Library imports

In [49]:
```
import pandas as pd
```

Reading the csv file and creating a dataframe

In [50]:
```
df = pd.read_csv("news.csv")
```

In [51]:
```
df.head()
```

Out[51]:

	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

x represents the Feature and y represents the Label / Target

In [52]:
```
x = df["text"]
y = df["label"]
```

In [53]:
```
print(x)
```

```
0       Daniel Greenfield, a Shillman Journalism Fello...
1       Google Pinterest Digg Linkedin Reddit Stumbleu...
2       U.S. Secretary of State John F. Kerry said Mon...
3       — Kaydee King (@KaydeeKing) November 9, 2016 T...
4       It's primary day in New York and front-runners...
                              ...
6330    The State Department told the Republican Natio...
6331    The ‘P’ in PBS Should Stand for ‘Plutocratic’ ...
6332     Anti-Trump Protesters Are Tools of the Oligar...
6333    ADDIS ABABA, Ethiopia —President Obama convene...
6334    Jeb Bush Is Suddenly Attacking Trump. Here's W...
Name: text, Length: 6335, dtype: object
```

In [54]:
```
print(y)
```

```
0       FAKE
1       FAKE
2       REAL
3       FAKE
4       REAL
        ...
6330    REAL
6331    FAKE
6332    FAKE
6333    REAL
6334    REAL
Name: label, Length: 6335, dtype: object
```

train_test_split : to decide the data to be considered for training and testing

In [55]:
```
from sklearn.model_selection import train_test_split
x_train , x_test , y_train , y_test = train_test_split(x , y , test_size=0.2 , random_state = 7)
```

TfidfVectorizer :

to convert a collection of raw documents or text into a matrix of TF-IDF features represented in a numerical format

In [56]:
```
from sklearn.feature_extraction.text import TfidfVectorizer

tfidf = TfidfVectorizer(stop_words = "english" , max_df = 0.7)

x_train = tfidf.fit_transform(x_train)
x_test = tfidf.transform(x_test)
```

In [57]:
```
print(x_test)
```

```
(0, 60731)    0.05899712902382916
(0, 60684)    0.033385466151529625
(0, 60271)    0.04581143542258741
(0, 60261)    0.07937859313949312
(0, 59116)    0.10997273171965094
(0, 59036)    0.08042180974421559
(0, 58654)    0.07128159375531905
(0, 58335)    0.0678398429566027
(0, 57086)    0.12429244186413906
(0, 55170)    0.20939665348422057
```

In [58]:
```
print(x_train)
```

```
(0, 56381)    0.03622223988286098
(0, 16314)    0.053492157980948106
(0, 19620)    0.030351855107005405
(0, 52607)    0.04266045446208797
(0, 14900)    0.039165339742818085
(0, 53749)    0.029756205182552464
(0, 15211)    0.07772572986248194
(0, 61154)    0.06726619958695557
(0, 59042)    0.047893261248723944
(0, 42972)    0.03152542343098286
(0, 54232)    0.038673616329284524
(0, 59249)    0.04106143649018827
(0, 28891)    0.06514397995138038
(0, 41708)    0.03983513460128018
(0, 50192)    0.045331181477256094
(0, 44691)    0.0318676439567658
(0, 11820)    0.046381950858248124
(0, 7682)     0.04137048243377956
(0, 50343)    0.10196965191544219
(0, 48095)    0.021092647294770877
(0, 17916)    0.03674587236023286
(0, 46027)    0.10236534701241509
(0, 16993)    0.02775494464904786
(0, 55006)    0.03368300200002207
(0, 51389)    0.03397042876291898
:     :
(5067, 32909) 0.09429823872256275
(5067, 59221) 0.11305513144362901
(5067, 14649) 0.03772971846597005
(5067, 55827) 0.2218263076177088
(5067, 10398) 0.029198031075976353
(5067, 46158) 0.037013973826002855
(5067, 60684) 0.022935168393493133
(5067, 53139) 0.025628375412833703
(5067, 14556) 0.04989667741743244
(5067, 59249) 0.09370008504693801
(5067, 48095) 0.09626467139586602
(5067, 54706) 0.02438296419332449
(5067, 54235) 0.18013712617861882
(5067, 23649) 0.11715980750719378
(5067, 60291) 0.02511088091736429
(5067, 30025) 0.07803429885512414
(5067, 4919)  0.039597295646358985
(5067, 24041) 0.031458613144788115
(5067, 51148) 0.03759621365205542
(5067, 47648) 0.051281746021764794
(5067, 40793) 0.12848334208123888
(5067, 57811) 0.028627812267104712
(5067, 51968) 0.04662099560930727
(5067, 57236) 0.053981424497667 98
```

Model Training / Fitting : to train the model using Passive Aggressive Classifier classification algorithm

In [59]:
```
from sklearn.linear_model import PassiveAggressiveClassifier

pac = PassiveAggressiveClassifier()
pac.fit(x_train,y_train)
```

Out[59]:
```
PassiveAggressiveClassifier()
```

Model Testing : predicting whether the news is real or fake

In [60]:
```
y_pred = pac.predict(x_test)
```

Comparison between y_test and y_pred :

to compare between the actual truth and the prediction of whether the news is real or fake as per the given dataset

In [61]:
```
print("Actual Vs Predicted classification is as follows : ")
comparison = pd.DataFrame({"Actual" : y_test , "Predicted" : y_pred})

comparison.head(10)
```

```
Actual Vs Predicted classification is as follows :
```

Out[61]:

	Actual	Predicted
3534	REAL	REAL
6265	FAKE	FAKE
3123	REAL	REAL
3940	REAL	REAL
2856	REAL	REAL
3031	REAL	REAL
4854	REAL	REAL
5861	REAL	REAL
307	REAL	REAL
2956	FAKE	FAKE

To verify the accuracy of the model by the following metrics:

1. Confusion Matrix

2. Accuracy Score

In [62]:
```
from sklearn import metrics

# 1.   Confusion Matrix
print(metrics.confusion_matrix(y_test,y_pred))

# 2.   Accuracy Score
print(metrics.accuracy_score(y_test,y_pred) * 100)
```

```
[[591  47]
 [ 43 586]]
92.89660615627466
```

Hence, the model has an accuracy score of 92.897 %