



**DUBLIN INSTITUTE  
of TECHNOLOGY**

*Institiúid Teicneolaíochta Bhaile Átha Cliath*

# **Medi alert System**

## **Final Year Project Report**

**DT228**

**BSc in Computer Science**

**David Long**

**Jane Ferris**

School of Computing

Dublin Institute of Technology



# Abstract

The project's aim was to create a system that could predict the presence of heart disease, this was done using Machine Learning (ML) and a dataset that contained instances both having heart disease present and not. The exploration of the features from the dataset obtained from the UCI ML repository, led to a selection of them being used to construct a model. The final features used in the construction of the model consisted of gender (sex), resting blood pressure (restbp), the maximum heart rate achieved during a thallium stress test (thalach), the number of major vessels coloured by fluoroscopy (ca), the type of chest pain recorded (cp), the peak exercise ST Segment of an ECG (oldpeak) and the thallium test result (thal).

There are a number of different ML algorithms that are tested for this project, they are as follows, Logistic Regression (LR), Random Forest (RF), K-Nearest Neighbours (KNN), Support Vector Machine (SVM) and Linear Discriminant Analysis (LDA). All were tested to see which was the most accurate and consistent. All had very similar outputs with only a difference in accuracy of between 1% and 2% on average when compared, this is covered in section 4 of this document.

The main focus is shifted to the optimisation of an LR model, this is due to it producing the most consistent outputs during the testing phase of the project. The accuracy achieved for the prediction model was 82%. This was the average for the algorithms that were tested, but the LR as stated, was the most consistent. This is covered in detail in two sections of this document, refer to sections 4 and 5.

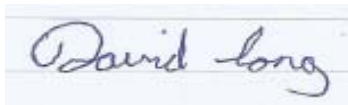
The project starts out with a data quality assessment, as there are both unprocessed and pre-processed datasets on the repository, both were assessed. There is a section that covers this with the use of graphs and there is a breakdown of the output from the training stage also included. The findings were that both datasets retained the same features bar two additional features in the unprocessed dataset. They were the family history feature and the resting heart rate feature.

Using ML it is possible to predict the presence of heart disease given that there is specific data for the training a model. This document covers everything from the loading and processing of the data to the understanding of the model outputs.

# Declaration

I hereby declare that the work described in this dissertation is, except where otherwise stated, entirely my own work and has not been submitted as an exercise for a degree at this or any other university.

Signed:

A handwritten signature in blue ink on a light blue background. The signature reads "David Long" in a cursive script.

---

David Long

13<sup>th</sup> April 2018

# Acknowledgements

I would like to thank my wife Rachael and children for the endless support and acceptance of my absence as for the last while I may have been home but only in body as my mind has stayed with my college work. I would like to thank Jane Ferris my final year project mentor also for the endless support, even in times of complete despair Jane was able to convince me that all would fine in the end and I was in fact making better progress than I thought. I would like to thank all the lecturers within DIT as everyone I have met has had a positive impact on me. Frank Duignan, Damian Bourke, Damian Gordon, Susan McKeever and Bryan Duggan to name a few, thank you for making the experience in DIT a memorable one.

I would like to take a moment in remembrance of Darragh Beatty, who left this life before his time. Darragh was a huge influence on me over the years and was even my inspiration to go into computers. From an early age Darragh had me hooked with computers, but it was only later in life that he would get to tell me “I told ya so”, when I told him I was going to college to study computers. Darragh didn’t gloat he was too much of a gentleman for that, instead he told me “don’t mess it up”.

## Table of Contents

1	Introduction.....	1
1.1	Project Overview.....	1
1.2	Project Objectives .....	2
1.3	Project Challenges.....	3
1.4	Structure of the document .....	4
2	Research.....	5
2.1	Researching the Problem.....	5
2.2	Researching the Solution.....	7
2.2.1	Machine Learning in Healthcare.....	7
2.2.2	Terminology and Concepts .....	9
2.3	Machine learning Algorithms.....	11
2.3.1	Logistic Regression.....	11
2.3.2	Random Forrest.....	12
2.3.3	Support Vector Machines .....	12
2.3.4	Linear Discriminant Analysis .....	13
2.3.5	Gaussian Naive Bayes.....	13
2.3.6	K Nearest Neighbours.....	14
2.4	Overview of the technologies.....	15
2.4.1	Weka .....	15
2.4.2	MALLET .....	16
2.4.3	NumPy .....	16
2.4.4	SciPy .....	16
2.4.5	Pandas .....	17
2.4.6	Scikit-Learn.....	17
2.4.7	Matplotlib.....	17
3	Dataset.....	19
3.1	Dataset Selection .....	19
3.2	Data Quality Analysis .....	21
3.3	Dataset Explained.....	24
3.4	Dataset Plotted.....	33
4	Testing with WEKA .....	37
4.1	Data Plots from Weka .....	37
4.1.1	Pre-processed Dataset Plots .....	37
4.1.2	Unprocessed Dataset Plots .....	39
4.2	Logistic Regression Outputs .....	42
4.2.1	Unprocessed Dataset Outputs Using a 10 fold Cross Validation .....	42
4.2.2	Pre-processed Dataset Outputs Using a 10 fold Cross Validation.....	43
4.3	Random Forrest results.....	44
4.3.1	Unprocessed Dataset Outputs Using a 10 fold Cross Validation .....	44
4.3.2	Pre-processed Dataset Outputs Using a 10 fold Cross Validation.....	45
4.4	Support Vector Machine .....	46
4.4.1	Unprocessed Dataset Outputs Using a 10 fold Cross Validation .....	46
4.4.2	Pre-processed Dataset Outputs Using a 10 fold Cross Validation.....	47
4.5	K Nearest Neighbours .....	48
4.5.1	Unprocessed Dataset Outputs Using a 10 fold Cross Validation .....	48
4.5.2	Pre-processed Dataset Outputs Using a 10 fold Cross Validation.....	49
4.6	Linear Discriminant Analysis.....	50
4.6.1	Unprocessed Dataset Outputs Using a 10 fold Cross Validation .....	50

4.6.2	Pre-processed Dataset Outputs Using a 10 fold Cross Validation.....	51
4.7	Naive Bayes.....	52
4.7.1	Unprocessed Dataset Outputs Using a 10 fold Cross Validation .....	52
4.7.2	Pre-processed Dataset Outputs Using a 10 fold Cross Validation.....	53
4.8	WEKA Test Findings.....	54
5	Scikit-Learn Outputs.....	55
5.1	Logistic Regression Outputs .....	55
5.1.1	Lasso Regression .....	55
5.1.2	Training Dataset Outputs .....	58
5.1.3	Testing Dataset Output .....	61
5.1.4	Complete Dataset Output.....	63
5.2	Gaussian Naïve Bayes Output.....	65
5.3	Random Forrest Output.....	65
5.4	Support Vector Machine Output .....	66
5.5	Findings for Scikit-Learn .....	66
6	Development.....	68
6.1	Scripts Explained.....	68
6.1.1	Load_hddf.py .....	68
6.1.2	Train_test_Split.py.....	68
6.1.3	DQR .....	68
6.1.4	TrainDset, TestDset and FullDset.....	69
6.1.5	FullVer .....	69
6.1.6	FYP.html.....	71
7	Project Plan .....	72
8	Conclusion .....	73
9	Bibliography .....	75
Appendix A.....		80
Full listing of attributes:.....		80
Appendix B.....		83
Original Data Plots.....		83
Pre-processed Data.....		86

## Table of Figures

Figure 1 Logistic Function Formula .....	11
Figure 2 Logistic Regression Formula .....	11
Figure 3 Random Forrest Regressing Formula .....	12
Figure 4 Bayes' theorem .....	14
Figure 5 Continuous Distance Functions.....	Figure 6 Categorical Distance Function .....
Figure 7 Data after the first round of manual feature selection.....	22
Figure 8 Data after second round of manual feature selection .....	23
Figure 9 Age dispersion across the dataset.....	24
Figure 10 Gender distribution across the feature .....	24
Figure 11 Pain type across the feature.....	25
Figure 12 Rest BP distribution across the dataset.....	Figure 13 BP Chart.....
Figure 14 CHOL distribution across the dataset .....	26
Figure 15 Family History of heart disease .....	26
Figure 16 Fasting Blood sugar .....	27
Figure 17 Resting ECG .....	27
Figure 18 Max Heart rate Achieved .....	28
Figure 19 Resting HR distribution for the dataset.....	28
Figure 20 Exercise Induced Angina across the dataset .....	29
Figure 21 OldPeak distribution across the dataset .....	29
Figure 22 Slope of peak exercise ST segment.....	30
Figure 23 major Vessels coloured by Fluoroscopy .....	30
Figure 24 Thal Representing heart state .....	31
Figure 25 Num feature representing presence of disease .....	31
Figure 26 Intrinsic discrepancy Formula.....	32
Figure 27 Intrinsic Discrepancy Table for the Features .....	32
Figure 28 WEKA Logistic Regression Output for Unprocessed Dataset .....	42
Figure 29 Weka Logistic Regression Class Accuracy for Unprocessed Dataset .....	42
Figure 30 WEKA Logistic Regression Confusion Matrix for Unprocessed Dataset.....	42
Figure 31 WEKA Logistic Regression Output for Pre-processed Dataset.....	43
Figure 32 WEKA Logistic Regression Class Accuracy for Pre-processed Dataset.....	43
Figure 33 WEKA Logistic Regression Confusion Matrix for Pre-processed Dataset .....	43
Figure 34 WEKA Random Forrest Output for Unprocessed Dataset .....	44
Figure 35 WEKA Random Forrest Class Accuracy for Unprocessed Dataset .....	44
Figure 36 WEKA Random Forrest Confusion Matrix for Unprocessed Dataset.....	44
Figure 37 WEKA Random Forrest Output for Pre-processed Dataset.....	45
Figure 38 WEKA Random Forrest Class Accuracy for Pre-processed Dataset.....	45
Figure 39 WEKA Random Forrest Confusion Matrix for Pre-processed Dataset .....	45
Figure 40 WEKA Support Vector Machine Output for Unprocessed Dataset.....	46
Figure 41 WEKA Support Vector Machine Class Accuracy for Unprocessed Dataset.....	46
Figure 42 WEKA Support Vector Machine Confusion Matrix for Unprocessed Dataset .....	46
Figure 43 WEKA Support Vector Machine Output for Pre-processed Dataset.....	47
Figure 44 WEKA Support Vector Machine Class Accuracy for Pre-processed Dataset .....	47
Figure 45 WEKA Support Vector Machine Confusion Matrix for Pre-processed Dataset .....	47
Figure 46 WEKA K Nearest Neighbours Output for Unprocessed dataset .....	48
Figure 47 WEKA K Nearest Neighbours Class Accuracy for Unprocessed dataset .....	48
Figure 48 WEKA K Nearest Neighbours Confusion Matrix for Unprocessed dataset.....	48
Figure 49 WEKA K Nearest Neighbours Output for Pre-processed Dataset.....	49

Figure 50 WEKA K Nearest Neighbours Class Accuracy for Pre-processed Dataset.....	49
Figure 51 WEKA K Nearest Neighbours Confusion Matrix for Pre-processed Dataset .....	49
Figure 52 WEKA Linear Discriminant Analysis Output for Unprocessed Dataset.....	50
Figure 53 WEKA Linear Discriminant Analysis Class Accuracy for Unprocessed Dataset .....	50
Figure 54 WEKA Linear Discriminant Analysis Confusion Matrix for Unprocessed Dataset ...	50
Figure 55 WEKA Linear Discriminant Analysis Output for Pre-processed Dataset .....	51
Figure 56 WEKA Linear Discriminant Analysis Class Accuracy for Pre-processed Dataset .....	51
Figure 57 WEKA Linear Discriminant Analysis Confusion Matrix for Pre-processed Dataset..	51
Figure 58 WEKA Naive Bayes Output for Unprocessed Dataset.....	52
Figure 59 WEKA Naive Bayes Class Accuracy for Unprocessed Dataset.....	52
Figure 60 WEKA Naive Bayes Confusion Matrix for Unprocessed Dataset .....	52
Figure 61 WEKA Naive Bayes Output for Pre-processed Dataset .....	53
Figure 62 WEKA Naive Bayes Class Accuracy for Pre-processed Dataset .....	53
Figure 63 WEKA Naive Bayes Confusion Matrix for Pre-processed Dataset.....	53
Figure 64 Logistic Regression coefficients for the features .....	56
Figure 65 Fitting results for the model .....	57
Figure 66 Intrinsic Discrepancies Output for Training Dataset .....	58
Figure 67 Logistic Regression Output Score for Training Dataset using Lasso .....	58
Figure 68 Classification Report for Training Dataset using Lasso .....	58
Figure 69 Confusion Matrix for Training Dataset using Lasso.....	58
Figure 70 Output for Automatic Feature Selection on Training Dataset .....	59
Figure 71 Training Data Probability Plot .....	60
Figure 72 Intrinsic Discrepancies Output for Testing Dataset .....	61
Figure 73 LR using Lasso for Testing Dataset.....	61
Figure 74 Classification Report for Testing Dataset using Lasso .....	61
Figure 75 LR using lasso Testing Dataset Confusion Matrix .....	61
Figure 76 Output for Automated Feature Selection for Testing Dataset.....	62
Figure 77 Testing Dataset Probability Plot .....	62
Figure 78 Intrinsic Discrepancies Output for Complete Dataset.....	63
Figure 79 LR using Lasso for Complete Dataset .....	63
Figure 80 Classification Report for Complete Dataset using Lasso.....	63
Figure 81 LR using lasso Complete Dataset Confusion Matrix.....	63
Figure 82 Output for Automated Feature Selection for the complete Dataset .....	64
Figure 83 Complete Dataset Probability Plot.....	64
Figure 84 Naive Bayes Output .....	65
Figure 85 Random Forrest Output.....	65
Figure 86 Support Vector Machine Output.....	66
Figure 87 Kullback-Leibler Distance for discrete features .....	70
Figure 88 Kullback-Leibler distance for Continuous features .....	70
Figure 89 Kullback-Leibler Distance Symmetrised .....	70



# 1 Introduction

## 1.1 Project Overview

The project aims to predict the presence of heart disease in a patient given a number of factors. To do this there will be an implementation of Machine Learning (ML) applied to the dataset. For this project the data set was obtained from the UCI ML repository(1). Using ML algorithms such as Logistic Regression (LR) it is possible to make predictions of probability given there is enough features and data to train a model.

Before considering what model to implement there was the question, what dataset to use as there are plenty of datasets out there that focus on heart disease?

The dataset that was selected for this project was a combination of the four sources in the UCI ML repository, the reason for this is they have been used for ML model training in the past with outcomes of success.

After selecting a dataset the data quality analysis and reporting was done, this is to determine the characteristics of the dataset, such as detection of outliers and anomalies. For this project there were two datasets a pre-processed and an unprocessed, both consisted of the four sources from the repository, also both were examined and analysed for the data quality report.

This was to ascertain if there were other features that may have been over looked when the original data was processed. This produced some interesting results as after completing the data quality analysis and feature selection the remaining features for the two datasets were the same bar the unprocessed dataset retaining two additional features after feature selection.

They were family history and the resting heart rate features. For the most part, the plotting of both datasets look the same, apart from some outliers in the unprocessed dataset which were expected due to Null values being replaced with a zero value. The final section of this report will cover the findings of the testing and development, along with any further plans for the project.

## 1.2 Project Objectives

This project will apply ML algorithms or models to a selected number of attributes from the dataset to try and find out just how precise a prediction can be made. This is to determine the presence of heart disease in a patient or instance of the dataset. The UCI ML website lists a number of attributes that are selected for the application of ML.

It consists of fourteen in total including a predictive attribute for the presence of heart disease. Aside from this there are in total seventy-five attributes in the unprocessed dataset, the project seeks to see if there are in fact other attributes that may be useful in the prediction process, that may have been overlooked in previous projects. With a larger selection of features it is more likely the model will be more accurate. This will be done by analysing the complete set of features in the unprocessed dataset, with the results being compared to the pre-processed dataset.

The main objective of the project is to use ML to predict the presence of heart disease. If an accuracy of above 80% is achieved, this would be sufficient for the purposes of this project, but would only be a starting point for a real world application. This project will provide an understanding of what is involved in the application of ML process to a dataset and how to select the right dataset/algorithms.

There will be a brief investigation into the different ML algorithms that can be used, before one is selected for the final model. The project seeks to demonstrate the use of ML for classification and prediction while providing and understanding of what the outputs from the algorithms mean.

## 1.3 Project Challenges

The challenges for this project are very clear, there is very little time until the deadline, there is the need to learn Python and other technologies in a very short time leading to a lot of hacking around problems to find solutions. This may end up leading to messy code or even inefficiency in the running of the code such as memory or CPU usage. Then there is the learning of the actual ML which is only a module being currently undertaken, this means there is a need to surpass what is covered in this module in order to complete the project.

Aside from this there are a whole host of applications and algorithms for implementation, meaning deciding on what is best to use is unknown to the author, thus also needing research.

This is because the ML algorithms that are needed to implement a project like this can be very complex and very difficult to both implement and understand. This means there is a need to get an in depth understanding of the data being used, to try and determine what ML algorithms or models are best suited to a project like this.

## 1.4 Structure of the document

In the research section, the research will address the different algorithms and languages that could have been used along with sources of data. There will be an analysis of the dataset and the performance of the algorithms that were applied to the dataset, this will include a data quality report.

Section 2.1 covers the research into the problem area that is heart disease, the research into the solution can be found after this in section 2.2.1. Section 2.3 of this document covers the ML algorithms that were investigated for the project. A discussion on some of the technologies that could have been used including the ones that were can be found in section 2.4.

The data selection is covered in section 3.1, while the data quality analysis and report can be found in section 3.2, this section also covers the feature selection stage. This is where the features for the training of the model are selected. This is to determine the best combination of features for the final training of the ML algorithms.

In sections 4.2 through to 4.7 the fitting and training of the ML models are investigated using Weka, and sections 5.1 through to 5.4 using Scikit-Learn. It is in these sections that the analysis of the different ML algorithms and technologies are compared. In section 5.5 the analysis of the model outputs from the selected technologies and algorithms are covered.

Section 6 covers the source code, models and implementation of the project. In this section there are details about the functions and process involved in a project like this. The final sections contain a discussion of the project findings and conclusions, refer to section 8, this is where the learning outcomes are discussed along with any further plans for the project, refer to section 7. It also addresses things the author would have done differently should the project be restarted.

## 2 Research

### 2.1 Researching the Problem

It is no secret that heart disease costs a fortune to treat in just about every country in the world. The World Heart Federation's World Congress of Cardiology & Cardiovascular Health (WCC 2016), published a report in June of that year. It stated that the cost of heart disease to Latin America alone was in excess of 30 billion dollars to treat(2).

This means that there is no shortage of reports and research into the area. Research conducted by the Heartbeat Trust in collaboration with the Irish Heart Foundation and the National University of Ireland Galway (NUIG), found that the total cost of heart failure is approximately 666 million euros. This is also predicted to increase in the future(3).

The findings of the report from the authors understanding into the problem area are that heart failure effects around 250,000 Irish people. The direct cost to the Health Service Executive (HSE), is estimated to be 158 million annually for heart disease alone. This includes hospitalisations, General Practitioners (GP) visits, nursing home care and other services where applicable.

The cost of heart failure in Ireland is now in excess of 660 million euros, this comprises of direct costs for informal care, equating to about a third of the total cost and the costs of premature deaths accounting for the remainder(3).

Just under half of the direct costs are hospital related, heart failure patients account for more than 231,000 hospital bed days each year, with 7% of all HSE inpatient bed days being a result heart failure. Primary care related costs, such as GP visits account for 25% of the total direct cost of heart failure(3).

It also states that the spending on cardiovascular health in Ireland is just 6% of the healthcare budget, this is lower than the reported EU average of 10 %(3).

With all the other research into the prediction of heart disease across the globe it is a very quickly advancing area, couple this with the right application of ML and a mass collection of data from as many credible sources as possible and this may be the best

solution for issues such as this. There are a number heart datasets that could have used for the purposes of this project but the most used is as mentioned previously the UCI ML dataset. It was collected from four sources which will be addressed further on in the report.

With the emergence of Artificial Intelligence (AI) over the past number of years it is only clear to see the benefit of applying ML to the processing and classification of large datasets in recent years.

## 2.2 Researching the Solution

### 2.2.1 Machine Learning in Healthcare

There are two main forms to ML, they are supervised and unsupervised. Unsupervised learning is more geared towards learning patterns or distributions about a dataset rather than trying to obtain a definite output from an ML algorithm. It is used for exploratory modelling on a dataset or to find the more important features(4).

Supervised learning is the other main form of ML, each set of feature values are supplied with a given label or class. This essentially means these features are mapped to the corresponding classes or labels, this is so they can be used for comparison in order to predict labels for future sets of features(4).

Companies like Google have used the application of ML algorithms to detect a wide range of health issues(5). Googles DeepMind Health Project Aims to tackle a number of different areas in the health sector including a new way of diagnosing people using ML.

There has been a lot of interest into this area and with companies like Google, Fitbit and Microsoft, collecting massive amounts of data. Be it from a phone or fitness tracker there is an endless supply of fresh data, meaning this is very possible.

While this is not a very new problem it is something that can be combatted using new technologies like ML. It is only in the recent years that such companies have started to pay real attention to what can be achieved using ML. There have been software's that have been developed to tackle some of these issues and have proven to be better than a doctor in some cases (6), such as the aforementioned Google project.

There are countless projects that are setting out to tackle issues in the medical field using ML it's not just prediction of illness. Disease identification and diagnosis, is at the forefront of ML research in medicine.

A report in 2015 issued by the Pharmaceutical Research and Manufacturers of America, more than 800 medicines and vaccines to treat cancer were in trial(7). This was addressed in an interview with Bloomberg Technology, Knight Institute Researcher Jeff

Tyner said that while exciting as this may be, it also presents challenges, like finding ways to work with all the resulting data. “That is where the idea of a biologist working with data scientists is so important”(6).

This is an interesting time to be involved in ML, largely due to the volume of areas that are currently being explored using ML. Areas like cancer identification and treatment were just some of the first areas tackled. In October 2016, IBM Watson Health announced IBM Watson Genomics, a partnership initiative with Quest Diagnostics, its aimed to make strides in precision medicine by integrating cognitive computing and genomic tumour sequencing(9).

Personalised medicine, is defined as a more effective treatment based on individual health data paired with predictive analytics, this massive research area and closely related to better disease assessment. The domain primarily consists of supervised learning. IBM Watson Oncology is a leading institution at the forefront of driving change in treatment decisions, using patient medical information and history to optimize the selection of treatment options(7).

In the coming future there will be an exponential increase in the use of Nano biosensors and devices, as well as mobile apps and sophisticated wearables with remote monitoring capabilities, which will provide an unfathomable amount of data that can be used to aid in the research and development of treatment efficiency.



## 2.2.2 Terminology and Concepts

This section covers some of the terms and concepts that are key to the application of ML. The explanations are the authors and are based on the learning and understanding gained throughout this project. They are high level “broad” descriptions of the terms. For a more in depth discussion on anything covered in this report please see the appendix for a listing of all research materials.

- ST: The ST segment is the flat, isoelectric section of the ECG between the end of the S wave (the J point) and the beginning of the T wave. It represents the interval between ventricular depolarization and repolarization
- Dataset: this is the data used in the model, it is normally stored in rows and columns.
- Feature: a feature is a column in the dataset.
- Classifier: it is a machine learning algorithm that maps features of a row to its label. In the case of this project it is disease or no-disease.
- Model: this is the result of the classifier being trained.
- Cleaning data: normalising each of the features so that they are in the same format.
- Training set: this is part of the dataset that is used to train the model with a classifier. This is usually like an 80% or 90% of the whole dataset.
- Test set: this is the other part of the dataset held back from training. This data is then given to a model without the labels to classify. As the model has not seen any of this data before, it can be taken as a validation set.
- Instance: one row of data in the dataset.
- Cross Validation: that is, separating the dataset into smaller sets, then using one set as the unseen test set and the rest as the training set. Then rotating the test set back into the training set and using a different one from the training set as the new testing set. This is used to ensure all of the data will be used as test and training data at some point in the validation.
- N-Fold Cross Validation: this involves splitting the dataset into N number of smaller subsets or folds.
- Average recall: also known as average class accuracy. This is the average accuracy of the model.

