Understanding the Signs of Cardiovascular Disease and Predicting the Presence of Cardiovascular Disease

By Stephen Hallett

GitHub: https://github.com/Stephen-Hallett/Cardiovascular_Disease_Prediction_BDAS

Table of Contents

1. Situation understanding

1.1 Identify the objectives of the situation

Cardiovascular disease is the leading cause of death in New zealand (Chan et al., 2008) and as such remains an active area of concern and research. This study aims to examine the strongest predictors of cardiovascular disease in order to develop a stronger understanding of what makes a person vulnerable to cardiovascular disease, and to develop a predictive model in order to predict a patient's likelihood of having cardiovascular disease.

1.2 Assess the situation

In order to complete this task, we will require a dataset which features a number of attributes related to cardiovascular health, which is accurate and unbiased. When undertaking this study we will assume that the dataset is representative of the total population and that the data is reliable, with no hidden or unrecorded variables which might impact the prediction of cardiovascular disease likelihood. Investigating this issue presents a number of constraints to take into consideration. Firstly there is a lack of usable data due to privacy concerns in releasing personal health data. This is a problem both in terms of the amount of data available, and also the absence of some likely helpful attributes to the data such as exercise rate and diet. This may interfere with being able to train the best possible model achievable, but we will achieve the most we can from the data we have access to. A risk we may face is that the best model for predicting cardiovascular disease likelihood may not be interpretable, which we will need to consider in the model fitting process. To address these concerns we will apply the following contingencies. First we will apply appropriate data augmentation techniques, as well as engineering as many potentially useful attributes as possible. And we will ensure that interpretable and explainable models are fit in order to gain a thorough understanding of the important predictors of cardiovascular disease.

1.3 Determine data mining objectives

This data mining endeavour comprises two objectives.
Objective 1: To uncover the most telling and important predictors of cardiovascular disease in patients
Objective 2: To develop a predictive model in order to identify the presence of cardiovascular disease in patients

Success in these goals would be a list of the significant predictors and critical levels for each, above which cardiovascular disease risk becomes significantly high. As well as a model which achieves 85% plus accuracy and 90% recall in classifying whether a patient has cardiovascular disease.

1.4 Produce a project plan

The plan for the coming weeks includes initially performing all data preprocessing and addressing the issues discovered in 2.3 and 2.4, followed by appropriate feature engineering and data augmentation. Next data reduction and transformation will be applied to ensure data is in the proper form to be investigated thoroughly. To address objective 1, a forward stepwise logistic regression will be performed to get an initial visualisation of the most relevant attributes in the data for predicting the presence of cardiovascular disease, followed by more advanced techniques such as ridge regression and decision trees to back up the logistic regression analysis. Additionally to address objective 2 I will experiment with a variety of machine learning models used to predict the presence of cardiovascular disease in patients, initially exploring several classification models including , random forest, naive bayes and SVM models. Once the results of these models have been visualised and explored, the data will be segmented into a train and test set to do a final assessment of the most informative models chosen to explore objective 1. I will then improve on the best model(s) chosen for objective 2 using the train and test set and create the final predictive model and visualise and describe the results. I will then interpret the results from data mining for both objectives and perform iterations to finalise all of the final models. A day to day timeline for this project is outlined in figure 1.

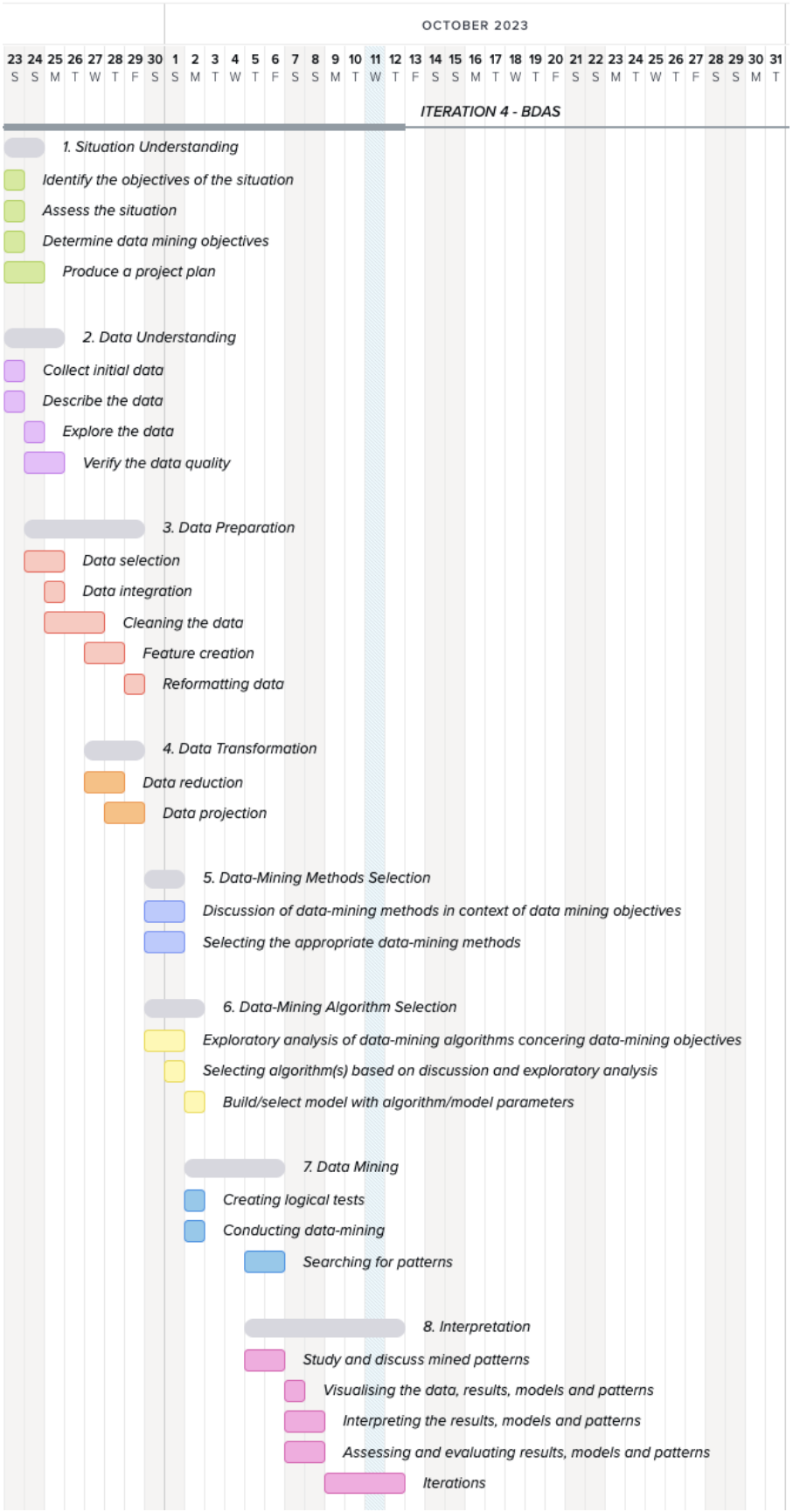


Figure 1: Gantt chart for day to day project plan

2. Data understanding

2.1 Collect initial data

The data set used for this study was downloaded as a csv file from kaggle and dates from 1988 consisting of four databases: Cleveland, Hungary, Switzerland, and Long Beach V (Lapp, 2019). In order to allow for more data mining to take place, I split this single dataset into two datasets - patient_data.csv and medical_data.csv. The patient dataset includes the age and sex of each patient, while the medical dataset includes more nitty gritty medical data such as resting blood pressure, cholesterol etc., and both datasets contain a matching id variable for integration. Since the data was completely clean, I introduced random NA values to some features, choosing to replace a uniformly random amount of 2.5-7.5% of data in three variables to NA values.

2.2 Describe the data

These two datasets each contain 1025 observations, and feature a collective 15 attributes, including an ID variable, and the response variable “target” which refers to the presence of cardiovascular disease in the patient (1 for present, 0 for not present).
The patient information dataset contains the following variables:

- **ID** - Arbitrary patient ID (consistent between both datasets)
- **Age** - age in years
- **Sex** - (1 = male, 0 = female)

The medical information dataset contains the following variables:

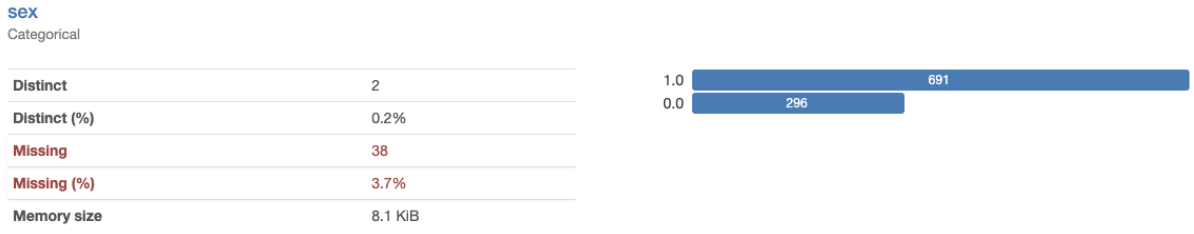
- **ID** - Arbitrary patient ID (consistent between both datasets)
- **cp** - chest pain type (4 values)
    - 0 - asymptomatic
    - 1 - atypical angina
    - 2 - non-anginal pain
    - 3 - typical angina
- **trestbps** - resting blood pressure
- **chol** - serum cholesterol (mg/dl)
- **fbs** - fasting blood sugar > 120mg/dl
- **restecg** - resting electrocardiographic results (0,1,2)
- **thalach** - Maximum heart rate achieved
- **exang** - Exercise induced angina
- **oldpeak** - Oldpeak = ST depression induced by exercise relative to rest
- **slope** - The slope of the peak exercise ST segment
- **ca** - Number of major vessels coloured by fluoroscopy (0-3)
- **thal** - A blood disorder called thalassemia (1 = fixed defect, 2 = normal, 3 = reversible defect)
- **target** - presence of cardiovascular disease in the patient (1 for present, 0 for not present)

2.3 Explore the data

An exploratory analysis of the raw data demonstrates some interesting findings which will need to be considered when performing data cleaning. One important observation in the raw data is the presence of null values in three attributes spanning both continuous variables as well as nominal variables. These null values will need to be addressed separately in order to get the most out of the small dataset. Additionally there appear to be some slightly skewed distributions within the dataset, particularly the trestbps and oldpeak variables show some right skew, and potentially also the chol variable, however this could simply be some outliers which need to be coerced or removed. Each of these variables will need to be inspected and addressed in the data cleaning stage. Additionally a closer look at this dataset indicates that the data is comprised primarily of categorical variables rather than continuous (real number) variables. Of the 14 non ID unique variables, only 5 of them are continuous, with the rest being categorical variables.

```
root
 |-- id: integer (nullable = true)
 |-- age: integer (nullable = true)
 |-- sex: integer (nullable = true)
```

```
|summary|count|mean               |stddev              |min|max |
|id     |1025 |513.0              |296.0363153398583   |1  |1025|
|age    |1025 |54.43414634146342  |9.072290233244278   |29 |77  |
|sex    |987  |0.7001013171225937 |0.45844563856310544 |0  |1   |
+-------+-----+-------------------+--------------------+---+----+
```



Figures 2,3 & 4: Raw data in patient_data.csv

```
root
 |-- id: integer (nullable = true)
 |-- cp: integer (nullable = true)
 |-- trestbps: integer (nullable = true)
 |-- chol: integer (nullable = true)
 |-- fbs: integer (nullable = true)
 |-- restecg: integer (nullable = true)
 |-- thalach: integer (nullable = true)
 |-- exang: integer (nullable = true)
 |-- oldpeak: double (nullable = true)
 |-- slope: integer (nullable = true)
 |-- ca: integer (nullable = true)
 |-- thal: integer (nullable = true)
 |-- target: integer (nullable = true)
```

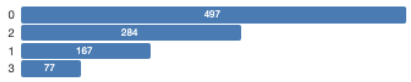
summary	count	mean	stddev	min	max
id	1025	513.0	296.0363153398583	1	1025
cp	1025	0.9424390243902439	1.029640743645865	0	3
trestbps	1025	131.61170731707318	17.516718005376408	94	200
chol	982	246.01221995926682	51.7080514274769	126	564
fbs	1025	0.14926829268292682	0.35652668972715756	0	1
restecg	1025	0.5297560975609756	0.5278775668748918	0	2
thalach	1025	149.11414634146342	23.00572374597721	71	202
exang	965	0.3378238341968912	0.4732134233484412	0	1
oldpeak	1025	1.0715121951219524	1.1750532551501767	0.0	6.2
slope	1025	1.3853658536585365	0.6177552671745918	0	2
ca	1025	0.7541463414634146	1.0307976650242825	0	4
thal	1025	2.32390243902439	0.6206602380510303	0	3
target	1025	0.5131707317073171	0.5000704980788011	0	1

cp

Categorical

HIGH CORRELATION

Distinct	4
Distinct (%)	0.4%
Missing	0
Missing (%)	0.0%
Memory size	8.1 KiB

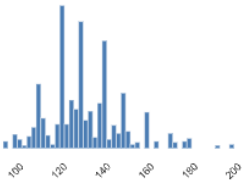


More details

trestbps

Real number (ℝ)

Distinct	49	**Minimum**	94
Distinct (%)	4.8%	**Maximum**	200
Missing	0	**Zeros**	0
Missing (%)	0.0%	**Zeros (%)**	0.0%
Infinite	0	**Negative**	0
Infinite (%)	0.0%	**Negative (%)**	0.0%
Mean	131.61171	**Memory size**	4.1 KiB



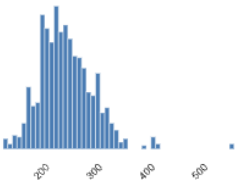
More details

chol

Real number (ℝ)

MISSING

Distinct	152	**Minimum**	126
Distinct (%)	15.5%	**Maximum**	564
Missing	43	**Zeros**	0
Missing (%)	4.2%	**Zeros (%)**	0.0%
Infinite	0	**Negative**	0
Infinite (%)	0.0%	**Negative (%)**	0.0%
Mean	246.01222	**Memory size**	8.1 KiB



More details

fbs

Categorical

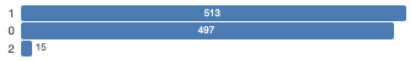
Distinct	2
Distinct (%)	0.2%
Missing	0
Missing (%)	0.0%
Memory size	8.1 KiB

More details

restecg

Categorical

Distinct	3
Distinct (%)	0.3%
Missing	0
Missing (%)	0.0%
Memory size	8.1 KiB

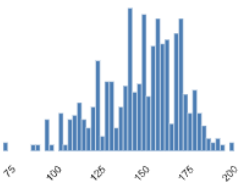


More details

thalach

Real number (ℝ)

Distinct	91	**Minimum**	71
Distinct (%)	8.9%	**Maximum**	202
Missing	0	**Zeros**	0
Missing (%)	0.0%	**Zeros (%)**	0.0%
Infinite	0	**Negative**	0
Infinite (%)	0.0%	**Negative (%)**	0.0%
Mean	149.11415	**Memory size**	4.1 KiB



More details

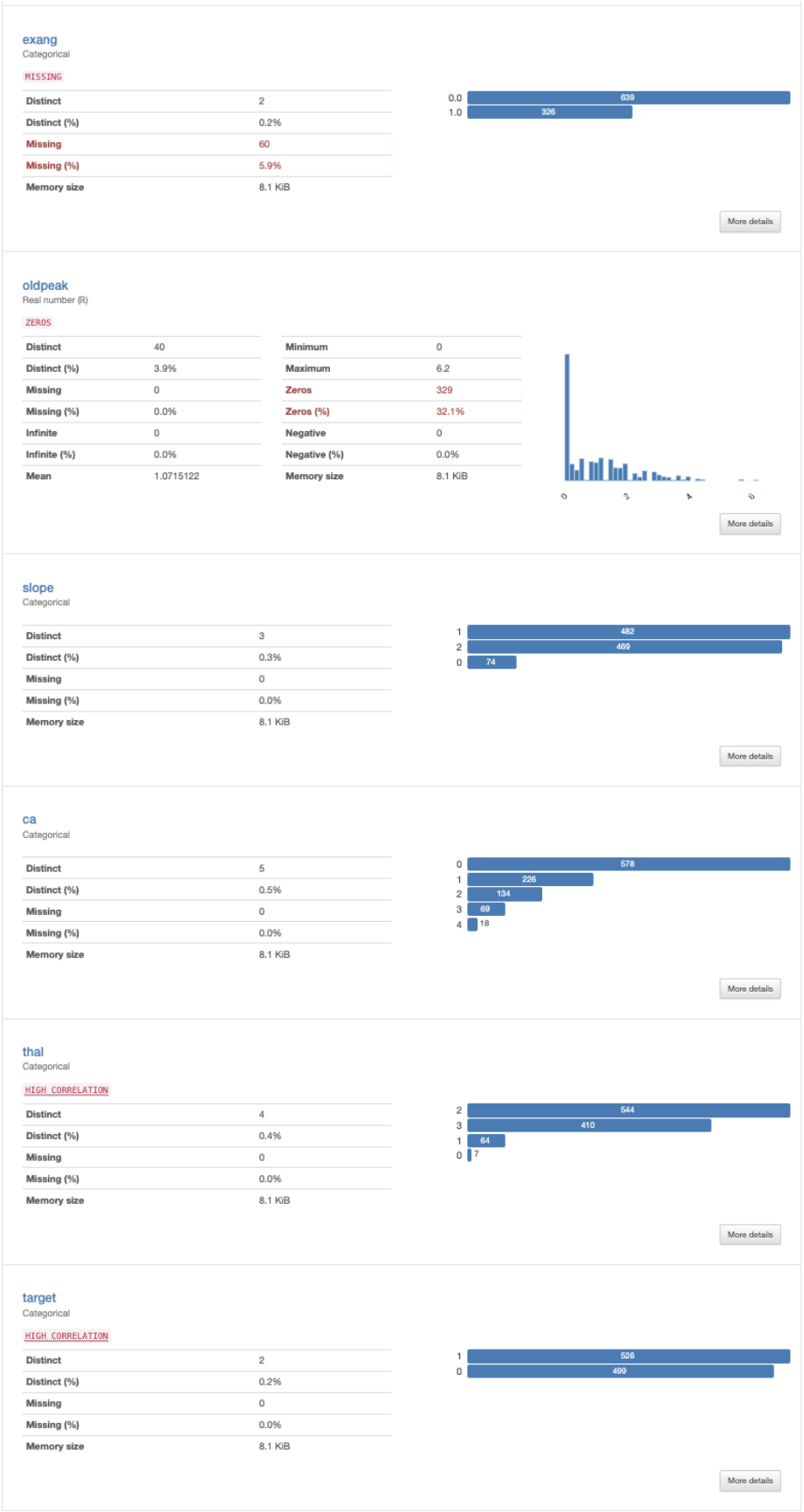
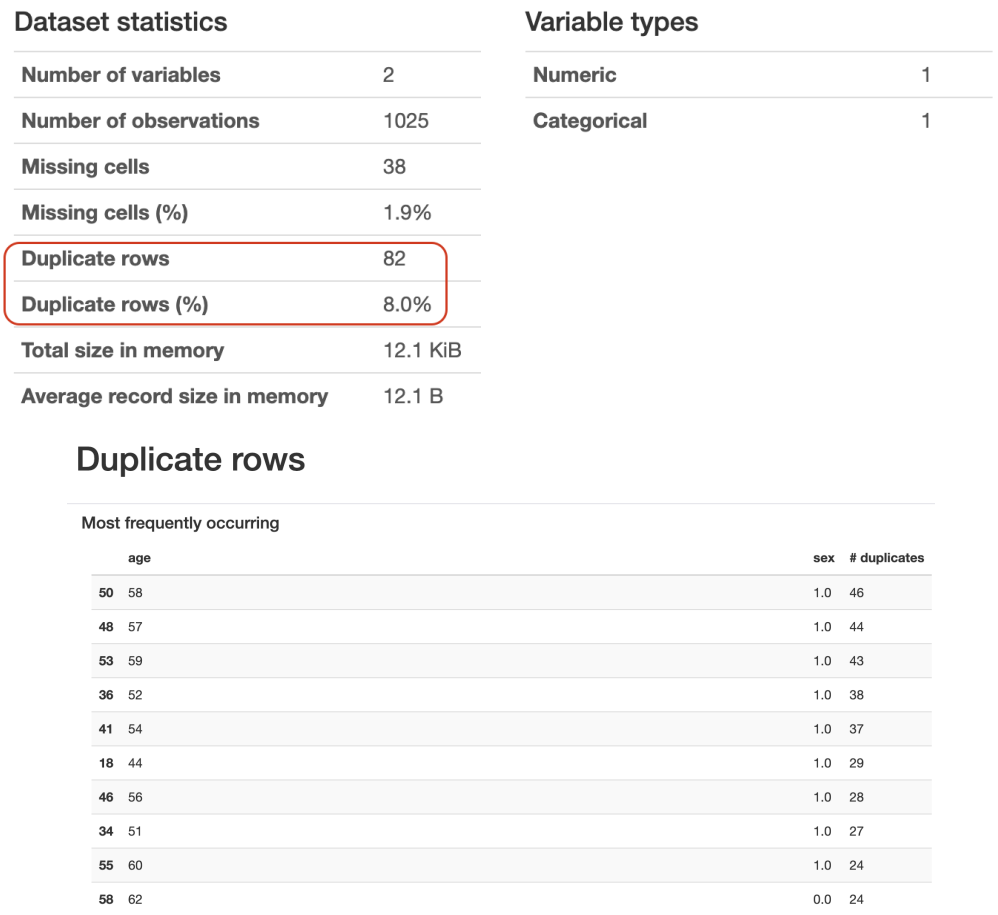


Figure 5,6,7 & 8: Raw data in medical_data.csv

2.4 Verify the data quality

As was mentioned in section 2.3, there are three variables which feature missing data being sex, chol and exang which will each need to be addressed individually and effectively so as to keep as much usable data as possible. An inspection into whether or not the data contained duplicate observations revealed a concerning observation in that it appears the majority of observations in the dataset have duplicate copies. We can see from figure 9 and 11 that the patient and medical datasets feature 82 and 298 rows with duplicate copies respectively. A further examination into the number of duplicate rows in the patient dataset in figure 10 doesn't tell us much because there are only two attributes, meaning that it is likely a lot of these duplicates are simply due to chance. On the other hand a closer look at the most frequently occurring duplicate rows in the medical dataset in figure 12 shows that most of these duplicates are duplicated 4 times. This suggests that these duplicate rows are possibly attributed to the dataset being a combination of four different databases of cardiovascular disease data. In order to address this we will need to combine the medical data with the patient data to examine if some of the duplicate medical data rows are random by chance, and then we will remove any remaining duplicates, keeping only one from each group.



Dataset statistics

Number of variables	2
Number of observations	1025
Missing cells	38
Missing cells (%)	1.9%
Duplicate rows	82
Duplicate rows (%)	8.0%
Total size in memory	12.1 KiB
Average record size in memory	12.1 B

Variable types

Numeric	1
Categorical	1

Duplicate rows

Most frequently occurring

	age	sex	# duplicates
50	58	1.0	46
48	57	1.0	44
53	59	1.0	43
36	52	1.0	38
41	54	1.0	37
18	44	1.0	29
46	56	1.0	28
34	51	1.0	27
55	60	1.0	24
58	62	0.0	24

Figures 9 & 10: Patient data duplicate record analysis

Dataset statistics

Number of variables	12
Number of observations	1025
Missing cells	103
Missing cells (%)	0.8%
Duplicate rows	298
Duplicate rows (%)	29.1%
Total size in memory	60.2 KiB
Average record size in memory	60.1 B

Variable types

Categorical	8
Numeric	4

Duplicate rows

Most frequently occurring

	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target	# duplicates
244	2	138	175.0	0	1	173	0.0	0.0	2	4	2	1	8
0	0	100	234.0	0	1	156	0.0	0.1	2	1	3	0	4
11	0	110	206.0	0	0	108	1.0	0.0	1	1	2	0	4
17	0	110	275.0	0	0	118	1.0	1.0	1	1	2	0	4
18	0	110	335.0	0	1	143	1.0	3.0	1	1	3	0	4
19	0	112	149.0	0	1	125	0.0	1.6	1	0	2	1	4
21	0	112	212.0	0	0	132	1.0	0.1	2	1	2	0	4
22	0	112	230.0	0	1	160	0.0	0.0	2	1	2	0	4
27	0	117	230.0	1	1	160	1.0	1.4	2	2	3	0	4
28	0	118	219.0	0	1	140	0.0	1.2	1	0	3	0	4

Figures 11 & 12: Medical data duplicate record analysis

3. Data Preparation

3.1 Data Selection

After taking a closer look at the available data for this project, we need to adequately choose which data to keep, alter or remove. Since we have such limited data we plan to take a more frugal approach to removing data, and instead will focus on refining the data and making the most of the scarce data available. One thing we will need to pay close attention to is the presence of duplicate records within the medical dataset as identified in section 2.4.

3.2 Data Integration

Given our data is split into two separate datasets, one containing patient data and the other having medical data, before we do too much data cleaning the first step is to combine these two datasets. Fortunately this process is simplified by the presence of the matching id variable between the two datasets. We start the merging process with a patient dataset containing three fields and a medical dataset containing 13 fields as seen in figure 5, merging the two results in a dataset containing one shared id attribute, and 14 other relevant attributes to our analysis. In the merging process I chose to remove the id variable as this

adds no value to the analysis from here forward, giving us the preliminary dataset for the data mining process with 14 total variables, seen in figure 14.

```
Patient data shape (Rows, Columns): (1025, 3)
Medical data shape (Rows, Columns): (1025, 13)
Merged data shape (Rows, Columns): (1025, 14)
```

Figure 13: Dataset shapes before and after integration

summary	count	mean	stddev	min	max
age	1025	54.43414634146342	9.072290233244278	29	77
sex	987	0.7001013171225937	0.45844563856310544	0	1
cp	1025	0.9424390243902439	1.029640743645865	0	3
trestbps	1025	131.61170731707318	17.516718005376408	94	200
chol	982	246.01221995926682	51.7080514274769	126	564
fbs	1025	0.14926829268292682	0.35652668972715756	0	1
restecg	1025	0.5297560975609756	0.5278775668748918	0	2
thalach	1025	149.11414634146342	23.00572374597721	71	202
exang	965	0.3378238341968912	0.4732134233484412	0	1
oldpeak	1025	1.0715121951219524	1.1750532551501767	0.0	6.2
slope	1025	1.3853658536585365	0.6177552671745918	0	2
ca	1025	0.7541463414634146	1.0307976650242825	0	4
thal	1025	2.32390243902439	0.6206602380510303	0	3
target	1025	0.5131707317073171	0.5000704980788011	0	1

Figure 14: Dataset after data integration

3.3 Cleaning the Data

With the data grouped together as as seen in 3.2, now we must clean the data so that it is fit for the purpose of data mining. Cleaning this dataset will be split into three main sections, dealing with duplicate rows, handling null values in the dataset, and handling extreme values and outliers. Each of these tasks come with their own challenges and questions which must be appropriately addressed in order to get the most out of this already small dataset. Additionally in order to ensure the test data has no influence on the training data, once the duplicate values have been removed we will split the data into a train and test set, on which only the train set will be used for data cleaning and imputation.

3.3.1 Duplicate values

An initial analysis in section 2.4 indicated the presence of multiple duplicate values within the medical dataset which is a key concern going forward with cleaning the data. A further

analysis into this anomaly showed that the initial medical dataset contained only 398 unique observations as seen in figure 16, cutting down our 1025 record dataset by over 60%.

```
medical_unique_rows = medical.drop("id").distinct().count()
print("Medical data unique rows:", medical_unique_rows)
```

```
Medical data unique rows: 398
```

Figures 15 & 16: Unique records in medical_data.csv

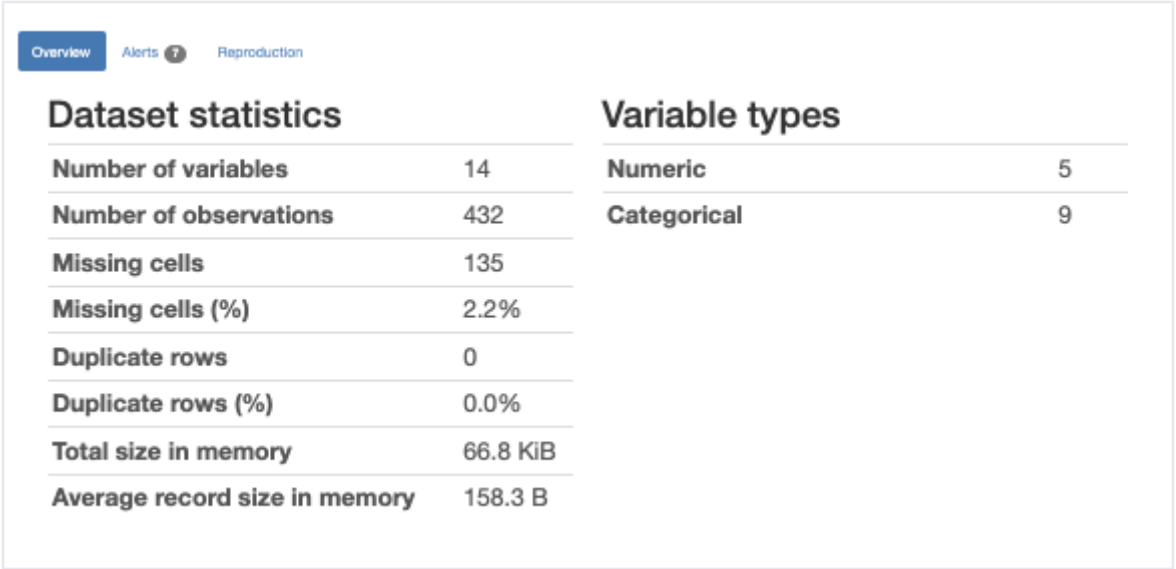
I chose to leave the removal of these duplicate values until after I had merged the two raw datasets together in order to assess whether the addition of sex and age variables would demonstrate that some of these duplicates were duplicates simply by chance. After merging the datasets a duplicate observation analysis was performed once more now grouped using all available variables as shown in figure 18.

```
heart = heart.dropDuplicates()
combined_unique_rows = heart.count()
print("Combined dataset unique rows:", combined_unique_rows)
```

```
Combined dataset unique rows: 432
```

Figures 17 & 18: Unique records in merged dataset

This revealed that in fact there were 34 observations which were duplicates simply by chance, and that the dataset now consisted of 432 unique data points which can be used for our analysis. With this analysis performed, the dataset was reduced by keeping only the first datapoint within each group, taking our dataset down to 432 records, and 14 features. The resulting dataset (excluding the target variable) at this stage can be seen in figures 19,20 & 21.

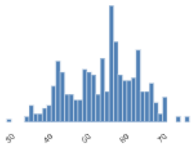


Select Columns

age

Real number (ℝ)

Distinct	41	Minimum	29
Distinct (%)	9.5%	Maximum	77
Missing	0	Zeros	0
Missing (%)	0.0%	Zeros (%)	0.0%
Infinite	0	Negative	0
Infinite (%)	0.0%	Negative (%)	0.0%
Mean	54.483795	Memory size	6.8 KiB



More details

sex

Categorical

Distinct	2
Distinct (%)	0.5%
Missing	36
Missing (%)	8.3%
Memory size	6.8 KiB

More details

cp

Categorical

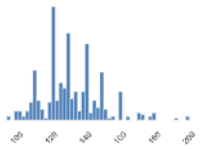
Distinct	4
Distinct (%)	0.9%
Missing	0
Missing (%)	0.0%
Memory size	6.8 KiB

More details

trestbps

Real number (ℝ)

Distinct	49	Minimum	94
Distinct (%)	11.3%	Maximum	200
Missing	0	Zeros	0
Missing (%)	0.0%	Zeros (%)	0.0%
Infinite	0	Negative	0
Infinite (%)	0.0%	Negative (%)	0.0%
Mean	131.2037	Memory size	6.8 KiB

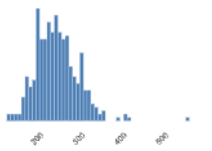


More details

chol

Real number (ℝ)

Distinct	152	Minimum	126
Distinct (%)	39.1%	Maximum	564
Missing	43	Zeros	0
Missing (%)	10.0%	Zeros (%)	0.0%
Infinite	0	Negative	0
Infinite (%)	0.0%	Negative (%)	0.0%
Mean	246.38817	Memory size	6.8 KiB



More details

fbs

Categorical

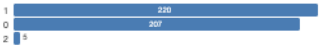
Distinct	2
Distinct (%)	0.5%
Missing	0
Missing (%)	0.0%
Memory size	6.8 KiB

More details

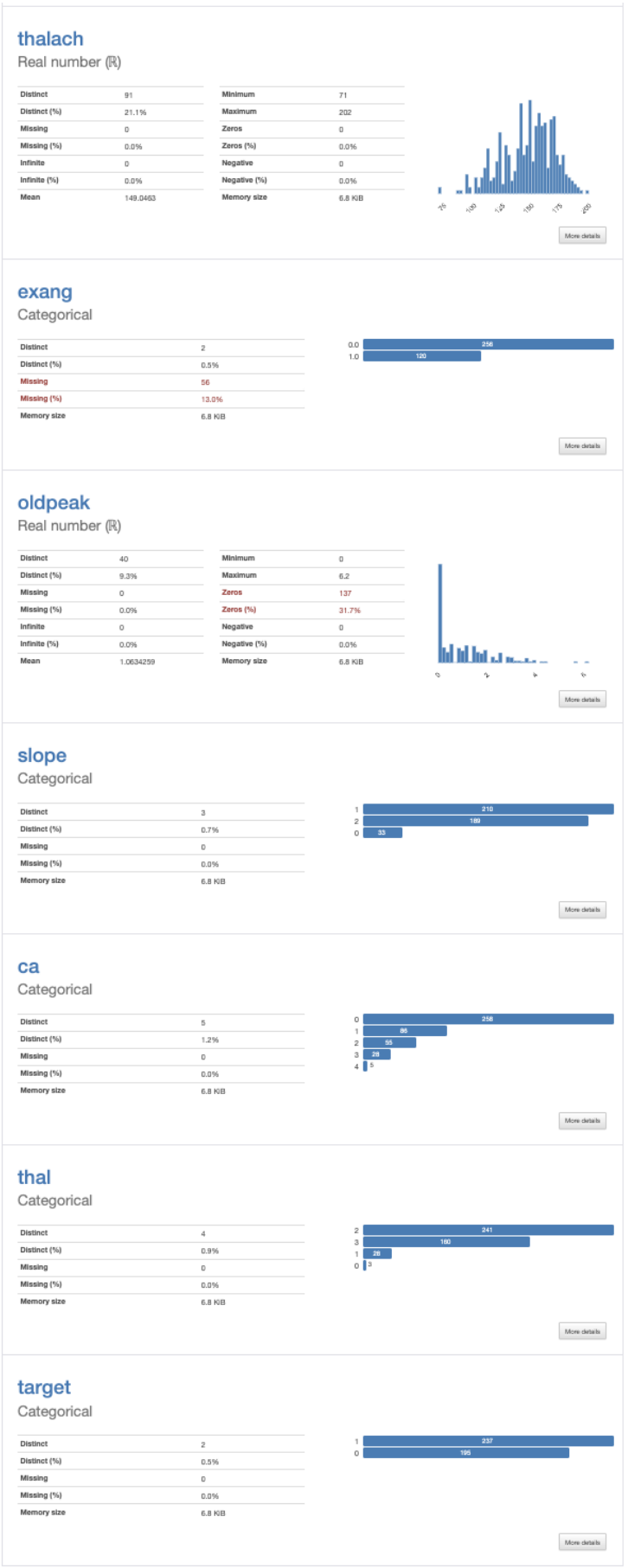
restecg

Categorical

Distinct	3
Distinct (%)	0.7%
Missing	0
Missing (%)	0.0%
Memory size	6.8 KiB



More details



Figures 19, 20 & 21: Raw data after integrating both datasets

3.3.2 Splitting the Dataset Into a Train and Test Set

In order to avoid using values from the test set to decide on what constitutes an outlier, or to be taken into consideration for imputation of missing values in the train dataset, before performing these data cleaning steps we will split the data into a train and test set as shown in figure 22. I chose a train/test data split of 80/20 in order to allow for sufficient samples in the test set to make valid predictions later on, and also to get as much out of the small dataset as possible. The sizes of the resulting datasets can be seen in figure 23.

```
train_ratio = 0.8

heart = heart.select("*").orderBy(F.rand()) #Randomize row order

train, test = heart.randomSplit([train_ratio, 1.0 - train_ratio], seed=722)

print("Train data observations:", train.count())
print("Test data observations:", test.count())
```

Figure 22: Splitting data into train and test datasets

```
Train data observations: 336
Test data observations: 83
```

Figure 23: Train/test set total observations

3.3.3 Handling Outliers and Extreme Values

3.3.3.1 Removing Undocumented Classes & Outlier Classes

An interesting observation made from the dataset is that there are some classes within the ca and thal variables which are not documented on the dataset page on kaggle. We can see from figure 24 that ca is documented to have classes from 0-3, and thal is documented to only have classes 1,2 and 3, however in figure 25 we can see there that thal has a 0 class and ca has a 4 class. In order to ensure only documented data is used in the data mining these classes will be removed from the dataset entirely. Additionally the restecg variable has 3 different values of 0,1 and 2, however given there are only 5 observations with a value of 2, it is best to remove them from the dataset to avoid them being overly influential in our models. The process of removing these observations and the value counts after their removal can be seen in figures 26 & 27.

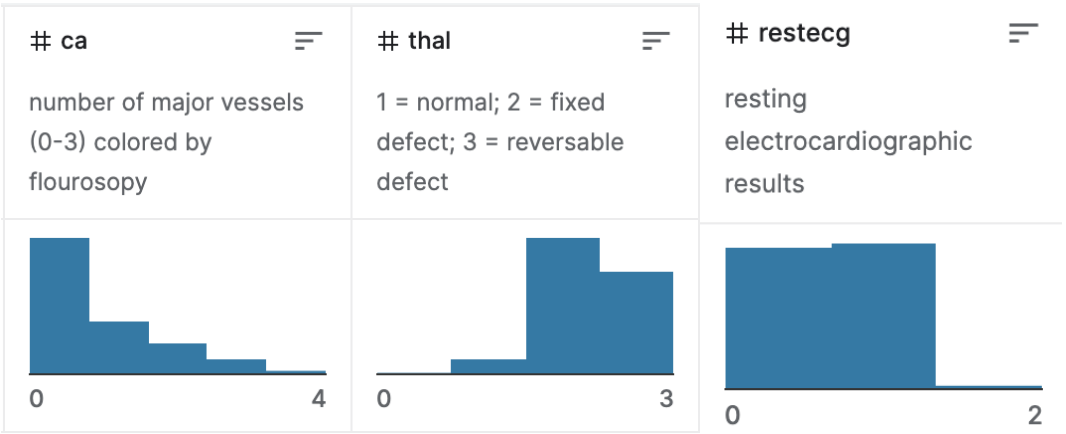


Figure 24: Dataset description of the ca, thal and restecg variables

```
Counts for 'thal' column:
+----+-----+
|thal|count|
+----+-----+
|   2|  241|
|   3|  160|
|   1|   28|
|   0|    3|
+----+-----+


Counts for 'ca' column:
+---+-----+
| ca|count|
+---+-----+
|  0|  258|
|  1|   86|
|  2|   55|
|  3|   28|
|  4|    5|
+---+-----+

Counts for 'restecg' column:
+-------+-----+
|restecg|count|
+-------+-----+
|      1|  220|
|      0|  207|
|      2|    5|
+-------+-----+
```

Figure 25: Value counts of thal, ca & restecg variables in whole dataset

```
heart = heart.filter(~((col("thal") == 0) | (col("ca") == 4) | (col("restecg") == 2)))
```

Figure 26: Process of removing undocumented classes & outlier classes from the dataset

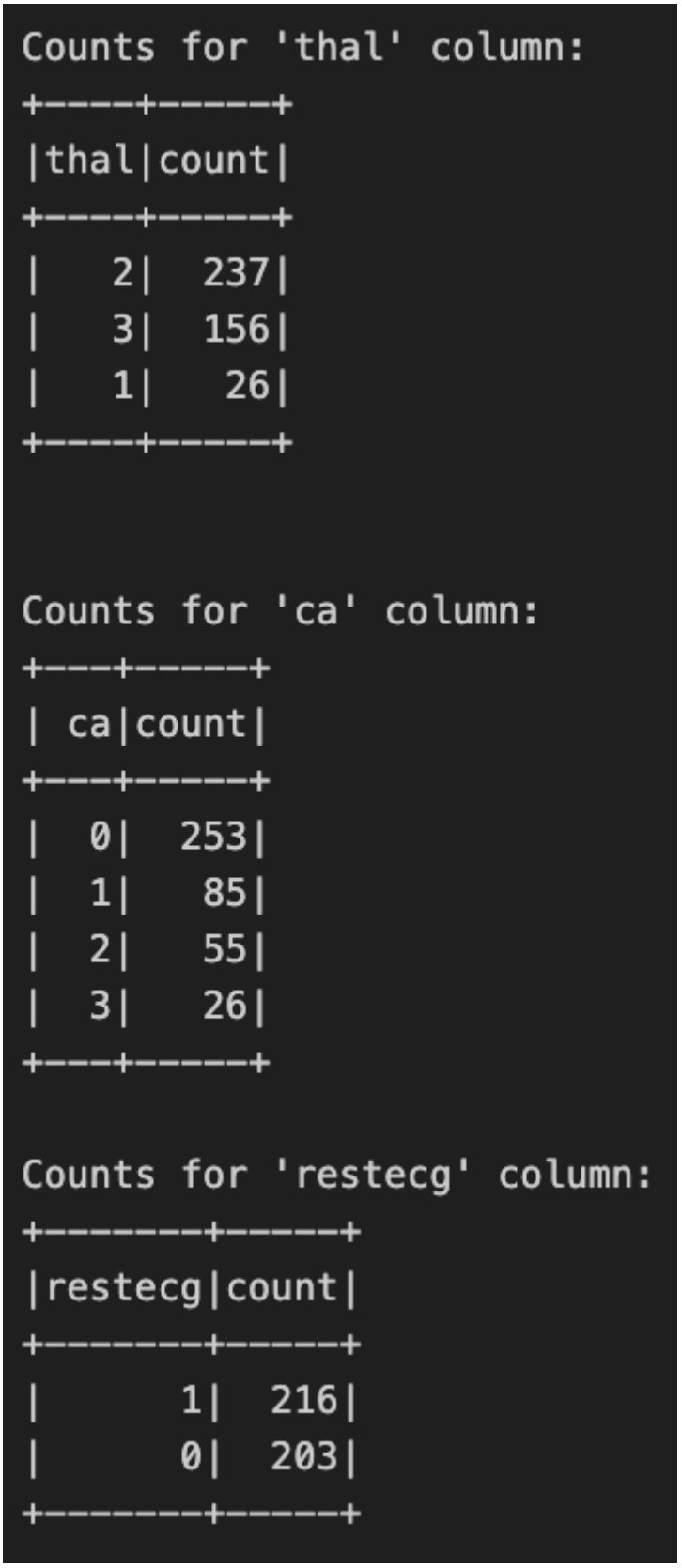


Figure 27: Value counts of thal, ca & restecg after class removal

3.3.3.2 Handling Outliers

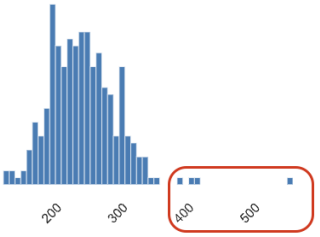
With the datasets merged and duplicates removed, less substantial issues in the dataset can be addressed such as outliers and extreme values. An examination of the train dataset after integration shows that outliers are present in two variables being chol and oldpeak as shown in figures 28, 29 & 30. Since there are so few outliers, I decided to simply remove them from the dataset to ensure the already small train set was not influenced by a handful of outliers.

chol

Real number (ℝ)

MISSING

Distinct	143	**Minimum**	126
Distinct (%)	47.5%	**Maximum**	564
Missing	35	**Zeros**	0
Missing (%)	10.4%	**Zeros (%)**	0.0%
Infinite	0	**Negative**	0
Infinite (%)	0.0%	**Negative (%)**	0.0%
Mean	246.40532	**Memory size**	2.8 KiB

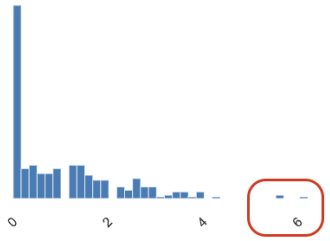


oldpeak

Real number (ℝ)

HIGH CORRELATION ZEROS

Distinct	37	**Minimum**	0
Distinct (%)	11.0%	**Maximum**	6.2
Missing	0	**Zeros**	109
Missing (%)	0.0%	**Zeros (%)**	32.4%
Infinite	0	**Negative**	0
Infinite (%)	0.0%	**Negative (%)**	0.0%
Mean	1.0333333	**Memory size**	2.8 KiB



Figures 28, 29 & 30: Outliers identified in train set

```
train = train.filter(~(col("oldpeak") > 5))
train = train.filter("chol < 380 OR ISNULL(chol)")
```

Figure 31: Process of removing outliers

```
Train set shape with outliers (Rows, Columns): (336, 18)
Train set shape without outliers (Rows, Columns): (329, 18)
```

Figure 32: Train set shape before and after outlier removal

We can see from figure 32 that the removal of these outliers has diminished the size of our train set slightly by 7 observations, but this diminishment is worth it given the removal of over influential points in the training set.

3.3.4 Handling Missing Values

With the outliers and extreme values handled the last step of the data cleaning process was to handle missing values in the dataset. There are three variables which have incomplete records as shown in figure 33, each of these need to be handled appropriately and differently depending on their relationships to other variables.

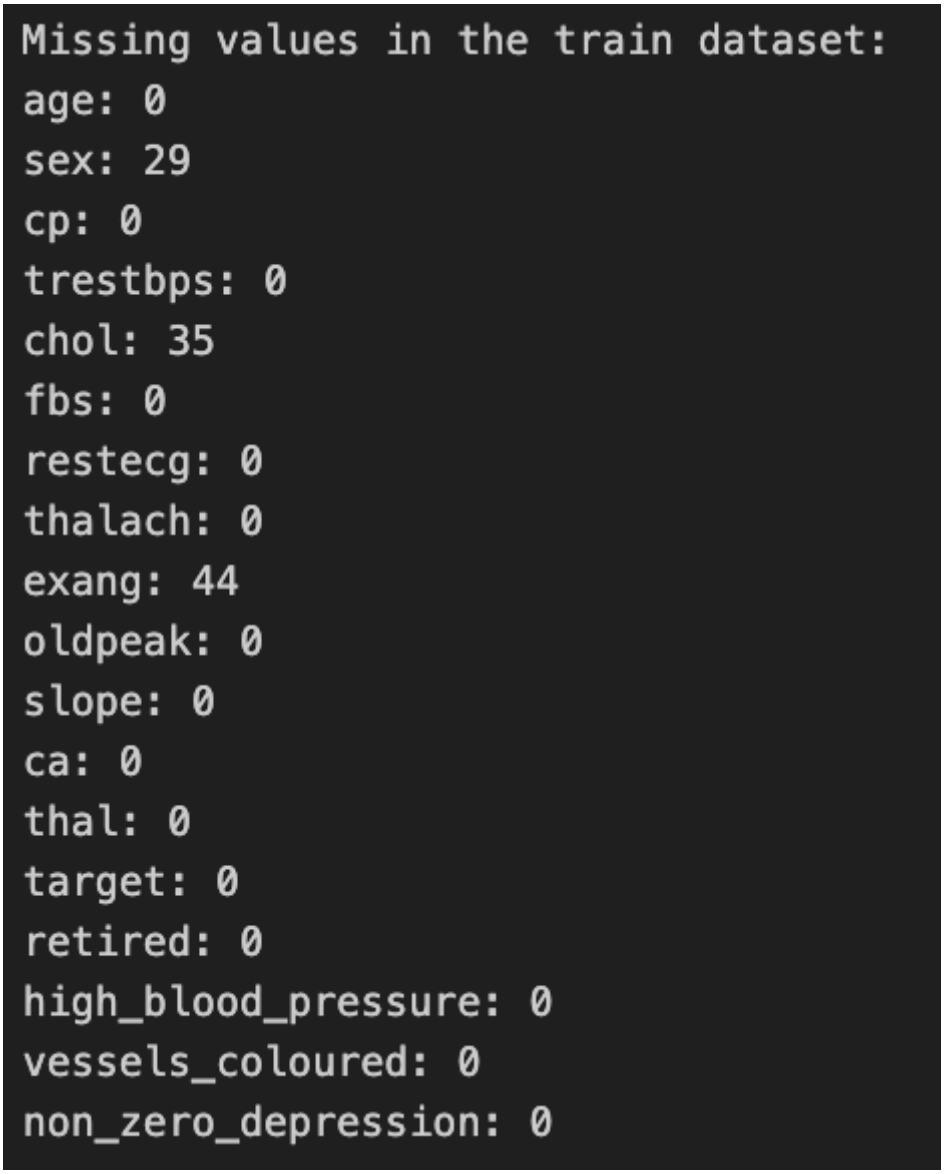


Figure 33: Current state of missing values in the dataset

Since the quantity of missing values is so small for each of these variables (largest proportion is exang at 44/329, just over 13%), the preferable method would be to develop a model to generate the missing values given there is adequate correlation between the

variable with missing values and the attributes used in the model. If there is no apparent correlation between variables however, another approach will need to be considered. To examine these correlations a heatmap was fit for the variables with missing values as shown in figure 34.

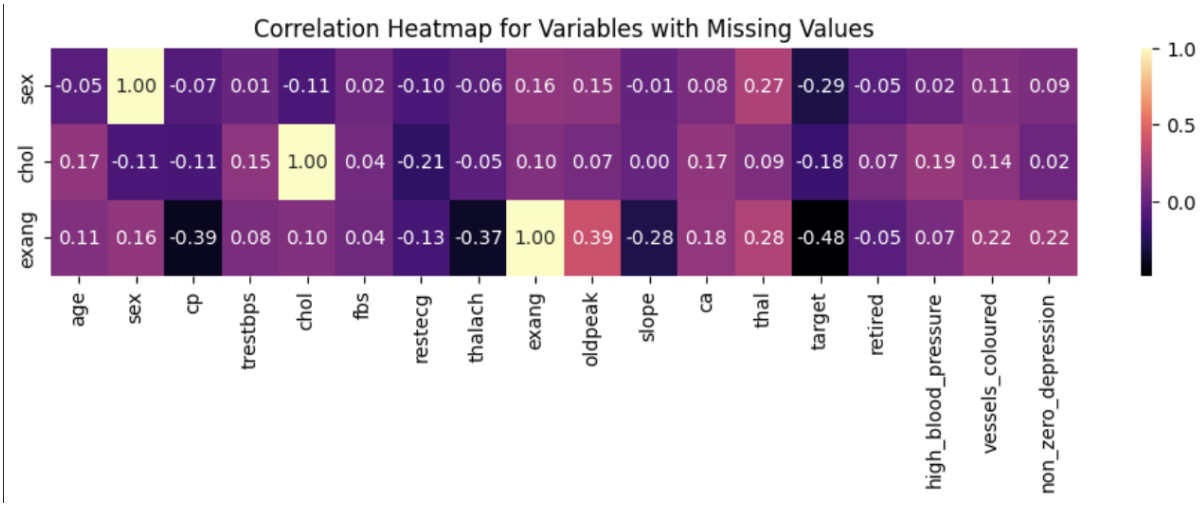


Figure 34: Correlation heatmap for exang, chol & sex with the rest of the dataset

3.3.4.1 Handling Missing Chol Values

An investigation into the correlations with the Chol variables shown in figure 34 demonstrated a lack of meaningful correlations with the chol variable. The only attributes with a semblance of a relationship with chol are age, restecg and high_blood_pressure (a feature created in section 3.4), however even these relationships are extremely weak, so we will opt to impute the chol variable using its median value rather than creating an imputation model. The implementation of this imputation can be seen in figure 35.

```
chol_median = train.approxQuantile("chol", [0.5], 0.001)[0]
train = train.fillna({"chol": chol_median})
test = test.fillna({"chol": chol_median})
```

Figure 35: Imputing missing chol values using its median value

3.3.4.2 Handling Missing sex Values

Similarly to the chol variable, an investigation into the correlations between sex and the other predictors available within the train set showed an overall lack of relationships. Given the highest level of correlation between sex and another predictor was only 0.29, and there was only one other variable with a correlation greater than 0.2, an imputation model would add very little value above the mode. Because of this the imputation method chosen for the sex variable was the mode value, where each missing sex value was replaced with the most common sex value, which was 1 (male). The implementation of this imputation can be seen in figure 36.

```
sex_mode = train.groupBy("sex").count().orderBy(col("count").desc()).first()["sex"]
train = train.fillna({"chol": chol_median, "sex": sex_mode})
test = test.fillna({"chol": chol_median, "sex": sex_mode})
```

Figure 36: Imputing missing sex values using its mode value

3.3.4.3 Handling Missing exang Values

An investigation into the correlations that the exang variable had with other variables in the train dataset demonstrated far more relationships to other variables within the dataset than was seen by the chol and sex attributes. We found that the exang variable had a correlation value with magnitude greater than 0.2 with 8 different variables, with 4 of those having correlation greater than 0.3. Although these correlation values on their own are still quite low, correlation with so many variables gives warrant to test imputation models.
To impute the exang variable using an imputation model, three models were tested using 5 fold cross validation to examine the accuracy of the imputation. The three models chosen to test were Logistic Regression, SVM (linear SVM), and random forest. The implementation of these preliminary imputation models and their results can be seen in figures 37 and 38.

```
from pyspark.ml.classification import RandomForestClassifier, LinearSVC, LogisticRegression
from pyspark.ml.evaluation import BinaryClassificationEvaluator
from pyspark.ml.tuning import ParamGridBuilder, CrossValidator
from pyspark.ml import Pipeline
from pyspark.ml.feature import VectorAssembler,StringIndexer

no_na_vals = train.na.drop()

feature_columns = [colname for colname in train.columns if colname != "exang"]

assembler = VectorAssembler(inputCols=feature_columns, outputCol="features")
indexer = StringIndexer(inputCol="exang", outputCol="label")
evaluator = BinaryClassificationEvaluator()

models = [RandomForestClassifier(featuresCol='features', labelCol="label"),
          LinearSVC(featuresCol='features', labelCol="label"),
          LogisticRegression(featuresCol='features', labelCol="label")]

model_results = []

folds = 5

for model in models: #Perform CV for each model
    pipeline = Pipeline(stages=[assembler, indexer, model])
    grid = ParamGridBuilder().build()

    crossval = CrossValidator(estimator=pipeline,
                              evaluator=evaluator,
                              estimatorParamMaps=grid,
                              numFolds=folds,
                              seed=722,
                              parallelism=20)

    cvModel = crossval.fit(no_na_vals)
    accuracy = sum(cvModel.avgMetrics) / len(cvModel.avgMetrics)
    model_results.append((model, cvModel, accuracy))
```

Figure 37: Implementation of imputation models for the exang variable

```
RandomForestClassifier accuracy: 0.842
LinearSVC accuracy: 0.821
LogisticRegression accuracy: 0.819
```

Figure 38: Accuracy of imputation models fit with 5-fold cross validation

Given the random forest imputation model performed so well achieving 84.2% accuracy, I chose to use this model to impute the missing exang values. The random forest was then used to impute the missing values as shown in figure 39.

```
rf = model_results[0][1].bestModel

no_na_vals = train.na.drop()
na_rows = train.filter("exang is NULL")
predicted = rf.transform(na_rows).select(na_rows.columns + ["prediction"])
predicted = predicted.withColumn("exang", predicted.prediction).select(na_rows.columns)
train = no_na_vals.unionByName(predicted)

test_no_na_vals = test.na.drop()
test_na_rows = test.filter("exang is NULL")
predicted = rf.transform(test_na_rows).select(test_na_rows.columns + ["prediction"])
predicted = predicted.withColumn("exang", predicted.prediction).select(test_na_rows.columns)
test = test_no_na_vals.unionByName(predicted)
```

Figure 39: Using random forest to impute missing exang values

3.4 Feature Creation

Before the data was split into a train and test set, some extra features were created in order to attempt to capture some interesting patterns which are not explicit from the data as it stands. Particularly these new features include the four flag variables retired, high_blood_pressure, vessels_coloured & non_zero_depression, as shown in figure 40.

```
heart = heart.withColumn("retired", when(col("age") < 65, 0).otherwise(1))
heart = heart.withColumn("high_blood_pressure", when(col("trestbps") <= 120, 0).otherwise(1))
heart = heart.withColumn("vessels_coloured", when(col("ca") == 0, 0).otherwise(1))
heart = heart.withColumn("non_zero_depression", when(col("oldpeak") == 0, 0).otherwise(1))
```

Figure 40: Generation of four additional variables

3.4.1 Retired Attribute

The first variable created was “retired”, which is a flag variable representing whether or not the patient was above or below the NZ retirement age of 65. This variable was created to examine whether the lack of activity and stress associated with retirement plays an additional role in the development of cardiovascular disease compared to age on its own. The formulation of this variable is shown in figure 40.

3.4.2 High Blood Pressure Attribute

The second variable created for this datamining project was “high_blood_pressure” a flag variable representing whether a blood pressure of greater than 120mm Hg was present in the patient. 120mm Hg and below is widely regarded as the standard for healthy blood pressure. The Canadian Journal of Cardiology states “intensive BP reduction to target a systolic BP (SBP) 120 mm Hg should be considered to lower the risk of cardiovascular events” (Harris et al., 2016). As a result this variable aims to determine if there is additional information given by patients specifically being below the “healthy” blood pressure threshold. The formulation of this attribute is shown in figure 40.

3.4.3 Vessels Coloured Attribute

The third variable created in order to capture additional information within the dataset was the vessels_coloured attribute. The vessels_coloured variable is a flag variable demonstrating whether or not there was more than 0 blood vessels coloured by Fluoroscopy. An investigation into the initial class values for the ca variable in figure 41, shows that the proportion of patients with 0 blood vessels coloured who had cardiovascular disease is much higher than that of those who didn't have cardiovascular disease. However, the opposite is true for all coloured blood vessel counts greater than 0. Since there is such a sharp change in cardiovascular disease expectation between patients with 0 blood vessels coloured, and patients with at least 1 blood vessel coloured by fluoroscopy, the vessels_coloured variable was introduced to capture this pattern, which it appears to do as seen in figure 42.

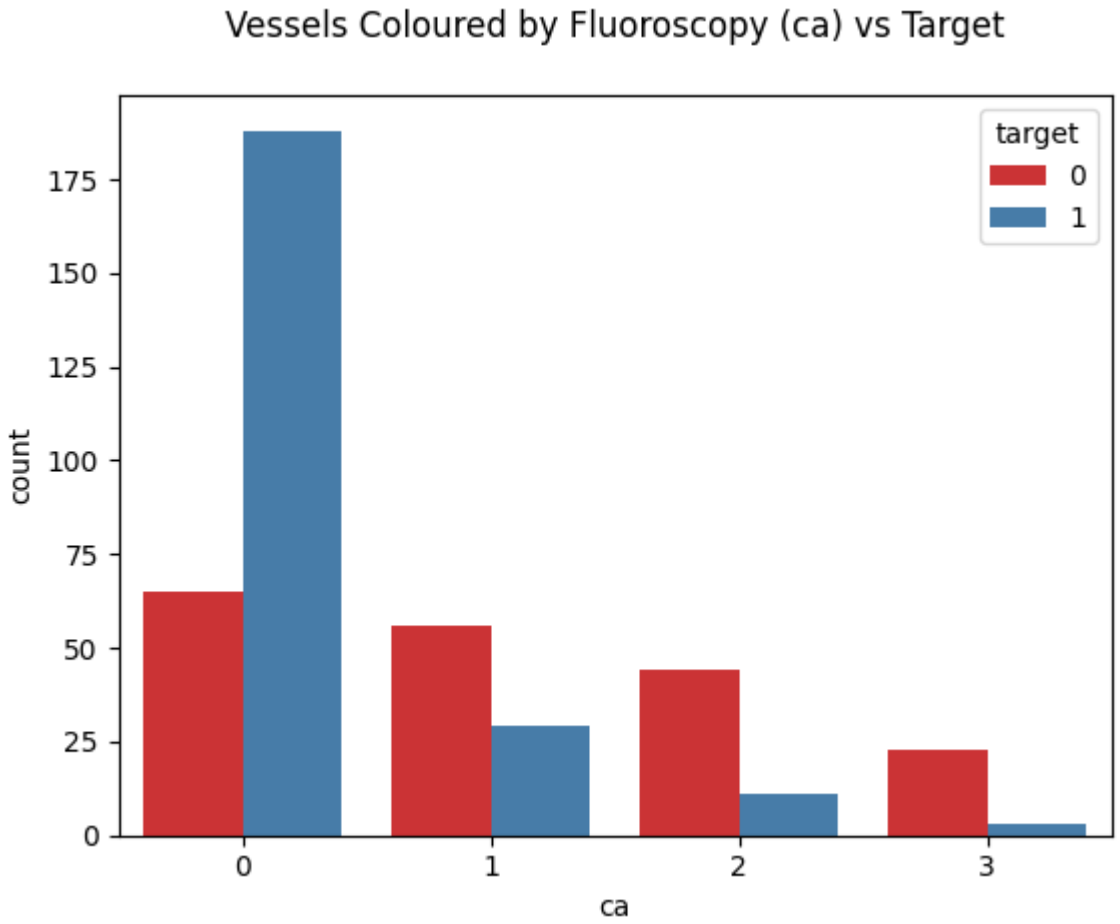


Figure 41: Investigation into the ca variable

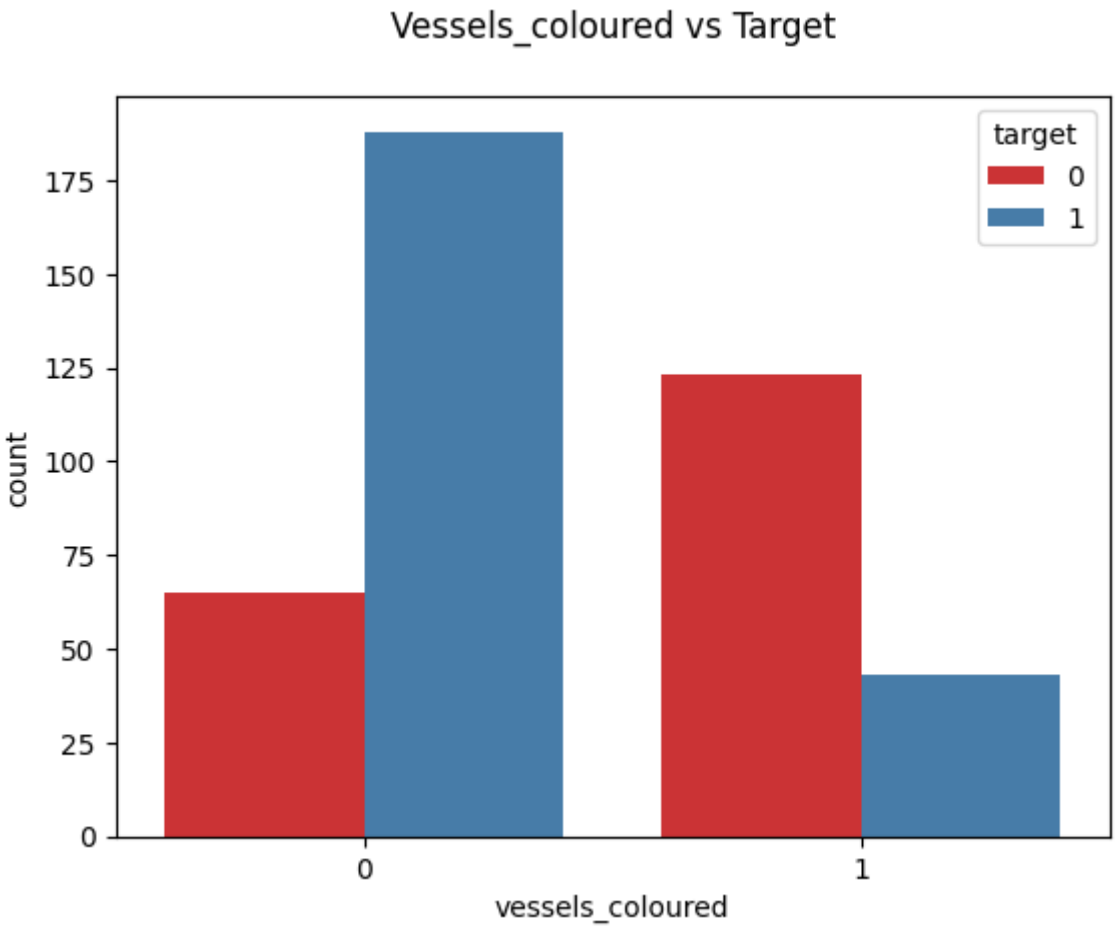


Figure 42: Investigation into vessels_coloured variable

3.4.4 Non Zero Depression Attribute

One interesting observation in regards to the oldpeak variable is the fact that there are a huge amount of 0 values (exactly 137 as seen in figure 21). Since there is such a clear distinction between patients having no ST depression induced by exercise relative to rest, and having some kind of ST depression, I introduced a non_zero_depression attribute to encompass this distinction. The non_zero_depression variable is a flag variable where it has the value 1 if the patient's oldpeak value is greater than 0, and 0 otherwise. The non_zero_depression attribute can be seen in figure 44.

