IMPORT LIBRARIES

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
from sklearn.metrics import classification_report
import re
import string
```

IMPORT **DATASET**

```
df_fake = pd.read_csv("/content/Fake.csv")
df_true = pd.read_csv("/content/True.csv")
```

```
df_fake.head()
```

```
df_true.head(5)
```

Inserting a column "class" as target feature

```
df_fake["class"] = 0
df_true["class"] = 1
```

```
# Removing last 10 rows for manual testing
df_fake_manual_testing = df_fake.tail(10)
for i in range(23480,23470,-1):
    df_fake.drop([i], axis = 0, inplace = True)
```

```
df_true_manual_testing = df_true.tail(10)
for i in range(21416,21406,-1):
    df_true.drop([i], axis = 0, inplace = True)
```

```
df_fake.shape, df_true.shape
```

```
df_fake_manual_testing["class"] = 0
df_true_manual_testing["class"] = 1
```

```
df_fake_manual_testing.head(10)
```

```
df_true_manual_testing.head(10)
```

```
df_manual_testing = pd.concat([df_fake_manual_testing,df_true_manual_testing], axis = 0)
df_manual_testing.to_csv("manual_testing.csv")
```

Merging True and Fake Dataframes

```
df_merge = pd.concat([df_fake, df_true], axis =0 )
df_merge.head(10)
```

```
df_merge.columns
```

Removing columns which are not required

```
df = df_merge.drop(["title", "subject","date"], axis = 1)
```

```
df.isnull().sum()
```

Random Shuffling the dataframe

```
df = df.sample(frac = 1)
```

```
df.head()
```

```
df.reset_index(inplace = True)
df.drop(["index"], axis = 1, inplace = True)
```

```
df.columns
```

```
df.head()
```

Creating a function to process the texts

```
def wordopt(text):
    text = text.lower()
    text = re.sub('\[.*?\]', '', text)
    text = re.sub("\\W"," ",text)
    text = re.sub('https?://\S+|www\.\S+', '', text)
    text = re.sub('<.*?>+', '', text)
    text = re.sub('[%s]' % re.escape(string.punctuation), '', text)
    text = re.sub('\n', '', text)
    text = re.sub('\w*\d\w*', '', text)
    return text
```

```
df["text"] = df["text"].apply(wordopt)
```

Defining dependent and independent variables

```
x = df["text"]
y = df["class"]
```

Splitting Training and Testing

```
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.25)
```

Convert text to vectors

```
from sklearn.feature_extraction.text import TfidfVectorizer

vectorization = TfidfVectorizer()
xv_train = vectorization.fit_transform(x_train)
xv_test = vectorization.transform(x_test)
```

Logistic Regression

```
from sklearn.linear_model import LogisticRegression

LR = LogisticRegression()
LR.fit(xv_train,y_train)
```

```
pred_lr=LR.predict(xv_test)
```

```
LR.score(xv_test, y_test)
```

```
print(classification_report(y_test, pred_lr))
```

Decision Tree Classification

```
from sklearn.tree import DecisionTreeClassifier

DT = DecisionTreeClassifier()
DT.fit(xv_train, y_train)
```

```
pred_dt = DT.predict(xv_test)
```

```
DT.score(xv_test, y_test)
```

```
print(classification_report(y_test, pred_dt))
```

Gradient Boosting Classifier

```
from sklearn.ensemble import GradientBoostingClassifier

GBC = GradientBoostingClassifier(random_state=0)
GBC.fit(xv_train, y_train)
```

```
pred_gbc = GBC.predict(xv_test)
```

```
GBC.score(xv_test, y_test)
```

```
print(classification_report(y_test, pred_gbc))
```

Random Forest Classifier

```
from sklearn.ensemble import RandomForestClassifier

RFC = RandomForestClassifier(random_state=0)
RFC.fit(xv_train, y_train)
```

```
pred_rfc = RFC.predict(xv_test)
```

```
RFC.score(xv_test, y_test)
```

```
print(classification_report(y_test, pred_rfc))
```

Model Testing

```
def output_lable(n):
    if n == 0:
        return "Fake News"
    elif n == 1:
        return "Not A Fake News"

def manual_testing(news):
    testing_news = {"text":[news]}
    new_def_test = pd.DataFrame(testing_news)
    new_def_test["text"] = new_def_test["text"].apply(wordopt)
    new_x_test = new_def_test["text"]
    new_xv_test = vectorization.transform(new_x_test)
    pred_LR = LR.predict(new_xv_test)
    pred_DT = DT.predict(new_xv_test)
    pred_GBC = GBC.predict(new_xv_test)
    pred_RFC = RFC.predict(new_xv_test)

    return print("\n\nLR Prediction: {} \nDT Prediction: {} \nGBC Prediction: {} \nRFC Prediction: {}".format(output_lable(pred_LR[0]),
                                                                                                              output_lable(pred_GBC[0]),
                                                                                                              output_lable(pred_RFC[0])))
```

```
news = str(input())
manual_testing(news)


news = str(input())
manual_testing(news)
```