- Overfitting: overfitting occurs when tailoring a model very specifically to a dataset. This can cause a much higher fluctuation in the accuracy for completely new data leading to inaccurate results.
- Under fitting: under fitting is also an issue, though it is far less common in ML when compared to overfitting. This normally happens when not enough features are used to effectively represent the data.
- Confusion Matrix: There are a number of things that can be calculated using this.
  - Accuracy such as, how often the classifier is correct
    - $(TP+TN)/total$
  - Misclassification Rate or how often the model is wrong
    - $(FP+FN)/total$
  - True Positive Rate/Recall
    - $TP/(FN+TP)$
  - False Positive Rate: When it's actually no, how often does it predict yes.
    - $FP/(TN+FP)$
  - Specificity, how often the model gets it right.
    - $TN/(TN+FP)$
  - Precision, when it predicts yes and is correct.
    - $TP/(FP+TP)$
  - Prevalence, or how often does the yes condition actually occur.
    - $(FN+TP)/total$

## 2.3 Machine learning Algorithms

### 2.3.1 Logistic Regression

LR classification is somewhat similar to Naïve Bayes (NB) classification in the respect that both implement a variation of probability in their algorithms. Then applying this probability to the test data to make a prediction. LR builds upon the logistic function(8). The function is defined in figure 1(9).

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

*Figure 1 Logistic Function Formula*

An LR classification model can have many input variables, this is a contrast to the logistic function above which has a single x value as input. It also consists of weights or coefficients for each of the input variables. This then means that the resulting LR algorithm is as defined in figure 2(10).

$$f(i) = \beta_0 + \beta_1 x_{1,i} + \dots + \beta_m x_{m,i},$$

*Figure 2 Logistic Regression Formula*

That is 1 over the right side of the formula above were the value,  $\beta$ , is the regression coefficient which generally depends on the other independent variables within the given set, meaning the value of a partial coefficient for one independent variable will vary, depending on the other independent variables included in the regression equation(11).

To date logistic regression has been used in a Six Sigma project in health care, that aimed to predict if a patient would go with a rehabilitation programme or not given specific features(12). It has also been used in predicting the length of a hospital stay(13).

### 2.3.2 Random Forrest

The Random Forest (RF) is a type of additive model, it makes predictions by combining decisions using a sequence of base models. The RF model is defined below in figure 3.

$$g(x) = f_0(x) + f_1(x) + f_2(x) + \dots$$

*Figure 3 Random Forrest Regressing Formula*

This is where the output or final model  $g$  is the sum of simple base models  $f_i$ . Meaning that each base classifier is in its self a simple decision tree. This is a very broad technique of using multiple models for getting a better predictive performance result. It is called model ensembling, in random forests all base models are constructed independently(14).

Applications within the health care sector, where RF has been applied include Diabetic Retinopathy Classification Analyses(15). It has also been used in the detection and prediction of Alzheimer's disease(16).

### 2.3.3 Support Vector Machines

Support Vector Machines (SVMs) work by plotting each instance as a point in either a 2D or 3D space. Linear SVMs aim to separate the classes within this plotted space using a straight-line that supports the 'optimum separating hyperplane', this is an area parallel to the separating line. The 'optimum separating hyperplane' is the plane that has the largest gap between the outer bounds of the plane, thus being the largest area. This allows for the plotting of new and unseen data in the same way. Depending on where it is plotted in relation to the optimal hyperplane, the label is predicted. In a linear SVM, that has less variance within the class instances, the hyperplane can be defined well and there is clear separation between classes(17).

SVM has been used in the prediction of diabetes(18). It has also been used for providing automatic medical diagnosis(19).

### 2.3.4 Linear Discriminant Analysis

Similar to an LR model, but without the limitations of having a two class classifier. What this means is that if you have more than two classes or prediction outcomes, then Linear Discriminant Analysis (LDA) would be a better classification technique.

A representation of LDA consists of statistical properties of the data, calculated for each class. For single variable (x) input, this is the mean and the variance of the variable for each class and with multiple variables it is the same, properties are calculated over the multivariate Gaussian, or the means and the covariance matrix.

These properties are estimated from the data and input into the LDA equation in order to make predictions. LDA makes predictions by estimating the probability that a new set of inputs belongs to each class.

The class that gets the highest probability is the output class and a prediction is made. The model uses Bayes Theorem to estimate the probabilities(20) this is depicted in figure 4 below.

LDA was been used for early detection of mental health disorders only last year(21).

### 2.3.5 Gaussian Naive Bayes

The Gaussian Naive Bayes (GNB) classifier uses the probability of each feature in each class to determine the prediction outcome. Then the probability for a feature value appearing in each class is calculated separately to the other features of the set.

It is these probabilities that are then combined to get the overall probability for that given instance for each class. The class that has the highest probability is selected based on the combined probabilities from the training data. GNB classifiers are probabilistic algorithms, they use Bayesian probability theory while assuming no correlation between the features of the set, hence the classifier is considered naive(22).

*“Given a hypothesis  $H$  and evidence  $E$ , Bayes' theorem states that the relationship between the probability of the hypothesis  $P(H)$  before getting the evidence and the probability of the hypothesis after getting the evidence  $P(E|H)$  is ”(23).*

$$P(H | E) = \frac{P(E | H)}{P(E)} P(H).$$

Figure 4 Bayes' theorem

GNB has been used in a whole multitude of applications such as heart disease detection(24).

### 2.3.6 K Nearest Neighbours

K nearest neighbours (KNN) is an algorithm that stores all available cases and classifies new cases based on a similarity measure using distance functions. This means a case is classified by a majority vote cast by its surrounding neighbours, then the case is simply assigned to the class of its nearest neighbour. In the instance of categorical variables the Hamming distance must be used, whereas when dealing with continuous variables there are 3 distance formula that can be applied, they are depicted on the next page(25).

Euclidean	$\sqrt{\sum_{i=1}^k (x_i - y_i)^2}$
Manhattan	$\sum_{i=1}^k  x_i - y_i $
Minkowski	$\left( \sum_{i=1}^k ( x_i - y_i )^q \right)^{1/q}$

$$D_H = \sum_{i=1}^k |x_i - y_i|$$

Figure 5 Continuous Variable Distance Functions

Figure 6 Categorical Variable Distance Function

The optimal value for K needs to be selected, this is done by first inspecting the data. A large K value is more precise as it reduces the overall noise, however there is no guarantee. Cross-validation is another way to determine a good K value by using an independent dataset to validate the K value. The optimal K for most datasets is in the range of 3 to 10. That produces much better results than 1NN(26). KNN has been used in common disease diagnosis(27).

## 2.4 Overview of the technologies

When undertaking a project in ML there are a number of things that need to be considered, starting with what language to use. This poses a dilemma as the best languages for doing anything with data and ML are R and Python(28).

Not having programmed in either language the next best choice was either SQL or Java. With SQL being a restricted but yet a powerful language, it does not lend its self well to functional programming.

This would mean having to use Python for functional aspects of the project. SQL is also susceptible to hacking using SQL injection rendering it possibly unsafe and this is why SQL has not been explored further.

Java while being far more flexible than SQL, also has some drawbacks. When dealing with very large data structures and also large volume of data, Java is worth considering as an implementation language. To implement such a project in Java there are libraries for ML implementation, some of them are outlined in sections 2.4.1 and 2.4.2 along with the most relevant ones for Python, covered in sections 2.4.3 through to 2.4.7.

### 2.4.1 Weka

Weka is often referred to as number one for the best Java library when it comes to ML implementation in Java(29). Weka is a fully Java-based workbench, it is well suited for the implementation of ML algorithms. It is primarily used for data mining, data analysis and predictive modelling. It's free, portable and has a simple and intuitive graphical user interface. Weka can cover classification, it supports clustering, association rule mining, time series prediction, feature selection and anomaly detection(30).

Weka's collection of ML algorithms can be applied either directly to a dataset or called from within Java code. It supports several standard data mining tasks, including data pre-processing, classification, clustering, visualization, regression, and feature selection(31).

This application was used for initial investigation of the ML algorithms to determine the accuracy, research on this can be found in section 4 of this document.

## 2.4.2 MALLET

Developed for the most part by Andrew McCallum along with some other students from UMASS and UPENN(32). MALLET is an open-source java ML toolkit for language to text. The Java-based package supports statistical and natural language processing, clustering, document classification, information extraction, topic modelling, and other ML applications. It includes sophisticated tools for document classification such as routines for converting text. It also supports a variety of algorithms which includes, GNB, Decision Trees, Maximum Entropy and code for evaluating classifier performance(33).

This means that it could be done in Java but it is not the number one choice for ML implementation(28). It comes down to the fact that languages like Python and R, are just better suited to ML than Java(28). This is due to there not being a library for ML in Java that comes with a vast amount of testing and documentation(34), and again this is due to R and Python as they are better suited to data manipulation(28).

## 2.4.3 NumPy

One of the most used libraries in Python, is the NumPy library, which stands for Numerical Python. It provides a whole host of useful features for operations on n-arrays and matrices in Python. The library can provide vectorization of mathematical operations on a NumPy array type, which improves performance and speeds up the execution of the calculations(35).

## 2.4.4 SciPy

Another Python library is SciPy, this is a library for engineering and science. There is a need to understand the difference between SciPy Stack and SciPy Library, Python's SciPy Stack is a collection of software specifically designed for scientific computing in Python, the SciPy library is part of this stack(36). SciPy contains modules used for linear algebra, optimization, integration and statistics(37). The main functionality of the SciPy library is built upon NumPy and its arrays(38). It can provide efficient numerical routines as numerical integration, optimization, and many others through its specific submodules which are very well documented(39).



## 2.4.5 Pandas

Pandas is another Python package it was designed to work with labelled and relational data. Pandas is a perfect tool for data manipulation, aggregation, and visualization(40). Pandas does not implement any advanced modelling functionality apart from linear and panel regression, to overcome this Scikit-learn will be applied.

Just a snippet of things that a developer can do with Pandas are listed below.

- Adding and dropping columns to a data frame
- Convert complex data structures to DataFrame objects
- Can handle missing data, represents with NaN
- Contains a powerful grouping by functionality

## 2.4.6 Scikit-Learn

Scikits, are the additional packages of the SciPy Stack designed for specific functionalities such as image processing and ML facilitation(41). When it comes to the ML implementation, one of the most important of these packages is Scikit-learn.

The package is built on the top of SciPy and makes heavy use of its math operations. Scikit-learn provides a clear and concise interface to the most common ML algorithms. The library combines a high standard code example set and it is very well documented. It offers high performance for ML with Python from even a basic machine.

## 2.4.7 Matplotlib

Matplotlib is another SciPy Stack package and Python Library that has been designed for the generation of simple but powerful visualizations(42). It is a very complete piece of software. Matplotlib combined with NumPy, SciPy, and Pandas are what makes Python a competitor to other scientific tools such as MatLab or Mathematica.

This being said the library is pretty low-level, meaning that there is a need to write more code to reach the advanced levels of visualizations.

Some of the things that can be done with Matplotlib include:

- Line plots
- Scatter plots
- Bar charts and Histograms
- Pie charts
- Stem plots
- Contour plots
- Quiver plots
- Spectrograms

There are facilities for creating grids, labels, legends, and many others for formatting entities with Matplotlib. The library is also supported by different platforms and makes use of different GUI kits for the depiction of resulting visualizations. The library is also supported in Integrated Development Environments like IPython, Spyder and Jupyter Notebooks offering full functionality across all platforms.

For this project Python has been elected as the implementation language, the reason for this is there are some very good libraries in Python, which are outlined below and with a basic knowledge of Python it is slightly less to learn for this project.

## 3 Dataset

### 3.1 Dataset Selection

For this project there are a number of sources that could have been used, some of them are pushed forward by competitions like the Kaggle's current one for heart disease prediction(43). Kaggle is a good source of data as it has a vast amount of datasets. Then there is the one which was selected for this project, it was obtained from the UCI ML repository(1). It was compiled from the four sources that follow:

- Hungarian Institute of Cardiology. Budapest: Andras Janosi, M.D.
- University Hospital, Zurich, Switzerland: William Steinbrunn, M.D.
- University Hospital, Basel, Switzerland: Matthias Pfisterer, M.D.
- V.A. Medical Center, Long Beach and Cleveland Clinic Foundation: Robert Detrano, M.D., Ph.D.

The unprocessed dataset has a total 76 attributes, almost all published experiments refer to using a subset of 14 of them, in particular.

*“The "goal" field refers to the presence of heart disease in the patient. It is integer valued from 0 (no presence) to 4. Experiments with the Cleveland database have concentrated on simply attempting to distinguish presence (values 1, 2, 3, 4) from absence (value 0). The names and social security numbers of the patients were recently removed from the database, replaced with dummy values(1).”*

The listed attribute in the dataset that are normally used when ML is being applied are listed below:

1. #3 (age) this is the patient age in years, type integer.
2. #4 (sex) this is the gender for the patient (0=Female, 1=Male), type binary
3. #9 (cp) Chest pain type (0=typical angina, 1=atypical angina, 2=non-angina and 3= asymptomatic angina), type categorical
4. #10 (trestbps) Resting blood pressure (mm Hg), type continuous
5. #12 (chol) Serum cholesterol (mg/dl), type continuous
6. #16 (fbs) Fasting blood sugar (< 120 mg/dl or > 120 mg/dl), type binary

7. #19 (restecg) Resting electrocardiography results (0=normal, 1=ST abnormality and 2=left ventricular hypertrophy), type categorical
8. #32 (thalach) Maximum heart rate achieved during a thallium stress test, type continuous
9. #38 (exang) Exercise induced angina (0=no, 1=yes), type binary
10. #40 (oldpeak) ST depression induced by exercise relative to rest, type continuous
11. #41 (slope) Slope of peak exercise ST segment (1=upsloping, 2=flat, 3=downsloping), type categorical
12. #44 (ca) The number of major vessels coloured by fluoroscopy, type integer in the range of 0 and 3
13. #51 (thal) Thallium stress test result (1=normal, 2=fixed defect, 3=reversible defect), type categorical
14. #58 (num) (the predicted attribute) Heart disease status: number of major vessels with >50% narrowing, type integer in the range of 0 and 4

The full listing of features can be found in the appendix of this document or at the UCI ML repo also listed in the bibliography of this document. This project also aims to see if there are any other features that may be used, for this reason the full list of features were analysed.

## 3.2 Data Quality Analysis

The first line in the Cleveland data set was as follows (1\_0\_63\_1\_9\_9\_9), replacing the underscore with a simple space was done manually as it was only the first row that had this issue.

There is a script to merge all the data from the four sources mentioned above to form the complete dataset. This script will be covered in a later section of this report. When reading in the data the script takes in null values and in place puts a (0), this is the first part of the data processing.

After creating another script to read in all the data in to a data frame for analysis it was clear to see that there were some features that had not been used at all as mentioned in the description of the dataset, this can be found in appendix A of this document, and marked as either junk or not used(44).

These features were removed from the data frame along with any other irrelevant features as indicated by the data description leaving the rest for further processing and analysis.

There is a real need to retain the integrity of the data, this is so there is no bias in the data when it comes to training the model. This will also be addressed by splitting the processed into training and testing sets, this will be covered in more detail below. With the remaining features a data quality table was constructed, it is presented below in figure 7.

	Count	Cardinality	Min	1st Quart	Mean	Median	3rd Quart	Max	Std. Dev.
<b>Features</b>									
<b>age</b>	899.0	51	0.0	47.0	53.42	54.0	60.0	77.0	9.60
<b>sex</b>	899.0	3	0.0	1.0	0.85	1.0	1.0	58.0	1.95
<b>relrest</b>	614.0	4	0.0	0.0	0.69	1.0	1.0	9.0	0.58
<b>cp</b>	898.0	5	1.0	3.0	3.25	4.0	4.0	4.0	0.93
<b>trestbps</b>	840.0	62	0.0	120.0	131.98	130.0	140.0	200.0	19.64
<b>chol</b>	869.0	215	0.0	174.0	198.48	224.0	269.0	603.0	112.03
<b>fbs</b>	809.0	4	0.0	0.0	0.22	0.0	0.0	40.0	1.45
<b>famhist</b>	476.0	3	0.0	0.0	0.56	1.0	1.0	1.0	0.50
<b>restecg</b>	897.0	4	0.0	0.0	0.60	0.0	1.0	2.0	0.80
<b>thalach</b>	844.0	121	7.0	120.0	137.12	140.0	157.0	202.0	26.34
<b>thalrest</b>	843.0	77	37.0	65.0	75.55	74.0	84.0	154.0	14.95
<b>trestbpd</b>	840.0	35	0.0	80.0	83.55	80.0	90.0	120.0	10.28
<b>exang</b>	844.0	4	0.0	0.0	0.48	0.0	1.0	80.0	2.78
<b>slope</b>	591.0	6	0.0	1.0	1.76	2.0	2.0	3.0	0.62
<b>ca</b>	291.0	7	0.0	0.0	1.03	0.0	1.0	97.0	5.74
<b>thal</b>	421.0	8	1.0	3.0	5.01	6.0	7.0	7.0	1.95
<b>num</b>	899.0	6	0.0	0.0	1.22	1.0	2.0	82.0	2.98

*Figure 7 Data after the first round of manual feature selection*

This shows that there are some anomalies' in the data as the cardinality for the sex feature should be two as there can be only be either zero for a female and one for a male, same with the feature smoker, it should only have cardinality of two as well.

After looking at the remaining features, the ca feature while having a low count number it is important in the determining the presence of heart disease so it will be retained. This data was then plotted out into histograms and bar charts to further explore and analyse it.

After further analysis and removing further features that had inconsistent data for the range displayed across the feature. Another data quality analysis is displayed in figure 8.

	Count	Cardinality	Min	1st Quart	Mean	Median	3rd Quart	Max	Std. Dev.
<b>Features</b>									
<b>age</b>	898.0	50	28.0	47.0	53.48	54.00	60.0	77.0	9.44
<b>sex</b>	898.0	2	0.0	1.0	0.79	1.00	1.0	1.0	0.41
<b>cp</b>	898.0	4	1.0	3.0	3.25	4.00	4.0	4.0	0.93
<b>restbp</b>	898.0	60	0.0	120.0	124.47	130.00	140.0	200.0	36.02
<b>chol</b>	898.0	213	0.0	161.5	192.07	222.00	267.0	603.0	115.66
<b>famhist</b>	898.0	2	0.0	0.0	0.15	0.00	0.0	1.0	0.36
<b>fbs</b>	898.0	2	0.0	0.0	0.30	0.00	1.0	1.0	0.46
<b>restecg</b>	898.0	3	0.0	0.0	0.60	0.00	1.0	2.0	0.80
<b>thalach</b>	898.0	120	0.0	115.0	128.87	137.00	155.0	202.0	41.45
<b>thalrest</b>	898.0	76	0.0	63.0	70.75	73.00	84.0	139.0	23.15
<b>exang</b>	898.0	2	0.0	0.0	0.37	0.00	1.0	1.0	0.48
<b>oldpeak</b>	898.0	42	0.0	0.0	0.84	0.25	1.5	6.2	1.05
<b>slope</b>	898.0	4	0.0	0.0	1.16	1.00	2.0	3.0	0.98
<b>ca</b>	898.0	4	0.0	0.0	0.22	0.00	0.0	3.0	0.63
<b>thal</b>	898.0	4	0.0	0.0	2.35	0.00	3.0	7.0	2.84
<b>num</b>	898.0	5	0.0	0.0	1.13	1.00	2.0	4.0	1.26

*Figure 8 Data after second round of manual feature selection*

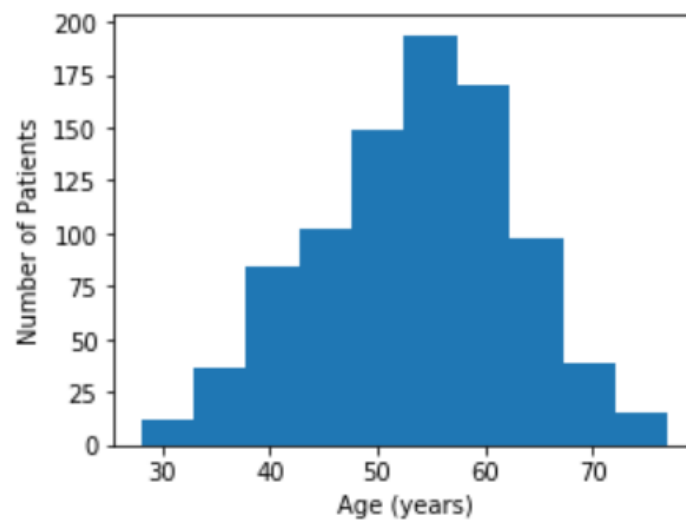
The thal and ca features have been retained as they were used in every other application of ML to this dataset, they will be accessed again after plotting of the remaining features to determine the validity of them as features. Then there is the famhist feature that has about half the values missing, this too will be accessed during the plotting of the remaining features.

Before plotting the dataset was brought into Microsoft Excel and viewed to see the missing data, the mean average could be added using the following function, this needs to be performed cell by cell.

`(=IF(D2="",AVERAGE(LOOKUP(2,1/(D$1:D1<>""),D$1:D1),INDEX(D3:D$12,MATCH(TRUE,INDEX((D3:D$12<>""),0),0))),D2))`

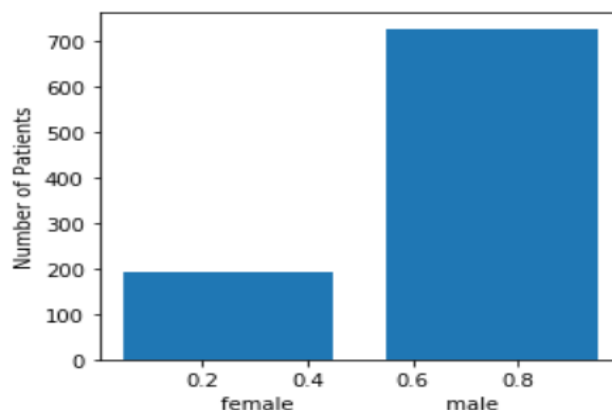
### 3.3 Dataset Explained

To address the number of missing values in the aforementioned features along with any other feature that had missing values had the (?), replaced with a zero as it has no effect on the plotting stage.



*Figure 9 Age dispersion across the dataset*

This shows that there is a nice bell curve or what is called a unimodal distribution across the feature with the central range between forty and sixty-five. This is exactly what is expected from the given dataset as most heart patients fall into the previously mentioned range.



*Figure 10 Gender distribution across the feature*

This shows that there is a far larger number of male instances in the dataset, this too is as expected given that males tend to have a higher rate of heart disease.



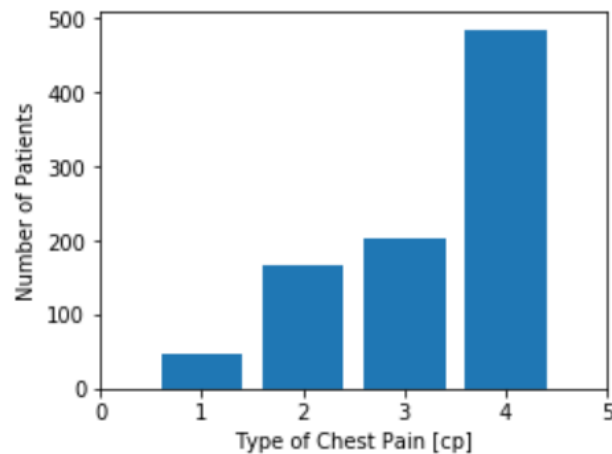


Figure 11 Pain type across the feature

This depicts the distribution of the chest pain type across the dataset they are as follows:

- Value 1: typical angina
- Value 2: atypical angina
- Value 3: non-angina pain
- Value 4: asymptomatic

This also shows that the majority of cases presented are complaining of asymptomatic pains in the chest.

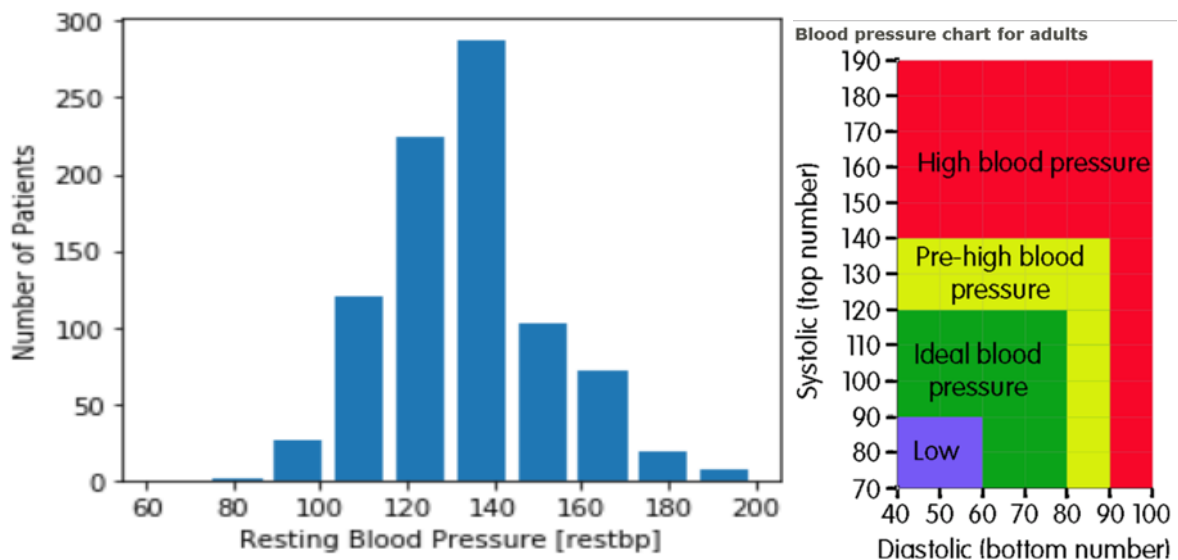
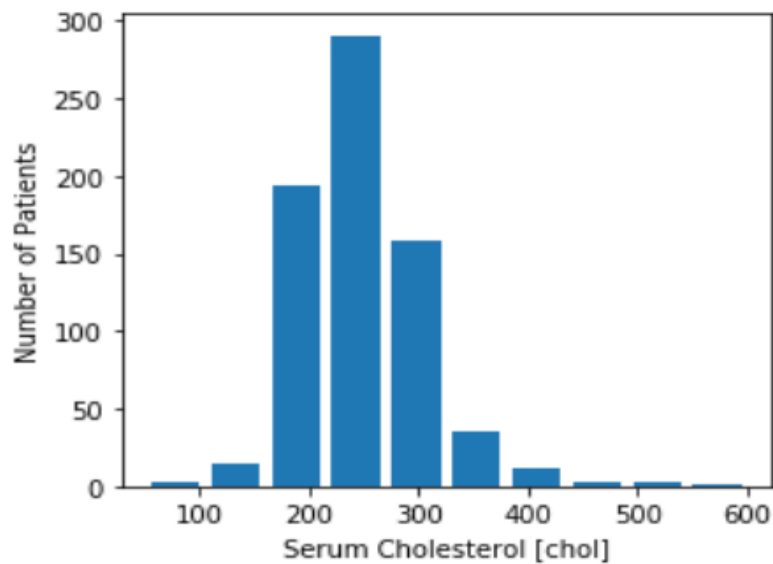


Figure 12 Rest BP distribution across the dataset

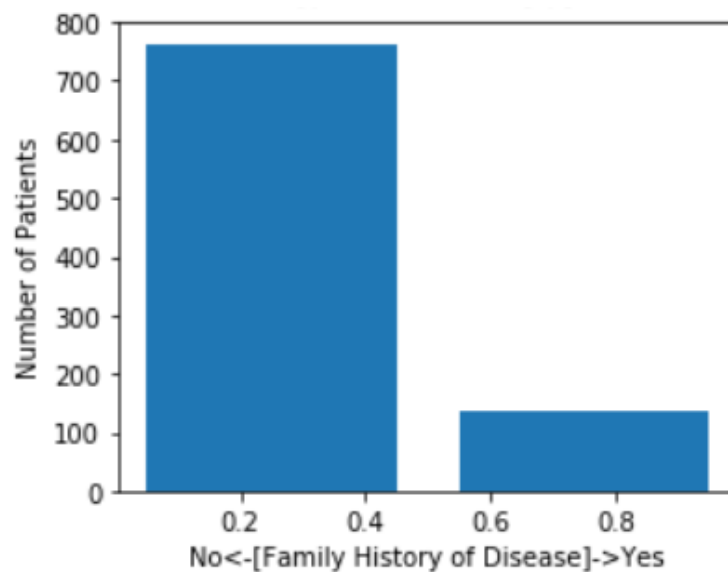
Figure 13 BP Chart

This shows a unimodal dispersion, as expected. The reason for the zero values entered in place of the missing values are not showing is they have been filtered out during the plotting. It is clear to see that there is a high number of instances that have either pre-high blood pressure or high blood pressure(45).



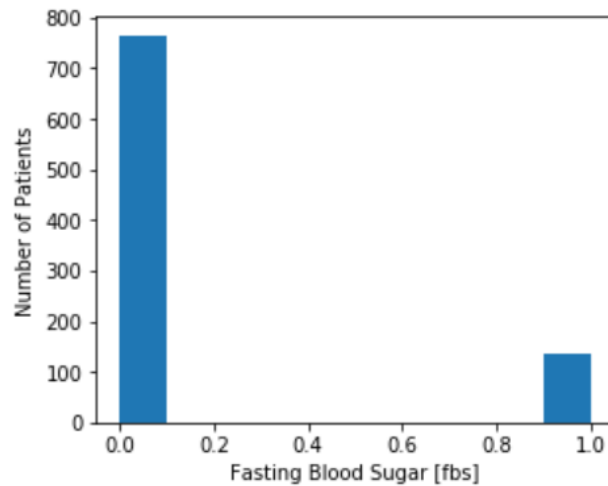
*Figure 14 CHOL distribution across the dataset*

This shows that the majority of instances in the dataset have very high levels of serum cholesterol as a healthy serum cholesterol level is less than 200 mg/dL(46).



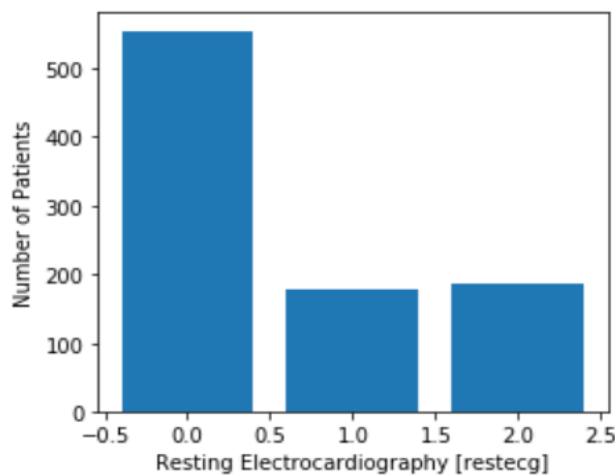
*Figure 15 Family History of heart disease*

This shows that the majority of the instances in the dataset have no family history of heart disease. This feature may still be discounted before the final dataset is completely ready for training the ML model.



*Figure 16 Fasting Blood sugar*

This data is represented using a true and false value where fasting blood sugar level is  $> 120$  mg/dl being stored as (1 = true; 0 = false). With the majority of the instances having normal levels as on average would be between 4.0 to 6.0 mmol/L (72 to 108 mg/dL) when fasting for a patient without diabetes(47).

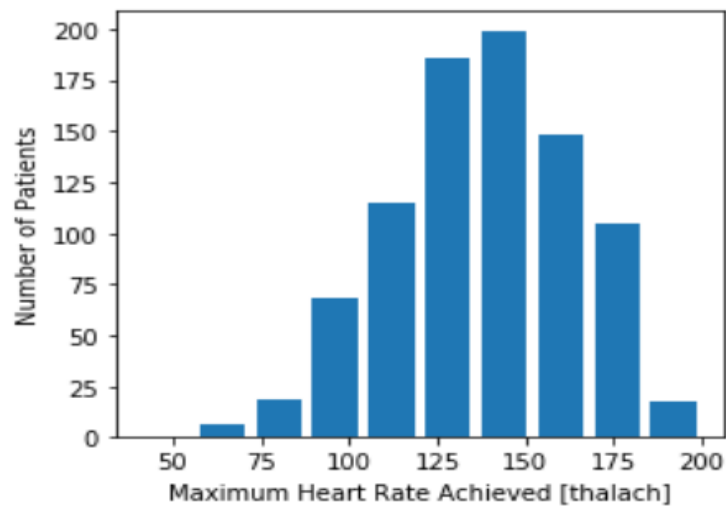


*Figure 17 Resting ECG*

This feature is stored with only three values they are as follows:

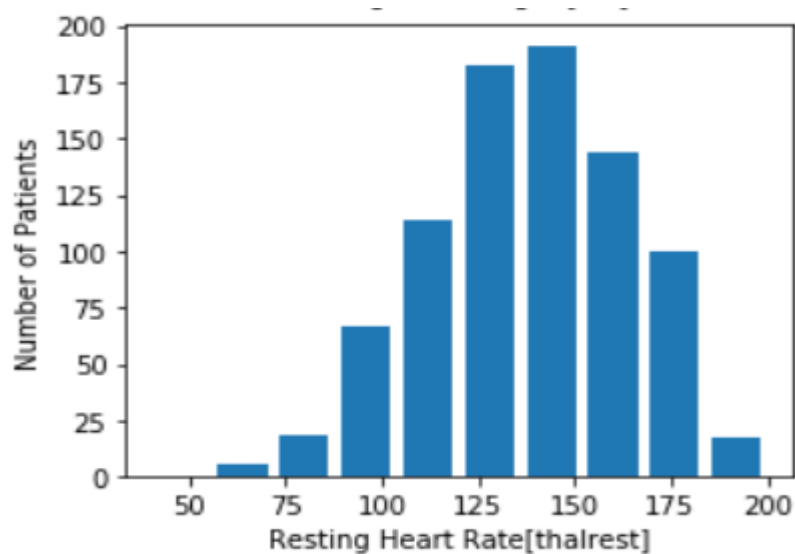
- Value 0: normal
- Value 1: having ST abnormality (T wave inversions and/or ST elevation or depression of  $> 0.05$  mV)
- Value 2: showing probable or definite left ventricular hypertrophy by Estes' criteria

It is then clear to see that the majority of the instances in the dataset have normal echocardiograph results.



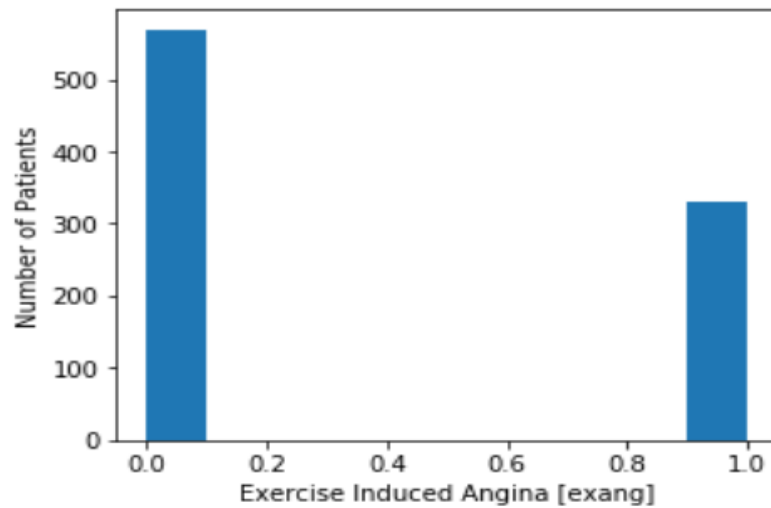
*Figure 18 Max Heart rate Achieved*

In this feature it is not clear to see that the zero values added in the previous stage of the data analysis as they have been filtered out during the plotting,



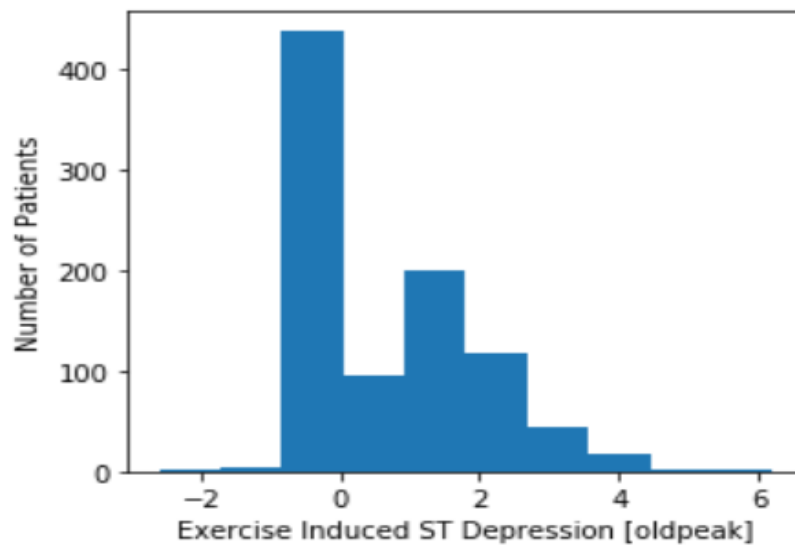
*Figure 19 Resting HR distribution for the dataset*

This feature is also showing the zero values as they were inserted to the dataset in place of the missing values, aside from them there is again the bell curve as expected. With the main range for the feature being 60 – 100, this is the normal range for a resting heart rate(48).



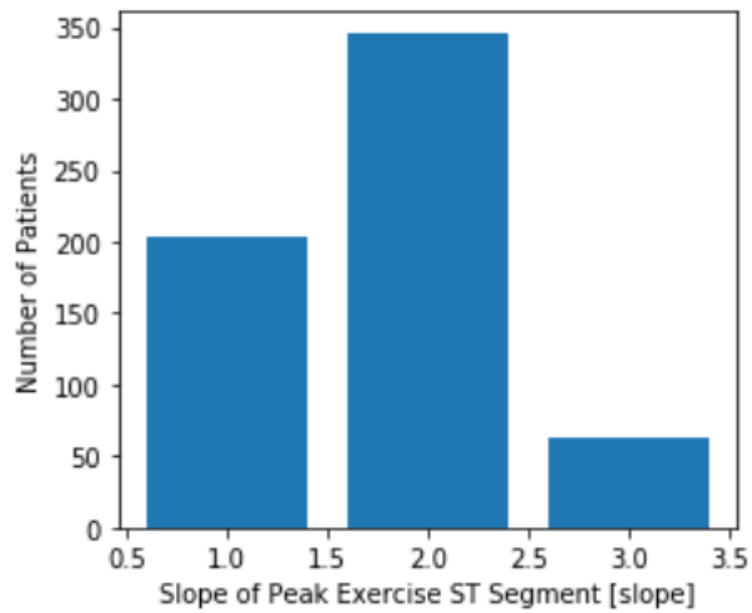
*Figure 20 Exercise Induced Angina across the dataset*

This feature is stored in the following way exercise induced angina (1 = yes; 0 = no). This is an angina type pain while exercising. Angina is caused by there not being enough blood getting through the arteries to supply the walls of the heart with enough blood flow to adequately pump(49).



*Figure 21 OldPeak distribution across the dataset*

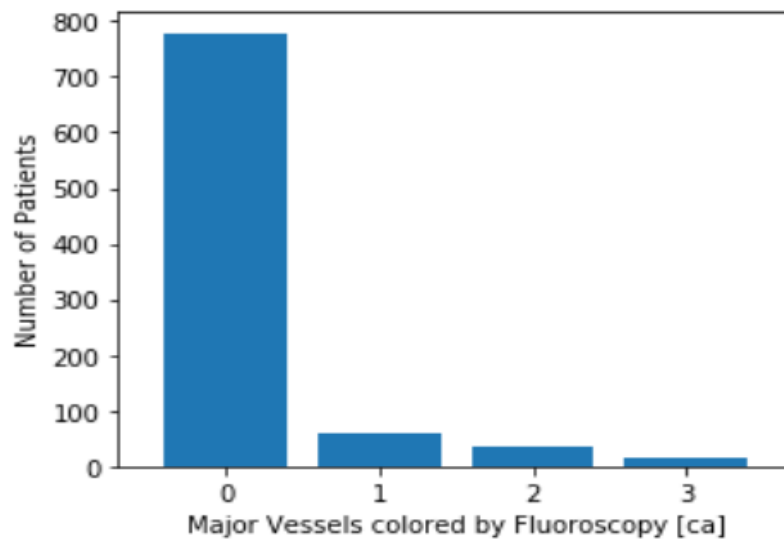
Oldpeak is ST depression induced by exercise relative to rest, this refers to an exercise stress test. This is where a patient normally uses a treadmill while connected to an ECG to determine heart rhythm and identifying heart conditions(50).



*Figure 22 Slope of peak exercise ST segment*

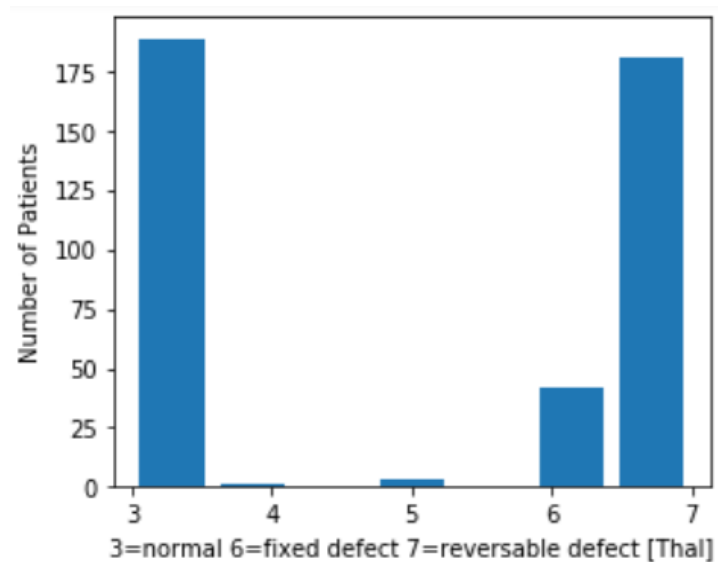
The feature slope has three values, what each value represents is listed below:

- Value 1: upsloping
- Value 2: flat
- Value 3: down sloping



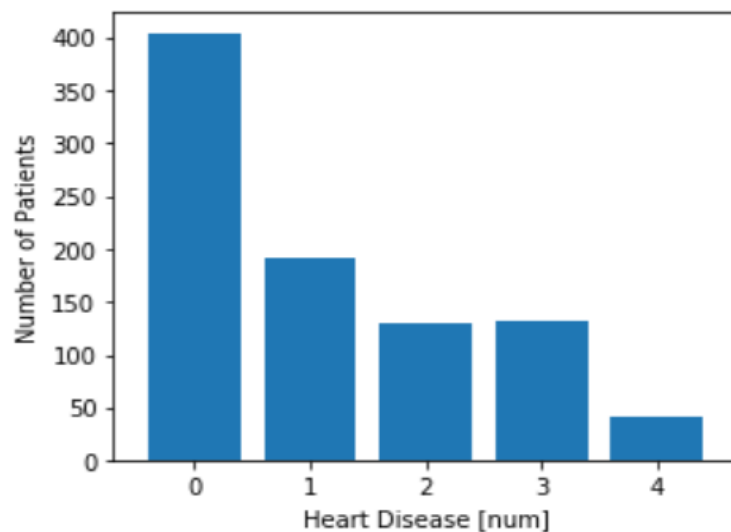
*Figure 23 major Vessels coloured by Fluoroscopy*

What this shows is that the majority of instances in the dataset have no major vessels that have been coloured and detected using fluoroscopy, this means that x-rays have not shown any issues with the major vessels.



*Figure 24 Thal Representing heart state*

As can be seen by the graph above the majority of the instances in the dataset consist of either normal or reversible defects. Then there are the data anomalies' showing in the 4 and 5 positions, this may be due to some inconsistent data.



*Figure 25 Num feature representing presence of disease*

This again shows that the majority of the dataset instances have no presence of heart disease. After plotting all the features there are still some inconsistencies in the results this may be due to the feature not being complete. After reading in the UCI processed datasets there are actually 920 records which is more than was left after processing the original data, this may be due to part of the Cleveland dataset being corrupted, please see the UCI LM repo for more info on this.

The pre-processed data has got zeros in place of the missing values, so it is plotting the same, the plots for this data set are in the appendix of this document also. The next set of diagrams display the comparison between disease and no-disease for the given dataset. They show the degree of obviousness, this is something that needs to be quantified. This can be done by calculating the intrinsic discrepancy between the occurrence of disease and non-disease instances for the given dataset. This is defined in a paper by “Jose M. Bernardo 1<sup>st</sup> “as follows.

*“Definition 1 (Intrinsic discrepancy) The intrinsic discrepancy  $\delta\{p_1, p_2\}$  between two probability distributions of a random vector  $x \in X$ , specified by their density functions  $p_1(x)$ ,  $x \in X_1 \subset X$ , and  $p_2(x)$ ,  $x \in X_2 \subset X$ , with either identical or nested supports, is”(51)*

$$\delta\{p_1, p_2\} = \min \left\{ \int_{\mathcal{X}_1} p_1(\mathbf{x}) \log \frac{p_1(\mathbf{x})}{p_2(\mathbf{x})} d\mathbf{x}, \int_{\mathcal{X}_2} p_2(\mathbf{x}) \log \frac{p_2(\mathbf{x})}{p_1(\mathbf{x})} d\mathbf{x} \right\}$$

*Figure 26 Intrinsic discrepancy Formula*

The author goes on then to state that this is provided at least one of the integrals are finite. The python code for this is covered in the next section of this report, below in figure 27 shows the calculated intrinsic discrepancies for each remaining feature in the dataset.

```
Intrinsic discrepancies between disease and no-disease for thal is: 0.633089
Intrinsic discrepancies between disease and no-disease for cp is: 0.596290
Intrinsic discrepancies between disease and no-disease for exang is: 0.382376
Intrinsic discrepancies between disease and no-disease for slope is: 0.325506
Intrinsic discrepancies between disease and no-disease for thalach is: 0.320208
Intrinsic discrepancies between disease and no-disease for oldpeak is: 0.319868
Intrinsic discrepancies between disease and no-disease for age is: 0.187274
Intrinsic discrepancies between disease and no-disease for chol is: 0.185248
Intrinsic discrepancies between disease and no-disease for sex is: 0.169414
Intrinsic discrepancies between disease and no-disease for ca is: 0.073099
Intrinsic discrepancies between disease and no-disease for restbp is: 0.040814
Intrinsic discrepancies between disease and no-disease for restecg is: 0.018863
Intrinsic discrepancies between disease and no-disease for fbs is: 0.016228
```

```
Number of patients in dataframe: 690, with disease: 381, without disease: 309
```

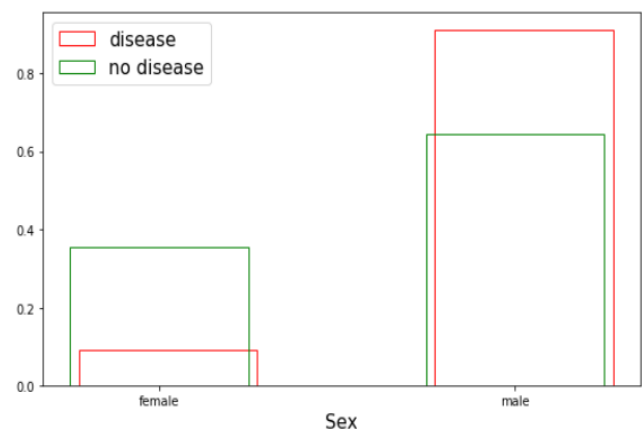
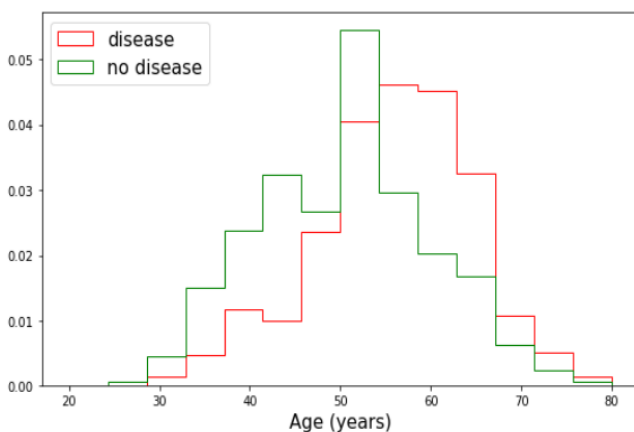
*Figure 27 Intrinsic Discrepancy Table for the Features*

These results provide an insight into which features would be best suited for inclusion in a ML model. They don’t take into account any correlations of the features. This is why they are better suited as a benchmark.

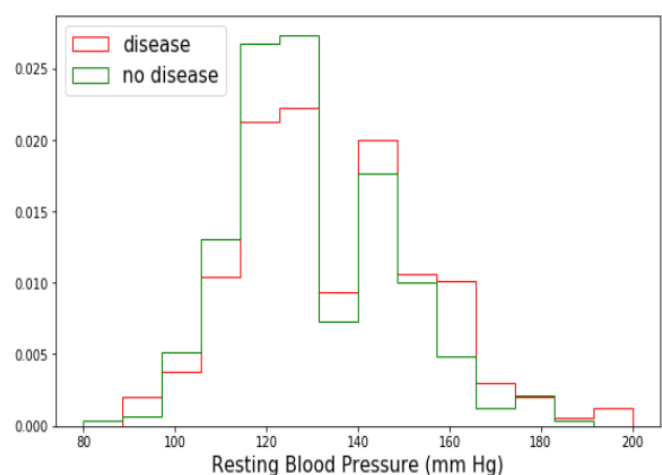
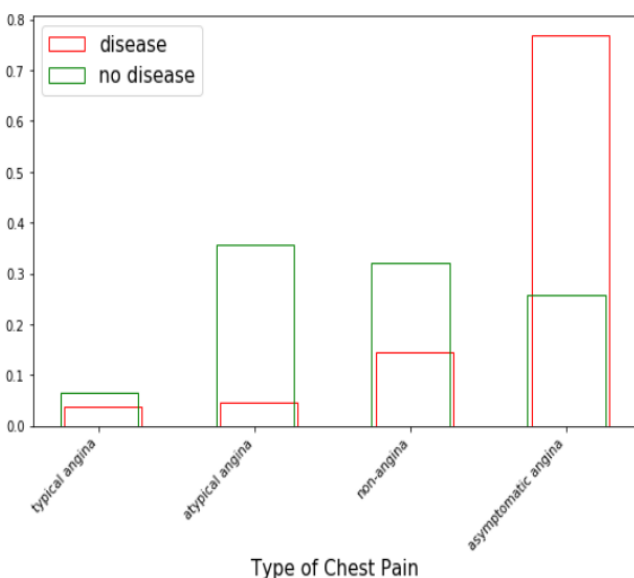


### 3.4 Dataset Plotted

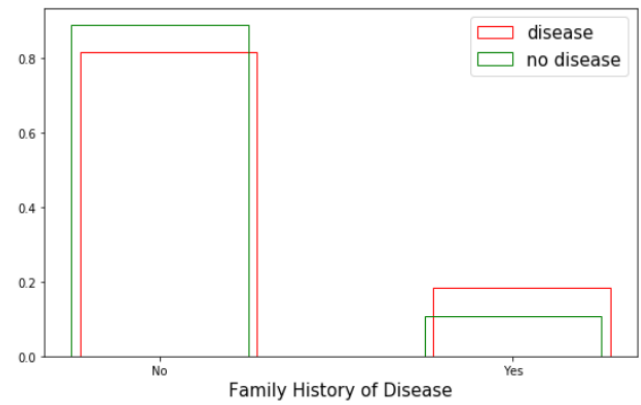
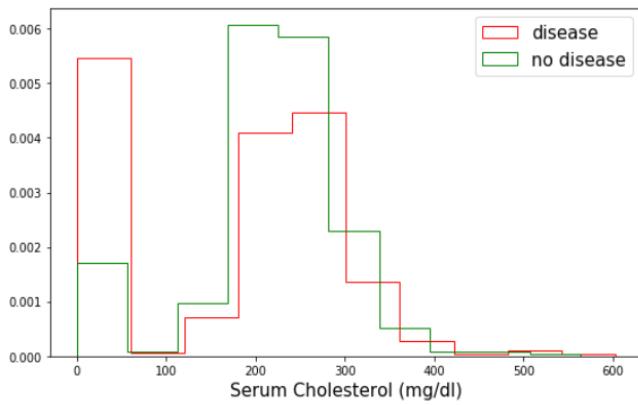
The following diagrams are of the entire dataset before there has been a split for the training and testing of the ML models. They have been plotted with respect to the intrinsic discrepancies listed above. When viewing the graphs it should be noted that the y axis represents the degree of obviousness for that that feature for disease and non-disease cases.



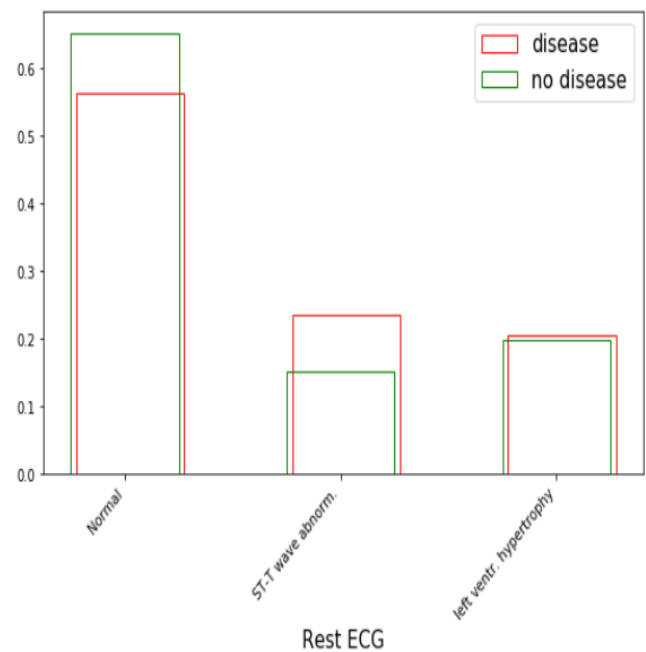
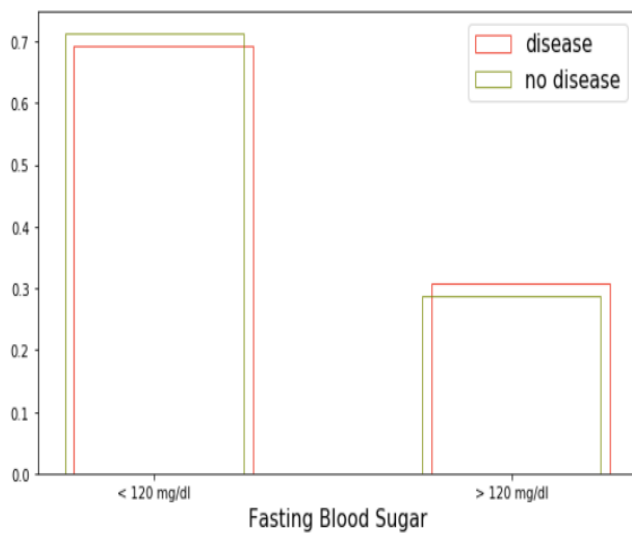
What the graphs above show is that there is a balanced split between disease and non-disease cases. It can also be seen that there are clearly far more male cases than there are female, with male having a far higher probability of having heart disease.



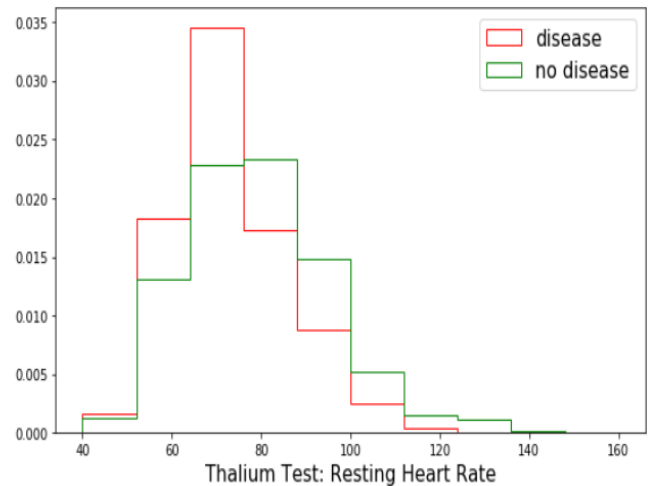
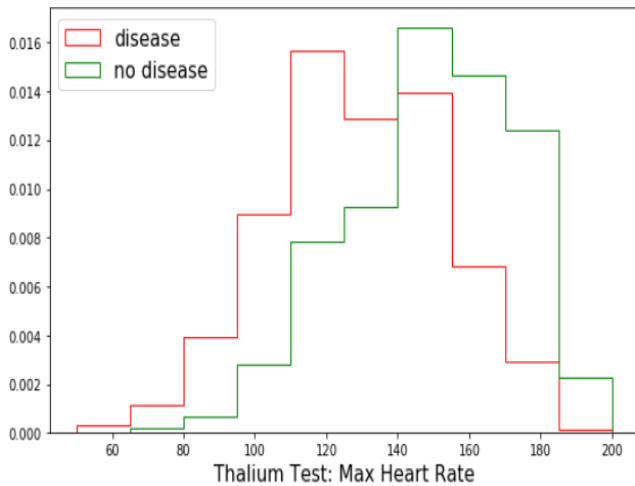
What can be inferred here is that it is asymptomatic angina pains that are the greatest concern when it comes to heart diseases, and that there tends to be lower blood pressure in cases where there is a presence of disease.



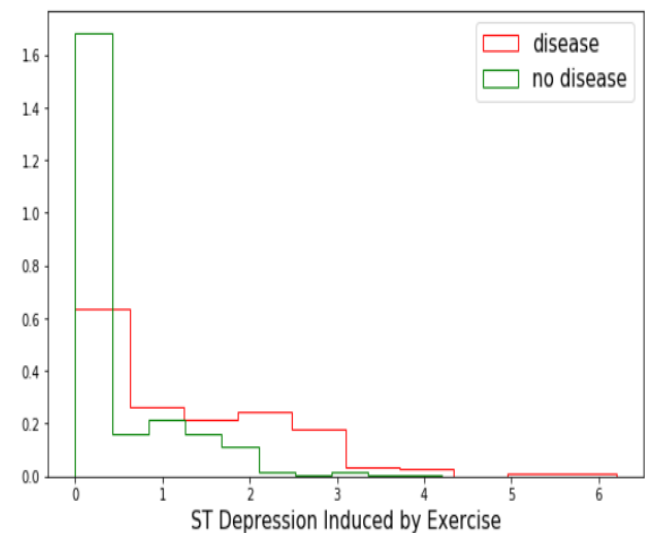
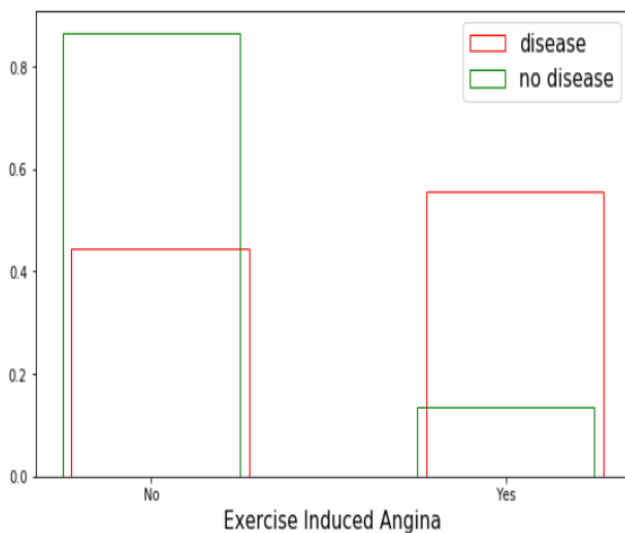
What this shows is there is there is a large number of cases where, family history is not present for the one where there is a presence of disease.



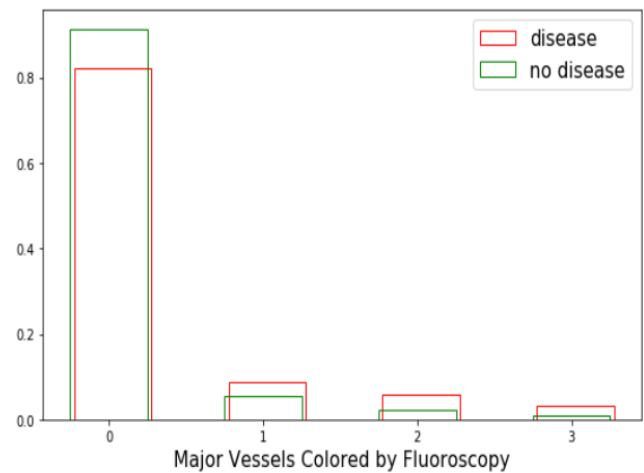
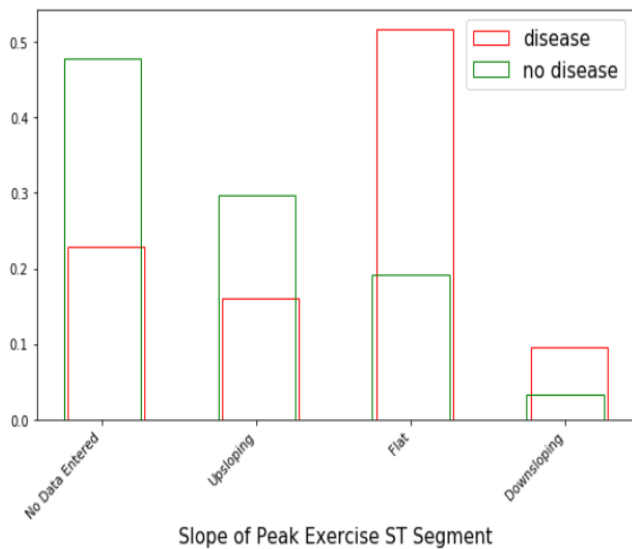
Fasting blood sugar appears very even for both disease and non-disease cases. This is showing that the data is evenly distributed for this feature. The same can be seen for the resting ECG, with the higher number of cases presenting as normal.



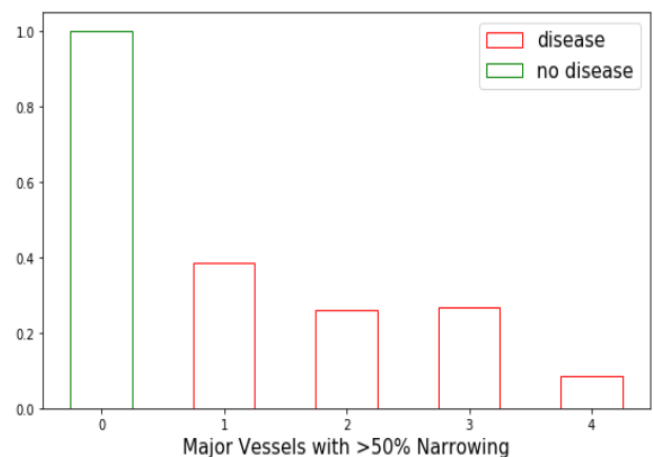
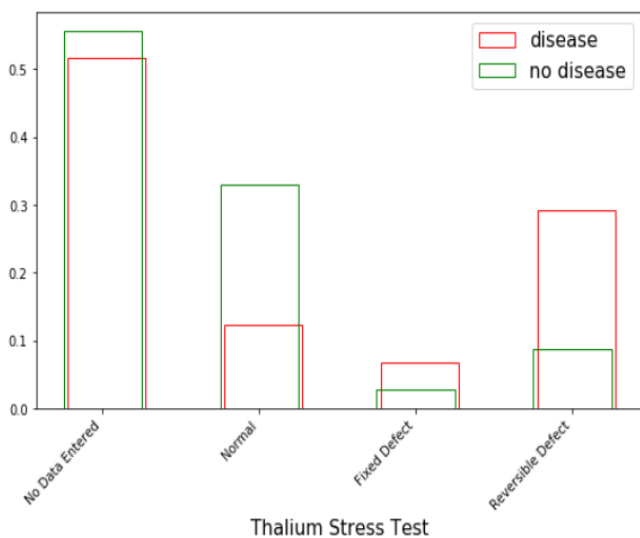
The interesting thing here is that there is a clear shift in the negative direction for the thallium stress test, with regard to maximum heart rate for disease cases. This would be an expected result as a healthy heart should achieve a higher rate during this test. The resting heart rate shows that there is also a shift in the negative direction for the feature with regard to disease cases. With most of the cases in the range of 50 to 75, while for non-disease it is 65 to 100.



Looking at the exercise induced angina graph it is clear that there more cases where this was not present. The ST depression induced by exercise has a massive peak at the zero point this should be ignored as it is a result of the data cleaning process.



What the slope graph shows us is there is the largest number of disease cases have a flat slope. Looking then at the vessels coloured by fluoroscopy, in both disease and non-disease most had none.



Looking at the data after plotting it's clear to see that there is visual representation of what could be described as the known facts with regard to heart disease. There are more cases of heart disease in males than there are in females. It also shows there is a spike of cases with disease in the age range of 45-65 and that in most cases there is no family history of heart disease.

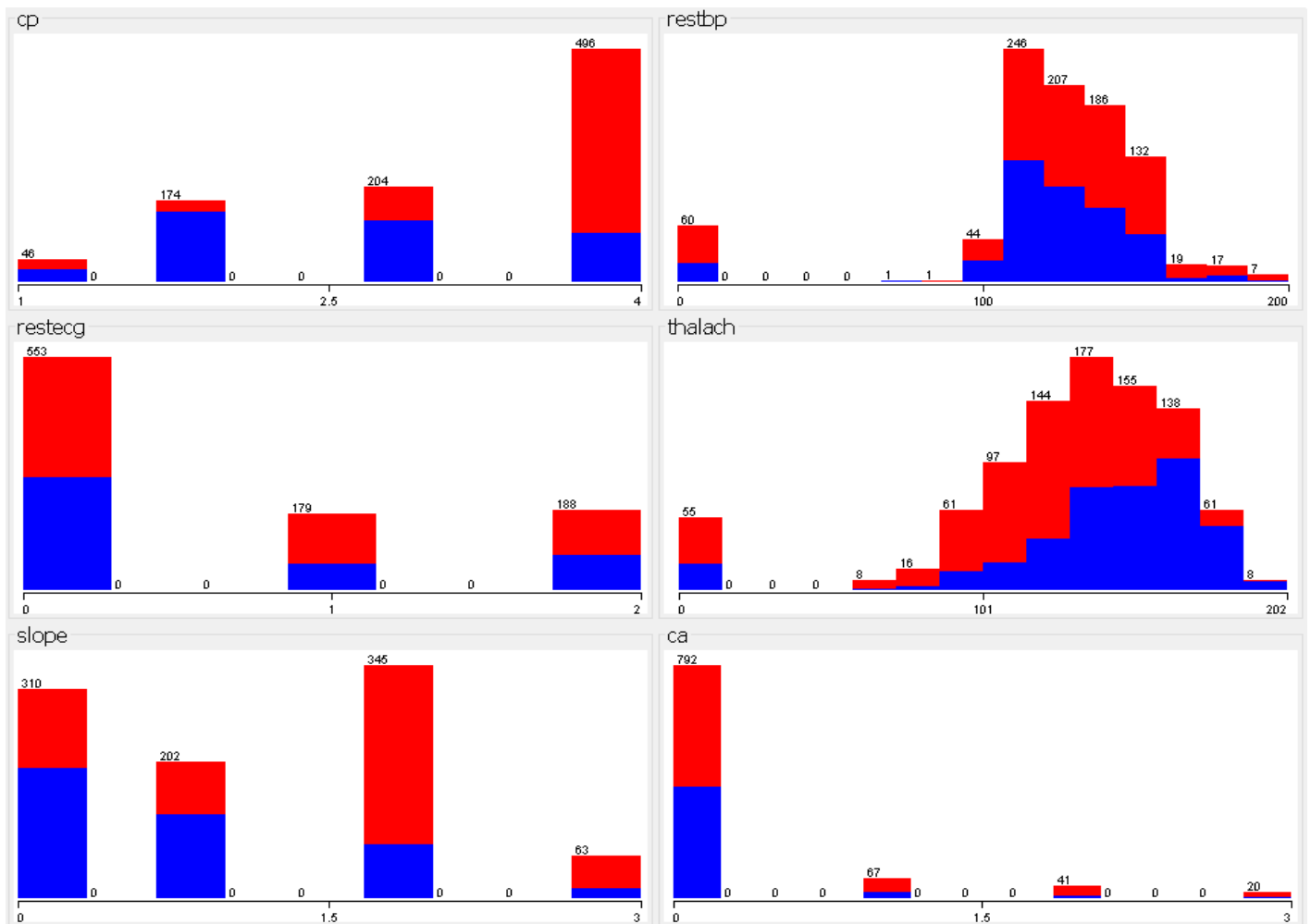
When this data was plotted it presented almost identical results the plots and data quality table can be found in appendix B of this document.

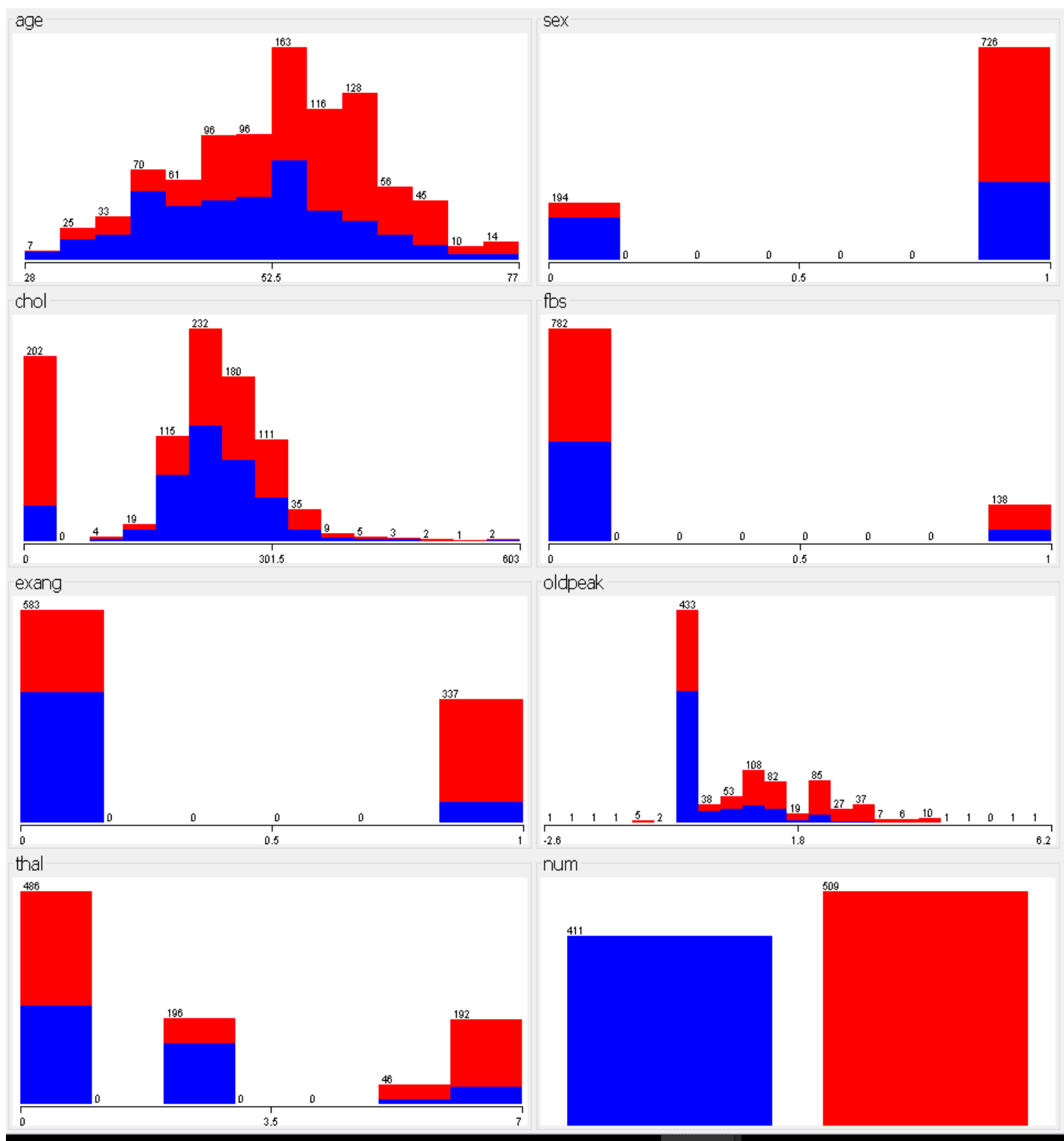
## 4 Testing with WEKA

Preparing the datasets for implementing any type of classifier in WEKA involves some alteration of the datasets. To do this a copy was first made. As there is only the need to know of either a presence of heart disease or not for this project, the (num) feature was changed. In place of a zero value, the string (tested\_negative) and any value that was greater than zero was replaced with the string (tested\_positive). After prepping the datasets they were plotted and as can be seen from the outputs for both datasets apart from the 2 extra features in the unprocessed dataset, they are almost identical to each other. This is apart from some outliers that are caused by missing values, this difference aside, they do in fact plot the same.

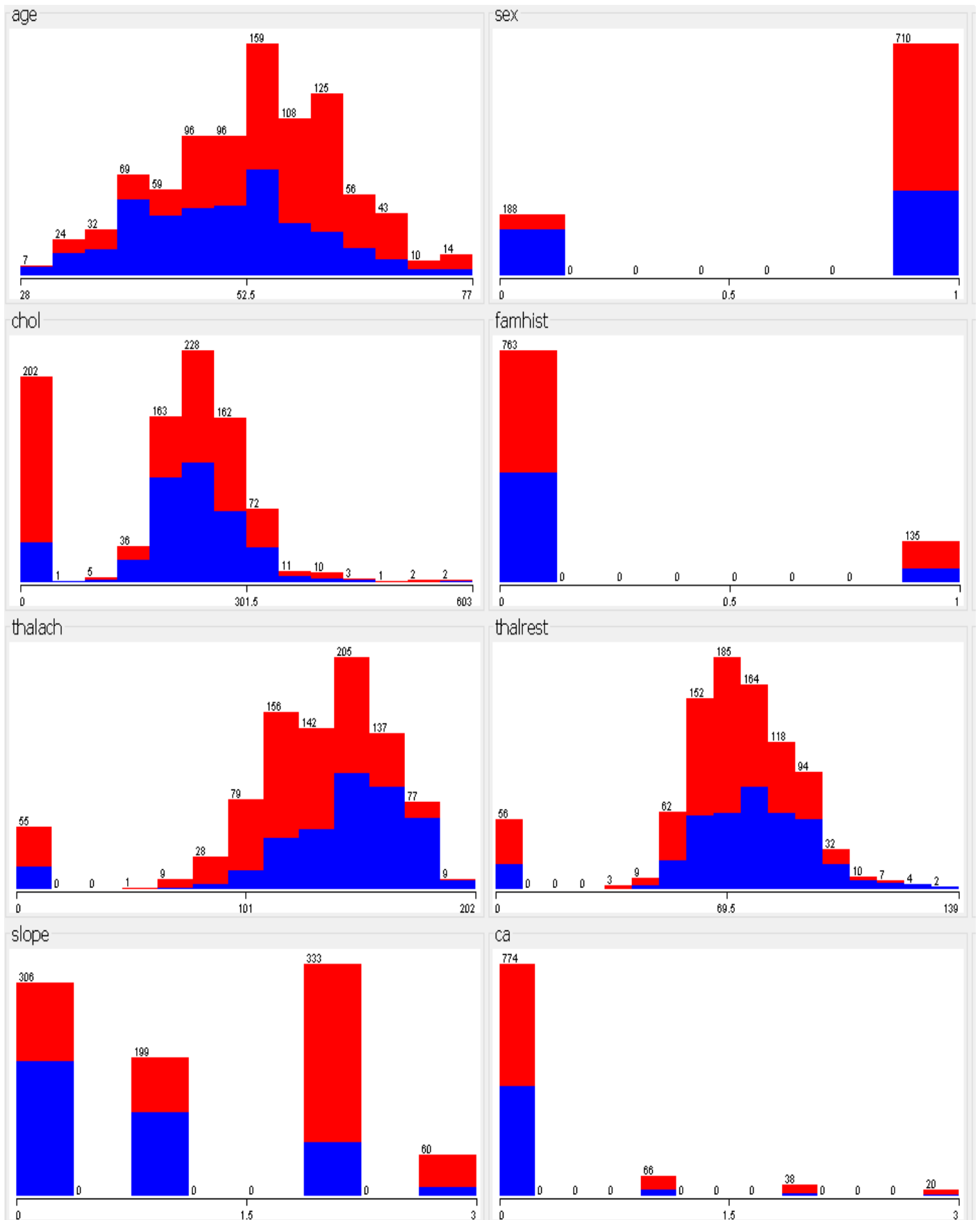
### 4.1 Data Plots from Weka

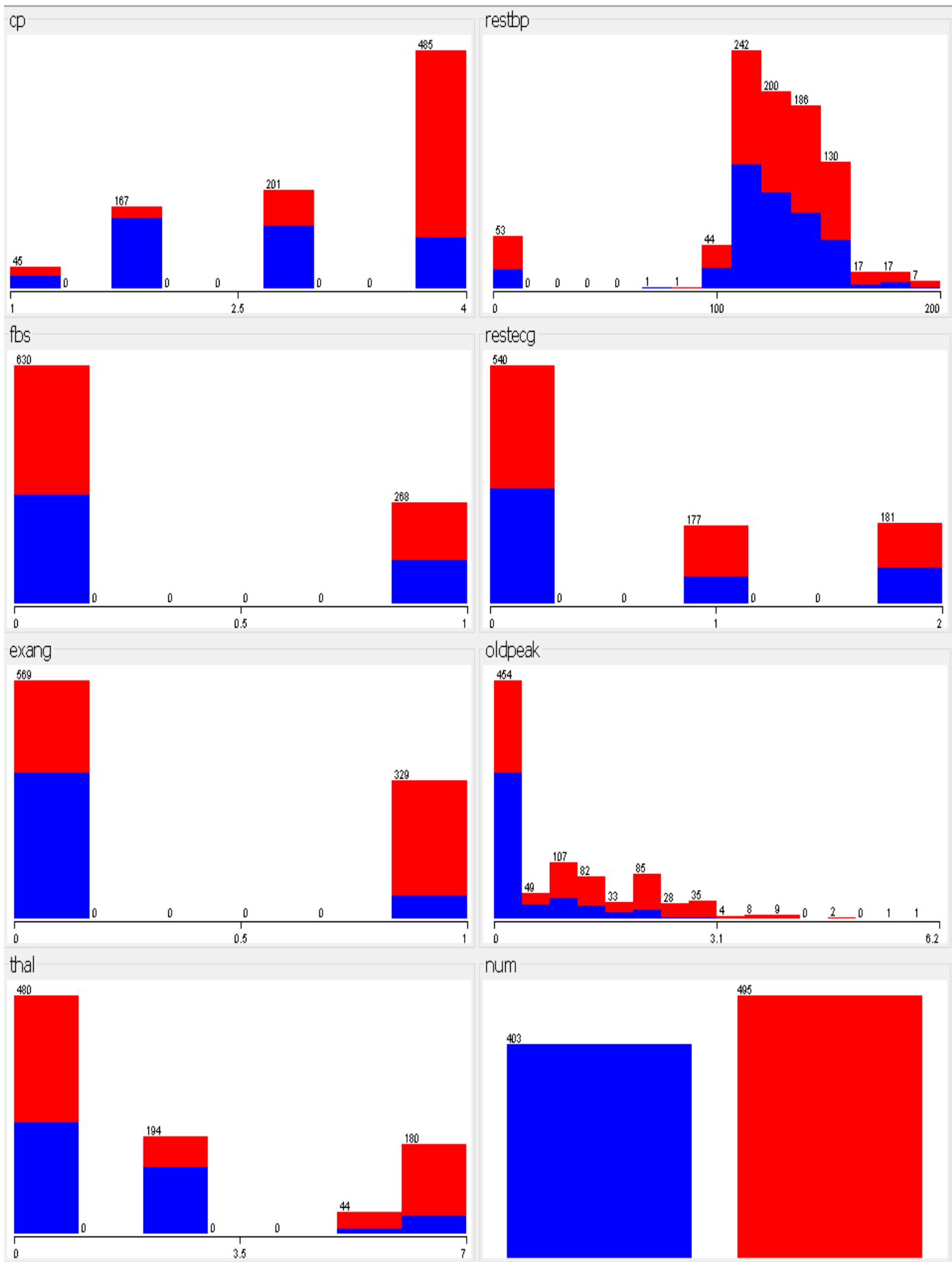
#### 4.1.1 Pre-processed Dataset Plots





## 4.1.2 Unprocessed Dataset Plots







With all the data plotted and there not being any real differences in the plots for them, it's time to see if there is any improvement in the accuracy of the output depending of the algorithm used for classification and on the two extra features in the unprocessed dataset.

The following section tests the various ML algorithms that could be used. This was to provide a better insight into the precision for the algorithms, while gaining a better understanding of how they differ in their operation, the results will be used to identify which algorithm is best suited..

Rather than using a testing and training split on the data here, N-fold was used. There are ten folds in the cross validation, this eliminates any bias in the training and testing of the model. Using N-fold with a selection of classification algorithms the outputs are explored below.

## 4.2 Logistic Regression Outputs

### 4.2.1 Unprocessed Dataset Outputs Using a 10 fold Cross Validation

```
=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances      729           81.1804 %
Incorrectly Classified Instances    169           18.8196 %
Kappa statistic                    0.6186
K&B Relative Info Score            43483.8792 %
K&B Information Score              431.4961 bits    0.4805 bits/instance
Class complexity | order 0         891.2105 bits    0.9924 bits/instance
Class complexity | scheme          562.284 bits    0.6262 bits/instance
Complexity improvement (Sf)        328.9265 bits    0.3663 bits/instance
Mean absolute error                 0.2692
Root mean squared error             0.3702
Relative absolute error             54.407 %
Root relative squared error         74.4347 %
Coverage of cases (0.95 level)     99.2205 %
Mean rel. region size (0.95 level) 91.2584 %
Total Number of Instances          898
```

*Figure 28 WEKA Logistic Regression Output for Unprocessed Dataset*

```
=== Detailed Accuracy By Class ===

      TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
      0.777    0.160    0.798    0.777    0.787      0.619    0.880    0.850    tested_negative
      0.840    0.223    0.822    0.840    0.831      0.619    0.880    0.892    tested_positive
Weighted Avg.  0.812    0.195    0.812    0.812    0.812      0.619    0.880    0.873
```

*Figure 29 Weka Logistic Regression Class Accuracy for Unprocessed Dataset*

```
=== Confusion Matrix ===

      a    b  <-- classified as
      313  90 |   a = tested_negative
      79 416 |   b = tested_positive
```

*Figure 30 WEKA Logistic Regression Confusion Matrix for Unprocessed Dataset*

What can be seen from the output above there is an accuracy of 81.1804% correct classification with a total of 729 cases out of 898 being correctly classified. A more detailed look at this result can be found by examining the confusion matrix for the model. It shows that there are in fact a total 79 cases where there is no presence of heart disease that have been incorrectly classified, meaning there are 79 cases that were told there is a presence of heart disease when there is not. With a further 90 cases where the presence of disease was not detected. This was tested in Weka using a ridge regression.

## 4.2.2 Pre-processed Dataset Outputs Using a 10 fold Cross Validation

```

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances      752           81.7391 %
Incorrectly Classified Instances    168           18.2609 %
Kappa statistic                     0.629
Mean absolute error                  0.2691
Root mean squared error              0.37
Relative absolute error              54.4437 %
Root relative squared error          74.4305 %
Total Number of Instances           920

=== Detailed Accuracy By Class ===

```

*Figure 31 WEKA Logistic Regression Output for Pre-processed Dataset*

```

=== Detailed Accuracy By Class ===

      TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
      0.774    0.147    0.809    0.774    0.791     0.630    0.880    0.848    tested_negative
      0.853    0.226    0.824    0.853    0.838     0.630    0.880    0.893    tested_positive
Weighted Avg.   0.817    0.191    0.817    0.817    0.817     0.630    0.880    0.873

```

*Figure 32 WEKA Logistic Regression Class Accuracy for Pre-processed Dataset*

```

=== Confusion Matrix ===

      a    b  <-- classified as
      318  93 |   a = tested_negative
      75 434 |   b = tested_positive

```

*Figure 33 WEKA Logistic Regression Confusion Matrix for Pre-processed Dataset*

Looking at the results for the above LR output on the pre-processed dataset, it shows a slight increase in the accuracy for the model. This may be due to there being some more instances in the dataset. Looking at the confusion matrix again it is clear that there is 93 cases that are incorrectly classified this time this is an increase on the unprocessed dataset output, however as there are 8 more cases in the pre-processed dataset it is best to look at this as a percentage, in this case 22.63% missed cases of heart disease for the pre-processed dataset verses a 22.33% for the unprocessed dataset. So while there is a slight increase in the accuracy of the overall prediction this is not the case for instances where the presence of heart disease was incorrectly classified, this has decreased slightly.

## 4.3 Random Forrest results

### 4.3.1 Unprocessed Dataset Outputs Using a 10 fold Cross Validation

```
=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances      733           81.6258 %
Incorrectly Classified Instances    165           18.3742 %
Kappa statistic                     0.6263
Mean absolute error                 0.2757
Root mean squared error            0.3659
Relative absolute error             55.7136 %
Root relative squared error        73.571 %
Total Number of Instances         898
```

*Figure 34 WEKA Random Forrest Output for Unprocessed Dataset*

```
=== Detailed Accuracy By Class ===

          TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
          0.762   0.139   0.816     0.762   0.788     0.627   0.887    0.865   tested_negative
          0.861   0.238   0.816     0.861   0.838     0.627   0.887    0.890   tested_positive
Weighted Avg.   0.816   0.194   0.816     0.816   0.816     0.627   0.887    0.879
```

*Figure 35 WEKA Random Forrest Class Accuracy for Unprocessed Dataset*

```
=== Confusion Matrix ===

      a    b  <-- classified as
307  96 |   a = tested_negative
 69 426 |   b = tested_positive
```

*Figure 36 WEKA Random Forrest Confusion Matrix for Unprocessed Dataset*

The output for the Random Forrest (RF) model with the unprocessed dataset shows an increase in the accuracy for the overall precision of the prediction. Again however there is an increase in the number of cases that have a presence of heart disease that are being incorrectly classified. The increase in accuracy for this model is only reflected in the unprocessed dataset.

### 4.3.2 Pre-processed Dataset Outputs Using a 10 fold Cross Validation

```

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances      751           81.6304 %
Incorrectly Classified Instances    169           18.3696 %
Kappa statistic                     0.6274
Mean absolute error                  0.2727
Root mean squared error              0.3664
Relative absolute error              55.1589 %
Root relative squared error          73.7079 %
Total Number of Instances          920

```

*Figure 37 WEKA Random Forrest Output for Pre-processed Dataset*

```

=== Detailed Accuracy By Class ===

```

	TP Rate	FP Rate	Precision	Recall	F-Measure	MCC	ROC Area	PRC Area	Class
	0.781	0.155	0.803	0.781	0.792	0.628	0.884	0.863	tested_negative
	0.845	0.219	0.827	0.845	0.836	0.628	0.884	0.886	tested_positive
Weighted Avg.	0.816	0.190	0.816	0.816	0.816	0.628	0.884	0.876	

*Figure 38 WEKA Random Forrest Class Accuracy for Pre-processed Dataset*

```

=== Confusion Matrix ===

      a    b  <-- classified as
321  90 |   a = tested_negative
 79 430 |   b = tested_positive

```

*Figure 39 WEKA Random Forrest Confusion Matrix for Pre-processed Dataset*

While the unprocessed dataset has increased in accuracy the pre-processed dataset has not, it has decreased, all be it only half a percent. The number of cases where there is a presence of heart disease that are being misclassified are still high.

## 4.4 Support Vector Machine

### 4.4.1 Unprocessed Dataset Outputs Using a 10 fold Cross Validation

```
=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances      726           80.8463 %
Incorrectly Classified Instances    172           19.1537 %
Kappa statistic                     0.6121
Mean absolute error                  0.1915
Root mean squared error              0.4376
Relative absolute error              38.7119 %
Root relative squared error          87.9909 %
Total Number of Instances          898
```

*Figure 40 WEKA Support Vector Machine Output for Unprocessed Dataset*

```
=== Detailed Accuracy By Class ===

      TP Rate  FP Rate  Precision  Recall  F-Measure  MCC      ROC Area  PRC Area  Class
      0.777   0.166   0.792   0.777   0.784   0.612   0.806   0.716   tested_negative
      0.834   0.223   0.821   0.834   0.828   0.612   0.806   0.776   tested_positive
Weighted Avg.   0.808   0.197   0.808   0.808   0.808   0.612   0.806   0.749
```

*Figure 41 WEKA Support Vector Machine Class Accuracy for Unprocessed Dataset*

```
=== Confusion Matrix ===

      a    b  <-- classified as
      313  90 |   a = tested_negative
      82 413 |   b = tested_positive
```

*Figure 42 WEKA Support Vector Machine Confusion Matrix for Unprocessed Dataset*

With the SVM model there is a decrease in accuracy overall when compared to both the LR and RF models. It does however have a very similar output to that of the for the confusion matrix for the other two models.

## 4.4.2 Pre-processed Dataset Outputs Using a 10 fold Cross Validation

```

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances      755           82.0652 %
Incorrectly Classified Instances    165           17.9348 %
Kappa statistic                     0.6366
Mean absolute error                  0.1793
Root mean squared error              0.4235
Relative absolute error              36.2801 %
Root relative squared error          85.1832 %
Total Number of Instances          920

```

*Figure 43 WEKA Support Vector Machine Output for Pre-processed Dataset*

```

=== Detailed Accuracy By Class ===

      TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
      0.791    0.155    0.804    0.791    0.798      0.637    0.818    0.730    tested_negative
      0.845    0.209    0.833    0.845    0.839      0.637    0.818    0.790    tested_positive
Weighted Avg.   0.821    0.185    0.820    0.821    0.820      0.637    0.818    0.763

=== Confusion Matrix ===

```

*Figure 44 WEKA Support Vector Machine Class Accuracy for Pre-processed Dataset*

```

=== Confusion Matrix ===

      a    b  <-- classified as
      325  86 |   a = tested_negative
      79 430 |   b = tested_positive

```

*Figure 45 WEKA Support Vector Machine Confusion Matrix for Pre-processed Dataset*

Looking at the output above again we can see there is very little in the increase in accuracy overall when compared to the other models. There is however an increase in the accuracy when only compared to the unprocessed dataset, this is most likely due to there being some more instances for the model to work with. It still amounts to an increase in accuracy of just over 1%, the may not be enough information alone to select a model for the project. The gained accuracy seems the have been gained in the number of cases being classified where there is a presence of heart disease, this may be a deciding factor for selection. This is accessed further on in this document after the feature selection stage.

## 4.5 K Nearest Neighbours

### 4.5.1 Unprocessed Dataset Outputs Using a 10 fold Cross Validation

```
=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances      717           79.8441 %
Incorrectly Classified Instances    181           20.1559 %
Kappa statistic                    0.5921
Mean absolute error                 0.2516
Root mean squared error            0.3887
Relative absolute error            50.846 %
Root relative squared error        78.1563 %
Total Number of Instances          898
```

*Figure 46 WEKA K Nearest Neighbours Output for Unprocessed dataset*

```
=== Detailed Accuracy By Class ===
```

	TP Rate	FP Rate	Precision	Recall	F-Measure	MCC	ROC Area	PRC Area	Class
	0.769	0.178	0.779	0.769	0.774	0.592	0.858	0.805	tested_negative
	0.822	0.231	0.814	0.822	0.818	0.592	0.858	0.843	tested_positive
Weighted Avg.	0.798	0.207	0.798	0.798	0.798	0.592	0.858	0.826	

*Figure 47 WEKA K Nearest Neighbours Class Accuracy for Unprocessed dataset*

```
=== Confusion Matrix ===

      a    b   <-- classified as
310  93 |   a = tested_negative
 88 407 |   b = tested_positive
```

*Figure 48 WEKA K Nearest Neighbours Confusion Matrix for Unprocessed dataset*

As can be seen again there is no improvement on the accuracy, it is in fact under 80% for the unprocessed dataset. This was tested with a number of different nearest neighbours settings, using 1, 3 and 5, nearest neighbours returned very similar results. With 5 nearest neighbours or the above output was the best accuracy by between 1% and 2%.



## 4.5.2 Pre-processed Dataset Outputs Using a 10 fold Cross Validation

```

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances      739           80.3261 %
Incorrectly Classified Instances    181           19.6739 %
Kappa statistic                     0.6015
Mean absolute error                  0.2485
Root mean squared error              0.388
Relative absolute error              50.2596 %
Root relative squared error          78.0532 %
Total Number of Instances          920

```

*Figure 49 WEKA K Nearest Neighbours Output for Pre-processed Dataset*

```

=== Detailed Accuracy By Class ===

      TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
      0.774    0.173    0.783    0.774    0.778      0.602    0.859    0.805    tested_negative
      0.827    0.226    0.819    0.827    0.823      0.602    0.859    0.843    tested_positive
Weighted Avg.   0.803    0.202    0.803    0.803    0.803      0.602    0.859    0.826

```

*Figure 50 WEKA K Nearest Neighbours Class Accuracy for Pre-processed Dataset*

```

=== Confusion Matrix ===

      a    b  <-- classified as
    318  93 |   a = tested_negative
      88 421 |   b = tested_positive

```

*Figure 51 WEKA K Nearest Neighbours Confusion Matrix for Pre-processed Dataset*

The pre-processed dataset is also showing no increase for the accuracy, however it is slightly better than the results for the unprocessed dataset. This could again be down to the fact that there are a few more instances in the dataset.

## 4.6 Linear Discriminant Analysis

### 4.6.1 Unprocessed Dataset Outputs Using a 10 fold Cross Validation

```
=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances      728           81.069 %
Incorrectly Classified Instances    170           18.931 %
Kappa statistic                     0.6165
Mean absolute error                  0.2548
Root mean squared error              0.3712
Relative absolute error              51.4916 %
Root relative squared error          74.6371 %
Total Number of Instances          898
```

*Figure 52 WEKA Linear Discriminant Analysis Output for Unprocessed Dataset*

```
=== Detailed Accuracy By Class ===

      TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
      0.777   0.162   0.796    0.777   0.786     0.617   0.881    0.848    tested_negative
      0.838   0.223   0.822    0.838   0.830     0.617   0.881    0.893    tested_positive
Weighted Avg.  0.811   0.196   0.810    0.811   0.810     0.617   0.881    0.873
```

*Figure 53 WEKA Linear Discriminant Analysis Class Accuracy for Unprocessed Dataset*

```
=== Confusion Matrix ===

      a    b  <-- classified as
      313  90 |   a = tested_negative
      80 415 |   b = tested_positive
```

*Figure 54 WEKA Linear Discriminant Analysis Confusion Matrix for Unprocessed Dataset*

There is again no advance beyond the 82% accuracy when compared to other models the Linear Discriminant Analysis (LDA) for the unprocessed dataset is again showing high misclassification rate.

## 4.6.2 Pre-processed Dataset Outputs Using a 10 fold Cross Validation

```

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances      749           81.413 %
Incorrectly Classified Instances    171           18.587 %
Kappa statistic                     0.625
Mean absolute error                  0.2136
Root mean squared error              0.3842
Relative absolute error              43.2165 %
Root relative squared error          77.2856 %
Total Number of Instances           920

```

*Figure 55 WEKA Linear Discriminant Analysis Output for Pre-processed Dataset*

```

=== Detailed Accuracy By Class ===

      TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
      0.805    0.179    0.784    0.805    0.795     0.625    0.876    0.848    tested_negative
      0.821    0.195    0.839    0.821    0.830     0.625    0.876    0.876    tested_positive
Weighted Avg.   0.814    0.188    0.815    0.814    0.814     0.625    0.876    0.864

```

*Figure 56 WEKA Linear Discriminant Analysis Class Accuracy for Pre-processed Dataset*

```

=== Confusion Matrix ===

      a    b  <-- classified as
    331  80 |   a = tested_negative
      91 418 |   b = tested_positive

```

*Figure 57 WEKA Linear Discriminant Analysis Confusion Matrix for Pre-processed Dataset*

Looking at the outputs for both the unprocessed and pre-processed datasets for the LDA model, there is like a switch in the incorrectly classified cases. The cases that have a presence of heart disease that were incorrectly classified in the unprocessed dataset was 90, while the number of cases where there was no presence of heart disease that were incorrectly classified was 80. Now looking to the confusion matrix above it's clear that there are only 80 cases in the pre-processed dataset that have a present in heart disease that were incorrectly classified, this is down when compared to the unprocessed dataset. Looking then to the cases that had no presence of heart disease that were incorrectly classified has increased to a total of 91.

## 4.7 Naive Bayes

### 4.7.1 Unprocessed Dataset Outputs Using a 10 fold Cross Validation

```
=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances      725           80.735 %
Incorrectly Classified Instances    173           19.265 %
Kappa statistic                    0.6107
Mean absolute error                 0.2177
Root mean squared error            0.3924
Relative absolute error            44.0029 %
Root relative squared error        78.9037 %
Total Number of Instances         898

- . . . - . .
```

*Figure 58 WEKA Naive Bayes Output for Unprocessed Dataset*

```
=== Detailed Accuracy By Class ===
```

	TP Rate	FP Rate	Precision	Recall	F-Measure	MCC	ROC Area	PRC Area	Class
	0.787	0.176	0.785	0.787	0.786	0.611	0.870	0.846	tested_negative
	0.824	0.213	0.826	0.824	0.825	0.611	0.870	0.868	tested_positive
Weighted Avg.	0.807	0.197	0.807	0.807	0.807	0.611	0.870	0.858	

*Figure 59 WEKA Naive Bayes Class Accuracy for Unprocessed Dataset*

```
=== Confusion Matrix ===

      a   b   <-- classified as
317  86 |   a = tested_negative
 87 408 |   b = tested_positive
```

*Figure 60 WEKA Naive Bayes Confusion Matrix for Unprocessed Dataset*

Naive Bayes (NB) again is sitting in the middle of accuracy results when compared to the previous models outputs. For the unprocessed dataset it is showing a far more balanced misclassification rate, this can be seen by looking at the confusion matrix above. It is clear to see that the misclassification rate, while increasing where there is no presence of heart disease has decreased for the classification where there is a presence.