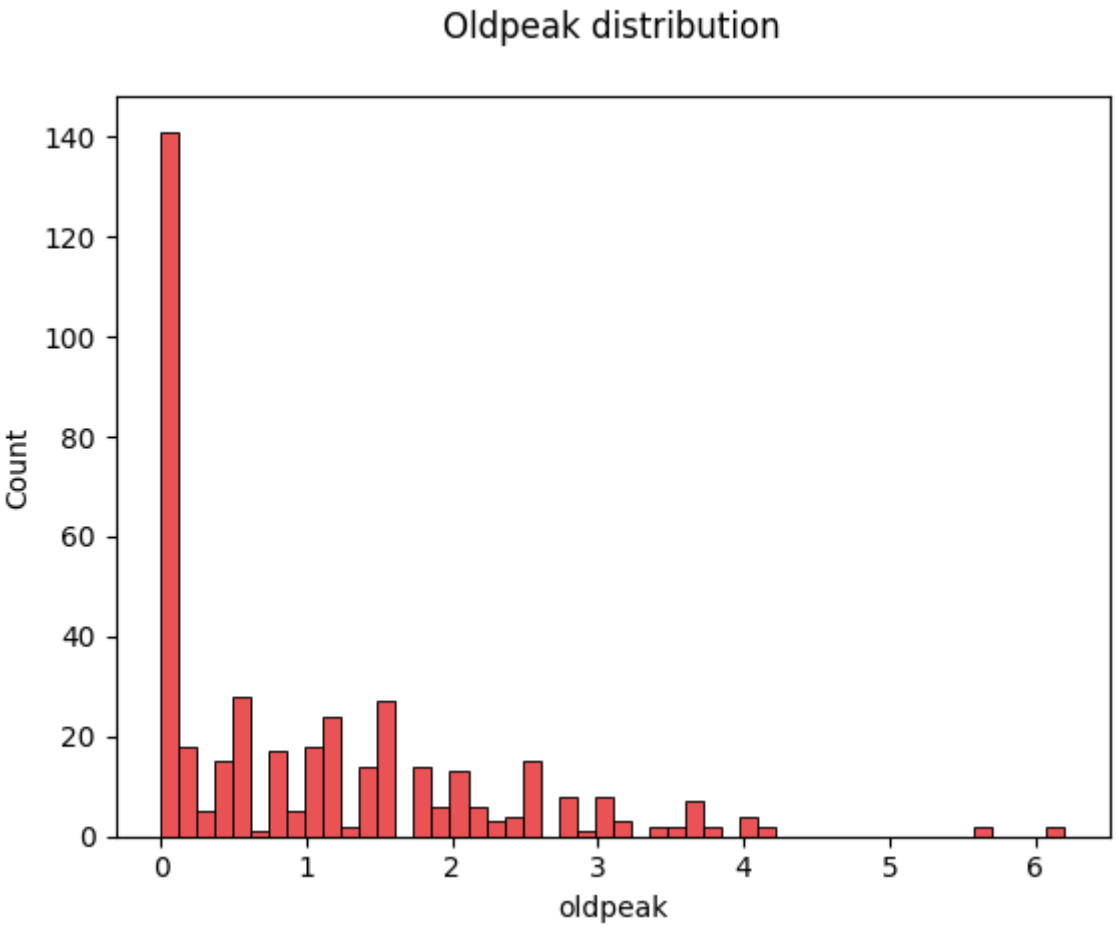


Figure 43: Investigation of the oldpeak variable distribution

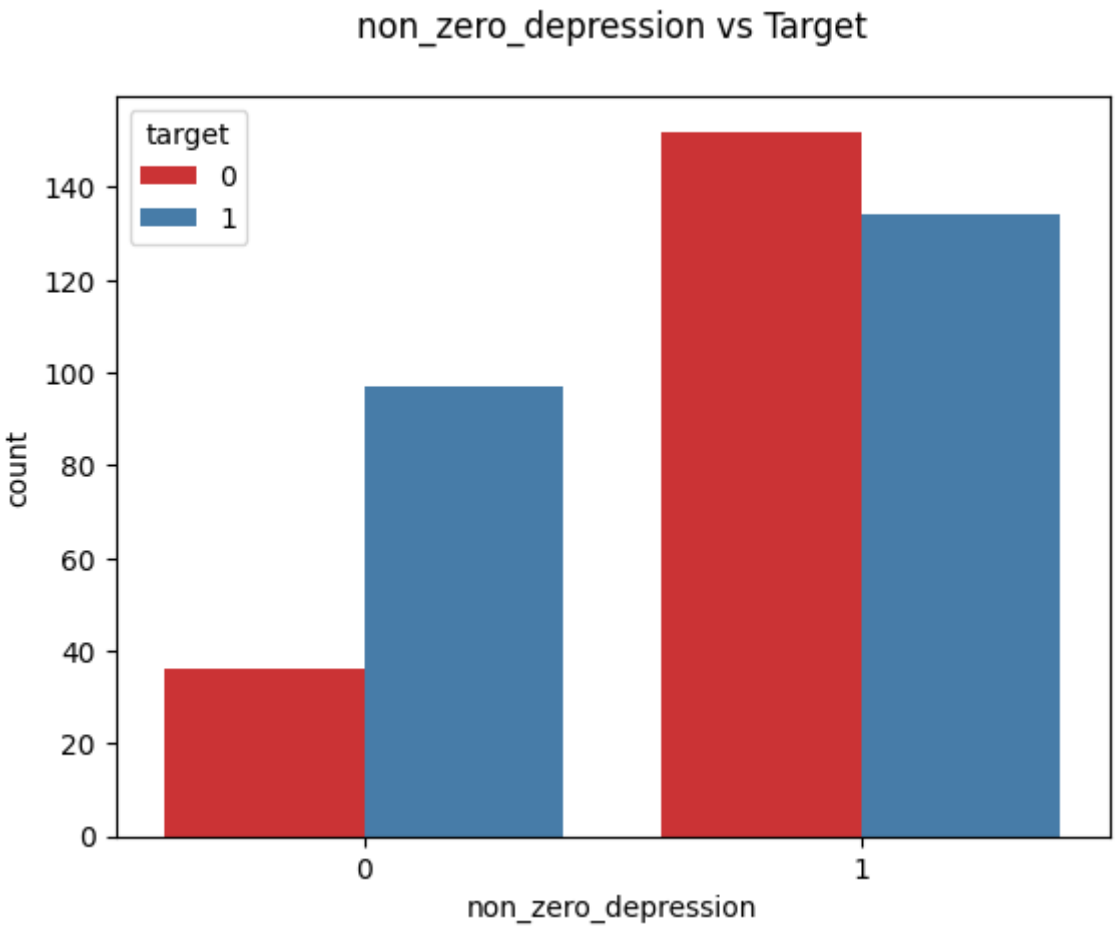


Figure 44: Investigation of the non_zero_depression attribute

3.5 Reformatting Data

With the data completely clean and with all relevant attributes added, the last step before moving on to data transformation is to ensure that all variables are in the correct format. The most important step in reformatting the data is to ensure that the results from the future analysis are fully understandable and are interpreted correctly by the models implemented. In light of this, firstly I will rename the variables to more interpretable names as seen in figure 45, the changes of which are seen in figures 46 & 47, and secondly I will turn categorical variables with multiple values into dummy flag variables. By changing categorical variables into dummy flag variables it ensures that models are able to select only the important classes from these variables rather than adding coefficients to account for unimportant variables. In order to turn these categorical variables into dummies first I change the variables numeric values (0-3, etc.) into strings which represent the value, and then I call pandas .get_dummies() function, with drop_first set to False in order to create a new variable for each individual class type of each categorical variable. The process of changing the categorical variables chest_pain_type, st_slope_type and thalassemia_type (formerly cp, slope and thal) into dummy variables can be seen in figures 48 & 49.

```
from pyspark.sql.functions import lit
from pprint import pprint

train = train.withColumn("partition", lit("train"))
test = test.withColumn("partition", lit("test"))

print("Train dataset shape (Rows, Columns):", (train.count(), len(train.columns)))
print("Test dataset shape (Rows, Columns):", (test.count(), len(test.columns)))

#Combine train and test
heart = train.unionByName(test)

print("Dataset shape (Rows, Columns):", (heart.count(), len(heart.columns)))

print("Current column names:")
pprint(heart.columns)

print()

#Update colnames to be more descriptive
new_colnames = ['age',
                'sex',
                'chest_pain_type',
                'resting_blood_pressure',
                'cholesterol',
                'fasting_blood_sugar',
                'rest_ecg_type',
                'max_heart_rate_achieved',
                'exercise_induced_angina',
                'st_depression',
                'st_slope_type',
                'num_major_vessels',
                'thalassemia_type',
                'target'] + list(heart.columns[14:])

for i in range(len(new_colnames)):
    heart = heart.withColumnRenamed(heart.columns[i], new_colnames[i])

print("New column names:")
pprint(heart.columns)
```

Figure 45: Changing column names to more informative versions

Figure 46: Current column names of the heart dataset

```
New column names:
['age',
 'sex',
 'chest_pain_type',
 'resting_blood_pressure',
 'cholesterol',
 'fasting_blood_sugar',
 'rest_ecg_type',
 'max_heart_rate_achieved',
 'exercise_induced_angina',
 'st_depression',
 'st_slope_type',
 'num_major_vessels',
 'thalassemia_type',
 'target',
 'retired',
 'high_blood_pressure',
 'vessels_coloured',
 'non_zero_depression',
 'partition']
```

Figure 47: New column names for heart dataset

```
# Replace non-flag variables with descriptive levels
# chest_pain_type
heart = heart.withColumn("chest_pain_type", when(heart["chest_pain_type"] == 0, "asymptomatic")
                                            .when(heart["chest_pain_type"] == 1, "atypical_angina")
                                            .when(heart["chest_pain_type"] == 2, "non-anginal_pain")
                                            .when(heart["chest_pain_type"] == 3, "typical_angina"))

# st_slope_type
heart = heart.withColumn("st_slope_type", when(heart["st_slope_type"] == 0, "downsloping")
                                          .when(heart["st_slope_type"] == 1, "flat")
                                          .when(heart["st_slope_type"] == 2, "upsloping"))

# thalassemia_type
heart = heart.withColumn("thalassemia_type", when(heart["thalassemia_type"] == 1, "fixed_defect")
                                             .when(heart["thalassemia_type"] == 2, "normal")
                                             .when(heart["thalassemia_type"] == 3, "reversible_defect"))
```

Figure 48: Changing numeric encodings into relevant string representations

```
categorical_cols = ["chest_pain_type", "st_slope_type", "thalassemia_type"]

for column in categorical_cols:
    categories = heart.select(column).distinct().rdd.flatMap(lambda x: x).collect()
    for category in categories:
        heart = heart.withColumn(column + "_" + category, F.when(F.col(column) == category, 1).otherwise(0))

heart = heart.drop(*categorical_cols)
```

Figure 49: Replacing categorical variables with dummy variables

The final cleaned dataset at this point consists of 28 variables (including partition and target) consisting of newly created variables in section 3.4, original variables from the dataset, and dummified categorical variables. The state of the final dataset can be seen in figure 50.

```
Dataset shape with outliers (Rows, Columns): (412, 26)
root
 |-- age: integer (nullable = true)
 |-- sex: integer (nullable = false)
 |-- resting_blood_pressure: integer (nullable = true)
 |-- cholesterol: integer (nullable = true)
 |-- fasting_blood_sugar: integer (nullable = true)
 |-- rest_ecg_type: integer (nullable = true)
 |-- max_heart_rate_achieved: integer (nullable = true)
 |-- exercise_induced_angina: integer (nullable = true)
 |-- st_depression: float (nullable = true)
 |-- num_major_vessels: integer (nullable = true)
 |-- target: integer (nullable = true)
 |-- retired: integer (nullable = false)
 |-- high_blood_pressure: integer (nullable = false)
 |-- vessels_coloured: integer (nullable = false)
 |-- non_zero_depression: integer (nullable = false)
 |-- partition: string (nullable = false)
 |-- chest_pain_type_asymptomatic: integer (nullable = false)
 |-- chest_pain_type_non-anginal_pain: integer (nullable = false)
 |-- chest_pain_type_atypical_angina: integer (nullable = false)
 |-- chest_pain_type_typical_angina: integer (nullable = false)
 |-- st_slope_type_flat: integer (nullable = false)
 |-- st_slope_type_downsloping: integer (nullable = false)
 |-- st_slope_type_upsloping: integer (nullable = false)
 |-- thalassemia_type_reversible_defect: integer (nullable = false)
 |-- thalassemia_type_normal: integer (nullable = false)
 |-- thalassemia_type_fixed_defect: integer (nullable = false)
```

Figure 50: Reformatted data

4. Data Transformation

4.1 Data Reduction

Given we now have a cleaned dataset with 26 predictors, we will trim out all unnecessary variables from our dataset to simplify the data mining process. Given we have such low dimensionality data to begin with, we will take a cautious approach to data reduction.
In order to reduce this data, Lasso regression will be utilised in order to fit a regression model on the train dataset in order to determine if any variables have no impact on the prediction of the target variable (a coefficient of 0). The results from the lasso regression shown in figure 51 demonstrate that there are a few variables which have a very low impact on the prediction of cardiovascular disease in a patient, but most importantly it features 9 predictors which have no impact at all. The removal of the variables which have no impact on prediction gives the dataset shown in figure 52.

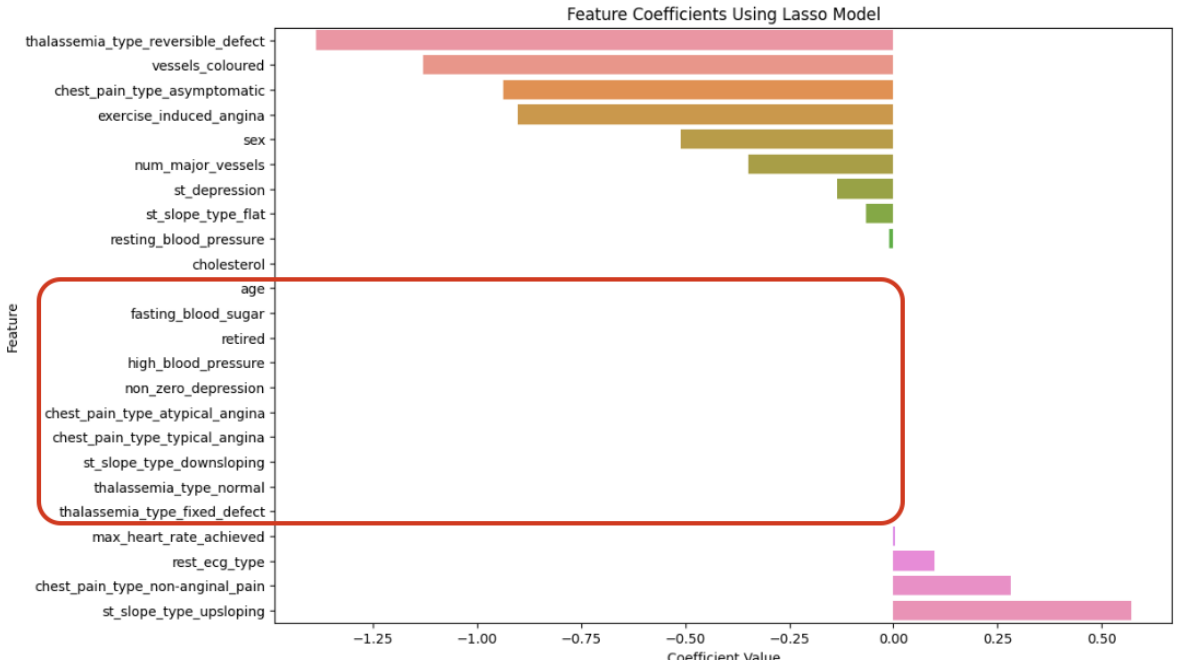


Figure 51: Coefficients of predictors for lasso regression model

```
root
 |-- sex: integer (nullable = true)
 |-- resting_blood_pressure: integer (nullable = true)
 |-- cholesterol: integer (nullable = true)
 |-- rest_ecg_type: integer (nullable = true)
 |-- max_heart_rate_achieved: integer (nullable = true)
 |-- exercise_induced_angina: integer (nullable = true)
 |-- st_depression: float (nullable = true)
 |-- num_major_vessels: integer (nullable = true)
 |-- target: integer (nullable = true)
 |-- vessels_coloured: integer (nullable = true)
 |-- chest_pain_type_asymptomatic: integer (nullable = true)
 |-- chest_pain_type_non-anginal_pain: integer (nullable = true)
 |-- st_slope_type_flat: integer (nullable = true)
 |-- st_slope_type_upsloping: integer (nullable = true)
 |-- thalassemia_type_reversible_defect: integer (nullable = true)
```

Figure 52: Data after feature selection

4.2 Data Projection

In terms of data projection there are two issues which need to be addressed, first there is the distribution of the numeric variables in the dataset, and the second is the unproportionate distribution of the target variable.

4.2.1 Transforming variables

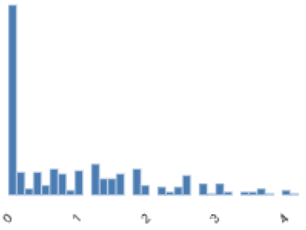
After the data reduction performed in section 4.1, there are four remaining continuous variables in the dataset which are resting_blood_pressure, cholesterol, max_heart_rate_achieved and st_depression. Most of these variables have a relatively standard normal distribution with the exception of st_depression which has a high skewness to the right, and resting_blood_pressure which has a slightly weaker skewness to the right also as seen in figure 53. Given this we will transform these variables in order to have a more appropriate distribution, and determine whether they are more useful for cardiovascular disease prediction than their uninfluenced counterparts.

st_depression

Real number (ℝ)

ZEROS

Distinct	35	**Minimum**	0
Distinct (%)	10.6%	**Maximum**	4.1999998
Missing	0	**Zeros**	109
Missing (%)	0.0%	**Zeros (%)**	33.1%
Infinite	0	**Negative**	0
Infinite (%)	0.0%	**Negative (%)**	0.0%
Mean	0.97933131	**Memory size**	0.0 B

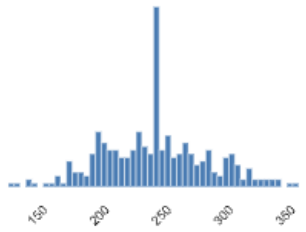


More details

cholesterol

Real number (ℝ)

Distinct	137	**Minimum**	126
Distinct (%)	41.6%	**Maximum**	360
Missing	0	**Zeros**	0
Missing (%)	0.0%	**Zeros (%)**	0.0%
Infinite	0	**Negative**	0
Infinite (%)	0.0%	**Negative (%)**	0.0%
Mean	244.05167	**Memory size**	0.0 B

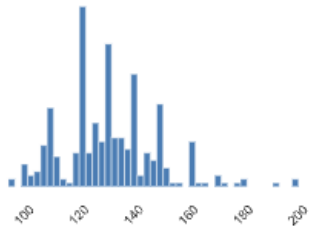


More details

resting_blood_pressure

Real number (ℝ)

Distinct	45	**Minimum**	94
Distinct (%)	13.7%	**Maximum**	200
Missing	0	**Zeros**	0
Missing (%)	0.0%	**Zeros (%)**	0.0%
Infinite	0	**Negative**	0
Infinite (%)	0.0%	**Negative (%)**	0.0%
Mean	130.89058	**Memory size**	0.0 B

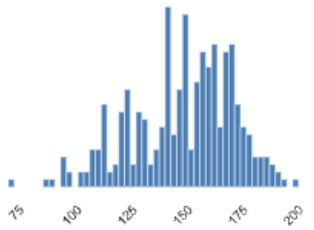


More details

max_heart_rate_achieved

Real number (ℝ)

Distinct	86	**Minimum**	71
Distinct (%)	26.1%	**Maximum**	202
Missing	0	**Zeros**	0
Missing (%)	0.0%	**Zeros (%)**	0.0%
Infinite	0	**Negative**	0
Infinite (%)	0.0%	**Negative (%)**	0.0%
Mean	149.93921	**Memory size**	0.0 B



More details

Figure 53: Analysis of continuous variables

One thing that must be taken into consideration when choosing the transformation for the st_depression variable however is the high presence of 0 values within the attribute. The presence of these 0 values takes a lot of transformation values off of the table such as log transformations and inverse transformations due to their inability to handle 0 values. With these limitations in mind I chose to apply a square root transformation to the st_depression variable as shown in figure 54. The square root transformation for the st_depression attribute allows for the presence of 0 values, whilst also removing the right skew present in the remainder of the st_depression values.
For the resting_blood_pressure attribute I chose a natural log transformation in order to minimise the right skew present in the variable. The log transformation applied to the resting_blood_pressure attribute visibly fixes the skewness present in the original distribution, shifting it to a more normal distribution as seen in figure 55.

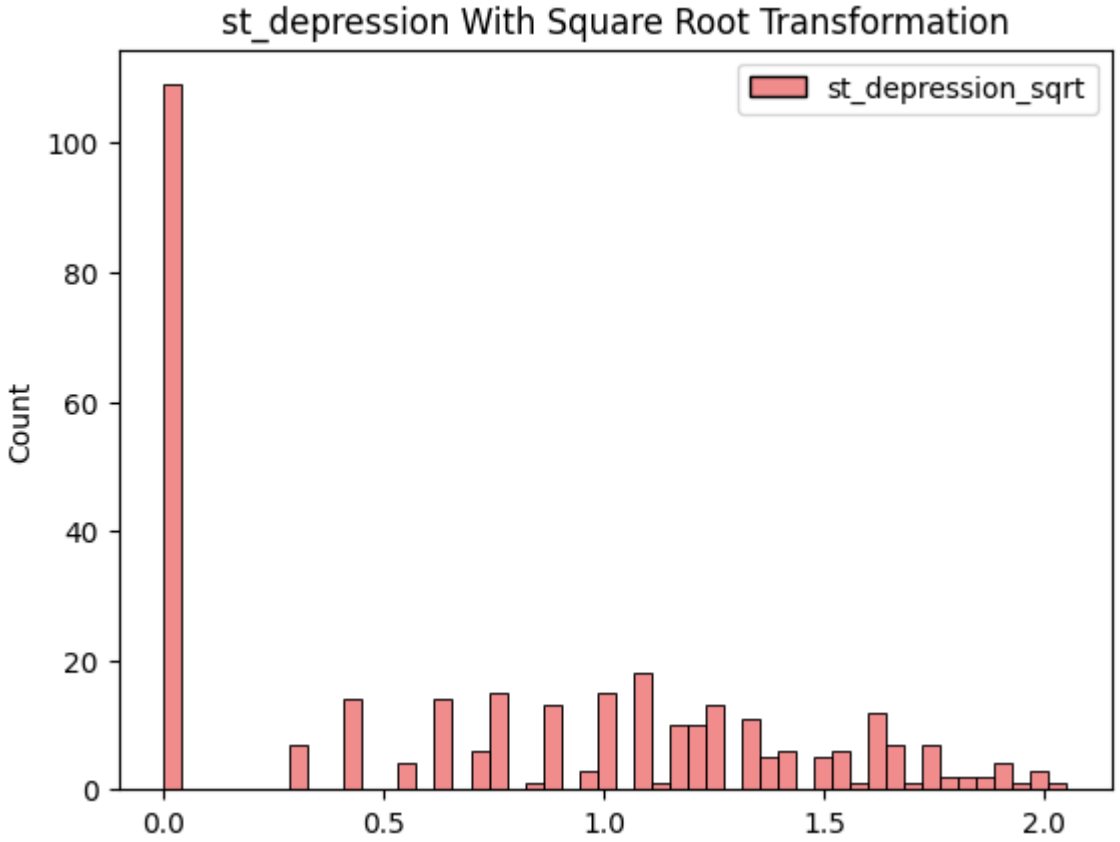


Figure 54: square root transformed st_depression variable

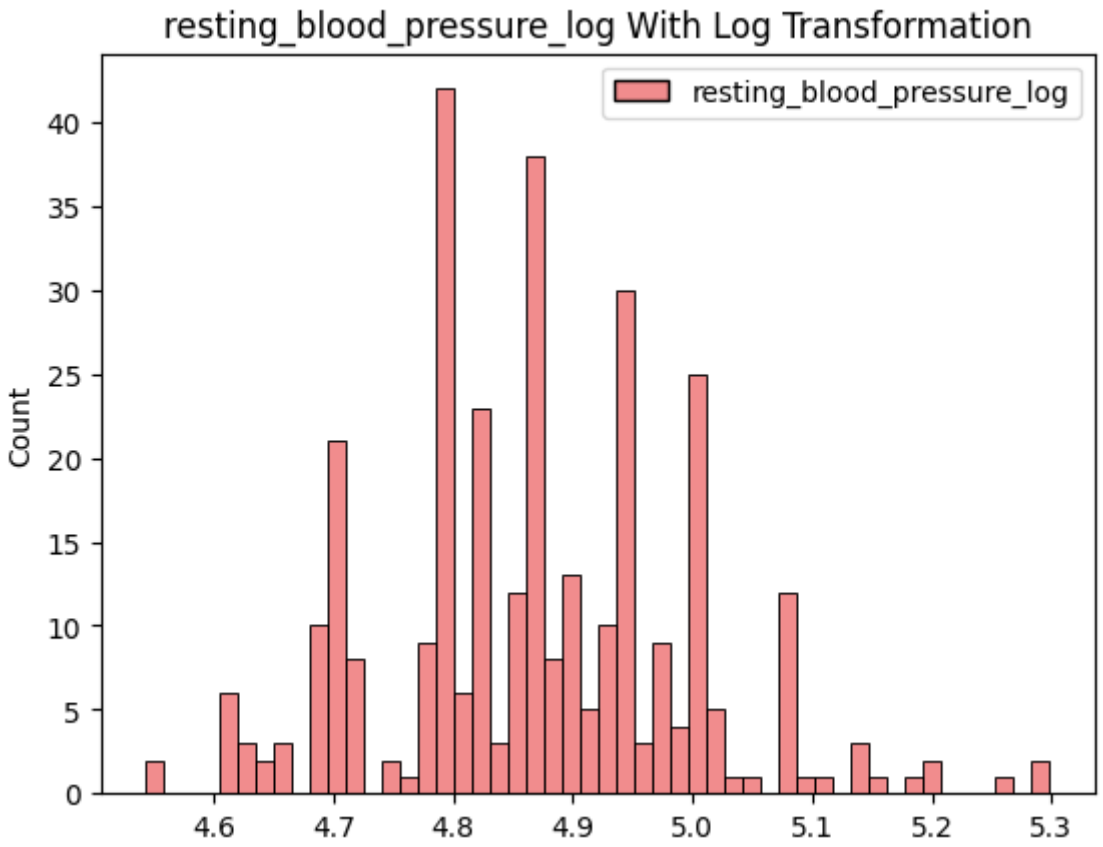


Figure 55: Log transformed resting_blood_pressure attribute

After implementing these transformations to the train dataset, I reimplemented the same lasso model fit in section 4.1 in order to determine whether they were important to the prediction of cardiovascular disease in patients. The results of this lasso model can be seen in figure 56, which shows as highlighted in the red box that the resting_blood_pressure_log variable was found to be important for the prediction of cardiovascular disease, while the st_depression_sqrt variable is not. Interestingly the addition of these variables also had the model find that cholesterol and st_slope_type_flat were no longer useful for the prediction of cardiovascular disease. As a result the three variables with a coefficient of 0 were removed from the dataset.

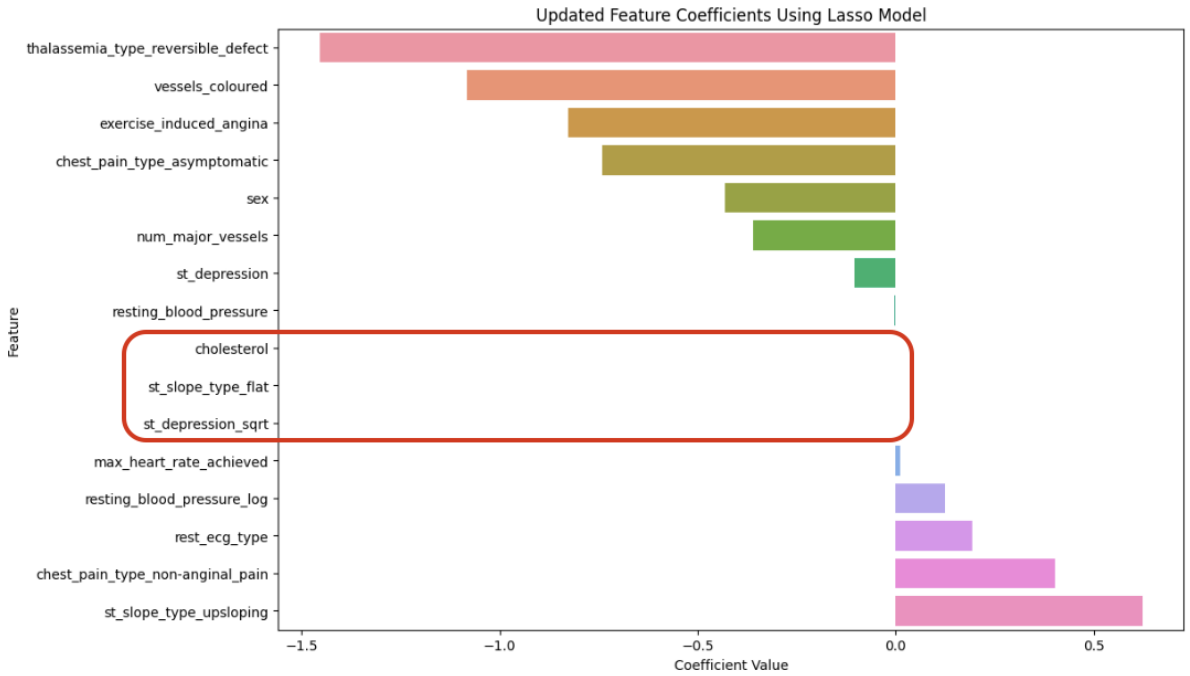


Figure 56: Feature importance check after adding transformed variables

4.2.2 Balancing the Target variable

The last remaining step in order to prepare the data for data mining is to ensure that the target variable classes are equal in quantity. A quick inspection at the proportion of the target variable in figure 57 shows that there are currently roughly 40 more observations in which the patient has cardiovascular disease than doesn't have cardiovascular disease in the train set.

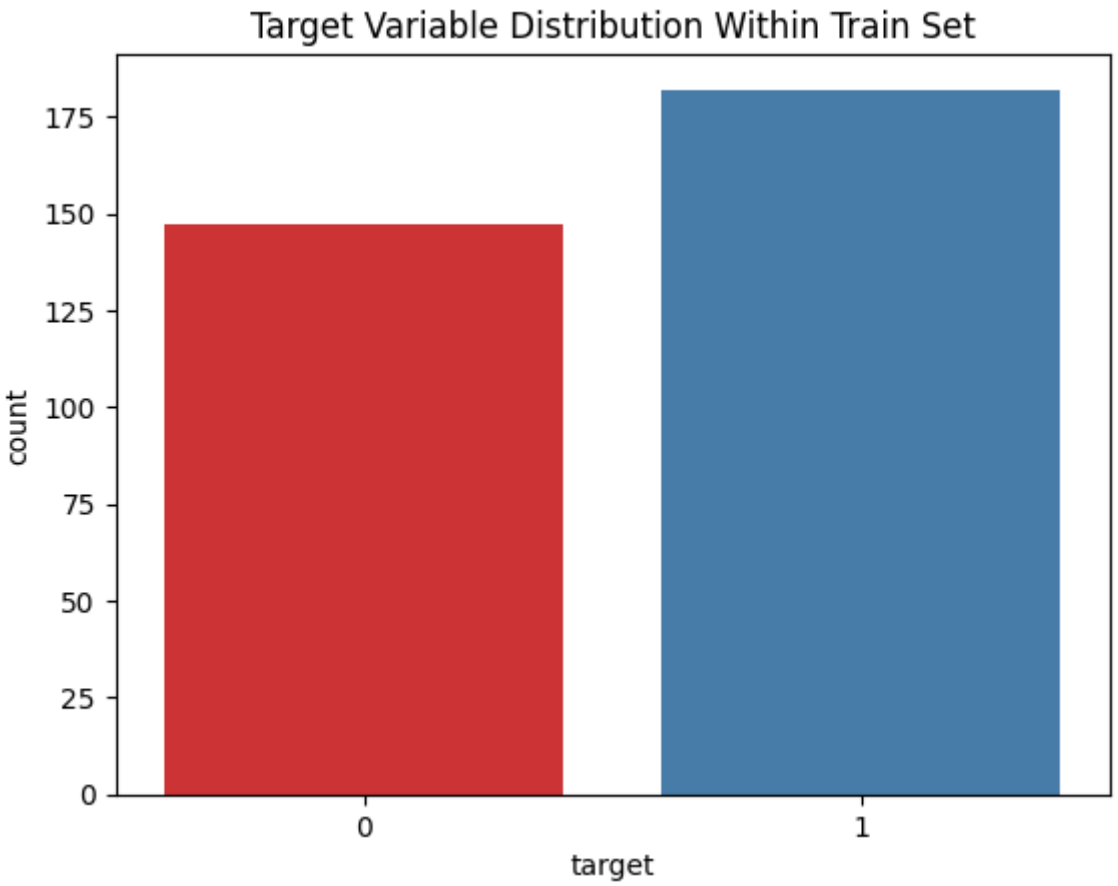


Figure 57: Target variable distribution

In order to address this I chose not to go with undersampling given the small size of the dataset, so instead it is preferred to use some kind of over sampling technique. In the end settled on performing a synthetic minority oversampling technique (SMOTE) to generate the extra observations rather than simply repeat observations. In order to ensure there is no test data bias, the SMOTE was applied to the training set, and not the entire dataset. One issue I encountered was that SMOTE was interpolating sex and exercise_induced_angina values which were between 1 and 0, when it should be interpolating as a categorical variable. To fix this I changed these variables typings such that they would not be coerced to float type values by the algorithm. The implementation of this SMOTE oversampling on the train set can be seen in figure 58.

```
from imblearn.over_sampling import SMOTE

train = train.toPandas()

train_x, train_y = train.drop("target",axis=1), train["target"]

sm = SMOTE(random_state=722)
new_x,new_y = sm.fit_resample(train_x, train_y)

train = new_x
train["target"] = new_y

train = spark.createDataFrame(train)
```

Figure 58: Implementation of SMOTE

The results of this over sampling can be seen in figure 59, now with an even split between observations with cardiovascular disease present and non present. A final view of the data at this stage can be seen in figures 60 & 61.

