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

Report submitted to the SASTRA Deemed to be University as the requirement for the course

BICCIC707: MINI PROJECT

Submitted by

GUNTAKAL TEJA TARUN

(121014019, ICT)

KUCHI SRI BHARGAV RAM

(121014023, ICT)

VANGALA BHUVANA NAGA SAI REDDY

(121014055, ICT)

December 2020

SCHOOL OF COMPUTING

THANJAVUR, TAMIL NADU, INDIA – 613 401

SCHOOL OF COMPUTING

THANJAVUR – 613 401

Bonafide Certificate

This is to certify that the report titled **"AUTOMATED CLASSIFICATION OF TWEETS INTO SMART CITIES' DIMENSIONS USING MACHINE LEARNING ALGORITHMS"** was submitted as a requirement for the course, BICCIC707: MINI PROJECT for B.Tech. is a bonafide record of the work done by **GUNTKALA TEJA TARUN** (121014019, Information and Communication Technology), **KUCHI SRI BHARGAV RAM** (121014023, Information and Communication Technology), **VANGALA BHUVANA NAGA SAI REDDY** (121014055, Information and Communication Technology) during the academic year 2020-21, in the School of Computing, under my supervision.

Signature of Project Supervisor :

Name with Affiliation : **Devika R**, AP-II, CSE Department, School of Computing, SASTRA DEEMED TO BE UNIVERSITY

Date : 24-12-2020

Mini Project Viva voce held on ________________

Examiner 1 **Examiner 2**

ACKNOWLEDGEMENT

We would forever remain grateful to honorable **Dr. S. Vaidhyasubramaniam**, Vice Chancellor for his encouragement and support in our academic life at SASTRA Deemed to be University. We also wish to express our profound gratitude to **Dr. R. Chandramouli**, Registrar for their overwhelming support provided during our course span at SASTRA University.

We are extremely thankful to **Dr. A. UmaMakeswari**, Dean School of Computing, and **Dr. V. S. Shankar Sriram**, Associate Dean, Department of Computer Science Technology for providing us the opportunity to do this project and for all the academic help extended in our project.

We would also like to express our special thanks of gratitude to our Guide **Prof. R. Devika**, Assistant Professor, SOC for her assistance and guidance for the successful implementation of the project systematically and professionally.

Table of Contents

LIST OF FIGURES

LIST OF TABLES

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

ABSTRACT

Smart cities work mutually with data and technology for efficient results and to make better decisions to improve the standard of life. Real-time data give the unfold picture from the citizens' perspective, understand the areas 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. One of those models is the standard ISO 37120 proposed by the International Organization for Standardization (ISO) which defines a set of dimensions and indicators (e.g. Transportation dimension, Emergency and fire dimension, Solid Waste dimension) for services and to improve the standard of living in cities and communities. Nowadays citizens are raising 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 different smart cities' services dimensions. The classifiers generated here can be integrated into various city services and systems like governmental support decision systems, customer complaints systems, police offices, transportation companies, and environmental agencies. Features selection algorithms used are CountVectorizer and TF-IDF Vectorizer. 8 supervised Machine Learning algorithms are used in this paper.

KEYWORDS: Machine Learning, Text Classification, CountVectorizer, TF-IDF Vectorizer

CHAPTER 1

SUMMARY OF THE BASE PAPER

Title: Automated classification of social network messages into Smart Cities dimensions
Journal: Future Generation Computer Systems - *Science Citation Index Expanded*
Year: 2020

Nowadays, 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. The distribution of the data is shown in Table 1.1 below. Based on this factor we cannot use accuracy as a metric to get the efficiency of the models and hence we are using the two variants of the F1 Score which are F1 macro and F1 micro.

