric(label="Trafic tweets fetched", value=df_size)
col_metric2.metric(label="Tweets classified as 'Traffic'", value=new_df_size)
csv = convert_df(df)
col_metric3.download_button(label="Download the Tweets as a CSV", data=csv,
file_name="search.csv", mime='text/csv')
```

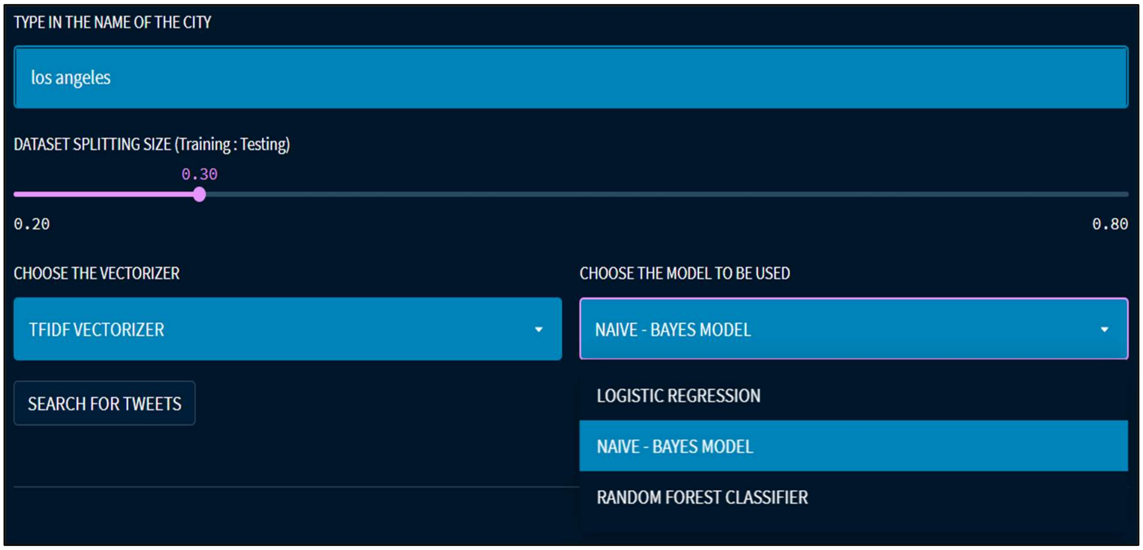
Sample code for plotting graphs:

```
plt.figure(figsize=(10,5))
plt.subplot(1,3,1)
plt.pie([df_size, new_df_size], labels = ['Total Tweets', 'Traffic Tweets'], explode=[0,0.25],
autopct='%.0f%%')
plt.subplot(1,3,3)
plt.bar(['Total Tweets','Traffic Tweets'], [df_size, new_df_size], color='violet')
st.pyplot()
```

Sample code for ROC plot:

```
exp1 = st.expander("SHOW ACCURACY SCORES FOR VARIOUS MODELS")
exp1.table(scores)
exp2 = st.expander("SHOW ROC CURVES FOR THE MODELS")
plt.plot(r_fpr, r_tpr, linestyle='--', label='Random prediction (AUROC = %0.3f)' % r_auc)
plt.plot(rf_fpr, rf_tpr, marker='.', label='Random Forest (AUROC = %0.3f)' % rf_auc)
plt.plot(nb_fpr, nb_tpr, marker='.', label='Naive Bayes (AUROC = %0.3f)' % nb_auc)
plt.plot(logreg_fpr, logreg_tpr, marker='.', label='Logistic Regression (AUROC = %0.3f)' %
logreg_auc)
# Title
plt.title('ROC Plot')
# Axis labels
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
# Show legend
plt.legend() #
# Show plot
plt.show()
exp2.pyplot()
```

10.3 Test Inputs and Outputs



TYPE IN THE NAME OF THE CITY

los angeles

DATASET SPLITTING SIZE (Training : Testing)

0.30

0.20 0.80

CHOOSE THE VECTORIZER

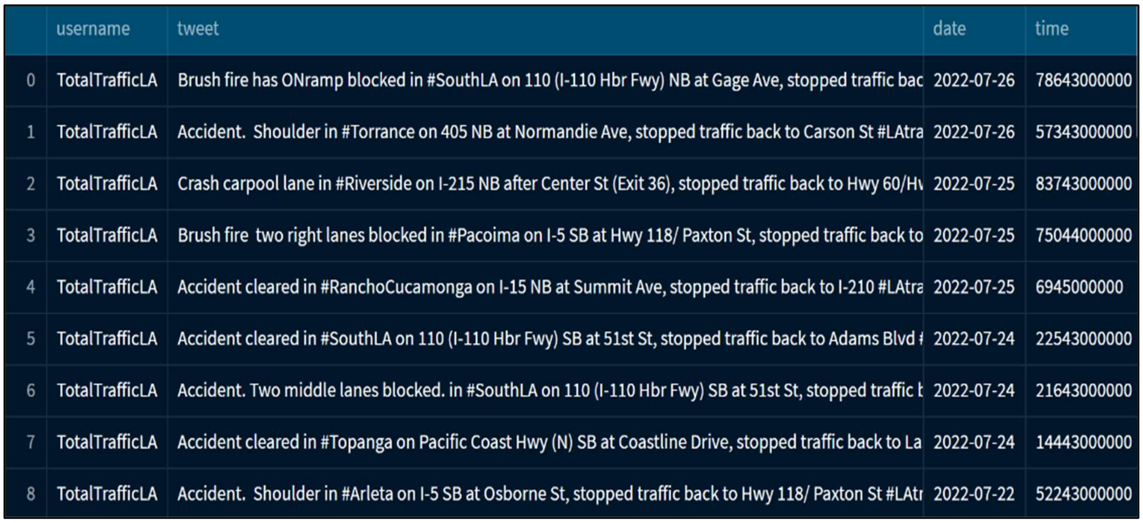
TFIDF VECTORIZER

CHOOSE THE MODEL TO BE USED

NAIVE - BAYES MODEL

Figure 11. Sample input

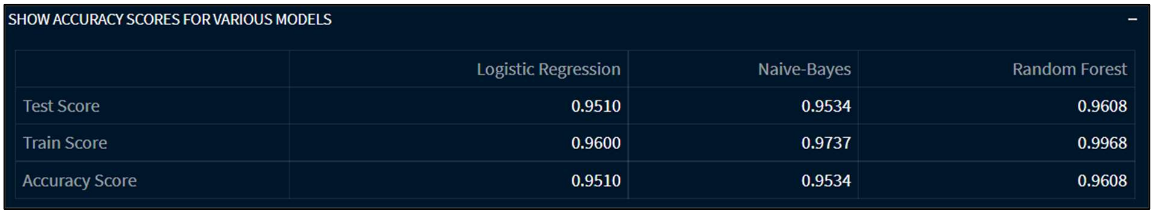
'Los Angeles', a city in the US was inputted in the text box. The split size was set to 0.3 and TFIDF – Naïve-Bayes combination was chosen for classification.



	username	tweet	date	time
0	TotalTrafficLA	Brush fire has ONramp blocked in #SouthLA on 110 (I-110 Hbr Fwy) NB at Gage Ave, stopped traffic bac	2022-07-26	78643000000
1	TotalTrafficLA	Accident. Shoulder in #Torrance on 405 NB at Normandie Ave, stopped traffic back to Carson St #LAtra	2022-07-26	57343000000
2	TotalTrafficLA	Crash carpool lane in #Riverside on I-215 NB after Center St (Exit 36), stopped traffic back to Hwy 60/Hv	2022-07-25	83743000000
3	TotalTrafficLA	Brush fire two right lanes blocked in #Pacoima on I-5 SB at Hwy 118/ Paxton St, stopped traffic back to	2022-07-25	75044000000
4	TotalTrafficLA	Accident cleared in #RanchoCucamonga on I-15 NB at Summit Ave, stopped traffic back to I-210 #LAtra	2022-07-25	6945000000
5	TotalTrafficLA	Accident cleared in #SouthLA on 110 (I-110 Hbr Fwy) SB at 51st St, stopped traffic back to Adams Blvd #	2022-07-24	22543000000
6	TotalTrafficLA	Accident. Two middle lanes blocked. in #SouthLA on 110 (I-110 Hbr Fwy) SB at 51st St, stopped traffic b	2022-07-24	21643000000
7	TotalTrafficLA	Accident cleared in #Topanga on Pacific Coast Hwy (N) SB at Coastline Drive, stopped traffic back to La	2022-07-24	14443000000
8	TotalTrafficLA	Accident. Shoulder in #Arleta on I-5 SB at Osborne St, stopped traffic back to Hwy 118/ Paxton St #LAtr	2022-07-22	52243000000

Figure 12. Fetched tweets

The successfully fetched tweets from the API were displayed in the form of a table which can later be downloaded as a CSV file for further reference.

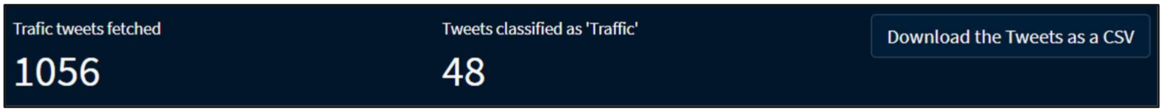


SHOW ACCURACY SCORES FOR VARIOUS MODELS

	Logistic Regression	Naive-Bayes	Random Forest
Test Score	0.9510	0.9534	0.9608
