ML model to tell if a news is fake or it is real

Textual data is used, with labels

fake news -> 1

real news -> 0

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import csv
import sys
```

In [2]:
```
csv.field_size_limit(sys.maxsize)

df= pd.read_csv('fake_news_train.csv',engine='python')
```

In [3]:
```
import re #used for searching text in a doc.
from nltk.corpus import stopwords
# nltk -> natural lang. tool kit, the text with which we are dealing
# corpus -> the body of the text
# stopwords -> words which doesn't add to much value to text ex. the, is
```

In [4]:
```
from nltk.stem.porter import PorterStemmer
# steming ->it takes a word & removes the prefix and sufix to returns root word
```

In [5]:
```
!pip install scikit-learn --upgrade
!pip install --upgrade scikit-learn
```

```
Requirement already satisfied: scikit-learn in /usr/local/lib/python3.10/dist-packages (1.5.2)
Requirement already satisfied: numpy>=1.19.5 in /usr/local/lib/python3.10/dist-packages (from scikit-learn) (1.2
6.4)
Requirement already satisfied: scipy>=1.6.0 in /usr/local/lib/python3.10/dist-packages (from scikit-learn) (1.13
.1)
Requirement already satisfied: joblib>=1.2.0 in /usr/local/lib/python3.10/dist-packages (from scikit-learn) (1.4
.2)
Requirement already satisfied: threadpoolctl>=3.1.0 in /usr/local/lib/python3.10/dist-packages (from scikit-lear
n) (3.5.0)
Requirement already satisfied: scikit-learn in /usr/local/lib/python3.10/dist-packages (1.5.2)
Requirement already satisfied: numpy>=1.19.5 in /usr/local/lib/python3.10/dist-packages (from scikit-learn) (1.2
6.4)
Requirement already satisfied: scipy>=1.6.0 in /usr/local/lib/python3.10/dist-packages (from scikit-learn) (1.13
.1)
Requirement already satisfied: joblib>=1.2.0 in /usr/local/lib/python3.10/dist-packages (from scikit-learn) (1.4
.2)
Requirement already satisfied: threadpoolctl>=3.1.0 in /usr/local/lib/python3.10/dist-packages (from scikit-lear
n) (3.5.0)
```

In [6]:
```
from sklearn.feature_extraction.text import TfidfVectorizer
# converting text data into feature vectore(number)
```

In [7]:
```
from sklearn.model_selection import train_test_split
```

In [8]:
```
import nltk
nltk.download('stopwords')
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Unzipping corpora/stopwords.zip.
```

Out[8]: True

In [9]:
```
print(stopwords.words('english')) #all the stopwords
```

```
['i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves', 'you', "you're", "you've", "you'll", "you'd", 'you
r', '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', 'were', 'be', 'been', 'being', 'have', 'h
as', 'had', 'having', 'do', 'does', 'did', 'doing', 'a', 'an', 'the', 'and', 'but', 'if', 'or', 'because', 'as',
'until', 'while', 'of', 'at', 'by', 'for', 'with', 'about', 'against', 'between', 'into', 'through', 'during', '
before', 'after', 'above', 'below', 'to', 'from', 'up', 'down', 'in', 'out', 'on', 'off', 'over', 'under', 'agai
n', 'further', 'then', 'once', 'here', 'there', 'when', 'where', 'why', 'how', 'all', 'any', 'both', 'each', 'fe
w', 'more', 'most', 'other', 'some', 'such', 'no', 'nor', 'not', 'only', 'own', 'same', 'so', 'than', 'too', 've
ry', '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', "hadn't",
'hasn', "hasn't", 'haven', "haven't", 'isn', "isn't", 'ma', 'mightn', "mightn't", 'mustn', "mustn't", 'needn', "
needn't", 'shan', "shan't", 'shouldn', "shouldn't", 'wasn', "wasn't", 'weren', "weren't", 'won', "won't", 'would
n', "wouldn't"]
```

In [10]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 20800 entries, 0 to 20799
Data columns (total 5 columns):
 #   Column  Non-Null Count  Dtype 
---  ------  --------------  ----- 
 0   id      20800 non-null  int64 
 1   title   20242 non-null  object
 2   author  18843 non-null  object
 3   text    20761 non-null  object
 4   label   20800 non-null  int64 
dtypes: int64(2), object(3)
memory usage: 812.6+ KB
```

