# information of the dataset :

1. id: unique id for a news article.
2. title: the title of a news article.
3. author: author of the news article.
4. text: the text of the article; could be incomplete.
5. lable: a label that marks whether the news article is real or fake.

```
1: Fake News
0: Real News
```

# Importing the Dependencies

In [7]:

```python
import pandas as pd
import numpy as np
import re
from nltk.corpus import stopwords
from nltk.stem.porter import PorterStemmer
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
```

In [8]:

```python
import nltk
nltk.download('stopwords')
```

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

Out[8]:

```
True
```

In [9]:

```python
# PRINTING THE STOPWORDS OF THE ENGLISH LANGUAGE

print(stopwords.words('english'))
```

```
['i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves', 'you', "you'r
e", "you've", "you'll", "you'd", 'your', 'yours', 'yourself', 'yourselves',
'he', 'him', 'his', 'himself', 'she', "she's", 'her', 'hers', 'herself', 'i
t', "it's", 'its', 'itself', 'they', 'them', 'their', 'theirs', 'themselves',
'what', 'which', 'who', 'whom', 'this', 'that', "that'll", 'these', 'those',
'am', 'is', 'are', 'was', 'were', 'be', 'been', 'being', 'have', 'has', 'ha
d', 'having', 'do', 'does', 'did', 'doing', 'a', 'an', 'the', 'and', 'but',
'if', 'or', 'because', 'as', 'until', 'while', 'of', 'at', 'by', 'for', 'wit
h', 'about', 'against', 'between', 'into', 'through', 'during', 'before', 'af
ter', 'above', 'below', 'to', 'from', 'up', 'down', 'in', 'out', 'on', 'off',
'over', 'under', 'again', 'further', 'then', 'once', 'here', 'there', 'when',
'where', 'why', 'how', 'all', 'any', 'both', 'each', 'few', 'more', 'most',
'other', 'some', 'such', 'no', 'nor', 'not', 'only', 'own', 'same', 'so', 'th
an', 'too', 'very', 's', 't', 'can', 'will', 'just', 'don', "don't", 'shoul
d', "should've", 'now', 'd', 'll', 'm', 'o', 're', 've', 'y', 'ain', 'aren',
"aren't", 'couldn', "couldn't", 'didn', "didn't", 'doesn', "doesn't", 'hadn',
"hadn't", 'hasn', "hasn't", 'haven', "haven't", 'isn', "isn't", 'ma', 'might
n', "mightn't", 'mustn', "mustn't", 'needn', "needn't", 'shan', "shan't", 'sh
ouldn', "shouldn't", 'wasn', "wasn't", 'weren', "weren't", 'won', "won't", 'w
ouldn', "wouldn't"]
```

# DATA PRE-PROCESSING

In [10]:
```
#  Loading the dataset to the Pandas DataFrame
news_dataset = pd.read_csv(r"C:\Users\Aditya\Downloads\train\Train.csv")
```

In [11]:
```
news_dataset.shape
```

Out[11]:
```
(20800, 5)
```

In [12]:
```
# print the first 5 rows of the DataFrame
news_dataset.head(5)
```

Out[12]:

| | id | title | author | text | label |
|---|---|---|---|---|---|
| **0** | 0 | House Dem Aide: We Didn’t Even See Comey’s Let... | Darrell Lucus | House Dem Aide: We Didn’t Even See Comey’s Let... | 1 |
| **1** | 1 | FLYNN: Hillary Clinton, Big Woman on Campus - ... | Daniel J. Flynn | Ever get the feeling your life circles the rou... | 0 |
| **2** | 2 | Why the Truth Might Get You Fired | Consortiumnews.com | Why the Truth Might Get You Fired October 29, ... | 1 |
| **3** | 3 | 15 Civilians Killed In Single US Airstrike Hav... | Jessica Purkiss | Videos 15 Civilians Killed In Single US Airstr... | 1 |
| **4** | 4 | Iranian woman jailed for fictional unpublished... | Howard Portnoy | Print \nAn Iranian woman has been sentenced to... | 1 |

In [13]:
```
# Counting the number of the missing data values from the dataset
news_dataset.isnull().sum()
```

Out[13]:
```
id           0
title      558
author    1957
text        39
label        0
dtype: int64
```

In [14]:
```
#   Replacing the null values with empty string
news_dataset = news_dataset.fillna('')
```

In [15]:
```
#  Merging the Author name and News title
news_dataset['content'] = news_dataset['author'] + ' '+news_dataset['title']
```

In [16]:
```
print(news_dataset['content'])
```

```
0        Darrell Lucus House Dem Aide: We Didn’t Even S...
1        Daniel J. Flynn FLYNN: Hillary Clinton, Big Wo...
2        Consortiumnews.com Why the Truth Might Get You...
3        Jessica Purkiss 15 Civilians Killed In Single ...
4        Howard Portnoy Iranian woman jailed for fictio...
                               ...                        
20795    Jerome Hudson Rapper T.I.: Trump a ’Poster Chi...
20796    Benjamin Hoffman N.F.L. Playoffs: Schedule, Ma...
20797    Michael J. de la Merced and Rachel Abrams Macy...
20798    Alex Ansary NATO, Russia To Hold Parallel Exer...
20799              David Swanson What Keeps the F-35 Alive
Name: content, Length: 20800, dtype: object
```

In [17]:
```
#  Separating the data & label
X = news_dataset.drop(columns = 'label', axis = 1)
Y = news_dataset['label']
```

In [18]:

```
print(X)
print(Y)
```

```
          id                                              title  \
0          0  House Dem Aide: We Didn’t Even See Comey’s Let...
1          1  FLYNN: Hillary Clinton, Big Woman on Campus - ...
2          2                  Why the Truth Might Get You Fired
3          3  15 Civilians Killed In Single US Airstrike Hav...
4          4  Iranian woman jailed for fictional unpublished...
...      ...                                                ...
20795  20795  Rapper T.I.: Trump a ’Poster Child For White S...
20796  20796  N.F.L. Playoffs: Schedule, Matchups and Odds -...
20797  20797  Macy’s Is Said to Receive Takeover Approach by...
20798  20798  NATO, Russia To Hold Parallel Exercises In Bal...
20799  20799                          What Keeps the F-35 Alive

                                          author  \
0                                  Darrell Lucus
1                                Daniel J. Flynn
2                             Consortiumnews.com
3                                Jessica Purkiss
4                                 Howard Portnoy
...                                          ...
20795                              Jerome Hudson
20796                           Benjamin Hoffman
20797  Michael J. de la Merced and Rachel Abrams
20798                                Alex Ansary
20799                              David Swanson

                                                    text  \
0      House Dem Aide: We Didn’t Even See Comey’s Let...
1      Ever get the feeling your life circles the rou...
2      Why the Truth Might Get You Fired October 29, ...
3      Videos 15 Civilians Killed In Single US Airstr...
4      Print \nAn Iranian woman has been sentenced to...
...                                                  ...
20795  Rapper T. I. unloaded on black celebrities who...
20796  When the Green Bay Packers lost to the Washing...
20797  The Macy’s of today grew from the union of sev...
20798  NATO, Russia To Hold Parallel Exercises In Bal...
20799    David Swanson is an author, activist, journa...

                                                 content
0      Darrell Lucus House Dem Aide: We Didn’t Even S...
1      Daniel J. Flynn FLYNN: Hillary Clinton, Big Wo...
2      Consortiumnews.com Why the Truth Might Get You...
3      Jessica Purkiss 15 Civilians Killed In Single ...
4      Howard Portnoy Iranian woman jailed for fictio...
...                                                  ...
20795  Jerome Hudson Rapper T.I.: Trump a ’Poster Chi...
20796  Benjamin Hoffman N.F.L. Playoffs: Schedule, Ma...
20797  Michael J. de la Merced and Rachel Abrams Macy...
20798  Alex Ansary NATO, Russia To Hold Parallel Exer...
20799             David Swanson What Keeps the F-35 Alive

[20800 rows x 5 columns]
0        1
1        0
2        1
3        1
4        1
        ..
20795    0
20796    0
20797    0
20798    1
20799    1
Name: label, Length: 20800, dtype: int64
```

## Stemming :

```
Stemming :
    Stemming is the process of reducing  word to its Root word

    Example :
    actor, actress , acting --> act
```

In [19]:
```python
port_stem = PorterStemmer()
```

In [20]:
```python
def stemming(content):
    stemmed_content = re.sub('[^a-zA-Z]',' ',content)
    stemmed_content = stemmed_content.lower()
    stemmed_content = stemmed_content.split()
    stemmed_content = [port_stem.stem(word) for word in stemmed_content if not
    stemmed_content = ' '.join(stemmed_content)
    return stemmed_content
```

In [21]:
```python
 news_dataset['content'] = news_dataset['content'].apply(stemming)
```

In [22]:
```python
print(news_dataset['content'])
```

```
0              : ’ ’
1            . : , -
2                  .
3                 15
4
            ...
20795       . .: ’ ’
20796    . . . : , -
20797        . ’ ’ -
20798              ,
20799            -35
Name: content, Length: 20800, dtype: object
```

In [23]:
```python
# separating the data and label
X = news_dataset['content'].values
Y = news_dataset['label'].values
```

In [24]:
```python
print(X)
```

```
[': ’ ’' '. : , -' '.' ... '. ’ ’ -' ',' '-35']
```

In [25]:
```python
print(Y)
```

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

In [26]:
```python
Y.shape
```

Out[26]:
```
(20800,)
```

In [27]:
```python
X.shape
```

Out[27]:
```
(20800,)
```

In [28]:
```python
# converting the textual data to numerical data
vectorizer = TfidfVectorizer()
vectorizer.fit(X)

X = vectorizer.transform(X)
```

In [29]:
```
print(X)
```

```
  (3, 62)	1.0
  (27, 62)	1.0
  (31, 145)	0.4249057758414498
  (31, 20)	0.49454392483696136
  (31, 7)	0.7582093299765108
  (36, 146)	0.7333171625424529
  (36, 131)	0.6798867105045412
  (37, 220)	0.8630627144132533
  (37, 145)	0.5050967738855863
  (40, 131)	1.0
  (47, 1204)	0.3691999620908306
  (47, 1124)	0.38440673900713157
  (47, 1058)	0.3691999620908306
  (47, 1040)	0.3691999620908306
  (47, 1014)	0.38440673900713157
  (47, 939)	0.38440673900713157
  (47, 869)	0.38440673900713157
  (53, 1386)	1.0
  (56, 251)	1.0
  (61, 47)	1.0
  (73, 348)	1.0
  (75, 263)	1.0
  (76, 263)	0.7876361228177006
  (76, 1)	0.6161406803910127
  (78, 243)	1.0
  :	:
  (20680, 1)	0.6161406803910127
  (20690, 1097)	0.3159830601044173
  (20690, 1089)	0.3159830601044173
  (20690, 1079)	0.3159830601044173
  (20690, 893)	0.3159830601044173
  (20690, 800)	0.2244087729413534
  (20690, 784)	0.2244087729413534
  (20690, 781)	0.3159830601044173
  (20690, 734)	0.3034830609713406
  (20690, 706)	0.2663660348465984
  (20690, 698)	0.29461417044453925
  (20690, 507)	0.2244087729413534
  (20690, 488)	0.3159830601044173
  (20696, 10)	1.0
  (20702, 167)	0.8587463239988116
  (20702, 1)	0.5124009670351218
  (20710, 140)	1.0
  (20731, 240)	1.0
  (20738, 10)	1.0
  (20743, 249)	1.0
  (20769, 2)	1.0
  (20779, 146)	1.0
  (20782, 226)	0.7472157197300603
  (20782, 221)	0.6645815737652436
  (20799, 212)	1.0
```

# Splitting the dataset to training & test data

In [30]:
```
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size = 0.2, str
```

# Training the Model : Logistic Regression

In [31]:
```
model = LogisticRegression()
```

In [32]:

```
model.fit(X_train, Y_train)
```

Out[32]:

```
▾ LogisticRegression
LogisticRegression()
```

# Evaluation

accuracy score

In [33]:

```
# Accuracy score on the training data
X_train_prediction = model.predict(X_train)
training_data_accuracy = accuracy_score(X_train_prediction, Y_train)
```

In [34]:

```
print('Accuracy score of the training data :', training_data_accuracy)
```

```
Accuracy score of the training data : 0.5409254807692307
```

In [35]:

```
# Accuracy score on the training data
X_test_prediction = model.predict(X_test)
test_data_accuracy = accuracy_score(X_test_prediction, Y_test)
```

In [36]:

```
print('Accuracy score of the test data :', test_data_accuracy)
```

```
Accuracy score of the test data : 0.5228365384615384
```

# Makeing a Predictive System

In [46]:

```
X_new = X_test[6]

prediction = model.predict(X_new)
print(prediction)

if (prediction[0] == 0):
    print('The news is Real ')
else:
    print('The news is Fake')
```

```
[0]
The news is Real
```

In [47]:

```
print(Y_test[6])
```

```
1
```

In [ ]:

```

```