FAKE NEWS PROJECT

SUBMITTED BY

KAYALVIZHI . P

ACKNOWLEDGMENT

I would like to express my sincere thanks and gratitude to my SME as well as “Flip Robo Technologies” team for letting me work on “Fake News Detection” project which also helped me in doing lots of research wherein I came to know about so many new things. Their suggestions and directions have helped me in the completion of this project successfully. This project also helped me in doing lots of research wherein I came to know about so many new things.

References:

I have also used few external resources that helped me to complete this project successfully. Below are the external resources that were used to create this project.

1. https://www.google.com/
2. https://www.youtube.com/
3. https://github.com/
4. https://www.kaggle.com/
5. https://medium.com/
6. https://towardsdatascience.com/
7. https://www.javatpoint.com/nlp
8. https://www.educative.io/answers/preprocessing-steps-in-natural-language-processing-nlp
9. https://www.youtube.com/watch?v=5ctbvkAMQO4
10. https://www.youtube.com/watch?v=X2vAabgKiuM

1. INTRODUCTION

Business Problem Framing

There was a time once if anyone required any news, he or she would sit up for the next-day newspaper. With the expansion of on-line newspapers UN agency update news nearly instantly, individuals have found a more robust and quicker thanks to learn of the matter of his/her interest. Today social-networking systems, on-line news portals, and alternative on-line media became the most sources of reports through that fascinating and breaking news are shared at a fast pace news are shared at a fast pace.

Conceptual Background of the Domain Problem

The term "Fake News" was a lot less unheard of and not prevalent a couple of decades ago but in this digital era of social media, it has surfaced as a huge monster. Fake news, information bubbles, news manipulation and the lack of trust in the media are growing problems within our society. . However, in order to start addressing this problem, an indepth understanding of fake news and its origins is required. Only then one can look into the different techniques and fields of machine learning (ML), natural language processing (NLP) and artificial intelligence (AI) that could help us fight this situation."Fake news" has been used in a multitude of ways in the last half a year and multiple definitions have been given. For instance, the New York Times defines it as "a made-up story with an intention to deceive".Measuring fake news or even defining it properly could very quickly become a subjective matter, rather than an objective metric.In its purest form, fake news is completely made up, manipulated to resemble credible journalism and attract maximum attention and, with it, advertising revenue. Despite all these shortcomings, several entities have tried to categorize fake news in different manners.

Review of Literature

Literature review covers relevant literature with the aim of gaining insight into the factors that are important to predict whether news is fake or not. In this study, we discuss various applications and methods which inspired us to build our supervised ML techniques.

This project is more about data exploration, feature engineering and preprocessing that can be done on this data. We can do better data exploration and derive some interesting features using the available columns. Different techniques like ensemble techniques, k-nearest neighbors, and decision trees have been used to make the predictions. The goal of this project is to build an application which can predict fake news with the help of other features.

Motivation for the Problem Undertaken

So, the main aim is to use machine learning algorithms to develop models for predicting used car prices.

- ✓ To build a supervised machine learning model for forecasting value of a used vehicle based on multiple attributes.
- ✓ The system that is being built must be feature based i.e., feature wise prediction must be possible.

2. Analytical Problem Framing

Mathematical/ Analytical Modeling of the Problem

We need to develop an efficient and effective Machine Learning model which predicts the price of a used cars. So, “Car_Price” is our target variable which is continuous in nature. Clearly it is a Regression problem where we need to use regression algorithms to predict the results.

Model Building Phase: After collecting the data, I built a machine learning model. Before model building, have done all data pre-processing steps. The complete life cycle of data science that I have used in this project are as follows:

- Data Cleaning
- Exploratory Data Analysis
- Data Pre-processing
- Model Building
- Model Evaluation
- Selecting the best model

Data Sources and their formats

Data set provided by Flip Robo was in the format of CSV (Comma Separated Values). The dimension of data is 44898 rows and 4 columns. There are 2 data sets that are given. One is Fake data and one is True data. After that we are adding Label to both datasets so 44898 rows and 5 columns.

1. Fake.csv file will have the fake news. It does not contains the target variable. We only adding the Label variable. Size of Fake.csv: 23481 records.

2. True.csv file will have the true news. It does not contains the target variable. We only adding the Label variable. Size of True.csv: 21417 records.

Data Preprocessing Done

Data pre-processing in Machine Learning refers to the technique of preparing (cleaning and organizing) the raw data to make it suitable for a building and training Machine Learning models. In other words, whenever the data is gathered from different sources it is collected in raw format which is not feasible for the analysis. Data pre-processing is an integral step in Machine Learning as the quality of data and the useful information that can be derived from it directly affects the ability of our model to learn; therefore, it is extremely important that we pre-process our data before feeding it into our model. In this project we use Natural Language Processing for treating the datas. Therefore, it is the first and crucial step while creating a machine learning model. I have used some following pre-processing steps:

- Importing necessary libraries and importing two dataset fake and true as a data frame.
- Then we are adding label to both the datasets.
- Label 0 for fake,csv and Label 1 for True.csv.
- We have two datasets so we are combining both datasets using concat funtion.
- Checked some statistical information like shape, info, data types etc.
- Then we are dropping unwanted columns from our dataset.
- Checking for Null values. We don’t have any null values in our dataset. We will confirm it with heatmap also.
- Checking some basic informations like columns, info.
- After that we are done with NLP steps.
- Steps for treating datas processing is Stemming, Stop words removal, Lemmatization.
- After these processes we are converting text format data as numerical data.
- Then we are checking Skewness and correlation for our label variable only, because we are having only two columns in our dataset.