Train Score	0.9600	0.9737	0.9968
Accuracy Score	0.9510	0.9534	0.9608

Figure 13. Accuracy scores

Different algorithms are seen to have different accuracy scores. This is because each one has their unique way of working. The one with higher score must be preferred.



Trafic tweets fetched

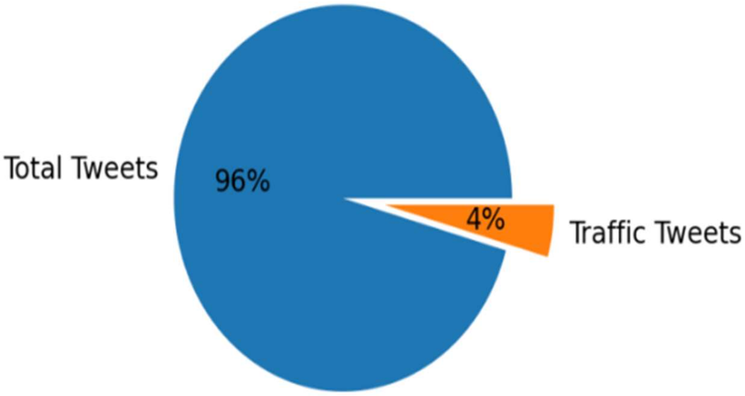
1056

Tweets classified as 'Traffic'

48

Figure 14. 48 Metrics

Not all the tweets which are being fetched are related to traffic. In this case, 48 tweets were actually seen to have been related to traffic.



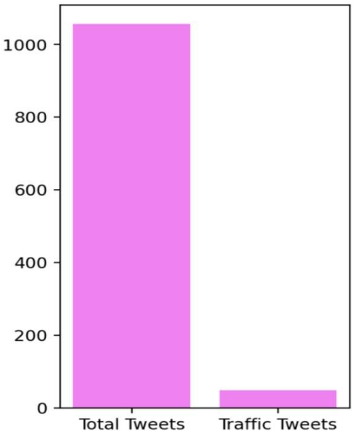


Figure 15. Pie-Chart (Sample Test)

Figure 16. Bar Plot (Sample Test)

The relation or proportion between total tweets fetched and actual classified-traffic tweets are depicted using various types of plots.

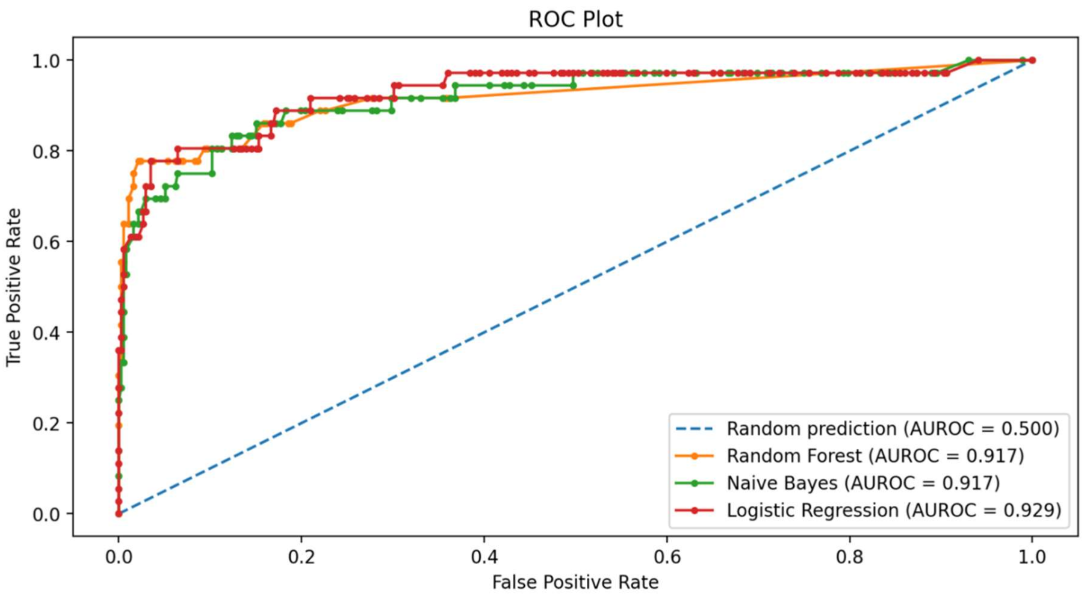
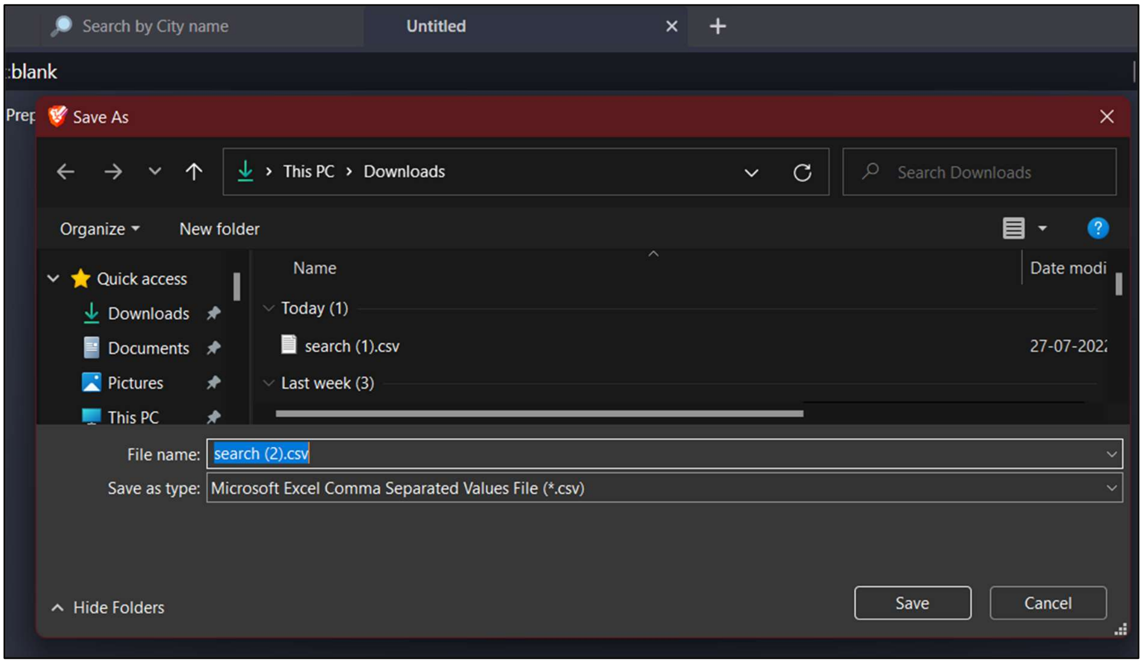


Figure 17. ROC plot and AUROC: Sample test

The AUROC values are actually the area under the curves which have been plotted. This plot helps in comparing accuracies of various algorithms.



search (1).csv 27-07-202

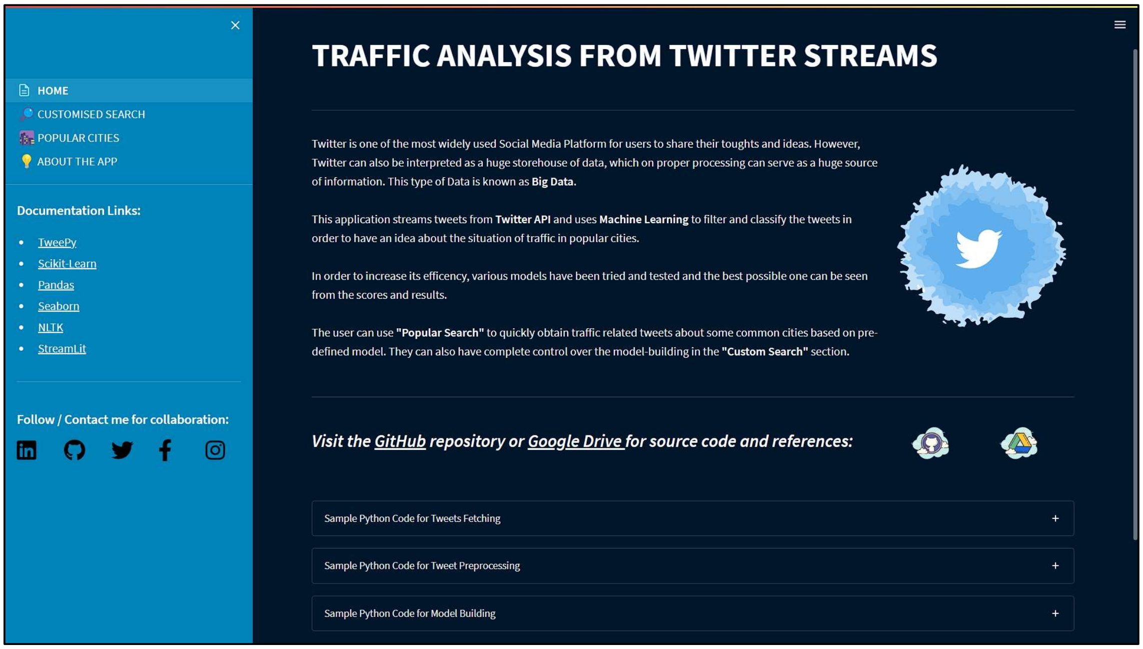
File name: search (2).csv

Save as type: Microsoft Excel Comma Separated Values File (*.csv)

Figure 18. Downloading CSV dialogue box

On clicking the ‘Download the Tweet as CSV’ button, a dialogue box appears which enables us to save the dataset as a CSV file in the device’s memory.

10.4 Deployment



TRAFFIC ANALYSIS FROM TWITTER STREAMS

Twitter is one of the most widely used Social Media Platform for users to share their toughts and ideas. However, Twitter can also be interpreted as a huge storehouse of data, which on proper processing can serve as a huge source of information. This type of Data is known as **Big Data.**

This application streams tweets from **Twitter API** and uses **Machine Learning** to filter and classify the tweets in order to have an idea about the situation of traffic in popular cities.

In order to increase its efficency, various models have been tried and tested and the best possible one can be seen from the scores and results.

The user can use **"Popular Search"** to quickly obtain traffic related tweets about some common cities based on pre-defined model. They can also have complete control over the model-building in the **"Custom Search"** section.

Visit the GitHub repository or Google Drive for source code and references:

Sample Python Code for Tweets Fetching +

Sample Python Code for Tweet Preprocessing +

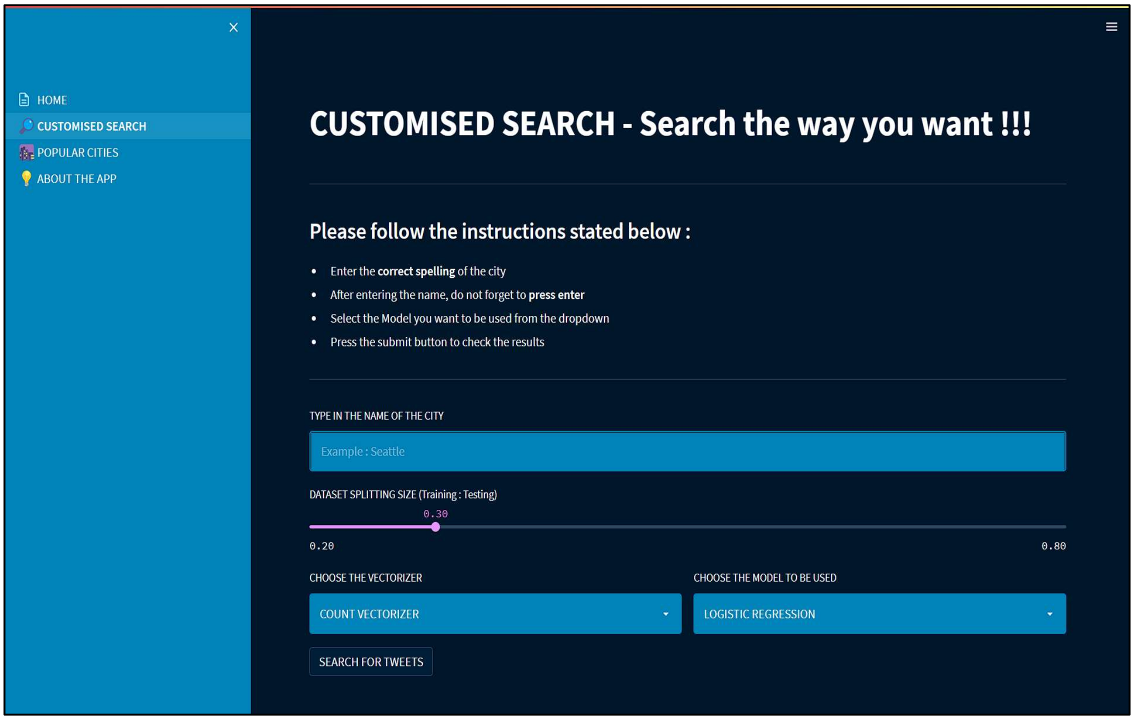
Sample Python Code for Model Building +

Figure 19. The Home Page

The Home Page serves as the landing page for the application. The general instructions for using the application, sample codes and various links to documentations are available in this page. Various HTML and CSS properties have also been used in order to make the page attractive.

