HEART-DISEASE PREDICTION APPLICATION

TABLE OF CONTENTS

Contents

INTRODUCTION..... 3

METHOD 6

Heart Disease prediction App (Using Sublime text & streamlit) 18

Conclusion 19

INTRODUCTION

Heart Disease Prediction

- It might have happened so many times that you or someone yours need doctors help immediately, but they are not available due to some reason.
- The Heart Disease Prediction application is an end user support and online consultation project
- Here, we propose a web application that allows users to get instant guidance on their heart disease through an intelligent system online.
- The application is fed with various details and the heart disease associated with those details.
- The application allows user to share their heart related issues.
- It then processes user specific details to check for various illness that could be associated with i
- Here we use some intelligent data mining techniques to guess the most accurate illness that could be associated with patient's details.
- Based on result, system automatically shows the result specific doctors for further treatment.
- The system allows user to view doctor's details.
- The system can be use in case of emergency.

The system comprises of 2 major modules as follows:

➢ Admin Module

1. Add Training Data
2. Add Doctor Details
3. View User Details
4. View Feedback
5. View Doc Details
6. View Training Data

➢ User Module

1. Register (With Details like Age, Sex, etc.)
2. Check Heart (By providing Details like
    - Age in Year
    - Gender
    - Chest Pain Type

    - Fasting Blood Sugar
    - Resting Electrographic Results(Restecg)
    - Exercise Induced Angina(Exang)
    - The slope of the peak exercise ST segment
    - CA – Number of major vessels colored by fluoroscopy
    - Thal
    - Trest Blood Pressure
    - Serum Cholesterol
    - Maximum heart rate achieved(Thalach)
    - ST depression induced by exercise(Oldpeak)
3. System will accordingly view Doctor to consult.
4. Give Feedback

- View Doctor

Software Requirements:

- Windows 7 or higher.
- Google Chrome
- Google Colab / Anaconda (Jupyter Notebook)

Hardware Components:

- Processor – i3
- Hard Disk – 5 GB
- Memory – 4GB RAM
- Internet Connection

Advantages: -

- User can search for doctor's help at any point of time.
- User can talk about their heart disease and get instant diagnosis.
- Doctors get more clients online.
- Very useful in case of emergency.

Disadvantages:

The system is not fully automated, it needs data from user for full diagnosis.

Application:

This application can be used by all patients or their family members who need help in emergency.

METHOD

Importing Libraries

```
[52] import numpy as np
     import pandas as pd
     import matplotlib as plt
     import matplotlib.pyplot as plt
     import seaborn as sns
```

```
[53] df = pd.read_csv('heart.csv')
```

```
[54] df.head() #To display our data in tabular form
```

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
0	63	1	3	145	233	1	0	150	0	2.3	0	0	1	1
1	37	1	2	130	250	0	1	187	0	3.5	0	0	2	1
2	41	0	1	130	204	0	0	172	0	1.4	2	0	2	1
3	56	1	1	120	236	0	1	178	0	0.8	2	0	2	1
4	57	0	0	120	354	0	1	163	1	0.6	2	0	2	1

EDA

```
EDA

[55] np.shape(df) #Shows the total number of rows and columns in our dataset

     (303, 14)

[56] df.columns #Shows the column names in our dataset

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

[57] df.nunique(axis=0) #No of unique values in different columns

     age         41
     sex          2
     cp           4
     trestbps    49
     chol       152
     fbs          2
     restecg      3
     thalach     91
     exang        2
     oldpeak     40
     slope        3
     ca           5
     thal         4
     target       2
     dtype: int64
```

```
[59] df.info() #Displays data types and count of non-null values in the columns
```

```
<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   sex       303 non-null    int64  
 2   cp        303 non-null    int64  
 3   trestbps  303 non-null    int64  
 4   chol      303 non-null    int64  
 5   fbs       303 non-null    int64  
 6   restecg   303 non-null    int64  
 7   thalach   303 non-null    int64  
 8   exang     303 non-null    int64  
 9   oldpeak   303 non-null    float64
 10  slope     303 non-null    int64  
 11  ca        303 non-null    int64  
 12  thal      303 non-null    int64  
 13  target    303 non-null    int64  
dtypes: float64(1), int64(13)
memory usage: 33.3 KB
```

```
[60] df.describe() #summarizes the mean, count, standard deviation, minimum and maximum for the numeric variables
```

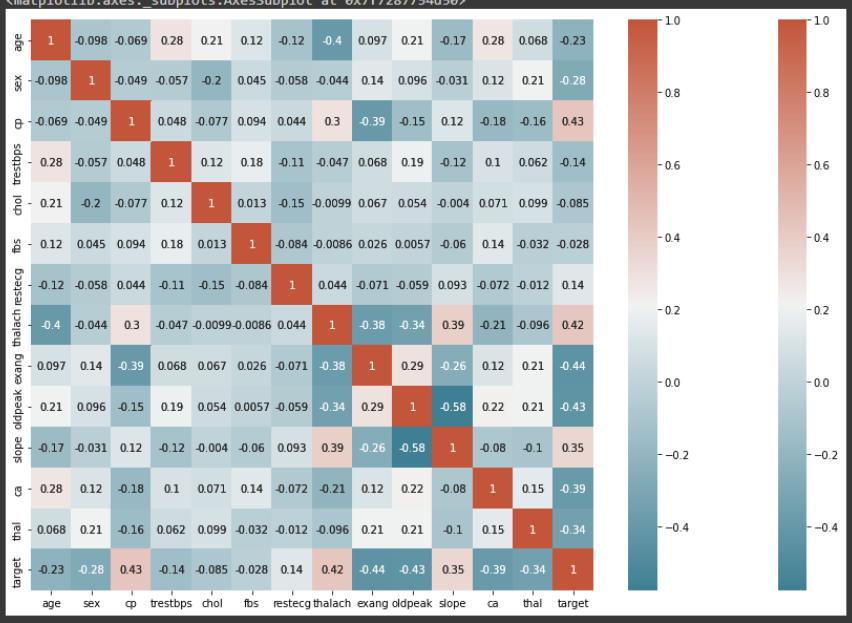
	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
count	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000
mean	54.366337	0.683168	0.966997	131.623762	246.264026	0.148515	0.528053	149.646865	0.326733	1.039604	1.399340	0.729373	2.313531	0.544554
std	9.082101	0.466011	1.032052	17.538143	51.830751	0.356198	0.525860	22.905161	0.469794	1.161075	0.616226	1.022606	0.612277	0.498835
min	29.000000	0.000000	0.000000	94.000000	126.000000	0.000000	0.000000	71.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	47.500000	0.000000	0.000000	120.000000	211.000000	0.000000	0.000000	133.500000	0.000000	0.000000	1.000000	0.000000	2.000000	0.000000
50%	55.000000	1.000000	1.000000	130.000000	240.000000	0.000000	1.000000	153.000000	0.000000	0.800000	1.000000	0.000000	2.000000	1.000000
75%	61.000000	1.000000	2.000000	140.000000	274.500000	0.000000	1.000000	166.000000	1.000000	1.600000	2.000000	1.000000	3.000000	1.000000
max	77.000000	1.000000	3.000000	200.000000	564.000000	1.000000	2.000000	202.000000	1.000000	6.200000	2.000000	4.000000	3.000000	1.000000

```
[61] df['target'].value_counts() #Displays the number of 1's and 0's in the 'target' column
```

```
1    165
0    138
Name: target, dtype: int64
```

```
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))
```

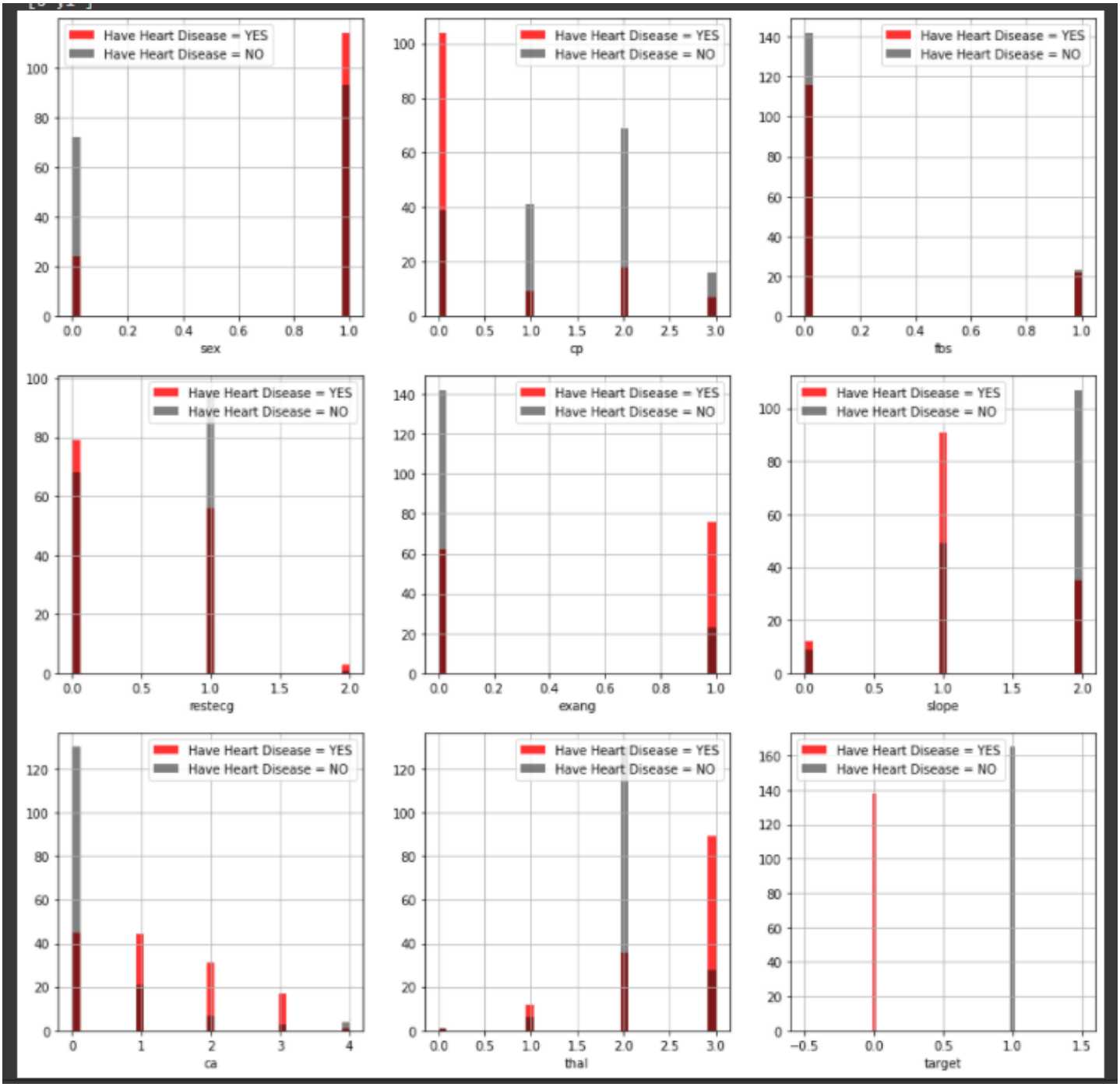
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f7287754d50>
```



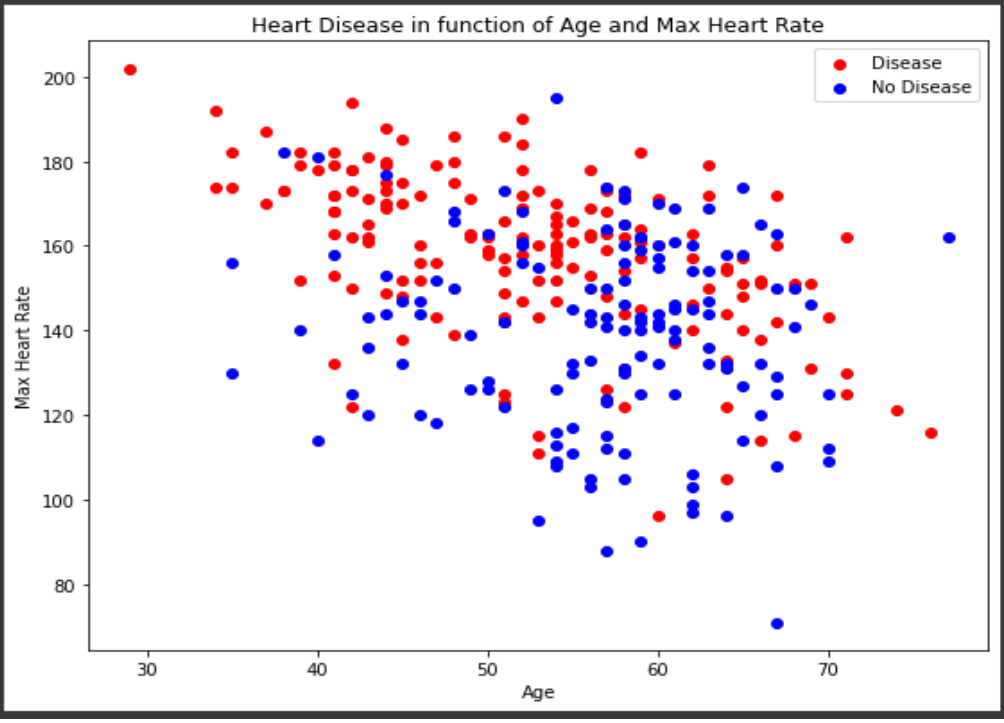
Visualization

```
[64] categorical_val = []
     continous_val = []
     for column in df.columns:
         if len(df[column].unique()) <= 10:
             categorical_val.append(column)
         else:
             continous_val.append(column)
```

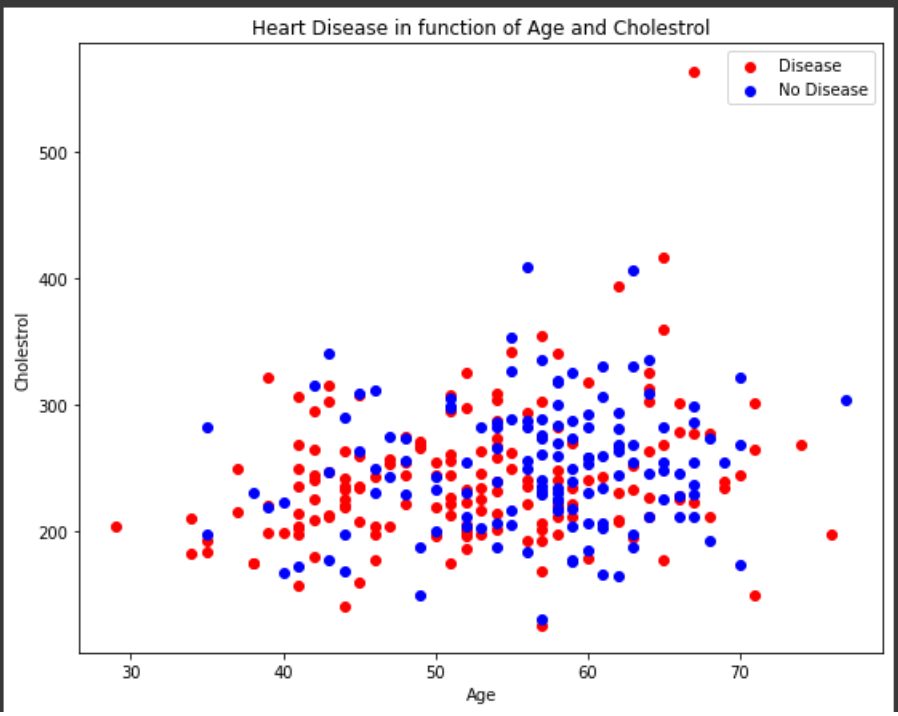
```
plt.figure(figsize=(15, 15))
print ("sex=[1 = male, 0 = female]")
print ("chest pain type =[0=Typical angina, 1=Atypical angina, 2=Non-anginal pain, 3=Asymptomatic]")
print ("Fasting Blood Sugar=[1 = true, 0 = false]")
print ("Resting Electrocardiographic=[0 = Normal, 1 = Non-Normal, 2 = Risk]")
print ("Exercise induced Angina =[1 = yes; 0 = no]")
print ("Peak Exercise=[0= better heart rate with excercise 1=typical healthy heart, 2=signs of unhealthy heart]")
print ("=[0=,1=,2=,3=,4=]")
print ("=[0=,1=,2=,3=]")
print ("=[0=,1=]")
for i, column in enumerate(categorical_val, 1):
    plt.subplot(3, 3, i)
    df[df["target"] == 0][column].hist(bins=35, color='Red', label='Have Heart Disease = YES', alpha=.8)
    df[df["target"] == 1][column].hist(bins=35, color='Black', label='Have Heart Disease = NO', alpha=.5)
    plt.legend()
    plt.xlabel(column)
```


Have Heart Disease = YES
Have Heart Disease = NO
sex
cp
fbs
restecg
exang
slope
ca
thal
target

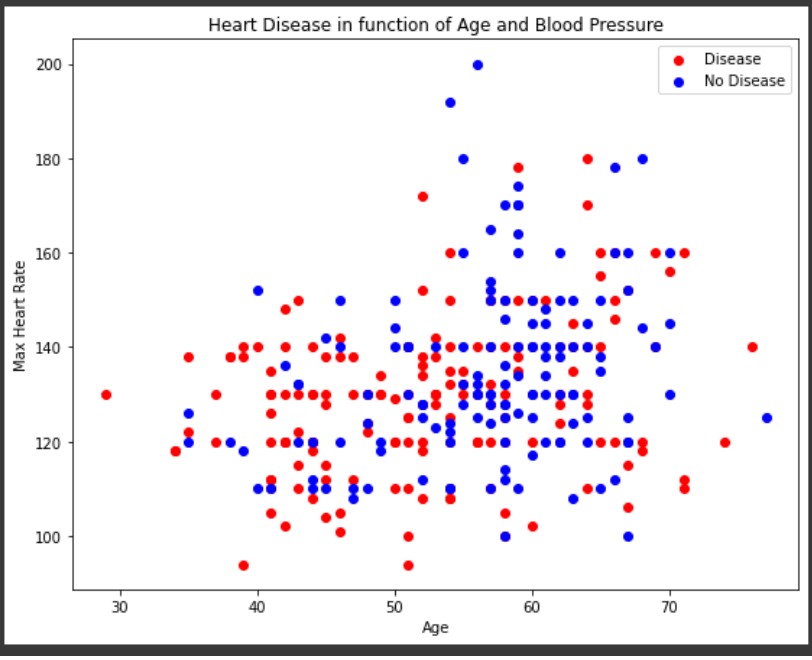
```
[66] plt.figure(figsize=(9, 7))

     # Scatter with postivie examples
     plt.scatter(df.age[df.target==1],
                 df.thalach[df.target==1],
                 c="red")

     # Scatter with negative examples
     plt.scatter(df.age[df.target==0],
                 df.thalach[df.target==0],
                 c="blue")

     # Add some helpful info
     plt.title("Heart Disease in function of Age and Max Heart Rate")
     plt.xlabel("Age")
     plt.ylabel("Max Heart Rate")
     plt.legend(["Disease", "No Disease"]);
```



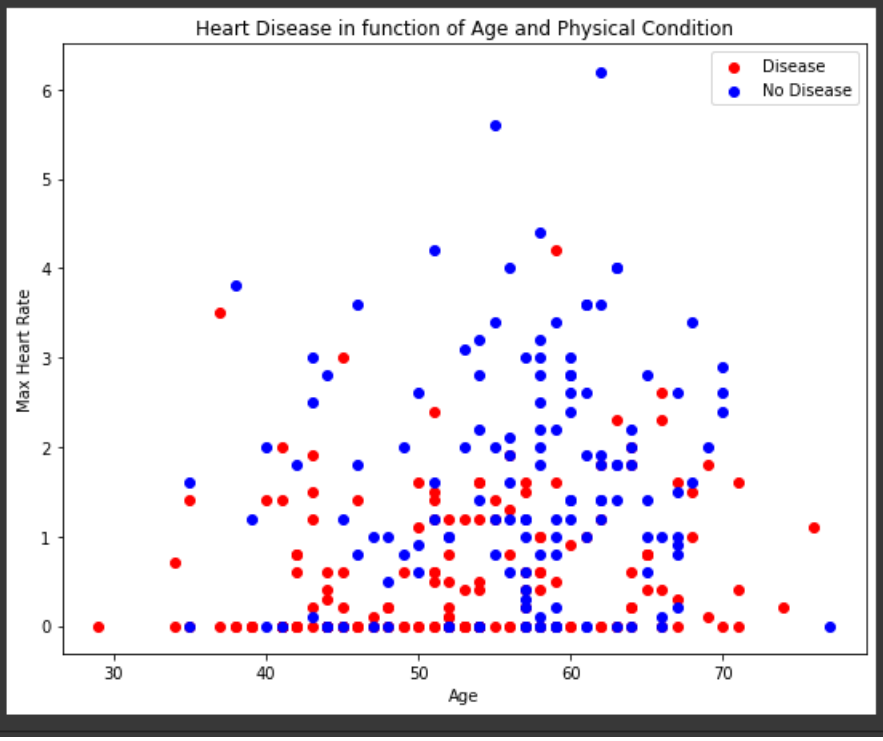
```
[67] plt.figure(figsize=(9, 7))

     # Scatter with postivie examples
     plt.scatter(df.age[df.target==1],
                 df.chol[df.target==1],
                 c="red")

     # Scatter with negative examples
     plt.scatter(df.age[df.target==0],
                 df.chol[df.target==0],
                 c="blue")

     # Add some helpful info
     plt.title("Heart Disease in function of Age and Cholestrol")
     plt.xlabel("Age")
     plt.ylabel("Cholestrol")
     plt.legend(["Disease", "No Disease"]);
```



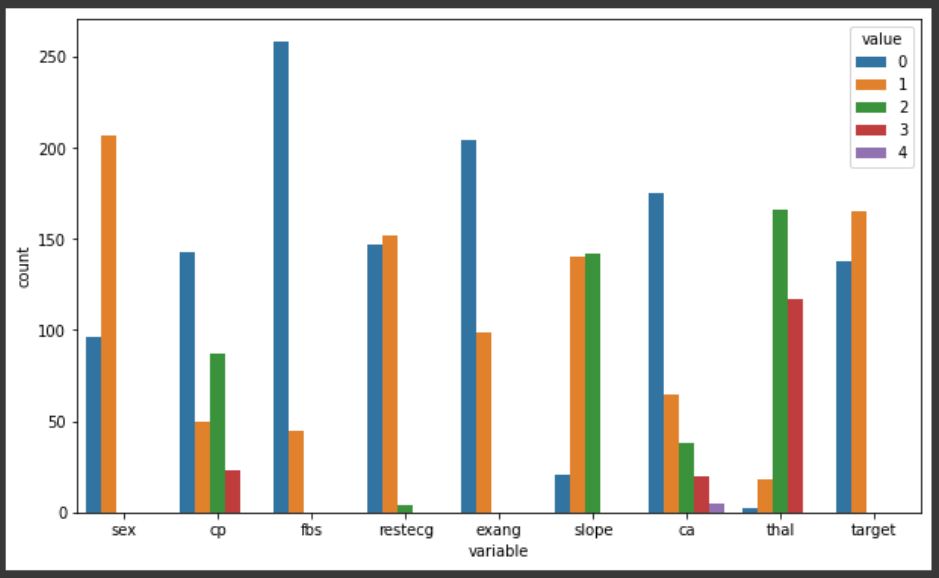
```
[68] # Scatter with postivie examples
     plt.scatter(df.age[df.target==1],
                 df.trestbps[df.target==1],
                 c="red")

     # Scatter with negative examples
     plt.scatter(df.age[df.target==0],
                 df.trestbps[df.target==0],
                 c="blue")

     # Add some helpful info
     plt.title("Heart Disease in function of Age and Blood Pressure")
     plt.xlabel("Age")
     plt.ylabel("Max Heart Rate")
     plt.legend(["Disease", "No Disease"]);
```



```
[69] plt.figure(figsize=(9, 7))

     # Scatter with postivie examples
     plt.scatter(df.age[df.target==1],
                 df.oldpeak[df.target==1],
                 c="red")

     # Scatter with negative examples
     plt.scatter(df.age[df.target==0],
                 df.oldpeak[df.target==0],
                 c="blue")

     # Add some helpful info
     plt.title("Heart Disease in function of Age and Physical Condition")
     plt.xlabel("Age")
     plt.ylabel("Max Heart Rate")
     plt.legend(["Disease", "No Disease"]);
```

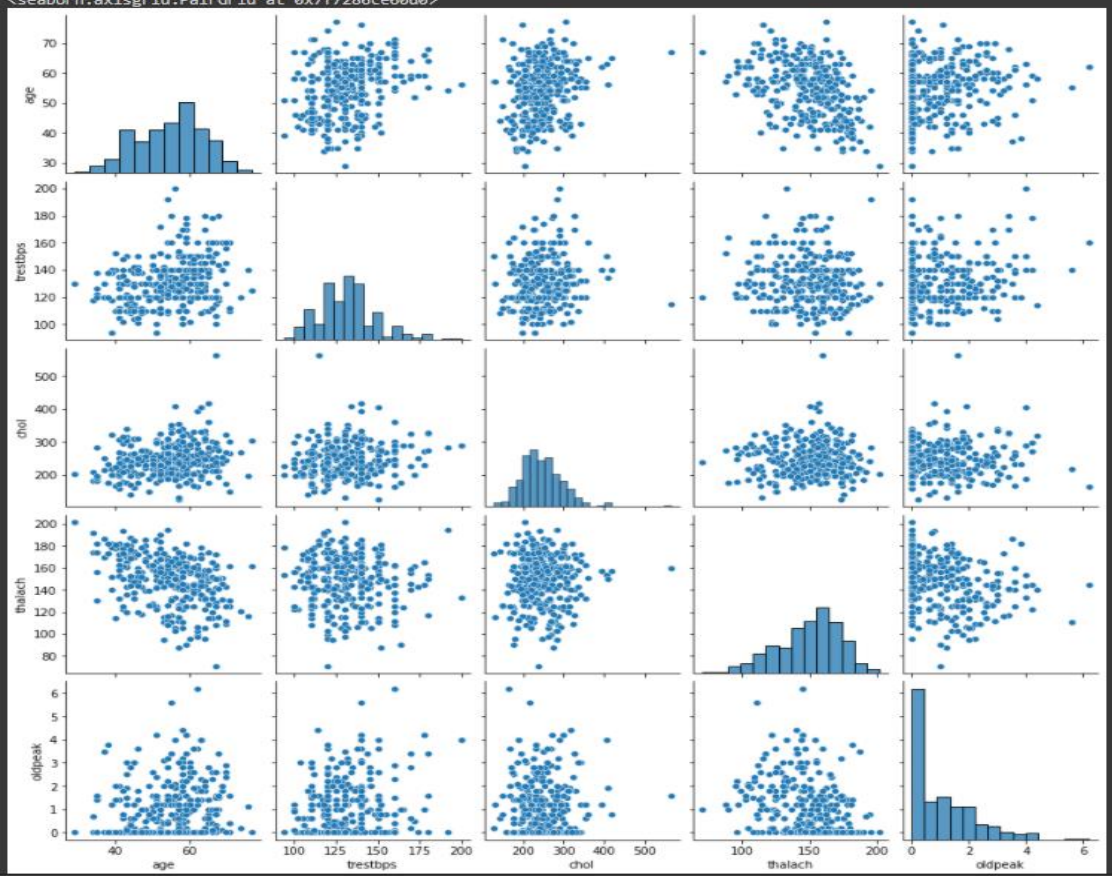


```
[70] plt.figure(figsize=(10,6))
     dataframe_categorical = df.loc[:,['sex', 'cp', 'fbs', 'restecg', 'exang', 'slope', 'ca', 'thal', 'target']]
     sns.countplot(x="variable", hue="value",data= pd.melt(dataframe_categorical));
```




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

```
<seaborn.axisgrid.PairGrid at 0x7f7286ce60d0>
```



Data Preprocessing

```
[72] ##### DATA FILTERING

[73] # To filter the data according to positive heart disease patients

pos_df = df[df['target']==1]

pos_df.describe()
```

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
count	165.000000	165.000000	165.000000	165.000000	165.000000	165.000000	165.000000	165.000000	165.000000	165.000000	165.000000	165.000000	165.000000	165.0
mean	52.496970	0.563636	1.375758	129.303030	242.230303	0.139394	0.593939	158.466667	0.139394	0.583030	1.593939	0.363636	2.121212	1.0
std	9.550651	0.497444	0.952222	16.169613	53.552872	0.347412	0.504818	19.174276	0.347412	0.780683	0.593635	0.848894	0.465752	0.0
min	29.000000	0.000000	0.000000	94.000000	126.000000	0.000000	0.000000	96.000000	0.000000	0.000000	0.000000	0.000000	0.000000	1.0
25%	44.000000	0.000000	1.000000	120.000000	208.000000	0.000000	0.000000	149.000000	0.000000	0.000000	1.000000	0.000000	2.000000	1.0
50%	52.000000	1.000000	2.000000	130.000000	234.000000	0.000000	1.000000	161.000000	0.000000	0.200000	2.000000	0.000000	2.000000	1.0
75%	59.000000	1.000000	2.000000	140.000000	267.000000	0.000000	1.000000	172.000000	0.000000	1.000000	2.000000	0.000000	2.000000	1.0
max	76.000000	1.000000	3.000000	180.000000	564.000000	1.000000	2.000000	202.000000	1.000000	4.200000	2.000000	4.000000	3.000000	1.0

```
[75] # To filter the data according to negative heart disease patients

[76] neg_df = df[df['target']==0]

neg_df.describe()
```

	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

```
[77] ####### USING MACHINE LEARNING ALGORITHMS

[78] #preparing the data for training and assigning values to variables X and Y

[79] X = df.iloc[:, :-1].values

     Y = df.iloc[:, -1].values

[80] #To split the data into training and testing sets

[81] 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)

[82] # Now, to normalize the data so its distribution will have a mean of 0

[83] from sklearn.preprocessing import StandardScaler

     sc = StandardScaler()

     x_train = sc.fit_transform(x_train)

     x_test = sc.transform(x_test)
```

Logistic Regression

Logistic Regression

```
[84] from sklearn.metrics import accuracy_score
```

```
[85] from sklearn.linear_model import LogisticRegression
     classifier = LogisticRegression()
     classifier.fit(x_train,y_train)
```

```
LogisticRegression()
```

```
[86] y_pred = classifier.predict(x_test)
     print(y_pred)
```

```
[0 1 0 0 0 0 0 1 0 1 0 0 1 0 0 1 1 0 0 1 1 0 1 1 1 1 0 1 1 1 1 1 1 1 0 0 0
 0 1 0 1 0 0 0 1 1 0 1 1 1 1 1 1 1 1 1 1 0 1 0 1]
```

```
[87] from sklearn.metrics import confusion_matrix
     cm = confusion_matrix(y_test, y_pred)
     print(cm)
```

```
[[20 10]
 [ 6 25]]
```

```
[88] (20+25)/(10+6+20+25)
```

```
0.7377049180327869
```

```
[89] from sklearn.model_selection import cross_val_score
     accuracies = cross_val_score(estimator = classifier, X = x_train, y = y_train, cv = 10)
     print("Accuracy: {:.2f} %".format(accuracies.mean()*100))
     print("Standard Deviation: {:.2f} %".format(accuracies.std()*100))
```

```
Accuracy: 81.85 %
Standard Deviation: 6.06 %
```

Random Forest Model

Random Forest Model

```
[90] from sklearn.metrics import classification_report

     from sklearn.ensemble import RandomForestClassifier
```

```
[91] random_model = RandomForestClassifier(random_state=1)# get instance of model

     random_model.fit(x_train, y_train) # Train model

     y_pred = random_model.predict(x_test) # get y predictions

     print(classification_report(y_test, y_pred)) # 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
```

```
[92] #The random forest model has 80% accuracy
```

```
[93] # get importance

     importance = random_model.feature_importances_

     # summarize feature importance

     for i,v in enumerate(importance):

        print('Feature: %0d, Score: %.5f' % (i,v))
```

```
Feature: 0, Score: 0.07814
Feature: 1, Score: 0.04206
Feature: 2, Score: 0.16580
Feature: 3, Score: 0.07477
Feature: 4, Score: 0.07587
Feature: 5, Score: 0.00828
Feature: 6, Score: 0.02014
Feature: 7, Score: 0.12772
Feature: 8, Score: 0.06950
Feature: 9, Score: 0.09957
Feature: 10, Score: 0.04677
Feature: 11, Score: 0.11667
Feature: 12, Score: 0.07473
```

Decision Tree

Decision Tree

```
[95] from sklearn.tree import DecisionTreeClassifier
```

```
[96] max_accuracy = 0


    for x in range(200):
        dt = DecisionTreeClassifier(random_state=x)
        dt.fit(x_train,y_train)
        y_pred_dt = dt.predict(x_test)
        current_accuracy = round(accuracy_score(y_pred_dt,y_test)*100,2)
        if(current_accuracy>max_accuracy):
            max_accuracy = current_accuracy
            best_x = x

    #print(max_accuracy)
    #print(best_x)


    dt = DecisionTreeClassifier(random_state=best_x)
    dt.fit(x_train,y_train)
    y_pred_dt = dt.predict(x_test)
```

```
[97] print(y_pred_dt.shape)

    (61,)
```

```
[98] score_dt = round(accuracy_score(y_pred_dt,y_test)*100,2)

    print("The accuracy score achieved using Decision Tree is: "+str(score_dt)+" %")

    The accuracy score achieved using Decision Tree is: 77.05 %
```

Decision Tree K – Nearest Neighbors

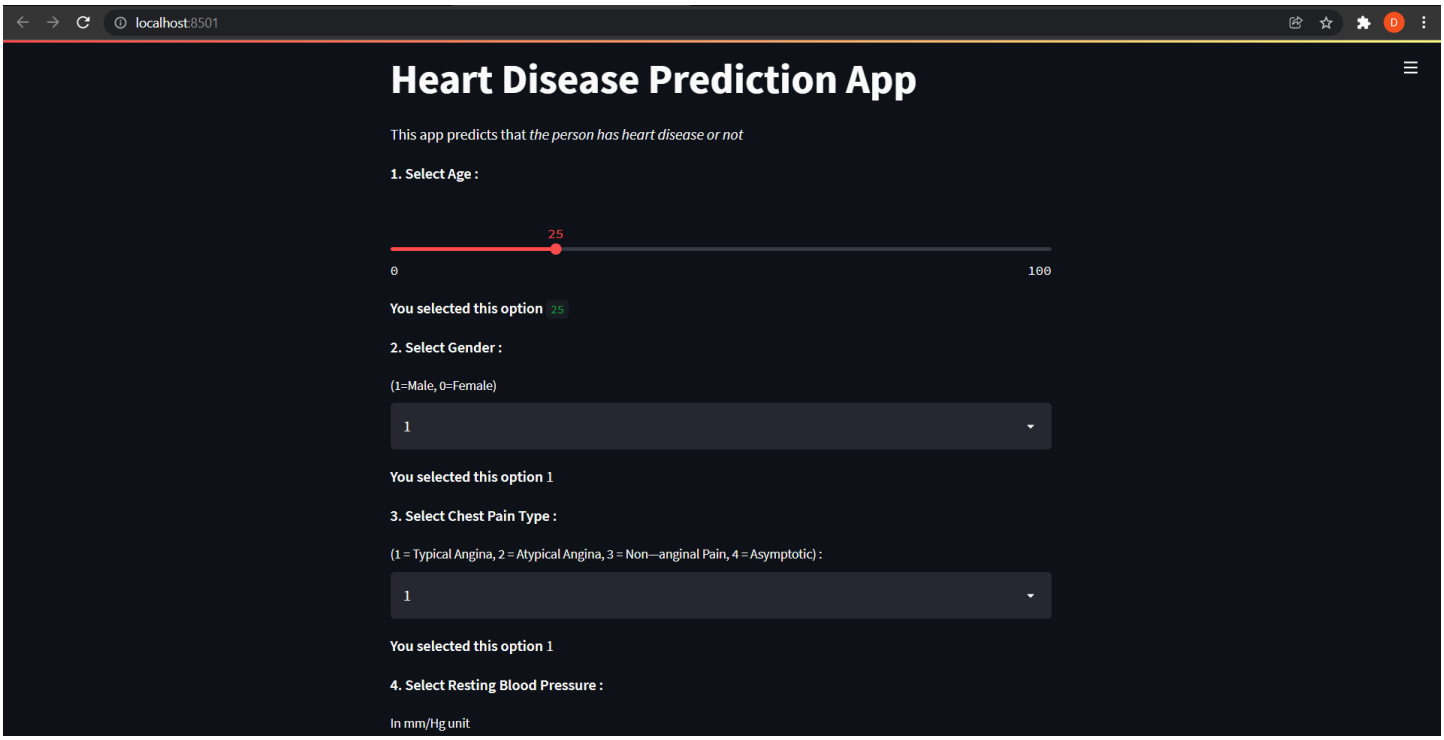
K-Nearest Neighbors

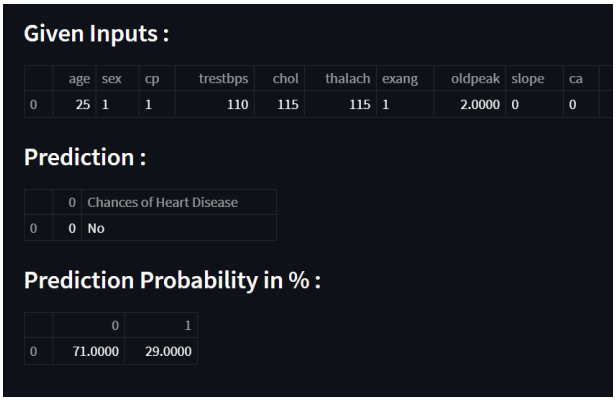
```
from sklearn.neighbors import KNeighborsClassifier

knn = KNeighborsClassifier(n_neighbors=7)
knn.fit(x_train,y_train)
y_pred_knn=knn.predict(x_test)
```

```
[100] y_pred_knn.shape

    (61,)
```

```
[101] score_knn = round(accuracy_score(y_pred_knn,y_test)*100,2)

    print("The accuracy score achieved using KNN is: "+str(score_knn)+" %")

    The accuracy score achieved using KNN is: 80.33 %
```

HEART DISEASE PREDICTION APP
(USING SUBLIME TEXT & STREAMLIT)





	age	sex	cp	trestbps	chol	thalach	exang	oldpeak	slope	ca
0	25	1	1	110	115	115	1	2.0000	0	0

	0	Chances of Heart Disease
0	0	No

	0	1
0	71.0000	29.0000

CONCLUSION

Heart diseases when aggravated spiral way beyond control. Heart diseases are complicated and take away lots of lives every year .When the early symptoms of heart diseases are ignored, the patient might end up with drastic consequences in a short span of time. Sedentary lifestyle and excessive stress in today's world have worsened the situation. If the disease is detected early then it can be kept under control. However, it is always advisable to exercise daily and discard unhealthy habits at the earliest. Tobacco consumption and unhealthy diets increase the chances of stroke and heart diseases. Eating at least 5 helpings of fruits and vegetables a day is a good practice. For heart disease patients, it is advisable to restrict the intake of salt to one teaspoon per day. One of the major drawbacks of these works is that the main focus has been on the application of classification techniques for heart disease prediction, rather than studying various data cleaning and pruning techniques that prepare and make a dataset suitable for mining. It has been observed that a properly cleaned and pruned dataset provides much better accuracy than an unclean one with missing values. Selection of suitable techniques for data cleaning along with proper classification algorithms will lead to the development of prediction systems that give enhanced accuracy. In future an intelligent system may be developed that can lead to selection of proper treatment methods for a patient diagnosed with heart disease. A lot of work has been done already in making models that can predict whether a patient is likely to develop heart disease or not. There are several treatment methods for a patient once diagnosed with a particular form of heart disease. Data mining can be of very good help in deciding the line of treatment to be followed by extracting knowledge from such suitable databases.