A PROJECT REPORT ON

FAKE NEWS DETECTION MODEL USING MACHINE LEARNING AND PYTHON

(A Project Report submitted in partial fulfillment of the requirements of Bachelor of Technology in Information Technology of the Maulana Abul Kalam Azad University of Technology, West Bengal)

Submitted To:
Prof. Md. Iqbal Qurashi
(Dept. Of Information Technology)

Submitted By:

Sohini Ghosh (10200216032)
Ankan Chattopadhyay (10200216071)
Bikash Prajapati (10200216063)
Ojas Vishwakarma (10200216047)

কল্যাণী - ৭৪১ ২৩৫
নদীয়া, পশ্চিমবঙ্গ

Kalyani 741 235
Nadia, West Bengal,India

পত্রাঙ্ক /Ref. No.:
তারিখ / Date :

কল্যাণী গভঃ ইঞ্জিনিয়ারিং কলেজ
Kalyani Government Engineering College
(Govt. of West Bengal)

Certificate of Approval

This is to certify that Ankan Chattopadhyay, Bikash Prajapati, Sohini Ghosh and Ojas Vishwakarma have done their final year project work entitled "FAKE NEWS DETECTION MODEL" under my direct supervision and they have fulfilled all the requirements relating to the Final Year Project. It is also certified that this project work being submitted, fulfills the norms of academic standard for B. Tech Degree in Information Technology of The Maulana Abul Kalam Azad University of Technology and it has not been submitted for any degree whatsoever by them or anyone else previously.

……

Head
Dept. Of Information Technology
Kalyani Govt. Engineering College

……

Supervisor
Dept. Of Information Technology
Kalyani Govt. Engineering College

……

Project Coordinator
Dept. Of Information Technology

Kalyani Govt. Engineering College

……

Examiner

Acknowledgement

We take this opportunity to express our profound gratitude and deep regards to our faculty Mr. Md. Iqbal Qurashi Sir for his exemplary guidance, monitoring and constant encouragement throughout the course of this project. The blessings, help and timely guidance given, shall carry us a long way in the journey of life on which we are about to embark. We are grateful for every team member's cooperation during the period of this project.

Ankan Chattopadhyay-10200216071
Sohini Ghosh-10200216032
Bikash Prajapati -10200216063
Ojas Vishwakarma -10200216047

ABSTRACT

There have been many instances of fake news circulation in recent times: misinformation related to COVID-19 is in the form of social media messages related to home remedies that have not been verified, fake advisories and conspiracy theories; there have been multiple instances of pictures from the Syrian and the Iraqi civil wars being passed off as from the Kashmir conflict with the intention of fuelling unrest and backing insurgencies; following the 2016 Indian banknote demonetisation, multiple fake news reports about "spying technology" added in the banknotes went viral on WhatsApp and had to be dismissed by the government and the list is endless. It is very important for us to understand which information is true and which isn't. Earlier efforts have been made in this field to analyse text properties of fake and real news articles, on the basis of term frequency. Here, we aim to create a software model based on machine learning and NLP techniques that artificially differentiates a real news article from a fake one. This gives our model an edge over the previous ones as it has helped us discover new features and their patterns. Our model is trained by a data set of about 9000 news articles and tested by 18000 of them, that have already been determined as real or fake. Upon receiving a news article as input, the model scans the text and identifies the part of speech of each word, tagging it with the same. Then these tagged words are divided into bigrams with the aim of extracting psycholinguistic features and performing discourse analysis. This has helped to figure out the textual properties of real and fake news articles, and hence predict a given news article as real or fake up to a certain level of accuracy. This model works only for news articles strictly in English. The results of our model have been satisfactory. It can be used to keep safe from undue allurance through misinformation, biasness, false connection of events as well as for cyber governance. However, there is a non-negligible chance of misprediction, one way of eliminating which is vigorous research in the extraction of linguistic cues and improved feature engineering for data preparation.

Table Of Contents

INTRODUCTION

Information shapes our world view: we make important decisions based on information. We form an idea about people or a situation by obtaining information. If the information we saw on the Web or otherwise is invented, false, exaggerated or distorted, we won't make good decisions. A type of yellow journalism, fake news encapsulates pieces of news that may be hoaxes and is generally spread through social media and other online media. This is often done to further or impose certain ideas and is often achieved with political agendas. Such news items may contain false and/or exaggerated claims and may end up being viralised by algorithms, and users may end up in a filter bubble. With the introduction of social media, the spread of fake news has increased and it became difficult to differentiate between true news and fake news. It is a matter of concern as it manipulates the public opinions. During the American Presidential elections of 2016, it was estimated that over 1 million tweets are related to fake news "Pizzagate" by the end of the elections. The extensive spread of fake news can have a huge negative impact on individuals and society as a whole.

The extensive spread of fake news can have a serious negative impact on individuals and society. It has brought down the authenticity of the news ecosystem as it is even more widely spread on social media than most popular authentic news. It is one of the biggest problems which has the ability to change opinions and influence decisions and interrupts the way in which people respond to real news. It has political influence, can encourage mistrust in legitimate media outlets, influence financial markets and cause damage to individuals reputation. During the American presidential elections of 2016, a survey revealed that many young men and teens in Veles were running hundreds of websites which published many false viral stories that supported Trump. These fake news influenced many people which affected the election results. This is just a small instance of how the spread of fake news can influence people. Many organizations have come forward to stop the spread of fake news. Eg. Google app uses Artificial Intelligence to select stories and stop fake news.

PROJECT OBJECTIVE

Fake news is a real menace as it can quickly spread panic among the public. It can also affect major world events, as was seen in the US Presidential Elections. With the flood of

news arising from online content generators, as well as various formats and genres, it is impossible to verify news using traditional fact checkers and vetting. To tackle this problem of quick and accurate classification of news as fake or authentic, we provide a computational tool.

The major objectives of this project are:

1. Taking the help of linguistic cues to develop a machine learning based model for accurately determining the likeliness of a news being fake or authentic.
2. To get high accuracy to determine a news is fake or true.

PROJECT SCOPE

The scope of this project is very diverse, it ranges from various online social media like Facebook, Twitter, Instagram etc. to fake blogs, fake websites that deceive the users in one way or the other. This project will try to enhance the user experience on the online social media platform and will also save a lot of time of users that they might spend on fake news otherwise.

DATA DESCRIPTION

We have used four datasets for this project-

1.data.csv-

This dataset has a shape of 4027×4. The first column identifies the URL of the news, the second and third are the Headline and body, and the fourth column has labels denoting whether the news is REAL or FAKE. In this particular dataset, we have used '0' to denote 'Real News' and '1' to denote 'Fake news'.

