Model Development Document - Fake News Classifier

By Simon Lowry

Student number: D20129695

Contents

Project Overview

1.1 Project Objective

The purpose of this project is to create and establish a solution which can be used to classify fake news. A number of different models were applied to the training and test data sets in order to establish which would be the best at predicting fake news.

1.2 Outline of Solution

The approach for the solution was to initially analyze the dataset, then to assess it's quality, clean the data where needed, merge in the submit.csv labels for the test data. After this, normalization techniques were applied to help with getting the data in a condition which will help optimize the performance of the models. Thereafter, vectorization was applied gaining further insight into the dataset and words which had a high value to the dataset. This was followed up by applying three different models to perform predictions on the data and then the results were assessed and conclusions drawn from the results. Finally, a section was drawn up to outline some further work that could be done to improve the solution.

The code for this work was applied in a jupyter notebook and some of the libraries that were used were sklearn, pandas, spacy, re (regular expression), and the time library as well.

2 Solutions Development

This section will outline the approach that will be taken for your project.

2.1 Data Sample

2.1.1 Data Sample Cohort Definition

Define the data sample you are using for analysis in terms of:

- **Sources of the data**

The data source for the Fake news solution I'm looking to create is from kaggle:

https://www.kaggle.com/c/fake-news/data

The data is made up of three csv files, containing the training data, test data and then another csv file which has the test labels, which are the actual results for the test set in binary format, 1 being positive and 0 being negative.

- **Number of rows, columns**

Details about the training dataset and it's columns:

Columns: 5
Rows: 20800

```
In [19]: dataframe.info()
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 20800 entries, 0 to 20799
Data columns (total 5 columns):
 #   Column  Non-Null Count  Dtype 
---  ------  --------------  ----- 
 0   id      20800 non-null  int64 
 1   title   20242 non-null  object
 2   author  18843 non-null  object
 3   text    20761 non-null  object
 4   label   20800 non-null  int64 
dtypes: int64(2), object(3)
memory usage: 812.6+ KB
```

Three columns initially of type object and two columns of type object.

Details about the test data:

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 5193 entries, 0 to 5199
Data columns (total 4 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   id      5193 non-null   int64
 1   title   5071 non-null   object
 2   author  4697 non-null   object
 3   text    5193 non-null   object
dtypes: int64(1), object(3)
memory usage: 202.9+ KB
```

As we can see here the test data csv does not directly contain the labels which show whether it's fake news or not. That's in another csv (submit.csv).

```
Test set Number of Rows and columns (Rows, Columns):

(5200, 4)
```

- **Specific details about when the sample is from or is it from a specific population.**

Sample output from the training dataset:

```
In [8]: dataframe.head()
```

Out[8]:

	id	title	author	text	label
0	0	House Dem Aide: We Didn't Even See Comey's Let...	Darrell Lucus	House Dem Aide: We Didn't Even See Comey's Let...	1
1	1	FLYNN: Hillary Clinton, Big Woman on Campus - ...	Daniel J. Flynn	Ever get the feeling your life circles the rou...	0
2	2	Why the Truth Might Get You Fired	Consortiumnews.com	Why the Truth Might Get You Fired October 29, ...	1
3	3	15 Civilians Killed In Single US Airstrike Hav...	Jessica Purkiss	Videos 15 Civilians Killed In Single US Airstr...	1
4	4	Iranian woman jailed for fictional unpublished...	Howard Portnoy	Print \nAn Iranian woman has been sentenced to...	1

2.1.2 Data Quality and Data Exclusions

Outline any data quality issues and how you addressed them

Some analysis showed that the main target feature which is the *text* parameter contains a host of null values.

```
In [64]: dataframe["text"].isnull().sum()
Out[64]: 39
```

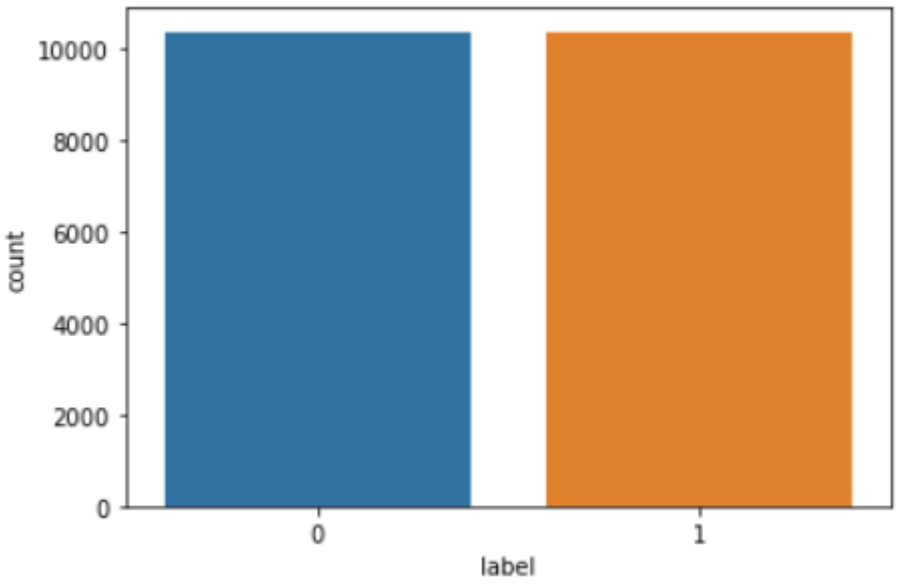
This could be an impediment to the accuracy of the results of the models that are intended to apply the data in. Since this is not a huge number, I am choosing to remove them from the data that will be used in the training set and not replace them and will do something similar in the test set if needed. There is still a large quantity of data to apply the models to in order to determine which is the most effective model to be used.

After dropping these 39 rows we're left with 20761 entries to apply to our models:

```
In [73]: d2 = dataframe.dropna(subset=['text'])
         d2.info()
<class 'pandas.core.frame.DataFrame'>
Int64Index: 20761 entries, 0 to 20799
Data columns (total 5 columns):
 #   Column  Non-Null Count  Dtype 
---  ------  --------------  ----- 
 0   id      20761 non-null  int64 
 1   title   20203 non-null  object
 2   author  18843 non-null  object
 3   text    20761 non-null  object
 4   label   20761 non-null  int64 
dtypes: int64(2), object(3)
memory usage: 973.2+ KB
```

Beyond this, the data has been normalized to try and get the data in the best possible state in order to obtain the best possible outcome for prediction models that are being applied. These techniques for doing this are outlined in the next section.

Distribution of Labels set to positive and negative (0 for negative, 1 for positive):



We see an even distribution of both the positive and negative cases of fake news.

2.1.2 Definition of Behavioral Features

List the features you are going to use for your analysis and definitions where the it is not obvious

The approach that will be used to normalize the data will include:

- removal of stopwords*
- removal of newlines, tabs and multiple spaces
- change all text to lowercase
- removal of all special characters and any other characters other alphanumeric characters

*stopwords are words which provide no semantic difference but add to the load of the computation needed to apply various models and as a result are best to be removed from the dataset.

2.2 Exploratory Analysis

Main Analysis Output

Analysis showed that there are a whole lot of different special characters in the data set as well as stopwords and punctuation marks and some potential uses of newlines as well as extra spacing between tokens. A lot of this data requires techniques to be applied to get it in a condition whereby the data quality has been normalized and is best fitted for applying the various models and obtaining the best results that we can obtain. The data was initially read in using the pandas libraries and placed in dataframes which provide the storage and capabilities for assessing the data and manipulating it as well. The assessment made use of a number of it's abilities to determine the best course of action for normalizing the data as well as manually reading the data from tools such as microsoft excel.

Also, as part of the analysis it became clear that the labels which denoted whether or not an entry in the test data was fake news or real was in a separate csv to the rest of the details of the test data. In order to make use of this data, the submit.csv was also ingested and it's column with the labels in it was added to the dataframe containing the test data. This was important for the later assessment of whether the prediction for the test data with the individual models was accurate or not.

Normalization

In order to normalize the data for the uses cases outlined above, the application is making use of a few different regexes to remove in conjunction with the sub method to remove characters which are not alphanumeric, remove newlines, any additional spaces and any undesired special characters.

Stopwords Removal

The application will make use of the library spacy which is a powerful natural language processing library that can be applied to a number of NLP tasks. In this instance it's being used to first illustrate the stopwords which it contains in it's English library as outlined below:

Spacy Stopwords:

```
In [18]: print(nlp.Defaults.stop_words)
```

```
{'amount', 'did', 'side', 'your', 'themselves', 'either', 'these', 'name', 'whose', 'see', 'could', 'this', 'ever', '‘ll',
"'d", 'which', 'had', 'well', 'through', 'less', '‘m', 'about', 'beforehand', "n't", 'still', 'why', '‘s', 'herein', 'for', 'fr
ont', 'to', 'throughout', '’m', 'somewhere', 'every', 'most', 'amongst', 'other', 'do', 'among', 'hence', 'because', 'some', 'i
ts', 'one', 'everyone', 'twenty', 'top', 'seeming', 'once', 'his', 'everything', 'full', 'mine', 'thereafter', 'off', 'are', 'w
ithout', 'due', 'everywhere', "'ve", 'very', 'quite', 'namely', 'sixty', 'also', 'take', 'anything', 'himself', 'an', 'they',
'‘re', 'herself', 'five', 'thus', 'us', 'has', 'toward', '’s', 'along', 'any', 'she', 'same', 'besides', 'by', 'seems', 'how',
'all', 'yourself', 'anyway', 'itself', 'wherein', 'few', 'ten', 'n‘t', 'yourselves', 'but', 'mostly', 'third', 'six', 'whereb
y', 'eight', 'anywhere', 'it', 'many', 'on', 'becomes', "'m", 'at', 'does', 'now', 'latterly', 'below', 'only', 'hers', 'much',
'enough', 'down', 'others', 'in', 'is', 'get', 'fifteen', 'here', 'under', 'above', 'except', 'he', 'before', 'into', 'became',
'whereafter', 'really', 'after', 'unless', 'although', 'out', 'latter', 'nowhere', 'a', 'him', 'forty', 'behind', 'three', 'ye
t', 'can', 'n’t', 'together', '‘d', 're', 'ours', 'something', "'s", 'call', 'however', 'been', 'hereby', 'the', 'indeed', 'alm
ost', 'bottom', 'whenever', 'otherwise', 'during', 'me', 'onto', 'whereupon', '’ve', 'that', 'rather', 'wherever', 'whither',
'doing', 'even', 'becoming', 'show', 'were', '’re', 'anyhow', 'whence', 'since', 'perhaps', 'fifty', 'beyond', 'across', 'there
in', '’d', 'or', 'again', 'while', 'each', 'meanwhile', 'neither', 'whole', 'always', 'between', 'regarding', 'back', 'anothe
r', 'seemed', 'being', 'nor', 'was', 'just', 'if', 'have', 'former', 'both', 'often', 'towards', 'you', 'never', 'thru', 'somet
imes', 'two', 'where', 'else', 'someone', 'her', 'moreover', 'more', 'when', 'beside', 'several', 'might', 'say', 'further', 'u
pon', 'used', 'afterwards', 'though', 'seem', 'nine', 'as', 'become', 'there', 'who', 'be', 'of', 'alone', 'with', 'against',
'none', 'would', 'formerly', 'whom', 'what', 'so', 'am', 'anyone', 'first', 'yours', "'ll", 'hereafter', '’ll', 'our', 'next',
'such', 'therefore', 'will', '‘ve', 'then', 'cannot', 'put', 'myself', 'somehow', 'give', 'around', 'nevertheless', 'own', 'hun
dred', 'too', 'hereupon', 'whereas', 'noone', 'my', 'move', 'over', 'thereupon', 'already', 'twelve', 'empty', 'must', 'keep',
'go', 'sometime', 'should', 'last', 'those', 'via', 'nobody', 'whether', 'least', 'may', 'whatever', 'part', 'from', 'whoever',
'ca', 'their', 'no', 'thence', 'four', 'using', 'within', 'done', 'thereby', 'elsewhere', 'them', 'ourselves', 'than', 'nothin
