```
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
```

```
data_fake = pd.read_csv('Fake.csv')
data_true = pd.read_csv('True.csv')
data_fake.head()
data_true.head()
```

	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
[illegible]	FBI Russia probe helped	WASHINGTON (Reuters) - Trump	[illegible]	December

```
data_fake["class"] = 0
data_true["class"] = 1
```

```
data_fake.shape, data_true.shape
```

```
((23481, 5), (21417, 5))
```

```
data_fake_manual_testing = data_fake.tail(10)
for i in range(23480,23470,-1):
    data_fake.drop([i], axis = 0, inplace = True)

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

```
data_fake_manual_testing['class'] = 0
data_true_manual_testing['class'] = 1
```

```
<ipython-input-216-90008d39c97b>:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus
  data_fake_manual_testing['class'] = 0
<ipython-input-216-90008d39c97b>:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus
  data_true_manual_testing['class'] = 1
```

```
data_merge = pd.concat([data_fake, data_true], axis = 0)
data_merge.head(10)
```

```
                                 title                    text  subject         date  class
```

```
data = data_merge.drop(['title','subject','date'], axis = 1)
```

```
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
```

```
data['text']= data['text'].apply(wordopt)
x = data['text']
y = data['class']
```

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

```
from sklearn.feature_extraction.text import TfidfVectorizer
vectorization = TfidfVectorizer()
xv_train = vectorization.fit_transform(x_train)
xv_test = vectorization.transform(x_test)
```

```
from sklearn.linear_model import LogisticRegression
LR = LogisticRegression()
LR.fit(xv_train, y_train)
```

```
▾ LogisticRegression
LogisticRegression()
```

```
pred_lr = LR. predict(xv_test)
LR.score(xv_test,y_test)
print(classification_report(y_test, pred_lr))
```

```
              precision    recall  f1-score   support

           0       0.99      0.98      0.99      5938
           1       0.98      0.99      0.98      5282

    accuracy                           0.99     11220
   macro avg       0.99      0.99      0.99     11220
weighted avg       0.99      0.99      0.99     11220
```

```
from sklearn.tree import DecisionTreeClassifier
DT = DecisionTreeClassifier()
DT.fit(xv_train, y_train)
```

```
▾ DecisionTreeClassifier
DecisionTreeClassifier()
```

```
pred_dt = DT.predict(xv_test)
DT.score(xv_test,y_test)
print(classification_report(y_test,pred_dt))
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00      5938
           1       1.00      1.00      1.00      5282

    accuracy                           1.00     11220
   macro avg       1.00      1.00      1.00     11220
weighted avg       1.00      1.00      1.00     11220
```

```
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_GB = GB.predict(new_xv_test)
    pred_RF = RF.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)
```

Colab paid products - Cancel contracts here

Executing (6m 30s) <cell line: 1> › raw_input() › _input_request() › select()