**SOURCE CODE**

import pandas as pd

import numpy as np

from sklearn.model\_selection import train\_test\_split

from sklearn.feature\_extraction.text import TfidfVectorizer

import warnings

warnings.filterwarnings('ignore')

from sklearn.linear\_model import LogisticRegression

from sklearn.metrics import accuracy\_score

df = pd.read\_csv("clean\_tweets\_1.csv")

df.head()

df.shape

dff = df.drop(['label'], axis = 1)

dff.head()

X\_temp, X\_test, y\_temp, y\_test = train\_test\_split(dff, list(df.label), test\_size=0.1)

X\_test.shape

X\_temp.shape

len(y\_test)

type(y\_temp)

X\_temp['label'] = y\_temp

X\_temp.head()

X\_temp.label[2]

type(X\_temp)

nonhate = X\_temp[X\_temp['label'] == 0]

nonhate.head()

hate = X\_temp[X\_temp.label == 1]

hate.shape[0]

nonhatesample = nonhate.sample(n = hate.shape[0])

nonhatesample.head()

nonhatesample.shape

ds = pd.concat([hate, nonhatesample], axis = 0)

ds.tail()

ds.shape

ds.to\_csv("trainset.csv")

ds = pd.read\_csv("trainset.csv")

ds.head()

ds\_temp = ds

testdf = X\_test

testdf['label'] = y\_test

testdf.shape

ds = pd.concat([ds\_temp, testdf], axis = 0)

ds.head()

ds.drop("Unnamed: 0",axis=1)

ds.shape

testdf.head()

ds = ds.drop("Unnamed: 0",axis=1)

ds.head()

list(testdf.index)

corpus = []

for i in range(ds.shape[0]):

corpus.append(ds.iloc[i][0])

corpus

cleaned\_corpus = [x for x in corpus if str(x) != 'nan']

print(cleaned\_corpus)

vectorizer = TfidfVectorizer()

X = vectorizer.fit\_transform(cleaned\_corpus)

feature\_names = vectorizer.get\_feature\_names()

dense = X.todense()

denselist = dense.tolist()

df2 = pd.DataFrame(denselist, columns=feature\_names)

df2

df2[0:7204]

tdf = df2

tdf['labelxyz'] = list(ds[0:len(tdf)].label)

tdf.tail()

print(tdf[tdf.labelxyz == 1])

tdf\_hate = tdf[tdf.labelxyz == 1]

tdf\_hate\_new = tdf\_hate

tdf\_hate\_new=tdf\_hate\_new.drop("labelxyz",axis=1)

print(tdf\_hate\_new)

tdf\_hate.shape

tdf\_nonhate = tdf[tdf.labelxyz == 0]

tdf\_nonhate\_new = tdf\_nonhate

tdf\_nonhate\_new=tdf\_nonhate\_new.drop("labelxyz",axis=1)

print(tdf\_nonhate\_new)

tdf\_nonhate.shape

X\_train\_hate = tdf\_hate.sample(frac=0.9, random\_state=0)

X\_test\_hate = tdf\_hate.drop(X\_train\_hate.index)

X\_train\_nonhate = tdf\_nonhate.sample(frac=0.406, random\_state=0)

X\_test\_nonhate = tdf\_nonhate.drop(X\_train\_nonhate.index)

X\_train\_df = pd.concat([X\_train\_hate, X\_train\_nonhate], axis = 0)

X\_train\_df

X\_train = X\_train\_df.drop(['labelxyz'], axis = 1)

y\_train = list(X\_train\_df.labelxyz)

X\_test\_df = pd.concat([X\_test\_hate, X\_test\_nonhate], axis = 0)

X\_test = X\_test\_df.drop(['labelxyz'], axis = 1)

y\_test = list(X\_test\_df.labelxyz)

from sklearn.ensemble import RandomForestClassifier

rand\_clf = RandomForestClassifier(n\_estimators=300,criterion='entropy',max\_depth=10,min\_samples\_split=5,min\_samples\_leaf=1,random\_state=0)

rand\_clf.fit(X\_train, y\_train)

predict = rand\_clf.predict(X\_test)

accuracy\_score(y\_test, predict)

from sklearn.metrics import confusion\_matrix

confusion\_matrix(y\_test, predict)

#Logistic Regression

log\_reg = LogisticRegression()

log\_reg.fit(X\_train, y\_train)

predict\_log = log\_reg.predict(X\_test)

accuracy\_score(y\_test, predict\_log)

from sklearn.metrics import confusion\_matrix

confusion\_matrix(y\_test, predict\_log)

# ## ROC CURVE FOR MODIFIED TFIDF

from sklearn.metrics import roc\_curve

from sklearn.metrics import roc\_auc\_score

print('Logistic Regression Score: ', roc\_auc\_score(y\_test, predict\_log))

print('Random Forest Score: ', roc\_auc\_score(y\_test, predict))

from sklearn.metrics import roc\_curve

log\_fpr, log\_tpr, threshold = roc\_curve(y\_test,predict\_log)

rand\_fpr, rand\_tpr, thresold = roc\_curve(y\_test, predict)

def graph\_roc\_curve\_multiple(rand\_fpr, rand\_tpr,log\_fpr, log\_tpr):

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

plt.title('ROC Curve \n Classifiers', fontsize=18)

plt.plot(rand\_fpr, rand\_tpr, label='random forest')

plt.plot(log\_fpr, log\_tpr, label='Logistic Regression')

plt.plot([0, 1], [0, 1], 'k--')

plt.axis([0, 1, 0, 1])

plt.xlabel('False Positive Rate', fontsize=16)

plt.ylabel('True Positive Rate', fontsize=16)

plt.annotate('Minimum ROC Score of 50% \n (This is the minimum score to get)', xy=(0.5, 0.5), xytext=(0.6, 0.3),

arrowprops=dict(facecolor='#6E726D', shrink=0.05),

)

plt.legend()

graph\_roc\_curve\_multiple(rand\_fpr, rand\_tpr, log\_fpr, log\_tpr)

plt.show()

#Prediction

A= 'HATE SPEECH'

B= 'GENUINE SPEECH'

predict = rand\_clf.predict(X\_test.iloc[1:5,:])

for i in range(len(predict)):

if predict[i] == 0:

print("{} :{} ".format(X\_test.iloc[i,:],A))

else:

print("{} :{} ".format(X\_test.iloc[i,:],B))

**RESULTS AND DISCUSSION**

label: This column represents the target variable or the label that indicates the category or class of a given tweet. It seems to be a binary label where:

0: Represents non-hate speech or genuine speech.

1: Represents hate speech.

clean\_tweet\_final: This column contains the original tweets.

Figure 1 is a representation of the initial dataset containing tweets that are used as input for building the hate speech detection model. The dataset comprises text data (tweets) along with their corresponding labels indicating whether the tweet contains hate speech or not.

A screenshot of a computer

Description automatically generated

Figure 1: Sample dataset of tweets for detecting the hate speech.

Figure 2 shows the dataset after removing the label column from the original dataset. This step is performed to separate the features (text data) from the labels, as proposed machine learning models require this separation for training and evaluation. Figure 3 is depicting a subset of the dataset that includes only non-hate speech labels. This subset is used for balancing the dataset to training the model. Figure 4 represents a portion of the dataset that's designated as the test set. It includes both hate speech and non-hate speech labels, which will be used to evaluate the model's performance.

A screenshot of a chat

Description automatically generated

Figure 2: Dataset after dropping the label column.

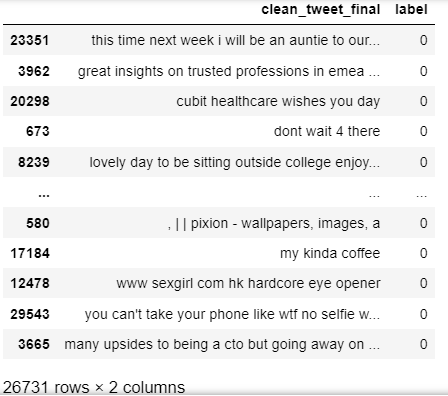


Figure 3: Illustration of dataset containing only non-hate speech labels.

Table 1 presents a performance comparison between two machine learning models, the Logistic Regression (LR) model and the Random Forest (RF) Classifier, for the task of hate speech detection. The table includes the accuracy percentages achieved by each model. Accuracy column displays the accuracy percentages achieved by each model in the hate speech detection task. Accuracy is a common evaluation metric in machine learning that measures the proportion of correctly classified instances out of the total instances. In this context, it indicates how well each model is able to correctly identify hate speech instances.

From Table 1, the LR Model achieved an accuracy of 79.77%. This means that out of all the instances the model evaluated, it correctly classified approximately 79.77% of them as either hate speech or non-hate speech. The RF Classifier achieved a higher accuracy of 84.05%. This indicates that the RF Classifier performed even better than the LR Model, correctly classifying about 84.05% of instances. In terms of improvement, the RF Classifier outperformed the LR Model by an increment of approximately 4.28 percentage points (84.05% - 79.77% = 4.28%). This suggests that the RF Classifier has a higher accuracy rate, meaning it's more adept at distinguishing between hate speech and non-hate speech instances compared to the LR Model.

A screenshot of a social media post

Description automatically generated

Figure 4: Sample test data frame with hate and non-hate speech labels.

Table 1: Performance comparison of LR model, and RF Classifier for hate speech detection.

|  |  |
| --- | --- |
| Model | Accuracy (%) |
| LR Model | 79.77 |
| RF Classifier | 84.05 |

Figure 5 displays the confusion matrix obtained when evaluating the hate speech detection model using the Logistic Regression (LR) model. The confusion matrix shows the number of true positives, true negatives, false positives, and false negatives, which are used to assess the model's performance. Similarly, Figure 6 represents the confusion matrix obtained from evaluating the hate speech detection model using the Random Forest (RF) Classifier.

A blue squares with white text

Description automatically generated

Figure 5: Obtained confusion matrix of hate speech detection using LR model.

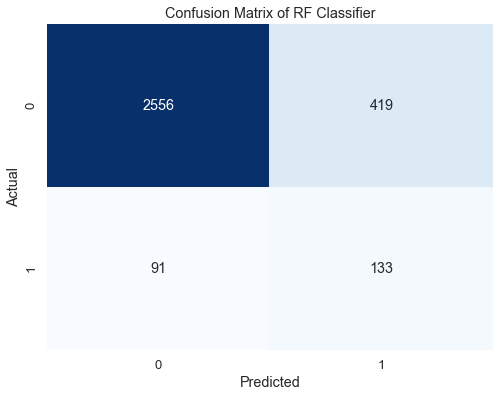


Figure 6: Displaying the confusion matrix obtained using RF-based hate speech detection model.