Detecting Fake News

Objective

Based on the existing dataset (news.csv) which has a shape of (7796, 4), we build a model to be able to accurately classify a news to see whether it is FAKE or REAL.

In [27]:
```
# Necessary imports as the first step
import numpy as np
import pandas as pd
import itertools
from sklearn.model_selection import train_test_split
# TfidfVectorizer (tf= term frequency; idf= inverse document frequency)
# a collection of raw documents is turned to a matrix of tf-idf features
from sklearn.feature_extraction.text import TfidfVectorizer
# Passive Aggressive Classifier
from sklearn.linear_model import PassiveAggressiveClassifier
from sklearn.metrics import accuracy_score, confusion_matrix
```

In [28]:
```
# Read the data
df = pd.read_csv('news.csv')
#Get shape and head
df.shape
df.head()
```

Out[28]:

	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 [29]:
```
# Get the labels
labels=df.label
labels.head()
```

Out[29]:
```
0    FAKE
1    FAKE
2    REAL
3    FAKE
4    REAL
Name: label, dtype: object
```

In [30]:
```
# Split the dataset into training and testing set
x_train, x_test, y_train, y_test = train_test_split(df['text'], labels, test_s
ize=0.2, random_state=7)
```

In [31]:
```
# Initialize a TfidfVectorizer
# stop words from the English language
# a maximum document frequency of 0.7
tf_idf_vectorizer = TfidfVectorizer(stop_words='english', max_df=0.7)
```

In [38]:
```
tf_idf_vectorizer
```

Out[38]:
```
TfidfVectorizer(analyzer='word', binary=False, decode_error='strict',
                dtype=<class 'numpy.float64'>, encoding='utf-8',
                input='content', lowercase=True, max_df=0.7, max_features=Non
e,
                min_df=1, ngram_range=(1, 1), norm='l2', preprocessor=None,
                smooth_idf=True, stop_words='english', strip_accents=None,
                sublinear_tf=False, token_pattern='(?u)\\b\\w\\w+\\b',
                tokenizer=None, use_idf=True, vocabulary=None)
```

In [32]:
```
# Fit and transform train set (x_train)
tfidf_train = tf_idf_vectorizer.fit_transform(x_train)
# transform test set (y_train)
tfidf_test = tf_idf_vectorizer.transform(x_test)
```

In [36]:
```
tfidf_train
```

Out[36]:
```
<5068x61651 sparse matrix of type '<class 'numpy.float64'>'
        with 1337098 stored elements in Compressed Sparse Row format>
```

In [37]:
```
tfidf_test
```

Out[37]:
```
<1267x61651 sparse matrix of type '<class 'numpy.float64'>'
        with 322056 stored elements in Compressed Sparse Row format>
```

In [33]:
```
# Initialize a PassiveAggressiveClassifier
pas = PassiveAggressiveClassifier(max_iter = 50)
pas.fit(tfidf_train, y_train)
```

Out[33]:
```
PassiveAggressiveClassifier(C=1.0, average=False, class_weight=None,
                            early_stopping=False, fit_intercept=True,
                            loss='hinge', max_iter=50, n_iter_no_change=5,
                            n_jobs=None, random_state=None, shuffle=True,
                            tol=0.001, validation_fraction=0.1, verbose=0,
                            warm_start=False)
```

In [34]:
```
# Predict on the test set and calculate accuracy
y_pred=pas.predict(tfidf_test)
score=accuracy_score(y_test,y_pred)
print(f'Accuracy: {round(score*100,2)}%')
```

```
Accuracy: 93.21%
```

In [35]:
```
# Build confusion matrix
confusion_matrix(y_test,y_pred, labels=['FAKE','REAL'])
```

Out[35]:
```
array([[592,  46],
       [ 40, 589]], dtype=int64)
```

Therefore, this model concludes that we get:

- **592** true positives
- **589** true negatives
- **40** false positives
- **46** false negatives

And the accuracy to detect a news from this model is **93.21%**