Sl. No	Label	%
1	Water	1.44433
2	Electricity	12.0798
3	Police	26.2605
4	Solid Waste	5.80357
5	Sanitation	0.971639
6	Education	4.33298
7	Environment	4.07038
8	Economy	13.4191
9	Fire	2.28466
10	Health	6.51261
11	Transport	22.0326
12	Telecommunication	0.787815

Table 1.1: Distribution of the Dataset

The process flow of the project is shown in Fig. 1.1. The extracted tweets are vectorized using the two vectorizers namely Countvectorizer and TF-IDF vectorizer. The Supervised Machine Learning (SML) models used in the project are Random Forest Classifier, Linear Support Vector Classifier, Multinomial Naive Bayes, Logistic Regression, K Neighbors Classifier, Complement Naive Bayes, Decision Tree Classifier, Support Vector Classifier. Firstly all the eight SML models are trained and tested using the two vectorizers with a min_df value, which selects the feature only if it is present in at least n tweets for a given n value, and No Maximum Number of Features (MNF) and the best three models out the above mentioned eight models.

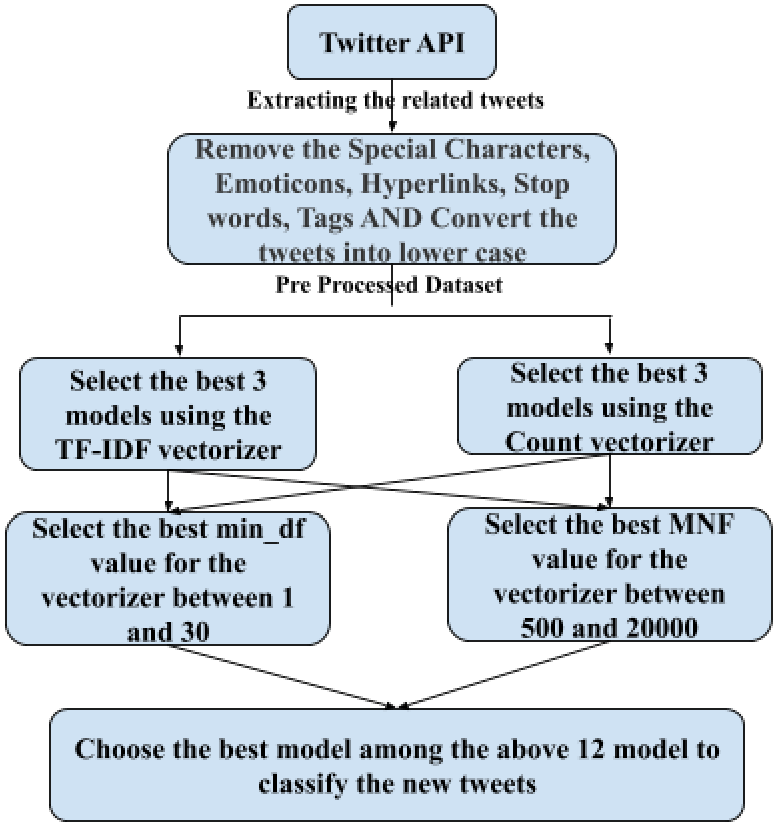


Fig. 1.1: Process Flow

TF-IDF Vectorizer:

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

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

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

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

CountVectorizer:

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

Using the above two vectorizers we get thousands of features which leads to overfitting and underfitting of the models and hence to select the most relevant features we use the min_df and MNF parameters. The min_df parameter selects the features that occur in at least N tweets for the given value of N, whereas the MNF parameter considers the top N frequent/essential features for a given value of N. With the help of these two parameters we could reduce the total number of features and hence can reduce the overfitting and underfitting and also can get more efficiency.

After selecting the top-performing algorithms from each of the above two methods(Countvectorizer and TF-IDF) we need to run these models with different values of parameters to know at which values of these parameters, the model is efficient. The parameters are the max number of features and minimum document frequency which refers to MNF and min-df respectively. We need to check at which values of these parameters, the top-performing algorithms from our first test will further perform better and increase the scores of the metrics. This is a trial and error method and we checked the scores of the models from min-df=5 to min-df=30 and mnf=500 to mnf=20000 separately. And this whole process needs to be run twice. Once with the Countvectorizer features and again with TF-IDF features. From these tests, we can conclude which algorithm with which features and parameters work most efficiently.

