**NAAN MUDHALVAN PROJECT(IBM)**

**IBM AI 101 ARTIFICIAL INTELLIGENCE-GROUP 1**

**Team name: Proj_224823_Team_1**

**Team members: SAI VIGNESH S (reg no 113321106079)**

**THIRUMARAN S (reg no 113321106106)**

**VIGNESH V (reg no 113321106115)**

**LINGESH V(reg no 113321106301)**

**SRINIVASA BHARATH S(reg 113321106094)**

# Problem Statement :

**Design and develop an NLP-based system that can accurately identify and classify news articles or information as either "fake" or "real" by analyzing the textual content, with the primary goal of mitigating the spread of misinformation and promoting the dissemination of trustworthy information.**

# DATASET

- This data set consists of 40000 fake and real news.
- Our goal is to train our model to accurately predict whether a particular piece of news is real or fake.
- Fake and real news data are given in two separate data sets, with each data set consisting of approximately 20000 articles.

# LOADING THE DATA SET

## LOAD REQUIRED LIBRARIES:(program)

```
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
import matplotlib.pyplot as plt
import seaborn as sns

from bs4 import BeautifulSoup
import re
import string
import nltk
from nltk.corpus import stopwords
from nltk.stem import WordNetLemmatizer
from wordcloud import WordCloud, STOPWORDS
from nltk.tokenize import word_tokenize
```

# LOADING THE DATA SET

**LOAD REQUIRED LIBRARIES:(program)**

```
from sklearn.model_selection import train_test_split
from sklearn.pipeline import Pipeline
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier,GradientBoostingClassifier
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
from sklearn.feature_extraction.text import TfidfVectorizer
```

# LOADING THE DATA SET

## IMPORT THE DATASET:

**Input 1:**

```
#import dataset
fake = pd.read_csv("../input/fake-and-real-news-dataset/Fake.csv")
true = pd.read_csv("../input/fake-and-real-news-dataset/True.csv")

#data exploration
fake.head()
```

**Output 1:**

| | title | text | subject | date |
|---|---|---|---|---|
| 0 | Donald Trump Sends Out Embarrassing New Year'... | Donald Trump just couldn t wish all Americans ... | News | December 31, 2017 |
| 1 | Drunk Bragging Trump Staffer Started Russian ... | House Intelligence Committee Chairman Devin Nu... | News | December 31, 2017 |
| 2 | Sheriff David Clarke Becomes An Internet Joke... | On Friday, it was revealed that former Milwauk... | News | December 30, 2017 |
| 3 | Trump Is So Obsessed He Even Has Obama's Name... | On Christmas day, Donald Trump announced that ... | News | December 29, 2017 |
| 4 | Pope Francis Just Called Out Donald Trump Dur... | Pope Francis used his annual Christmas Day mes... | News | December 25, 2017 |

# LOADING THE DATA SET

## DISPLAY THE DATASET :

```
true.head()
```

**Output 2:** :

| | title | text | subject | date |
|---|---|---|---|---|
| 0 | As U.S. budget fight looms, Republicans flip t... | WASHINGTON (Reuters) - The head of a conservat... | politicsNews | December 31, 2017 |
| 1 | U.S. military to accept transgender recruits o... | WASHINGTON (Reuters) - Transgender people will... | politicsNews | December 29, 2017 |
| 2 | Senior U.S. Republican senator: 'Let Mr. Muell... | WASHINGTON (Reuters) - The special counsel inv... | politicsNews | December 31, 2017 |
| 3 | FBI Russia probe helped by Australian diplomat... | WASHINGTON (Reuters) - Trump campaign adviser ... | politicsNews | December 30, 2017 |
| 4 | Trump wants Postal Service to charge 'much mor... | SEATTLE/WASHINGTON (Reuters) - President Donal... | politicsNews | December 29, 2017 |

# LOADING THE DATA SET

## READ & DISPLAY THE DATASET:

```
#add column
true_data['target'] = 1
fake_data['target'] = 0
 true_data.tail()
```

OUTPUT:

| | title | text | subject | date | target |
|---|---|---|---|---|---|
| 21412 | 'Fully committed' NATO backs new U.S. approach... | BRUSSELS (Reuters) - NATO allies on Tuesday we... | worldnews | August 22, 2017 | 1 |
| 21413 | LexisNexis withdrew two products from Chinese ... | LONDON (Reuters) - LexisNexis, a provider of l... | worldnews | August 22, 2017 | 1 |
| 21414 | Minsk cultural hub becomes haven from authorities | MINSK (Reuters) - In the shadow of disused Sov... | worldnews | August 22, 2017 | 1 |
| 21415 | Vatican upbeat on possibility of Pope Francis ... | MOSCOW (Reuters) - Vatican Secretary of State ... | worldnews | August 22, 2017 | 1 |
| 21416 | Indonesia to buy $1.14 billion worth of Russia... | JAKARTA (Reuters) - Indonesia will buy 11 Sukh... | worldnews | August 22, 2017 | 1 |

