Fake News Detection

- **Group ID : 12**

  Guide : ***Prof. Yogita Fatangare***

  Group Members:

  1. ***Shraddha Bhoite (35009)***
  2. ***Pooja Jadhav (35033)***
  3. ***Prajakta Kank (35040)***
  4. ***Sanket Kurle (35051)***

  PES's MCOE, BE - Information Technology 2021-22

Importing Libraries

```
import numpy as np
import pandas as pd
import seaborn as sns
import re
import nltk
from nltk.corpus import stopwords
from nltk.stem.porter import PorterStemmer
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import f1_score
```

The Dataset

```
data_train = pd.read_csv("traindata.csv")
print("Data shape = ",data_train.shape)
data_train.head()
```

```
Data shape =  (7613, 5)
```

Dropping the not required columns

```
0   1   NaN   NaN   Our Deeds are the Reason of this #earthquake M...   1
```

```
data_train = data_train.drop(['location','keyword'], axis=1)
print("location and keyword columns droped successfully")

data_train = data_train.drop('id', axis=1)
print("id column droped successfully")
data_train.columns
```

```
location and keyword columns droped successfully
id column droped successfully
Index(['text', 'target'], dtype='object')
```

```
data_train
```

	text	target
0	Our Deeds are the Reason of this #earthquake M...	1
1	Forest fire near La Ronge Sask. Canada	1
2	All residents asked to 'shelter in place' are ...	1
3	13,000 people receive #wildfires evacuation or...	1
4	Just got sent this photo from Ruby #Alaska as ...	1
...	...	...
7608	Two giant cranes holding a bridge collapse int...	1
7609	@aria_ahrary @TheTawniest The out of control w...	1
7610	M1.94 [01:04 UTC]?5km S of Volcano Hawaii. htt...	1
7611	Police investigating after an e-bike collided ...	1
7612	The Latest: More Homes Razed by Northern Calif...	1

7613 rows × 2 columns

Create corpus a feature of NLP

```
corpus  = []
pstem = PorterStemmer()
for i in range(data_train['text'].shape[0]):
    #Remove unwanted words
    token = re.sub("[^a-zA-Z]", ' ', data_train['text'][i])
    #Transform words to lowercase
    token = token.lower()
```

```
    token = token.split()
    #Remove stopwords then Stemming it
    token = [pstem.stem(word) for word in token if not word in set(stopwords.words('english')
    token = ' '.join(token)
    #Append cleaned token to corpus
    corpus.append(token)

print("Corpus created successfully")
```

```
Corpus created successfully
```

Create dictionary

```
uniqueWordFrequents = {}
for token in corpus:
    for word in token.split():
        if(word in uniqueWordFrequents.keys()):
            uniqueWordFrequents[word] += 1
        else:
            uniqueWordFrequents[word] = 1

#Convert dictionary to dataFrame
uniqueWordFrequents = pd.DataFrame.from_dict(uniqueWordFrequents,orient='index',columns=['Wor
uniqueWordFrequents.sort_values(by=['Word Frequent'], inplace=True, ascending=False)
uniqueWordFrequents.head(10)
```

	Word Frequent
co	4746
http	4721
like	411
fire	363
amp	344
get	311
bomb	239
new	228
via	220
u	216

```
uniqueWordFrequents['Word Frequent'].unique()
```

```
array([4746, 4721,  411,  363,  344,  311,  239,  228,  220,  216,  213,
        210,  209,  201,  183,  181,  180,  178,  175,  169,  166,  164,
```

```
       162,  156,  155,  153,  151,  145,  144,  143,  137,  133,  132,
       131,  130,  129,  128,  125,  124,  123,  122,  121,  120,  119,
       118,  117,  116,  114,  111,  110,  109,  108,  106,  105,  104,
       103,  102,  101,  100,   99,   98,   97,   96,   95,   94,   93,
        91,   90,   89,   88,   87,   86,   84,   83,   82,   79,   78,
        77,   76,   75,   74,   73,   72,   71,   70,   69,   68,   67,
        66,   65,   64,   63,   62,   61,   60,   59,   58,   57,   56,
        55,   54,   53,   52,   51,   50,   49,   48,   47,   46,   45,
        44,   43,   42,   41,   40,   39,   38,   37,   36,   35,   34,
        33,   32,   31,   30,   29,   28,   27,   26,   25,   24,   23,
        22,   21,   20,   19,   18,   17,   16,   15,   14,   13,   12,
        11,   10,    9,    8,    7,    6,    5,    4,    3,    2,    1],
      dtype=int64)
```

```
uniqueWordFrequents = uniqueWordFrequents[uniqueWordFrequents['Word Frequent'] >= 20]
print(uniqueWordFrequents.shape)
uniqueWordFrequents
```

```
(787, 1)
```

	Word Frequent
co	4746
http	4721
like	411
fire	363
amp	344
...	...
cnn	20
gem	20
captur	20
arriv	20
carri	20

787 rows × 1 columns

Bag of word and CountVectorizer

```
counVec = CountVectorizer(max_features = uniqueWordFrequents.shape[0])
bagOfWords = counVec.fit_transform(corpus).toarray()
```

```
X = bagOfWords
y = data_train['target']
print("X shape = ",X.shape)
```

```
print("y shape = ",y.shape)
X_train , X_test , y_train , y_test = train_test_split(X,y,test_size=0.20, random_state=55, s
print('data splitting successfully')
```

```
X shape =  (7613, 787)
y shape =  (7613,)
data splitting successfully
```

Multinomial DB

```
multinomialNBModel = MultinomialNB(alpha=0.1)
multinomialNBModel.fit(X_train,y_train)
print("multinomialNB model run successfully")
```

```
multinomialNB model run successfully
```

Evaluation Details

```
models = [multinomialNBModel]
for model in models:
····print(type(model).__name__,'·Train·Score·is···:·'·,model.score(X_train,·y_train))
····print(type(model).__name__,'·Test·Score·is····:·'·,model.score(X_test,·y_test))
····
····y_pred·=·model.predict(X_test)
    print(type(model).__name__,' F1 Score is      : ' ,f1_score(y_test,y_pred))
    print('----------------------')
```

```
MultinomialNB  Train Score is   :  0.8022988505747126
MultinomialNB  Test Score is    :  0.7734734077478661
MultinomialNB  F1 Score is      :  0.7165160230073953
----------------------
```