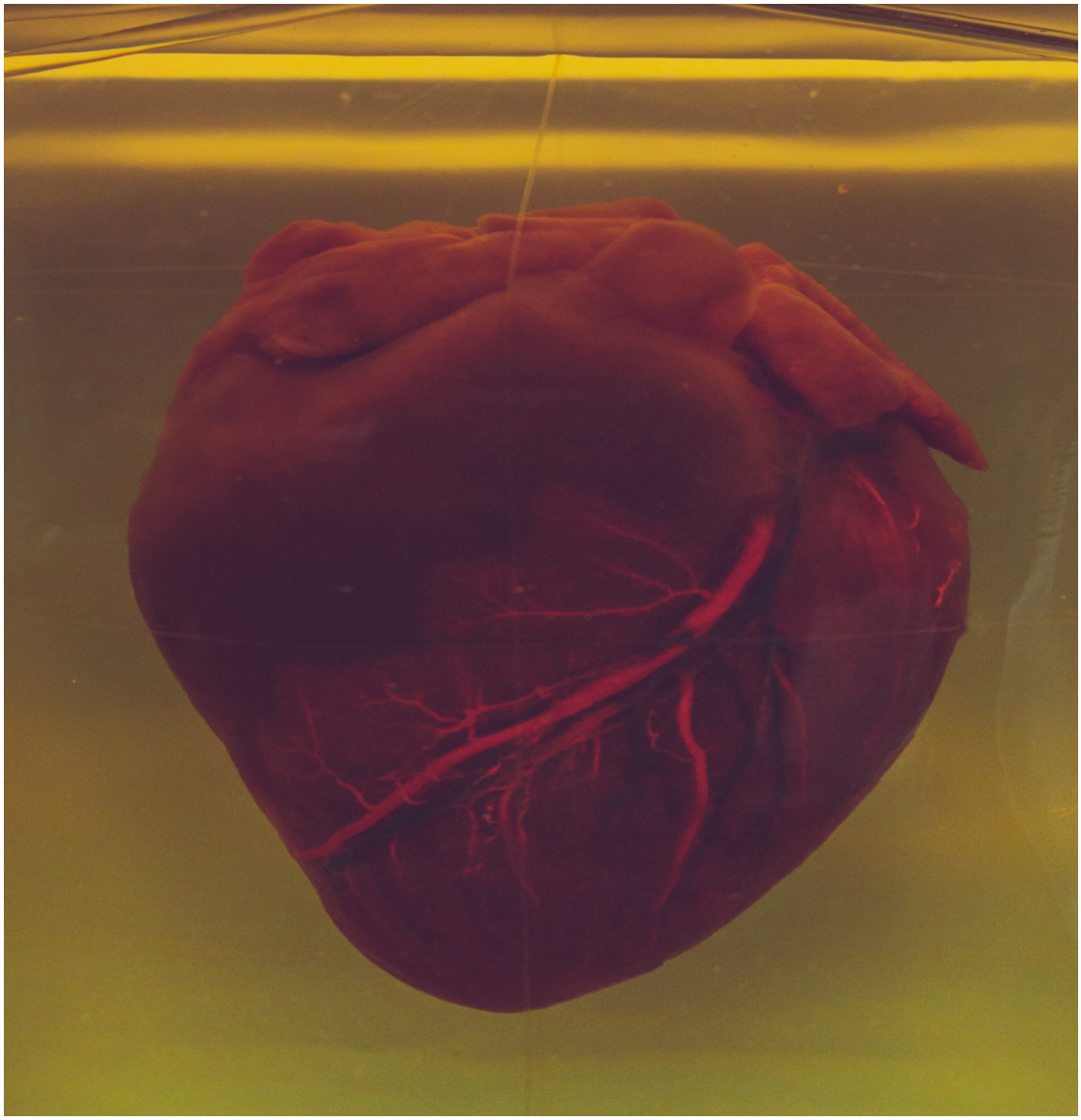
Project: Predicting Heart Disease

Submitted By

SARAVANAKUMAR G (EBSLEM0322454664)



Abstract

ABSTRACT

Heart plays significant role in living organisms. Diagnosis and prediction of heart related diseases requires more precision, perfection and correctness because a little mistake can cause fatigue problem or death of the person, there are numerous death cases related to heart and their counting is increasing exponentially day by day. To deal with the problem there is essential need of prediction system for awareness about diseases. Machine learning is the branch of Artificial Intelligence(AI), it provides prestigious support in predicting any kind of event which take training from natural events. In this paper, we calculate accuracy of machine learning algorithms for predicting heart disease, for this algorithms are k-nearest neighbor, decision tree,Logistic Regression, support vectomachine(SVM),Naives Bayes Classifier, Random Forest and XGBoost by using Kaggle repository dataset. For implementation of Python programming Anaconda(jupyter) notebook is best tool, which have many type of library, header file, that make the work more accurate and precise.

Contents

Table of Contents

CHAPTER	PARTICULARS	PAGE NO:
Chapter 1	Introductions: Scenario & Goals, Features & Predictor	6
Chapter 2	Features & Predictor	9
Chapter 3	Methodology: (i).Datasets (ii).Data Cleaning Preprocessing (iii). Machine Learning Algorithms (iv).Implementation Steps	11
Chapter 4	Analysis of the Result	31
Chapter 5	Conclusions	35
	Reference	37

Chapter-I

Chapter I
INTRODUCTION

The most crucial part of our body is Heart. It's a muscular organ which placed directly behind and slightly left of breastbone. Heart Disease causes highest number of deaths globally, with approximately around 17.9 million people died from it every year which means around 31% of deaths are from the heart disease as per the WHO (World Health Organization). Heart disease are also called as Cardiovascular Disease which are group of complication of the blood vessels and heart which include cerebrovascular disease, rheumatic heart disease and some other heart conditions. Four out of five heart disease deaths are from the strokes and heart attacks. Heart Disease is the most life-threatening disease in the world nowadays. Therefore, heart disease should be predicted at their early stage and healthy lifestyle are ways to prevent them. People who are at risk of heart disease may have symptoms like high blood pressure, obesity, cholesterol, diabetes, age, etc. As there is recent improvement in medical health care is observed. The health care system has collected massive amount of data about heart disease and they have all those data and created datasets which consist of different medical parameter or features such as age, sex, blood pressure, cholesterol, chest type and so on, etc. Datasets consist of around 13 to 15 different medical parameters. These datasets are now available for analysis and to extract crucial information from it. So, we can predict the heart disease at their early stage by applying machine learning algorithms on this massive amount of data to extract features (information/medical parameters) that we will extract from datasets. Various machine learning techniques like logistic regression, naïve bayes, support vector machine, k nearest neighbor (knn), etc. we can use for predicting heart disease means to classify whether a person is having cardiovascular disease or not, after applying them on the features extraction from datasets. Different algorithms will give different accuracy, so after comparing among them we can find the best algorithm which predicts heart disease with highest accuracy. The main objective of our project is to enhance efficiency for predicting heart disease rate.

Scenario:

you have just been hired as a Data Scientist at a Hospital with an alarming number of patients coming in reporting various cardiac symptoms. A cardiologist measures vitals & hands you this data to perform Data Analysis and predict whether certain patients have Heart Disease. We would like to make a Machine Learning algorithm where we can train

our AI to learn & improve from experience. Thus, we would want to classify patients as either positive or negative for Heart Disease.

Goal:

- Predict whether a patient should be diagnosed with Heart Disease. This is a binary outcome.

  Positive (+) = 1, patient diagnosed with Heart Disease

  Negative (-) = 0, patient not diagnosed with Heart Disease
- Experiment with various Classification Models & see which yields greatest accuracy.
- Examine trends & correlations within our data
- Determine which features are most important to Positive/Negative Heart Disease diagnosis

`

Chapter-II

Features & Predictor

Our Predictor (Y, Positive or Negative diagnosis of Heart Disease) is determined by 13 features (X):

1. age (#)
2. sex : 1= Male, 0= Female (Binary)
3. (cp)chest pain type (4 values -Ordinal):Value 1: typical angina ,Value 2: atypical angina, Value 3: non-anginal pain , Value 4: asymptomatic (
4. (trestbps) resting blood pressure (#)
5. (chol) serum cholesterol in mg/dl (#)
6. (fbs)fasting blood sugar > 120 mg/dl(Binary)(1 = true; 0 = false)
7. (restecg) resting electrocardiographic results(values 0,1,2)
8. (thalach) maximum heart rate achieved (#)
9. (exang) exercise induced angina (binary) (1 = yes; 0 = no)
10. (oldpeak) = ST depression induced by exercise relative to rest (#)
11. (slope) of the peak exercise ST segment (Ordinal) (Value 1: upsloping , Value 2: flat , Value 3: downsloping)
12. (ca) number of major vessels (0-3, Ordinal) colored by fluoroscopy
13. (thal) maximum heart rate achieved - (Ordinal): 3 = normal; 6 = fixed defect; 7 = reversible defect

Note: Our data has 3 types of data:

Continuous: Which is quantitative data that can be measured

Ordinal Data: Categorical data that has a order to it (0,1,2,3,etc)

Binary Data: Data whose unit can take on only two possible states(0&1)

Chapter-III

Chapter:III
Methodology

Data Cleaning and Preprocessing:

The datasets which were collected from UCI machine learning repository and Kaggle website contain unfiltered data which must be filtered before the final data set can be used to train the model. Also, data has some categorical variables which must be modified into numerical values for which we used Pandas library of Python. In data cleaning step, first we checked whether there are any missing or junk values in the dataset for which we used the isnull() function. Then for handling categorical variables we converted them into numerical variables

Machine Learning Algorithms:

a) Random Forest :

Random Forest is the most famous and it is considered as the best algorithm for machine learning. It is a supervised learning algorithm. To achieve more accurate and consistent prediction, random forest creates several decision trees and combines them together. The major benefit of using it is its ability to solve both regression and classification issues. When building each individual tree, it employs bagging and feature randomness in order to produce an uncorrelated tree forest whose collective forecast has much better accuracy than any individual tree's prediction. Bagging enhances accuracy of machine learning methods by grouping them together. In this algorithm, during the splitting of nodes it takes only random subset of nodes into an account. When splitting a node, it looks for the best feature from a random group of features rather than the most significant feature. This results into getting better accuracy. It efficiently deals with the huge datasets. It also solves the issue of overfitting in datasets. It works as follows: First, it'll select random samples from the provided dataset. Next, for every selected sample it'll create a decision tree and it'll receive a forecasted result from every created decision tree. Then for each result which was predicted, it'll perform voting and through voting it will select the best predicted result.

b) Logistic Regression :

Logistic regression is often used a lot of times in machine learning for predicting the likelihood of response attributes when a set of explanatory independent attributes are given. It is used when the target attribute is also known as a dependent variable having categorical values like yes/no or true/false, etc. It's widely used for solving classification problems. It falls under the category of supervised machine learning. It efficiently solves linear and

binary classification problems. It is one of the most commonly used and easy to implement algorithms. It's a statistical technique to predict classes which are binary. When the target variable has two possible classes in that case it predicts the likelihood of occurrence of the event. In our dataset the target variable is categorical as it has only two classes-yes/no.

c) Naive Bayes : It is a probabilistic machine learning algorithm which is mainly used in classification problems. 11 | Page It's based on Bayes theorem. It is simple and easy to build. It deals with huge datasets efficiently. It can solve complicated classification problems. The existence of a specific feature in a class is assumed to be independent of the presence of any other feature according to naïve bayes theorem. It's formula is as follows : P(S|T) = P(T|S) * P(S) / P(T) Here, T is the event to be predicted, S is the class value for an event. This equation. will find out the class in which the expected feature for classification.

d) Support Vector Machine (SVM) :

It is a powerful machine learning algorithm that falls under the category of supervised learning. Many people use SVM to solve both regression and classification problems. The primary role of SVM algorithm is that it separates two classes by creating a line of hyperplanes. Data points which are closest to the hyperplane or points of the data set that, if deleted, would change the position of dividing the hyperplane are known as support vectors. As a result, they might be regarded as essential components of the data set. The margin is the distance between hyperplane and nearest data point from either collection. The goal is to select the hyperplane with the maximum possible margin between it and any point in the training set increasing the likelihood of a new data being properly classified. SVM's main objective is to find a hyperplane in N-dimensional space which will classify all the data points. The dimension of a hyperplane is actually dependent on the quantity of input features. If input has two features in that case the hyperplane will be a line and two dimensional plane.

e) K Nearest Neighbor (KNN) :

KNN is a supervised machine learning algorithm. It assumes similar objects are nearer to one another. When the parameters are continuous in that case knn is preferred. In this algorithm it classifies objects by predicting their nearest neighbor. It's simple and easy to implement and also has high speed because of which it is preferred over the other algorithms when it comes to solving classification problems. The algorithm classifies whether or not the patient has disease by taking the heart disease dataset as an input. It takes input parameters like age, sex, chol, etc and classify person with heart disease. Algorithm takes following steps :-

Step 1: Select the value for K.

Step 2 : Find the Euclidean distance of K no. of neighbors.

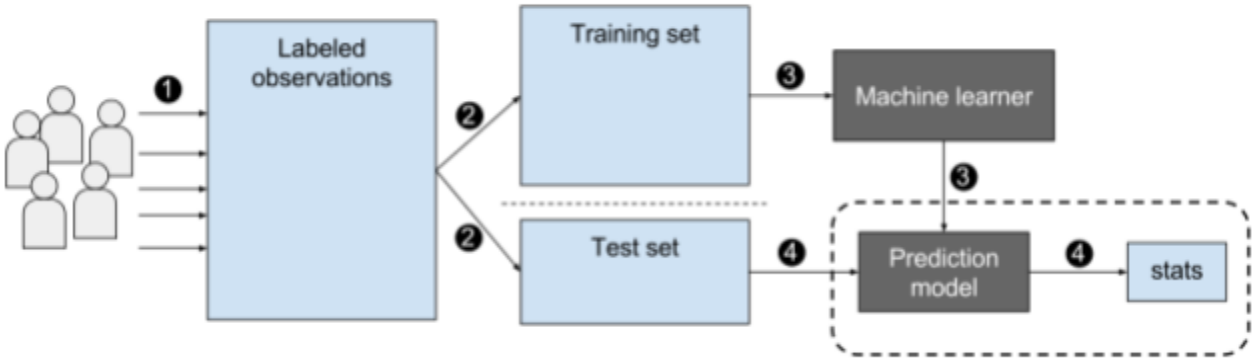
Step 3 : Based on calculated distance, select the K nearest neighbors in the training

data which are nearest to unknown data points.

Step 4 : Calculate no. of data points in each category among these K neighbors.

Step 5 : Assign new data points to the category which has the maximum no. of neighbors.

Step 6 : Stop.



Implementation Steps:

As we already discussed in the methodology section about some of the implementation details. So, the language used in this project is Python programming. We're running python code in anaconda navigator's Jupyter notebook. Jupyter notebook is much faster than Python IDE tools like PyCharm or Visual studio for implementing ML algorithms. The advantage of Jupyter notebook is that while writing code, it's really helpful for Data visualization and plotting some graphs like histogram and heatmap of correlated matrices. Let's revise implementation steps :

a) Dataset collection.

b) Importing Libraries : Numpy, Pandas, Scikit-learn, Matplotlib and Seaborn libraries were used.

c) Exploratory data analysis : For getting more insights about data.

d) Data cleaning and preprocessing : Checked for null and junk values using isnull() and isna().sum() functions of python.In Preprocessing phase, we did feature engineering on our dataset. As we converted categorical variables into numerical variables using function of Pandas library. Both our datasets contains some categorical variables

e) Feature Scaling : In this step, we normalize our data by applying Standardization by using StandardScalar() and fit_transform() functions of scikit-learn library.

f) Model selection : We first separated X's from y's. X's are features or input variables of our datasets and y's are dependent or target variables which are crucial for predicting disease. Then using by the importing model_selection function of the sklearn library, we splitted our X's and y's into train and test split using train_test_split() function

of sklearn. We splitted 80% of our data for training and 20% for testing. g) Applied ML models and created a confusion matrix of all models. h) Deployment of the model which gave the best accuracy.

LIbRARYS

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from matplotlib import rcParams
from matplotlib.cm import rainbow
%matplotlib inline
import warnings
warnings.filterwarnings('ignore')

from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn import metrics
```

Data Wrangling

```
data=pd.read_csv("heart_disease_data.csv")
data.head()
```

	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

```
data.shape
```

```
(303, 14)
```

```
data.nunique(axis=0) #returns the number of unique value for each variable
```

```
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
```

```
data.info()
```

```
#summarizes the count, mean, standard deviation, min, and max for numeric variables.
data.describe()
```

	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

```
print(data.isna().sum()) # Display the Missing values
```

```
age         0
sex         0
cp          0
trestbps    0
chol        0
fbs         0
restecg     0
thalach     0
exang       0
oldpeak     0
slope       0
ca          0
thal        0
target      0
dtype: int64
```

Let's see if theirs a good proportion between our positive and negative results. It appears we have a good balance between the two.

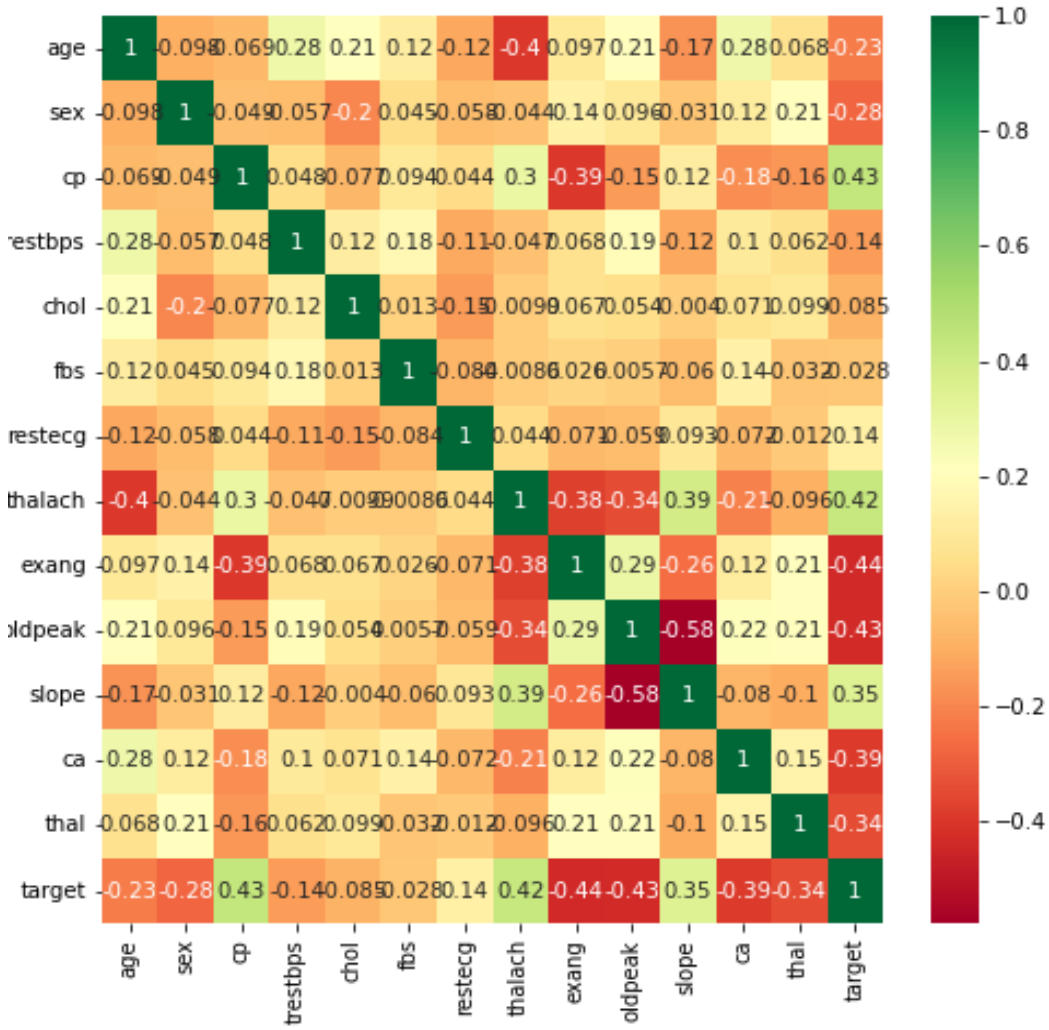
```
data['target'].value_counts()
```

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

Exploratory Data Analysis

Correlation Matrix- let's you see correlations between all variables. Within seconds, you can see whether something is positively or negatively correlated with our predictor (target).

```
import seaborn as sns
#get correlations of each features in dataset
corrmat = data.corr()
top_corr_features = corrmat.index
plt.figure(figsize=(20,20))
#plot heat map
sns.heatmap(data[top_corr_features].corr(),annot=True,cmap="RdYlGn")
```



We can see there is a positive correlation between chest pain (cp) & target (our predictor). This makes sense since, The greater amount of chest pain results in a greater chance of having heart disease. Cp (chest pain), is a ordinal feature with 4 values: Value 1: typical angina ,Value 2: atypical angina, Value 3: non-anginal pain , Value 4: asymptomatic.

In addition, we see a negative correlation between exercise induced angina (exang) & our predictor. This makes sense because when you exercise, your heart requires more blood, but narrowed arteries slow down blood flow.

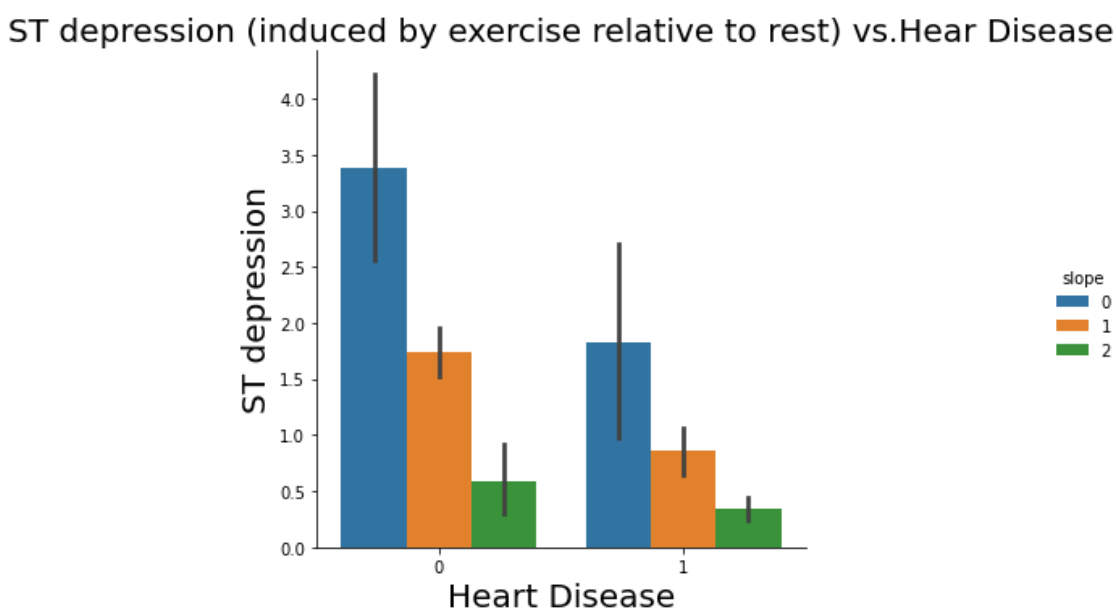
Pair Plots are also a great way to immediately see the correlations between all variables. But you will see me make it with only continuous columns from our data, because with so many features, it can be difficult to see each one. So instead I will make a pairplot with only our continuous features.

```
data.hist(figsize=(15,16))

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

Chose to make a smaller pairplot with only the continuous variables, to dive deeper into the relationships. Also a great way to see if theirs a positive or negative correlation!

```
sns.catplot(x="target",y="oldpeak",hue="slope",kind="bar",data=data);
plt.title("ST depression (induced by exercise relative to rest) vs Hear Disease",size=20)
plt.xlabel("Heart Disease",size=20)
plt.ylabel("ST depression",size=20)
```



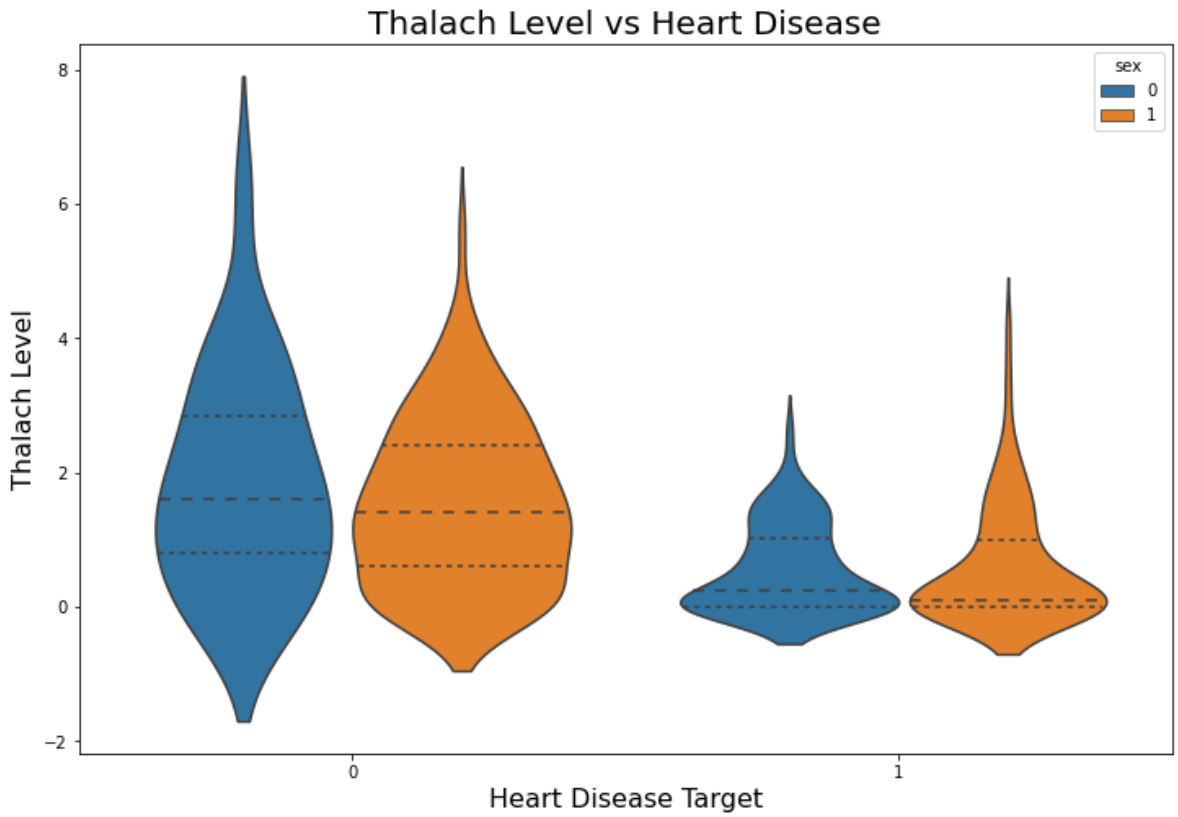
ST segment depression occurs because when the ventricle is at rest and therefore repolarized. If the trace in the ST segment is abnormally low below the baseline, this can lead to this Heart Disease. This is supports the plot above because low ST Depression

yields people at greater risk for heart disease. While a high ST depression is considered normal & healthy. The "slope" hue, refers to the peak exercise ST segment, with values: 0: upsloping , 1: flat , 2: downsloping). Both positive & negative heart disease patients exhibit equal distributions of the 3 slope categories.

Violin & Box plots

The advantages of showing the Box & Violin plots is that it shows the basic statistics of the data, as well as its distribution. These plots are often used to compare the distribution of a given variable across some categories.

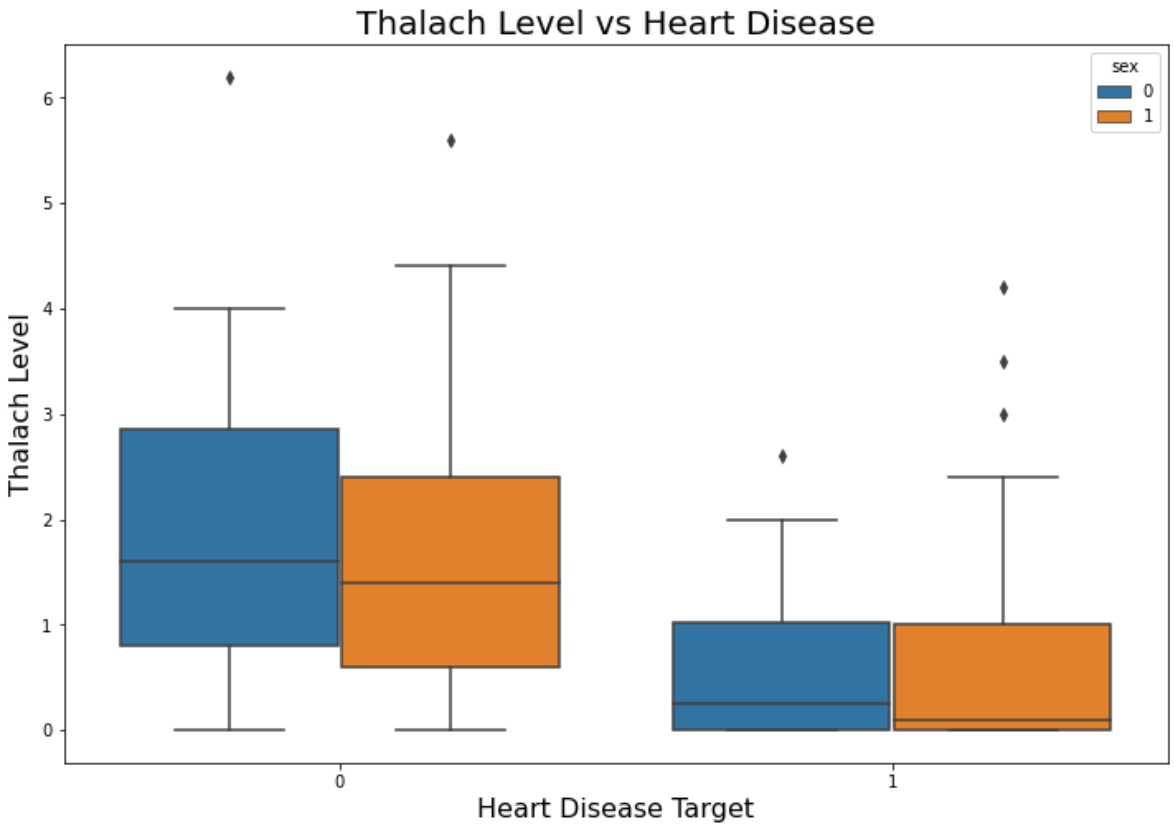
We can see that the overall shape & distribution for negative & positive patients differ vastly. Positive patients exhibit a lower median for ST depression level & thus a great distribution of their data is between 0 & 2, while negative patients are between 1 & 3. In addition, we dont see many differences between male & female target outcomes.



BOX Plots

```
plt.figure(figsize=(12,8))
sns.boxplot(x= 'target', y= 'thalach',hue="sex", data=data )
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)
```

Positive patients exhibit a heightened median for ST depression level, while negative patients have lower levels. In addition, we don't see many differences between male & female target outcomes, expect for the fact that males have slightly larger ranges of ST Depression.



Filtering data by positive & negative Heart Disease patient

```
#positive
pos_data=data[data["target"]==1]
pos_data.describe()

#negative
neg_data=data[data["target"]==0]
```

neg_data.describe()

Positive&Negative Patients ST depression

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

```
(Positive Patients ST depression):0.5830303030303029
(Negative Patients ST depression):1.5855072463768118
```

Positive&Negative Patients thalach

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

```
Positive Patients thalach):158.46666666666667
Negative Patients thalach):139.1014492753623
```

From comparing positive and negative patients we can see there are vast differences in means for many of our Features. From examining the details, we can observe that positive patients experience heightened maximum heart rate achieved (thalach) average. In addition, positive patients exhibit about 1/3rd the amount of ST depression induced by exercise relative to rest (oldpeak).

Machine learning and Predictive Analytics:

(i) Prepare the data

To prepare data for modeling, just remember **ASN (Assign,Split, Normalize).**

- **Assign** the 13 features to X, & the last column to our classification predictor,y

  X=data.iloc[:,:-1].values

  y=data.iloc[:,-1].values

- **Split**: the data set into the Training set and Test set,

  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)

- **Normalize**: Standardizing the data will transform the data so that its distribution will have a mean of 0 and a standard deviation of 1.

  from sklearn.preprocessing import StandardScaler

  sc = StandardScaler()

  X_train = sc.fit_transform(X_train)

  X_test = sc.transform(X_test)

(iii).Modeling /Training

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.

Note: these are all supervised learning models.

Model 1:Logistic Regression

```
model = LogisticRegression()
```

```
model.fit(X_train, y_train)
```

```
LogisticRegression()
```

```python
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_test) # get y predictions
print(classification_report(y_test, y_pred1)) # output accuracy
```

```
              precision    recall  f1-score   support

           0       0.77      0.67      0.71        30
           1       0.71      0.81      0.76        31

    accuracy                           0.74        61
   macro avg       0.74      0.74      0.74        61
weighted avg       0.74      0.74      0.74        61
```

ACCURACY 74%

Model2: K-NN (K-Nearest Neighbors)

```python
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
```

ACCURACY 75%

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
```

ACCURACY 75%

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
```

ACCURACY 77%

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
```

ACCURACY 69%

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
```

ACCURACY 80%

Model 7: XGBoost

```
!pip install xgboost
```

```
Requirement already satisfied: xgboost in c:\users\saravana\anaconda3\lib\site-packages (1.6.1)
Requirement already satisfied: numpy in c:\users\saravana\anaconda3\lib\site-packages (from xgboost) (1.20.3)
Requirement already satisfied: scipy in c:\users\saravana\anaconda3\lib\site-packages (from xgboost) (1.7.1)
```

```
import xgboost as xgb
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
```

ACCURACY 79%

From comparing the 7 models, we can conclude that Model 6: Random Forest yields the highest accuracy. With an accuracy of 80%.

We have precision, recall, f1-score and support:

Precision : be "how many are correctly classified among that class"

Recall : "how many of this class you find over the whole number of element of this class"

F1-score : harmonic mean of precision and recall values. F1 score reaches its best value at 1 and worst value at 0. F1 Score = 2 x ((precision x recall) / (precision + recall))

Support: of samples of the true response that lie in that class.

Measuring Model Performance

While there are other ways of measuring model performance (precision, recall, F1 Score, ROC Curve, etc), let's keep this simple and use accuracy as our metric. To do this are going to see how the model performs on new data (test set)

Accuracy is defined as: (fraction of correct predictions): correct predictions / total number of data points

		Predicted	
Actual		Positive	Negative
	Positive	True Positive (TP)	False Negative (FN)
	Negative	False Positive (FP)	True Negative (TN)

Making the Confusion Matrix

```
from sklearn.metrics import confusion_matrix, accuracy_score
cm = confusion_matrix(y_test, y_pred6)
print(cm)
accuracy_score(y_test, y_pred6)
```

```
[[21  9]
 [ 3 28]]
0.8032786885245902
```

21 is the amount of True Positives in our data, while 28 is the amount of True Negatives.

9 & 3 are the number of errors.

There are 9 type 1 error (False Positives)- You predicted positive and it's false.

There are 3 type 2 error (False Negatives)- You predicted negative and it's false.

Hence if we calculate the accuracy its # Correct Predicted/ # Total. In other words, where TP, FN, FP and TN represent the number of true positives, false negatives, false positives and true negatives.

(TP + TN)/(TP + TN + FP + FN). (21+28)/(21+28+9+3) = 0.80 = 80% accuracy

Note: A good rule of thumb is that any accuracy above 70% is considered good, but be careful because if your accuracy is extremly high, it may be too good to be true (an example of Overfitting). Thus, 80% is the ideal accuracy!

Feature Importance

Feature Importance provides a score that indicates how helpful each feature was in our model.

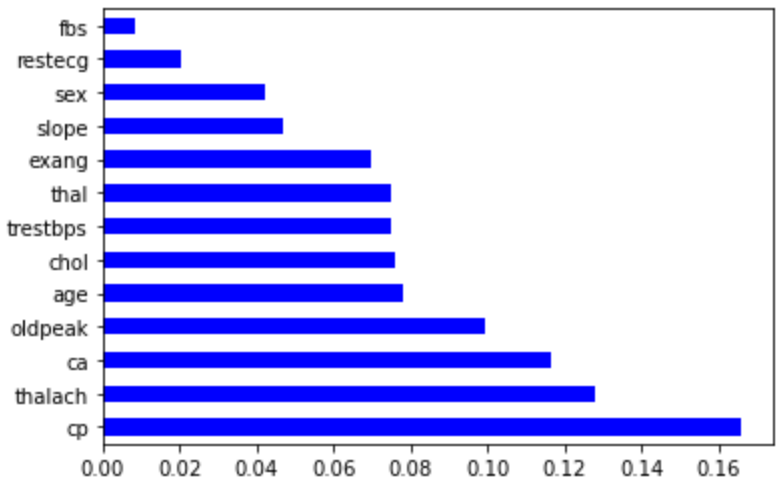
The higher the Feature Score, the more that feature is used to make key decisions & thus the more important it is.

```
# get importance
importance = model6.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
```

```
index= data.columns[:-1]
importance = pd.Series(model6.feature_importances_, index=index)
importance.nlargest(13).plot(kind='barh', colormap='winter')
#plt.xticks(rotation=45)
```



From the Feature Importance graph above, we can conclude that the top 4 significant features were chest pain type (cp), maximum heart rate achieved (thalach), number of major vessels (ca), and ST depression induced by exercise relative to rest (oldpeak).

Chapter-IV

Analysis of the Result

We used precision, F1-score, recall and accuracy evaluation metrics for evaluating our models. False Positive(FP) is when a model incorrectly predicts a positive outcome. False Negative(FN) is when a model incorrectly predicts the negative outcome. True Positive(TP) is when model correctly predicts a positive outcome. True Negative(TN) is when a model correctly predicts a negative outcome. 13 | Page Precision=TP/(TP+FP) Recall = TP / (TP + FN)

Machine Learning Models	Accuracy
Logistic Regression	74%
KNN	75%
SVM	75%
Naive Bayes	77%
Decision Tree	69%
Random Forest	80%
XGBoost	79%

Predictions:

Scenario: A patient develops cardiac symptoms & you input his vitals into the Machine Learning Algorithm.

He is a 20 year old male, with a chest pain value of 2 (atypical angina), with resting blood pressure of 110.

In addition he has a serum cholestoral of 230 mg/dl.

He is fasting blood sugar > 120 mg/dl.

He has a resting electrocardiographic result of 1.

The patients maximum heart rate achieved is 140.

Also, he was exercise induced angina.

His ST depression induced by exercise relative to rest value was 2.2.

The slope of the peak exercise ST segment is flat.

He has no major vessels colored by fluoroscopy, and in addition his maximum heart rate achieved is a reversable defect.

Based on this information, can you classify this patient with Heart Disease?

```
print(model6.predict(sc.transform([[20,1,2,110,230,1,1,140,1,2.2,2,0,2]])))
```

```
[1]
```

Yes! Our machine learning algorithm has classified this patient with Heart Disease. Now we can properly diagnose him, & get him the help he needs to recover. By diagnosing him early, we may prevent worse symptoms from arising later.

Predicting the Test set results:

First value represents our predicted value, Second value represents our actual value.

If the values match, then we predicted correctly. We can see that our results are very accurate!

```
y_pred = model6.predict(X_test)
print(np.concatenate((y_pred.reshape(len(y_pred),1), y_test.reshape(len(y_test),1)),1))
```

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

Building a Predictive System

```
input_data = (37,1,2,130,250,0,1,187,0,3.0,0,2,1)

# change the input data to a numpy array
input_data_as_numpy_array= np.asarray(input_data)

# reshape the numpy array as we are predicting for only on instance
input_data_reshaped = input_data_as_numpy_array.reshape(1,-1)

prediction = model.predict(input_data_reshaped)
print(prediction)

if (prediction[0]== 0):
  print('The Person does not have a Heart Disease')
else:
  print('The Person has Heart Disease')
```

```
[1]
The Person has Heart Disease
```

Chapter-V

Conclusions

- Our Random Forest algorithm yields the highest accuracy, 80%. Any accuracy above 70% is considered good, but be careful because if your accuracy is extremely high, it may be too good to be true (an example of Overfitting). Thus, 80% is the ideal accuracy!
- Out of the 13 features we examined, the top 4 significant features that helped us classify between a positive & negative Diagnosis were chest pain type (cp), maximum heart rate achieved (thalach), number of major vessels (ca), and ST depression induced by exercise relative to rest (oldpeak).
- Our machine learning algorithm can now classify patients with Heart Disease. Now we can properly diagnose patients, & get them the help they needs to recover. By diagnosing detecting these features early, we may prevent worse symptoms from arising later.

Here is access to the data set & code from my GitHub page:

https://github.com/Saravanakumargovin/Predicting-Heart-Disease---final-Project

References

References

1. UMAIR SHAFIQUE, FIAZ MAJEED, HASEEB QAISER, IRFAN UL MUSTAFA "DATA MINING IN HEALTHCARE FOR HEART DISEASES", INTERNATIONAL JOURNAL OF INNOVATION AND APPLIED STUDIES ISSN 2028-9324 VOL. 10, ISSUE 4, (2015), PP. 1312-1322

2. V. SABARINATHAN "DIAGNOSIS OF HEART DISEASE USING DECISION TREE", INTERNATIONAL JOURNAL OF RESEARCH IN COMPUTER APPLICATIONS & INFORMATION TECHNOLOGY VOL. 2, ISSUE 6, (2014), PP. 74-79

3. VIKAS CHAURASIA, "DATA MINING APPROACH TO DETECT HEART DISESES", INTERNATIONAL JOURNAL OF ADVANCED COMPUTER SCIENCE AND INFORMATION TECHNOLOGY (IJACSIT) VOL. 2, ISSUE 4, 2013, PAGE: 56-66, ISSN: 2296-1739

4. G. SARIKA SINDHU, "ANALYSIS AND PREDICTION OF CARDIOVASCULAR DISEASE USING MACHINE LEARNING CLASSIFIERS", INTERNATIONAL CONFERENCE ON ADVANCED COMPUTING & COMMUNICATION SYSTEMS (ICACCS) APRIL 2020.

5. MALKARI BHARGAV AND J. RAGHUNATH, "A STUDY ON RISK PREDICTION OF CARDIOVASCULAR DISEASE USING MACHINE LEARNING ALGORITHMS", INTERNATIONAL JOURNAL OF EMERGING TECHNOLOGIES AND INNOVATIVE RESEARCH (WWW.JSTOR.ORG), ISSN:2349-5162, VOL.7, ISSUE 8, PAGE NO.683-688, AUGUST 2020