```
#https://github.com/Abhradipta/Fake-News-Detection/blob/master/Fake%20News%20Detection
```

Data Preprocessing

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
!pip install wordcloud
```

```
Requirement already satisfied: wordcloud in /usr/local/lib/python3.7/dist-packages (1.5
Requirement already satisfied: numpy>=1.6.1 in /usr/local/lib/python3.7/dist-packages (
Requirement already satisfied: pillow in /usr/local/lib/python3.7/dist-packages (from w
```

```
!unzip news.zip
```

```
Archive:  news.zip
replace news.csv? [y]es, [n]o, [A]ll, [N]one, [r]ename: y
  inflating: news.csv
```

Dataset

```
dataset = pd.read_csv('news.csv')
dataset.head()
```

	Unnamed: 0	title	text	label
0	8476	You Can Smell Hillary’s Fear	Daniel Greenfield, a Shillman Journalism Fello...	FAKE
1	10294	Watch The Exact Moment Paul Ryan Committed Pol...	Google Pinterest Digg Linkedin Reddit Stumbleu...	FAKE
2	3608	Kerry to go to Paris in gesture of sympathy	U.S. Secretary of State John F. Kerry said Mon...	REAL

```
dataset.shape
```

```
(6335, 4)
```

Importing other required libraries

```
import re
import nltk
nltk.download('stopwords')
from nltk.corpus import stopwords
from nltk.stem.porter import PorterStemmer
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

Removing stopwords from datset

```
corpus = []
ps = PorterStemmer()
for i in range(0,3000):
  news = re.sub('[^^a-zA-Z0-9]',' ',str(dataset['text'][i]))
  news = news.lower()
  news = news.split()
  news = [ps.stem(word) for word in news if len(news)<=2 or not word in set(stopwords.w
  news = ' '.join(news)
  corpus.append(news)
```

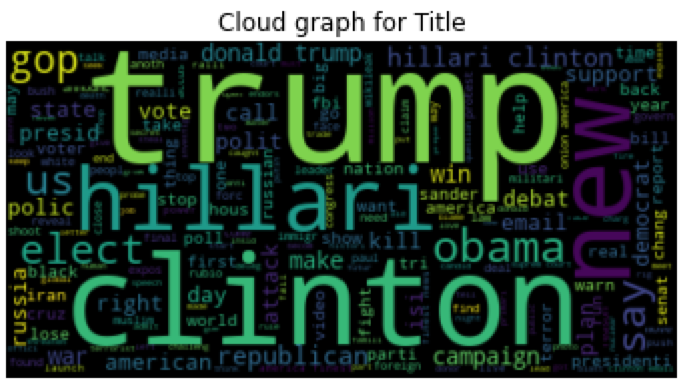
```
corpus2 = []
for i in range(0,3000):
  news = re.sub('[^^a-zA-Z0-9]',' ',str(dataset['title'][i]))
  news = news.lower()
  news = news.split()
  news = [ps.stem(word) for word in news if len(news)<=2 or not word in set(stopwords.w
  news = ' '.join(news)
  corpus2.append(news)
```

WordCloud visualization (Most Frequent words)

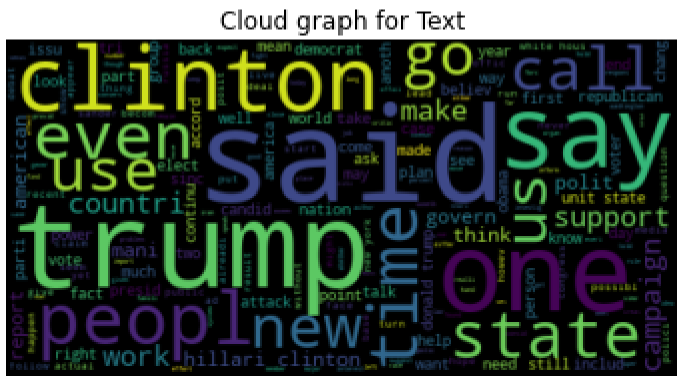
```
textVisualiseTitle = ""
textVisualiseText = ""
for word in corpus:
  textVisualiseText = textVisualiseText + word
for word in corpus2:
  textVisualiseTitle = textVisualiseTitle + word
```

```
from wordcloud import WordCloud
# Create and generate a word cloud image:
wordcloud = WordCloud().generate(textVisualiseTitle)

# Display the generated image:
```

```
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis("off")
plt.title("Cloud graph for Title")
plt.show()
```



```
# Create and generate a word cloud image:
wordcloud = WordCloud().generate(textVisualiseText)

# Display the generated image:
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis("off")
plt.title("Cloud graph for Text")
plt.show()
```