In [11]:
```
#df.reset_index(drop=True, inplace=True)
df
```

Out[11]:

	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
...	...	...	...	...	...
20795	20795	Rapper T.I.: Trump a 'Poster Child For White S...	Jerome Hudson	Rapper T. I. unloaded on black celebrities who...	0
20796	20796	N.F.L. Playoffs: Schedule, Matchups and Odds -...	Benjamin Hoffman	When the Green Bay Packers lost to the Washing...	0
20797	20797	Macy’s Is Said to Receive Takeover Approach by...	Michael J. de la Merced and Rachel Abrams	The Macy’s of today grew from the union of sev...	0
20798	20798	NATO, Russia To Hold Parallel Exercises In Bal...	Alex Ansary	NATO, Russia To Hold Parallel Exercises In Bal...	1
20799	20799	What Keeps the F-35 Alive	David Swanson	David Swanson is an author, activist, journa...	1

20800 rows × 5 columns

In [12]:
```
df.isnull().sum()
```

Out[12]:

	0
id	0
title	558
author	1957
text	39
label	0

dtype: int64

In [13]:
```
df.duplicated().sum()
```

Out[13]:
```
0
```

In [14]:
```
df[df['title'].isnull()]
```

Out[14]:

	id	title	author	text	label
53	53	NaN	Dairy✓ TRUMP	Sounds like he has our president pegged. What ...	1
120	120	NaN	Anonymous	Same people all the time , i dont know how you...	1
124	124	NaN	SeekSearchDestory	You know, outside of any morality arguments, i...	1
140	140	NaN	Anonymous	There is a lot more than meets the eye to this...	1
196	196	NaN	Raffie	They got the heater turned up on high.	1
...	...	...	...	...	...
20568	20568	NaN	Cathy Milne	Amusing comment Gary! “Those week!” So, are ...	1
20627	20627	NaN	Ramona	No she doesn’t have more money than God, every...	1
20636	20636	NaN	Dave Lowery	Trump all the way!	1
20771	20771	NaN	Letsbereal	DYN's Statement on Last Week's Botnet Attack h...	1
20772	20772	NaN	beersession	Kinda reminds me of when Carter gave away the ...	1

558 rows × 5 columns

In [15]:
```
df['title']=df['title'].fillna('missing title')
```

In [16]:
```
df[df['id']==20772]
```

Out[16]:

	id	title	author	text	label
20772	20772	missing title	beersession	Kinda reminds me of when Carter gave away the ...	1

In [17]:
```
df['author']=df['author'].fillna('no author')
```

In [18]:
```
df['text']=df['text'].fillna('')
```

For prediction we are going to use title and author, we will not be using text as it would take a lot of time for processing, but if we want we can use text too

we will be combining author and title together

In [19]:
```
#combining title and author
df['joined_content']=df['author']+' '+df['title']
```

In [20]:
```
y=df['label']
```

In [21]:
```
x=df.drop('label',axis=1)
```

Stemming -> process of reducing a word to it's root word

ex-> actor,acting= act would be returned

after the process of stemming, we would be doing

Vectorization -> converting text data to numeric data

In [22]:
```
# stemming
pc=PorterStemmer()
```

In [23]:
```
def stemming(joined_content):
  stemmed_content=re.sub('[^a-zA-Z]',' ' ,joined_content)
  #the above would remove all no.'s from the given text(joined content)

  stemmed_content= stemmed_content.lower()
  #converting all the letters to lower case,for easier processing

  stemmed_content=stemmed_content.split()
  #all the text will be splitted into indiviual words and converted to list

  stemmed_content=[pc.stem(word) for word in stemmed_content if word not in stopwords.words('english')]
  #stem function is applied for each word
  #and only those words are selected which are not included in stopwords

  stemmed_content= ' '.join(stemmed_content)
  # returned words are than joined

  return stemmed_content
```

