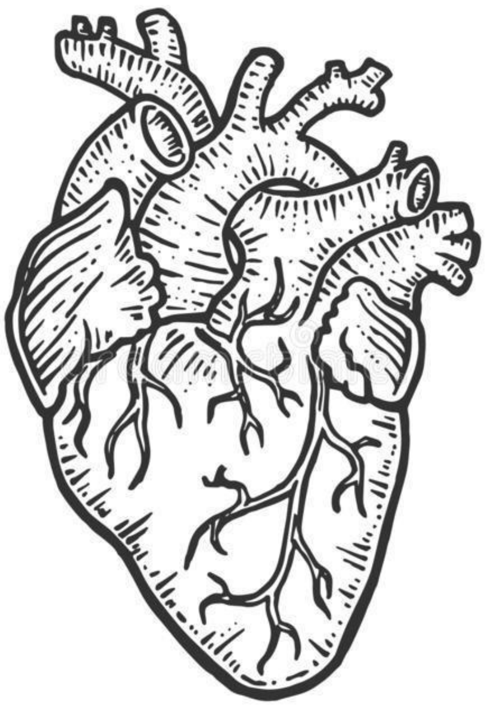
MAJOR PROJECT BY TEAM I



HEART CLASSIFICATION PREDICTION MODEL

Done By:
MOHAMMED YOONUS
BINDU A
MALKAN ARSHID
VIJIYA SURYA

CONTENT

INTRODUCTION

Heart disease describes a range of conditions that affect the heart. Heart diseases include Blood vessel diseases, such as coronary artery disease, Irregular heartbeats (arrhythmias), Heart problems from birth (congenital heart defects), Heart problems you're born with (congenital heart defects), Disease of the heart muscle, and heart valve disease.

According to the World Health Organization, consistently 12 million deaths occur worldwide due to heart disease. It is one of the biggest causes of morbidity and mortality among the population. Heart disease prediction is regarded as the most important subject in the section of data analysis.

Machine Learning is used across many ranges around the world. The healthcare industry is not excluded. Machine Learning can play an essential role in predicting the presence/absence of locomotors disorders, and heart diseases and that's only the tip of the iceberg. Such data, whenever anticipated well ahead of time, can give significant instincts to the specialists. The early guess of heart disease can support decisions on lifestyle changes in high-risk patients and thusly decrease the complexities, which can be an extraordinary achievement in the field of medicine. The usage of reasonable technology support in this respect can end up being profoundly gainful to the medical crew and patients.

DATA DESCRIPTION

The dataset used in this article is the Cleveland Heart Disease dataset taken from the UCI repository. The dataset consists of 76 attributes. There are 14 columns in the dataset, which are described below:

1) **Age**- displays the individual's age.
2) **Sex**-displays the gender of the individual using the following format:
   1=male
   0=female
3) **Chest-pain type**(cp)-displays the type of chest pain experienced by the individual using the following format:
   1=typical angina
   2=atypical angina
   3=non-anginal pain
   4=asymptotic
4) **Resting blood pressure**(trestbps)-displays the resting blood pressure value of an individual in mm Hg (unit).
5) **Serum cholesterol** (chol)- displays the serum cholesterol in mg/dl(unit).
6) **Fasting blood sugar**(fbs)- compares an individual's fasting blood sugar value with 120mg/dl.
   If fasting blood sugar>120mg/dl then: 1 [true]
   Else: 0 [false]
7) **Resting ECG**(restecg)-displays resting electrocardiographic results
   0=normal
   1=having ST-T wave abnormality
   2=left ventricular hypertrophy
8) **Max heart rate achieved**(thalach)-displays the max heart rate achieved by an individual.
9) **Exercised induced angina**(exang)-

1=yes
0=no

10) ***S*T depression induced by exercise relative to rest**(Oldpeak)-displays the value which is an integer or float.

11) **Peak exercise ST segment*(***Slope)-
1 = upsloping
2=flat
3=downsloping

12) **Number of major vessels colored by fluoroscopy(**Ca)-displays the value as integer or float.

13) **thal-displays the thalassemia** :

3 = normal

6 = fixed defect

7 = reversible defect

14) **Diagnosis of heart disease** (num)-displays whether the individual is suffering from heart disease or not:

Value 0: <50% diameter narrowing

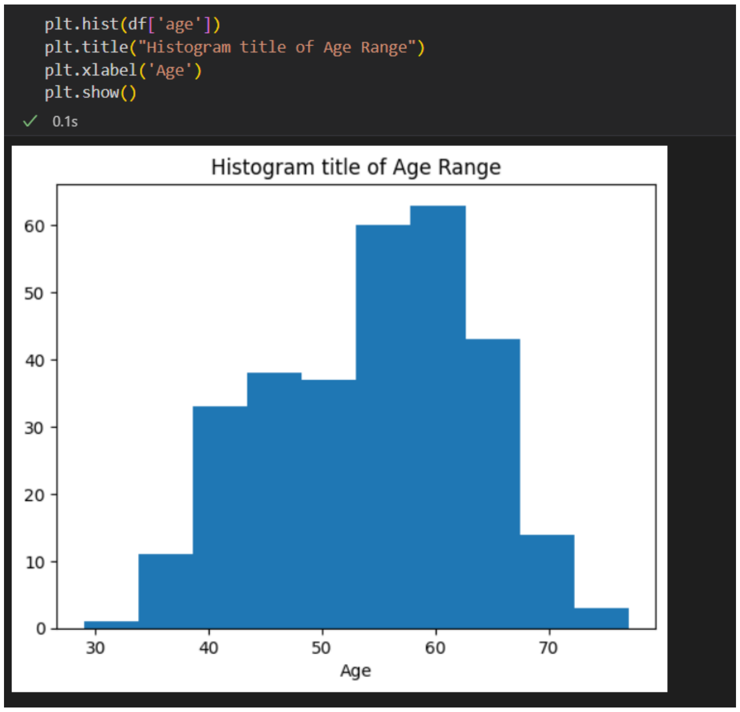
Value 1: >50% diameter narrowing

ALGORITHM

Numerous modules, packages, and classifiers are used in this project. The following Machine learning algorithms /classifiers are utilized on the dataset:

- Numpy as np
- Pandas as pd
- Matplotlib as plt
- Seaborn as sns
- Sklearn.model_selection for train_test_split
- Sklearn.linear_model for Logistic regression
- Sklearn.metrics for accuracy_score
- Sklearn.preprocessing for StandardScaler
- Sklearn.metrics for classification_report
- Sklearn.neighbors for KNeighborsClassifier
- Sklearn.svm for SVC
- Sklearn.naive_bayes for GaussianNB
- Sklearn.tree for DecisionTreeClassifier
- Sklearn.ensemble for RandomForestClassifier
- sklearn.metrics for confusion_matrix, accuracy_score
- xgboost for XGBClassifier

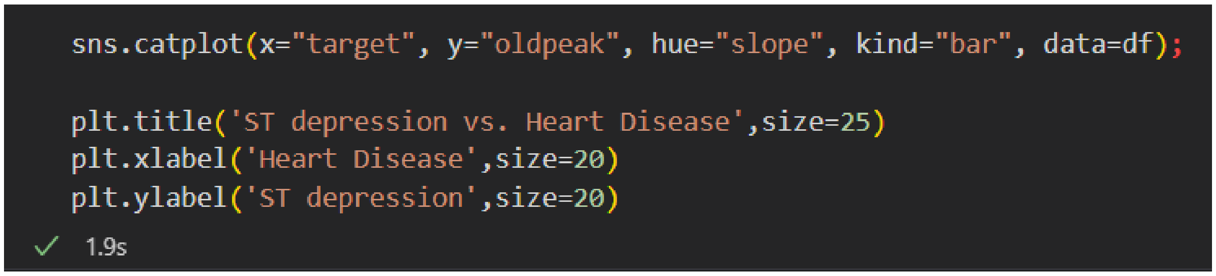
Data Visualization



```
plt.hist(df['age'])
plt.title("Histogram title of Age Range")
plt.xlabel('Age')
plt.show()
```

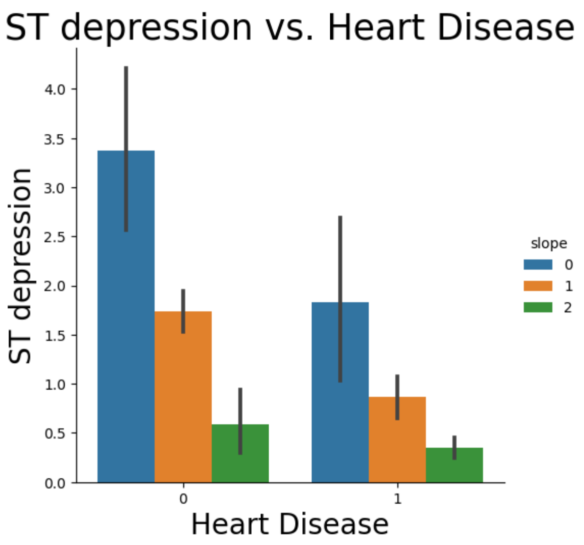
✓ 0.1s

Heart Disease is very common in seniors which are composed of the age group 60 and above and common among adults which belong to the age group of 41 to 60. But it's rare among the age group of 19 to 40 and very rare among the age group of 0 to 18.

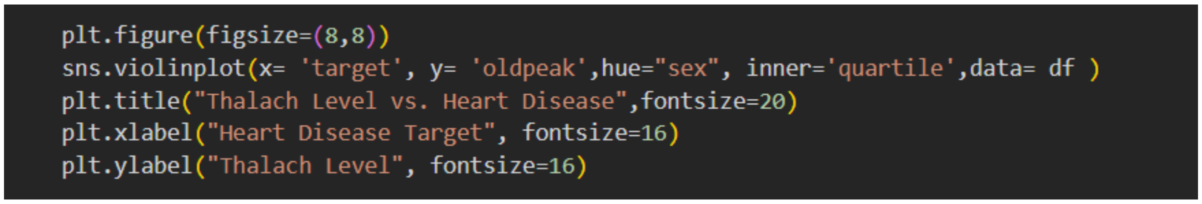


```
sns.catplot(x="target", y="oldpeak", hue="slope", kind="bar", data=df);

plt.title('ST depression vs. Heart Disease',size=25)
plt.xlabel('Heart Disease',size=20)
plt.ylabel('ST depression',size=20)
```

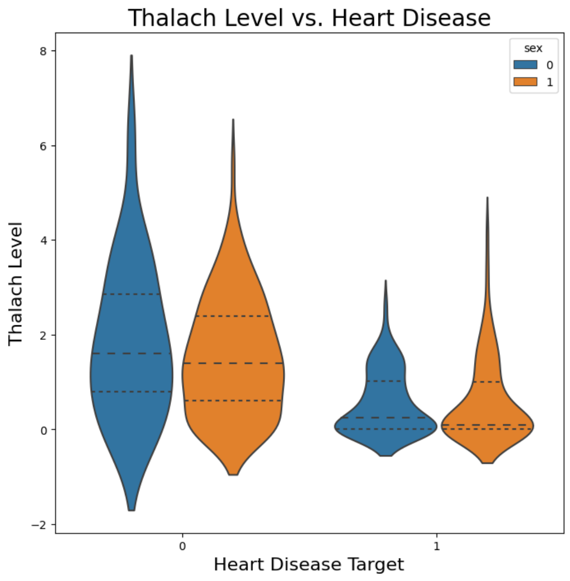
✓ 1.9s



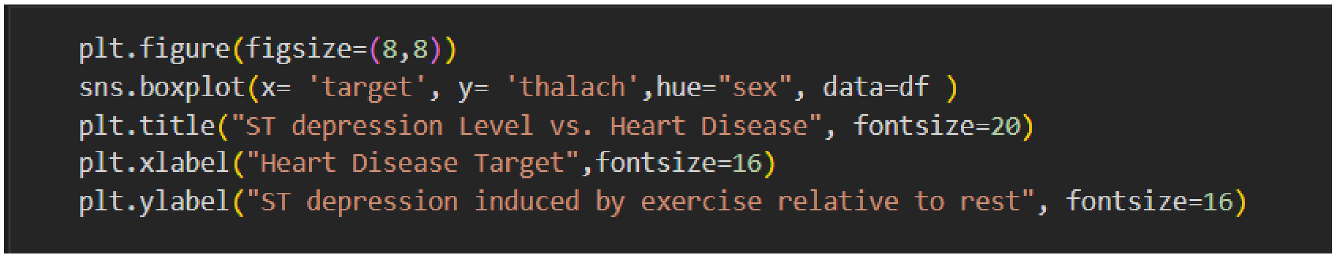
Heart disease may result from an ST segment trace that is abnormally low or below the baseline. This is consistent with the plot above since those who have low ST Depression are more likely to get heart disease. Although a high ST depression is natural and healthy. The "slope" color, which has values of 0 for upsloping, 1 for flat, and 2, relates to the peak workout ST segment. Equal distributions of the three slope groups can be seen in both positive and negative heart disease patients.



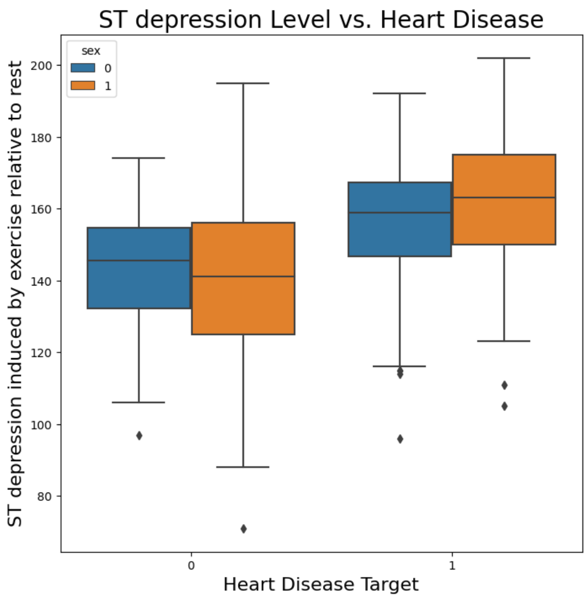
```
plt.figure(figsize=(8,8))
sns.violinplot(x= 'target', y= 'oldpeak',hue="sex", inner='quartile',data= df )
plt.title("Thalach Level vs. Heart Disease",fontsize=20)
plt.xlabel("Heart Disease Target", fontsize=16)
plt.ylabel("Thalach Level", fontsize=16)
```



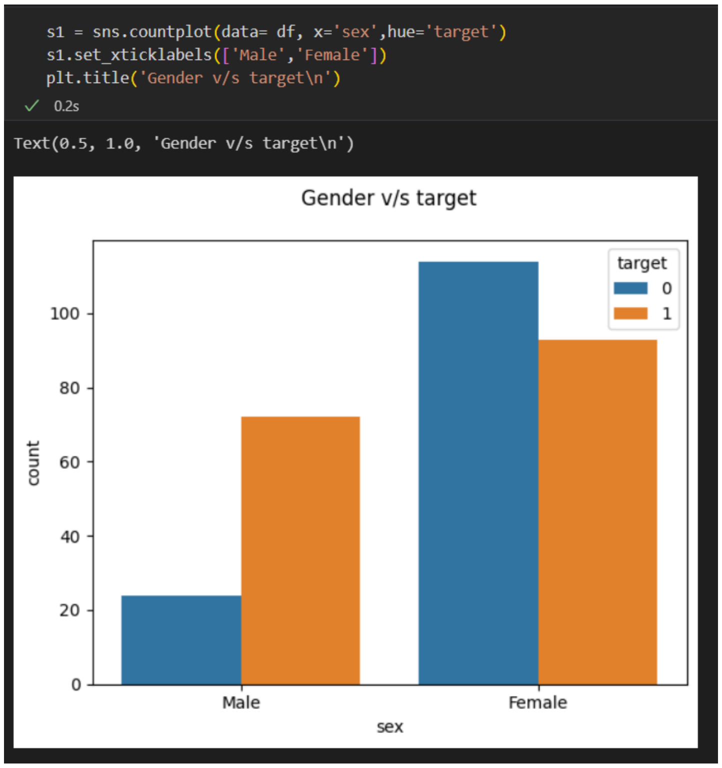
Overall shapes and distribution for negative & positive patients are shown to differ greatly.
Positive patients have lower median ST depression levels, and a large portion of their data fall between 0 and 2, whereas negative patients' data fall between 1 and 3. Additionally, there aren't many distinctions between target outcomes for men and women.



```
plt.figure(figsize=(8,8))
sns.boxplot(x= 'target', y= 'thalach',hue="sex", data=df )
plt.title("ST depression Level vs. Heart Disease", fontsize=20)
plt.xlabel("Heart Disease Target",fontsize=16)
plt.ylabel("ST depression induced by exercise relative to rest", fontsize=16)
```



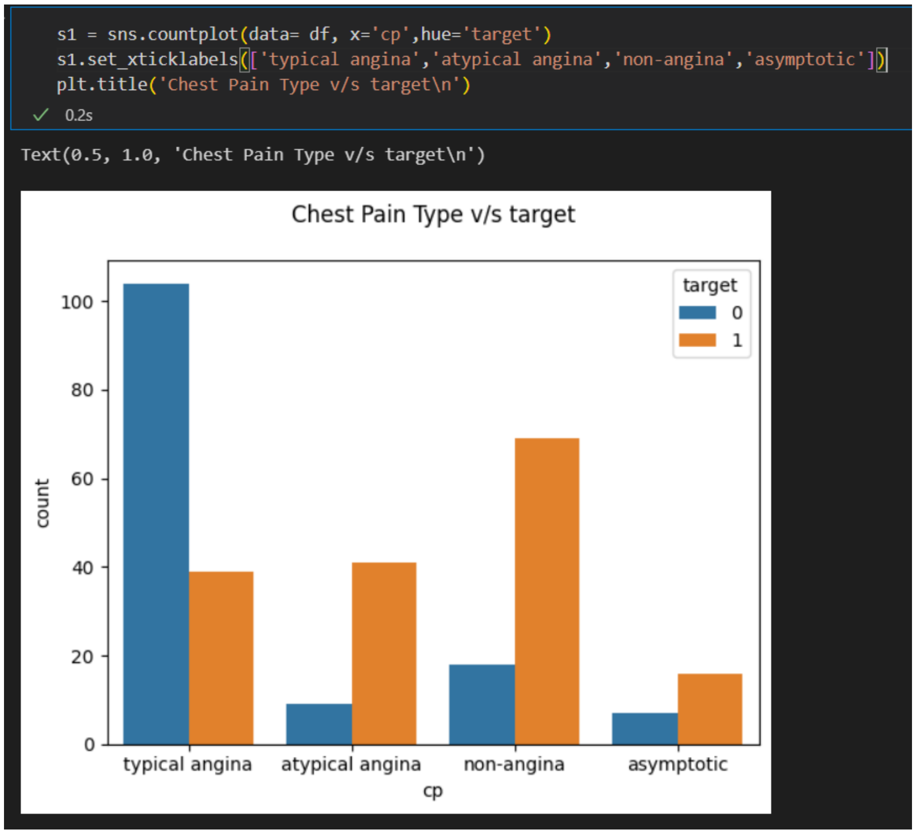
While negative individuals had lower levels, positive patients show an elevated median for ST depression.
In addition, there aren't many distinctions between male and female goal outcomes, with the exception of the fact that ST Depression ranges are a little bit larger in men.



```
s1 = sns.countplot(data= df, x='sex',hue='target')
s1.set_xticklabels(['Male','Female'])
plt.title('Gender v/s target\n')
```

```
Text(0.5, 1.0, 'Gender v/s target\n')
```

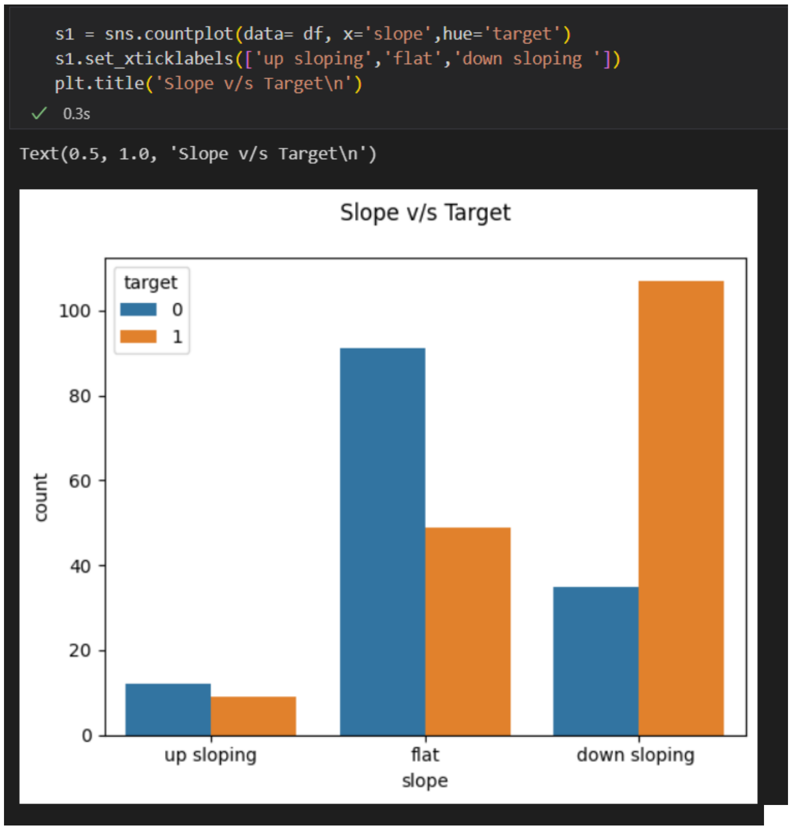
According to this data set, Males are more susceptible to getting heart disease than females.
Sudden Heart Attacks are experienced by Men between 70%-89%. Women may experience a heart attack with no chest pressure at all, they usually experience nausea or vomiting which are often confused with acid reflux or the flu.



```
s1 = sns.countplot(data= df, x='cp',hue='target')
s1.set_xticklabels(['typical angina','atypical angina','non-angina','asymptotic'])
plt.title('Chest Pain Type v/s target\n')
```

```
Text(0.5, 1.0, 'Chest Pain Type v/s target\n')
```

From this bar graph, it is evident that having chest pain does not indicate or be an indicator of heart disease.

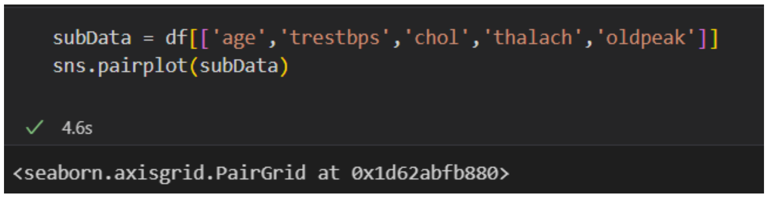


```
s1 = sns.countplot(data= df, x='slope',hue='target')
s1.set_xticklabels(['up sloping','flat','down sloping '])
plt.title('Slope v/s Target\n')
```

✓ 0.3s

```
Text(0.5, 1.0, 'Slope v/s Target\n')
```

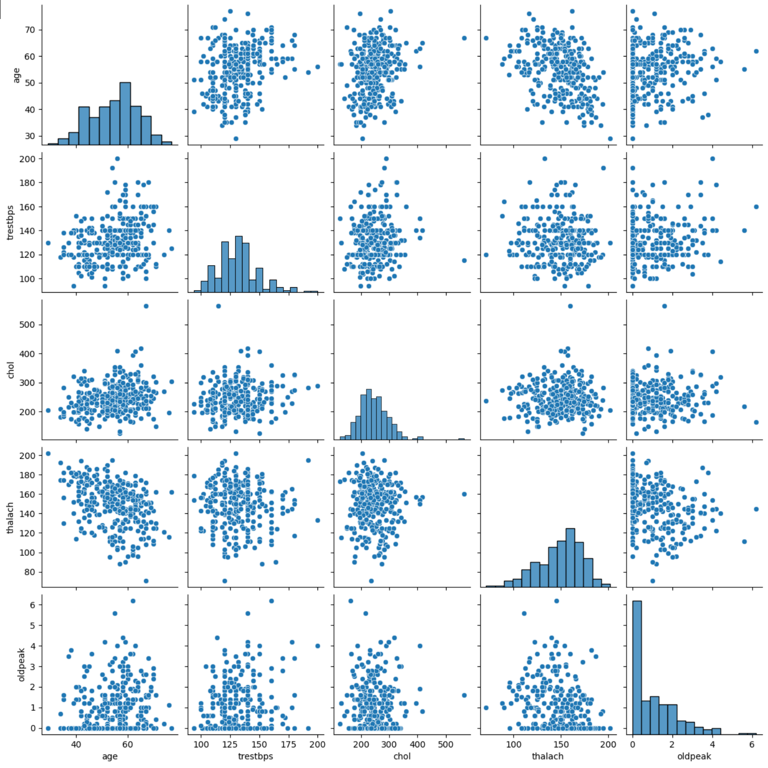
The peak workout ST segment is identified by the "slope" hue. Equal distributions of the three slope groups are present in patients with positive and negative heart disease.



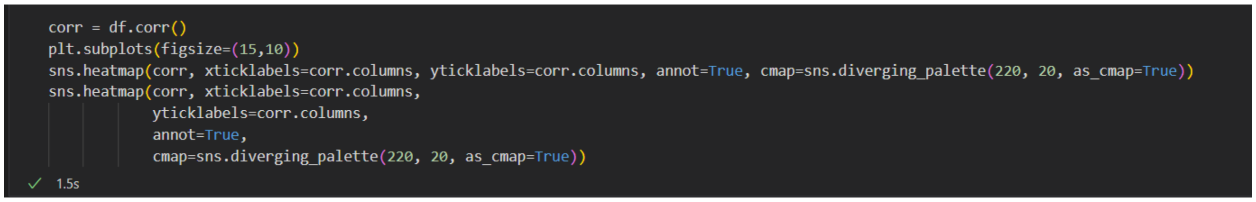
```
subData = df[['age','trestbps','chol','thalach','oldpeak']]
sns.pairplot(subData)
```

✓ 4.6s

```
<seaborn.axisgrid.PairGrid at 0x1d62abfb880>
```

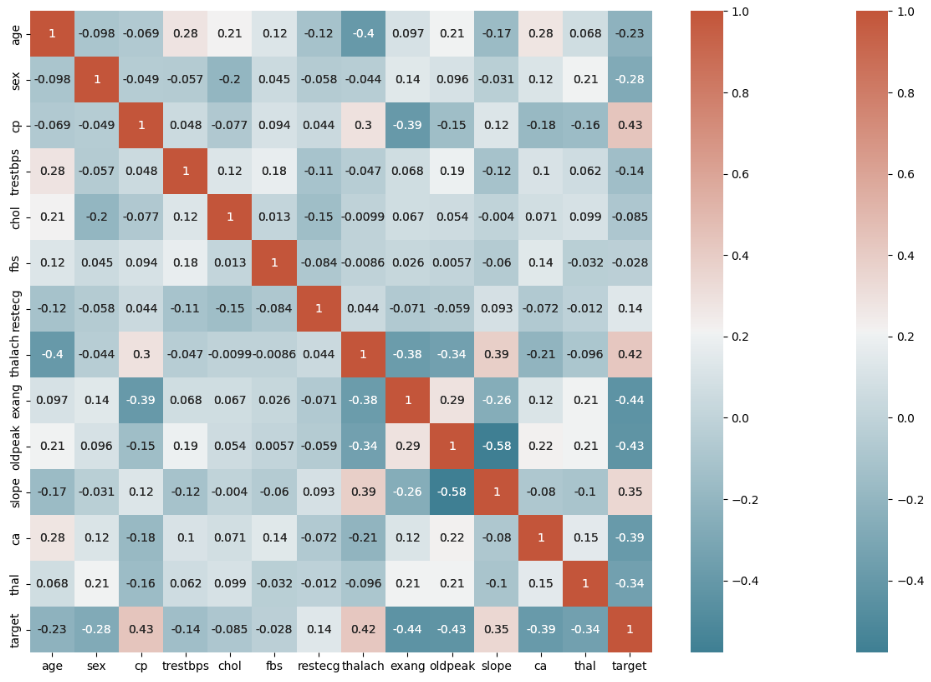


This is a pair plot against a few of the attributes such as age, trestbps, chol, thalach, and oldpeak.



```
corr = df.corr()
plt.subplots(figsize=(15,10))
sns.heatmap(corr, xticklabels=corr.columns, yticklabels=corr.columns, annot=True, cmap=sns.diverging_palette(220, 20, as_cmap=True))
sns.heatmap(corr, xticklabels=corr.columns,
            yticklabels=corr.columns,
            annot=True,
            cmap=sns.diverging_palette(220, 20, as_cmap=True))
```

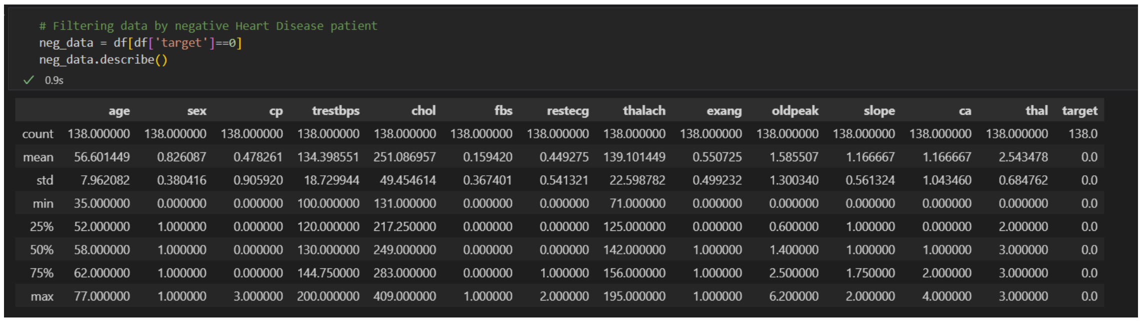
✓ 1.5s



Correlation shows whether the characteristics are related to each other or to the target variable. Correlation can be positive (The value of the target variable increased) or negative (The value of the target variable decreased). From this heatmap, we can observe that “cp” Chest pain is highly related to the target variable. Compared to the relation between the other two variables, we can say that Chest Pain contributes the most to the prediction of the presence of heart disease.

Evaluation and Comparison

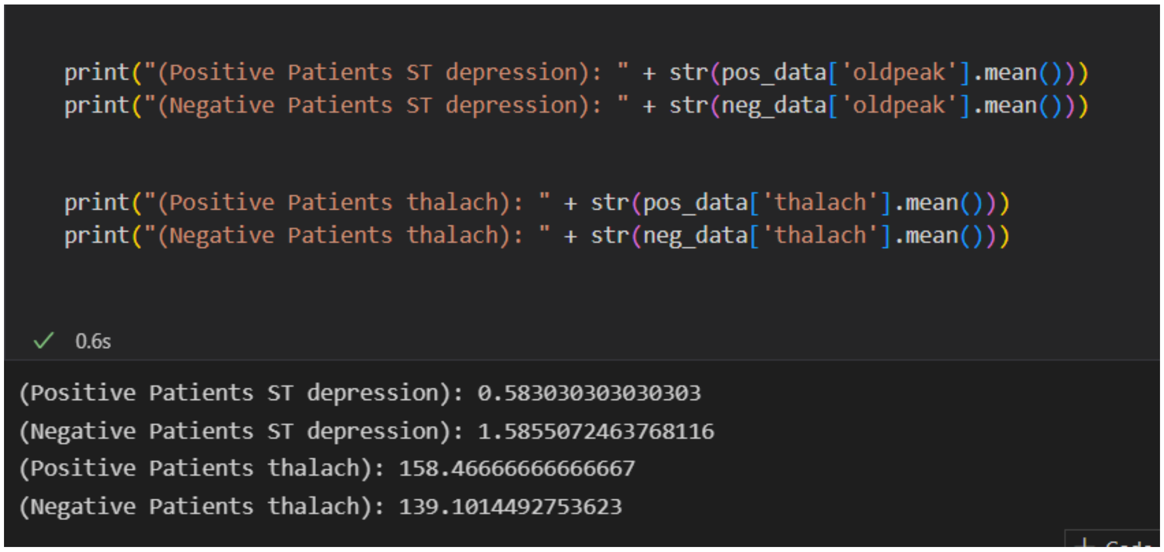
A classifier in machine learning is an algorithm that automatically orders or categorizes data into one or more of a set of 'classes'. There are different types of classifiers in ML like Perceptron, Naive Bayes, Decision Tree, Artificial Neural Networks, etc. For this project, we have used Logistic Regression, K-NN, SVM, Naives Bayes, Decision Trees, Random Forest, and XGBoost.



```
# Filtering data by negative Heart Disease patient
neg_data = df[df['target']==0]
neg_data.describe()
```

✓ 0.9s

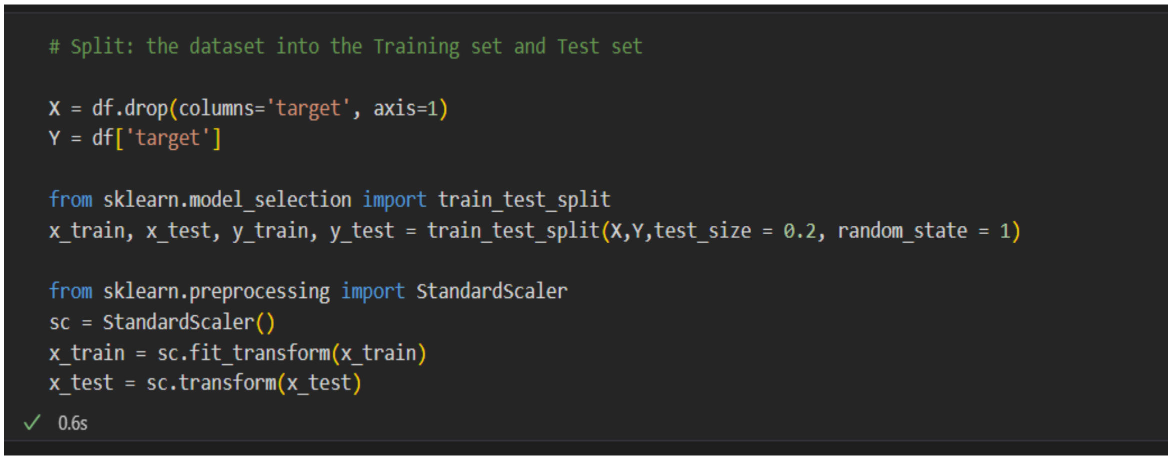
	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
count	138.000000	138.000000	138.000000	138.000000	138.000000	138.000000	138.000000	138.000000	138.000000	138.000000	138.000000	138.000000	138.000000	138.0
mean	56.601449	0.826087	0.478261	134.398551	251.086957	0.159420	0.449275	139.101449	0.550725	1.585507	1.166667	1.166667	2.543478	0.0
std	7.962082	0.380416	0.905920	18.729944	49.454614	0.367401	0.541321	22.598782	0.499232	1.300340	0.561324	1.043460	0.684762	0.0
min	35.000000	0.000000	0.000000	100.000000	131.000000	0.000000	0.000000	71.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.0
25%	52.000000	1.000000	0.000000	120.000000	217.250000	0.000000	0.000000	125.000000	0.000000	0.600000	1.000000	0.000000	2.000000	0.0
50%	58.000000	1.000000	0.000000	130.000000	249.000000	0.000000	0.000000	142.000000	1.000000	1.400000	1.000000	1.000000	3.000000	0.0
75%	62.000000	1.000000	0.000000	144.750000	283.000000	0.000000	1.000000	156.000000	1.000000	2.500000	1.750000	2.000000	3.000000	0.0
max	77.000000	1.000000	3.000000	200.000000	409.000000	1.000000	2.000000	195.000000	1.000000	6.200000	2.000000	4.000000	3.000000	0.0



```
print("(Positive Patients ST depression): " + str(pos_data['oldpeak'].mean()))
print("(Negative Patients ST depression): " + str(neg_data['oldpeak'].mean()))


print("(Positive Patients thalach): " + str(pos_data['thalach'].mean()))
print("(Negative Patients thalach): " + str(neg_data['thalach'].mean()))
```

✓ 0.6s

```
(Positive Patients ST depression): 0.583030303030303
(Negative Patients ST depression): 1.5855072463768116
(Positive Patients thalach): 158.46666666666667
(Negative Patients thalach): 139.1014492753623
```

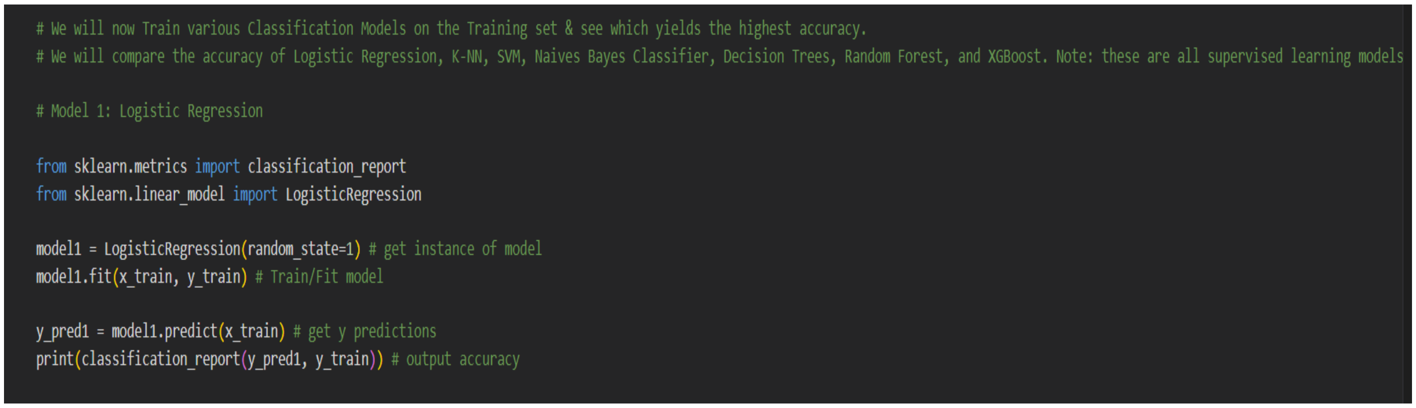


```
# Split: the dataset into the Training set and Test set

X = df.drop(columns='target', axis=1)
Y = df['target']

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 = 1)

from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
x_train = sc.fit_transform(x_train)
x_test = sc.transform(x_test)
```

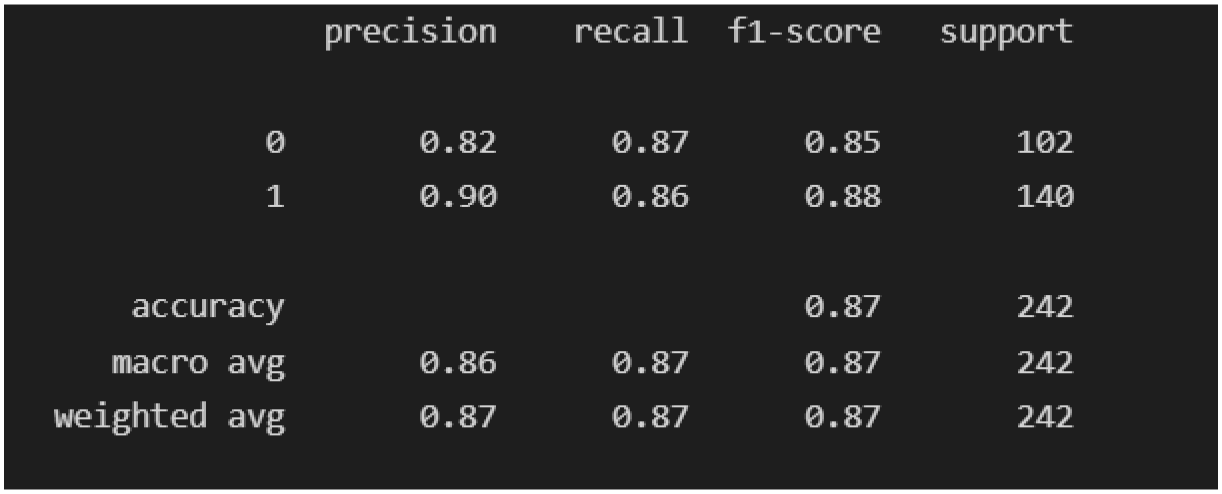
✓ 0.6s

LOGISTIC REGRESSION

Similar to linear regression, logistic regression is also used to estimate the relationship between a dependent variable and one or more independent variables, but it is used to make a prediction about a categorical variable versus a continuous one. Logistic regression predicts the output of a categorical dependent variable. Therefore the outcome must be a categorical or discrete value. It can be either Yes or no, 0 or 1, true or False, etc. but instead of giving the exact value as 0 and 1, it gives the probabilistic values which lie between 0 and 1.



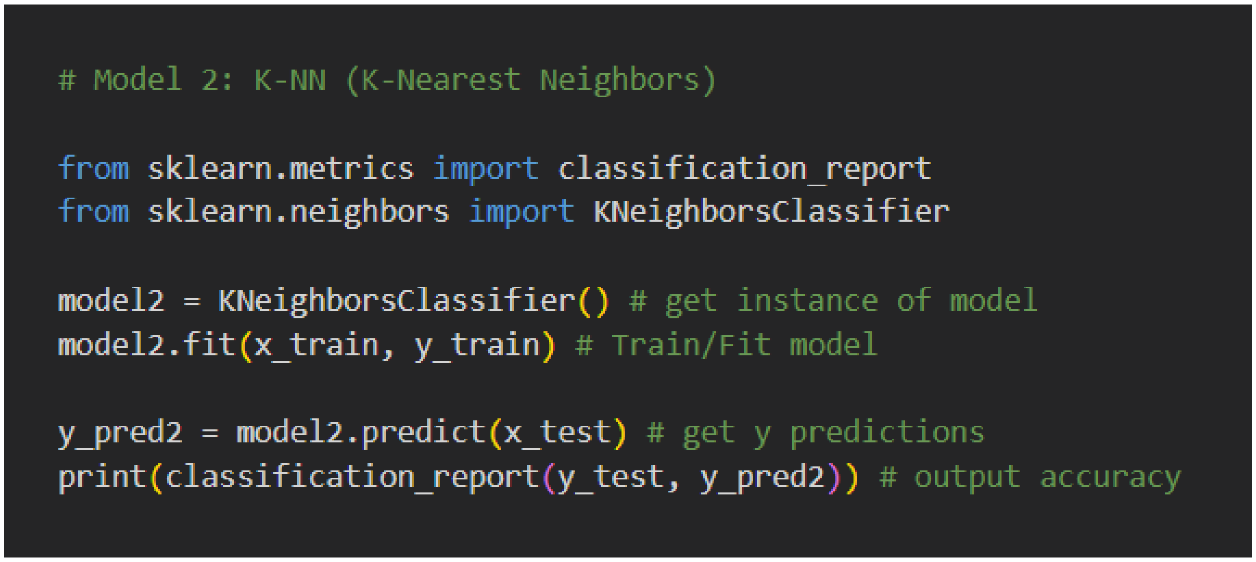
```
# We will now Train various Classification Models on the Training set & see which yields the highest accuracy.
# We will compare the accuracy of Logistic Regression, K-NN, SVM, Naives Bayes Classifier, Decision Trees, Random Forest, and XGBoost. Note: these are all supervised learning models

# Model 1: Logistic Regression

from sklearn.metrics import classification_report
from sklearn.linear_model import LogisticRegression

model1 = LogisticRegression(random_state=1) # get instance of model
model1.fit(x_train, y_train) # Train/Fit model

y_pred1 = model1.predict(x_train) # get y predictions
print(classification_report(y_pred1, y_train)) # output accuracy
```



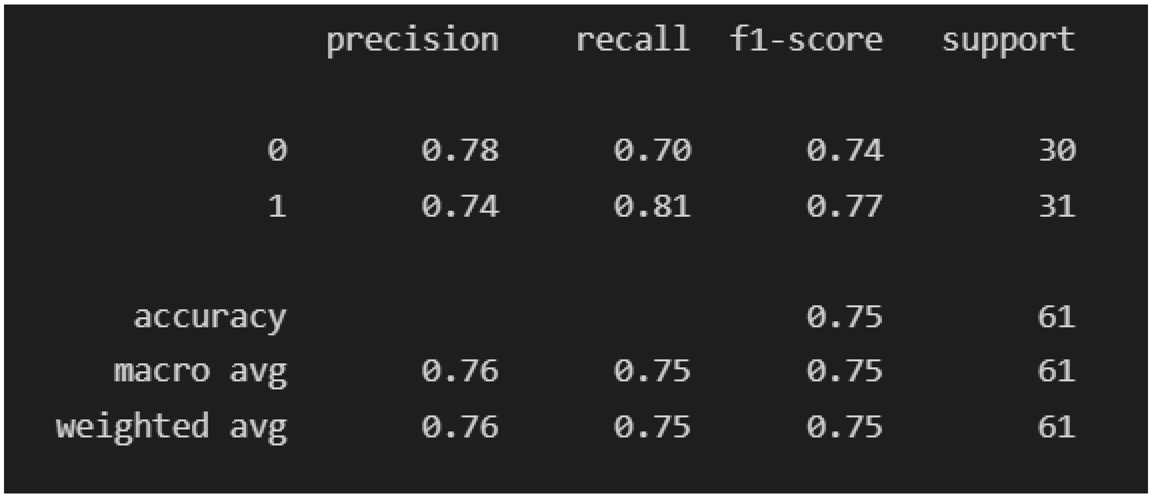
```
              precision    recall  f1-score   support

           0       0.82      0.87      0.85       102
           1       0.90      0.86      0.88       140

    accuracy                           0.87       242
   macro avg       0.86      0.87      0.87       242
weighted avg       0.87      0.87      0.87       242
```

K-NEAREST NEIGHBOUR

K-NN is a non-parametric algorithm, which means it does not make any assumption on underlying data. It is also called a lazy learner algorithm because it does not learn from the training set immediately instead it stores the dataset and at the time of classification, it performs an action on the dataset. KNN algorithm at the training phase just stores the dataset and when it gets new data, then it classifies that data into a category that is much similar to the new data.



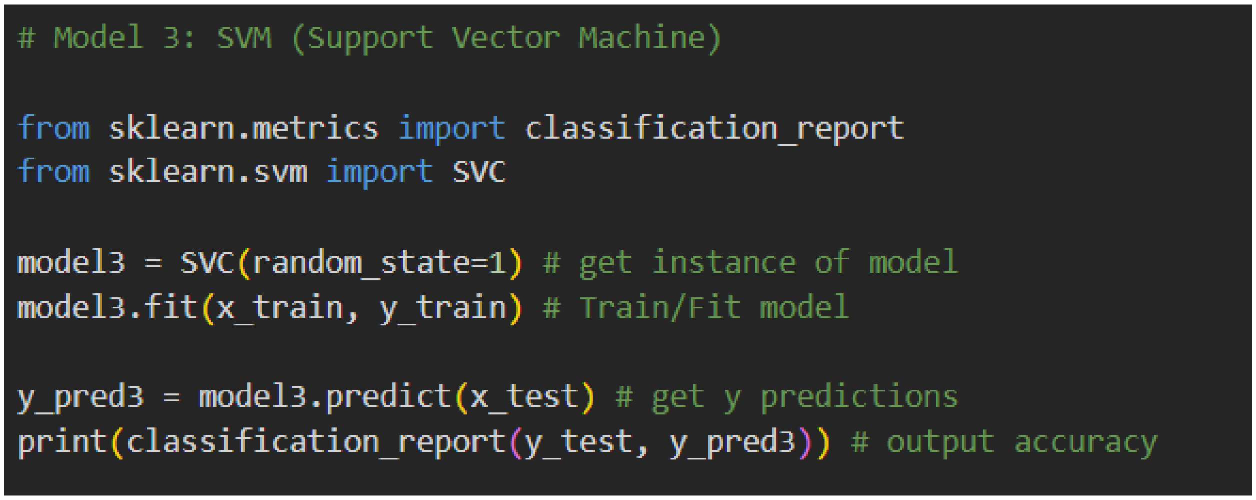
```
# Model 2: K-NN (K-Nearest Neighbors)

from sklearn.metrics import classification_report
from sklearn.neighbors import KNeighborsClassifier

model2 = KNeighborsClassifier() # get instance of model
model2.fit(x_train, y_train) # Train/Fit model

y_pred2 = model2.predict(x_test) # get y predictions
print(classification_report(y_test, y_pred2)) # output accuracy
```



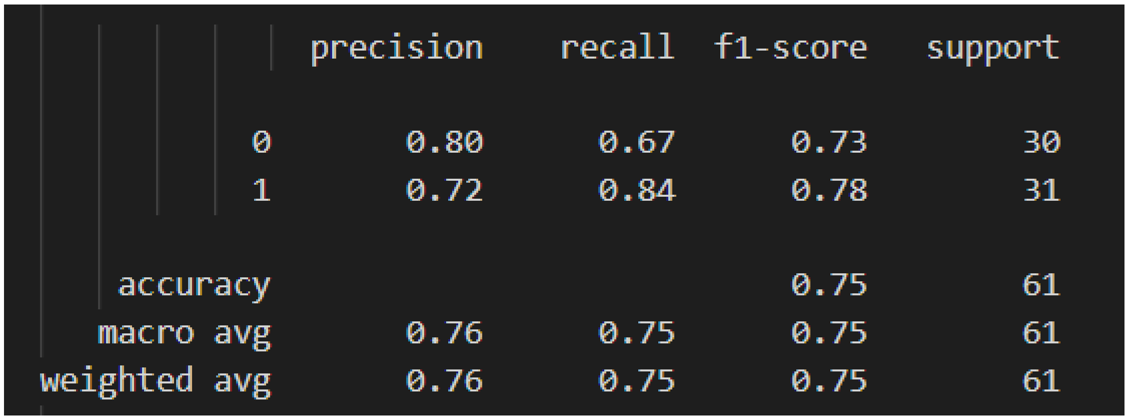
```
              precision    recall  f1-score   support

           0       0.78      0.70      0.74        30
           1       0.74      0.81      0.77        31

    accuracy                           0.75        61
   macro avg       0.76      0.75      0.75        61
weighted avg       0.76      0.75      0.75        61
```

SUPPORT VECTOR MACHINE

Support Vector Machine(SVM) is a supervised machine learning algorithm used for both classification and regression but its best suited for classification.
The objective of the SVM algorithm is to find a hyperplane in an N-dimensional space that distinctly classifies the data points.



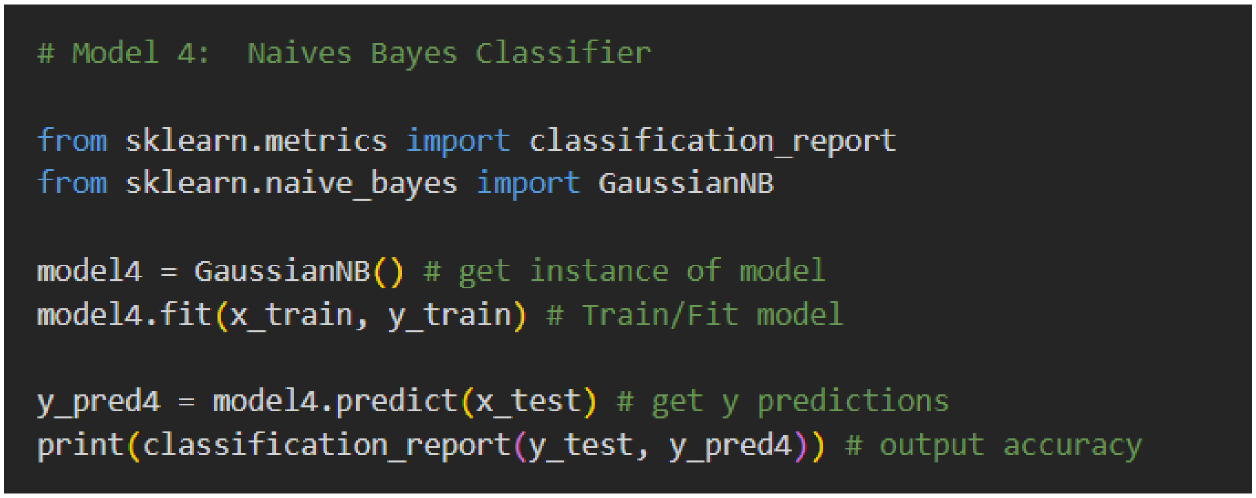
```
# Model 3: SVM (Support Vector Machine)

from sklearn.metrics import classification_report
from sklearn.svm import SVC

model3 = SVC(random_state=1) # get instance of model
model3.fit(x_train, y_train) # Train/Fit model

y_pred3 = model3.predict(x_test) # get y predictions
print(classification_report(y_test, y_pred3)) # output accuracy
```



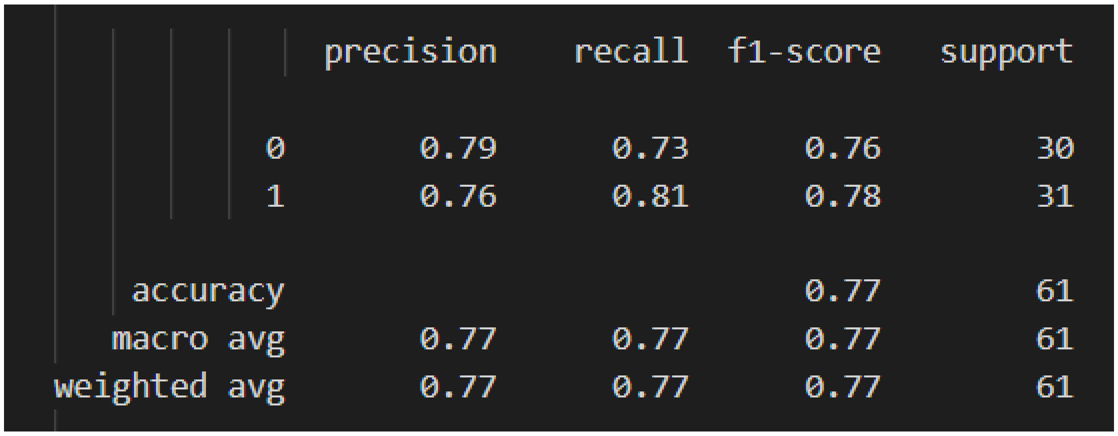
```
              precision    recall  f1-score   support

           0       0.80      0.67      0.73        30
           1       0.72      0.84      0.78        31

    accuracy                           0.75        61
   macro avg       0.76      0.75      0.75        61
weighted avg       0.76      0.75      0.75        61
```

NAIVES BAYES CLASSIFIER

Naïve Bayes is one of the fast and easy ML algorithms to predict a class of datasets.
It can be used for Binary as well as Multi-class Classifications. It performs well in Multi-class predictions as compared to the other Algorithms.
It is the most popular choice for text classification problems.



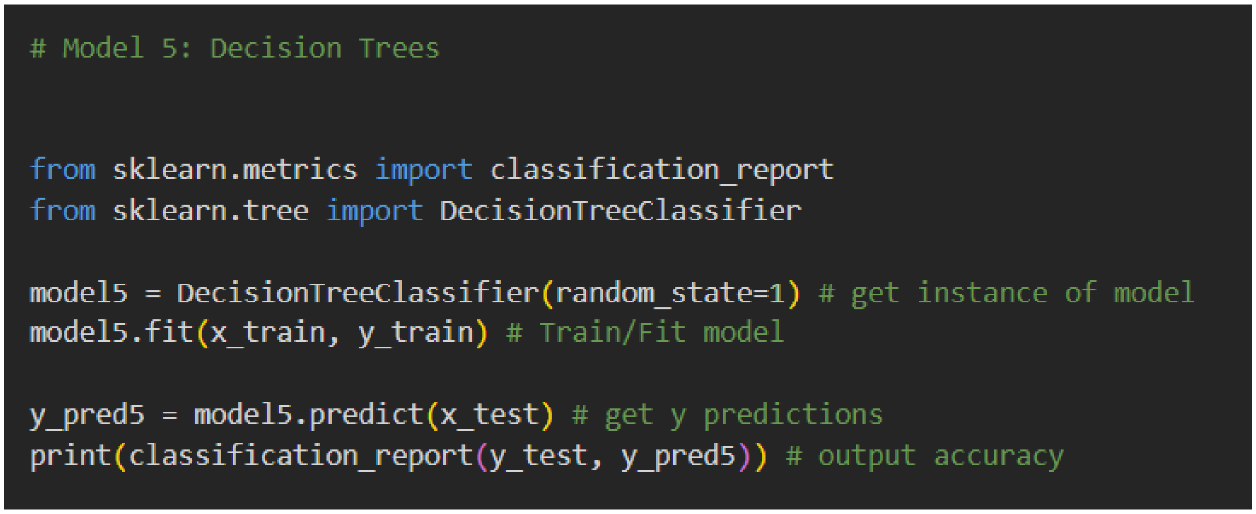
```
# Model 4:  Naives Bayes Classifier

from sklearn.metrics import classification_report
from sklearn.naive_bayes import GaussianNB

model4 = GaussianNB() # get instance of model
model4.fit(x_train, y_train) # Train/Fit model

y_pred4 = model4.predict(x_test) # get y predictions
print(classification_report(y_test, y_pred4)) # output accuracy
```



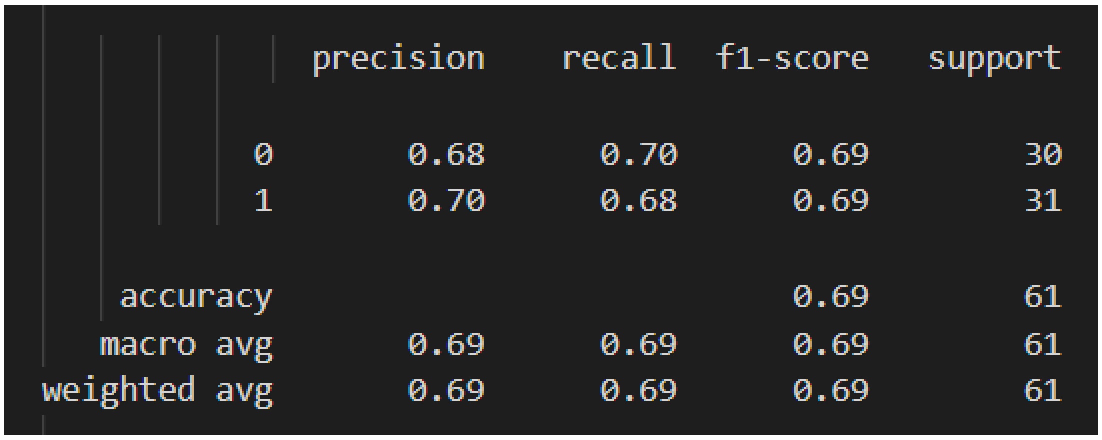
```
              precision    recall  f1-score   support

           0       0.79      0.73      0.76        30
           1       0.76      0.81      0.78        31

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

DECISION TREE

Decision trees are used to calculate the potential success of different series of decisions made to achieve a specific goal. The concept of a decision tree existed long before machine learning, as it can be used to manually model operational decisions like a flowchart. They are commonly taught and utilized in business, economics, and operation management sectors as an approach to analyzing organizational decision-making. The Decision Tree Algorithm belongs to the family of supervised machine learning algorithms. It can be used for both classification problems as well as for regression problems. The goal of this algorithm is to create a model that predicts the value of a target variable, for which the decision tree uses the tree representation to solve the problem in which the leaf node corresponds to a class label and attributes are represented on the internal node of the tree.



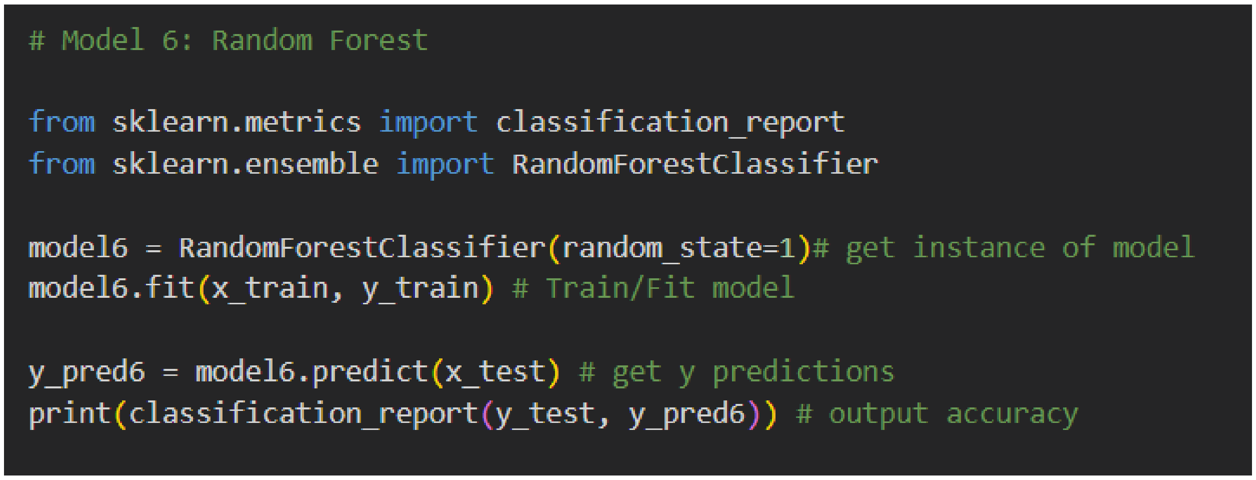
```
# Model 5: Decision Trees


from sklearn.metrics import classification_report
from sklearn.tree import DecisionTreeClassifier

model5 = DecisionTreeClassifier(random_state=1) # get instance of model
model5.fit(x_train, y_train) # Train/Fit model

y_pred5 = model5.predict(x_test) # get y predictions
print(classification_report(y_test, y_pred5)) # output accuracy
```



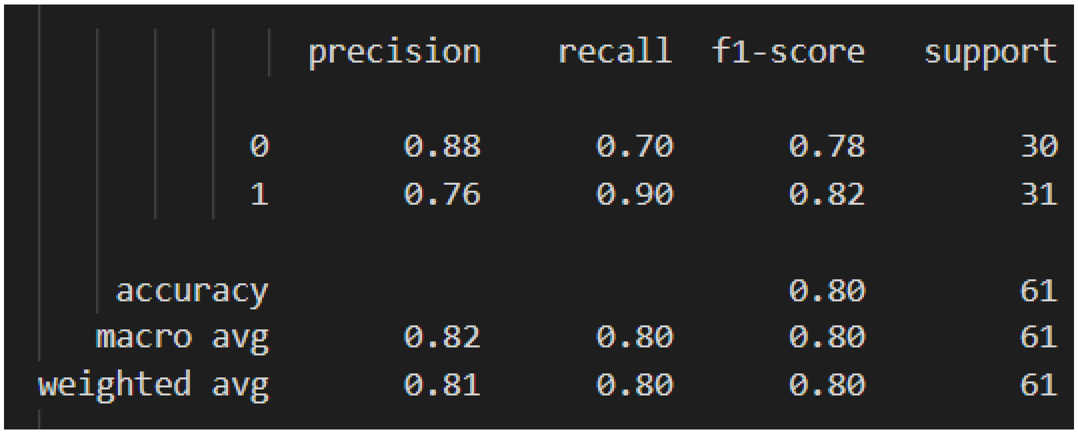
```
              precision    recall  f1-score   support

           0       0.68      0.70      0.69        30
           1       0.70      0.68      0.69        31

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

RANDOM FOREST

Random Forest is a powerful and versatile supervised machine learning algorithm that grows and combines multiple decision trees to create a “forest.” It can be used for both classification and regression problems in R and Python As the name suggests, "Random Forest is a classifier that contains a number of decision trees on various subsets of the given dataset and takes the average to improve the predictive accuracy of that dataset." Instead of relying on one decision tree, the random forest takes the prediction from each tree, and based on the majority votes of predictions, it predicts the final output. The greater number of trees in the forest leads to higher accuracy and prevents the problem of overfitting.



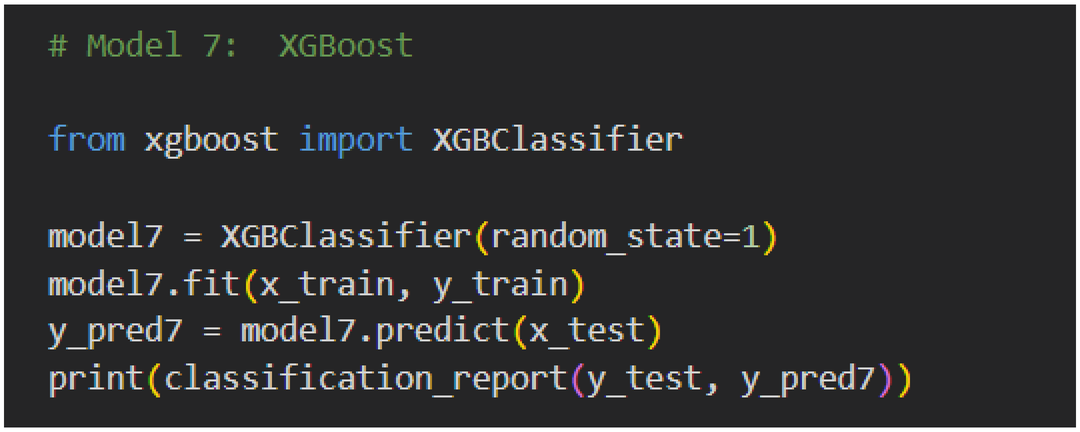
```
# Model 6: Random Forest

from sklearn.metrics import classification_report
from sklearn.ensemble import RandomForestClassifier

model6 = RandomForestClassifier(random_state=1)# get instance of model
model6.fit(x_train, y_train) # Train/Fit model

y_pred6 = model6.predict(x_test) # get y predictions
print(classification_report(y_test, y_pred6)) # output accuracy
```



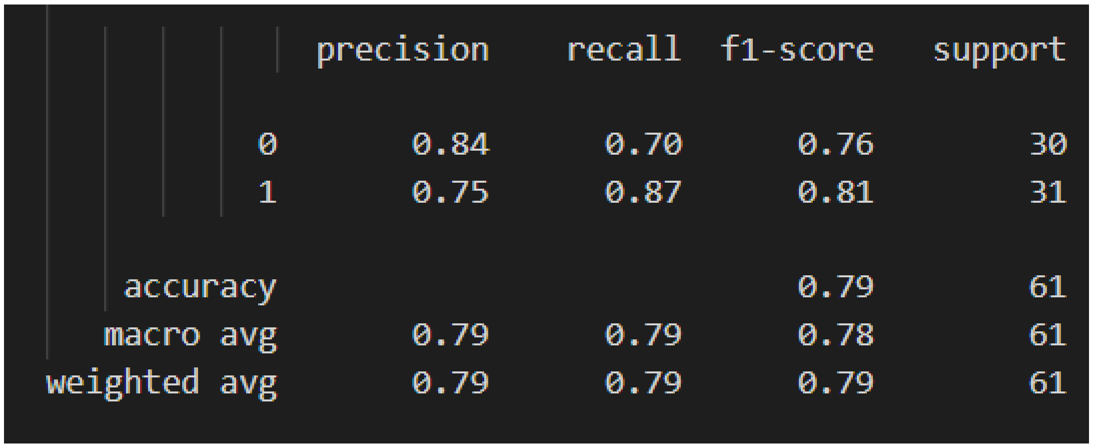
```
              precision    recall  f1-score   support

           0       0.88      0.70      0.78        30
           1       0.76      0.90      0.82        31

    accuracy                           0.80        61
   macro avg       0.82      0.80      0.80        61
weighted avg       0.81      0.80      0.80        61
```

XGBOOST

The GPU-accelerated XGBoost algorithm makes use of fast parallel prefix sum operations to scan through all possible splits, as well as parallel radix sorting to repartition data. It builds a decision tree for a given boosting iteration, one level at a time, processing the entire dataset concurrently on the GPU.



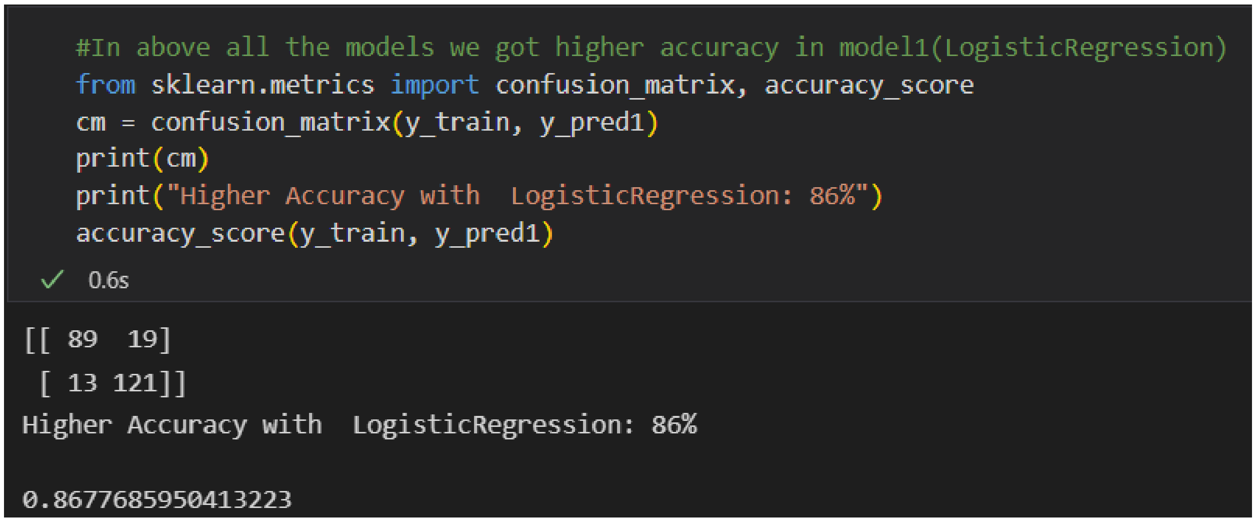
```
# Model 7:  XGBoost

from xgboost import XGBClassifier

model7 = XGBClassifier(random_state=1)
model7.fit(x_train, y_train)
y_pred7 = model7.predict(x_test)
print(classification_report(y_test, y_pred7))
```



```
              precision    recall  f1-score   support

           0       0.84      0.70      0.76        30
           1       0.75      0.87      0.81        31

    accuracy                           0.79        61
   macro avg       0.79      0.79      0.78        61
weighted avg       0.79      0.79      0.79        61
```

CONFUSION MATRIX

Confusion Matrix is a useful machine learning method that allows you to measure Recall, Precision, Accuracy, and AUC-ROC curve. It is used to determine the performance of the classification models for a given set of test data. It can only be determined if the true values for test data are known. The F1 score is balancing precision and recall on the positive class while the accuracy score given by the confusion matrix looks at correctly classified observations both positive and negative.



```
#In above all the models we got higher accuracy in model1(LogisticRegression)
from sklearn.metrics import confusion_matrix, accuracy_score
cm = confusion_matrix(y_train, y_pred1)
print(cm)
print("Higher Accuracy with  LogisticRegression: 86%")
accuracy_score(y_train, y_pred1)
```

✓ 0.6s

```
[[ 89  19]
 [ 13 121]]
Higher Accuracy with  LogisticRegression: 86%

0.8677685950413223
```

RESULT

a). Logistic Regression gives us an accuracy of 86%.

b). K-Nearest Neighbours gives us an accuracy of 75%.

c). Support Vector Machine gives us an accuracy of 75%.

d). Naives Bayes Classifier gives us an accuracy of 77%.

e). Decision Trees gives us an accuracy of 69%.

f). Random Forest gives us an accuracy of 80%.

g). XGBoost gives us an accuracy of 79%.

The highest accuracy for this data is achieved by Logistic Regression which is 86%.

CONCLUSION

The researchers develop prediction models for heart diagnosis using pattern recognition and data mining techniques.
In order to forecast the outcome, the computer learns patterns from the known dataset and applies them to the unknown dataset.
In order to demonstrate the algorithm's usefulness in early disease prediction, this research focuses not only on improving the accuracy of weak classification algorithms but also on how to implement the method using a medical dataset.
These Trained and tested models can be used in a commercial way for business strategy for the purpose of prediction and analysis purpose in the field of medicine.

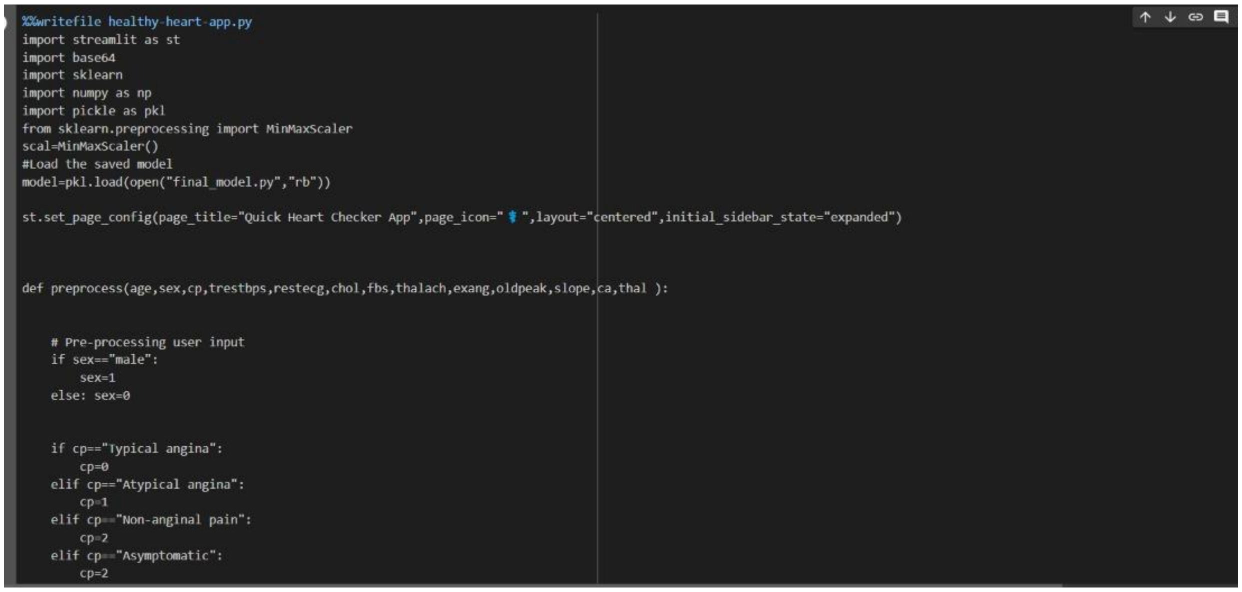
REFERENCES:

1. kaggle (https://www.kaggle.com/)
2. geeks for geeks (https://www.geeksforgeeks.org/)
3. stackoverflow (https://stackoverflow.com/)
4. sklearn (https://scikit-learn.org/stable/index.html)
5. steamlit (https://streamlit.io/)
6. youtube (https://www.youtube.com/)
7. Wikipedia (https://www.wikipedia.org/)
8. w3scools (https://www.w3schools.com/html/default.asp)
9. tutorialspoint (https://www.tutorialspoint.com/tutorialslibrary.htm)
10. towards data science (https://towardsdatascience.com/)

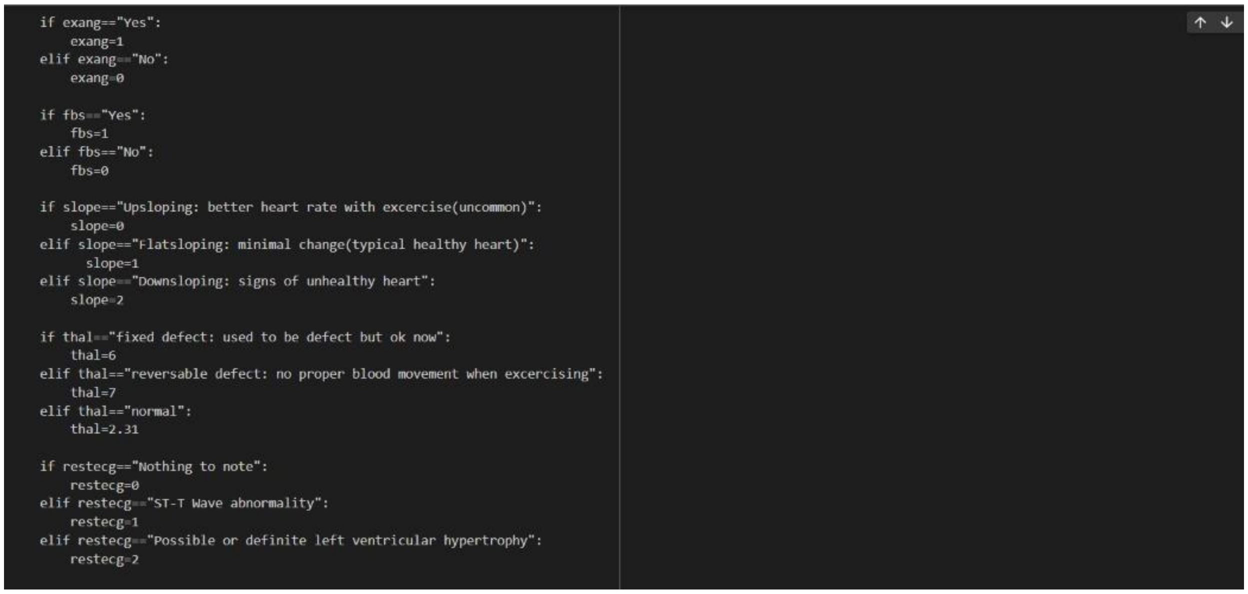
CODING LINK

https://colab.research.google.com/drive/10smFKgYbE0rfezbYKWdEDkeVOJeDgHOu

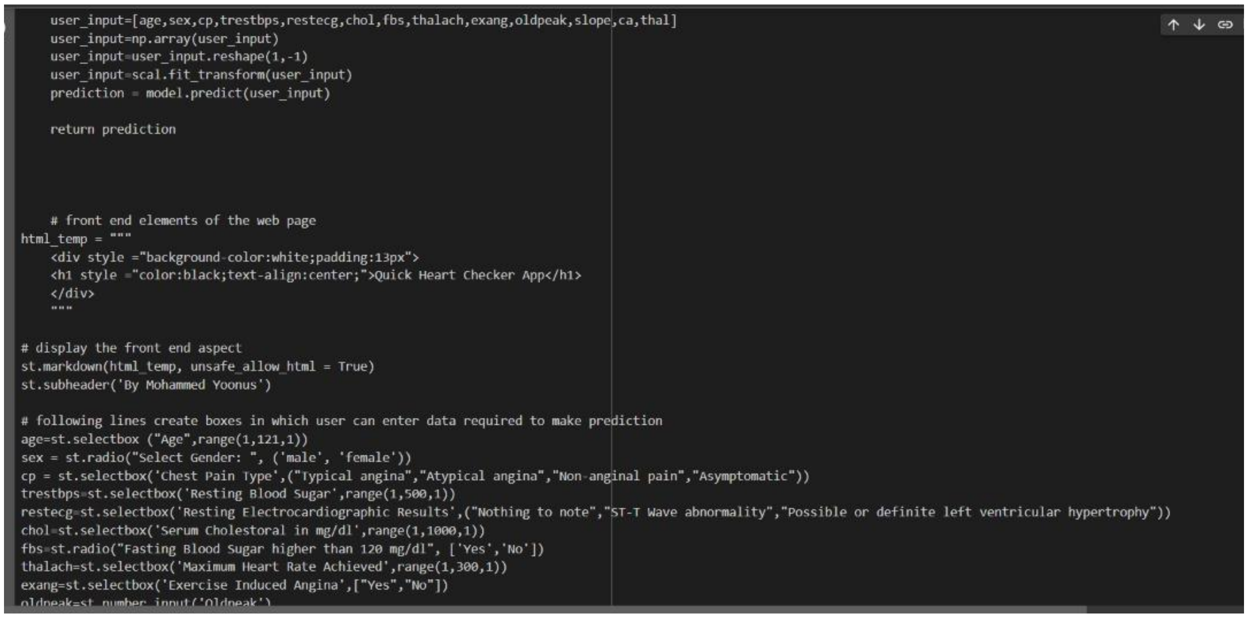
CODING FOR WEB APP



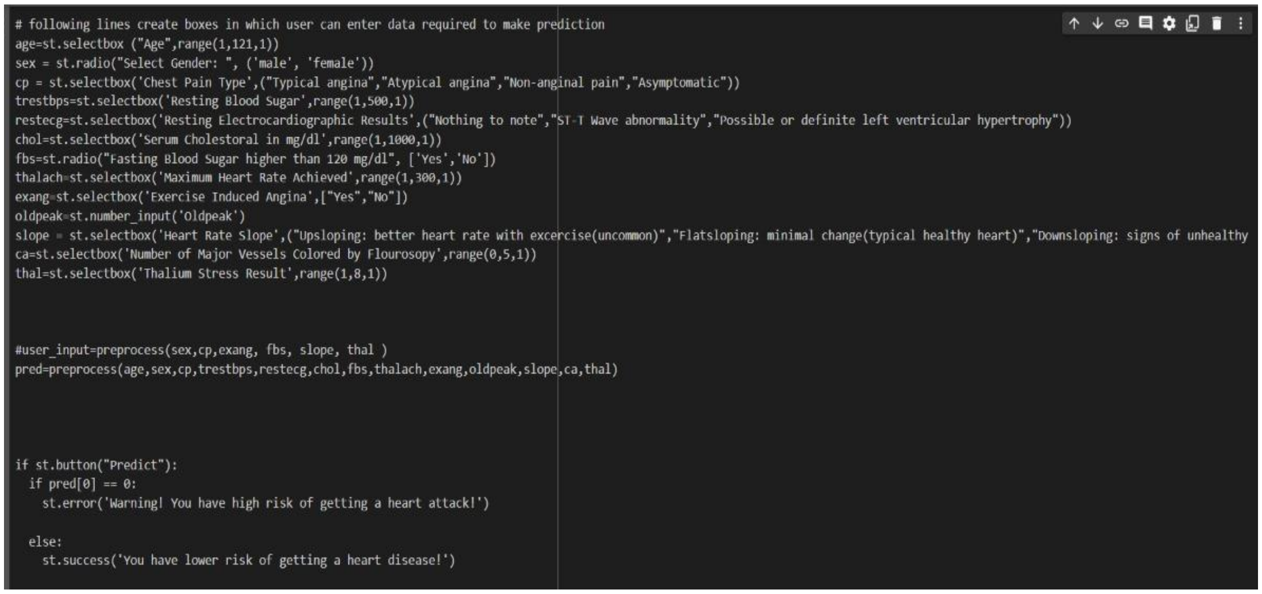
```
%%writefile healthy-heart-app.py
import streamlit as st
import base64
import sklearn
import numpy as np
import pickle as pkl
from sklearn.preprocessing import MinMaxScaler
scal=MinMaxScaler()
#Load the saved model
model=pkl.load(open("final_model.py","rb"))

st.set_page_config(page_title="Quick Heart Checker App",page_icon=" ⚕ ",layout="centered",initial_sidebar_state="expanded")


def preprocess(age,sex,cp,trestbps,restecg,chol,fbs,thalach,exang,oldpeak,slope,ca,thal ):


    # Pre-processing user input
    if sex=="male":
        sex=1
    else: sex=0


    if cp=="Typical angina":
        cp=0
    elif cp=="Atypical angina":
        cp=1
    elif cp=="Non-anginal pain":
        cp=2
    elif cp=="Asymptomatic":
        cp=2
```



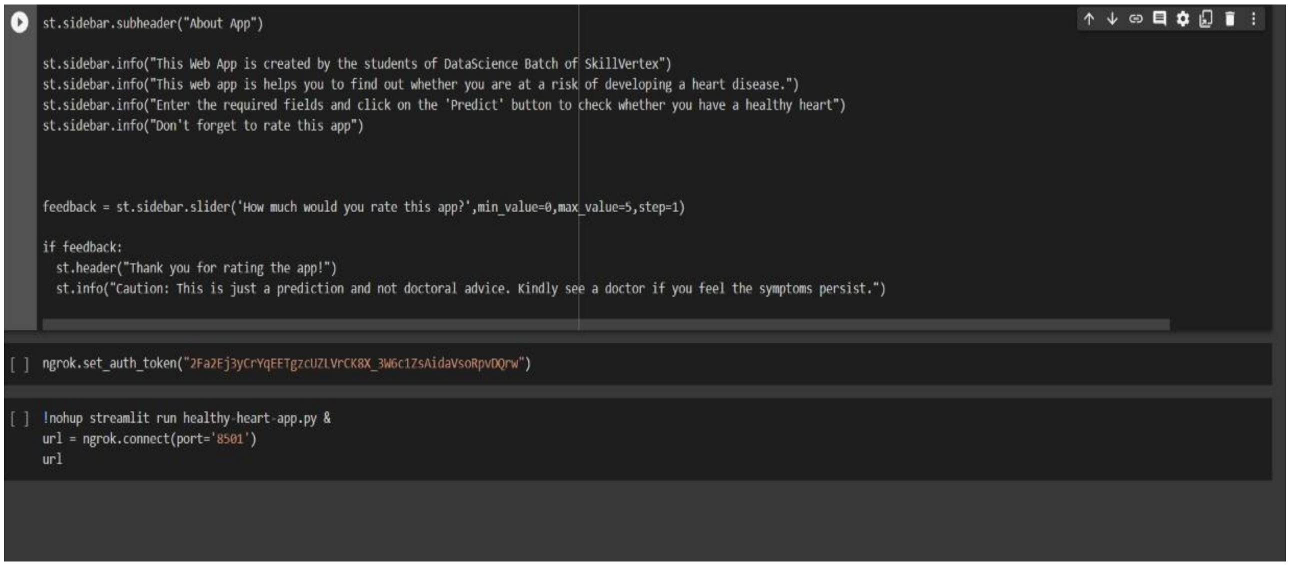
```
    if exang=="Yes":
        exang=1
    elif exang=="No":
        exang=0

    if fbs=="Yes":
        fbs=1
    elif fbs=="No":
        fbs=0

    if slope=="Upsloping: better heart rate with excercise(uncommon)":
        slope=0
    elif slope=="Flatsloping: minimal change(typical healthy heart)":
          slope=1
    elif slope=="Downsloping: signs of unhealthy heart":
        slope=2

    if thal=="fixed defect: used to be defect but ok now":
        thal=6
    elif thal=="reversable defect: no proper blood movement when excercising":
        thal=7
    elif thal=="normal":
        thal=2.31

    if restecg=="Nothing to note":
        restecg=0
    elif restecg=="ST-T Wave abnormality":
        restecg=1
    elif restecg=="Possible or definite left ventricular hypertrophy":
        restecg=2
```



```
    user_input=[age,sex,cp,trestbps,restecg,chol,fbs,thalach,exang,oldpeak,slope,ca,thal]
    user_input=np.array(user_input)
    user_input=user_input.reshape(1,-1)
    user_input=scal.fit_transform(user_input)
    prediction = model.predict(user_input)

    return prediction



    # front end elements of the web page
html_temp = """
    <div style ="background-color:white;padding:13px">
    <h1 style ="color:black;text-align:center;">Quick Heart Checker App</h1>
    </div>
    """

# display the front end aspect
st.markdown(html_temp, unsafe_allow_html = True)
st.subheader('By Mohammed Yoonus')

# following lines create boxes in which user can enter data required to make prediction
age=st.selectbox ("Age",range(1,121,1))
sex = st.radio("Select Gender: ", ('male', 'female'))
cp = st.selectbox('Chest Pain Type',("Typical angina","Atypical angina","Non-anginal pain","Asymptomatic"))
trestbps=st.selectbox('Resting Blood Sugar',range(1,500,1))
restecg=st.selectbox('Resting Electrocardiographic Results',("Nothing to note","ST-T Wave abnormality","Possible or definite left ventricular hypertrophy"))
chol=st.selectbox('Serum Cholestoral in mg/dl',range(1,1000,1))
fbs=st.radio("Fasting Blood Sugar higher than 120 mg/dl", ['Yes','No'])
thalach=st.selectbox('Maximum Heart Rate Achieved',range(1,300,1))
exang=st.selectbox('Exercise Induced Angina',["Yes","No"])
oldpeak=st.number_input('Oldpeak')
```



```
# following lines create boxes in which user can enter data required to make prediction
age=st.selectbox ("Age",range(1,121,1))
sex = st.radio("Select Gender: ", ('male', 'female'))
cp = st.selectbox('Chest Pain Type',("Typical angina","Atypical angina","Non-anginal pain","Asymptomatic"))
trestbps=st.selectbox('Resting Blood Sugar',range(1,500,1))
restecg=st.selectbox('Resting Electrocardiographic Results',("Nothing to note","ST-T Wave abnormality","Possible or definite left ventricular hypertrophy"))
chol=st.selectbox('Serum Cholestoral in mg/dl',range(1,1000,1))
fbs=st.radio("Fasting Blood Sugar higher than 120 mg/dl", ['Yes','No'])
thalach=st.selectbox('Maximum Heart Rate Achieved',range(1,300,1))
exang=st.selectbox('Exercise Induced Angina',["Yes","No"])
oldpeak=st.number_input('Oldpeak')
slope = st.selectbox('Heart Rate Slope',("Upsloping: better heart rate with excercise(uncommon)","Flatsloping: minimal change(typical healthy heart)","Downsloping: signs of unhealthy
ca=st.selectbox('Number of Major Vessels Colored by Flourosopy',range(0,5,1))
thal=st.selectbox('Thalium Stress Result',range(1,8,1))



#user_input=preprocess(sex,cp,exang, fbs, slope, thal )
pred=preprocess(age,sex,cp,trestbps,restecg,chol,fbs,thalach,exang,oldpeak,slope,ca,thal)




if st.button("Predict"):
  if pred[0] == 0:
    st.error('Warning! You have high risk of getting a heart attack!')

  else:
    st.success('You have lower risk of getting a heart disease!')
```

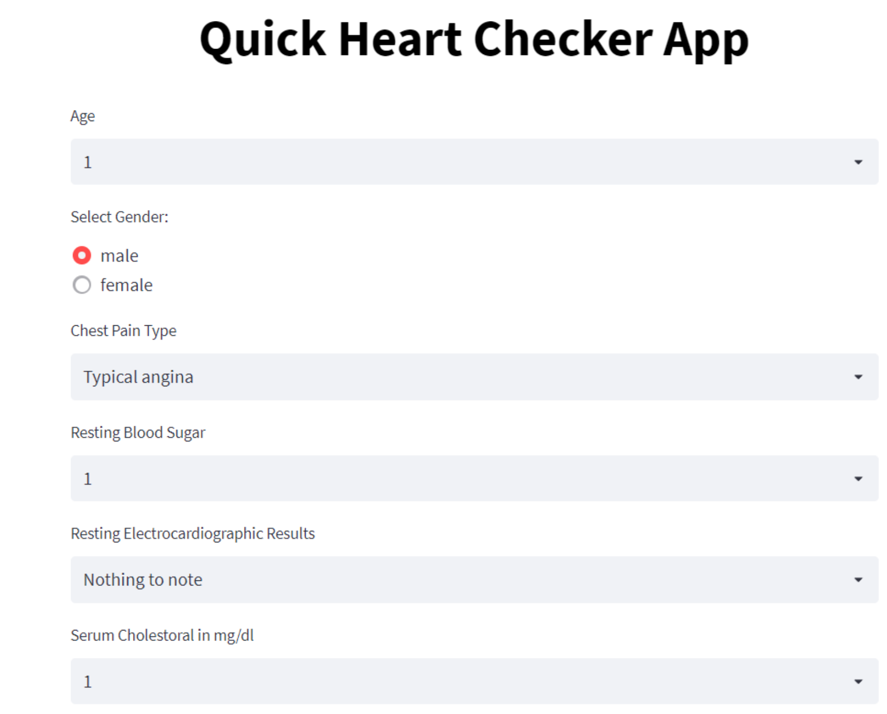


```
st.sidebar.subheader("About App")

st.sidebar.info("This Web App is created by the students of DataScience Batch of SkillVertex")
st.sidebar.info("This web app is helps you to find out whether you are at a risk of developing a heart disease.")
st.sidebar.info("Enter the required fields and click on the 'Predict' button to check whether you have a healthy heart")
st.sidebar.info("Don't forget to rate this app")



feedback = st.sidebar.slider('How much would you rate this app?',min_value=0,max_value=5,step=1)

if feedback:
  st.header("Thank you for rating the app!")
  st.info("Caution: This is just a prediction and not doctoral advice. Kindly see a doctor if you feel the symptoms persist.")
```

```
[ ] ngrok.set_auth_token("2Fa2Ej3yCrYqEETgzcUZLVrCK8X_3W6c1ZsAidaVsoRpvDQrw")
```

```
[ ] !nohup streamlit run healthy-heart-app.py &
    url = ngrok.connect(port='8501')
    url
```

WORKING OF WEB APP



Quick Heart Checker App

Age

1

Select Gender:

[✓ male] female

Chest Pain Type

Typical angina

Resting Blood Sugar

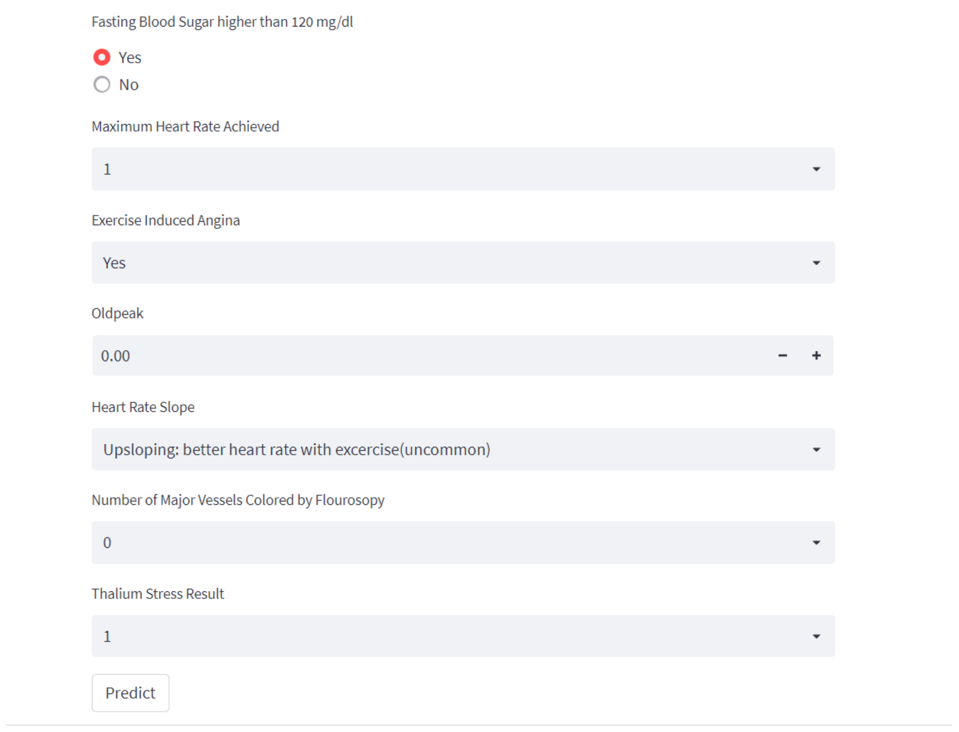
1

Resting Electrocardiographic Results

Nothing to note

Serum Cholestoral in mg/dl

1



Fasting Blood Sugar higher than 120 mg/dl

[✓ Yes] No

Maximum Heart Rate Achieved

1

Exercise Induced Angina

Yes

Oldpeak

0.00

Heart Rate Slope

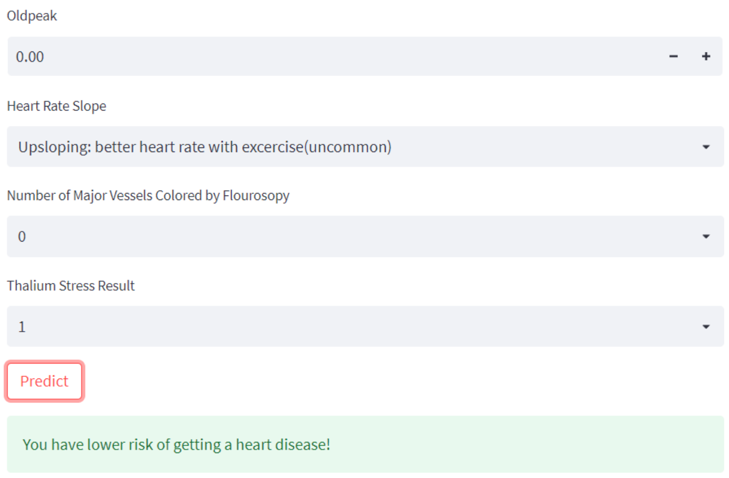
Upsloping: better heart rate with excercise(uncommon)

Number of Major Vessels Colored by Flourosopy

0

Thalium Stress Result

1


Oldpeak

0.00

Heart Rate Slope

Upsloping: better heart rate with excercise(uncommon)

Number of Major Vessels Colored by Flourosopy

0

Thalium Stress Result

1

You have lower risk of getting a heart disease!