Multiple Features Based Approach for Automatic Fake News Detection on Social Networks

Project Report

Machine Learning (CS351)

Department of Computer Science and Engineering

National Institute of Technology Karnataka, Surathkal

Done By:

Mohammed Rushad
(mohammedrushad@gmail.com)
181CO232

Akshat Nambiar
(akshat1804@gmail.com)
181CO204

Abstract

With widespread adoption of internet use and social media websites like Facebook, the news that people obtain from these sources become very important factors in influencing people’s opinions. This is why fake news articles can cause a major problem and their spread needs to be kept in check. Fake news is defined as fictitious articles deliberately fabricated to deceive readers. Untrustworthy social media and news outlets publish fake news to increase viewership or to influence public opinion. This report details the work done as part of our Machine Learning course for detecting and classifying fake news.

1. Introduction

With the spread of online social networks with their ease of use, faster transformation and less expensive nature, it has become a significant way of communication and information sharing. These days, almost all the social network users access the news through online channels. And because of this increasing popularity of social networks, its use has become a target for communication and spreading of fake news. The spreading of fake news is done in the form of misleading content, fake reviews, rumors, advertisements, false speech content for politicians, satires, hate speech etc. Currently, fake news is spread faster in social media rather than mainstream media. To tackle this problem, we need efficient and effective ways to detect and classify fake news.

One of the major approaches in this regard is building and using large datasets in machine learning and deep learning algorithms to train models to detect fake news. Traditionally, this method is implemented using the title and text of the article in question. This method has proved to be largely effective but still leaves a lot of room for improvement. The paper [1], improves on this traditional method by using features beyond the news title and text and incorporating features like user profile id, name, friends etc. By including user features in training the model to detect fake news, the authors obtain improved accuracy compared to the traditional method.

In our approach, we have chosen to incorporate article performance features like number of replies, shares likes, comments, spam score etc. along with the article title and text using the dataset in [2]. This approach also uses multiple features beyond the traditional approach and is shown to provide an improvement in performance and accuracy.

2. Dataset Description

The dataset [2] has many features giving information about the news article that was crawled. We use a subset of these features and have divided them into the following categories.

1. **News Features**:

	Feature	**Description**
1	Title	Title of the news article
2	Text	Content of the news article
3	Author	Author of the news article

2. **Article Performance Features**:

	Feature	**Description**
1	Shares	Number of times article has been shared
2	Comments	Number of comments
3	Likes	Number of likes
4	Participants Count	Total number of participants engaged in the article
5	Replies Count	Total number of replies
6	Spam Score	Score given to article

3. **Miscellaneous Features**

	Feature	**Description**
1	Country	Country of origin of article
2	URL	Website URL of the article

These make a total of 11 features for assessing the quality of the news article, with the focus here being on the Article Performance Features.

Each of these articles can initially be classifies into 8 labels: *state, satire, junkSci, hate, fake, conspiracy, bs, bias*. But for the classification we have mapped the data as:

▪ *state, satire, junkSci, hate, bias*	--->	1
▪ *fake, conspiracy, bs*	--->	0

3. Models Used and Tested

3.1 Models Implemented

Several algorithms were implemented using machine learning and deep learning as shown below:

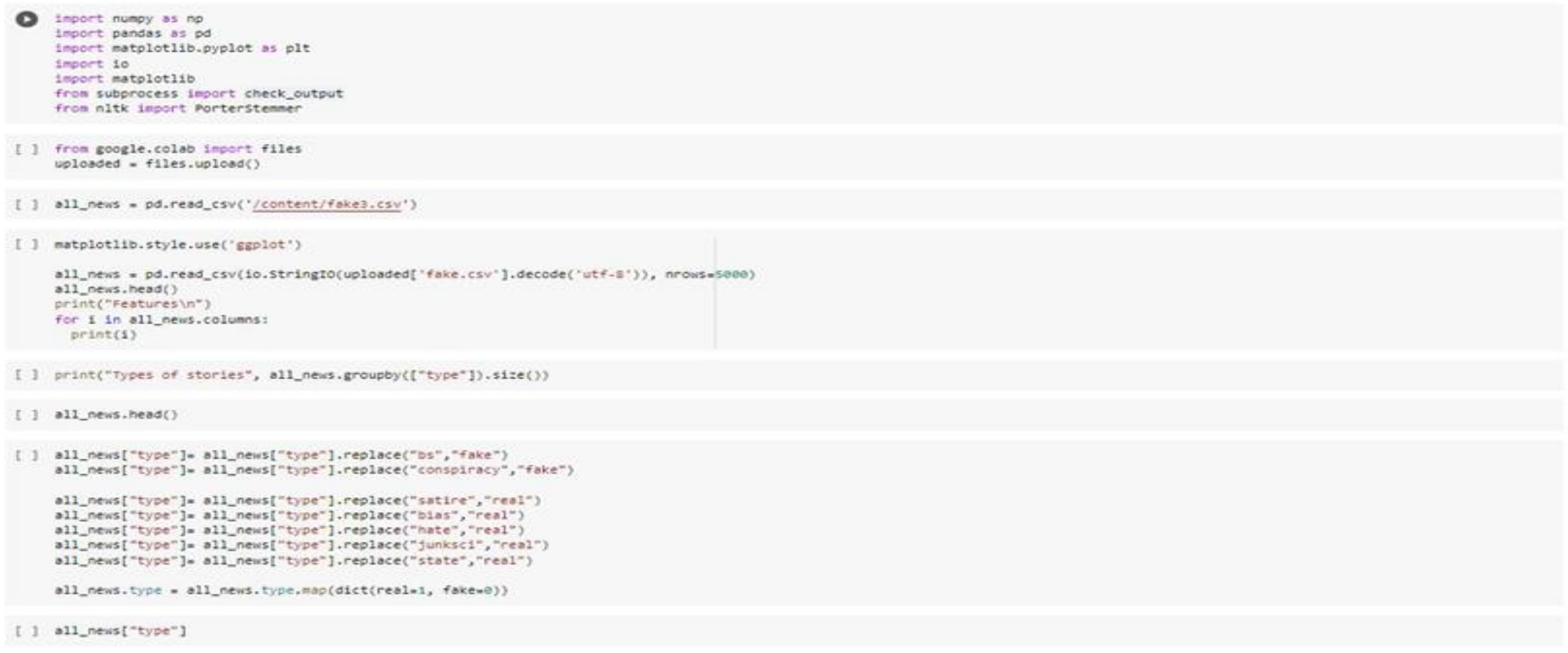
- Machine Learning
  - Logistic Regression
  - KNN
  - SVM
  - Decision Tree (Using gini Index)
  - Naïve Bayes
- Deep Learning
  - LSTM

3.2 Implementation

Overview:

1) **Import Data** – Importing dataset with the article performance and news article features
2) **Replace Labels and Data Pre-processing -** Appropriately classifying news articles as real and fake and filling in null values. Concatenate the text and title of the news article
3) **NLP Pre-processing –** Using CountVectorizer from sklearn to tokenize the words after removing punctuation, stopwords, converting to lowercase etc. This is done on all text-based columns of the dataset.
4) **Preparing Final Dataset –** Combining the numerical and vectorized text columns into a new dataset.
5) **Splitting the Dataset into Test-Train Data -** Splitting the data appropriately into testing and training data to provide as input to the models.
6) **Implement ML and Deep Learning Models –** Using machine learning models to classify the test data, after training the models with the training data. Models like Naïve-Bayesian, Decision Tree (Gini), SVM, Logistic Regression and KNN are used. Using Deep Learning Model LSTM (Long Short-term Memory)

Code:



```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import io
import matplotlib
from subprocess import check_output
from nltk import PorterStemmer
```

```
[ ] from google.colab import files
    uploaded = files.upload()
```

```
[ ] all_news = pd.read_csv('/content/fake3.csv')
```

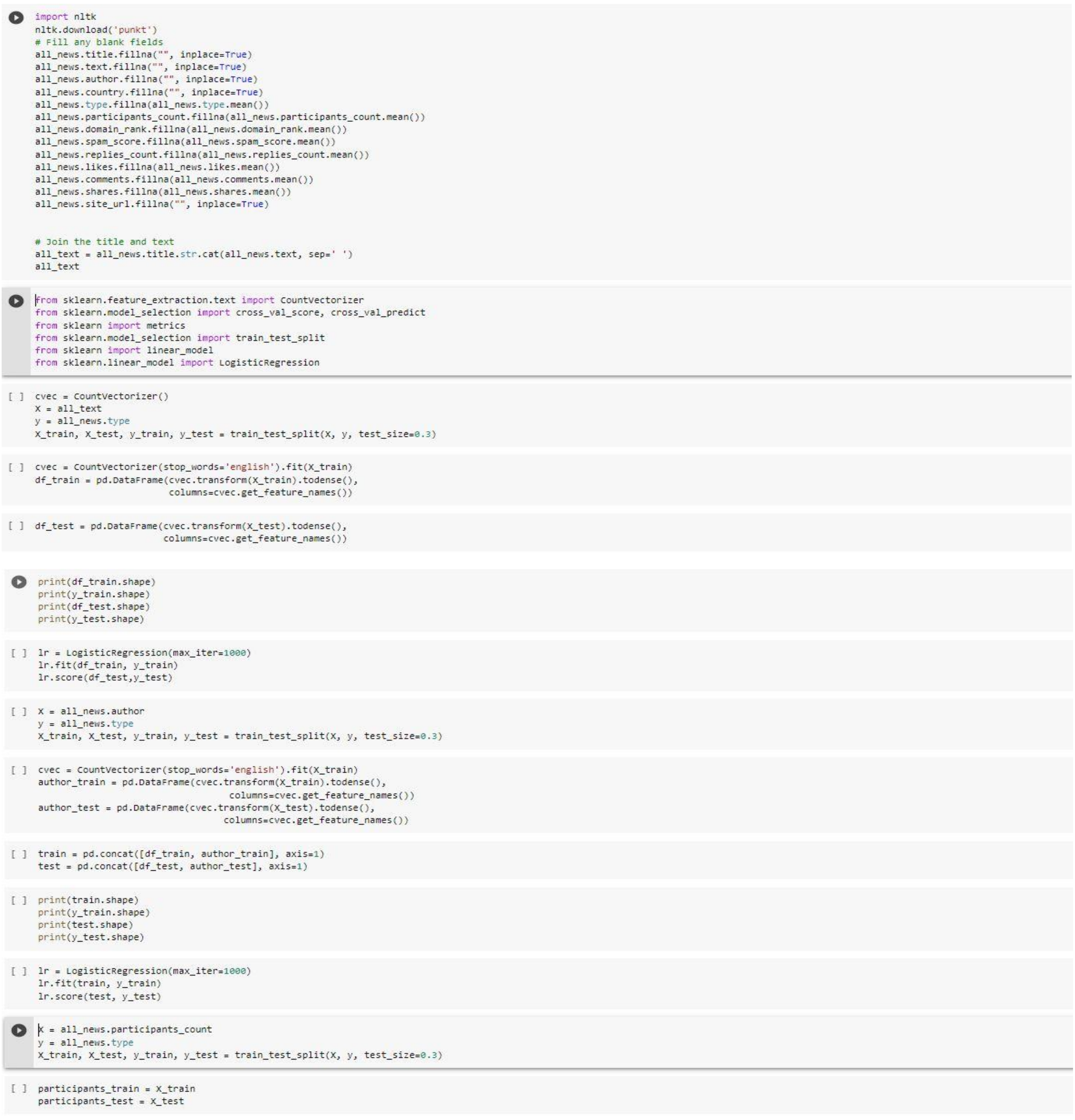
```
[ ] matplotlib.style.use('ggplot')

    all_news = pd.read_csv(io.StringIO(uploaded['fake.csv'].decode('utf-8')), nrows=5000)
    all_news.head()
    print("Features\n")
    for i in all_news.columns:
      print(i)
```

```
[ ] print("Types of stories", all_news.groupby(["type"]).size())
```

```
[ ] all_news.head()
```

```
[ ] all_news["type"]= all_news["type"].replace("bs","fake")
    all_news["type"]= all_news["type"].replace("conspiracy","fake")

    all_news["type"]= all_news["type"].replace("satire","real")
    all_news["type"]= all_news["type"].replace("bias","real")
    all_news["type"]= all_news["type"].replace("hate","real")
    all_news["type"]= all_news["type"].replace("junksci","real")
    all_news["type"]= all_news["type"].replace("state","real")

    all_news.type = all_news.type.map(dict(real=1, fake=0))
```

```
[ ] all_news["type"]
```



```
import nltk
nltk.download('punkt')
# Fill any blank fields
all_news.title.fillna("", inplace=True)
all_news.text.fillna("", inplace=True)
all_news.author.fillna("", inplace=True)
all_news.country.fillna("", inplace=True)
all_news.type.fillna(all_news.type.mean())
all_news.participants_count.fillna(all_news.participants_count.mean())
all_news.domain_rank.fillna(all_news.domain_rank.mean())
all_news.spam_score.fillna(all_news.spam_score.mean())
all_news.replies_count.fillna(all_news.replies_count.mean())
all_news.likes.fillna(all_news.likes.mean())
all_news.comments.fillna(all_news.comments.mean())
all_news.shares.fillna(all_news.shares.mean())
all_news.site_url.fillna("", inplace=True)


# Join the title and text
all_text = all_news.title.str.cat(all_news.text, sep=' ')
all_text
```

```
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.model_selection import cross_val_score, cross_val_predict
from sklearn import metrics
from sklearn.model_selection import train_test_split
from sklearn import linear_model
from sklearn.linear_model import LogisticRegression
```

```
[ ] cvec = CountVectorizer()
    X = all_text
    y = all_news.type
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3)
```

```
[ ] cvec = CountVectorizer(stop_words='english').fit(X_train)
    df_train = pd.DataFrame(cvec.transform(X_train).todense(),
                            columns=cvec.get_feature_names())
```

```
[ ] df_test = pd.DataFrame(cvec.transform(X_test).todense(),
                           columns=cvec.get_feature_names())
```

```
print(df_train.shape)
print(y_train.shape)
