GMR Institute of Technology
An Autonomous Institute Affiliated to JNTUK, Kakinada

GMRIT
Training Tomorrow's Engineers Today

AUTOMATED CLASSIFICATION OF TWEETS INTO SMART CITIES DIMENSIONS USING MACHINE LEARNING ALGORITHMS

A project report submitted in partial fulfilment of the requirement

for the award of degree of

BACHELOR OF TECHNOLOGY

In

COMPUTER SCIENCE AND ENGINEERING

Submitted by

P V N MAHESH
(17341A05E5)

S CHANDRA SEKHAR
(17341A05F5)

M ROHINI
(16341A05B4)

A SUNIL KUMAR
(16341A0516)

Under the esteemed guidance of

Mrs. N. Lakshmi Devi

Designation, Dept. of CSE

GMR Institute of Technology

An Autonomous Institute Affiliated to JNTUK, Kakinada

(Accredited by NBA, NAAC with 'A' Grade & ISO 9001:2008 Certified Institution)

GMR Nagar, Rajam – 532127,
Andhra Pradesh, India

Aug 2021

GMR Institute of Technology
An Autonomous Institute Affiliated to JNTUK, Kakinada

GMRIT
Training Tomorrow's Engineers Today

Department of Computer Science and Engineering

CERTIFICATE

This is to certify that the thesis entitled **AUTOMATED CLASSIFICATION OF TWEETS INTO SMART CITIES DIMENSIONS USING MACHINE LEARNING ALGORITHMS** submitted by **P.V.N.MAHESH(17341A05E5), S.CHANDRA SEKHAR(17341A05F5), M.ROHINI(16341A05B4), A.SUNIL KUMAR(16341A0516)** has been carried out in partial fulfilment of the requirement for the award of degree of **Bachelor of Technology** in **Computer Science and Engineering** of **GMRIT, Rajam** affiliated to **JNTUK, KAKINADA** is a record of bonafide work carried out by them under my guidance & supervision. The results embodied in this report have not been submitted to any other University or Institute for the award of any degree.

Signature of Supervisor
Mrs. N. Lakshmi Devi
Assistant Professor
Department of CSE
GMRIT, Rajam.

Signature of HOD
Dr. A. V. Ramana
Professor & Head
Department of CSE
GMRIT, Rajam.

The report is submitted for the viva-voce examination held on ………………..

Signature of Internal Examiner

Signature of External Examiner

ACKNOWLEDGEMENT

It gives us an immense pleasure to express deep sense of gratitude to my guide **Mrs. N. Lakshmi Devi,** Assistant Professor, Department of Computer Science and Engineering for his whole hearted and invaluable guidance throughout the project work. Without his sustained and sincere effort, this project work would not have taken this shape. He encouraged and helped us to overcome various difficulties that we have faced at various stages of our project work.

We would like to sincerely thank our Head of the department **Dr. A. V. Ramana**, for providing all the necessary facilities that led to the successful completion of our project work.

We would like to take this opportunity to thank our beloved Principal **Dr.C.L.V.R.S.V.Prasad**, for providing all the necessary facilities and a great support to us in completing the project work.

We would like to thank all the faculty members and the non-teaching staff of the Department of Electronics and Communication Engineering for their direct or indirect support for helping us in completion of this project work.

Finally, we would like to thank all of our friends and family members for their continuous help and encouragement.

P.V.N.MAHESH(17341A05E5)
S.CHANDRA SEKHAR(17341A05F5)
M.ROHINI(16341A05B4)
A.SUNIL KUMAR(16341A0516)

ABSTRACT

Smart cities work mutually with data and technology for efficient results and make better decisions to improve the standard of life . Real-time data give the unfold picture from the citizen’s perspective , understand the area of concern, and respond faster. Various city models are proposed with various dimensions and indicators to follow for the evolution of a city into a smart city. Now a days citizens are rising complaints and problems about the services by interacting directly with the respective social network profiles(water, transportation, energy etc) of the government entities using social networks as a gateway. In this paper we applied machine learning algorithms over the preprocessed data collected from Twitter to create classifiers that categorize citizens messages into smart cities services dimension. The classifiers generated here can be integrated into various city services and systems like Government Support decision systems, customer complaint systems, police offices , transportation companies, environmental agencies. The extracted tweets are vectorized using countVectorizer and TF_IDF vectorizer. The various supervised machine learning algorithms used in this project are Random Forest, Decision Tree Classifier, K Nearest Neighbor Classifier. Supervised machine learning models are trained and tested using the vectorizers.

Keywords:

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS & ABBREVIATIONS

ISO	International organization for standardization
SML	Supervised Machine Learning
LR	Logistic Regression
RF	Random Forest
LSVC	Linear Support Vector Classifier
SVC	Support Vector Classifier
MNF	Maximum Number of Features
TF	Term Frequency
IDF	Inverse Document Frequency

INTRODUCTION

1.1 INTRODUCTION

People are using social media networks to raise complaints in their locality on issues related to different kinds of problems like no water supply, electricity, telecommunication, environment, economy, solid wastes, education, fire emergency, sanitation, health emergency, transport, criminal cases, etc. Hence in this project, all the complaints posted on Twitter are examined to classify the new tweets into these dimensions which helps an organization dealing with that particular issue to get to know about the issue and resolve it at the earliest. This model also helps to get to know about the opinion of the citizens and also to monitor and manage various government bodies in an effective way. In this project, the data was extracted using different keywords, the extracted data were filtered by removing the hyperlinks, emoticons, converting the texts into the lower case for easy processing, the extracted data was labeled according to the dimensions mentioned above. The data extracted is the complaints made by the citizens of a particular city, which need not be the same on all the above-mentioned labels hence the dataset is biased.

1.2 OBJECTIVE

Smart City models have been proposed looking for standard metrics to follow cities development and compare their performance into the various domains. ISO 37120 establishes a set of themes/dimensions related to city services and quality of life, and each theme has a set of indicators. Some examples can be verified on Fig. 1. In this work each dimension of ISO corresponds to a topic in the classification approach that will be discussed in next sessions The International Organization for Standardization (ISO) developed this standard to provide a recommendation of what to measure and how [21]. The main objective of ISO with this model is to help cities to measure the management of municipal services performance and quality of life over time, facilitate the learning of one city with the other, allowing comparison in a wide range of measures and share best practices.

1.3 PROPOSED WORK

There is a good volume of published research applying OSN data to cities context. Regarding this paper scope and purposes it is important to understand:

(a) how social media has been used to deal with urban issues

(b) if there are cases integrating OSN data into city services

(c) classifying OSN data into Smart City indicators models.

In the three scenarios the main interest is for the kind of classification performed in those researches, looking specially for those approaches using machine learning techniques.

LITERATURE SURVEY

1. Wang, Q., Bhandal, J., Huang, S., &Luo, B.(2017), IEEE explore "***classification of private tweets using tweet content***".

   This project deals with the detailed information on the four different levels of text classification and the optimization of machine learning classifiers for text analysis prediction. One is Naive Bayes, which needs small dataset for training on text classification and is quite fast in learning. Others are tree based classifiers which are J48, BF Tree and One R. When compared to other classifiers J48 can work with larger training datasets. Here both unigrams and bigrams are considered with J48 algorithm.

2. Moschen, S. A, Macke , J., Bebber, S., &da Silva, M.B.C (2019). "***Sustainable development of communities***".

   In this project, calculated text score at which opinion lexicons are considered, proposed combination of dictionary based technique with Machine Learning approach, worked on six different products reviews taken from amazon and used SVM and concluded that machine learning techniques gives the best results on product reviews, considered performance metrics here are accuracy, precision and F1 score.

3. De Oliveira, M., de Souza Baptista, C., Campelo, C. E., & Bertolotto(2017) "***A gold-standard social media corpus for urban issues. In proceedings of the Symposium on Applied computing***"

   This project is focused on the effect of gender and age in text classification, because it can make process easy for retailers of e-commerce to marketing their goods based on different age groups. Demographics play a prominent part while determining the retailing techniques for various goods. BOW, Word2Vec feature extraction techniques was used. Dictionary based technique was also discussed. For determining text accuracy Naive Bayes, Maximum Entropy, SVM and Long Short Term Memory algorithms are used. Finally showed the experimental results of Machine Learning approaches and their result analysis on a dataset created based on a questionnaire. Here, among gender, female data got good accuracy and among different age groups, age group above 50 got predictions more correctly.

SYSTEM ANALYSIS

3.1 FUNCTIONAL REQUIREMENTS

The functional requirement of the system defines a function of software system or its components. A function is described as set of inputs, behavior of a system and output.

- Fast and efficient
- Simple Computation

3.2 NON FUNTIONAL REQUIREMENTS

In systems engineering and requirements engineering, a non-functional requirement (NFR) is a requirement that specifies criteria that can be used to judge the operation of a system, rather than specific behaviors. They are contrasted with functional requirements that define specific behavior or functions.

3.3 SOFTWARE REQUIREMENTS

- IDE : Jupyter notebook. (Google colab)
- Programming Language : Python

3.4 HARDWARE REQUIREMENTS

- RAM : 4 GB

METHODOLOGY

4.1 MODULES AND LIBRARIES

Matplotlib : Matplotlib is one of the most popular Python packages used for data visualization. It is a cross-platform library for making 2D plots from data in arrays. It provides an objectoriented API that helps in embedding plots in applications using Python GUI toolkits such as PyQt, WxPythonotTkinter. It can be used in Python and IPython shells, Jupyter notebook and web application servers also.

Pyplot : Matplotlib.pyplot is a collection of functions that make matplotlib work like MATLAB. Each pyplot function makes some change to a figure: e.g., creates a figure, creates a plotting area in a figure, plots some lines in a plotting area, decorates the plot with labels, etc.

Numpy : Numpy is a python library used for working with arrays. It also has functions for working in domain of linear algebra,fourier transform and matrices.Numpy Stands for numerical Python. In python there are lists to use but they are slow to process where as numpy provides an object of array which is almost 50 times faster than the traditional python lists because numpy arrays are stored at one continuous place in memory unlike lists,so processes can access and manipulate them very efficiently.

Import os : The OS module in Python provides a way of using operating system dependent functionality.The functions that the OS module provides allows you to interface with the underlying operating system that Python is running on – be that Windows, Mac or Linux.You can find important information about your location or about the process. In this post I will show some of these functions.

Import random : The random module is a built-in module to generate the pseudo-random variables. It can be used perform some action randomly such as to get a random number, selecting a random elements from a list, shuffle elements randomly, etc.

