Tech Saksham

Capstone Project Report

ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING FUNDAMENTALS

“Heart Disease Prediction using Logistic Regression”

“UNIVERSITY COLLEGE OF ENGINEERING (BIT CAMPUS) TIRUCHIRAPALLI”

NM ID	NAME
au810021214031	SHREYA S R

Trainer Name

P. Ramar Bose

Master Trainer

ABSTRACT

Cardiovascular diseases (CVDs) continue to cause a large number of deaths globally, with heart attacks representing a substantial portion of CVD-related fatalities, as per the World Health Organization. This study centers on predicting CVD risk through logistic regression analysis. Leveraging concepts such as artificial intelligence and logistic regression, we aim to identify individuals at heightened risk of CVD development. By employing logistic regression techniques and considering diverse factors associated with CVDs, our goal is to determine the probability of individuals being affected by CVD based on various risk indicators. Through the analysis of pertinent medical data and the application of predictive modeling, we endeavor to contribute to early detection and prevention strategies for heart disease, ultimately striving to mitigate CVD mortality rates.

INDEX

CHAPTER 1

INTRODUCTION

1.1 Problem Statement

A heart attack is caused when a vessel of the heart gets blocked and therefore causes death of cardiomyocytes and cardiac muscles in the area. Although anyone can get a heart attack, some factors can increase the risk of a heart attack such as age, gender, cholesterol, weight, Family histories, etc. However, manual analysis of this risk can be difficult as there are far too many factors and the results cannot be given in a precise and consistent manner. Being able to correctly analyze and predict heart diseases in a timely manner can save the lives of many people. Furthermore, the presentation of symptoms of heart diseases can vary a lot based on gender and age.

1.2 Proposed Solution

The proposed solution involves using the **Google collaboration** website. Logistic Regression is recommended for Heart Disease Prediction, which can be involved in data preprocessing, feature selection, and model building. Utilizing the Logistic Regression model, it becomes feasible to evaluate an individual's likelihood of developing cardiovascular disease. This model aids healthcare providers in pinpointing high-risk patients, enabling the implementation of tailored interventions and lifestyle adjustments to mitigate the consequences associated with CVD. The critical steps encompass data preparation, feature selection, model training, and assessment. Techniques like correlation analysis and domain expertise contribute to identifying the most informative predictors. Evaluation of model performance relies on metrics such as accuracy and precision.

1.3 Features

- **Feature Relevance**: Logistic regression selects pertinent features like age, blood pressure, and cholesterol levels, streamlining the identification of key risk factors for heart disease prediction.

- **Probabilistic Output**: Providing a probability estimate of heart disease occurrence, logistic regression offers a clear indication of risk, aiding clinicians in decision-making regarding further tests or preventive actions.
- **Efficient Scalability**: Logistic regression's computational efficiency makes it apt for analyzing extensive healthcare datasets, facilitating real-time risk assessment and intervention for individuals prone to heart disease.

1.4 Advantages

- **Early Intervention**: Predictive models enable early identification of heart disease risk, allowing for timely preventive measures to be implemented.
- **Personalized Care**: By considering diverse factors like medical history and lifestyle, predictive models facilitate tailored interventions, optimizing treatment outcomes.
- **Cost Savings**: Early detection and intervention reduce the need for expensive procedures, resulting in significant cost savings and more efficient resource allocation within healthcare systems.

1.5 Scope

Predicting heart disease encompasses a multifaceted approach, integrating traditional risk factors such as age, family history, and lifestyle choices with advanced medical tests like lipid profiles and genetic predisposition assessments. Emerging technologies, including machine learning and artificial intelligence, offer new avenues for predictive modeling, analyzing vast datasets to identify subtle patterns indicative of heart disease risk. Biomarkers and imaging techniques provide additional insight, while behavioral and lifestyle factors play a pivotal role in prevention. By leveraging predictive analytics, healthcare providers can tailor interventions and population-level strategies, ultimately aiming to mitigate the burden of heart disease through early detection and targeted interventions. As research progresses and technology evolves, the scope of prediction expands, promising more accurate risk assessments and improved outcomes for individuals and populations alike.

CHAPTER 2

SERVICES AND TOOLS REQUIRED

2.1 Services Used

Services for Data Collection and Storage: We utilize healthcare databases such as MIMIC-III, NHANES, or the Framingham Heart Study dataset, housing anonymized patient data pertinent to heart disease. For this project, Kaggle was used.

Services for Data Processing: We employ Python libraries including Pandas, NumPy, and scikit-learn for comprehensive data cleaning, preprocessing, and feature engineering tasks. Visualize data distributions, correlations, and trends using Matplotlib in conjunction with Seaborn.

Services for Machine Learning: We harness machine learning frameworks like Google Colaboratory, Jupyter, scikit-learn, or PyTorch to implement and train Logistic Regression models on available datasets, facilitating robust predictive analytics for heart disease assessment.

2.2 Tools and Software used

- **Data collection and storage:** Kaggle was used, and the heart disease prediction dataset was downloaded.

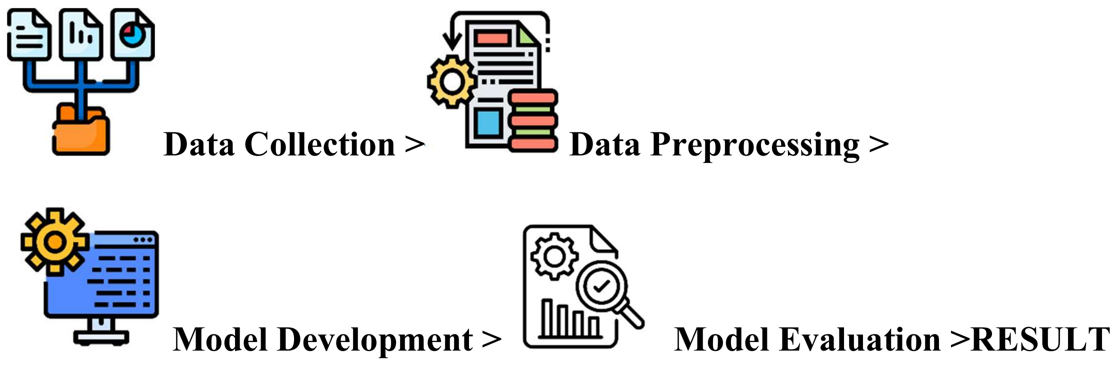
- **Data Processing:** Various python libraries including NumPy, matplotlib, SciPy, Statsmodel API etc. are used

- **Model Evaluation and interpretation**: Google collab software is used to run the program and predict the results.

CHAPTER 3

PROJECT ARCHITECTURE

3.1 Architecture



An overview of the project architecture is given below:

- **Data Collection**: Gather a comprehensive dataset encompassing demographic details, lifestyle behaviors (e.g., smoking, diet, exercise), clinical metrics (e.g., blood pressure, cholesterol levels, family history of CVD), and outcome data (presence or absence of CVD).
- **Data Preprocessing**: Conduct thorough data cleanup, including handling missing values, encoding categorical variables, and normalizing numerical attributes.
- **Exploratory Data Analysis (EDA):** Perform in-depth exploration of the dataset to uncover trends, correlations, and outliers that may impact CVD risk assessment.
- **Model Development**: Employ logistic regression to construct a predictive model capable of estimating the probability of CVD occurrence based on input features.
- **Model Evaluation**: Assess the performance of the logistic regression model using relevant evaluation metrics, such as accuracy and precision, to gauge its effectiveness in predicting CVD risk.

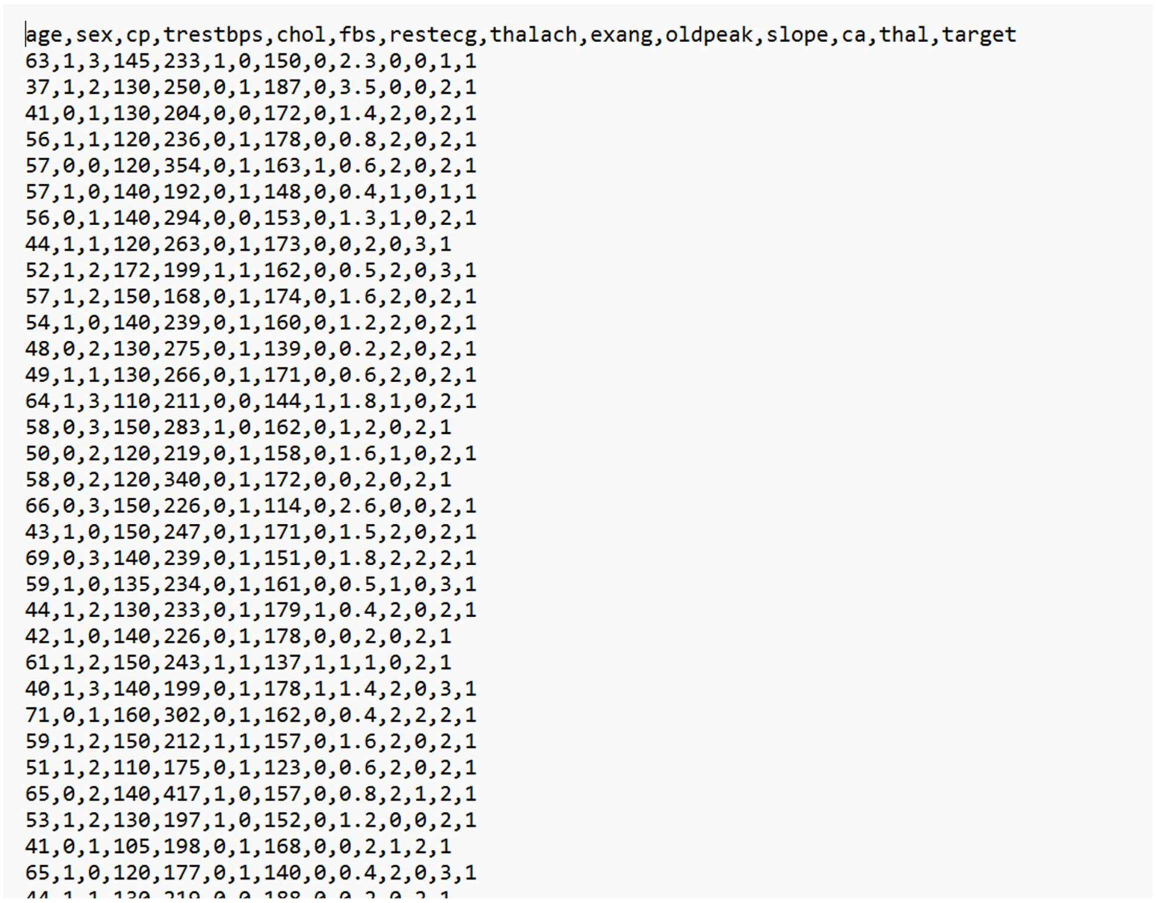
Interpretation and Insights: Analyze the logistic regression model coefficients to gain insights into the influence of different risk factors on CVD risk, offering actionable conclusions for preventive healthcare strategies.

This architectural framework provides valuable insights into the predictive capabilities of logistic regression in identifying individuals at heightened risk of heart disease. Moreover, the developed model holds promise for facilitating early intervention and personalized risk assessment by healthcare practitioners, ultimately contributing to the mitigation of CVD-related morbidity and mortality rates of individual afflicted by these diseases.

CHAPTER 4

MODELING AND PROJECT OUTCOME

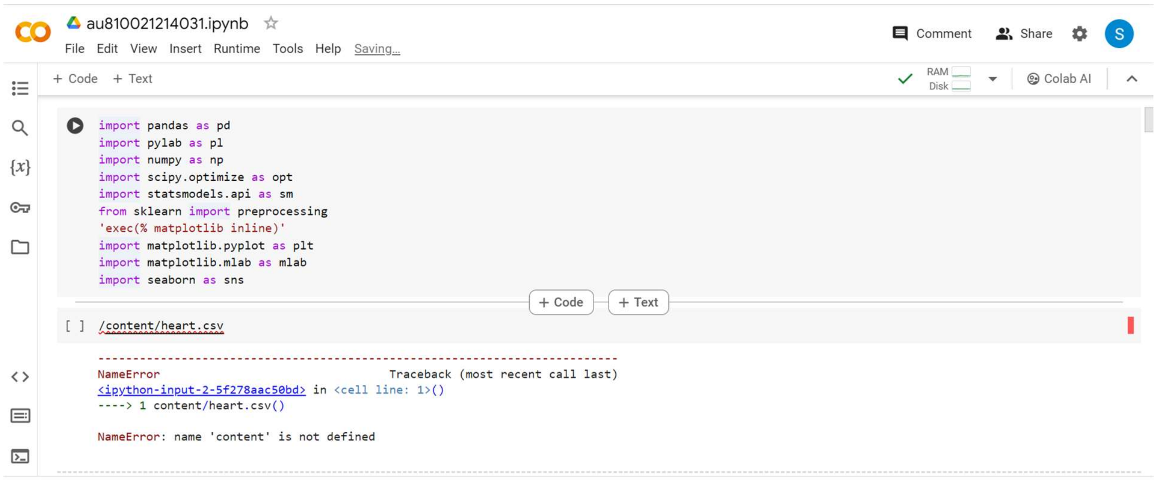
1. The datasets are retrieved from the Kaggle (link: https://www.kaggle.com/datasets/johnsmith88/heart-disease-dataset) platform and analyzed



```
age,sex,cp,trestbps,chol,fbs,restecg,thalach,exang,oldpeak,slope,ca,thal,target
63,1,3,145,233,1,0,150,0,2.3,0,0,1,1
37,1,2,130,250,0,1,187,0,3.5,0,0,2,1
41,0,1,130,204,0,0,172,0,1.4,2,0,2,1
56,1,1,120,236,0,1,178,0,0.8,2,0,2,1
57,0,0,120,354,0,1,163,1,0.6,2,0,2,1
57,1,0,140,192,0,1,148,0,0.4,1,0,1,1
56,0,1,140,294,0,0,153,0,1.3,1,0,2,1
44,1,1,120,263,0,1,173,0,0,2,0,3,1
52,1,2,172,199,1,1,162,0,0.5,2,0,3,1
57,1,2,150,168,0,1,174,0,1.6,2,0,2,1
54,1,0,140,239,0,1,160,0,1.2,2,0,2,1
48,0,2,130,275,0,1,139,0,0.2,2,0,2,1
49,1,1,130,266,0,1,171,0,0.6,2,0,2,1
64,1,3,110,211,0,0,144,1,1.8,1,0,2,1
58,0,3,150,283,1,0,162,0,1,2,0,2,1
50,0,2,120,219,0,1,158,0,1.6,1,0,2,1
58,0,2,120,340,0,1,172,0,0,2,0,2,1
66,0,3,150,226,0,1,114,0,2.6,0,0,2,1
43,1,0,150,247,0,1,171,0,1.5,2,0,2,1
69,0,3,140,239,0,1,151,0,1.8,2,2,2,1
59,1,0,135,234,0,1,161,0,0.5,1,0,3,1
44,1,2,130,233,0,1,179,1,0.4,2,0,2,1
42,1,0,140,226,0,1,178,0,0,2,0,2,1
61,1,2,150,243,1,1,137,1,1,1,0,2,1
40,1,3,140,199,0,1,178,1,1.4,2,0,3,1
71,0,1,160,302,0,1,162,0,0.4,2,2,2,1
59,1,2,150,212,1,1,157,0,1.6,2,0,2,1
51,1,2,110,175,0,1,123,0,0.6,2,0,2,1
65,0,2,140,417,1,0,157,0,0.8,2,1,2,1
53,1,2,130,197,1,0,152,0,1.2,0,0,2,1
41,0,1,105,198,0,1,168,0,0,2,1,2,1
65,1,0,120,177,0,1,140,0,0.4,2,0,3,1
44,1,1,130,219,0,0,188,0,0,2,0,2,1
```

Google colaboratory (colab) software is used to run the program and predict the heart disease in the people community

2. Import packages (Numpy, Pandas) for processing the dataset from Kaggle . URL: https://www.kaggle.com/datasets/johnsmith88/heart-disease-dataset

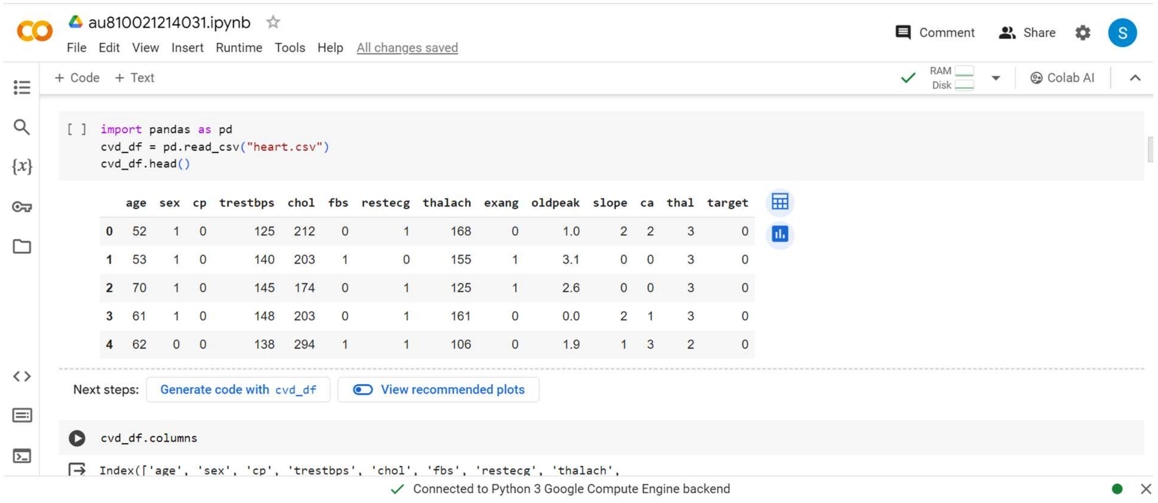


```
import pandas as pd
import pylab as pl
import numpy as np
import scipy.optimize as opt
import statsmodels.api as sm
from sklearn import preprocessing
'exec(% matplotlib inline)'
import matplotlib.pyplot as plt
import matplotlib.mlab as mlab
import seaborn as sns
```

```
/content/heart.csv
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-2-5f278aac50bd> in <cell line: 1>()
----> 1 content/heart.csv()

NameError: name 'content' is not defined
```

3.Read data from the URL with help of pandas



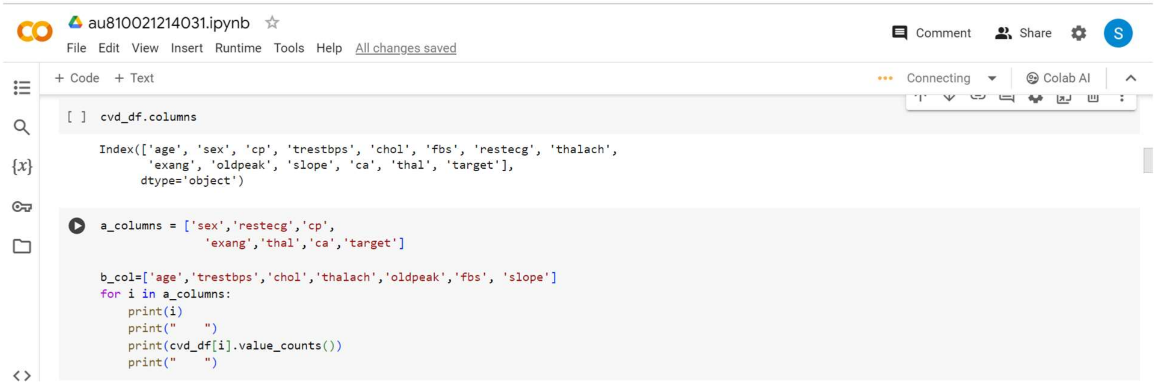
```
import pandas as pd
cvd_df = pd.read_csv("heart.csv")
cvd_df.head()
```

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
0	52	1	0	125	212	0	1	168	0	1.0	2	2	3	0
1	53	1	0	140	203	1	0	155	1	3.1	0	0	3	0
2	70	1	0	145	174	0	1	125	1	2.6	0	0	3	0
3	61	1	0	148	203	0	1	161	0	0.0	2	1	3	0
4	62	0	0	138	294	1	1	106	0	1.9	1	3	2	0

```
cvd_df.columns
```

```
Index(['age', 'sex', 'cp', 'trestbps', 'chol', 'fbs', 'restecg', 'thalach',
```

3. Split the dataset values into two

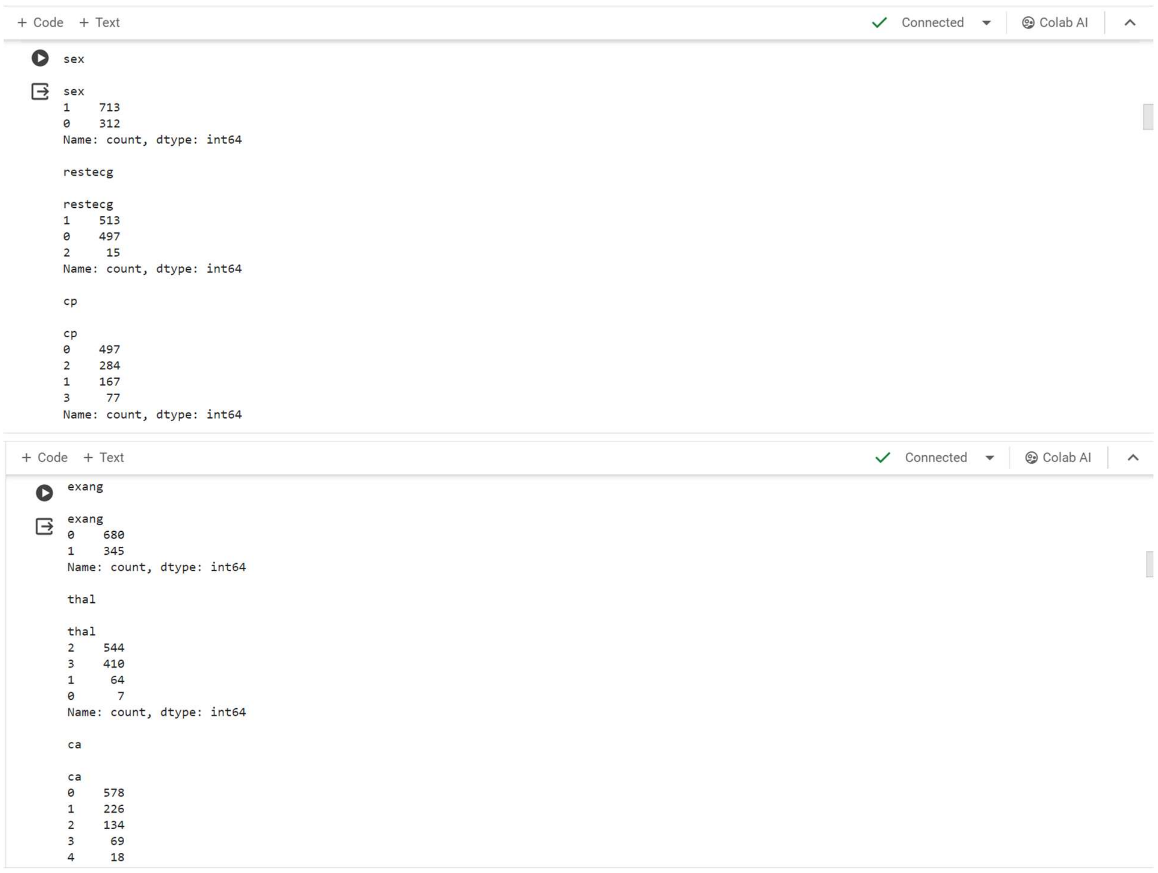


```
cvd_df.columns
```

```
Index(['age', 'sex', 'cp', 'trestbps', 'chol', 'fbs', 'restecg', 'thalach',
       'exang', 'oldpeak', 'slope', 'ca', 'thal', 'target'],
      dtype='object')
```

```
a_columns = ['sex','restecg','cp',
             'exang','thal','ca','target']

b_col=['age','trestbps','chol','thalach','oldpeak','fbs', 'slope']
for i in a_columns:
    print(i)
    print("     ")
    print(cvd_df[i].value_counts())
    print("     ")
```

Output:

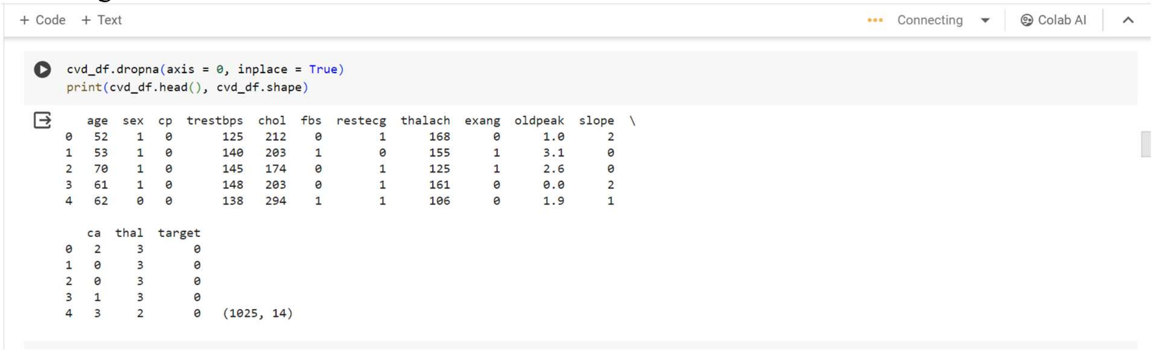


```
sex

sex
1    713
0    312
Name: count, dtype: int64

restecg

restecg
1    513
0    497
2     15
Name: count, dtype: int64

cp

cp
0    497
2    284
1    167
3     77
Name: count, dtype: int64
```

```
exang

exang
0    680
1    345
Name: count, dtype: int64

thal

thal
2    544
3    410
1     64
0      7
Name: count, dtype: int64

ca

ca
0    578
1    226
2    134
3     69
4     18
```

Connected to Python 3 Google Compute Engine backend

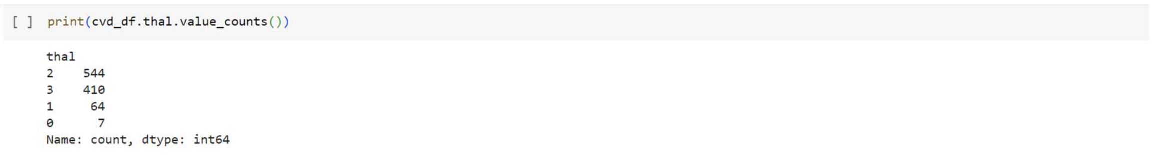
4. Handling null values:



```
cvd_df.dropna(axis = 0, inplace = True)
print(cvd_df.head(), cvd_df.shape)
```

```
   age  sex  cp  trestbps  chol  fbs  restecg  thalach  exang  oldpeak  slope  \
0   52    1   0       125   212    0        1      168      0      1.0      2
1   53    1   0       140   203    1        0      155      1      3.1      0
2   70    1   0       145   174    0        1      125      1      2.6      0
3   61    1   0       148   203    0        1      161      0      0.0      2
4   62    0   0       138   294    1        1      106      0      1.9      1

   ca  thal  target
0   2     3       0
1   0     3       0
2   0     3       0
3   1     3       0
4   3     2       0  (1025, 14)
```

5. Final count

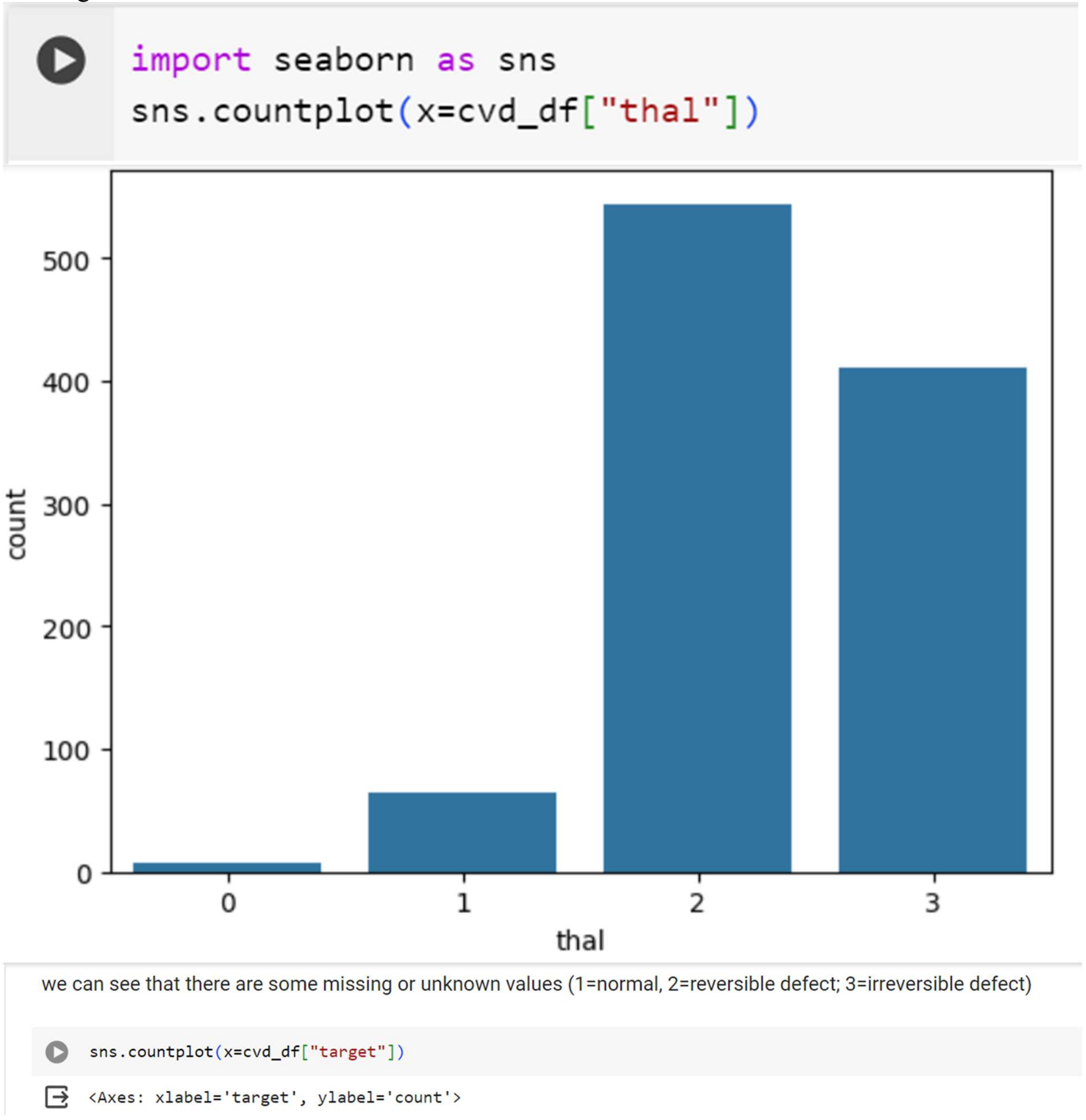


```
[ ] print(cvd_df.thal.value_counts())
```

```
thal
2    544
3    410
1     64
0      7
Name: count, dtype: int64
```

With this step, the dataset preprocessing is over

6. Exploratory data analysis is done and wrangling of data is also done to further find any missing or null values as shown below.

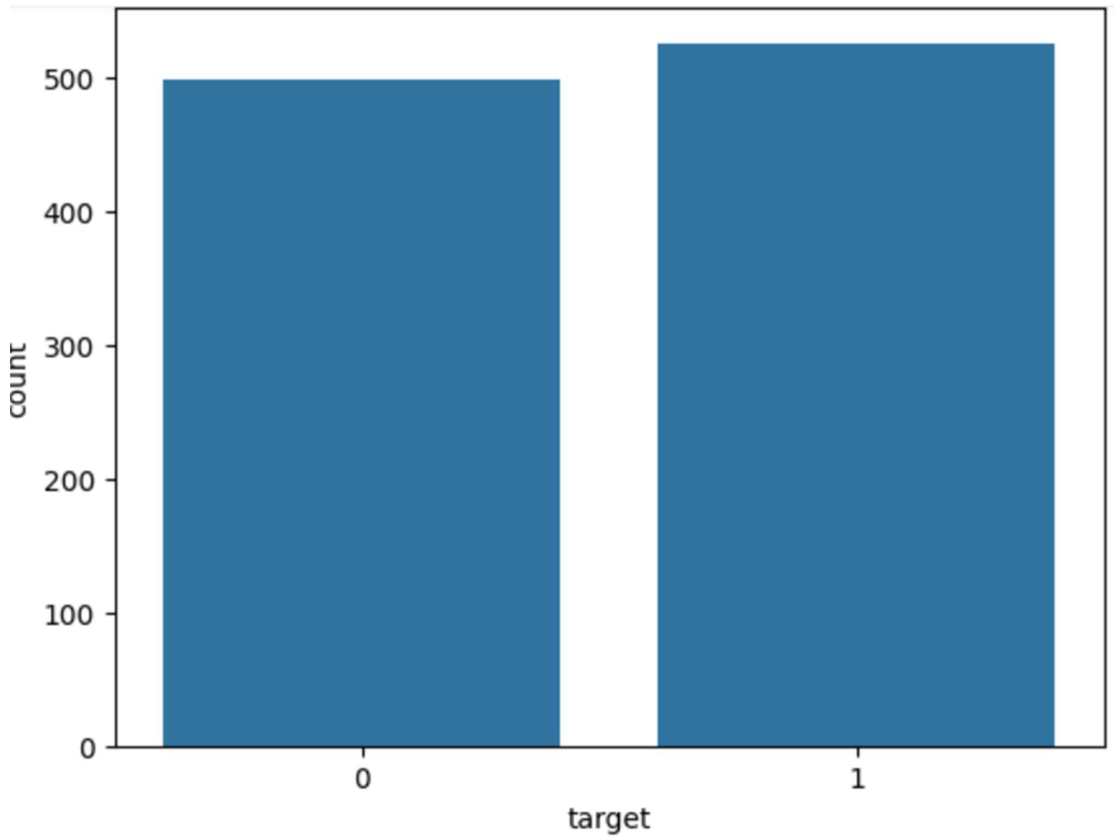


```
import seaborn as sns
sns.countplot(x=cvd_df["thal"])
```

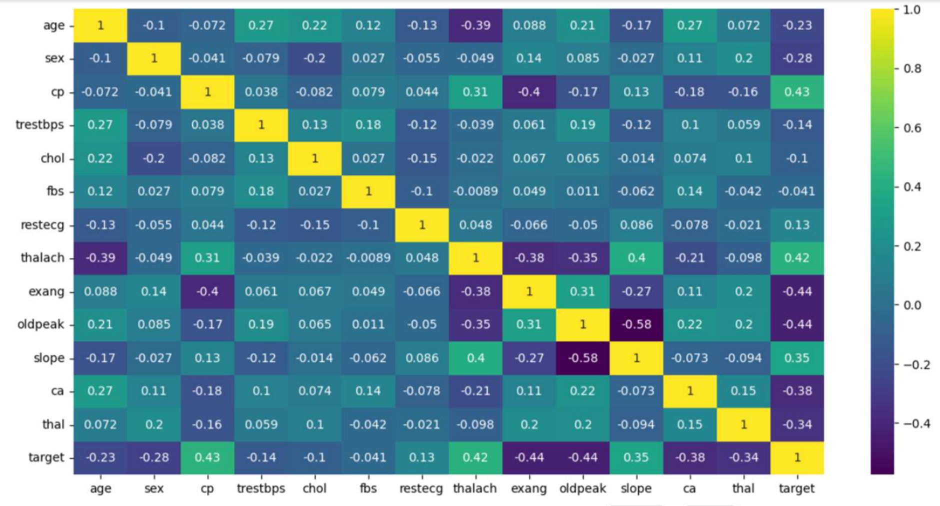
we can see that there are some missing or unknown values (1=normal, 2=reversible defect; 3=irreversible defect)

```
sns.countplot(x=cvd_df["target"])
```

```
<Axes: xlabel='target', ylabel='count'>
```

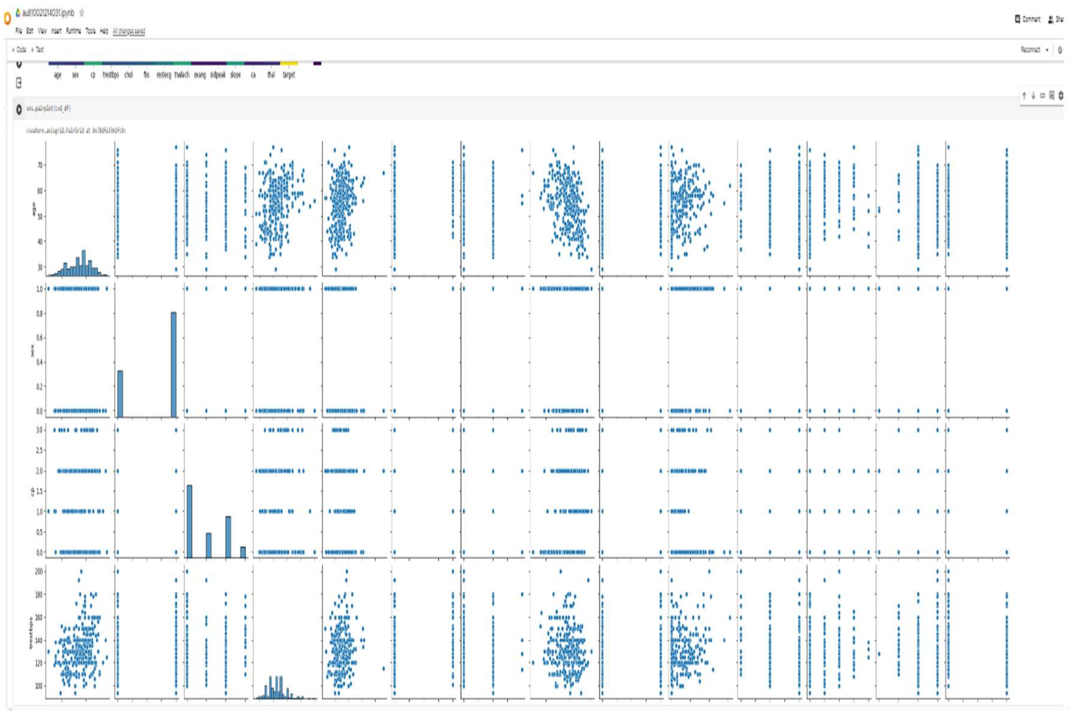


```
import matplotlib.pyplot as plt
plt.figure(figsize= (14,7))
sns.heatmap(cvd_df.corr(), annot = True, cmap= 'viridis');
```



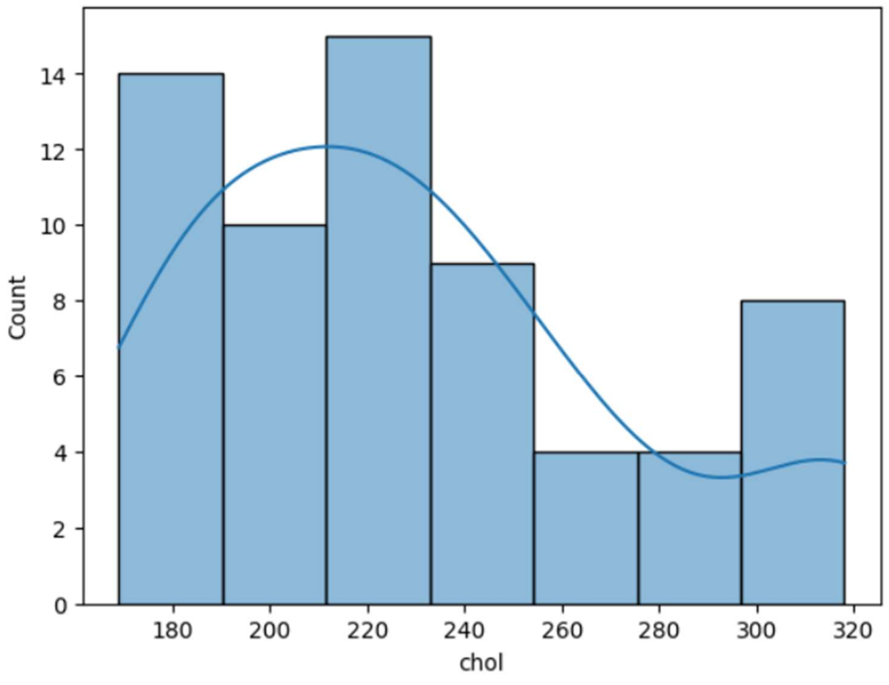
	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
age	1	-0.1	-0.072	0.27	0.22	0.12	-0.13	-0.39	0.088	0.21	-0.17	0.27	0.072	-0.23
sex	-0.1	1	-0.041	-0.079	-0.2	0.027	-0.055	-0.049	0.14	0.085	-0.027	0.11	0.2	-0.28
cp	-0.072	-0.041	1	0.038	-0.082	0.079	0.044	0.31	-0.4	-0.17	0.13	-0.18	-0.16	0.43
trestbps	0.27	-0.079	0.038	1	0.13	0.18	-0.12	-0.039	0.061	0.19	-0.12	0.1	0.059	-0.14
chol	0.22	-0.2	-0.082	0.13	1	0.027	-0.15	-0.022	0.067	0.065	-0.014	0.074	0.1	-0.1
fbs	0.12	0.027	0.079	0.18	0.027	1	-0.1	-0.0089	0.049	0.011	-0.062	0.14	-0.042	-0.041
restecg	-0.13	-0.055	0.044	-0.12	-0.15	-0.1	1	0.048	-0.066	-0.05	0.086	-0.078	-0.021	0.13
thalach	-0.39	-0.049	0.31	-0.039	-0.022	-0.0089	0.048	1	-0.38	-0.35	0.4	-0.21	-0.098	0.42
exang	0.088	0.14	-0.4	0.061	0.067	0.049	-0.066	-0.38	1	0.31	-0.27	0.11	0.2	-0.44
oldpeak	0.21	0.085	-0.17	0.19	0.065	0.011	-0.05	-0.35	0.31	1	-0.58	0.22	0.2	-0.44
slope	-0.17	-0.027	0.13	-0.12	-0.014	-0.062	0.086	0.4	-0.27	-0.58	1	-0.073	-0.094	0.35
ca	0.27	0.11	-0.18	0.1	0.074	0.14	-0.078	-0.21	0.11	0.22	-0.073	1	0.15	-0.38
thal	0.072	0.2	-0.16	0.059	0.1	-0.042	-0.021	-0.098	0.2	0.2	-0.094	0.15	1	-0.34
target	-0.23	-0.28	0.43	-0.14	-0.1	-0.041	0.13	0.42	-0.44	-0.44	0.35	-0.38	-0.34	1

```
sns.pairplot(cvd_df)
```

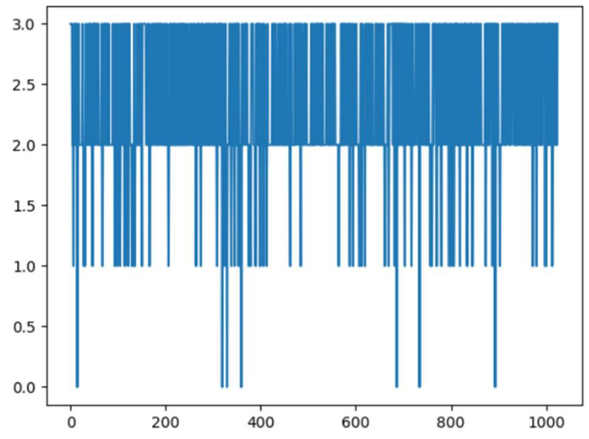


as we can see, there is high correlation between thal and chol

```
[ ]  sns.histplot(data=cvd_df,x=cvd_df[cvd_df["thal"]==1]["chol"],kde=True)
     plt.show()
```

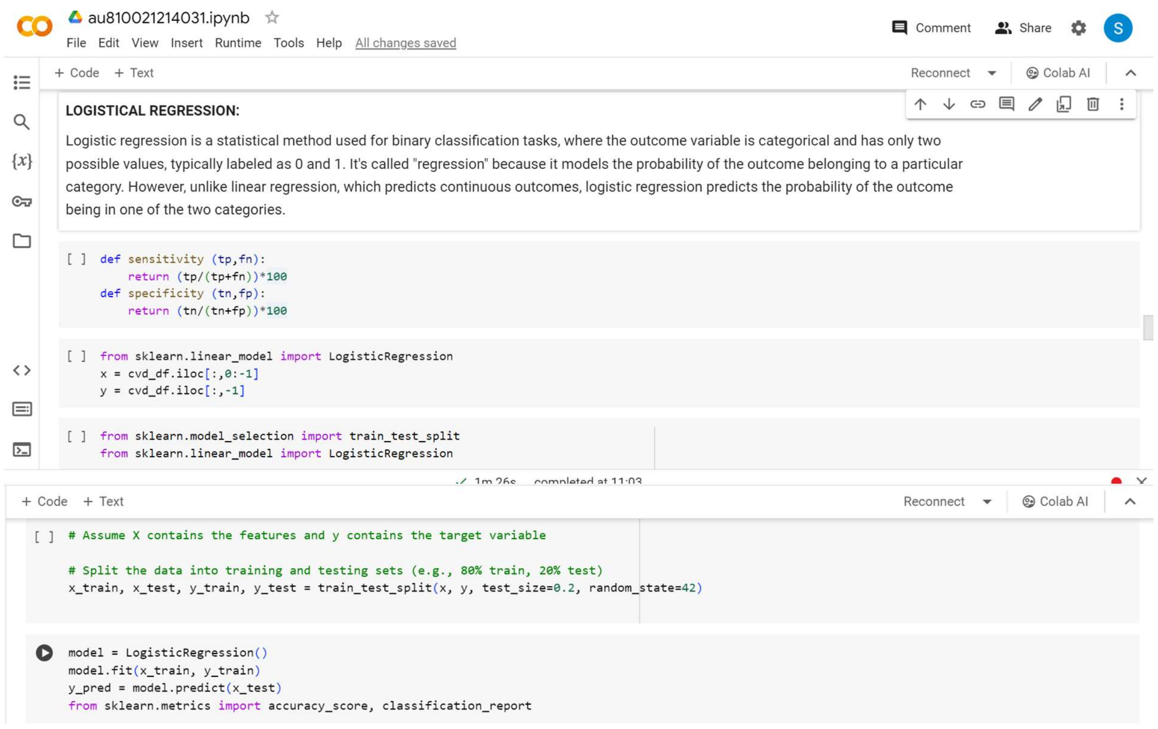


```
[ ] laste = cvd_df['thal'].plot()
    plt.show(laste)
```



From the EDA, we conclude there is high correlation between thal and chol and that the 0 value in thal refers to those who had heart disease but were not predicted. This means that we must also take sensitivity and specificity into account

7. Logistical regression model is constructed as follows:



LOGISTICAL REGRESSION:

Logistic regression is a statistical method used for binary classification tasks, where the outcome variable is categorical and has only two possible values, typically labeled as 0 and 1. It's called "regression" because it models the probability of the outcome belonging to a particular category. However, unlike linear regression, which predicts continuous outcomes, logistic regression predicts the probability of the outcome being in one of the two categories.

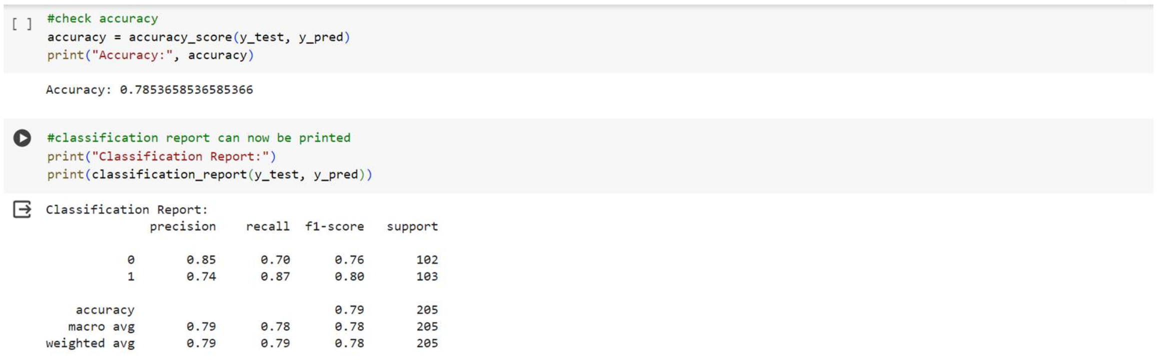
```
[ ] def sensitivity (tp,fn):
        return (tp/(tp+fn))*100
    def specificity (tn,fp):
        return (tn/(tn+fp))*100
```

```
[ ] from sklearn.linear_model import LogisticRegression
    x = cvd_df.iloc[:,0:-1]
    y = cvd_df.iloc[:,-1]
```

```
[ ] from sklearn.model_selection import train_test_split
    from sklearn.linear_model import LogisticRegression
```

```
[ ] # Assume X contains the features and y contains the target variable

    # Split the data into training and testing sets (e.g., 80% train, 20% test)
    x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_state=42)
```

```
model = LogisticRegression()
model.fit(x_train, y_train)
y_pred = model.predict(x_test)
from sklearn.metrics import accuracy_score, classification_report
```



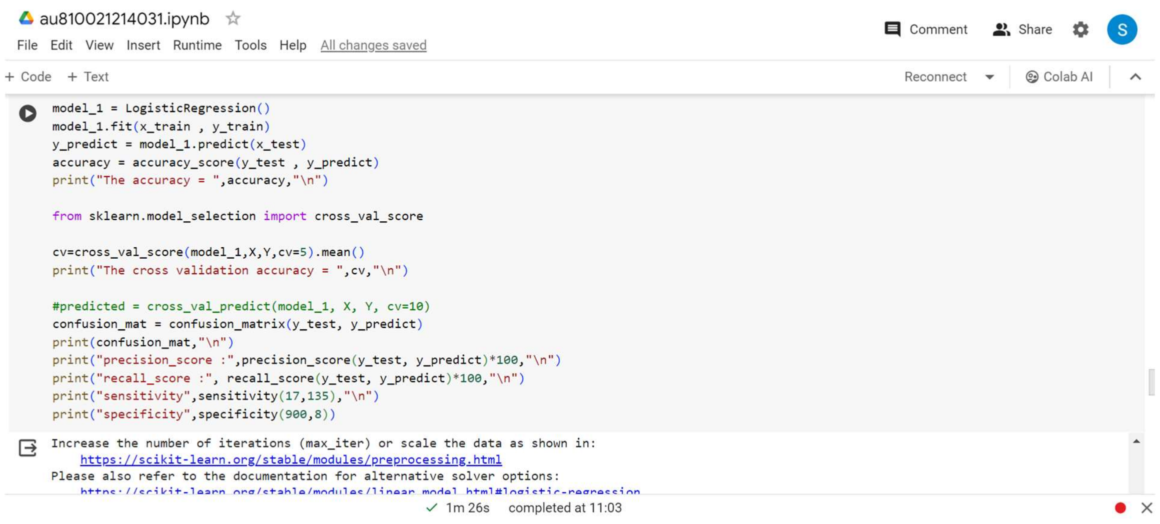
```
#check accuracy
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)
```

```
Accuracy: 0.7853658536585366
```

```
#classification report can now be printed
print("Classification Report:")
print(classification_report(y_test, y_pred))
```

```
Classification Report:
              precision    recall  f1-score   support

           0       0.85      0.70      0.76       102
           1       0.74      0.87      0.80       103

    accuracy                           0.79       205
   macro avg       0.79      0.78      0.78       205
weighted avg       0.79      0.79      0.78       205
```

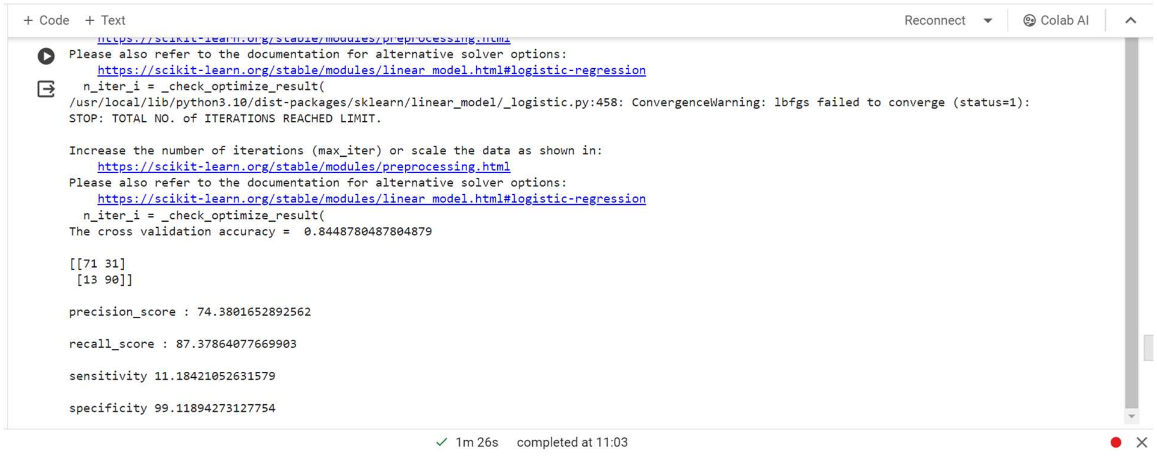
8. Now the model is prepared as follows:



au810021214031.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment Share
+ Code + Text
Reconnect
Colab AI
✓ 1m 26s completed at 11:03

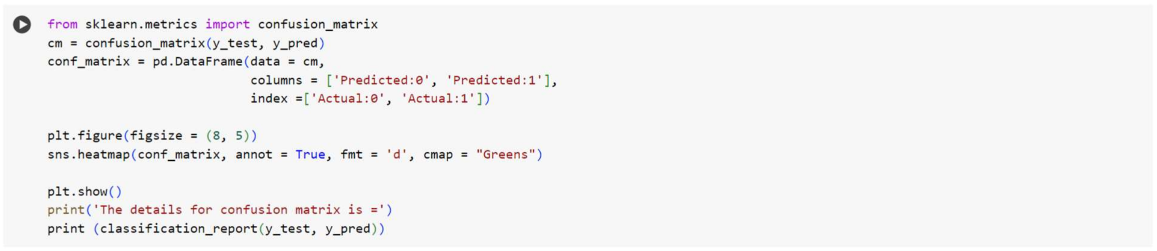
```
model_1 = LogisticRegression()
model_1.fit(x_train , y_train)
y_predict = model_1.predict(x_test)
accuracy = accuracy_score(y_test , y_predict)
print("The accuracy = ",accuracy,"\n")

from sklearn.model_selection import cross_val_score

cv=cross_val_score(model_1,X,Y,cv=5).mean()
print("The cross validation accuracy = ",cv,"\n")

#predicted = cross_val_predict(model_1, X, Y, cv=10)
confusion_mat = confusion_matrix(y_test, y_predict)
print(confusion_mat,"\n")
print("precision_score :",precision_score(y_test, y_predict)*100,"\n")
print("recall_score :", recall_score(y_test, y_predict)*100,"\n")
print("sensitivity",sensitivity(17,135),"\n")
print("specificity",specificity(900,8))
```

```
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:
```

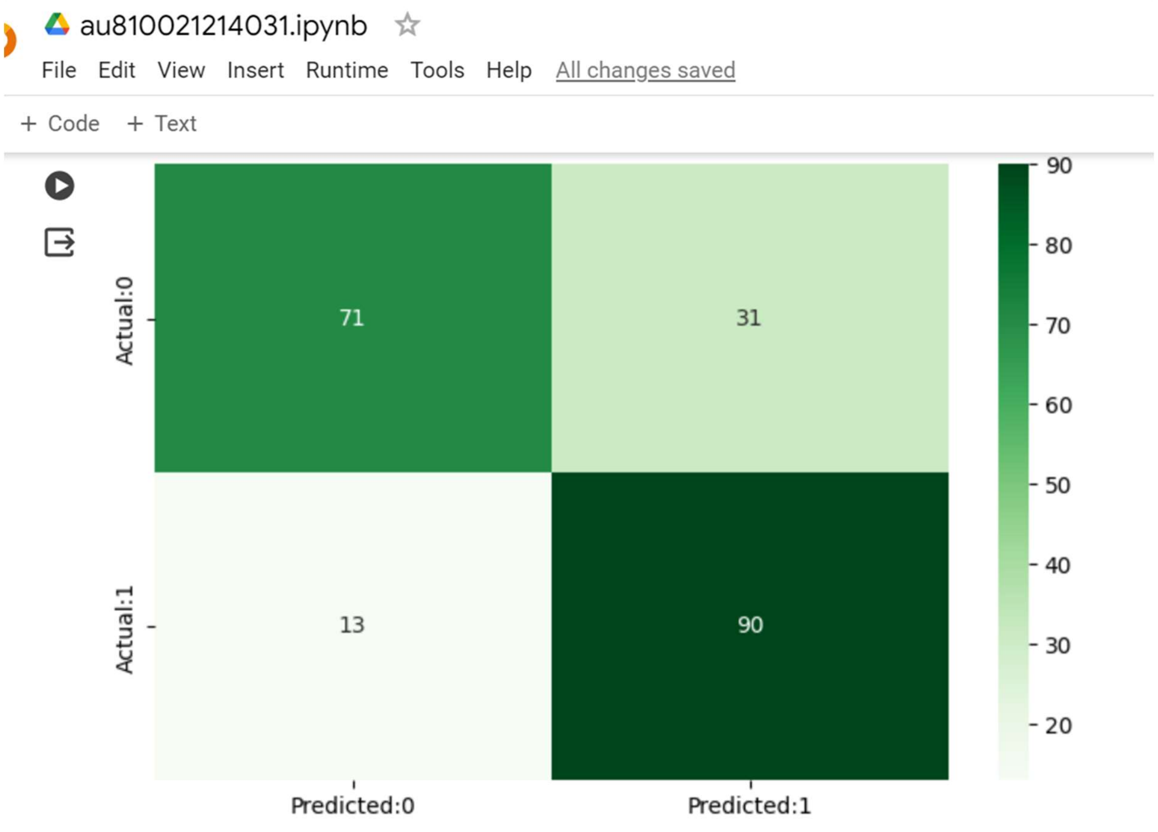


```
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
/usr/local/lib/python3.10/dist-packages/sklearn/linear_model/_logistic.py:458: ConvergenceWarning: lbfgs failed 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(
The cross validation accuracy =  0.8448780487804879

[[71 31]
 [13 90]]

precision_score : 74.3801652892562

recall_score : 87.37864077669903

sensitivity 11.18421052631579

specificity 99.11894273127754
```

9. Now, the confusion matrix is creates as shown:

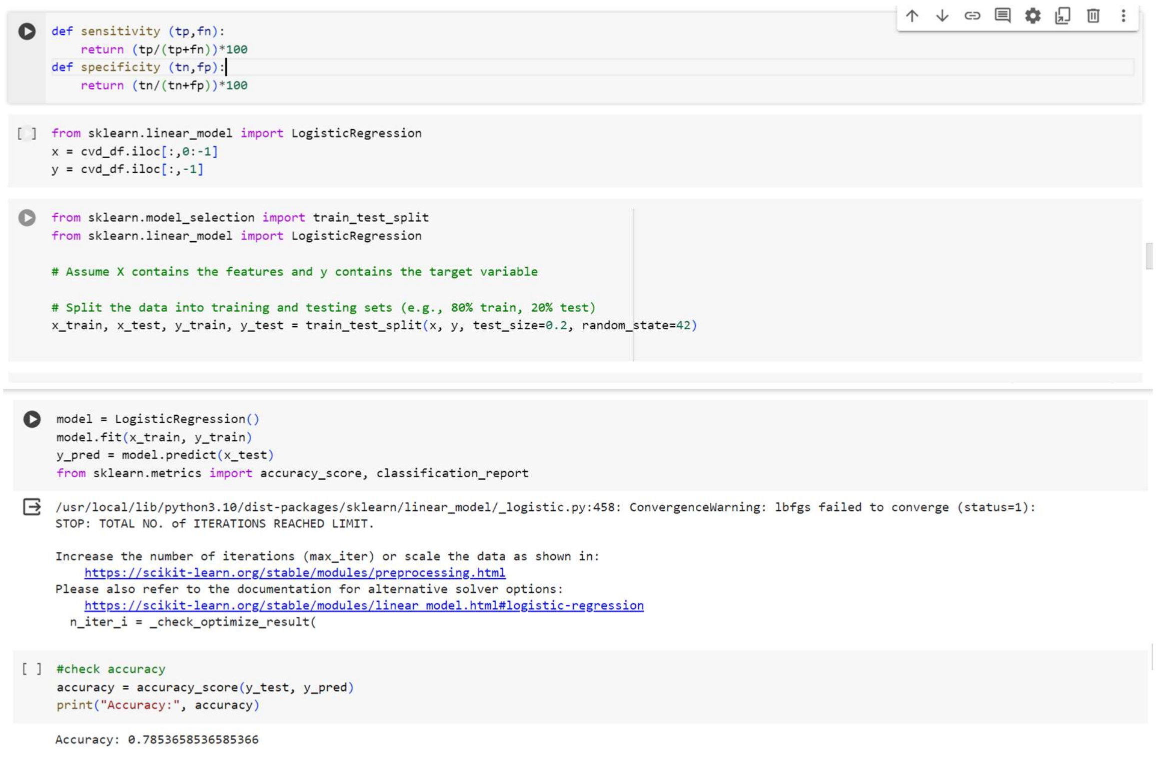


```
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred)
conf_matrix = pd.DataFrame(data = cm,
                           columns = ['Predicted:0', 'Predicted:1'],
                           index =['Actual:0', 'Actual:1'])

plt.figure(figsize = (8, 5))
sns.heatmap(conf_matrix, annot = True, fmt = 'd', cmap = "Greens")

plt.show()
print('The details for confusion matrix is =')
print (classification_report(y_test, y_pred))
```



```
The details for confusion matrix is =
              precision    recall  f1-score   support

           0       0.85      0.70      0.76       102
           1       0.74      0.87      0.80       103

    accuracy                           0.79       205
   macro avg       0.79      0.78      0.78       205
weighted avg       0.79      0.79      0.78       205
```

10.Now, we create a model which uses logistical regression.


```
def sensitivity (tp,fn):
    return (tp/(tp+fn))*100
def specificity (tn,fp):
    return (tn/(tn+fp))*100
```

```
from sklearn.linear_model import LogisticRegression
x = cvd_df.iloc[:,0:-1]
y = cvd_df.iloc[:,-1]
```

```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression

# Assume X contains the features and y contains the target variable

# Split the data into training and testing sets (e.g., 80% train, 20% test)
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_state=42)
```

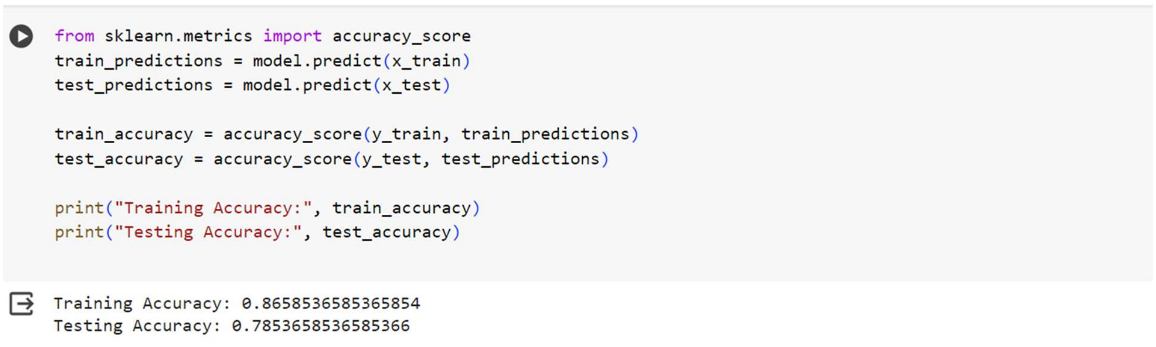
```
model = LogisticRegression()
model.fit(x_train, y_train)
y_pred = model.predict(x_test)
from sklearn.metrics import accuracy_score, classification_report
```

```
/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(
```

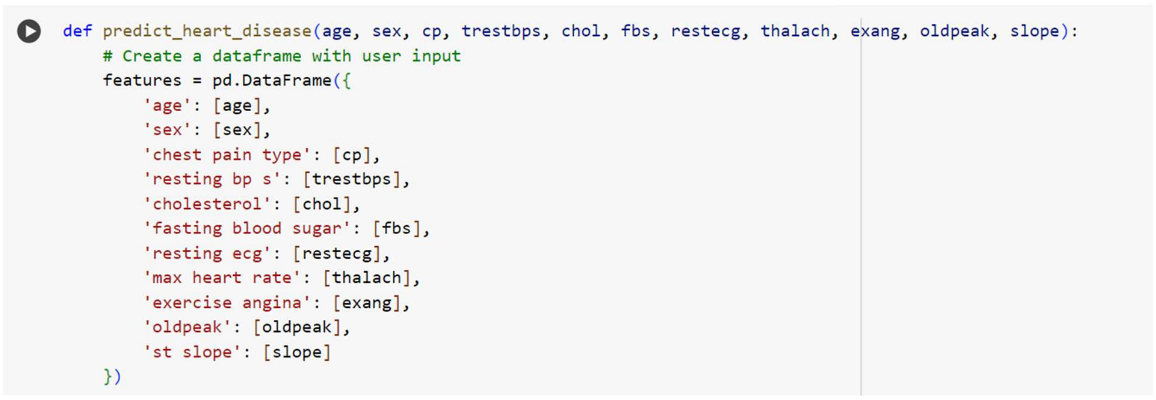
```
#check accuracy
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)
```

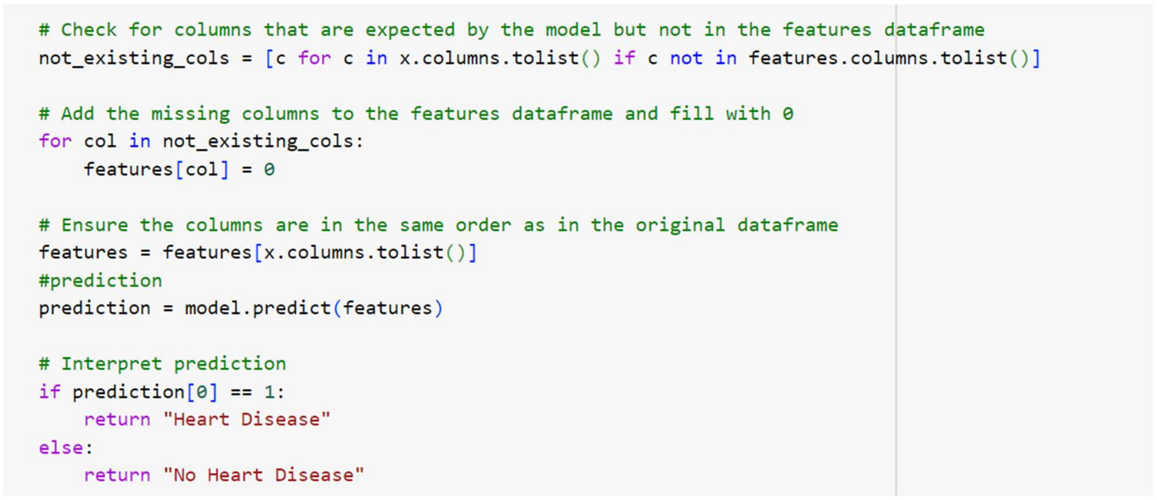
```
Accuracy: 0.7853658536585366
```

11. The final step is to make a prediction tool which can receive, read ,interpret and analyze input from the user.

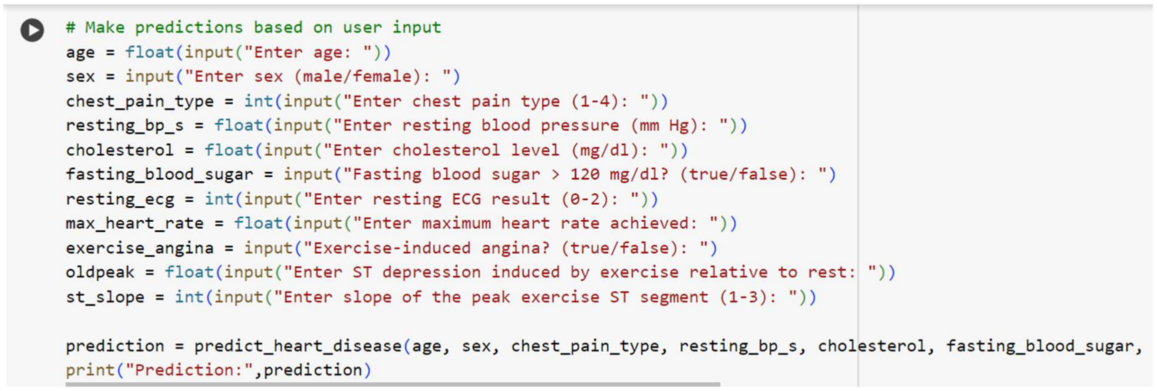

```
from sklearn.metrics import accuracy_score
train_predictions = model.predict(x_train)
test_predictions = model.predict(x_test)

train_accuracy = accuracy_score(y_train, train_predictions)
test_accuracy = accuracy_score(y_test, test_predictions)

print("Training Accuracy:", train_accuracy)
print("Testing Accuracy:", test_accuracy)
```

```
Training Accuracy: 0.8658536585365854
Testing Accuracy: 0.7853658536585366
```

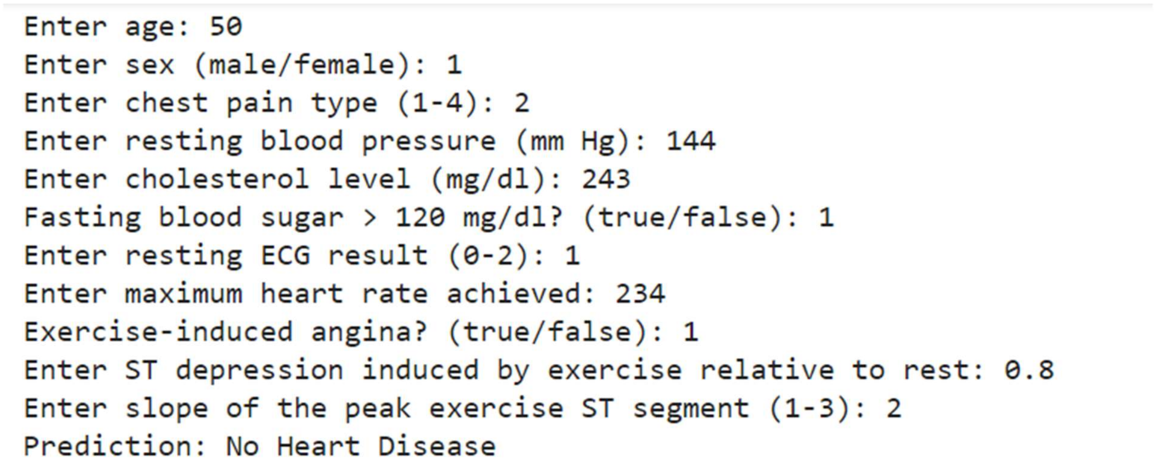

```
def predict_heart_disease(age, sex, cp, trestbps, chol, fbs, restecg, thalach, exang, oldpeak, slope):
    # Create a dataframe with user input
    features = pd.DataFrame({
        'age': [age],
        'sex': [sex],
        'chest pain type': [cp],
        'resting bp s': [trestbps],
        'cholesterol': [chol],
        'fasting blood sugar': [fbs],
        'resting ecg': [restecg],
        'max heart rate': [thalach],
        'exercise angina': [exang],
        'oldpeak': [oldpeak],
        'st slope': [slope]
    })
```



```
# Check for columns that are expected by the model but not in the features dataframe
not_existing_cols = [c for c in x.columns.tolist() if c not in features.columns.tolist()]

# Add the missing columns to the features dataframe and fill with 0
for col in not_existing_cols:
    features[col] = 0

# Ensure the columns are in the same order as in the original dataframe
features = features[x.columns.tolist()]
#prediction
prediction = model.predict(features)

# Interpret prediction
if prediction[0] == 1:
    return "Heart Disease"
else:
    return "No Heart Disease"
```



```
# Make predictions based on user input
age = float(input("Enter age: "))
sex = input("Enter sex (male/female): ")
chest_pain_type = int(input("Enter chest pain type (1-4): "))
resting_bp_s = float(input("Enter resting blood pressure (mm Hg): "))
cholesterol = float(input("Enter cholesterol level (mg/dl): "))
fasting_blood_sugar = input("Fasting blood sugar > 120 mg/dl? (true/false): ")
resting_ecg = int(input("Enter resting ECG result (0-2): "))
max_heart_rate = float(input("Enter maximum heart rate achieved: "))
exercise_angina = input("Exercise-induced angina? (true/false): ")
oldpeak = float(input("Enter ST depression induced by exercise relative to rest: "))
st_slope = int(input("Enter slope of the peak exercise ST segment (1-3): "))

prediction = predict_heart_disease(age, sex, chest_pain_type, resting_bp_s, cholesterol, fasting_blood_sugar,
print("Prediction:",prediction)
```



```
Enter age: 50
Enter sex (male/female): 1
Enter chest pain type (1-4): 2
Enter resting blood pressure (mm Hg): 144
Enter cholesterol level (mg/dl): 243
Fasting blood sugar > 120 mg/dl? (true/false): 1
Enter resting ECG result (0-2): 1
Enter maximum heart rate achieved: 234
Exercise-induced angina? (true/false): 1
Enter ST depression induced by exercise relative to rest: 0.8
Enter slope of the peak exercise ST segment (1-3): 2
Prediction: No Heart Disease
```

CHAPTER 5

Project result

The dataset which was collected from many countries was used to train the AI model. This can be further enhanced by getting more newer data and adding more variables to make the data more precise. The user can now input the result which can then be analyzed by the given model. The model has a testing accuracy of 79%(approximately). To increase the accuracy, we must add more input data. The methods of testing for the variables such as cholesterol may also have some error. This is covered under the sensitivity and specificity section of the coding. The final result is an fairly accurate model which uses logistical regression to predict the chances of getting a CVD.

CONCLUSION

The application which has been developed in this project can be used to detect any heart disease before it even occurs based on data such as age, gender, ECG date, etc. This model, which uses logistic regression can also be further developed with an higher number of input data and further refinement of the techniques used. New applications like there could possibly increase the chances of saving the life of a patient without compromising time and valuable resources which would otherwise be needed to diagnose a heart condition.

The data used in the project was found to be highly effective for the training of Artificial Intelligence and Machine learning. Based on the exploratory data analysis conducted, there seems to be very high correlation between the patients who have high cholesterol or get exercise angina (pain in the chest during or after exercise/strenuous activities) and those who have cardiovascular diseases. Thus, the given AI model is highly effective for finding out the more important parameters to predict heart disease in a given patient.

FUTURE SCOPE

The future of Artificial intelligence in the prediction of heart diseases is very important. Simple yet powerful machine learning tools such as logistic regression can alter the course of human development by saving the lives of many people and work in the diagnosis of various illnesses which plague humanity. With further developments such as newer datasets with higher number of risk factors, more accurate results from highly sensitive and specific test results can help to improve the scope of this model. By promoting hospitals and other organizations to enter data into these models can help improve their accuracy over the course of time

REFERENCES

1. Project Github link, Ramar Bose , 2024
2. Project video recorded link (youtube/github), Ramar Bose , 2024
3. Project PPT & Report github link, Ramar Bose , 2024

GIT Hub Link of Project Code:

https://github.com/shreyaSR-au810021214031/Naan-Mudhalvan-Heart_Disease_Prediction-Shreya-SR

YouTube link: https://youtu.be/1I19M80riqw