URLs	Headline	Body	Label

http://www... .	Four ways Bob ..	Image copyright Getty Images...	1
http://www... .	Linklater's war..	LONDON (Reuters) ….	1
http://www... .	Trump’s Fight..	Trump’s Fight With...	1
http://www... .	Egypt's Cheiron..	MEXICO CITY (Reuters)- ...	1

2.fake.csv-

This dataset has a shape of 17925×20. The columns of this dataset include news URL, language of the news, date of publication, Country,type and title along with other attributes.

uuid	**title**	**text**	**type**
6a175f46b	Muslims BUSTED...	Print they should...	bias
2bdc29d12	Re: Why Did Att...	Why Did Attorney..	bias
c70e149fd	BREAKING: Weine...	Red State:...	bias
7cf7c15731	PIN DROP SPEECH...	Email Kayla...	bias

3.Fake_or_real_news.csv-

This dataset has a shape of 7772×4. The first column identifies the URL of the news, the second and third are the title and text, and the fourth column has labels denoting whether the news is REAL or FAKE.

id	**title**	**text**	**label**
8476	You can Smell..	Daniel Greenfield...	FAKE

10294	Watch the exac..	Google Pinterest Digg..	FAKE
3608	Kerry to go to...	U. S. Secretary of state...	REAL
10142	Bernie Support...	â€" Kaydee King...	FAKE

4.train.csv - This dataset has a shape of 21,305 x 5. The first column denotes id followed by title,author name, text and label. The label column indicates whether the particular news is real or fake by flagging real as '0' and '1' as fake.

id	**title**	**author**	**text**	**label**
0	House Dem Ai...	Darrell	House dem aide...	1
1	FLYNN: Hillary...	Daniel	Ever get the feeling	0
2	Why The Truth...	Consortiumnews	Why the truth might	1
3	15 Civilians...	Jessicca	Videos 15 Civilians	1

MODEL

We have used four classifiers-MULTINOMIAL NAIVE BAYES , RANDOM FORESTS GRADIENT BOOSTING AND LINEAR SVC to check the accuracy of Fake news detection in the first part of our project. In the second part we have used NATURAL LANGUAGE PROCESSING technique coupled with the above mentioned four classifiers to improve the performance to check on a news being fake. Based on the

accuracy level and consistency of all four classifiers, we will select our FINAL CLASSIFICATION MODEL to differentiate between real and fake news.

1. **NAIVE-BAYES:**

   Naive bayes is a supervised learning algorithm which is used for classification. It is based on Bayes' theorem assuming that features are independent of each other. It calculates the probability of every class, the class with maximum probability is chosen as the output.

2. **RANDOM FOREST:**

   Random Forests is a bagging type of ensemble model in which the base model for bagging is decision tree. In addition to bagging (i.e taking random samples of total data and train those samples independently and then take the majority voting of numerous samples of the total dataset to find the output of the total dataset), there is feature bagging (i.e column sampling in which not all the columns/features are taken into consideration while training but rather random samples of features are considered while training different samples.
   Random Forest = Bagging with decision tree as base model + feature bagging.

3. **GRADIENT BOOSTING:**

   Gradient boosting is an example of boosting ensemble models. Boosting is an ensemble technique in which the predictors are not made independently, but sequentially. Boosting tries to convert a weak learner to become better. It learns from its mistakes/errors and tries to reduce the error.

4. **LINEAR SVC:**

Linear Support Vector Classifier is a linear model for classification and regression problems. Linear SVC is an algorithm that takes the data as an input and outputs a line that separates those classes if possible The objective of a Linear SVC (Support Vector Classifier) is to fit to the input data, returning a "best fit" hyperplane that divides, or categorizes the data. From there, after getting the hyperplane, some features can be added to the classifier to see what the "predicted" class is.

NATURAL LANGUAGE PROCESSING(NLP):

Natural Language Processing(NLP) is a field of Artificial Intelligence(AI) that focuses on quantifying human language to make it intelligible to machines. It combines the power of linguistics and computer science to study the rules and structure of language, and create intelligent systems capable of understanding, analyzing, extracting meaning from text and speech.

Linguistics is used to understand the structure and meaning of a text by analyzing different aspects like syntax, semantics, pragmatics, parts of speech. Then computer science transforms this linguistic knowledge into rule based or machine learning algorithms that can solve specific complications and perform the needed tasks.

SOURCE CODE

The Jupyter notebook has been used for implementing our machine learning algorithm and it has many files including dataset files and python notebooks which have following extensions i.e. ".tsv" ,".csv ",".pynb" . We also tried to use python libraries like torch and the famous numpy. Some excerpts from our project are shown below.

- **Making Necessary Imports:**

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
```

```
from sklearn.feature_extraction.text import TfidfVectorizer, CountVectorizer
import matplotlib.pyplot as plt
import itertools
from sklearn import svm
from sklearn.naive_bayes import MultinomialNB
from sklearn.ensemble import RandomForestClassifier, GradientBoostingClassifier
from sklearn import metrics
```

- **Reading the data into DataFrame and getting the labels:**

```
df = pd.read_csv('Dataset/data.csv')
df.loc[df['Label']== 0, 'Label'] = 'REAL'
df.loc[df['Label']== 1, 'Label'] = 'FAKE'
Df.columns
df['Label'].value_counts()
```

- **Filtering out the duplicate data and news with small body to improve the performance:**

```
df = df.drop_duplicates()
cnt = 0
ind = []
for art in df['Body']:
if len(str(art)) < 10:
 ind.append(cnt)
   cnt+=1
df = df.drop(df.index[ind])
```

- **Calculating total classified records in our dataset and plotting:**

```
df['Label'].value_counts()
```

[12]: REAL 15343
FAKE 12522
Name: Label, dtype: int64

```
#Bar graph

df['Label'].value_counts().plot(kind = 'bar')
```

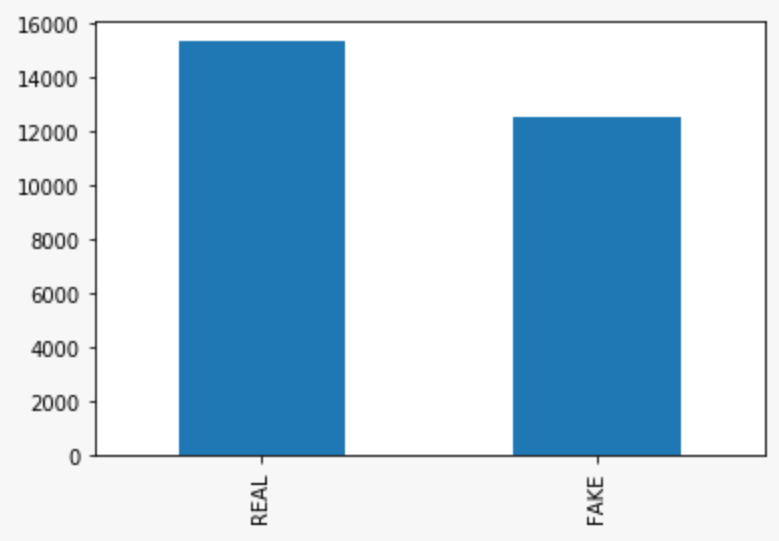

Fig. 1.1 Plotting real and fake data of the entire dataset

- **Splitting the dataset into training and testing sets where test size = 0.33:**

```
df["Text"] = df["Headline"].map(str) + df["Body"]
y = df.Label
y = y.astype('str')
X_train, X_test, Y_train, Y_test = train_test_split(df['Text'],y, test_size=0.33)
X_train
```

- **Initializing the TF-IDF Vectorizer and Pickle:**