CHAPTER 2

MERITS AND DEMERITS

The model trained in this project is helpful to classify the issue raised by the citizens into a dimension that helps the public or private organizations to look into the issue at the earliest and solve it quicker, which helps the Govt. to resolve the issues faster than usual. Not only to the government it will be also helpful for other private social organizations like Businesses, enterprises, and other stakeholders.

The model developed in this paper also helps the government and other stakeholders to get to know about the public opinion about different organizations. This model also helps to understand the key aspects to be developed or to improve in the organizations in order to transform the city into a smart city.

The Authors of the base paper extracted every tweet and classified them into the dimensions through which 49% of the dataset contained the tweet that falls in none of the categories, hence the efficiency of the model is limited at around 50%

In our project, we extracted only the related tweets regarding the complaints on the daily essentials or the economy by the citizens so we managed to get an efficiency of the classification models around 80%.and here the only classification into dimensions is done but there is no option of classifying tweets into compliments and complaints which will be useful to rate the services provided by the government automatically.

CHAPTER 3

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','D
TC_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])))
```

CHAPTER 4

SNAPSHOTS

The distribution of the dataset:

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. The distribution is shown in Fig 4.1

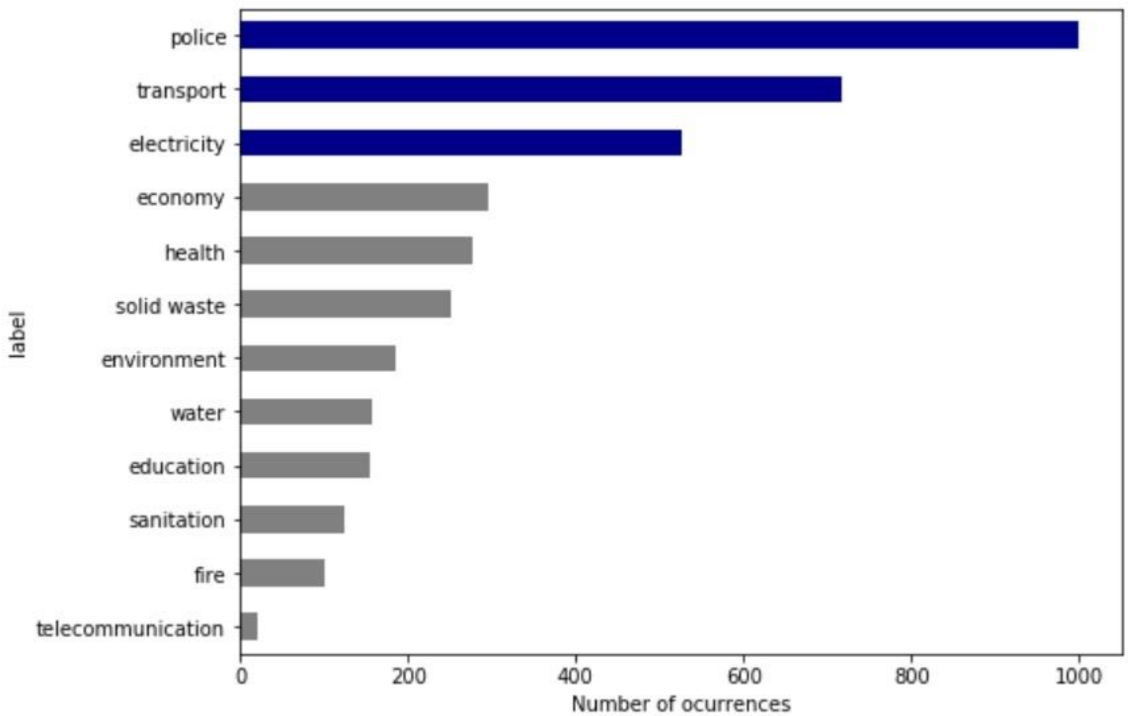


Fig 4.1: Distribution of the dataset

The F1 macro plot for TF-IDF vectorizer:

The F1 macro for all the eight models with five cross-validations

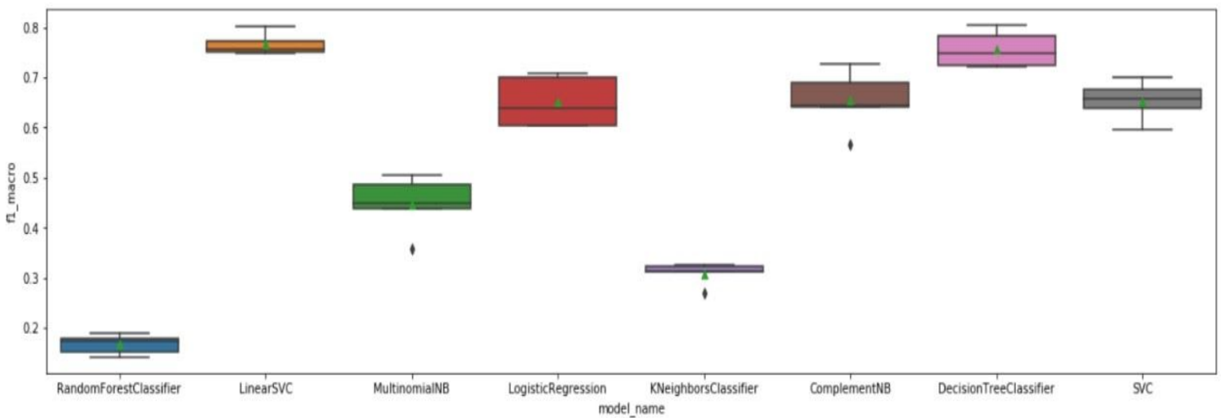


Fig 4.2: F1 macro for TF-IDF vectorizer

The F1 macro plot for Count vectorizer:

The F1 macro for all the eight models with five cross-validation

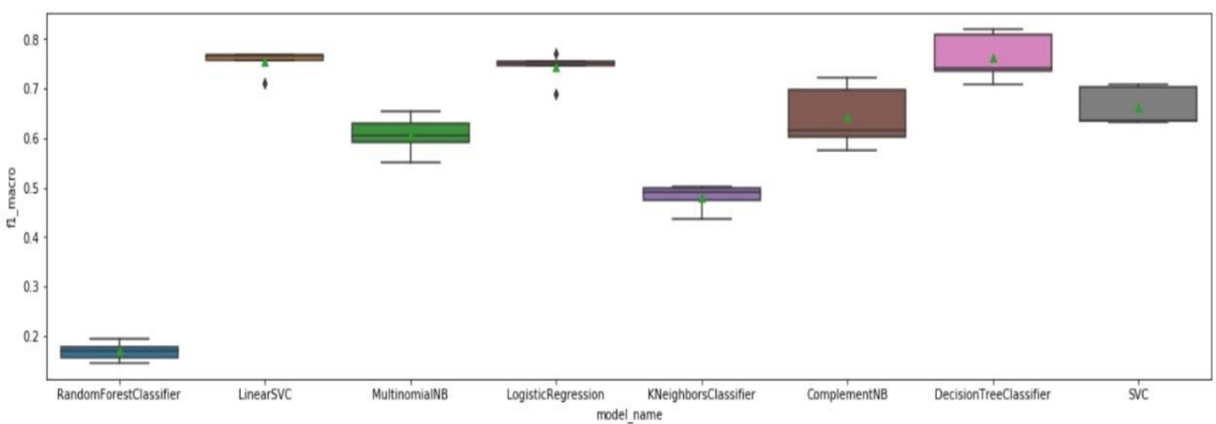


Fig 4.3: F1 macro for Count vectorizer

The F1 macro mean of the models with TF-IDF:

The mean of the five F1 macro scores obtained by using five cross-validations with TF-IDF vectorizer is given in the following table.

SL No.	Model	F1 macro (TF-IDF Vectorizer)
