In [1]:
```
import numpy as np
import pandas as pd
import nltk
from nltk.corpus import stopwords
import string
```

In [2]:
```
df = pd.read_csv('C:/Users/Yash/Desktop/Fake News/fake_or_real_news.csv')

df.head(5)
```

Out[2]:

	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
3	10142	Bernie supporters on Twitter erupt in anger ag...	— Kaydee King (@KaydeeKing) November 9, 2016 T...	FAKE
4	875	The Battle of New York: Why This Primary Matters	It's primary day in New York and front-runners...	REAL

In [3]:
```
df.shape
```

Out[3]:
```
(6335, 4)
```

In [4]:
```
df.columns
```

Out[4]:
```
Index(['Unnamed: 0', 'title', 'text', 'label'], dtype='object')
```

In [5]:
```
df.drop_duplicates(inplace = True)
```

In [6]:
```
df.shape
```

Out[6]:
```
(6335, 4)
```

In [7]:
```
df.loc[df["label"]=="FAKE","label"]=0
df.loc[df["label"]=="REAL","label"]=1
```

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

Out[8]:

	Unnamed: 0	title	text	label
0	8476	You Can Smell Hillary's Fear	Daniel Greenfield, a Shillman Journalism Fello...	0
1	10294	Watch The Exact Moment Paul Ryan Committed Pol...	Google Pinterest Digg Linkedin Reddit Stumbleu...	0
2	3608	Kerry to go to Paris in gesture of sympathy	U.S. Secretary of State John F. Kerry said Mon...	1
3	10142	Bernie supporters on Twitter erupt in anger ag...	— Kaydee King (@KaydeeKing) November 9, 2016 T...	0
4	875	The Battle of New York: Why This Primary Matters	It's primary day in New York and front-runners...	1

In [9]:
```
df.isnull().sum()
```

Out[9]:
```
Unnamed: 0    0
title         0
text          0
label         0
dtype: int64
```

In [10]:
```
nltk.download('stopwords')
```

```
[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\Yash\AppData\Roaming\nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

Out[10]: True

In [11]:
```
def process_text(text):
    '''
    1. Remove punctuation
    2. Remove stopwords
    3. Return list of clean text words
    '''

    #1
    nopunc = [char for char in text if char not in string.punctuation]
    nopunc = ''.join(nopunc)

    #2
    clean_words = [word for word in nopunc.split() if word.lower() not in stop
words.words('english')]

    #3
    return clean_words
```

In [12]:
```
#Show the Tokenization
df['text'].head().apply(process_text)
```

Out[12]:
```
0    [Daniel, Greenfield, Shillman, Journalism, Fel...
1    [Google, Pinterest, Digg, Linkedin, Reddit, St...
2    [US, Secretary, State, John, F, Kerry, said, M...
3    [—, Kaydee, King, KaydeeKing, November, 9, 201...
4    [primary, day, New, York, frontrunners, Hillar...
Name: text, dtype: object
```

In [13]:
```
from sklearn.feature_extraction.text import CountVectorizer
News = CountVectorizer(analyzer=process_text).fit_transform(df['text'])
```

In [14]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(News, df['label'], test_si
ze = 0.20, random_state = 0)
```

In [65]:
```
#Get the shape of News
News.shape
```

Out[65]:
```
(6335, 128019)
```

In [16]:
```
from sklearn.naive_bayes import MultinomialNB
classifier = MultinomialNB()
classifier.fit(X_train, y_train)
```

Out[16]:
```
MultinomialNB(alpha=1.0, class_prior=None, fit_prior=True)
```

In [17]:
```
#Print the predictions
print(classifier.predict(X_train))
```

```
[1 1 0 ... 0 0 1]
```

In [18]:
```
#Print the actual values
print(y_train.values)
```

```
[1 1 0 ... 0 0 1]
```

In [19]:
```
#Evaluate the model on the training data set
from sklearn.metrics import classification_report,confusion_matrix, accuracy_s
core
pred = classifier.predict(X_train)
print(classification_report(y_train ,pred ))
print('Confusion Matrix: \n',confusion_matrix(y_train,pred))
print()
print('Accuracy: ', accuracy_score(y_train,pred))
```

```
              precision    recall  f1-score   support

           0       0.98      0.94      0.96      2549
           1       0.94      0.99      0.96      2519

    accuracy                           0.96      5068
   macro avg       0.96      0.96      0.96      5068
weighted avg       0.96      0.96      0.96      5068

Confusion Matrix:
 [[2397  152]
 [  37 2482]]

Accuracy:  0.962707182320442
```