Creating bag of words

```
from sklearn.feature_extraction.text import CountVectorizer
vectorizer = CountVectorizer()
# tokenize and build vocab
vectorizer.fit(corpus)
# summarize
print(vectorizer.vocabulary_)
# encode document
news_body = vectorizer.transform(corpus).todense()
```

```
# summarize encoded vector
print(news_body.shape)
print(type(news_body))
```

```
{'daniel': 7892, 'greenfield': 12637, 'shillman': 26220, 'journal': 15673, 'fellow': 10
(3000, 32530)
<class 'numpy.matrix'>
```

```
# tokenize and build vocab
vectorizer.fit(corpus2)
# summarize
print(vectorizer.vocabulary_)
# encode document
news_title = vectorizer.transform(corpus2).todense()
# summarize encoded vector
print(news_title.shape)
print(type(news_title))
```

```
{'smell': 4117, 'hillari': 2105, 'fear': 1702, 'watch': 4853, 'exact': 1609, 'moment':
(3000, 5003)
<class 'numpy.matrix'>
```

```
final_vector = np.hstack((news_title,news_body))
features = final_vector
features.shape
```

```
(3000, 37533)
```

```
y = dataset.iloc[:3000,-1].values
```

Splitting dataset into train and test

```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test = train_test_split(features,y,test_size = 0.25,random_st
```

Support Vector Machine

```
from sklearn.svm import SVC
svm_clf = SVC(kernel='sigmoid')
svm_clf.fit(x_train,y_train)
```

```
SVC(C=1.0, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,
          decision_function_shape='ovr', degree=3, gamma='scale', kernel='sigmoid',
          max_iter=-1, probability=False, random_state=None, shrinking=True,
          tol=0.001, verbose=False)
```

```
y_pred = svm_clf.predict(x_test)
print(np.concatenate((y_pred.reshape(len(y_pred),1), y_test.reshape(len(y_test),1)),1)
```

```
[['REAL' 'REAL']
 ['REAL' 'REAL']
 ['FAKE' 'REAL']
 ...
 ['REAL' 'FAKE']
 ['FAKE' 'FAKE']
 ['REAL' 'REAL']]
```

```
from sklearn.metrics import confusion_matrix,accuracy_score,classification_report
print(confusion_matrix(y_test,y_pred))
```

```
[[290  86]
 [ 89 285]]
```

```
print(classification_report(y_test,y_pred))
```

```
              precision    recall  f1-score   support

        FAKE       0.77      0.77      0.77       376
        REAL       0.77      0.76      0.77       374

    accuracy                           0.77       750
   macro avg       0.77      0.77      0.77       750
weighted avg       0.77      0.77      0.77       750
```

```
np.set_printoptions(precision=2)
print(accuracy_score(y_test,y_pred)*100)
```

```
76.66666666666667
```

Decision Tree Classifier

```
from sklearn.tree import DecisionTreeClassifier
Dclassifier = DecisionTreeClassifier(criterion = 'entropy', random_state = 0)
Dclassifier.fit(x_train, y_train)
```

```
DecisionTreeClassifier(ccp_alpha=0.0, class_weight=None, criterion='entropy',
                       max_depth=None, max_features=None, max_leaf_nodes=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, presort='deprecated',
                       random_state=0, splitter='best')
```

```
# Predicting the Test set results
y_pred = Dclassifier.predict(x_test)
print(np.concatenate((y_pred.reshape(len(y_pred),1), y_test.reshape(len(y_test),1)),1)
```

```
[['REAL' 'REAL']
 ['REAL' 'REAL']
 ['REAL' 'REAL']
 ...
 ['REAL' 'FAKE']
 ['FAKE' 'FAKE']
 ['REAL' 'REAL']]
```

```
print(confusion_matrix(y_test,y_pred))
```

```
[[304  72]
 [ 82 292]]
```

```
print(classification_report(y_test,y_pred))
```

```
              precision    recall  f1-score   support

        FAKE       0.79      0.81      0.80       376
        REAL       0.80      0.78      0.79       374

    accuracy                           0.79       750
   macro avg       0.79      0.79      0.79       750
weighted avg       0.79      0.79      0.79       750
```

```
np.set_printoptions(precision=2)
print(accuracy_score(y_test,y_pred)*100)
```

```
79.46666666666667
```