```
tfidf_vectorizer = TfidfVectorizer(stop_words='english', ngram_range = (2,2))
tfidf1_train = tfidf_vectorizer.fit_transform(X_train.astype('str'))
tfidf1_test = tfidf_vectorizer.transform(X_test.astype('str'))
pickle.dump(tfidf1_train, open("tfidf1_train.pickle", "wb"))
pickle.dump(tfidf1_test, open("tfidf1_test.pickle", "wb"))
```

- **Getting the confusion matrix:**

```
def plot_confusion_matrix(cm, classes, title='Confusion matrix',cmap=plt.cm.Blues):
  plt.imshow(cm, interpolation='nearest', cmap=cmap)
 plt.title(title)
plt.colorbar()
   tick_marks = np.arange(len(classes))
  plt.xticks(tick_marks, classes, rotation=45)
  plt.yticks(tick_marks, classes)
  thresh = cm.max() / 2.
 for i, j in itertools.product(range(cm.shape[0]), range(cm.shape[1])):
    plt.text(j, i, cm[i, j],
         horizontalalignment="center",
         color="white" if cm[i, j] > thresh else "black")
plt.tight_layout()
plt.ylabel('True label')
  plt.xlabel('Predicted label')
```

- **Applying Naive Bayes Algorithm:**

```
clf = MultinomialNB()
clf.fit(tfidf1_train, Y_train)
pickle.dump(clf, open('tfidf_nb', 'wb'))
pred = clf.predict(tfidf1_test)
score = metrics.accuracy_score(Y_test, pred)
print("Accuracy with Multinomial Naive Bayes:   %0.3f" % score)
```

Accuracy with MultinomialNB: 0.825

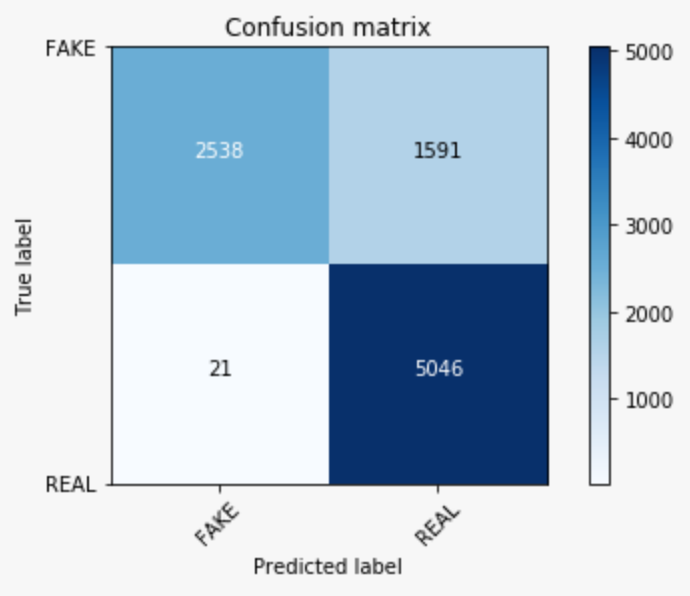

Fig. 2.1 Plotting confusion matrix for Naive-Bayes algorithm

- **Applying Random Forests algorithm:**

```
clf = RandomForestClassifier()
clf.fit(tfidf1_train, Y_train)
pickle.dump(clf, open('tfidf_rf', 'wb'))
pred = clf.predict(tfidf1_test)
score = metrics.accuracy_score(Y_test, pred)
print("Accuracy with RandomForestClassifier:   %0.3f" % score)
```

Accuracy with RandomForestClassifier: 0.876

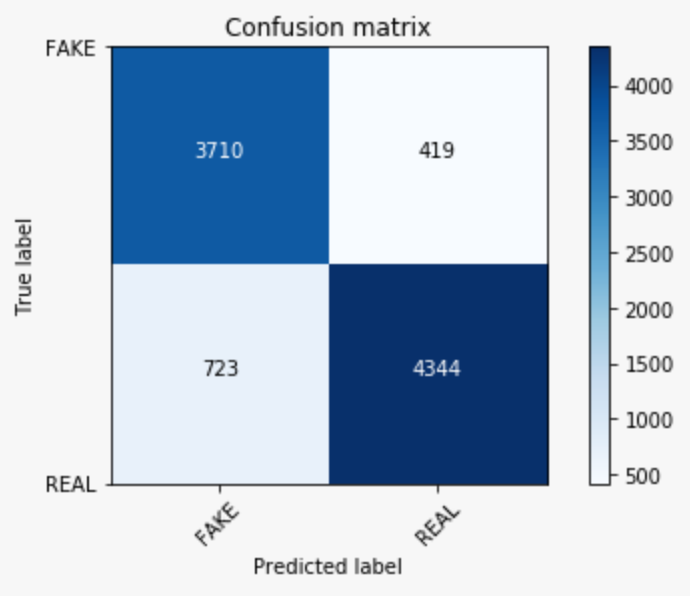

Fig. 2.2 Plotting confusion matrix for Random-Forests algorithm

- **Applying Gradient Boosting algorithm:**

```
clf_GBC = GradientBoostingClassifier()
clf_GBC.fit(tfidf1_train, Y_train)
pickle.dump(clf_GBC, open('tfidf_gb', 'wb'))
pred = clf_GBC.predict(tfidf1_test)
score = metrics.accuracy_score(Y_test, pred)
print("Accuracy with Gradient Boosting: %0.3f" % score)
Accuracy with Gradient Boosting: 0.848
```

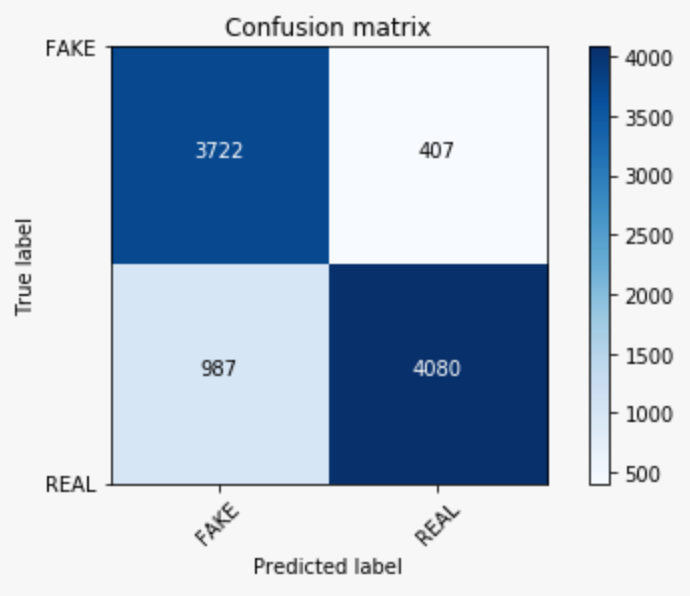

Fig. 2.3 Plotting confusion matrix for Gradient-Boosting algorithm

- **Applying Linear SVC algorithm:**

```
clf_svm = svm.SVC(kernel='linear')
clf_svm.fit(tfidf1_train, Y_train)
pickle.dump(clf_svm, open('tfidf_svc', 'wb'))
pred = clf_svm.predict(tfidf1_test)
score = metrics.accuracy_score(Y_test, pred)
print("Accuracy with SVC: %0.3f" % score)
```

