

Predicting Users' Film Ratings and Classifying Heart Conditions from ECGs

G14SML Statistical Machine Learning

Group project report

2018/19

School of Mathematical Sciences

University of Nottingham

Group A:

Tim Gardiner

Jue Hou

Joshua Joyce

Yesser Naji

Anni She

We have read and understood the School and University guidelines on plagiarism. We confirm that this work is our own, apart from the acknowledged references.

Abstract

This project aims to create accurate models to predict potential movie ratings from users as well as to diagnose patients' heart conditions based on their ECG readings. Previous user ratings were used in the training set to extract any particular preferences and also to group users with similar preferences together, whilst the diagnosis required an analysis of how ECGs look for particular conditions. The movies ratings part of the project utilised Linear Regression, K-Nearest Neighbours and Collaborative Filtering for predictions on the Test data. Here it was found that a simple linear model outperformed any other model tried, obtaining an $MSE_{test} = 0.82$. The heart Diagnosis part of the project focused on extracting important features from the ECGs and applying K-Nearest Neighbours, decision Trees and random Forests to predict each patients health condition in the test set. Here it was found that random forests using features, similar to what cardiologists use when interpreting ECGs, performed better than any other model achieving an $MCE_{test} = 0.24$

Contents

I	Predicting Users' Film Ratings	6
1	Introduction	6
1.1	Background	6
1.2	Aim	7
1.3	The Data	7
1.4	Overview of methods	9
2	Methodology	12
2.1	Linear Regression model	12
2.2	Simple Linear Models	14
2.3	Collaborative Filtering	19
3	Conclusions	26
3.1	Results	26
3.2	Report Limitations	28
II	Diagnosing Heart Conditions from ECGs	28
4	Introduction	29
5	Motivation	30
5.1	Project Motivation	30
5.2	Theoretical motivation: Cardiologist's methods	31
6	The Data	35
7	Previous research	38
7.1	Feature extraction	39
7.2	Methodologies	40

8	Method	41
8.1	Heart Beat separation	42
8.2	Analytical method on mean heart beats	44
8.3	Feature Extraction	49
8.4	Limitations	60
9	Decision Trees and Random Forests	61
10	Conclusions	65
III	Appendices	68
A	Film Test Data	68
B	Feature Extraction Algorithm	69
IV	Bibilography	73

Part I

Predicting Users' Film Ratings

1 Introduction

The first problem that this report will concern itself with is predicting how users' would rate a film based on how they have already rated other films. For this project the data set consists of a total of 72753 user movie ratings based off of 8090 films. However, in this huge database, it should be noted that some movies have not been rated by some users which can end up causing problems in predictive models as will be discussed later on. The aim of this report is to use this training set to train a model to make predictions of how users will rate movies in the test set, possible ratings options are $\{0.5, 1, 1.5, 2, 2.5, 3, 3.5, 4, 4.5, 5\}$.

One of the main problems for machine learning programmes in tasks such as this one is the sparse nature of the data. Not every individual will watch and rate every film and as a result many of the ratings for each individual will be missing. A quote from Montgomery et al. explains how this issue grows the longer the rating system has been around for, thus resulting in the need to account for factors, such as taste, that change with time. Montgomery et al. [1] states that problems with sparsity arise as the number of users and items increase in a system which 'affects the performance of a recommender system as the accuracy of predictions decreases. Thus, there is a need for a technique that one can perform efficiently under sparse environments' [1]. This is an important issue in recommender systems and will be continually addressed as this report progresses.

1.1 Background

This challenge is central to a famous group of systems called recommender systems. Companies such as Amazon, YouTube and Netflix invest a substantial amount of research into developing these systems so they can predict what items users will be interested in and recommend them accordingly. Taking Youtube as an example, it is easy to see how important these systems are to companies. It is predicted that 70% of watch time on Youtube

is a result of recommended videos [32] and seeing as in 2018 it was projected that Youtube generated \$3.9 B, [28] in advertising revenue, this accounts for roughly \$2.7 B, illustrating the necessity of these algorithm to large companies. Referring back to the project in this report, how many users like a movie is evaluated by ratings in the range of 0.5 to 5. These recommender systems are also important as companies such as Netflix need users to enjoy the content in order to keep subscribers, so they need to give good recommendations meaning having an accurate recommender system.

1.2 Aim

In this report several methods are going to be built in order to predict the ratings \widehat{r}_{ij} , where \widehat{r}_{ij} is the predicted rating of user i for film j . How well these models are performing will be measured using the mean square error (MSE) of the prediction, calculated as:

$$\frac{1}{n_{test}} \sum_{i=1}^{n_{test}} (r_{ij} - \widehat{r}_{ij})^2 \quad (1.1)$$

where r_{ij} are the true rating systems.

In order to avoid overfitting the model, since the true rating values of the test set are unknown, k-folds cross validation will be used to split the training set in order to evaluate the performance of the models we use [2].

1.3 The Data

There are three relevant data sets for this problem, these are ‘ratings_train’, ‘ratings_test’ and the ‘movies’. The training set is comprised of 671 users, 8090 films and a combined number of 72753 film ratings meaning that not every user has rated every film, in fact one user rated just 9 films while the user who rated the most films still only rated 1727. This shows just how sparse the data is as even the most avid film assessor has only rated 21% of the available films. The data set also includes the time when the user made the rating as can be seen in the extract of the data in figure 1.

The time stamp here, however, is not in the standard date-time format of YYYY-MM-DD and thus it must be transformed into the usual version for ease of interpretation. This can be seen in figure 2.

	userId	movieId	rating	timestamp
1	89	1385	3.0	1257610479
2	294	4816	4.0	1047072493
3	397	413	2.0	1268904250
4	273	5378	4.0	1466946309
5	213	4343	5.0	1462644087
6	213	54001	3.5	1462636406
7	388	2011	3.0	946523740
8	564	818	4.0	974838390
9	461	2108	2.0	1091959304
10	547	3886	3.5	1053171261

Figure 1: The first 10 lines of the 'ratings_train' data set

	userId	movieId	rating	timestamp
1	89	1385	3.0	2009-11-07 16:14:39
2	294	4816	4.0	2003-03-07 21:28:13
3	397	413	2.0	2010-03-18 09:24:10
4	273	5378	4.0	2016-06-26 14:05:09
5	213	4343	5.0	2016-05-07 19:01:27
6	213	54001	3.5	2016-05-07 16:53:26
7	388	2011	3.0	1999-12-30 03:15:40
8	564	818	4.0	2000-11-21 20:26:30
9	461	2108	2.0	2004-08-08 11:01:44
10	547	3886	3.5	2003-05-17 12:34:21

Figure 2: The first 10 lines of the 'ratings_train' data set with the amended timestamp

The dataset 'ratings_test' is a CSV consists of gives the userId, movieId and the timestamp but without the 'ratings' column as that is what the developed models will be predicting. An excerpt from this data set with the timestamp transformed into the standard format can be seen in appendix A.

The final dataset, 'movies', allows us to find the issue year, title and genres of each movieId contained in the 'ratings_train' and 'ratings_test' data sets. This can be seen below in figure 3.

This causes some issues with data extraction as the release year needs to be extracted from the 'title' column and it is also necessary to know what possible genres a film has as this information is important for the model building that will take place later.

The lists of possible years and genres are shown below.

	movieId	title	genres
1	1	Toy Story (1995)	Adventure Animation Children Comedy Fantasy
2	2	Jumanji (1995)	Adventure Children Fantasy
3	3	Grumpier old Men (1995)	Comedy Romance
4	4	Waiting to Exhale (1995)	Comedy Drama Romance
5	5	Father of the Bride Part II (1995)	Comedy
6	6	Heat (1995)	Action Crime Thriller
7	7	Sabrina (1995)	Comedy Romance
8	8	Tom and Huck (1995)	Adventure Children
9	9	Sudden Death (1995)	Action
10	10	GoldenEye (1995)	Action Adventure Thriller
11	11	American President, The (1995)	Comedy Drama Romance
12	12	Dracula: Dead and Loving It (1995)	Comedy Horror
13	13	Balto (1995)	Adventure Animation Children
14	14	Nixon (1995)	Drama
15	15	Cutthroat Island (1995)	Action Adventure Romance

Figure 3: The first 10 lines of the ‘movies’ dataset.

```
[1] "1902" "1915" "1916" "1917" "1918" "1919" "1920" "1921" "1922" "1923" "1924" "1925" "1926" "1927" "1928" "1929" "1930"
[18] "1931" "1932" "1933" "1934" "1935" "1936" "1937" "1938" "1939" "1940" "1941" "1942" "1943" "1944" "1945" "1946" "1947"
[35] "1948" "1949" "1950" "1951" "1952" "1953" "1954" "1955" "1956" "1957" "1958" "1959" "1960" "1961" "1962" "1963" "1964"
[52] "1965" "1966" "1967" "1968" "1969" "1970" "1971" "1972" "1973" "1974" "1975" "1976" "1977" "1978" "1979" "1980" "1981"
[69] "1982" "1983" "1984" "1985" "1986" "1987" "1988" "1989" "1990" "1991" "1992" "1993" "1994" "1995" "1996" "1997" "1998"
[86] "1999" "2000" "2001" "2002" "2003" "2004" "2005" "2006" "2007" "2008" "2009" "2010" "2011" "2012" "2013" "2014" "2015"
[103] "2016"
```

Figure 4: List of all of the years that the movies were released in

```
[1] "Action" "Adventure" "Animation" "children" "Comedy" "Crime" "Documentary" "Drama"
[9] "Fantasy" "Film-Noir" "Horror" "IMAX" "Musical" "Mystery" "Romance" "Sci-Fi"
[17] "Thriller" "War" "Western"
```

Figure 5: List of all possible genres from the ‘movies’ dataset

1.4 Overview of methods

The following section will give a brief outline of the methods that this report used to predict users’ film ratings. Through previous research on the topic of recommender system, regression methods are used frequently with regression analysis being a statistical technique for investigating and modelling the relationship between variables [1]. Of these analytical methods the most basic is linear regression so that is where this report will begin its approach. Other reports such as Schafer et al. [27] and Ekstrand [26] use models based on collaborative filtering, finding users with similar interests and basing predictions off of their known ratings, a simplified version of this is K-nearest neighbours (KNN), used to find similar users or ‘neighbours’.

1.4.1 Linear Regression

The linear model has been chosen to begin with due to its simplicity. One issue with the model is that it makes the assumption that ratings can be continuous, while this is not

true it is a reasonable assumption to be made in the range of 0.5 to 5, and any values that fall outside this range can be scaled accordingly. There are a number of steps to take when considering the linear model. Firstly, features of each movie, such as genres and year of distribution, must be extracted to be used as the covariates. Then, based on each user, a corresponding objective function will be found that needs to be minimised, thus finding the relevant parameters. Finally, these parameters will be used along with the test covariate vectors to get the predicted values. Some of the simple linear regression models that will be investigated in the methodology include:

- A preliminary model using mean user ratings to make predictions produced a low MSE, making it ideal to use simple linear models based on the mean user ratings and possibly mean movie ratings to improve on an already low MSE .
- The simple model with a new covariate added to reflect the interaction between users and movies. This interaction is movie and user specific and helps to account for a users' affinity to a specific film due to a particular actor or director taking part.
- A final covariate will be added representing the relationship between the mean movie ratings and issue years to account for how ratings change based on when a film was released.

1.4.2 Collaborative Filtering

The main idea of the user-based collaborative filtering is to find a user's neighbours who have similar movie preferences. How the original user will rate a movie is then assumed to depend on how their neighbours have rated the movie. The method that will be used to find the user's neighbours is KNN and the similarities between users will be defined by Pearson Theorem, this will be discussed in depth in section 2.3.1 . Also in this section the optimum k will be found for KNN by plotting the mean MSE based on 10-fold cross validation with different values of K ranging from 30 to 350. This investigation will indicate that the MSE is the lowest when k is 185.

Three main issues arise for this method, all of which result in still getting NA values

among the output predicted ratings. This is because 672 of the movies in the test set are not in the training set and have, therefore, not been rated by anyone in the training set. Other films may not have been rated by any of a users' neighbours, again resulting in NA ratings. And finally there is a phenomenon known as the Grey Sheep phenomenon which is when a users' taste is unique that they do not have any neighbours. It would be expected that in a data set as large as the one used in this report that the Grey Sheep phenomenon will be nullified, however it is still something to be taken into account. These issues can be solved by assigning a value to all of the NA ratings, the value chosen was 3.5 because this is the mean of all ratings in the training set. It will be seen that this near arbitrary value of 3.5 does not perform very well, this report will therefore combine the collaborative filtering model with the linear model by assigning the ratings predicted by linear model to the corresponding "NA" value in collaborative filtering model, thus overcoming the NA issue in a more informed way. Though in this case, the MSE we get by predicting ratings by the users mean rating and the NA values are assigned by 3.5 is 0.922, which is relatively accurate. However, it is careless to predict the ratings by using quick baselines because different users have different preferences for different movies, and rating may be biased.

1.4.3 Baseline

One final issue that should be addressed is what is known as the 'baseline' effect. This is a fault of recommender systems that occur due to differences between people, their tastes and how they use rating systems. One problem here arises because one user may have a strong preference towards action movies and will, therefore, be more inclined to give an action film a higher rating. Another issue occurs with how a user's preference evolve over time, just because they used to like comedies 5 years ago does not mean they like them as much now. All of these considerations can create issues in the model and need to be carefully considered in order to avoid inaccurate predictions. These baseline effects will be mentioned and considered throughout the methodology section as will all of the issues talked about here.

2 Methodology

The following section will cover the methods tried, their performance and how they were built. This will lead into the conclusion where full model comparisons will be made with a final model presented and appropriate conclusions being drawn.

2.1 Linear Regression model

In this model it is assumed that the response variable y , is a linear function of $p - 1$ covariates labelled x_1, \dots, x_{p-1} , then the general linear regression model is shown as [3]:

$$y = \theta_0 + \theta_1 x_1 + \dots + \theta_{p-1} x_{p-1} \quad (2.1)$$

where $\theta_i, i = 0, 1, \dots, p - 1$ are the parameters corresponding the relevant x_i 's.

For this method, we assume that we know the value of each feature for each film and thus we denote the features of the movies as the covariates $x_i, i = 1, \dots, p - 1$ and the response variable y is assumed by movie ratings r_{ij} . Now we are going to discuss which covariates were chosen to be used in this model.

Firstly, from the data set 'movies' discussed in 1.3, we find that each movie has its own genres, i.e. each movie can have its corresponding feature values. To begin with, each genre will be regarded as an individual feature. In this case there are 19 genres:

we denote $\mathbf{X}_j = (x_1, x_2, \dots, x_{19})$ as the feature vector for movie j . The value x_1 represents action films, x_2 represents adventure films and so on. The elements of \mathbf{X}_j are defined as indicator variables, so if movie i 's genres were action and comedy then the values of x_1 and x_5 in \mathbf{X}_i would be set equal to 1 and all of the other indicators would be set equal to 0. If a movie has no listed genres then every element of \mathbf{X}_j would be set equal to 0.

The parameter vector for user i can then be defined as $\boldsymbol{\theta}_i = (\theta_1, \dots, \theta_{19})$. This vector can be interpreted as the extent that user i likes each specific genre.

The rating of movie j by user i , denoted \hat{r}_{ij} , is then calculate using the following linear combination of $\boldsymbol{\theta}_i$ and \mathbf{X}_j [4]:

$$\hat{r}_{ij} = \boldsymbol{\theta}_i^T \mathbf{X}_j \quad (2.2)$$

Since the parameter vectors are unknown, we now treat predicting the ratings of each user as a linear regression model and therefore we get the corresponding objective function as:

$$\min_{\theta_i} \frac{1}{2m_i} \sum_j (\theta_i^T \mathbf{X}_j - y_{ij})^2 \quad (2.3)$$

Where m_i is the number of movies that user i has rated; y_{ij} is the true ratings for user i and the 2 in the denominator is for easier computation. It should be noted that when calculating the MSE_{test} based on the movie set that user i has rated, movies that the user has not rated are not taken into consideration. After optimizing equation 2.3 a parameter vector θ_i is generated for each user.

Since we have obtained θ_i , we can then predict the ratings using 2.2. The performance of this model, however, is fairly poor with an $\text{MSE}_{test} \approx 2.037$ when calculated using k-folds on the training data. It is therefore necessary to reflect on the model and look at what covariates could be added in order to improve it.

Recall the 'movies' dataset, it was mentioned in 1.3 that each movie has an associated year of distribution. Hence, this could be used as a feature to improve on the first model. An sample of the extracted years from the data is shown in the figure below.

Here, it is decided to separate the feature into 3 time periods and treat them as indicator variables to reflect how people tend to view movies from different eras. These three time periods are:

- before 1970
- from 1970 to before 2000
- after and including 2000

With these three new features, three more covariates, x_{20}, x_{21}, x_{22} , must be added to \mathbf{X}_j and 3 more parameters, $\theta_{20}, \theta_{21}, \theta_{22}$ must be added to θ_i . The three new covariates are again indicator variables with $x_{20} = 1$ and $x_{21} = x_{22} = 0$ if the film occurs before 1970 and so on. If a movie does not have a year of distribution then all three of the new covariates are set equal to 0. The new parameters, $\theta_{20}, \theta_{21}, \theta_{22}$, can be interpreted as how much the user likes a film from the corresponding time period. This method did provide some issues as one of the distribution years was '2007-' and '1975-1979', in these situations the earlier

date was chosen arbitrarily. This split into three time periods could also be improved upon by testing how many time periods to split the data up into, however, three was chosen in order to help create a simple model without too many parameters. Using the same idea as before, the MSE_{test} using k-fold cross validation was around 1.1902, and the MSE for the test data was 1.22, this is a good improvement from the first model, implying that the inclusion of the year covariates improved the model. This means that when considering further models the year of distribution will be strongly considered as a variable.

This method carries with it numerous pros and cons as will now be discussed:

Advantages: This method eliminates the need to deal with sparse data, this saves time and confusion in deciding what to assign in place of the 'NA' values.

Disadvantages Some of the predicted ratings are greater than 5 and some are smaller than 0.5 which is not only invalid but also increases the MSE. Our solution is to change these out of range ratings to 5 and 0.5 respectively as those are the maximum and minimum ratings that can be given. The aforementioned MSE values were calculated after this change was made. This model performed well however there is still room for improvement. The next model that will be tried is a simplified version of the first two models in an attempt to improve the performance.

2.2 Simple Linear Models

2.2.1 Foreword

Before introducing simple linear models, it will be beneficial to first introduce three simple ways of predicting user ratings. These are:

1. Choose the population mean of the ratings in the "ratings_train" data set as predicted ratings:

$$\widehat{r}_{ij} = \text{mean rating of "ratings_train"} \quad (2.4)$$

2. Predict the rating of any film rated by user i using their mean rating from "ratings_train"

$$\widehat{r}_{ij} = \text{user } i\text{'s mean rating from ratings_train} \quad (2.5)$$

In this case there may be NA values while calculating the mean since the user may not have existed in the training set but does exist in the test set. In these situations the ‘NA’ value is replaced with the population mean for all ratings in “ratings_train” which is 3.5.

3. Similar to the model in equation 2.5, but instead any rating for film j is predicted using that films mean from “ratings_train”.

$$\widehat{r}_{ij} = \text{film } j\text{'s mean rating from ratings_train} \quad (2.6)$$

Again ‘NA’ values can arise due to the film not being in the training set and therefore these values will again be set as 3.5.

The resulting MSEs’ calculated using 10-fold cross validation can be seen in table 1

	Mean Rating	Users’ Mean	Movie’s Mean
MSE	1.1220	0.9318	1.0099

Table 1: Predicted ratings for the three methods given in 2.4, 2.5 and 2.6 respectively

From table 1 it can be seen that the model based off of users’ mean ratings, corresponding to equation 2.5, performed noticeably better than the other two models, giving an MSE of 0.9318. When testing this model on the test data it achieved an MSE of 0.922. The best MSE of any of these models is still fairly close to 1, however, and can therefore be improved. One possible route would be to consider the users’ mean and movie’s means models and form a multivariate model based off of these two.

The following sections will now investigate the general linear model and which covariates should be included in it.

2.2.2 Model Introduction

Three main models will now be introduced and compared

Model 1: For this model, it is assumed that the predicted ratings are composed of a baseline rating, denoted c_0 , a user specific effect and a movie specific effect [5]:

$$r_{ij} = c_0 + c_1 * \text{user specific effect}_i + c_2 * \text{movie specific effect}_j \quad (2.7)$$

The user specific effect refers to the fact that some users naturally give higher ratings than other users, the following equation shows how the user specific effect was calculated.

$$\text{user specific effect}_i = \text{user } i\text{'s mean rating} \quad (2.8)$$

The mean of a users' ratings was used as it perfectly reflects how they like to rate films. A user with a lower mean rating score should have a user specific effect that reflects this and vice versa.

For the same reason as above the movie specific effect is calculated using the movie's mean, as shown in equation 2.9.

$$\text{movie specific effect}_j = \text{movie } j\text{'s mean rating} \quad (2.9)$$

As before, 'NA' values are replaced with 3.5 where necessary.

The ratings do not, however, solely depend on the user specific effect and the movie specific effect. It also makes logical sense for them to be dependent on how much user i likes movie j

Model 2 In order to reflect this interaction between user i and movie j a new covariate, called specific interaction, is introduced to the model. The predicted model now becomes:

$$r_{ij} = c_0 + c_1 * \text{user specific effect}_i + c_2 * \text{movie specific effect}_j + c_3 * \text{specific interaction}_{ij} \quad (2.10)$$

The values of $\theta_i^T \mathbf{X}_j$ from section 2.1 are used as the values of the specific interaction,

$$\text{specific interaction}_{ij} = \theta_i^T \mathbf{X}_j \quad (2.11)$$

This model still does not, however, consider all of the interactive effects present in the data. An example of this is that the mean of the overall movies' ratings may be influenced by their year of distribution, figure 6 illustrates the relationship between these two variables. Model 2 can be used with \mathbf{X}_j being defined in either form from 2.1, that is either (x_1, \dots, x_{19}) or (x_1, \dots, x_{22}) . When using model 2 to make predictions, the same problems as before arise, that is that some of the ratings that the model is giving as predictions are outside of the desired range. This problem is solved in the same way as before, by mapping those predictions that fall outside of the [0.5:5] range to the closest value within said range.

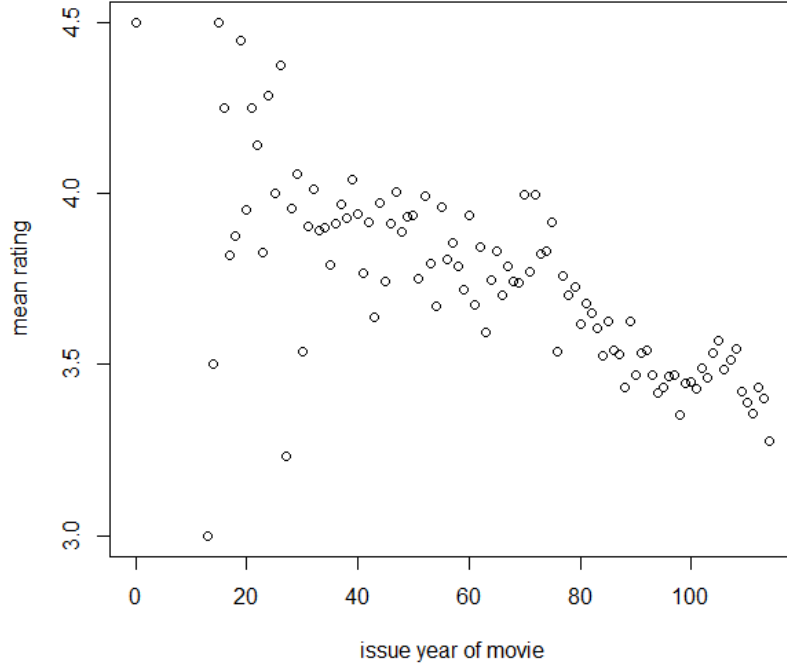


Figure 6: A plot showing how movie mean ratings have a decreasing trend with the issue year of the movies

Model 3: The last model that will be considered adds a final covariate, denoted m_t , which accounts for the relationship between movie mean ratings and the year in which the movie was distributed:

$$r_{ij} = c_0 + c_1 * \text{user specific effect}_i + c_2 * \text{movie specific effect}_j + c_3 * \text{specific interaction}_{ij} + c_4 * m_t \quad (2.12)$$

This movie rating and issue year interaction term is defined in the following way; first, the linear regression model for mean movie rating based on age must be found:

$$\text{mean movie rating} = 4.146897 - 0.006392 * \text{movie's age} \quad (2.13)$$

Then, each movie's age is substituted into the linear regression model in 2.13. This then returns the corresponding mean movie rating based on when the film was distributed, which will be denoted as m_t . In this model, specific interaction only considers genres, $\mathbf{X}_j = (x_1, \dots, x_{19})$ but does not consider the year in which the movie was distributed, since this is reasoning for the inclusion of m_t . This model can have problems as some

films do not have an associated year of distribution, in these situations the corresponding m_t was set to the mean m_t which is the mean of the mean movie ratings by movie's age.

2.2.3 Simple Linear Models Conclusion

All three of these methods performed well when calculated using cross validation on the training set as can be seen in table 2 below.

Model	MSE_{test}
model 1	0.8466
model 2 (a) ($\mathbf{X}_j = (x_1, \dots, x_{19})$)	0.8426
model 2 (b) ($\mathbf{X}_j = (x_1, \dots, x_{22})$)	0.8373
model 3	0.8423

Table 2: Predicted ratings for the three methods given in 2.7, 2.10 and 2.12 respectively

The results in table 2 show little to no difference between model 1, model 2 (a) and model 3, while model 2 (b) outperforms all of the other models.

Aside from the results there are a couple of other comments that can be made about the simple linear models:

1. Situations arise where movies or users do not appear in the training set, so users may not have made ratings before and some movies may not have been rated yet. In these models this problem is solved by setting the missing data equal to 3.5 however this can be improved upon.
2. The linear models also have some issues with predicting inside of the standard rating range of 0.5 to 5 due to linear regression predicting values in \mathbb{R} . This has been solved by setting ratings above 5 to be equal to 5 and those below 0.5 to be equal to 0.5. This, however, can also be improved upon.

The following section will now introduce the idea of using a collaborative filtering model.

2.3 Collaborative Filtering

User based collaborative filtering is a method used to mine a small number of users, who have similar preferences to one another, amongst large datasets. In collaborative filtering, these users are referred to as neighbours and the method predicts users' ratings based on how their neighbours have already rated films.

One of the most classic examples of collaborative filtering is watching movies. More often than not, users are unable to decide which movie they would like to watch as they are spoilt for choice. The usual method of deciding is to ask friends and see what good movie recommendations they give. When an individual asks their friends for movie recommendations, they are more inclined to ask someone with a similar taste in films to themselves. This is the essence of collaborative filtering.

As was mentioned in section 1.4, many previous papers such as [10] have included collaborative filtering in their recommender systems. Through learning about collaborative filtering this report concluded that user-based collaborative filtering will be used. This is the the idea of finding users' with the same film preferences and is performed using the KNN method.

2.3.1 KNN

As stated by Cover and Hart in their report 'Nearest Neighbour pattern classification', 'the KNN algorithm is an intuitive and effective non parametric model used for both classification and regression purposes' [11]. This part of the project will focus on regression properties. The algorithm uses 'feature similarity' to predict values of any new data points. This means that the new points are assigned values based on how closely they resemble the points in the training set.

The first step to this method is to choose a measure of distance to help define what is meant by 'near' in k-nearest neighbours. For two vectors, \mathbf{x} and \mathbf{y} , each of length k , the two most common distance measures used are [12]:

1. The Euclidean distance:

$$\sqrt{\sum_{i=1}^k (x_i - y_i)^2}$$

2. The Manhattan distance:

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

After calculating the distances between each of the users, the k-nearest users to the individual in question are selected. Predictions are made by calculating the mean of the ratings given by these k neighbours for the film in question, film j , in the following way.

$$\frac{1}{k} \sum_{i=1}^k r_{ij}$$

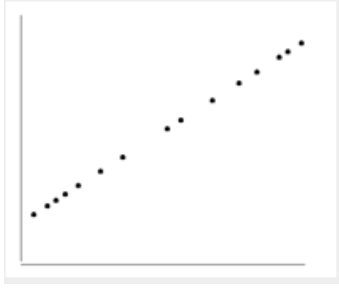
where r_{ij} is the rating given by user i to film j . It is also possible to weight these k-nearest neighbours so that the closer the users' ratings are, the higher they are weighted. This is done using the formula:

$$\sum_{i=1}^k w_i r_{ij} \quad \text{with} \quad \sum_i w_i = 1$$

where w_i is the weight given to user i . While this report will not include these weightings in the method, they were strongly considered before, ultimately, being left out in favour of a simpler model.

The Baseline Effect: The baseline effect, as mentioned before, is to do with the fact that different users have different rating habits, for example some users rate movies they like as a 5 while others still only rate them as a 3. This is something that KNN must be trained to account for because, if not, these habits may affect the neighbours that are used for predictions. This concept is best explained through an example. Take two users, A and B. User A and user B have very different tastes in movies but user B rates everything lower than normal, giving a low baseline, and user A rates everything higher than normal, giving a high baseline. The films that user B likes could, therefore, end up with similar ratings to the films that user A does not like, implying that they are neighbours even though they have very different tastes.

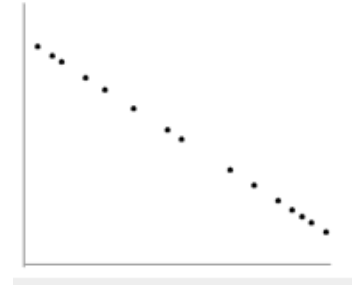
In this situation, the distance may not be a good way of defining the neighbours. An alternative approach taken, in order to try and minimise the effect of the baseline, is to use the Pearson correlation coefficient as a measure of 'nearness' in terms of similarity rather than ratings.



(a) Plot of two users with
 $sim(i, j) = 1$



(b) Plot of two users with
 $sim(i, j) = 0$



(c) Plot of two users with
 $sim(i, j) = -1$

Figure 7: Plots displaying the extreme cases for the Pearsons similarity measure [14].

The Pearson correlation coefficient, shown in equation 2.14 is a measure of the strength of association between two variables [13].

$$sim(i, j) = \frac{\sum_{x \in I_{i,j}} (r_{i,x} - \bar{r}_i)(r_{j,x} - \bar{r}_j)}{\sqrt{\sum_{x \in I_{i,j}} (r_{i,x} - \bar{r}_i)^2} \sqrt{\sum_{x \in I_{i,j}} (r_{j,x} - \bar{r}_j)^2}} \quad (2.14)$$

The formula is used to calculate how similarly user i and user j rate films. In the formula $r_{i,x}$ represents the rating of movie x by user i , $r_{j,x}$ represents the rating of movie x by user j , the \bar{r}_i and \bar{r}_j represent the mean ratings of users i and j respectively and $I_{i,j}$ represents the set of all films rated by both user i and user j . What is happening here is that the users ratings are being normalised to remove the baseline effect.

Figure 7 represents the three extreme cases that can occur with the Pearson correlation. Sub figure 7a shows a plot of two users whose $sim(i, j) = 1$ meaning that their scores are identical. Sub figure 7b shows a plot of two users whose $sim(i, j) = 0$ meaning that their scores have no correlation with one another and finally sub figure 7c shows a plot of two users whose $sim(i, j) = -1$ implying they rate films oppositely to one another, so if user I rates a film a 5 then user j would likely rate it a 0.5.

In the context of film ratings it is only necessary to focus on the $sim(i, j)$ which are close to 1 and it is only necessary to consider correlations between 0 and 1 as all those which are below 0 have the same theoretical effect as no correlation when it comes to finding similar neighbours. The closer to 1 the coefficient is the more similar the two users ratings are.

To begin with a choice of $k=30$ was arbitrarily chosen and the 30 nearest neighbours

based on the Pearson correlation coefficient were found based on the mean scores given by users rather than weighting scores. It would be possible to including weighting here and that may be something to consider in future reports, however, it was not done in this case as it was felt that the mean scores would be adequate.

All of the prediction outputs were then put into a CSV file similar to the one shown in figure 8

	A	B	C	D
1	M1	M2	M3	M4
2	2.54404851	2.54325619	2.42440816	2.46440816
3	3.59460739	3.47020204	3.55457471	3.456289
4	3.62046919	3.56591304	3.72642177	3.6
5	4.42191471	4.27463898	4.39248104	4.26072594
6	4.01227993	3.88539309	NA	3.87148005
7	3.25095238	3.27888889	3.21936937	3.21666667
8	NA	3.40677966	3.40677966	3.40677966
9	3.92651005	3.83462909	3.97713782	3.88271605

Figure 8: Excerpt from file showing predicted values

From figure 8 it can be seen that there are still some 'NA' values in the data, this could be due to a number of reasons. The first is that the film in the test set may not have appeared in the training set and therefore no user would have predicted it so it is impossible to find any users with similar ratings for predictions. The second reason could be that, while the film was in the training set, none of the neighbours of a specific user had rated the film, so again, predicting a rating is impossible.

In these situations 3.5 was used as predictions for reasons mentioned in section 1.4.2. After these 'NA' values were dealt with the data obtained an MSE_{test} score of 1.2077617 using cross validation. While this is not as good as the linear models, it is promising as there are a lot of areas to improve.

Over and Under-fitting Before going any further with developing the model, it is necessary to consider the problem of over or underfitting. Overfitting occurs when too much emphasis is put on the training set [15] and the model is fit too closely to predict

these known values rather than the underlying relationships in the data. This can result in low MSE for the training set, however, it can lead to much higher MSE values when using the test set as the model has not been generalised enough. Underfitting is the opposite problem. This can occur when the training set is not focused on enough and too much emphasis is put on making an overly general model. In these situations, none of the MSE values will be particularly good as the model again does not match up with the underlying relationships in the data. When fitting models it is important to find a balance between these two situations.

In the context of KNN, overfitting can occur when the value of k is very small. In these situations, it is equivalent to lots of very small data sets. This matches the training rating predictions very closely to their true values but results in thousands of points being used to predict and the model becomes overly complicated creating overfitting.

Conversely a large value of k is equivalent to using loads of friends recommend which film to watch. This method does have its advantages as the error for the test set can be reduced however this is at the cost of a greatly increased training error which would result in the model not being chosen. Using an overly large value of k would result in users whose tastes are not particularly similar still being used to predict film ratings causing erroneous predictions. Figure 9 below illustrates how altering the value of k can affect how well the model performs.

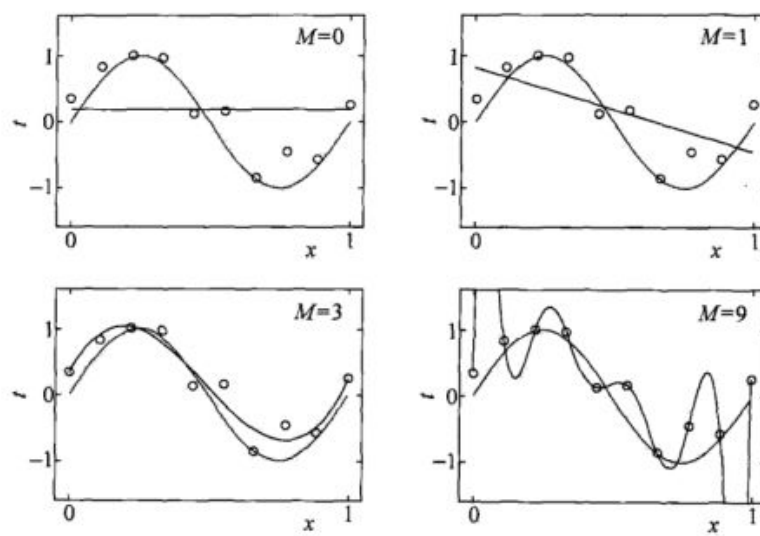


Figure 9: A plot of how well different values of k fit an underlying true distribution

In this report the method of k-fold cross validation will be used in order to select an optimal k, which generally takes a relatively small value. K-fold Cross validation revolves around separating the training set into k ‘folds’ or groups and systematically removing each of the k groups from the data, training the model, testing it on the removed fold, replacing the fold and removing a different one to repeat the process. This allows for the predictive power of each model to be tested numerous times so it can be seen which model performs well for data sets with varying properties. In this project 10-fold cross validation will be performed.

Improving the value of k In the KNN model given previously in this section a value of k was arbitrarily chosen to be 30, however, as has just been stated, the value of k is incredibly important to how the model performs. The process of choosing an optimum value of k will now be investigated.

The idea of choosing a value of k can be thought of, again, in the context of asking friends for recommendations. Asking one friend for a recommendation is equivalent to setting $k=1$. Conversely suppose you talked to 100 friends and asked them for recommendations, this would be the equivalent of using $k=100$ however, due to asking so many people there may be conflicting recommendations and it is therefore tough to decide who to trust. It would be expected that this algorithm would be more effective because it reduces the influence of oddballs in the dataset however it is necessary to find an optimum value of k to get the best recommendations and this is what will be focused on now.

Using 10-fold cross validation, it is possible to plot the MSE of each model for k ranging from 30 to 350. The aim is to find the value of k which minimises this MSE. From figure 10, the value of k which minimises the MSE can be seen to be in the region of 175 to 190 with the actual value being 185. When k is chosen to be any larger than this the MSE begins to increase again implying that the model is being under-fitted. This is again due to users with too dissimilar tastes being used to predict an individual’s rating because of the overly large value of k. When $k=185$ is used in the model the corresponding $MSE= 0.8265968$ which is a huge improvement on the MSE for $k=30$ which equalled 1.2077617. KNN and linear regression from section 2.1 will now be combined in order to try and further improve the MSE values for the test dataset.

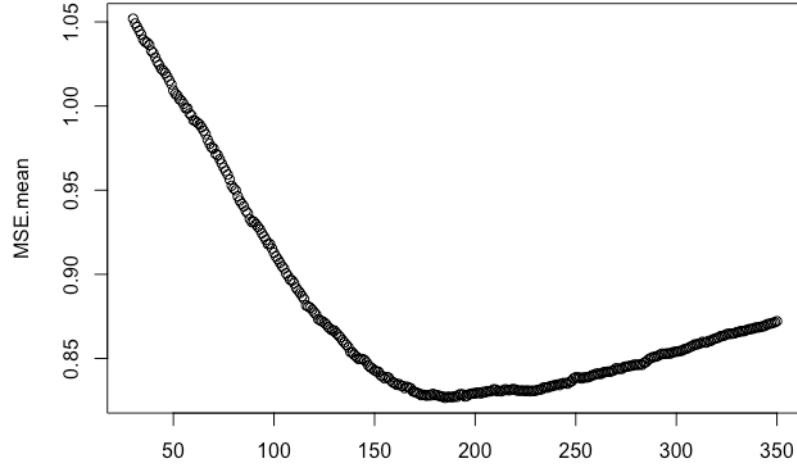


Figure 10: A plot showing the MSE values for k between 30 and 350.

2.3.2 Combining KNN and Linear Regression

The final method that will be briefly touched on is using the linear regression model in section ... to predict values for the ‘NA’ data that arises when using KNN. This is to avoid using a fairly meaningless value like 3.5, which is what the previous KNN model was using.

This method was tried using two of the linear regression models that were previously developed, the first was the model given in section 2.1 with $\theta_i \mathbf{X}_j$ and $x_j = (x_1, \dots, x_{22})$, i.e. the linear regression model with indicative genre and year covariates. This performed better than any previous model obtaining an MSE of 0.8142 based on 10-fold cross validation on the training set. The second model that was tried comes from equation 2.10 in section 1.4.2 and is the second variant of model, named model 2 (b). This model improved the MSE even further achieving a score of 0.81175, again based on 10-fold cross validation on the training set illustrating how combining the methods greatly improved the model performance.

2.3.3 Collaborative Filtering and KNN conclusion

In this section the general idea of collaborative filtering methods were investigated using KNN as a way of finding the neighbours to each user. At first the model performance was

fairly average achieving an MSE of 1.208, however, purely by finding the optimum value of $k=185$, this value decreased to 0.827. There are still some considerations that need to be made regarding this method.

Advantages:

- The method is incredibly simple. This is seen throughout the method as it is easy to understand, implement and interpret. This method does not require any parameter estimation, nor does it have to be trained other than to find an optimum k . Not only this, but its simplicity makes it easily interpretable as the method is very intuitive and fits with how users would naturally look for recommendations from their friends.
- The method is also robust to outliers. Due to the fact that the predictions are based off of similar users, outliers do not appear in the set of nearest neighbours and therefore have a limited impact on the model.

Disadvantages:

- Calculations can be very computationally expensive and slow, especially with such large datasets. This is because the similarity values must be calculated for each individual in order to find which are closest to 1, which results in thousands of calculations that have to be made. This can be solved by searching for the best value of k over different regions in order to narrow down the search that must be undertaken, helping to reduce how long the computation takes.
- The model also provides no insight into which variables are important in predicting the ratings, this is because the model only considers other users ratings and not features of the film such as genre and distribution year which may help to improve predictions.

3 Conclusions

3.1 Results

Table 3 provides the MSE_{test} scores for each of the best models from each of the four broad methods that were investigated. All the values given in table 3 were evaluated

using k-fold cross validation with k=10.

Model	Linear Model with $\theta_i \mathbf{X}_j =$ (x_1, \dots, x_{22})	Simple linear model 2 (b)	Collaborative Filtering, k=185	Combined model (CF with model 2 (b))
MSE_{test}	1.1902	0.8373	0.8266	0.8118

Table 3: Predictive ratings for the best model from each section

All these models were then used in order to predict the ratings for the ‘ratings_test’ dataset. Out of the MSE scores that the models achieved there are two that should be highlighted. The first is the score achieved by the simple linear model which was discussed in section 2.1 . This based its predictions on the genres and distribution year of each film and was incredibly simple and easy to interpret and yet it achieved an MSE score of 1.217 when evaluated on the ‘ratings_test’ set. For such a simple model this is a very good score when considering the complexities of the data, such as the existence of ‘NA’ values. This score is noteworthy as it helped to motivate the rest of the work that took place in this project, and paved the way for other, relatively simple models to be investigated.

The other score that should be highlighted is that achieved by the simple linear model 2 (b), discussed in section 2.2. This was the final model whose test values were submitted, and it achieved an MSE of just 0.82. Again, for such a simple to understand model this score is incredibly low, and it highlights the predictive power of linear models when the appropriately developed.

The model with the best MSE score from the training set, the model that combines KNN with the simple linear model 2 (b) using k=185, was not put forwards due to time constraints however a preliminary version was investigated. This preliminary version used k=30 and the original linear model discussed in section 2.1 and achieved an MSE of 1.039 when using the predicted test set ratings. While this may not be as good as the simple linear model 2 (b) by itself, it is thought that if another prediction were made using k=185 and simple linear model 2 (b) to remove the ‘NA’ values then the predictions would have an MSE even smaller than 0.82.

3.2 Report Limitations

It is necessary in a report such as this one to briefly discuss the limitations of its results. Firstly, the time stamp variable that is displayed in table ... could have been employed as a new covariate. A users' taste in films naturally changes [6] as they grow older however none of the models that have been presented in this report take this into consideration. Using the timestamp variable could penalise ratings made long ago so that they have less of an impact on the predicted ratings, possibly leading to more accurate predictions.

The second major limitation of this report is to do with the collaborative filtering model. In this model Pearson's correlation coefficient was used in order to find the most similar users' to an individual and then use these neighbours for predictive purposes. There are numerous other similarity measures that can be used in place of Pearson's method including the Cosine similarity method [7] and the JacUOD [8] index. If this report were to be repeated it could be interesting to perform the KNN method using all these similarity measures in order to find which one gives the most accurate predictions. It could also be useful to use the weighting method, briefly mentioned in section ..., to help prioritise those neighbours who are more like the user in question.

A final consideration of this report which can lead on to further work is the inclusion of a more complex model. When reading around the topic of recommender systems it was found that there are many, much more complex methods that can be used to perform the task of predicting ratings. Some of these include singular value decomposition and Kumar's [9] regression-based latent factor model. Hence with more time, further investigations could be undertaken in building towards these models in a hope of further improving on the models presented in this report.

Part II

Diagnosing Heart Conditions from ECGs

The second project that will be covered in this report concerns itself with the group of machine learning methods known as classifiers. In classification problems, models are created with the aim of assigning a single value, or class, to the response variable, based on the relationships between the available data. This section of the report will seek to use these methods in the context of diagnosing patients based on electrocardiogram (ECG) readings.

4 Introduction

An ECG is a medical test that is used to diagnose heart conditions, they are widely regarded as 'one of the most common, enduring and important tests in all of medicine' [20] and are performed frequently in order to help doctors make informed decisions about what conditions a patient may be suffering from. The standard ECG consists of 12 leads attached to a patient by a series of 10 sensors: one on each ankle, one on each wrist and six strategically placed around the chest. These sensors pick up on electrical signals produced by the heart when it beats, these signals are then recorded and plotted on a graph ready for doctors to interpret. While the 12 lead ECG is standard across the medical profession, this report will use data from a single lead ECG, while this creates a much simpler data set it can cause issues, something that will be touched on in section 7.2

The main aim of this report was to create a model, based on classification methods, which can take a vector of covariates, \mathbf{x} , from a given patient's ECG and return a single value, \hat{y} , which corresponds to a medical diagnosis. The three groups that our model will need to classify are:

- $\hat{y} = 0$: Denoting a healthy individual

- $\hat{y} = 1$: Denoting an individual suffering from Myocardial Infarction
- $\hat{y} = 2$: Denoting an individual suffering from Cardiomyopathy.

The model will be trained on 115 individuals with known diagnoses and then tested on 100 individuals with unknown diagnoses in order to evaluate the true performance of the model. This performance will be measured using the metric:

$$\sum_{i=1}^{n_{test}} \mathbb{I}(\hat{y}_i = y_i) \quad (4.1)$$

In order to achieve the main aim of creating a model to perform this classification, a series of sub-aims were made which will dictate how the rest of the report will progress. Firstly, the motivations behind the project and the research into cardiologist’s ECG methods will be discussed, these will help in understanding the wider context and the methods used currently to classify heart conditions. Next the data will be presented along with any problems it causes and useful exploratory plots. This will be followed by an in depth review of previous research in this topic helping to guide this report in the model selection amongst other areas. Then the methods themselves will be presented including any feature extraction that was performed on the data set, this section will include the MCE_{train} values for the models as these were used to justify certain model improvements. Finally, the results for our proposed models calculated using equation 4.1 will be given and compared with appropriate conclusions being drawn.

5 Motivation

5.1 Project Motivation

The importance of being able to read ECGs quickly and accurately cannot be overstated and many question the necessity to create a computer programme to do this instead of doctors, however, there are many reasons as to why this may be useful. The first and most obvious is that it frees up a doctors time to perform tasks that computers are not able to do such as surgeries. It is no secret that doctor’s ‘workloads have increased substantially in recent years’ [29] with some reports stating that ‘consultations grew by more than 15%

between 2010/11 and 2014/15' [29] showing the necessity to free up as much time as possible. Roughly 7,000,000 people in the UK suffer from some form of heart condition [30], if each of these individuals requires one ECG per year, with each ECG taking a doctor 2 minutes to read then this amounts to roughly 230,000 hours analysing ECGs nationwide, again illustrating the need for a computerised system to perform this task instead.

Another valid reason to use a programme rather than a professional task is that 'reading an ECG is a matter of pattern recognition' [20]. This is a task that computers are more than capable of performing which, in turn, can help reduce inaccurate diagnoses, as will be discussed in section 7. Computers are constantly capable being improved, thus allowing them to get arbitrarily close to 100% classification accuracy as computational methods improve, people however will always be subject to human error. Not only this but it may even allow for the classification of conditions that ECGs were previously thought to be unable to classify. This is due to the fact that computers can make connections between the underlying data rather than relying on interpreting its graphical representation. A final justification for the importance of this topic is that at the very least the model will provide a second opinion for the doctor, therefore picking up on things that they may have missed or making them more confident in their diagnoses. Either way, computational methods would be contributing to increasing the accuracy of diagnoses and saving lives which, as heart disease accounts for 'more than a quarter (26%) of all deaths in the UK' [30], is clearly justification enough.

5.2 Theoretical motivation: Cardiologist's methods

The ECG was first used in 1902 [33] and has been a crucial part of a doctor's tool box ever since. For most of this history doctors have been reading ECGs themselves and with great success, some reports put the accuracy of trained cardiologists as high as 94% [34]. It is logical, therefore, to begin by understanding how specialists read an ECG so that this information can be used to motivate the model choices made in this report.

To do this it is first necessary to examine a single heart beat from a standard ECG to introduce all the terms and definitions that will be required throughout this report.

Figure 11 shows a diagram of a single heartbeat with 6 specific points labelled: P, Q, R, S, T and U.

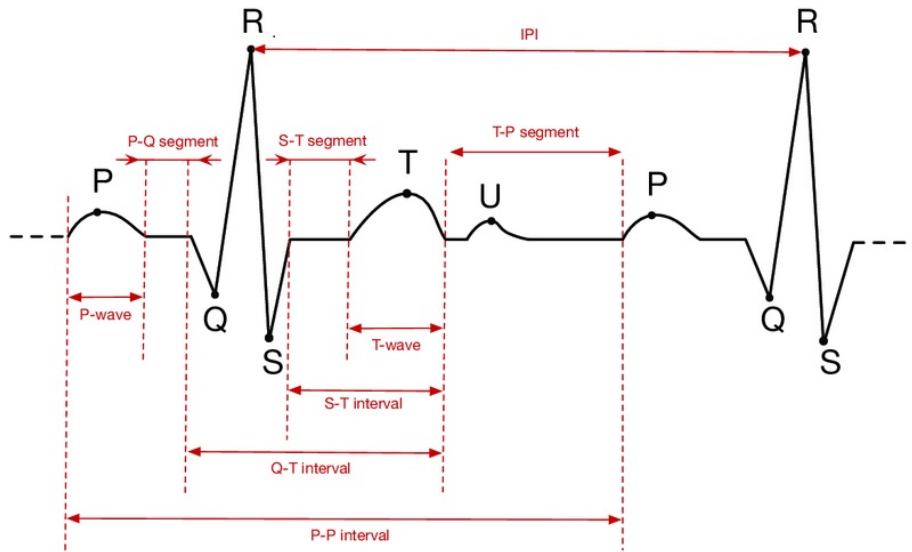


Figure 11: ECG with labelled peaks and feature [23]

The 6 points can be broken up into three main sections:

- The P-wave: this is a peak before the main activity of the ECG which shows the atrial 'activation'.
- The QRS complex: This is the section of the ECG people traditionally think of when talking about a heart beat monitor. It is always referred to as the QRS complex however it may not always contain all 3 waves.
- The T and U waves: The T wave is the first wave after the QRS complex. It is not always a maximum and can become inverted. The U wave is a supplementary wave that can sometimes be seen on ECGs, however this is rarely used in diagnoses as the 'genesis of the U-wave remains elusive' [23].

These points make up the basic anatomy of an ECG heartbeat and, in addition to this, they can be used to calculate a series of values which can be very useful when it comes to diagnosing heart conditions. Most of these are displayed in figure 11 however figure 12 has been included for completeness.

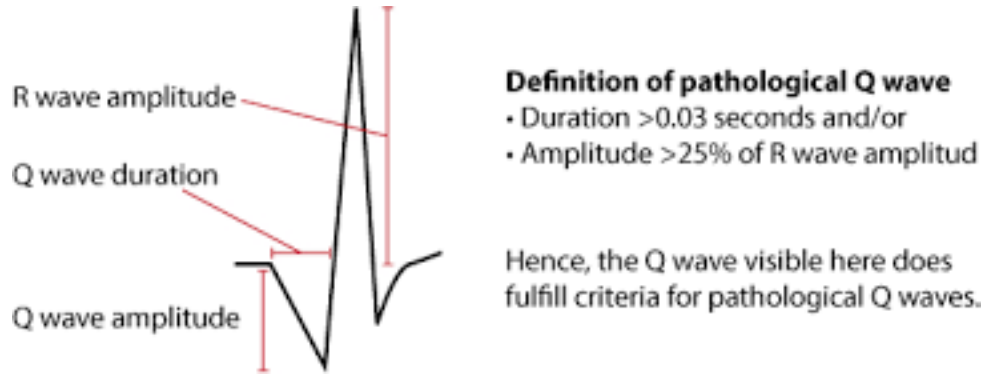


Figure 12: Graph displaying Q amplitude and interval [23]

The key features that can now be observed from an ECG are:

- The PQ-segment: This is a relatively flat portion of the ECG which is often referred to as the baseline and is used as a reference point to calculate the heights of the rest of the features.
- RR-interval (labelled IPI in figure 11): This is the distance between consecutive R peaks and is seen as the beginning and end of one full heartbeat.
- Q-wave amplitude: This is the depth of the Q wave below the baseline
- Q-wave interval: This is the amount of time that the Q wave occurs over. It is measured from the onset of the QRS complex until the point when the ECG passes above the baseline again as seen in figure 12.
- QT-interval: This is from the onset of the QRS complex until the end of the T-wave. This can be adjusted based on the heart rate to give the QTc interval using the formula:

$$QTc = \frac{QTinterval}{\sqrt{RRinterval}}$$

- T-wave amplitude: This is measured similarly to the Q-wave amplitude, i.e. from the baseline to the maximum (or minimum) of the T-wave.
- ST-segment: This refers to the plateau after the completion of the QRS complex. This segment can deviate from being in line with the baseline causing ST-segment elevation or depression.

There are numerous other variables that can be extracted from ECGs, however the above variables are the only ones relevant to this report.

As was previously stated this report is concerned with classifying ECG waves into one of three classes: patients that are healthy, those that are suffering from myocardial infarction and those that are suffering from cardiomyopathy. The ECGs of people suffering from these two diseases have several tell-tale signs, based around the aforementioned features, that will be discussed now.

5.2.1 Myocardial Infaction:

Myocardial infarction (MI) is the official name for a heart attack. It occurs when blood flow decreases or stops to a part or parts of the heart. The most common cause of MI is coronary heart disease which is where blood vessels become blocked by a build-up of cholesterol. Complications caused by MI can be picked up on ECGs long after the heart attack has taken place due to ongoing symptoms such as arrhythmias. The telling features that cardiologists look out for in an ECG in order to diagnose MI are the following:

- ST-segment elevation of 2mm or more above the baseline.
- Pathological Q-waves with a duration greater than 0.03 seconds and an amplitude greater than 25% of the R-wave amplitude.
- Inverted T-waves meaning that the T-wave occurs below the baseline. While this is not a certain sign of MI, it is often a sign that a patient has suffered from MI in the past.

5.2.2 Cardiomyopathy:

The second condition that this report will aim to diagnose is cardiomyopathy. It is thought that roughly 1 in 500 people in the UK suffer from the disease [35] but cardiomyopathy 'isn't a single condition, but a group of condtions that affect the structure of the heart and reduce its ability to pump blood around the body' [35]. There are many causes of cardiompyopathy ranging from genetics to medication side effects, however, due to

the combinatorial nature of the condition there are few unique ECG features to help cardiologists diagnose it. Some of the features that have proved useful are:

- Pathological Q-waves mimicking those seen in MI ECGs.
- T-wave inversion
- Prolonged QTc duration, usually more than 0.45 seconds.
- ST-depression

It is clear that the features are very similar to those described for MI. The first two, relating to Q-waves and T-wave inversion, are identical to those described previously and, while QTc is not explicitly linked to MI, a pathological Q-wave would still result in a prolonged QTc to some extent.

The only distinguishing feature between the ECGs of the two heart diseases is, therefore, related to the ST-segment. If this segment is elevated then the patient would be expected to be suffering from MI, while if it's depressed then the patient would be expected to be suffering from cardiomyopathy. When the models are built around this theory it is this difference that would be expected to separate the two conditions.

Finally, it should be noted that just because a patient's ECG may appear to be 'normal' that doesn't mean said patient is healthy as there are numerous diseases which do not show up on an ECG. This report will therefore seek to classify 'healthy' individuals as those whose ECGs do not exhibit characteristics of a patient suffering from MI or cardiomyopathy rather than attempting to 'diagnose' someone as healthy.

6 The Data

Before delving too deeply into the methodology it is key to have a solid understanding of the data provided. For this project the data consists of 115 training individuals and 100 test individuals, each represented by a vector of length 30000 that, when plotted, generates an ECG graph like the one shown in figure 13. Figure 14 is a zoomed in version of the graph shown in figure 13 and clearly displays the P-wave, QRS-complex and T-wave being repeated for each heartbeat.

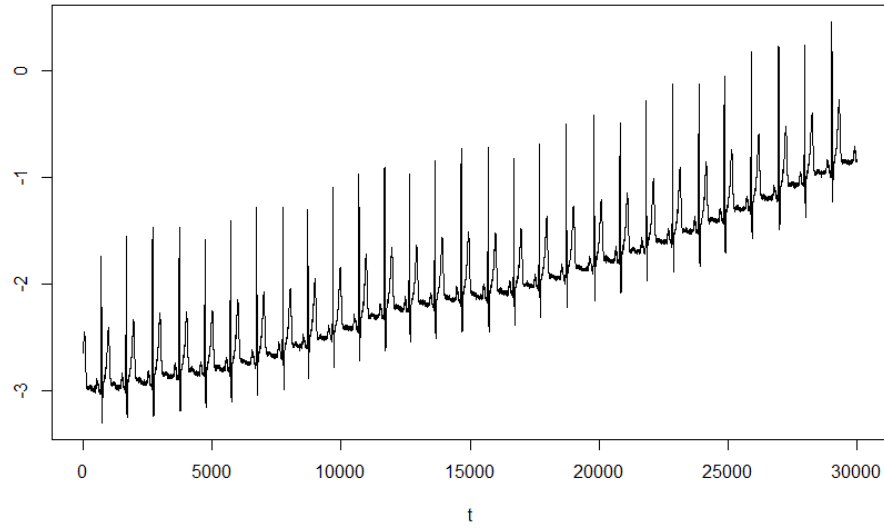


Figure 13: A plot of a standard ECG from the training data

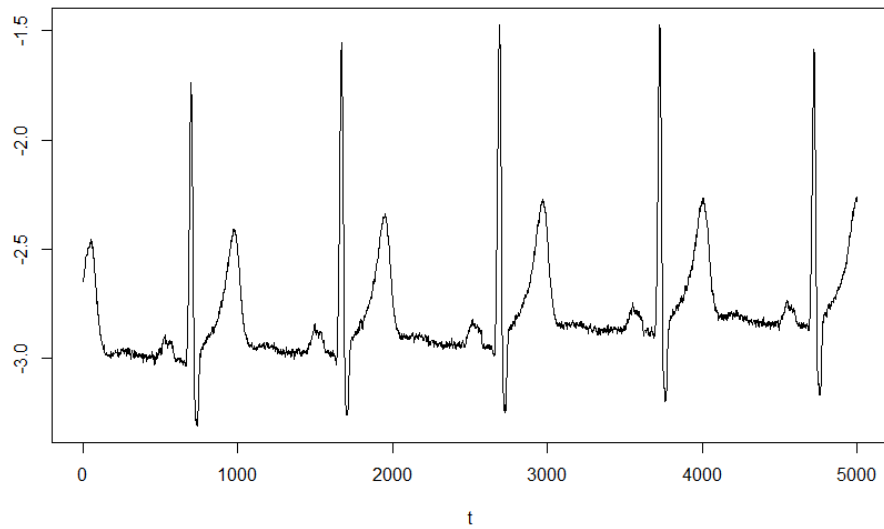


Figure 14: A zoomed in version of figure 13

Each of the 115 training individuals also has an associated class variable. This is either a 0, 1 or 2 as mentioned in section 4. These classes are treated as factors and therefore their associated number is irrelevant and they could equally be labelled 'H', 'M' and 'C'.

It is sensible when looking at a new data set, especially one with 30000 points per

individual, to make some preliminary plots in order to visualise the data set. Figure 15 is a pie chart displaying the proportions of each of the classes. From this graph it can be seen that the vast majority of training individuals suffer from MI. While this does not tell us anything about the test set, as it cannot be assumed that the proportions are the same, it does highlight some of the problems that may be encountered. The most notable of these problems is that only 7.8% of the training set suffer from cardiomyopathy, this equates to just 9 individuals. This creates issues for several reasons, one being that characteristic features of a cardiomyopathic ECG, such as ST-depression, may not be represented in such a small sample making it difficult to train a model to pick up on all feature's indicative of the disease. It also means that if cross validation is used in a method, to reduce over-fitting, then it is highly likely that some of the folds may contain no cardiomyopathic individuals which can again lead to models which are poor at diagnosing diseases.

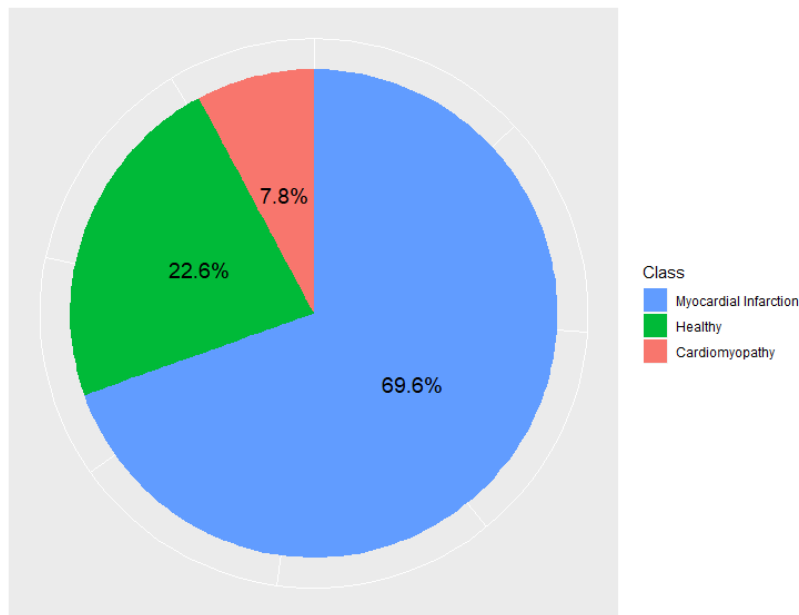
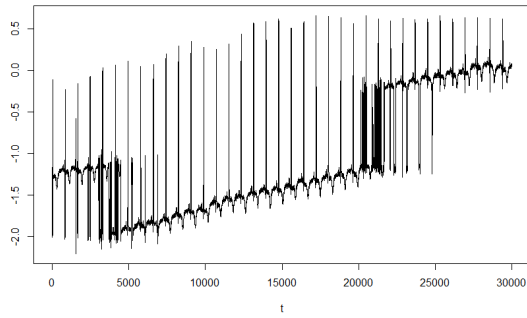
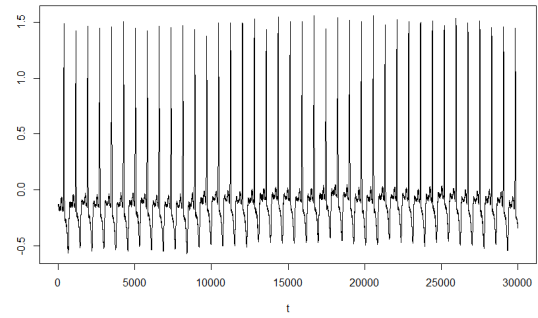


Figure 15: A pie chart displaying the proportions of each class in the training set

A small sample of individuals suffering from cardiomyopathy is not the only issue with the data, another is the training data's size in general. While each individual may have 30000 data points there are only 115 individuals in total to train the models from. This is enough data for most models however it may limit this reports ability to utilise some more complex models, such as neural networks, as they will perform poorly with such a



(a) ECG for someone who suffers from myocardial infarction



(b) ECG for someone who suffers from cardiomyopathy

Figure 16: Graphs showing the differences between MI and cardiomyopathic ECGs

small training sample.

A final issue is that one of the intentions of this report is to create a function to perform feature extraction on the ECGs, this means creating an algorithm that works well for all of the variations that an ECG can take. This could prove difficult because the algorithm must extract features from vastly different ECG plots. ECGs for individuals suffering from MI or cardiomyopathy can look incredibly different, as shown in figure 16 below, this can make writing a single algorithm to perform all of the feature extraction problematic. There are ways to get around this issue such as transforming the data or brute forcing the feature extraction. This, however, can result in loss of information, can be incredibly time consuming and could end up with incorrect data values due to human error. In the following section regarding previous research, this report will investigate how other papers have overcome these issues so that their methods can be considered going forwards.

7 Previous research

The aim of automating ECG readings and creating a model to diagnose patients is not a new topic, it has been the subject of numerous reports with many different approaches taken. This section will give a brief overview to a series of papers, their approaches and

their results, therefore, allowing for informed decisions to be made regarding this papers methodology. There are three main areas to look at when comparing the past research for this project; these are the feature extraction methods, the models chosen and their overall performance. First the methods for feature extraction will be discussed.

7.1 Feature extraction

As mentioned in section 5.2 feature extraction is a crucial part of creating a model that predicts heart conditions based on ECGs. Previous papers differ greatly on the methods they use to perform feature extraction, some papers such as that by Celin and Vasanth [24] start by rigorously transforming the data into a more standardised form by removing high frequency noise caused by muscle contractions from muscles other than the heart. They do this using a series of filters, such as the Butterworth filter, to remove high and low frequency noise. They then use an unspecified peak detection algorithm, presumably of their own making, to find the peaks of the QRS-complexes allowing them to separate the ECG into individual heart beats before finally extracting the relevant features. These features include pulse transit time and pulse rate variability which are calculated using information that this report doesn't have, such as blood vessel radius. While the features being extracted are irrelevant the method used seems to be fairly easy to implement and will be strongly considered by this report.

Other papers such as that by Heden et al. [25] do not specify the method used to extract the features but instead appear to have gotten the information from a more advanced ECG machine: 'the ST-T measurements used as input to the artificial neural networks were obtained from the measurement programme of the computerised ECG recorders' [25]. A final method frequently discussed amongst previous papers, including that by Acharya et al. [31] is the use of neural networks on the raw data, resulting in minimum data pre-processing and relying on the neural networks to learn how to spot the relevant features before classifying them.

This report aims to use a similar method to the one used by Celin and Valanth [24] to extract the desired features. The method of finding individual heart beats and then extracting the specific features appears achievable based on the data provided when

compared to the other two methods that have been discussed. Unlike Celin and Valanth, however, this report will then extract the features talked about in subsection 5.2 from these individual heart beats ready to be used for classification.

7.2 Methodologies

Previous paper's models and their relative performance will now be dealt with in tandem. Researching the topics showed that reports use a wide variety of machine learning techniques of varying complexity to classify ECGs. Celin and Valanth [24] again proved useful as they compared four methods, namely: Support vector machines (SVMs), SVMs with adaboost, artificial neural networks (ANN) and the Naive-Bayes classifier (NB). The results of their report provide a lot of insight into the nature of classification methods. As is often the case the most basic method, NB, proved the most accurate providing an accuracy of 99.7% using cross validation to avoid over-fitting. SVM was the worst performing model, recording an accuracy of 87.5% while the adaboost and ANN methods achieved accuracies of 93% and 94% respectively. These results could be attributed to several reasons, one of the most likely being that the parameter selection for the SVMs and ANNs could be improved upon as there is such a wide variety of parameters that can be passed to these two methods. Celin and Valanth do, however, illustrate an important point that no method, regardless of how simple it may be, should be ignored.

Strodthoff and Strodthoff [17] also chose to use neural networks, resulting in a sensitivity of 93.3% and a specificity of 89.7% when using 10-fold cross validation. Sensitivity and specificity are performance measures specific to binary classification problems and therefore won't be needed in this report, however these are very respectable scores and again display how well ANNs can work on problems such as this one. Strodthoff and Strodthoff's paper goes on to state that 'the proposed method outperformed state-of-the-art approaches and reaches the level of human cardiologists for detection of myocardial infarction' [17], confirming what was discussed in section 5.1 that computational methods can be as good, if not better than, trained cardiologists.

A final report of noteworthy significance is by Rahman et al. [18]. In their report, which focuses on the diagnosis of cardiomyopathy from ECGs, they compare several meth-

ods including SVMs and random forests using both 5 and 10 fold cross validation. This is significant because random forests are based off of decision trees which make classifications based on binary decisions. An example in the context of ECGs may be whether the Q-interval is over or under 0.03 seconds in length. This method is therefore very intuitive as it makes decisions in ways similar to those discussed in section 5.2. The results show that the random forests and SVMs performed similarly, with both recording a precision of 84%. This may be lower than other methods discussed however the intuitive nature of the approach could make it a worthwhile model to investigate.

This report will seek to compare a variety of methods. Taking note from the lessons learned from Celin and Valanth, this report will start with more basic methods, such as the KNN method, before building upon these in an attempt to improve on their results rather than rushing to more complex 'black box' models that are difficult to understand. As for the models that this report will try to build towards, neural networks appear to be used universally however they may not perform well in this situation due to the reasons mentioned in section 6. SVMs perform well throughout all the previous papers and may be taken into consideration as this report moves forwards. Random forests, however, seem to be the most useful method in the ECG context due to their binary decisions based on whether or not an ECG does or does not exhibit specific behaviours. It is also clear from the past research that cross validation is key, especially when working with such a small training data set.

The following section will now cover the methodology of this report. It will start by describing the methods used to extract the features from the raw data before utilising them in a series of classification models, justifying the reasoning between each model's inclusion as it goes.

8 Method

The methodology will be dealt with in four parts, first the methods used to transform the data into a more usable set will be presented, allowing for the average heartbeats to be found. Then this data will be analysed using KNN and discriminant analysis. It will be observed that these models perform poorly on the data set so feature extraction will

then be performed to find the necessary features, discussed in section 5.2. Finally decision trees and random forests will be used to analyse these new data points.

8.1 Heart Beat separation

In order to perform analysis on this data it is first necessary to extract the relevant features. One obvious direction to start in is to somehow extract individual heartbeats from the time series. To do this an algorithm is developed that works as follows.

Owing to the high frequency content of the QRS region the derivatives of these regions have higher amplitudes and thus can be used to locate the QRS complex [16]. Here the first difference of the time series is taken, i.e $d1(i) = e(i+1) - e(i), i = 1, \dots, n-1$, which is proportional to the first derivative. Then in order to maximise the amplitudes second differences are taken, $d2(j) = d1(j+1) - d1(j), j = 1, \dots, n-2$, and then squared, i.e. $D(j) = [d2(j)]^2$. Here $e(n)$ corresponds to the n th point in the ECG signal.

The peaks of this differenced series are found by checking the magnitude of each point against it's neighbours within a certain region. The region here was an extrapolation of the average heartbeat length of the observation as logically heartbeats of an individual will be a certain duration as maximums will only appear in these areas. Figure 17 shows the peaks located on the differenced plot for the 1st individual.

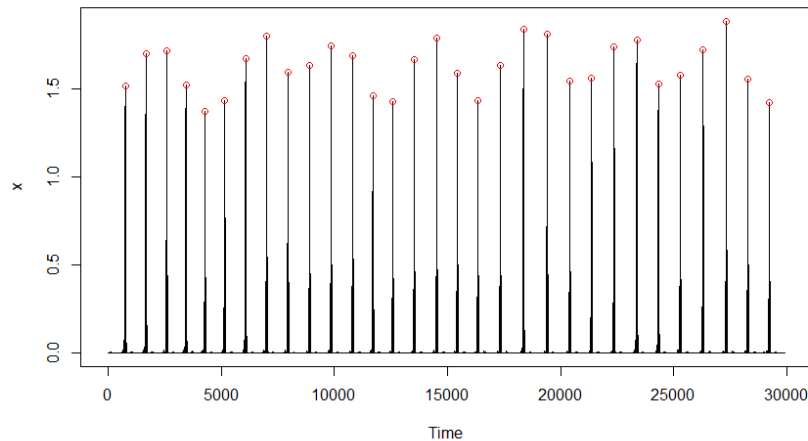


Figure 17: Graph showing peaks located on the second differenced series of the first individual.

The next step is to find the corresponding indices on the original ECG time series and find the maximum point in this QRS region via magnitude comparison once again. These points will then be the R peaks of each heartbeat within the ECG time series as can be seen in figure 18. Through this method 100% accuracy is obtained in locating the peaks of the R wave for every individual in both the training set and test set. It should be noted here that because of the differencing method used some information is lost as the n th point $e(n)$ of the ECG data will be discarded.

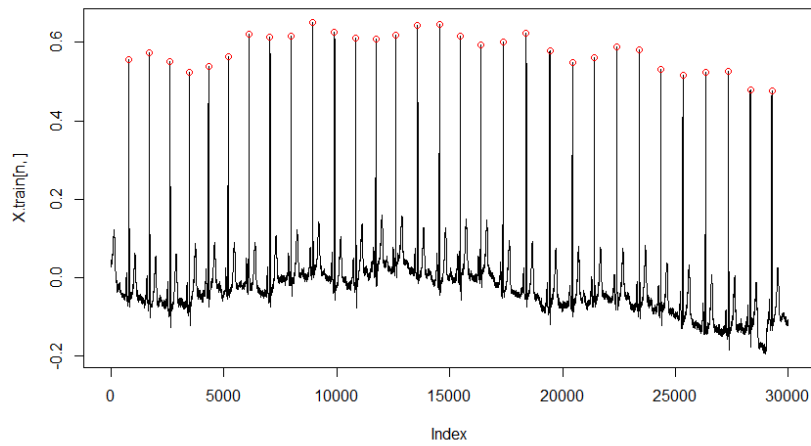


Figure 18: Graph showing peaks located on original ECG data of the first individual.

Next the ECG wave is ‘chopped up’ into individual heartbeats by extracting the information between R peaks and representing this as a vector, the length we chose for this vector was $p=1000$ points in order to lose the least amount of information contained in the heartbeat as possible. This leads us to gain a number of different heartbeats for each individual as can be seen in figure 19a and we then decided that the best way to represent this individual was to then take an average of these heartbeats as seen in figure 19b. Each individual’s average heartbeat is then demeaned by subtracting the average of the heartbeat in order to standardise the heartbeats and remove any baseline drift.

At the end of this process an n -by- p matrix of heartbeats is obtained for every individual in the training set and also the test set with which the analysis can be performed in the coming sections.

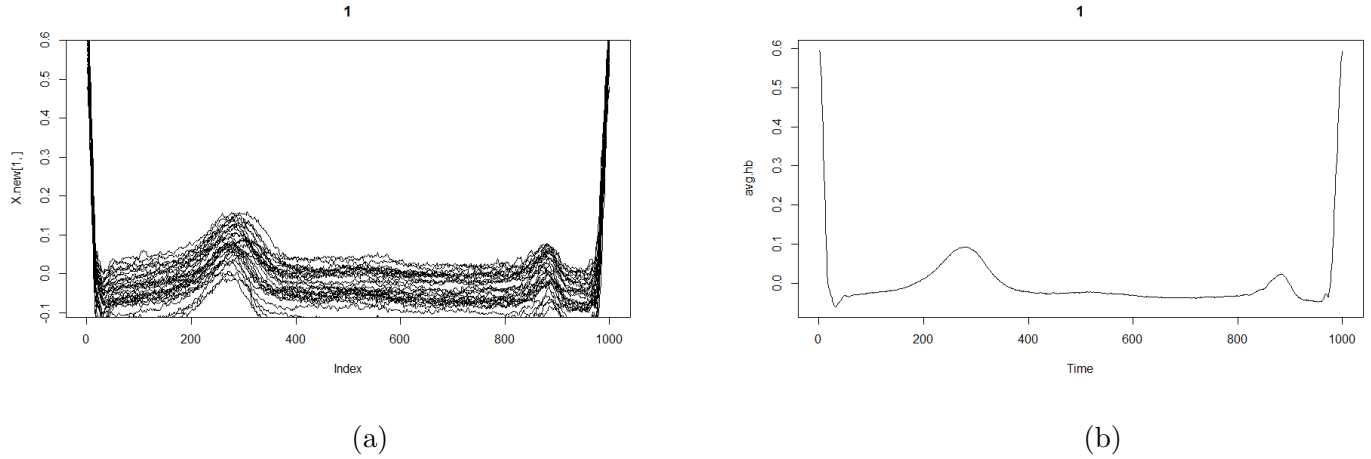


Figure 19: (a) Shows every heartbeat extracted from the ECG signal of the first individual, (b) Shows the average heartbeat taken to represent the individual .

8.2 Analytical method on mean heart beats

As was previously mentioned, the first step taken when extracting the features from the data sets was finding the average heart beat of each individual. Through inspecting these average heart beats it became clear that they varied greatly depending on the health status of the individual that they referred to, it was therefore decided that analysis would be undertaken on these average heart beat vectors in order to examine their predictive properties. The two methods that were used to perform this task were k-nearest neighbours (KNN) and discriminant analysis. Both of these methods were chosen due to their simple approaches allowing for quick examination of the predictive properties, if either of these models performs well then it would be possible to develop them into more complicated models and if not then this report could conclude that the average heart beat provided little predictive power without extracting the necessary features. It will be found that the latter was correct in this situation. KNN will now be discussed.

8.2.1 KNN

The first model that was built using these average heartbeats as features was a k-nearest neighbours model using Euclidean distance as the distance metric. Please refer back to Section 2.3.1 for the theoretical idea of the model. The reasoning for using this model was that different pathologies would be exhibited in similar looking waveforms e.g. it would

be expected that the majority of healthy heartbeats would look similar to each other, myocardial infarction heartbeats would look similar and so on. K-nearest neighbours would be useful here as it would classify waves most similar in form to the same class. It is also a simple model to implement and would give a good "baseline" accuracy with which to compare performance of other potential models to.

Some basic analysis of the optimal k was done and $k=10$ was chosen for the model. In order to test performance of the model a simple validation method, was used by splitting the training data into two different sets - one to train the model on, the other a holdout set to test the model on. This model achieved an MCE of 0.36 on this holdout set but when applied to the test set it achieved an MCE of 0.47 indicating that the model suffers from overfitting, one reason for this could be due to the validation used, using a holdout set is a very simple form of cross-validation and using a more advanced form such a k -fold could have revealed a more optimal model. This was a consideration taken into account when future models were developed as will be seen in the next sections.

8.2.2 Discriminant Analysis

Theory: Discriminant analysis is a standard machine learning tool used universally in classification methods. Before covering the basics of the method some notation needs to be introduced:

- \mathbf{z} : a new observation to be classified.
- g : the number of classes that an observation can be assigned to.
- p : the dimension of the observation.

Here, \mathbf{z} is an ECG from the test set, $g=3$ and $p=1000$ as this is the length that each average heartbeat was set to.

The fundamental idea of discriminant analysis is to assign each \mathbf{z} to one of the possible classes, here labelled as 0, 1 or 2, with as small a probability for error as possible. A graphical representation of the method can be seen in figure 20 with $g=3$ classes in 2 dimensions, the method splits the p -dimensional space into g disjoint regions so that the location of each data point allows it to be classified accordingly.

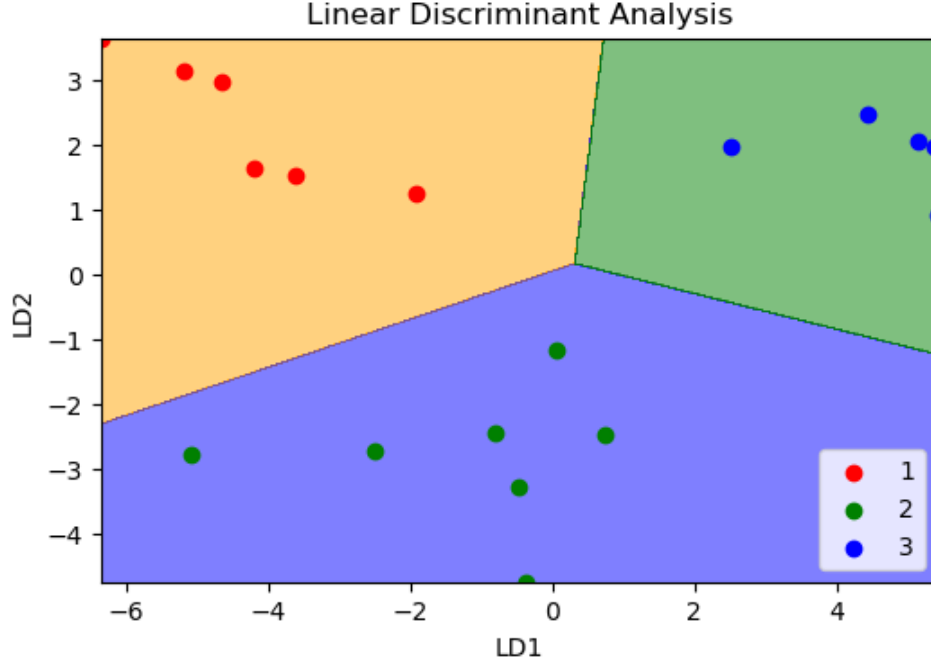


Figure 20: A plot giving an example of LDA [21]

There are numerous types of discriminant analysis that can be performed. The most basic and well known is linear discriminant analysis (LDA), this is where the boundaries between classification regions are linear. While this is very simple to implement it can be poor at predicting more complex relationships due to its rigid structure. As a result other methods such as quadratic discriminant analysis (QDA) and regularised discriminant analysis (RDA) can be used in more complex situations.

In this report the intent was to first apply LDA, before following this up with Fisher discriminant analysis (FDA). FDA is a variant of LDA with less assumptions to be satisfied, making it better suited to the data that this report is investigating. It will also be discussed as to why QDA and RDA were not investigated in this report.

Method: LDA is performed by computing the value of a discriminant rule that is derived from the maximum likelihood. The derivation is not relevant to this report and therefore will not be included but the rule is presented as follows. Each new \mathbf{z} is allocated to the group, j , that minimises the following equation:

$$(\mathbf{z} - \boldsymbol{\mu}_k)^T \boldsymbol{\Sigma}^{-1} (\mathbf{z} - \boldsymbol{\mu}_k) \text{ for } k = 1, \dots, g \quad (8.1)$$

Where $\boldsymbol{\mu}_k$ is a g by 1 vector containing the population means for each group, and $\boldsymbol{\Sigma}$ is the population covariance matrix. Here lies the first issue with LDA. The method requires the population values for the mean vector and covariance matrix, this is easily solved as it can be shown that the sample mean vector and pooled covariance matrix can be used to the same effect, the formula for this is shown below:

$$(\mathbf{z} - \bar{\mathbf{x}}_k)^T \mathbf{S}^{-1} (\mathbf{z} - \bar{\mathbf{x}}_k) \text{ for } k = 1, \dots, g \quad (8.2)$$

Where \mathbf{z} is again allocated to the group j which minimises equation 8.2.

Another issue that occurs is that the data set that LDA is being applied to is far too large. In the calculation of the discriminant rule the inverse of the pooled covariance matrix must be found however the determinant of this matrix is approximately equal to 0 resulting in R returning errors. It is therefore necessary to carry out principal component analysis (PCA) on the data set in order to reduce its dimensions. PCA works by projecting the data into the desired dimension such that it's variance is maximised, thereby losing as little of the explanatory information from the data as possible. What dimension to choose is up for debate therefore cross validation will be performed in order to find which dimension gives the most accurate results.

Another problem that arises with LDA is that it is assumed that the covariance, or sample covariance, matrices are equal. This can be tested through a procedure called Box's M test [19]. Using the biotools package in R it was found that the covariance matrices for this dataset were not equal as the test returned a p-value=1.898e-08, which clearly means the null hypothesis is rejected. As a result, a new approach needed to be found, this is what led to Fishers discriminant analysis. FDA is very similar to LDA in its approach however, instead of assuming the covariance matrices are the same it creates a new criterion to classify the observations.

The Fisher Criterion is to find a unit vector, λ , which maximises:

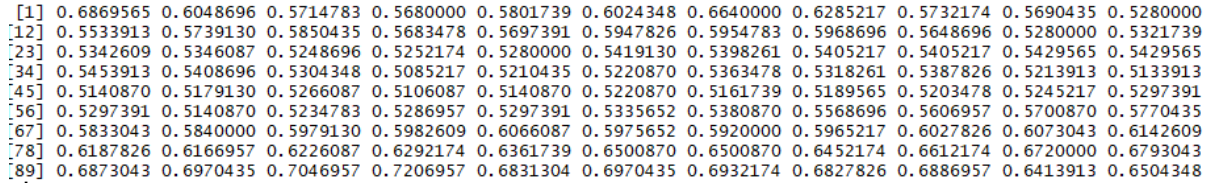
$$\frac{\lambda^T \mathbf{B} \lambda}{\lambda^T \mathbf{W} \lambda} \quad (8.3)$$

where:

$$\mathbf{W} = \sum_{j=1}^g n_j \mathbf{S}_j \text{ and } \mathbf{B} = \sum_{j=1}^g n_j (\bar{\mathbf{x}}_j - \bar{\mathbf{x}})(\bar{\mathbf{x}}_j - \bar{\mathbf{x}})^T \quad (8.4)$$

with $n = \sum_j n_j$ and S_j is the sample covariance matrix for class j . It can be shown that this criterion is satisfied by the largest eigenvalue of $\mathbf{W}^{-1}\mathbf{B}$, \mathbf{z} is then classified to the class that minimises $|\boldsymbol{\lambda}^T \mathbf{z} - \boldsymbol{\lambda}^T \bar{\mathbf{x}}_k|$.

FDA was then used with leave-5-out cross validation 25 times, with average MCEs being taken in order to find which dimension gives the most accurate predicted scores. The MCE was calculated using PCA to reduce the average heartbeats from 2 dimensions up to 100 dimensions with the results presented in figure 21.



[1]	0.6869565	0.6048696	0.5714783	0.5680000	0.5801739	0.6024348	0.6640000	0.6285217	0.5732174	0.5690435	0.5280000
[12]	0.5533913	0.5739130	0.5850435	0.5683478	0.5697391	0.5947826	0.5954783	0.5968696	0.5648696	0.5280000	0.5321739
[23]	0.5342609	0.5346087	0.5248696	0.5252174	0.5280000	0.5419130	0.5398261	0.5405217	0.5405217	0.5429565	0.5429565
[34]	0.5453913	0.5408696	0.5304348	0.5085217	0.5210435	0.5220870	0.5363478	0.5318261	0.5387826	0.5213913	0.5133913
[45]	0.5140870	0.5179130	0.5266087	0.5106087	0.5140870	0.5220870	0.5161739	0.5189565	0.5203478	0.5245217	0.5297391
[56]	0.5297391	0.5140870	0.5234783	0.5286957	0.5297391	0.5335652	0.5380870	0.5568696	0.5606957	0.5700870	0.5770435
[67]	0.5833043	0.5840000	0.5979130	0.5982609	0.6066087	0.5975652	0.5920000	0.5965217	0.6027826	0.6073043	0.6142609
[78]	0.6187826	0.6166957	0.6226087	0.6292174	0.6361739	0.6500870	0.6500870	0.6452174	0.6612174	0.6720000	0.6793043
[89]	0.6873043	0.6970435	0.7046957	0.7206957	0.6831304	0.6970435	0.6932174	0.6827826	0.6886957	0.6413913	0.6504348

Figure 21: A screen shot of MCEs from FDA with 99 different levels of dimension reduction

From this figure we can see that when reducing the dimension of the data down to around 45 the model consistently achieves an MCE of roughly 0.51 which is worse than the KNN model used in section 8.2.1 and is only classifying the variables correctly 50% of the time which clearly isn't good enough.

Using QDA and RDA were briefly looked into however these rely on using each of the class covariance matrices separately, rather than using a weighted sum like FDA does. As a result of the small training dataset, which only includes 9 ECGs classified as type 2, these methods were unable to be implemented with any sort of accuracy. Cross validation was also not possible to use as some folds would include no ECGs of class 2. This is an issue in this model specifically because if there are no observations of a class then it becomes impossible to compute said classes covariance matrix.

As a consequence of these results and an inability to develop the model further using QDA, discriminant analysis was not used any further in this project. The model does, however, provide valuable insight into how necessary it is to extract the relevant features from the ECGs. Once these features have been extracted the data set becomes much more concentrated with only the most relevant information being included. The methods used to perform this feature analysis will be discussed in section 8.3.

8.2.3 Mean Heartbeat Analysis Conclusion

From the two methods of mean heartbeat analysis that have been discussed it is clear that the KNN model with $k=10$ performed far better, achieving an MCE of 0.36 against FDAs 0.51. It is possible that the KNN model could be improved up by performing a rigorous evaluation on how k was selected, however this was not investigated in this report. The KNN model was still put forwards using the test data and will serve as a useful, simple starting model to compare the more complex methods to later on.

Never the less, neither of these models achieved MCE values anywhere near the sort of scores that previous papers have been achieving. This paper will therefore take the feature extraction a step further by attempting to find the specific features that cardiologists use to diagnose individuals. The methods through which this was achieved will be discussed now.

8.3 Feature Extraction

Appendix B is the algorithm coded in Python used to extract certain features which were determined to be crucial in identifying whether a patient is ill or not and further to determine what they may be suffering from.

The algorithm had been used to find the peaks S, T, P, Q and R as well as evaluating the following

- $T_{gradient}$ - the gradient approaching the point T from the left
- $Baseline$ - the value at the flat segment between P and Q
- ST - the y -value at the flat segment between S and T
- Q_{width} - the width of the peak Q
- $Q_{Amplitude}$ - the steepness of the peak Q
- $QT_{interval}$ - the length of the interval between Q and T

Due to the individuality of each averaged ECG, certain boundaries used for evaluation techniques had to be amended per case.

S,T,P,Q & R

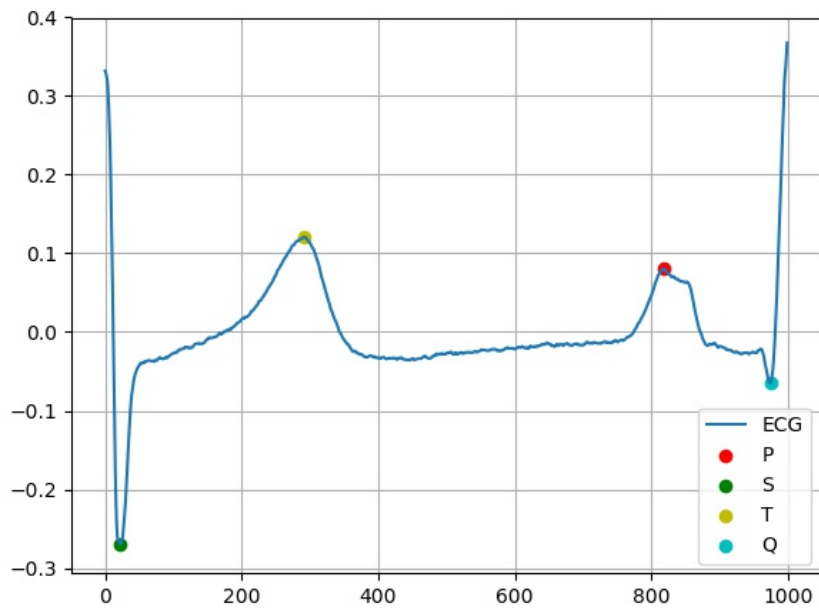


Figure 22: An ECG of a healthy person

Figure 22 is a typical example of a ‘*healthy*’ averaged ECG. In order to attain the relevant *peaks* from the graph, the averaged ECG must be segmented into four parts, namely from 0 - 150, 150 - 550, 550 - 850 and 850 - 1000 as the diagram below shows.

Each individual portion will be used to find S , T , P and Q respectively. The value of R can be found by finding that maximum of the whole averaged ECG (unless R is not the maximum value, in which case the position of R is to be determined). It is important to point out however, that the boundaries for these portions vary for each averaged ECG graph as they may have peaks in slightly different places albeit being in the same general area.

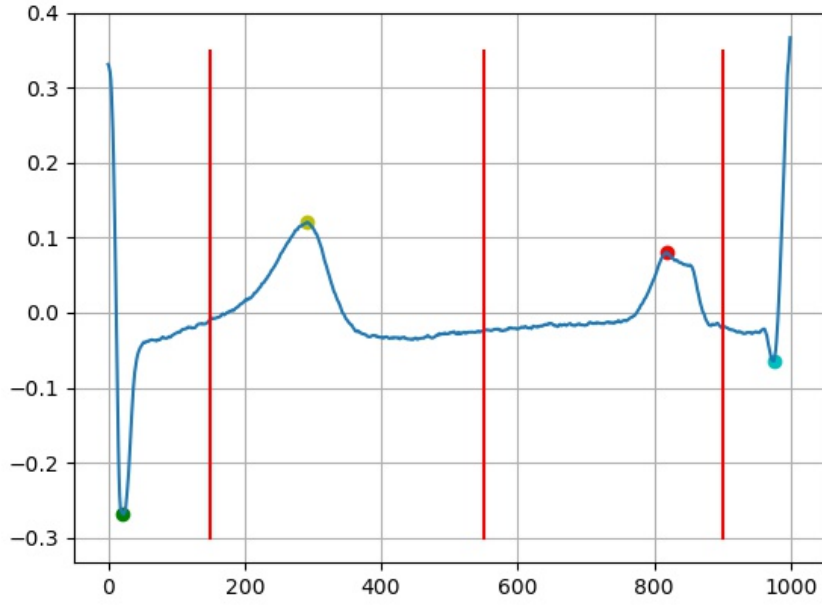


Figure 23: The *segmented* ECG graph

T-Wave

Inverted T-Wave

At this point it will be clear whether or not the T -wave is '*inverted*' or not. In other words, if its a maximum or minimum in the second portion.

Figure 24 shows an example of an inverted T -wave, the first indication that the patient may not be '*healthy*'. To take into account this inversion, a reference value must be introduced to evaluate the relative severity of the peak of T . This will be referred to as *Baseline*.

Baseline is defined as the point between P and Q where the ECG is seemingly flat. Thus to evaluate the value of *Baseline*, the algorithm in Appendix B finds the first point between P and Q where the gradient is below 0.001. One problem with this approach is that there might be some flat segments on the P -wave, prompting the algorithm to evaluate *Baseline* at the incorrect position. To overcome this, instead of starting at point P , the

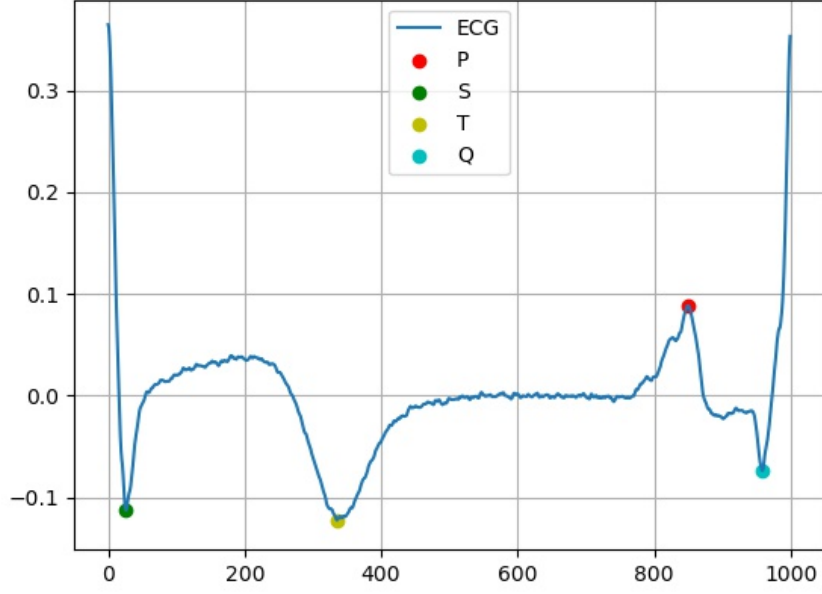


Figure 24: An inverted T -Wave

algorithm starts at the half way point between P and Q .

Now that the Baseline can be identified, the peak T relative to $Baseline$ can be defined as

$$T_{Baseline} = T - Baseline$$

The values for $T_{Baseline}$ for the two ECGs earlier are 0.14678 for the ‘*healthy*’ averaged ECG and 0.05425 for the averaged ECG with the inverted T -wave. A lower value of $T_{Baseline}$ implies a higher likelihood of a patient suffering from Myocardial Infarction.

Note that the algorithm must take into account whether T is a local maximum or minimum. In other words, the averaged ECG must be analysed beforehand to program the algorithm to either find a maximum or minimum. An initial approach to this issue was to calculate the mean value of the ECG local to T and implementing logic that suggests that if this value is below $Baseline$, then T should be the minimum and vice versa. The problem with the approach, however, is that complications arise when $Baseline$ is relatively high or low relative to the whole averaged ECG, which will be investigated in a

later section.

Gradient approaching T

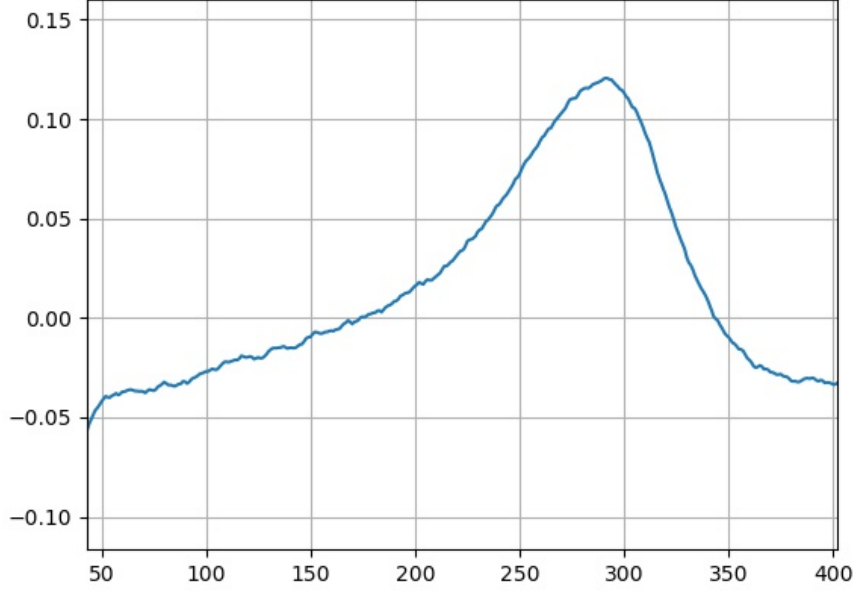


Figure 25: A steep gradient when approaching point T

There is a steep gradient approaching T for the ‘*healthy*’ patient, as can be seen in figure 25. One indication of Myocardial Infarction is a *less steep* approach of T , as the T -point as shown in 26.

A good approximation of evaluating the gradient approaching T would be the gradient from the midpoint of the ST-segment and T to T . Let the variables ST_x and T_x denote the x -value at ST and T respectively and let $ECG(x)$ be the value of the Average ECG at the point x . Then define

$$T_{gradient} = \frac{ECG(\frac{1}{2}(ST_x + T_x)) - T}{\frac{1}{2}ST_x - \frac{1}{2}T_x}$$

The less positive $T_{gradient}$ is, the more likely the patient will suffering from Myocardial Infarction. Note that if $T_{gradient}$ is negative, then this means that the averaged ECG has a negative $T_{gradient}$.

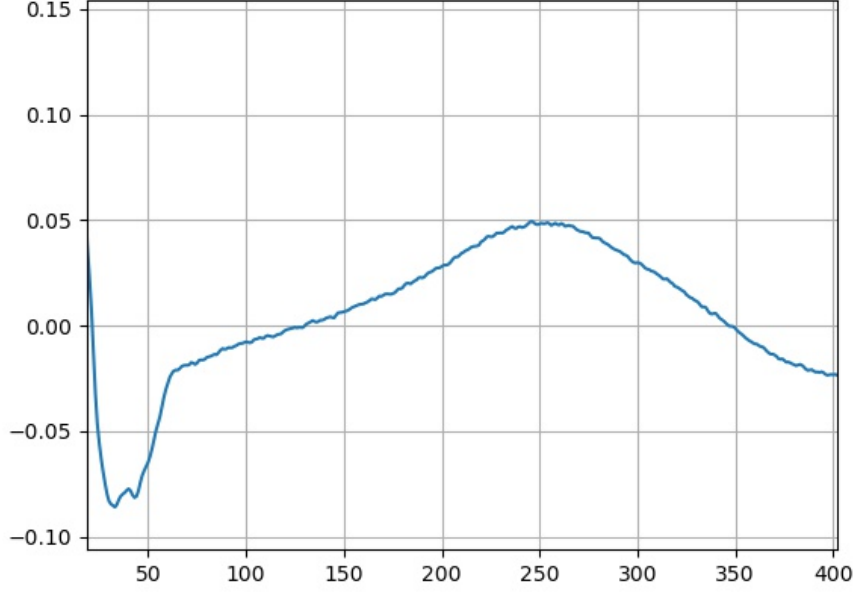


Figure 26: A *flatter* gradient when approaching point T

Baseline vs ST-Segment

As highlighted in section 5.2, the position of the baseline relative to the ST-Segment is crucial in identifying the potential illness, if any. The ST-Segment is the seemingly flat line that follows after S and before T . The algorithm in Appendix B evaluates this point in a similar fashion to *Baseline*, only starting at S this time. The y -value of the ST-segment will be referred to as ST .

For the case of the healthy averaged ECG, *Baseline* and ST are relatively level.

The Baseline and ST-Segment of the averaged ECG in Figure 27 are roughly similar in value, indicating that the patient is healthy. The ST-elevation is defined as

$$ST_{elevation} = ST - Baseline.$$

The sign (and magnitude) of $ST_{elevation}$ is a good indicator discerning what illness the patient is suffering from. A '*healthy*' averaged ECG for a patient should have a value relatively close to 0. The averaged ECG below shows a case where ST is higher than

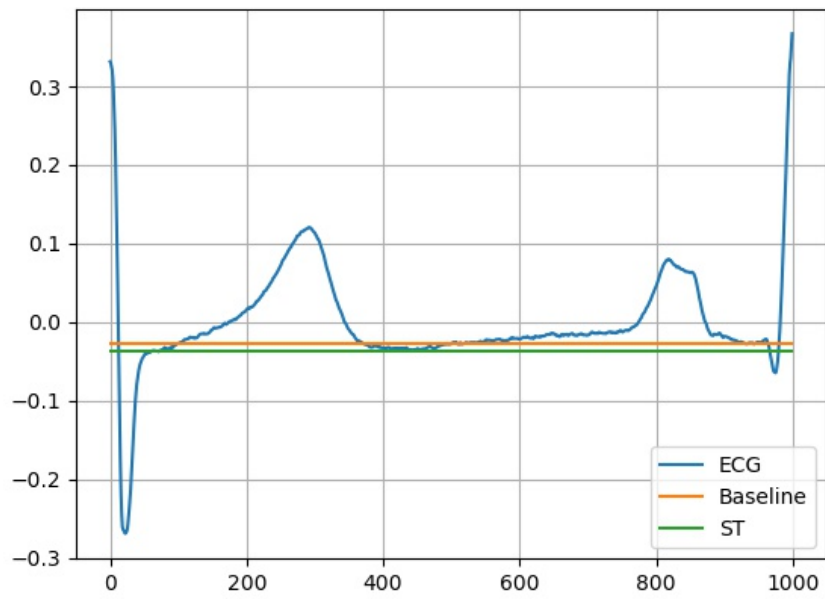


Figure 27: The Baseline and ST-Segment

Baseline.

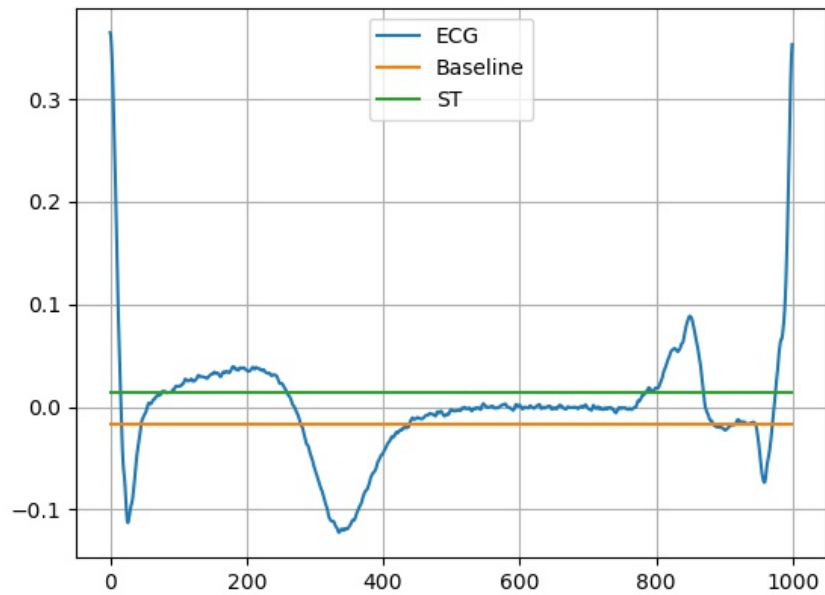


Figure 28: The Baseline in this ECG is considerably lower than ST

For the ECG in Figure 28, the value of $ST_{elevation}$ is 0.030366. It is very likely this patient is suffering from Cardiomyopathy as the Baseline is lower than the ST-Segment. Similarly, Figure 29 depicts an averaged ECG where *Baseline* is higher than *ST*.

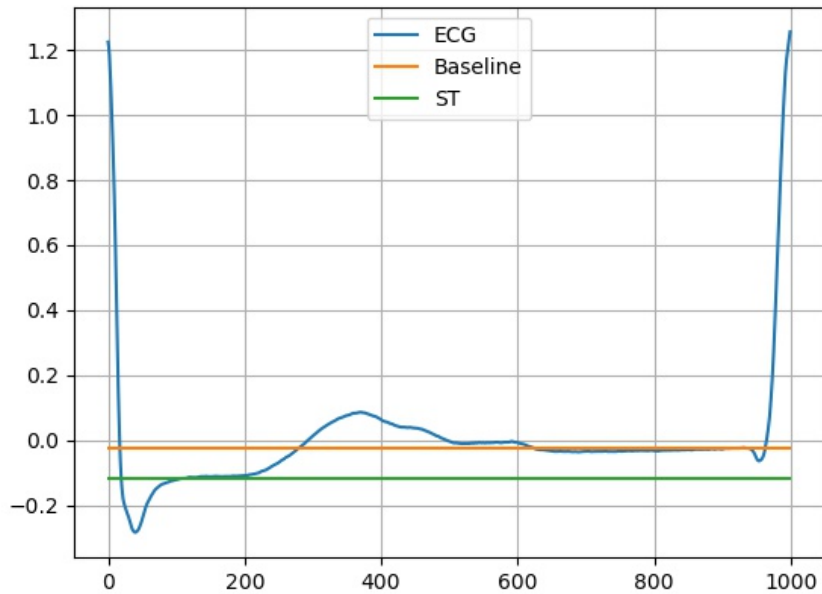


Figure 29: The Baseline in this ECG is higher than *ST*

For this case, value of $ST_{elevation}$ is -0.11502.

Summarising, a more *negative* value of $ST_{elevation}$ would imply that the patient may be suffering from Myocardial Infarction, and a more *positive* value implies that they may be suffering from Cardiomyopathy.

Q-Wave Steepness & Width

A steep and wide *Q*-wave implies that the patient is not '*healthy*', either suffering from Myocardial Infarction or Cardiomyopathy.

The averaged ECG in Figure 30 has been zoomed at the *Q*-wave to demonstrate an example of a '*healthy*' *Q*-Wave. The width is only 17 units wide (0.017 seconds) and the

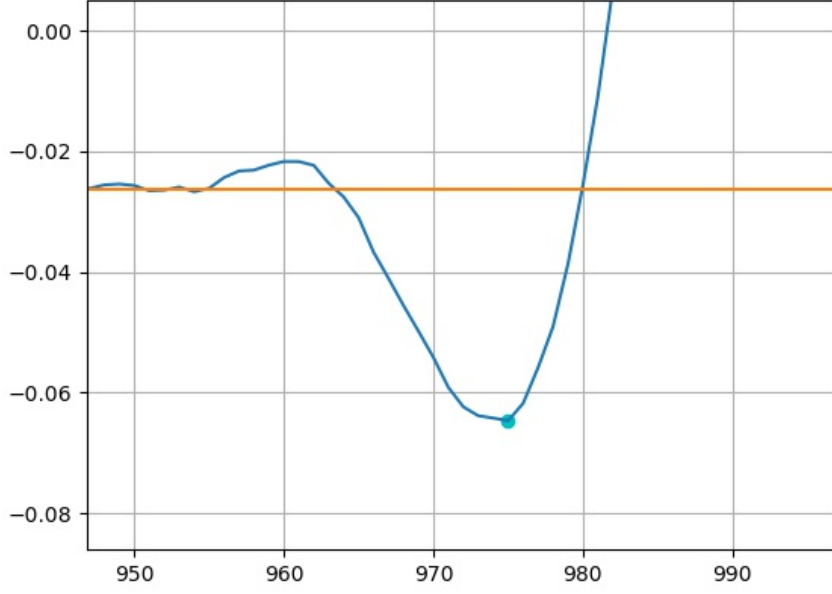


Figure 30: A close up of the Q -peak for the ‘*healthy*’ averaged ECG

Q -peak is relatively small in comparison to the R -peak and has an amplitude of 0.046631. However, consider the ECG in Figure 31.

The final portion of the averaged ECG is very different to that of the healthy ECG. Figure 32 shows the same ECG zoomed into the Q -point.

The width of the Q -peak is around 75 units long (0.075 seconds) and has a steepness that takes up a considerable proportion of the R peak, with an amplitude of 0.17887, relatively higher than that of the ‘*healthy*’ ECG.

Notice in the averaged ECG above that the Baseline can act as measure for the width of the Q -peak. By taking values of the averaged ECG that intercept the Baseline locally around the peak Q , the width, defined as Q_{Width} can be obtained in the following way

$$Q_{Width} = Q_{right} - Q_{left} \text{ where } Q_{left} < Q_{right} \text{ and } Q_{left}, Q_{right} \in [Q - \delta, 1000)$$

for some $\delta > 0$ such that $T_x < Q - \delta$ where T_x , as before, is the x -value at T . Q_{left} and

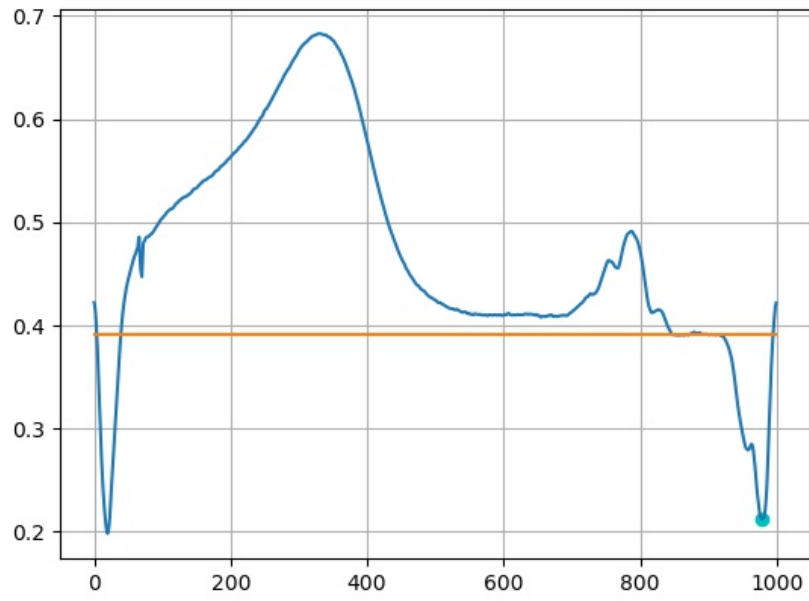


Figure 31: A averaged ECG with a high amplitude at the Q -peak

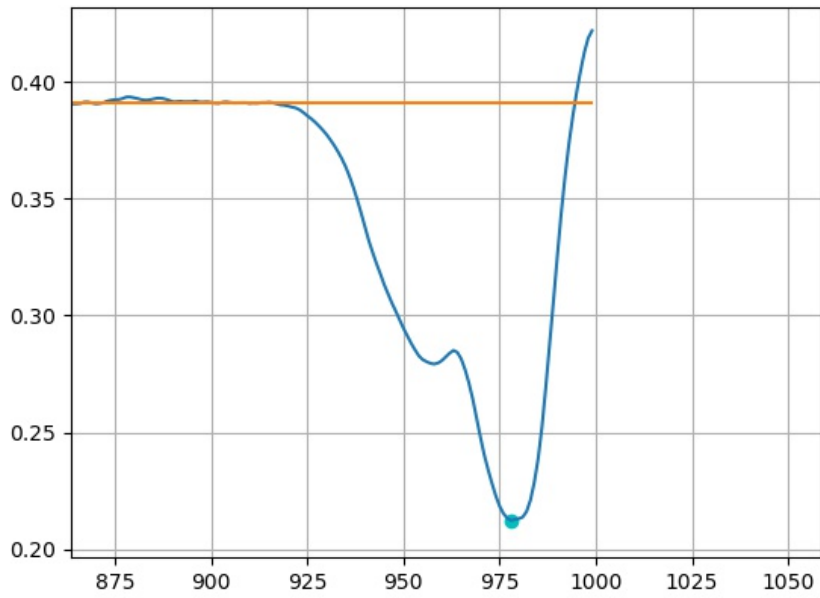


Figure 32: A close up of the Q -peak from the previous diagram

Q_{right} are chosen such that

$$\text{ECG}(Q_{\text{left}}) \approx \text{Baseline} \approx \text{ECG}(Q_{\text{right}}).$$

Q_{left} and Q_{right} are attained by evaluating each data point locally around Q until $Q_{\text{left}/\text{right}} \leq \text{Baseline}$ and $Q_{\text{left}+1/\text{right}+1} > \text{Baseline}$.

The amplitude of Q is defined as

$$Q_{\text{Amplitude}} = \text{Baseline} - Q.$$

QT-Interval

The distance between the points Q and T are also indicators of whether a patient is healthy or not.

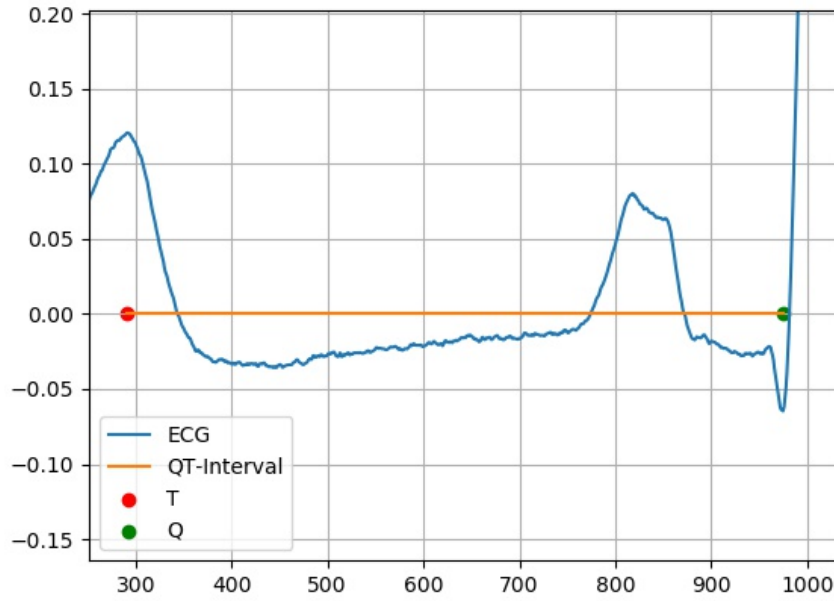


Figure 33: A ‘healthy’ QT-interval

The distance between T to Q in 33 is 684 units (0.684 seconds). However, consider the ECG in Figure 34.

The QT-interval here is 705 units (0.705 seconds), which is higher than normal, indicating signs of Cardiomyopathy.

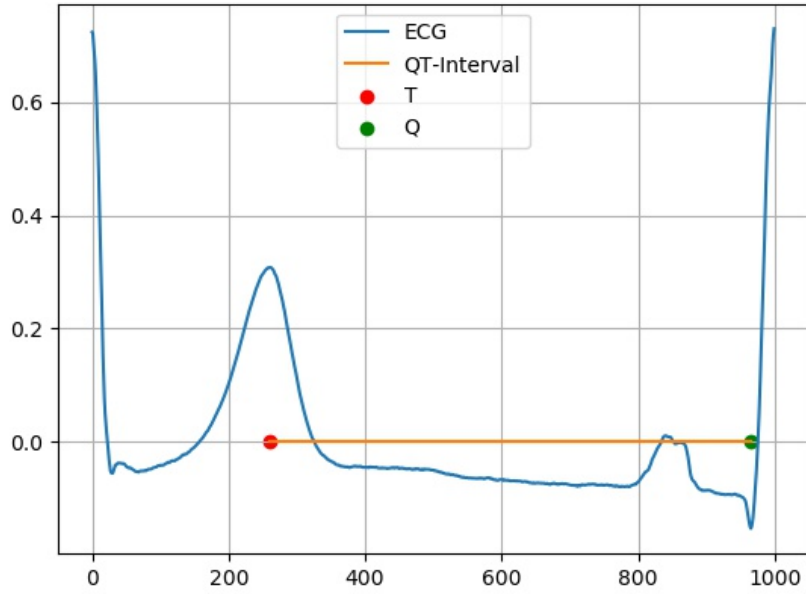


Figure 34: QT-interval showing signs of Cardiomyopathy

8.4 Limitations

As mentioned before, the limitations to the numerical approach of feature extraction is that since there is a substantial amount of variation in the trends of the average ECGs, the algorithm used consistently needed to be tuned in line with what the data looked like. This was quite time consuming and not practical if going through large amounts of data.

The means of extracting particular types of data, such as using derivatives close to 0, were troublesome since generally, the average ECG plots were not smooth. The algorithm in some cases detected a compatible derivative too prematurely and in other cases, does not find a derivative that meets the sufficient condition at all. For such cases, the information sought had to be found manually, which is more time consuming.

Furthermore, some of the average ECG plots do not show typical signs of the conventional ECG type shape. Rather some showed strange behaviour such as being mostly flat, which was not only difficult to interpret, but would not be suitable for the algorithm to be run on. On such cases, the relevant values had to be determined manually.

9 Decision Trees and Random Forests

9.0.1 Decision tree approach

With the features we developed in section 8.3 we decided that the logical model to use was a decision tree-based model as this would closely mirror how an actual cardiologist would diagnose these pathologies. A decision tree is a connection of nodes and works by first selecting a feature - the first node - and selecting a value in this feature's range to be a cut-off point whereby it will split the observations into two different categories. This process is then repeated with different features each time in order to classify each observation into a particular class. This seemed a good fit for the features in this report because as previously mentioned in section 5.2 certain features such as a longer Q wave duration or an elevated ST segment can be used to detect myocardial infarction and this could easily be modelled by a decision tree. An example of a possible classification tree is shown in Figure 34.

To determine the value of a feature a split is made and the value of p that minimises the Gini impurity which is defined as $I_G(p) = 1 - \sum_{i=1}^J p_i^2$ where J is the number of distinct classes and p_i is the probability of an observation being correctly labelled, is found.

Decision trees also implement a parameter called the *complexity parameter* or CP for short. This parameter is a value in the range (0,1) and is used to select the tree's optimal size. If the cost of adding another variable to the decision tree from the current node is above the value of CP, then the tree building does not continue. Put another way it could be said that tree construction does not continue unless it would decrease the overall lack of fit by a factor of CP.

Some initial exploration of this model revealed that an optimal value for the CP was 0.053 with a final tree size of 7 (see figure 36a). As can be seen in figure 36b the features used here were ST elevation, Q, T, R and T grad. The performance of this tree was evaluated using 10-fold cross validation and achieved an MCE of 0.314 on the training set which is a significant increase on our previous models. However the performance of this

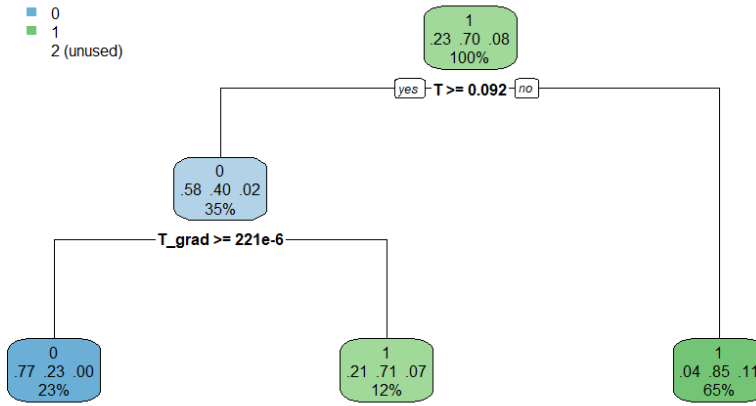
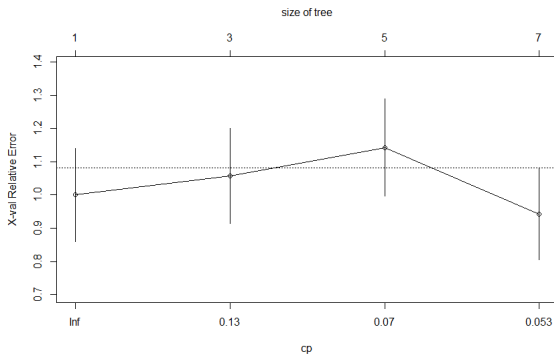
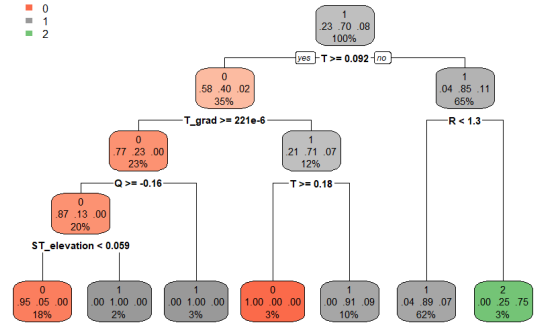


Figure 35: Example of a possible decision tree to classify heartbeats

model comes with some potential caveats as decision trees have a tendency to follow the noise in the training set and overfit. For this reason it was decided to extend this idea by building a random forest model.



(a)



(b)

Figure 36: (a) shows the range of possible CP values and their associated errors, (b) shows the optimal tree model developed on the training data.

9.0.2 Extension into random forests

Methodology Random forests are an ensemble method that work as thus, they apply the technique of bootstrap aggregating - known as bagging - to decision trees. Namely, given a training set $X = x_1, \dots, x_n$ with responses $Y = y_1, \dots, y_n$ a random sample is

selected with replacement (say B times) from the training set and fit trees to the samples.

The algorithm is as follows:

For $b=1,\dots,B$:

1. Sample with replacement n examples from X and Y , called X_b, Y_b say.
2. Train a classification tree on X_b, Y_b .

After this, training predictions are made for unseen examples by taking the majority vote across all trees.

This method leads to better overall performance as it decreases the variance of the model with no change in bias. So by taking the average of many trees the sensitivity to noise is reduced when compared to what is observed when using just one decision tree. One potential problem with this method is the possibility of highly correlated trees, if some of the features are very strong predictors for the response variable Y these particular features will be selected in many of the trees. This will cause trees to look very similar i.e. they will be similar to the performance of a single tree and therefore the problem of high variance will not have been solved. This is where another feature of random forests comes into play.

The above algorithm describes an approach in which all features are considered for splitting the data. Random forests instead consider a random subset $n < N$ of the features when considering a potential split.