4.2 FLOWCHART

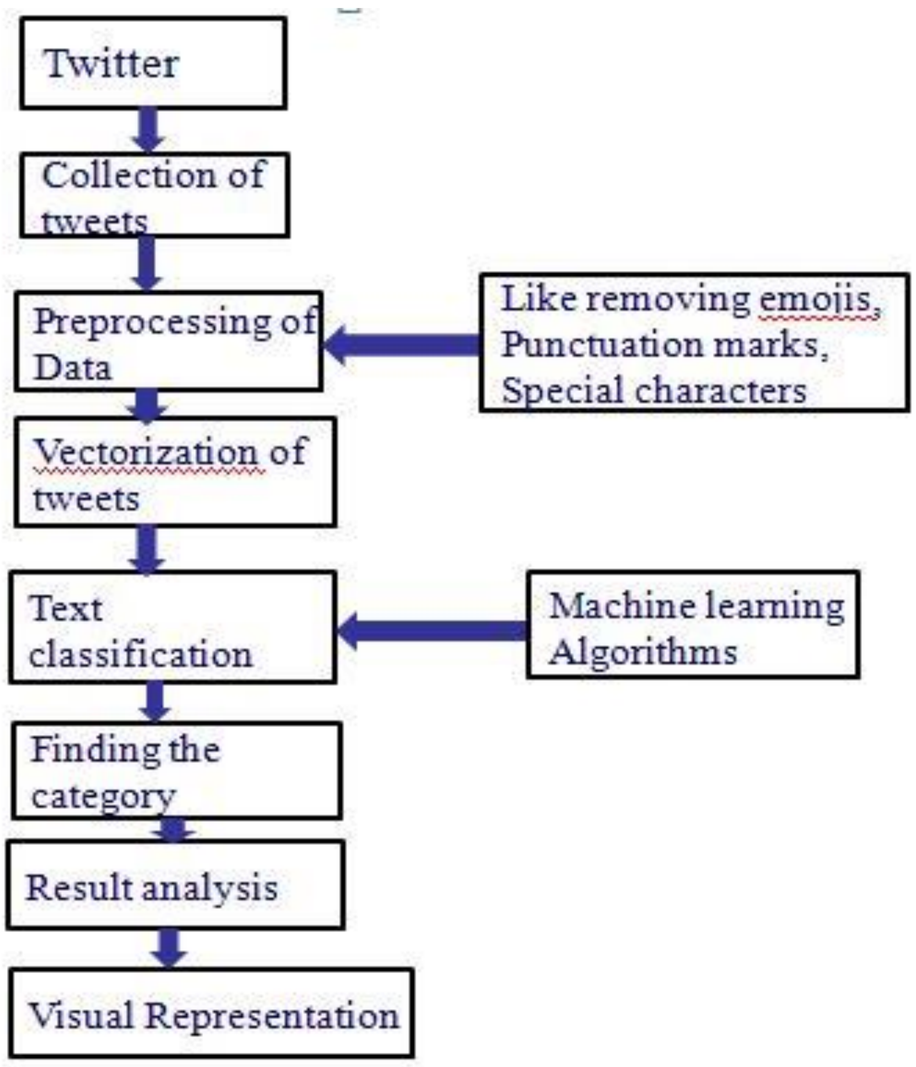


Fig 4.2.1 Flowchart

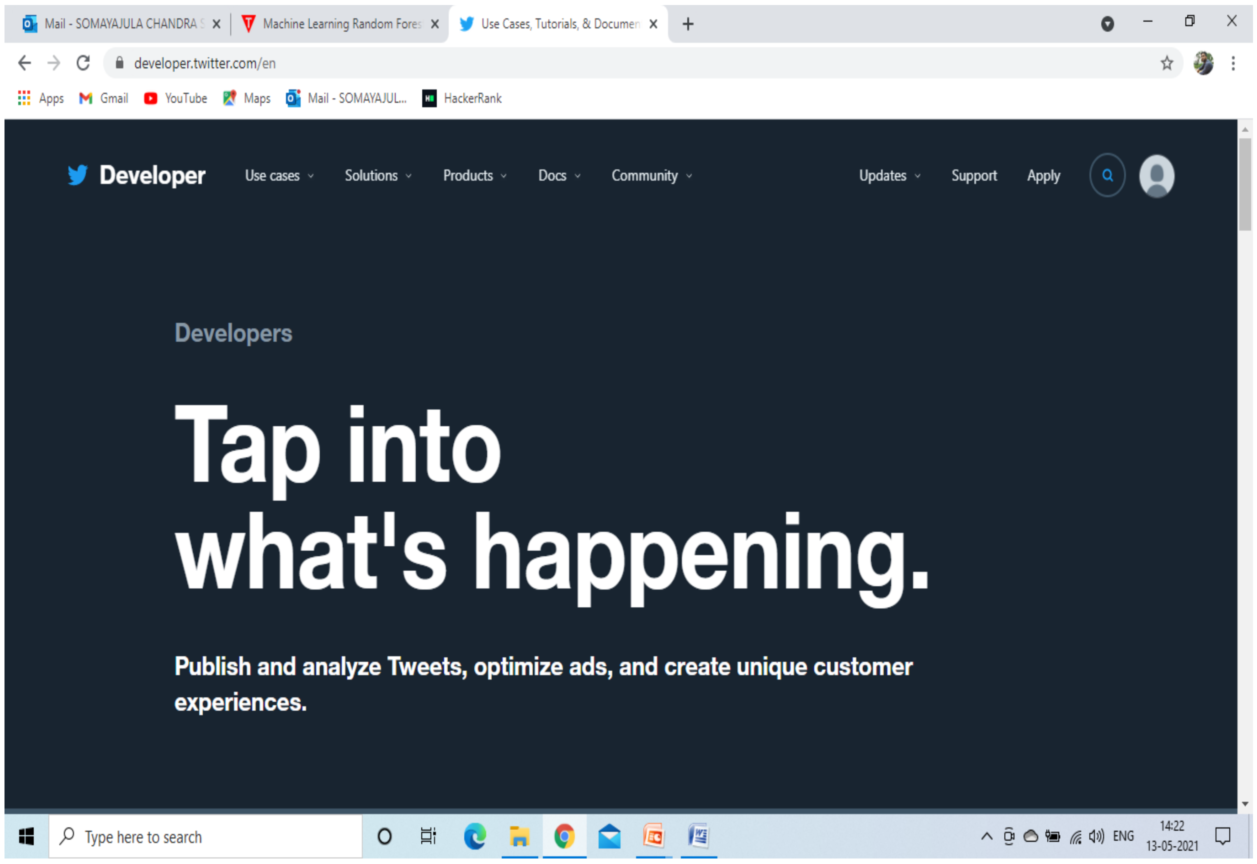
4.3 DATA SET DESCRIPTION

We have collected data from twitter by using API key. So, in order to retrieve data from twitter using API keys we need a developer account in twitter.

Here are the following steps to create a developer account

1) Open Chrome browser and login to your twitter account using the given URL ***developer.twitter.com***

2) Then you find a screen like this. Then click on Apply



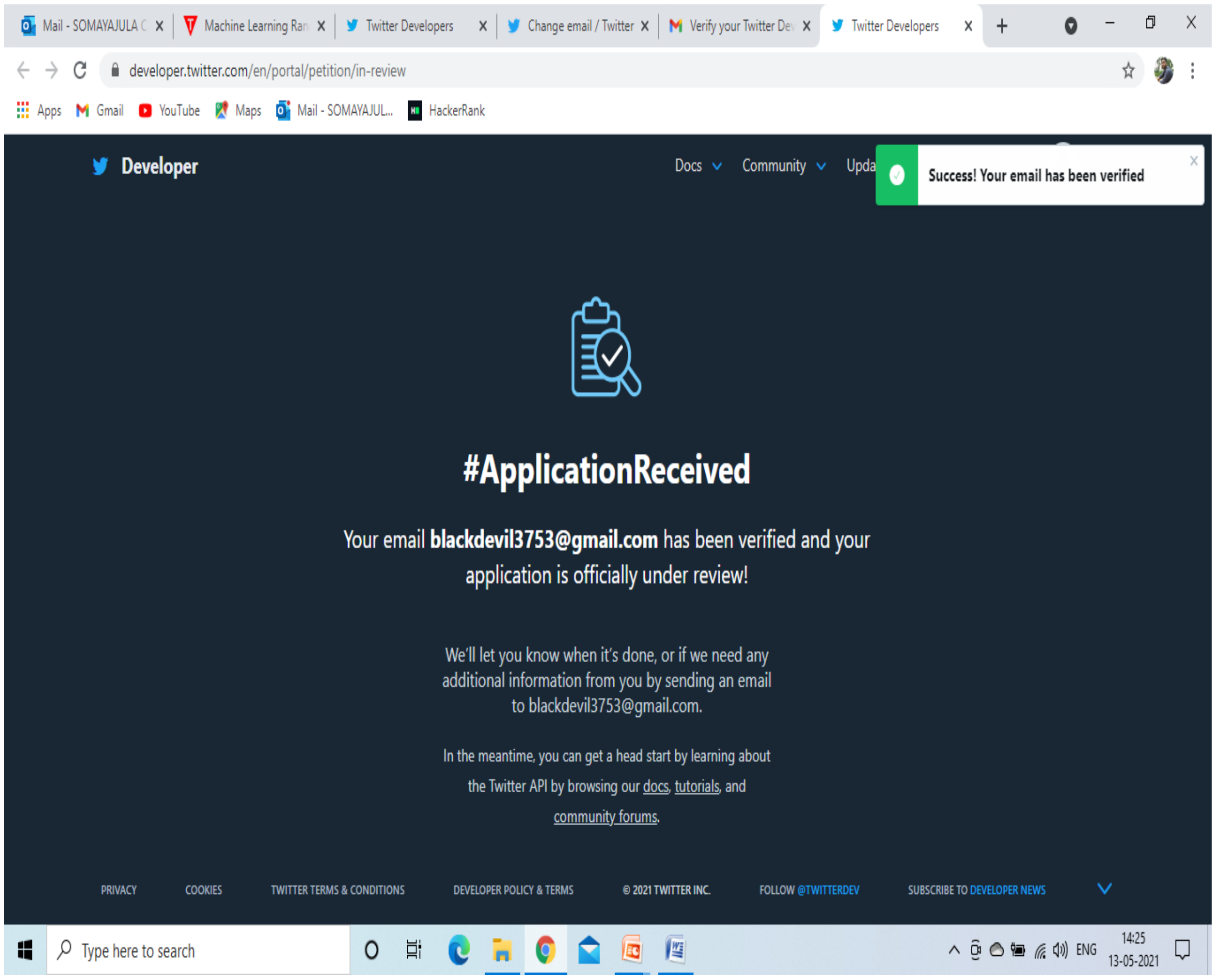
Developers

Tap into what's happening.

Publish and analyze Tweets, optimize ads, and create unique customer experiences.

3)Give all the credentials which are required to create a developer account in Twitter.

4) After completing all the steps , you will get a screen like this.



Success! Your email has been verified

#ApplicationReceived

Your email **blackdevil3753@gmail.com** has been verified and your application is officially under review!

We'll let you know when it's done, or if we need any additional information from you by sending an email to blackdevil3753@gmail.com.