Hardware & Software Requirements & Tools Used

To build the machine learning projects it is important to have the following hardware and software requirements and tools.

Hardware required:

- Processor: core i3
- RAM: 8 GB
- ROM/SSD: 250 GB or above

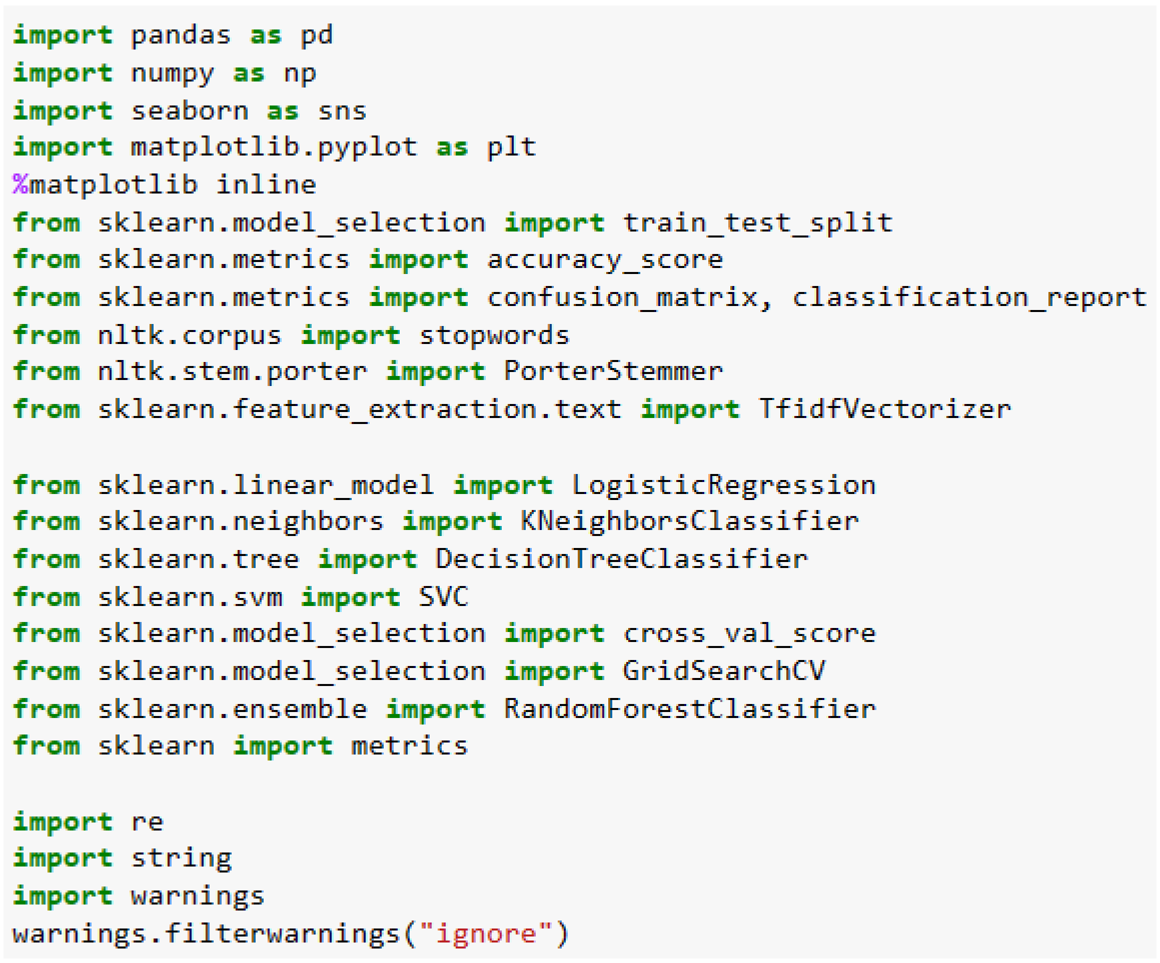
Software required:

- Distribution: Anaconda Navigator
- Programming language: Python
- Browser based language shell: Jupyter Notebook

Libraries/Packages Used:

Pandas, NumPy, matplotlib, seaborn, scikit-learn, NLP, Re.

