Data Science

Final Report

Heart Attack/Disease Predictor

Presented to:

Dr. Irfan Younas

Assistant Professor

FAST-NUCES

Presented by:

SECTION B

Tasmia Rana	(14L-4178)
Syed Hussain Haider Zaidi	(14L-4187)
Harris Irfan	(14L-4188)
Abdul Rehman Sarohy	(14L-4196)

Table of Contents

Abstract & Introduction

Today, heart disease is one of the most prevalent causes of death in the world. Hence its early prediction and diagnosis is important in medical field which could help in on time treatment, and thereby decrease the deaths caused by it.

Our application aims at facilitating both the doctors, and the patients who are prone to heart disease in the near future. Technology is advancing and the world is accelerating in the field of computer science. So now it's about time that we invest on computer technology for medical diagnosis.

The patients who are having a high risk of heart disease can be warned beforehand and they can take precautionary measures. It could greatly reduce their chances of having a heart disease in the near future. On the other hand, doctors spend a handful of time trying to diagnose a disease by evaluating the symptoms of the patients. This is also where our application comes to the rescue by providing the doctors with a reliable machine to do the diagnosis for them or confirm their diagnosis. They can spend the designated time for diagnosis over something more useful, while leaving our application to do the diagnosis.

Data Set Description

We have trained and tested our algorithm on a data set which was gathered by Robert Detrano, M.D., Ph.D. from Cleveland Clinic Foundation [1]. The database, originally collected and compiled by Dr. Robert contained 76 attributes and **303** entries. However, researchers have used only the relevant 14 attributes from the original data in all their published experiments. Therefore, we also used only those relevant **14** attributes which are listed below:

1. age
2. sex
3. cp: chest pain type

    -- Value 1: typical angina

    -- Value 2: atypical angina

    -- Value 3: non-anginal pain

    -- Value 4: asymptomatic
4. trestbps: resting blood pressure (in mm Hg on admission to the hospital)
5. chol: serum cholesterol in mg/dl
6. fbs: (fasting blood sugar > 120 mg/dl) (1 = true; 0 = false)
7. restecg: resting electrocardiographic results

    -- Value 0: normal

    -- Value 1: having ST-T wave abnormality (T wave inversions and/or ST elevation or depression of > 0.05 mV)

    -- Value 2: showing probable or definite left ventricular hypertrophy by Estes' criteria
8. thalach: maximum heart rate achieved
9. exang: exercise induced angina (1 = yes; 0 = no)
10. oldpeak = ST depression induced by exercise relative to rest
11. slope: the slope of the peak exercise ST segment

    -- Value 1: upsloping

    -- Value 2: flat

    -- Value 3: downsloping
12. ca: number of major vessels (0-3) colored by fluoroscopy

13. thal: 3 = normal; 6 = fixed defect; 7 = reversible defect
14. num-the predicted attribute: diagnosis of heart disease (angiographic disease status)

    -- Value 0: < 50% diameter narrowing

    -- Value 1: > 50% diameter narrowing

    (in any major vessel: attributes 59 through 68 are vessels)

    It is an integer valued from 0 (no presence of heart disease) to 4. Experiments with the Cleveland database have concentrated on simply attempting to distinguish presence (values 1,2,3,4) from absence (value 0).

A modification in the database has been made to account for values of unknown attributes. In the original database, the unknown attributes are represented by '?'. In the **processed data** that we downloaded from the UCI Machine Learning Repository [1], the unknown attributes are replaced by a value of -9.0 to allow them to be incorporated in the experimentations. This replacement is already done in the available dataset and **was not introduced by us**. The results gathered after inputting that data into our algorithm are shown later. First let us visualize the data.

Data Visualization

For data visualization, Tableau was used. To visualize the data, we had to further process the processed data downloaded from UCI Machine Learning Repository [1]. The data downloaded was completely numerical. However, for visualization purposes, we had to change some of the attributes from measures to dimensions, i.e. from numeric to non-numeric. For example, the attribute *sex* was initially labeled as 0 or 1. It had to be changed to 'male' and 'female' for visualization purposes. All the other discreet attributes were assigned their relative labels, for better visualization with easily identifiable and comprehendible labels so that a layman can also understand the visuals.

Below, the relationship between all the 13 attributes with the 14th attribute, i.e. the severity of the presence of heart disease is shown.

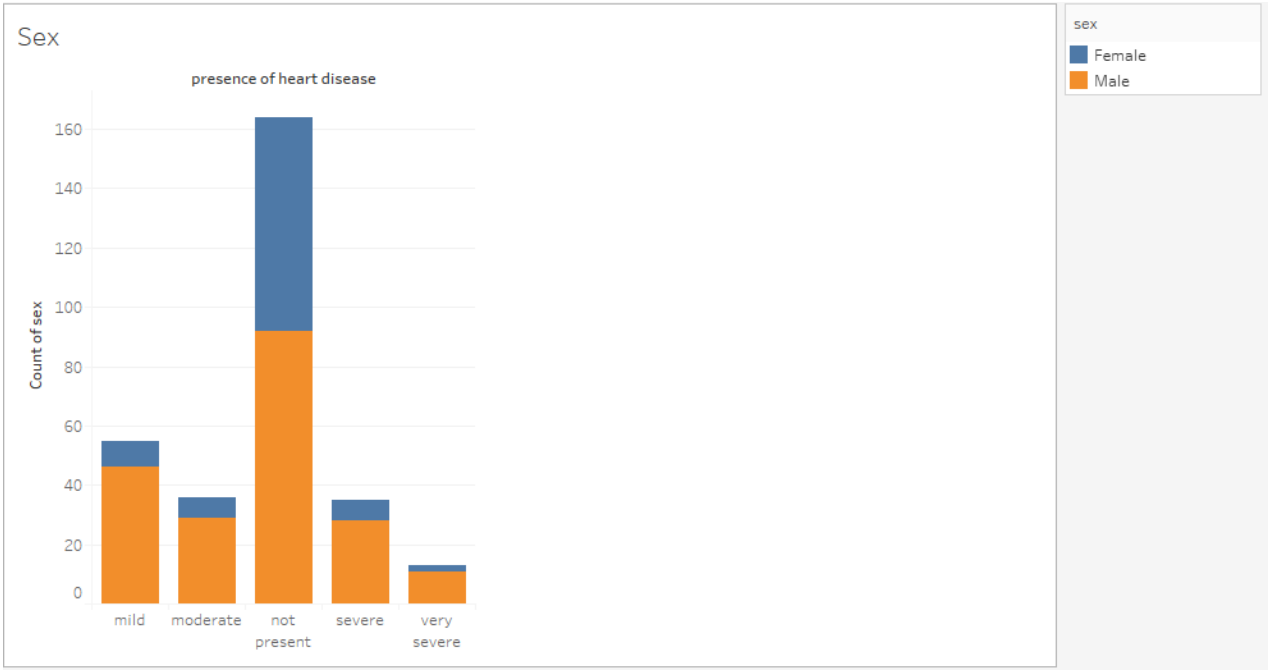

Figure 1: Ratio of presence of heart disease among different sexes

It can be seen clearly that males are more prone to heart disease than females.

