Fake News Detection

- You'll need to install the following libraries with pip:
  `pip install numpy pandas sklearn`
- You'll need to install Jupyter Lab to run code.
- Dataset File name: **newsdataset.csv**
- Source Code: **Fake News.ipynb**

Steps for Detecting fake news with Python

1. Make necessary imports:

```
In [1]: import numpy as np
        import pandas as pd
        import itertools
        from sklearn.model_selection import train_test_split
        from sklearn.feature_extraction.text import TfidfVectorizer
        from sklearn.linear_model import PassiveAggressiveClassifier
        from sklearn.metrics import accuracy_score, confusion_matrix
```

2. Now, let's read the data into a data frame, and get the shape of the data and the first 5 records:

```
In [7]: #Read the data
        df=pd.read_csv('C:\\Users\\BUG SLAYERS\\Desktop\\newsdataset.csv')
        #Get shape and head
        df.shape
        df.head()
```

Out[7]:

	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

3. Get the labels from the data frame:

```
In [8]: #Get the labels
        labels=df.label
        labels.head()
```

```
Out[8]: 0    FAKE
        1    FAKE
        2    REAL
        3    FAKE
        4    REAL
        Name: label, dtype: object
```

4. Split the dataset into training and testing sets:

```
In [9]: #Split the dataset
        x_train,x_test,y_train,y_test=train_test_split(df['text'], labels, test_size=0.2, random_state=7)
```

5. **Initialize a TfidfVectorizer** with stop words from the English language and a maximum document frequency of 0.7 (terms with a higher document frequency will be discarded). Stop words are the most common words in a language that are to be filtered out before processing the natural language data. And a TfidfVectorizer turns a collection of raw documents into a matrix of TF-IDF features.
   Now, **fit and transform the vectorizer on the train set, and transform the vectorizer on the test set:**

```
In [10]: #Initialize a TfidfVectorizer
         tfidf_vectorizer=TfidfVectorizer(stop_words='english', max_df=0.7)
         #Fit and transform train set, transform test set
         tfidf_train=tfidf_vectorizer.fit_transform(x_train)
         tfidf_test=tfidf_vectorizer.transform(x_test)
```

6. We'll **initialize a Passive Aggressive Classifier.** This is. We'll fit this on tfidf_train and y_train.

   Then, we'll **predict on the test set from the Tfidf Vectorizer and calculate the accuracy with accuracy_score() from sklearn.metrics:**

```
In [11]: #Initialize a Passive Aggressive Classifier
         pac=PassiveAggressiveClassifier(max_iter=50)
         pac.fit(tfidf_train,y_train)
         #Predict on the test set and calculate accuracy
         y_pred=pac.predict(tfidf_test)
         score=accuracy_score(y_test,y_pred)
         print(f'Accuracy: {round(score*100,2)}%')

         C:\ProgramData\Anaconda3\lib\site-packages\sklearn\linear_model\stochastic_gradient.py:183: FutureWarning: max_iter and tol
         parameters have been added in PassiveAggressiveClassifier in 0.19. If max_iter is set but tol is left unset, the default val
         ue for tol in 0.19 and 0.20 will be None (which is equivalent to -infinity, so it has no effect) but will change in 0.21 to
         1e-3. Specify tol to silence this warning.
           FutureWarning)

         Accuracy: 92.74%
```

7. **We got an accuracy of 92.74% with this model. Finally, let's print out a confusion matrix to gain insight into the number of false and true negatives and positives:**

```
In [12]: #Build confusion matrix
         confusion_matrix(y_test,y_pred, labels=['FAKE','REAL'])

Out[12]: array([[587,  51],
                [ 41, 588]], dtype=int64)
```

So with this model, we have **589 true positives**, **587 true negatives**, **42 false positives**, and **49 false negatives.**