Libraries required:



```python
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
from sklearn.metrics import confusion_matrix, classification_report
from nltk.corpus import stopwords
from nltk.stem.porter import PorterStemmer
from sklearn.feature_extraction.text import TfidfVectorizer

from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.svm import SVC
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import GridSearchCV
from sklearn.ensemble import RandomForestClassifier
from sklearn import metrics

import re
import string
import warnings
warnings.filterwarnings("ignore")
```

- **import numpy as np:** It is defined as a Python package used for performing the various numerical computations and processing of the multidimensional and single dimensional array elements. The calculations using Numpy arrays are faster than the normal Python array.
- **import pandas as pd:** Pandas is a Python library that is used for faster data analysis, data cleaning and data pre-processing. The data-frame term comes from Pandas itself.
- **import matplotlib.pyplot as plt:** Matplotlib and Seaborn acts as the backbone of data visualization through Python.
- **Matplotlib:** It is a Python library used for plotting graphs with the help of other libraries like Numpy and Pandas. It is a powerful tool for visualizing data in Python. It is used for creating statical interferences and plotting 2D graphs of arrays.

- **import seaborn as sns:** Seaborn is also a Python library used for plotting graphs with the help of Matplotlib, Pandas, and Numpy. It is built on the roof of Matplotlib and is considered as a superset of the Matplotlib library. It helps in visualizing univariate and bivariate data.
- **import TfidfVectorizer:** Transforms text to feature vectors that can be used as input to estimator. vocabulary_ Is a dictionary that converts each token (word) to feature index in the matrix, each unique token gets a feature index.
- **import PorterStemmer:** process for removing the commoner morphological and inflexional endings from words in English.
- **import stopwords:** Stop words are commonly used in Text Mining and Natural Language Processing (NLP) to eliminate words that are so commonly used that they carry very little useful information.

3.MODEL/S DEVELOPMENT AND EVALUATION

Identification of possible Problem-solving approaches (Methods):

In this project, we are using some machine learning and Natural language processing libraries like NLTK, re (Regular Expression), Scikit Learn. Machine learning data only works with numerical features so we have to convert text data into numerical columns. So we have to preprocess the text and that is called natural language processing.

In-text preprocess we are cleaning our text by steaming, lemmatization, remove stopwords, remove special symbols and numbers, etc. After cleaning the data we have to feed this text data into a vectorizer which will convert this text data into numerical features.

Stemming: Stemming is the process of producing morphological variants of a root/base word. Stemming programs are commonly referred to as stemming algorithms or stemmers. A stemming algorithm reduces the words “chocolates”, “chocolatey”, and “choco” to the root word, “chocolate” and “retrieval”, “retrieved”, “retrieves” reduce to the stem “retrieve”.

Stopwords Removal: The process of converting data to something a computer can understand is referred to as **pre-processing.** One of the major forms of pre-processing is to filter out useless data. In natural language processing, useless words (data), are referred to as stop words.

What are Stop words?

Stop Words: A stop word is a commonly used word (such as “the”, “a”, “an”, “in”) that a search engine has been programmed to ignore, both when indexing entries for searching and when retrieving them as the result of a search query.

Lemmatization: Lemmatization is the process of grouping together the different inflected forms of a word so they can be analyzed as a single item. Lemmatization is similar to stemming but it brings context to the words. So it links words with similar meanings to one word. Text preprocessing includes both Stemming as well as Lemmatization. Many times people find these two terms confusing. Some treat these two as the same. Actually, lemmatization is preferred over Stemming because lemmatization does morphological analysis of the words.

Vectorizer: It is a great tool provided by the scikit-learn library in Python. It is used to transform a given text into a vector on the basis of the frequency (count) of each word that occurs in the entire text. This is helpful when we have multiple such texts, and we wish to convert each word in each text into vectors (for using in further text analysis).

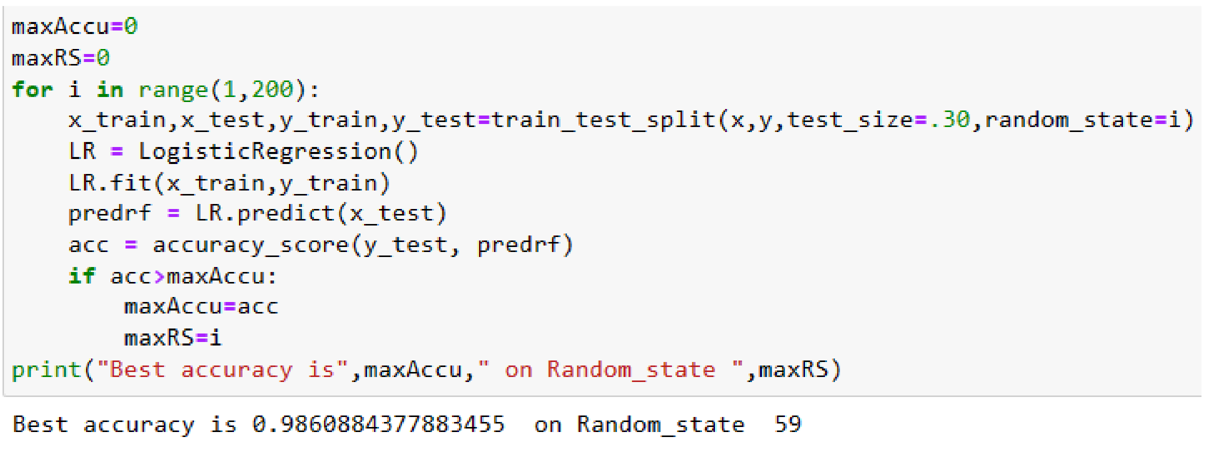
Testing of Identified Approaches (Algorithms)

The algorithms used on training the data are as follows:

1. Logistic Regression
2. Decision Tree Classifier
3. KNNeighbors Classifier
4. Random Forest Classifier
5. Ada Boost Classifier

Run and evaluate selected models

I have used 5 Classification algorithms after choosing random state amongst 1-200 number. I have used Logistic Regression to find best random state and the code is as below:



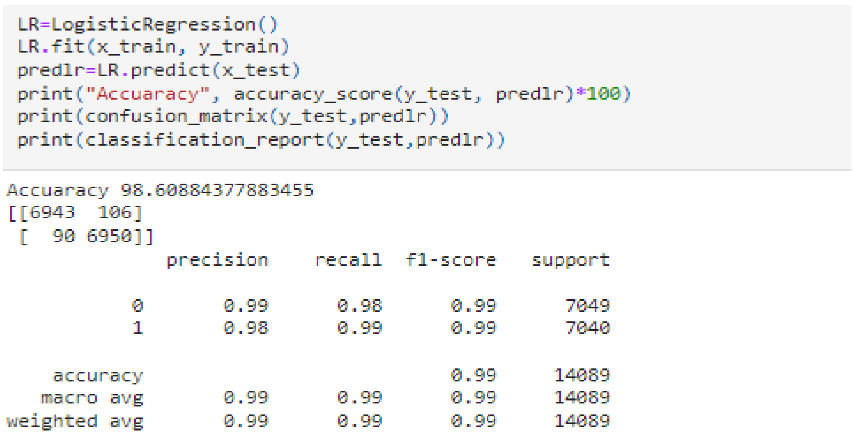
```
maxAccu=0
maxRS=0
for i in range(1,200):
    x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=.30,random_state=i)
    LR = LogisticRegression()
    LR.fit(x_train,y_train)
    predrf = LR.predict(x_test)
    acc = accuracy_score(y_test, predrf)
    if acc>maxAccu:
        maxAccu=acc
        maxRS=i
print("Best accuracy is",maxAccu," on Random_state ",maxRS)
```

```
Best accuracy is 0.9860884377883455  on Random_state  59
```

Model Building:

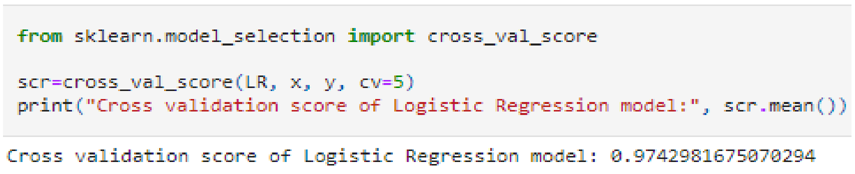
1. Logistic Regression:

Logistic regression aims to solve classification problems. It does this by predicting categorical outcomes, unlike linear regression that predicts a continuous outcome.



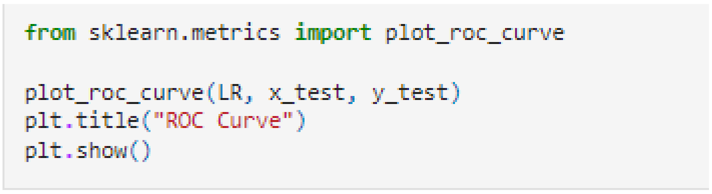
```
LR=LogisticRegression()
LR.fit(x_train, y_train)
predlr=LR.predict(x_test)
print("Accuaracy", accuracy_score(y_test, predlr)*100)
print(confusion_matrix(y_test,predlr))
print(classification_report(y_test,predlr))
```

```
Accuaracy 98.60884377883455
[[6943  106]
 [  90 6950]]
              precision    recall  f1-score   support

           0       0.99      0.98      0.99      7049
           1       0.98      0.99      0.99      7040

    accuracy                           0.99     14089
   macro avg       0.99      0.99      0.99     14089
weighted avg       0.99      0.99      0.99     14089
```