print(df_test.shape)
print(y_test.shape)
```

```
[ ] lr = LogisticRegression(max_iter=1000)
    lr.fit(df_train, y_train)
    lr.score(df_test,y_test)
```

```
[ ] X = all_news.author
    y = all_news.type
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3)
```

```
[ ] cvec = CountVectorizer(stop_words='english').fit(X_train)
    author_train = pd.DataFrame(cvec.transform(X_train).todense(),
                                columns=cvec.get_feature_names())
    author_test = pd.DataFrame(cvec.transform(X_test).todense(),
                               columns=cvec.get_feature_names())
```

```
[ ] train = pd.concat([df_train, author_train], axis=1)
    test = pd.concat([df_test, author_test], axis=1)
```

```
[ ] print(train.shape)
    print(y_train.shape)
    print(test.shape)
    print(y_test.shape)
```

```
[ ] lr = LogisticRegression(max_iter=1000)
    lr.fit(train, y_train)
    lr.score(test, y_test)
```

```
X = all_news.participants_count
y = all_news.type
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3)
```

```
[ ] participants_train = X_train
    participants_test = X_test
```

```
participants_train = X_train
participants_test = X_test
```

```
participants_test.reset_index(drop=True, inplace=True)
participants_train.reset_index(drop=True, inplace=True)
test.reset_index(drop=True, inplace=True)
train.reset_index(drop=True, inplace=True)
```

```
train = pd.concat([participants_train, df_train, author_train], axis=1)
test = pd.concat([participants_test, df_test, author_test], axis=1)
```

```
print(train.shape)
print(y_train.shape)
print(test.shape)
print(y_test.shape)
```

```
lr = LogisticRegression(max_iter=800)
lr.fit(train, y_train)
lr.score(test, y_test)
```

```
X = all_news.country
y = all_news.type
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3)
```

```
cvec = CountVectorizer(stop_words='english').fit(X_train)
country_train = pd.DataFrame(cvec.transform(X_train).todense(),
                              columns=cvec.get_feature_names())
country_test = pd.DataFrame(cvec.transform(X_test).todense(),
                             columns=cvec.get_feature_names())
```

```
train = pd.concat([country_train, participants_train, df_train, author_train], axis=1)
test = pd.concat([country_test, participants_test, df_test, author_test], axis=1)
```

```
print(train.shape)
print(y_train.shape)
print(test.shape)
print(y_test.shape)
```

```
lr = LogisticRegression(max_iter=800)
lr.fit(train, y_train)
lr.score(test, y_test)
```

```
X = pd.concat([all_news.spam_score, all_news.replies_count, all_news.likes, all_news.comments, all_news.shares],axis=1)
X.shape
```

```
X
```

```
y = all_news.type
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3)
```

```
X_train.head()
```

```
rest_train = X_train
rest_test = X_test
```