In the meantime, you can get a head start by learning about the Twitter API by browsing our docs, tutorials, and community forums.

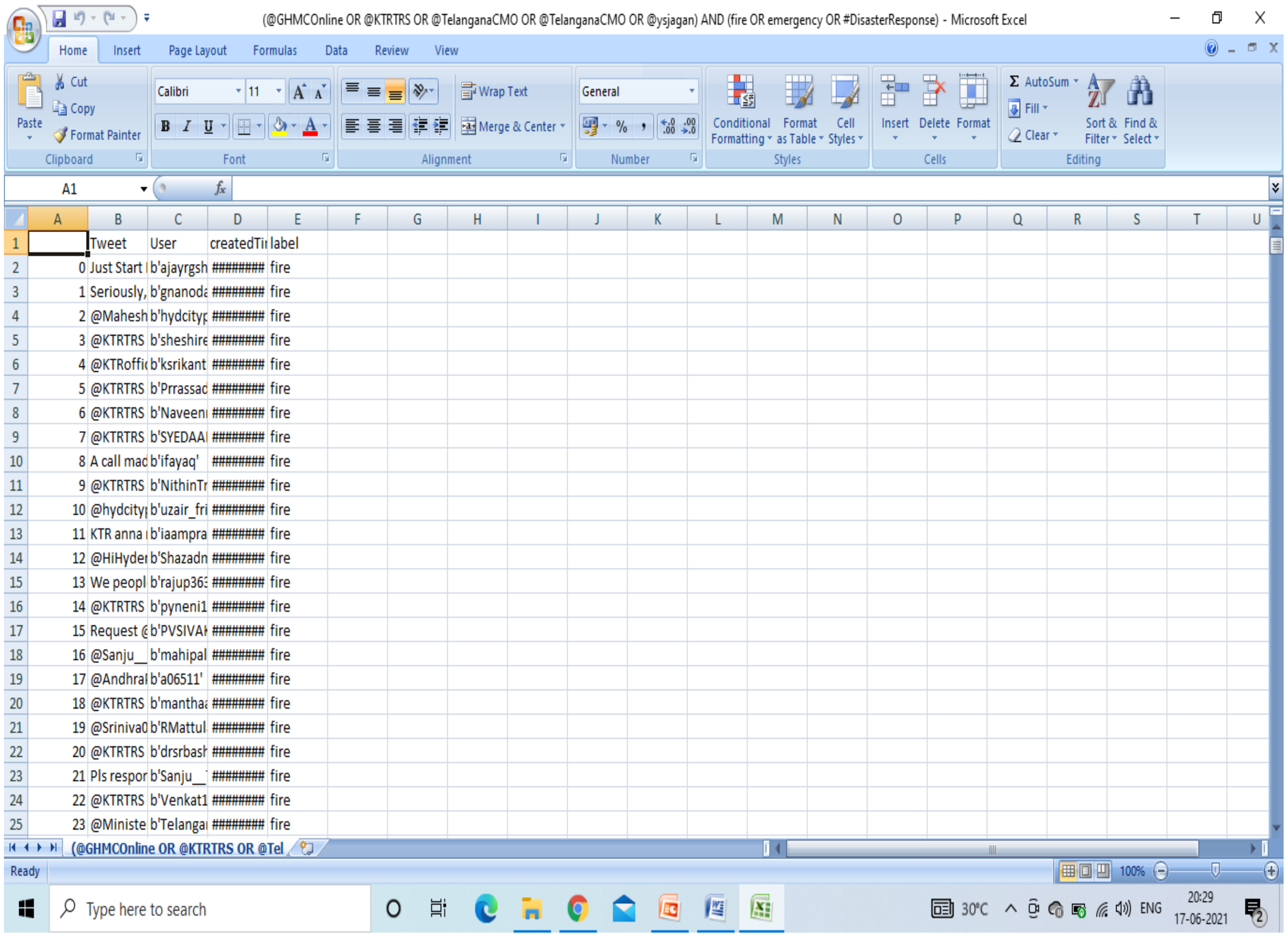
Data Extraction :

```
def Extract_data(label,qu,n):
  df = pd.DataFrame()
  q = qu + ' -filter:retweets'
  for status in tweepy.Cursor(api.search, q=q, lang='en',
tweet_mode='extended').items(n):
      df = df.append({'createdTime' : status.created_at, 'Tweet' :
status.full_text.replace('\n',' '), 'User': status.user.screen_name.encode('utf-8'),
'label': label}, ignore_index=True)
  fname = qu + '.csv'
  df.to_csv(fname)
  return df
```

Different keys of Twitter:

```
consumer_key=“jEEWinIsUKeoiUH0a4I8MKkUp”
consumer_secret=“2eDqARYiYaPn7ewcAOs1oKHPOVyrOaSwuGhOSWU96ZSvOyGBpi”
access_token=“1394502294490607617i6pyYLrmZsoQtM17BD78uFKmd6jZyX”
access_token_secret=“xYdmZh7nkXqlDCln3fHCVEYUlTwI8TP7BcTkE8p0WUAwI”
```

Sample dataset:

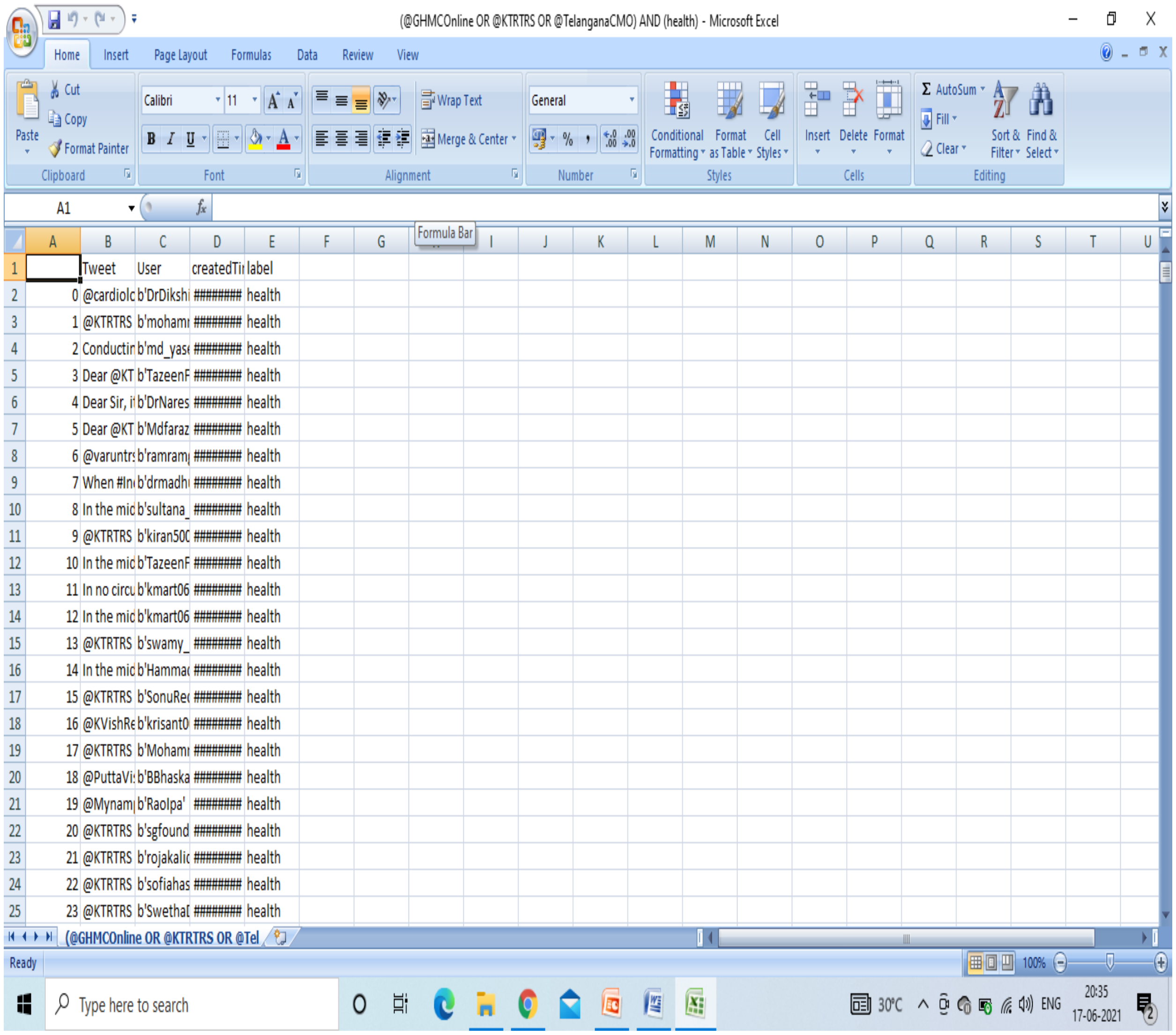


(@GHMCOnline OR @KTRTRS OR @TelanganaCMO OR @TelanganaCMO OR @ysjagan) AND (fire OR emergency OR #DisasterResponse) - Microsoft Excel

	Tweet	User	createdTi	label
0	Just Start	b'ajayrgsh	########	fire
1	Seriously,	b'gnanoda	########	fire
2	@Mahesh	b'hydcityp	########	fire
3	@KTRTRS	b'sheshire	########	fire
4	@KTRoffi	b'ksrikant	########	fire
5	@KTRTRS	b'Prrassad	########	fire
6	@KTRTRS	b'Naveen	########	fire
7	@KTRTRS	b'SYEDAA	########	fire
8	A call mad	b'ifayaq'	########	fire
9	@KTRTRS	b'NithinTr	########	fire
10	@hydcity	b'uzair_fri	########	fire
11	KTR anna	b'iaampra	########	fire
12	@HiHyder	b'Shazadn	########	fire
13	We peopl	b'rajup363	########	fire
14	@KTRTRS	b'pyneni1	########	fire
15	Request @	b'PVSIVAK	########	fire
16	@Sanju__	b'mahipal	########	fire
17	@Andhral	b'a06511'	########	fire
18	@KTRTRS	b'manthaa	########	fire
19	@Sriniva0	b'RMattul	########	fire
20	@KTRTRS	b'drsrbash	########	fire
21	Pls respor	b'Sanju__	########	fire
22	@KTRTRS	b'Venkat1	########	fire
23	@Ministe	b'Telanga	########	fire

(@GHMCOnline OR @KTRTRS OR @Tel

Fig 4.3.1 Fire department dataset


(@GHMCOnline OR @KTRTRS OR @TelanganaCMO) AND (health) - Microsoft Excel

	Tweet	User	createdTi	label
0	@cardiolc	b'DrDikshi	########	health
1	@KTRTRS	b'mohamı	########	health
2	Conductin	b'md_yas	########	health
3	Dear @KT	b'TazeenF	########	health
4	Dear Sir, i	b'DrNares	########	health
5	Dear @KT	b'Mdfaraz	########	health
6	@varuntrs	b'ramram	########	health
7	When #In	b'drmadh	########	health
8	In the mic	b'sultana_	########	health
9	@KTRTRS	b'kiran50C	########	health
10	In the mic	b'TazeenF	########	health
11	In no circu	b'kmart06	########	health
12	In the mic	b'kmart06	########	health
13	@KTRTRS	b'swamy_	########	health
14	In the mic	b'Hammac	########	health
15	@KTRTRS	b'SonuRe	########	health
16	@KVishRe	b'krisant0	########	health
17	@KTRTRS	b'Mohamı	########	health
18	@PuttaVi:	b'BBhaska	########	health
19	@Mynam	b'RaoIpa'	########	health
20	@KTRTRS	b'sgfound	########	health
21	@KTRTRS	b'rojakalic	########	health
22	@KTRTRS	b'sofiahas	########	health
23	@KTRTRS	b'SwethaI	########	health

(@GHMCOnline OR @KTRTRS OR @Tel

Fig 4.3.2 Health department dataset

4.4 PROPOSED MODELS AND METHODS

Here, we have used 8 supervised learning models(SLM) to classify our data. They are

- Decision Tree classifier
- Random Forest classifier
- K Nearest Neighbor classifier
- Compliment Naïve Bayes
- Logistic regression
- Multinomial regression
- Support Vector classifier
- Linear SVC

Decision Tree classifier:

- Decision Tree is a supervised learning technique that can be used for both classification and regression problems but mostly it is preferred for solving classification problems.
- It is tree-structured classifier, where internal nodes represent the features of a dataset, branches represent the decision rules and each leaf node represents the outcome.
- In a decision tree, for prediction the class of the given dataset the algorithm starts from the root node of the tree. This algorithm compares the values of

root attribute with the record attribute and based on the comparison, follows the branch and jumps to the next node.

- For the next node, the algorithm again compares the attribute value with the other sub-nodes and move further. It continues the process until it reaches the leaf node of the tree.
- Decision tree usually mimic human thinking ability while making a decision, so it is easy to understand.

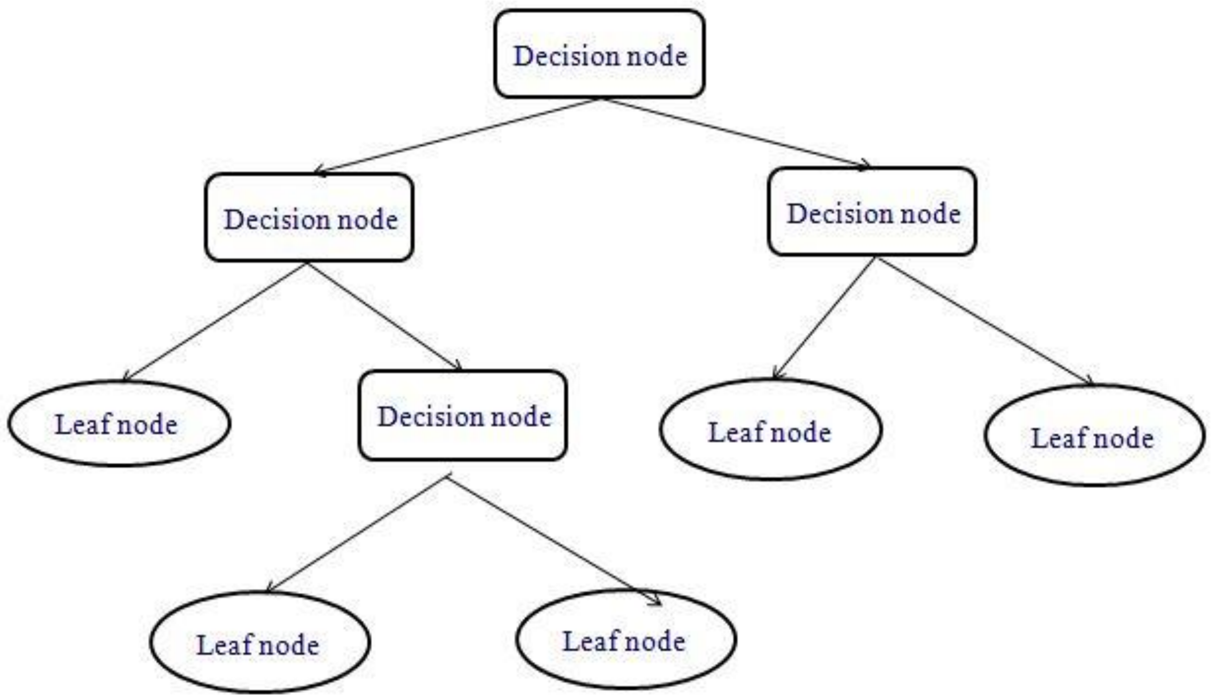


Fig 4.4.1 Decision tree classifier

Random Forest Classifier:

- Random Forest is a classifier that contains a number of decision tress on various subsets of the given dataset and takes the average to improve the predictive accuracy of that dataset.
- Instead of relying on one decision tree, the random forest takes the prediction from each tree and based on the majority votes of predictions, and it predicts the final output.
- 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.
- Random Forest combines multiple trees to predict the class of the data set it is possible that some decision trees may predict the correct output while others may not but together all the trees predict the correct output .
- It takes less training time as compared to other algorithms .
- Although random forest can be used for both classification and regression tasks it is not more suitable for regression tasks.

K- Nearest Neighbor Classifier :

- K-NN algorithm assumes the similarity between new case and available cases and put the new case in the category that is most similar to the available categories .
- K-NN algorithm stores all the available data and classifies a new data point based on the similarity .This means when new data appears than it can be easily classified into a well suite category by using KNN algorithm .
- It is also called a lazy learner algorithm because it does not learn from the training set immediately instead it stores the data set and at the time of classification it performs an action on the dataset .
- Firstly , we will choose the number of neighbors next we will calculate the Euclidean distance between the data points .By calculating the Euclidean distance we will get the nearest neighbors by finding nearest neighbors new data point will form .

Complement Naïve Bayes:

- Complement Naïve Bayes is somewhat an adaptation of the multinomial Naive bayes algorithm.
- Multinomial Naïve Bayes does not perform well on imbalanced datasets.

- Imbalanced datasets are datasets where the number of examples of some classes is higher than the number of examples belonging to their classes. This means that the Distribution of examples is not uniform.

Logistic Regression:

- The logistic model (or logit model) is used to model the probability of a certain class or event existing such as pass/fail, win/lose, alive/dead or healthy/sick.
- This can be extended to model several classes of events such as determining whether an image contains a cat, dog, lion, etc.
- Each object being detected in the image would be assigned a probability between 0 and 1, with a sum of one.

Multinomial Naïve Bayes :

- The multinomial Naive Bayes classifier is suitable for classification with discrete features
- The multinomial distribution normally requires integer feature counts. However, in practice, fractional counts such as tf-idf may also work.
- The algorithm is based on the Bayes Theorem and predicts the tag of a text such as piece of email or newspaper article.

Support vector Classifier:

- Support vector classifier is to fit to the data you provide, returning a “best fit” hyperplane that divides or categorize our data.
- The fit time scales at least quadratically with the number of samples and may be impractical beyond tens of thousands of samples.
- The multiclass support is handled according to a one-vs-one scheme.

Linear SVC:

- Similar to SVC with parameter kernel = “ linear “.
- but implemented in terms of liblinear rather than libsvm, so it has more flexibility in the choice of penalties and loss functions and should scale better to large numbers of samples.
- This class supports both dense and sparse input and the multiclass support is handled according to a one-vs-the-rest scheme.

PROPOSED METHODS:

Here we have used two vectorizer methods to vectorize the data.

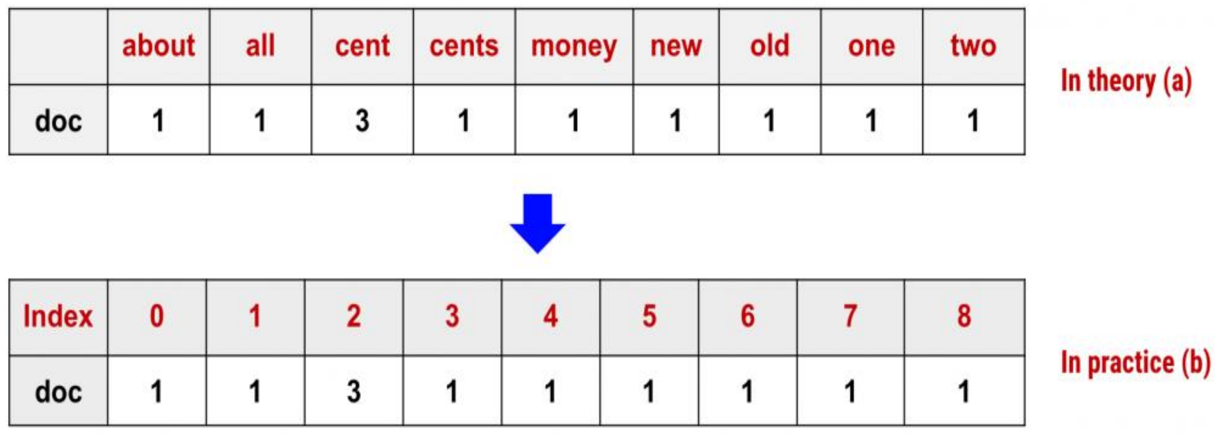
The vectorized methods proposed in this project are

- Count vectorizer
- Tf-Idf vectorizer

Count Vectorizer:

- The count vectorizer assigns a unique value for each unique word in the whole data set and all the unique words are the features.

  doc=["One Cent, Two Cents, Old Cent, New Cent: All About Money"]



	about	all	cent	cents	money	new	old	one	two
doc	1	1	3	1	1	1	1	1	1

In theory (a)

Index	0	1	2	3	4	5	6	7	8
doc	1	1	3	1	1	1	1	1	1

In practice (b)

Fig 4.4.2 Count vectorizer

TF-IDF vectorizer:

- Stands for Term Frequency Inverse Document Frequency.
- In the TF-IDF vectorizer, we calculate the Term Frequency, the Inverse Document Frequency, and the TF-IDF score using the below formulae. Words with a higher TF-IDF score are more important, and those with a lower score are less important.
- Term frequency can be calculated by dividing the no of times a particular word appears in a document to the total no of words in a document.

$$TF_{t,d} = n_{t,d} \text{ / No.of terms in the document}$$

- Inverse document Frequency can be calculated by performing the logarithm of ratio of no of documents to the total no of documents with the term “t” .

$$IDF_t = \log\{\text{No. of documents / No. of documents with term ‘t’}\}$$

- The TFIDF is calculated by performing multiplication of TF and IDF.

$$TF\text{-}IDF_{t,d} = TF_{t,d} * IDF_t$$

IMPLEMENTATION

5.1 STEPS FOR IMPLEMENTATION

1. Load the data from Twitter.
2. Preprocessing of data like removing punctuation marks,emojis and any other hyperlinks.
3. Vectorization of data using count and Tf-idf vectorizer.
4. Text classification using given supervised learning models.
5. Finding the category.
6. Result analysis
7. Visual representation.