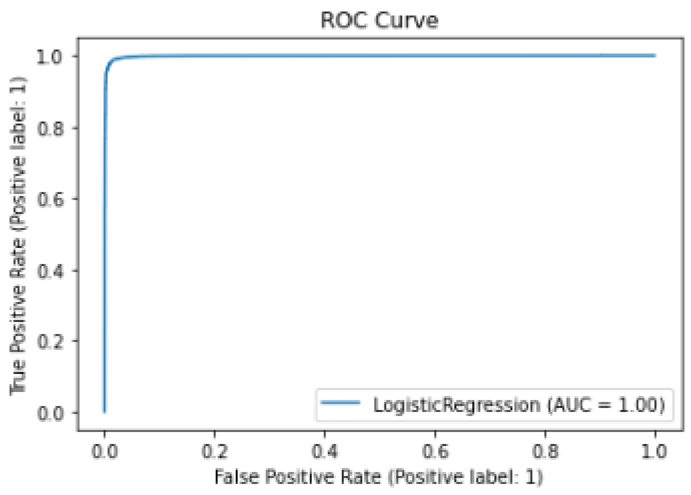


```
from sklearn.model_selection import cross_val_score

scr=cross_val_score(LR, x, y, cv=5)
print("Cross validation score of Logistic Regression model:", scr.mean())
```

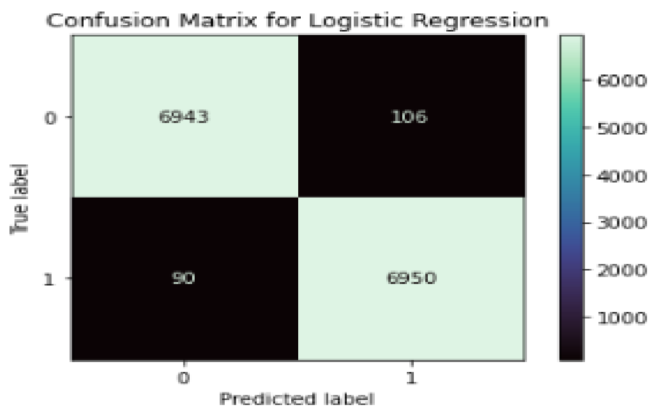
```
Cross validation score of Logistic Regression model: 0.9742981675070294
```



```
from sklearn.metrics import plot_roc_curve

plot_roc_curve(LR, x_test, y_test)
plt.title("ROC Curve")
plt.show()
```



```
metrics.plot_confusion_matrix(LR, x_test, y_test, cmap='mako')
plt.title('Confusion Matrix for Logistic Regression')
plt.show()
```



Created Logistic Regression model and checked for it's evaluation metrics. The model is giving accuracy_score as 98.6% and Cross validation Score as 97%.

2. Decision Tree Classifier

Decision Tree is a white box type of ML algorithm. It shares internal decision-making logic, which is not available in the black box type of algorithms such as Neural Network. Its training time is faster compared to the neural network algorithm.

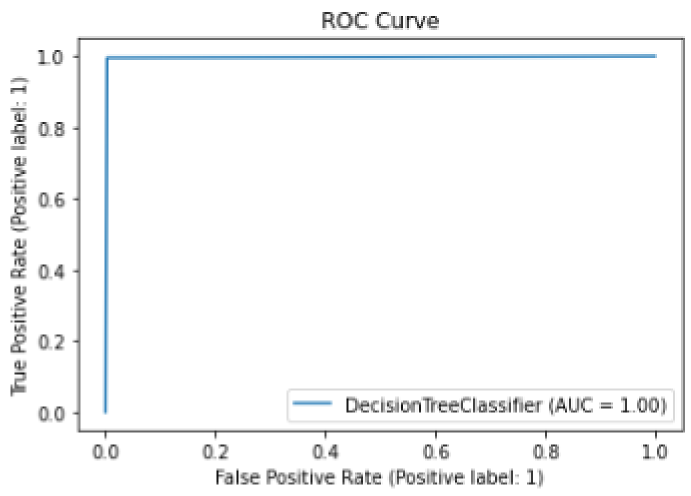
```
dtr=DecisionTreeClassifier()
dtr.fit(x_train, y_train)
preddtr=dtr.predict(x_test)
print("Accuaracy", accuracy_score(y_test, preddtr)*100)
print(confusion_matrix(y_test,preddtr))
print(classification_report(y_test,preddtr))
```

```
Accuaracy 99.61672226559728
[[7030   19]
 [  35 7005]]
              precision    recall  f1-score   support

           0       1.00      1.00      1.00      7049
           1       1.00      1.00      1.00      7040

    accuracy                           1.00     14089
   macro avg       1.00      1.00      1.00     14089
weighted avg       1.00      1.00      1.00     14089
```