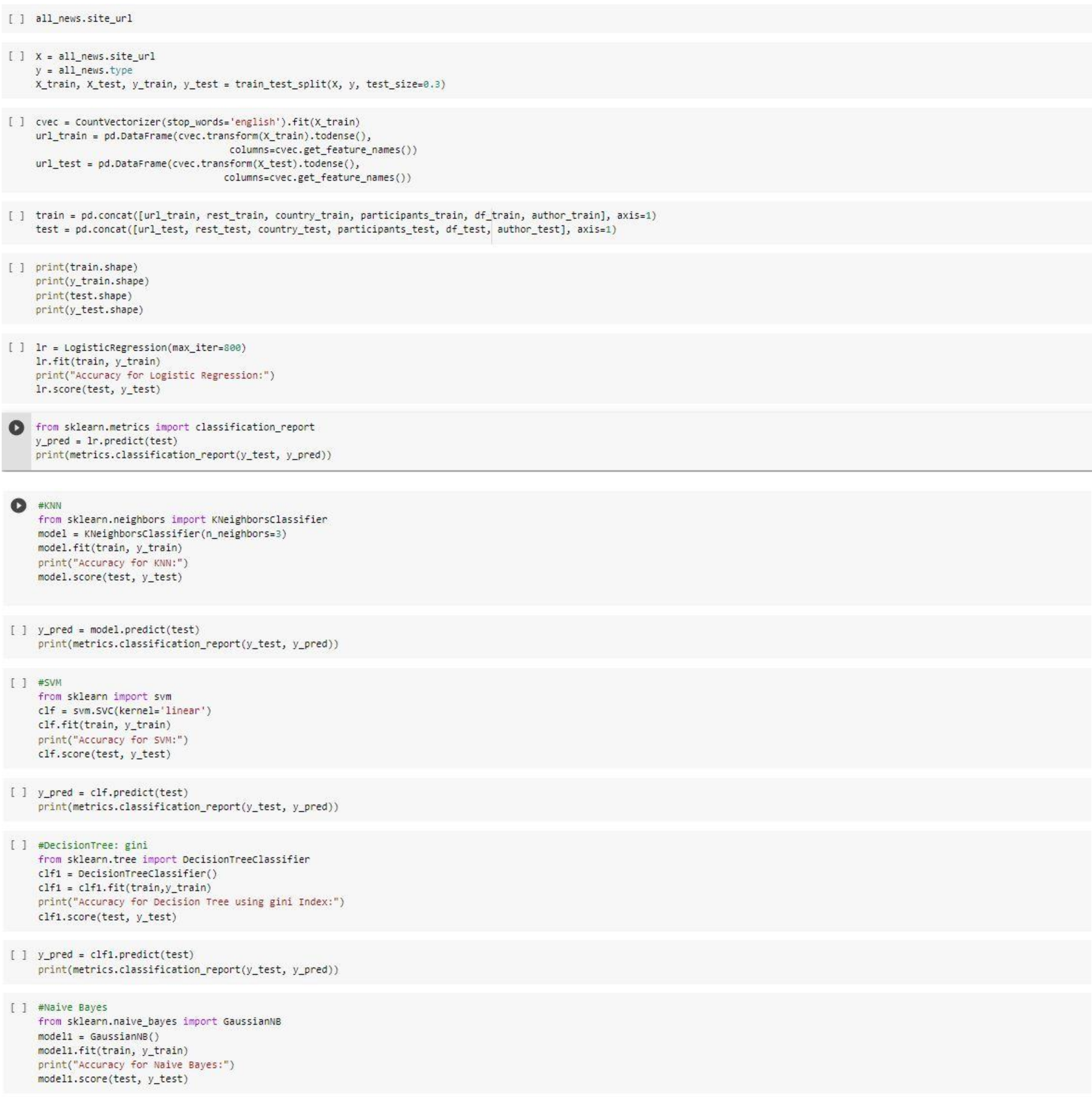
```
rest_test.reset_index(drop=True, inplace=True)
rest_train.reset_index(drop=True, inplace=True)
test.reset_index(drop=True, inplace=True)
train.reset_index(drop=True, inplace=True)
```

```
rest_train.head()
```

```
train = pd.concat([rest_train, country_train, participants_train, df_train, author_train], axis=1)
test = pd.concat([rest_test, country_test, participants_test, df_test, author_test], axis=1)
```

```
print(train.shape)
print(y_train.shape)
print(test.shape)
print(y_test.shape)
```

```
lr = LogisticRegression(max_iter=800)
lr.fit(train, y_train)
lr.score(test, y_test)
```



```
all_news.site_url
```

```
X = all_news.site_url
y = all_news.type
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3)
```

```
cvec = CountVectorizer(stop_words='english').fit(X_train)
url_train = pd.DataFrame(cvec.transform(X_train).todense(),
                                  columns=cvec.get_feature_names())
url_test = pd.DataFrame(cvec.transform(X_test).todense(),
                                 columns=cvec.get_feature_names())
```

```
train = pd.concat([url_train, rest_train, country_train, participants_train, df_train, author_train], axis=1)
test = pd.concat([url_test, rest_test, country_test, participants_test, df_test, author_test], axis=1)
```

```
print(train.shape)
print(y_train.shape)
print(test.shape)
print(y_test.shape)
```

```
lr = LogisticRegression(max_iter=800)
lr.fit(train, y_train)
print("Accuracy for Logistic Regression:")
lr.score(test, y_test)
```

```
from sklearn.metrics import classification_report
y_pred = lr.predict(test)
print(metrics.classification_report(y_test, y_pred))
```

```
#KNN
from sklearn.neighbors import KNeighborsClassifier
model = KNeighborsClassifier(n_neighbors=3)
model.fit(train, y_train)
print("Accuracy for KNN:")
model.score(test, y_test)
```

```
y_pred = model.predict(test)
print(metrics.classification_report(y_test, y_pred))
```