# DATA PREPROCESSING:

## MISSING VALUES:

- Data cleaning is a very crucial step in any machine learning model, but more so for NLP.
- Without the cleaning process, the dataset is often a cluster of words that the computer doesn't understand.
- Here, It will go over steps done in a typical machine learning text pipeline to clean data.

PROGRAM:

```
#data cleaning
#combining the title and text columns
df['text'] = df['title'] + " " + df['text']
#deleting few columns from the data
del df['title']
del df['subject']
del df['date']
df.head()
```

# DATA PREPROCESSING:

DATA CLEANING:

OUTPUT:

| | text | target |
|---|---|---|
| 0 | politicsNews As U.S. budget fight looms, Repub... | 1 |
| 1 | politicsNews U.S. military to accept transgend... | 1 |
| 2 | politicsNews Senior U.S. Republican senator: '... | 1 |
| 3 | politicsNews FBI Russia probe helped by Austra... | 1 |
| 4 | politicsNews Trump wants Postal Service to cha... | 1 |

# DATA PREPROCESSING:

## REMOVE STOPWORDS:

```
nltk.download("stopwords")
from nltk.corpus import stopwords

# we can use tokenizer instead of split
first_text = nltk.word_tokenize(first_text)
first_text = [ word for word in first_text if not word in set(stopwords.words("english"))]
```

## LEMMATIZATION:

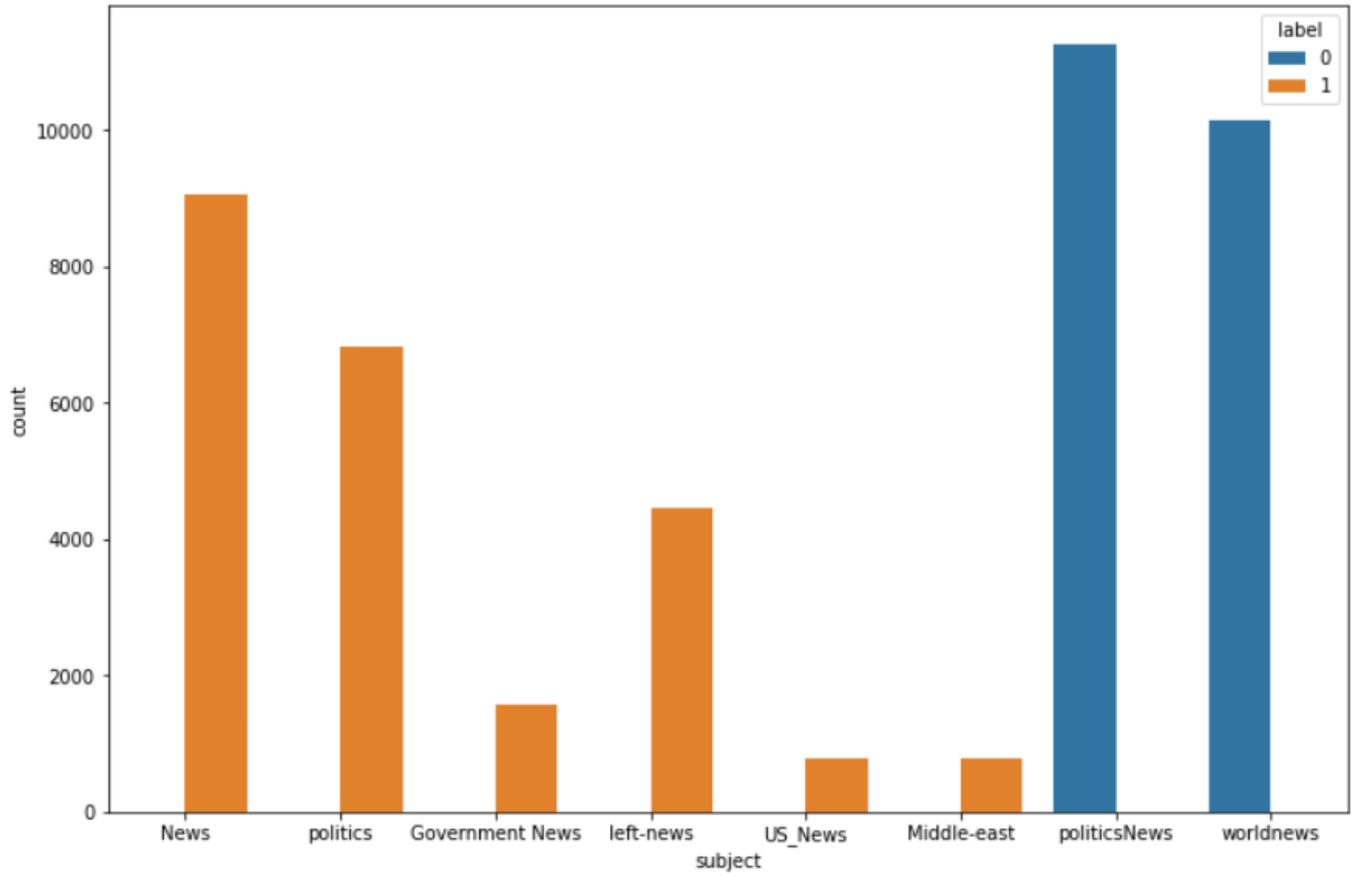
```
lemma = nltk.WordNetLemmatizer()
first_text = [ lemma.lemmatize(word) for word in first_text]

first_text = " ".join(first_text)
first_text
```