```
Accuracy with SVC: 0.940
```

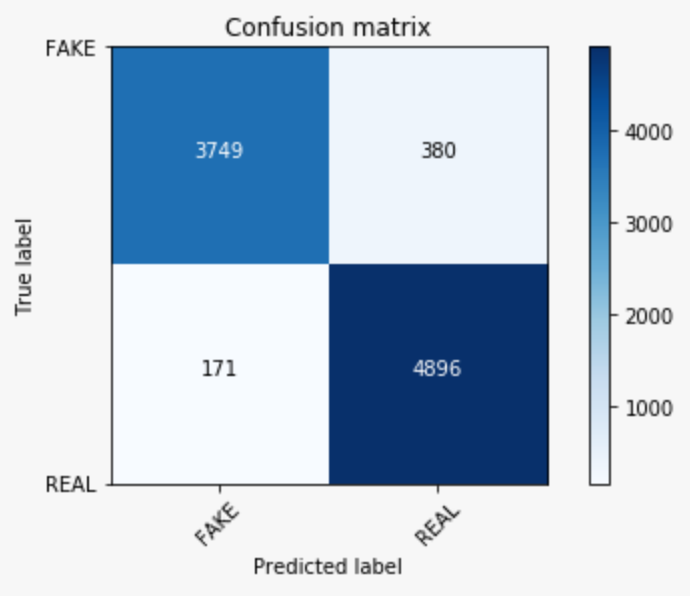

Fig. 2.4 Plotting confusion matrix for Linear-SVC algorithm

- **Determining ROC curve with classifiers:**

```
plt.figure(0).clf()
for model, name in [(clf_MNB, 'Multinomial Naive Bayes '), (clf_RFC,'Random Forest Classifier'),(clf_GBC,'Gradient␣ ,→Boosting'),(clf_svm, 'Support Vector classifier')                    ]:
if name == 'Support Vector classifier':
        pred = model.decision_function(tfidf1_test)
Else:
        pred = model.predict_proba(tfidf1_test)[:,1] fpr, tpr, thresh = metrics.roc_curve(Y_test.values, pred, pos_label='REAL')
plt.plot(fpr,tpr,label="{}".format(name))

        plt.legend(loc=0)
```

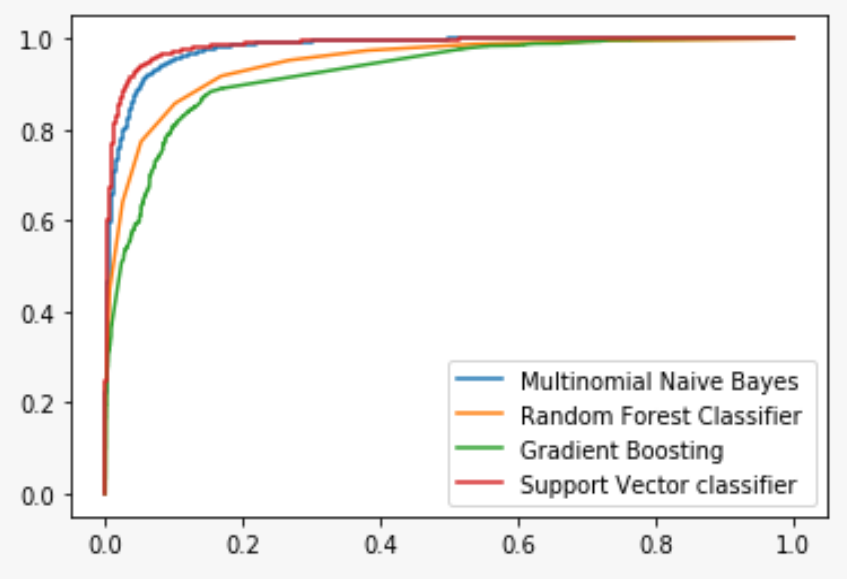

Fig. 2.5 ROC curve for displaying the performance of all the classifiers

- **Calculating Training time for each classifier:**

```
import logging
import sys
from time import time
def benchmark(clf,name):
print('_' * 80)
print("Training: "+name)
t0 = time()
clf.fit(tfidf_train, y_train)
train_time = time() - t0
print("train time: %0.3fs" % train_time)

# Train Random Forest Classifier
print('=' * 80)
print("RandomForestClassifier")
results.append(benchmark(RandomForestClassifier(),"RandomForestClassifier"))

# Train Naive Bayes classifier
print('=' * 80)
print("Naive Bayes")
 results.append(benchmark(MultinomialNB(),"Naive Bayes"))
```

```
# Train Gradient Boosting Classifier
print('=' * 80)
print("GradientBoosting")
results.append(benchmark(GradientBoostingClassifier(),"GradientBoosting"))

# Train SVC Classifier
 print('=' * 80)
print("SVC")
results.append(benchmark(svm.SVC(kernel='linear'),"SVC"))
```

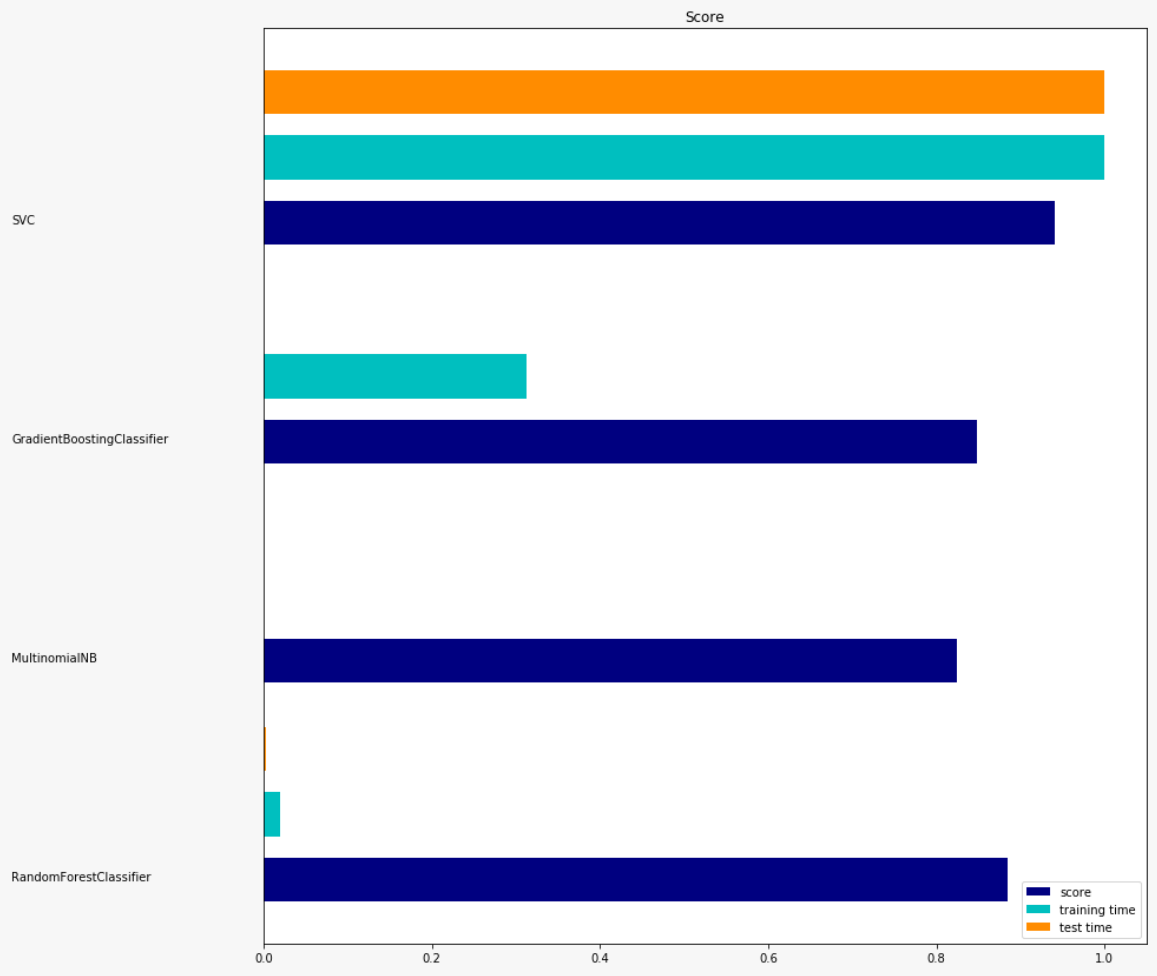

Fig. 2.6 Bar graph showing training time for every classifier

- **Applying Natural Language Processing(NLP):**

We have used Spacy library from NLP to extract more information about input data and analyze it better to improve the fake news detection accuracy. Spacy is designed specifically for production use and helps to build applications that process and "understand" large volumes of text. It can be used to build information extraction or natural language understanding systems, or to pre-process text for deep learning.

#Loading Spacy and tokenizing entire text using it:

```
nlp = spacy.load('en_core_web_sm')
txt = ('We will overcome.'')
doc = nlp(txt)
print([token.text for token in doc])
```

#Generating POS tags for all the tokens and adding a new column by replacing text with their POS tags:

```
nlp = spacy.load('en_core_web_sm')
x = []
df["Text"] = df["Headline"].map(str) + df["Body"]
for txt in df['Text']:
  text_new = []
doc = nlp(str(txt))
for token in doc:
        text_new.append(token.pos_)
Txt = ' '.join(text_new)
x.append(Txt)
df['Text_pos'] = x
df.to_pickle('newdata.pkl')
```

#Splitting the data set into training and testing sets:

```
y = df.Label
y = y.astype('str')
```

```
x_train, x_test, y_train, y_test = train_test_split(df['Text_pos'],y, ␣
,→test_size=0.33)
x_train
```

#After training data set with pos tagging:

```
19745 PROPN PROPN VERB VERB SCONJ ADP PROPN PROPN PU…
 9021 NUM PROPN PROPN AUX VERB ADP PROPN PROPN NUM N…
6728 PROPN PROPN VERB PRON DET ADJ PUNCT ADV PROPN …
16882 DET PROPN PROPN ADP PROPN PROPN PUNCT CCONJ AD…
30147 PROPN PROPN VERB SYM NUM NOUN ADP PROPN PROPN … …
25864 PROPN PUNCT NOUN ADP PROPN PUNCT AUX PART VERB…
26106 PROPN PROPN PUNCT ADP PUNCT PROPN PROPN PROPN …
27146 PROPN PROPN PROPN PROPN ADP PROPN PROPN PROPN …
30864 PROPN PROPN PUNCT PUNCT NOUN PUNCT PUNCT PROPN…
23232 PROPN PROPN PUNCT PUNCT ADJ PART PROPN PROPN P…
Name: Text_pos, Length: 20755, dtype: object
```

```
#Initializing the TF-IDF Vectorizer:

tfidf_vectorizer = TfidfVectorizer(stop_words='english', ngram_range = (2,2))

# Fit and transform the training data

 tfidf_train = tfidf_vectorizer.fit_transform(x_train.astype('str'))

 # Transform the test set

tfidf_test = tfidf_vectorizer.transform(x_test.astype('str'))

#We use pickle so that we can again use the model without running it again. ␣
,→Pickle stores the model in memory which can be used later.

pickle.dump(tfidf_train, open("tfidf_train.pickle", "wb")) pickle.dump(tfidf_test,
open("tfidf_test.pickle", "wb"))
```

- **Applying all the classifiers coupled with NLP:**

#Applying Naive Bayes Algorithm:

```
clf_MNB_pos = MultinomialNB()
clf_MNB_pos.fit(tfidf_train, y_train)
pickle.dump(clf_MNB_pos, open('pos_nb', 'wb'))
pred = clf_MNB_pos.predict(tfidf_test)
score = metrics.accuracy_score(y_test, pred)
print("Accuracy with Multinomial Naive Bayes: %0.3f" % score)
```

Accuracy with Multinomial Naive Bayes: 0.795

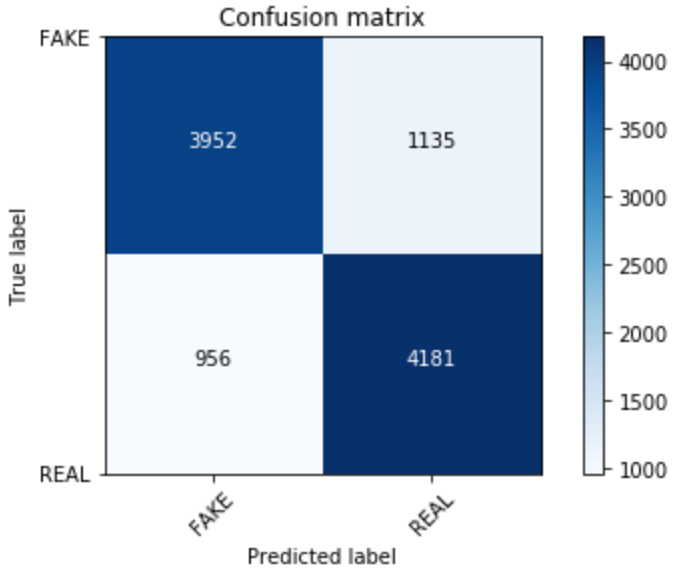

Fig. 3.1 Plotting confusion matrix for Naive-Bayes algorithm coupled with NLP

#Applying Random Forests Algorithm:

```
clf_RFC_pos = RandomForestClassifier()
clf_RFC_pos.fit(tfidf_train, y_train)
pickle.dump(clf_RFC_pos, open('pos_rf', 'wb'))
pred = clf_RFC_pos.predict(tfidf_test)
 score = metrics.accuracy_score(y_test, pred)
print("Accuracy with RandomForestClassifier: %0.3f" % score)
```

Accuracy with RandomForestClassifier: 0.870

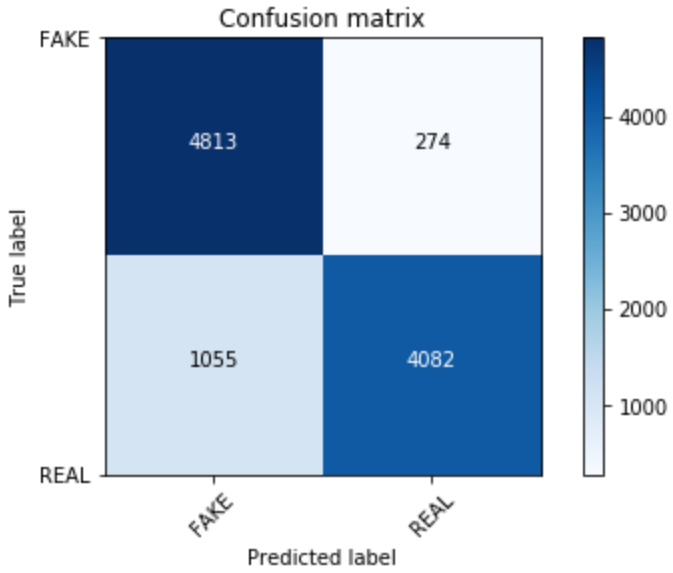

Fig. 3.2 Plotting confusion matrix for Random-Forests algorithm coupled with NLP

#Applying Gradient Boosting Algorithm:

```
clf_GBC_pos = GradientBoostingClassifier()
clf_GBC_pos.fit(tfidf_train, y_train)
 pickle.dump(clf_GBC_pos, open('pos_gb', 'wb')) 19
pred = clf_GBC_pos.predict(tfidf_test)
score = metrics.accuracy_score(y_test, pred)
print("Accuracy with Gradient Boosting: %0.3f" % score)
```

Accuracy with Gradient Boosting: 0.880

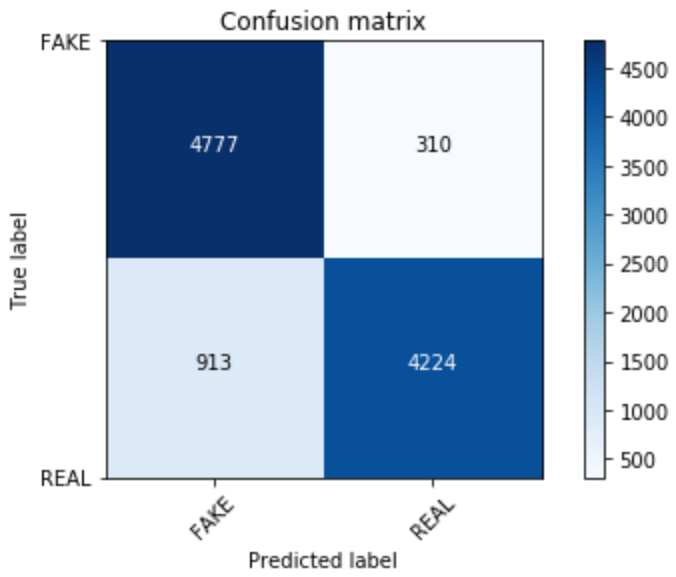

Fig. 3.3 Plotting confusion matrix for Gradient-Boosting algorithm coupled with NLP

#Applying Linear SVC Algorithm:

```
clf_svm_pos = svm.SVC(kernel='linear')
clf_svm_pos.fit(tfidf_train, y_train)
pickle.dump(clf_svm_pos, open('pos_svc', 'wb'))
pred = clf_svm_pos.predict(tfidf_test)
score = metrics.accuracy_score(y_test, pred)
print("Accuracy with SVC: %0.3f" % score)
```

Accuracy with SVC: 0.836

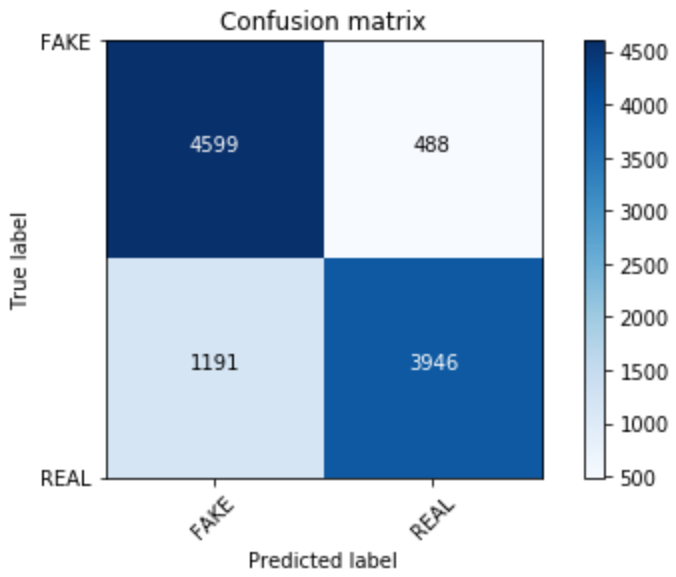

Fig. 3.4 Plotting confusion matrix for Linear-SVC algorithm coupled with NLP

- **Determining ROC curve for NLP technique along with classifiers:**

```
plt.figure(0).clf()
for model, name in [(clf_MNB_pos, 'Multinomial Naive Bayes '),
(clf_RFC_pos,'Random Forest␣ ,→Classifier'),(clf_GBC_pos,'Gradient
Boosting'),(clf_svm_pos, 'Support Vector␣ ,→classifier')
                                        ]:
 if name == 'Support Vector classifier':
        pred = model.decision_function(tfidf_test)
else:
        pred = model.predict_proba(tfidf_test)[:,1]
fpr, tpr, thresh = metrics.roc_curve(y_test.values, pred, pos_label='REAL')
plt.plot(fpr,tpr,label="{}".format(name))

plt.legend(loc=0)
```

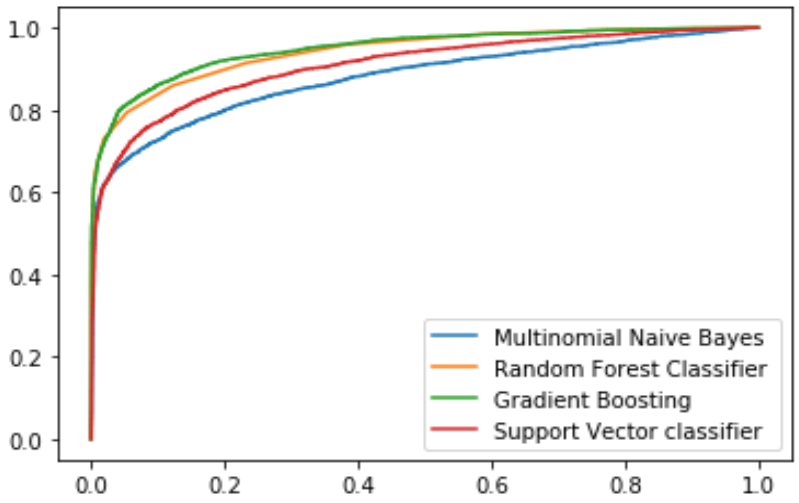

Fig. 3.5 ROC curve for displaying the performance of all the classifiers coupled with NLP

- **Merging Two analysis done above and applying on input data:**

```
tfidf_train_wt = tfidf_train
tfidf_test_wt = tfidf_test
tfidf1_train_wt = tfidf1_train
tfidf1_test_wt = tfidf1_test

#dif_row_tr is difference between no. of rows in syntax vector taining set and␣
,→bigram vector training set.
 dif_rows_tr = tfidf_train_wt.shape[0] - tfidf1_train_wt.shape[0]
X_ = sp.vstack((tfidf1_train_wt, sp.csr_matrix((dif_rows_tr, tfidf1_train_wt.
,→shape[1])))