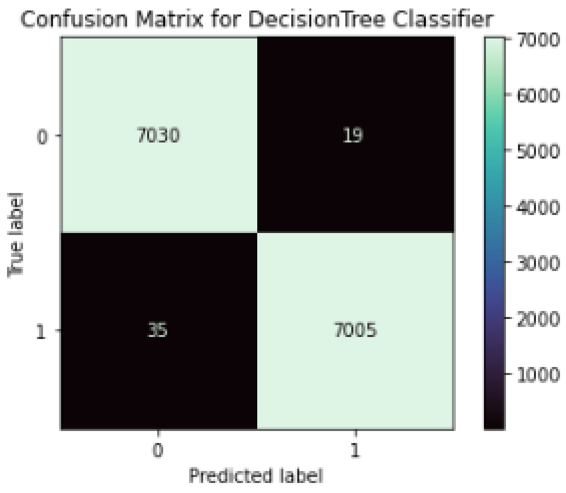
```
scr=cross_val_score(dtr, x, y, cv=5)
print("Cross validation score of Decision Tree Classifier model:", scr.mean())
```

```
Cross validation score of Decision Tree Classifier model: 0.9926110385946547
```

```
plot_roc_curve(dtr, x_test, y_test)
plt.title("ROC Curve")
plt.show()
```



```
metrics.plot_confusion_matrix(dtr, x_test, y_test, cmap='mako')
plt.title('Confusion Matrix for DecisionTree Classifier')
plt.show()
```



Created Decision Tree Classification model and checked for it's evaluation metrics. The model is giving accuracy_score as 99.6% and Cross validation Score as 99%.

3. KNNeighbors Classifier

K Nearest Neighbor(KNN) is a very simple, easy to understand, versatile and one of the topmost machine learning algorithms. KNN used in the variety of applications such as finance, healthcare, political science, handwriting detection, image recognition and video recognition.

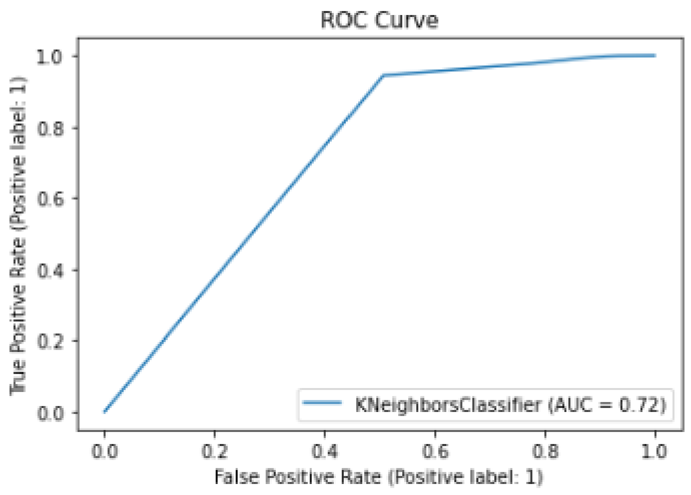
```
knc=KNeighborsClassifier()
knc.fit(x_train, y_train)
predknc=knc.predict(x_test)
print("Accuaracy", accuracy_score(y_test, predknc)*100)
print(confusion_matrix(y_test,predknc))
print(classification_report(y_test,predknc))
```

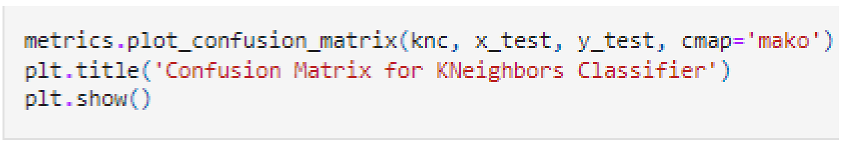
```
Accuaracy 56.12179714671019
[[ 916 6133]
 [  49 6991]]
              precision    recall  f1-score   support

           0       0.95      0.13      0.23      7049
           1       0.53      0.99      0.69      7040

    accuracy                           0.56     14089
   macro avg       0.74      0.56      0.46     14089
weighted avg       0.74      0.56      0.46     14089
```

```
scr=cross_val_score(knc, x, y, cv=5)
print("Cross validation score of KNeighbors Classifier model:", scr.mean())
```

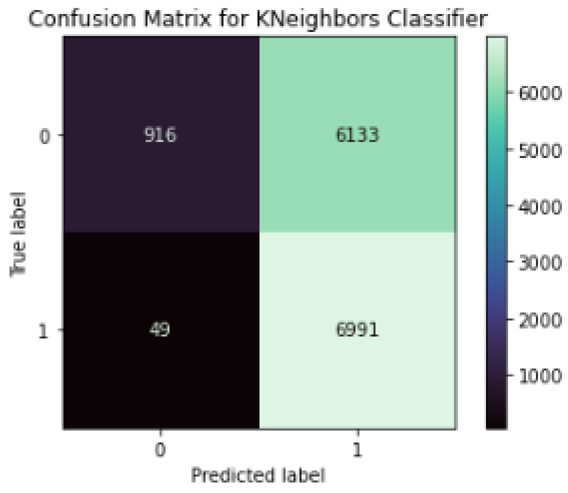
```
Cross validation score of KNeighbors Classifier model: 0.5711881138242967
```

```
plot_roc_curve(knc, x_test, y_test)
plt.title("ROC Curve")
plt.show()
```