In [20]:
```
 #Print the predictions
print('Predicted value: ',classifier.predict(X_test))
```

```
Predicted value:  [1 1 0 ... 1 1 1]
```

In [21]:
```
#Print Actual Label
print('Actual value: ',y_test.values)
```

```
Actual value:  [1 0 0 ... 0 1 1]
```

In [22]:
```
#Evaluate the model on the test data set
from sklearn.metrics import classification_report,confusion_matrix, accuracy_s
core
pred = classifier.predict(X_test)
print(classification_report(y_test ,pred ))

print('Confusion Matrix: \n', confusion_matrix(y_test,pred))
print()
print('Accuracy: ', accuracy_score(y_test,pred))
```

```
              precision    recall  f1-score   support

           0       0.94      0.86      0.90       615
           1       0.88      0.94      0.91       652

    accuracy                           0.90      1267
   macro avg       0.91      0.90      0.90      1267
weighted avg       0.91      0.90      0.90      1267

Confusion Matrix:
 [[529  86]
 [ 36 616]]

Accuracy:  0.9037095501183899
```

In [23]:
```
from sklearn.ensemble import RandomForestClassifier
```

In [24]:
```
clf1 = RandomForestClassifier(random_state=1)
clf1.fit(X_train, y_train)
```

```
C:\Users\Yash\Anaconda3\lib\site-packages\sklearn\ensemble\forest.py:245: Fut
ureWarning: The default value of n_estimators will change from 10 in version
0.20 to 100 in 0.22.
  "10 in version 0.20 to 100 in 0.22.", FutureWarning)
```

Out[24]:
```
RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini',
                       max_depth=None, max_features='auto', max_leaf_nodes=No
ne,
                       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=10,
                       n_jobs=None, oob_score=False, random_state=1, verbose=
0,
                       warm_start=False)
```

In [25]:
```
#Print the predictions
print(clf1.predict(X_train))
```

```
[1 1 0 ... 0 0 1]
```

In [26]:
```
#Print the actual values
print(y_train.values)
```

```
[1 1 0 ... 0 0 1]
```

In [27]:
```
#Evaluate the model on the training data set
from sklearn.metrics import classification_report,confusion_matrix, accuracy_s
core
pred = clf1.predict(X_train)
print(classification_report(y_train ,pred ))
print('Confusion Matrix: \n',confusion_matrix(y_train,pred))
print()
print('Accuracy: ', accuracy_score(y_train,pred))
```

```
              precision    recall  f1-score   support

           0       0.99      1.00      1.00      2549
           1       1.00      0.99      1.00      2519

    accuracy                           1.00      5068
   macro avg       1.00      1.00      1.00      5068
weighted avg       1.00      1.00      1.00      5068

Confusion Matrix:
 [[2545    4]
 [  20 2499]]

Accuracy:  0.9952644041041832
```

In [28]:
```
 #Print the predictions
print('Predicted value: ',clf1.predict(X_test))
```

```
Predicted value:  [1 0 0 ... 0 1 0]
```

In [29]:
```
#Print Actual Label
print('Actual value: ',y_test.values)
```

```
Actual value:  [1 0 0 ... 0 1 1]
```

In [30]:
```
#Evaluate the model on the test data set
from sklearn.metrics import classification_report,confusion_matrix, accuracy_s
core
pred = clf1.predict(X_test)
print(classification_report(y_test ,pred ))

print('Confusion Matrix: \n', confusion_matrix(y_test,pred))
print()
print('Accuracy: ', accuracy_score(y_test,pred))
```

```
              precision    recall  f1-score   support

           0       0.83      0.90      0.86       615
           1       0.89      0.82      0.86       652

    accuracy                           0.86      1267
   macro avg       0.86      0.86      0.86      1267
weighted avg       0.86      0.86      0.86      1267

Confusion Matrix:
 [[551  64]
 [116 536]]

Accuracy:  0.8579321231254933
```

In [31]:
```
from sklearn.ensemble import BaggingClassifier,AdaBoostClassifier,VotingClassi
fier
from sklearn.tree import DecisionTreeClassifier
from sklearn import model_selection
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
```

In [123]:
```
#descision tree
dt = DecisionTreeClassifier(criterion='gini',max_depth=None)
dt.fit(X_train,y_train)
```

Out[123]:
```
DecisionTreeClassifier(class_weight=None, criterion='gini', 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=False,
                       random_state=None, splitter='best')
```