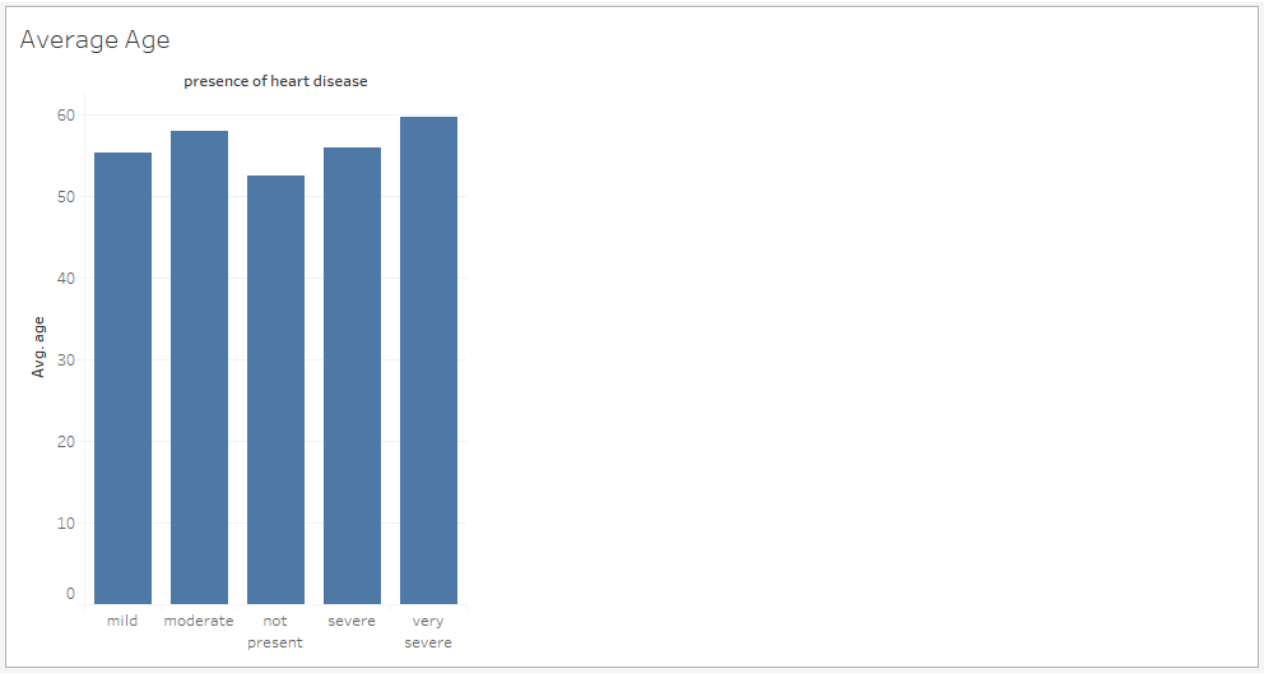

Figure 2: Average age against presence of heart disease

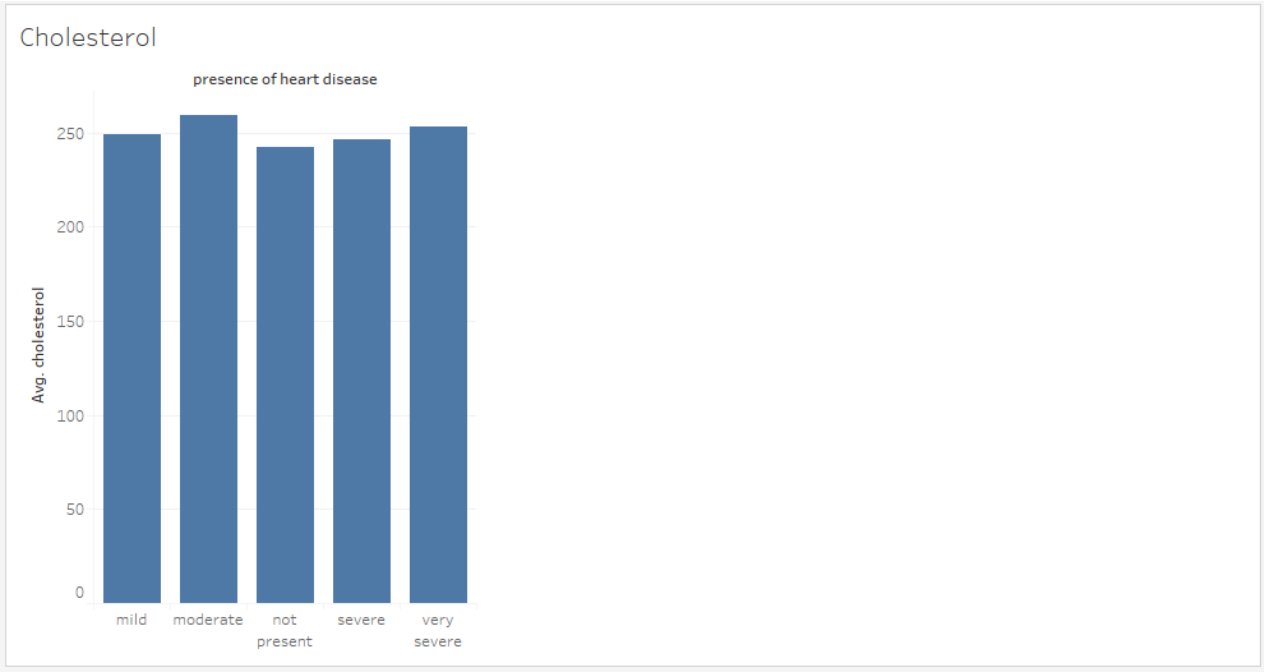

Figure 3: Average cholesterol levels against presence of heart disease

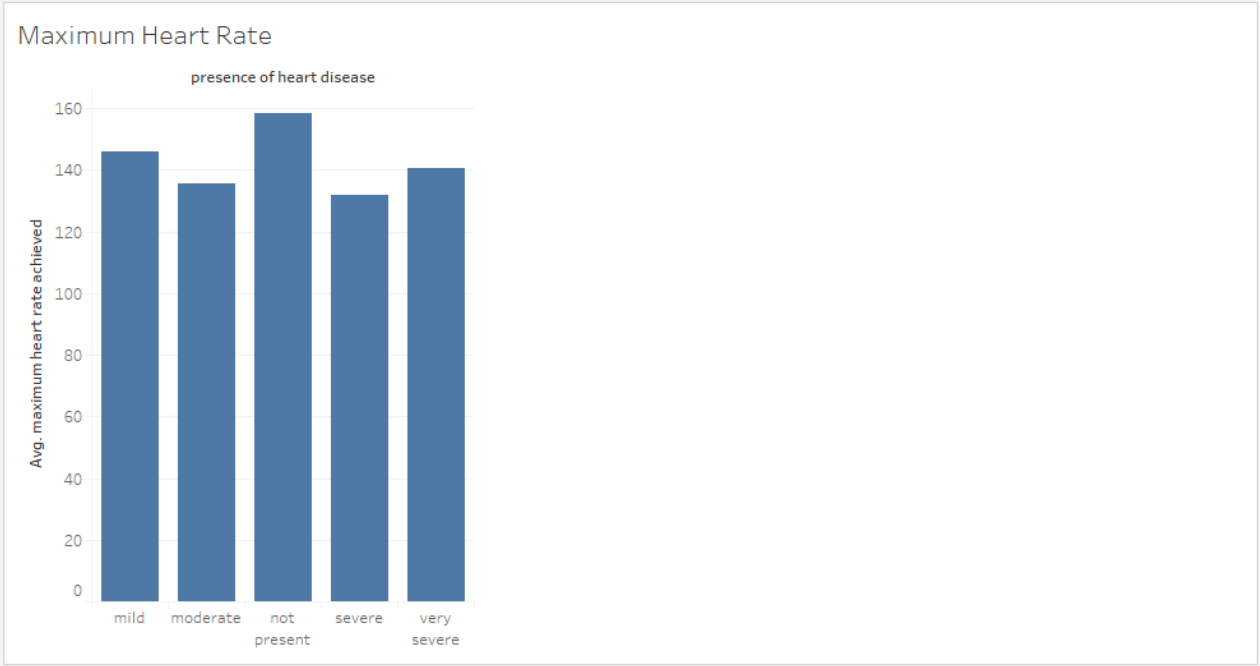

Figure 4: Maximum heart rate achieved against presence of heart disease

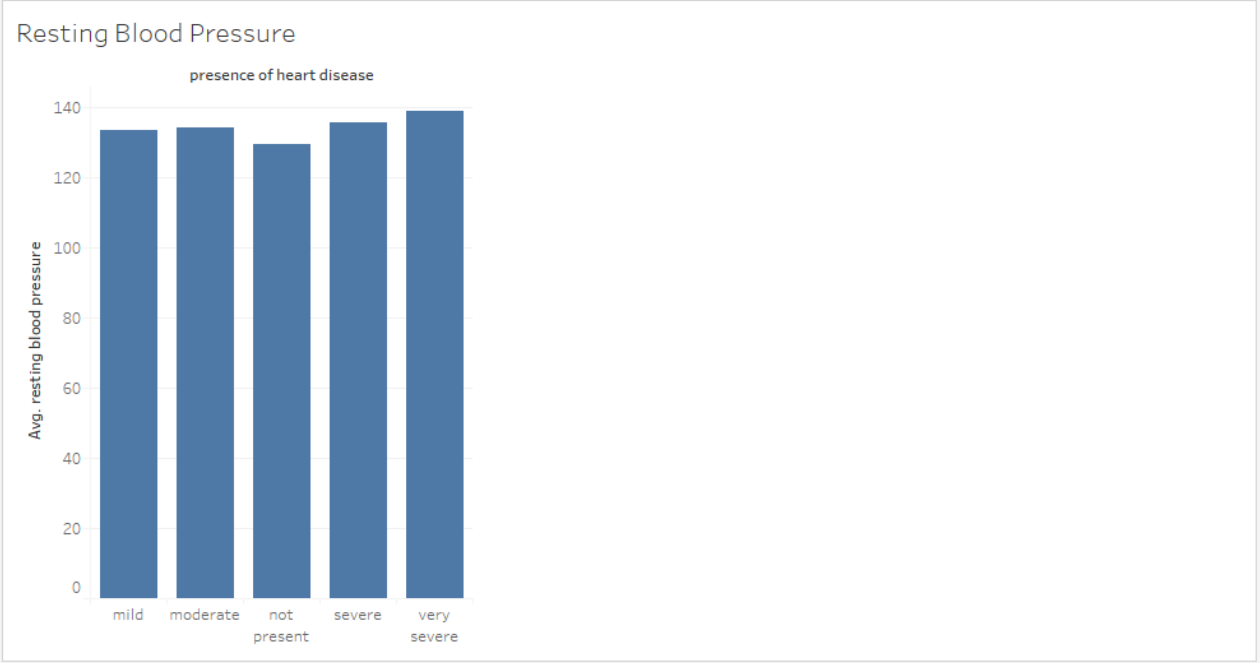

Figure 5: Resting blood pressure against presence of heart disease

A trend can be seen that as the severity of heart disease increase, resting blood pressure also increases among patients.

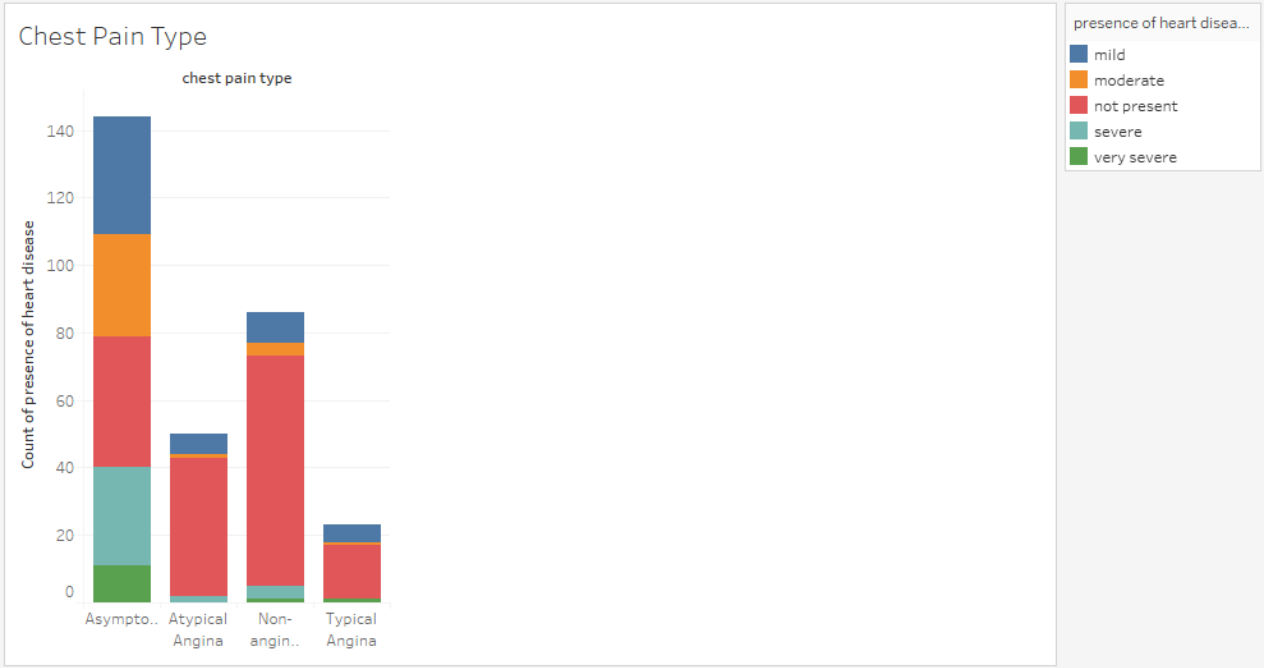

Figure 6: Chest pain types against presence of heart disease

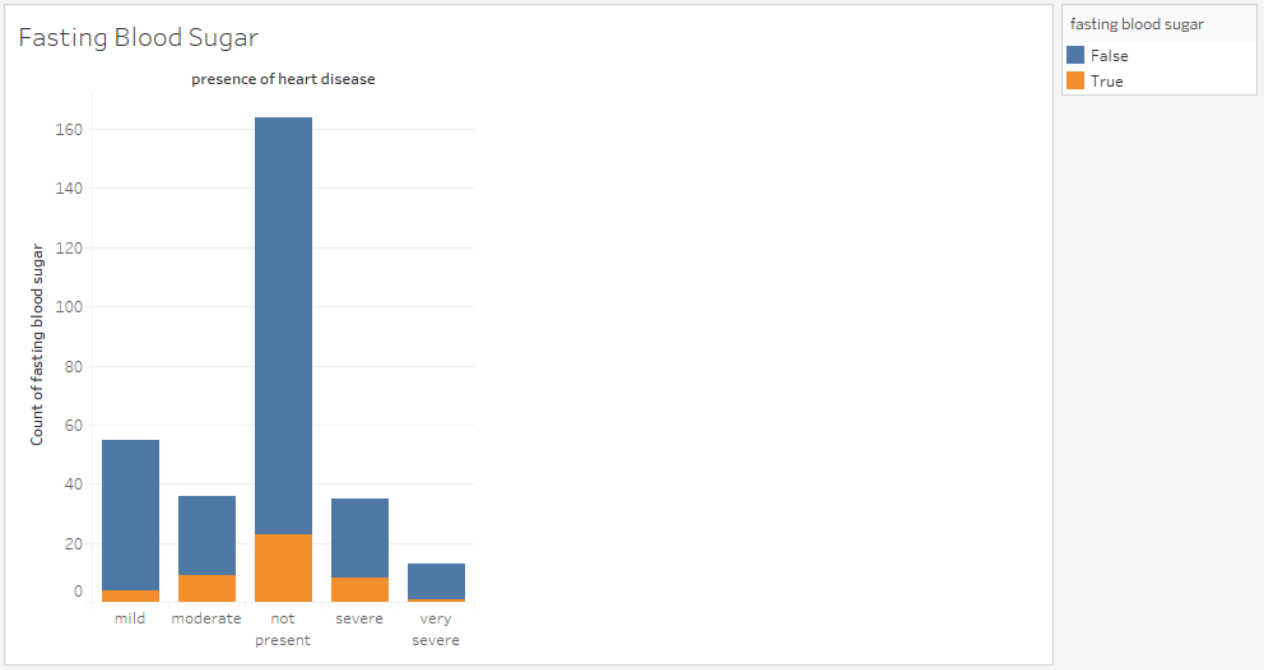

Figure 7: Fasting blood sugar against presence of heart disease

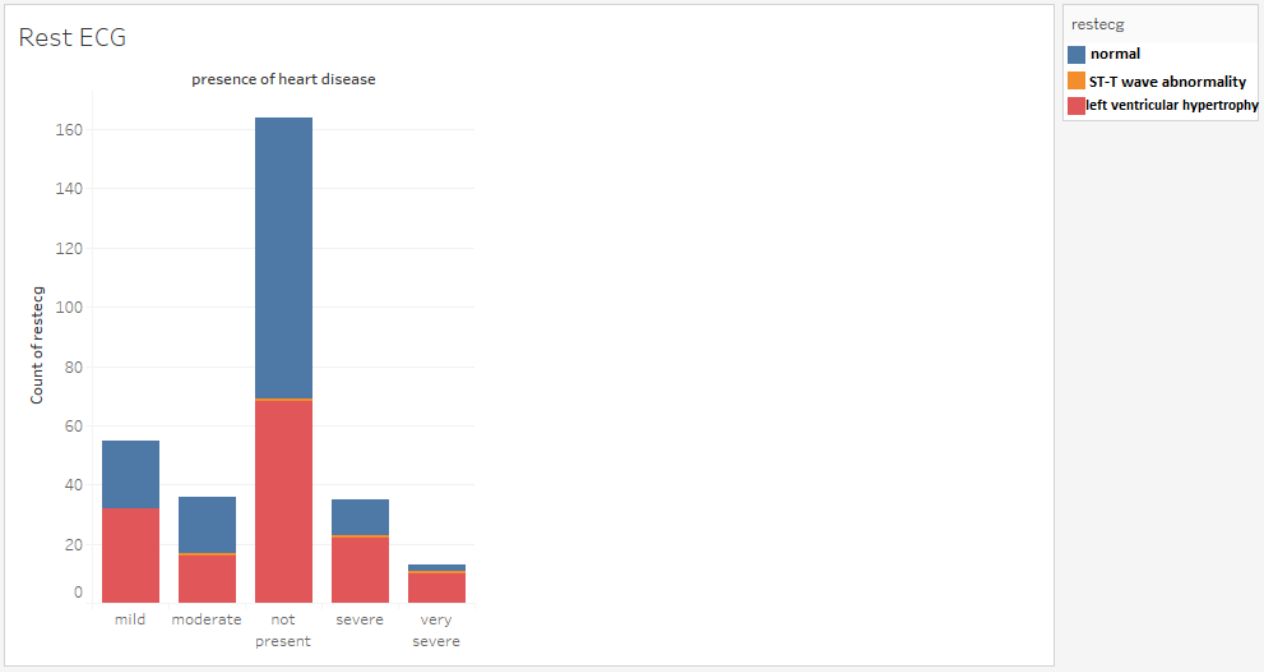

Figure8: Rest ECG against presence of heart disease

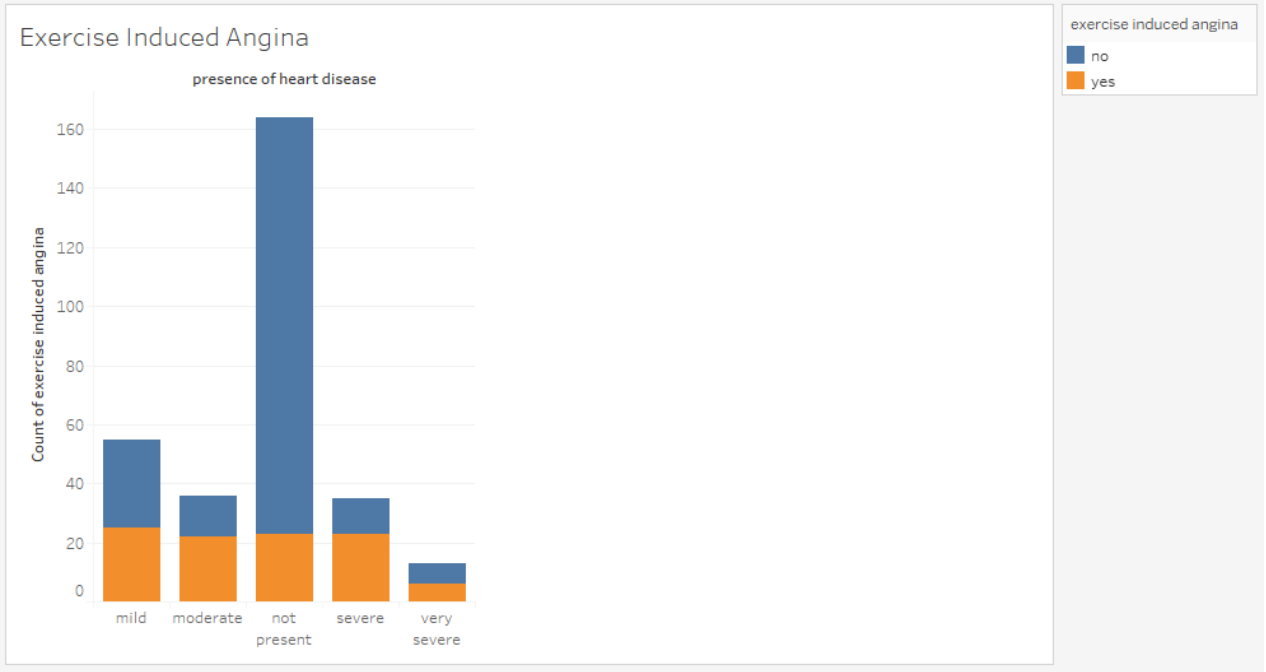

Figure 9: Exercise induced angina against presence of heart disease

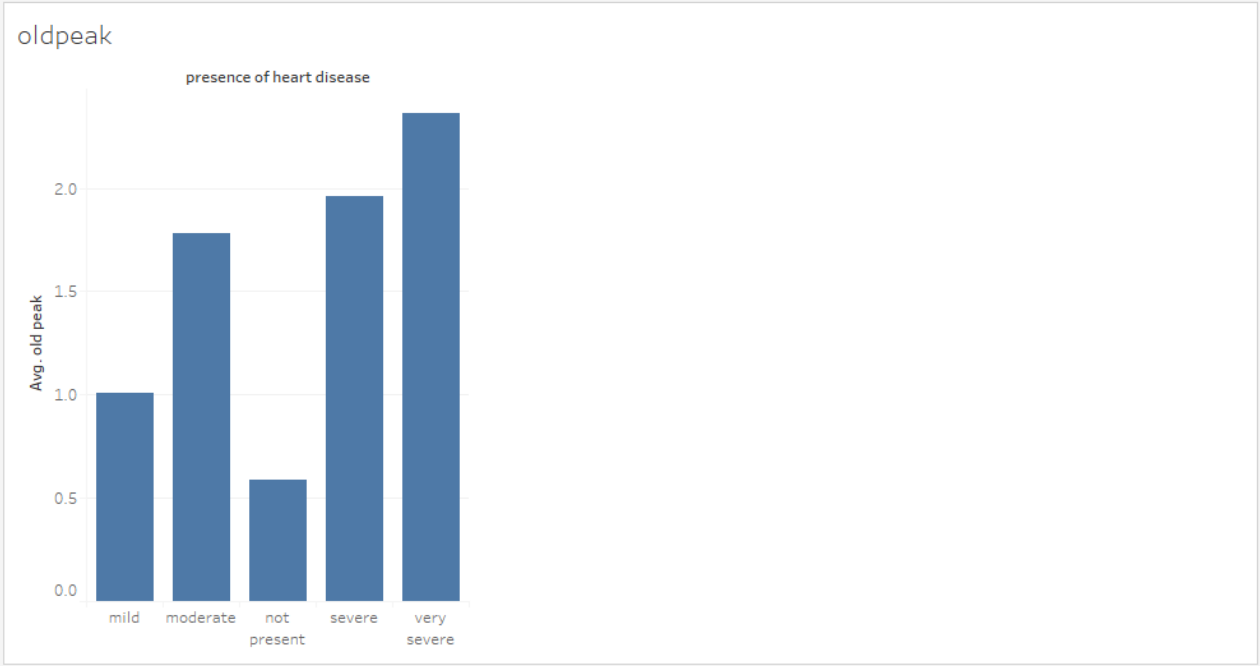

Figure 10: Old peak against presence of heart disease

Again, a clear trend can be seen. There is a direct relation with increasing severity of heart disease. This means that it is an important feature

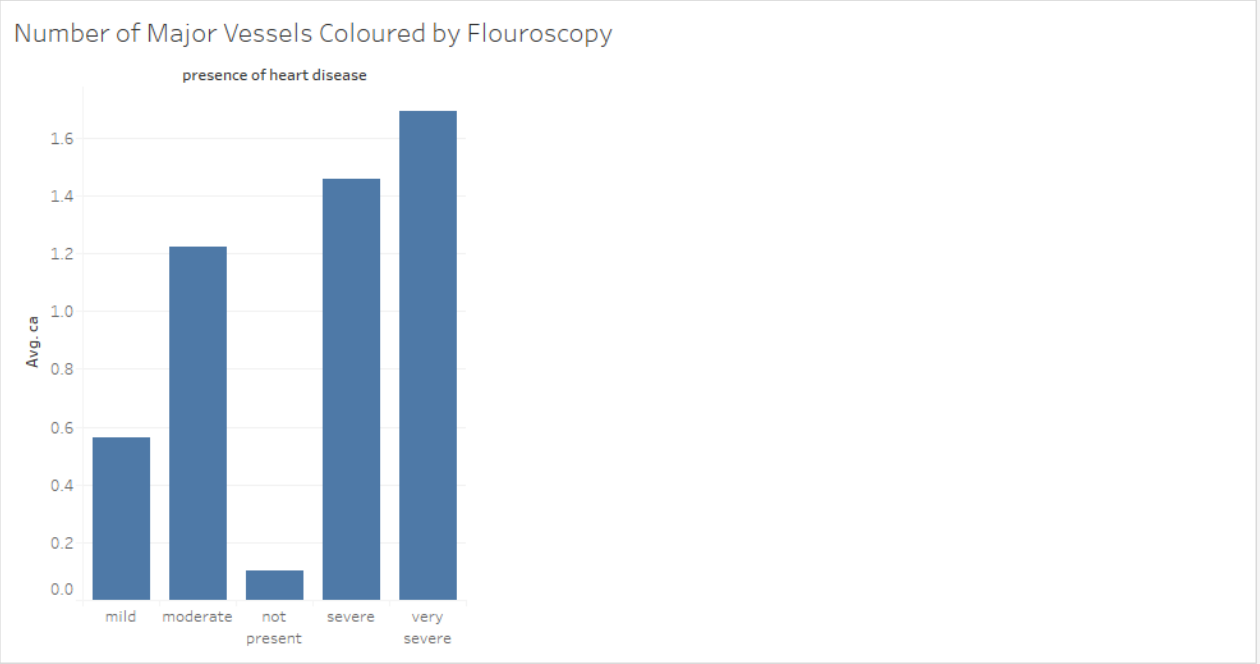

Figure 11: Number of major vessels coloured by fluoroscopy against presence of heart disease

It also serves as an important feature as there is a direct relation between it and the output.

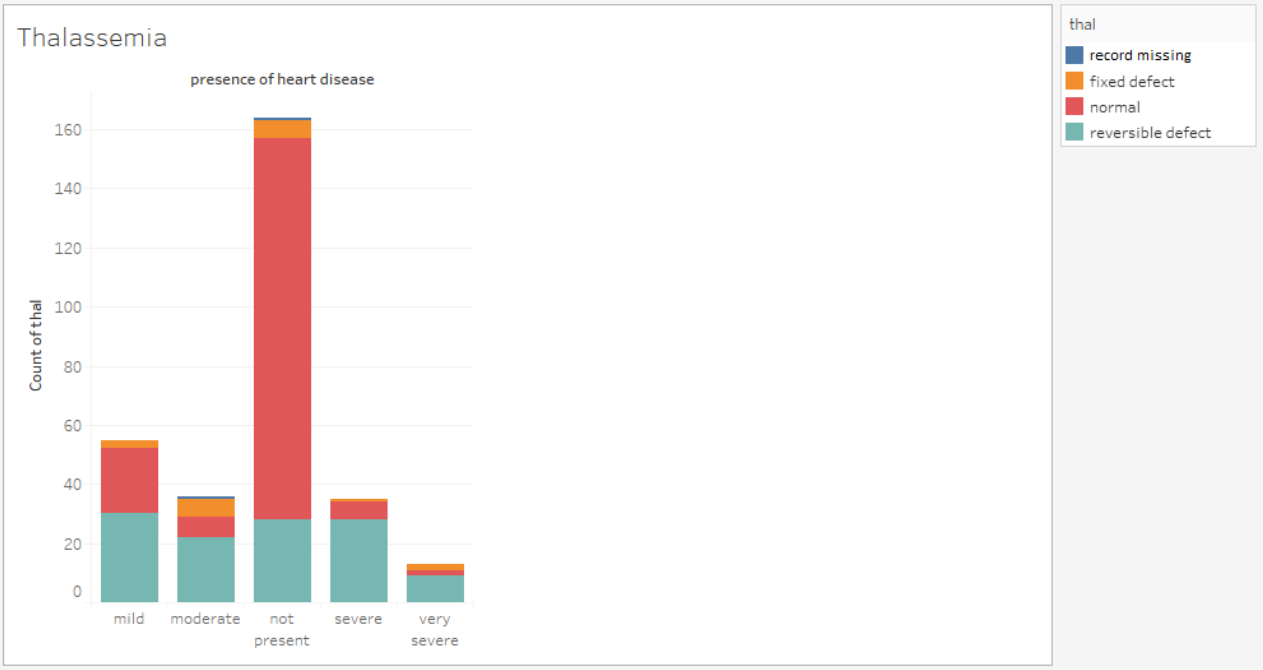

Figure 12: Thalassemia against presence of heart disease

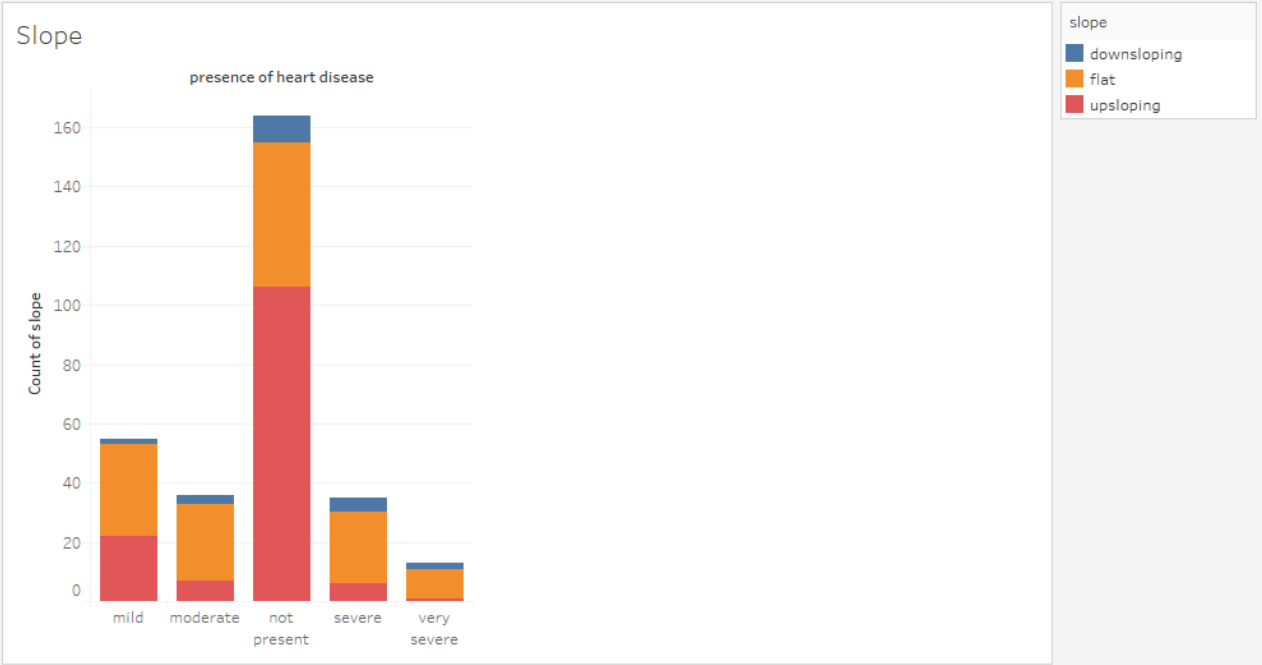

Figure 13: Slope against presence of heart disease

Experimentation & Method

We have implemented **logistic regression** to solve this problem of predicting heart disease. The tools that we have used are Python 3.6.4 along with JetBrains PyCharm IDE. We did some pre-processing with the data to enable it for our logistic regression model. All the features were normalized before training by subtracting mean from them and dividing them by the standard deviation because the features lied in different ranges.

Our model only predicts presence of heart disease, not the severity of it. We have reduced the multi-classification problem, which had to be solved by the 'one-vs-all' approach, to a simpler problem, which has only two classes, i.e. presence and absence of heart disease. Therefore, all the output values greater than equal to 1 (1, 2, 3, 4) were converted to 1. They originally represented severity of heart disease. Now they all represent presence of heart disease only. The 0 values were left as it is, and they represent absence of heart disease.

We divided the data set of 303 entries into **testing** and **training data**. 282 examples were used for training and 21 were used for testing. We also experimented with varied values of learning rate and number of iterations. The best values that resulted in best results were then finalized for our model.

We first take data in CSV format and convert it into a Numpy array. After obtaining the features in Matrix X, we append a column of 1's as the x0 feature vector. The first 13 attributes of the data set, as mentioned above, are used as the 13 features, from x1-x13. The last attribute, i.e. the presence or absence of heart disease is used as a target (output) value. We have implemented three functions, a hypothesis function, cost function, and gradient descent function.

The hypothesis function is a sigmoid of the weighted sum of thetas and the feature vector:

$$h_\theta(x) = g(\theta^T x), \qquad g(z) = \frac{1}{1+e^{-z}}.$$

The cost function on the other hand, which represents the average cost of all the training examples, is as follows:

$$J(\theta) = \frac{1}{m}\sum_{i=1}^{m}\left[-y^{(i)}\log(h_\theta(x^{(i)})) - (1-y^{(i)})\log(1-h_\theta(x^{(i)}))\right],$$

And, the derivative of the cost function, used to update thetas in the gradient descent algorithm:

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

For gradient descent, in a loop we add to every value of theta, the negative of the cost function's partial derivative with respect to that feature weight. The feature vector theta is initially initialized to 0. We have also calculated the cost on respective thetas values and saved it to observe the trend of decreasing cost. We keep on running the iterations until the cost function converges to a minimum or the change in its value becomes negligible. In every iteration, the thetas are updated simultaneously. The implementation is generic and vectorized.

Then we use the thetas predicted by our classifier for testing purpose. The test data is loaded from a separate csv file and feature normalization is also done on the test data by the same method as used for the training data. Feature vector x0 = 1 is also appended. The test data is passed to the hypothesis function, that produces a single value between 0 and 1 as the output for every test example. The threshold that we have set is 0.5. If the value of the hypothesis is greater than 0.5, we classify it as presence of heart disease and otherwise, the example is classified as absence of heart disease.

Numpy has proven to be extremely useful to carry out all the above functionalities. Here is a list of some of the Numpy functions that we used:

- concatenate
- append
- array
- std
- mean
- dot
- ones
- empty
- size
- shape

We also imported a library *csv* and used its function *csv.reader* to initially load/read the data into the program.

To carry out the logarithmic and exponential operations, *math* library was used.

In short, we used the following libraries:

- numpy
- matplotlib
- csv
- math

We experimented with various learning rates and number of iterations. Below, the results of different experiments' effect on the cost function is shown, with varying levels of learning rate(alpha) and number of iterations. The construction of these plots is done with the *matplotlib* library.

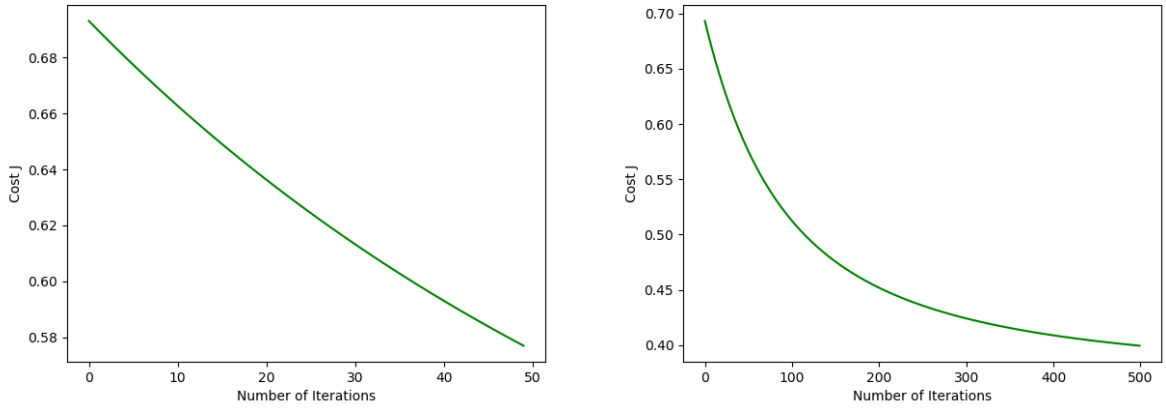

Figure 14: Learning rate = 0.01

Since the learning rate (step size) is very low, the algorithm converges very slowly.

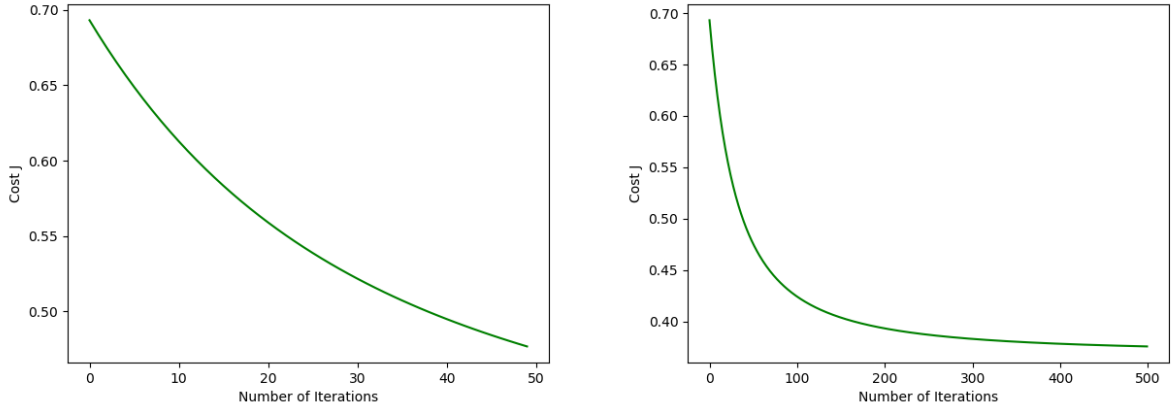

Figure 15: Learning rate = 0.03

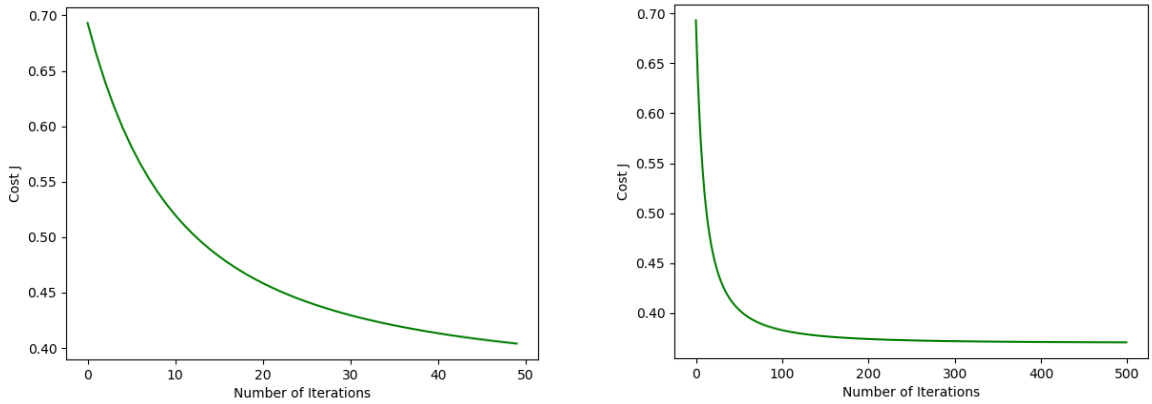

Figure 16: Learning rate = 0.10

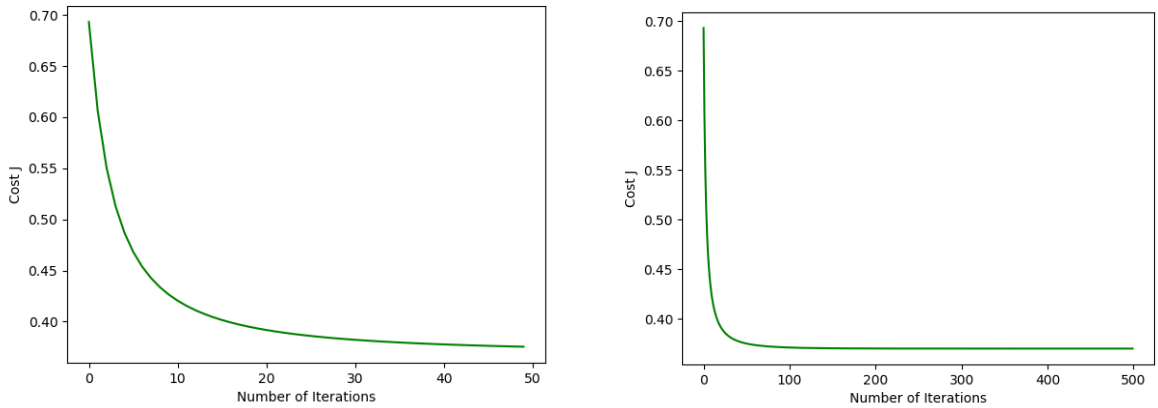

Figure 17: Learning rate = 0.30

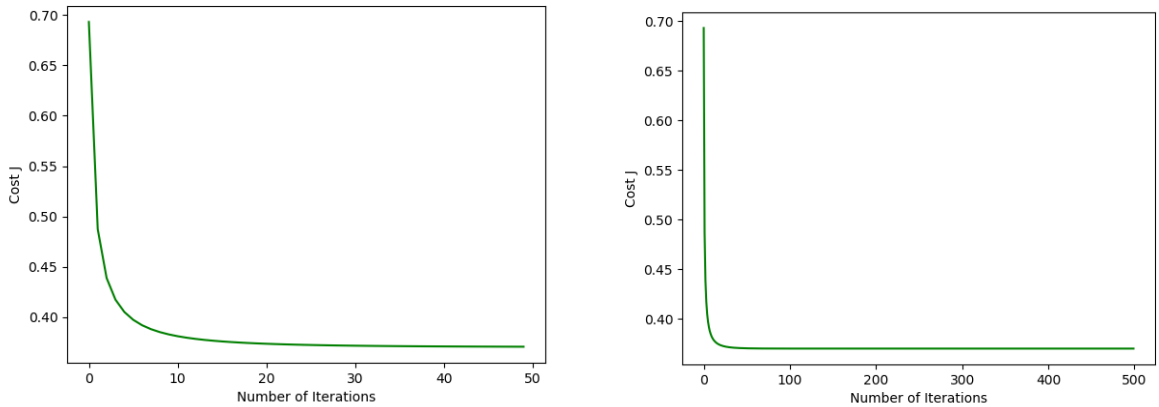

Figure 18: Learning rate = 0.90

As we increase the learning rate, it can be clearly seen that the algorithm converges much more quickly i.e. in much less number of iterations. Therefore, there is no need to run the algorithm for 500 iterations. 50 iterations would be more than enough.

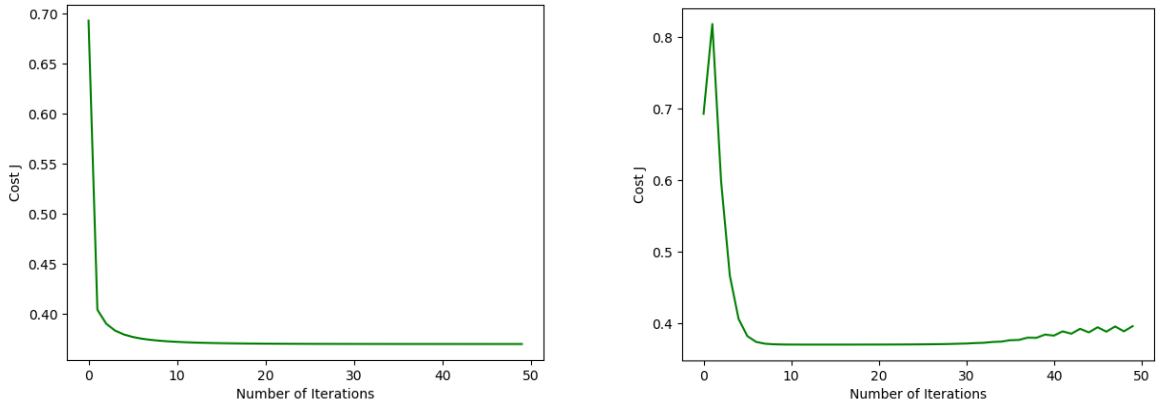

Figure 19: Learning rate = 2.0 (left) and 10.0 (right)

The algorithm performs okay with alpha = 2.0 and even alpha = 4.0. However, when the value of alpha exceeds to 10, the gradient descent algorithm diverges. The reason is that the step size becomes so great, that the algorithm misses the minima of the convex cost function and jumps to the other side of the minima.

Keeping these things in mind, we reached a compromise and chose the value of alpha as 2.0 and 50 number of iterations.

For more details regarding the implementation, our open-source code can be viewed.

Results & Analysis

Out of the 21 examples of test data, which was completely distinct from training data, 18 were classified correctly by our algorithm, and only 3 were misclassified. It showed an accuracy of 85.7%.

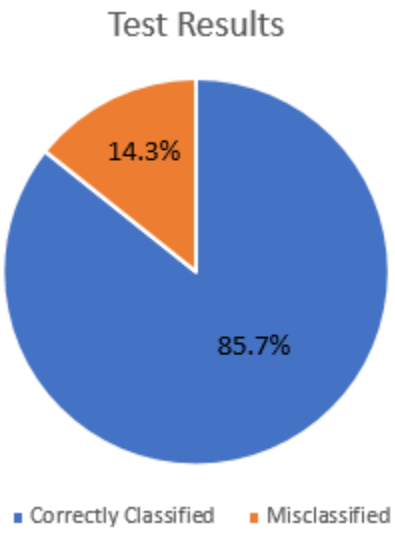

Figure 20: Test Results

However, all three misclassified examples are false negatives, i.e. the patient actually has heart disease, but he/she is cleared by the system. This can be troublesome if the application is deployed in the real world. To deal with this problem, the threshold was brought down from 0.5 to 0.25, i.e. to classify an example with a hypothesis value of greater than equal to 0.25 as 1(presence of heart disease). Still there was no improvement.

In total, there were 11 true positives, 7 true negatives, 0 false positives and 3 false negatives.

The precision, recall and F-measure of the system are as follows:

Precision = TP/(TP + FP) 11/(11+0) = 1

Recall = TP/(TP+FN) = 11/(11+3) = 11/14 = 0.79

F Measure = 2PR/(P+R) = 2(1)(0.79)/(1+0.79) = 0.88

Conclusion

The results are convincing and it can be seen that they are quite accurate. However, this application cannot serve as a completely independent diagnosis machine. Rather, doctors or physicians can use this application to confirm or reject their diagnosis and can use it as a helping tool. One important thing to note is that this program can be trained very well with a small set of data. The database of a single hospital can be used to train this program and it will produce satisfying results, as it proved in this case. This helping tool can greatly save time, resources and ultimately lives. Life is precious, let's value it.

In the future, we plan to train our model with more training data from different hospitals from different locations all over the globe. This way, we plan to improve the accuracy and robustness of our model and minimize false negatives. We also plan to develop a model that can predict the severity of heart disease as well, along with its presence. In the future, we also plan to develop more medical applications that are based on machine learning.

References

[1] Archive.ics.uci.edu. (2018). *UCI Machine Learning Repository: Heart Disease Data Set*. [online] Available at: http://archive.ics.uci.edu/ml/datasets/heart+disease [Accessed 25 May 2018].