## 4.7.2 Pre-processed Dataset Outputs Using a 10 fold Cross Validation

```

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances      749           81.413 %
Incorrectly Classified Instances    171           18.587 %
Kappa statistic                    0.625
Mean absolute error                 0.2136
Root mean squared error             0.3842
Relative absolute error             43.2165 %
Root relative squared error         77.2856 %
Total Number of Instances          920

```

*Figure 61 WEKA Naive Bayes Output for Pre-processed Dataset*

```

=== Detailed Accuracy By Class ===

```

	TP Rate	FP Rate	Precision	Recall	F-Measure	MCC	ROC Area	PRC Area	Class
	0.805	0.179	0.784	0.805	0.795	0.625	0.876	0.848	tested_negative
	0.821	0.195	0.839	0.821	0.830	0.625	0.876	0.876	tested_positive
Weighted Avg.	0.814	0.188	0.815	0.814	0.814	0.625	0.876	0.864	

*Figure 62 WEKA Naive Bayes Class Accuracy for Pre-processed Dataset*

```

=== Confusion Matrix ===

      a    b   <-- classified as
331  80 |   a = tested_negative
 91 418 |   b = tested_positive

```

*Figure 63 WEKA Naive Bayes Confusion Matrix for Pre-processed Dataset*

The balance in the incorrectly classified cases for the unprocessed dataset is not present in the pre-processed dataset, instead the output for NB for the pre-processed dataset bears more resemblance to that of the LDA and RF model outputs only minor variation in the overall accuracy for the prediction.

## 4.8 WEKA Test Findings

After testing both datasets there is no real significant difference in the overall prediction accuracy, there are some cases where the model showed a slight increase in accuracy. This, in most cases, was over shadowed by the number of incorrectly classified cases where there was a presence of heart disease.

For this project that may be the deciding factor as the system aims to identify cases that would have a higher probability of having heart disease. But while this is only one factor that needs consideration, there is the factor of which has the least deviance in the results from the pre-processed and unprocessed datasets. This then highlights the LR model as that has the smallest distance in the classification errors for both datasets.

What this has shown is just how powerful WEKA can be, it also provide an in depth look into the output for some of the models that could be used for a project with ML. The selected model for this project will be the LR model, this will be implemented in python using Scikit-Learn to see if the accuracy can be increased using only the best selected features.

As there is no real difference in the outputs for either dataset the next section of this document covers the data quality report for both datasets. After the research conducted at this stage of the project the unprocessed dataset is going to be dropped as there is not a significant enough difference in the outputs to retain both.

For the reason that it has some more instances in the dataset the pre-processed dataset is going to be retained. This and the fact that the figures when closely examined, show that it has slightly more accurate outputs for most of the models.

While the SVM model was the most accurate during the testing phase it had some inconsistencies in the outputs. The LR model while being half a percent less accurate than the SVM, it did produce a more consistent output regardless of the amount of data used for training the model.

## 5 Scikit-Learn Outputs

### 5.1 Logistic Regression Outputs

Now the data has been plotted there are a number of ML models that can be applied to the dataset in order to make a prediction for the presence of heart disease, for this project not all will be explored.

The complete dataset was split into a training and testing set, with a 75/25 split. This is done so that the testing data is new to the model and will provide a less biased result on comparison with training and testing the model with the same data.

There is a function in the Scikit-learn for doing a training and testing split of data, using this method there is no need to split the dataset before loading the data into a data frame as the function will read in the complete dataset and split it before fitting the model. For testing purposes the dataset will be split using a script and the subsets of the dataset will all be tested as too will the Scikit-Learn train, test split function.

Having randomly split the dataset into the 75/25 training and testing files using a python script, the training data was loaded into an LR model using a Lasso regression, there is a brief overview of what Lasso Regression is below.

#### 5.1.1 Lasso Regression

Lasso regression is a form of LR that uses shrinkage, this is where data values are shrunk towards the mean. “LASSO” or Least Absolute Shrinkage and Selection Operator, this procedure promotes simple models. This type of regression is well-suited for models where you want to automate certain parts of model selection, such as variable selection or parameter elimination(52).

L1 Regularization Lasso regression performs L1 regularization, which adds a penalty equal to the absolute value of the magnitude of the coefficients. This can result in sparse models, with very few coefficients. It can cause other strange things to happen also, some coefficients may become zero and then get eliminated from the model. Greater penalties will result in coefficient values closer to zero. Then there is L2 regularization, this doesn't result in elimination of coefficients or sparse models. Instead it makes the

Lasso far easier to interpret than Ridge(52). This is a part of the logistic regression process. The next section covers the outputs of a logistic regression model using Scikit-Learn and statsmodels. In Scikit-learn it is possible to specify a  $\text{penalty} = l1$  or  $\text{penalty} = l2$  with a corresponding  $C=c$  value. This is the inverse of the regularization strength.

```
LogisticRegression coefficients:
  age : 0.13917
  sex : 1.11117
 restbp : 0.24926
  chol : -0.42199
  fbs : 0.34453
 thalach : -0.59222
  exang : 1.14064
 oldpeak : 0.45623
   ca : 3.22337
  cp1 : -1.34303
  cp2 : -1.98953
  cp3 : -1.35932
 recg1 : -0.12491
 recg2 : -0.38506
 slope1 : -0.74014
 slope3 : -0.65820
  thal6 : 0.64047
  thal7 : 1.11493
Intercept : -0.170862
```

*Figure 64 Logistic Regression coefficients for the features*

The coefficients from the output are far from intuitive to most people, that is why they are sometimes change to odds ratios or vice versa. This is possible as the log of the odds ratio are the coefficient, meaning it is also true that the exponential of the coefficients are the odds ratios(53).

Looking at the figure below it is very clear what the resulting columns indicate, this is due to them being represented as log odds. To understand the output, there is a need to understand the difference between odds, log odds and proportions. It is most often that binary outcomes are looked at in the terms of proportions, such as a percentage of correctness in a given condition.

While LR is actually based on the concept of [log] odds. This is calculated from a proportion using the logit function. This then means that odds can be interpreted as the proportion for some target outcome. With this outcome expressed in terms of how many non-target outcomes there are per target outcome.



What this means that given a proportion of 50%, also seen as an accuracy. It is this accuracy that corresponds directly to an odds ratio of exactly 1|1, this is true as there is a correct instance for every incorrect one. Considering a less straight forward proportion, say 33%. This will yield an odds ratio of 1|2, this is expected as there as there is there is in fact going to be 2 incorrect outputs for every 1 correct one(54). While log odds can difficult to interpret on their own, they can be converted to normal odds using the exponential function. While logistic regression coefficients are sometimes reported this way, the coefficients are mainly used as parameters for formulas that give predicted probabilities(55).

```

Optimization terminated successfully.
      Current function value: 0.392943
      Iterations 7

                        Logit Regression Results
=====
Dep. Variable:          HD      No. Observations:          690
Model:                 Logit    Df Residuals:              671
Method:                 MLE      Df Model:                18
Date:                  Thu, 15 Mar 2018    Pseudo R-squ.:          0.4286
Time:                  22:43:01    Log-Likelihood:         -271.13
converged:              True      LL-Null:                 -474.51
                               LLR p-value:          3.568e-75
=====

```

	coef	std err	z	P> z	[0.025	0.975]
age	0.1391	0.133	1.050	0.294	-0.121	0.399
sex	1.1126	0.294	3.782	0.000	0.536	1.689
restbtp	0.2493	0.157	1.592	0.111	-0.058	0.556
chol	-0.4220	0.116	-3.653	0.000	-0.648	-0.196
fbs	0.3447	0.318	1.085	0.278	-0.278	0.967
thalach	-0.5923	0.174	-3.396	0.001	-0.934	-0.250
exang	1.1412	0.264	4.325	0.000	0.624	1.658
oldpeak	0.4562	0.137	3.339	0.001	0.188	0.724
ca	3.2238	0.791	4.073	0.000	1.673	4.775
cp1	-1.3429	0.445	-3.016	0.003	-2.216	-0.470
cp2	-1.9892	0.326	-6.100	0.000	-2.628	-1.350
cp3	-1.3592	0.266	-5.109	0.000	-1.881	-0.838
recg1	-0.1249	0.293	-0.426	0.670	-0.699	0.449
recg2	-0.3849	0.318	-1.212	0.226	-1.008	0.238
slope1	-0.7398	0.300	-2.469	0.014	-1.327	-0.153
slope3	-0.6586	0.509	-1.293	0.196	-1.657	0.339
thal6	0.6409	0.556	1.152	0.249	-0.449	1.731
thal7	1.1151	0.310	3.593	0.000	0.507	1.723
intercept	-0.1726	0.322	-0.536	0.592	-0.804	0.459

```

=====

```

*Figure 65 Fitting results for the model*

This too is a result of for logistic regression using Lasso regression, this was done using the statsmodels API for Python(56), although it provides the same information as the Scikit-Learn module it can be harder to interpret the results, this is due to the output displaying the results in terms of log odds, this was explained above.

## 5.1.2 Training Dataset Outputs

```
Intrinsic discrepancies between disease and no-disease for thal is: 0.633089
Intrinsic discrepancies between disease and no-disease for cp is: 0.596290
Intrinsic discrepancies between disease and no-disease for exang is: 0.382376
Intrinsic discrepancies between disease and no-disease for slope is: 0.325506
Intrinsic discrepancies between disease and no-disease for thalach is: 0.320208
Intrinsic discrepancies between disease and no-disease for oldpeak is: 0.319868
Intrinsic discrepancies between disease and no-disease for age is: 0.187274
Intrinsic discrepancies between disease and no-disease for chol is: 0.185248
Intrinsic discrepancies between disease and no-disease for sex is: 0.169414
Intrinsic discrepancies between disease and no-disease for ca is: 0.073099
Intrinsic discrepancies between disease and no-disease for restbp is: 0.040814
Intrinsic discrepancies between disease and no-disease for restecg is: 0.018863
Intrinsic discrepancies between disease and no-disease for fbs is: 0.016228
```

Number of patients in training set: 690, with disease: 381, without disease: 309

*Figure 66 Intrinsic Discrepancies Output for Training Dataset*

```
LogisticRegression(C=1000.0, class_weight=None, dual=False,
    fit_intercept=True, intercept_scaling=1, max_iter=100,
    multi_class='ovr', n_jobs=1, penalty='l1', random_state=None,
    solver='liblinear', tol=0.0001, verbose=0, warm_start=False)
```

LogisticRegression score on the training data set: 0.823188

*Figure 67 Logistic Regression Output Score for Training Dataset using Lasso*

```
Classification report on training data set:
              precision    recall  f1-score   support

     0       0.81         0.79         0.80         309
     1       0.83         0.85         0.84         381

 avg / total         0.82         0.82         0.82         690
```

*Figure 68 Classification Report for Training Dataset using Lasso*

```
Confusion matrix:
[[244  65]
 [ 57 324]]
```

*Figure 69 Confusion Matrix for Training Dataset using Lasso*

This shows that there is a miss classification rate as there are 65 false positives and 57 false negatives, this is why the score is .823 and not 1. The output is as expected in the 80% area, this would be in line with the outputs from the testing in Weka.

```

Initial -log(L)=271.13042; fit method = newton

Dropping      recg1, -log(L)=271.22128...
Dropping intercept, -log(L)=271.42583...
Dropping      fbs, -log(L)=271.94468...
Dropping      recg2, -log(L)=272.67042...
Dropping      age, -log(L)=273.27281...
Dropping      thal6, -log(L)=274.02606...
Dropping      slope3, -log(L)=274.97959...

Final selection includes 12 features:

(<statsmodels.discrete.discrete_model.BinaryResultsWrapper at 0x1ee4a6074e0>,
 [2, 3, 4, 6, 7, 8, 9, 10, 11, 12, 15, 18])

```

*Figure 70 Output for Automatic Feature Selection on Training Dataset*

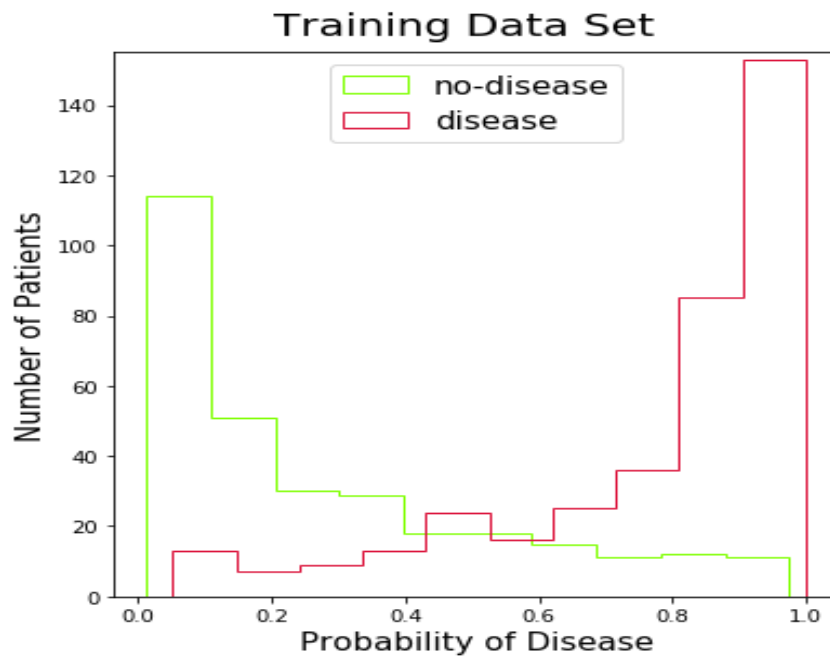
This is the output for the function to select features for a logistic regression model. There are now in fact 18 features as the categorical variables that had more than 2 outcome. Needed to be extracted and split into as many new features as there were categories in the original feature. For example the feature thal, it had 3 possible outcomes. This then means that there are in fact thal, thal3, thal6 and thal7, meaning that from that one feature, a further 3 features were extracted, the original feature can be now dropped as it is now represented by 3 new features that combined account for all instances of the original feature. The working for this function will be explained further in the development section of this document.

What it does at a high level is it starts by including all features in the dataset, and then it works through and eliminates any features deemed to be insignificant one-by-one. It does this in such a way that it minimises the increase in deviance after each elimination.

This method of feature selection has determined that the best combination of features for training a model, while listed above this output needs to be considered in array style with the first element being the zero element, this means the above output needs to be rolled back by 1 for each output variable.

Therefore the features that have been selected are as follows: (HD, sex, restbp, chol, thalach, exang, oldpeak, ca, cp1, cp2, cp3, slope1 and thal7). This combination of features were loaded into an LR model, producing very similar results to the previous implementation of the LR model.

Number of disease cases: 381, no-disease cases: 309



*Figure 71 Training Data Probability Plot*

As can be seen from the graph above there is a fairly balanced distribution of instances for either predicted outcome. This is the predicted probability for the above listed features.

As can be seen here there are 381 cases in the dataset that have a presence of heart disease and 309 where there is no presence. This dataset represents 3/4 of the complete dataset.

### 5.1.3 Testing Dataset Output

```
Intrinsic discrepancies between disease and no-disease for cp is: 0.730646
Intrinsic discrepancies between disease and no-disease for exang is: 0.393862
Intrinsic discrepancies between disease and no-disease for thal is: 0.349541
Intrinsic discrepancies between disease and no-disease for thalach is: 0.307208
Intrinsic discrepancies between disease and no-disease for age is: 0.269510
Intrinsic discrepancies between disease and no-disease for slope is: 0.210215
Intrinsic discrepancies between disease and no-disease for chol is: 0.198709
Intrinsic discrepancies between disease and no-disease for oldpeak is: 0.181328
Intrinsic discrepancies between disease and no-disease for sex is: 0.177069
Intrinsic discrepancies between disease and no-disease for restbp is: 0.074501
Intrinsic discrepancies between disease and no-disease for fbs is: 0.049228
Intrinsic discrepancies between disease and no-disease for restecg is: 0.045428
Intrinsic discrepancies between disease and no-disease for ca is: -0.034749

Number of patients in dataframe: 230, with disease: 128, without disease: 102
```

*Figure 72 Intrinsic Discrepancies Output for Testing Dataset*

```
LogisticRegression(C=1000.0, class_weight=None, dual=False,
    fit_intercept=True, intercept_scaling=1, max_iter=100,
    multi_class='ovr', n_jobs=1, penalty='l1', random_state=None,
    solver='liblinear', tol=0.0001, verbose=0, warm_start=False)

LogisticRegression score on the testing data set: 0.852174
```

*Figure 73 LR using Lasso for Testing Dataset*

```
Classification report on testing data set:
              precision    recall  f1-score   support

    0.0         0.85      0.78      0.82         102
    1.0         0.84      0.89      0.86         128

 avg / total         0.84      0.84      0.84         230
```

*Figure 74 Classification Report for Testing Dataset using Lasso*

```
Confusion matrix:
[[ 81  21]
 [ 13 115]]
```

*Figure 75 LR using lasso Testing Dataset Confusion Matrix*

What can be seen here from the output from the testing set that consisted of 1/4 of the complete dataset, the overall accuracy has increased by almost 3%. While this is a significant increase it is still around the expected average for the model.

```

Initial -log(L)=80.84669; fit method = newton

Dropping    slope3, -log(L)=80.89402...
Dropping    ca, -log(L)=80.96924...
Dropping    thal6, -log(L)=81.09537...
Dropping    thal7, -log(L)=81.24160...
Dropping    thalach, -log(L)=81.48830...
Dropping    age, -log(L)=81.63424...
Dropping    slope1, -log(L)=82.05206...
Dropping    recg2, -log(L)=82.62124...
Dropping intercept, -log(L)=83.07360...
Dropping    fbs, -log(L)=83.78602...
Dropping    restbp, -log(L)=85.29299...
Dropping    exang, -log(L)=86.35216...

Final selection includes 7 features:

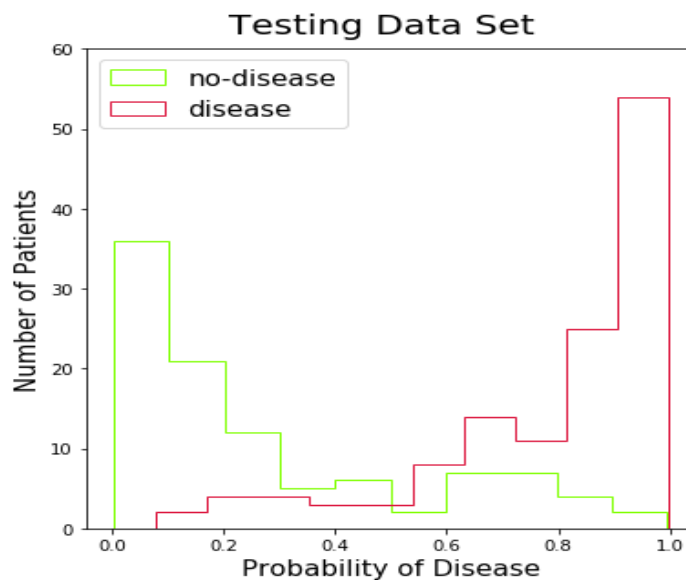
(<statsmodels.discrete.discrete_model.BinaryResultsWrapper at 0x20f04391828>,
 [2, 4, 8, 10, 11, 12, 13])

```

*Figure 76 Output for Automated Feature Selection for Testing Dataset*

It is also interesting to see that there are only 7 feature retained after the feature selection stage this time, they are as follows: (HD, sex, chol, oldpeak, cp1, cp2, cp3 and recg1). This then has only (HD, sex, chol, oldpeak, cp1, cp2 and cp3) in common, while the other features retained in the testing dataset are discarded here. This was furthered accessed after running the function on the complete dataset.

Number of disease cases: 128, no-disease cases: 102



*Figure 77 Testing Dataset Probability Plot*

Again there is a nice balance between the 2 defining outcomes for the model, with again an almost even split. It is also clear to see that the graph has all the same characteristics as that of the one for the training dataset.

## 5.1.4 Complete Dataset Output

```
Intrinsic discrepancies between disease and no-disease for cp is: 0.627439
Intrinsic discrepancies between disease and no-disease for thal is: 0.554669
Intrinsic discrepancies between disease and no-disease for exang is: 0.384881
Intrinsic discrepancies between disease and no-disease for oldpeak is: 0.313605
Intrinsic discrepancies between disease and no-disease for slope is: 0.298476
Intrinsic discrepancies between disease and no-disease for thalach is: 0.289996
Intrinsic discrepancies between disease and no-disease for chol is: 0.178247
Intrinsic discrepancies between disease and no-disease for age is: 0.177981
Intrinsic discrepancies between disease and no-disease for sex is: 0.170816
Intrinsic discrepancies between disease and no-disease for ca is: 0.054324
Intrinsic discrepancies between disease and no-disease for restbp is: 0.041973
Intrinsic discrepancies between disease and no-disease for restecg is: 0.024604
Intrinsic discrepancies between disease and no-disease for fbs is: 0.022830
```

```
Number of patients in dataframe: 920, with disease: 509, without disease: 411
```

*Figure 78 Intrinsic Discrepancies Output for Complete Dataset*

```
[0, 1, 3, 5, 6, 7, 8, 9, 10, 11, 14, 17]
LogisticRegression(C=1000.0, class_weight=None, dual=False,
                    fit_intercept=False, intercept_scaling=1, max_iter=100,
                    multi_class='ovr', n_jobs=1, penalty='l1', random_state=None,
                    solver='liblinear', tol=0.0001, verbose=0, warm_start=False)
```

```
LogisticRegression score on full data set: 0.816304
```

*Figure 79 LR using Lasso for Complete Dataset*

```
Classification report on full data set:
              precision    recall  f1-score   support

     0         0.81         0.77         0.79         411
     1         0.82         0.85         0.84         509

 avg / total         0.82         0.82         0.82         920
```

*Figure 80 Classification Report for Complete Dataset using Lasso*

```
Confusion matrix:
[[323  88]
 [ 76 433]]
```

*Figure 81 LR using lasso Complete Dataset Confusion Matrix*

With the output for the complete dataset looking more like the output from the training dataset. This shows that the results are consistent as there is only slight differences between the training dataset and that of the complete dataset. When comparing them its best again to look at the classification report in both cases and it is clear to see they are almost identical.

```

Initial -log(L)=366.93024; fit method = newton

Dropping      recg2, -log(L)=367.10827...
Dropping      recg1, -log(L)=367.39577...
Dropping      restbp, -log(L)=367.81215...
Dropping      intercept, -log(L)=368.27934...
Dropping      slope3, -log(L)=369.05953...
Dropping      fbs, -log(L)=370.20610...
Dropping      thal6, -log(L)=371.87211...

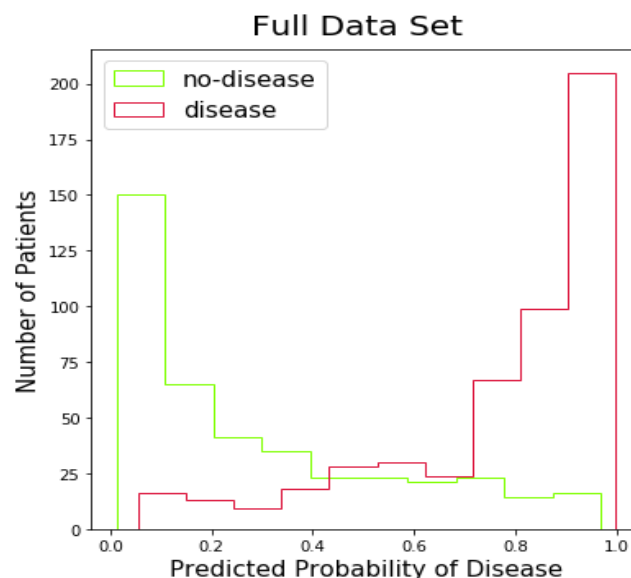
Final selection includes 12 features:

(<statsmodels.discrete.discrete_model.BinaryResultsWrapper at 0x1eb6e1230b8>,
 [1, 2, 4, 6, 7, 8, 9, 10, 11, 12, 15, 18])

```

*Figure 82 Output for Automated Feature Selection for the complete Dataset*

After automatic feature selection on the complete dataset there are again 12 features left just as there was for the training dataset, the features retained this time are as follows: (HD, age, sex, chol, thalach, exang, oldpeak, ca, cp1, cp2, cp3, slope1 and thal7). When this is compared to the output for the training dataset there is only one feature that differs in the two selection, the result from the training dataset is as follows: (HD, sex, restbp, chol, thalach, exang, oldpeak, ca, cp1, cp2, cp3, slope1 and thal7). Notice that there is an age feature in the complete dataset feature selection and there is not in the training dataset. This is the same for the restbp feature in the training dataset, as this feature is not in the complete dataset.



*Figure 83 Complete Dataset Probability Plot*

This is again showing that there is very little in the difference for the prediction probability for both the training and the complete datasets.



## 5.2 Gaussian Naïve Bayes Output

```
GaussianNB(priors=None)

GaussianNB score on full data set: 0.815217

Classification report on full data set:
      precision    recall  f1-score   support

     0       0.79      0.79      0.79       411
     1       0.83      0.83      0.83       509

 avg / total       0.82      0.82      0.82       920

Confusion matrix:
[[326  85]
 [ 85 424]]
```

*Figure 84 Naive Bayes Output*

The output for the GNB model using Scikit-Learn produced almost identical results to that of the output for the model using Weka. This then goes to show there is consistency in the results and that overfitting of the model has not occurred.

## 5.3 Random Forrest Output

```
RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini',
                        max_depth=3, max_features='auto', max_leaf_nodes=None,
                        min_impurity_decrease=0.0, min_impurity_split=None,
                        min_samples_leaf=1, min_samples_split=2,
                        min_weight_fraction_leaf=0.0, n_estimators=10, n_jobs=1,
                        oob_score=False, random_state=0, verbose=0, warm_start=False)
\Random Forrest score on full data set: 0.710870

Classification report on full data set:
      precision    recall  f1-score   support

     0       0.73      0.67      0.70       460
     1       0.69      0.76      0.72       460

 avg / total       0.71      0.71      0.71       920

Confusion matrix:
[[306 154]
 [112 348]]
```

*Figure 85 Random Forrest Output*

The RF output is showing less accuracy then when it was tested in the Weka, this is with a max depth of 3, when the max depth is removed the accuracy increases. But when loading the complete set of features into the model the accuracy drops to the 60% region.

## 5.4 Support Vector Machine Output

```
SVC(C=1.0, cache_size=200, class_weight=None, coef0=0.0,  
    decision_function_shape='ovr', degree=3, gamma='auto', kernel='rbf',  
    max_iter=-1, probability=False, random_state=None, shrinking=True,  
    tol=0.001, verbose=False)
```

Support Vector Machine score on full data set: 0.716304

```
Classification report on full data set:  
              precision    recall  f1-score   support  
  
     0           0.72       0.70       0.71         460  
     1           0.71       0.73       0.72         460  
  
avg / total           0.72       0.72       0.72        920
```

```
Confusion matrix:  
[[322 138]  
 [123 337]]
```

*Figure 86 Support Vector Machine Output*

This is down when compared to the output from the Weka testing outputs, even after inclusion of selected combinations of features the accuracy did not improve with any great significance.

## 5.5 Findings for Scikit-Learn

As the results for the LR model appear to be the most consistent and with little time left this is going to be the model used for the implementation of the visual system (webpage). To classify an instance as having a presence of disease the requirement for the logistic probability needs to be at least 50%.

This is a standard choice, other choices are also possible, this if there is more concern about one type of classification error than another. The combination of the four datasets were used for the LR model with varying results.

After changing the combination of features, the results also varied, this is something that would be part of the future plan for this project. This is that there are just too many combinations of features and different models to test in the time frame for this project.

For building a model that can predict the presence of heart disease there are only a couple of features needed. As the results for the best combination have had slight inconsistencies in the output but the combination that came out more than 90% of the time was as follows:

- Sex
- Restbp
- Thalach
- Ca
- Cp1, Cp2, Cp3
- Slope1
- Thal7

Using this combination of features in the LR model yielded an accuracy of 82%, this is by no means the best that could be achieved with this dataset and hence there is a need for this project to have a future plan.

There was an article published on Rpubs(57), that conducted a similar project but only using the Cleveland dataset, this dataset on its own consists of around 300 instances. The author Brigitte Mueller, reports an accuracy of 87% with an LR model(58). When this is compared to the testing set output from the LR model, as seen in figure 73, where the accuracy for the LR model is 85%. This suggests that with more data the accuracy is decreasing for the LR model. This is something that will be addressed in the project plan, section 7 of this document.

## 6 Development

### 6.1 Scripts Explained

#### 6.1.1 Load\_hddf.py

This is quite a simple programme, what it does is goes off to the internet and finds the UCI ML repo using a programmed URL for the desired file. Then using the pandas function read csv it reads in the required csv files and then appends the four file to one data frame. This is then written out to another csv file for reading into the main programme.

#### 6.1.2 Train\_test\_Split.py

This is also a rather simple programme that splits the data using a random split of the instances. It does this by reading in the input file in a binary form then it performs a random split of the instances keeping the features for an instance linked to that instance. It then writes the split instances out to separate csv files. This programme performs a 25/75 split on the data, it can be set to split the data into whatever size partitions are desired.

#### 6.1.3 DQR

This programme is used to build a data quality table for the features in the dataset. It is used for determining things like whether the data has a large number of missing values. This programme starts by reading in the relevant files and then performs a number of calculations. This is facilitated by the libraries in the python language as discussed at the start of this document. After the features that are not being retained have been removed the remaining features are then placed into a pandas data frame for further processing. It is at this point that the calculations are performed and then the results are then written out to a file. This file is an IPython file, it was constructed using Jupyter Notebooks.

### 6.1.4 TrainDset, TestDset and FullDset

The three of these programmes do the same thing just with different datasets getting loaded into them, this was done for testing purposes the see if there was great significance in the different outputs. The contents of these notebooks are covered in the FullVar section below as they are derived directly from that source code.

### 6.1.5 FullVer

This notebook file contains the entire working of this project except the above mentioned python files marked with the .py extension. Below is the code for the function that calculates the intrinsic discrepancies between two given histograms.

```
def intrinsic_discrepancy(x,y):
    assert len(x)==len(y)
    sumx = sum(xval for xval in x)
    sumy = sum(yval for yval in y)
    id1 = 0.0
    id2 = 0.0
    for (xval,yval) in zip(x,y):
        if (xval>0) and (yval>0):
            id1 += (float(xval)/sumx) * np.log((float(xval)/sumx)/(float(yval)/sumy))
            id2 += (float(yval)/sumy) * np.log((float(yval)/sumy)/(float(xval)/sumx))
    return min(id1,id2)
```

This function starts by confirming that the length of both x and y are equal, after doing this the sum of the values for both x and y are calculated, in this case the x and y represent the classification of either disease or no disease.

This is considered a symmetrised Kullback-Leibler distance, this function is a natural distance function. The way in which it works is it takes a probability distribution of true, to a targeted probability distribution. This is denoted as “p” for the true probability and “q” for the targeted probability. When dealing with variables that are discrete or that may not be finite for example the probability distributions for,

$p=\{p_1, ..., p_n\}$  and  $q=\{q_1, ..., q_n\}$ , is defined as follows(59):

$$KL(p, q) = \sum_i p_i \cdot \log_2(p_i / q_i)$$

*Figure 87 Kullback-Leibler Distance for discrete features*

When using this calculation for continuous features the sum of the probabilities needs to be replaced with an integral.

$$\begin{aligned} KL(p, p) &= 0 \\ KL(p, q) &\geq 0 \end{aligned}$$

*Figure 88 Kullback-Leibler distance for Continuous features*

While this function is not naturally symmetric it can be made to produce a symmetric distance. To do this we sum the distances and then put the whole formula shown below over 2(46).

$$KL(p, q) + KL(q, p)$$

*Figure 89 Kullback-Leibler Distance Symmetrised*

This is the technique that was used to ascertain the intrinsic discrepancies for the plotting of the probabilities. The programme then starts by creating a Pandas data frame for holding the data from the csv file.

There is a number of different plots then created, this was done for the purposes of checking that the data is as expected and there are no anomalies that can't be explained. After plotting all the features to check the distribution of the data, the intrinsic discrepancies for the features was calculated and the features that contained more than 2 categorical variables were extracted into their own features, using 'thal' as an example it had 3 possible outcomes that were important, when extracted out the result was 4 features that included 'thal', 'thal3', 'thal6' and 'thal7', as there is no longer a need for the 'thal' feature it gets discarded. This is also the case for the 'slope', 'cp' and 'restecg' features, while the 'num' feature is changed to reflect a binary output with cases where the 'num' feature is greater than zero, it is replaced with the value of 1, this represents a presence of heart disease. The data frame is then standardised and plotted for the distribution of the data with regard to its classification, class of either disease or no disease in this case.

After this has been completed there is an implementation of LR using Lasso on the dataset. This was the starting point of the project with regard to test the accuracy of the prediction, the workings of Lasso have been covered in a previous section of this document. An LR model is then fitted using the statsmodels api, while this method and the previous do the same thing, the output from the statsmodels api can be harder to interpret than the first method.

The next section of this notebook is where the automatic feature selection is performed. This is done by selecting all the features in the standardised data frame and then eliminating the one by one based on the significance. For a feature to be deemed insignificant if the absolute z-value for the feature is less than 2.

The pseudo code for this function would be as follows:

```
Load all features and fit the model
Calculate the deviance from the fitted model
Check if all features sequentially to see if the absolute z-value is less than 2
    If Yes:
        For each one of these features:
            Remove the feature
            Refit the model
            Recalculate the deviance
            Replace the feature
        Drop the feature with the lowest change in deviance after removal
    If No:
        Return 0;
```

What this produces is the biggest subset of features where the fit coefficient for the features are at the very least two standard deviations from zero, while also retaining the smallest fit deviance possible for the model. This is the algorithm that was applied to the optimisation for the LR, GNB, SVM and RF models. After this there is an implementation of an LR model. This is followed by the optimisation for the other listed models above.

### 6.1.6 FYP.html

This is a single page site that illustrates the prediction when given different values for the model. This was done only for demonstration purposes, if the implementation was being aimed at the medical sector it would be far more advanced than what is being used.

## 7 Project Plan

The planned project was not just to detect the presence of heart disease, it was to use real time data, transmitted from a Fitbit. With the heart rate data from the Fitbit, it was planned to use this as an indicator, with a number of user submitted values for the other features in the ML model. There was some pitfall in this as the Fitbit while it can transmit the real time heart rate of a user, it needed to be send from an application that ran from inside the actual Fitbit application. This then had a latent delay in the receiving of the data to a 3<sup>rd</sup> party application, the delay was at best between 10 and 14 minutes. As the main aim was to have this signal send out an SOS message and kill the engine to the wearer's vehicle.

The project then shifted from this into an exploration of the application of ML, with the aim of predicting the presence of heart disease for a given instance in a dataset. The application of ML to the heart data became the new main objective. This has led to a better understanding of what is needed to implement an ML project.

For the most part the project after the initial shift there have been no real issues that could be overcome with some research and large amounts of reading. If this project was to be repeated, the advice that could be given is do not underestimate the time takes to select the best combination of features from the dataset and also the time to train a model.

This is something that impacted this project, time was in short supply. With it taking up to a day to run all the combinations of features to find the best ones, it became clear that a laptop with 16GB of RAM and running an i5 processor was inferior for this type of task. This leads to the suggestion of using a cloud service such as Amazon AWS for the application of the ML model as this would reduce the time taken to perform the features selection and training of the model.

This being said, I feel that the project has achieved what was set out at the start as objectives, for this reason there is nothing the author would do different. This is because the knowledge that was acquired during this project, came from a lot of research.



## 8 Conclusion

At the beginning of this project the aim was to determine if it was possible to predict the presence of heart disease, this has been successful. The initial aim was to obtain an accuracy of around 80%. This too has been achieved, in fact the accuracy for most models came in around this figure. This was a very surprising fact in this project and will encourage further exploration of this type of application for ML.

The research conducted throughout the course of this project included many interesting aspects. Starting with the data quality report, the findings for this section of the project were vital in identifying that there is not real need for the full listing of features in the dataset as the combinations of features that were left after the final feature selection consisted of less than 10 features. The best combination of features was dependant on the model.

After looking at the outputs for the different algorithms the most accurate when testing with Weka was the SVM, this however was not the case when compared to the implementation in python. The python implementation for the SVM model only had an accuracy of 71%, this is a significant decrease in accuracy from 82%. Then there was the RF model in Weka having an accuracy of 81%, again had a decreased accuracy of 71% for the python implementation.

This is something that caused concern and this is why they were ruled out for the final model selection. It is however an avenue that could be investigated further in order to determine why there is such a decrease in the accuracy. It was not further explored as it was not in line with the main objective for this project.

Another interesting finding was that the best combination for feature selection while changing for most models, it also changed depending the amount of data used. When doing the features selection on smaller datasets the number of features retained after the selection was still usually less than 10, the actual combination of features changed.

This is something that is still not quite clear as to why this was happening, I feel that it may be caused by missing data. The reason for this is the feature selection process calculates the best combination of features based on the fit coefficient, with missing data this will change depending on the size of the dataset.

While there is an interface for users to enter details and get a probability for the presence of heart disease, this interface is not a complex application. Rather a simple one page website, that works for demonstration purposes.

This project has provided many things in terms of personal and learning objectives. There have been so many things that were never considered and most things needed far more time than I initially anticipated. This can be marked down to inexperience in the field of ML.

With limited prior knowledge in many of the areas applied other than an understanding of most basic concepts, also never having extensively used the Python language before. This project was a massive learning curve which also provided a new understanding into other aspects such as data analysis.

With a new in-depth understanding of Python, it is clear that it's a far more powerful language than previously believed to be, it is also clear now just how well it is suited to ML. The findings and research for this project have resulted in a detailed understanding in some aspects of ML and its application within the medical sector.

On a personal level this project has sparked a deep need for a further understanding into the application ML. This is not to just have the understanding but to then explore all avenues for the application of ML to classification and prediction for heart related conditions. It has also made me think I may look to this sector as a career choice, with the intent of perhaps studying a masters in ML.

## 9 Bibliography

1. UCI Machine Learning Repository: Heart Disease Data Set [Internet]. [cited 2018 Feb 11]. Available from: <http://archive.ics.uci.edu/ml/datasets/heart+Disease>
2. Cost of heart disease in Latin America tops \$30bn [Internet]. World Heart Federation. 2016 [cited 2018 Apr 10]. Available from: <https://www.world-heart-federation.org/cost-heart-disease-latin-america-tops-30bn/>
3. Cost-of-Heart-Failure-Report-web.pdf [Internet]. [cited 2018 Feb 11]. Available from: <http://heartbeat-trust.ie/wp-content/uploads/2015/12/Cost-of-Heart-Failure-Report-web.pdf>
4. ESLII.pdf [Internet]. [cited 2018 Mar 28]. Available from: <https://web.stanford.edu/~hastie/Papers/ESLII.pdf>
5. Top Artificial Intelligence Companies in Healthcare to Keep an Eye On [Internet]. The Medical Futurist. 2017 [cited 2018 Feb 11]. Available from: <http://medicalfuturist.com/top-artificial-intelligence-companies-in-healthcare/>
6. Microsoft Develops AI to Help Cancer Doctors Find the Right Treatments. Bloomberg.com [Internet]. 2016 Sep 20 [cited 2018 Feb 16]; Available from: <https://www.bloomberg.com/news/articles/2016-09-20/microsoft-develops-ai-to-help-cancer-doctors-find-the-right-treatments>
7. IBM Watson Health - IBM Watson for Oncology [Internet]. IBM Watson Health. 2017 [cited 2018 Feb 16]. Available from: <https://www.ibm.com/watson/health/oncology-and-genomics/oncology/>
8. Salkind NJ, Rasmussen K, editors. Encyclopedia of measurement and statistics. Thousand Oaks, Calif: SAGE Publications; 2007. 3 p.
9. Logistic function [Internet]. [cited 2018 Mar 28]. Available from: [https://ipfs.io/ipfs/QmXoypizjW3WknFiJnKLwHCnL72vedxjQkDDP1mXWo6uco/wiki/Logistic\\_function.html](https://ipfs.io/ipfs/QmXoypizjW3WknFiJnKLwHCnL72vedxjQkDDP1mXWo6uco/wiki/Logistic_function.html)
10. Logistic regression [Internet]. [cited 2018 Mar 28]. Available from: [https://ipfs.io/ipfs/QmXoypizjW3WknFiJnKLwHCnL72vedxjQkDDP1mXWo6uco/wiki/Logistic\\_regression.html](https://ipfs.io/ipfs/QmXoypizjW3WknFiJnKLwHCnL72vedxjQkDDP1mXWo6uco/wiki/Logistic_regression.html)
11. Linear predictor function [Internet]. [cited 2018 Mar 28]. Available from: [https://ipfs.io/ipfs/QmXoypizjW3WknFiJnKLwHCnL72vedxjQkDDP1mXWo6uco/wiki/Linear\\_predictor\\_function.html](https://ipfs.io/ipfs/QmXoypizjW3WknFiJnKLwHCnL72vedxjQkDDP1mXWo6uco/wiki/Linear_predictor_function.html)
12. van der Meulen F, Vermaat T, Willems P. Case Study: An Application of Logistic Regression in a Six Sigma Project in Health Care. Qual Eng. 2011 Mar 2;23(2):113–24.

13. HAL PDF Full Text [Internet]. [cited 2018 Mar 10]. Available from: <https://hal.archives-ouvertes.fr/hal-01081557/document>
14. Random Forest Regression | Turi Machine Learning Platform User Guide [Internet]. [cited 2018 Mar 28]. Available from: [https://turi.com/learn/userguide/supervised-learning/random\\_forest\\_regression.html](https://turi.com/learn/userguide/supervised-learning/random_forest_regression.html)
15. PubMed Central Full Text PDF [Internet]. [cited 2018 Mar 10]. Available from: <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4062420/pdf/pone.0098587.pdf>
16. ScienceDirect Full Text PDF [Internet]. [cited 2018 Mar 10]. Available from: [https://ac.els-cdn.com/S2213158214001326/1-s2.0-S2213158214001326-main.pdf?\\_tid=ccd9d81b-f979-4f7c-ab00-e8f9968cf8a9&acdnat=1523380958\\_c5b78435b2e01bb0a70b3f25501ed4bf](https://ac.els-cdn.com/S2213158214001326/1-s2.0-S2213158214001326-main.pdf?_tid=ccd9d81b-f979-4f7c-ab00-e8f9968cf8a9&acdnat=1523380958_c5b78435b2e01bb0a70b3f25501ed4bf)
17. Kotsiantis et al. - 2006 - Machine learning a review of classification and c.pdf [Internet]. [cited 2018 Mar 28]. Available from: [http://www.cs.bham.ac.uk/~pxt/IDA/class\\_rev.pdf](http://www.cs.bham.ac.uk/~pxt/IDA/class_rev.pdf)
18. Yu et al. - 2010 - Application of support vector machine modeling for.pdf [Internet]. [cited 2018 Mar 10]. Available from: <https://bmcmedinformdecismak.biomedcentral.com/track/pdf/10.1186/1472-6947-10-16>
19. Kampouraki et al. - 2013 - e-Doctor A Web based Support Vector Machine for A.pdf [Internet]. [cited 2018 Mar 10]. Available from: [https://ac.els-cdn.com/S1877042813003716/1-s2.0-S1877042813003716-main.pdf?\\_tid=c0eaaf15-bb14-4cfd-9ebb-4ee28e46f7ff&acdnat=1523381527\\_7461fbf863d60e315506733bb4747285](https://ac.els-cdn.com/S1877042813003716/1-s2.0-S1877042813003716-main.pdf?_tid=c0eaaf15-bb14-4cfd-9ebb-4ee28e46f7ff&acdnat=1523381527_7461fbf863d60e315506733bb4747285)
20. Linear Discriminant Analysis [Internet]. Dr. Sebastian Raschka. 2014 [cited 2018 Mar 29]. Available from: [sebastianraschka.com/Articles/2014\\_python\\_lda.html](http://sebastianraschka.com/Articles/2014_python_lda.html)
21. Full Text PDF [Internet]. [cited 2018 Mar 10]. Available from: [https://www.researchgate.net/profile/Haoyi\\_Xiong/publication/309203021\\_Daehr\\_A\\_Discriminant\\_Analysis\\_Framework\\_for\\_Electronic\\_Health\\_Record\\_Data\\_and\\_an\\_Application\\_to\\_Early\\_Detection\\_of\\_Mental\\_Health\\_Disorders/links/59cc0af90f7e9bbf3b77cf/Daehr-A-Discriminant-Analysis-Framework-for-Electronic-Health-Record-Data-and-an-Application-to-Early-Detection-of-Mental-Health-Disorders.pdf](https://www.researchgate.net/profile/Haoyi_Xiong/publication/309203021_Daehr_A_Discriminant_Analysis_Framework_for_Electronic_Health_Record_Data_and_an_Application_to_Early_Detection_of_Mental_Health_Disorders/links/59cc0af90f7e9bbf3b77cf/Daehr-A-Discriminant-Analysis-Framework-for-Electronic-Health-Record-Data-and-an-Application-to-Early-Detection-of-Mental-Health-Disorders.pdf)
22. Naive Bayes Classifier [Internet]. [cited 2018 Mar 29]. Available from: <http://www.statsoft.com/Textbook/Naive-Bayes-Classifier>
23. Bayes' Theorem and Conditional Probability | Brilliant Math & Science Wiki [Internet]. [cited 2018 Mar 29]. Available from: <https://brilliant.org/wiki/bayes-theorem/>

24. Bhuvaneswari and Kalaiselvi - Naive Bayesian Classification Approach in Healthca.pdf [Internet]. [cited 2018 Mar 10]. Available from: <https://pdfs.semanticscholar.org/9ac9/feade95495780dc7f6d245a41e1a3686c7ff.pdf>
25. KNN Classification [Internet]. [cited 2018 Mar 29]. Available from: [http://www.saedsayad.com/k\\_nearest\\_neighbors.htm](http://www.saedsayad.com/k_nearest_neighbors.htm)
26. k-Nearest Neighbors [Internet]. [cited 2018 Mar 29]. Available from: <http://www.statsoft.com/Textbook/k-Nearest-Neighbors>
27. Punjani et al. - 2016 - Automated Medical Diagnosis using K-Nearest Neighb.pdf [Internet]. [cited 2018 Apr 10]. Available from: <http://euroasiapub.org/wp-content/uploads/2016/10/16IMApril-3417-1.pdf>
28. Brownlee J. Best Programming Language for Machine Learning [Internet]. Machine Learning Mastery. 2014 [cited 2018 Feb 16]. Available from: <https://machinelearningmastery.com/best-programming-language-for-machine-learning/>
29. Snapshot [Internet]. [cited 2018 Feb 10]. Available from: <https://www.datasciencecentral.com/profiles/blogs/10-popular-java-machine-learning-tools-libraries>
30. Weka 3 - Data Mining with Open Source Machine Learning Software in Java [Internet]. [cited 2018 Feb 11]. Available from: <https://www.cs.waikato.ac.nz/ml/weka/documentation.html>
31. Weka 3 - Data Mining with Open Source Machine Learning Software in Java [Internet]. [cited 2018 Feb 11]. Available from: <https://www.cs.waikato.ac.nz/ml/weka/documentation.html>
32. MALLET homepage [Internet]. [cited 2018 Feb 10]. Available from: <http://mallet.cs.umass.edu/>
33. Classification Developer's Guide [Internet]. [cited 2018 Feb 11]. Available from: <http://mallet.cs.umass.edu/classifier-devel.php>
34. Posted by Demnag on September 13 2015 at 8:07pm, Blog V. 10 Popular Java Machine Learning Tools & Libraries [Internet]. [cited 2018 Feb 10]. Available from: <https://www.datasciencecentral.com/profiles/blogs/10-popular-java-machine-learning-tools-libraries>
35. Contributing to NumPy — NumPy v1.13.dev0 Manual [Internet]. [cited 2018 Feb 16]. Available from: <https://docs.scipy.org/doc/numpy-dev/dev/>
36. The SciPy Stack specification — SciPy.org [Internet]. [cited 2018 Feb 16]. Available from: <https://www.scipy.org/stackspec.html>

37. SciPy — SciPy v1.1.0.dev0+e7d9c0d Reference Guide [Internet]. [cited 2018 Feb 16]. Available from: <http://scipy.github.io/devdocs/>
38. The N-dimensional array (ndarray) — NumPy v1.13.dev0 Manual [Internet]. [cited 2018 Feb 16]. Available from: <https://docs.scipy.org/doc/numPy-dev/reference/arrays.ndarray.html>
39. API - importing from Scipy — SciPy v1.0.0 Reference Guide [Internet]. [cited 2018 Feb 16]. Available from: <https://docs.scipy.org/doc/scipy/reference/api.html>
40. Python Data Analysis Library — pandas: Python Data Analysis Library [Internet]. [cited 2018 Feb 16]. Available from: <https://pandas.pydata.org/>
41. scikit-learn: machine learning in Python — scikit-learn 0.19.1 documentation [Internet]. [cited 2018 Feb 16]. Available from: <http://scikit-learn.org/stable/>
42. User's Guide — Matplotlib 2.1.2 documentation [Internet]. [cited 2018 Feb 16]. Available from: <https://matplotlib.org/users/index.html>
43. Heart disease prediction [Internet]. [cited 2018 Feb 16]. Available from: <https://www.kaggle.com/c/SAheart>
44. [cited 2018 Feb 25]. Available from: <http://archive.ics.uci.edu/ml/machine-learning-databases/heart-disease/heart-disease.names>
45. Blood Pressure : Blood pressure chart [Internet]. [cited 2018 Mar 4]. Available from: <http://www.bloodpressureuk.org/BloodPressureandyou/Thebasics/Bloodpressurechart>
46. Serum Cholesterol: Understanding Your Levels [Internet]. Healthline. 2017 [cited 2018 Mar 4]. Available from: <https://www.healthline.com/health/serum-cholesterol>
47. Normal and Diabetic Blood Sugar Level Ranges - Blood Sugar Levels for Diabetes [Internet]. [cited 2018 Mar 4]. Available from: [https://www.diabetes.co.uk/diabetes\\_care/blood-sugar-level-ranges.html](https://www.diabetes.co.uk/diabetes_care/blood-sugar-level-ranges.html)
48. 2 easy, accurate ways to measure your heart rate [Internet]. Mayo Clinic. [cited 2018 Mar 4]. Available from: <http://www.mayoclinic.org/healthy-lifestyle/fitness/expert-answers/heart-rate/faq-20057979>
49. Exercise Induced Angina [Internet]. Rehabilitate Your Heart. 2013 [cited 2018 Mar 4]. Available from: <https://rehabilitateyourheart.wordpress.com/2013/01/16/exercise-induced-angina/>
50. Mater Private - Treadmill Stress Test [Internet]. [cited 2018 Mar 4]. Available from: <http://www.materprivate.ie/dublin/centre-services/all-services/treadmill-stress-test/index.xml>

51. RefAna.pdf [Internet]. [cited 2018 Mar 14]. Available from: <https://www.uv.es/~bernardo/RefAna.pdf>
52. A Complete Tutorial on Ridge and Lasso Regression in Python [Internet]. Analytics Vidhya. 2016 [cited 2018 Mar 20]. Available from: <https://www.analyticsvidhya.com/blog/2016/01/complete-tutorial-ridge-lasso-regression-python/>
53. Interpreting logistic regression coefficients [Internet]. [cited 2018 Mar 2]. Available from: <http://www.mypolyuweb.hk/~sjpolit/logisticregression.html>
54. Interpreting logistic regression coefficients [Internet]. [cited 2018 Mar 2]. Available from: <http://www.mypolyuweb.hk/~sjpolit/logisticregression.html>
55. Snapshot [Internet]. [cited 2018 Feb 2]. Available from: <https://stats.idre.ucla.edu/other/mult-pkg/faq/general/faq-how-do-i-interpret-odds-ratios-in-logistic-regression/>
56. StatsModels: Statistics in Python — statsmodels 0.8.0 documentation [Internet]. [cited 2018 Mar 20]. Available from: <https://www.statsmodels.org/stable/index.html>
57. RPubs [Internet]. [cited 2018 Mar 11]. Available from: <https://rpubs.com/>
58. RPubs - Machine learning for heart disease prediction [Internet]. [cited 2018 Mar 11]. Available from: <https://rpubs.com/mbbrigitte/HeartDisease>
59. Kullback Leibler (KL) Distance (Divergence) of two Probability Distributions [Internet]. [cited 2018 Feb 3]. Available from: <http://www.allisons.org/ll/MML/KL/>

# Appendix A

## Full listing of attributes:

- 1 id: patient identification number
- 2 ccf: social security number (I replaced this with a dummy value of 0)
- 3 age: age in years
- 4 sex: sex (1 = male; 0 = female)
- 5 painloc: chest pain location (1 = substernal; 0 = otherwise)
- 6 painexer (1 = provoked by exertion; 0 = otherwise)
- 7 relrest (1 = relieved after rest; 0 = otherwise)
- 8 pncaden (sum of 5, 6, and 7)
- 9 cp: chest pain type
  - Value 1: typical angina
  - Value 2: atypical angina
  - Value 3: non-anginal pain
  - Value 4: asymptomatic
- 10 trestbps: resting blood pressure (in mm Hg on admission to the hospital)
- 11 htn
- 12 chol: serum cholesterol in mg/dl
- 13 smoke: I believe this is 1 = yes; 0 = no (is or is not a smoker)
- 14 cigs (cigarettes per day)
- 15 years (number of years as a smoker)
- 16 fbs: (fasting blood sugar > 120 mg/dl) (1 = true; 0 = false)
- 17 dm (1 = history of diabetes; 0 = no such history)
- 18 famhist: family history of coronary artery disease (1 = yes; 0 = no)
- 19 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: probable or definite left ventricular hypertrophy by Estes' criteria
- 20 ekgmo (month of exercise ECG reading)
- 21 ekgday(day of exercise ECG reading)
- 22 ekgyr (year of exercise ECG reading)
- 23 dig (digitalis used during exercise ECG: 1 = yes; 0 = no)
- 24 prop (Beta blocker used during exercise ECG: 1 = yes; 0 = no)
- 25 nitr (nitrates used during exercise ECG: 1 = yes; 0 = no)
- 26 pro (calcium channel blocker used during exercise ECG: 1 = yes; 0 = no)
- 27 diuretic (diuretic used during exercise ECG: 1 = yes; 0 = no)
- 28 proto: exercise protocol
  - 1 = Bruce
  - 2 = Kottus
  - 3 = McHenry
  - 4 = fast Balke
  - 5 = Balke
  - 6 = Noughton
  - 7 = bike 150 kpa min/min



- 8 = bike 125 kpa min/min
- 9 = bike 100 kpa min/min
- 10 = bike 75 kpa min/min
- 11 = bike 50 kpa min/min
- 12 = arm ergometer

29 thaldur: duration of exercise test in minutes

30 thaltime: time when ST measure depression was noted

31 met: mets achieved

32 thalach: maximum heart rate achieved

33 thalrest: resting heart rate

34 tpeakbps: peak exercise blood pressure (first of 2 parts)

35 tpeakbpd: peak exercise blood pressure (second of 2 parts)

36 dummy

37 trestbpd: resting blood pressure

38 exang: exercise induced angina (1 = yes; 0 = no)

39 xhypo: (1 = yes; 0 = no)

40 oldpeak = ST depression induced by exercise relative to rest

41 slope: the slope of the peak exercise ST segment

- Value 1: upsloping
- Value 2: flat
- Value 3: downsloping

42 rldv5: height at rest

43 rldv5e: height at peak exercise

44 ca: number of major vessels (0-3) colored by flourosopy

45 restckm: irrelevant

46 exerckm: irrelevant

47 restef: rest raidonuclid (sp?) ejection fraction

48 restwm: rest wall (sp?) motion abnormality

- 0 = none
- 1 = mild or moderate
- 2 = moderate or severe
- 3 = akinesis or dyskmem (sp?)

49 exeref: exercise radinalid (sp?) ejection fraction

50 exerwm: exercise wall (sp?) motion

51 thal: 3 = normal; 6 = fixed defect; 7 = reversable defect

52 thalsev: not used

53 thalpul: not used

54 earlobe: not used

55 cmo: month of cardiac cath (sp?) (perhaps "call")

56 cday: day of cardiac cath (sp?)

57 cyr: year of cardiac cath (sp?)

58 num: diagnosis of heart disease (angiographic disease status)

- Value 0: < 50% diameter narrowing
- Value 1: > 50% diameter narrowing

59 lmt

60 ladprox

61 laddist

62 diag

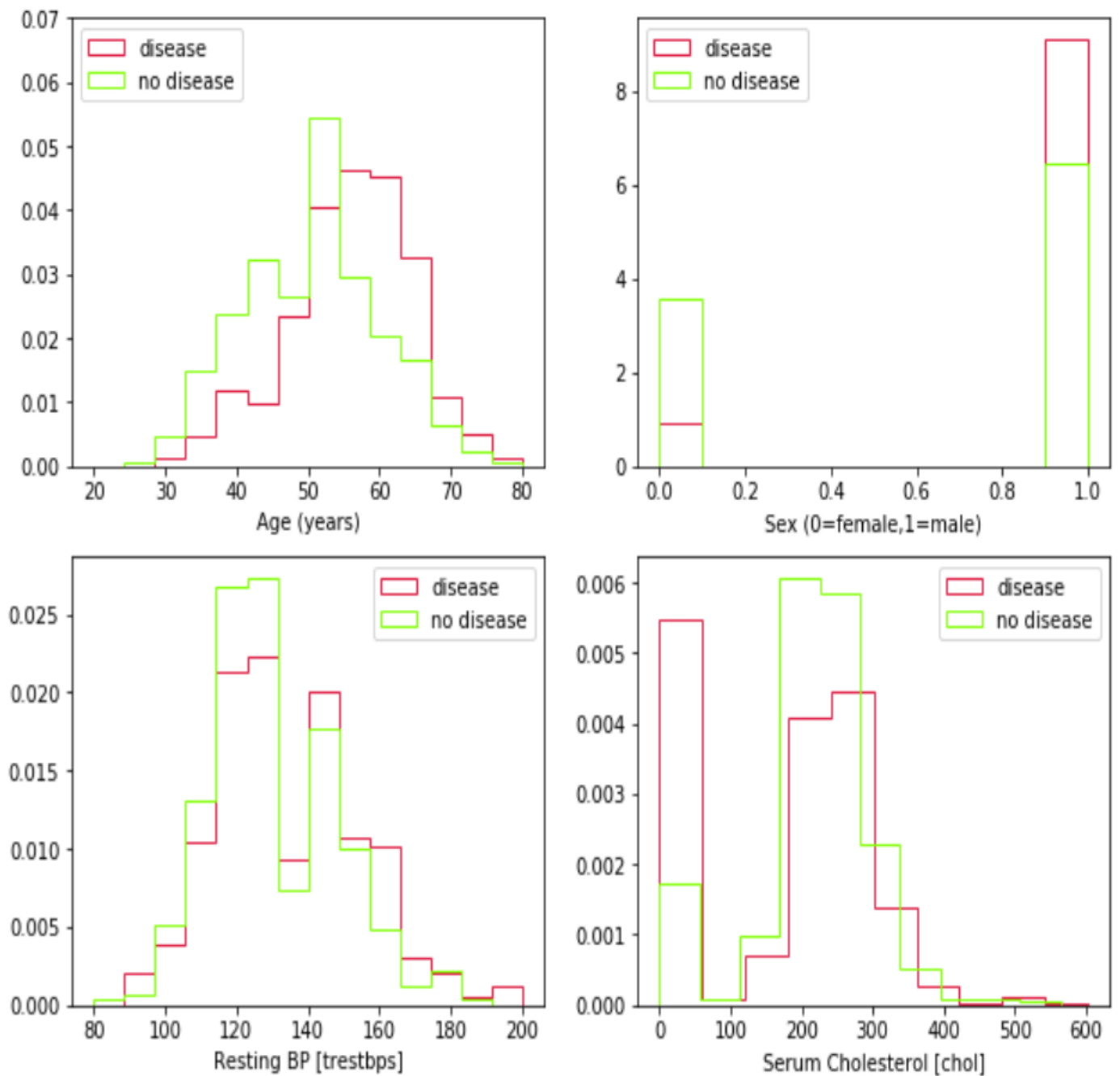
63 cxmain

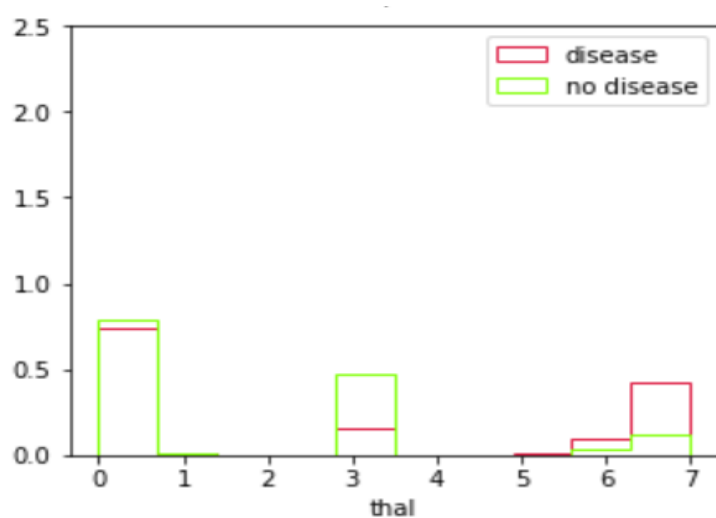
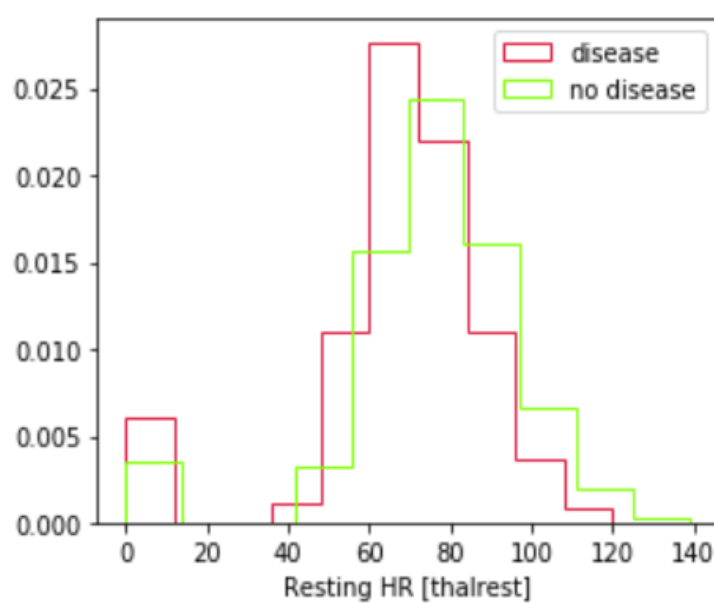
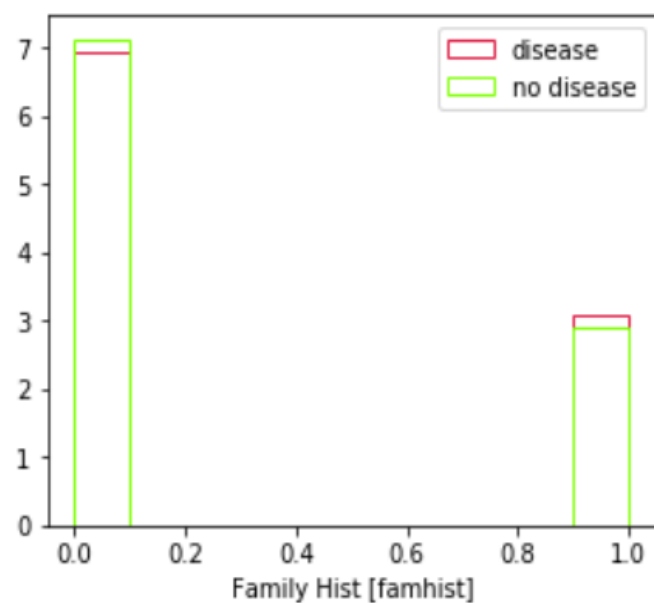
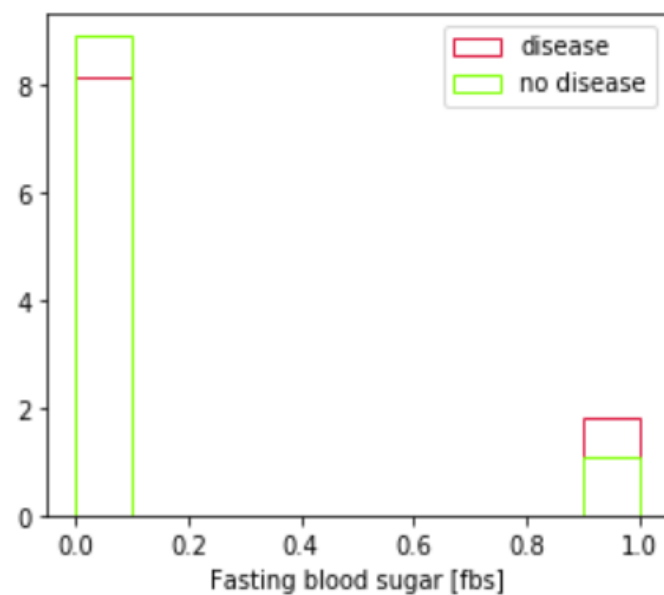
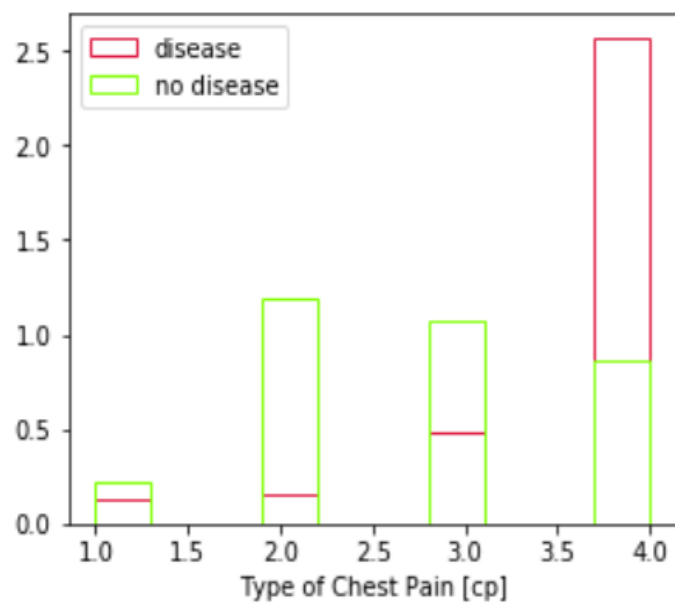
64 ramus

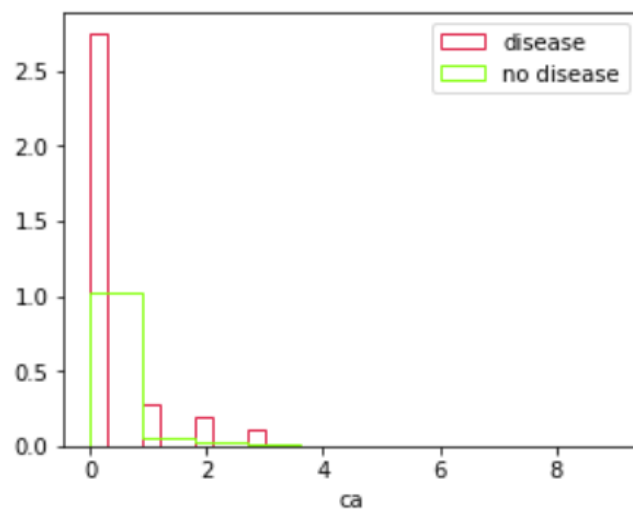
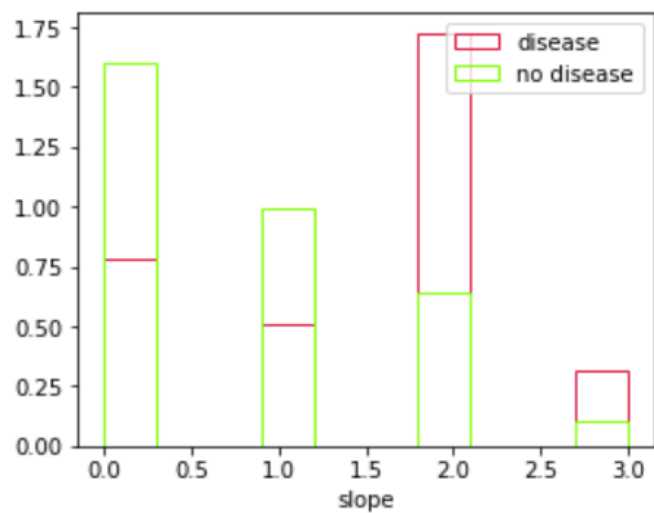
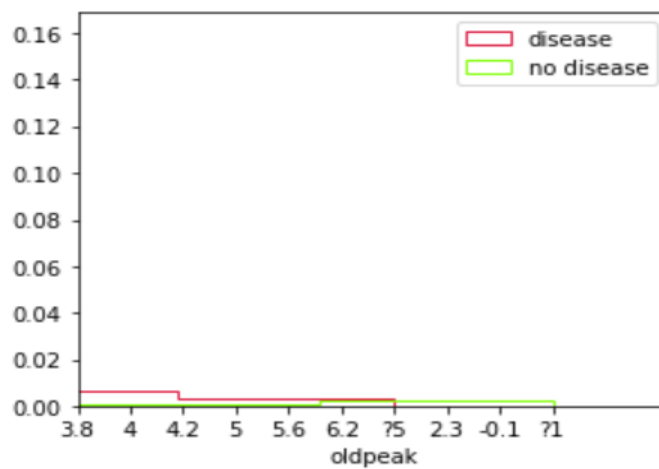
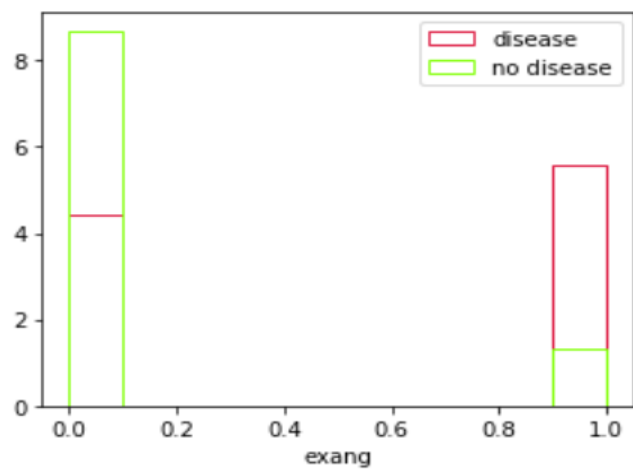
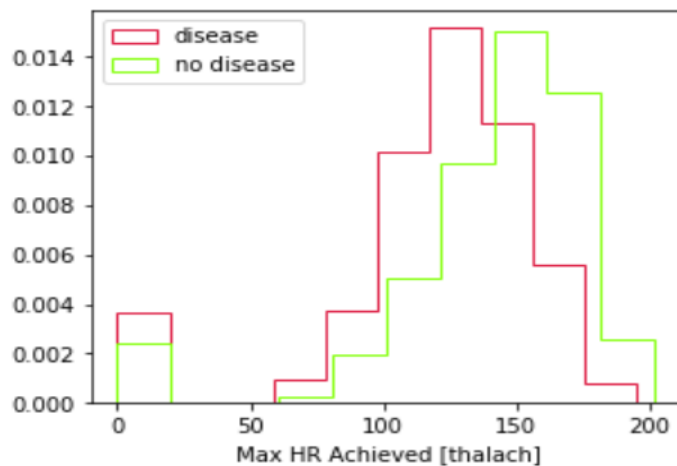
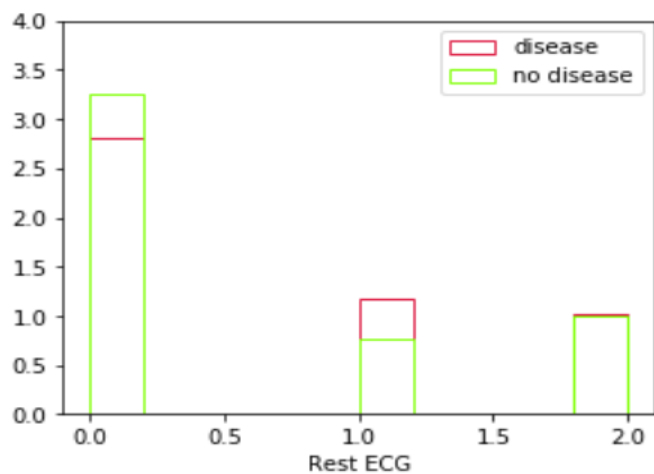
65 om1  
66 om2  
67 rcaprox  
68 readist  
69 lvx1: not used  
70 lvx2: not used  
71 lvx3: not used  
72 lvx4: not used  
73 lvf: not used  
74 cathef: not used  
75 junk: not used  
76 name: last name of patient (this was replaced this with the dummy string "name")

# Appendix B

## Original Data Plots







## Pre-processed Data