# merging the syntax vector training set(POS tagging) and bigram
vector(TF-IDF␣ ,→bigrams)
training set. X_tr = sp.hstack((tfidf_train_wt, X_))

#dif_row_ts is difference between no. of rows in syntax vector test set and␣
,→bigram vector test set. 25

dif_rows_ts = tfidf_test_wt.shape[0] - tfidf1_test_wt.shape[0]
```

```
Y_ = sp.vstack((tfidf1_test_wt, sp.csr_matrix((dif_rows_ts, tfidf1_test_wt.
,→shape[1]))))

 # merging the POS tagging test set and TF-IDF test set.
X_ts = sp.hstack((tfidf_test_wt, Y_))
```

- **Applying classifiers coupled with merging analysis:**

#Applying Naive-Bayes Algorithm:

```
clf_MNB_tp = MultinomialNB()
clf_MNB_tp.fit(X_tr, y_train)
pickle.dump(clf_MNB_tp, open('tp_nb', 'wb'))
pred = clf_MNB_tp.predict(X_ts)
score = metrics.accuracy_score(y_test, pred)
print("Accuracy with Multinomial Naive Bayes: %0.3f" % score)
```

Accuracy with Multinomial Naive Bayes: 0.543

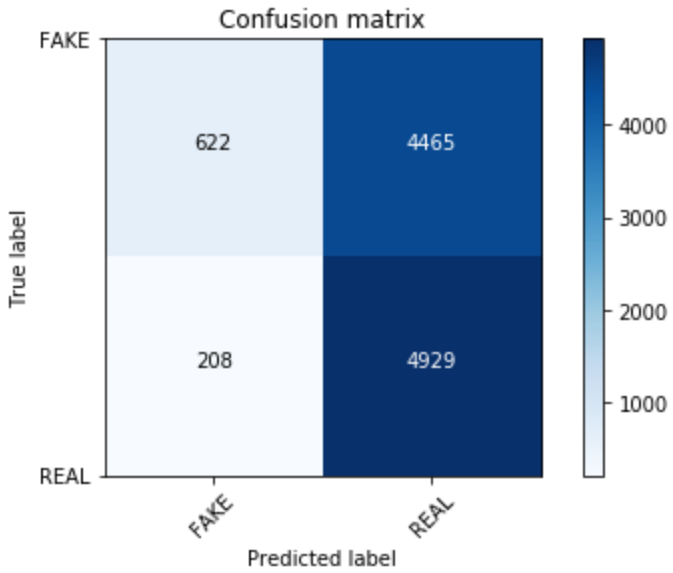

Fig. 4.1 Plotting confusion matrix for Naive-Bayes algorithm coupled with merging model

#Applying Random-Forests Algorithm:

```
clf_RFC_tp = RandomForestClassifier()
```

```
clf_RFC_tp.fit(X_tr, y_train)
pickle.dump(clf_RFC_tp, open('tp_rf', 'wb'))
pred = clf_RFC_tp.predict(X_ts)
score = metrics.accuracy_score(y_test, pred)
print("Accuracy with RandomForestClassifier: %0.3f" % score)
```

Accuracy with RandomForestClassifier: 0.739

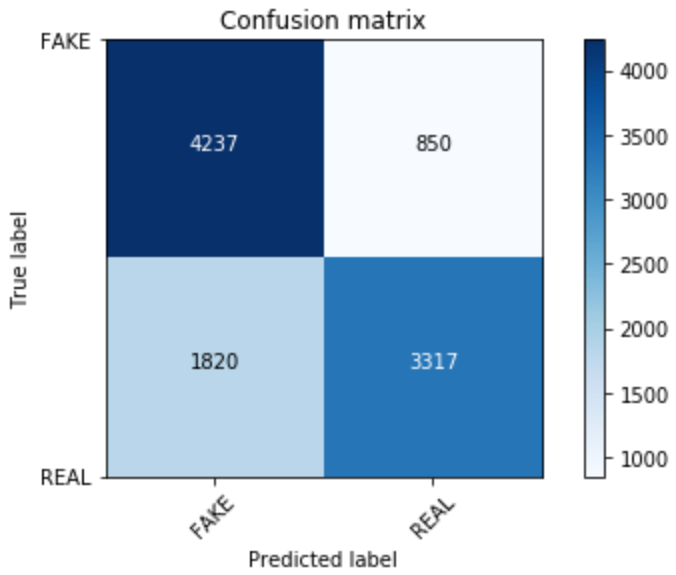

Fig. 4.2 Plotting confusion matrix Random-Forests algorithm coupled with merging model.

#Applying Gradient-Boosting Algorithm:

```
clf_GBC_tp = GradientBoostingClassifier()
clf_GBC_tp.fit(X_tr, y_train)
pickle.dump(clf_GBC_tp, open('tp_gb', 'wb'))
pred = clf_GBC_tp.predict(X_ts)
score = metrics.accuracy_score(y_test, pred)
print("Accuracy with Gradient Boosting: %0.3f" % score)
```

Accuracy with Gradient Boosting: 0.879

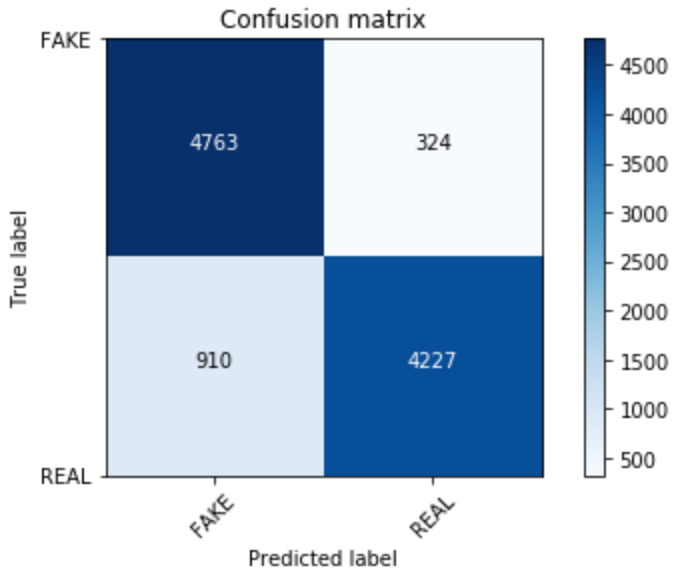

Fig. 4.3 Plotting confusion matrix for Gradient-Boosting algorithm coupled with merging model

#Applying Linear-SVC algorithm:

```
clf_svm_tp = svm.SVC(kernel='linear')
clf_svm_tp.fit(X_tr, y_train)
pickle.dump(clf_svm_tp, open('tp_svc', 'wb'))
pred = clf_svm_tp.predict(X_ts)
score = metrics.accuracy_score(y_test, pred)
print("Accuracy with SVC: %0.3f" % score)
```

Accuracy with SVC: 0.829

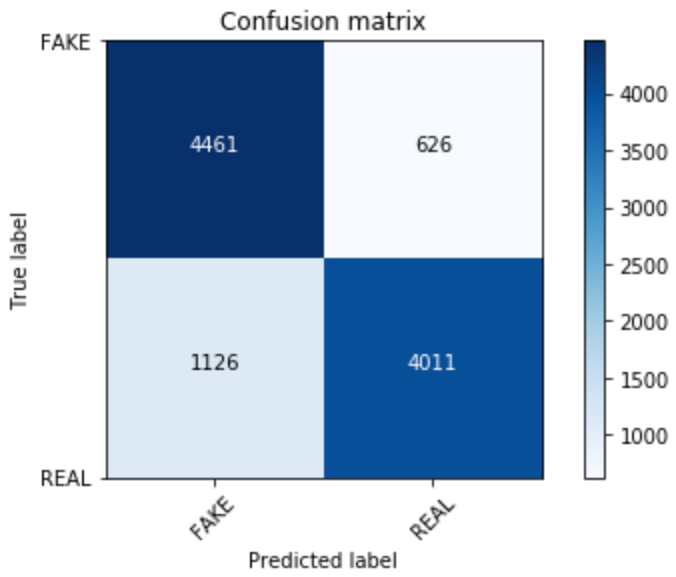

Fig. 4.4 Plotting confusion matrix for Linear-SVC algorithm coupled with merging model

- **Determining ROC curve for Merging technique along with classifiers:**

```
plt.figure(0).clf()
for model, name in [(clf_MNB_tp, 'Multinomial Naive Bayes '),
(clf_RFC_tp,'Random Forest␣ ,→Classifier'),(clf_GBC_tp,'Gradient
Boosting'),(clf_svm_tp, 'Support Vector␣ ,→classifier')
                                    ]:
if name == 'Support Vector classifier':
        pred = model.decision_function(X_ts)
else:
pred = model.predict_proba(X_ts)[:,1]
fpr, tpr, thresh = metrics.roc_curve(y_test.values, pred, pos_label='REAL')
plt.plot(fpr,tpr,label="{}".format(name))

plt.legend(loc=0)
```

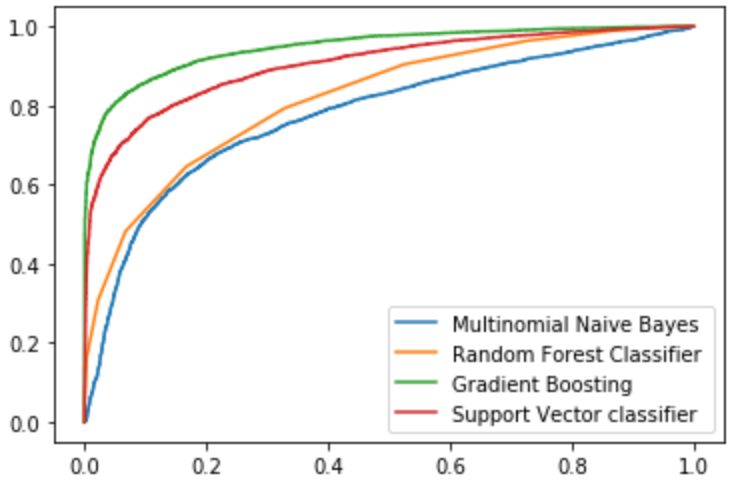

Fig. 4.5 ROC curve for displaying the performance of all the classifiers coupled with the merging model.

CONCLUSION

We have used three approaches in our project to increase the accuracy of fake news detection-

1. Applying four classifiers only with TF-IDF Vectorizer
2. Applying NLP on input data set before using the TF-IDF Vectorizer and then using four classifiers on resultant data set
3. Merging above two techniques to train the input data set and then using four classifiers

In the first approach,
We observed, among all the four classifiers(Multinomial Naive-Bayes, Random-Forest, Gradient-Boosting and Linear SVC), Linear SVC gives the most accurate prediction with 94.2% accuracy followed by Random-Forest with 87.6%, Gradient-Boosting with 84.8% and Multinomial Naive-Bayes with 82.5% accuracy.

In the second one,
We applied the same four classifiers but coupled with Natural Language Processing technique. This time,Gradient Boosting topped the accuracy of fake news detection with 88% accuracy with Random-Forest,Linear SVC and Multinomial Naive-Bayes clocking 87%,83.6% and 79.5 % accuracy.

In the last one,
We merged the two models mentioned above and used it to train the input data set before applying the four classifiers on the resultant data set. This time also, the Gradient-Boosting algorithm performed best with 87.9 % accuracy. Linear SVC, Random-Forest and Multinomial Naive-Bayes algorithm followed Gradient Boosting with 82.9%,73.9% and 54.3% accuracy.

From the three approaches used above,
We can see that Gradient Boosting has performed best in two of the three cases we applied, with 88% and 87.9% accuracy. It is also clear from the accuracy percentage, that Gradient Boosting algorithm has been the most consistent one in differentiating the real and fake news, which is also a major deciding factor in detecting fake news.

Therefore, we can deduce from the approaches used, that Gradient Boosting algorithm with Natural Language Processing is the best fit for our project to differentiate real and fake news. And hence we select it as the Final Selection Model to detect fake news.

FUTURE PROSPECTS

This project involves designing and implementing a facility that can distinguish between fake and real news, by learning using linguistic cues. The essentiality of such an application software has prevailed for quite some time now and is the need of the hour, owing to the political disruptions, religious polarization and other mishaps in the society that find their root in misinformation. This very fact has been the propelling force for our project.

Of the 12 months of planning, research study and experimentation with data, we have managed to yield twelve mathematical models using four classification algorithms to suit our data sets and label any new instance as “real” or “fake”. However, there is always room for improvement of the accuracy and expansion of the use cases. The effectiveness of the software could be enhanced if the following can be realised in future:

- Involving advanced elements of semantic analysis for better extraction of linguistic cues.
- Introducing sentiment analysis for better discourse analysis.
- Using better techniques of feature engineering for optimized data preparation upon availability of faster hardware resources.
- Accepting feedback from users who might rely on multiple fact checkers to verify information, as to whether the prediction is correct. This feedback can be used to increase the training data size and hence obtain a better fitting model.
- Tracing the origin of generation of fake news so that it can be used in cyber governance.
- Incorporate facilities for live detection and restraint from fake news.

Enhancement of the model is a continuous procedure. Detecting the falsity of news on the go would depend a lot on the authenticity of their sources. Also, the prevalent state of affairs would play a handsome role in framing this provision. With streamlined efforts on this domain and development of more efficient tools of classification, this effort could potentially mature into the answer to “Fake News Epidemic”.

REFERENCES

[1] K. Shu, A. Sliva, S. Wang, J. Tang, and H. Liu, “Fake news detection on social media: A data mining perspective,” CoRR, vol. abs/1708.01967, 2017.

[2]T1 - Fake news detection using naive Bayes classifier DO-10.1109/ UKRCON.2017.8100379.

[3]S. Gilda, "Notice of Violation of IEEE Publication Principles: Evaluating machine learning algorithms for fake news detection," *2017 IEEE 15th Student Conference on Research and Development (SCOReD)*, Putrajaya, 2017, pp. 110-115.
doi: 10.1109/SCORED.2017.8305411

WEBSITES REFERENCES USED:
1.https://www.tutorialspoint.com/python/index.htm
2.https://stackoverflow.com/
3.https://scikit-learn.org/stable/
4.https://www.kaggle.com/
5.https://www.datacamp.com/community/tutorials/scikit-learn-fake-news