5.2 SOURCE CODE

```
import csv

import re

import tweepy

import nltk

import pandas as pd

import numpy as np

import sklearn

import matplotlib.pyplot as plt

import seaborn as sns

from sklearn.feature_extraction.text import TfidfVectorizer

from sklearn.feature_extraction.text import CountVectorizer

from sklearn.feature_selection import chi2

from sklearn.model_selection import train_test_split

from sklearn.naive_bayes import MultinomialNB

from sklearn.linear_model import LogisticRegression

from sklearn.ensemble import RandomForestClassifier

from sklearn.svm import LinearSVC

from sklearn.neighbors import KNeighborsClassifier

from sklearn.naive_bayes import ComplementNB
from sklearn.tree import DecisionTreeClassifier

from sklearn.svm import SVC

from sklearn.model_selection import cross_val_score

import matplotlib.pyplot as plt

from sklearn.metrics import f1_score

from sklearn.metrics import confusion_matrix

from sklearn import metrics

from sklearn.metrics import make_scorer, accuracy_score, precision_score,
recall_score, f1_score

import sklearn.model_selection as model_selection

from sklearn.model_selection import cross_validate
```

#Different keys of Twitter developer account

```
consumer_key = “CONSUMER KEY”

consumer_secret = ‘CONSUMER SECRET”

access_token = “ACCESS TOKEN”

access_token_secret = “ACCESS TOKEN SECRET”


auth = tweepy.OAuthHandler(consumer_key, consumer_secret)
```

```
auth.set_access_token(access_token, access_token_secret)

api = tweepy.API(auth, wait_on_rate_limit = True)
```

#Data Extraction

```
          def Extract_data(label,qu,n):

            df = pd.DataFrame()

            q = qu + ' -filter:retweets'

            for status in tweepy.Cursor(api.search, q=q, lang='en',

     tweet_mode='extended').items(n):

                df = df.append({'createdTime' : status.created_at, 'Tweet' :

     status.full_text.replace('\n',' '), 'User': status.user.screen_name.encode('utf-

     8'), 'label': label}, ignore_index=True)

            fname = qu + '.csv'

            df.to_csv(fname)

            return df


d2 = Extract_data('water', '@GHMCOnline AND water', 1000)

d3 = Extract_data('electricity', '@GHMCOnline AND (electricity OR electric)',

1000)

d4 = Extract_data('electricity', '@TsspdclCorporat AND power', 1000)
```

```
d5 = Extract_data('police', '(@HYDTP OR @TelanganaCOPs)', 1000)
d6 = Extract_data('solid waste', '(@GHMCOnline OR @KTRTRS) AND (garbage
OR waste)', 1000)
d7 = Extract_data('sanitation', '(@GHMCOnline OR @KTRTRS) AND sanitation',
1000)
d8 = Extract_data('education', '(@GHMCOnline OR @KTRTRS) AND education',
1000)
d9 = Extract_data('environment', '(@GHMCOnline OR @KTRTRS) AND
(environment OR trees OR plant OR tree OR pollution)', 1000)
d10 = Extract_data('economy', '(@GHMCOnline OR @KTRTRS OR
@TelanganaCMO OR @trsharish) AND (economy OR finance OR financial OR
gdp)', 1000)
d11 = Extract_data('fire', '(@GHMCOnline OR @KTRTRS OR @TelanganaCMO
OR @TelanganaCMO OR @ysjagan) AND (fire OR emergency OR
#DisasterResponse)', 1000)
d12 = Extract_data('health', '(@GHMCOnline OR @KTRTRS OR
@TelanganaCMO) AND (health)', 1000)
d13 = Extract_data('health', '(@TelanganaHealth OR @Eatala_Rajender) AND
(hospital OR medical OR health OR suffering)', 1000)
```

```
d14 = Extract_data('transport', '(@GHMCOnline OR @KTRTRS) AND (road OR
bus OR transport OR train OR metro)', 1000)
d15 = Extract_data('telecommunication', '(@GHMCOnline OR @KTRTRS) AND
(bsnl OR network OR communications OR 5G OR 4G OR fibre OR fiber OR
broadband )', 1000)

df = pd.concat([d2,d3,d4,d5,d6,d7,d8,d9,d10,d11,d12,d13,d14,d15])
```

#Data Preprocessing

```
from nltk.stem import PorterStemmer
from nltk.tokenize import word_tokenize as wt
nltk.download('stopwords')
nltk.download('punkt')
from nltk.corpus import stopwords

def preProcessing(tweets):
  tweets = tweets.tolist()
  ps = PorterStemmer()
  processedTweets = []
  punct = ['.',',','?','!',';',':','-
','_','(',')','{','}','[',']','&','*','+','=','"',"'",'<','>','|','\\','/','`','~','@','#']
  sWords = set(stopwords.words('english'))
  for tweet in tweets:
    tweet = tweet.encode().decode()
    tweet = tweet.encode('ascii', 'ignore').decode('ascii') #removing the emojis
    tweet = tweet.encode('latin-1', 'ignore').decode('latin-1')
    tweet = re.sub(r'http\S+', '', tweet) #removing the urls
    tweet = tweet.lower() #converting the text into lower cases
    words = wt(tweet) #tokenization
    processedWords = []
    for word in words:
      if word in punct or word in sWords:
        continue
      if words[words.index(word) - 1] != '@' and words[words.index(word) - 1] !=
'#':
        processedWords.append(word)
    processedWords.remove(processedWords[0])
    processedTweets.append(' '.join(processedWords))
  return processedTweets
```

```
df.Tweet = preProcessing(df.Tweet)

df1=df[['Tweet','label']].copy()

df1.columns=['tweet','label']

df1['category_id'] = df1['label'].factorize()[0]

category_id_df = df1[['label', 'category_id']].drop_duplicates()


# Dictionaries for future use

category_to_id = dict(category_id_df.values)

id_to_category = dict(category_id_df[['category_id', 'label']].values)

fig = plt.figure(figsize=(8,6))

colors = ['grey','grey','grey','grey','grey','grey','grey','grey',

    'grey','darkblue','darkblue','darkblue']

df1.groupby('label').tweet.count().sort_values().plot.barh(

    ylim=0, color=colors, title= 'NUMBER OF TWEETS IN EACH

CATEGORY\n')

plt.xlabel('Number of occurrences', fontsize = 10);
```

#CountVectorizer Feature Selection

```
vectorizer = CountVectorizer(ngram_range=(1, 3),
                    stop_words='english')
features = vectorizer.fit_transform(df1.tweet).toarray()
labels = df1.category_id

print("Each of the %d tweets is represented by %d features" %(features.shape))
```

#Training the models

```
X = df1['tweet']
y = df1['label']

X_train, X_test, y_train, y_test = train_test_split(X, y,
                                                    test_size=0.2,
                                                   random_state = 0)
models = [ RandomForestClassifier(n_estimators=100, max_depth=5,
random_state=0),
    LinearSVC(),
    MultinomialNB(),
    LogisticRegression(random_state=0),

    KNeighborsClassifier(n_neighbors=1),

    ComplementNB(),

    DecisionTreeClassifier(random_state=0),

    SVC(),

]

CV = 5

cv_df = pd.DataFrame(index=range(CV * len(models)))


entries = []

score=['f1_macro','f1_micro']

for model in models:

    model_name = model.__class__.__name__

    accuracies = cross_validate(model, features, labels, scoring=score, cv=CV)

    for f1_macro,f1_micro in

zip(accuracies['test_f1_macro'],accuracies['test_f1_micro']):

        entries.append((model_name,f1_macro,f1_micro))

cv_df = pd.DataFrame(entries, columns=['model_name','f1_macro','f1_micro'])

f1_macro = cv_df.groupby('model_name').f1_macro.mean()

f1_micro = cv_df.groupby('model_name').f1_micro.mean()
```

```
acc = pd.concat([f1_macro,f1_micro], axis= 1,
          ignore_index=True)
acc.columns = ['f1-macro_mean','f1-micro_mean']
acc=acc.sort_values(by=['f1-macro_mean'], ascending=False)
print(acc)   #Accuracy scores
```

#Plotting the metrics of 8 algorithms

```
plt.figure(figsize=(20,5))
sns.boxplot(x='model_name', y='f1_macro',
          data=cv_df,
          showmeans=True)
plt.title("f1-macro (cv = 5)\n", size=14);
plt.figure(figsize=(20,5))
sns.boxplot(x='model_name', y='f1_micro',
          data=cv_df,
          showmeans=True)
plt.title("f1-micro (cv = 5)\n", size=14);


X_train, X_test, y_train, y_test,indices_train,indices_test =
train_test_split(features,labels,df1.index,test_size=0.2,random_state=1)
```

```
model = DecisionTreeClassifier(random_state=0)

model.fit(X_train, y_train)

y_pred = model.predict(X_test)

print('\t\t\t\tCLASSIFICATIION METRICS\n')

print(metrics.classification_report(y_test, y_pred,

                                    target_names= df1['label'].unique()))
```

#Minimum Document Frequency method

```
final2=[]

x=1

while(x<=30):

    vectorizer = CountVectorizer(min_df=x,

                    ngram_range=(1, 3),

                    stop_words='english',

                            )

    features = vectorizer.fit_transform(df1.tweet).toarray()

    labels = df1.category_id

    print("\nFor min-df=",x,", each of the %d tweets is represented by %d

features\n" %(features.shape))
```

```
X = df1['tweet'] # Collection of documents
y = df1['label'] # Target or the labels we want to predict (i.e., the 13 different
complaints of products)
X_train, X_test, y_train, y_test = train_test_split(X, y,
                                                    test_size=0.2,
                                                    random_state = 0)
models = [
LinearSVC(max_iter=100000),
LogisticRegression(random_state=0,max_iter=10000),
DecisionTreeClassifier(random_state=0),
]

# 5 Cross-validation
CV = 5
cv_df = pd.DataFrame(index=range(CV * len(models)))

entries = []
score=['f1_macro','f1_micro']
for model in models:
    model_name = model.__class__.__name__
    accuracies = cross_validate(model, features, labels, scoring=score, cv=CV)
    i=0
    for f1_macro,f1_micro in
zip(accuracies['test_f1_macro'],accuracies['test_f1_micro']):
            entries.append((i,model_name,f1_macro,f1_micro))
          i+=1
cv_df = pd.DataFrame(entries,
columns=['index','model_name','f1_macro','f1_micro'])
f1_macro = cv_df.groupby('model_name').f1_macro.mean()
f1_micro = cv_df.groupby('model_name').f1_micro.mean()
demo_acc = pd.concat([f1_macro,f1_micro], axis= 1, ignore_index=True)
demo_acc.columns = ['f1-macro_mean','f1-micro_mean']
print("Scores for min-df=",x," are given below:\n")
print(demo_acc)
l1=demo_acc['f1-macro_mean'].tolist()
l2=demo_acc['f1-micro_mean'].tolist()
l3=[]
l3=[x]+l1+l2
final2.append(l3)
    x+=1
temp=pd.DataFrame(final2,columns=['mnf','DTC_f1-macro','LSVC_f1-
macro','LR_f1-macro','DTC_f1-micro','LSVC_f1-micro','LR_f1-micro'])
temp.to_csv('scores from countvectorizer(mnf).csv')
print(temp)
```