In [33]:
```
# dt.score(X_test,y_test)
#Print the predictions
print(dt.predict(X_train))
```

```
[1 1 0 ... 0 0 1]
```

In [34]:
```
# dt.score(X_train,y_train)
#Print the actual values
print(y_train.values)
```

```
[1 1 0 ... 0 0 1]
```

In [35]:
```
predictions = dt.predict(X_test)
```

In [36]:
```
from sklearn.metrics import classification_report,confusion_matrix
```

In [37]:
```
print(classification_report(y_test,predictions))
```

```
              precision    recall  f1-score   support

           0       0.86      0.84      0.85       615
           1       0.85      0.87      0.86       652

    accuracy                           0.86      1267
   macro avg       0.86      0.86      0.86      1267
weighted avg       0.86      0.86      0.86      1267
```

In [38]:
```
cm=confusion_matrix(y_test,predictions)
print(cm)
print ("Accuracy of prediction:",accuracy_score(y_test,pred))
```

```
[[516  99]
 [ 83 569]]
Accuracy of prediction: 0.8579321231254933
```

In [39]:
```
#Evaluate the model on the training data set
from sklearn.metrics import classification_report,confusion_matrix, accuracy_s
core
pred = dt.predict(X_train)
print(classification_report(y_train ,pred ))
print('Confusion Matrix: \n',confusion_matrix(y_train,pred))
print()
print('Accuracy: ', accuracy_score(y_train,pred))
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00      2549
           1       1.00      1.00      1.00      2519

    accuracy                           1.00      5068
   macro avg       1.00      1.00      1.00      5068
weighted avg       1.00      1.00      1.00      5068

Confusion Matrix:
 [[2549    0]
 [   0 2519]]

Accuracy:  1.0
```

In [40]:
```
 #Print the predictions
print('Predicted value: ',dt.predict(X_test))
```

```
Predicted value:  [1 0 0 ... 0 1 1]
```

In [41]:
```
#Print Actual Label
print('Actual value: ',y_test.values)
```

```
Actual value:  [1 0 0 ... 0 1 1]
```

In [42]:
```
#Evaluate the model on the test data set
from sklearn.metrics import classification_report,confusion_matrix, accuracy_s
core
pred = dt.predict(X_test)
print(classification_report(y_test ,pred ))

print('Confusion Matrix: \n', confusion_matrix(y_test,pred))
print()
print('Accuracy: ', accuracy_score(y_test,pred))
```

```
              precision    recall  f1-score   support

           0       0.86      0.84      0.85       615
           1       0.85      0.87      0.86       652

    accuracy                           0.86      1267
   macro avg       0.86      0.86      0.86      1267
weighted avg       0.86      0.86      0.86      1267

Confusion Matrix:
 [[516  99]
 [ 83 569]]

Accuracy:  0.856353591160221
```

In [49]:
```
#Bagging

bg = BaggingClassifier(DecisionTreeClassifier(), max_samples= 0.5, max_feature
s = 1.0, n_estimators = 20)
bg.fit(X_train,y_train)
```

Out[49]:
```
BaggingClassifier(base_estimator=DecisionTreeClassifier(class_weight=None,
                                                        criterion='gini',
                                                        max_depth=None,
                                                        max_features=None,
                                                        max_leaf_nodes=None,
                                                        min_impurity_decrease
=0.0,
                                                        min_impurity_split=No
ne,
                                                        min_samples_leaf=1,
                                                        min_samples_split=2,
                                                        min_weight_fraction_l
eaf=0.0,
                                                        presort=False,
                                                        random_state=None,
                                                        splitter='best'),
                  bootstrap=True, bootstrap_features=False, max_features=1.0,
                  max_samples=0.5, n_estimators=20, n_jobs=None,
                  oob_score=False, random_state=None, verbose=0,
                  warm_start=False)
```

In [50]:
```
bg.score(X_test,y_test)
```

Out[50]:
```
0.8950276243093923
```

In [51]:
```
bg.score(X_train,y_train)
```

Out[51]:
```
0.9704025256511445
```

In [52]:
```
#Boosting - Ada Boost

adb = AdaBoostClassifier(DecisionTreeClassifier(),n_estimators = 5, learning_r
ate = 1)
adb.fit(X_train,y_train)
```