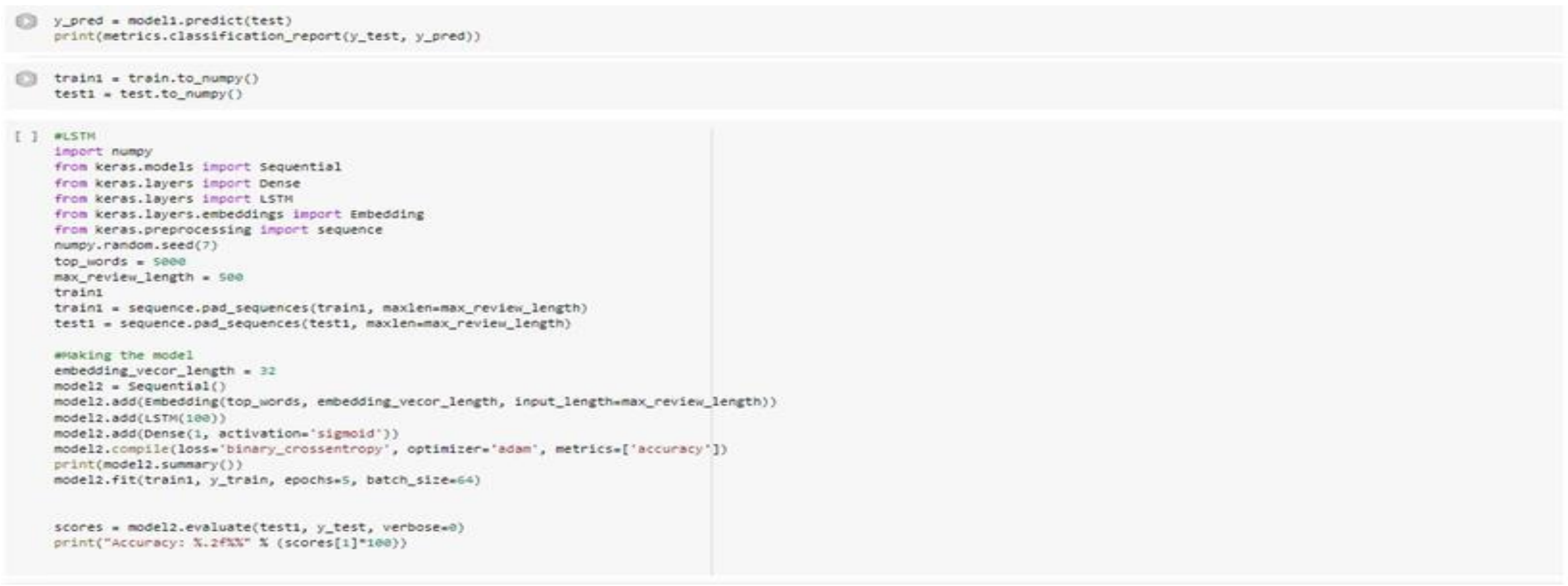
```
#SVM
from sklearn import svm
clf = svm.SVC(kernel='linear')
clf.fit(train, y_train)
print("Accuracy for SVM:")
clf.score(test, y_test)
```

```
y_pred = clf.predict(test)
print(metrics.classification_report(y_test, y_pred))
```

```
#DecisionTree: gini
from sklearn.tree import DecisionTreeClassifier
clf1 = DecisionTreeClassifier()
clf1 = clf1.fit(train,y_train)
print("Accuracy for Decision Tree using gini Index:")
clf1.score(test, y_test)
```

```
y_pred = clf1.predict(test)
print(metrics.classification_report(y_test, y_pred))
```

```
#Naive Bayes
from sklearn.naive_bayes import GaussianNB
model1 = GaussianNB()
model1.fit(train, y_train)
print("Accuracy for Naive Bayes:")
model1.score(test, y_test)
```



```
y_pred = model1.predict(test)
print(metrics.classification_report(y_test, y_pred))
```

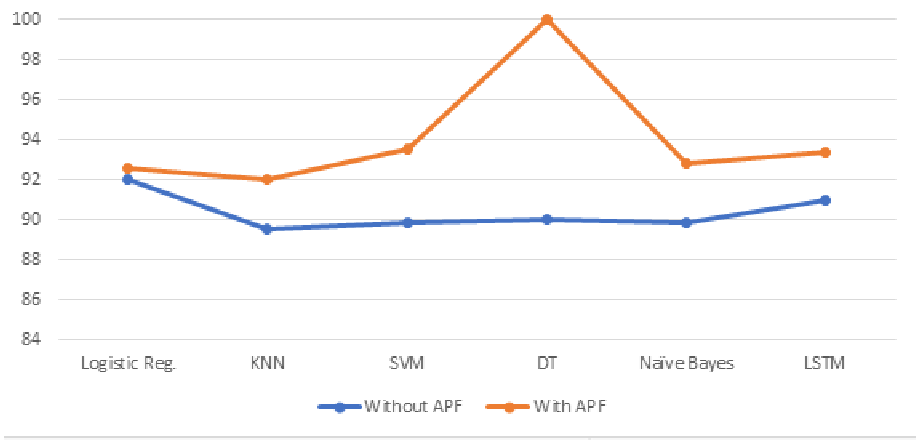
```
train1 = train.to_numpy()
test1 = test.to_numpy()
```

```
[ ] #LSTM
import numpy
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import LSTM
from keras.layers.embeddings import Embedding
from keras.preprocessing import sequence
numpy.random.seed(7)
top_words = 5000
max_review_length = 500
train1
train1 = sequence.pad_sequences(train1, maxlen=max_review_length)
test1 = sequence.pad_sequences(test1, maxlen=max_review_length)

#Making the model
embedding_vecor_length = 32
model2 = Sequential()
model2.add(Embedding(top_words, embedding_vecor_length, input_length=max_review_length))
model2.add(LSTM(100))
model2.add(Dense(1, activation='sigmoid'))
model2.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
print(model2.summary())
model2.fit(train1, y_train, epochs=5, batch_size=64)


scores = model2.evaluate(test1, y_test, verbose=0)
print("Accuracy: %.2f%%" % (scores[1]*100))
```

4. Results:

In our experimentation and analysis, using Article Performance Features (APF) along with News Features has shown to boost accuracy in all the algorithms implemented. The results have been summarized in the following table:

Algorithm		Accuracy	
		Without AFP (%)	With AFP (%)
Machine Learning			
1	Logistic Regression	92.00	92.66
2	KNN	89.50	92.00
3	SVM	89.83	93.50
4	Decision Tree (gini Index)	90.00	100.00
5	Naïve Bayes	89.83	92.83
Deep Learning			
1	LSTM	91.00	93.33



Hence, the incorporation of article performance features (APF) gives an increase in accuracy in all implemented models.

5. Contributions and Conclusion:

In this report we have summarized the results of using article performance features (APF) along with news features for detecting and classifying fake news. The method implemented by us is unique in its inclusion of APF as shown in section 2. Several models were implemented (Logistic Regression, KNN, SVM, DT, Naive Bayes, LSTM) and tested for accuracy. Our addition of the APF has shown to provide an increase in accuracy for all the models mentioned above as shown in Section 4. Hence, this incorporation of APF is shown to be a viable method for efficient fake news detection.

6. Future work:

The following can be explored further:

- Incorporate User + News + Article Performance features
- Incorporate Data from different social networks
- Incorporate model into web-extensions for easier article tagging.

References

[1] Somya Ranjan Sahoo, B.B. Gupta, Multiple features based approach for automatic fake news detection on social networks using deep learning, https://doi.org/10.1016/j.asoc.2020.106983.

[2] Kaggle: https://www.kaggle.com/mrisdal/fake-news