1	Linear SVC	0.765677
2	Multinomial NB	0.447405
3	Logistic Regression	0.651364
4	KNN Classifier	0.308341
5	Complement NB	0.653574
6	Decision Tree Classifier	0.755719
7	Random Forest Classifier	0.166807
8	SVC	0.652760

Table 4.1: F1 Macro scores for eight models with TF-IDF vectorizer

The F1 macro mean of the models with Countvectorizer:

The mean of the five F1 macro scores obtained by using five cross-validations with Count vectorizer is given in the following table.

Sl. No	Model	F1 macro (CountVectorizer)
1	Linear SVC	0.753848
2	Multinomial NB	0.605923
3	Logistic Regression	0.743108
4	KNN Classifier	0.480227
5	Complement NB	0.642085
6	Decision Tree Classifier	0.762284
7	Random Forest Classifier	0.168826
8	SVC	0.662879

Table 4.2: F1 Macro scores for eight models with Countvectorizer

The Classification Metrics for the best model:

The Decision Tree classifier with TF-IDF vectorizer when the MNF value is set to 16000, performed better than any other model. Here is the confusion matrix [Fig 4.4 (a)] and the Classification Metrics [Fig. 4.4 (b)] for this model

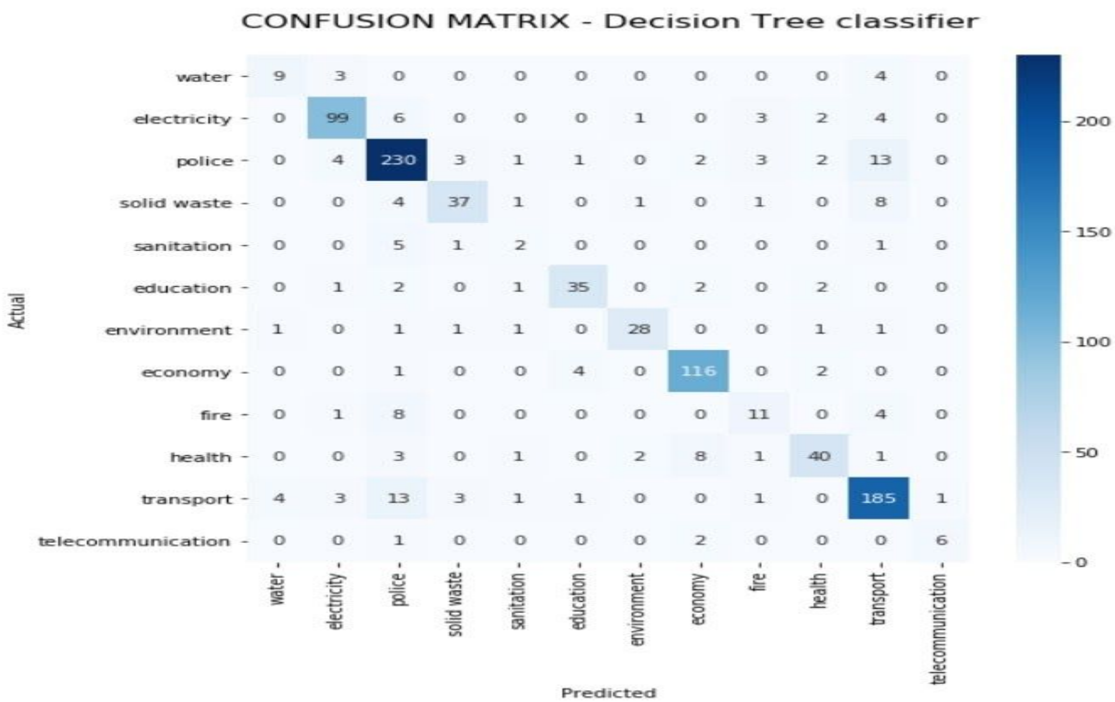


Fig 4.4: (a) Confusion Matrix

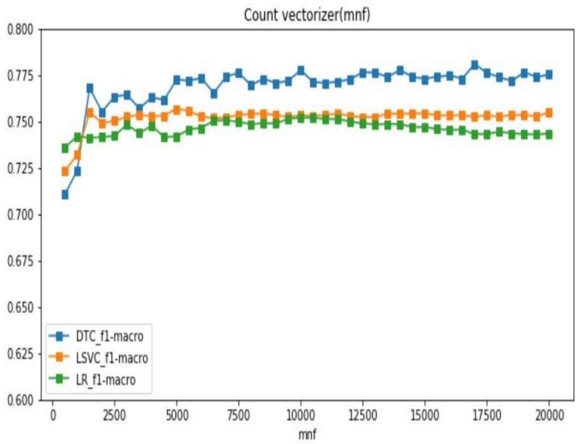