#Plotting the results

```
temp.plot(x='min_df',y=['DTC_f1-macro','LSVC_f1-macro','LR_f1-
macro'],marker='^',figsize=(10,5),ylim=(0.7,0.8));
```

#Maximum number of Features method

```
final1=[]
w=500
while(w<=20000):
    vectorizer = CountVectorizer(max_features=w,
                    ngram_range=(1, 3),
                    stop_words='english',
                              )
    # We transform each complaint into a vector
    demo_features = vectorizer.fit_transform(df1.tweet).toarray()
    labels = df1.category_id
    print("\nFor max-features=",w,", each of the %d tweets is represented by %d
features\n" %(demo_features.shape))

    X = df1['tweet'] # Collection of documents
    y = df1['label'] # Target or the labels we want to predict (i.e., the 13 different
complaints of products)
    X_train, X_test, y_train, y_test = train_test_split(X, y,
                                                test_size=0.2,
                                                random_state = 0)
    models = [
    LinearSVC(max_iter=100000),
    LogisticRegression(random_state=0,max_iter=10000),
    DecisionTreeClassifier(random_state=0),
    ]

    # 5 Cross-validation
    CV = 5
    cv_df = pd.DataFrame(index=range(CV * len(models)))


    entries = []
    score=['f1_macro','f1_micro']
    for model in models:
        model_name = model.__class__.__name__
        accuracies = cross_validate(model, demo_features, labels, scoring=score,
cv=CV)
        i=0
        for f1_macro,f1_micro in
zip(accuracies['test_f1_macro'],accuracies['test_f1_micro']):
            entries.append((i,model_name,f1_macro,f1_micro))
          i+=1
    cv_df = pd.DataFrame(entries,
columns=['index','model_name','f1_macro','f1_micro'])
    f1_macro = cv_df.groupby('model_name').f1_macro.mean()
    f1_micro = cv_df.groupby('model_name').f1_micro.mean()
    demo_acc = pd.concat([f1_macro,f1_micro], axis= 1, ignore_index=True)
    demo_acc.columns = ['f1-macro_mean','f1-micro_mean']
    print("Scores for max features=",w," are given below:\n")
    print(demo_acc)
    l1=demo_acc['f1-macro_mean'].tolist()
    l2=demo_acc['f1-micro_mean'].tolist()
    l3=[]
    l3=[w]+l1+l2
    final1.append(l3)
    w+=500
temp1=pd.DataFrame(final1,columns=['mnf','DTC_f1-macro','LSVC_f1-macro','LR_f1-macro','DTC_f1-micro','LSVC_f1-micro','LR_f1-micro'])
temp1.to_csv('scores from countvectorizer(mnf).csv')
print(temp1)
```

#Plotting the results

```
temp1.plot(x='mnf',y=['DTC_f1-macro','LSVC_f1-macro','LR_f1-macro'],marker='^',figsize=(10,5),ylim=(0.7,0.8));
```

#TF-IDF Feature Selection

```
tfidf = TfidfVectorizer(sublinear_tf=True,
                ngram_range=(1, 3),
                stop_words='english')

features1 = tfidf.fit_transform(df1.tweet).toarray()

labels = df1.category_id

print("Each of the %d tweets is represented by %d features (TF-IDF score of
unigrams and bigrams)" %(features1.shape))
X = df1['tweet']
y = df1['label']

X_train, X_test, y_train, y_test = train_test_split(X, y,
                                                    test_size=0.2,
                                                     random_state = 0)
models = [
   RandomForestClassifier(n_estimators=100, max_depth=5, random_state=0),
   LinearSVC(),
   MultinomialNB(),
   LogisticRegression(random_state=0),
    KNeighborsClassifier(n_neighbors=1),

    ComplementNB(),

    DecisionTreeClassifier(random_state=0),

    SVC(),

]

CV = 5

cv_df = pd.DataFrame(index=range(CV * len(models)))


entries = []

score=['f1_macro','f1_micro']

for model in models:

    model_name = model.__class__.__name__

    accuracies = cross_validate(model, features1, labels, scoring=score, cv=CV)

    for f1_macro,f1_micro in

zip(accuracies['test_f1_macro'],accuracies['test_f1_micro']):

        entries.append((model_name,f1_macro,f1_micro))

cv_df = pd.DataFrame(entries, columns=['model_name','f1_macro','f1_micro'])


f1_macro = cv_df.groupby('model_name').f1_macro.mean()

f1_micro = cv_df.groupby('model_name').f1_micro.mean()
```

```
acc1 = pd.concat([f1_macro,f1_micro], axis= 1,
          ignore_index=True)
acc1.columns = ['f1-macro_mean','f1-micro_mean']
acc1=acc1.sort_values(by=['f1-macro_mean'], ascending=False)
print(acc1)
```

#Plotting the metrics of 8 algorithms

```
plt.figure(figsize=(20,5))
sns.boxplot(x='model_name', y='f1_macro',
          data=cv_df,
          showmeans=True)
plt.title("F1-MACRO MEAN (cv = 5)\n", size=14);
plt.figure(figsize=(20,5))
sns.boxplot(x='model_name', y='f1_micro',
          data=cv_df,
          showmeans=True)
plt.title("F1-MICRO MEAN (cv = 5)\n", size=14);
```

#Minimum number of features method

```
final3=[]

x=1

while(x<=30):

    vectorizer = TfidfVectorizer(sublinear_tf=True,

                            min_df=x,

                              ngram_range=(1, 3),

                              stop_words='english',

                            )


    demo_features1 = vectorizer.fit_transform(df1.tweet).toarray()

    labels = df1.category_id

    print("\nFor min-df=",x,", each of the %d tweets is represented by %d

features\n" %(demo_features1.shape))


    X = df1['tweet']

    y = df1['label']

    X_train, X_test, y_train, y_test = train_test_split(X, y,

                                                    test_size=0.2,

                                                    random_state = 0)
    models = [
    LinearSVC(max_iter=100000),
    LogisticRegression(random_state=0,max_iter=10000),
    DecisionTreeClassifier(random_state=0),
    ]

    # 5 Cross-validation
    CV = 5
    cv_df = pd.DataFrame(index=range(CV * len(models)))

    entries = []
    score=['f1_macro','f1_micro']
    for model in models:
        model_name = model.__class__.__name__
        accuracies = cross_validate(model, demo_features1, labels, scoring=score,
cv=CV)
        i=0
        for f1_macro,f1_micro in
zip(accuracies['test_f1_macro'],accuracies['test_f1_micro']):
            entries.append((i,model_name,f1_macro,f1_micro))
        i+=1
    cv_df = pd.DataFrame(entries,
columns=['index','model_name','f1_macro','f1_micro'])
    f1_macro = cv_df.groupby('model_name').f1_macro.mean()
    f1_micro = cv_df.groupby('model_name').f1_micro.mean()
    demo_acc = pd.concat([f1_macro,f1_micro], axis= 1, ignore_index=True)
    demo_acc.columns = ['f1-macro_mean','f1-micro_mean']
    print("Scores for min-df=",x," are given below:\n")
    print(demo_acc)
    l1=demo_acc['f1-macro_mean'].tolist()
    l2=demo_acc['f1-micro_mean'].tolist()
    l3=[]
    l3=[x]+l1+l2
    final3.append(l3)
    x+=1
temp3=pd.DataFrame(final1,columns=['min_df','DTC_f1-macro','LSVC_f1-
macro','LR_f1-macro','DTC_f1-micro','LSVC_f1-micro','LR_f1-micro'])
temp3.to_csv('scores from tf-idf.csv')
print(temp3)
```

#Plotting the results

```
temp3.plot(x='min_df',y=['DTC_f1-macro','LSVC_f1-macro','LR_f1-
macro'],marker='^',figsize=(10,5),ylim=(0.7,0.8));
```

#Maximum number of features method

```
final4=[]
w=500
while(w<=20000):
    vectorizer = TfidfVectorizer(sublinear_tf=True,
                                 max_features=w,
                                 ngram_range=(1, 3),
                                 stop_words='english',
                              )
    demo_features2 = vectorizer.fit_transform(df1.tweet).toarray()
    labels = df1.category_id
    print("\nFor max-features=",w,", each of the %d tweets is represented by %d
features\n" %(demo_features2.shape))
```

```
X = df1['tweet']

y = df1['label']

X_train, X_test, y_train, y_test = train_test_split(X, y,

                                        test_size=0.2,

                                        random_state = 0)

models = [

LinearSVC(max_iter=100000),

LogisticRegression(random_state=0,max_iter=10000),

DecisionTreeClassifier(random_state=0),

]


# 5 Cross-validation

CV = 5

cv_df = pd.DataFrame(index=range(CV * len(models)))


entries = []

score=['f1_macro','f1_micro']

for model in models:

    model_name = model.__class__.__name__
```

```
        accuracies = cross_validate(model, demo_features2, labels, scoring=score,
cv=CV)
        i=0
        for f1_macro,f1_micro in
zip(accuracies['test_f1_macro'],accuracies['test_f1_micro']):
          entries.append((i,model_name,f1_macro,f1_micro))
          i+=1
    cv_df = pd.DataFrame(entries,
columns=['index','model_name','f1_macro','f1_micro'])
    f1_macro = cv_df.groupby('model_name').f1_macro.mean()
    f1_micro = cv_df.groupby('model_name').f1_micro.mean()
    demo_acc = pd.concat([f1_macro,f1_micro], axis= 1, ignore_index=True)
    demo_acc.columns = ['f1-macro_mean','f1-micro_mean']
    print("Scores for max features=",w," are given below:\n")
    print(demo_acc)
    l1=demo_acc['f1-macro_mean'].tolist()
    l2=demo_acc['f1-micro_mean'].tolist()
    l3=[]
    l3=[w]+l1+l2
    final4.append(l3)
    w+=500
temp4=pd.DataFrame(final1,columns=['mnf','DTC_f1-macro','LSVC_f1-
macro','LR_f1-macro','DTC_f1-micro','LSVC_f1-micro','LR_f1-micro'])
temp4.to_csv('scores from tf-idf(mnf).csv')
print(temp4)
```

#Plotting the results

```
temp4.plot(x='mnf',y=['DTC_f1-macro','LSVC_f1-macro','LR_f1-
macro'],marker='^',figsize=(10,5));
```

#Classification of new texts

```
X_train, X_test, y_train, y_test = train_test_split(X, y,
                                                    test_size=0.2,
                                                    random_state = 0)

tfidf = TfidfVectorizer(sublinear_tf=True,
                        max_features=16000,
                    ngram_range=(1, 2),
                    stop_words='english')


fitted_vectorizer = tfidf.fit(X_train)

tfidf_vectorizer_vectors = fitted_vectorizer.transform(X_train)


model = DecisionTreeClassifier(random_state=0).fit(tfidf_vectorizer_vectors,

y_train)


new_tweet = input('Enter any text to classify:')

print(model.predict(fitted_vectorizer.transform([new_tweet])))
```

RESULTS AND DISCUSSION

Here we have used a new metric called F1 score. Accuracy is used when True positives and True negatives are important while F1 score is used when False positives and False Negatives are crucial. Here we are using Biased data.So there's a high possibility that we might get more then 4 categories . So, Accuracy mightnot be better metric for this data. That's why we are using F1 score .