Out[52]:
```
AdaBoostClassifier(algorithm='SAMME.R',
                   base_estimator=DecisionTreeClassifier(class_weight=None,
                                                         criterion='gini',
                                                         max_depth=None,
                                                         max_features=None,
                                                         max_leaf_nodes=None,
                                                         min_impurity_decreas
e=0.0,
                                                         min_impurity_split=N

one,
                                                         min_samples_leaf=1,
                                                         min_samples_split=2,
                                                         min_weight_fraction_

leaf=0.0,
                                                         presort=False,
                                                         random_state=None,
                                                         splitter='best'),
                   learning_rate=1, n_estimators=5, random_state=None)
```

In [53]:
```
adb.score(X_test,y_test)
```

Out[53]:
```
0.8484609313338595
```

In [105]:
```
adb.score(X_train,y_train)
```

Out[105]:
```
1.0
```

In [55]:
```
pip install mlxtend
```

```
Requirement already satisfied: mlxtend in c:\users\yash\anaconda3\lib\site-pa
ckages (0.17.2)
Requirement already satisfied: matplotlib>=3.0.0 in c:\users\yash\anaconda3\l
ib\site-packages (from mlxtend) (3.1.1)
Requirement already satisfied: joblib>=0.13.2 in c:\users\yash\anaconda3\lib
\site-packages (from mlxtend) (0.13.2)
Requirement already satisfied: setuptools in c:\users\yash\anaconda3\lib\site
-packages (from mlxtend) (41.4.0)
Requirement already satisfied: scikit-learn>=0.20.3 in c:\users\yash\anaconda
3\lib\site-packages (from mlxtend) (0.21.3)
Requirement already satisfied: scipy>=1.2.1 in c:\users\yash\anaconda3\lib\si
te-packages (from mlxtend) (1.3.1)
Requirement already satisfied: pandas>=0.24.2 in c:\users\yash\anaconda3\lib
\site-packages (from mlxtend) (0.25.1)
Requirement already satisfied: numpy>=1.16.2 in c:\users\yash\anaconda3\lib\s
ite-packages (from mlxtend) (1.16.5)
Requirement already satisfied: cycler>=0.10 in c:\users\yash\anaconda3\lib\si
te-packages (from matplotlib>=3.0.0->mlxtend) (0.10.0)
Requirement already satisfied: kiwisolver>=1.0.1 in c:\users\yash\anaconda3\l
ib\site-packages (from matplotlib>=3.0.0->mlxtend) (1.1.0)
Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.1 in
c:\users\yash\anaconda3\lib\site-packages (from matplotlib>=3.0.0->mlxtend)
(2.4.2)
Requirement already satisfied: python-dateutil>=2.1 in c:\users\yash\anaconda
3\lib\site-packages (from matplotlib>=3.0.0->mlxtend) (2.8.0)
Requirement already satisfied: pytz>=2017.2 in c:\users\yash\anaconda3\lib\si
te-packages (from pandas>=0.24.2->mlxtend) (2019.3)
Requirement already satisfied: six in c:\users\yash\anaconda3\lib\site-packag
es (from cycler>=0.10->matplotlib>=3.0.0->mlxtend) (1.12.0)
Note: you may need to restart the kernel to use updated packages.
```

In [56]:
```
from mlxtend.classifier import StackingClassifier
from sklearn.neighbors import KNeighborsClassifier
```

In [57]:
```
import warnings

warnings.simplefilter('ignore')
```

In [120]:
```
clf1 = KNeighborsClassifier(n_neighbors=1)
clf2 = RandomForestClassifier(random_state=1)
clf3 = MultinomialNB()
lr   = LogisticRegression()
sclf = StackingClassifier(classifiers=[clf1, clf2, clf3],
                          meta_classifier=lr)
```

In [122]:
```
print('4-fold cross validation:\n')

for clf, label in zip([clf1, clf2, clf3,lr, sclf],
                      ['KNN',
                       'Random Forest',
                       'Naive Bayes',
                       'StackingClassifier','Logistic Regression']):

    scores = model_selection.cross_val_score(clf, X_train, y_train,cv=4, scori
ng='accuracy')
    print("Accuracy: %0.2f (+/- %0.2f) [%s]"
          % (scores.mean(), scores.std(), label))
```

```
4-fold cross validation:

Accuracy: 0.79 (+/- 0.01) [KNN]
Accuracy: 0.84 (+/- 0.01) [Random Forest]
Accuracy: 0.91 (+/- 0.01) [Naive Bayes]
Accuracy: 0.93 (+/- 0.01) [StackingClassifier]
Accuracy: 0.80 (+/- 0.02) [Logistic Regression]
```