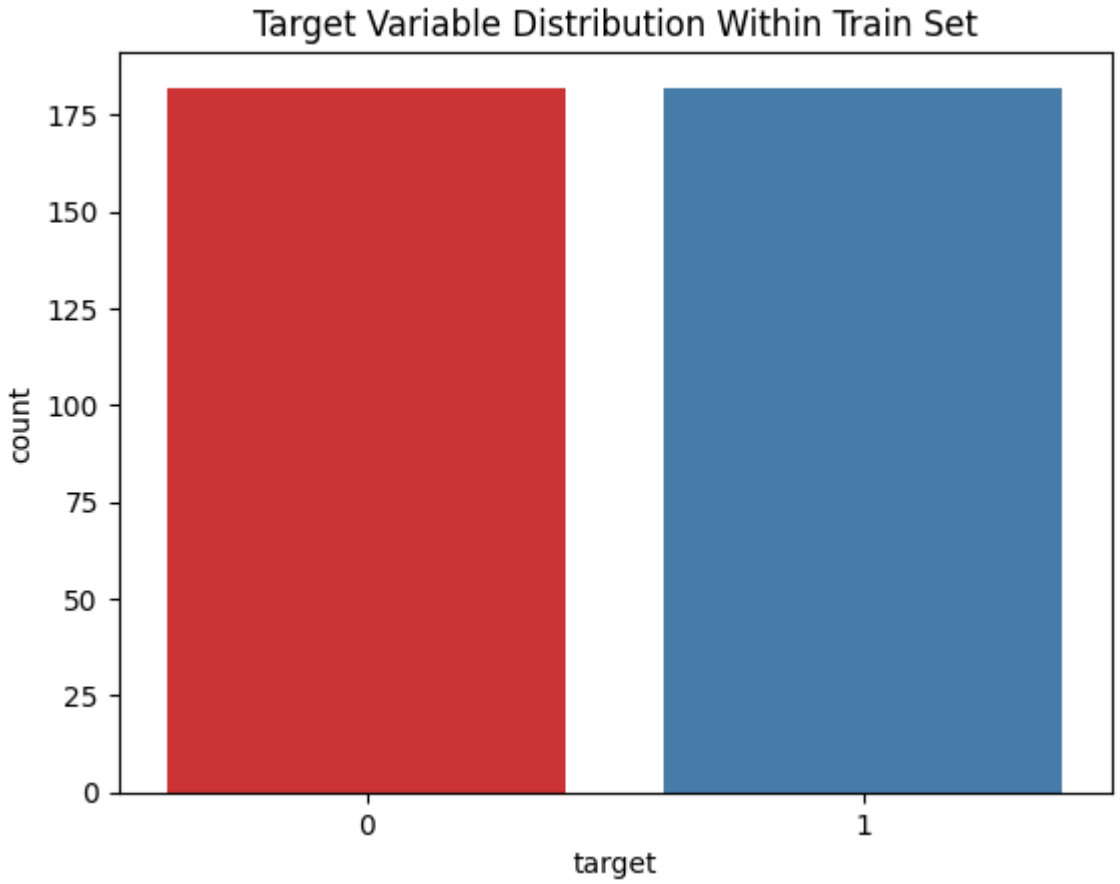


Figure 59: Distribution of target variable post SMOTE

summary	count	mean	stddev	min	max
sex	447	0.6868008948545862	0.4643142522497525	0	1
resting_blood_pressure	447	131.40939597315437	17.289806254177034	94	200
rest_ecg_type	447	0.4899328859060403	0.5004587536405618	0	1
max_heart_rate_achieved	447	148.48322147651007	23.25668929090527	71	202
exercise_induced_angina	447	0.319910514541387	0.4669642356279164	0	1
st_depression	447	1.0342954569152438	1.081581962513034	0.0	6.199999809265137
num_major_vessels	447	0.6666666666666666	0.9270877945342015	0	3
vessels_coloured	447	0.3959731543624161	0.48960666852143203	0	1
chest_pain_type_asymptomatic	447	0.49217002237136465	0.5004988432592138	0	1
chest_pain_type_non-anginal_pain	447	0.25727069351230425	0.4376196063183999	0	1
st_slope_type_upsloping	447	0.41834451901565994	0.4938400254400386	0	1
thalassemia_type_reversible_defect	447	0.378076062639821	0.4854500602715446	0	1
resting_blood_pressure_log	447	4.870193653457915	0.12838059391385961	4.543294782270004	5.298317366548036
target	447	0.5100671140939598	0.5004587536405617	0	1
partition	447	NULL	NULL	test	train

Figure 60: Description of the whole dataset

```
root
 |-- sex: integer (nullable = true)
 |-- resting_blood_pressure: integer (nullable = true)
 |-- rest_ecg_type: integer (nullable = true)
 |-- max_heart_rate_achieved: integer (nullable = true)
 |-- exercise_induced_angina: integer (nullable = true)
 |-- st_depression: float (nullable = true)
 |-- num_major_vessels: integer (nullable = true)
 |-- vessels_coloured: integer (nullable = true)
 |-- chest_pain_type_asymptomatic: integer (nullable = true)
 |-- chest_pain_type_non-anginal_pain: integer (nullable = true)
 |-- st_slope_type_upsloping: integer (nullable = true)
 |-- thalassemia_type_reversible_defect: integer (nullable = true)
 |-- resting_blood_pressure_log: integer (nullable = true)
 |-- target: integer (nullable = true)
 |-- partition: string (nullable = false)
```

Figure 61: Info of the whole dataset

5. Data-Mining Method(s) Selection

5.1 Discussion of Data Mining Methods in Context of Data Mining Objectives

In section 1.3 we defined the data mining goals for the project, being to identify key variables which indicate a higher risk of developing cardiovascular disease, and to additionally create a model which is able to predict the likelihood of somebody developing cardiovascular disease based on various predictors. When choosing the most appropriate models for this project, each goal will need to be addressed differently. As a result we will need to strongly consider the different modelling approaches, as well as the different requirements and assumptions made by each modelling technique.

5.1.1 Data Mining Goals and Objectives

The goal of the data mining is split into two distinct categories. Firstly to identify which variables are key variables in the prediction of developing cardiovascular disease and key thresholds at which the risk of having cardiovascular disease is high for each of these key variables. And secondly to develop a high accuracy and high recall predictive model for the presence of cardiovascular disease.

5.1.1.1 Identifying Key Variables and Thresholds to Predict High Risk

In order to identify the key cardiovascular disease predictors, only highly explainable models will be chosen, given black box models give us no indication of exactly what patients should be keeping an eye on in order to limit their risk of developing cardiovascular disease. As a

result we will choose to fit models which both are easily explainable, and which also importantly limit the amount of variables which are kept in the final model. In line with these objectives, I will fit a logistic regression model, and a decision tree which will each demonstrate key variables and high risk thresholds.

5.1.1.2 Fitting a High Accuracy Predictive Model

In fitting a high accuracy predictive model, different considerations will be undertaken compared with those discussed in section 5.1.1.1, given we are willing to sacrifice explainability in order to achieve the best performing predictive model possible. In light of this, I will start by training multiple different classification models on the train dataset using cross validation, and then assess the performance of each model in order to get a list of the highest performers. I then plan to fine tune these high performers in order to achieve a predictive accuracy higher than 85% and a recall value greater than 90%.

5.1.2 Modelling Requirements, Assumptions and Criteria

5.1.2.1 Requirements

With the models discussed in section 5.1.1 in mind, there are several requirements which must be met for an adequate and trustworthy data mining project. Firstly the data must be sufficiently preprocessed such that there are no blatant outliers or missing values. The presence of missing values will severely impact mathematical models such as Logistic regression, and so must be sufficiently dealt with before training takes place. Additionally outliers must be sufficiently handled such that they don't dominate the input space and cause undue influence on models. This is specifically important for the logistic regression model which is sensitive to outliers. Secondly the data will need to be separated sufficiently into train and test segments such that the results from the test set(s) can be trusted to be true. For preliminary analysis these splits will be through cross validation, and for the final models this will be the dedicated train and test set created in section 3.3.2 to ensure a consistent and fair comparison between models.

5.1.2.2 Assumptions

Given several different models are going to be fit, the assumptions of each model will need to be addressed. Firstly for the logistic regression model, we assume that there is a linear relationship between predictive variables and the log odds of the target variable. Additionally for logistic regression we assume that there is little to no multicollinearity between variables. General assumptions for the model fitting process include that there is a clear separation between the train and test sets, such that the data from the train set has no undue influence on the data in the test set.

5.1.2.3 Criteria

The criteria used for modelling will be different depending on the objective. For our first objective of uncovering the most influential predictive variables on the presence of cardiovascular disease, the modelling criteria will be focussed on generality and overall accuracy. These criteria are chosen since in order to determine the influence of each variable on the presence of cardiovascular disease we have no preference over which

outcome is predicted, we are only concerned with obtaining the models of best fit for the data.
For our second objective our modelling criteria will be slightly different, since the most effective use of the resulting model from this analysis will be to predict the onset of cardiovascular disease. Since this model has the intention of preventing cardiovascular disease by alerting when patients are at high risk, the model will be fit with according criteria. This means instead of focussing on maximising overall accuracy of the model, we will instead focus on maximising the recall value of the model (true positives / true positives + false negatives) where possible. In cases where this score is unavailable we will then rank models based on AUC, and then lastly by overall accuracy. Evaluating models in this way will lead to a model which is more cautious about classifying cardiovascular disease as not present so that more at-risk patients will be identified sooner.

5.2 Selecting the Appropriate Data-Mining Method(s)

Based on our discussion in part 5.1, we decided the best course of action for the first objective is to fit multiple explainable models in order to get a list of the most influential variables in terms of cardiovascular disease risk. In pursuit of this, we will first fit a binomial logistic regression model. Secondly a decision tree will be fit to the whole training set. In order to fit towards generality, the decision tree will have a limited depth of just 4 to ensure the model does not overfit too heavily to the training data and instead captures just the most important and general trends.
For our second objective, multiple machine learning models will be implemented to select high performing predictive models. Preliminarily we will fit 6 different models as shown in figure 62 in order to determine the highest performing models for our dataset. To ensure trustworthy results and to allow for models which overfit to the train set, the models will be fit using 10 fold cross validation and evaluated based on their average results across the left out folds. As discussed in section 5.1.2.3, the models will be ranked based on their recall score to ensure that the model can predict the onset of cardiovascular disease preventatively.

```
classifiers = [
    LogisticRegression(featuresCol="features", labelCol="target", maxIter=10000),
    RandomForestClassifier(featuresCol="features", labelCol="target", seed=722),
    DecisionTreeClassifier(featuresCol="features", labelCol="target", seed=722),
    GBTClassifier(featuresCol="features", labelCol="target", seed=722),
    NaiveBayes(featuresCol="features", labelCol="target"),
]
```

Figure 62: Preliminary models fit for objective 2

6. Data-Mining Algorithm(s) Selection

6.1 Exploratory Analysis of Data-Mining Algorithms Concerning DM Objectives

In section 5 we identified the data mining techniques we plan to use to address both of the two objectives of this exploration. In this section we plan to begin the implementation process of these techniques.

6.1.1 Exploratory Analysis for Objective 1

As discussed in section 5.1.1.1, the exploratory plan for the objective of determining the highest influence factors related to the presence of cardiovascular disease involves fitting a logistic regression, and decision tree. The implementation of these exploratory models can be seen in figures 63 & 64.

```
from pyspark.ml.classification import LogisticRegression
logistic = LogisticRegression(featuresCol="features", labelCol=target_column)
logistic = logistic.fit(transformed_train)
```

Figure 63: Implementation of logistic regression model

```
from pyspark.ml.classification import DecisionTreeClassifier
dt = DecisionTreeClassifier(featuresCol="features", labelCol="target", seed=722)
dt_model = dt.fit(transformed_train)
```

Figure 64: Implementation of decision tree model

6.1.1.1 Logistic Regression

When implementing the logistic regression on the dataset, we implement the model using a penalty value of None in order to preliminarily have a standard logistic regression model with no penalty given to large coefficients. In the future we plan to experiment with different penalties, but for our exploratory analysis no penalty is needed. Given there is no penalty, all 14 features are used in the logistic regression and have corresponding coefficients. Given there are some continuous variables which have varying scales, it is difficult to properly examine the relative importance of these variables, however the coefficients we can see in figure 65 give a strong indication of some potentially important variables and the direction of their influence, such as the thalassemia_type and chest_pain_type.

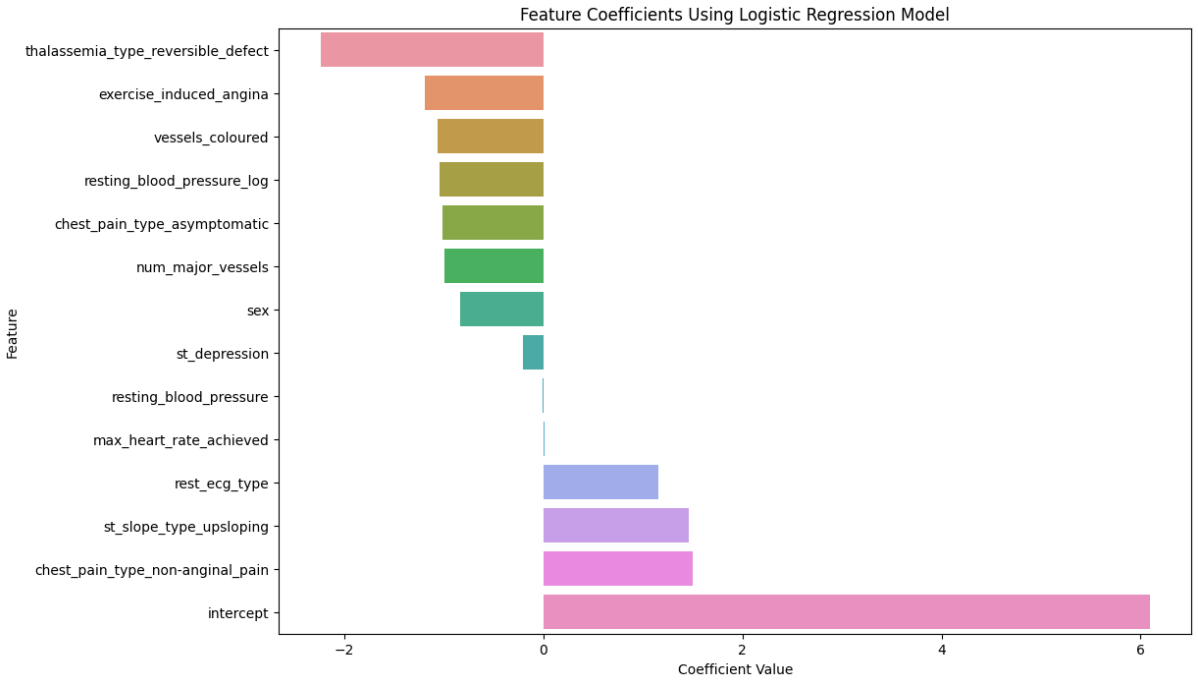


Figure 65: Logistic regression parameters

6.1.1.2 Decision Tree

The final step of exploratory analysis for objective one was to fit a decision tree. Given this model is simply for an exploratory analysis, I chose to leave the parameters untouched in order to get a general idea of what a fully fit decision tree finds to be the most important variables in the model. A view of the predictor importance shown in figure 66 demonstrates that it is a helpful model in terms of identifying predictive variables, however an investigation into the size of the decision tree shows it is not effective as an explanatory model on its own given its large size, having 33 nodes as seen in figure 67. While on its own the model is not effective as an explanatory model, it does bring unique insight into the relationships between different variables which is a subset of information missed by the logistic regression model. These results show that the full tree and uncapped rulesets are not of great use to us, but with limited depth the tree is helpful in showing unique information which is not shown through other methods.

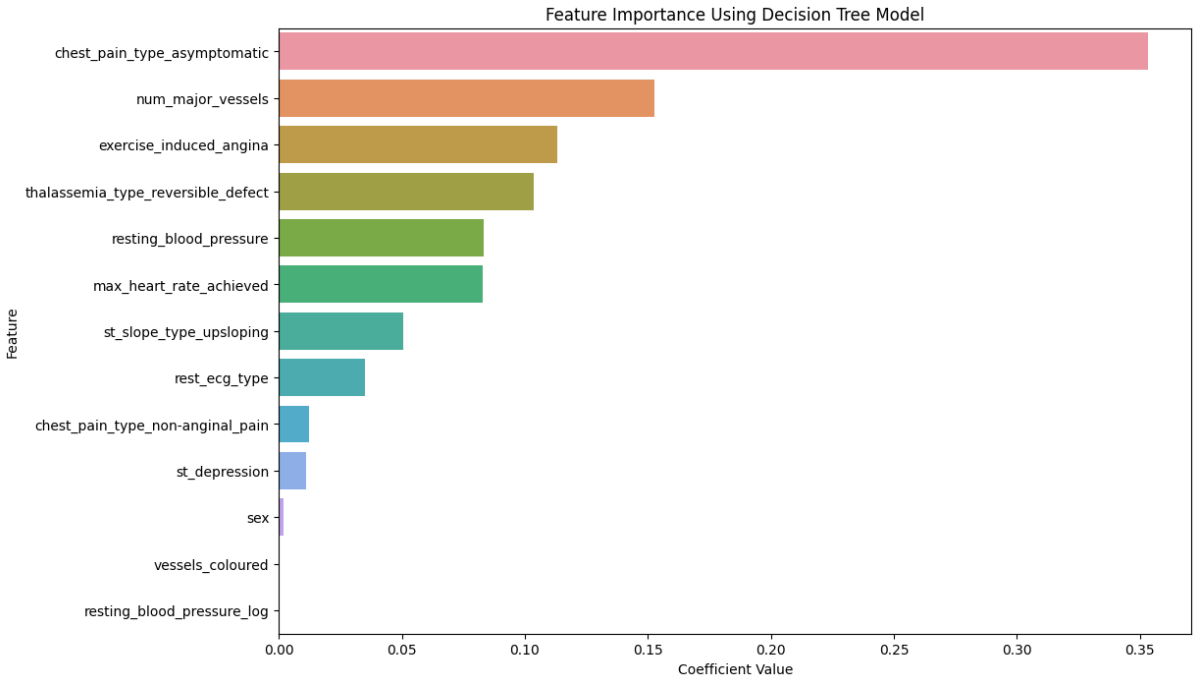


Figure 66: Predictor importance from C5.0 Decision tree model

```
Decision Tree Structure (Text Representation):
depth=5, numNodes=33, numClasses=2, numFeatures=14
  If (chest_pain_type_asymptomatic <= 0.5)
   If (st_depression <= 1.9984288000000001)
    If (resting_blood_pressure <= 167.5)
     If (num_major_vessels <= 1.5)
      Predict: 1.0
     Else (num_major_vessels > 1.5)
      If (max_heart_rate_achieved <= 152.5)
       Predict: 0.0
      Else (max_heart_rate_achieved > 152.5)
       Predict: 1.0
    Else (resting_blood_pressure > 167.5)
     Predict: 0.0
   Else (st_depression > 1.9984288000000001)
    If (num_major_vessels <= 0.5)
     If (cholesterol <= 239.5)
      If (st_slope_type_flat <= 0.5)
       Predict: 1.0
      Else (st_slope_type_flat > 0.5)
       Predict: 0.0
     Else (cholesterol > 239.5)
      Predict: 1.0
    Else (num_major_vessels > 0.5)
     Predict: 0.0
  Else (chest_pain_type_asymptomatic > 0.5)
   If (num_major_vessels <= 0.5)
    If (exercise_induced_angina <= 0.5)
     If (thalassemia_type_reversible_defect <= 0.5)
```

```
    If (cholesterol <= 305.5)
     Predict: 1.0
    Else (cholesterol > 305.5)
     Predict: 0.0
   Else (thalassemia_type_reversible_defect > 0.5)
    If (rest_ecg_type <= 0.5)
     Predict: 0.0
    Else (rest_ecg_type > 0.5)
     Predict: 1.0
  Else (exercise_induced_angina > 0.5)
   If (resting_blood_pressure <= 109.0)
    Predict: 1.0
   Else (resting_blood_pressure > 109.0)
    Predict: 0.0
Else (num_major_vessels > 0.5)
 If (resting_blood_pressure <= 109.0)
  If (max_heart_rate_achieved <= 147.5)
   Predict: 1.0
  Else (max_heart_rate_achieved > 147.5)
   Predict: 0.0
 Else (resting_blood_pressure > 109.0)
  Predict: 0.0
```

Figures 67 & 68: Decision tree rulesets

6.1.2 Exploratory Analysis for Objective 2

As discussed in section 5.1.1.2, the exploratory plan for the objective of developing a highly accurate predictive model is to test multiple different classification models using cross validation. The implementation of these different classification models can be seen in figure 69, in which I chose to fit gradient boosted trees, random forest, SVM, naive bayes, decision tree, and stochastic gradient descent models. The classifiers were fit using 10 fold cross validation in order to minimise bias in the results, since some classifiers innately perform overly well on the training set such as random forest and gradient boosted trees models. Given this is an exploratory analysis I chose to fit these models using only the default parameters to get a baseline understanding of the best performing models.

```
from pyspark.ml.classification import (
    LogisticRegression,
    RandomForestClassifier,
    DecisionTreeClassifier,
    GBTClassifier,
    NaiveBayes,
)
from pyspark.ml.evaluation import BinaryClassificationEvaluator, MulticlassClassificationEvaluator
from pyspark.ml.tuning import CrossValidator, ParamGridBuilder


classifiers = [
    LogisticRegression(featuresCol="features", labelCol="target", maxIter=10000),
    RandomForestClassifier(featuresCol="features", labelCol="target", seed=722),
    DecisionTreeClassifier(featuresCol="features", labelCol="target", seed=722),
    GBTClassifier(featuresCol="features", labelCol="target", seed=722),
    NaiveBayes(featuresCol="features", labelCol="target"),
]

eval_metrics = ["AUC", "accuracy", "weightedRecall", "weightedPrecision", "f1"]
eval_results = {metric:[] for metric in eval_metrics}


for metric in eval_metrics:
    if metric == "AUC":
        evaluator = BinaryClassificationEvaluator(labelCol="target", rawPredictionCol="prediction", metricName="areaUnderROC")
    else:
        evaluator = MulticlassClassificationEvaluator(labelCol="target", predictionCol="prediction", metricName=metric)
    for classifier in classifiers:
        # Create a parameter grid for hyperparameter tuning if needed
        paramGrid = ParamGridBuilder().build()

        # Create a CrossValidator
        cv = CrossValidator(
            estimator=classifier,
            estimatorParamMaps=paramGrid,
            evaluator=evaluator,
            numFolds=10,
            seed=722
        )

        # Fit the model
        cvModel = cv.fit(transformed_train)
        eval_results[metric] += [round(cvModel.avgMetrics[0],4)]
```

Figure 69: Implementation of classifier tests

The results of the classifier model tests can be seen in figure 70. We can see that all the models fit except for naive bayes manage to achieve greater than 85% accuracy, but none achieve over 90% recall. The fact that the logistic regression and decision tree models perform so well is somewhat surprising given we would expect more robust models such as random forest and gradient boosted trees to perform better in terms of accuracy. The reason for this could potentially be that the oversampled features created by SMOTE would impact the validity of cross validation. Because of this we will fit the random forest and gradient

boosted trees models for objective 2, and keep an eye on the simpler decision tree and logistic regression models to see if they perform just as well on the left out test set in our datamining section.

	AUC	accuracy	weightedRecall	weightedPrecision	f1	average
Logistic Regression	0.8721	0.8718	0.8718	0.8761	0.8718	0.8727
Random Forest	0.8697	0.8707	0.8707	0.8768	0.8703	0.8716
Decision Tree	0.8921	0.8896	0.8896	0.8955	0.8898	0.8913
Gradient Boosted Trees	0.8917	0.8886	0.8886	0.8958	0.8887	0.8907
Naive bayes	0.8459	0.8435	0.8435	0.8572	0.8425	0.8465

Figure 70: Classifier test results

6.2 Selecting Algorithm(s) Based on Discussion and Exploratory Analysis

6.2.1 Selecting Algorithms for Objective 1

An exploration into the methods proposed in section 5 demonstrated some expected and unexpected results. Firstly an insight into the logistic regression proved effective at demonstrating the directional impact of each variable on the target variable, but was less effective at selecting the most important variables as it is. This model will be kept in the data mining analysis done in section 7, however we will adjust it in order to look deeper into the results and ascertain the level of importance of each variable as well as the important thresholds for determining a patient's risk of having cardiovascular disease.
Secondly, a decision tree demonstrated that as a whole rule set it was not coherent, however it did give a strong indication of the most important and influential variables. It also effectively expanded upon the relationships between variables in a more in depth way than we saw in the logistic regression model. This decision tree will be kept for data mining in section 7 however it will have a smaller scope than before, where I will restrict the tree to have a smaller more interpretable depth.

6.2.2 Selecting Algorithms for Objective 2

An exploration of the results of the classification models as suggested in section 5.2 demonstrated that a few models perform exceptionally well on the dataset, however to limit the analysis, only the top two models will be considered. Consequently, the model building process to meet objective two will include fitting and parameter tuning a random forest model and gradient boosted trees model. Between these models we should be capable of finding an exceptionally accurate model which is capable of achieving a strong accuracy and recall rate on the test set.

6.3 Build/Select Model with Algorithm/Model Parameter(s)

With the models selected in section 6.2, we are ready to build the final models for the purpose of data mining. These models will be fit unique to their purpose, therefore models

for objective 1 will be built with the goal of being general, while the models for objective 2 will be built with the purpose of being accurate.

6.3.1 Building Models for Objective 1

We have chosen to keep both models tested in section 6.1.1, however we plan to implement some alterations in order to make the models more general. The logistic regression model fit will now be altered and optimised using a gridsearch to find the best parameters in terms of penalties, penalty strength, fitting of an intercept, and standardisation using 10-fold cross validation in order to develop a model which is generalised to perform well on unseen data. This approach will give more validity to the results such that the findings from the regression can be generalised to the wider population. Additionally, if the optimal model does not find predictor standardisation to be effective, a second logistic model will be integrated with standardisation enabled, using all other chosen parameters from the first model. This will allow us to understand the importance of each parameter given they are all on the same scale. The parameters used for the grid search can be seen in figure 71.

```
paramGrid = (
    ParamGridBuilder()
    .addGrid(lr.regParam, [0, 0.01, 0.03, 0.1, 1.0])
    .addGrid(lr.elasticNetParam, [0.0, 0.5, 1.0])
    .addGrid(lr.fitIntercept, [True, False])
    .addGrid(lr.standardization, [True, False])
    .build()
)
```

Figure 71: Logistic regression potential parameters

The next model fit is the decision tree. Here we will undergo a similar process as with the logistic regression by using a gridsearch using 10 fold cross validation on the test set so that the best parameters for generalisation will be chosen. The most important parameter which will be chosen is the maximum depth, which will be limited to choose between the range [1,5] so that the model will be forced to generalise. Other parameters to promote generalisability are also fine tuned as seen in figure 72.

```
paramGrid = (
    ParamGridBuilder()
    .addGrid(dt.impurity, ["gini", "entropy"])
    .addGrid(dt.maxDepth, list(range(1, 5)))
    .addGrid(dt.minInstancesPerNode, list(np.cumprod([2] * 4)))
    .build()
)
```

Figure 72: Decision tree potential parameters

6.3.2 Building Models for Objective 2

In section 6.2.2 we decided to limit our predictive model search to just a random forest model and a gradient boosted trees model.
The random forest model was implemented with various additional parameters introduced in order to allow for hyper parameter tuning as well as for generalisation to the test set. The final parameters chosen for the fine tuning process can be seen in figure 73. Given the random forest model is dependent on overfitting multiple decision trees, the range of tested parameters is smaller, only finetuning the number of estimators to ensure convergence, the decision tree impurity criterion and max_features considered for each decision tree in the model.
By not influencing many of the decision tree parameters within the model we ensure that each decision tree can be sufficiently overfit in order to maximise the variance between each model, and as a result improve the overall random forest accuracy.

```
paramGrid = (
    ParamGridBuilder()
    .addGrid(rf.numTrees, [300, 400, 500, 1000])
    .addGrid(rf.featureSubsetStrategy, ["auto", "sqrt", "log2"])
    .addGrid(rf.impurity, ["gini", "entropy"])
    .build()
)
```

Figure 73: Random forest potential model parameters

The second model to be fit is a tree based gradient boosted trees model. This model has less available parameters to be altered than the random forest model however a larger range is tested. I have chosen to hyper parameter tune the learning rate and maximum depth of each sub tree of the gradient boosted trees model in order to optimise performance. The final parameters to be tuned by grid search can be seen in figure 75.

```
paramGrid = (
    ParamGridBuilder()
    .addGrid(gbt.maxDepth, [3, 4, 5, 6])
    .addGrid(gbt.maxBins, [32, 64])
    .addGrid(gbt.stepSize, [0.05, 0.1, 0.15])
    .build()
)
```

Figure 74: gradient boosted trees potential model parameters

7. Data Mining

7.1 Creating Logical Test(s)

In order to build and fairly test our models, the data was split into a train and test set of 80% and 20% of the data respectively as seen in section 3.3.2. The train set has 364 observations while the test set has 83 observations as seen in figure 75. This split was chosen such that there was a large enough proportion on the dataset which was left out in order to get trustworthy and believable results from all testing on the final models, while still allowing a sufficient amount of data to train on. I believe any smaller of a training set or any smaller of a test set would lead to a lack of training data to get the most effective model, or a lack of test data to get trustworthy results.

```
Train set observations: 364
Train set observations: 83
```

Figure 75: Train/Test set observations

Furthermore in order to develop the best models using this 80% test set, all the models implemented will be done using cross validation. We will implement cross validation with 10 folds in order to ensure all models fit are robust to generalise to unseen data, so that there will be no question of the trustworthiness of our final results. Out of the industry standards of 5 fold, 10 fold and leave one out cross validation, I settled on 10 fold cross validation. I made this choice given the dataset is small enough that computation of 10 folds will not be an issue and will give a more accurate and generalizable model than those fit with 5-fold cross validation, and since the dataset is a little bit too large for leave one out cross validation to be computationally efficient.

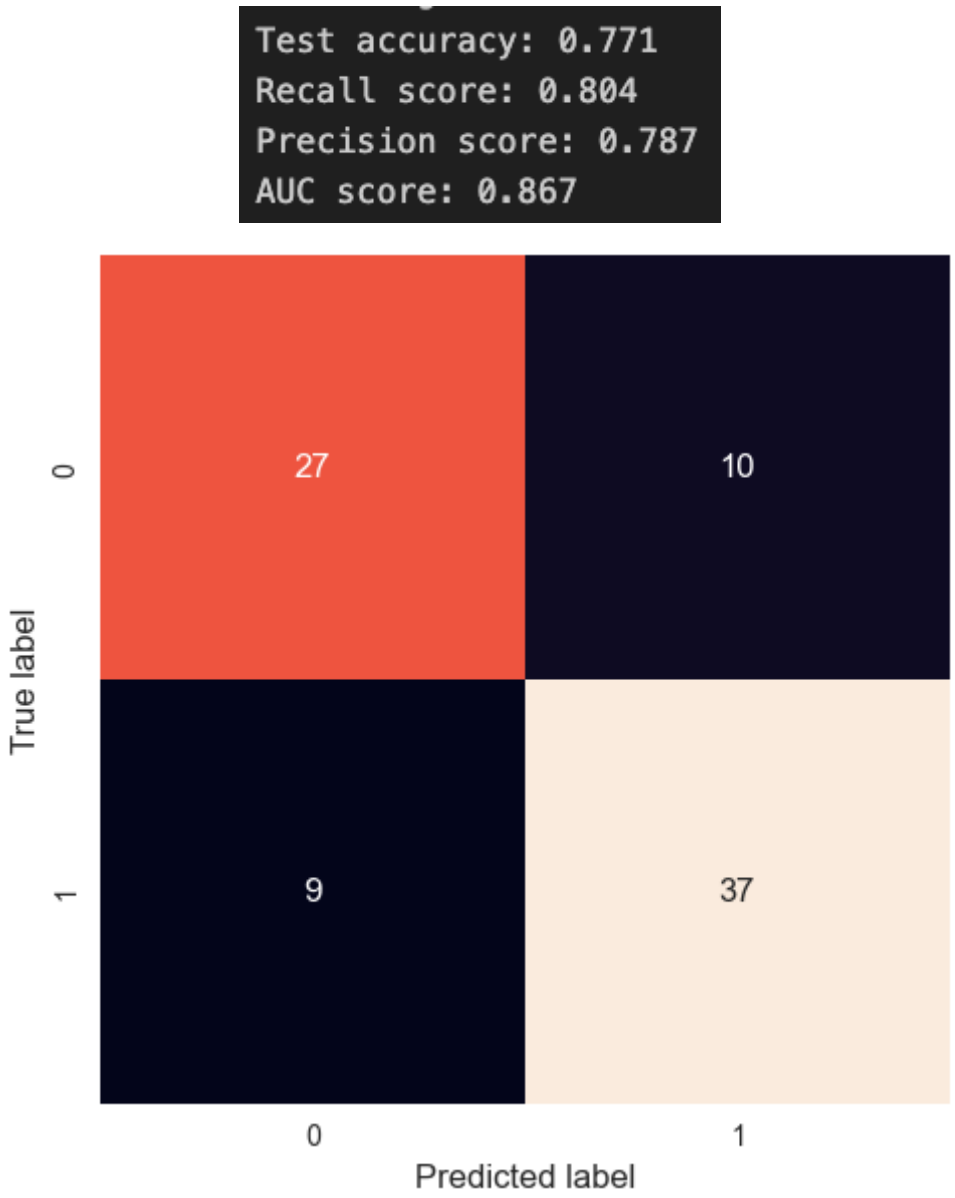
Additionally to generate the most effective models for our objective as mentioned in section 5.1, the ranking that we will be using for the predictive models will be based on Recall primarily, followed by AUC and then overall accuracy. We will do this as the models created from this research are created with the intention of identifying cardiovascular disease before it is diagnosed. As such we would rather have models which are more liberal with labelling patients as having cardiovascular disease so that they can visit a doctor before they are in major danger.

7.2 Conducting Data Mining

7.2.1 Data Mining for Objective 1

7.2.1.1 Logistic regression

The first step of data mining for objective 1 is to run the logistic regression model with hyper parameter tuning in order to find the most generalised model for the data.



Figures 76 & 77: Logistic regression performance

After fitting the model on the train set, we can observe it performed with 77% accuracy on the left out test samples, which suggests that its results are somewhat trustworthy. Additionally the model performs with an AUC of almost 87% which suggests the logistic regression model should be investigated further for its ability to meet our second objective.

We find as shown in figure 78 that the optimal logistic regression model has standardisation enabled such that the train samples are first scaled before being fit. As such we can interpret the feature importances as they are without concern of scaling.

```
Best params:
Regularization Param: 0.03
Elastic Net Param: 0.0
Fit Intercept: True
Standardization: True
```

Figure 78: Logistic regression optimal parameters

The predictor importance plot shown in figure 79 somewhat mimics the coefficients we saw in section 6.1 with their order of importance shuffled around. For instance resting_blood_pressure_log is found to be much less important in the new standardised model. An interesting observation from these results is that some variables seem to act counterintuitively, for instance resting_blood_pressure which apparently decreases the chance of cardiovascular disease as it increases. These counter intuitive results suggest that the effect we would expect resting_blood_pressure to have on the results are instead being explained by another variable. The coefficients attributed to each predictor is shown in figure 79 which we will assess further in section 8.1.

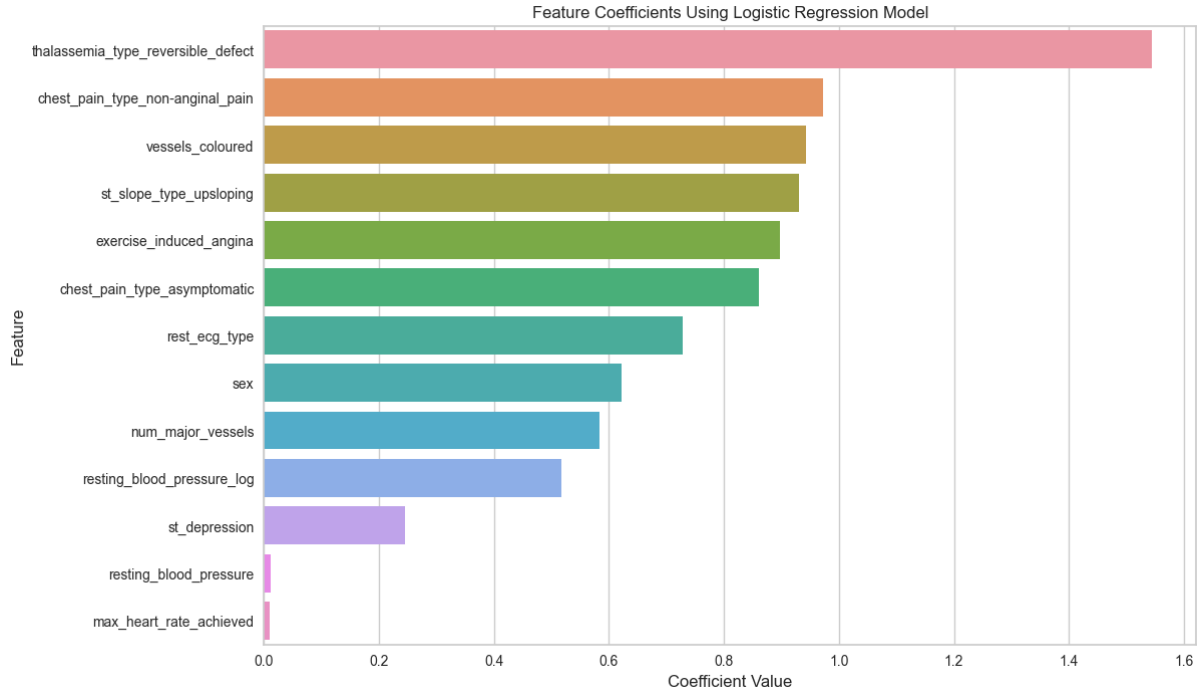


Figure 79: Logistic regression predictor importance

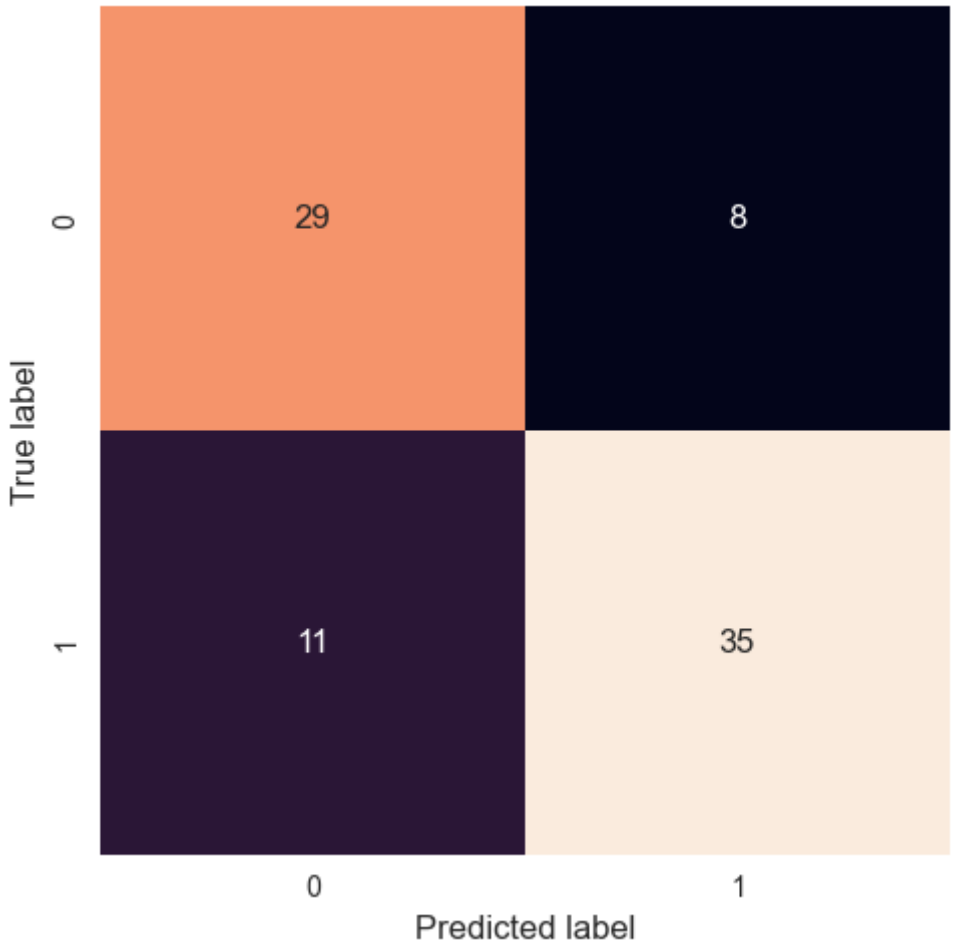
Coefficient	Value
thalassemia_type_reversible_defect	-1.544103
vessels_coloured	-0.942043
exercise_induced_angina	-0.897877
chest_pain_type_asymptomatic	-0.860955
sex	-0.622108
num_major_vessels	-0.583086
resting_blood_pressure_log	-0.517404
st_depression	-0.245686
resting_blood_pressure	-0.012484
max_heart_rate_achieved	0.010650
rest_ecg_type	0.727957
st_slope_type_upsloping	0.929738
chest_pain_type_non-anginal_pain	0.972688
intercept	3.888024

Figure 80: Logistic regression predictor coefficients

7.2.1.2 Decision tree

The decision tree fit with 10 fold cross validation and grid search hyperparameter tuning was capable of reaching a testing accuracy of 77.1% despite being slightly simplified in the aim of being more understandable. Given this accuracy is not very impressive, nor are any of its other model performance metrics, we will explore this model purely for objective 1, and not include it for objective 2.

```
Test accuracy: 0.771
Recall score: 0.761
Precision score: 0.814
AUC score: 0.809
```



Figures 81 & 82: Performance of decision tree

The decision tree and predictor importance graph shown in figures 82/83 & 81 demonstrate that the chest_pain_type_atypical_angina is by far the most important variable for cardiovascular disease classification, with num_major_vessels, exercise_induced_angina, thalassemia_type_reversible_defect being very important also.

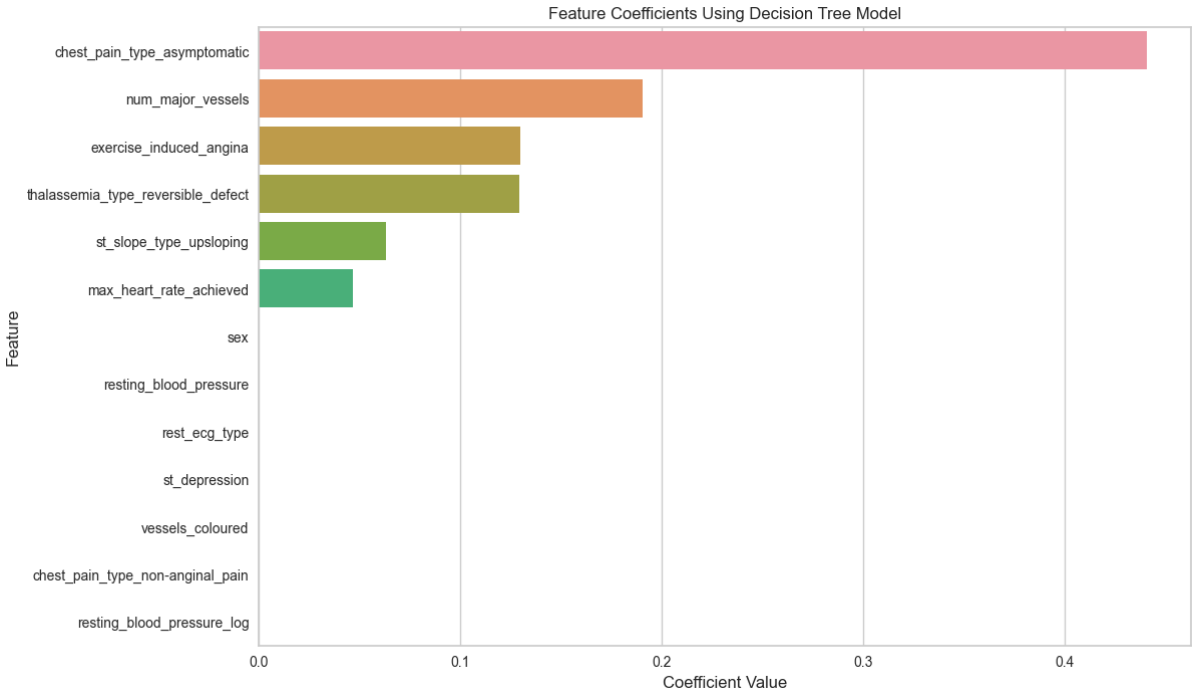


Figure 82: Decision tree predictor importance

```
Decision Tree Structure (Text Representation):
depth=4, numNodes=17, numClasses=2, numFeatures=13
  If (chest_pain_type_asymptomatic <= 0.5)
   If (thalassemia_type_reversible_defect <= 0.5)
    If (num_major_vessels <= 1.5)
     Predict: 1.0
    Else (num_major_vessels > 1.5)
     If (max_heart_rate_achieved <= 152.5)
      Predict: 0.0
     Else (max_heart_rate_achieved > 152.5)
      Predict: 1.0
   Else (thalassemia_type_reversible_defect > 0.5)
    If (st_slope_type_upsloping <= 0.5)
     Predict: 0.0
    Else (st_slope_type_upsloping > 0.5)
     Predict: 1.0
  Else (chest_pain_type_asymptomatic > 0.5)
   If (num_major_vessels <= 0.5)
    If (exercise_induced_angina <= 0.5)
     If (thalassemia_type_reversible_defect <= 0.5)
      Predict: 1.0
     Else (thalassemia_type_reversible_defect > 0.5)
      Predict: 0.0
    Else (exercise_induced_angina > 0.5)
     Predict: 0.0
   Else (num_major_vessels > 0.5)
    Predict: 0.0
```

Figure 83: Decision tree text representation

7.2.2 Data Mining for Objective 2

7.2.2.1 Random forest

The first predictive model fit is the random forest model discussed in section 6.3.2, which was not the best performing model when fit without hyperparameter tuning in section 6.1.2, however we expect it to perform better than other models on unseen data. The results from analysing the performance of the newly fit random forest model are shown in figures 84 & 85.

```
Test accuracy: 0.795
Recall score: 0.804
Precision score: 0.822
AUC score: 0.88
```

Figure 84: Random Forest train/test performance & metrics

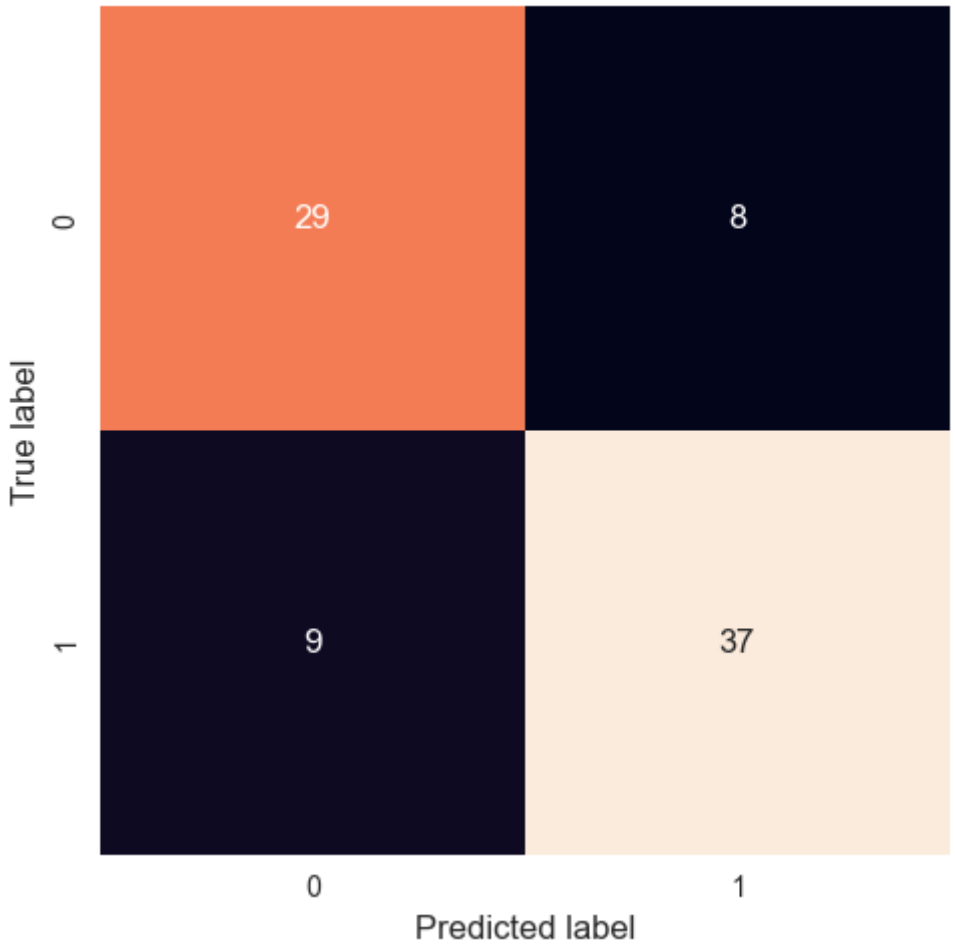


Figure 85: Random forest test set classification matrix

Similarly to the logistic regression model, we can see the accuracy is not very impressive, coming in at just under 80% accuracy, however this is slightly redeemed by the 0.88 auc value which suggests that with some tweaking we can get a sufficient model.

7.2.2.2 gradient boosted trees

The second predictive model to be implemented for data mining is the gradient boosted trees model described in section 6.3.2, which was one of the best performing models when fit with no parameter tuning. The results of the newly trained gradient boosted trees model can be seen in figures 86 & 87.

```
Train Accuracy: 0.867
Test accuracy: 0.795
Recall score: 0.804
Precision score: 0.822
AUC score: 0.86
```

Figure 86: gradient boosted trees train/test performance & metrics

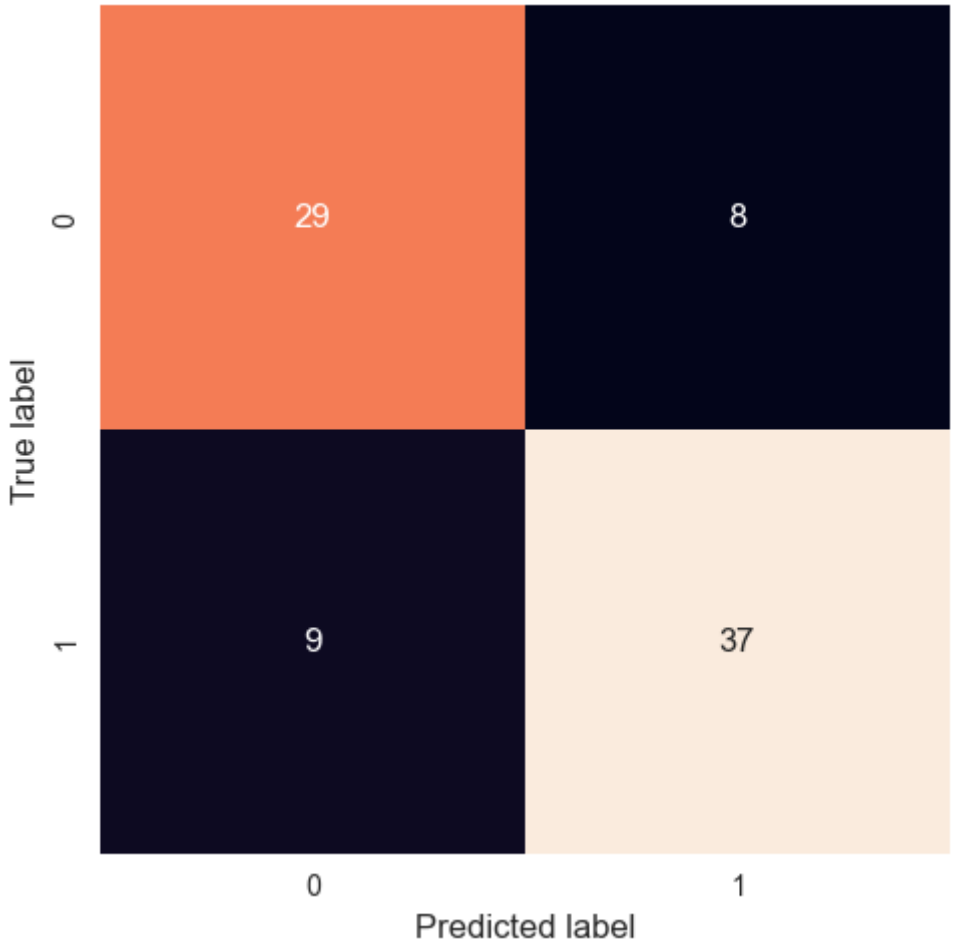


Figure 87: gradient boosted trees test set classification matrix

When evaluated on the test set we find that the gradient boosted trees model performs exactly the same as the random forest model in terms of test accuracy, recall, precision and overall classifications, but we find that actually the AUC is lower. Because of this we will need to investigate further which of our predictive models ROC curve is more favourable towards recall.

7.3 Searching for Patterns

The data mining done in section 7.2 has demonstrated several interesting patterns that are worth taking note of. The first pattern which was apparent from the feature importance plots from the decision tree and logistic regression models was the emergence of some of the most important variables in each model. Particularly we can see that chest_pain,

vessels_coloured/num_major_vessels, exercise_induced_angina and thalassemia_type, and st_type_upsloping are consistently important across both models.

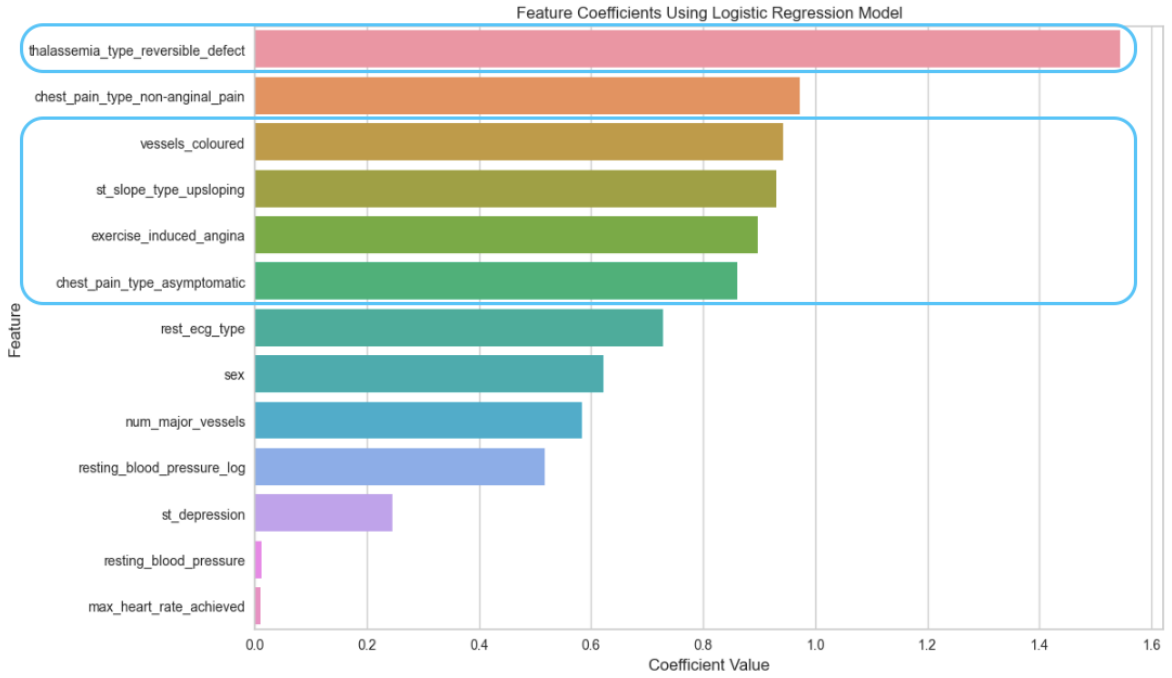


Figure 88: Stand out important predictors in Logistic Regression model

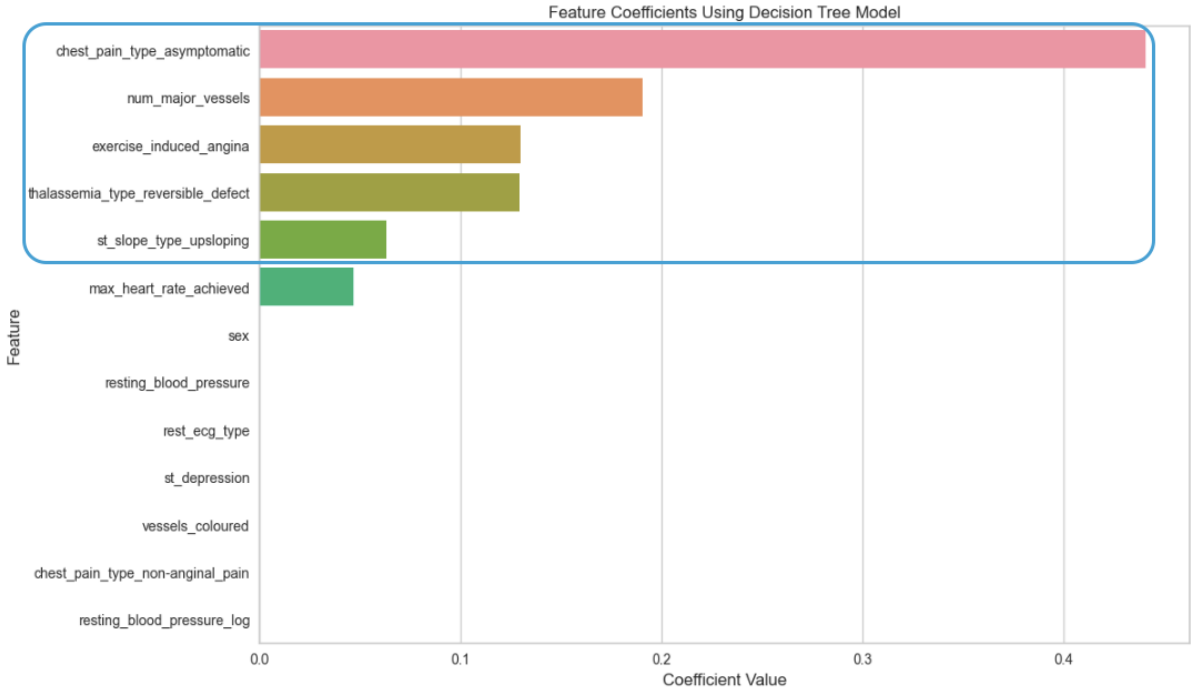


Figure 89: Stand out important predictors in Decision Tree model

The second pattern which was quite apparent from the coefficients of the logistic regression was that most variables decreased the likelihood of cardiovascular disease, including some which were counterintuitive. We can see from figure 90 that only 4 variables out of 14

(excluding the intercept) increased the likelihood of cardiovascular disease rather than decreasing it. The reasons behind this will need to be explored further in section 8.1.

Coefficient	Value
thalassemia_type_reversible_defect	-1.544103
vessels_coloured	-0.942043
exercise_induced_angina	-0.897877
chest_pain_type_asymptomatic	-0.860955
sex	-0.622108
num_major_vessels	-0.583086
resting_blood_pressure_log	-0.517404
st_depression	-0.245686
resting_blood_pressure	-0.012484
max_heart_rate_achieved	0.010650
rest_ecg_type	0.727957
st_slope_type_upsloping	0.929738
chest_pain_type_non-anginal_pain	0.972688
intercept	3.888024

Figure 90: all variables which increased likelihood of cardiovascular disease

The third pattern which is observable from the datamining performed in section 7.2 is that our predictive models tend to be more trustworthy when predicting the presence of heart disease than the absence of it. This is seen by the fact that the false omission rate (FN/PN) is higher than the false discovery rate (FP/PP) for both predictive models at 23.7% compared to 17.8% as shown in figure 91. This pattern is also observed in the decision tree and logistic regression models, and will need to be explored further in section 8.1

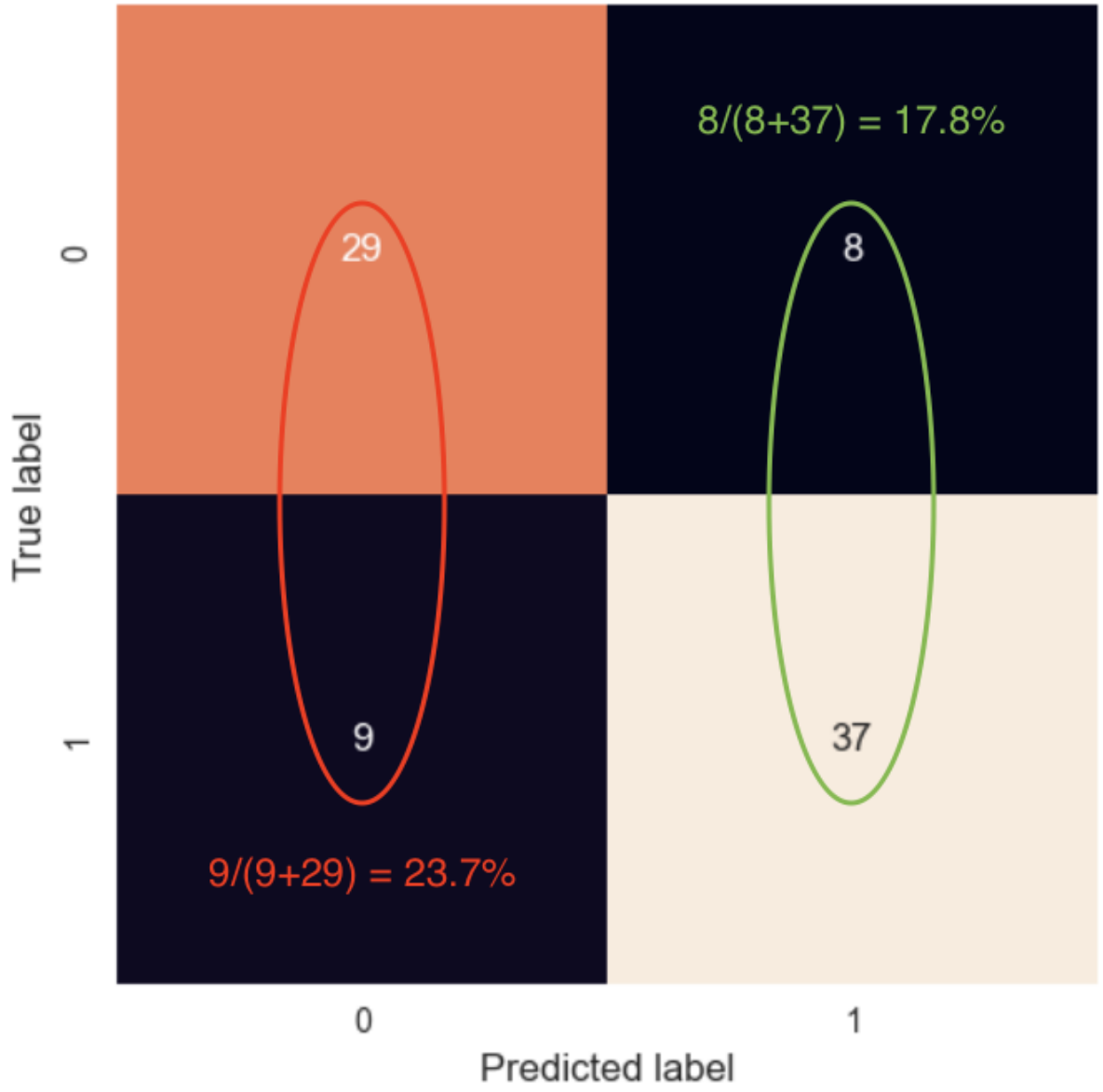


Figure 91: Predictive model preference for false positives

8. Interpretation

8.1 Study and Discuss Mined Patterns

Pattern 1: A pattern we observed earlier in section 7.3 was that both the optimised logistic regression model fit in section 7.2.1.1 and decision tree model fit in section 7.2.1.2 shared the same important variables. We can see that the models shared the 5 most important variables, where the 5 most important variables for our generalised decision tree were within the top 6 most important variables for our logistic regression model. To look further into these patterns we first fit refit our scaled logistic regression model and decision tree model to examine the feature importance as shown in figures 92 and 93.

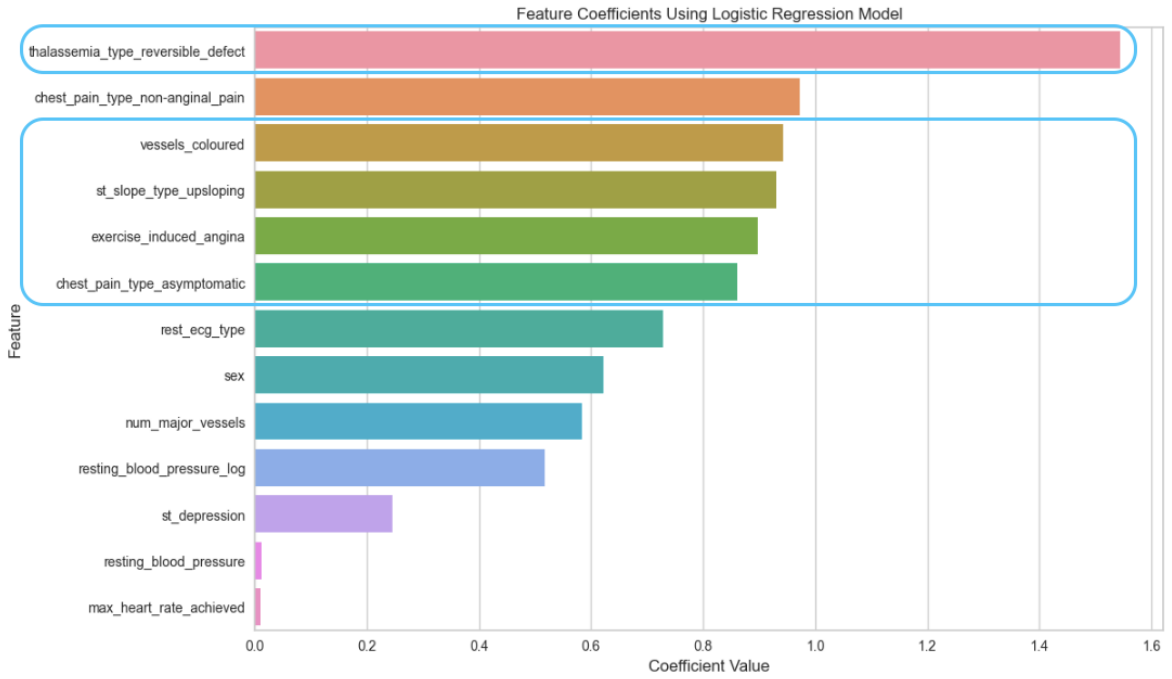


Figure 92: Coefficient magnitude of scaled logistic regression model

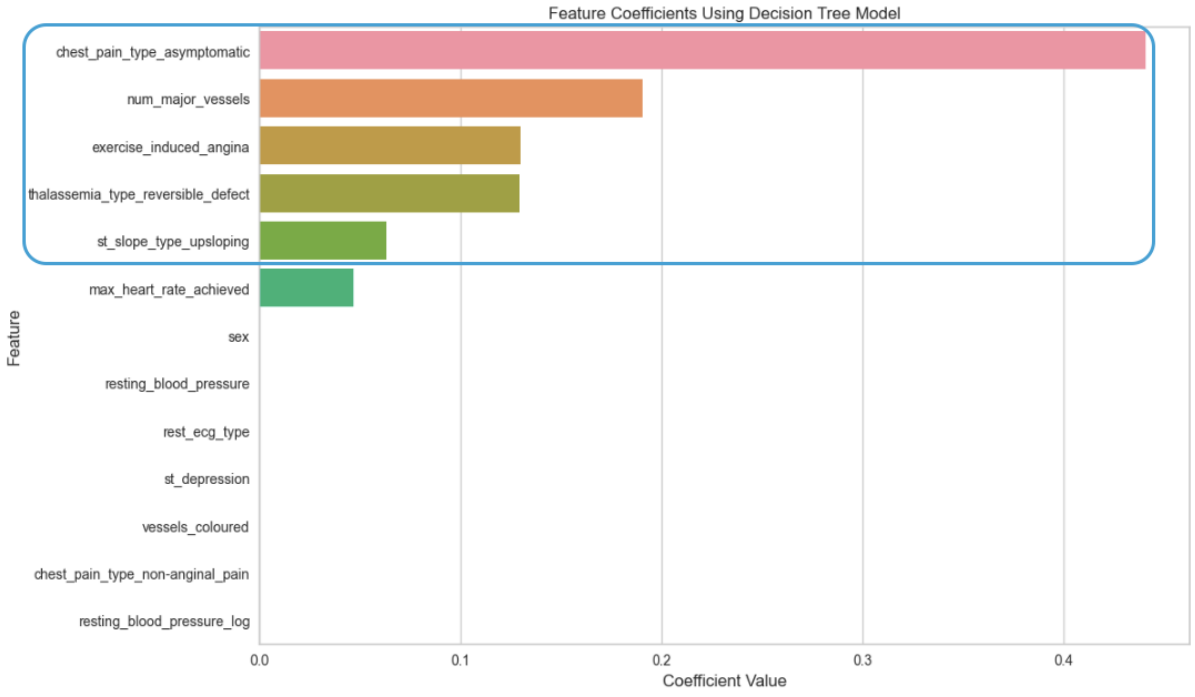
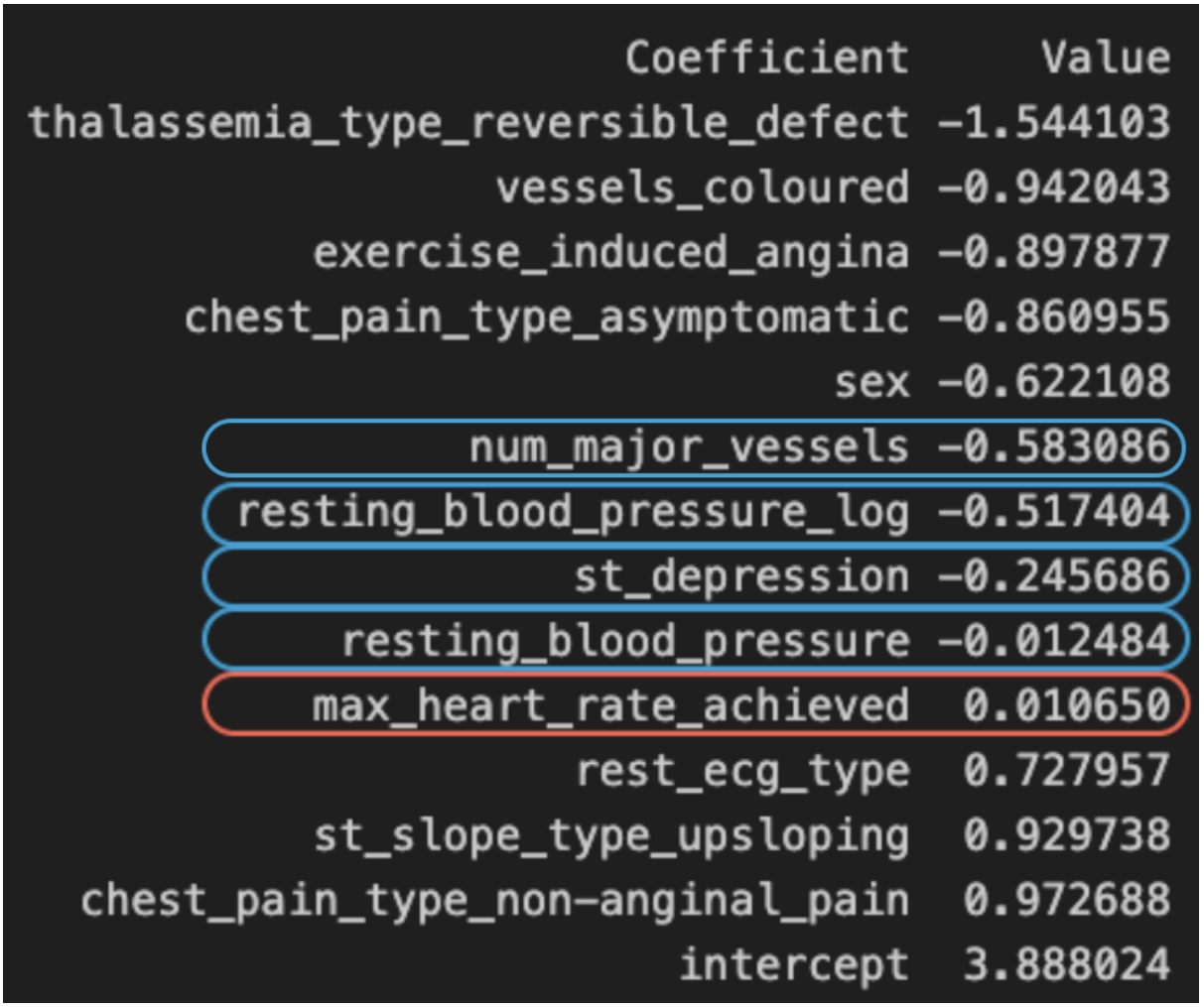


Figure 93: Feature importance of decision tree model

We can see that both models share the same important variables, which can be broken down into thalassemia_type_reversable_defect (a blood disorder), chest_pain_type_asymptomatic (asymptomatic chest pain rather than one of typical angina, atypical angina, non-anginal pain), num_major_vessels/vessels_coloured (blood vessels coloured by fluoroscopy), exercise_induced_angina (chest pain induced by exercise) and st_slope_type_upsloping (ecg results during exercise). Each of these variables make a lot of sense in predicting cardiovascular disease, which makes me very confident in the models

produced and their ability to fulfil the success criteria of objective 1. For instance having a blood disorder (thalassemia) which is reversible suggests that you are less likely to have cardiovascular disease than if it was irreversible. Having asymptomatic pain makes you less likely to have cardiovascular disease than otherwise. Having blood vessels coloured by fluoroscopy suggests that arteries are not blocked, so healthy blood flow allows for multiple blood vessels to be coloured. Having chest pain induced by exercise is less indicitive of heart disease than if the chest pain arose with no obvious trigger. Having a climbing ECG rate during exercise is more indicative of a heart disease patient. Although these variables make perfect sense, another pattern we notice is that there are many variables which the layman would assume to be important to cardiovascular disease prediction were not found to be too important. An example of some of these variables is resting blood pressure and sex which were not found to be as important compared to the five categories discussed previously.

Pattern 2: The second pattern observed in section 7.3 was that there were more variables which had a negative influence on the presence of cardiovascular disease than a positive one. An investigation into these coefficients shows there are 9 variables which decrease the likelihood of cardiovascular disease and 4 which increase it (not counting the intercept). A closer look into this however reveals that 5 of the 9 variables decreasing the likelihood of cardiovascular disease and 3 of the 4 variables increasing the likelihood are flag variables. Since these are flag variables the direction of their influence is simply based on the coding of the flag variable, so the important variables to look at are any count or continuous variables, as highlighted in figure 94.



Coefficient	Value
thalassemia_type_reversible_defect	-1.544103
vessels_coloured	-0.942043
exercise_induced_angina	-0.897877
chest_pain_type_asymptomatic	-0.860955
sex	-0.622108
num_major_vessels	-0.583086
resting_blood_pressure_log	-0.517404
st_depression	-0.245686
resting_blood_pressure	-0.012484
max_heart_rate_achieved	0.010650
rest_ecg_type	0.727957
st_slope_type_upsloping	0.929738
chest_pain_type_non-anginal_pain	0.972688
intercept	3.888024

Figure 94: Influence of count and continuous variables within the dataset

We can see with only the count and continuous variables examined, the same pattern is true. It isn't entirely clear why this is the case. The most likely explanation of this is that most of the variables which had a positive influence of the likelihood of cardiovascular disease were removed during the feature selection process in section 4.1, and now their influence is being encoded into the large intercept coefficient. The model is setting the intercept value quite highly in favour of the patient having cardiovascular disease, and most of the variables push this likelihood lower rather than higher.

Pattern 3: The third and final pattern that we observed in section 7.3 is that the models fit for prediction are inherently more accurate when predicting the presence of heart disease than predicting the absence of it. To investigate this further we took a closer look at the false omission rates and false discovery rates for each of the three models we are planning on using for prediction, being random forest, gradient boosted trees and logistic regression. These rates can be seen in figure 95, 96 & 97.

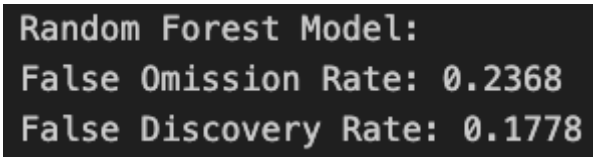


Figure 95: Random forest False Omission Rate vs False Discovery Rate

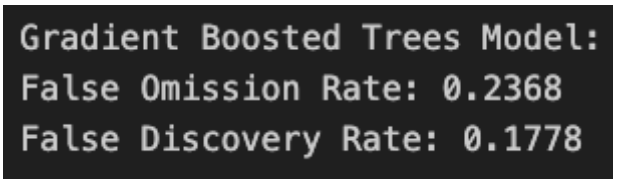


Figure 96: gradient boosted trees False Omission Rate vs False Discovery Rate

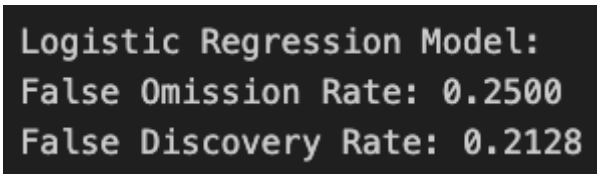


Figure 97: Logistic regression False Omission Rate vs False Discovery Rate

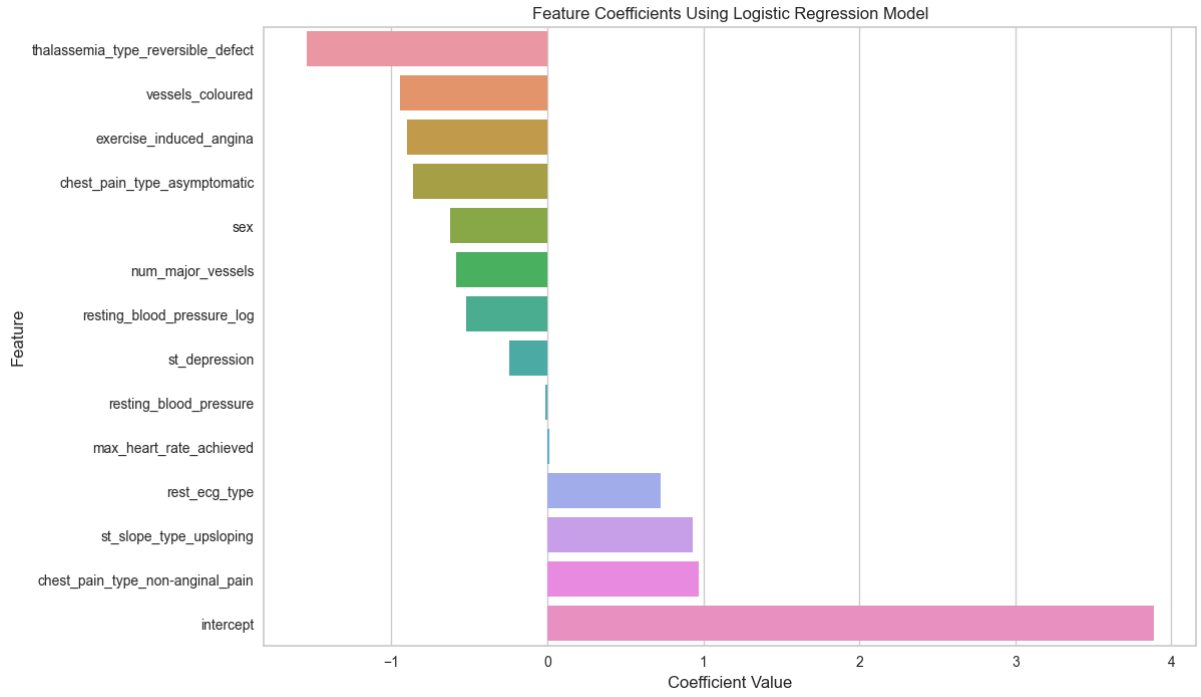
We can see this higher false omission rate is common between all three models, which suggests that our model is more accurate and confident when predicting the presence of heart disease than the opposite. This could be a result of the fact that our train set had more positive samples for cardiovascular disease inherently, with extra negative samples needing to be generated by SMOTE. As a result of this the training on the authentic positive samples is more trustworthy than the training on partially inauthentic negative samples. This is an issue for our models given we are aiming for a high recall, meaning the false omission rate should ideally be 0 (No false negatives). Because of this we will need to do some altering of our models in order to modify them to have a lower false omission rate.

8.2 Visualising the Data, Results, Models and Patterns

We will visualise the models obtained from section 7.2 by demonstrating the relationships and important variables for objective 1, and by showing the false omission and false discovery rates for the models fit for objective 2, accompanied by some other helpful information.

8.2.1 Visualising results for objective 1

8.2.1.1 Logistic regression variables included and excluded from the model & overall performance



	Coefficient	Value
thalassemia_type_reversible_defect		-1.544103
vessels_coloured		-0.942043
exercise_induced_angina		-0.897877
chest_pain_type_asymptomatic		-0.860955
sex		-0.622108
num_major_vessels		-0.583086
resting_blood_pressure_log		-0.517404
st_depression		-0.245686
resting_blood_pressure		-0.012484
max_heart_rate_achieved		0.010650
rest_ecg_type		0.727957
st_slope_type_upsloping		0.929738
chest_pain_type_non-anginal_pain		0.972688
intercept		3.888024

Figures 98 & 99: Predictors used in logistic regression model

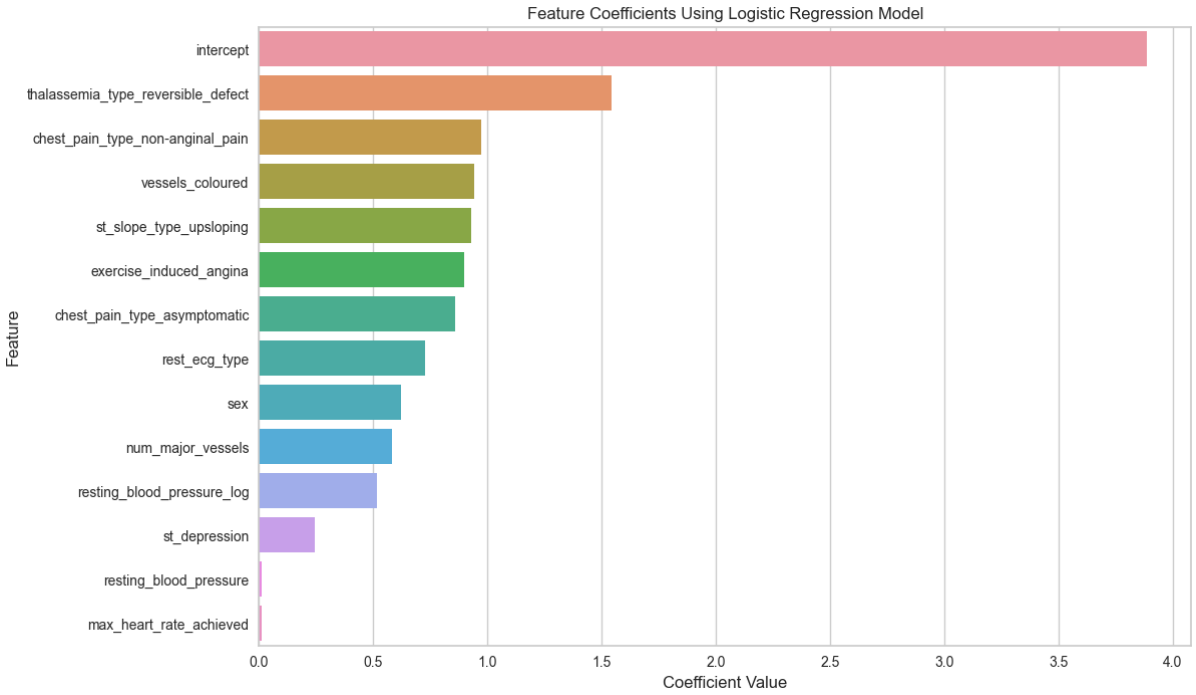


Figure 100: Feature importance using scaled logistic regression model

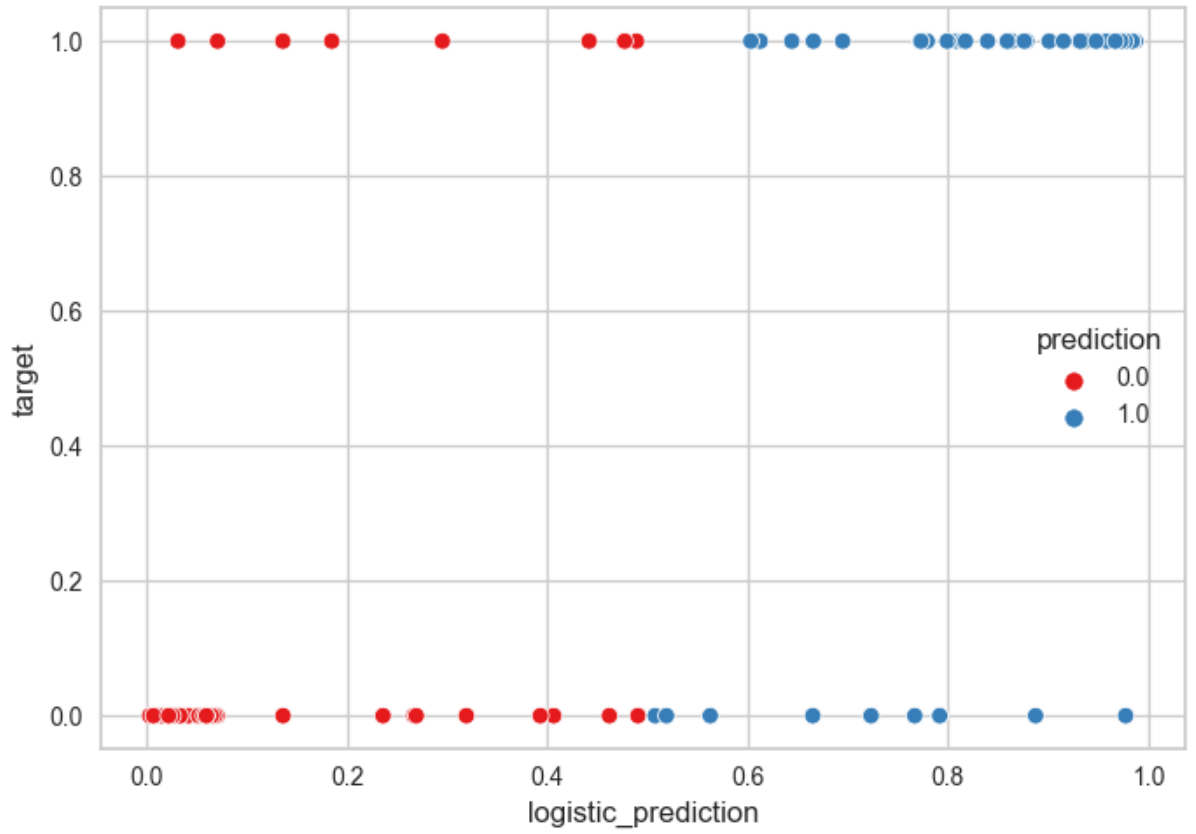


Figure 101: Logistic regression probability prediction vs target variable

8.2.1.2 Decision tree

```
Decision Tree Structure (Text Representation):
depth=4, numNodes=17, numClasses=2, numFeatures=13
  If (chest_pain_type_asymptomatic <= 0.5)
   If (thalassemia_type_reversible_defect <= 0.5)
    If (num_major_vessels <= 1.5)
     Predict: 1.0
    Else (num_major_vessels > 1.5)
     If (max_heart_rate_achieved <= 152.5)
      Predict: 0.0
     Else (max_heart_rate_achieved > 152.5)
      Predict: 1.0
   Else (thalassemia_type_reversible_defect > 0.5)
    If (st_slope_type_upsloping <= 0.5)
     Predict: 0.0
    Else (st_slope_type_upsloping > 0.5)
     Predict: 1.0
  Else (chest_pain_type_asymptomatic > 0.5)
   If (num_major_vessels <= 0.5)
    If (exercise_induced_angina <= 0.5)
     If (thalassemia_type_reversible_defect <= 0.5)
      Predict: 1.0
     Else (thalassemia_type_reversible_defect > 0.5)
      Predict: 0.0
    Else (exercise_induced_angina > 0.5)
     Predict: 0.0
   Else (num_major_vessels > 0.5)
    Predict: 0.0
```

Figure 102: Decision tree text representation

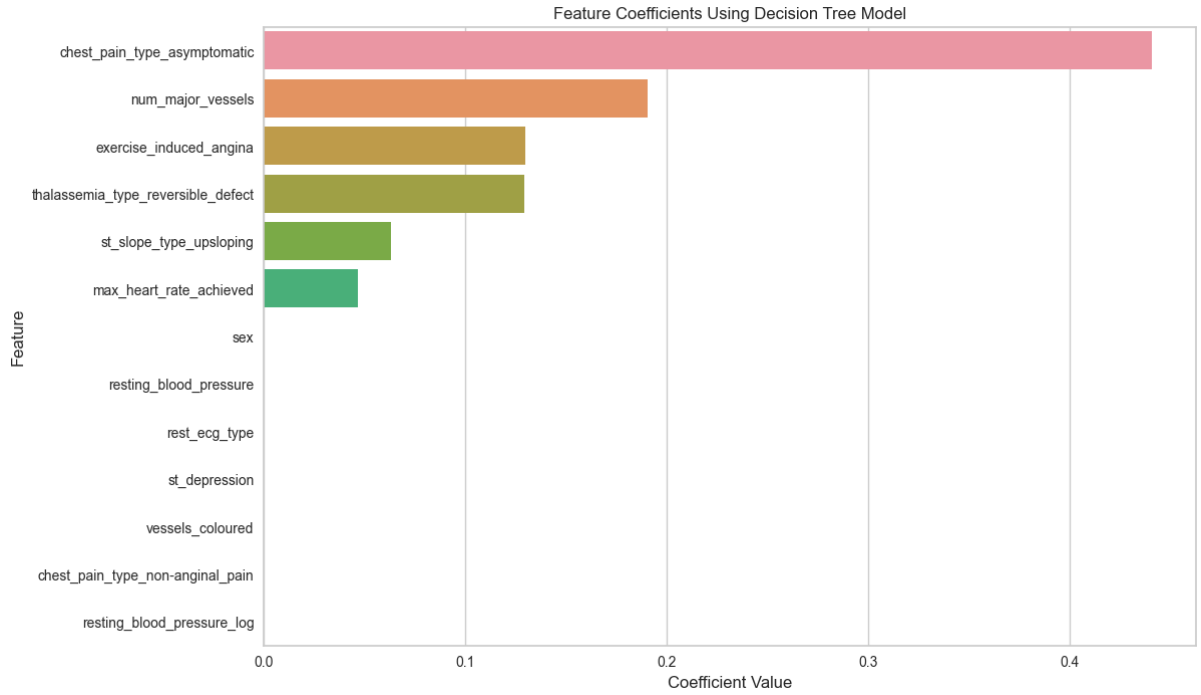


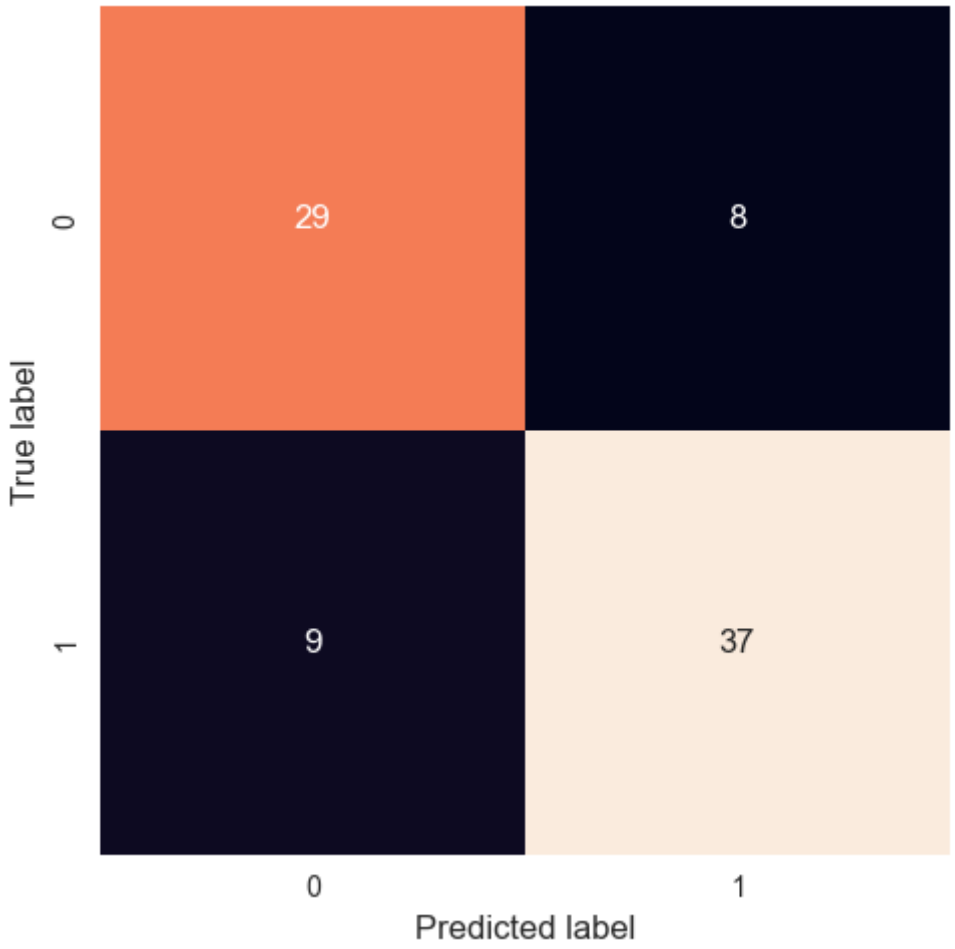
Figure 103 Decision tree predictor importance

8.2.2 Visualising results for objective 2

In order to gain a deeper understanding of how the predictive models tradeoff recall for specificity, ROC plots of the models performance on the test set are implemented alongside general performance statistics.

8.2.2.1 Test set analysis and ROC curve for random forest model

```
Test accuracy: 0.795
Recall score: 0.804
Precision score: 0.822
AUC score: 0.88
```



Figures 104 & 105: Performance of random forest model

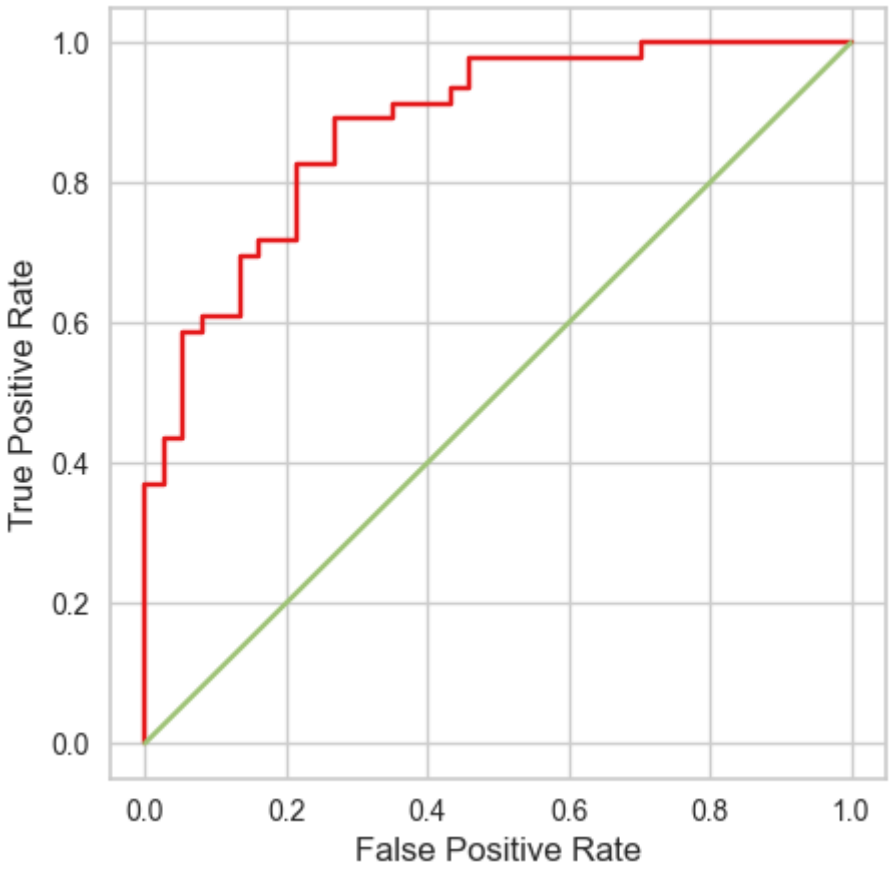
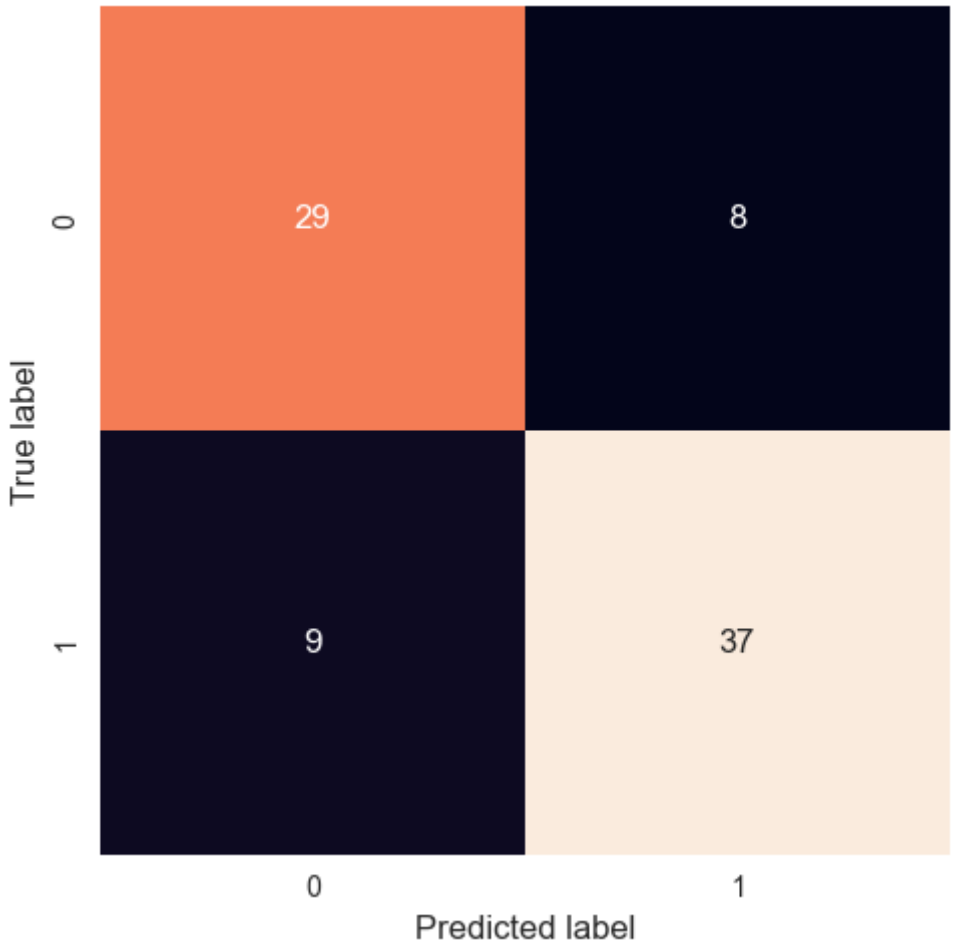