# EXPLORATORY ANAYLSIS 1:

CODING:

```
plt.figure(figsize=(12,8))
sns.countplot(x = "subject", data=df, hue = "label")
plt.show()
```

OUTPUT:

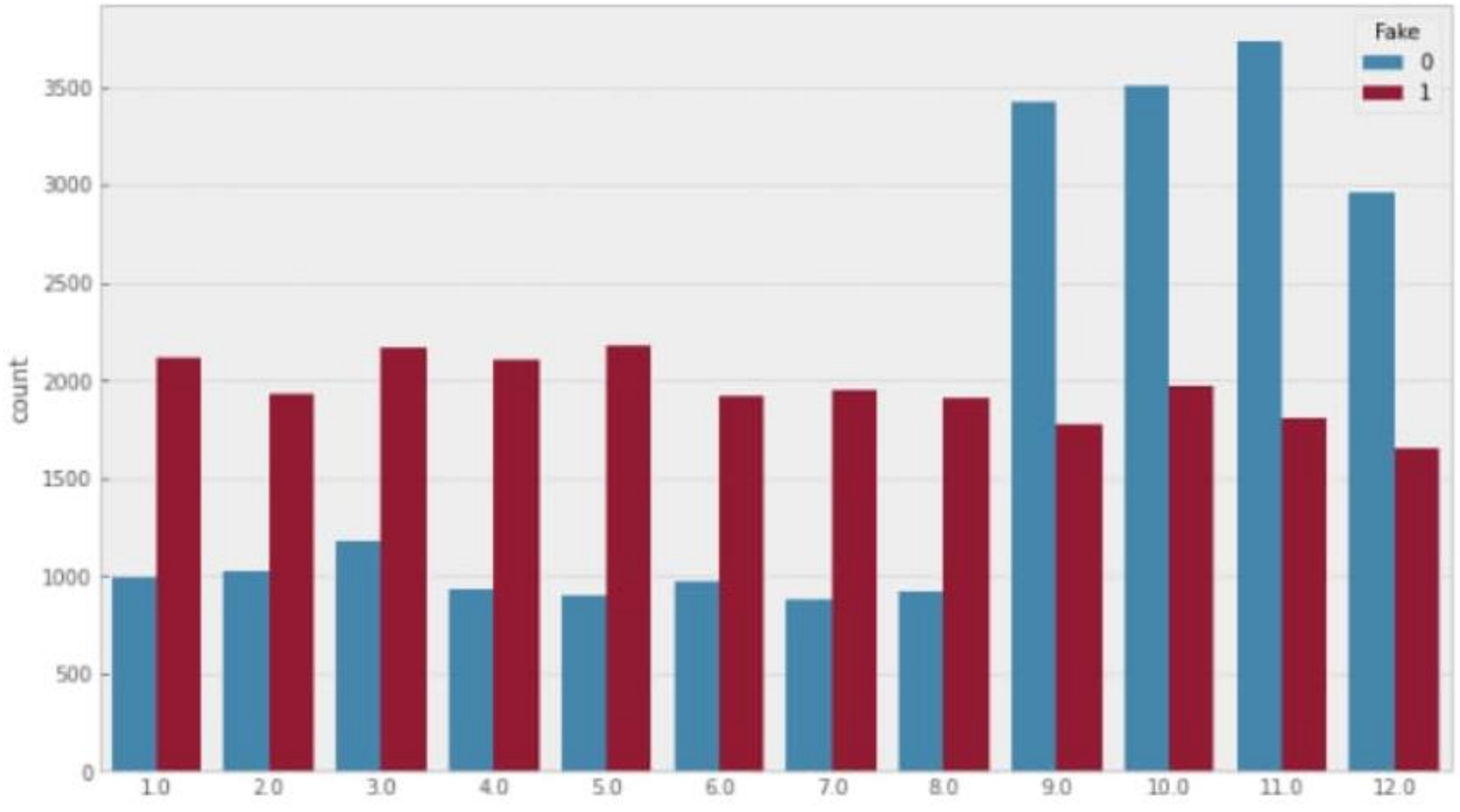


# EXPLORATORY ANAYLSIS 2:

CODING:

```
# Correlation between months and news
plt.style.use('bmh')
plt.figure(figsize=(12, 7))
sns.countplot(data=df, x='Month', hue='Fake')
```

OUTPUT:



# N GRAM ANAYLSIS :

**Unigram Analysis (1-gram**):

- In unigram analysis, each word is considered independently without any regard to its neighboring words.

This is the simplest form of n gram analysis.

**Bigram Analysis (2-gram**):

- Bigram analysis considers pairs of consecutive words.

This type of analysis capture some level of context.

**Trigram Analysis (3-gram):**

- Trigram analysis looks at sequences of three consecutive words.

This provides a bit more context compared to bigrams.

# N GRAM ANAYLSIS :

## This is Big Data AI Book

| | | | | | | |
|---|---|---|---|---|---|---|
| *Uni-Gram* | This | Is | Big | Data | AI | Book |
| *Bi-Gram* | This is | Is Big | Big Data | Data AI | AI Book | |
| *Tri-Gram* | This is Big | Is Big Data | Big Data AI | Data AI Book | | |

# N GRAM ANAYLSIS :

**Unigram Analysis (1-gram)**:

**CODE:**

```
draw_n_gram(string,1)
```

**OUTPUT:**

| | word | count |
|---|---|---|
| 0 | (trump,) | 149603 |
| 1 | (said,) | 133030 |
| 2 | (u,) | 78516 |
| 3 | (state,) | 62726 |
| 4 | (president,) | 58790 |

# ALGORITHM FOR NLP:

- HERE , WE ARE USING LSTM ALGORITHM FOR TRAINING MODEL
- LSTM STANDS FOR LONG SHORT-TERM MEMORY NETWORKS, USED IN THE FIELD OF DEEP LEARNING
- IT IS ONE OF THE BEST ALGORITHM IN NLP TECHNIQUE
- IT IS USED FOR PROCESSING, PREDICTING, AND CLASSIFYING ON THE BASIS OF TIME-SERIES DATA.
- 
- LONG SHORT-TERM MEMORY (LSTM) IS A TYPE OF RECURRENT NEURAL NETWORK (RNN) THAT

IS SPECIFICALLY DESIGNED TO HANDLE SEQUENTIAL DATA, SUCH AS TIME SERIES, SPEECH, AND TEXT.

## TRAIN THE MODEL:

## CODING FOT TRAIN TESTSPLIT:

```
X_train, X_test, y_train, y_test = train_test_split(df['text'], df['target'], random_state=0)
```

## TOKENIZATION:

```
max_features = 10000
maxlen = 300
tokenizer = text.Tokenizer(num_words=max_features)
tokenizer.fit_on_texts(X_train)
tokenized_train = tokenizer.texts_to_sequences(X_train)
X_train = sequence.pad_sequences(tokenized_train, maxlen=maxlen)
tokenized_test = tokenizer.texts_to_sequences(X_test)
X_test = sequence.pad_sequences(tokenized_test, maxlen=maxlen)
```

-

# PROGRAM FOR TRAINING LSTM MODEL:

```
batch_size = 256
epochs = 10
embed_size = 100
model = Sequential()

#Non-trainable embeddidng layer
model.add(Embedding(max_features, output_dim=embed_size, input_length=maxlen, trainable=False))

#LSTM
model.add(LSTM(units=128 , return_sequences = True , recurrent_dropout = 0.25 , dropout = 0.25))
model.add(LSTM(units=64 , recurrent_dropout = 0.1 , dropout = 0.1))
model.add(Dense(units = 32 , activation = 'relu'))
model.add(Dense(1, activation='sigmoid'))
model.compile(optimizer=keras.optimizers.Adam(lr = 0.01), loss='binary_crossentropy', metrics=['accuracy'])
history = model.fit(X_train, y_train, validation_split=0.3, epochs=10, batch_size=batch_size, shuffle=True, verbose = 1)
```