Calculating F1 score for the models with count vectorizer :

Model	**F1 score**
Decision tree classifier	0.785344
Linear SVC	0.719764
Logistic regression	0.680612
Complement NB	0.625890
Multinomial NB	0.548618
SVC	0.481840
K Neighbors classifier	0.284201
Random Forest classifier	0.123618

Calculating F1 score for the models with TF-IDF vectorizer :

Model	**F1 score**
Decision tree classifier	0.866512
Linear SVC	0.852839
Logistic regression	0.840093
Complement NB	0.746002
Multinomial NB	0.734878
SVC	0.732793
K Neighbors classifier	0.541136
Random Forest classifier	0.466280

Table

Plotting Graph for Supervised Learning Models with TF-IDF vectorizer

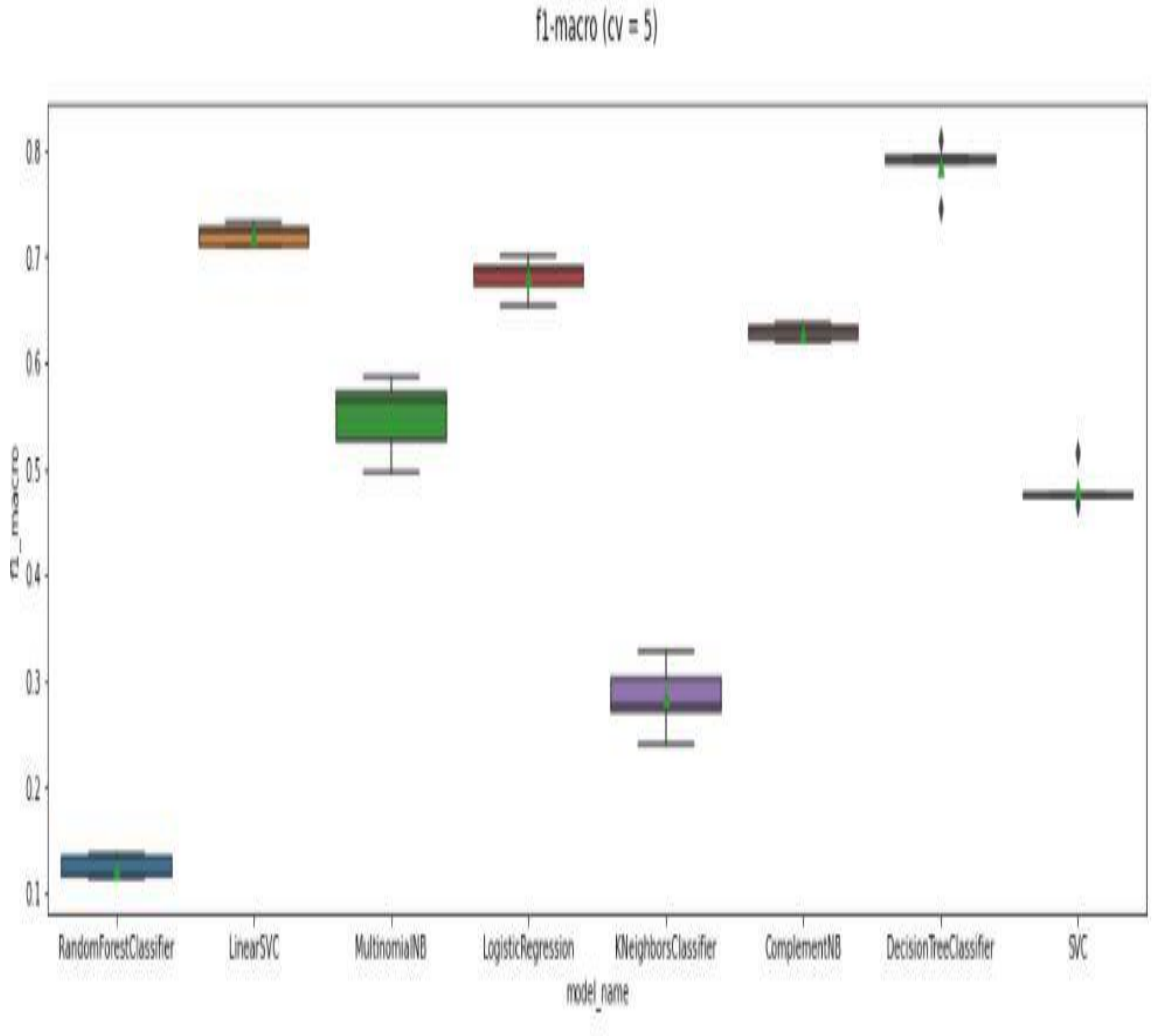


Fig F1 scores of SLM on TF-IDF vectorizers

Plotting Graph for Supervised Learning Models with count vectorizer

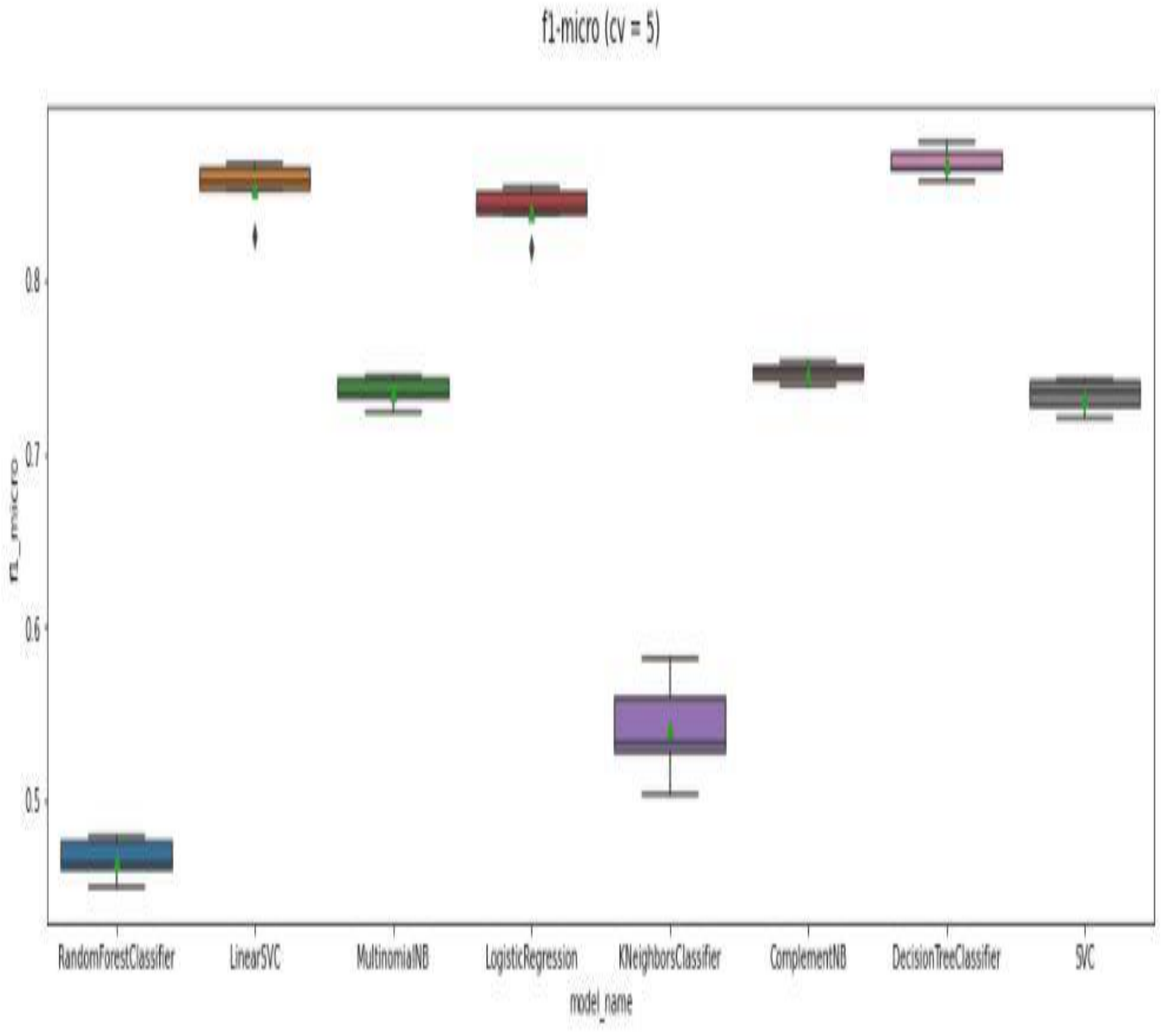
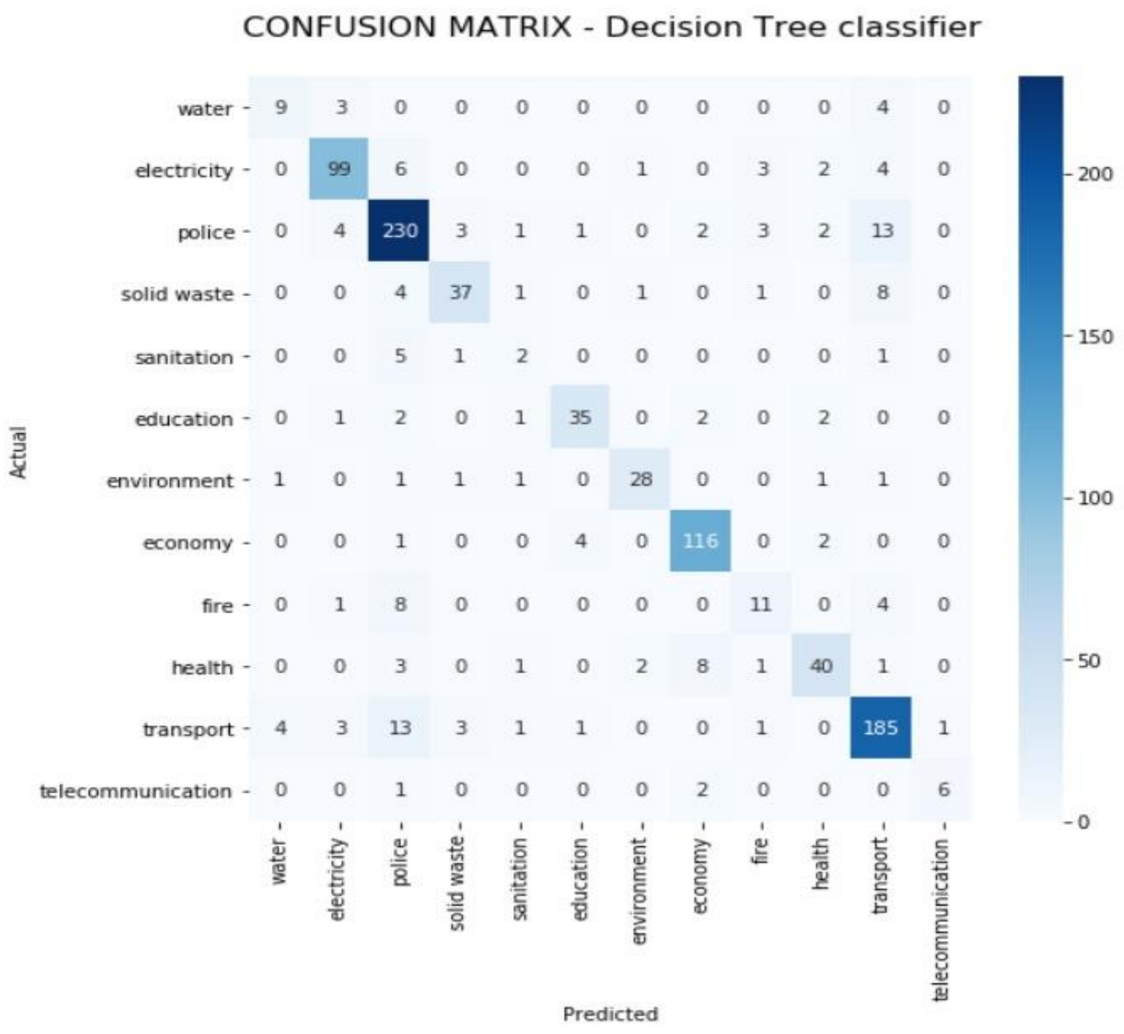