Random Forest Classifier

```
from sklearn.ensemble import RandomForestClassifier
Rclassifier = RandomForestClassifier(n_estimators = 65, criterion = 'entropy', random_s
Rclassifier.fit(x_train, y_train)
```

```
RandomForestClassifier(bootstrap=True, ccp_alpha=0.0, class_weight=None,
                       criterion='entropy', max_depth=None, max_features='auto',
                       max_leaf_nodes=None, max_samples=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, n_estimators=65,
                       n_jobs=None, oob_score=False, random_state=0, verbose=0,
                       warm_start=False)
```

```
# Predicting the Test set results
y_pred = Rclassifier.predict(x_test)
print(np.concatenate((y_pred.reshape(len(y_pred),1), y_test.reshape(len(y_test),1)),1)
```

```
[['REAL' 'REAL']
 ['REAL' 'REAL']
 ['REAL' 'REAL']
 ...
 ['REAL' 'FAKE']
 ['FAKE' 'FAKE']
 ['REAL' 'REAL']]
```

```
from sklearn.metrics import confusion_matrix,accuracy_score,classification_report
print(confusion_matrix(y_test,y_pred))
```

```
[[334  42]
 [ 55 319]]
```

```
print(classification_report(y_test,y_pred))
```

```
              precision    recall  f1-score   support

        FAKE       0.86      0.89      0.87       376
        REAL       0.88      0.85      0.87       374

    accuracy                           0.87       750
   macro avg       0.87      0.87      0.87       750
weighted avg       0.87      0.87      0.87       750
```

```
np.set_printoptions(precision=2)
print(accuracy_score(y_test,y_pred)*100)
```

```
87.06666666666666
```

Fake prediction

```
text = '''
"House Dem Aide: We Didnâ€™t Even See Comeyâ€™s Letter Until Jason Chaffetz Tweeted It
With apologies to Keith Olbermann, there is no doubt who the Worst Person in The World
As we now know, Comey notified the Republican chairmen and Democratic ranking members o
â€” Jason Chaffetz (@jasoninthehouse) October 28, 2016
Of course, we now know that this was not the case . Comey was actually saying that it w
```

```
But according to a senior House Democratic aide, misreading that letter may have been 
So letâ€™s see if weâ€™ve got this right. The FBI director tells Chaffetz and other GOI
There has already been talk on Daily Kos that Comey himself provided advance notice of
What it does suggest, however, is that Chaffetz is acting in a way that makes Dan Burt
Granted, itâ€™s not likely that Chaffetz will have to answer for this. He sits in a ri
Darrell is a 30-something graduate of the University of North Carolina who considers hi

'''
title = '''
House Dem Aide: We Didnâ€™t Even See Comeyâ€™s Letter Until Jason Chaffetz Tweeted It
'''
```

```
text = re.sub('[^a-zA-Z]', ' ', text)
text = text.lower()
text = text.split()
ps = PorterStemmer()
text = [ps.stem(word) for word in text if not word in set(stopwords.words('english'))]
text = ' '.join(text)
new_corpus_text = [text]
vectorizer.fit(corpus)
new_X_text = vectorizer.transform(new_corpus_text).todense()
title = re.sub('[^a-zA-Z]', ' ', title)
title = title.lower()
title = title.split()
ps = PorterStemmer()
title = [ps.stem(word) for word in title if not word in set(stopwords.words('english'))
title = ' '.join(title)
new_corpus_title = [title]
vectorizer.fit(corpus2)
new_X_title = vectorizer.transform(new_corpus_title).todense()
final_X_test = np.hstack((new_X_title,new_X_text))
new_y_pred = Rclassifier.predict(final_X_test)
print(new_y_pred)
```

```
['FAKE']
```

Real Prediction

```
text = '''
Donald J. Trump is scheduled to make a highly anticipated visit to an   church in Detr
'''

title = '''
Excepts From a Draft Script for Donald Trumpâ€™s Q&ampA With a Black Churchâ€™s Pastor
'''
```

```
text = re.sub('[^a-zA-Z]', ' ', text)
text = text.lower()
text = text.split()
ps = PorterStemmer()
text = [ps.stem(word) for word in text if not word in set(stopwords.words('english'))]
text = ' '.join(text)
new_corpus_text = [text]
vectorizer.fit(corpus)
new_X_text = vectorizer.transform(new_corpus_text).todense()
title = re.sub('[^a-zA-Z]', ' ', title)
title = title.lower()
title = title.split()
ps = PorterStemmer()
title = [ps.stem(word) for word in title if not word in set(stopwords.words('english'))
title = ' '.join(title)
new_corpus_title = [title]
vectorizer.fit(corpus2)
new_X_title = vectorizer.transform(new_corpus_title).todense()
final_X_test = np.hstack((new_X_title,new_X_text))
new_y_pred = Rclassifier.predict(final_X_test)
print(new_y_pred)
```

```
['REAL']
```