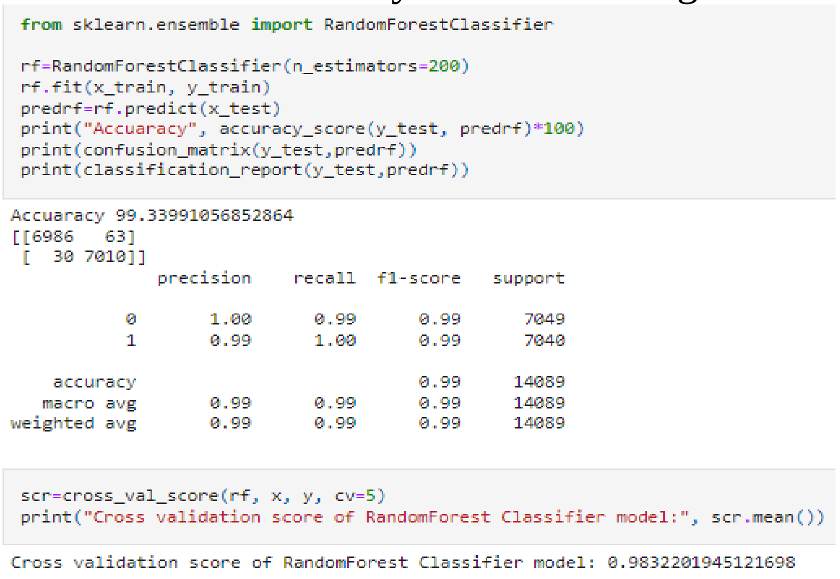
```
metrics.plot_confusion_matrix(knc, x_test, y_test, cmap='mako')
plt.title('Confusion Matrix for KNeighbors Classifier')
plt.show()
```



Created KNNeighbors Classification model and checked for it's evaluation metrics. The model is giving accuracy_score as 56.2% and Cross validation Score as 57%.

4. Random Forest Classifier

Random forests is a supervised learning algorithm. It can be used both for classification and regression. It is also the most flexible and easy to use algorithm. A forest is comprised of trees. It is said that the more trees it has, the more robust a forest is. Random forests creates decision trees on randomly selected data samples, gets prediction from each tree and selects the best solution by means of voting.



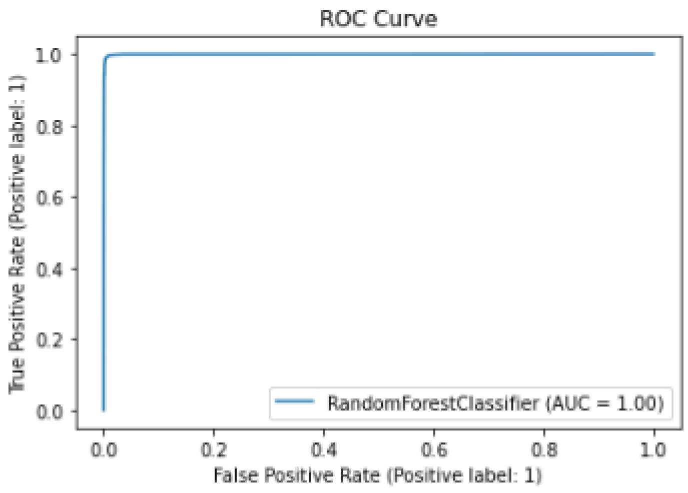
```
from sklearn.ensemble import RandomForestClassifier

rf=RandomForestClassifier(n_estimators=200)
rf.fit(x_train, y_train)
predrf=rf.predict(x_test)
print("Accuaracy", accuracy_score(y_test, predrf)*100)
print(confusion_matrix(y_test,predrf))
print(classification_report(y_test,predrf))
```

```
Accuaracy 99.33991056852864
[[6986   63]
 [  30 7010]]
              precision    recall  f1-score   support

           0       1.00      0.99      0.99      7049
           1       0.99      1.00      0.99      7040

    accuracy                           0.99     14089
   macro avg       0.99      0.99      0.99     14089
weighted avg       0.99      0.99      0.99     14089
```

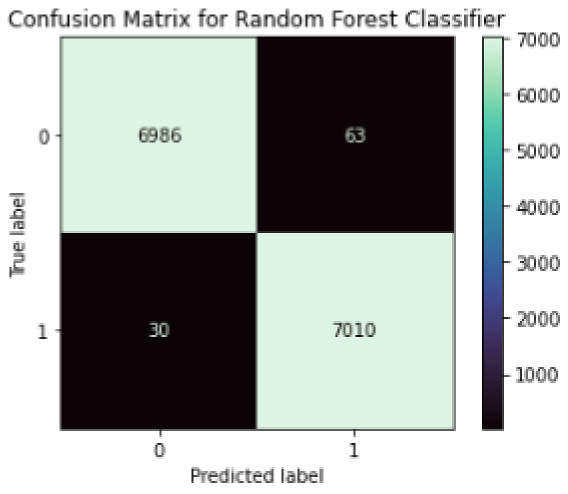
```
scr=cross_val_score(rf, x, y, cv=5)
print("Cross validation score of RandomForest Classifier model:", scr.mean())
```

```
Cross validation score of RandomForest Classifier model: 0.9832201945121698
```

```
plot_roc_curve(rf, x_test, y_test)
plt.title("ROC Curve")
plt.show()
```



```
metrics.plot_confusion_matrix(rf, x_test, y_test, cmap='mako')
plt.title('Confusion Matrix for Random Forest Classifier')
plt.show()
```



Created Random Forest Classification model and checked for it's evaluation metrics. The model is giving accuracy_score as 99.3% and Cross validation Score as 98%.