g', 'make', 'not', 'i', 'we', 'serious', 'up', 'eleven', 'made', 'per', 'please', 'until', 'and', "'re", 'various'}
```

These are removed from the dataset in both the training and test data sets respectively.

Vectorization

After getting the data into an appropriate condition and improving it's quality, the next step was to vectorize the data. For this step, the term frequency-inverse document frequency (TF-IDF) method was applied. Making use of the TF-IDF establishes a weighting which gives some insight into the importance of a given word in a document. This matters because it allows the application to determine which words have a high predictive value in ascertaining if a particular news article is real or fake.

TF-IDF formula:

$$W_{x,y} = tf_{x,y^*} \log\left(\frac{N}{df_x}\right)$$

$W_{x,y}$= Word x within document y

$tf_{x,y}$= frequency of x in y

df_x= number of documents containing x

N=total number of documents

The tool makes use of sklearns TfIdfVectorizer in order to perform this operation on the datasets.

Results and Conclusions

Document the main findings/impact of the work that you have done.

After the data was cleaned, normalized and vectorized it was now in a condition to then apply the various predictive models and ascertain the effectiveness of each of the models applied to the test data. Again the libraries of sklearn became an invaluable tool here which pre-packaged models which could be applied to the vectorized data in conjunction with the labels of both the training and test set.

The models that were applied were as follows:

- Naive Bayes
- Random Forrest
- Support Vector Machines

Each of the models were initially applied and trained on the training set data and then applied to the test set. A classification report was then output to display the accuracy, precision, recall and f1-score for the model applied on the test set. The classification reports showed that Random Forrest had the greatest effectiveness overall with these metrics. It scored better in all categories from accuracy to precision, recall and F1-Score. It obtained an overall score of 67%, to SVM's 64% and Naive Bayes' score of 58%. One thing to note is that while the Random Forrest algorithm performed 3% better than the SVM, it was also drastically faster than the SVM's model, which took a very long time to train. Naive Bayes was shown to be the fastest with only taking 0.12 seconds, this is due to the fact that this is a simpler model than the other two. Random Forrest took 25.89 seconds to complete which is significantly longer but not an unreasonable amount of time. SVM drastically lagged behind the other two taking a staggering

So, in conclusion, we were able to apply each of the models and the Random Forrest model was shown to be the most effective model choice.

Model Performance:

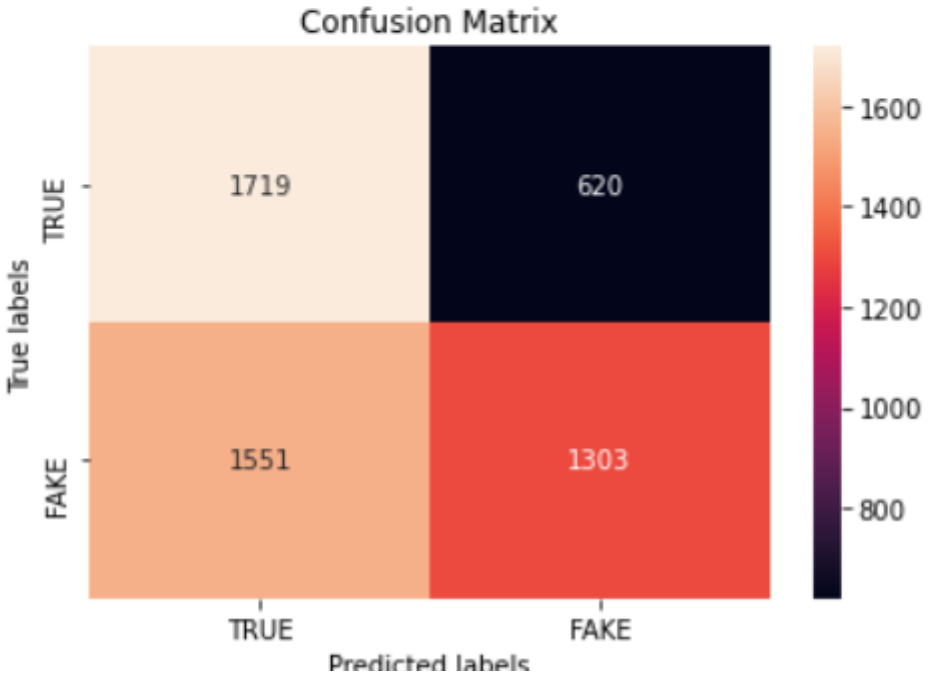
Naive Bayes:

```
Applying Naive Bayes Model:
----------------------------
accuracy:   0.582
              precision    recall  f1-score   support

    Positive       0.53      0.73      0.61      2339
    Negative       0.68      0.46      0.55      2854

    accuracy                           0.58      5193
   macro avg       0.60      0.60      0.58      5193
weighted avg       0.61      0.58      0.58      5193

Time taken to apply model: 0.262702 seconds
```



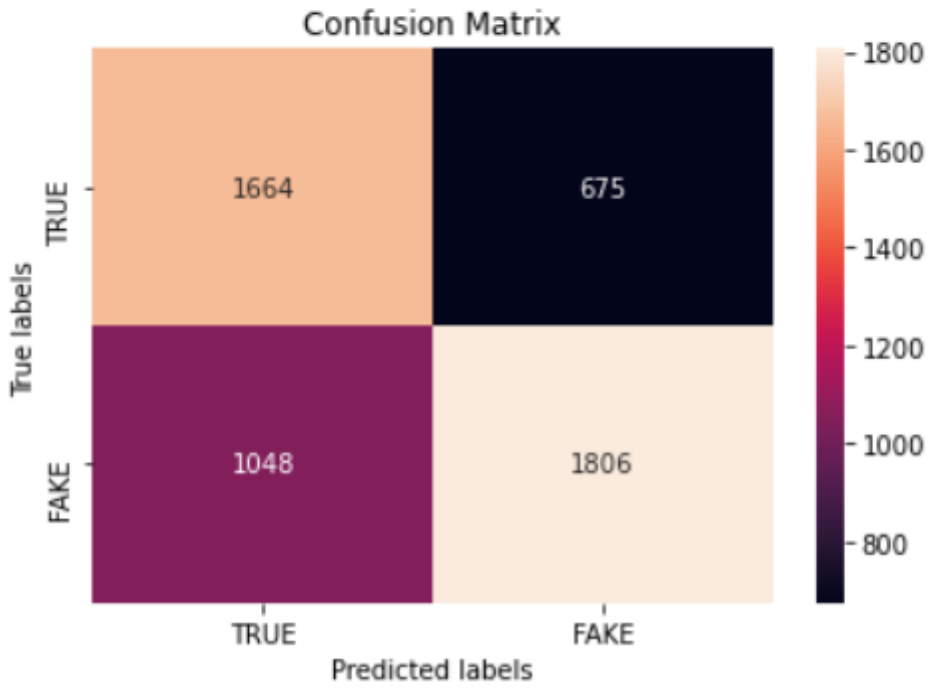
Random Forrest:

```
Applying Random Forrest Model:
-------------------------------
accuracy:   0.668
              precision    recall  f1-score   support

    Positive       0.61      0.71      0.66      2339
    Negative       0.73      0.63      0.68      2854

    accuracy                           0.67      5193
   macro avg       0.67      0.67      0.67      5193
weighted avg       0.68      0.67      0.67      5193

Time taken to apply model: 36.406860 seconds
```



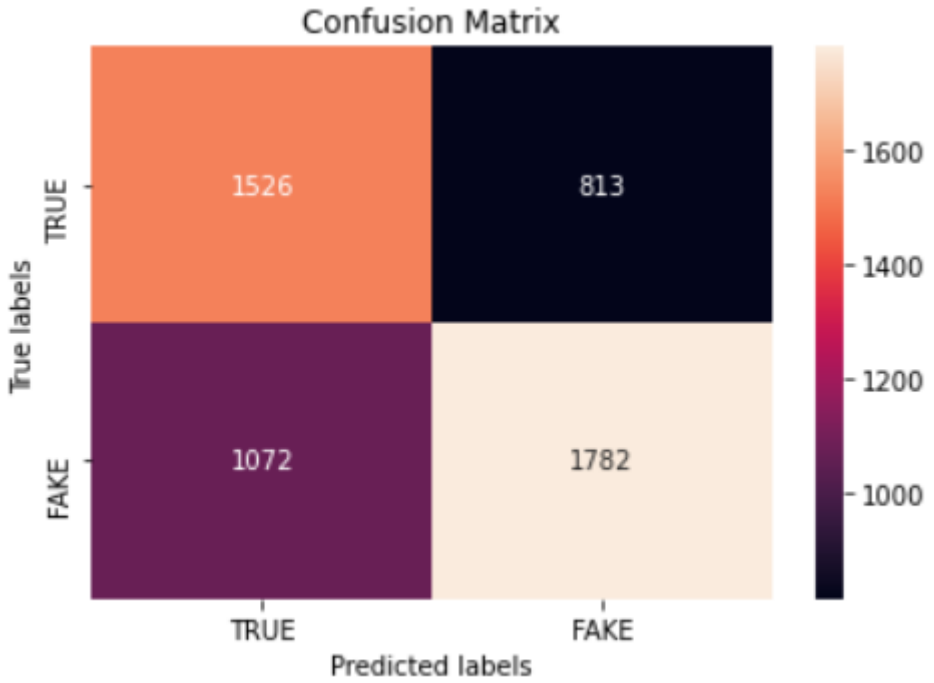
Support Vector Machines:

```
Applying SVM Model:
---------------------------
accuracy:   0.637
              precision    recall  f1-score   support

    Positive       0.59      0.65      0.62      2339
    Negative       0.69      0.62      0.65      2854

    accuracy                           0.64      5193
   macro avg       0.64      0.64      0.64      5193
weighted avg       0.64      0.64      0.64      5193
```

```
Time taken to apply model: 888.007661 seconds
```



Possible Further Improvements

Improvements and approaches that were not applied due to time constraints. Some of which could help the performance of the application.

- Apply different processing methods to the textual data to see how it impacts on the performance of the models
    - stemming
    - lemmatization
    - word embeddings
    - ngram usage with the TFIdfVectorizer instead of single words

Some of these could help to improve the classification results of the different models.

- Concatenating the title of the article to the text for an added source of information to be leveraged for gaining insights for the predictions
- The anomaly of why SVM took so long to be applied could be further investigated. This may be due the way I applied it or something related to the data applied with SVM. This is unclear at this point.
- Other approaches could be considered and applied such as deep learning methods which extract features from the textual data and make use of those features and patterns when applying it’s predictions for this binary classification problem.

Appendix

Code for the project from a jupyter notebook (contains both the code and some of the text that was originally in markup in the jupyter notebook.

```
3  import spacy

4

5  # disabling tagger and parser of spacy for time considerations

6  nlp = spacy.load('en_core_web_sm', disable=[ "parser", "ner"])

7

8  import pandas as pd

9  # read in the test and training data to dataframes

10 trainingData = pd.read_csv("C:/Users/simon/Documents/AI/Foundatoins in AI - TU Dublin Course/Project/Project Data/train.csv")

11 testData = pd.read_csv("C:/Users/simon/Documents/AI/Foundatoins in AI - TU Dublin Course/Project/Project Data/test.csv")

12 print(trainingData.head())

13 print(testData.head())
```

```
   id                                              title              author  \
0   0  House Dem Aide: We Didn’t Even See Comey’s Let...       Darrell Lucus
1   1  FLYNN: Hillary Clinton, Big Woman on Campus - ...     Daniel J. Flynn
2   2                  Why the Truth Might Get You Fired  Consortiumnews.com
3   3  15 Civilians Killed In Single US Airstrike Hav...     Jessica Purkiss
4   4  Iranian woman jailed for fictional unpublished...      Howard Portnoy

                                                text  label
0  House Dem Aide: We Didn’t Even See Comey’s Let...      1
1  Ever get the feeling your life circles the rou...      0
2  Why the Truth Might Get You Fired October 29, ...      1
3  Videos 15 Civilians Killed In Single US Airstr...      1
4  Print \nAn Iranian woman has been sentenced to...      1
      id                                              title  \
0  20800  Specter of Trump Loosens Tongues, if Not Purse...
1  20801  Russian warships ready to strike terrorists ne...
2  20802  #NoDAPL: Native American Leaders Vow to Stay A...
3  20803  Tim Tebow Will Attempt Another Comeback, This ...
4  20804                    Keiser Report: Meme Wars (E995)

                  author                                               text
0        David Streitfeld  PALO ALTO, Calif.  —   After years of scorning...
1                     NaN  Russian warships ready to strike terrorists ne...
2           Common Dreams  Videos #NoDAPL: Native American Leaders Vow to...
3           Daniel Victor  If at first you don’t succeed, try a different...
4  Truth Broadcast Network  42 mins ago 1 Views 0 Comments 0 Likes 'For th...
```

14

15 print("Training set Number of Rows and columns (Rows, Columns): ")

16 trainingData.shape

```
Training set Number of Rows and columns (Rows, Columns):
Test set Number of Rows and columns (Rows, Columns):

(5200, 4)
```

17

18 print("Test set Number of Rows and columns (Rows, Columns): ")

19 testData.shape

```
Test set Number of Rows and columns (Rows, Columns):

(5193, 4)
```

20

21 trainingData.info()

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 20800 entries, 0 to 20799
Data columns (total 5 columns):
 #   Column  Non-Null Count  Dtype 
---  ------  --------------  ----- 
 0   id      20800 non-null  int64 
 1   title   20242 non-null  object
 2   author  18843 non-null  object
 3   text    20761 non-null  object
 4   label   20800 non-null  int64 
dtypes: int64(2), object(3)
memory usage: 812.6+ KB
```

22

23 # Take test labels from submit csv dataframe and add them to the test dataframe (they were originally separate in two csv files)

24 testLabels = pd.read_csv('C:/Users/simon/Documents/AI/Foundatoins in AI - TU Dublin Course/Project/Project Data/submit.csv')

25 testData['labels'] = testLabels['label']

26 testData.head()

27

	id	title	author	text	labels
0	20800	Specter of Trump Loosens Tongues, if Not Purse...	David Streitfeld	PALO ALTO, Calif. — After years of scorning...	0
1	20801	Russian warships ready to strike terrorists ne...	NaN	Russian warships ready to strike terrorists ne...	1
2	20802	#NoDAPL: Native American Leaders Vow to Stay A...	Common Dreams	Videos #NoDAPL: Native American Leaders Vow to...	0
3	20803	Tim Tebow Will Attempt Another Comeback, This ...	Daniel Victor	If at first you don't succeed, try a different...	1

28 print("Training data nulls: ", trainingData['text'].isnull().sum())

29 print("Test data nulls: ", testData['text'].isnull().sum())

30 trainingData = trainingData.dropna(subset=['text'])

31 trainingData.info()

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 20761 entries, 0 to 20799
Data columns (total 5 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   id      20761 non-null  int64
 1   title   20203 non-null  object
 2   author  18843 non-null  object
 3   text    20761 non-null  object
 4   label   20761 non-null  int64
dtypes: int64(2), object(3)
memory usage: 973.2+ KB
```

```
testData = testData.dropna(subset=['text'])
testData.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 5193 entries, 0 to 5199
Data columns (total 4 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   id      5193 non-null   int64
 1   title   5071 non-null   object
 2   author  4697 non-null   object
 3   text    5193 non-null   object
dtypes: int64(1), object(3)
memory usage: 202.9+ KB
```

```
 Removing Stop Words and Normalizing text
def tokenizeAndNormalizeArticle(article):
    article_split = article.split()
    normalizedTokens = []

    for word in article_split:
        word = re.sub(r"[\n\t]*", "", word).strip()   # remove newlines, tabs and any additional spaces
```

```
        word = word.lower()                    # set to lower case
        if (nlp.vocab[word].is_stop == False):       # remove stop words
            normalizedTokens.append(word)
            word = re.sub('[^A-Za-z0-9]+', '', word)       # remove all special characters & punctutation

    return " ".join(normalizedTokens)


# call method removeStopWordsInTweet for each row of the text
# parameter to remove all stop words from the dataset
articleData = trainingData["text"]
normalizedArticleData = []

counter = 0
for article in articleData:
    if (type(article) is str):
        normalizedArticleData.append(tokenizeAndNormalizeArticle(article))
# repeat for test data
articleData = testData["text"]
normalizedTestData = []

counter = 0
```

```
for article in articleData:
    if (type(article) is str):
        normalizedTestData.append(tokenizeAndNormalizeArticle(article))

print(len(normalizedTestData))
Vectorization
from sklearn.feature_extraction.text import TfidfVectorizer

tfidfVectorizer=TfidfVectorizer(use_idf=True)

# vectorize training & test data
xTrainingTf = tfidfVectorizer.fit_transform(normalizedArticleData)

xTestTf = tfidfVectorizer.transform(normalizedTestData)
from time import time
from sklearn.naive_bayes import MultinomialNB
from sklearn import metrics

t = time()

yTraining = trainingData['label']
naiveBayesClassifier = MultinomialNB()
```

```
naiveBayesClassifier.fit(xTrainingTf, yTraining)

# predict the new document from the testing dataset
yPredictions = naiveBayesClassifier.predict(xTestTf)
yTest = testData['labels']

print("Applying Naive Bayes Model:")
print("---------------------------")

# compute the performance measures
score1 = metrics.accuracy_score(yTest, yPredictions)
print("accuracy:   %0.3f" % score1)

print(metrics.classification_report(yTest, yPredictions,
     target_names=['Positive', 'Negative']))

naiveBayesConfusionMatrix = metrics.confusion_matrix(yTest, yPredictions)

ax= plt.subplot()
sns.heatmap(naiveBayesConfusionMatrix, annot=True, fmt='g', ax=ax);

ax.set_xlabel('Predicted labels');ax.set_ylabel('True labels');
```

```
ax.set_title('Confusion Matrix');
ax.xaxis.set_ticklabels(['TRUE', 'FAKE']);
ax.yaxis.set_ticklabels(['TRUE', 'FAKE']);


duration = time() - t
print("\nTime taken to apply model: %f seconds" % (duration))

Random Forrest Classifier
from sklearn.ensemble import RandomForestClassifier

t = time()
randomForrestClassifier = RandomForestClassifier(n_estimators=100, random_state=0,
    n_jobs=-1)
randomForrestClassifier.fit(xTrainingTf, yTraining)
yPredictions = randomForrestClassifier.predict(xTestTf)

print("Applying Random Forrest Model:")
print("------------------------------")

# compute the performance measures
score1 = metrics.accuracy_score(yTest, yPredictions)
print("accuracy:   %0.3f" % score1)
```

```
131
132 print(metrics.classification_report(yTest, yPredictions,
133       target_names=['Positive', 'Negative']))
134
135 randomForrestConfusionMatrix = metrics.confusion_matrix(yTest, yPredictions)
136
137 ax= plt.subplot()
138 sns.heatmap(randomForrestConfusionMatrix, annot=True, fmt='g', ax=ax);
139
140 ax.set_xlabel('Predicted labels');ax.set_ylabel('True labels');
141 ax.set_title('Confusion Matrix');
142 ax.xaxis.set_ticklabels(['TRUE', 'FAKE']);
143 ax.yaxis.set_ticklabels(['TRUE', 'FAKE']);
144
145 duration = time() - t
146 print("\nTime taken to apply model: %f seconds" % (duration))
147
148 ,
149 Applying SVM Model Classifier
150 from sklearn import svm
151
152 #Create a svm Classifier
```

```
svmClassifier = svm.SVC(kernel='linear') # Linear Kernel

# Perform training for the svm model
svmClassifier.fit(xTrainingTf, yTraining)

# Predict the response for test dataset
yPredictions = svmClassifier.predict(xTestTf)

print("Applying SVM Model:")
print("---------------------------")

# compute the performance measures
score1 = metrics.accuracy_score(yTest, yPredictions)
print("accuracy:   %0.3f" % score1)

print(metrics.classification_report(yTest, yPredictions,
     target_names=['Positive', 'Negative']))

svmConfusionMatrix = metrics.confusion_matrix(yTest, yPredictions)

ax= plt.subplot()
sns.heatmap(svmConfusionMatrix, annot=True, fmt='g', ax=ax);
```

```
175
176 ax.set_xlabel('Predicted labels');ax.set_ylabel('True labels');
177 ax.set_title('Confusion Matrix');
178 ax.xaxis.set_ticklabels(['TRUE', 'FAKE']);
179 ax.yaxis.set_ticklabels(['TRUE', 'FAKE']);
180
181 duration = time() - t
182 print("\nTime taken to apply model: %f seconds" % (duration))
183
```