Hyperparameter tunning

```
from sklearn.model_selection import GridSearchCV
param = {
    "n_estimators":[0,20,40,50]
}
```

```
clf_grid_Random = GridSearchCV(estimator=Rclassifier,param_grid=param,cv=3,n_jobs=-1)
clf_grid_Random.fit(x_train,y_train)
```

```
/usr/local/lib/python3.7/dist-packages/joblib/externals/loky/process_executor.py:691: U
  "timeout or by a memory leak.", UserWarning
GridSearchCV(cv=3, error_score=nan,
             estimator=RandomForestClassifier(bootstrap=True, ccp_alpha=0.0,
                                              class_weight=None,
                                              criterion='entropy',
                                              max_depth=None,
                                              max_features='auto',
                                              max_leaf_nodes=None,
                                              max_samples=None,
                                              min_impurity_decrease=0.0,
                                              min_impurity_split=None,
                                              min_samples_leaf=1,
                                              min_samples_split=2,
                                                  min_weight_fraction_leaf=0.0,
                                                  n_estimators=65, n_jobs=None,
                                                  oob_score=False, random_state=0,
                                                  verbose=0, warm_start=False),
                     iid='deprecated', n_jobs=-1,
                     param_grid={'n_estimators': [0, 20, 40, 50]},
                     pre_dispatch='2*n_jobs', refit=True, return_train_score=False,
                     scoring=None, verbose=0)
```

```
clf_grid_Random.best_params_
```

```
{'n_estimators': 50}
```

```
hyper_pred = clf_grid_Random.predict(x_test)
print(np.concatenate((hyper_pred.reshape(len(hyper_pred),1), y_test.reshape(len(y_test)
```

```
[['REAL' 'REAL']
 ['REAL' 'REAL']
 ['REAL' 'REAL']
 ...
 ['REAL' 'FAKE']
 ['FAKE' 'FAKE']
 ['REAL' 'REAL']]
```

```
print(confusion_matrix(y_test,hyper_pred))
```

```
[[339  37]
 [ 61 313]]
```

```
print(classification_report(y_test,hyper_pred))
```

```
              precision    recall  f1-score   support

        FAKE       0.85      0.90      0.87       376
        REAL       0.89      0.84      0.86       374

    accuracy                           0.87       750
   macro avg       0.87      0.87      0.87       750
weighted avg       0.87      0.87      0.87       750
```

```
np.set_printoptions(precision=2)
print(accuracy_score(y_test,hyper_pred)*100)
```

```
86.93333333333332
```

Fake News Prediction

```
text = '''
"House Dem Aide: We Didnâ€™t Even See Comeyâ€™s Letter Until Jason Chaffetz Tweeted It
With apologies to Keith Olbermann, there is no doubt who the Worst Person in The World
As we now know, Comey notified the Republican chairmen and Democratic ranking members o
â€” Jason Chaffetz (@jasoninthehouse) October 28, 2016
Of course, we now know that this was not the case . Comey was actually saying that it w
But according to a senior House Democratic aide, misreading that letter may have been t
So letâ€™s see if weâ€™ve got this right. The FBI director tells Chaffetz and other GOP
There has already been talk on Daily Kos that Comey himself provided advance notice of
What it does suggest, however, is that Chaffetz is acting in a way that makes Dan Burto
Granted, itâ€™s not likely that Chaffetz will have to answer for this. He sits in a ric
Darrell is a 30-something graduate of the University of North Carolina who considers hi

'''
title = '''
House Dem Aide: We Didnâ€™t Even See Comeyâ€™s Letter Until Jason Chaffetz Tweeted It
'''
```

```
text = re.sub('[^a-zA-Z]', ' ', text)
text = text.lower()
text = text.split()
ps = PorterStemmer()
text = [ps.stem(word) for word in text if not word in set(stopwords.words('english'))]
text = ' '.join(text)
new_corpus_text = [text]
vectorizer.fit(corpus)
new_X_text = vectorizer.transform(new_corpus_text).todense()
title = re.sub('[^a-zA-Z]', ' ', title)
title = title.lower()
title = title.split()
ps = PorterStemmer()
title = [ps.stem(word) for word in title if not word in set(stopwords.words('english')
title = ' '.join(title)
new_corpus_title = [title]
vectorizer.fit(corpus2)
new_X_title = vectorizer.transform(new_corpus_title).todense()
final_X_test = np.hstack((new_X_title,new_X_text))
new_y_pred = clf_grid_Random.predict(final_X_test)
print(new_y_pred)
```

```
['FAKE']
```