5. Ada Boost Classifier

An AdaBoost classifier is a meta-estimator that begins by fitting a classifier on the original dataset and then fits additional copies of the classifier on the same dataset but where the weights of incorrectly classified instances are adjusted such that subsequent classifiers focus more on difficult cases.

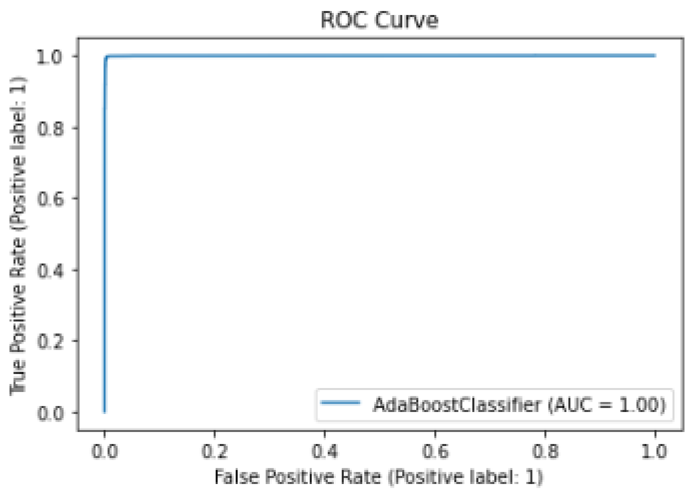
```
from sklearn.ensemble import AdaBoostClassifier

ad=AdaBoostClassifier()
ad.fit(x_train, y_train)
predad=ad.predict(x_test)
print("Accuaracy", accuracy_score(y_test, predad)*100)  # testing accuracy
print(confusion_matrix(y_test,predad))
print(classification_report(y_test,predad))
```

```
Accuaracy 99.62382000141955
[[7020   29]
 [  24 7016]]
              precision    recall  f1-score   support

           0       1.00      1.00      1.00      7049
           1       1.00      1.00      1.00      7040

    accuracy                           1.00     14089
   macro avg       1.00      1.00      1.00     14089
weighted avg       1.00      1.00      1.00     14089
```

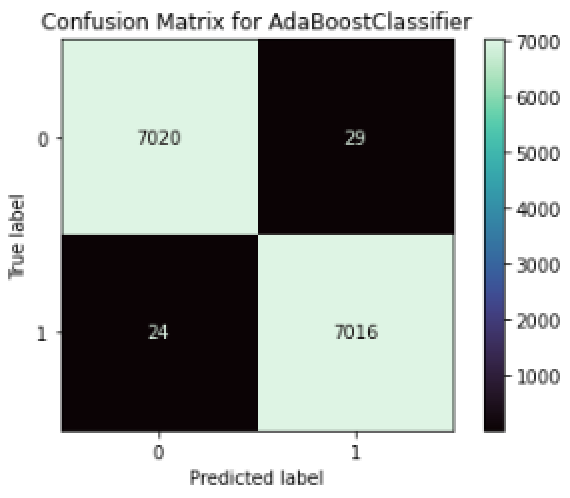
```
scr=cross_val_score(ad, x, y, cv=5)
print("Cross validation score of AdaBoostClassifier model:", scr.mean())
```

```
Cross validation score of AdaBoostClassifier model: 0.9936756181113523
```

```
plot_roc_curve(ad, x_test, y_test)
plt.title("ROC Curve")
plt.show()
```



```
metrics.plot_confusion_matrix(ad, x_test, y_test, cmap='mako')
plt.title('Confusion Matrix for AdaBoostClassifier')
plt.show()
```



Created Ada Boost Classification model and checked for it's evaluation metrics. The model is giving accuracy_score as 99.6% and Cross validation Score as 99%.

Hyper Parameter Tuning:

I am choosing Ada Boost Classifier as my best model.

```
from sklearn.model_selection import GridSearchCV

parameters={'algorithm':['SAMME','SAMME.R'],
           'learning_rate':[0.0001, 0.001, 0.01, 0.1, 1.0],
           'n_estimators':[10,50,100,500]}
```

After comparing all the classification models I have selected AdaBoost Classifier as my best model and have listed down it's parameters above referring the sklearn webpage.

```
GCV=GridSearchCV(estimator=AdaBoostClassifier(),
                param_grid=parameters,
                n_jobs=-1,
                cv=5,
                scoring='accuracy')
```

```
GCV.fit(x_train,y_train)
```