CLASSIFICATIION METRICS

	precision	recall	f1-score	support
water	0.64	0.56	0.60	16
electricity	0.89	0.86	0.88	115
police	0.84	0.89	0.86	259
solid waste	0.82	0.71	0.76	52
sanitation	0.25	0.22	0.24	9
education	0.85	0.81	0.83	43
environment	0.88	0.82	0.85	34
economy	0.89	0.94	0.92	123
fire	0.55	0.46	0.50	24
health	0.82	0.71	0.76	56
transport	0.84	0.87	0.85	212
telecommunication	0.86	0.67	0.75	9
accuracy			0.84	952
macro avg	0.76	0.71	0.73	952
weighted avg	0.84	0.84	0.84	952

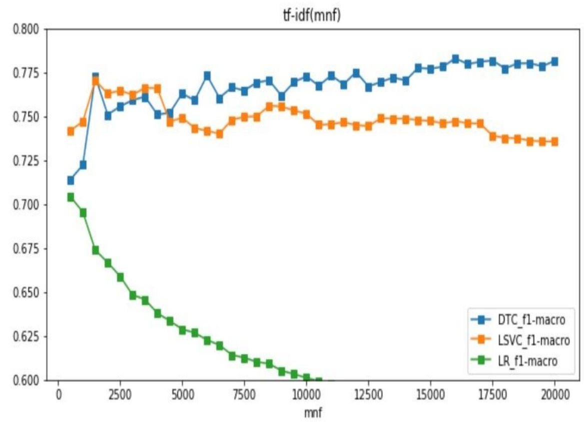
Fig: 4.4: (b) Classification metrics of the best model

The MNF plots:

The plot between the MNF value between 500 and 20000 and the corresponding F1 macro mean for the top three models



5.1(a)

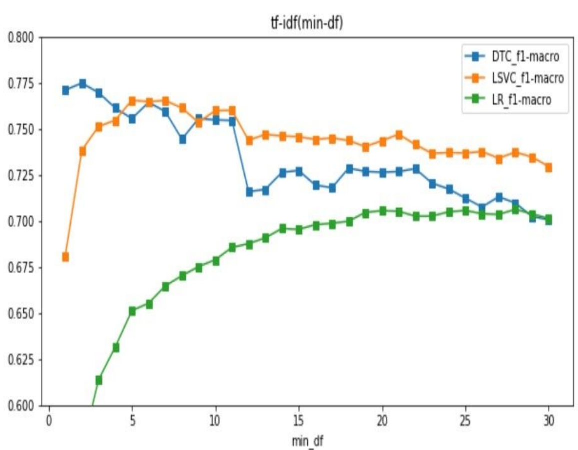


5.1(b)

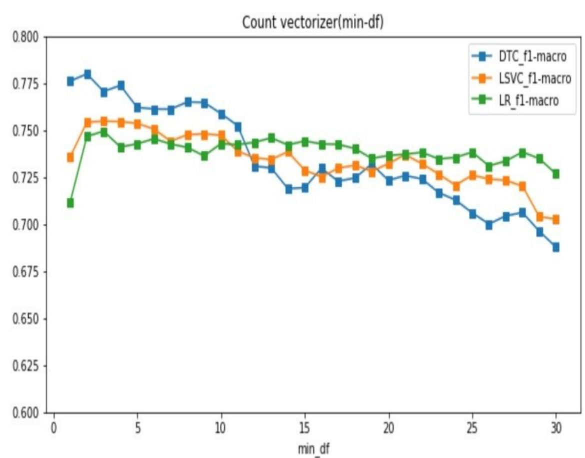
Fig. 4.5: (a) The MNF plot for TF-IDF **(b)** The MNF plot for Count

The min_df plots:

The plot between the min_df value between 1 and 30 and the corresponding F1 macro mean for the top three models



5.2(a)



5.2(b)

Fig 4.6: (a) The min_df plot for TF-IDF **(b)** The min_df plot for CountVectorizer

The Classification:

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

```
new_tweet = "my colony roads are filled with drainage water. please look into this problem sir"
print("This tweet belongs to",model.predict(fitted_vectorizer.transform([new_tweet]))[0],"dimension")
```

```
This tweet belongs to water dimension
```

```
new_tweet = "There is power cut from 3hrs in our kukatpally dharmareddy colony. please look into this problem sir"
print("This tweet belongs to",model.predict(fitted_vectorizer.transform([new_tweet]))[0],"dimension")
```

```
This tweet belongs to electricity dimension
```

```
new_tweet = "News coming from health department of telangana, covid19 cases registered on 28th july are 2900 in hyderabad"
print("This tweet belongs to",model.predict(fitted_vectorizer.transform([new_tweet]))[0],"dimension")
```

```
This tweet belongs to health dimension
```

Fig 4.7: The Classification Code snippet

CHAPTER 5

CONCLUSION AND FUTURE WORKS

Using the Count vectorizer we get a total of about 78000 features for the data extracted.
To minimize the no of features, the Maximum number of features (MNF) is used. The two vectorizers are applied with the default min_df and MNF values along with the above mentioned eight SML models and the best three for each vectorizer are identified using the F1 macro score as the efficiency metric. The F1 macro scores of the eight models are given in Table 4.1 and 4.2

From the F1 macro scores and the above table, we get to know that Decision Tree Classifier, Linear SVC, and Logistic Regression are the three models that performed well with both the vectorizers. So these three models are used to get the best min_df and MNF values with the vectorizers

Maximum Number of Features (MNF):

The MNF parameter in the vectorizer is used to consider only the N most frequent/important for the given N value. To get the best MNF value, compute the f1 macro score for the three models that performed well with the default values and assign the MNF parameter from 500 through 20000 and plot a graph between the MNF value and the F1 macro score. The graphs plotted are given in Fig. 4.5

From the above graphs, we can conclude that the Decision Tree Classifier with TF-IDF vectorizer at MNF "16000" achieved the better F1 macro score, which is nearly 78.31%, the Decision Tree Classifier with Count vectorizer at MNF "17000" achieved the better F1 macro score, which is nearly 78.09%

Min_df :

The min_df parameter in the vectorizer is used to consider only the features that occur in minimum N tweets for a given value of N. To get the best min_df value, compute the f1 macro score for the three models that performed well with the default values and assign the min_df parameter from 1 through 30 and plot a graph between the min_df value and the F1 macro score. The graphs plotted are given in Fig. 4.6

From the above graphs, we can conclude that the Decision Tree Classifier with TF-IDF vectorizer at min_df "2" achieved the better F1 macro score, which is nearly 77.49%, the Decision Tree Classifier with Count vectorizer at min_df "2" achieved the better F1 macro score, which is nearly 78.01%

CHAPTER 6
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 Sustainability in Higher Education.

[3]: De Oliveira, M. G., 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).