Real News *Prediction*

```
text = '''
Donald J. Trump is scheduled to make a highly anticipated visit to an   church in Detro
Donald J. Trump is scheduled to make a highly anticipated visit to an   church in Detro
'''

title = '''
Excepts From a Draft Script for Donald Trumpâ€™s Q&ampA With a Black Churchâ€™s Pastor
'''
```

```
text = re.sub('[^a-zA-Z]', ' ', text)
text = text.lower()
text = text.split()
ps = PorterStemmer()
text = [ps.stem(word) for word in text if not word in set(stopwords.words('english'))]
text = ' '.join(text)
new_corpus_text = [text]
vectorizer.fit(corpus)
new_X_text = vectorizer.transform(new_corpus_text).todense()
title = re.sub('[^a-zA-Z]', ' ', title)
title = title.lower()
title = title.split()
ps = PorterStemmer()
title = [ps.stem(word) for word in title if not word in set(stopwords.words('english')
title = ' '.join(title)
new_corpus_title = [title]
vectorizer.fit(corpus2)
new_X_title = vectorizer.transform(new_corpus_title).todense()
final_X_test = np.hstack((new_X_title,new_X_text))
new_y_pred = clf_grid_Random.predict(final_X_test)
print(new_y_pred)
```

```
['REAL']
```

XGBoost Classifier

```
from xgboost import XGBClassifier
classifier = XGBClassifier()
classifier.fit(x_train, y_train)
```

```
XGBClassifier(base_score=0.5, booster='gbtree', colsample_bylevel=1,
              colsample_bynode=1, colsample_bytree=1, gamma=0,
              learning_rate=0.1, max_delta_step=0, max_depth=3,
              min_child_weight=1, missing=None, n_estimators=100, n_jobs=1,
              nthread=None, objective='binary:logistic', random_state=0,
              reg_alpha=0, reg_lambda=1, scale_pos_weight=1, seed=None,
              silent=None, subsample=1, verbosity=1)
```

```
# Predicting the Test set results
y_pred = classifier.predict(x_test)
```

```
print(np.concatenate((y_pred.reshape(len(y_pred),1), y_test.reshape(len(y_test),1)),1)
```

```
[['REAL' 'REAL']
 ['REAL' 'REAL']
 ['REAL' 'REAL']
 ...
 ['REAL' 'FAKE']
 ['FAKE' 'FAKE']
 ['REAL' 'REAL']]
```

```
print(confusion_matrix(y_test,y_pred))
```

```
[[344  32]
 [ 57 317]]
```

```
print(classification_report(y_test,y_pred))
```

```
              precision    recall  f1-score   support

        FAKE       0.86      0.91      0.89       376
        REAL       0.91      0.85      0.88       374

    accuracy                           0.88       750
   macro avg       0.88      0.88      0.88       750
weighted avg       0.88      0.88      0.88       750
```

```
np.set_printoptions(precision=2)
print(accuracy_score(y_test,y_pred)*100)
```

```
88.13333333333333
```