Figure 106: Random forest model ROC curve

8.2.2.2 Test set analysis and ROC curve for gradient boosted trees model

```
Test accuracy: 0.795
Recall score: 0.804
Precision score: 0.822
AUC score: 0.86
```



Figures 107 & 108: gradient boosted trees model performance

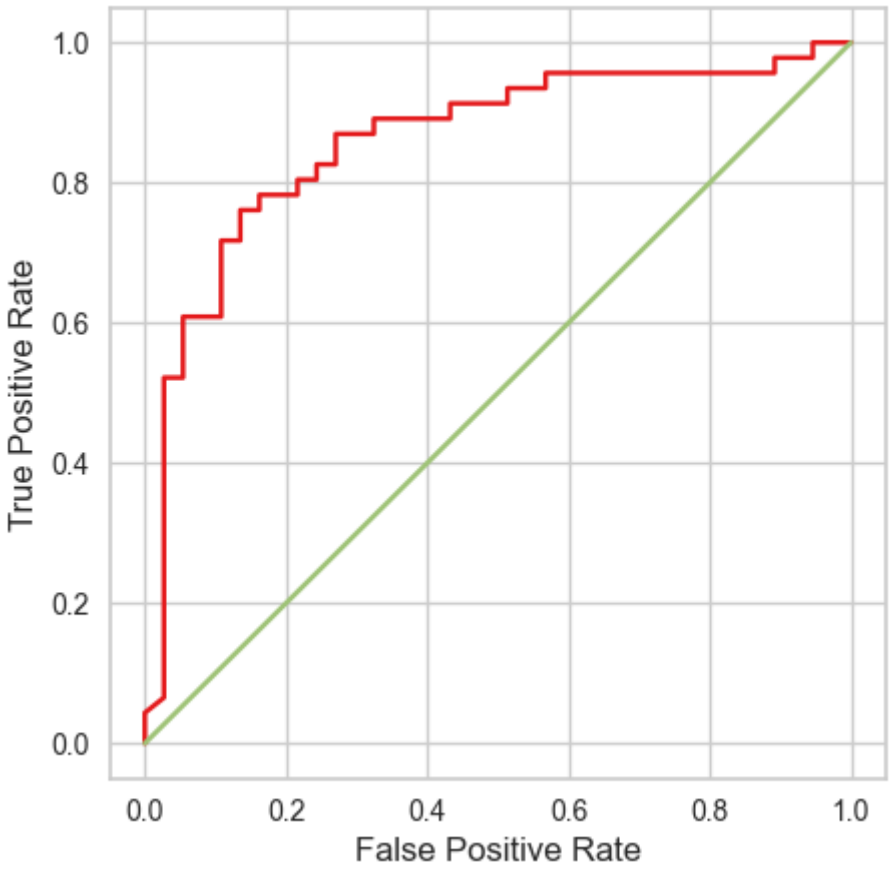


Figure 109: gradient boosted trees model ROC curve

8.2.2.3 Logistic regression

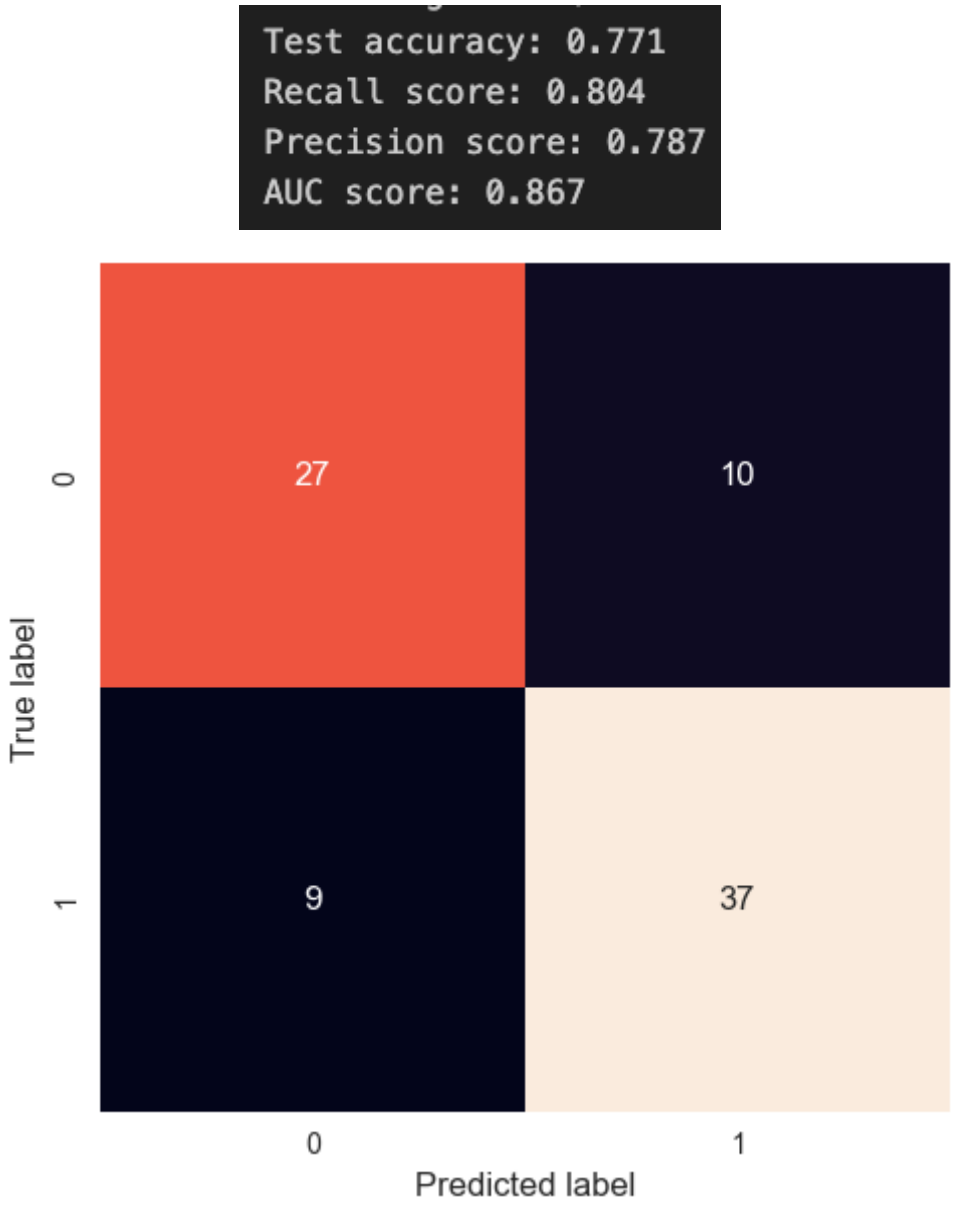


Figure 110 & 111: Logistic regression performance

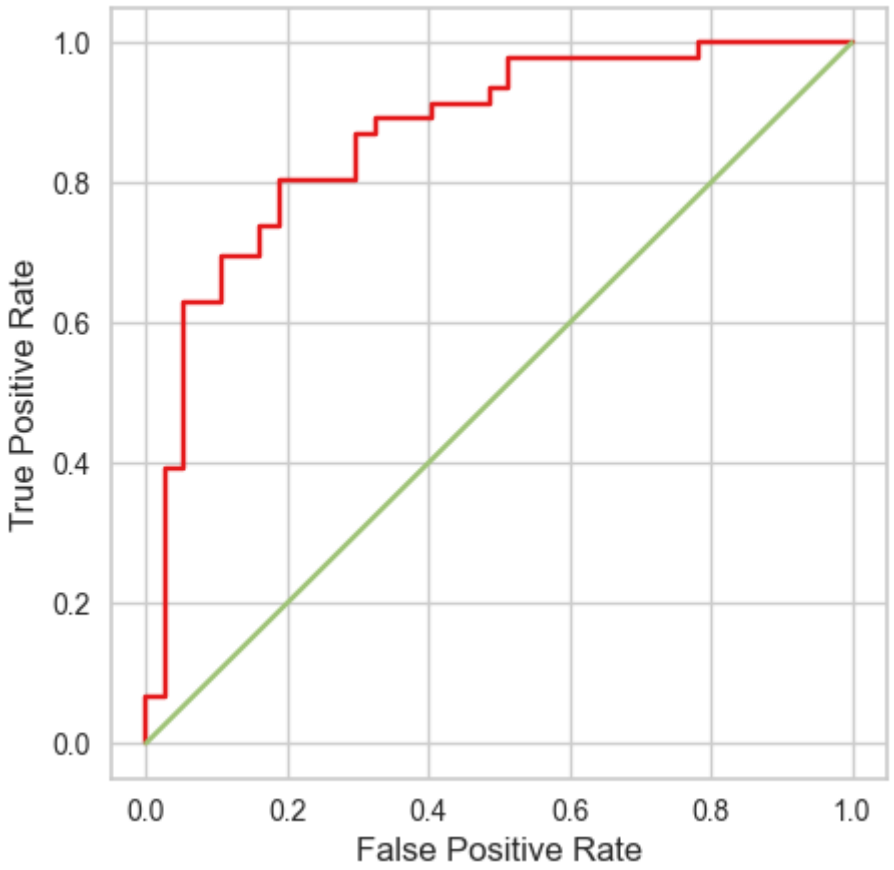


Figure 112: Logistic regression ROC curve

8.3 Interpreting the Results, Models and Patterns

8.3.1 Interpreting results and patterns for objective 1

In section 8.2.1.1 and 8.2.1.2 we take a close look into the most important predictive attributes and how they affect the models. We can see from the logistic regression model that very large portions of the final prediction are made up by highly influential categorical variables, the same of which is seen in the decision tree. As seen in figure 115, we can see that the first three levels of the decision tree are solely made up of categorical variables making them the splits which give the highest entropy gain and thus accuracy. Similarly we can see that 8 of the most important coefficients in the logistic regression model are flag variables, and the 9th most important variable beyond those 8 was also categorical. This suggests that for the most part categorical variables are extremely effective at categorising a large majority of samples, with the more fine grained samples being singled out by numeric variables.

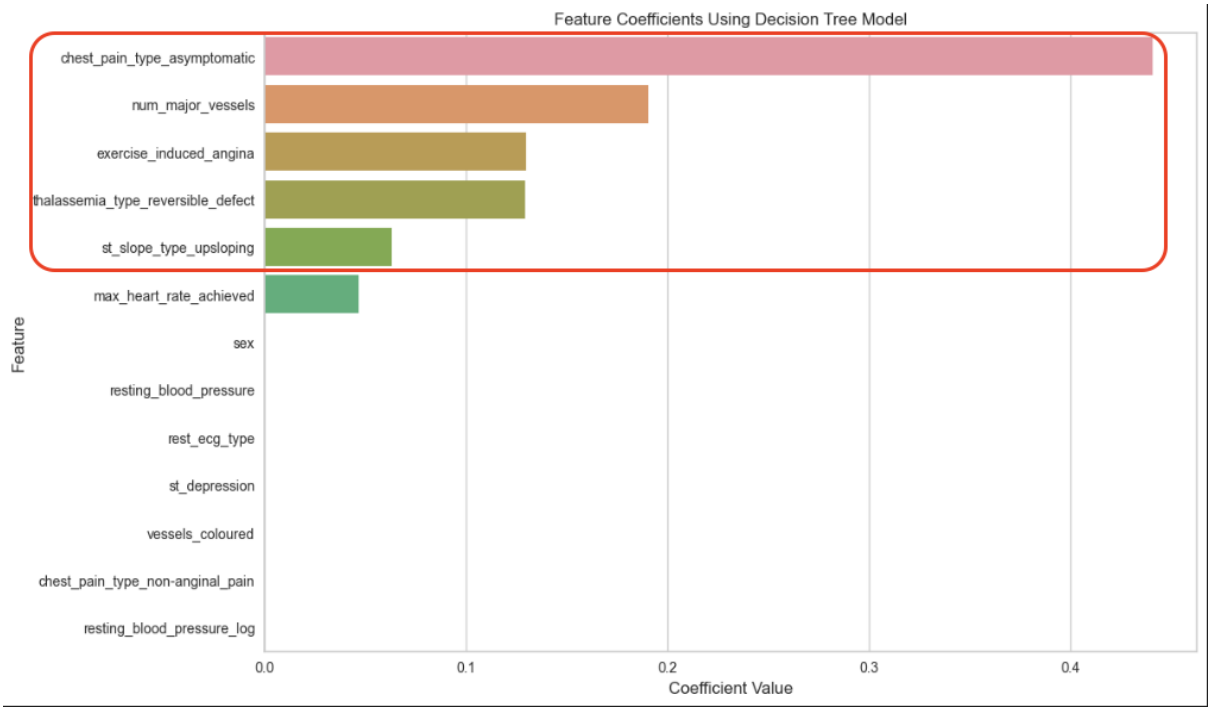


Figure 113: High influence of categorical variables on decision tree

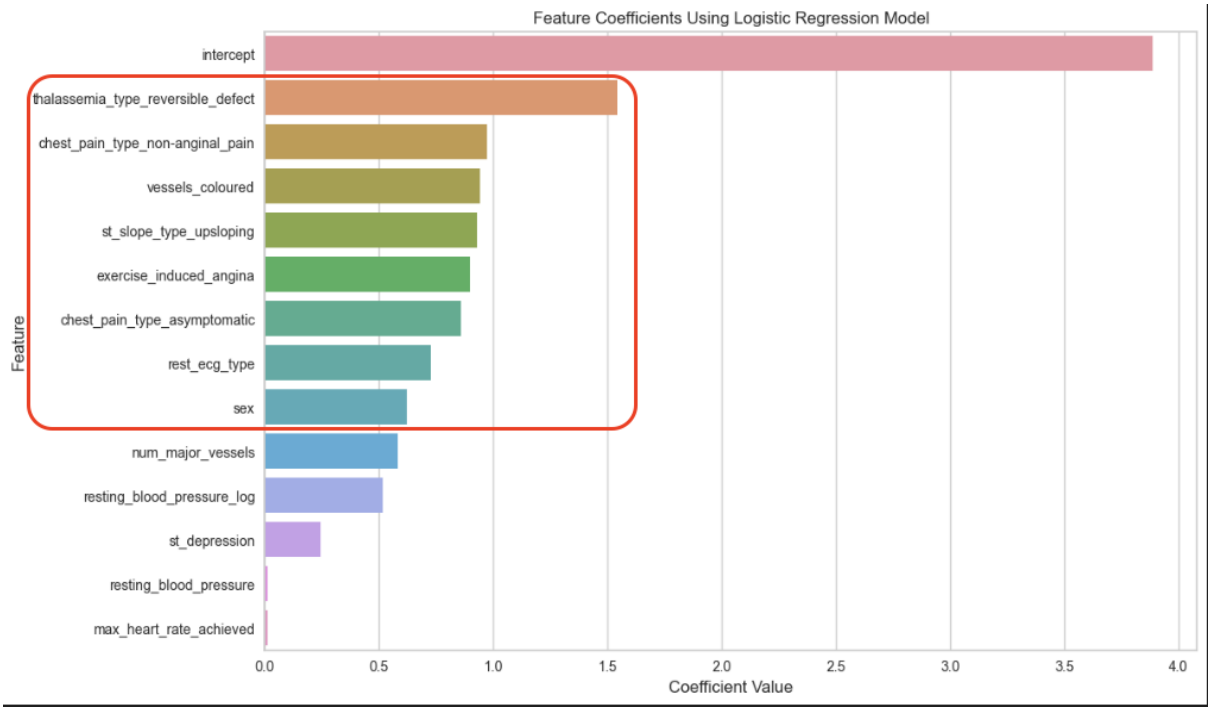


Figure 114: Influence of categorical variables in logistic regression model

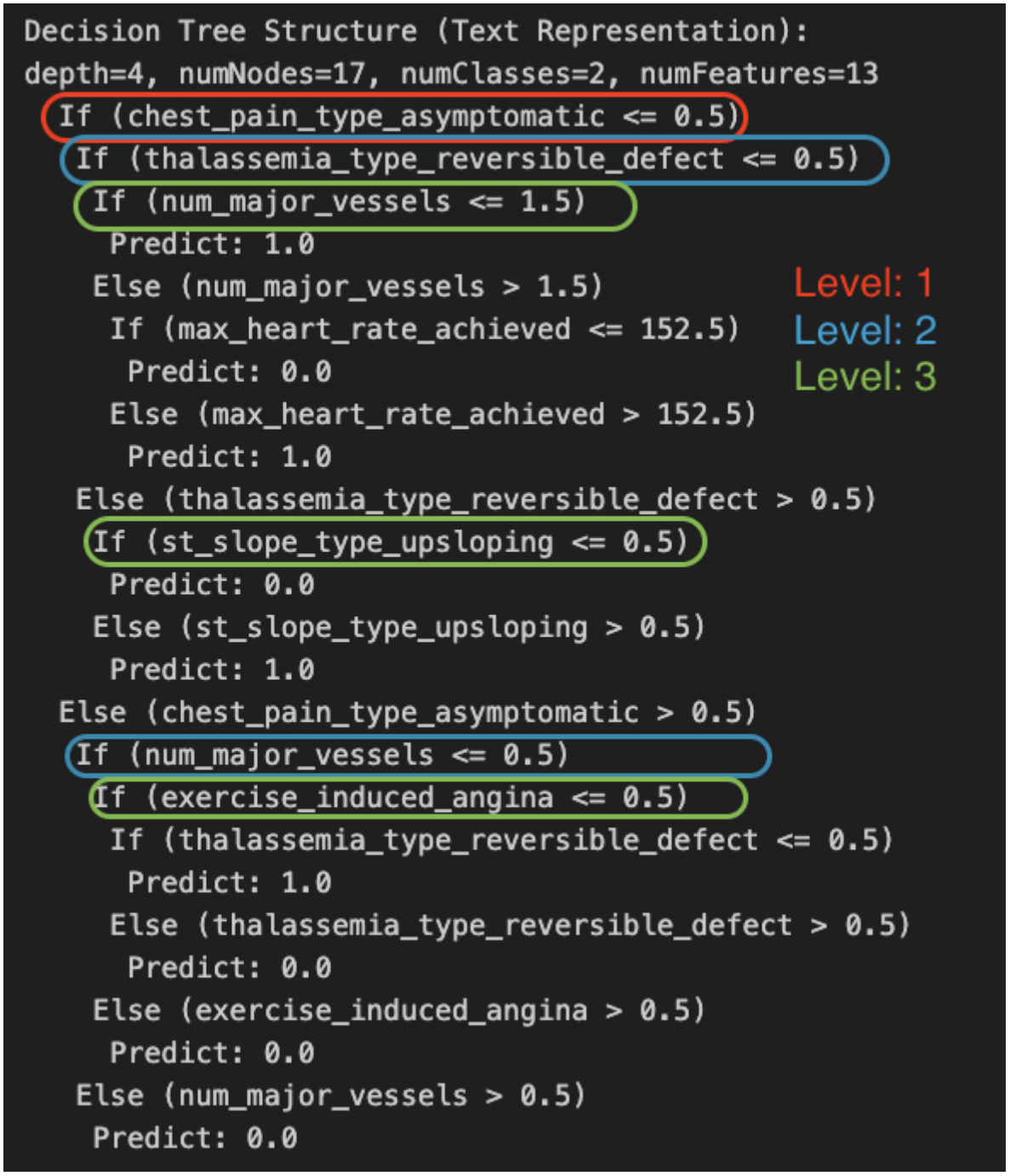


Figure 115: Prevalence of categorical variables in decision tree

Additionally a plot of the logistic models predictions was fit against the true labels and indicates that the model tends to favour extremes which we can observe in figure 120 below. This is a further indication that categorical variables make up the most important factors, since if continuous variables were the most important predictors we would expect to see a more spread out distribution of predictions. This domination of categorical variables further confirms the results we have found from our models where the thalassemia_type, vessels_coloured/num_major_vessels and chest_pain_type, exercise_induced_angina and st_slope_upsloping are some of the most influential variables when it comes to the presence of cardiovascular disease. Our results indicate that these categorical variables dominate the predictions, while other continuous variables mostly serve to fine tune the predictions,

particularly this is observable by the clump of absent heart disease predictions on the bottom left, which are likely due to the fact that most of our important categorical variables decrease the likelihood of heart disease. This compounds to a visible grouping of negative cardiovascular disease predictions.

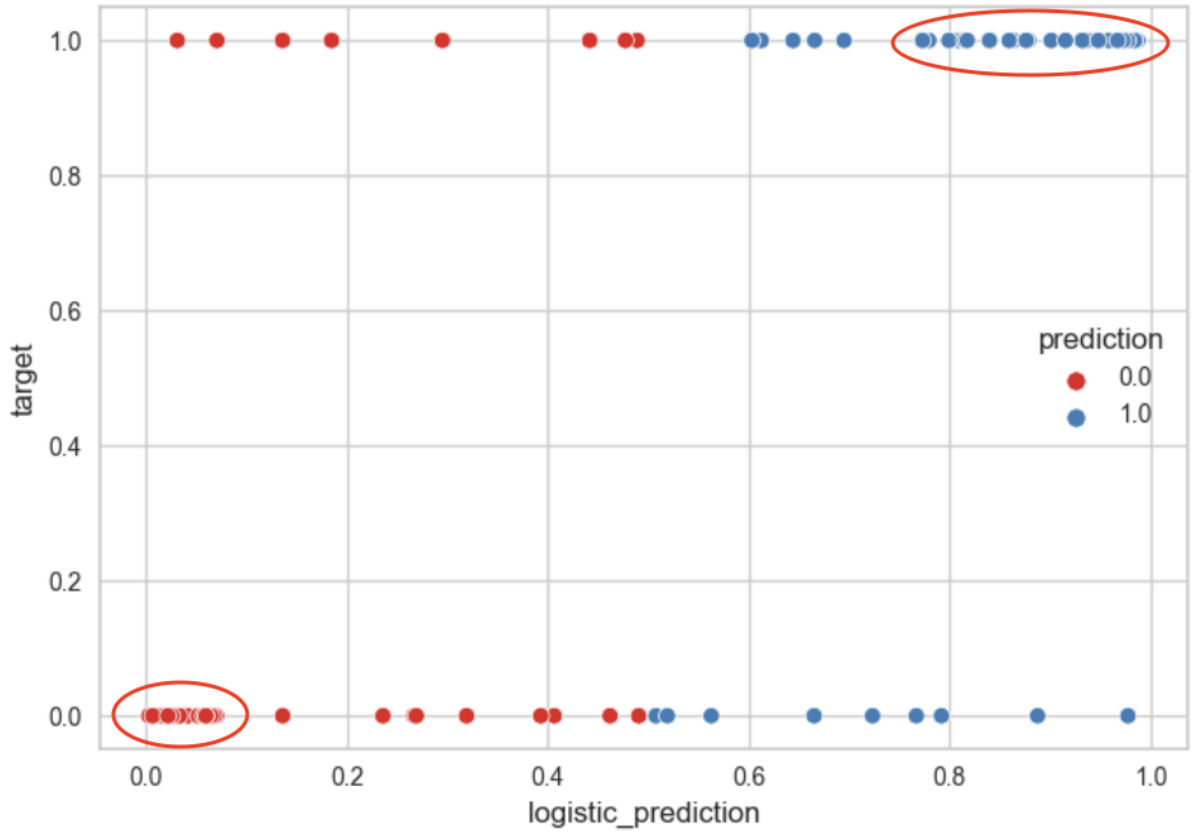


Figure 116: Logistic regression models preference for extremes

8.3.2 Interpreting the results and patterns for objective 2

The output from the ROC plots demonstrated in section 8.2.2 actually demonstrate that our models perform better in terms of recall (true positive rate) than it first appeared from our datamining results in section 7.3.2. These ROC plots demonstrate that a model with 90% recall is in fact a realistic possibility, one that did'nt look probable from the results seen in section 7.3.2. The ROC plots insights into the false positive rate we can expect if we alter our models for 90% recall. This is important since it is likely what would be recommended in practice, given the cost of extra testing and care on patients who may not need it is worth it to ensure every patient possible who does need care is identified. We can see that our random forest model is capable of achieving 90% recall with only roughly 27% false positive rate as seen in figure 117. Our gradient boosted trees model appears to perform the worst in terms of FPR and TPR tradeoff, only able to achieve an FPR of roughly 33% when getting 90% recall as shown in figure 118. Lastly the logistic regression model is able to achieve the second lowest FPR of the three models of roughly 32% when predicting with 90% recall as seen in figure 119.

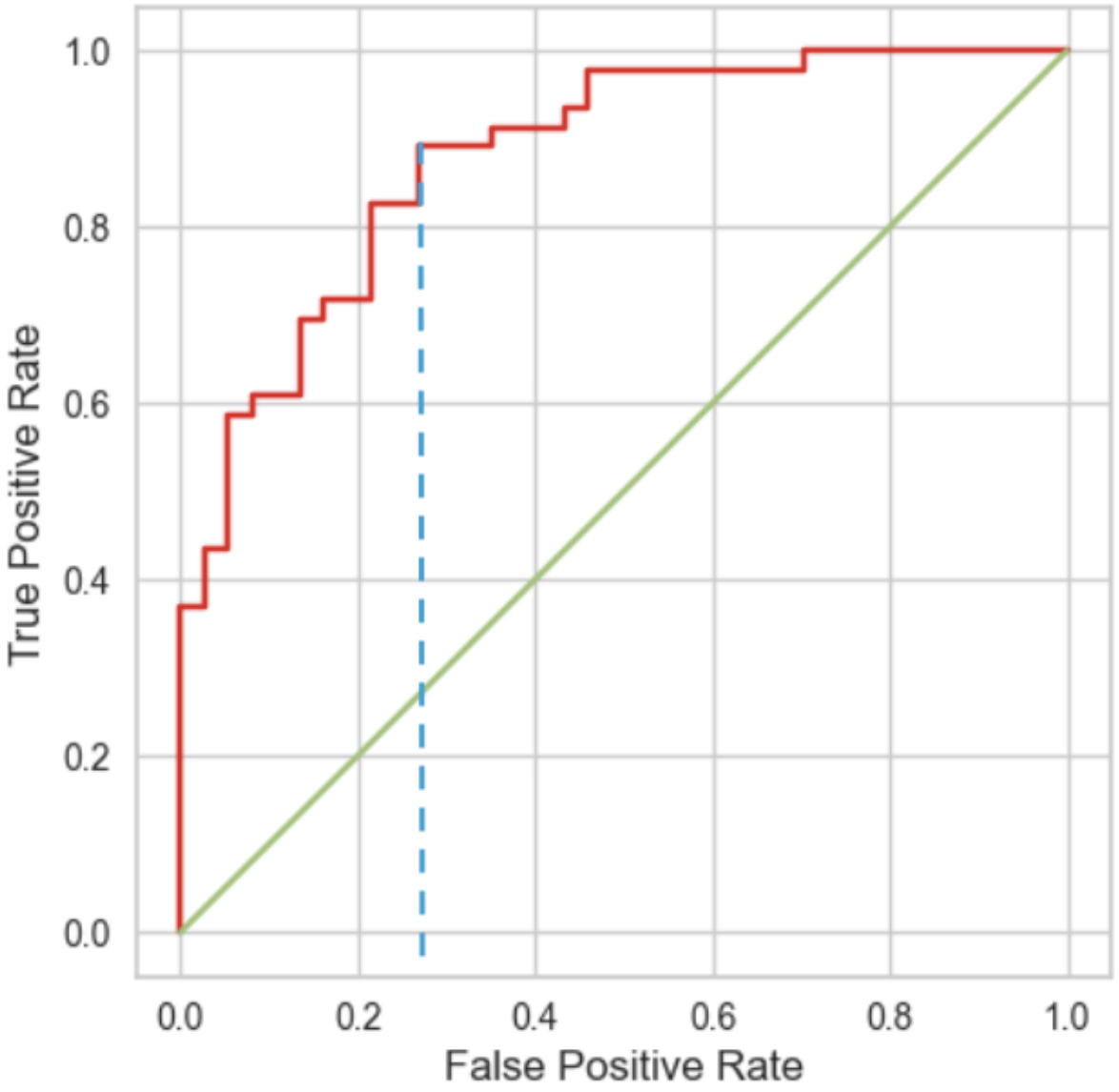


Figure 117: Random forest achievable FPR with 90% recall

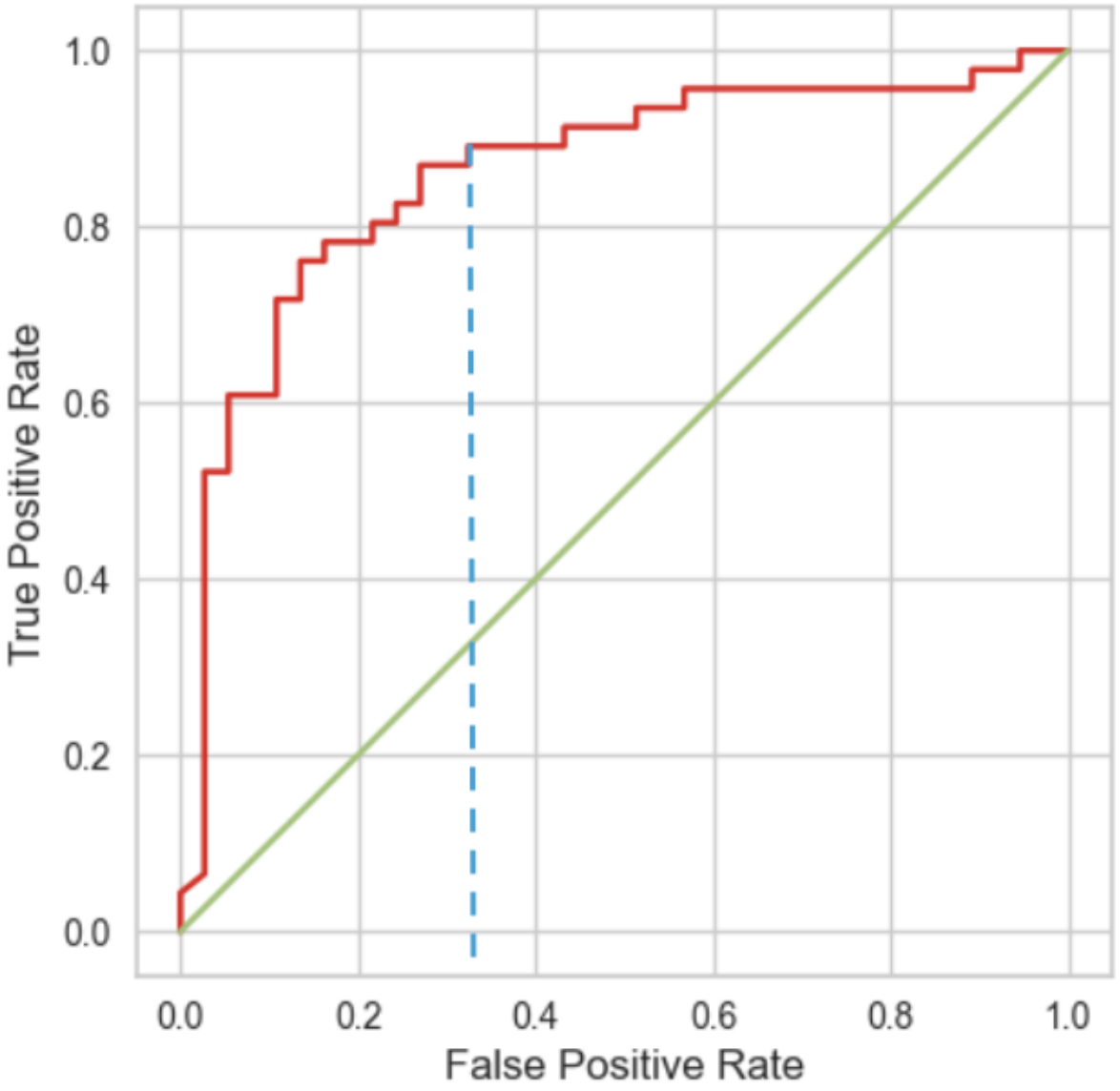


Figure 118: gradient boosted trees achievable FPR with 90% recall

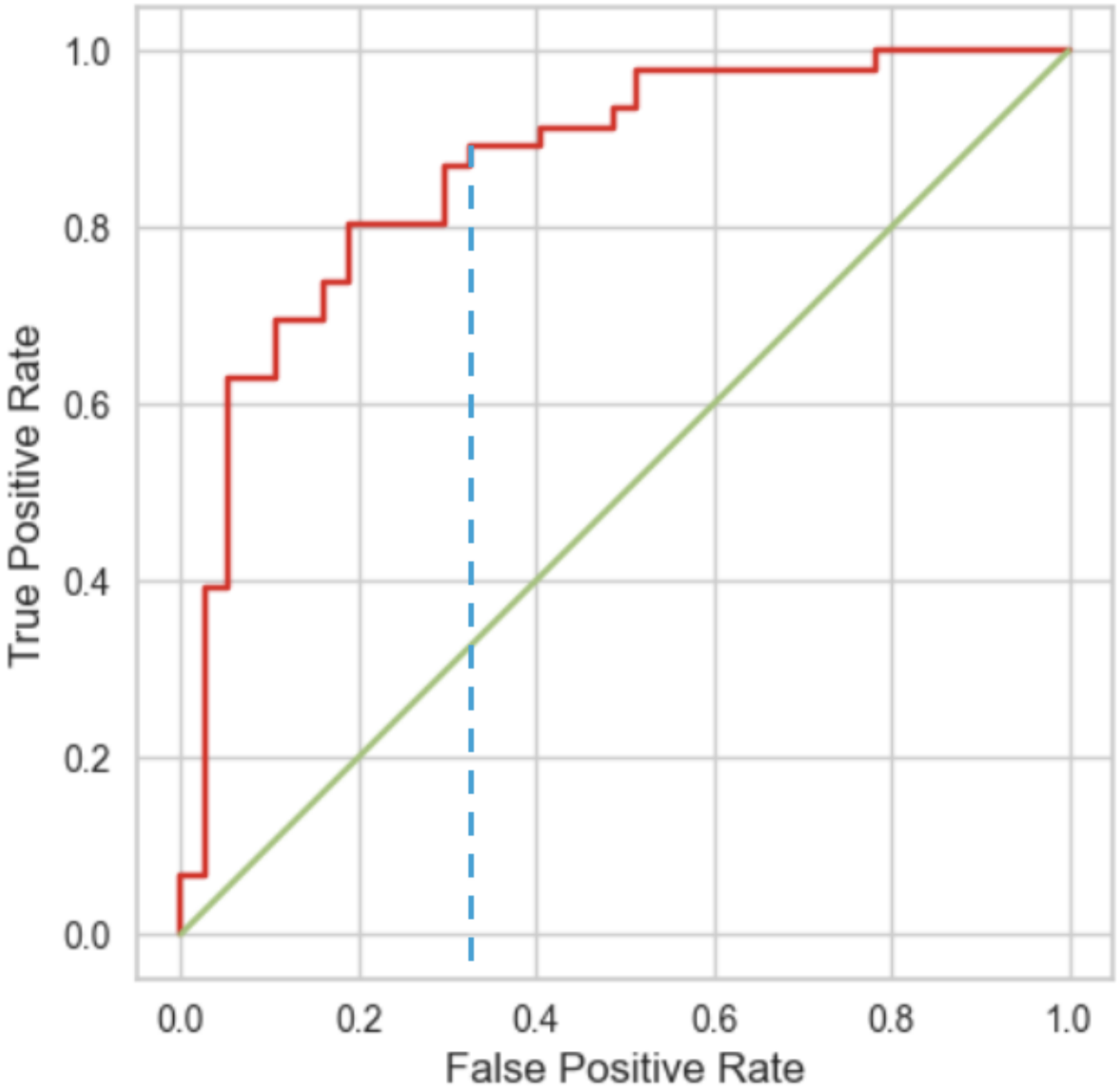


Figure 119: Logistic regression achievable FPR with 90% recall

In general, the large area under these ROC curves demonstrates that our models are effective in terms of the second objective of this data mining expedition, and with some iterative improvements will be capable of meeting the success criteria outlined in section 1.3.

8.4 Assessing and Evaluating Results, Models and Patterns

8.4.1 Assessing and Evaluating Results Models and Patterns for Objective 1

Our analysis of different models which revealed several different important predictors signified to us that the most important variables for the prediction of the presence of cardiovascular disease are num_major_vessels/vessels_coloured, thalassemia_reversable_defect, chest_pain_asymptomatic, exercise_induced_angina and st_slope_type_upsloping. The success criteria for objective 1 outlined in section 1.3 were “a

list of the significant predictors and critical levels for each, above which cardiovascular disease risk becomes significantly high", to explore the critical levels of these important variables we will exponentiate the coefficients of the final logistic model in order to get an understanding of the relative risk of cardiovascular disease attributed to each variable.

Coefficient	Value
thalassemia_type_reversible_defect	0.213503
vessels_coloured	0.389831
exercise_induced_angina	0.407434
chest_pain_type_asymptomatic	0.422758
sex	0.536812
num_major_vessels	0.558173
resting_blood_pressure_log	0.596066
st_depression	0.782168
resting_blood_pressure	0.987594
max_heart_rate_achieved	1.010707
rest_ecg_type	2.070845
st_slope_type_upsloping	2.533845
chest_pain_type_non-anginal_pain	2.645045
intercept	48.814311

Figure: 120: Exponentiated variable coefficients for logistic regression model

chest_pain_type:
Our decision tree predictor variable importance analysis revealed that for the decision tree, the most important variable was chest_pain_type_asymptomatic, a variable which was also found to be very important for our logistic regression. Our results from logistic regression demonstrate that patients with asymptomatic chest pain have 57.7% (1-0.423) reduced risk of cardiovascular disease relative to patients who had one of typical angina or atypical angina.
Additionally our logistic regression found that chest_pain_type_non-anginal_pain was important to prediction also. Our results demonstrate that patients with this chest pain were 164.5% (2.645-1) more likely to have cardiovascular disease than patients who had one of typical angina or atypical angina
This suggests the critical point where cardiovascular disease risk becomes concerning in terms of chest pain is when the pain is not non-anginal, and that cardiovascular disease risk is least concerning when the pain type is asymptomatic.

num_major_vessels:
Our analysis shows that when 0 major vessels coloured by fluoroscopy the risk of cardiovascular disease is significantly higher than if there are one or more blood vessels coloured. Furthermore we know the risk of cardiovascular disease decreases for each blood vessel that is coloured by fluoroscopy. We can see from figure 120 that the risk of

cardiovascular disease decreases by 61% (1-0.390) by having at least one blood vessel coloured by fluoroscopy, and is decreased by an additional 44.2% (1-0.558) for every blood vessel coloured by fluoroscopy. As a result we can say the risk of cardiovascular disease is concerning when no blood vessels are coloured by fluoroscopy, and is significantly less so otherwise.

thalassemia_type:
Our feature selection process in section 4.1 found that only the variable thalassemia_type_reversable_defect was worth keeping in the model, thus discarding the other two thalassemia types fixed_defect and normal, effectively grouping them together. Our exponentiated logistic regression coefficients in figure 120 demonstrate that a patient with a reversible defect type of thalassemia is 78.6% (1-0.214) less likely to have cardiovascular disease than otherwise. This demonstrates that patients at the highest risk are those with normal thalassemia or thalassemia with a fixed defect.

st_slope_type_upsloping:
Our feature selection process in section 4.1 found that only the variable upsloping st slope type was worth keeping in the model, thus discarding the other two slope types flat and downsloping, again effectively grouping them together. Our exponentiated logistic regression coefficients in figure 120 demonstrate that a patient with an upward sloping ecg segment is 152.4% (2.524-1) more likely to have cardiovascular disease than otherwise. This demonstrates that patients at the highest risk of cardiovascular disease are those with an upsloping st slope.

exercise_induced_angina:
The last variable which was found to be consistently important between both explanatory models was the exercise_induced_angina variable. The coefficients from our logistic regression model demonstrate that the patients who have chest pain triggered from exercising are 59.3% (1-0.407) less likely to have cardiovascular disease than patients who develop chest pain without any obvious trigger.

These results fulfil the success criteria set forth in section 1.3 and give valuable insights into which variables play the largest and most important parts in predicting the presence of cardiovascular disease, and exactly how much of a role they play.

8.4.2 Assessing and Evaluating Results Models and Patterns for Objective 2

After several iterations of model refinement unfortunately none of the three efficient models for prediction were able to satisfy the success criteria set forth in section 1.3. Our most effective models in terms of accuracy and recall were the random forest and gradient boosted trees models, which had exactly the same results for both of these metrics. However the most effective model of the three models fit was the random forest model, which managed to achieve 88% AUC, 2% larger than the gradient boosted trees model.

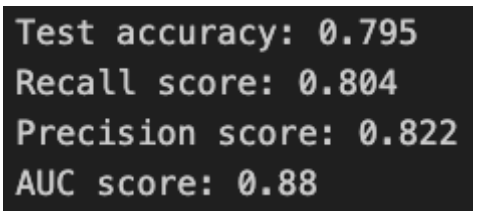


Figure 121: Random forest model performance

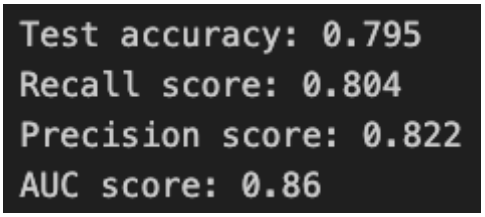


Figure 122: Gradient boosted trees model performance

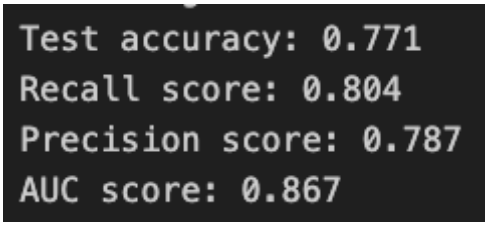


Figure 123: Logistic regression model performance

We can see from the results demonstrated in figures 121, 122 and 123 that all three models out of random forest, gradient boosted trees, and logistic regression achieved the same recall of 80.4%. However in terms of AUC the random forest out performed the other two models. Our initial rating criteria defined in section 5.1.2.3 was recall > AUC > accuracy, our investigation into the ROC curves of the two models found that the random forest model was able to achieve 90% recall while achieving a lower false positive rate than the gradient boosted trees model and logistic regression model. The combination of this fact alongside the superior AUC and accuracy demonstrate that the random forest model is the superior model for the problem at hand given in practice we will want to implement a model with at least 90% recall.

Considering the success criteria defined in section 1.3 as “a predictive model capable of reaching an overall accuracy of 85% and a recall value greater than 0.9”, we have unfortunately fallen short, so will need to implement iterative improvements in section 8.5 to attempt to get closer to this success criteria.

8.5 Iterations

The outcomes of the data mining expedition were promising, however there was still room for significant improvement. Because of this I undertook the iterative process of improving the research done thus far.

The main space for improvement within this research is generating a final decisive model which is the best model for the problem. In section 8.4 we discuss that our best model was

the random forest mode, however it was not able to meet either of the elements of the objective 2 success criteria set forth in section 1.3. As a result I decided to attempt to improve the model further within this section.

My first idea to improve the models was to implement an ensemble voting classifier which consisted of the three effective models which we fit throughout the data mining process being the logistic regression, random forest and gradient boosted trees models. I combined these models using a voting method by taking the average probability of each model in order to hopefully find a bit more consistency and benefit from the advantages of all models. The implementation of this averaging process can be seen in figure 124

```
from pyspark.sql.functions import col, lit, when
from pyspark.sql.functions import monotonically_increasing_id, row_number
from pyspark.sql.window import Window

voting_rf_predictions = rf_predictions.withColumnRenamed("probability", "rf_probability").select("rf_probability")
voting_logistic_predictions = logistic_predictions.withColumnRenamed("probability", "logistic_probability").select("logistic_probability")
voting_gbt_predictions = gbt_predictions.withColumnRenamed("probability", "gbt_probability").select("gbt_probability")
target = rf_predictions.select("target")
target = target.withColumn('index', row_number().over(Window.orderBy(monotonically_increasing_id())))

voting_rf_predictions = spark.createDataFrame(voting_rf_predictions.toPandas()["rf_probability"].apply(lambda x: x[1]).reset_index())
voting_logistic_predictions = spark.createDataFrame(voting_logistic_predictions.toPandas()["logistic_probability"].apply(lambda x: x[1]).reset_index())
voting_gbt_predictions = spark.createDataFrame(voting_gbt_predictions.toPandas()["gbt_probability"].apply(lambda x: x[1]).reset_index())

voting_predictions = voting_rf_predictions.join(voting_logistic_predictions, on=["index"]).join(voting_gbt_predictions, on=["index"])
voting_predictions = voting_predictions.select(((col("rf_probability") + col("logistic_probability") + col("gbt_probability")) / lit(3)).alias("probability"))
voting_predictions = voting_predictions.withColumn("prediction", when(col("probability") < 0.5, 0).otherwise(1))

voting_predictions = voting_predictions.withColumn('index', row_number().over(Window.orderBy(monotonically_increasing_id())))
target = target.withColumn('index', row_number().over(Window.orderBy(monotonically_increasing_id())))

voting_predictions = voting_predictions.join(target, on=["index"]).drop("index")

y_real = voting_predictions.select("target").toPandas()
y_pred = voting_predictions.select("probability").toPandas()

fpr, tpr, _ = roc_curve(y_real, y_pred)
roc_display = RocCurveDisplay(fpr=fpr, tpr=tpr).plot()
roc_display.figure_.set_size_inches(5,5)
plt.plot([0, 1], [0, 1], color = 'g')
plt.show()
```

Figure 124: Implementing ensemble voting method

When making the voting classifier I had to choose whether or not I wanted to use hard voting or soft voting to find the ensemble prediction. Soft voting involves taking the probability predictions of the presence of cardiovascular disease or lack thereof for each of the three models, adding them together and then taking the highest probability sum as the final class prediction. Alternatively hard voting involves taking the final target prediction from each of the three models and choosing the mode prediction as the final model prediction. In the end I decided soft voting made more sense for the problem given all the models performed so similarly, instead of taking the similar predictions, more information would be gained from the level of confidence each model had in their predictions. Interestingly the results of the voting classifier in figure 125 shows us that the model actually performs significantly worse than the initial random forest model, as shown by a very lacklustre ROC curve. These results suggest that the random forest model fit in section 7.3 is the best model that we can hope to achieve on our dataset.

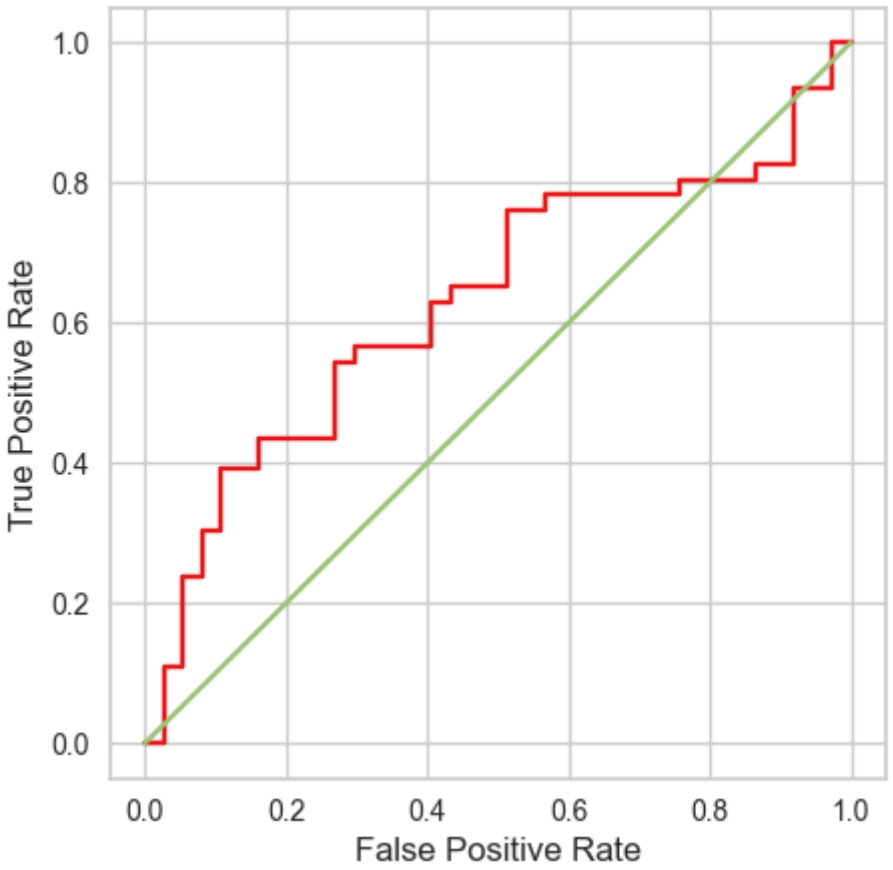


Figure 125: Voting ensemble model ROC curve

The next iterative improvement I made on the predictive model was to explore how shifting the prediction threshold affected the recall. By default all of the models fit thus far have had a prediction threshold of 0.5, meaning that if the predicted probability of cardiovascular disease was <= 0.5, then the model would predict no cardiovascular disease, and if the predicted probability of cardiovascular disease was > 0.5 then it would predict the presence of cardiovascular disease. As discussed previously however, in practice this cardiovascular disease model will be implemented to determine when patients need care, and as such should predict as few false negatives as possible. In order to ensure that our highest performing model - the random forest - is able to fulfil this, we will need to lower the prediction threshold such that any predicted probability of cardiovascular disease above this threshold will be classified as cardiovascular disease being present, and that there are only very few false negatives predicted. Here we will aim for recall of 90% as set forth in section 1.3. In order to determine the threshold at which this is possible, I calculated the predictions for a range of thresholds between [0,1], and evaluated the tradeoff between recall and accuracy over this threshold range, as shown in figure 126.

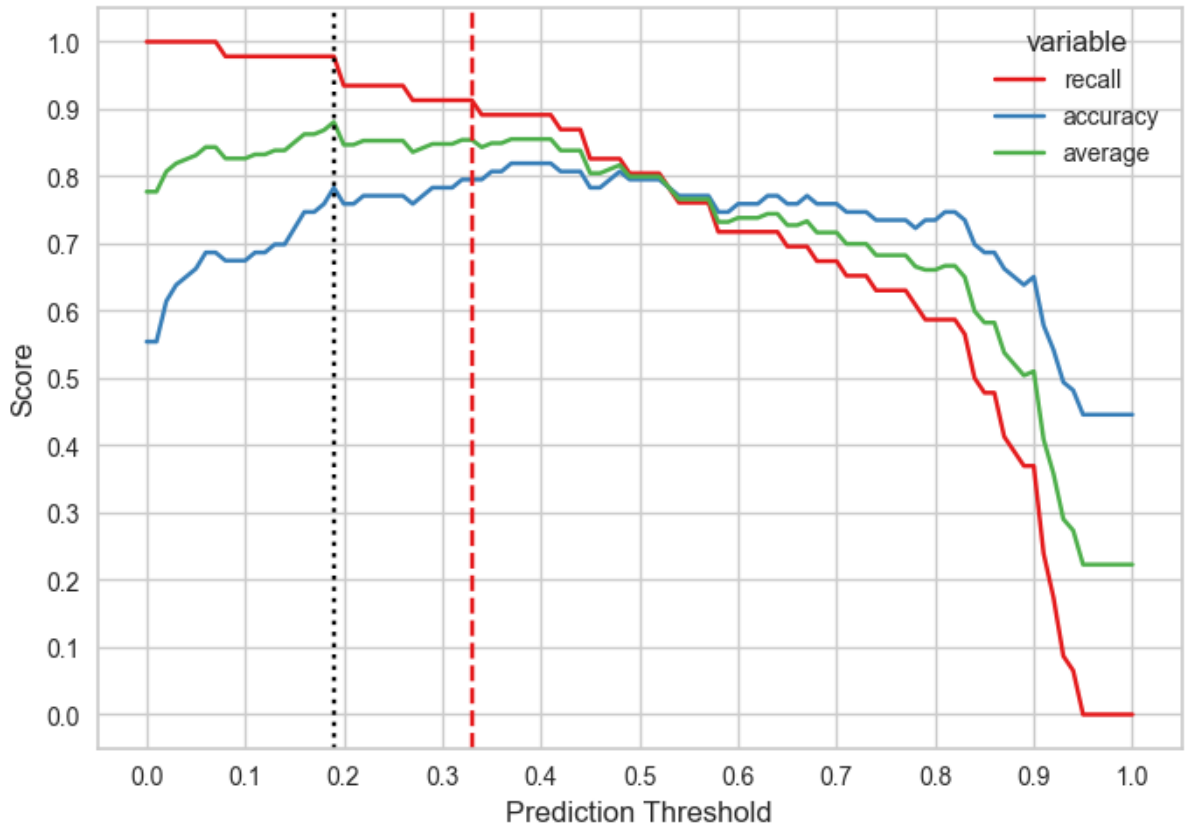
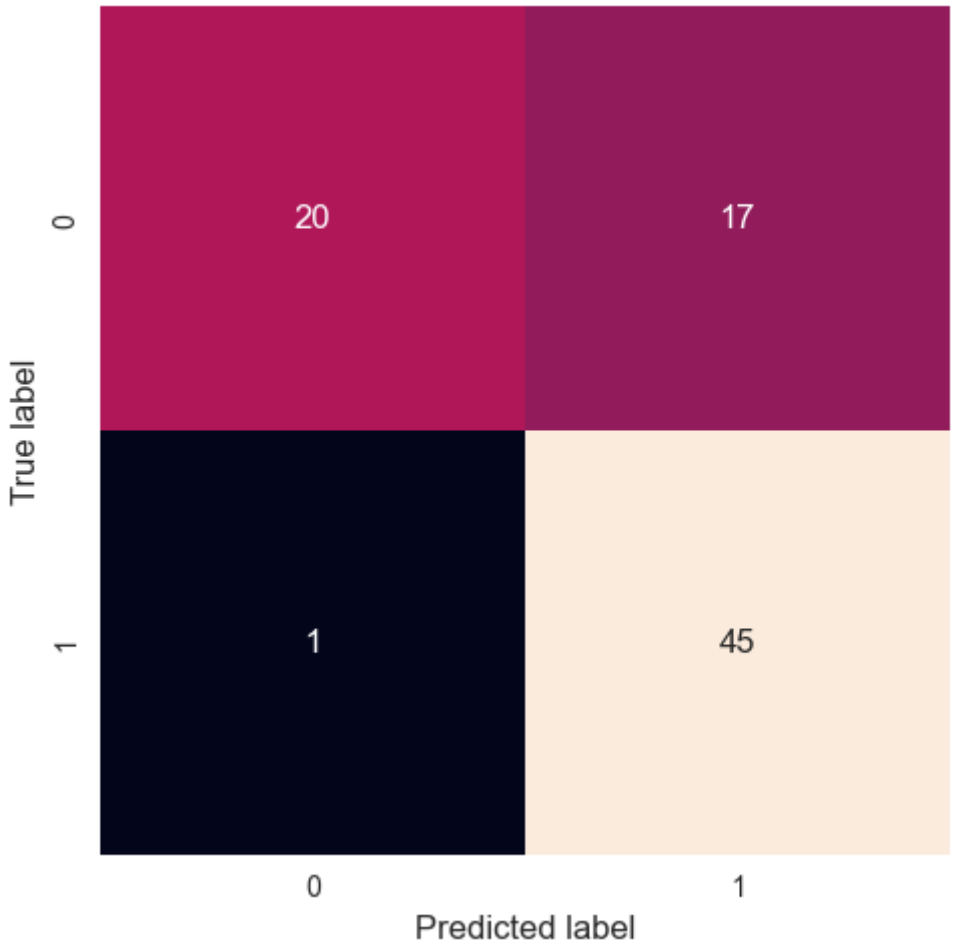


Figure 126: Tradeoff between recall and accuracy with prediction threshold

We find from figure 126 that the random forest model has the highest accuracy on the test set when the prediction threshold is set between 0.37 & 0.41, but more importantly we find that the recall of the model remains above 0.9 when the prediction threshold is set to 0.33 or lower. Unfortunately this plot demonstrates that there is no threshold at which the model can perform with over 85% accuracy as we set out in section 1.3. The threshold demonstrated by the dashed red line, demonstrates the threshold below which the model has over 90% recall. Furthermore we can see the highest average of accuracy and recall can be seen when the threshold is 0.19 as shown by the dotted black line in figure 126. While this model does not meet the success criteria, it is a very strong model for practical application as it manages to only generate a single false negative while attaining just shy of 80% accuracy. The accuracy metrics of the optimised model with a threshold of 0.19 can be seen in figures 127 and 128.

```
Test accuracy: 0.783
Recall score: 0.978
Precision score: 0.726
```



Figures 127 & 128: Random forest results with optimal prediction threshold of 0.19

The final iterative improvement I wanted to make on the final random forest model was to create a new model object which took a prediction threshold parameter natively. This was implemented such that in practice the model can be implemented with a more conservative or liberal prediction threshold as the user wishes. This model object is implemented in a simple nature, defining just a fit() and transform() method, such that the model can be fit to new data, and will predict positive classes according to the user defined predictive threshold which has a default value of 0.19, the optimum discovered from figure 126. The simple implementation of this new cardiovascular prediction model can be seen in figure 129.

```python
from pyspark.ml.classification import RandomForestClassifier

class CardiovascularModel():
    def __init__(self, estimator = None, threshold=0.33):
        self.threshold=threshold
        if estimator != None:
            self.base_estimator = estimator
        else:
            self.base_estimator = RandomForestClassifier(featuresCol="features", labelCol="target", seed=722,
                                                         numTrees=400,
                                                         impurity="entropy",
                                                         featureSubsetStrategy="auto")

    def fit(self,x):
        self.base_estimator = self.base_estimator.fit(x)
        return self.base_estimator

    def transform(self,x):
        x = self.base_estimator.transform(x)
        new = x.toPandas()
        new["probability"] = new["probability"].apply(lambda x: x[1])
        new["prediction"] = new["probability"].apply(lambda x: 0 if x < self.threshold else 1)
        x = spark.createDataFrame(new.reset_index())
        return x
```

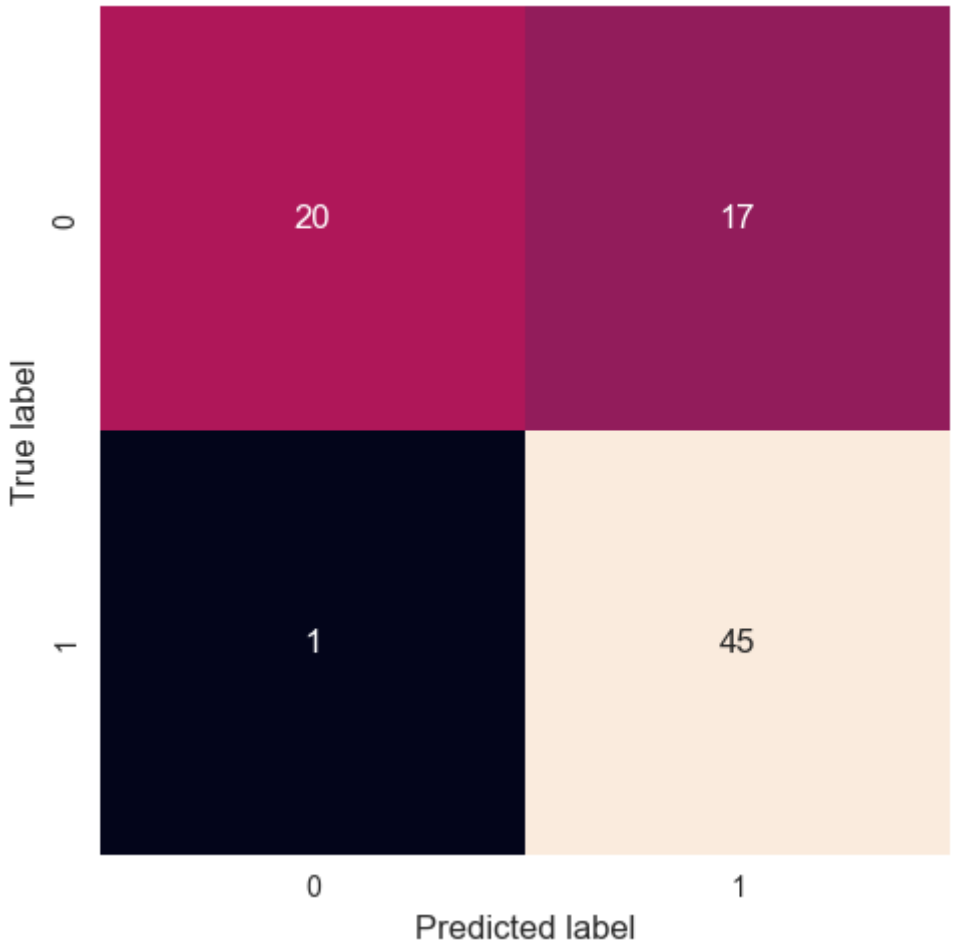
Figure 129: CardiovascularModel implementation with predictive threshold compatibility

I allowed this CardiovascularModel class to accept a predefined classifier, or to create a new random forest classifier with the optimal parameters of the model fit in section 7.2. To demonstrate the models capability I first implemented it with the saved random forest model from section 7.2 as shown in figures 130, 131 & 132, and also trained a new model with the threshold of 0.33 as shown in figures 133, 134 & 135. Both models work as intended and generate recall values greater than 90%, with near optimal accuracy.

```python
#final results
cardio_model = CardiovascularModel(estimator=rf)
cardio_results = cardio_model.transform(test_assembled)
accuracy_metrics(cardio_results)
```

Figure 130: Implementing and fitting final cardiovascular disease prediction model

```
Test accuracy: 0.783
Recall score: 0.978
Precision score: 0.726
AUC score: 0.88
```



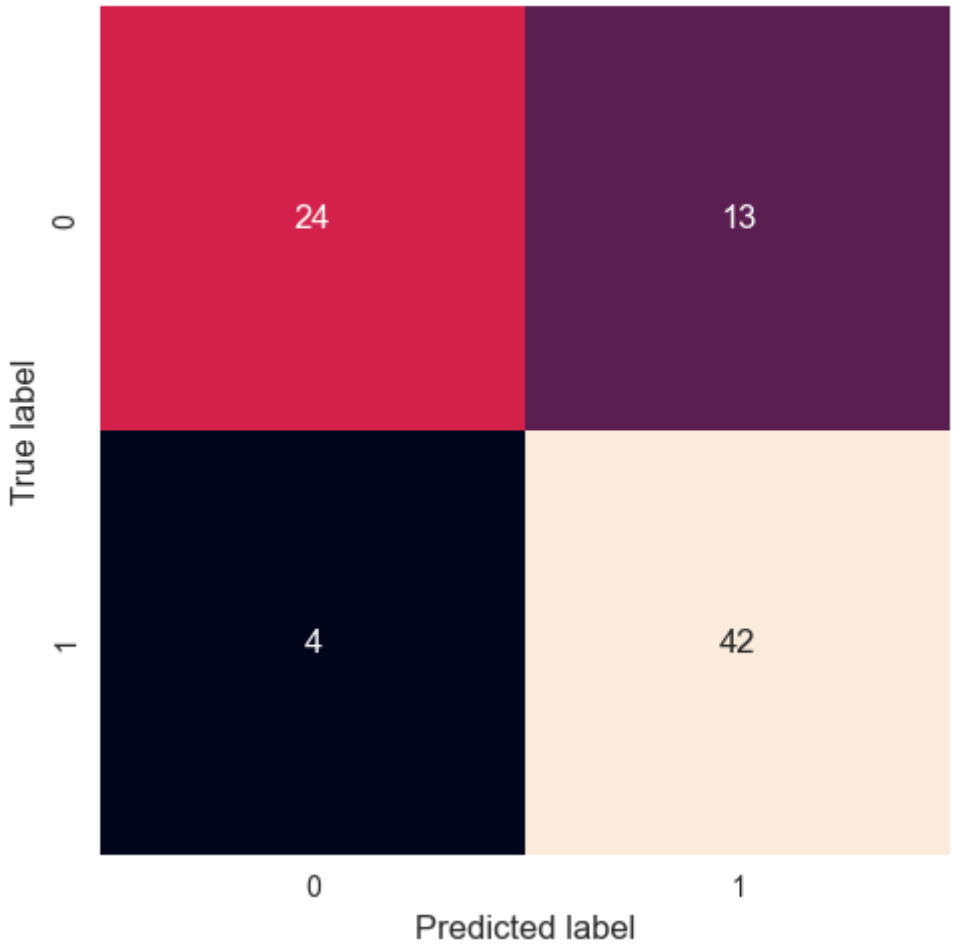
Figures 131 & 132: Final cardiovascular model results

```
new_cardio_model = CardiovascularModel(threshold=0.33)
new_cardio_model.fit(train_assembled)
new_cardio_results = new_cardio_model.transform(test_assembled)

accuracy_metrics(new_cardio_results)
```

Figure 133: Demonstration of new model being fit and tested

```
Test accuracy: 0.795
Recall score: 0.913
Precision score: 0.764
AUC score: 0.88
```



Figures 134 & 135 cardiovascular model with 0.33 threshold results

With these final improvements in place I am happy with the ability of the final model to effectively identify the presence of cardiovascular disease in patients. These final improvements to the recall of the model make it a model which can be used effectively in practice to root out and recommend testing to many at-risk patients. Unfortunately the model is incapable of meeting both of the success criteria set forth for the model in section 1.3, as it is incapable of achieving accuracy greater than 85%, however it is capable of 97.8% recall, with 78.3% accuracy, performance which would certainly be sufficient in a practical environment.

References

Chan, W. C., Wright, C., Riddell, T., Wells, S., Kerr, A. J., Gala, G., & Jackson, R. (2008). Ethnic and socioeconomic disparities in the prevalence of cardiovascular disease in New Zealand. *The New Zealand Medical Journal (Online)*, *121*(1285).

Harris, K. C., Benoit, G., Dionne, J., Feber, J., Cloutier, L., Zarnke, K. B., Padwal, R. S., Rabi, D. M., & Fournier, A. (2016). Hypertension Canada's 2016 Canadian hypertension education program guidelines for blood pressure measurement, diagnosis, and assessment of risk of

pediatric hypertension. Canadian Journal of Cardiology, 32(5), 589–597. https://doi.org/10.1016/j.cjca.2016.02.075

Lapp, D. (2019, June 6). heart disease dataset. Kaggle. https://www.kaggle.com/datasets/johnsmith88/heart-disease-dataset

Disclaimer

I acknowledge that the submitted work is my own original work in accordance with the University of Auckland guidelines and policies on academic integrity and copyright. (See: https://www.auckland.ac.nz/en/students/forms-policies-and-guidelines/student-policies-and-guidelines/academic-integrity-copyright.html).

I also acknowledge that I have appropriate permission to use the data that I have utilised in this project. (For example, if the data belongs to an organisation and the data has not been published in the public domain then the data must be approved by the rights holder.) This includes permission to upload the data file to Canvas. The University of Auckland bears no responsibility for the student's misuse of data.