## MODEL AFTER TRAINING WITH LSTM ALGORITHM:

### CODE:

```
model.summary()
```

### OUTPUT:

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding (Embedding)        (None, 300, 100)          1000000
_________________________________________________________________
lstm (LSTM)                  (None, 300, 128)          117248
_________________________________________________________________
lstm_1 (LSTM)                (None, 64)                49408
_________________________________________________________________
dense (Dense)                (None, 32)                2080
_________________________________________________________________
dense_1 (Dense)              (None, 1)                 33
=================================================================
Total params: 1,168,769
Trainable params: 168,769
Non-trainable params: 1,000,000
_________________________________________________________________
```

## ANALYSIS OF MODEL AFTER TRAINING WITH LSTM ALGORITHM:

### CODE:

```
print("Accuracy of the model on Training Data is - " , model.evaluate(X_train,y_train)[1]*100 , "%")
print("Accuracy of the model on Testing Data is - " , model.evaluate(X_test,y_test)[1]*100 , "%")
```

### OUTPUT:

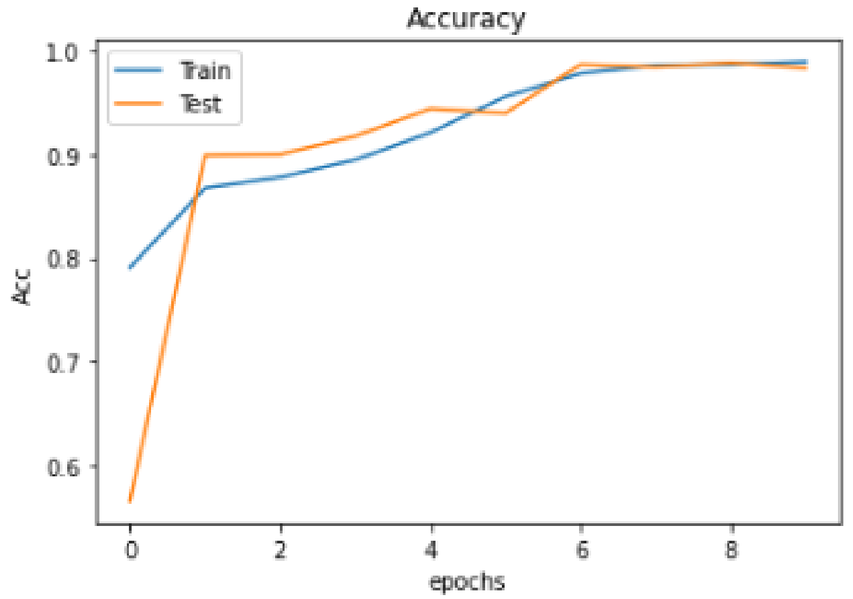
```
1053/1053 [==============================] - 101s 96ms/step - loss: 0.0393 - accuracy: 0.9
843
Accuracy of the model on Training Data is -  98.42603802680969 %
351/351 [==============================] - 34s 97ms/step - loss: 0.0397 - accuracy: 0.9840
Accuracy of the model on Testing Data is -  98.39643836021423 %
```

## MODEL ANALYSIS AFTER TRAINING:

### CODE:

```
plt.figure()
plt.plot(history.history["accuracy"], label = "Train")
plt.plot(history.history["val_accuracy"], label = "Test")
plt.title("Accuracy")
plt.ylabel("Acc")
plt.xlabel("epochs")
plt.legend()
plt.show()
```

### OUTPUT:



## MODEL ANALYSIS AFTER TRAINING:

### CODE:

```
pred = model.predict_classes(X_test)
print(classification_report(y_test, pred, target_names = ['Fake','Real']))
```

### OUTPUT:

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| Fake | 1.00 | 0.97 | 0.98 | 5858 |
| Real | 0.97 | 1.00 | 0.98 | 5367 |
| accuracy | | | 0.98 | 11225 |
| macro avg | 0.98 | 0.98 | 0.98 | 11225 |
| weighted avg | 0.98 | 0.98 | 0.98 | 11225 |

# THANK YOU