The contents of the page include:

- Overview of the App
- General instructions
- GitHub and Drive links
- Sample codes for reference
- Documentation links
- Social – Media links



CUSTOMISED SEARCH - Search the way you want !!!

Please follow the instructions stated below :

- Enter the **correct spelling** of the city
- After entering the name, do not forget to **press enter**
- Select the Model you want to be used from the dropdown
- Press the submit button to check the results

TYPE IN THE NAME OF THE CITY

Example : Seattle

DATASET SPLITTING SIZE (Training : Testing)

0.30

0.20 0.80

CHOOSE THE VECTORIZER: COUNT VECTORIZER

CHOOSE THE MODEL TO BE USED: LOGISTIC REGRESSION

Figure 20. Customised Search page

This is the most important page of the app, where the user can have absolute versatility in obtaining tweets from Twitter API. One can set his own sets of parameters and choices for the prediction-classification model. The results of the search are displayed within this page itself, with the additional option of downloading the fetched tweets as a CSV file.

The contents of the page include:

- General Instructions
- City name text input box
- Slider to input the test-train split ratio
- Dropdowns to select the vectorizer and machine learning algorithm to be used for the model
- Search button to submit our choices

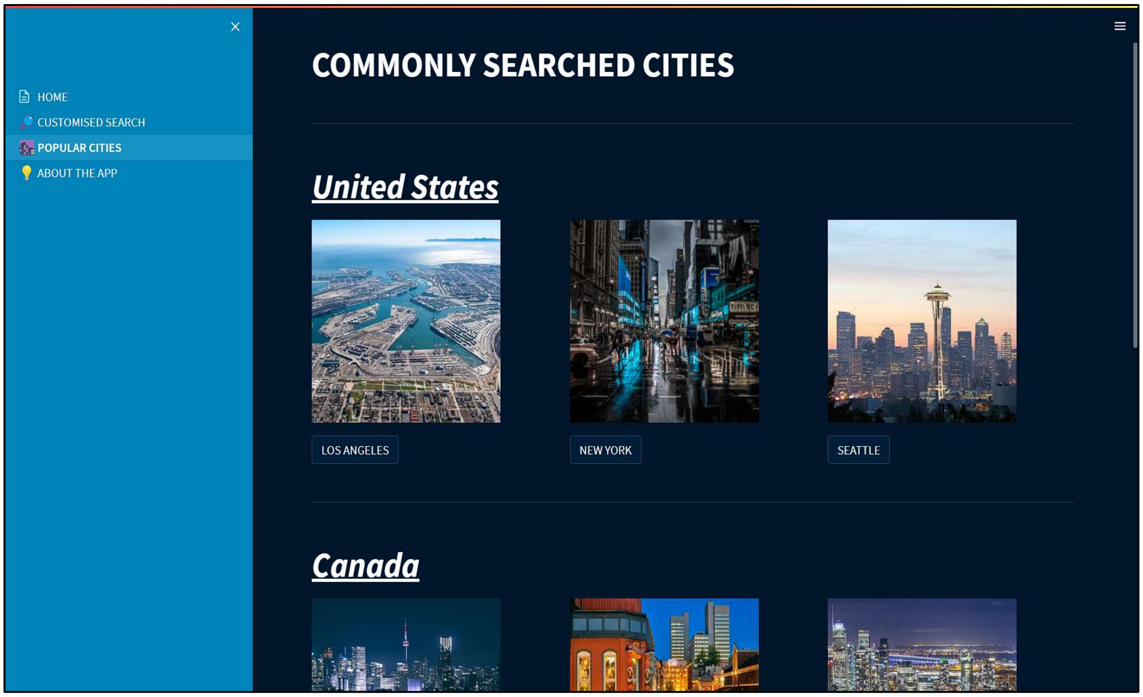
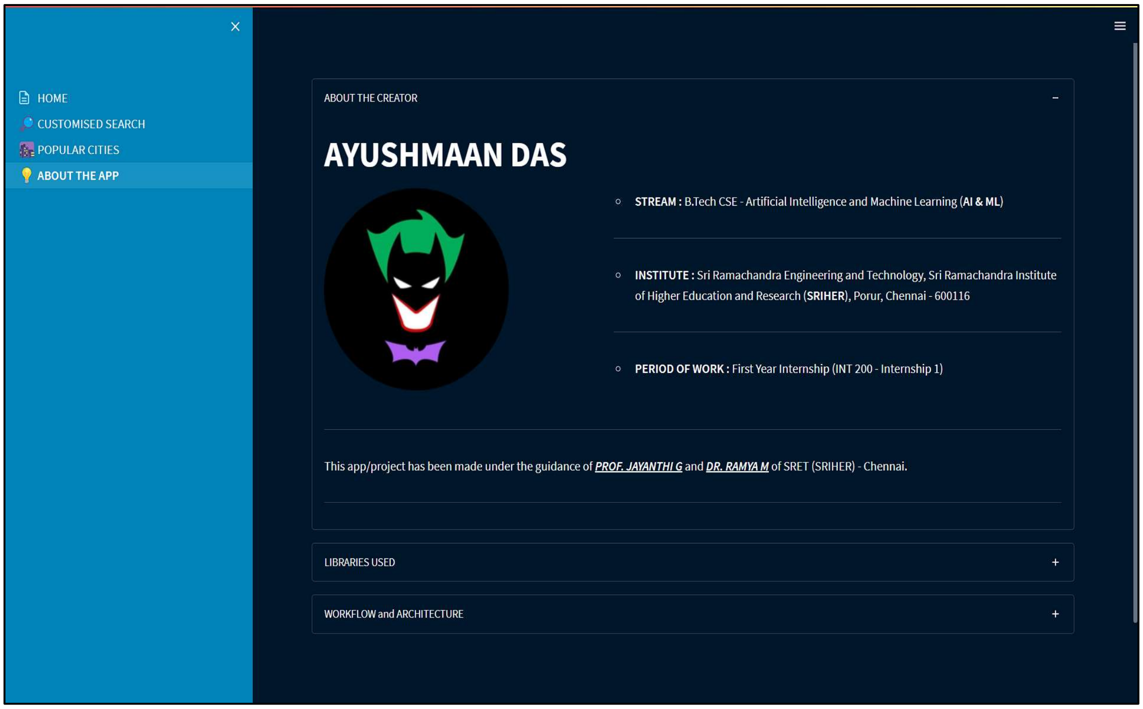


Figure 21. Commonly searched cities page

This page has been designed for time-saving and user accessibility purpose. Tweets can be fetched for the cities with just a single click of the button. All other functionalities, such as fetched tweets dataset, downloading the dataset, visualization and ROC plots, are similar to that of the Customised Search page.

The following cities have been included:

- Los Angeles (USA)
- New York (USA)
- Seattle (USA)
- London (UK)
- Glasgow (UK)
- Manchester (UK)
- Ottawa (Canada)
- Toronto (Canada)
- Montreal (Canada)



ABOUT THE CREATOR

AYUSHMAAN DAS

- **STREAM :** B.Tech CSE - Artificial Intelligence and Machine Learning (**AI & ML**)
- **INSTITUTE :** Sri Ramachandra Engineering and Technology, Sri Ramachandra Institute of Higher Education and Research (**SRIHER**), Porur, Chennai - 600116
- **PERIOD OF WORK :** First Year Internship (INT 200 - Internship 1)

This app/project has been made under the guidance of ***PROF. JAYANTHI G*** and ***DR. RAMYA M*** of SRET (SRIHER) - Chennai.

Figure 22. About the App page

The ‘About the App’ page, as the name suggests, relays information about the app top the user about the developer and working of the app. The libraries used for the deployment of the app are mentioned and also, the workflow of the application can be accessed for the user to understand about the way the application fetches and classifies tweets from the user.

The contents of the page include:

- User information
- Organization information
- Modules and libraries of Python utilised for this project
- The Data Pipeline: Data Pipeline: Pre-processing tweets and Machine learning workflow for classification

11. SCOPE FOR FURTHER ENHANCEMENT

This project was developed as a part of the internship for the first year. Due to limited time and also limited access to resources, there were certain aspects which couldn't be covered. Hence, it is intended to enhance the web-application in all possible ways in the near future.

The following tasks are intended to be included within the app:

- Increasing the number of Machine Learning algorithms being used in order to provide more versatility and wider scope for the users.
- Using various other types of vectorizers available.
- Development of a proper user registration and login system.
- Linking the Application to a Database.
- Development of a SMS-based or Email-based alert system.
- Solve the problems regarding to app deployment in streamlit cloud.

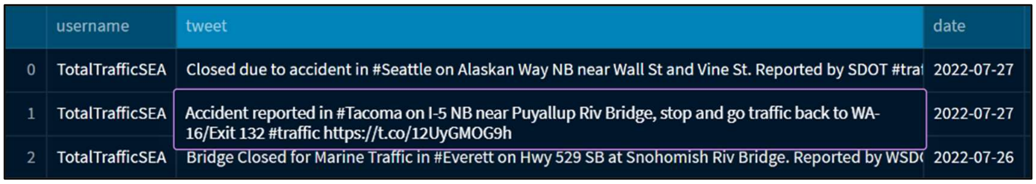
The aforementioned concepts can be implemented with further study and research work. This project has a very wide scope which can be explored properly in the future with the gain of information. It can be made even more flexible, versatile and environment-independent.

12. CONCLUSION

In this project, traffic-related tweets related to a particular city have been successfully mined from Twitter API. A proper pipeline was designed for Pre-Processing of tweets followed by model building and Machine Learning for classification of the fetched tweets.

Various kinds of machine learning algorithms as well as vectorization approaches were used which gave a wide variety of results. Thus, we concluded that different approaches had different accuracy values, which were visualised using the ROC Plots. Versatility and efficiency were achieved by creating multiple approaches for model building.

In the end, all the modules were integrated into a web-application which was designed and deployed using StreamLit. The app was made versatile and interactive which enabled the users to fetch traffic-related tweets from the city of their choice.



	username	tweet	date
0	TotalTrafficSEA	Closed due to accident in #Seattle on Alaskan Way NB near Wall St and Vine St. Reported by SDOT #tra	2022-07-27
1	TotalTrafficSEA	Accident reported in #Tacoma on I-5 NB near Puyallup Riv Bridge, stop and go traffic back to WA-16/Exit 132 #traffic https://t.co/12UyGMOG9h	2022-07-27
2	TotalTrafficSEA	Bridge Closed for Marine Traffic in #Everett on Hwy 529 SB at Snohomish Riv Bridge. Reported by WSD	2022-07-26

Figure 23. Traffic tweets for 'Seattle'

For the test case of 'Seattle', the user can interpret that they should avoid going through 'Tacoma' on 27th July, 2022 as there has been a roadblock.

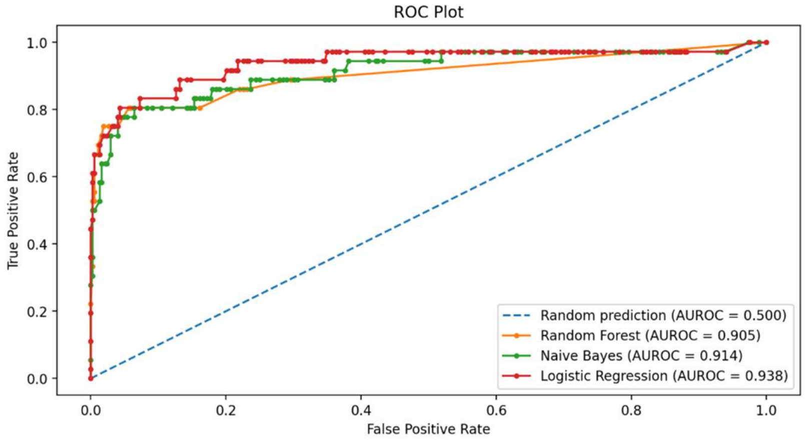


Figure 24. ROC Plot for 'Seattle'

The ROC Curve for the city of 'Seattle' using Count Vectorizer showed that Logistic Regression was having the greatest AUROC, hence the best accuracy.

13. PROJECT DEPLOYMENT LINK

This project has been deployed in GitHub under an organisation of the Project Mentor. The sample codes, documents and other additional files can be referred to through this repository.

The repository specifications are:

- Organization name: B Tech 2021-25 INT 200
- Repository name: E Traffic Alert System
- Licensing: GPL-3.0

Link: https://github.com/B-TECH-2021-25-INT200/E-Traffic-Alert-System

The folder 'Final review' and 'App' contains all the source codes and associated files. The python file 'HOME.py' must be run using streamlit.

The app can be run in streamlit using the command:

- "*Streamlit run HOME.py*"

14. TIMELINE

Gantt Chart depicting the timeline of work done during the course of the internship:

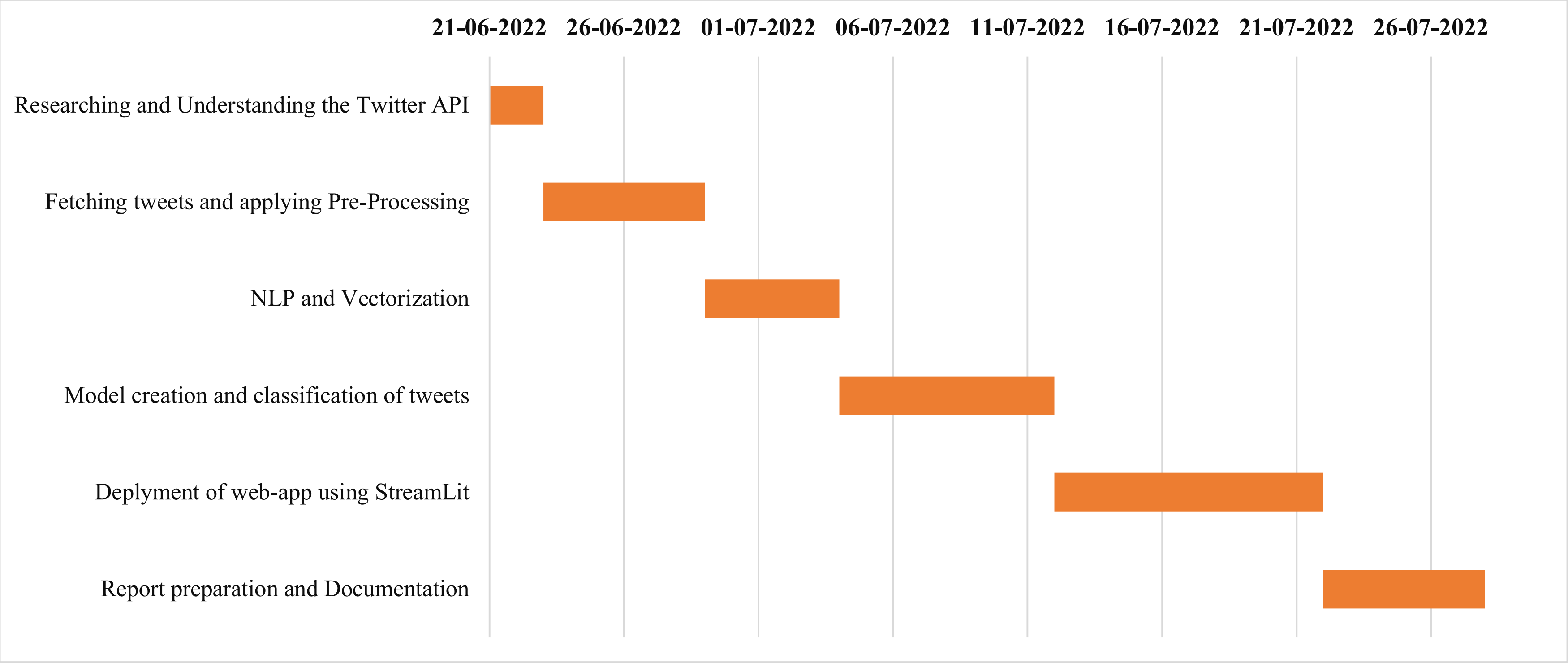


Figure 25. Gantt Chart

15. WORKLOG

Table 4. Daily worklog for the internship

Day	Date	Work
Day 1	21-06-2022	Studying and Researching about Data Scrapping, going through existing products
Day 2	22-06-2022	Twitter Developer Account, understanding how Twitter API works
Day 3	23-06-2022	Obtaining and Extracting tweets using 'TweePy'
Day 4	24-06-2022	Basic Pre-Processing of tweets extracted
Day 5	25-06-2022	Introduction to 'Scikit-Learn' and implementing it
Day 6	26-06-2022	Storing the fetched tweets, going through machine learning models
Day 7	27-06-2022	Fetching Tweets from Twitter, storing and applying preprocessing
Day 8	28-06-2022	Preparation for 1st Review – Presentation, Modules and Workflow
Day 9	29-06-2022	1st Review
Day 10	30-06-2022	Understanding Tokenization, implementation using Python
Day 11	01-07-2022	Learning Stemming & Lemmatization
Day 12	02-07-2022	Introduction to Text Classification, researching about NLP
Day 13	03-07-2022	Bag of Word approach for classification; unifying all concepts
Day 14	04-07-2022	Attempt to classify the tweets – Training and Predicting
Day 15	05-07-2022	Continuation of Tweet Classification using multiple algorithms
Day 16	06-07-2022	Unifying all concepts and testing a sample case
Day 17	07-07-2022	Roc Curves for different classification models used for the process
Day 18	08-07-2022	Creating a jupyter notebook in sequential format, implementing the entire pipeline until now
Day 19	09-07-2022	Researching about StreamLit, going through applications and existing works

Day 20	10-07-2022	Preparation for Second Review - Reviewing the objectives
Day 21	11-07-2022	2nd Review
Day 22	12-07-2022	Implementation of Streamlit - introduction of creating an app
Day 23	13-07-2022	Web App - designs, layouts, contents, etc.
Day 24	14-07-2022	Creation of Web App
Day 25	15-07-2022	Creation of Web App (continued)
Day 26	16-07-2022	Creation of Web App (continued)
Day 27	17-07-2022	Creation of Web App (continued)
Day 28	18-07-2022	Creation of Web App (continued)
Day 29	19-07-2022	Applying styles to the app and making it interactive
Day 30	20-07-2022	Final touches to the app and Testing of the app
Day 31	21-07-2022	Preparation for third review, demo of the application
Day 32	22-07-2022	3rd Review
Day 33	23-07-2022	Creation of project report – going through the documentation and formats to create a report
Day 34	24-07-2022	Preparation of report
Day 35	25-07-2022	Preparation of Report
Day 36	26-07-2022	Preparation of Report, making changes suggested by the project mentor
Day 37	27-07-2022	Preparation for the final review, creation of presentation and finalising report
Day 38	28-07-2022	Final Review

16. REFERENCES

- ***"Traffic Detection from Twitter using Spark"* –** Sivagurunathan (https://github.com/SivagurunathanV/Traffic-Detection-from-Twitter-using-Spark)

- ***"From Streaming Data to Twitter Analysis: Using Spark and AWS Kinesis"* –** Zhong Hongsheng (https://ieeexplore.ieee.org/document/7057672)

- ***"Real-Time Detection of Traffic from Twitter Stream Analysis"* –** Eleonora D'Andrea, Pietro Ducange, Beatrice Lazzerini, Francesco Marcellon (https://ieeexplore.ieee.org/document/7057672)

- ***"Detection Traffic Congestion Based on Twitter Data using Machine Learning"* –** Muhammed Taufiq Zulfiqar, Suharto (https://www.sciencedirect.com/science/article/pii/S187705091931066X)

- ***"DataScrapping using Twitter"* –** Kaushal Bundel (https://colab.research.google.com/github/kaushalbundel/blog/blob/master/_notebooks/2020-12-03-Twitter_Data_Scrapping.ipynb)

- ***"Traffic Detection from Real Time Twitter Stream Analysis and Navigation System"* -** Kavita Sawant, Shital Pawar, Miss. Poonam Jadhav, Sayali Vidhate, Nirasha Bule, Snehal Patil

  (https://ijesc.org/upload/331793030f1f54872bf54cb97a160214.Traffic%20Detection%20from%20Real%20Time%20Twitter%20Stream%20Analysis%20and%20Navigation%20System.pdf)

- ***"ROC Curve"* –** Chanin Nantasenamat (https://github.com/dataprofessor/code/blob/master/python/ROC_curve.ipynb)

- ***"Detection, Classification and Location Identification of Traffic Congestion from Twitter Stream"* –** Pouza Rezai (https://github.com/pouyarz/DETECTION-CLASSIFICATION-LOCATION-IDENTIFICATION-OF-TRAFFIC-CONGESTION-FROM-TWITTER-STREAM-ANALYSIS)

- ***"Covid 19 - Italy"* –** Tommaso Bonomo (https://github.com/tommasobonomo/covid19-italy)

APPENDIX

Streaming tweets from twitter

```
import tweepy, configparser
import pandas as pd

# AUTHENTICATION:
config = configparser.ConfigParser()
config.read('config.ini')
consumer_key = config['twitter']['api_key']
consumer_secret = config['twitter']['api_key_secret']
access_token = config['twitter']['access_token']
access_secret = config['twitter']['access_token_secret']
auth = tweepy.OAuthHandler(consumer_key, consumer_secret)
auth.set_access_token(access_token, access_secret)
api = tweepy.API(auth)

# SEARCHING FOR TWEETS:
query1 = 'nyc traffic -filter:retweets'
tweets_array = []
tweets1 = tweepy.Cursor(api.search_tweets, q=query1, lang='en', tweet_mode='extended',
result_type='recent').items(500)
for tweet in tweets1:
    tweets_array.append([tweet.user.screen_name, tweet.user.id, str(tweet.created_at),
tweet.user.location, tweet.full_text])

# EXPORTING TWEETS AS A CSV:
df = pd.DataFrame(tweets_array, columns=['UserName', 'UserID', 'TimeStamp', 'Location',
'Content / Tweet'])
df.to_csv("traffic_tweets.csv")
```

Pre-Processing Techniques and Lemmatizer

```
from nltk.stem import WordNetLemmatizer
import re
import nltk

def remove_emojis(data):
    emoj = re.compile("["
                  u"\U0001F600-\U0001F64F" u"\U0001F300-\U0001F5FF"
                  u"\U0001F680-\U0001F6FF" u"\U0001F1E0-\U0001F1FF"
                  u"\U00002500-\U00002BEF" u"\U00002702-\U000027B0"
                  u"\U00002702-\U000027B0" u"\U000024C2-\U0001F251"
                  u"\U0001f926-\U0001f937" u"\U00010000-\U0010ffff"
                  u"\u2640-\u2642"  u"\u2600-\u2B55" "\u200d" u"\u23cf"
                  u"\u23e9" u"\u231a" u"\ufe0f" u"\u3030"
                  "]+", re.UNICODE)
    return re.sub(emoj, '', data)
```

```
def preprocess_tweets(tweet):
    tweet = re.sub(r'https?://[^ ]+', '', tweet)
    tweet = re.sub(r'@[^ ]+', '', tweet)
    tweet = re.sub(r'0 ', 'zero', tweet)
    tweet = re.sub(r'[^A-Za-z ]', '', tweet)
    tweet = tweet.lower()
    tweet = remove_emojis(tweet)
    return tweet

def lemmatization(x):
    punctuations = '.?:!,;'
    words = nltk.word_tokenize(x)
    new_text=""
    for word in words:
        if word in punctuations:
            words.remove(word)
    word_net_lemmatizer = WordNetLemmatizer()
    for word in words:
        lem_word = word_net_lemmatizer.lemmatize(word,"v")
        new_text = new_text + lem_word+" "
    new_text.strip()
    return new_text
```

Vectorization and Model Building

Reading the Training Dataset:

```
samp_df = pandas.read_csv(".\\pages\\trainingData.csv")
samp_df['Classification'] = samp_df['class'].apply(pp.convert)
samp_df['processed_text'] = samp_df['text'].apply(pp.lemmatization)
samp_df['processed_text'].apply(pp.preprocess_tweets)
```

Splitting of the Dataset:

```
x_train, x_test, y_train, y_test = train_test_split(samp_df.processed_text,
samp_df.Classification, test_size=split_size, random_state=101)
```

Vectorization:

```
vectorizer = CountVectorizer()
tfidf = TfidfVectorizer()

def count_vectorizer(x_train, x_test):
    x_train  = vectorizer.fit_transform(x_train)
    x_test = vectorizer.transform(x_test)
    return x_train, x_test

def tfid_vectorizer(X_train_tfidf, X_test_tfidf):
    X_train_tfidf = tfidf.fit_transform(X_train_tfidf)
    X_test_tfidf = tfidf.transform(X_test_tfidf)
    return X_train_tfidf, X_test_tfidf
```

Training the model:

```
rf = RandomForestClassifier()
rf.fit(x_train, y_train)
rf_train_score = rf.score(x_train,y_train)
rf_test_score = rf.score(x_test,y_test)

nb = MultinomialNB()
nb.fit(x_train, y_train)
nb_train_score = nb.score(x_train,y_train)
nb_test_score = nb.score(x_test,y_test)

logreg = LogisticRegression()
logreg.fit(x_train, y_train)
logreg_train_score = logreg.score(x_train,y_train)
logreg_test_score = logreg.score(x_test,y_test)
```

Predicting the class:

```
predictions1 = logreg.predict(x_test)
logreg_acc_score = accuracy_score(y_test, predictions1)
predictions2 = nb.predict(x_test)
nb_acc_score = accuracy_score(y_test, predictions2)
predictions3 = rf.predict(x_test)
rf_acc_score = accuracy_score(y_test, predictions3)

raw_tweets = df['tweet']
raw_tweets = raw_tweets.apply(pp.preprocess_tweets)
raw_tweets = raw_tweets.apply(pp.lemmatization)

bow = vectorizer.transform(raw_tweets)

df['predicted_class'] = predictions
new_df = df[df['predicted_class'] == 1]
new_df = new_df.drop('predicted_class', axis=1)
new_df = new_df.reset_index(drop=True)
```

ROC Plot and AUROC

```
r_probs = [0 for _ in range(len(y_test))]
rf_probs = rf.predict_proba(x_test)
nb_probs = nb.predict_proba(x_test)
logreg_probs = logreg.predict_proba(x_test)
rf_probs = rf_probs[:, 1]
nb_probs = nb_probs[:, 1]
```

```
logreg_probs = logreg_probs[:, 1]

r_auc = roc_auc_score(y_test, r_probs)
rf_auc = roc_auc_score(y_test, rf_probs)
nb_auc = roc_auc_score(y_test, nb_probs)
logreg_auc = roc_auc_score(y_test, logreg_probs)

r_fpr, r_tpr, _ = roc_curve(y_test, r_probs)
rf_fpr, rf_tpr, _ = roc_curve(y_test, rf_probs)
nb_fpr, nb_tpr, _ = roc_curve(y_test, nb_probs)
logreg_fpr, logreg_tpr, _ = roc_curve(y_test, logreg_probs)


plt.figure(figsize = (15,8))

plt.plot(r_fpr, r_tpr, linestyle='--', label='Random prediction (AUROC = %0.3f)' % r_auc)
plt.plot(rf_fpr, rf_tpr, marker='.', label='Random Forest (AUROC = %0.3f)' % rf_auc)
plt.plot(nb_fpr, nb_tpr, marker='.', label='Naive Bayes (AUROC = %0.3f)' % nb_auc)
plt.plot(logreg_fpr, logreg_tpr, marker='.', label='Logistic Regression (AUROC = %0.3f)' %
logreg_auc)

plt.title('ROC Plot')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.legend()
plt.show()
```