Fake prediction

```
text = '''
"House Dem Aide: We Didnâ€™t Even See Comeyâ€™s Letter Until Jason Chaffetz Tweeted It
With apologies to Keith Olbermann, there is no doubt who the Worst Person in The World
As we now know, Comey notified the Republican chairmen and Democratic ranking members 
â€” Jason Chaffetz (@jasoninthehouse) October 28, 2016
Of course, we now know that this was not the case . Comey was actually saying that it 
But according to a senior House Democratic aide, misreading that letter may have been 
So letâ€™s see if weâ€™ve got this right. The FBI director tells Chaffetz and other GO
There has already been talk on Daily Kos that Comey himself provided advance notice of
What it does suggest, however, is that Chaffetz is acting in a way that makes Dan Burt
Granted, itâ€™s not likely that Chaffetz will have to answer for this. He sits in a ri
Darrell is a 30-something graduate of the University of North Carolina who considers h

'''
```

```
title = '''
House Dem Aide: We Didnâ€™t Even See Comeyâ€™s Letter Until Jason Chaffetz Tweeted It
'''
```

```
text = re.sub('[^a-zA-Z]', ' ', text)
text = text.lower()
text = text.split()
ps = PorterStemmer()
text = [ps.stem(word) for word in text if not word in set(stopwords.words('english'))]
text = ' '.join(text)
new_corpus_text = [text]
vectorizer.fit(corpus)
new_X_text = vectorizer.transform(new_corpus_text).todense()
title = re.sub('[^a-zA-Z]', ' ', title)
title = title.lower()
title = title.split()
ps = PorterStemmer()
title = [ps.stem(word) for word in title if not word in set(stopwords.words('english'))
title = ' '.join(title)
new_corpus_title = [title]
vectorizer.fit(corpus2)
new_X_title = vectorizer.transform(new_corpus_title).todense()
final_X_test = np.hstack((new_X_title,new_X_text))
new_y_pred = classifier.predict(final_X_test)
print(new_y_pred)
```

```
['FAKE']
```

Real Prediction

```
text = '''
Donald J. Trump is scheduled to make a highly anticipated visit to an   church in Detr
'''


title = '''
Excepts From a Draft Script for Donald Trumpâ€™s Q&ampA With a Black Churchâ€™s Pastor
'''
```

```
text = re.sub('[^a-zA-Z]', ' ', text)
text = text.lower()
text = text.split()
ps = PorterStemmer()
text = [ps.stem(word) for word in text if not word in set(stopwords.words('english'))]
text = ' '.join(text)
new_corpus_text = [text]
vectorizer.fit(corpus)
```

```
new_X_text = vectorizer.transform(new_corpus_text).todense()
title = re.sub('[^a-zA-Z]', ' ', title)
title = title.lower()
title = title.split()
ps = PorterStemmer()
title = [ps.stem(word) for word in title if not word in set(stopwords.words('english'))
title = ' '.join(title)
new_corpus_title = [title]
vectorizer.fit(corpus2)
new_X_title = vectorizer.transform(new_corpus_title).todense()
final_X_test = np.hstack((new_X_title,new_X_text))
new_y_pred = classifier.predict(final_X_test)
print(new_y_pred)
```

```
['REAL']
```

Logisitic Regression

```
from sklearn.linear_model import LogisticRegression
clf_reg = LogisticRegression(max_iter=200)
clf_reg.fit(x_train,y_train)
```

```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=200,
                   multi_class='auto', n_jobs=None, penalty='l2',
                   random_state=None, solver='lbfgs', tol=0.0001, verbose=0,
                   warm_start=False)
```

```
pred_reg = clf_reg.predict(x_test)
print(np.concatenate((pred_reg.reshape(len(pred_reg),1),y_test.reshape(len(y_test),1))
```

```
[['REAL' 'REAL']
 ['REAL' 'REAL']
 ['FAKE' 'REAL']
 ...
 ['REAL' 'FAKE']
 ['FAKE' 'FAKE']
 ['REAL' 'REAL']]
```

```
from sklearn.metrics import confusion_matrix,accuracy_score,classification_report
print(confusion_matrix(y_test,pred_reg))
```

```
[[346  30]
 [ 43 331]]
```

```
print(classification_report(y_test,pred_reg))
```

```
              precision    recall  f1-score   support

        FAKE       0.89      0.92      0.90       376
        REAL       0.92      0.89      0.90       374

    accuracy                           0.90       750
   macro avg       0.90      0.90      0.90       750
weighted avg       0.90      0.90      0.90       750
```

```
np.set_printoptions(precision=2)
print(accuracy_score(y_test,pred_reg)*100)
```

```
90.26666666666667
```

Fake News Prediction

```
text = '''
"House Dem Aide: We Didnâ€™t Even See Comeyâ€™s Letter Until Jason Chaffetz Tweeted It
With apologies to Keith Olbermann, there is no doubt who the Worst Person in The World
As we now know, Comey notified the Republican chairmen and Democratic ranking members c
â€" Jason Chaffetz (@jasoninthehouse) October 28, 2016
Of course, we now know that this was not the case . Comey was actually saying that it w
But according to a senior House Democratic aide, misreading that letter may have been t
So letâ€™s see if weâ€™ve got this right. The FBI director tells Chaffetz and other GOF
There has already been talk on Daily Kos that Comey himself provided advance notice of
What it does suggest, however, is that Chaffetz is acting in a way that makes Dan Burto
Granted, itâ€™s not likely that Chaffetz will have to answer for this. He sits in a ric
Darrell is a 30-something graduate of the University of North Carolina who considers hi

'''
title = '''
House Dem Aide: We Didnâ€™t Even See Comeyâ€™s Letter Until Jason Chaffetz Tweeted It
'''
```

```
text = re.sub('[^a-zA-Z]', ' ', text)
text = text.lower()
text = text.split()
ps = PorterStemmer()
text = [ps.stem(word) for word in text if not word in set(stopwords.words('english'))]
text = ' '.join(text)
new_corpus_text = [text]
vectorizer.fit(corpus)
new_X_text = vectorizer.transform(new_corpus_text).todense()
title = re.sub('[^a-zA-Z]', ' ', title)
title = title.lower()
title = title.split()
ps = PorterStemmer()
title = [ps.stem(word) for word in title if not word in set(stopwords.words('english')
title = ' '.join(title)
new_corpus_title = [title]
vectorizer.fit(corpus2)
new_X_title = vectorizer.transform(new_corpus_title).todense()
final_X_test = np.hstack((new_X_title,new_X_text))
new_y_pred = clf_reg.predict(final_X_test)
print(new_y_pred)
```

```
['FAKE']
```

Real News *Prediction*

```
text = '''
Donald J. Trump is scheduled to make a highly anticipated visit to an   church in Detr
'''


title = '''
Excepts From a Draft Script for Donald Trumpâ€™s Q&ampA With a Black Churchâ€™s Pastor
'''
```

```
text = re.sub('[^a-zA-Z]', ' ', text)
text = text.lower()
text = text.split()
ps = PorterStemmer()
text = [ps.stem(word) for word in text if not word in set(stopwords.words('english'))]
text = ' '.join(text)
new_corpus_text = [text]
vectorizer.fit(corpus)
new_X_text = vectorizer.transform(new_corpus_text).todense()
title = re.sub('[^a-zA-Z]', ' ', title)
title = title.lower()
title = title.split()
ps = PorterStemmer()
title = [ps.stem(word) for word in title if not word in set(stopwords.words('english')
title = ' '.join(title)
new_corpus_title = [title]
vectorizer.fit(corpus2)
new_X_title = vectorizer.transform(new_corpus_title).todense()
final_X_test = np.hstack((new_X_title,new_X_text))
new_y_pred = clf_reg.predict(final_X_test)
print(new_y_pred)
```

```
['REAL']
```

Hyperparameter tuning for Logistic regression

```
from sklearn.model_selection import GridSearchCV
param = {
    'penalty' : ['l1', 'l2'],
    'C' : np.logspace(-4, 4, 20),
    'solver' : ['liblinear']
}
```

```
clf_grid = GridSearchCV(estimator=clf_reg,param_grid=param,cv=5,verbose=True, n_jobs=-
clf_grid.fit(x_train,y_train)
```

```
Fitting 5 folds for each of 40 candidates, totalling 200 fits
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 2 concurrent workers.
/usr/local/lib/python3.7/dist-packages/joblib/externals/loky/process_executor.py:691: U
  "timeout or by a memory leak.", UserWarning
[Parallel(n_jobs=-1)]: Done  46 tasks      | elapsed:  1.0min
[Parallel(n_jobs=-1)]: Done 196 tasks      | elapsed:  3.7min
[Parallel(n_jobs=-1)]: Done 200 out of 200 | elapsed:  3.8min finished
GridSearchCV(cv=5, error_score=nan,
             estimator=LogisticRegression(C=1.0, class_weight=None, dual=False,
                                          fit_intercept=True,
                                          intercept_scaling=1, l1_ratio=None,
                                          max_iter=200, multi_class='auto',
                                          n_jobs=None, penalty='l2',
                                          random_state=None, solver='lbfgs',
                                          tol=0.0001, verbose=0,
                                          warm_start=False),
             iid='deprecated', n_jobs=-1,
             param_grid={'C': array([1.00e-04, 2.64e-04, 6.95e-04, 1.83e-03, 4.83e-03,
       3.36e-02, 8.86e-02, 2.34e-01, 6.16e-01, 1.62e+00, 4.28e+00,
       1.13e+01, 2.98e+01, 7.85e+01, 2.07e+02, 5.46e+02, 1.44e+03,
       3.79e+03, 1.00e+04]),
                         'penalty': ['l1', 'l2'], 'solver': ['liblinear']},
             pre_dispatch='2*n_jobs', refit=True, return_train_score=False,
             scoring=None, verbose=True)
```

```
clf_grid.best_params_
```

```
{'C': 0.03359818286283781, 'penalty': 'l2', 'solver': 'liblinear'}
```

```
hyper_pred = clf_grid.predict(x_test)
print(np.concatenate((hyper_pred.reshape(len(hyper_pred),1), y_test.reshape(len(y_test
```

```
[['REAL' 'REAL']
 ['REAL' 'REAL']
 ['REAL' 'REAL']
 ...
 ['REAL' 'FAKE']
 ['FAKE' 'FAKE']
 ['REAL' 'REAL']]
```

```
print(confusion_matrix(y_test,hyper_pred))
```

```
[[349  27]
 [ 45 329]]
```

```
print(classification_report(y_test,hyper_pred))
```

```
              precision    recall  f1-score   support

        FAKE       0.89      0.93      0.91       376
        REAL       0.92      0.88      0.90       374

    accuracy                           0.90       750
   macro avg       0.90      0.90      0.90       750
weighted avg       0.90      0.90      0.90       750
```

```
np.set_printoptions(precision=2)
print(accuracy_score(y_test,hyper_pred)*100)
```

```
90.4
```

