Chapter 1

ABSTRACT

As per the recent study by WHO, heart related diseases are increasing. 17.9 million people die every-year due to this. With a growing population, it gets further difficult to diagnose and start treatment at an early stage. But due to the recent advancement in technology, Machine Learning techniques have accelerated the health sector by multiple researches. Thus, the objective of this paper is to build a ML model for heart disease prediction based on the related parameters. We have used a benchmark dataset of UCI Heart disease prediction for this research work, which consist of 14 different parameters related to Heart Disease. Machine Learning algorithms such as Random Forest, Support Vector Machine (SVM), Naive Bayes and Decision tree have been used for the development of models. In our research we have also tried to find the correlations between the different attributes available in the dataset with the help of standard Machine Learning methods and then using them efficiently in the prediction of chances of Heart disease. Result shows that compared to other ML techniques, Random Forest gives more accuracy in less time for the prediction. This model can be helpful to the medical practitioners at their clinic as a decision support system.

Chapter 2

INTRODUCTION

Predicting and diagnosing heart disease is the biggest challenge in the medical industry and relies on factors such as the physical examination, symptoms and signs of the patient.

Factors that influence heart disease are body cholesterol levels, smoking habit and obesity, family history of illnesses, blood pressure, and work environment. Machine learning algorithms play an essential and precise role in the prediction of heart disease.

Advances in technology allow machine language to combine with Big Data tools to manage unstructured and exponentially growing data. Heart disease is seen as the world's deadliest disease of human life. In particular, in this type of disease, the heart is not able to push the required amount of blood to the remaining organs of the human body to perform regular functions.

Heart disease can be predicted based on various symptoms such as age, gender, heart rate, etc. and reduces the death rate of heart patients.

Due to the increasing use of technology and data collection, we can now predict heart disease using machine learning algorithms. Now let's go further with the task of heart disease prediction using machine learning with Python.

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

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 electrocardiography 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: up sloping , Value 2: flat , Value 3: down sloping)
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 3

SOFTWARE REQUIREMENTS

Python :

- Python is a high-level, general-purpose programming language.
- Its language constructs and object-oriented approach aim to help programmers write clear, logical code for small and large-scale projects. Python is dynamically typed AND supports multiple programming paradigms, including procedural, object-oriented, and functional programming.

Libraries :

- **numpy :-**
- **pandas :-**
- **matplotlib :-**
- **seaborn :-**
- **sklearn :-**

Operating System :

- Program is tested on Windows 10

Hardware Requirements Specification :

- Laptop / Computer

Chapter 4

Logistic Regression Algorithm

Logistic Regression is a popular statistical model used for binary classification, that is for predictions of the type this or that, yes or no, A or B, etc. Logistic regression can, however, be used for multiclass classification, but here we will focus on its simplest application. It is one of the most frequently used machine learning algorithms for binary classifications that translates the input to 0 or 1. For example,

- 0: negative class
- 1: positive class

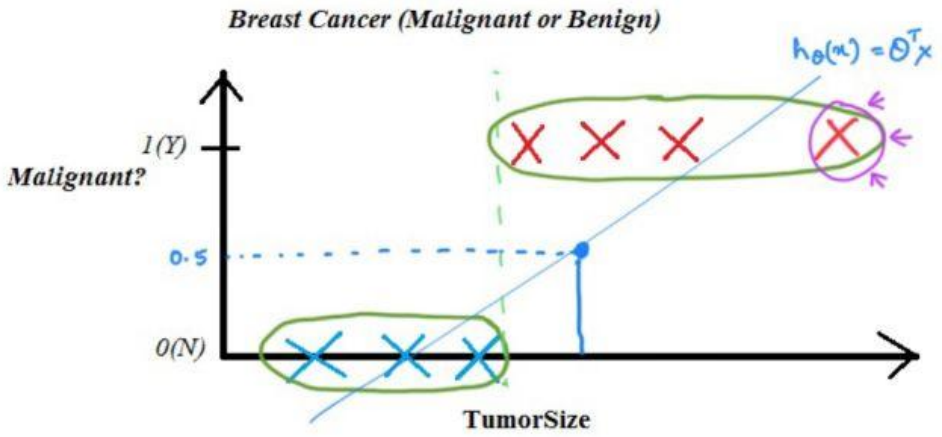
Some examples of classification are mentioned below:

- Email: spam / not spam
- Online transactions: fraudulent / not fraudulent
- Tumor: malignant / not malignant

Let us look at the issues we encounter in Linear Regression.

Issue 1 of Linear Regression

As you can see on the graph mentioned below, the prediction would leave out malignant tumors as the gradient becomes less steep with an additional data point on the extreme right.



→ Threshold classifier output $h_\theta(x)$ at 0.5:

→ If $h_\theta(x) \geq 0.5$, predict "y = 1"

If $h_\theta(x) < 0.5$, predict "y = 0"

Issue 2 of Linear Regression

- Hypothesis can be larger than 1 or smaller than zero
- Hence, we have to use logistic regression

What is Logistic Regression?

Logistic Regression is the appropriate regression analysis to conduct when the dependent variable has a binary solution. Similar to all other types of regression systems, Logistic Regression is also a type of predictive regression system. Logistic regression is used to evaluate the relationship between one dependent binary variable and one or more independent variables. It gives discrete outputs ranging between 0 and 1.

A simple example of Logistic Regression is: Does calorie intake, weather, and age have any influence on the risk of having a heart attack? The question can have a discrete answer, either "yes" or "no".

Logistic Regression Hypothesis

The logistic regression classifier can be derived by analogy to the linear regression hypothesis which is:

$$h_\theta(\mathbf{x}) = \theta^\top \mathbf{x}$$

Linear regression hypothesis

However, the logistic regression hypothesis generalizes from the linear regression hypothesis in that it uses the logistic function:

$$h_\theta(\mathbf{x}) = g(\theta^\top \mathbf{x})$$
$$g(z) = \frac{1}{1+e^{-z}}$$

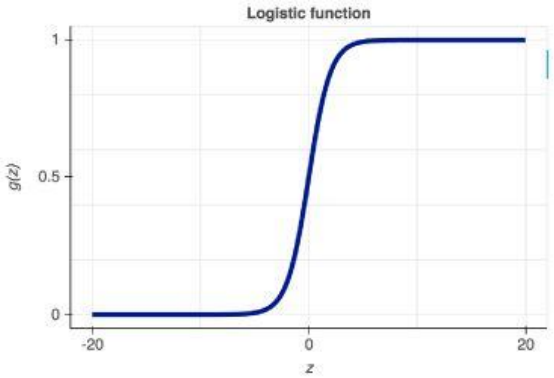
The result is the logistic regression hypothesis:

$$h_\theta(\mathbf{x}) = \frac{1}{1+e^{-\theta^\top \mathbf{x}}}$$

Logistic regression hypothesis

The function g(z) is the logistic function, also known as the sigmoid function.

