ONLINE FOOD DELIVERY PLATFORMS FEEDBACK APP

A machine learning approach for feedback on online food delivery platforms like Zomato and Swiggy.

STEPS FOR ML MODEL DEVELOPMENT

STEP-1:

Understanding the business problem:

Nowadays many people are ordering food through online platforms like Zomato and Swiggy and other food delivery companies. so, there is high demand for online food orders.so these companies are trying to improve their business by deliver the food to the right persons at right time.

So, I am implementing an online food delivery feedback app in order to help the companies to increase their business

These online food delivery feedback app which helps the companies to deliver the food faster to the customers and this app which is used to take the feedback from the customers and it helps to the companies to identify the majority of the areas who orders food and different types of the customers who order food more and helps to food delivery companies to increase their business profits.

STEP-2:

Collecting or gathering the data.

So, the food delivery companies have the customers data. I have taken the dataset from the Kaggle website which consists of the data about the customers.

It consists of following details of the customers.

1.Age: The age of the customer

2.Marital status: the marriage status of the customer

3.Occupation: work of the customer

4.monthly income: the monthly income of a customer

5. education qualification: education background of the customers

6.family size: the total family members of the customer

7.latitide and longitude: the location of the customer

8.pin code: the pin code of residence of the customer

9.Output: is customer order again

10.Feedback: feedback of the last order

STEP-3

EXPLORATORY DATA ANALYSIS

1) Load the necessary packages to perform EDA

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
```

2) Load the dataset

	Age	Gender	Marital Status	Occupation	Monthly Income	Educational Qualifications	Family size	latitude	longitude	Pin code	Output	Feedback
0	20	Female	Single	Student	No Income	Post Graduate	4	12.9766	77.5993	560001	Yes	Positive
1	24	Female	Single	Student	Below Rs.10000	Graduate	3	12.9770	77.5773	560009	Yes	Positive
2	22	Male	Single	Student	Below Rs.10000	Post Graduate	3	12.9551	77.6593	560017	Yes	Negative
3	22	Female	Single	Student	No Income	Graduate	6	12.9473	77.5616	560019	Yes	Positive
4	22	Male	Single	Student	Below Rs.10000	Post Graduate	4	12.9850	77.5533	560010	Yes	Positive
...	...	...	...	...	...	...	...	...	...	...	...	...
383	23	Female	Single	Student	No Income	Post Graduate	2	12.9766	77.5993	560001	Yes	Positive
384	23	Female	Single	Student	No Income	Post Graduate	4	12.9854	77.7081	560048	Yes	Positive
385	22	Female	Single	Student	No Income	Post Graduate	5	12.9850	77.5533	560010	Yes	Positive
386	23	Male	Single	Student	Below Rs.10000	Post Graduate	2	12.9770	77.5773	560009	Yes	Positive
387	23	Male	Single	Student	No Income	Post Graduate	5	12.8988	77.5764	560078	Yes	Positive

388 rows × 13 columns

\

3) Understanding about the dataset

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 388 entries, 0 to 387
Data columns (total 12 columns):
 #   Column                      Non-Null Count  Dtype  
---  ------                      --------------  -----  
 0   Age                         388 non-null    int64  
 1   Gender                      388 non-null    object 
 2   Marital Status              388 non-null    object 
 3   Occupation                  388 non-null    object 
 4   Monthly Income              388 non-null    object 
 5   Educational Qualifications  388 non-null    object 
 6   Family size                 388 non-null    int64  
 7   latitude                    388 non-null    float64
 8   longitude                   388 non-null    float64
 9   Pin code                    388 non-null    int64  
 10  Output                      388 non-null    object 
 11  Feedback                    388 non-null    object 
dtypes: float64(2), int64(3), object(7)
memory usage: 36.5+ KB
```

4) Checking the null values

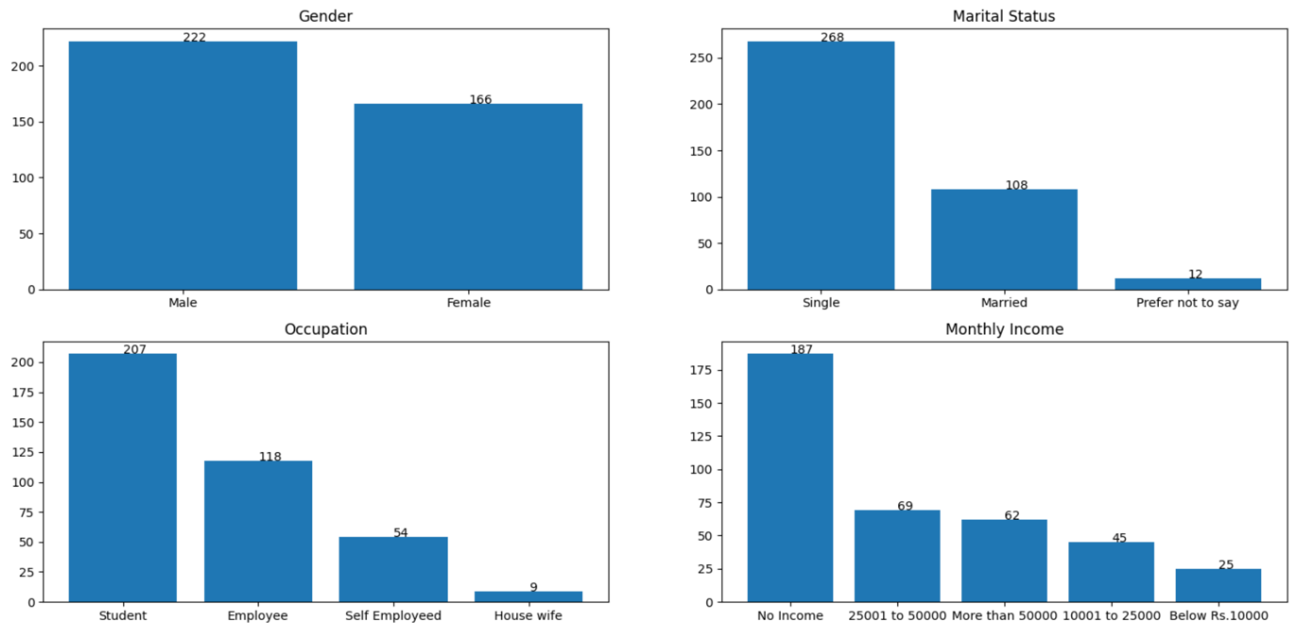
```
Age                           0
Gender                        0
Marital Status                0
Occupation                    0
Monthly Income                0
Educational Qualifications    0
Family size                   0
latitude                      0
longitude                     0
Pin code                      0
Output                        0
Feedback                      0
```

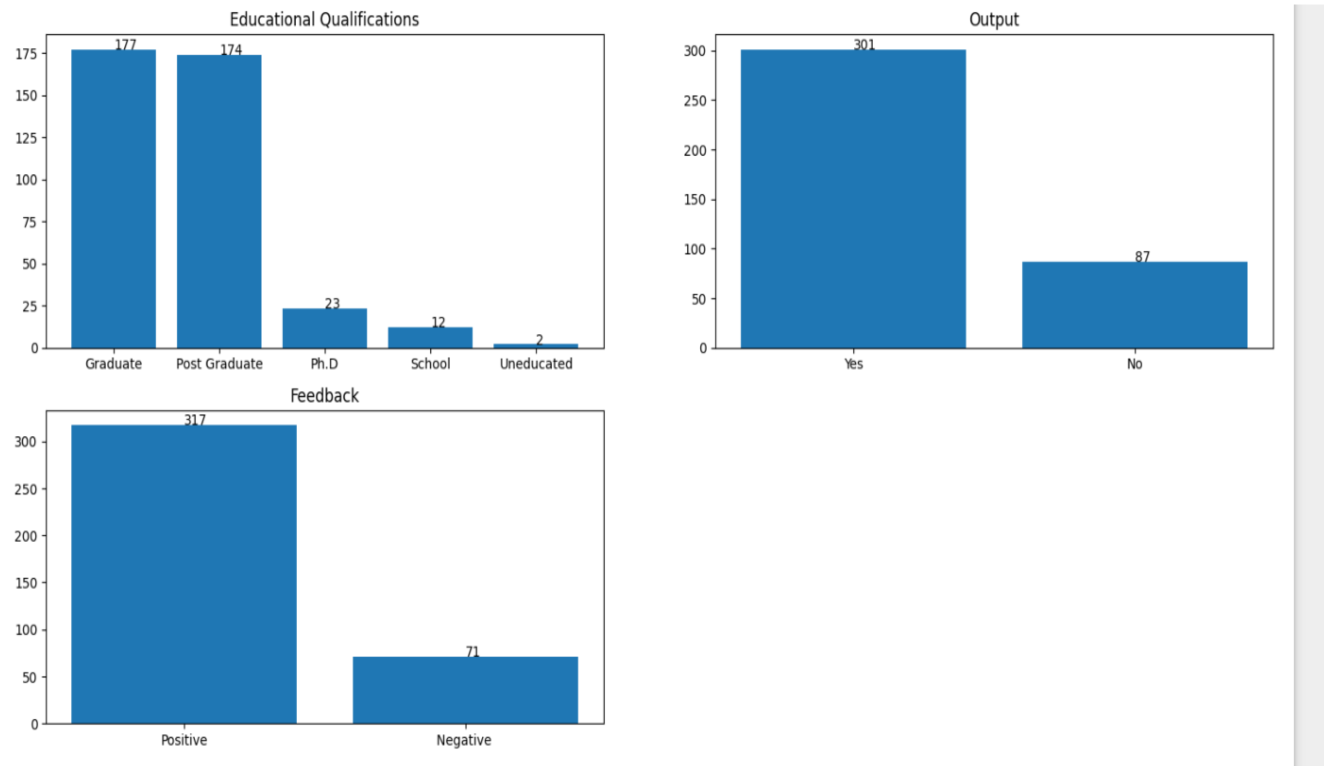
5) Separation of categorical and numerical columns

	categorical_columns
0	Gender
1	Marital Status
2	Occupation
3	Monthly Income
4	Educational Qualifications
5	Output
6	Feedback

	numerical_columns
0	Age
1	Family size
2	latitude
3	longitude
4	Pin code

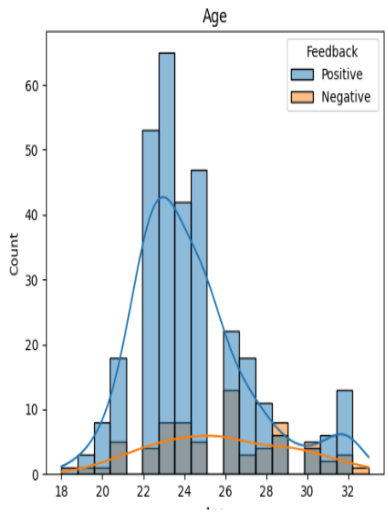
6) Working on the categorical columns

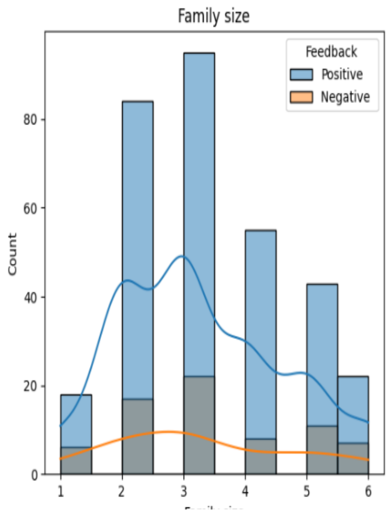


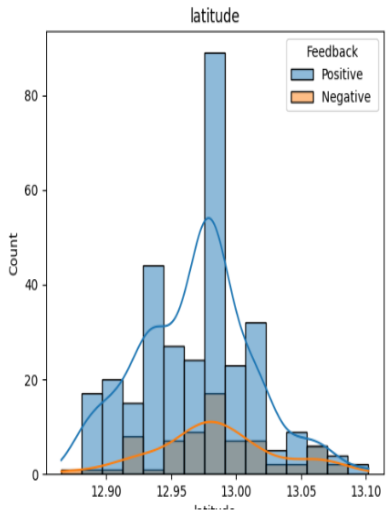


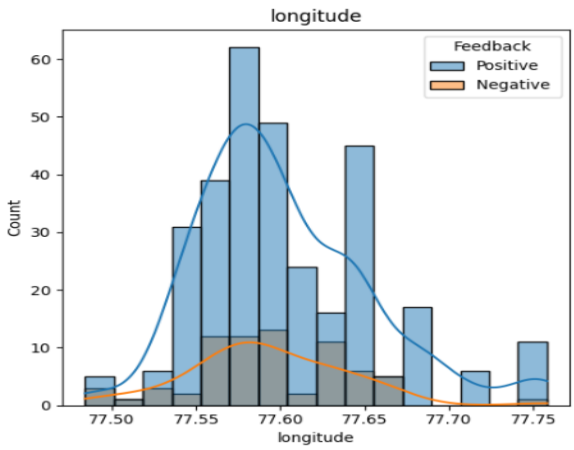
7) Working on the numerical columns

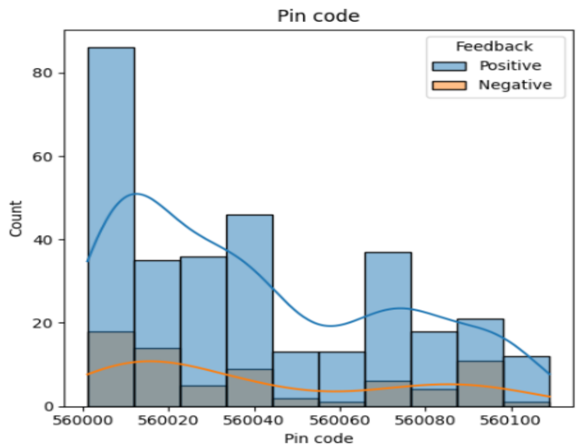
	Age	Family size	latitude	longitude	Pin code
count	388.000000	388.000000	388.000000	388.000000	388.000000
mean	24.628866	3.280928	12.972058	77.600160	560040.113402
std	2.975593	1.351025	0.044489	0.051354	31.399609
min	18.000000	1.000000	12.865200	77.484200	560001.000000
25%	23.000000	2.000000	12.936900	77.565275	560010.750000
50%	24.000000	3.000000	12.977000	77.592100	560033.500000
75%	26.000000	4.000000	12.997025	77.630900	560068.000000
max	33.000000	6.000000	13.102000	77.758200	560109.000000



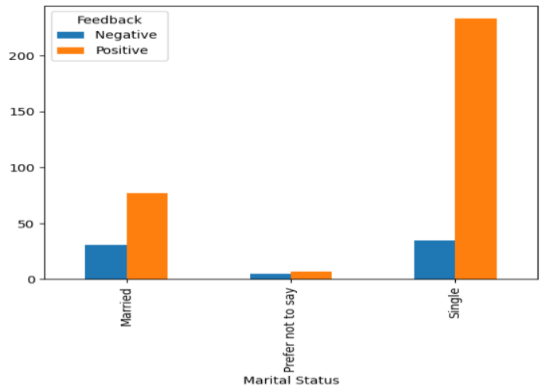


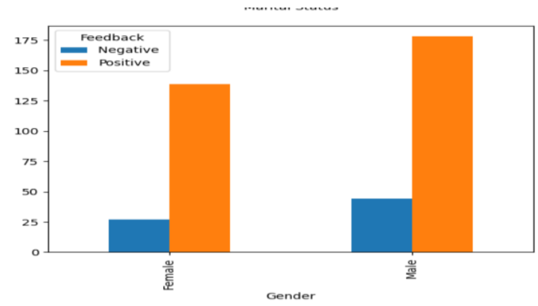


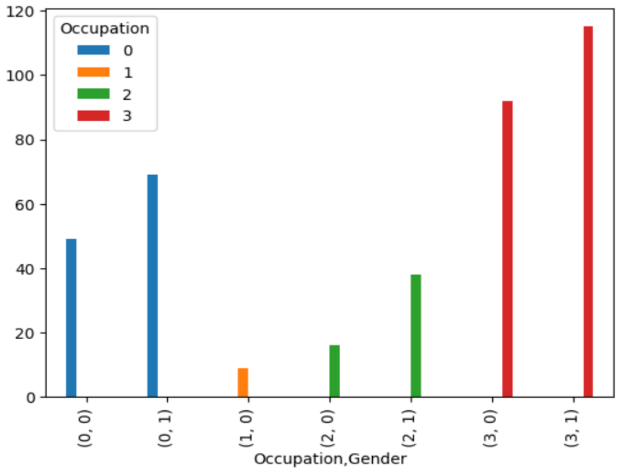


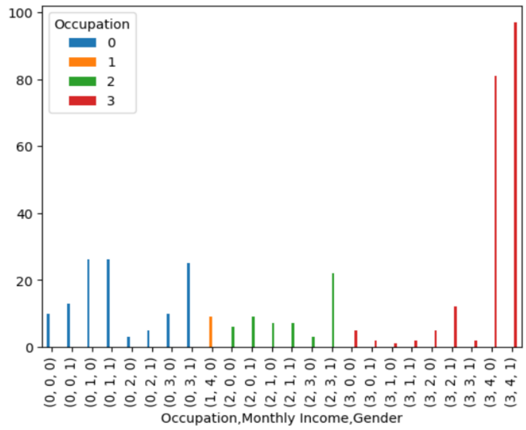


8) Bivariate and multivariate analysis









9) Converting the categorical to numerical column

	Age	Gender	Marital Status	Occupation	Monthly Income	Educational Qualifications	Family size	latitude	longitude	Pin code	Output	Feedback
0	20	0	2	3	4	2	4	12.9766	77.5993	560001	1	1
1	24	0	2	3	2	0	3	12.9770	77.5773	560009	1	1
2	22	1	2	3	2	2	3	12.9551	77.6593	560017	1	0
3	22	0	2	3	4	0	6	12.9473	77.5616	560019	1	1
4	22	1	2	3	2	2	4	12.9850	77.5533	560010	1	1
...	...	...	...	...	...	...	...	...	...	...	...	...
383	23	0	2	3	4	2	2	12.9766	77.5993	560001	1	1
384	23	0	2	3	4	2	4	12.9854	77.7081	560048	1	1
385	22	0	2	3	4	2	5	12.9850	77.5533	560010	1	1
386	23	1	2	3	2	2	2	12.9770	77.5773	560009	1	1
387	23	1	2	3	4	2	5	12.8988	77.5764	560078	1	1

388 rows × 12 columns

10. Feature selection:

	Age	Gender	Marital Status	Occupation	Monthly Income	Educational Qualifications	Family size	Pin code	Output	Feedback
0	20	0	2	3	4	2	4	560001	1	1
1	24	0	2	3	2	0	3	560009	1	1
2	22	1	2	3	2	2	3	560017	1	0
3	22	0	2	3	4	0	6	560019	1	1
4	22	1	2	3	2	2	4	560010	1	1
...	...	...	...	...	...	...	...	...	...	...
383	23	0	2	3	4	2	2	560001	1	1
384	23	0	2	3	4	2	4	560048	1	1
385	22	0	2	3	4	2	5	560010	1	1
386	23	1	2	3	2	2	2	560009	1	1
387	23	1	2	3	4	2	5	560078	1	1

388 rows × 10 columns

INSIGHTS/PATTERNS

1) **Majority of the singles nearly 60.05% are giving the positive feed back for the food delivery which senses majority of people who ordering the online food are singles**
2) **Students mostly ordering the online food nearly 47.9% of students are giving the positive feedback**
3) **Students who are males are nearly 29.65% are giving the positive feedback to online food delivery platforms**
4) **Age and family size have a good correlation.**
5) **Students with the n o income people are ordering the food most and they only giving nearly 45% of them are giving positive feedback**

STEP-4:

Splitting the data:

- Divide the data into train data and test data
- Train data is used for train the ML model

- We have train input and train output
- Train input will have all the independent columns
- Train output will have the dependent column or target or output column
- Test data is used to test the ML model\
- Test input will have all the independent columns
- Test output will have the dependent column or target or output column
- We have test input and test output
- Generally, train data should be more compared with test data then the model is well trained with large number of observations it will able to understand the more relations and patterns from the data and perform well on the test data and unseen data.
- Splitting the data randomly is most important in order to understand the more patterns about the data

```
X=food_csv.drop("Feedback",axis=1)
y=food_csv["Feedback"]
```

```
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.2,random_state=42)
```

STEP-5:

CHOOSE THE ML MODEL

As I am working on the feedback (positive or negative) it is classification problem so we can choose the classification algorithms

RANDOM FOREST

- random forest classifier is best example for the bagging
- in the random forest we will construct the multiple decision trees
- each decision tree will generate the output

- final output is considered based on majority of voting among the decision trees.

Training the data using random forest classifier

```
]: from sklearn.ensemble import RandomForestClassifier
   RFC=RandomForestClassifier()
   RFC.fit(X_train,y_train)
```

```
]: ▾ RandomForestClassifier
   RandomForestClassifier()
```

Predictions on test data

```
]: y_pred=RFC.predict(X_test)
```

```
]: y_pred
```

```
]: array([0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
          1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 0, 1,
          1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1,
          1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1])
```

STEP-6:

EVALUTION OF MODEL PERFORMANCE:

Metrics are used to evaluate the performance of the model

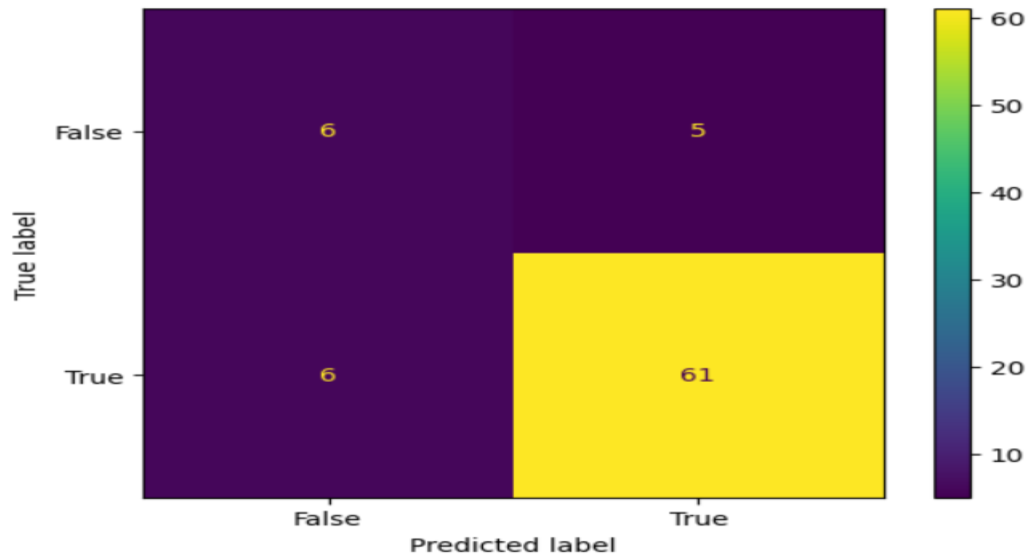
As it is classification problem

We have

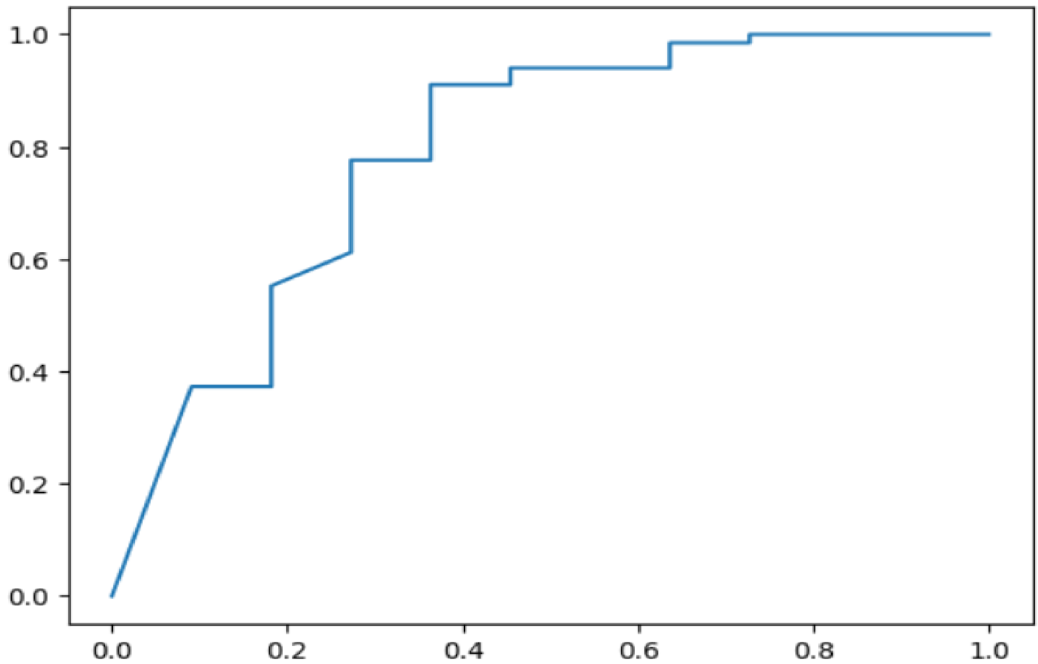
- Confusion matrix
- Accuracy
- Precision
- Recall
- F1 score
- Roc Auc curve

```
from sklearn.metrics import accuracy_score, f1_score, precision_score, recall_score, classification_report,roc_curve
acc_RF= round(accuracy_score(y_test,y_pred)*100,2)
f1_RF=round(f1_score(y_test,y_pred),2)
precision_RF=round(precision_score(y_test,y_pred),2)
recall_RF=round(recall_score(y_test,y_pred),2)
print("accuray is:",acc_RF)
print("F1 is:",f1_RF)
print("Precision is:",precision_RF)
print("Recall is:",recall_RF)
```

```
accuray is: 85.9
F1 is: 0.92
Precision is: 0.92
Recall is: 0.91
```




```
True negative:  6
False postive:  5
False negative:  6
True postive:  61
```



STEP-7

Hyperparameter tuning

Now we will choose the parameters in order to improve the performance of the model.

```
param_grid = {
    'n_estimators': [50, 100, 200],
    'max_features': ['auto', 'sqrt', 'log2'],
    'max_depth': [None, 10, 20],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4],
    'bootstrap': [True, False]
}
```