onlinefoods project

Predict Output Feature of Dataset

```
#importing libraries

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# importing csv file

path =r"C:\Users\venky\Downloads\onlinefoods.csv"
df = pd.read_csv(path)

df.head() #checking first 5 rows
```

```
   Age  Gender Marital Status Occupation  Monthly Income  \
0   20  Female         Single    Student       No Income
1   24  Female         Single    Student  Below Rs.10000
2   22    Male         Single    Student  Below Rs.10000
3   22  Female         Single    Student       No Income
4   22    Male         Single    Student  Below Rs.10000

  Educational Qualifications  Family size  latitude  longitude  Pin
code  \
0              Post Graduate            4   12.9766    77.5993
560001
1                   Graduate            3   12.9770    77.5773
560009
2              Post Graduate            3   12.9551    77.6593
560017
3                   Graduate            6   12.9473    77.5616
560019
4              Post Graduate            4   12.9850    77.5533
560010

  Output   Feedback Unnamed: 12
0    Yes   Positive         Yes
1    Yes   Positive         Yes
2    Yes  Negative          Yes
3    Yes   Positive         Yes
4    Yes   Positive         Yes
```

```
df.tail() #checking last 5 rows
```

```
     Age  Gender Marital Status Occupation  Monthly Income  \
383   23  Female         Single    Student       No Income
```

```
384   23  Female         Single    Student       No Income
385   22  Female         Single    Student       No Income
386   23    Male         Single    Student  Below Rs.10000
387   23    Male         Single    Student       No Income

    Educational Qualifications  Family size  latitude  longitude  Pin
code  \
383                Post Graduate            2   12.9766    77.5993
560001
384                Post Graduate            4   12.9854    77.7081
560048
385                Post Graduate            5   12.9850    77.5533
560010
386                Post Graduate            2   12.9770    77.5773
560009
387                Post Graduate            5   12.8988    77.5764
560078

    Output  Feedback Unnamed: 12
383    Yes  Positive         Yes
384    Yes  Positive         Yes
385    Yes  Positive         Yes
386    Yes  Positive         Yes
387    Yes  Positive         Yes
```

I] data preprocessing

01 duplicate data

```
# checking the shape of data frame
df.shape
print(f"data frame has shape of {df.shape}")

# checking duplicate values in data frame
df.duplicated().sum()
print(f"data frame has {df.duplicated().sum()} of duplicate values")

data frame has shape of (388, 13)
data frame has 103 of duplicate values

# dropping the duplicate values
df.drop_duplicates(keep='first',inplace=True) #keep ='first' keeps
first value and drops next duplicate value keep='last' keeps last
duplicate value

# checking the shape after duplicates have been removed
print(f"AFTER DROP DUPLICATE: data frame has shape of {df.shape}")

#checking if any duplicate data is present after removing duplicate
```

```
print(f"AFTER DROP DUPLICATE:  data frame has {df.duplicated().sum()}
of duplicate values")

AFTER DROP DUPLICATE: data frame has shape of (285, 13)
AFTER DROP DUPLICATE:  data frame has 0 of duplicate values

df.head()

   Age  Gender Marital Status Occupation   Monthly Income  \
0   20  Female         Single    Student        No Income
1   24  Female         Single    Student   Below Rs.10000
2   22    Male         Single    Student   Below Rs.10000
3   22  Female         Single    Student        No Income
4   22    Male         Single    Student   Below Rs.10000

  Educational Qualifications  Family size  latitude  longitude  Pin
code  \
0             Post Graduate            4    12.9766     77.5993
560001
1                  Graduate            3    12.9770     77.5773
560009
2             Post Graduate            3    12.9551     77.6593
560017
3                  Graduate            6    12.9473     77.5616
560019
4             Post Graduate            4    12.9850     77.5533
560010

  Output   Feedback Unnamed: 12
0    Yes   Positive         Yes
1    Yes   Positive         Yes
2    Yes  Negative          Yes
3    Yes   Positive         Yes
4    Yes   Positive         Yes
```

All 103 duplicate values are dropped

02 missing value treatment

```
# checking the data atypes of column
df.dtypes

Age                             int64
Gender                         object
Marital Status                 object
Occupation                     object
Monthly Income                 object
Educational Qualifications     object
Family size                     int64
latitude                      float64
```

```
longitude                     float64
Pin code                        int64
Output                         object
Feedback                       object
Unnamed: 12                    object
dtype: object
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 285 entries, 0 to 386
Data columns (total 13 columns):
 #   Column                      Non-Null Count  Dtype  
---  ------                      --------------  -----  
 0   Age                         285 non-null    int64  
 1   Gender                      285 non-null    object 
 2   Marital Status              285 non-null    object 
 3   Occupation                  285 non-null    object 
 4   Monthly Income              285 non-null    object 
 5   Educational Qualifications  285 non-null    object 
 6   Family size                 285 non-null    int64  
 7   latitude                    285 non-null    float64
 8   longitude                   285 non-null    float64
 9   Pin code                    285 non-null    int64  
 10  Output                      285 non-null    object 
 11  Feedback                    285 non-null    object 
 12  Unnamed: 12                 285 non-null    object 
dtypes: float64(2), int64(3), object(8)
memory usage: 31.2+ KB
```

```
#checking total number of null values in every column
df.isnull().sum()
```

```
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
Unnamed: 12                   0
dtype: int64
```

no missing values present in data frame

03 outlier treatment

```
df['Output'].value_counts(normalize=True)
```

```
Output
Yes    0.761404
No     0.238596
Name: proportion, dtype: float64
```

```
#checking data types , number of unique values and list of unique
values in a column
for i in df.columns:
    print("-"*30,i,"-"*30)
    print(f"data type of {i} = ",df[i].dtype) # gives column data type

    print(f"number of unique values in {i} = ",df[i].nunique()) #
gives number of unique values in a column
    print(f"list of unique value present in {i} is = ",df[i].unique())

    print("\n")
```

```
------------------------------ Age ------------------------------
data type of Age =  int64
number of unique values in Age =  16
list of unique value present in Age is =  [20 24 22 27 23 21 28 25 32
30 31 26 18 19 33 29]


------------------------------ Gender ------------------------------
data type of Gender =  object
number of unique values in Gender =  2
list of unique value present in Gender is =  ['Female' 'Male']


------------------------------ Marital Status
------------------------------
data type of Marital Status =  object
number of unique values in Marital Status =  3
list of unique value present in Marital Status is =  ['Single'
'Married' 'Prefer not to say']


------------------------------ Occupation
------------------------------
data type of Occupation =  object
number of unique values in Occupation =  4
list of unique value present in Occupation is =  ['Student' 'Employee'
'Self Employeed' 'House wife']
```

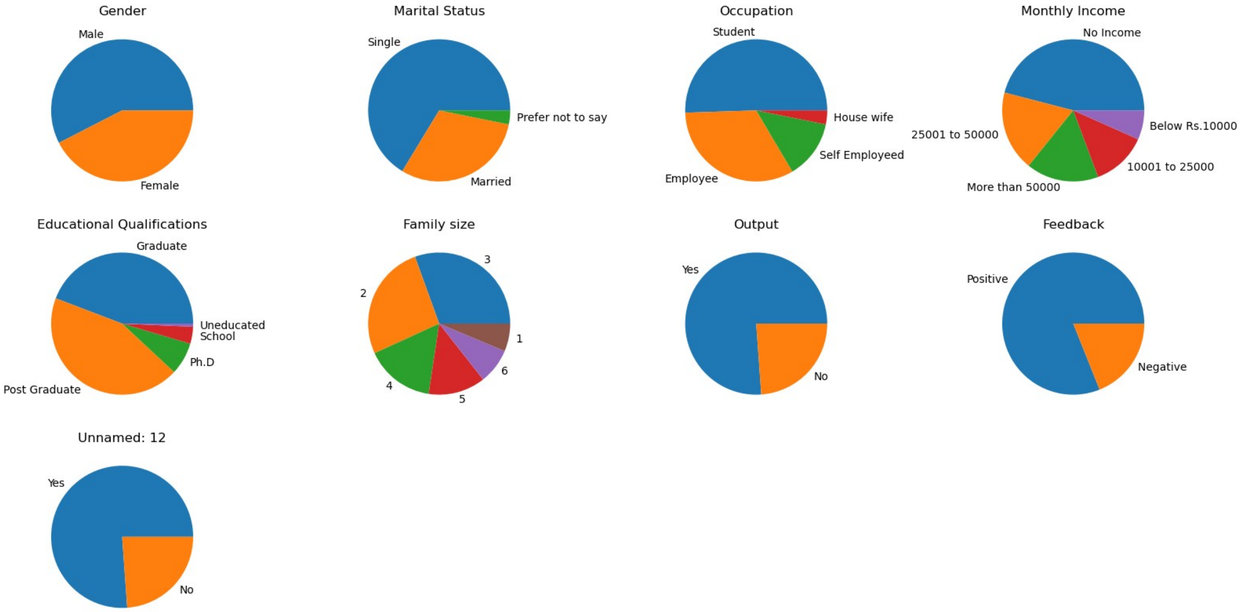
```
------------------------------ Monthly Income
------------------------------
data type of Monthly Income =  object
number of unique values in Monthly Income =  5
list of unique value present in Monthly Income is =  ['No Income'
'Below Rs.10000' 'More than 50000' '10001 to 25000'
 '25001 to 50000']


------------------------------ Educational Qualifications
------------------------------
data type of Educational Qualifications =  object
number of unique values in Educational Qualifications =  5
list of unique value present in Educational Qualifications is =
['Post Graduate' 'Graduate' 'Ph.D' 'Uneducated' 'School']


------------------------------ Family size
------------------------------
data type of Family size =  int64
number of unique values in Family size =  6
list of unique value present in Family size is =  [4 3 6 2 5 1]


------------------------------ latitude ------------------------------
data type of latitude =  float64
number of unique values in latitude =  77
list of unique value present in latitude is =  [12.9766 12.977
12.9551 12.9473 12.985  12.9299 12.9828 12.9854 12.8988
 12.9438 12.8893 12.9783 12.982  13.0298 12.9983 12.9925 12.9306
12.9353
 12.9155 13.0019 12.9698 12.9261 12.9119 12.9662 12.9565 13.0206
12.9635
 13.0067 12.8845 13.0158 12.9343 13.0012 12.9442 13.0487 12.9889
12.9335
 13.102  12.9048 12.9337 12.9037 13.0289 12.9561 12.9579 13.014
13.0138
 12.9537 12.998  13.0496 13.0166 13.0503 12.9883 13.0626 12.957
12.8652
 12.9757 12.9621 12.9217 13.0223 13.0262 13.0078 12.9105 12.8834
12.9149
 12.9706 13.0103 13.0641 12.9369 13.0809 12.9859 12.9866 12.9847
12.989
 12.9251 12.9967 13.0734 12.9515 12.9719]


------------------------------ longitude
------------------------------
data type of longitude =  float64
number of unique values in longitude =  76
```

```
list of unique value present in longitude is =  [77.5993 77.5773
77.6593 77.5616 77.5533 77.6848 77.6131 77.7081 77.5764
 77.5738 77.6399 77.6408 77.6256 77.6047 77.6409 77.5633 77.5434
77.5585
 77.5135 77.5713 77.75   77.6221 77.6446 77.6068 77.5484 77.6479
77.5821
 77.545  77.6036 77.539  77.6044 77.5995 77.6076 77.5923 77.5741
77.5691
 77.5864 77.6821 77.59   77.5376 77.54   77.5921 77.6309 77.5658
77.5877
 77.6176 77.6227 77.4941 77.6804 77.5529 77.5987 77.5284 77.5637
77.524
 77.5586 77.5936 77.7132 77.62   77.5577 77.4842 77.5486 77.5635
77.6529
 77.5796 77.5931 77.6407 77.5565 77.6713 77.4904 77.5491 77.5332
77.4992
 77.7582 77.5464 77.4921 77.5128]


------------------------------ Pin code ------------------------------
data type of Pin code =  int64
number of unique values in Pin code =  77
list of unique value present in Pin code is =  [560001 560009 560017
560019 560010 560103 560042 560048 560078 560004
 560068 560038 560008 560032 560033 560021 560085 560050 560098 560003
 560066 560034 560102 560025 560026 560043 560002 560086 560076 560096
 560029 560046 560030 560024 560020 560028 560064 560036 560011 560061
 560022 560027 560007 560012 560006 560047 560005 560073 560016 560013
 560051 560015 560018 560109 560023 560104 560041 560049 560045 560055
 560060 560062 560070 560075 560080 560092 560095 560097 560093 560091
 560100 560079 560059 560067 560014 560056 560072]


------------------------------ Output ------------------------------
data type of Output =  object
number of unique values in Output =  2
list of unique value present in Output is =  ['Yes' 'No']


------------------------------ Feedback ------------------------------
data type of Feedback =  object
number of unique values in Feedback =  2
list of unique value present in Feedback is =  ['Positive' 'Negative
']


------------------------------ Unnamed: 12
------------------------------
data type of Unnamed: 12 =  object
number of unique values in Unnamed: 12 =  2
```

```
list of unique value present in Unnamed: 12 is =  ['Yes' 'No']



df.info()

<class 'pandas.core.frame.DataFrame'>
Index: 285 entries, 0 to 386
Data columns (total 13 columns):
 #   Column                      Non-Null Count  Dtype  
---  ------                      --------------  -----  
 0   Age                         285 non-null    int64  
 1   Gender                      285 non-null    object 
 2   Marital Status              285 non-null    object 
 3   Occupation                  285 non-null    object 
 4   Monthly Income              285 non-null    object 
 5   Educational Qualifications  285 non-null    object 
 6   Family size                 285 non-null    int64  
 7   latitude                    285 non-null    float64
 8   longitude                   285 non-null    float64
 9   Pin code                    285 non-null    int64  
 10  Output                      285 non-null    object 
 11  Feedback                    285 non-null    object 
 12  Unnamed: 12                 285 non-null    object 
dtypes: float64(2), int64(3), object(8)
memory usage: 31.2+ KB
```

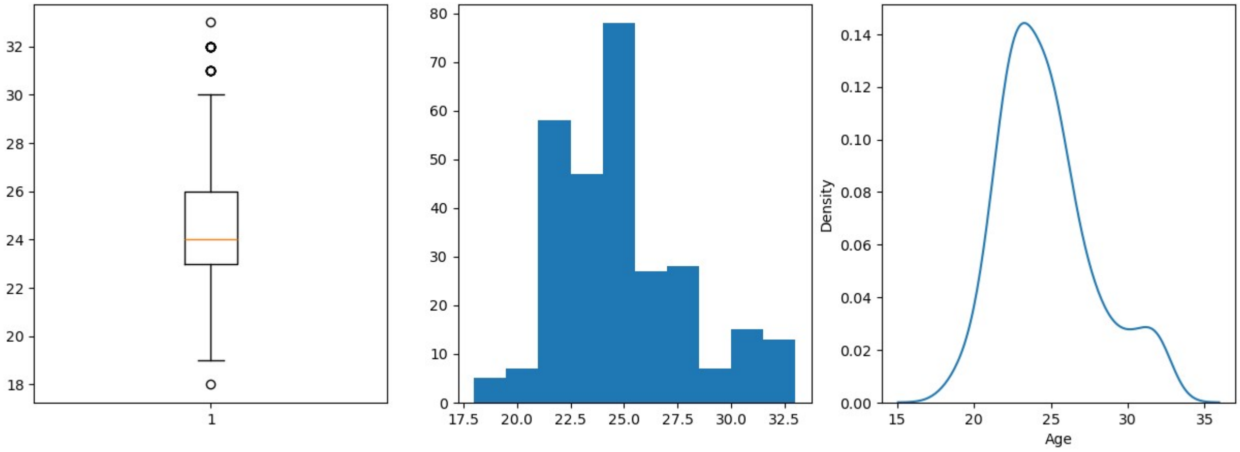
pie charts for columns having less than or equal to 10 unique values

```
count=1
fig = plt.figure(figsize =(20,10))
for i in df.columns:
    if df[i].nunique()<=10:
        plt.subplot(3,4,count)
        count+=1
        figure=plt

plt.pie(df[i].value_counts().values,labels=df[i].value_counts().index)
        plt.title(i)
plt.show()
```



Age is the only numerical columns

```
fog=plt.figure(figsize=(15,5))
plt.subplot(1,3,1)
plt.boxplot(df['Age'])
plt.subplot(1,3,2)
plt.hist(df['Age'])
plt.subplot(1,3,3)
sns.kdeplot(data=df,x='Age')
plt.show()
```



Z-Score Outlier Detection

```
# importing scipy library for z-score and iqr method
import scipy.stats as stats
```

```
count=0
for i in stats.zscore(df['Age']):
    if i>3 or i<-3:
        count+=1
print("outliers in Age ",count)
```

```
outliers in Age  0
```

Checked through Z-score, There are no outliers present

04 Feature Selection

dropping Unnamed: 12 column as we can see that Unnamed: 12 and Output both are same column

```
c=0
for i,j in zip(df['Output'],df['Unnamed: 12']):
    if i==j:
        c+=1
print(c)
```

```
285
```

```
c=(df['Output']==df['Unnamed: 12']).sum()
print(c)
```

```
285
```

```
df.drop('Unnamed: 12', axis=1, inplace=True)
```

dropping latitude and longitude column as we already have pincode column for location

```
df.drop('latitude', axis=1, inplace=True)
```

```
df.drop('longitude', axis=1, inplace=True)
```

05 Data Splitting

splitting data in x and y

```
# spiltting data in x and y
x=df.drop(columns='Output')
y=df['Output']
```

06 label encoding on categorical columns

```
# performing label encoding on categorical columns
from sklearn.preprocessing import LabelEncoder
le=LabelEncoder()
y=pd.DataFrame(le.fit_transform(y))
```

```
for i in x.columns:
    if i!= 'Age':
        x[i]=le.fit_transform(x[i])
    else:
        continue


x.head()
```

```
   Age  Gender  Marital Status  Occupation  Monthly Income  \
0   20       0               2           3               4
1   24       0               2           3               2
2   22       1               2           3               2
3   22       0               2           3               4
4   22       1               2           3               2

   Educational Qualifications  Family size  Pin code  Feedback
0                           2            3         0         1
1                           0            2         8         1
2                           2            2        16         0
3                           0            5        18         1
4                           2            3         9         1
```

```
y.head()
```

```
   0
0  1
1  1
2  1
3  1
4  1
```

splitting data into train test

```
# splitting data into train test
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=30)

for i in [x_train,x_test,y_train,y_test]:
    print(i.shape)
```

```
(228, 9)
(57, 9)
(228, 1)
(57, 1)
```

07 standard scaling on numerical column in x_train and x_test

```
# standard scaling on numerical column in y_train and y_test
from sklearn.preprocessing import StandardScaler
ss=StandardScaler()
x_train['Age']=(ss.fit_transform(x_train[['Age']]))
x_test['Age']=(ss.fit_transform(x_test[['Age']]))

x_train.head()
```

```
          Age  Gender  Marital Status  Occupation  Monthly Income  \
65   0.124063       1               2           3               4
101 -0.533758       1               2           3               4
386 -0.533758       1               2           3               2
73  -0.533758       1               2           3               4
288  0.124063       0               2           3               4

     Educational Qualifications  Family size  Pin code  Feedback
65                            2            5        39         1
101                           2            1        29         1
386                           2            1         8         1
73                            2            1        59         1
288                           2            2        13         1
```

```
x_test.head()
```

```
          Age  Gender  Marital Status  Occupation  Monthly Income  \
69  -0.297560       0               0           0               3
239 -0.630126       0               2           3               4
62  -0.630126       1               2           3               1
224  0.035007       0               0           3               4
131  2.362973       0               0           0               3

     Educational Qualifications  Family size  Pin code  Feedback
69                            1            3         3         1
239                           2            2        68         1
62                            2            0        69         1
224                           2            1        50         1
131                           0            0         2         1
```

II] model building

01 LogisticRegression

```
from sklearn.linear_model import LogisticRegression

model=LogisticRegression()

model.fit(x_train,y_train)
```

```
C:\Users\venky\anaconda3\Lib\site-packages\sklearn\utils\
validation.py:1339: DataConversionWarning: A column-vector y was
passed when a 1d array was expected. Please change the shape of y to
(n_samples, ), for example using ravel().
  y = column_or_1d(y, warn=True)
```

```
LogisticRegression()
```

```
y_train_predict=model.predict(x_train)
```

```
from sklearn.metrics import
accuracy_score,confusion_matrix,classification_report,roc_curve,roc_au
c_score
```

```
train_accuracy=accuracy_score(y_train,y_train_predict)
print(train_accuracy)
```

```
0.8596491228070176
```

```
cm=confusion_matrix(y_train,y_train_predict)
cm
```

```
array([[ 32,  24],
       [  8, 164]], dtype=int64)
```

```
tn,fp,fn,tp=confusion_matrix(y_train,y_train_predict).ravel()
```

```
for i in [tn,fp,fn,tp]:
    print(i)
```

```
32
24
8
164
```

```
y_test_predict=model.predict(x_test)
```

```
test_accuracy=accuracy_score(y_test,y_test_predict)
print(test_accuracy)
```

```
0.8421052631578947
```

```
test_cm=confusion_matrix(y_test,y_test_predict)
test_cm
```

```
array([[ 8,  4],
       [ 5, 40]], dtype=int64)
```

```
# def calculate_acc(xtrain ,x_test ,y_train ,y_test):
#     models
=[LogisticRegression(),DecisionTreeClassifier(),SVC(),RandomForestClas
sifier(),XGBClassifier()]
#     data_frame = pd.DataFrame()
#     acc =[]
#     recall =[]
#     precision =[]
#     f1=[]
#     for mod in models :
#         model_ = mod
#         model_.fit(x_train ,y_train)

#         y_pred_test =model_.predict(x_test)
#         acc.append(np.round(accuracy_score(y_pred_test,y_test),2))
#         recall.append(np.round(recall_score(y_pred_test,y_test),2))
#         precision.append(precision_score(y_pred_test,y_test))
#         f1.append(f1_score(y_pred_test,y_test).round(2))


#     tabel
=pd.DataFrame(index=["LogisticRegression","DecisionTreeClassifier","SV
C","RandomForestClassifier","XGBClassifier"],
#                         columns=["acc" ,"recall","precision","F1"] )
#     tabel["acc"]    = acc
#     tabel["recall"] =recall
#     tabel["precision"] = precision
#     tabel["F1"] =f1
#     return tabel
#     print("           Accuracy Measurement")
# calculate_acc(x_train ,x_test ,y_train ,y_test)
```

02 DecisionTreeClassifier

```
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import precision_score, \
    recall_score, confusion_matrix, classification_report, \
    accuracy_score, f1_score

dtc=DecisionTreeClassifier(max_depth=2,criterion='entropy') # Using
hyper parameter max_depth=2 and criterion='entropy'
dtc.fit(x_train,y_train)

# Training data set
y_train_predict=dtc.predict(x_train)
print("-"*30,"Train_Result","-"*30)
print("\n")
print ('Accuracy:', accuracy_score(y_train, y_train_predict) )
print ('F1 score:', f1_score(y_train, y_train_predict) )
print ('Recall:', recall_score(y_train, y_train_predict) )
print ('Precision:', precision_score(y_train, y_train_predict) )
print ('clasification report:\n', classification_report(y_train,
y_train_predict) )
print("\n")
```

```
# Testing data set
y_test_predict=dtc.predict(x_test)
print("-"*30,"Test_Result","-"*30)
print("\n")
print("test_accuracy :",test_accuracy)
print ('Accuracy:', accuracy_score(y_test, y_test_predict))
print ('F1 score:', f1_score(y_test, y_test_predict))
print ('Recall:', recall_score(y_test, y_test_predict))
print ('Precision:', precision_score(y_test, y_test_predict))
print ('clasification report:\n', classification_report(y_test,
y_test_predict))
```

```
------------------------------ Train_Result
------------------------------


Accuracy: 0.8640350877192983
F1 score: 0.9141274238227147
Recall: 0.9593023255813954
Precision: 0.873015873015873
clasification report:
               precision    recall  f1-score   support

           0       0.82      0.57      0.67        56
           1       0.87      0.96      0.91       172

    accuracy                           0.86       228
   macro avg       0.85      0.77      0.79       228
weighted avg       0.86      0.86      0.86       228



------------------------------ Test_Result
------------------------------


test_accuracy : 0.8421052631578947
Accuracy: 0.8070175438596491
F1 score: 0.8791208791208791
Recall: 0.8888888888888888
Precision: 0.8695652173913043
clasification report:
               precision    recall  f1-score   support

           0       0.55      0.50      0.52        12
           1       0.87      0.89      0.88        45

    accuracy                           0.81        57
   macro avg       0.71      0.69      0.70        57
weighted avg       0.80      0.81      0.80        57
```

03 RandomForestClassifier

```
from sklearn.ensemble import RandomForestClassifier

rfc=RandomForestClassifier(n_estimators = 100,max_depth=4)

rfc=RandomForestClassifier(n_estimators = 100,max_depth=4)
rfc.fit(x_train,y_train)

# Training data set
y_train_predict=rfc.predict(x_train)
print("-"*30,"Train_Result","-"*30)
print("\n")
print ('Accuracy:', accuracy_score(y_train, y_train_predict) )
print ('F1 score:', f1_score(y_train, y_train_predict) )
print ('Recall:', recall_score(y_train, y_train_predict) )
print ('Precision:', precision_score(y_train, y_train_predict) )
print ('clasification report:\n', classification_report(y_train,
y_train_predict) )
print("\n")

# Testing data set
y_test_predict=rfc.predict(x_test)
print("-"*30,"Test_Result","-"*30)
print("\n")
print("test_accuracy :",test_accuracy)
print ('Accuracy:', accuracy_score(y_test, y_test_predict))
print ('F1 score:', f1_score(y_test, y_test_predict))
print ('Recall:', recall_score(y_test, y_test_predict))
print ('Precision:', precision_score(y_test, y_test_predict))
print ('clasification report:\n', classification_report(y_test,
y_test_predict))
```

```
------------------------------ Train_Result
------------------------------


Accuracy: 0.8903508771929824
F1 score: 0.9318801089918256
Recall: 0.9941860465116279
Precision: 0.8769230769230769
clasification report:
               precision    recall  f1-score   support

           0       0.97      0.57      0.72        56
           1       0.88      0.99      0.93       172

    accuracy                           0.89       228
   macro avg       0.92      0.78      0.83       228
weighted avg       0.90      0.89      0.88       228
```

```
------------------------------ Test_Result
------------------------------


test_accuracy : 0.8421052631578947
Accuracy: 0.8245614035087719
F1 score: 0.8913043478260869
Recall: 0.9111111111111111
Precision: 0.8723404255319149
clasification report:
               precision    recall  f1-score   support

           0       0.60      0.50      0.55        12
           1       0.87      0.91      0.89        45

    accuracy                           0.82        57
   macro avg       0.74      0.71      0.72        57
weighted avg       0.82      0.82      0.82        57


C:\Users\venky\anaconda3\Lib\site-packages\sklearn\base.py:1473:
DataConversionWarning: A column-vector y was passed when a 1d array
was expected. Please change the shape of y to (n_samples,), for
example using ravel().
  return fit_method(estimator, *args, **kwargs)
```

```
rfc.fit(x_train,y_train)

# Training data set
y_train_predict=rfc.predict(x_train)
print("-"*30,"Train_Result","-"*30)
print("\n")
print ('Accuracy:', accuracy_score(y_train, y_train_predict) )
print ('F1 score:', f1_score(y_train, y_train_predict) )
print ('Recall:', recall_score(y_train, y_train_predict) )
print ('Precision:', precision_score(y_train, y_train_predict) )
print ('clasification report:\n', classification_report(y_train,
y_train_predict) )
print("\n")

# Testing data set
y_test_predict=rfc.predict(x_test)
print("-"*30,"Test_Result","-"*30)
print("\n")
print("test_accuracy :",test_accuracy)
print ('Accuracy:', accuracy_score(y_test, y_test_predict))
print ('F1 score:', f1_score(y_test, y_test_predict))
print ('Recall:', recall_score(y_test, y_test_predict))
print ('Precision:', precision_score(y_test, y_test_predict))
print ('clasification report:\n', classification_report(y_test,
y_test_predict))
```

```
C:\Users\venky\anaconda3\Lib\site-packages\sklearn\base.py:1473:
DataConversionWarning: A column-vector y was passed when a 1d array
was expected. Please change the shape of y to (n_samples,), for
example using ravel().
  return fit_method(estimator, *args, **kwargs)

------------------------------ Train_Result
------------------------------


Accuracy: 0.8947368421052632
F1 score: 0.9344262295081968
Recall: 0.9941860465116279
Precision: 0.8814432989690721
clasification report:
               precision    recall  f1-score   support

           0       0.97      0.59      0.73        56
           1       0.88      0.99      0.93       172

    accuracy                           0.89       228
   macro avg       0.93      0.79      0.83       228
weighted avg       0.90      0.89      0.89       228



------------------------------ Test_Result
------------------------------


test_accuracy : 0.8421052631578947
Accuracy: 0.8245614035087719
F1 score: 0.8913043478260869
Recall: 0.9111111111111111
Precision: 0.8723404255319149
clasification report:
               precision    recall  f1-score   support

           0       0.60      0.50      0.55        12
           1       0.87      0.91      0.89        45

    accuracy                           0.82        57
   macro avg       0.74      0.71      0.72        57
weighted avg       0.82      0.82      0.82        57
```

```
number_trees=[i for i in range(100,2100,100)]
oob_errors=[]
```

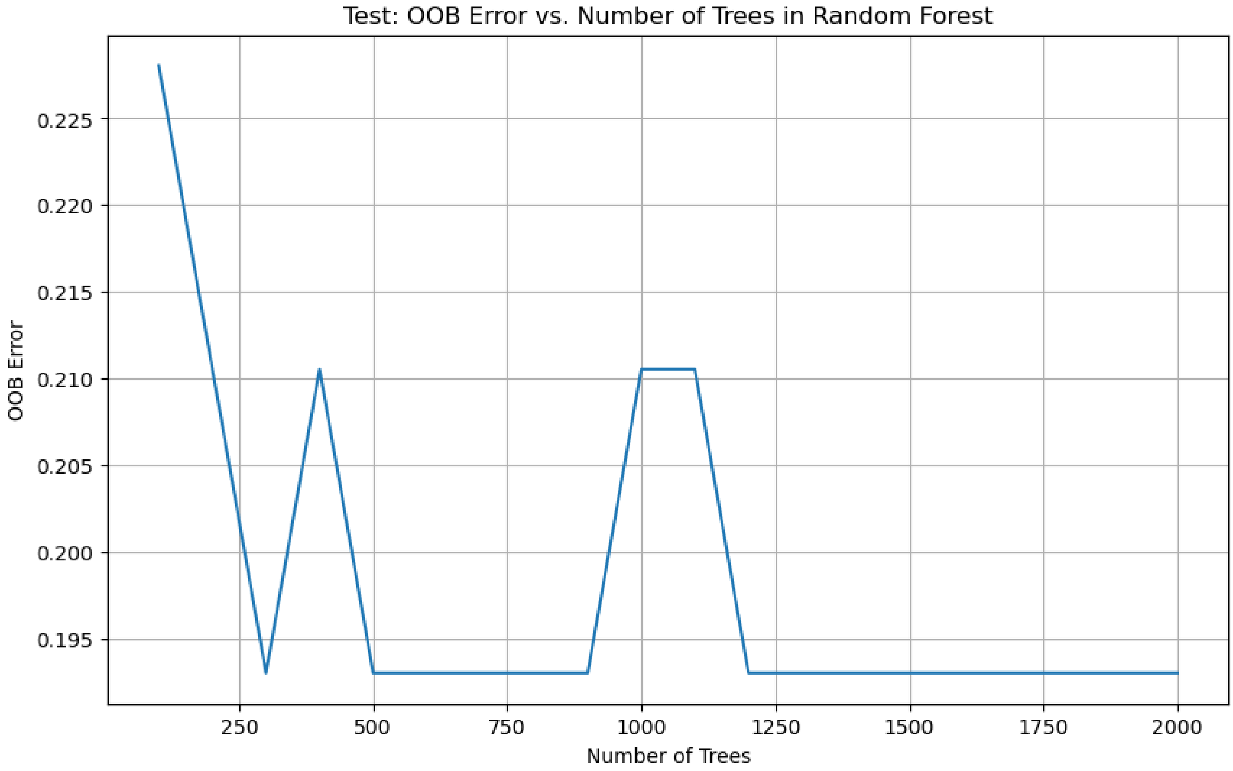
```
for i in number_trees:
    rfc=RandomForestClassifier(n_estimators=i, oob_score=True,
random_state=42, bootstrap=True)
    rfc.fit(x_test,y_test)
    y_test_predict=rfc.predict(x_test)
    oob_errors.append(1 - rfc.oob_score_)
```

```
C:\Users\venky\anaconda3\Lib\site-packages\sklearn\base.py:1473:
DataConversionWarning: A column-vector y was passed when a 1d array
was expected. Please change the shape of y to (n_samples,), for
example using ravel().
  return fit_method(estimator, *args, **kwargs)
C:\Users\venky\anaconda3\Lib\site-packages\sklearn\base.py:1473:
DataConversionWarning: A column-vector y was passed when a 1d array
was expected. Please change the shape of y to (n_samples,), for
example using ravel().
  return fit_method(estimator, *args, **kwargs)
C:\Users\venky\anaconda3\Lib\site-packages\sklearn\base.py:1473:
DataConversionWarning: A column-vector y was passed when a 1d array
was expected. Please change the shape of y to (n_samples,), for
example using ravel().
  return fit_method(estimator, *args, **kwargs)
C:\Users\venky\anaconda3\Lib\site-packages\sklearn\base.py:1473:
DataConversionWarning: A column-vector y was passed when a 1d array
was expected. Please change the shape of y to (n_samples,), for
example using ravel().
  return fit_method(estimator, *args, **kwargs)
C:\Users\venky\anaconda3\Lib\site-packages\sklearn\base.py:1473:
DataConversionWarning: A column-vector y was passed when a 1d array
was expected. Please change the shape of y to (n_samples,), for
example using ravel().
  return fit_method(estimator, *args, **kwargs)
C:\Users\venky\anaconda3\Lib\site-packages\sklearn\base.py:1473:
DataConversionWarning: A column-vector y was passed when a 1d array
was expected. Please change the shape of y to (n_samples,), for
example using ravel().
  return fit_method(estimator, *args, **kwargs)
C:\Users\venky\anaconda3\Lib\site-packages\sklearn\base.py:1473:
DataConversionWarning: A column-vector y was passed when a 1d array
was expected. Please change the shape of y to (n_samples,), for
example using ravel().
  return fit_method(estimator, *args, **kwargs)
C:\Users\venky\anaconda3\Lib\site-packages\sklearn\base.py:1473:
DataConversionWarning: A column-vector y was passed when a 1d array
was expected. Please change the shape of y to (n_samples,), for
example using ravel().
  return fit_method(estimator, *args, **kwargs)
C:\Users\venky\anaconda3\Lib\site-packages\sklearn\base.py:1473:
DataConversionWarning: A column-vector y was passed when a 1d array
```

```
was expected. Please change the shape of y to (n_samples,), for
example using ravel().
  return fit_method(estimator, *args, **kwargs)
C:\Users\venky\anaconda3\Lib\site-packages\sklearn\base.py:1473:
DataConversionWarning: A column-vector y was passed when a 1d array
was expected. Please change the shape of y to (n_samples,), for
example using ravel().
  return fit_method(estimator, *args, **kwargs)
C:\Users\venky\anaconda3\Lib\site-packages\sklearn\base.py:1473:
DataConversionWarning: A column-vector y was passed when a 1d array
was expected. Please change the shape of y to (n_samples,), for
example using ravel().
  return fit_method(estimator, *args, **kwargs)
C:\Users\venky\anaconda3\Lib\site-packages\sklearn\base.py:1473:
DataConversionWarning: A column-vector y was passed when a 1d array
was expected. Please change the shape of y to (n_samples,), for
example using ravel().
  return fit_method(estimator, *args, **kwargs)
C:\Users\venky\anaconda3\Lib\site-packages\sklearn\base.py:1473:
DataConversionWarning: A column-vector y was passed when a 1d array
was expected. Please change the shape of y to (n_samples,), for
example using ravel().
  return fit_method(estimator, *args, **kwargs)
C:\Users\venky\anaconda3\Lib\site-packages\sklearn\base.py:1473:
DataConversionWarning: A column-vector y was passed when a 1d array
was expected. Please change the shape of y to (n_samples,), for
example using ravel().
  return fit_method(estimator, *args, **kwargs)
C:\Users\venky\anaconda3\Lib\site-packages\sklearn\base.py:1473:
DataConversionWarning: A column-vector y was passed when a 1d array
was expected. Please change the shape of y to (n_samples,), for
example using ravel().
  return fit_method(estimator, *args, **kwargs)
C:\Users\venky\anaconda3\Lib\site-packages\sklearn\base.py:1473:
DataConversionWarning: A column-vector y was passed when a 1d array
was expected. Please change the shape of y to (n_samples,), for
example using ravel().
  return fit_method(estimator, *args, **kwargs)
C:\Users\venky\anaconda3\Lib\site-packages\sklearn\base.py:1473:
DataConversionWarning: A column-vector y was passed when a 1d array
was expected. Please change the shape of y to (n_samples,), for
example using ravel().
  return fit_method(estimator, *args, **kwargs)
C:\Users\venky\anaconda3\Lib\site-packages\sklearn\base.py:1473:
DataConversionWarning: A column-vector y was passed when a 1d array
was expected. Please change the shape of y to (n_samples,), for
example using ravel().
  return fit_method(estimator, *args, **kwargs)
C:\Users\venky\anaconda3\Lib\site-packages\sklearn\base.py:1473:
```

```
DataConversionWarning: A column-vector y was passed when a 1d array
was expected. Please change the shape of y to (n_samples,), for
example using ravel().
  return fit_method(estimator, *args, **kwargs)
C:\Users\venky\anaconda3\Lib\site-packages\sklearn\base.py:1473:
DataConversionWarning: A column-vector y was passed when a 1d array
was expected. Please change the shape of y to (n_samples,), for
example using ravel().
  return fit_method(estimator, *args, **kwargs)
```

```
plt.figure(figsize=(10, 6))
plt.plot(number_trees, oob_errors)
plt.title("Test: OOB Error vs. Number of Trees in Random Forest")
plt.xlabel("Number of Trees")
plt.ylabel("OOB Error")
plt.grid()
plt.show()
```