In [24]:
```
df['joined_content']= df['joined_content'].apply(stemming)
```

```
# stemming function is applied on joined content col
```

In [25]:
```
print(df['joined_content'])
```

```
0        darrel lucu hous dem aid even see comey letter...
1        daniel j flynn flynn hillari clinton big woman...
2                   consortiumnew com truth might get fire
3        jessica purkiss civilian kill singl us airstri...
4        howard portnoy iranian woman jail fiction unpu...
                               ...
20795    jerom hudson rapper trump poster child white s...
20796    benjamin hoffman n f l playoff schedul matchup...
20797    michael j de la merc rachel abram maci said re...
20798    alex ansari nato russia hold parallel exercis ...
20799                            david swanson keep f aliv
Name: joined_content, Length: 20800, dtype: object
```

In [26]:
```
# seperating data and label
x=df['joined_content'].values
y=df['label'].values
```

In [27]:
```
#vectorizing
vectorizer=TfidfVectorizer() # Tf->term frequency, idf-> inverse doc. frequency
#Tf-> it counts the no. of times a word is repeating in a doc.
#idf-> it removes words which doesn't give a significant meaning or repeating no of times

vectorizer.fit(x)
```

Out[27]:
```
TfidfVectorizer
TfidfVectorizer()
```

In [28]:
```
x=vectorizer.transform(x)
```

In [29]:
```
# train and test split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,stratify=y,random_state=2)
```

In [30]:
```
!pip install xgboost
```

```
Requirement already satisfied: xgboost in /usr/local/lib/python3.10/dist-packages (2.1.1)
Requirement already satisfied: numpy in /usr/local/lib/python3.10/dist-packages (from xgboost) (1.26.4)
Requirement already satisfied: nvidia-nccl-cu12 in /usr/local/lib/python3.10/dist-packages (from xgboost) (2.23.
4)
Requirement already satisfied: scipy in /usr/local/lib/python3.10/dist-packages (from xgboost) (1.13.1)
```

In [31]:
```
from xgboost import XGBClassifier
```

In [32]:
```
xgb_model=XGBClassifier()
```

In [33]:
```
xgb_model.fit(x_train,y_train)
```

Out[33]:
```
XGBClassifier
XGBClassifier(base_score=None, booster=None, callbacks=None,
              colsample_bylevel=None, colsample_bynode=None,
              colsample_bytree=None, device=None, early_stopping_rounds=Non
e,
              enable_categorical=False, eval_metric=None, feature_types=Non
e,
              gamma=None, grow_policy=None, importance_type=None,
              interaction_constraints=None, learning_rate=None, max_bin=Non
e,
```

In [34]:
```
y_pred=xgb_model.predict(x_test)
```

In [35]:
```
from sklearn.metrics import accuracy_score
```

In [36]:
```
xgb_accuracy=round(accuracy_score(y_pred,y_test),4)
xgb_accuracy
```

Out[36]:
```
0.9897
```

In [37]:
```
from sklearn.linear_model import LogisticRegression
```

In [38]:
```
log_model=LogisticRegression()
```

In [39]:
```
log_model.fit(x_train,y_train)
```

Out[39]:
```
LogisticRegression
LogisticRegression()
```

In [40]:
```
y_predict=log_model.predict(x_test)
```

In [41]:
```
log_accuracy=round(accuracy_score(y_predict,y_test),4)
log_accuracy
```

Out[41]:
```
0.9837
```

XG classifier will give more accuracy than logistic regression

In [44]:
```
# making predictive system
x_new=x_test[0]

predict=xgb_model.predict(x_new)
print(predict)
if predict[0]==0:
  print('news is real')
else:
  print('news is fake')
```

```
[1]
news is fake
```

In [ ]: