About the dataset:

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

    1: Fake news
    0: Real news
```

In [1]:
```
import numpy as np
import pandas as pd
import re
from nltk.corpus import stopwords
from nltk.stem.porter import PorterStemmer
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.feature_extraction.text import TfidfVectorizer
import warnings
warnings.filterwarnings("ignore")
```

In [2]:
```
import nltk
nltk.download("stopwords")
```

```
[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\s323\AppData\Roaming\nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

Out[2]:
```
True
```

In [3]:
```
print(stopwords.words("English"))
#printing the stop words in english
```

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

Loading dataset and data preprocessing

In [4]:
```
df_train=pd.read_csv(r"C:\Users\s323\Desktop\Gatherings\Data Science\Datasets\trai
df_submit=pd.read_csv(r"C:\Users\s323\Desktop\Gatherings\Data Science\Datasets\subm
```

In [5]:
```
df_train.shape
```

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

In [6]:
```
df_train.head()
```

Out[6]:

	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 [7]:
```
df_train.isnull().sum()
```

Out[7]:
```
id           0
title      558
author    1957
text        39
label        0
dtype: int64
```

In this case we have a very big dataset so we can drop or replace else we had to do various inputation dataset

In [8]:
```
# replacing the null values with empty string
df_train = df_train.fillna("")
```

we will use title and author column to predict news is correct or false, rest are very big and it will take more time to process

In [9]:
```
# merging author name and news title into a new column called content
df_train["content"]=df_train["author"]+""+df_train["title"]
```

In [10]:
```
print(df_train["content"])
```

```
0        Darrell LucusHouse Dem Aide: We Didn’t Even Se...
1        Daniel J. FlynnFLYNN: Hillary Clinton, Big Wom...
2        Consortiumnews.comWhy the Truth Might Get You ...
3        Jessica Purkiss15 Civilians Killed In Single U...
4        Howard PortnoyIranian woman jailed for fiction...
                               ...
20795    Jerome HudsonRapper T.I.: Trump a ’Poster Chil...
20796    Benjamin HoffmanN.F.L. Playoffs: Schedule, Mat...
20797    Michael J. de la Merced and Rachel AbramsMacy’...
20798    Alex AnsaryNATO, Russia To Hold Parallel Exerc...
20799              David SwansonWhat Keeps the F-35 Alive
Name: content, Length: 20800, dtype: object
```

Seperating the data and label

In [11]:
```
X = df_train.drop("label",axis = 1)
Y = df_train["label"]
# we made this show orginal dataset can be introdcued like this as well later on sp
```

Stemming Procedure - Take root word only (remove suffix and prefix to the word)

- It is the most imp feature to do because it will reduce the word to it's root words eg- act,acting,actor,actoress - act
- We had to reduce as much as for better accuracy of the model
- Once we reduce then we do tfidf vectorizer - convert words into numerical value so that we can fill into ML model

In [12]:
```
port_stem = PorterStemmer()
```

In [13]:
```
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 word
    stemmed_content = " ".join(stemmed_content)
    return stemmed_content
```

In [14]:
```
df_train["content"] = df_train["content"].apply(stemming)
```

In [15]:
```
#separating the data and label
X = df_train['content'].values
Y = df_train['label'].values
```

In [16]:
```
print (X)
```

```
['darrel lucushous dem aid even see comey letter jason chaffetz tweet'
 'daniel j flynnflynn hillari clinton big woman campu breitbart'
 'consortiumnew comwhi truth might get fire' ...
 'michael j de la merc rachel abramsmaci said receiv takeov approach hudson bay ne
w york time'
 'alex ansarynato russia hold parallel exercis balkan'
 'david swansonwhat keep f aliv']
```

In [17]:
```
print (Y)
```

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

In [18]:
```
#### Still all the values are in textual formal need to convert into number with th
```

Converting textual data into numerical data

In [19]:
```
vectorizer = TfidfVectorizer()
vectorizer.fit(X)
# Tf - term frequency and if - inverse frequency
```

Out[19]:
```
TfidfVectorizer()
```

In [20]:
```
X = vectorizer.transform(X)
```

In [21]:
```
print (X)
```

```
  (0, 26340)	0.28088379401596425
  (0, 22724)	0.2552336018069161
  (0, 15019)	0.43006226759639316
  (0, 14555)	0.29177259684200296
  (0, 12782)	0.24619727512767195
  (0, 8022)	0.23133661742488731
  (0, 6273)	0.2839932825877813
  (0, 5969)	0.35488202138141456
  (0, 5006)	0.2472595823572816
  (0, 4211)	0.3625320323150658
  (0, 578)	0.2694167078545385
  (1, 27923)	0.36911845953845024
  (1, 11313)	0.24166773097712638
  (1, 8772)	0.5258635625386451
  (1, 5916)	0.31810058109638056
  (1, 4767)	0.23338756776626793
  (1, 3859)	0.45980466668763476
  (1, 3281)	0.18652439327549428
  (1, 2622)	0.3562953366945267
  (2, 26235)	0.3665032495181434
  (2, 16361)	0.43295215406038445
  (2, 9454)	0.30743020569262086
  (2, 8567)	0.3411947414020896
  (2, 5240)	0.40440534260277944
  (2, 5121)	0.5511414848555652
  :	:
  (20797, 25776)	0.08220218573989037
  (20797, 25319)	0.3119640221826561
  (20797, 22086)	0.24902354987792552
  (20797, 20778)	0.2729578683228216
  (20797, 20493)	0.249994989010826
  (20797, 17505)	0.08090456115716123
  (20797, 16315)	0.1785200594251359
  (20797, 16217)	0.3273246827604847
  (20797, 14104)	0.22761807337911874
  (20797, 11692)	0.2992170910232368
  (20797, 6088)	0.21253094503918346
  (20797, 2257)	0.3357782642976524
  (20797, 1249)	0.3072223353708335
  (20797, 72)	0.38829670969848273
  (20798, 21937)	0.2284042880065583
  (20798, 18760)	0.43981843518920394
  (20798, 11434)	0.3219420705942853
  (20798, 8095)	0.40266358130888547
  (20798, 1921)	0.43981843518920394
  (20798, 1081)	0.4638903157542853
  (20798, 697)	0.2827933658592677
  (20799, 25148)	0.6713314187498636
  (20799, 13329)	0.4138037375613909
  (20799, 6018)	0.345590335823275
  (20799, 732)	0.5085743925573473
```

Splitting the dataset to training & test data

In [46]:
```python
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, stratify = Y
```

Training the model

In [47]:
```python
from sklearn.linear_model import LogisticRegression
model = LogisticRegression()
```

In [48]:
```
model.fit(x_train,y_train)
```

Out[48]:
```
LogisticRegression()
```

Accuracy Score

In [49]:
```
from sklearn.metrics import accuracy_score
```

In [50]:
```
x_train_predictions = model.predict(x_train)
```

In [51]:
```
training_data_accuracy = accuracy_score(x_train_predictions,y_train)
```

In [52]:
```
training_data_accuracy
```

Out[52]:
```
0.9719951923076923
```

In [53]:
```
from sklearn.metrics import accuracy_score
```

In [54]:
```
model.score(x_test,y_test)
```

Out[54]:
```
0.9548076923076924
```

In [55]:
```
predictions = model.predict(x_test)
```

In [56]:
```
predictions
```

Out[56]:
```
array([1, 0, 1, ..., 1, 1, 0], dtype=int64)
```

In [57]:
```
# accuracy score on the test data
# X_test_prediction = model.predict(X_test)
# test_data_accuracy = accuracy_score(X_test_prediction, Y_test)
```

In [58]:
```
test_data_accuracy = accuracy_score(predictions, y_test)
```

In [59]:
```
test_data_accuracy
```

Out[59]:
```
0.9548076923076924
```

In [60]:
```
# accuracy score on the test data
x_test_prediction = model.predict(x_test)
test_data_accuracy = accuracy_score(x_test_prediction, y_test)
```

In [61]:
```
test_data_accuracy
```

Out[61]:
```
0.9548076923076924
```

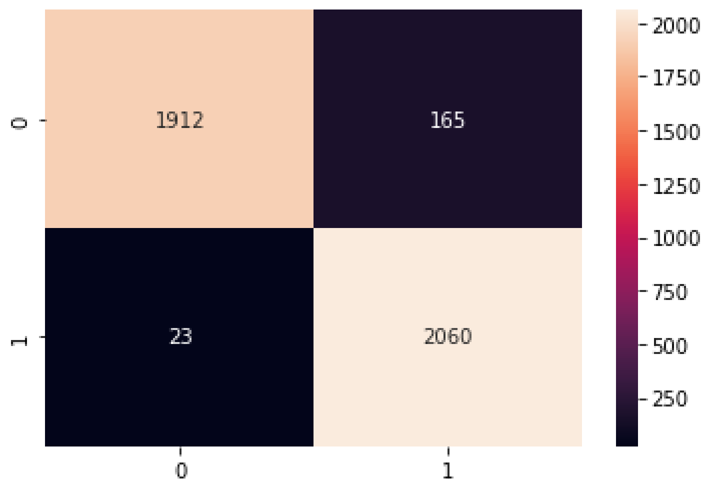
Now prediciction for the unknown data, untrained dataset

In [62]:
```
from sklearn.metrics import confusion_matrix, classification_report
confusion_matrix(y_test, predictions)
```

Out[62]:
```
array([[1912,  165],
       [  23, 2060]], dtype=int64)
```

In [63]:
```
sns.heatmap(confusion_matrix(y_test, predictions),annot=True,fmt="0.0f")
```

Out[63]:
```
<AxesSubplot:>
```



Out of 1872 + 159 = 2031 - 1872 correct and 159 false

In [64]:
```
print(classification_report(y_test, predictions))
```

```
              precision    recall  f1-score   support

           0       0.99      0.92      0.95      2077
           1       0.93      0.99      0.96      2083

    accuracy                           0.95      4160
   macro avg       0.96      0.95      0.95      4160
weighted avg       0.96      0.95      0.95      4160
```

Making a Predictive System

In [69]:
```
X_new = x_test[3]

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 [70]:
```
print(y_test[3])
```

```
0
```

In [ ]: