ml-project-heart-disease-2

November 2, 2025

1 Introduction

1.1 Heart Attack Prediction System

Heart disease is a leading cause of death worldwide. Early detection and prevention are crucial in reducing the risk and impact of heart attacks. This script is designed to predict the likelihood of a heart attack based on various medical and personal attributes provided by the user. The system uses a machine learning model trained on a dataset of patient information to make these predictions.

2 How It Works

3 Data Loading and Preprocessing:

The script starts by loading a dataset containing historical patient data, which includes various attributes such as age, gender, chest pain type, blood pressure, cholesterol levels, and more. The dataset is cleaned and prepared for training a machine learning model.

4 Model Training:

A logistic regression model is trained using the historical data. This model learns the relationships between the input features and the likelihood of a heart attack.

5 User Input:

The script then prompts the user to enter specific details about their health and personal information. These details include age, gender, chest pain type, resting blood pressure, cholesterol levels, fasting blood sugar, electrocardiographic results, maximum heart rate achieved, exercise-induced angina, ST depression, slope of peak exercise ST segment, number of major vessels colored by fluoroscopy, and thalassemia status.

6 Prediction:

Using the trained model, the script predicts whether the user is likely to experience a heart attack and provides the probability of this prediction. The prediction is based on the user-provided data and the learned patterns from the historical dataset.

7 Importing the Dependencies

[ ]:
```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
```

[ ]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score, classification_report,␣
 ↪confusion_matrix, recall_score, precision_score, cohen_kappa_score
from sklearn import preprocessing
from sklearn.preprocessing import StandardScaler, LabelEncoder
from sklearn.linear_model import  LogisticRegression
from sklearn.ensemble import RandomForestClassifier, RandomForestClassifier,␣
 ↪GradientBoostingClassifier
from imblearn.over_sampling import SMOTE
import xgboost as xgb
!pip install catboost
from catboost import CatBoostRegressor
```

```
Collecting catboost
  Downloading catboost-1.2.5-cp310-cp310-manylinux2014_x86_64.whl.metadata (1.2
kB)
Requirement already satisfied: graphviz in /usr/local/lib/python3.10/dist-
packages (from catboost) (0.20.3)
Requirement already satisfied: matplotlib in /usr/local/lib/python3.10/dist-
packages (from catboost) (3.7.1)
Requirement already satisfied: numpy>=1.16.0 in /usr/local/lib/python3.10/dist-
packages (from catboost) (1.25.2)
Requirement already satisfied: pandas>=0.24 in /usr/local/lib/python3.10/dist-
packages (from catboost) (2.0.3)
Requirement already satisfied: scipy in /usr/local/lib/python3.10/dist-packages
(from catboost) (1.11.4)
Requirement already satisfied: plotly in /usr/local/lib/python3.10/dist-packages
(from catboost) (5.15.0)
Requirement already satisfied: six in /usr/local/lib/python3.10/dist-packages
(from catboost) (1.16.0)
Requirement already satisfied: python-dateutil>=2.8.2 in
/usr/local/lib/python3.10/dist-packages (from pandas>=0.24->catboost) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-
packages (from pandas>=0.24->catboost) (2023.4)
```

```
Requirement already satisfied: tzdata>=2022.1 in /usr/local/lib/python3.10/dist-
packages (from pandas>=0.24->catboost) (2024.1)
Requirement already satisfied: contourpy>=1.0.1 in
/usr/local/lib/python3.10/dist-packages (from matplotlib->catboost) (1.2.1)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.10/dist-
packages (from matplotlib->catboost) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in
/usr/local/lib/python3.10/dist-packages (from matplotlib->catboost) (4.53.1)
Requirement already satisfied: kiwisolver>=1.0.1 in
/usr/local/lib/python3.10/dist-packages (from matplotlib->catboost) (1.4.5)
Requirement already satisfied: packaging>=20.0 in
/usr/local/lib/python3.10/dist-packages (from matplotlib->catboost) (24.1)
Requirement already satisfied: pillow>=6.2.0 in /usr/local/lib/python3.10/dist-
packages (from matplotlib->catboost) (9.4.0)
Requirement already satisfied: pyparsing>=2.3.1 in
/usr/local/lib/python3.10/dist-packages (from matplotlib->catboost) (3.1.2)
Requirement already satisfied: tenacity>=6.2.0 in
/usr/local/lib/python3.10/dist-packages (from plotly->catboost) (8.5.0)
Downloading catboost-1.2.5-cp310-cp310-manylinux2014_x86_64.whl (98.2 MB)
                         98.2/98.2 MB
7.9 MB/s eta 0:00:00
Installing collected packages: catboost
Successfully installed catboost-1.2.5
```

8 Data Collection and Processing

[ ]:
```
# loading the csv data to a Pandas DataFrame

df= pd.read_csv('/content/HeartDisease.csv')
df
```

[ ]:

	age	gender	chest_pain	rest_bps	cholestrol	fasting_blood_sugar	\
0	63	1	3	145	233	1	
1	37	1	2	130	250	0	
2	41	0	1	130	204	0	
3	56	1	1	120	236	0	
4	57	0	0	120	354	0	
..	…	…	…	…	…	…	
298	57	0	0	140	241	0	
299	45	1	3	110	264	0	
300	68	1	0	144	193	1	
301	57	1	0	130	131	0	
302	57	0	1	130	236	0	

	rest_ecg	thalach	exer_angina	old_peak	slope	ca	thalassemia	target
0	0	150	0	2.3	0	0	1	1
1	1	187	0	3.5	0	0	2	1

```
2           0      172            0       1.4      2   0            2       1
3           1      178            0       0.8      2   0            2       1
4           1      163            1       0.6      2   0            2       1
..        ...      ...          ...       ...    ...  ..          ...     ...
298         1      123            1       0.2      1   0            3       0
299         1      132            0       1.2      1   0            3       0
300         1      141            0       3.4      1   2            3       0
301         1      115            1       1.2      1   1            3       0
302         0      174            0       0.0      1   1            2       0

[303 rows x 14 columns]
```

[ ]:
```
df.head(5)  # print first 5 rows of the dataset
```

[ ]:
```
   age  gender  chest_pain  rest_bps  cholestrol  fasting_blood_sugar  \
0   63       1           3       145         233                    1
1   37       1           2       130         250                    0
2   41       0           1       130         204                    0
3   56       1           1       120         236                    0
4   57       0           0       120         354                    0

   rest_ecg  thalach  exer_angina  old_peak  slope  ca  thalassemia  target
0         0      150            0       2.3      0   0            1       1
1         1      187            0       3.5      0   0            2       1
2         0      172            0       1.4      2   0            2       1
3         1      178            0       0.8      2   0            2       1
4         1      163            1       0.6      2   0            2       1
```

[ ]:
```
# print last 5 rows of the dataset

df.tail()
```

[ ]:
```
     age  gender  chest_pain  rest_bps  cholestrol  fasting_blood_sugar  \
298   57       0           0       140         241                    0
299   45       1           3       110         264                    0
300   68       1           0       144         193                    1
301   57       1           0       130         131                    0
302   57       0           1       130         236                    0

     rest_ecg  thalach  exer_angina  old_peak  slope  ca  thalassemia  target
298         1      123            1       0.2      1   0            3       0
299         1      132            0       1.2      1   0            3       0
300         1      141            0       3.4      1   2            3       0
301         1      115            1       1.2      1   1            3       0
302         0      174            0       0.0      1   1            2       0
```

[ ]:
```
# number of rows and columns in the dataset
df.shape
```

[ ]:
```
(303, 14)
```

[ ]:
```
# getting some info about the data
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 303 entries, 0 to 302
Data columns (total 14 columns):
 #   Column                Non-Null Count  Dtype  
---  ------                --------------  -----  
 0   age                   303 non-null    int64  
 1   gender                303 non-null    int64  
 2   chest_pain            303 non-null    int64  
 3   rest_bps              303 non-null    int64  
 4   cholestrol            303 non-null    int64  
 5   fasting_blood_sugar   303 non-null    int64  
 6   rest_ecg              303 non-null    int64  
 7   thalach               303 non-null    int64  
 8   exer_angina           303 non-null    int64  
 9   old_peak              303 non-null    float64
 10  slope                 303 non-null    int64  
 11  ca                    303 non-null    int64  
 12  thalassemia           303 non-null    int64  
 13  target                303 non-null    int64  
dtypes: float64(1), int64(13)
memory usage: 33.3 KB
```

[ ]:
```
# getting some info about the data
df.isnull().sum()
```

[ ]:
```
age                   0
gender                0
chest_pain            0
rest_bps              0
cholestrol            0
fasting_blood_sugar   0
rest_ecg              0
thalach               0
exer_angina           0
old_peak              0
slope                 0
ca                    0
thalassemia           0
target                0
```

```
dtype: int64
```

[ ]:
```
# statistical measures about the data
df.describe()
```

[ ]:
```
              age      gender  chest_pain    rest_bps  cholestrol  \
count  303.000000  303.000000  303.000000  303.000000  303.000000
mean    54.366337    0.683168    0.966997  131.623762  246.264026
std      9.082101    0.466011    1.032052   17.538143   51.830751
min     29.000000    0.000000    0.000000   94.000000  126.000000
25%     47.500000    0.000000    0.000000  120.000000  211.000000
50%     55.000000    1.000000    1.000000  130.000000  240.000000
75%     61.000000    1.000000    2.000000  140.000000  274.500000
max     77.000000    1.000000    3.000000  200.000000  564.000000

       fasting_blood_sugar    rest_ecg     thalach  exer_angina    old_peak  \
count           303.000000  303.000000  303.000000   303.000000  303.000000
mean              0.148515    0.528053  149.646865     0.326733    1.039604
std               0.356198    0.525860   22.905161     0.469794    1.161075
min               0.000000    0.000000   71.000000     0.000000    0.000000
25%               0.000000    0.000000  133.500000     0.000000    0.000000
50%               0.000000    1.000000  153.000000     0.000000    0.800000
75%               0.000000    1.000000  166.000000     1.000000    1.600000
max               1.000000    2.000000  202.000000     1.000000    6.200000

            slope          ca  thalassemia      target
count  303.000000  303.000000   303.000000  303.000000
mean     1.399340    0.729373     2.313531    0.544554
std      0.616226    1.022606     0.612277    0.498835
min      0.000000    0.000000     0.000000    0.000000
25%      1.000000    0.000000     2.000000    0.000000
50%      1.000000    0.000000     2.000000    1.000000
75%      2.000000    1.000000     3.000000    1.000000
max      2.000000    4.000000     3.000000    1.000000
```

[ ]:
```
# checking the distribution of Target Variable

df['target'].value_counts()
```

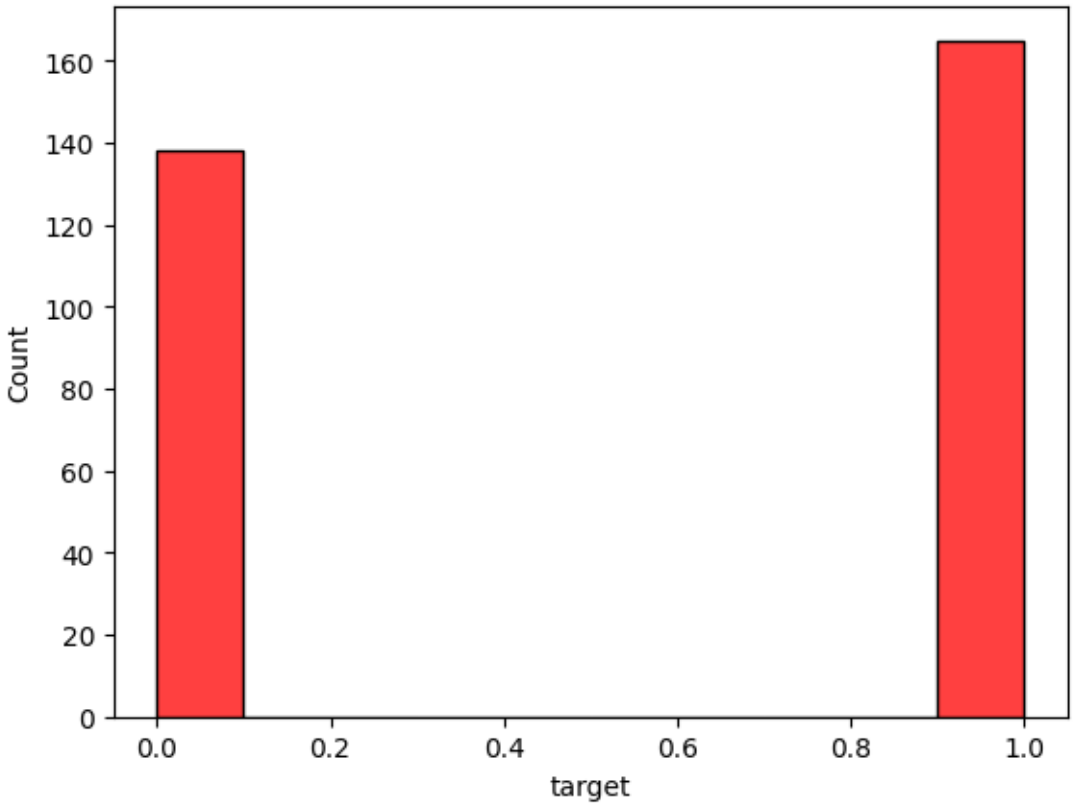
[ ]:
```
target
1    165
0    138
Name: count, dtype: int64
```

1 –> Defective Heart

0 –> Healthy Heart

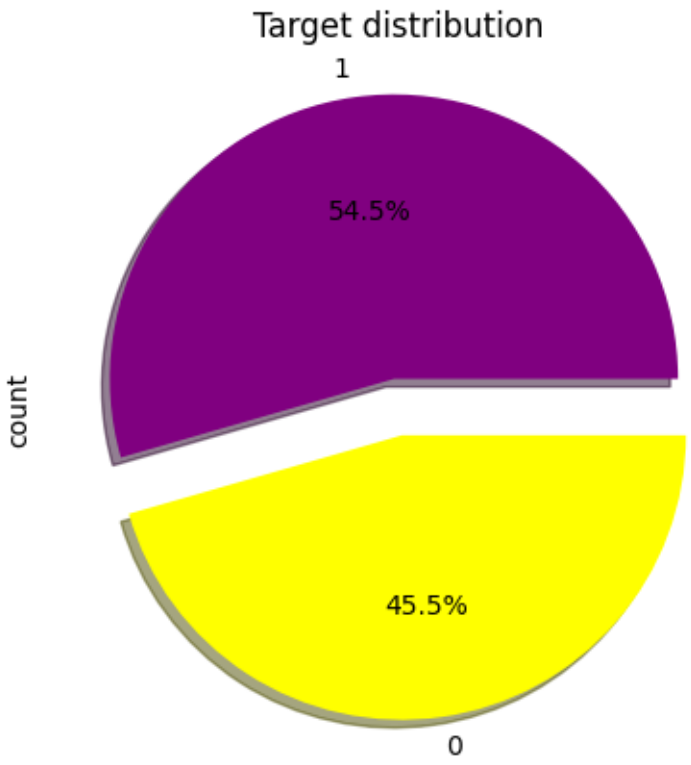
[ ]:
```
sns.histplot(df['target'],color='red')
```

[ ]:
```
<Axes: xlabel='target', ylabel='Count'>
```



[ ]:
```
df['target'].value_counts().plot.pie(explode=[0.1,0.1],autopct='%1.
↪1f%%',shadow=True, colors=['purple', 'yellow'],textprops = {'fontsize':10}).
↪set_title("Target distribution")
```

[ ]:
```
Text(0.5, 1.0, 'Target distribution')
```



[ ]:
```
# drop duplicates
df.drop_duplicates()
```

[ ]:

	age	gender	chest_pain	rest_bps	cholestrol	fasting_blood_sugar	\
0	63	1	3	145	233	1	
1	37	1	2	130	250	0	
2	41	0	1	130	204	0	
3	56	1	1	120	236	0	
4	57	0	0	120	354	0	
..	…	…	…	…	…	…	
298	57	0	0	140	241	0	
299	45	1	3	110	264	0	
300	68	1	0	144	193	1	
301	57	1	0	130	131	0	
302	57	0	1	130	236	0	

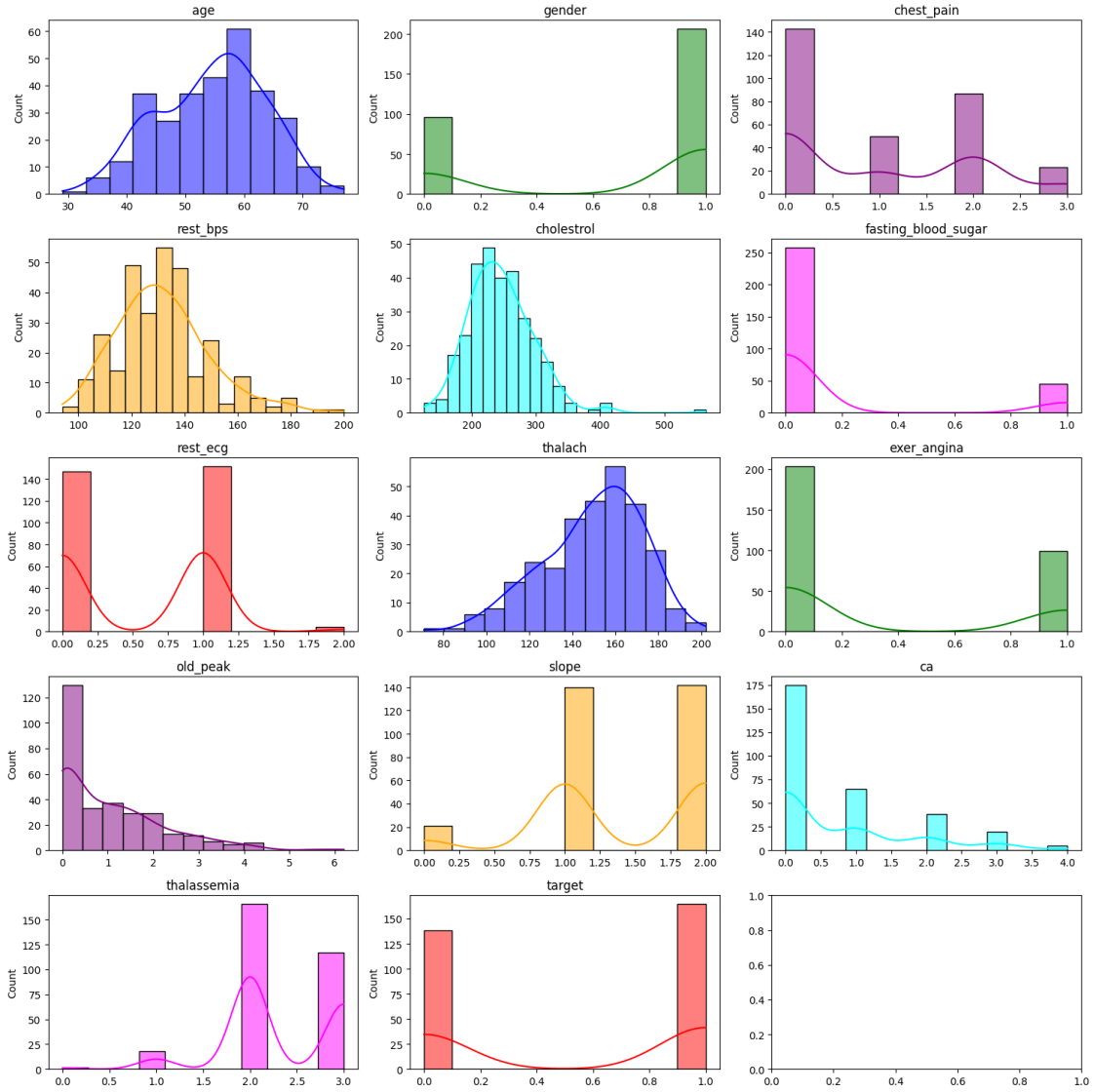
	rest_ecg	thalach	exer_angina	old_peak	slope	ca	thalassemia	target
0	0	150	0	2.3	0	0	1	1
1	1	187	0	3.5	0	0	2	1
2	0	172	0	1.4	2	0	2	1
3	1	178	0	0.8	2	0	2	1
4	1	163	1	0.6	2	0	2	1

```
..        …     …          …      …    …  ..      …     …
298         1     123          1      0.2     1   0          3      0
299         1     132          0      1.2     1   0          3      0
300         1     141          0      3.4     1   2          3      0
301         1     115          1      1.2     1   1          3      0
302         0     174          0      0.0     1   1          2      0

[302 rows x 14 columns]
```

[ ]:
```
df.columns
```

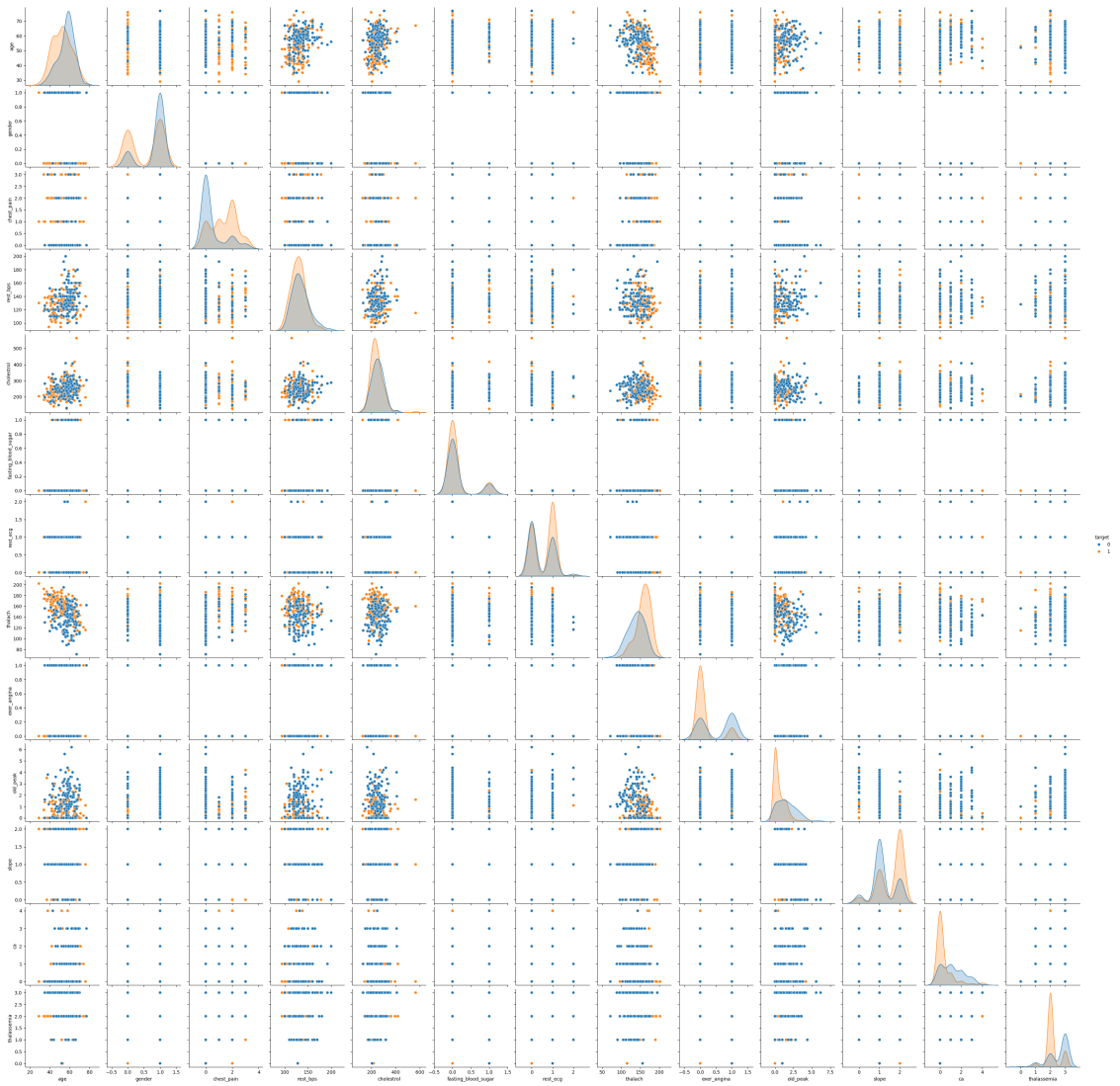
[ ]:
```
Index(['age', 'gender', 'chest_pain', 'rest_bps', 'cholestrol',
       'fasting_blood_sugar', 'rest_ecg', 'thalach', 'exer_angina', 'old_peak',
       'slope', 'ca', 'thalassemia', 'target'],
      dtype='object')
```

[ ]:
```
fig, axes = plt.subplots(5, 3, figsize=(15, 15))
axes = axes.flatten()
colors = ['blue', 'green', 'purple', 'orange', 'cyan', 'magenta', 'red']
for i, col in enumerate(df.columns):
    sns.histplot(df[col], ax=axes[i], kde=True, color=colors[i % len(colors)])
    axes[i].set_title(col)
    axes[i].set_xlabel('')

plt.tight_layout()
plt.show()
```



[ ]:
```
colors = ['blue', 'green', 'purple', 'orange', 'cyan', 'magenta', 'red']
sns.pairplot(df, hue='target') # Use 'hue' to specify the column to determine␣
↪colors
```

[ ]: <seaborn.axisgrid.PairGrid at 0x7dcdd3971c90>



[ ]:
```
# Heat Map
df.corr()
```

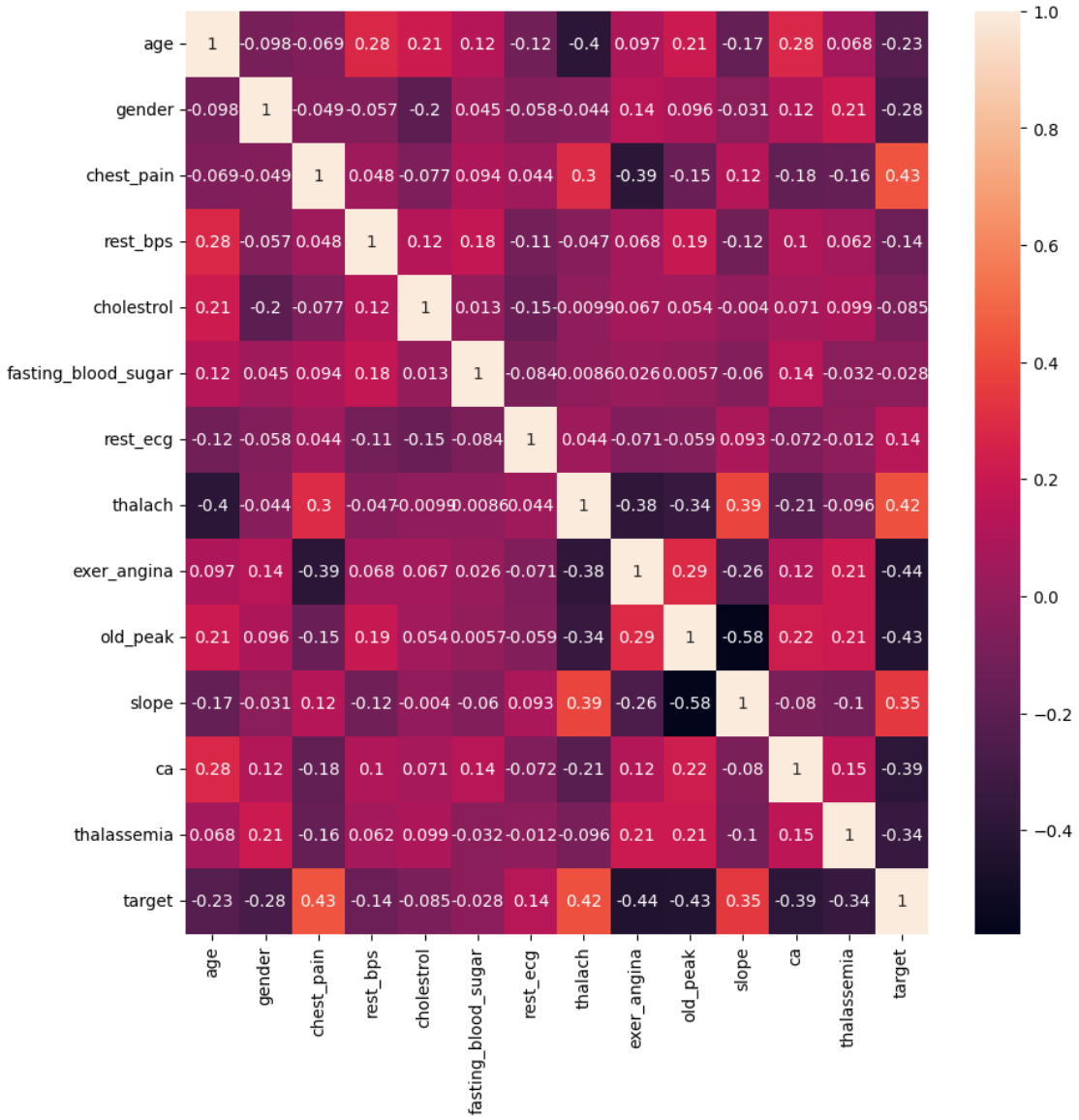
[ ]:

	age	gender	chest_pain	rest_bps	cholestrol	\
age	1.000000	-0.098447	-0.068653	0.279351	0.213678	
gender	-0.098447	1.000000	-0.049353	-0.056769	-0.197912	
chest_pain	-0.068653	-0.049353	1.000000	0.047608	-0.076904	
rest_bps	0.279351	-0.056769	0.047608	1.000000	0.123174	
cholestrol	0.213678	-0.197912	-0.076904	0.123174	1.000000	
fasting_blood_sugar	0.121308	0.045032	0.094444	0.177531	0.013294	
rest_ecg	-0.116211	-0.058196	0.044421	-0.114103	-0.151040	
thalach	-0.398522	-0.044020	0.295762	-0.046698	-0.009940	
exer_angina	0.096801	0.141664	-0.394280	0.067616	0.067023	
old_peak	0.210013	0.096093	-0.149230	0.193216	0.053952	

```
slope                -0.168814 -0.030711    0.119717 -0.121475   -0.004038
ca                    0.276326  0.118261   -0.181053  0.101389    0.070511
thalassemia           0.068001  0.210041   -0.161736  0.062210    0.098803
target               -0.225439 -0.280937    0.433798 -0.144931   -0.085239

                     fasting_blood_sugar  rest_ecg   thalach  exer_angina  \
age                             0.121308 -0.116211 -0.398522     0.096801
gender                          0.045032 -0.058196 -0.044020     0.141664
chest_pain                      0.094444  0.044421  0.295762    -0.394280
rest_bps                        0.177531 -0.114103 -0.046698     0.067616
cholestrol                      0.013294 -0.151040 -0.009940     0.067023
fasting_blood_sugar             1.000000 -0.084189 -0.008567     0.025665
rest_ecg                       -0.084189  1.000000  0.044123    -0.070733
thalach                        -0.008567  0.044123  1.000000    -0.378812
exer_angina                     0.025665 -0.070733 -0.378812     1.000000
old_peak                        0.005747 -0.058770 -0.344187     0.288223
slope                          -0.059894  0.093045  0.386784    -0.257748
ca                              0.137979 -0.072042 -0.213177     0.115739
thalassemia                    -0.032019 -0.011981 -0.096439     0.206754
target                         -0.028046  0.137230  0.421741    -0.436757

                     old_peak     slope        ca  thalassemia    target
age                  0.210013 -0.168814  0.276326     0.068001 -0.225439
gender               0.096093 -0.030711  0.118261     0.210041 -0.280937
chest_pain          -0.149230  0.119717 -0.181053    -0.161736  0.433798
rest_bps             0.193216 -0.121475  0.101389     0.062210 -0.144931
cholestrol           0.053952 -0.004038  0.070511     0.098803 -0.085239
fasting_blood_sugar  0.005747 -0.059894  0.137979    -0.032019 -0.028046
rest_ecg            -0.058770  0.093045 -0.072042    -0.011981  0.137230
thalach             -0.344187  0.386784 -0.213177    -0.096439  0.421741
exer_angina          0.288223 -0.257748  0.115739     0.206754 -0.436757
old_peak             1.000000 -0.577537  0.222682     0.210244 -0.430696
slope               -0.577537  1.000000 -0.080155    -0.104764  0.345877
ca                   0.222682 -0.080155  1.000000     0.151832 -0.391724
thalassemia          0.210244 -0.104764  0.151832     1.000000 -0.344029
target              -0.430696  0.345877 -0.391724    -0.344029  1.000000
```

[ ]:
```
plt.figure(figsize=(10,10))
sns.heatmap(df.corr(),annot=True)
```

[ ]:
```
<Axes: >
```



9 machine Model Training

[ ]:
```
x = df.drop(['target'],axis=1)
y=df['target']
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.
  ↪2,random_state=42)
x_train,x_test,y_train,y_test
```

[ ]:
```
(     age  gender  chest_pain  rest_bps  cholestrol  fasting_blood_sugar  \
 132   42       1           1       120         295                    0
```

```
202   58       1           0       150         270                    0
196   46       1           2       150         231                    0
75    55       0           1       135         250                    0
176   60       1           0       117         230                    1
..   …     …         …       …         …                  …
188   50       1           2       140         233                    0
71    51       1           2        94         227                    0
106   69       1           3       160         234                    1
270   46       1           0       120         249                    0
102   63       0           1       140         195                    0

     rest_ecg  thalach  exer_angina  old_peak  slope  ca  thalassemia
132         1      162            0       0.0      2   0            2
202         0      111            1       0.8      2   0            3
196         1      147            0       3.6      1   0            2
75          0      161            0       1.4      1   0            2
176         1      160            1       1.4      2   2            3
..        …      …          …       …    …  ..          …
188         1      163            0       0.6      1   1            3
71          1      154            1       0.0      2   1            3
106         0      131            0       0.1      1   1            2
270         0      144            0       0.8      2   0            3
102         1      179            0       0.0      2   2            2

[242 rows x 13 columns],
     age  gender  chest_pain  rest_bps  cholestrol  fasting_blood_sugar  \
179   57       1           0       150         276                    0
228   59       1           3       170         288                    0
111   57       1           2       150         126                    1
246   56       0           0       134         409                    0
60    71       0           2       110         265                    1
..   …     …         …       …         …                  …
249   69       1           2       140         254                    0
104   50       1           2       129         196                    0
300   68       1           0       144         193                    1
193   60       1           0       145         282                    0
184   50       1           0       150         243                    0

     rest_ecg  thalach  exer_angina  old_peak  slope  ca  thalassemia
179         0      112            1       0.6      1   1            1
228         0      159            0       0.2      1   0            3
111         1      173            0       0.2      2   1            3
246         0      150            1       1.9      1   2            3
60          0      130            0       0.0      2   1            2
..        …      …          …       …    …  ..          …
249         0      146            0       2.0      1   3            3
104         1      163            0       0.0      2   0            2
```

```
300         1      141            0       3.4      1   2            3
193         0      142            1       2.8      1   2            3
184         0      128            0       2.6      1   0            3

[61 rows x 13 columns],
132    1
202    0
196    0
75     1
176    0
      ..
188    0
71     1
106    1
270    0
102    1
Name: target, Length: 242, dtype: int64,
179    0
228    0
111    1
246    0
60     1
      ..
249    0
104    1
300    0
193    0
184    0
Name: target, Length: 61, dtype: int64)
```

10 Logisitic Regression

[ ]:
```
from sklearn.linear_model import LogisticRegression # Import LogisticRegression

from sklearn.model_selection import train_test_split
```

[ ]:
```
model  = LogisticRegression()
model.fit(x_train,y_train)
y_pred = model.predict(x_test)
accuracy=accuracy_score(y_test,y_pred)
cfm=confusion_matrix(y_test,y_pred)
classreport=classification_report(y_test,y_pred)
recall = recall_score(y_test, y_pred)
precision = precision_score(y_test, y_pred)
print("Accuracy",accuracy)
print("confusion matrix",cfm)
```

```
print("Precision :",precision)
print("Recall :",recall)
print("classification report",classreport)
print("Training score:",model.score(x_train,y_train))
print('Testing score:',model.score(x_test,y_test))
print("Kappa score:",cohen_kappa_score(y_test, y_pred))
```

```
Accuracy 0.8852459016393442
confusion matrix [[25  4]
 [ 3 29]]
Precision : 0.8787878787878788
Recall : 0.90625
classification report               precision    recall  f1-score   support

           0       0.89      0.86      0.88        29
           1       0.88      0.91      0.89        32

    accuracy                           0.89        61
   macro avg       0.89      0.88      0.88        61
weighted avg       0.89      0.89      0.89        61

Training score: 0.8553719008264463
Testing score: 0.8852459016393442
Kappa score: 0.7695628710199676
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_logistic.py:458:
ConvergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-
regression
  n_iter_i = _check_optimize_result(
```

11 Decision Tree

[ ]:
```
model1=DecisionTreeClassifier()
model1.fit(x_train,y_train)
y_pred1=model1.predict(x_test)
accuracy1=accuracy_score(y_test,y_pred1)
cfm1=confusion_matrix(y_test,y_pred1)
classreport1=classification_report(y_test,y_pred1)
recall1 = recall_score(y_test, y_pred1)
precision1 = precision_score(y_test, y_pred1)
print("Accuracy",accuracy1)
```

```
print("confusion matrix",cfm1)
print("Precision :",precision1)
print("Recall :",recall1)
print("classification report",classreport1)
print("Training score:",model1.score(x_train,y_train))
print('Testing score:',model1.score(x_test,y_test))
print("Kappa score:",cohen_kappa_score(y_test, y_pred1))
```

```
Accuracy 0.8360655737704918
confusion matrix [[26  3]
 [ 7 25]]
Precision : 0.8928571428571429
Recall : 0.78125
classification report               precision    recall  f1-score   support

           0       0.79      0.90      0.84        29
           1       0.89      0.78      0.83        32

    accuracy                           0.84        61
   macro avg       0.84      0.84      0.84        61
weighted avg       0.84      0.84      0.84        61

Training score: 1.0
Testing score: 0.8360655737704918
Kappa score: 0.6734475374732334
```

12 Random Forest

[ ]:
```
model2=RandomForestClassifier()
model2.fit(x_train,y_train)
y_pred2=model2.predict(x_test)
accuracy2=accuracy_score(y_test,y_pred2)
cfm2=confusion_matrix(y_test,y_pred2)
classreport2=classification_report(y_test,y_pred2)
recall2 = recall_score(y_test, y_pred2)
precision2 = precision_score(y_test, y_pred2)
print("Accuracy",accuracy2)
print("confusion matrix",cfm2)
print("Precision :",precision2)
print("Recall :",recall2)
print("classification report",classreport2)
print("Training score:",model2.score(x_train,y_train))
print('Testing score:',model1.score(x_test,y_test))
print("Kappa score:",cohen_kappa_score(y_test, y_pred2))
```

```
Accuracy 0.8688524590163934
confusion matrix [[24  5]
```

```
 [ 3 29]]
Precision : 0.8529411764705882
Recall : 0.90625
classification report               precision    recall  f1-score   support

           0       0.89      0.83      0.86        29
           1       0.85      0.91      0.88        32

    accuracy                           0.87        61
   macro avg       0.87      0.87      0.87        61
weighted avg       0.87      0.87      0.87        61

Training score: 1.0
Testing score: 0.8360655737704918
Kappa score: 0.7362162162162162
```

13 Gradient Boosting Classifier

[ ]:
```
from sklearn.ensemble import GradientBoostingClassifier
model3=GradientBoostingClassifier()
model3.fit(x_train,y_train)
y_pred3=model3.predict(x_test)
accuracy3=accuracy_score(y_test,y_pred3)
cfm3=confusion_matrix(y_test,y_pred3)
classreport3=classification_report(y_test,y_pred3)
recall3 = recall_score(y_test, y_pred3)
precision3 = precision_score(y_test, y_pred3)
print("Accuracy",accuracy3)
print("confusion matrix",cfm3)
print("Precision :",precision3)
print("Recall :",recall3)
print("classification report",classreport3)
print("Training score:",model3.score(x_train,y_train))
print('Testing score:',model3.score(x_test,y_test))
print("Kappa score:",cohen_kappa_score(y_test, y_pred3))
```

```
Accuracy 0.7704918032786885
confusion matrix [[23  6]
 [ 8 24]]
Precision : 0.8
Recall : 0.75
classification report               precision    recall  f1-score   support

           0       0.74      0.79      0.77        29
           1       0.80      0.75      0.77        32

    accuracy                           0.77        61
```

```
   macro avg       0.77      0.77      0.77        61
weighted avg       0.77      0.77      0.77        61

Training score: 1.0
Testing score: 0.7704918032786885
Kappa score: 0.5413533834586466
```

14 AdaBoostClassifier

[ ]:
```
model4 = GradientBoostingClassifier(learning_rate=0.1, n_estimators=500,␣
 ↪subsample=0.5, random_state=10)
model4.fit(x_train, y_train)
y_pred4 = model4.predict(x_test)
accuracy4 = accuracy_score(y_test, y_pred4)
cfm4 = confusion_matrix(y_test, y_pred4)
classreport4 = classification_report(y_test, y_pred4)
recall4 = recall_score(y_test, y_pred4)
precision4 = precision_score(y_test, y_pred4)
print("Accuracy", accuracy4)
print("confusion matrix", cfm4)
print("Precision :", precision4)
print("Recall :", recall4)
print("classification report", classreport4)
print("Training score:", model4.score(x_train, y_train))
print('Testing score:', model4.score(x_test, y_test))
print("Kappa score:", cohen_kappa_score(y_test, y_pred4))
```

```
Accuracy 0.8524590163934426
confusion matrix [[25  4]
 [ 5 27]]
Precision : 0.8709677419354839
Recall : 0.84375
classification report               precision    recall  f1-score   support

           0       0.83      0.86      0.85        29
           1       0.87      0.84      0.86        32

    accuracy                           0.85        61
   macro avg       0.85      0.85      0.85        61
weighted avg       0.85      0.85      0.85        61

Training score: 1.0
Testing score: 0.8524590163934426
Kappa score: 0.7046799354491662
```

15 xgb.XGBClassifier

```
[ ]: model5 = xgb.XGBClassifier(learning_rate=0.1, n_estimators=500, subsample=0.5,␣
      ↪random_state=10)
     model5.fit(x_train, y_train)
     y_pred5 = model5.predict(x_test)
     accuracy5 = accuracy_score(y_test, y_pred5)
     cfm5 = confusion_matrix(y_test, y_pred5)
     classreport5 = classification_report(y_test, y_pred5)
     recall5 = recall_score(y_test, y_pred5)
     precision5 = precision_score(y_test, y_pred5)
     print("Accuracy", accuracy5)
     print("confusion matrix", cfm5)
     print("Precision :", precision5)
     print("Recall :", recall5)
     print("classification report", classreport5)
     print("Training score:", model5.score(x_train, y_train))
     print('Testing score:', model5.score(x_test, y_test))
     print("Kappa score:", cohen_kappa_score(y_test, y_pred5))
```

```
Accuracy 0.8524590163934426
confusion matrix [[25  4]
 [ 5 27]]
Precision : 0.8709677419354839
Recall : 0.84375
classification report               precision    recall  f1-score   support

           0       0.83      0.86      0.85        29
           1       0.87      0.84      0.86        32

    accuracy                           0.85        61
   macro avg       0.85      0.85      0.85        61
weighted avg       0.85      0.85      0.85        61

Training score: 1.0
Testing score: 0.8524590163934426
Kappa score: 0.7046799354491662
```

16 catboost Classifier

```
[ ]: !pip install catboost
     from catboost import CatBoostClassifier
```

```
Requirement already satisfied: catboost in /usr/local/lib/python3.10/dist-
packages (1.2.5)
Requirement already satisfied: graphviz in /usr/local/lib/python3.10/dist-
packages (from catboost) (0.20.3)
```

```
Requirement already satisfied: matplotlib in /usr/local/lib/python3.10/dist-
packages (from catboost) (3.7.1)
Requirement already satisfied: numpy>=1.16.0 in /usr/local/lib/python3.10/dist-
packages (from catboost) (1.25.2)
Requirement already satisfied: pandas>=0.24 in /usr/local/lib/python3.10/dist-
packages (from catboost) (2.0.3)
Requirement already satisfied: scipy in /usr/local/lib/python3.10/dist-packages
(from catboost) (1.11.4)
Requirement already satisfied: plotly in /usr/local/lib/python3.10/dist-packages
(from catboost) (5.15.0)
Requirement already satisfied: six in /usr/local/lib/python3.10/dist-packages
(from catboost) (1.16.0)
Requirement already satisfied: python-dateutil>=2.8.2 in
/usr/local/lib/python3.10/dist-packages (from pandas>=0.24->catboost) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-
packages (from pandas>=0.24->catboost) (2023.4)
Requirement already satisfied: tzdata>=2022.1 in /usr/local/lib/python3.10/dist-
packages (from pandas>=0.24->catboost) (2024.1)
Requirement already satisfied: contourpy>=1.0.1 in
/usr/local/lib/python3.10/dist-packages (from matplotlib->catboost) (1.2.1)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.10/dist-
packages (from matplotlib->catboost) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in
/usr/local/lib/python3.10/dist-packages (from matplotlib->catboost) (4.53.1)
Requirement already satisfied: kiwisolver>=1.0.1 in
/usr/local/lib/python3.10/dist-packages (from matplotlib->catboost) (1.4.5)
Requirement already satisfied: packaging>=20.0 in
/usr/local/lib/python3.10/dist-packages (from matplotlib->catboost) (24.1)
Requirement already satisfied: pillow>=6.2.0 in /usr/local/lib/python3.10/dist-
packages (from matplotlib->catboost) (9.4.0)
Requirement already satisfied: pyparsing>=2.3.1 in
/usr/local/lib/python3.10/dist-packages (from matplotlib->catboost) (3.1.2)
Requirement already satisfied: tenacity>=6.2.0 in
/usr/local/lib/python3.10/dist-packages (from plotly->catboost) (8.5.0)
```

[ ]:
```
model6 = CatBoostClassifier(learning_rate=0.1, n_estimators=500, subsample=0.5,␣
  ↪random_state=10)
model6.fit(x_train, y_train)
y_pred6 = model6.predict(x_test)
accuracy6 = accuracy_score(y_test, y_pred6)
cfm6 = confusion_matrix(y_test, y_pred6)
classreport6 = classification_report(y_test, y_pred6)
recall6 = recall_score(y_test, y_pred6)
precision6 = precision_score(y_test, y_pred6)
print("Accuracy", accuracy6)
print("confusion matrix", cfm6)
print("Precision :", precision6)
```

```
print("Recall :", recall6)
print("classification report", classreport6)
print("Training score:", model6.score(x_train, y_train))
print('Testing score:', model6.score(x_test, y_test))
print("Kappa score:", cohen_kappa_score(y_test, y_pred6))
```

```
0:      learn: 0.6292063        total: 6.82ms   remaining: 3.4s
1:      learn: 0.5679420        total: 8.48ms   remaining: 2.11s
2:      learn: 0.5229161        total: 9.81ms   remaining: 1.62s
3:      learn: 0.4789023        total: 11.1ms   remaining: 1.37s
4:      learn: 0.4425827        total: 12.3ms   remaining: 1.22s
5:      learn: 0.4155897        total: 13.5ms   remaining: 1.11s
6:      learn: 0.3964624        total: 14.7ms   remaining: 1.04s
7:      learn: 0.3790341        total: 15.9ms   remaining: 978ms
8:      learn: 0.3510948        total: 17.1ms   remaining: 930ms
9:      learn: 0.3377328        total: 18.2ms   remaining: 892ms
10:     learn: 0.3220351        total: 19.4ms   remaining: 861ms
11:     learn: 0.3059821        total: 20.6ms   remaining: 838ms
12:     learn: 0.2903827        total: 21.8ms   remaining: 815ms
13:     learn: 0.2838766        total: 22.9ms   remaining: 796ms
14:     learn: 0.2754118        total: 24.1ms   remaining: 779ms
15:     learn: 0.2649277        total: 25.3ms   remaining: 765ms
16:     learn: 0.2581780        total: 26.6ms   remaining: 755ms
17:     learn: 0.2488869        total: 27.7ms   remaining: 743ms
18:     learn: 0.2414497        total: 28.9ms   remaining: 732ms
19:     learn: 0.2324046        total: 30.1ms   remaining: 722ms
20:     learn: 0.2305497        total: 31ms     remaining: 708ms
21:     learn: 0.2229945        total: 32.2ms   remaining: 699ms
22:     learn: 0.2201897        total: 33.4ms   remaining: 692ms
23:     learn: 0.2166929        total: 34.5ms   remaining: 685ms
24:     learn: 0.2114445        total: 35.7ms   remaining: 678ms
25:     learn: 0.2074186        total: 36.8ms   remaining: 671ms
26:     learn: 0.2002667        total: 37.9ms   remaining: 665ms
27:     learn: 0.1948818        total: 39.2ms   remaining: 661ms
28:     learn: 0.1931387        total: 40.1ms   remaining: 652ms
29:     learn: 0.1897620        total: 41.3ms   remaining: 647ms
30:     learn: 0.1859346        total: 42.5ms   remaining: 643ms
31:     learn: 0.1815277        total: 43.6ms   remaining: 638ms
32:     learn: 0.1806296        total: 44.6ms   remaining: 631ms
33:     learn: 0.1783546        total: 45.8ms   remaining: 627ms
34:     learn: 0.1734368        total: 46.9ms   remaining: 624ms
35:     learn: 0.1709913        total: 48.3ms   remaining: 622ms
36:     learn: 0.1687719        total: 49.5ms   remaining: 619ms
37:     learn: 0.1642614        total: 50.8ms   remaining: 618ms
38:     learn: 0.1595910        total: 52.5ms   remaining: 620ms
39:     learn: 0.1569203        total: 53.8ms   remaining: 619ms
40:     learn: 0.1538807        total: 55ms     remaining: 616ms
```

```
41:     learn: 0.1501174        total: 56.2ms   remaining: 613ms
42:     learn: 0.1484172        total: 57.4ms   remaining: 610ms
43:     learn: 0.1467974        total: 58.6ms   remaining: 607ms
44:     learn: 0.1465546        total: 59.4ms   remaining: 601ms
45:     learn: 0.1430683        total: 60.6ms   remaining: 598ms
46:     learn: 0.1380652        total: 61.7ms   remaining: 595ms
47:     learn: 0.1359240        total: 65.3ms   remaining: 615ms
48:     learn: 0.1330689        total: 66.5ms   remaining: 612ms
49:     learn: 0.1298328        total: 67.6ms   remaining: 609ms
50:     learn: 0.1273504        total: 68.9ms   remaining: 606ms
51:     learn: 0.1251747        total: 70ms     remaining: 604ms
52:     learn: 0.1220147        total: 71.3ms   remaining: 601ms
53:     learn: 0.1177906        total: 72.5ms   remaining: 599ms
54:     learn: 0.1163448        total: 73.7ms   remaining: 596ms
55:     learn: 0.1152744        total: 74.9ms   remaining: 594ms
56:     learn: 0.1136831        total: 76.1ms   remaining: 592ms
57:     learn: 0.1124541        total: 77.3ms   remaining: 589ms
58:     learn: 0.1107799        total: 78.5ms   remaining: 587ms
59:     learn: 0.1069459        total: 79.7ms   remaining: 585ms
60:     learn: 0.1053552        total: 80.9ms   remaining: 582ms
61:     learn: 0.1040590        total: 82.1ms   remaining: 580ms
62:     learn: 0.1027810        total: 83.3ms   remaining: 578ms
63:     learn: 0.1014696        total: 84.5ms   remaining: 575ms
64:     learn: 0.0989862        total: 85.7ms   remaining: 573ms
65:     learn: 0.0978290        total: 86.9ms   remaining: 571ms
66:     learn: 0.0951944        total: 88ms     remaining: 569ms
67:     learn: 0.0934224        total: 89.3ms   remaining: 567ms
68:     learn: 0.0925301        total: 90.5ms   remaining: 565ms
69:     learn: 0.0907169        total: 91.7ms   remaining: 563ms
70:     learn: 0.0901062        total: 92.9ms   remaining: 561ms
71:     learn: 0.0880383        total: 94.1ms   remaining: 559ms
72:     learn: 0.0864376        total: 95.2ms   remaining: 557ms
73:     learn: 0.0837871        total: 96.4ms   remaining: 555ms
74:     learn: 0.0824669        total: 97.6ms   remaining: 553ms
75:     learn: 0.0812216        total: 98.7ms   remaining: 551ms
76:     learn: 0.0792308        total: 99.9ms   remaining: 549ms
77:     learn: 0.0785896        total: 101ms    remaining: 547ms
78:     learn: 0.0773696        total: 102ms    remaining: 545ms
79:     learn: 0.0757840        total: 103ms    remaining: 543ms
80:     learn: 0.0739815        total: 105ms    remaining: 542ms
81:     learn: 0.0732788        total: 106ms    remaining: 540ms
82:     learn: 0.0711753        total: 107ms    remaining: 538ms
83:     learn: 0.0691156        total: 108ms    remaining: 537ms
84:     learn: 0.0685245        total: 110ms    remaining: 536ms
85:     learn: 0.0677568        total: 111ms    remaining: 535ms
86:     learn: 0.0661474        total: 112ms    remaining: 533ms
87:     learn: 0.0648349        total: 114ms    remaining: 532ms
88:     learn: 0.0633079        total: 115ms    remaining: 530ms
```

```
89:     learn: 0.0623557        total: 116ms    remaining: 529ms
90:     learn: 0.0608173        total: 117ms    remaining: 527ms
91:     learn: 0.0600331        total: 118ms    remaining: 525ms
92:     learn: 0.0585071        total: 120ms    remaining: 524ms
93:     learn: 0.0575150        total: 121ms    remaining: 522ms
94:     learn: 0.0562049        total: 122ms    remaining: 520ms
95:     learn: 0.0551645        total: 123ms    remaining: 518ms
96:     learn: 0.0538207        total: 124ms    remaining: 517ms
97:     learn: 0.0526105        total: 126ms    remaining: 515ms
98:     learn: 0.0518475        total: 127ms    remaining: 513ms
99:     learn: 0.0511939        total: 128ms    remaining: 512ms
100:    learn: 0.0507426        total: 129ms    remaining: 510ms
101:    learn: 0.0498227        total: 130ms    remaining: 509ms
102:    learn: 0.0492733        total: 132ms    remaining: 508ms
103:    learn: 0.0483383        total: 133ms    remaining: 507ms
104:    learn: 0.0481294        total: 134ms    remaining: 505ms
105:    learn: 0.0474603        total: 136ms    remaining: 504ms
106:    learn: 0.0468816        total: 137ms    remaining: 502ms
107:    learn: 0.0461520        total: 138ms    remaining: 501ms
108:    learn: 0.0453477        total: 139ms    remaining: 499ms
109:    learn: 0.0449262        total: 141ms    remaining: 500ms
110:    learn: 0.0441868        total: 142ms    remaining: 498ms
111:    learn: 0.0434718        total: 143ms    remaining: 497ms
112:    learn: 0.0428716        total: 145ms    remaining: 495ms
113:    learn: 0.0421833        total: 146ms    remaining: 494ms
114:    learn: 0.0415716        total: 147ms    remaining: 492ms
115:    learn: 0.0406529        total: 148ms    remaining: 491ms
116:    learn: 0.0397798        total: 149ms    remaining: 489ms
117:    learn: 0.0392240        total: 151ms    remaining: 488ms
118:    learn: 0.0385774        total: 152ms    remaining: 487ms
119:    learn: 0.0377438        total: 153ms    remaining: 485ms
120:    learn: 0.0374195        total: 154ms    remaining: 484ms
121:    learn: 0.0367815        total: 156ms    remaining: 482ms
122:    learn: 0.0363976        total: 157ms    remaining: 481ms
123:    learn: 0.0359189        total: 158ms    remaining: 480ms
124:    learn: 0.0352608        total: 160ms    remaining: 479ms
125:    learn: 0.0346792        total: 161ms    remaining: 478ms
126:    learn: 0.0342402        total: 162ms    remaining: 476ms
127:    learn: 0.0338694        total: 163ms    remaining: 475ms
128:    learn: 0.0334086        total: 165ms    remaining: 474ms
129:    learn: 0.0327496        total: 166ms    remaining: 473ms
130:    learn: 0.0323794        total: 167ms    remaining: 472ms
131:    learn: 0.0320483        total: 169ms    remaining: 470ms
132:    learn: 0.0316304        total: 170ms    remaining: 469ms
133:    learn: 0.0312800        total: 171ms    remaining: 468ms
134:    learn: 0.0308872        total: 172ms    remaining: 466ms
135:    learn: 0.0306453        total: 174ms    remaining: 465ms
136:    learn: 0.0302320        total: 175ms    remaining: 464ms
```

```
137:    learn: 0.0298931        total: 176ms    remaining: 462ms
138:    learn: 0.0296966        total: 177ms    remaining: 461ms
139:    learn: 0.0293135        total: 179ms    remaining: 459ms
140:    learn: 0.0289384        total: 180ms    remaining: 458ms
141:    learn: 0.0286006        total: 181ms    remaining: 456ms
142:    learn: 0.0284436        total: 182ms    remaining: 455ms
143:    learn: 0.0280518        total: 183ms    remaining: 453ms
144:    learn: 0.0274094        total: 185ms    remaining: 452ms
145:    learn: 0.0270179        total: 186ms    remaining: 451ms
146:    learn: 0.0265895        total: 187ms    remaining: 450ms
147:    learn: 0.0263911        total: 188ms    remaining: 448ms
148:    learn: 0.0262100        total: 192ms    remaining: 453ms
149:    learn: 0.0259933        total: 193ms    remaining: 451ms
150:    learn: 0.0256563        total: 194ms    remaining: 449ms
151:    learn: 0.0253189        total: 195ms    remaining: 448ms
152:    learn: 0.0251592        total: 197ms    remaining: 446ms
153:    learn: 0.0247294        total: 198ms    remaining: 444ms
154:    learn: 0.0244745        total: 199ms    remaining: 443ms
155:    learn: 0.0241707        total: 200ms    remaining: 441ms
156:    learn: 0.0238929        total: 201ms    remaining: 440ms
157:    learn: 0.0236660        total: 202ms    remaining: 438ms
158:    learn: 0.0234165        total: 203ms    remaining: 436ms
159:    learn: 0.0231005        total: 205ms    remaining: 435ms
160:    learn: 0.0228949        total: 206ms    remaining: 433ms
161:    learn: 0.0225024        total: 207ms    remaining: 432ms
162:    learn: 0.0222196        total: 208ms    remaining: 430ms
163:    learn: 0.0220099        total: 209ms    remaining: 429ms
164:    learn: 0.0217577        total: 210ms    remaining: 427ms
165:    learn: 0.0215956        total: 212ms    remaining: 426ms
166:    learn: 0.0213555        total: 213ms    remaining: 424ms
167:    learn: 0.0211707        total: 214ms    remaining: 423ms
168:    learn: 0.0210618        total: 215ms    remaining: 421ms
169:    learn: 0.0209376        total: 216ms    remaining: 420ms
170:    learn: 0.0206507        total: 217ms    remaining: 418ms
171:    learn: 0.0204182        total: 219ms    remaining: 417ms
172:    learn: 0.0202541        total: 220ms    remaining: 415ms
173:    learn: 0.0200859        total: 221ms    remaining: 414ms
174:    learn: 0.0199602        total: 222ms    remaining: 412ms
175:    learn: 0.0197808        total: 223ms    remaining: 411ms
176:    learn: 0.0195337        total: 224ms    remaining: 409ms
177:    learn: 0.0193590        total: 225ms    remaining: 408ms
178:    learn: 0.0191957        total: 227ms    remaining: 406ms
179:    learn: 0.0190627        total: 228ms    remaining: 405ms
180:    learn: 0.0187422        total: 229ms    remaining: 403ms
181:    learn: 0.0185163        total: 230ms    remaining: 402ms
182:    learn: 0.0183968        total: 231ms    remaining: 400ms
183:    learn: 0.0183159        total: 232ms    remaining: 399ms
184:    learn: 0.0181964        total: 234ms    remaining: 398ms
```

```
185:    learn: 0.0179839        total: 235ms    remaining: 397ms
186:    learn: 0.0178675        total: 236ms    remaining: 395ms
187:    learn: 0.0176860        total: 237ms    remaining: 394ms
188:    learn: 0.0175389        total: 239ms    remaining: 393ms
189:    learn: 0.0174257        total: 240ms    remaining: 391ms
190:    learn: 0.0172638        total: 241ms    remaining: 390ms
191:    learn: 0.0171283        total: 242ms    remaining: 388ms
192:    learn: 0.0169544        total: 243ms    remaining: 387ms
193:    learn: 0.0167925        total: 245ms    remaining: 386ms
194:    learn: 0.0166495        total: 246ms    remaining: 384ms
195:    learn: 0.0165292        total: 247ms    remaining: 383ms
196:    learn: 0.0162960        total: 248ms    remaining: 382ms
197:    learn: 0.0161920        total: 249ms    remaining: 380ms
198:    learn: 0.0160504        total: 251ms    remaining: 379ms
199:    learn: 0.0158909        total: 252ms    remaining: 378ms
200:    learn: 0.0157415        total: 253ms    remaining: 376ms
201:    learn: 0.0155974        total: 254ms    remaining: 375ms
202:    learn: 0.0154511        total: 255ms    remaining: 373ms
203:    learn: 0.0153990        total: 256ms    remaining: 372ms
204:    learn: 0.0152898        total: 258ms    remaining: 371ms
205:    learn: 0.0151571        total: 259ms    remaining: 369ms
206:    learn: 0.0150132        total: 260ms    remaining: 368ms
207:    learn: 0.0149211        total: 261ms    remaining: 366ms
208:    learn: 0.0147254        total: 262ms    remaining: 365ms
209:    learn: 0.0146568        total: 263ms    remaining: 364ms
210:    learn: 0.0145384        total: 265ms    remaining: 363ms
211:    learn: 0.0143984        total: 266ms    remaining: 361ms
212:    learn: 0.0142967        total: 267ms    remaining: 360ms
213:    learn: 0.0141800        total: 268ms    remaining: 359ms
214:    learn: 0.0141126        total: 270ms    remaining: 357ms
215:    learn: 0.0139822        total: 271ms    remaining: 356ms
216:    learn: 0.0138722        total: 272ms    remaining: 355ms
217:    learn: 0.0137362        total: 274ms    remaining: 354ms
218:    learn: 0.0136456        total: 275ms    remaining: 353ms
219:    learn: 0.0135550        total: 276ms    remaining: 351ms
220:    learn: 0.0134588        total: 277ms    remaining: 350ms
221:    learn: 0.0133646        total: 278ms    remaining: 348ms
222:    learn: 0.0132445        total: 279ms    remaining: 347ms
223:    learn: 0.0131499        total: 281ms    remaining: 346ms
224:    learn: 0.0130770        total: 282ms    remaining: 345ms
225:    learn: 0.0129934        total: 283ms    remaining: 344ms
226:    learn: 0.0129063        total: 285ms    remaining: 342ms
227:    learn: 0.0128078        total: 286ms    remaining: 341ms
228:    learn: 0.0127558        total: 287ms    remaining: 340ms
229:    learn: 0.0126845        total: 288ms    remaining: 338ms
230:    learn: 0.0125976        total: 289ms    remaining: 337ms
231:    learn: 0.0125451        total: 290ms    remaining: 335ms
232:    learn: 0.0124309        total: 292ms    remaining: 334ms
```

```
233:    learn: 0.0123949        total: 293ms    remaining: 333ms
234:    learn: 0.0123164        total: 294ms    remaining: 331ms
235:    learn: 0.0122205        total: 295ms    remaining: 330ms
236:    learn: 0.0121225        total: 296ms    remaining: 329ms
237:    learn: 0.0120225        total: 297ms    remaining: 327ms
238:    learn: 0.0119127        total: 298ms    remaining: 326ms
239:    learn: 0.0118068        total: 300ms    remaining: 325ms
240:    learn: 0.0117272        total: 301ms    remaining: 323ms
241:    learn: 0.0116391        total: 302ms    remaining: 322ms
242:    learn: 0.0115638        total: 303ms    remaining: 320ms
243:    learn: 0.0114947        total: 304ms    remaining: 319ms
244:    learn: 0.0114429        total: 306ms    remaining: 319ms
245:    learn: 0.0114161        total: 308ms    remaining: 318ms
246:    learn: 0.0113417        total: 309ms    remaining: 316ms
247:    learn: 0.0113061        total: 310ms    remaining: 315ms
248:    learn: 0.0112488        total: 311ms    remaining: 313ms
249:    learn: 0.0111811        total: 312ms    remaining: 312ms
250:    learn: 0.0111042        total: 313ms    remaining: 311ms
251:    learn: 0.0110535        total: 314ms    remaining: 309ms
252:    learn: 0.0109597        total: 316ms    remaining: 308ms
253:    learn: 0.0109199        total: 317ms    remaining: 307ms
254:    learn: 0.0108655        total: 318ms    remaining: 306ms
255:    learn: 0.0108176        total: 319ms    remaining: 304ms
256:    learn: 0.0107406        total: 320ms    remaining: 303ms
257:    learn: 0.0106779        total: 322ms    remaining: 302ms
258:    learn: 0.0106156        total: 323ms    remaining: 301ms
259:    learn: 0.0105815        total: 324ms    remaining: 299ms
260:    learn: 0.0105017        total: 325ms    remaining: 298ms
261:    learn: 0.0104282        total: 326ms    remaining: 297ms
262:    learn: 0.0103701        total: 328ms    remaining: 295ms
263:    learn: 0.0102897        total: 329ms    remaining: 294ms
264:    learn: 0.0102404        total: 330ms    remaining: 293ms
265:    learn: 0.0101866        total: 331ms    remaining: 291ms
266:    learn: 0.0101288        total: 332ms    remaining: 290ms
267:    learn: 0.0100791        total: 334ms    remaining: 289ms
268:    learn: 0.0100525        total: 334ms    remaining: 287ms
269:    learn: 0.0100122        total: 336ms    remaining: 286ms
270:    learn: 0.0099150        total: 337ms    remaining: 285ms
271:    learn: 0.0098805        total: 338ms    remaining: 283ms
272:    learn: 0.0098602        total: 339ms    remaining: 282ms
273:    learn: 0.0098034        total: 341ms    remaining: 281ms
274:    learn: 0.0097612        total: 342ms    remaining: 280ms
275:    learn: 0.0096919        total: 343ms    remaining: 278ms
276:    learn: 0.0096580        total: 344ms    remaining: 277ms
277:    learn: 0.0096453        total: 345ms    remaining: 276ms
278:    learn: 0.0095701        total: 346ms    remaining: 274ms
279:    learn: 0.0095202        total: 351ms    remaining: 276ms
280:    learn: 0.0094621        total: 353ms    remaining: 275ms
```

```
281:    learn: 0.0094082        total: 354ms    remaining: 274ms
282:    learn: 0.0093597        total: 355ms    remaining: 272ms
283:    learn: 0.0093204        total: 356ms    remaining: 271ms
284:    learn: 0.0092710        total: 358ms    remaining: 270ms
285:    learn: 0.0092348        total: 359ms    remaining: 268ms
286:    learn: 0.0091630        total: 360ms    remaining: 267ms
287:    learn: 0.0090812        total: 361ms    remaining: 266ms
288:    learn: 0.0090495        total: 362ms    remaining: 264ms
289:    learn: 0.0090033        total: 363ms    remaining: 263ms
290:    learn: 0.0089672        total: 365ms    remaining: 262ms
291:    learn: 0.0089035        total: 366ms    remaining: 261ms
292:    learn: 0.0088646        total: 367ms    remaining: 259ms
293:    learn: 0.0088381        total: 368ms    remaining: 258ms
294:    learn: 0.0088336        total: 371ms    remaining: 258ms
295:    learn: 0.0087973        total: 372ms    remaining: 256ms
296:    learn: 0.0087356        total: 373ms    remaining: 255ms
297:    learn: 0.0086715        total: 374ms    remaining: 253ms
298:    learn: 0.0086244        total: 375ms    remaining: 252ms
299:    learn: 0.0085786        total: 377ms    remaining: 251ms
300:    learn: 0.0085232        total: 378ms    remaining: 250ms
301:    learn: 0.0084876        total: 379ms    remaining: 248ms
302:    learn: 0.0084511        total: 380ms    remaining: 247ms
303:    learn: 0.0084433        total: 381ms    remaining: 246ms
304:    learn: 0.0084216        total: 382ms    remaining: 245ms
305:    learn: 0.0083714        total: 384ms    remaining: 243ms
306:    learn: 0.0083391        total: 385ms    remaining: 242ms
307:    learn: 0.0082839        total: 386ms    remaining: 241ms
308:    learn: 0.0082373        total: 387ms    remaining: 239ms
309:    learn: 0.0081943        total: 389ms    remaining: 238ms
310:    learn: 0.0081595        total: 390ms    remaining: 237ms
311:    learn: 0.0081428        total: 391ms    remaining: 236ms
312:    learn: 0.0081123        total: 392ms    remaining: 234ms
313:    learn: 0.0080776        total: 394ms    remaining: 233ms
314:    learn: 0.0080241        total: 395ms    remaining: 232ms
315:    learn: 0.0079498        total: 396ms    remaining: 231ms
316:    learn: 0.0079495        total: 398ms    remaining: 230ms
317:    learn: 0.0079297        total: 399ms    remaining: 229ms
318:    learn: 0.0079032        total: 401ms    remaining: 227ms
319:    learn: 0.0078732        total: 402ms    remaining: 226ms
320:    learn: 0.0078619        total: 403ms    remaining: 225ms
321:    learn: 0.0078087        total: 404ms    remaining: 223ms
322:    learn: 0.0077757        total: 405ms    remaining: 222ms
323:    learn: 0.0077389        total: 406ms    remaining: 221ms
324:    learn: 0.0077268        total: 408ms    remaining: 220ms
325:    learn: 0.0076951        total: 409ms    remaining: 218ms
326:    learn: 0.0076683        total: 410ms    remaining: 217ms
327:    learn: 0.0076436        total: 411ms    remaining: 216ms
328:    learn: 0.0076127        total: 412ms    remaining: 214ms
```

```
329:    learn: 0.0075818        total: 414ms    remaining: 213ms
330:    learn: 0.0075444        total: 415ms    remaining: 212ms
331:    learn: 0.0074950        total: 416ms    remaining: 211ms
332:    learn: 0.0074566        total: 417ms    remaining: 209ms
333:    learn: 0.0074095        total: 419ms    remaining: 208ms
334:    learn: 0.0073788        total: 420ms    remaining: 207ms
335:    learn: 0.0073634        total: 421ms    remaining: 206ms
336:    learn: 0.0073327        total: 422ms    remaining: 204ms
337:    learn: 0.0072964        total: 424ms    remaining: 203ms
338:    learn: 0.0072583        total: 425ms    remaining: 202ms
339:    learn: 0.0072368        total: 426ms    remaining: 201ms
340:    learn: 0.0072096        total: 427ms    remaining: 199ms
341:    learn: 0.0071785        total: 429ms    remaining: 198ms
342:    learn: 0.0071547        total: 430ms    remaining: 197ms
343:    learn: 0.0071220        total: 431ms    remaining: 195ms
344:    learn: 0.0070875        total: 432ms    remaining: 194ms
345:    learn: 0.0070647        total: 433ms    remaining: 193ms
346:    learn: 0.0070368        total: 434ms    remaining: 192ms
347:    learn: 0.0070034        total: 436ms    remaining: 190ms
348:    learn: 0.0069664        total: 437ms    remaining: 189ms
349:    learn: 0.0069336        total: 438ms    remaining: 188ms
350:    learn: 0.0069209        total: 439ms    remaining: 186ms
351:    learn: 0.0068959        total: 440ms    remaining: 185ms
352:    learn: 0.0068675        total: 442ms    remaining: 184ms
353:    learn: 0.0068439        total: 443ms    remaining: 183ms
354:    learn: 0.0068007        total: 444ms    remaining: 181ms
355:    learn: 0.0067783        total: 446ms    remaining: 180ms
356:    learn: 0.0067458        total: 447ms    remaining: 179ms
357:    learn: 0.0067322        total: 448ms    remaining: 178ms
358:    learn: 0.0067075        total: 449ms    remaining: 176ms
359:    learn: 0.0066936        total: 451ms    remaining: 176ms
360:    learn: 0.0066590        total: 453ms    remaining: 174ms
361:    learn: 0.0066149        total: 454ms    remaining: 173ms
362:    learn: 0.0065803        total: 455ms    remaining: 172ms
363:    learn: 0.0065610        total: 456ms    remaining: 171ms
364:    learn: 0.0065409        total: 458ms    remaining: 169ms
365:    learn: 0.0065259        total: 459ms    remaining: 168ms
366:    learn: 0.0065074        total: 460ms    remaining: 167ms
367:    learn: 0.0064923        total: 462ms    remaining: 166ms
368:    learn: 0.0064712        total: 463ms    remaining: 164ms
369:    learn: 0.0064656        total: 464ms    remaining: 163ms
370:    learn: 0.0064593        total: 466ms    remaining: 162ms
371:    learn: 0.0064234        total: 467ms    remaining: 161ms
372:    learn: 0.0063929        total: 468ms    remaining: 159ms
373:    learn: 0.0063590        total: 469ms    remaining: 158ms
374:    learn: 0.0063422        total: 471ms    remaining: 157ms
375:    learn: 0.0063168        total: 472ms    remaining: 156ms
376:    learn: 0.0062894        total: 475ms    remaining: 155ms
```

```
377:    learn: 0.0062712        total: 476ms    remaining: 154ms
378:    learn: 0.0062424        total: 477ms    remaining: 152ms
379:    learn: 0.0062225        total: 478ms    remaining: 151ms
380:    learn: 0.0061905        total: 480ms    remaining: 150ms
381:    learn: 0.0061766        total: 481ms    remaining: 148ms
382:    learn: 0.0061464        total: 482ms    remaining: 147ms
383:    learn: 0.0061179        total: 483ms    remaining: 146ms
384:    learn: 0.0060963        total: 484ms    remaining: 145ms
385:    learn: 0.0060714        total: 485ms    remaining: 143ms
386:    learn: 0.0060539        total: 486ms    remaining: 142ms
387:    learn: 0.0060275        total: 488ms    remaining: 141ms
388:    learn: 0.0060034        total: 489ms    remaining: 139ms
389:    learn: 0.0059805        total: 490ms    remaining: 138ms
390:    learn: 0.0059592        total: 491ms    remaining: 137ms
391:    learn: 0.0059144        total: 492ms    remaining: 136ms
392:    learn: 0.0058966        total: 493ms    remaining: 134ms
393:    learn: 0.0058773        total: 494ms    remaining: 133ms
394:    learn: 0.0058583        total: 496ms    remaining: 132ms
395:    learn: 0.0058446        total: 497ms    remaining: 130ms
396:    learn: 0.0058366        total: 498ms    remaining: 129ms
397:    learn: 0.0058068        total: 499ms    remaining: 128ms
398:    learn: 0.0057873        total: 500ms    remaining: 127ms
399:    learn: 0.0057663        total: 502ms    remaining: 125ms
400:    learn: 0.0057455        total: 503ms    remaining: 124ms
401:    learn: 0.0057259        total: 504ms    remaining: 123ms
402:    learn: 0.0057076        total: 505ms    remaining: 122ms
403:    learn: 0.0056956        total: 506ms    remaining: 120ms
404:    learn: 0.0056796        total: 507ms    remaining: 119ms
405:    learn: 0.0056651        total: 508ms    remaining: 118ms
406:    learn: 0.0056563        total: 511ms    remaining: 117ms
407:    learn: 0.0056348        total: 512ms    remaining: 115ms
408:    learn: 0.0056077        total: 513ms    remaining: 114ms
409:    learn: 0.0055901        total: 516ms    remaining: 113ms
410:    learn: 0.0055683        total: 518ms    remaining: 112ms
411:    learn: 0.0055371        total: 520ms    remaining: 111ms
412:    learn: 0.0055182        total: 521ms    remaining: 110ms
413:    learn: 0.0054928        total: 522ms    remaining: 108ms
414:    learn: 0.0054740        total: 523ms    remaining: 107ms
415:    learn: 0.0054626        total: 524ms    remaining: 106ms
416:    learn: 0.0054441        total: 525ms    remaining: 105ms
417:    learn: 0.0054431        total: 527ms    remaining: 103ms
418:    learn: 0.0054284        total: 528ms    remaining: 102ms
419:    learn: 0.0054157        total: 529ms    remaining: 101ms
420:    learn: 0.0053948        total: 530ms    remaining: 99.5ms
421:    learn: 0.0053810        total: 532ms    remaining: 98.2ms
422:    learn: 0.0053635        total: 533ms    remaining: 97ms
423:    learn: 0.0053403        total: 534ms    remaining: 95.7ms
424:    learn: 0.0053403        total: 535ms    remaining: 94.4ms
```

```
425:    learn: 0.0053200        total: 536ms    remaining: 93.2ms
426:    learn: 0.0053169        total: 538ms    remaining: 91.9ms
427:    learn: 0.0053022        total: 539ms    remaining: 90.7ms
428:    learn: 0.0052777        total: 540ms    remaining: 89.4ms
429:    learn: 0.0052650        total: 541ms    remaining: 88.1ms
430:    learn: 0.0052445        total: 542ms    remaining: 86.8ms
431:    learn: 0.0052247        total: 544ms    remaining: 85.6ms
432:    learn: 0.0052024        total: 545ms    remaining: 84.3ms
433:    learn: 0.0051855        total: 546ms    remaining: 83ms
434:    learn: 0.0051707        total: 547ms    remaining: 81.7ms
435:    learn: 0.0051574        total: 550ms    remaining: 80.7ms
436:    learn: 0.0051440        total: 551ms    remaining: 79.4ms
437:    learn: 0.0051318        total: 553ms    remaining: 78.3ms
438:    learn: 0.0051165        total: 554ms    remaining: 77ms
439:    learn: 0.0050985        total: 555ms    remaining: 75.7ms
440:    learn: 0.0050935        total: 556ms    remaining: 74.4ms
441:    learn: 0.0050788        total: 558ms    remaining: 73.2ms
442:    learn: 0.0050498        total: 559ms    remaining: 71.9ms
443:    learn: 0.0050274        total: 560ms    remaining: 70.6ms
444:    learn: 0.0050008        total: 561ms    remaining: 69.3ms
445:    learn: 0.0049869        total: 562ms    remaining: 68.1ms
446:    learn: 0.0049712        total: 563ms    remaining: 66.8ms
447:    learn: 0.0049557        total: 565ms    remaining: 65.5ms
448:    learn: 0.0049386        total: 566ms    remaining: 64.3ms
449:    learn: 0.0049294        total: 567ms    remaining: 63ms
450:    learn: 0.0049292        total: 568ms    remaining: 61.8ms
451:    learn: 0.0049293        total: 570ms    remaining: 60.5ms
452:    learn: 0.0049155        total: 571ms    remaining: 59.2ms
453:    learn: 0.0049028        total: 572ms    remaining: 58ms
454:    learn: 0.0048916        total: 573ms    remaining: 56.7ms
455:    learn: 0.0048796        total: 575ms    remaining: 55.5ms
456:    learn: 0.0048647        total: 576ms    remaining: 54.2ms
457:    learn: 0.0048476        total: 577ms    remaining: 53ms
458:    learn: 0.0048257        total: 579ms    remaining: 51.7ms
459:    learn: 0.0048138        total: 580ms    remaining: 50.4ms
460:    learn: 0.0047944        total: 581ms    remaining: 49.2ms
461:    learn: 0.0047807        total: 583ms    remaining: 47.9ms
462:    learn: 0.0047807        total: 584ms    remaining: 46.7ms
463:    learn: 0.0047704        total: 585ms    remaining: 45.4ms
464:    learn: 0.0047701        total: 586ms    remaining: 44.1ms
465:    learn: 0.0047420        total: 587ms    remaining: 42.9ms
466:    learn: 0.0047276        total: 589ms    remaining: 41.6ms
467:    learn: 0.0047079        total: 590ms    remaining: 40.3ms
468:    learn: 0.0046897        total: 591ms    remaining: 39.1ms
469:    learn: 0.0046755        total: 593ms    remaining: 37.8ms
470:    learn: 0.0046605        total: 594ms    remaining: 36.6ms
471:    learn: 0.0046459        total: 595ms    remaining: 35.3ms
472:    learn: 0.0046457        total: 596ms    remaining: 34ms
```

```
473:    learn: 0.0046284        total: 598ms    remaining: 32.8ms
474:    learn: 0.0046152        total: 599ms    remaining: 31.5ms
475:    learn: 0.0046080        total: 600ms    remaining: 30.3ms
476:    learn: 0.0046080        total: 601ms    remaining: 29ms
477:    learn: 0.0045926        total: 603ms    remaining: 27.7ms
478:    learn: 0.0045778        total: 604ms    remaining: 26.5ms
479:    learn: 0.0045640        total: 605ms    remaining: 25.2ms
480:    learn: 0.0045501        total: 606ms    remaining: 23.9ms
481:    learn: 0.0045376        total: 608ms    remaining: 22.7ms
482:    learn: 0.0045222        total: 609ms    remaining: 21.4ms
483:    learn: 0.0045090        total: 610ms    remaining: 20.2ms
484:    learn: 0.0044955        total: 611ms    remaining: 18.9ms
485:    learn: 0.0044832        total: 613ms    remaining: 17.6ms
486:    learn: 0.0044705        total: 614ms    remaining: 16.4ms
487:    learn: 0.0044668        total: 615ms    remaining: 15.1ms
488:    learn: 0.0044595        total: 617ms    remaining: 13.9ms
489:    learn: 0.0044486        total: 618ms    remaining: 12.6ms
490:    learn: 0.0044298        total: 619ms    remaining: 11.3ms
491:    learn: 0.0044190        total: 620ms    remaining: 10.1ms
492:    learn: 0.0044107        total: 621ms    remaining: 8.82ms
493:    learn: 0.0043975        total: 623ms    remaining: 7.56ms
494:    learn: 0.0043869        total: 624ms    remaining: 6.3ms
495:    learn: 0.0043747        total: 625ms    remaining: 5.04ms
496:    learn: 0.0043635        total: 626ms    remaining: 3.78ms
497:    learn: 0.0043517        total: 628ms    remaining: 2.52ms
498:    learn: 0.0043394        total: 629ms    remaining: 1.26ms
499:    learn: 0.0043208        total: 630ms    remaining: 0us
Accuracy 0.8360655737704918
confusion matrix [[24  5]
 [ 5 27]]
Precision : 0.84375
Recall : 0.84375
classification report               precision    recall  f1-score   support

           0       0.83      0.83      0.83        29
           1       0.84      0.84      0.84        32

    accuracy                           0.84        61
   macro avg       0.84      0.84      0.84        61
weighted avg       0.84      0.84      0.84        61

Training score: 1.0
Testing score: 0.8360655737704918
Kappa score: 0.6713362068965517
```

17 svm

[ ]:
```
from sklearn.svm import SVC

model7 = SVC(kernel='linear', C=1.0)
model7.fit(x_train, y_train)
y_pred7 = model7.predict(x_test)
accuracy7 = accuracy_score(y_test, y_pred7)
cfm7 = confusion_matrix(y_test, y_pred7)
classreport7 = classification_report(y_test, y_pred7)
recall7 = recall_score(y_test, y_pred7)
precision7 = precision_score(y_test, y_pred7)
print("Accuracy", accuracy7)
print("confusion matrix", cfm7)
print("Precision :", precision7)
print("Recall :", recall7)
print("classification report", classreport7)
print("Training score:", model7.score(x_train, y_train))
print('Testing score:', model7.score(x_test, y_test))
print("Kappa score:", cohen_kappa_score(y_test, y_pred7))
```

```
Accuracy 0.8688524590163934
confusion matrix [[25  4]
 [ 4 28]]
Precision : 0.875
Recall : 0.875
classification report               precision    recall  f1-score   support

           0       0.86      0.86      0.86        29
           1       0.88      0.88      0.88        32

    accuracy                           0.87        61
   macro avg       0.87      0.87      0.87        61
weighted avg       0.87      0.87      0.87        61

Training score: 0.8636363636363636
Testing score: 0.8688524590163934
Kappa score: 0.7370689655172413
```

18 Naive Bayes

[ ]:
```
from sklearn.naive_bayes import GaussianNB
model8 = GaussianNB()
model8.fit(x_train, y_train)
y_pred8 = model7.predict(x_test)
accuracy8 = accuracy_score(y_test, y_pred8)
cfm8 = confusion_matrix(y_test, y_pred)
```

```
classreport8 = classification_report(y_test, y_pred8)
recall8 = recall_score(y_test, y_pred8)
precision8 = precision_score(y_test, y_pred8)
print("Accuracy", accuracy8)
print("confusion matrix", cfm8)
print("Precision :", precision8)
print("Recall :", recall8)
print("classification report", classreport8)
print("Training score:", model7.score(x_train, y_train))
print('Testing score:', model7.score(x_test, y_test))
print("Kappa score:", cohen_kappa_score(y_test, y_pred8))
```

```
Accuracy 0.8688524590163934
confusion matrix [[25  4]
 [ 3 29]]
Precision : 0.875
Recall : 0.875
classification report               precision    recall  f1-score   support

           0       0.86      0.86      0.86        29
           1       0.88      0.88      0.88        32

    accuracy                           0.87        61
   macro avg       0.87      0.87      0.87        61
weighted avg       0.87      0.87      0.87        61

Training score: 0.8636363636363636
Testing score: 0.8688524590163934
Kappa score: 0.7370689655172413
```

19 KNN

[ ]:
```
from sklearn.neighbors import KNeighborsClassifier

model9 = KNeighborsClassifier()
model9.fit(x_train, y_train)
y_pred9 = model9.predict(x_test)
accuracy9 = accuracy_score(y_test, y_pred9)
cfm9 = confusion_matrix(y_test, y_pred9)
classreport9 = classification_report(y_test, y_pred9)
recall9 = recall_score(y_test, y_pred9)
precision9 = precision_score(y_test, y_pred9)
print("Accuracy", accuracy9)
print("confusion matrix", cfm9)
print("Precision :", precision9)
print("Recall :", recall9)
print("classification report", classreport9)
```

```
print("Training score:", model9.score(x_train, y_train))
print('Testing score:', model9.score(x_test, y_test))
print("Kappa score:", cohen_kappa_score(y_test, y_pred9))
```

```
Accuracy 0.6885245901639344
confusion matrix [[18 11]
 [ 8 24]]
Precision : 0.6857142857142857
Recall : 0.75
classification report               precision    recall  f1-score   support

           0       0.69      0.62      0.65        29
           1       0.69      0.75      0.72        32

    accuracy                           0.69        61
   macro avg       0.69      0.69      0.69        61
weighted avg       0.69      0.69      0.69        61

Training score: 0.7603305785123967
Testing score: 0.6885245901639344
Kappa score: 0.37249593936112624
```

20 hyperparameter tuning Decision Tree Classifier

[ ]:
```
from sklearn.model_selection import GridSearchCV
param_grid = {
    'criterion': ['gini', 'entropy'],
    'max_depth': [None, 5, 10, 20],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4]
}
dt_classifier = DecisionTreeClassifier()
grid_search = GridSearchCV(dt_classifier, param_grid, cv=5)
grid_search.fit(x_train, y_train)
best_params = grid_search.best_params_
print("Best hyperparameters:", best_params)
best_dt_classifier = DecisionTreeClassifier(**best_params)
best_dt_classifier.fit(x_train, y_train)
accuracy10 = accuracy_score(y_test, best_dt_classifier.predict(x_test))
print("Accuracy:", accuracy10)
```

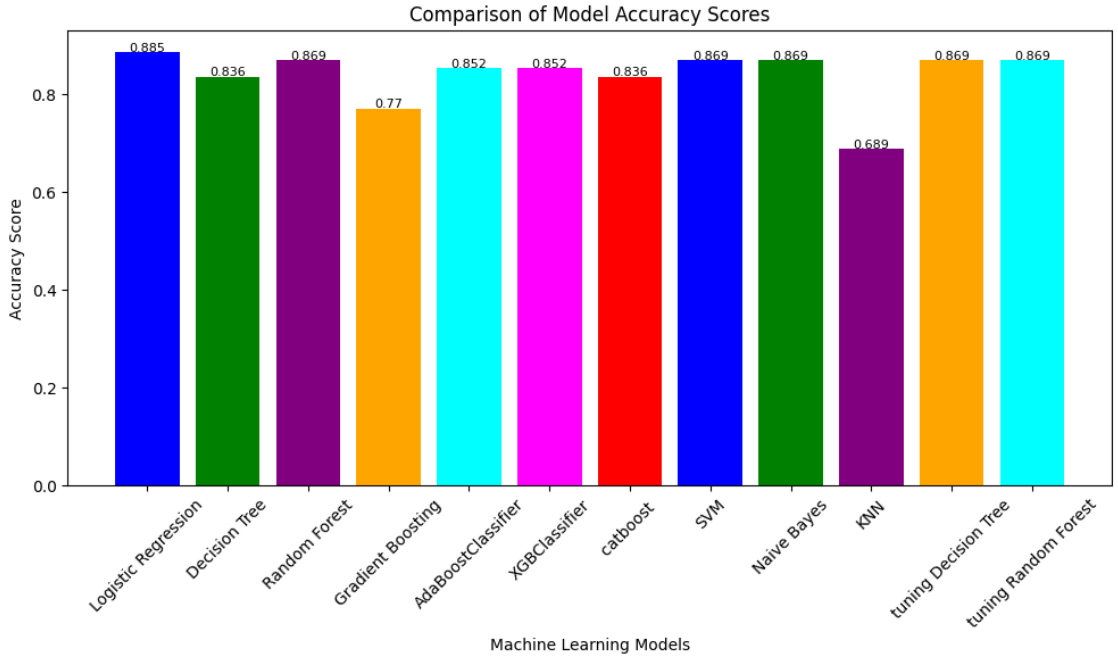
```
Best hyperparameters: {'criterion': 'entropy', 'max_depth': 10,
'min_samples_leaf': 1, 'min_samples_split': 5}
Accuracy: 0.819672131147541
```

21 Hyperparameter tuning Random Forest

[ ]:
```
from sklearn.model_selection import GridSearchCV
param_grid = {
    'criterion': ['gini', 'entropy'],
    'max_depth': [None, 5, 10, 20],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4]
}
rf_classifier = RandomForestClassifier()
grid_search = GridSearchCV(rf_classifier, param_grid, cv=5)
grid_search.fit(x_train, y_train)
best_params = grid_search.best_params_
print("Best hyperparameters:", best_params)
best_rf_classifier = RandomForestClassifier(**best_params)
best_rf_classifier.fit(x_train, y_train)
accuracy11 = accuracy_score(y_test, best_rf_classifier.predict(x_test))
print("Accuracy:", accuracy11)
```

```
Best hyperparameters: {'criterion': 'entropy', 'max_depth': 5,
'min_samples_leaf': 2, 'min_samples_split': 10}
Accuracy: 0.8688524590163934
```

[ ]:
```
accuracies = [accuracy, accuracy1, accuracy2, accuracy3, accuracy4, accuracy5,␣
 ↪accuracy6, accuracy7, accuracy8, accuracy9, accuracy10, accuracy11]

# Create a list of model names
model_names = ["Logistic Regression", "Decision Tree", "Random␣
 ↪Forest","Gradient Boosting",'AdaBoostClassifier','XGBClassifier','catboost␣
 ↪','SVM','Naive Bayes','KNN','tuning Decision Tree','tuning Random Forest']
colors = ['blue', 'green', 'purple', 'orange', 'cyan', 'magenta', 'red']

plt.figure(figsize=(10, 6))
bars = plt.bar(model_names, accuracies, color=colors)
plt.xlabel('Machine Learning Models')
plt.ylabel('Accuracy Score')
plt.title('Comparison of Model Accuracy Scores')
plt.xticks(rotation=45)
plt.tight_layout()
for bar, score in zip(bars, accuracies):
    yval = bar.get_height()
    plt.text(bar.get_x() + bar.get_width()/2, yval + 0.01, round(score, 3),␣
 ↪ha='center', va='center_baseline', fontsize=8)
```



[ ]:
```
# prompt: show all values of the model

for i, model_name in enumerate(model_names):
  print(f"{i+1}. {model_name}: {accuracies[i]}")
```

```
1. Logistic Regression: 0.8852459016393442
2. Decision Tree: 0.8360655737704918
3. Random Forest: 0.8688524590163934
4. Gradient Boosting: 0.7704918032786885
5. AdaBoostClassifier: 0.8524590163934426
6. XGBClassifier: 0.8524590163934426
7. catboost : 0.8360655737704918
8. SVM: 0.8688524590163934
9. Naive Bayes: 0.8688524590163934
10. KNN: 0.6885245901639344
11. tuning Decision Tree: 0.8688524590163934
12. tuning Random Forest: 0.8688524590163934
```

[ ]:
```
df.head(2)
```

[ ]:

	age	gender	chest_pain	rest_bps	cholestrol	fasting_blood_sugar	\
0	63	1	3	145	233	1	
1	37	1	2	130	250	0	

	rest_ecg	thalach	exer_angina	old_peak	slope	ca	thalassemia	target
0	0	150	0	2.3	0	0	1	1

1	1	187	0	3.5	0	0	2	1

22 Building a Predictive System get a user from input

[ ]:
```
def get_user_input():
    age = int(input("Enter age: "))
    gender = int(input("Enter gender (1 = male, 0 = female): "))
    chest_pain = int(input("Enter chest pain type (0-3) (0 for typical angina,␣
 ↪1 for atypical angina, 2 for non-anginal pain, 3 for asymptomatic): "))
    rest_bps = int(input("Enter resting blood pressure: "))
    cholestrol = int(input("Enter cholesterol level: "))
    fasting_blood_sugar = int(input("Enter fasting blood sugar (> 120 mg/dl, 1␣
 ↪= true; 0 = false): "))
    rest_ecg = int(input("Enter resting electrocardiographic results (0-2) (0␣
 ↪for normal, 1 for ST-T wave abnormality, 2 for left ventricular hypertrophy):
 ↪ "))
    thalach = int(input("Enter maximum heart rate achieved: "))
    exer_angina = int(input("Enter exercise induced angina (1 = yes; 0 = no):␣
 ↪"))
    old_peak = float(input("Enter ST depression induced by exercise relative to␣
 ↪rest: "))
    slope = int(input("Enter the slope of the peak exercise ST segment (0-2) (0␣
 ↪for upsloping, 1 for flat, 2 for downsloping:): "))
    ca = int(input("Enter number of major vessels colored by fluoroscopy (0-3):␣
 ↪"))
    thalassemia = int(input("Enter thalassemia (1 = normal; 2 = fixed defect; 3␣
 ↪= reversable defect): "))

    return [[age, gender, chest_pain, rest_bps, cholestrol,␣
 ↪fasting_blood_sugar, rest_ecg, thalach, exer_angina, old_peak, slope, ca,␣
 ↪thalassemia]]

# Get user input
user_data = get_user_input()

# Make prediction
user_prediction = model.predict(user_data)
user_prediction_proba = model.predict_proba(user_data)

# Display the prediction
print(f'Prediction: {"Heart Attack" if user_prediction[0] == 1 else "No Heart␣
 ↪Attack"}')
print(f'Prediction Probability: {user_prediction_proba[0][1] * 100:.2f}% for␣
 ↪Heart Attack')
```

```
Enter age: 35
```

```
Enter gender (1 = male, 0 = female): 1
Enter chest pain type (0-3) (0 for typical angina, 1 for atypical angina, 2 for
non-anginal pain, 3 for asymptomatic): 2
Enter resting blood pressure: 200
Enter cholesterol level: 300
Enter fasting blood sugar (> 120 mg/dl, 1 = true; 0 = false): 1
Enter resting electrocardiographic results (0-2) (0 for normal, 1 for ST-T wave
abnormality, 2 for left ventricular hypertrophy): 5
Enter maximum heart rate achieved: 250
Enter exercise induced angina (1 = yes; 0 = no): 1
Enter ST depression induced by exercise relative to rest: 8
Enter the slope of the peak exercise ST segment (0-2) (0 for upsloping, 1 for
flat, 2 for downsloping:): 2
Enter number of major vessels colored by fluoroscopy (0-3): 1
Enter thalassemia (1 = normal; 2 = fixed defect; 3 = reversable defect): 0
Prediction: Heart Attack
Prediction Probability: 83.93% for Heart Attack

/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does
not have valid feature names, but LogisticRegression was fitted with feature
names
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/sklearn/base.py:439: UserWarning: X does
not have valid feature names, but LogisticRegression was fitted with feature
names
  warnings.warn(
```

23 Conclusion

23.1 This heart attack prediction system serves as a tool to assist individuals in understanding their risk of heart attack based on specific health indicators. It is important to note that this system is not a substitute for professional medical advice. Always consult with a healthcare provider for an accurate diagnosis and treatment plan.