Fake News Prediction

```
text = '''
"House Dem Aide: We Didnâ€™t Even See Comeyâ€™s Letter Until Jason Chaffetz Tweeted It
With apologies to Keith Olbermann, there is no doubt who the Worst Person in The World
As we now know, Comey notified the Republican chairmen and Democratic ranking members o
â€” Jason Chaffetz (@jasoninthehouse) October 28, 2016
Of course, we now know that this was not the case . Comey was actually saying that it w
But according to a senior House Democratic aide, misreading that letter may have been t
So letâ€™s see if weâ€™ve got this right. The FBI director tells Chaffetz and other GOP
There has already been talk on Daily Kos that Comey himself provided advance notice of
What it does suggest, however, is that Chaffetz is acting in a way that makes Dan Burto
Granted, itâ€™s not likely that Chaffetz will have to answer for this. He sits in a ric
Darrell is a 30-something graduate of the University of North Carolina who considers hi

'''
title = '''
House Dem Aide: We Didnâ€™t Even See Comeyâ€™s Letter Until Jason Chaffetz Tweeted It
'''
```

```
text = re.sub('[^a-zA-Z]', ' ', text)
text = text.lower()
text = text.split()
ps = PorterStemmer()
text = [ps.stem(word) for word in text if not word in set(stopwords.words('english'))]
text = ' '.join(text)
new_corpus_text = [text]
vectorizer.fit(corpus)
new_X_text = vectorizer.transform(new_corpus_text).todense()
title = re.sub('[^a-zA-Z]', ' ', title)
title = title.lower()
title = title.split()
ps = PorterStemmer()
title = [ps.stem(word) for word in title if not word in set(stopwords.words('english')
title = ' '.join(title)
new_corpus_title = [title]
vectorizer.fit(corpus2)
new_X_title = vectorizer.transform(new_corpus_title).todense()
final_X_test = np.hstack((new_X_title,new_X_text))
new_y_pred = clf_grid.predict(final_X_test)
print(new_y_pred)
```

```
['FAKE']
```

Real News *Prediction*

```
text = '''
Donald J. Trump is scheduled to make a highly anticipated visit to an   church in Detr
'''


title = '''
Excepts From a Draft Script for Donald Trumpâ€™s Q&ampA With a Black Churchâ€™s Pastor
'''
```

```
text = re.sub('[^a-zA-Z]', ' ', text)
text = text.lower()
text = text.split()
ps = PorterStemmer()
text = [ps.stem(word) for word in text if not word in set(stopwords.words('english'))]
text = ' '.join(text)
new_corpus_text = [text]
vectorizer.fit(corpus)
new_X_text = vectorizer.transform(new_corpus_text).todense()
title = re.sub('[^a-zA-Z]', ' ', title)
title = title.lower()
title = title.split()
ps = PorterStemmer()
```

```
title = [ps.stem(word) for word in title if not word in set(stopwords.words('english')
title = ' '.join(title)
new_corpus_title = [title]
vectorizer.fit(corpus2)
new_X_title = vectorizer.transform(new_corpus_title).todense()
final_X_test = np.hstack((new_X_title,new_X_text))
new_y_pred = clf_grid.predict(final_X_test)
print(new_y_pred)
```

```
['REAL']
```

✓ 0s completed at 4:38 PM