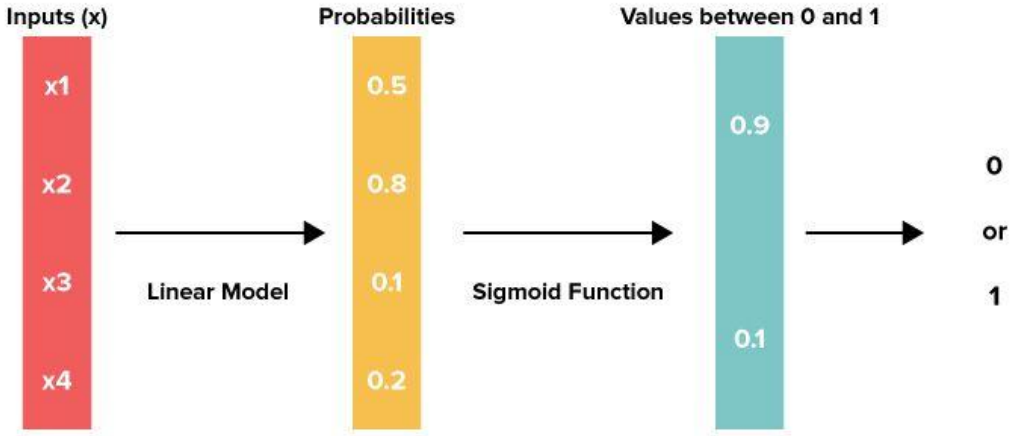
The logistic function has asymptotes at 0 and 1, and it crosses the y-axis at 0.5.



How Logistic Regression works?

Logistic Regression uses a more complex cost function than Linear Regression, this cost function is called the 'Sigmoid function' or also known as the 'logistic function' instead of a linear function.

The hypothesis of logistic regression tends to limit the cost function between 0 and 1. Therefore linear functions fail to represent it as it can have a value greater than 1 or less than 0 which is not possible as per the hypothesis of logistic regression.



Sigmoid function maps any real value into another value between 0 and 1. In machine learning, we use sigmoid to map predictions to probabilities.

Formula:

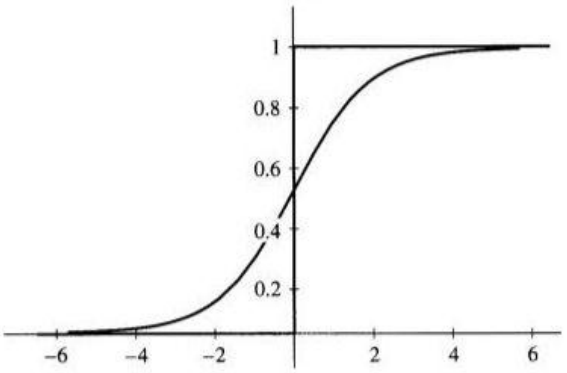
$$f(x) = \frac{1}{1+e^{-(x)}}$$

Where,

f(x) = output between 0 and 1 (probability estimate)

x = input to the function

e = base of natural log



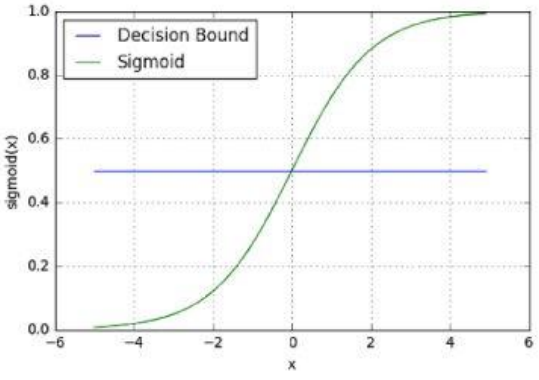
Decision Boundary

The prediction function returns a probability score between 0 and 1. If you want to map the discrete class (true/false, yes/no), you will have to select a threshold value above which you will be classifying values into class 1 and below the threshold value into class 2.

$p \geq 0.5, class=1$

$p < 0.5, class=0$

For example, suppose the threshold value is 0.5 and your prediction function returns 0.7, it will be classified as positive. If your predicted value is 0.2, which is less than the threshold value, it will be classified as negative. For logistic regression with multiple classes we could select the class with the highest predicted probability.



Our aim should be to maximize the likelihood that a random data point gets classified correctly, which is called Maximum Likelihood Estimation. Maximum Likelihood Estimation is a general approach to estimating parameters in statistical models. The likelihood can be

maximized using an optimization algorithm. Newton's Method is one such algorithm which can be used to find maximum (or minimum) of many different functions, including the likelihood function. Other than Newton's Method, you can also use Gradient Descent.

Cost Function

We have covered Cost Function earlier in the blog on Linear Regression. In brief, a cost function is created for optimization purpose so that we can minimize it and create a model with minimum error.

Cost function for Logistic Regression are:

- $\text{Cost}(h\theta(x),y) = -\log(h\theta(x))$ if $y = 1$
- $\text{Cost}(h\theta(x),y) = -\log(1-h\theta(x))$ if $y = 0$

The above functions can be written together as:

$$J(\theta) = -\frac{1}{m}\sum\left[y^{(i)}\log(h\theta(x(i))) + \left(1-y^{(i)}\right)\log(1-h\theta(x(i)))\right]$$

Gradient Descent

After finding out the cost function for Logistic Regression, our job should be to minimize it i.e. min J(θ). The cost function can be reduced by using Gradient Descent.

The general form of gradient descent:

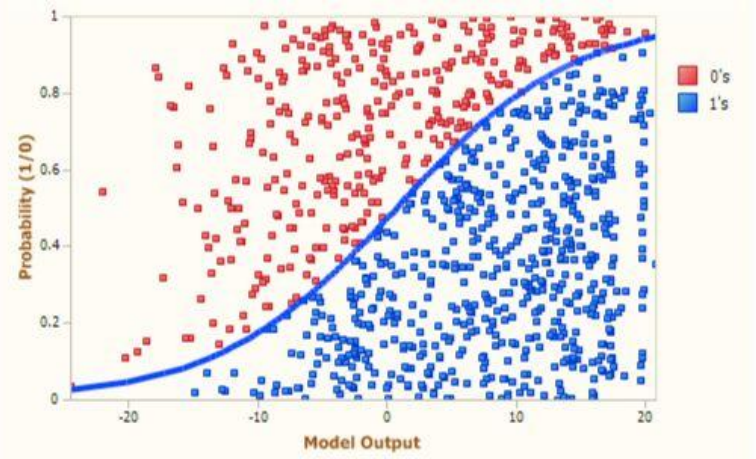
$$\theta_j := \theta_j - \alpha\frac{\partial}{\partial\theta_j}J(\theta)$$

The derivative part can be solved using calculus so the equation comes to:

$$\theta_j := \theta_j - \frac{\alpha}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})x_j^{(i)}$$

When to use Logistic Regression?

Logistic Regression is used when the input needs to be separated into “two regions” by a linear boundary. The data points are separated using a linear line as shown:



Based on the number of categories, Logistic regression can be classified as:

1. binomial: target variable can have only 2 possible types: “0” or “1” which may represent “win” vs “loss”, “pass” vs “fail”, “dead” vs “alive”, etc.
2. multinomial: target variable can have 3 or more possible types which are not ordered(i.e. types have no quantitative significance) like “disease A” vs “disease B” vs “disease C”.
3. ordinal: it deals with target variables with ordered categories. For example, a test score can be categorized as:“very poor”, “poor”, “good”, “very good”. Here, each category can be given a score like 0, 1, 2, 3.

Let us explore the simplest form of Logistic Regression, i.e Binomial Logistic Regression. It can be used while solving a classification problem, i.e. when the y-variable takes on only two values. Such a variable is said to be a “binary” or “dichotomous” variable. “Dichotomous” basically means two categories such as yes/no, defective/non-defective, success/failure, and so on. “Binary” refers to the 0's and 1’s.

Chapter 4

Heart Disease Prediction Using Logistic Regression in Machine Learning

4.1 Importing Libraries

We are going to use the Python programming language for this task of heart disease prediction so let's start by importing some necessary libraries:

```
import numpy as np
import pandas as pd
import matplotlib as plt
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
from sklearn.model_selection import train_test_split
from sklearn import metrics
from sklearn.linear_model import LogisticRegression
```

Now let's import the data and move further:

```
data = pd.read_csv('heart_disease.csv')
data.head(10)
```

	Age	Sex	ChestPain	RestBP	Chol	Fbs	RestECG	MaxHR	ExAng	Oldpeak	Slope	Ca	Thal	AHD
1	63	1	typical	145	233	1	2	150	0	2.3	3	0.0	fixed	No
2	67	1	asymptomatic	160	286	0	2	108	1	1.5	2	3.0	normal	Yes
3	67	1	asymptomatic	120	229	0	2	129	1	2.6	2	2.0	reversable	Yes
4	37	1	nonanginal	130	250	0	0	187	0	3.5	3	0.0	normal	No
5	41	0	nontypical	130	204	0	2	172	0	1.4	1	0.0	normal	No
6	56	1	nontypical	120	236	0	0	178	0	0.8	1	0.0	normal	No
7	62	0	asymptomatic	140	268	0	2	160	0	3.6	3	2.0	normal	Yes
8	57	0	asymptomatic	120	354	0	0	163	1	0.6	1	0.0	normal	No
9	63	1	asymptomatic	130	254	0	2	147	0	1.4	2	1.0	reversable	Yes
10	53	1	asymptomatic	140	203	1	2	155	1	3.1	3	0.0	reversable	Yes

4.2 Data analysis

Before training the logistic regression we need to observe and analyse the data to see what we are going to work with. The goal here is to learn more about the data

find answers to some important questions such as:

- What question (s) are you trying to solve?
- What kind of data do we have and how do we handle the different types?
- What is missing in the data and how do you deal with it?
- Where are the outliers and why should you care?
- How can you add, change, or remove features to get the most out of your data?

```
Int64Index: 303 entries, 1 to 303
Data columns (total 14 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   Age        303 non-null    int64
 1   Sex        303 non-null    int64
 2   ChestPain  303 non-null    object
 3   RestBP     303 non-null    int64
 4   Chol       303 non-null    int64
 5   Fbs        303 non-null    int64
 6   RestECG    303 non-null    int64
 7   MaxHR      303 non-null    int64
 8   ExAng      303 non-null    int64
 9   Oldpeak    303 non-null    float64
 10  Slope      303 non-null    int64
 11  Ca         299 non-null    float64
 12  Thal       301 non-null    object
 13  AHD        303 non-null    object
dtypes: float64(2), int64(9), object(3)
memory usage: 35.5+ KB
None
```

In our Data there are some Null spaces, before starting to analyze the data we have to fill that Null space with ‘Zero’ OR any value which is suitable in that column.

```
data.isnull().sum()
```

```
Age          0
Sex          0
ChestPain    0
RestBP       0
Chol         0
Fbs          0
RestECG      0
MaxHR        0
ExAng        0
Oldpeak      0
Slope        0
Ca           4
Thal         2
AHD          0
dtype: int64
```

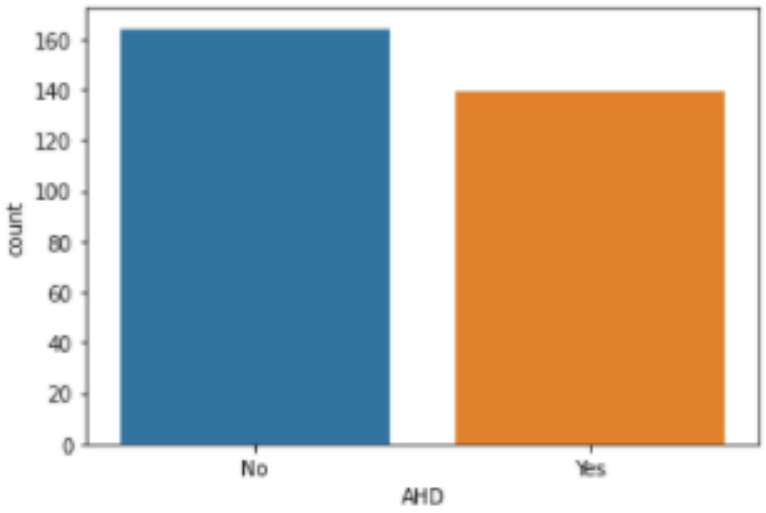
```
data["Ca"].replace(to_replace = np.nan, value = 0,inplace = True)
data["Thal"].replace(to_replace = np.nan, value = "normal",inplace = True)
data.isnull().sum()
```

```
Age          0
Sex          0
ChestPain    0
RestBP       0
Chol         0
Fbs          0
RestECG      0
MaxHR        0
ExAng        0
Oldpeak      0
Slope        0
Ca           0
Thal         0
AHD          0
dtype: int64
```

Let's see if there's a good proportion between our positive & negative **binary predictor.**

```
sns.countplot(x="AHD", data=data)
```

```
<AxesSubplot:xlabel='AHD', ylabel='count'>
```



For Better optimization we are replacing some values

Yes to 1 and No to 0 in column ‘AHD’

typical to 0, non typical to 1, nonanginal to 2, and asymptomatic to 3 in column ‘Chestpain’

normal to 1, fixed to 2, reversible to 3 in column ‘Thal’

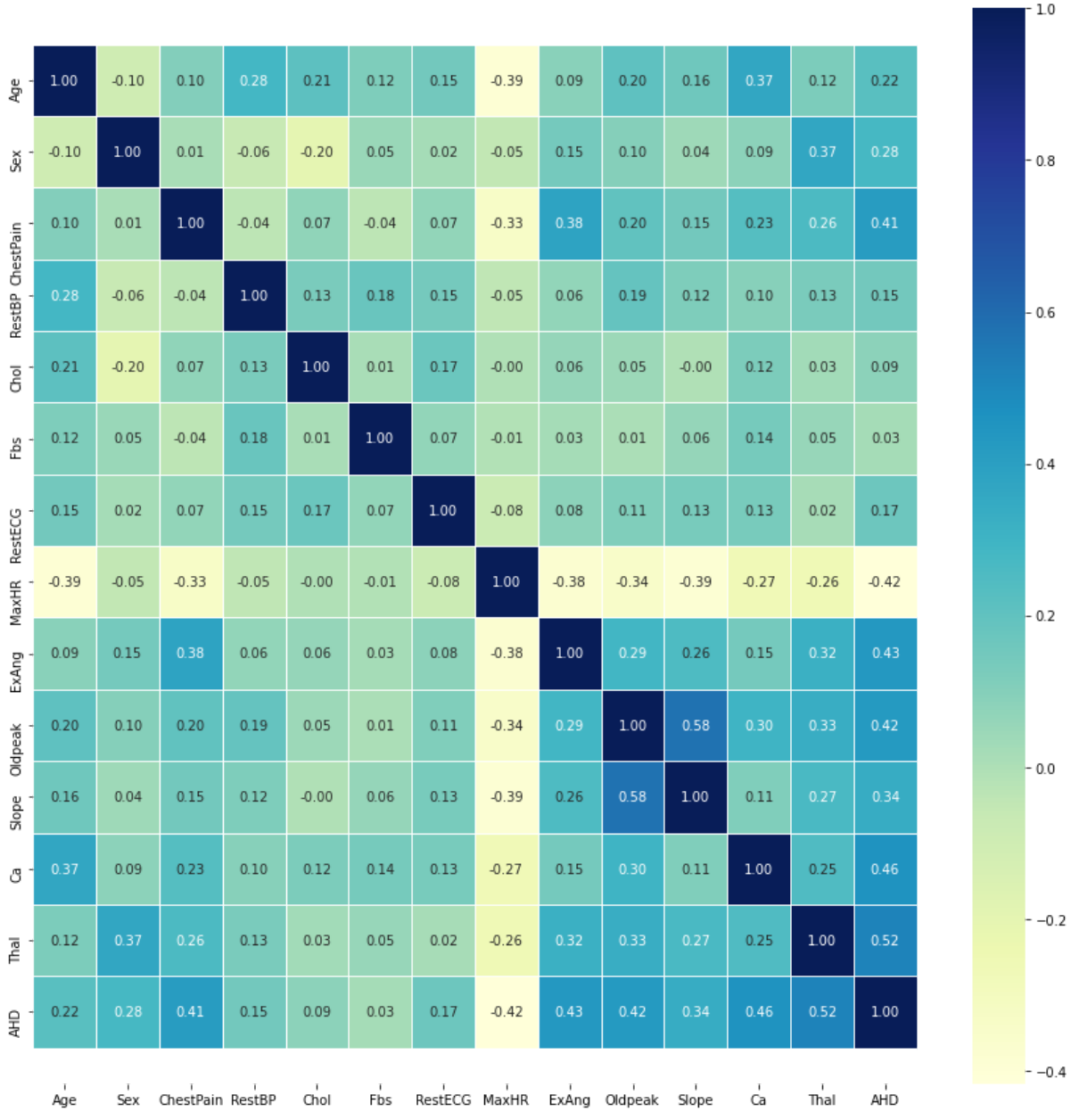
```
data = data.replace({'AHD':{'Yes': 1,'No': 0}})
data = data.replace({'ChestPain':{'typical': 0,'nontypical': 1,'nonanginal': 2,'asymptomatic': 3}})
data = data.replace({'Thal':{'normal': 1,'fixed': 2,'reversable': 3}})
data
```

	Age	Sex	ChestPain	RestBP	Chol	Fbs	RestECG	MaxHR	ExAng	Oldpeak	Slope	Ca	Thal	AHD
1	63	1	0	145	233	1	2	150	0	2.3	3	0.0	2	0
2	67	1	3	160	286	0	2	108	1	1.5	2	3.0	1	1
3	67	1	3	120	229	0	2	129	1	2.6	2	2.0	3	1
4	37	1	2	130	250	0	0	187	0	3.5	3	0.0	1	0
5	41	0	1	130	204	0	2	172	0	1.4	1	0.0	1	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
299	45	1	0	110	264	0	0	132	0	1.2	2	0.0	3	1
300	68	1	3	144	193	1	0	141	0	3.4	2	2.0	3	1
301	57	1	3	130	131	0	0	115	1	1.2	2	1.0	3	1
302	57	0	1	130	236	0	2	174	0	0.0	2	1.0	1	1
303	38	1	2	138	175	0	0	173	0	0.0	1	0.0	1	0

303 rows × 14 columns

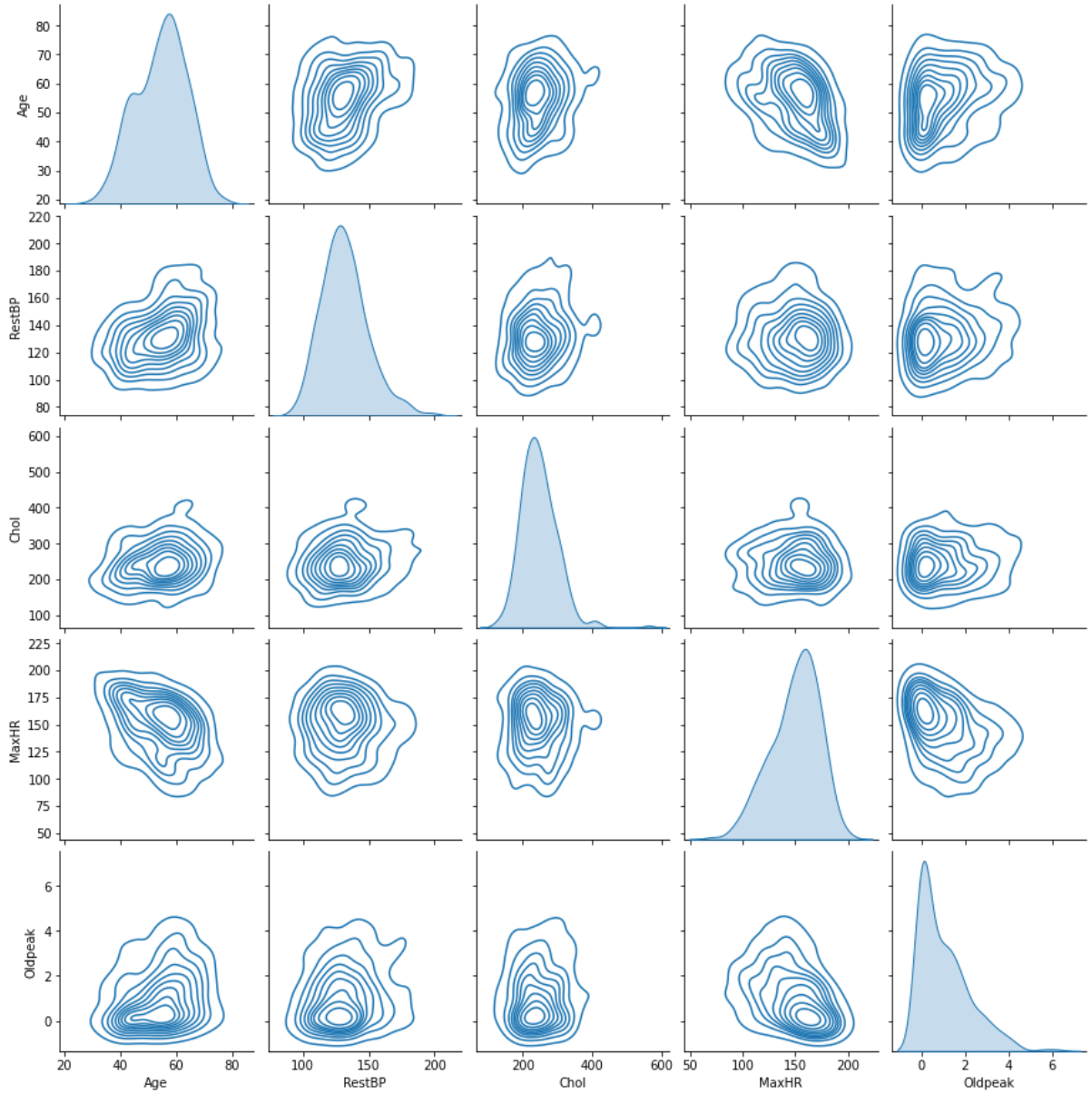
Correlation between all the variables

```
corr_matrix = data.corr()
fig, ax = plt.subplots(figsize=(15, 15))
ax = sns.heatmap(corr_matrix,
                 annot=True,
                 linewidths=0.5,
                 fmt=".2f",
                 cmap="YlGnBu");
bottom, top = ax.get_ylim()
ax.set_ylim(bottom + 0.5, top - 0.5)
```



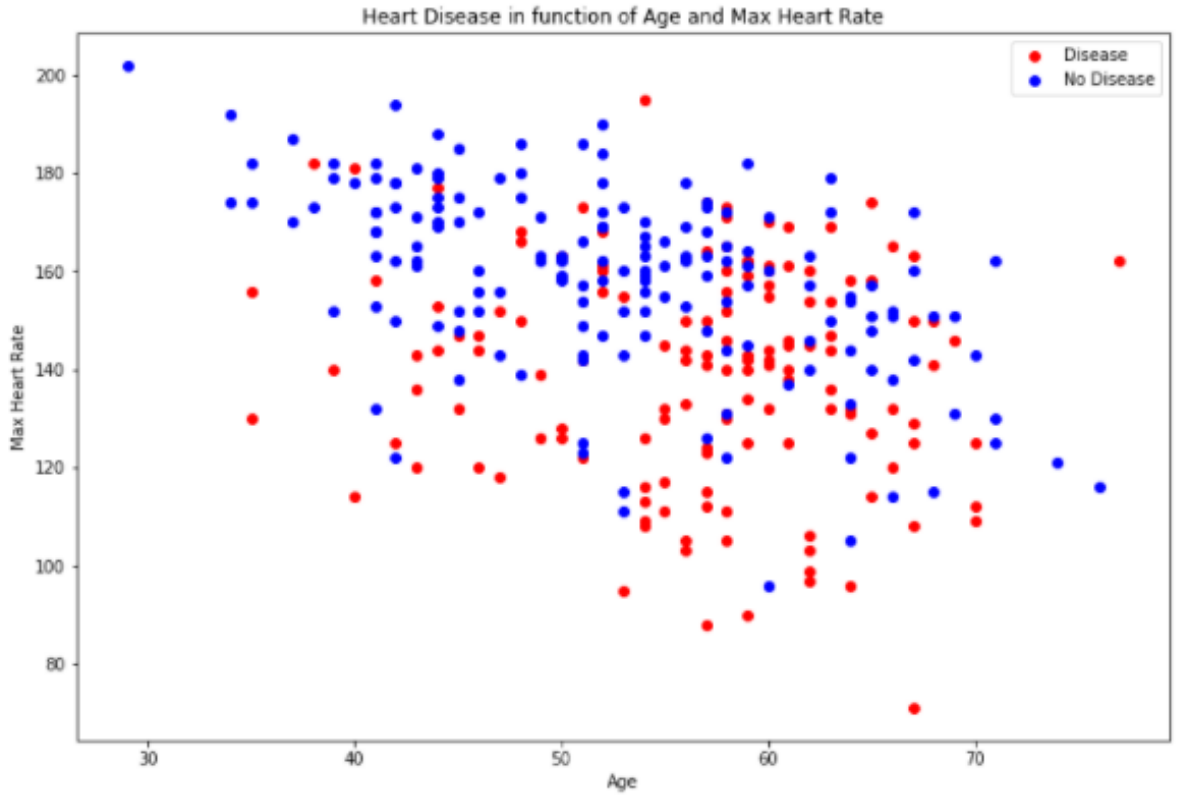
Pairplots are also a great way to immediately see the correlations between all variables

```
subData = data[['Age','RestBP','Chol','MaxHR','Oldpeak']]
sns.pairplot(subData,kind="kde")
```

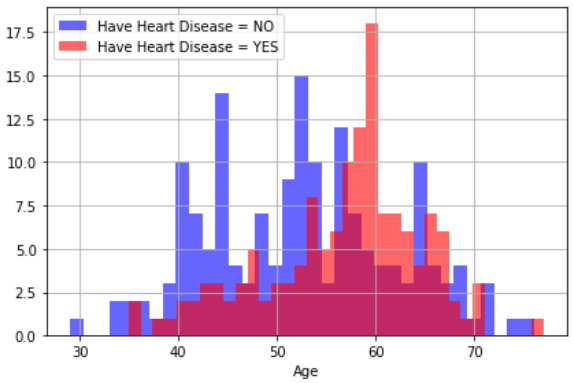


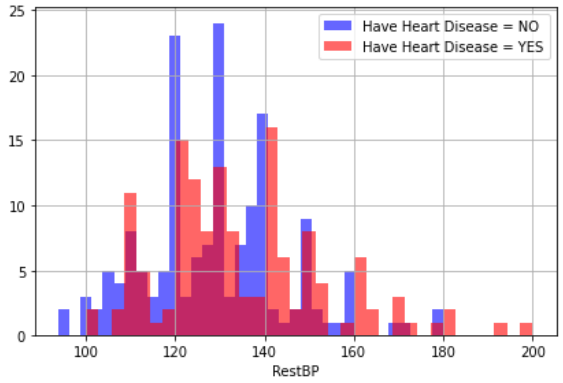
Below scatter plot shows the how Max Heart rate and Age is related to Heart Disease

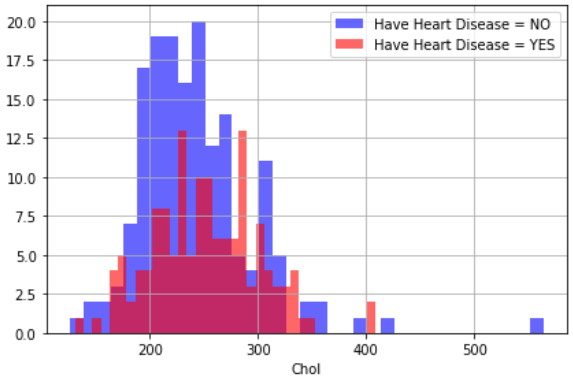
```
plt.figure(figsize=(12, 8))

plt.scatter(data.Age[data.AHD==1],data.MaxHR[data.AHD==1],c="r")
plt.scatter(data.Age[data.AHD==0],data.MaxHR[data.AHD==0],c="b")

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"]);
```

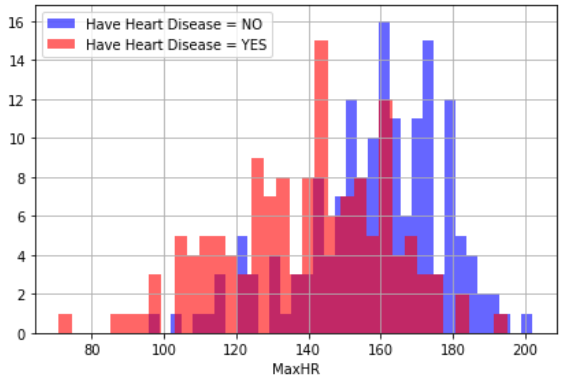


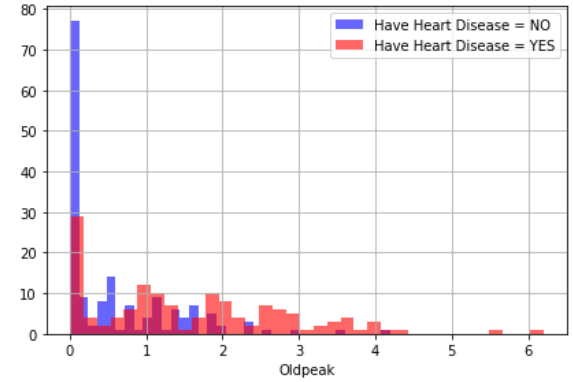
```
plt.figure(figsize=(15, 15))

for i, column in enumerate(continous_val, 1):
    plt.subplot(3, 2, i)
    data[data["AHD"] == 0][column].hist(bins=35, color='blue', label='Have Heart Disease = NO', alpha=0.6)
    data[data["AHD"] == 1][column].hist(bins=35, color='red', label='Have Heart Disease = YES', alpha=0.6)
    plt.legend()
    plt.xlabel(column)
```







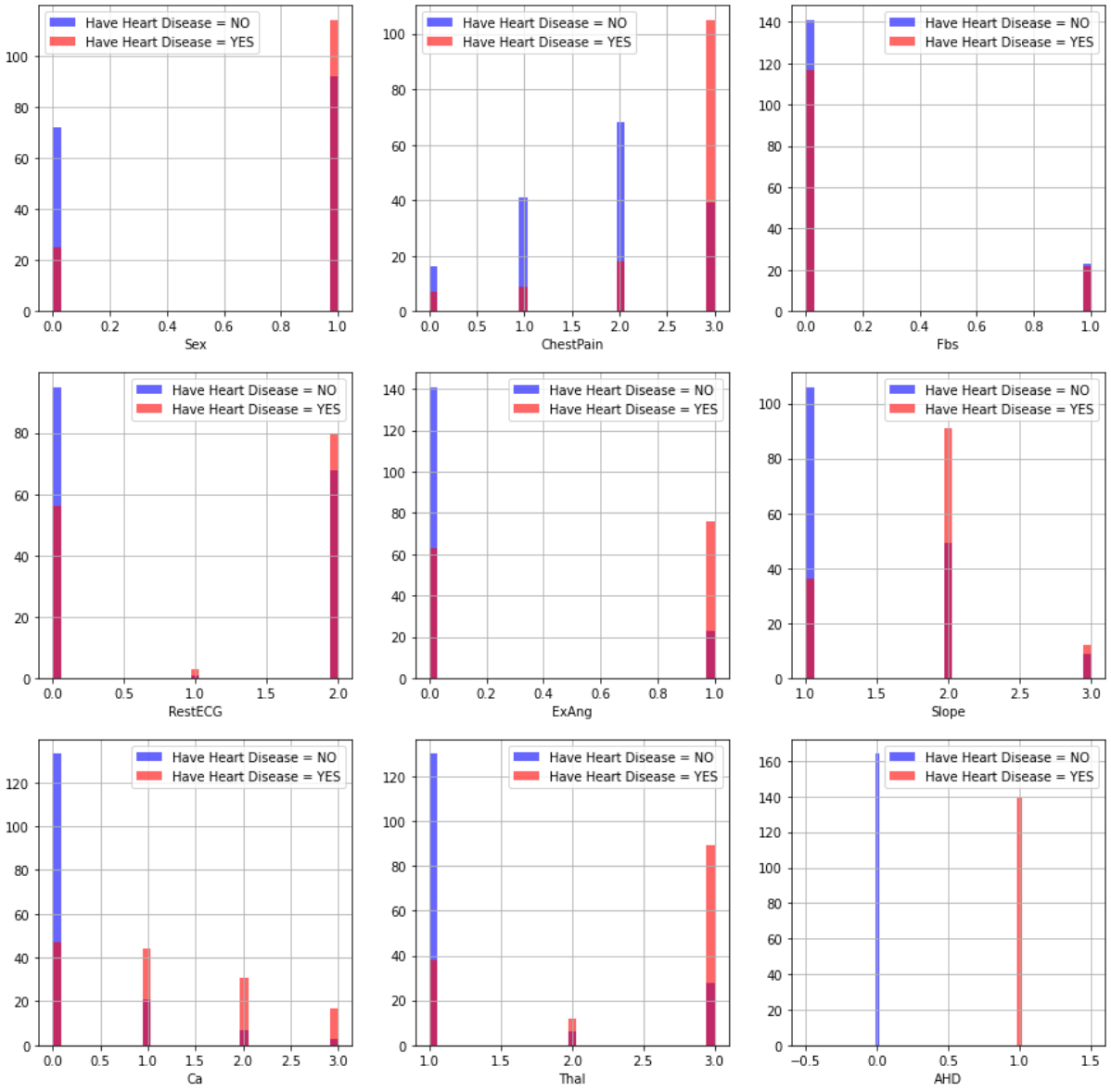




Observations from the above plot:

- trestbps: resting blood pressure anything above 130-140 is generally of concern
- chol: greater than 200 is of concern.
- thalach: People with a maximum of over 140 are more likely to have heart disease.
- The old peak of exercise-induced ST depression vs. rest looks at heart stress during exercise. An unhealthy heart will stress more.

```
plt.figure(figsize=(15, 15))

for i, column in enumerate(categorical_val, 1):
    plt.subplot(3, 3, i)
    data[data["AHD"] == 0][column].hist(bins=35, color='blue', label='Have Heart Disease = NO', alpha=0.6)
    data[data["AHD"] == 1][column].hist(bins=35, color='red', label='Have Heart Disease = YES', alpha=0.6)
    plt.legend()
    plt.xlabel(column)
```



Observations from the above plot:

- cp {Chest pain}: People with cp 1, 2, 3 are more likely to have heart disease than people with cp 0.
- restecg {resting EKG results}: People with a value of 1 (reporting an abnormal heart rhythm, which can range from mild symptoms to severe problems) are more likely to have heart disease.
- exang {exercise-induced angina}: people with a value of 0 (No ==> angina induced by exercise) have more heart disease than people with a value of 1 (Yes ==> angina induced by exercise)

- slope {the slope of the ST segment of peak exercise}: People with a slope value of 2 (Downsloping: signs of an unhealthy heart) are more likely to have heart disease than people with a slope value of 2 slope is 0 (Upsloping: best heart rate with exercise) or 1 (Flat Sloping: minimal change (typical healthy heart)).
- ca {number of major vessels (0-3) stained by fluoroscopy}: the more blood movement the better, so people with ca equal to 0 are more likely to have heart disease.
- thal {thallium stress result}: People with a thal value of 2 (defect corrected: once was a defect but ok now) are more likely to have heart disease.

4.3 Prepare Data for modeling

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

Now let's split the data into training and test sets. I will split the data into 80% training and 20% testing:

```
x = data.iloc[:, :-1].values
y = data.iloc[:, -1].values
x_train, x_test, y_train, y_test=train_test_split(x,y, test_size=0.2, random_state=2)
```

Printing dimension of train and test

```
print(x_train.shape)
print(y_train.shape)
print(x_test.shape)
print(y_test.shape)
```

```
(242, 13)
(242,)
(61, 13)
(61,)
```

4.4 Training

Now let's train the machine learning model and print the classification report of our logistic regression mode

```
reg=LogisticRegression()
reg.fit(x_train, y_train)
```

```
c:\users\sanke\appdata\local\programs\python\python39\lib\site-packages\sklearn\linear_model\_logistic.py:763: ConvergenceWarni
ng: 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(
```

```
LogisticRegression()
```

```
print(reg.predict(x_test[0].reshape(1,-1)))
reg.predict(x_test[0:10])
```

```
[0]
```

```
array([0, 1, 0, 0, 0, 1, 0, 0, 1, 0], dtype=int64)
```

```
pred=reg.predict(x_test)
print(pred)
score=reg.score(x_test, y_test)
print(score)
```

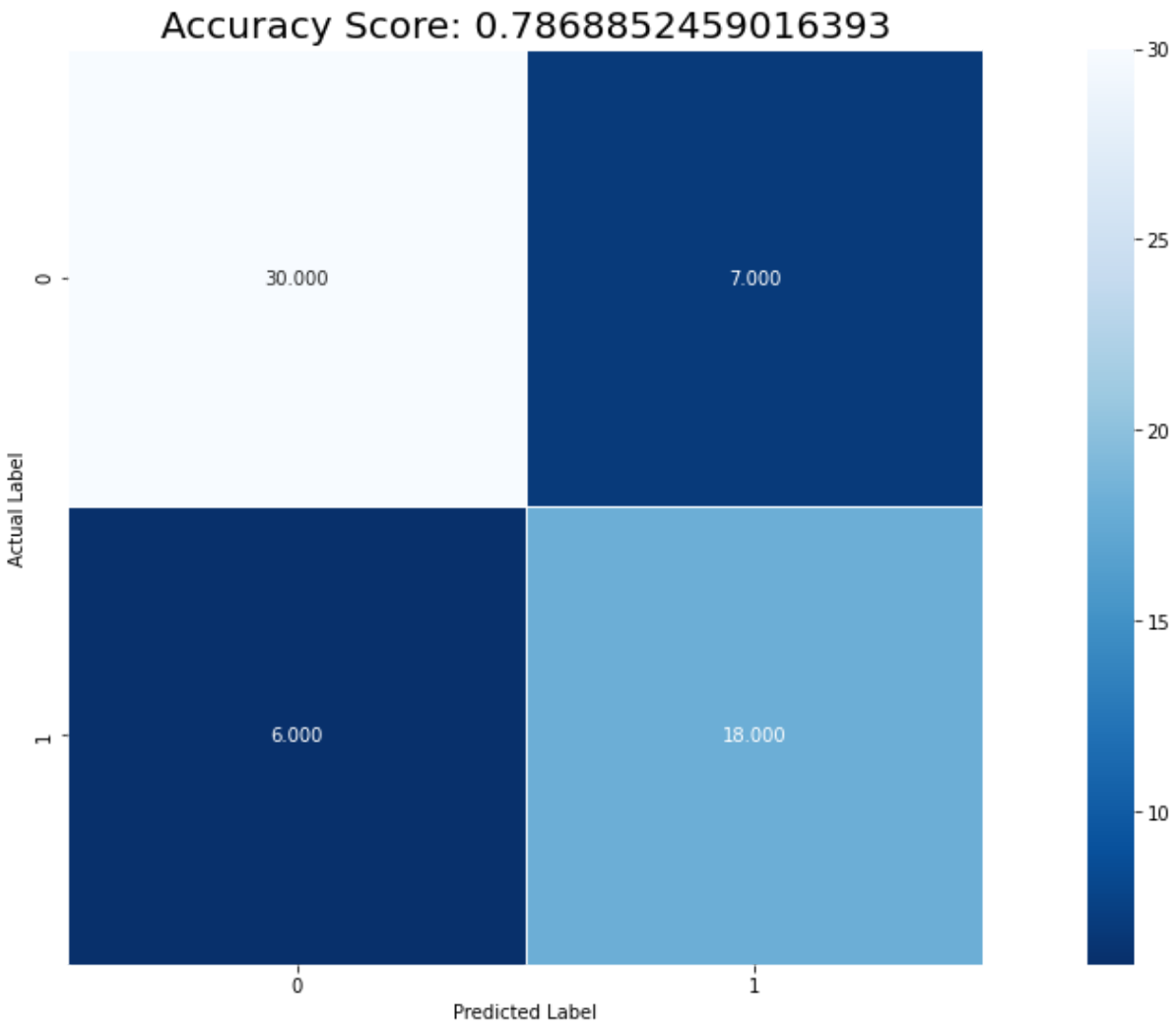
```
[0 1 0 0 0 1 0 0 1 0 0 1 1 1 0 0 1 1 0 1 0 1 0 0 0 1 0 0 0 0 0 0 0 1 0 1 1 0
 1 0 1 0 1 1 0 0 1 1 1 0 0 0 1 0 0 0 1 0 0 1 0 1 0 1]
0.7868852459016393
```

Now let's train the machine learning model and print the classification report of our logistic regression model

```
from sklearn.metrics import confusion_matrix
cm=confusion_matrix(y_test, pred)
print(cm)
plt.figure(figsize=(20,9))
sns.heatmap(cm, annot=True, fmt=".3f", linewidths=.5, square=True, cmap='Blues_r')
plt.ylabel('Actual Label')
plt.xlabel('Predicted Label')
all_sample_title=f"Accuracy Score: {score}"
plt.title(all_sample_title, size=20)
```

```
[[30  7]
 [ 6 18]]
```

```
Text(0.5, 1.0, 'Accuracy Score: 0.7868852459016393')
```



As you can see the model performs very well of the test set as it is giving almost the same accuracy in the test set as in the training set

Chapter 5

CONCLUSION

Here we successfully created the model of Machine Learning for Heart Disease Prediction. In this model we use Logistic Regression Algorithm

Chapter 6

REFERENCES

Website Referenced :-

- https://towardsdatascience.com/project-predicting-heart-disease-with-classification-machine-learning-algorithms-fd69e6fdc9d6
- https://www.geeksforgeeks.org/ml-heart-disease-prediction-using-logistic-regression/
- https://www.javatpoint.com/logistic-regression-in-machine-learning
- https://www.knowledgehut.com/blog/data-science/logistic-regression-for-machine-learning