04 3-Models in 1-Function

```
import numpy as np
def calculate_acc(xtrain ,x_test ,y_train ,y_test):
    models
=[LogisticRegression(),DecisionTreeClassifier(max_depth=2,criterion='e
ntropy'),RandomForestClassifier(n_estimators=600, random_state=42,
bootstrap=True)]
```

```
    data_frame = pd.DataFrame()
    acc =[]
    recall =[]
    precision =[]
    f1=[]
    for mod in models :
        model_ = mod
        model_.fit(x_train ,y_train)

        y_pred_test =model_.predict(x_test)
        acc.append(np.round(accuracy_score(y_pred_test,y_test),2))
        recall.append(np.round(recall_score(y_pred_test,y_test),2))
        precision.append(precision_score(y_pred_test,y_test))
        f1.append(f1_score(y_pred_test,y_test).round(2))


    tabel
=pd.DataFrame(index=["LogisticRegression","DecisionTreeClassifier" ,"R
andomForestClassifier"],
                        columns=["acc" ,"recall","precision","F1"] )
    tabel["acc"]    = acc
    tabel["recall"] =recall
    tabel["precision"] = precision
    tabel["F1"] =f1
    return tabel
    print("Accuracy Measurement")
calculate_acc(x_train, x_test, y_train, y_test)
```

```
C:\Users\venky\anaconda3\Lib\site-packages\sklearn\utils\
validation.py:1339: DataConversionWarning: A column-vector y was
passed when a 1d array was expected. Please change the shape of y to
(n_samples, ), for example using ravel().
  y = column_or_1d(y, warn=True)
C:\Users\venky\anaconda3\Lib\site-packages\sklearn\base.py:1473:
DataConversionWarning: A column-vector y was passed when a 1d array
was expected. Please change the shape of y to (n_samples,), for
example using ravel().
  return fit_method(estimator, *args, **kwargs)
```

```
                        acc  recall  precision    F1
LogisticRegression      0.84    0.91   0.888889  0.90
DecisionTreeClassifier  0.81    0.87   0.888889  0.88
RandomForestClassifier  0.82    0.87   0.911111  0.89
```