Fig F1 scores of all SLM with count vectorizer

By applying Supervised learning models on both the vectorizers, Decision Tree Classifier performed better among all the other Supervised Learning Models.

Confusion Matrix for the best learning model

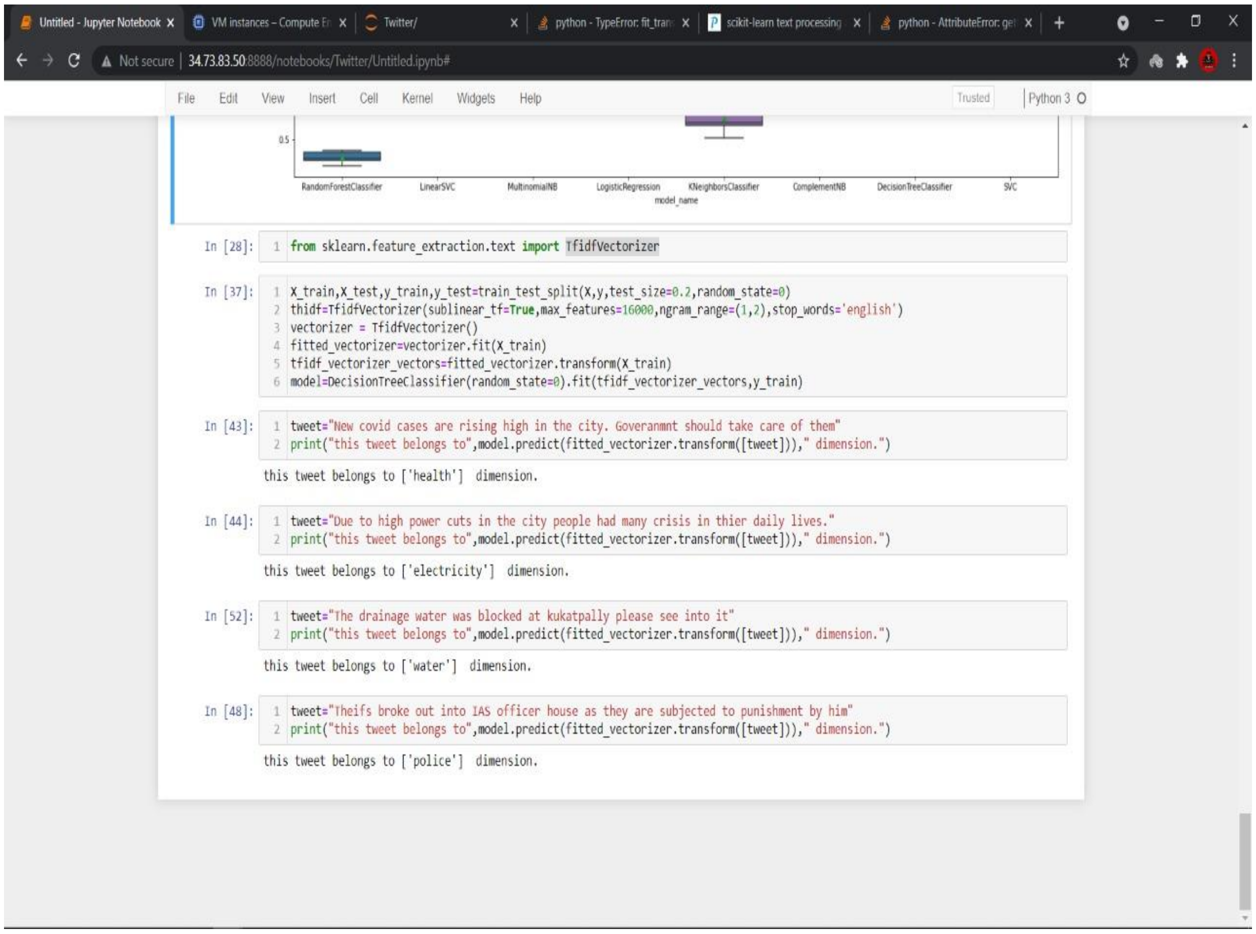


Actual \ Predicted	water	electricity	police	solid waste	sanitation	education	environment	economy	fire	health	transport	telecommunication
water	9	3	0	0	0	0	0	0	0	0	4	0
electricity	0	99	6	0	0	0	1	0	3	2	4	0
police	0	4	230	3	1	1	0	2	3	2	13	0
solid waste	0	0	4	37	1	0	1	0	1	0	8	0
sanitation	0	0	5	1	2	0	0	0	0	0	1	0
education	0	1	2	0	1	35	0	2	0	2	0	0
environment	1	0	1	1	1	0	28	0	0	1	1	0
economy	0	0	1	0	0	4	0	116	0	2	0	0
fire	0	1	8	0	0	0	0	0	11	0	4	0
health	0	0	3	0	1	0	2	8	1	40	1	0
transport	4	3	13	3	1	1	0	0	1	0	185	1
telecommunication	0	0	1	0	0	0	0	2	0	0	0	6

Fig Confusion matrix for decision tree classifier

The classification

The code snippet of Classification of the new tweets into the dimensions mentioned above is shown below



Untitled - Jupyter Notebook
VM instances – Compute Engine
Twitter/
python - TypeError: fit_transform
scikit-learn text processing
python - AttributeError: get
Not secure | 34.73.83.50:8888/notebooks/Twitter/Untitled.ipynb#
File
Edit
View
Insert
Cell
Kernel
Widgets
Help
Trusted
Python 3
0.5
RandomForestClassifier
LinearSVC
MultinomialNB
LogisticRegression
KNeighborsClassifier
ComplementNB
DecisionTreeClassifier
SVC
model_name

```
In [28]: from sklearn.feature_extraction.text import TfidfVectorizer
```

```
In [37]: X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.2,random_state=0)
         thidf=TfidfVectorizer(sublinear_tf=True,max_features=16000,ngram_range=(1,2),stop_words='english')
         vectorizer = TfidfVectorizer()
         fitted_vectorizer=vectorizer.fit(X_train)
         tfidf_vectorizer_vectors=fitted_vectorizer.transform(X_train)
         model=DecisionTreeClassifier(random_state=0).fit(tfidf_vectorizer_vectors,y_train)
```

```
In [43]: tweet="New covid cases are rising high in the city. Goveranmnt should take care of them"
         print("this tweet belongs to",model.predict(fitted_vectorizer.transform([tweet]))," dimension.")

this tweet belongs to ['health']  dimension.
```

```
In [44]: tweet="Due to high power cuts in the city people had many crisis in thier daily lives."
         print("this tweet belongs to",model.predict(fitted_vectorizer.transform([tweet]))," dimension.")

this tweet belongs to ['electricity']  dimension.
```

```
In [52]: tweet="The drainage water was blocked at kukatpally please see into it"
         print("this tweet belongs to",model.predict(fitted_vectorizer.transform([tweet]))," dimension.")

this tweet belongs to ['water']  dimension.
```

```
In [48]: tweet="Theifs broke out into IAS officer house as they are subjected to punishment by him"
         print("this tweet belongs to",model.predict(fitted_vectorizer.transform([tweet]))," dimension.")

this tweet belongs to ['police']  dimension.
```

Fig Classification of code snippet

CONCLUSION AND FUTURE SCOPE

This study approached Topic Classification in social networks datasets collected from Twitter. The main question was how to determine a SLM to classify messages into Smart City dimensions using ISO 37120 as the standard model.The feature selection process method was defined in a way to optimize the classifiers selected, which is the essence of the wrapper approach. Due to the fact the datasets were extremely imbalanced, the classifier's evaluation was done using firstly F1 macro global averages in the features selection process. Considering that the top models were not statistically different from the others following approach was used to determine the best SLM.

REFERENCES

[1].Wang, Q., Bhandal, J., Huang, S., &Luo, B.(2017). "*Classification of private tweets using tweet content*". In 2017 IEEE 11Th International conference on Semantic Computing(ICSC)(pp. 65-68), IEEE.

[2]. Moschen, S. A, Macke , J., Bebber, S., &da Silva, M.B.C (2019). "*Sustainable development of communities*:" ISO 37120 and UN goals. International Journal of Sustainablility in Higher Education.

[3]. De Oliveira, M., de Souza Baptista, C., Campelo, C. E., & Bertolotto, M.(2017 , April). *"A gold-standard social media corpus for urban issues*. In proceedings of the Symposium on Applied computing (pp. 1011-1016).