Performance Analysing this model using 10 fold cross validation it was found that optimal settings for the number of trees to be used was 500 and performance was overall best when randomly selecting just one feature to be used at each split (see figure 37).

Average MCE was found to be around 0.24 and the final model selected achieved an MCE of 0.226, indicating a substantial increase in model performance over using just a single decision tree. Although as can be seen in the confusion matrix, given in table 4, the model actually achieved 0% accuracy when it came to classifying cardiomyopathy. The main reason for this is thought to be the incredibly small cardiomyopathy sample size.

Additionally the performance of this random forest was also compared against other potential classification methods, namely Support Vector Machines, Linear Discriminant

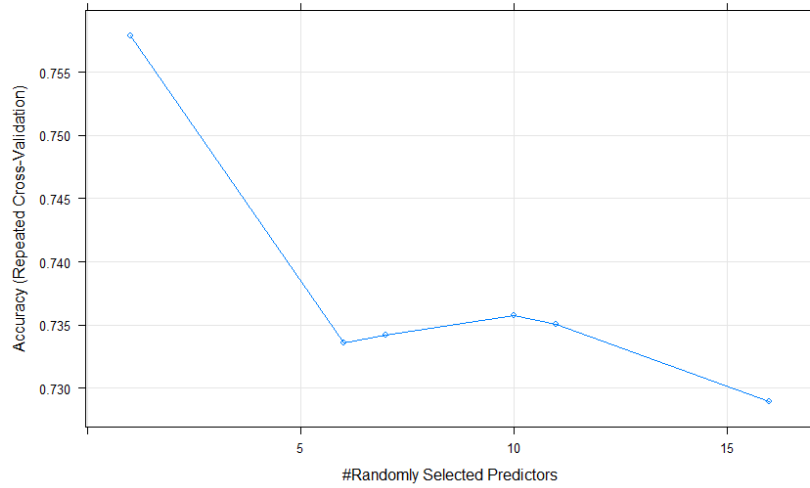


Figure 37: Graph showing accuracy performance randomly selecting from different amounts of predictors.

Analysis and Gradient Boosting Machines. Each model was tested using 10-fold cross validation once again. The results of this comparison can be seen in Figure 38 and it can be seen that the model seems to outperform these other methods further justifying our selection.

	0	1	2	Class Error
0	16	10	0	0.3846
1	6	73	1	0.0875
2	0	9	0	1.0000

Table 4: Confusion matrix of final model

On the test set this model achieved an MCE of 0.24 which was by far the best performance from any model in this report. Although it should be noted that the predictions this model made on the test set once again failed to classify any heartbeats as cardiomyopathy and this is an issue which will be discussed in our conclusions.

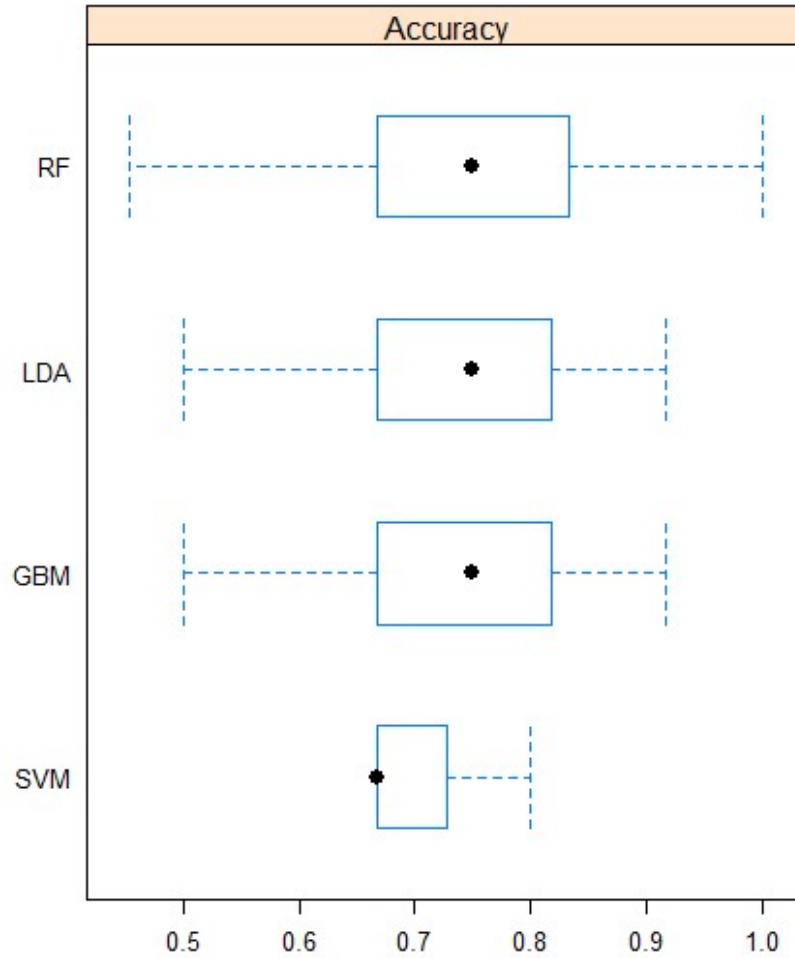


Figure 38: Boxplots of accuracy of a range of classification methods on the training data

10 Conclusions

Table 5 shows the MCE_{test} obtained by the best model from each method used in this report. These were: KNN using a holdout method of cross validation, discriminant analysis using leave-5-out cross validation 25 times and random forest using 10-fold cross validation.

Model	K-NN	FDA	Random Forest
MCE_{test}	0.47	0.6	0.24

Table 5: Classification error on test data using each model

To summarise the results, it was found that the models developed using the average heart beat data did not perform to the desired level achieving MCEs of 0.47 and 0.6 respectively meaning that roughly 1 in 2 ECGs were classified incorrectly. Once an algorithm was developed to extract the necessary morphological features from the heart-beat the performances increased across multiple models on the training data. Through comparison of these new models in 9.0.2 it was found that the best method to use was random forests. This was decided for two reasons, the first was that the model performed better than any other model that was considered. The second was that decision trees, and therefore random forests, are perfectly suited to this problem creating accurate and interpretable solutions. When the MCE for this model was evaluated using the test data it achieved a low MCE of 0.24, massively improving on any scores that other models had achieved. When looking at the features that frequently arose in the decision tree models, such as the one in figure 36b, it is not surprising to see ST-elevation being used to classify the final layer as this is one of the features that was seen to distinguish healthy, myocardial and cardiomyopathic individuals. On the whole this report largely succeeded in classifying different pathologies accurately although the problem of correctly classifying cardiomyopathy was not solved by any of the models. This is due to the features that were developed not easily determining the difference between cardiomyopathy and myocardial infarction as symptoms for these diseases can often be very similar.

It is thought that the test accuracy achieved with the random forest model is the upper limit of what this model can achieve given the current features. If further work were to be done on this problem it is the view of this report that time should be spent developing new features and new methods of extracting them, specifically features that would enable the classification of cardiomyopathy accurately. From reading around the topic, there are many methods that could be looked into in order to process the data more cleanly, such as the aforementioned filters to remove noise. This could be a very important step towards reducing the models MCE as some waves are incredibly noisy which may be causing inaccurate average heartbeats being extracted, throwing off all of the subsequent feature extraction. This could also be done through the use of discrete wavelet transforms, which are also used to extract features. Finally, given more time, another model that would

be interesting to implement would be neural networks. The background reading made it clear that neural networks were used frequently in this problem both for extracting features and then classifying ECGs. Due to this reports focus on creating interpretable models neural networks were ignored however they are a model which would strongly be considered if this report were to be redone.

Part III

Appendices

A Film Test Data

	userId	movieId		timestamp
1	73	49526	2009-10-15	07:01:18
2	187	47518	2009-03-16	00:22:15
3	150	788	2005-04-24	02:40:21
4	216	8830	2004-09-21	19:47:29
5	242	1227	2000-04-25	18:57:56
6	400	454	1996-08-16	15:19:14
7	256	3	1996-12-29	12:24:34
8	99	468	1999-09-29	07:43:27
9	102	1644	2000-05-10	18:40:51
10	480	31410	2010-05-01	00:58:20

Figure 39: Excerpt from the test data

B Feature Extraction Algorithm

```
import pandas as pd

dataset = pd.read_csv('avghb2.csv')

#Independent Variable
x=[]

for i in range(0,1000):
    x.append(i)

#Go through each row in the Training Set
for k in range(0,115):
    y = dataset.iloc[k,:].values

    S_point = -999
    S = 999
    for i in range(0,150):
        if y[i] < S:
            S = y[i]
            S_point = int(x[i])

    #Used if T is a maximum
    T = -999
    for i in range(S_point, 550):
        if y[i] > T:
            T = y[i] # y value of T
            T_point = int(x[i]) # x value of T

    #Used if T is a minimum
    T = 999
```

```

for i in range(151, 550):
    if y[i] < T:
        T = y[i]
        T_point = int(x[i])

P_point = -999
P = -999
for i in range(T_point, 900):
    if y[i] > P:
        P = y[i] # y value of P
        P_point = int(x[i]) # x value of P

Q_point = -999
Q = 999
for i in range(940, 1000):
    if y[i] < Q:
        Q = y[i] # y value of Q
        Q_point = int(x[i]) # x value of Q

# Calculating Baseline by assessing "flatness"
# i.e. stop when the derivative is relatively close to 0
# AND check the y-values PQ-push to
# find the midpoint between Q_point and P_point
# and "push" the indexes to avoid code
# stopping prematurely

Baseline = -999
PQ_push = int((Q_point - P_point)/2)  stoping prematurely
for i in range(P_point, Q_point):

```

```

if (y[i+PQ_push]-y[i+PQ_push+1])/(x[i+PQ_push]-x[i+PQ_push+1])
    <= 0
and y[i+PQ_push]-y[i+PQ_push+1] <= 10**-3:
    Baseline = y[i+PQ_push]
    break

# interpolating ('int') to find the
# first instance where the gradient
# is less than or equal to 0.001
S_int = -999
for i in range(S_point, 150):
    if y[i+50]-y[i+51] <= 10**-3:
        S_int = y[i+50]
        ST_int_point = int(x[i+50])
        break

# find the largest value after S up to 250
S_max = -999
S_max_point = 999
for i in range(S_point, 250):
    if y[i] >= S_max:
        S_max = y[i]
        S_max_point = x[i]

#Take the lower value to be defined as the ST-Segment
ST_point = min(S_max_point, ST_int_point)
ST_segment = y[ST_point] # set ST

# ST-elevation
ST_elevation = ST_segment - Baseline

```

```

# T_Baseline
T_Baseline = T - Baseline

# Find the gradient between the midpoint of ST and T to T
ST_half_point = int((T_point-ST_point)/2)
T_grad = (ST_segment - y[ST_half_point])/(ST_point - ST_half_point)

# QT Interval
QT_interval = Q_point-T_point

# Q_amplitude
Q_Amp = Baseline - Q

# Find Q_left
Q_left = -999
for i in range(0,1000):
    if y[Q_point-i-1] - y[Q_point-i] <= 0:
        Q_left_point = int(x[Q_point - i]) # x value at Q_left
        Q_left = y[Q_point-i] # y value at Q_right
        break

# Find Q_right
Q_right_point = -999
for i in range(Q_point, 999):
    if y[i] <= Q_left and y[i+1] > Q_left:
        Q_right_point = int(x[i+1]) # x value at Q_right

#Q_width
Q_width = Q_right_point - Q_left_point

```


Part IV

Bibliography

References

- [1] D. C. Montgomery, E. A. Peck and G. G. Vining *Introduction to Linear Regression Analysis* John Wiley & Sons (2012)
- [2] G. Jiang and W. Wang, *Error estimation based on variance analysis of k-fold cross-validation* Pattern Recognition 69 (2017): 94-106
- [3] G. Seber *The linear Model and Hypothesis: A General Unifying Theory* (Springer, 2015)
- [4] R. Bell, Y. Koren and C. Volinsky, *Matrix Factorization Techniques for Recommender Systems* Computer, vol. 42, no. 08, pp. 30-37, 2009.
- [5] E. Chen. *Is there any summary of top models for the Netflix prize? What are the high level and intuitive ideas behind the winning models that were finally used in the ensemble learning by top teams?* (2011) Available at: <https://www.quora.com/Netflix-Prize/Is-there-any-summary-of-top-models-for-the-Netflix-prize-What-are-the-high-level-and-intuitive-ideas-behind-the-winning-models-that-were-finally-used-in-the-ensemble-learning-by-top-teams> [Accessed 03/04/2019]
- [6] R. M. Bell, Y. Koren, C. Volinsky *The bellkor 2008 solution to the netflix prize* Statistics Research Department at AT& T Research 1 (2008)
- [7] P. Xia, L. Zhang, F. Li *Learning similarity with cosine similarity ensemble* Information Sciences 307 (2015):39-52
- [8] H-F, Sun, J-L. Chen, G. Yu, C-C. Liu, Y. Peng, G. Chen, B. Cheng *JacUOD: A new similarity measurement for collaborative filtering* Journal of Computer Science and Technology 27.6 (2012): 1252-1260

- [9] B. Kumar *A novel latent factor model for recommender systems* JISTEM-Journal of Information Systems and Technology Management 13.3 (2016): 497-514.
- [10] S. Kant and T. Mahara *Nearest biclusters collaborative framework with fusion* Journal of Computational Science, vol. 25 (2018): 204-212
- [11] T. Cover and P. Hart *Nearest neighbour pattern classification* IEEE Transactions on Information Theory 13.1 (1967): 21-27
- [12] A. Singh *A practical introduction to K-Nearest Neighbours algorithm for regression* (2018) Available at: <https://www.analyticsvidhya.com/blog/2018/08/k-nearest-neighbor-introduction-regression-python/> [Accessed on 03/04/2019]
- [13] S. Gaw, D. Gerahrd, C. N. Glover, F. Mohamed, E. Moltchanova *On correlation analysis of many-to-many observations: an alternative to Pearson's correlation coefficient and its application to an ecotoxicological study* Australian & New Zealand Journal of Statistics 59.4 (2017): 371-87
- [14] *Pearson's Correlation Coefficient* University of the West of England Available at: <http://learntech.uwe.ac.uk/da/Default.aspx?pageid=1442> [Accessed on 03/04/2019]
- [15] The Shape of Data *K-Nearest Neighbours* (2013) Available at: <https://shapeofdata.wordpress.com/2013/05/07/k-nearest-neighbors/> [Accessed on 03/04/2019]
- [16] D. Sadhukhan and M. Mitra *R-Peak Detection Algorithm for Ecg using Double Difference And RR Interval Processing* Procedia Technology vol. 4 (2012): 873-877
- [17] N. Strodthoff and C. Strodthoff *Detecting and interpreting myocardial infarction using fully convolutional neural networks* Institute of Physics and Engineering in Medicine (2019)

- [18] Q. A. Rahman, L. G. Tereshchenko, M. Kongkatong, T. Abraham, M. R. Abraham, H. Shatkay *Utilizing ECG-Based Heartbeat Classification for Hypertrophic Cardiomyopathy Identification* IEEE Trans Nanobioscience (2015)
- [19] NCSS Statistical Software *Equality of Covariance* Available at: https://ncss-wpengine.netdna-ssl.com/wp-content/themes/ncss/pdf/Procedures/NCSS/Equality_of_Covariance.pdf [Accessed on 03/04/2019]
- [20] Harvard Health Letter *Understanding the ECG: Reading the waves* Harvard Health Publishing Available at: <https://www.health.harvard.edu/heart-health/understanding-the-ecg-reading-the-waves> [Accessed on 03/04/2019]
- [21] P. Kharkar *Linear Discriminant Analysis using Python* The Java Geek Available at: <http://www.thejavageek.com/2018/04/30/linear-discriminant-analysis-using-python/> (2018) [Accessed on 03/04/2019]
- [22] O. Martin, L. Sanchez, P. Sanchez, P. Lopez, P. Lopez, P. Juan T. Juan. *Heartbeats Do Not Make Good Pseudo-Random Number Generators: An Analysis of the Randomness of Inter-Pulse Intervals Entropy* (2018)
- [23] Clinical ECG Interpretation *ECG Interpretation Part 1: definitions, criteria, and characteristics of the normal ECG waves, intervals, durations and rhythm* Available at: <https://ecgwaves.com/ecg-normal-p-wave-qrs-complex-st-segment-t-wave-j-point/> [Accessed on 03/04/2019]
- [24] S. Celin and K. Vasanth *ECG Signal Classification Using Various Machine Learning Techniques* Journal of Medical Systems (2018): 42-241
- [25] B. Heden, H. Ohlin, R. Rittner, K. Edenbrandt *Acute Myocardial Infarction Detected in the 12-Lead ECG by Artificial Neural Networks* Circulation (1997)
- [26] M. D. Ekstrand, J. T. Riedl and J. A. Konstan *Collaborative Filtering Recommender Systems* Foundations and Trends in Human-Computer Interaction (2011), vol.4:No.2: 81-173

- [27] J. B. Schafer, D. Frankowski, J. Herlocker, S. Sen *Collaborative Filtering Recommender Systems* Lecture notes in computer science vol. 4321
- [28] The statistics portal *Net advertising revenues of YouTube in the United States from 2015 to 2018* Available at: <https://www.statista.com/statistics/289660/youtube-us-net-advertising-revenues/> [Accessed on 03/04/2019]
- [29] B. Baird, A. Charles, M. Honeyman, D. Maguire, P. Das *Understanding Pressures in general practice* The King's Fund (2016)
- [30] British Heart Foundation *Facts and figures* Available at: <https://www.bhf.org.uk/for-professionals/press-centre/facts-and-figures> [Accessed on 03/04/2019]
- [31] U. R. Acharya, P. S. Bhat, S. S. Iyengar, A. Rao, S. Dua *Classification of heart rate data using artificial neural network and fuzzy equivalence relation* Pattern Recognition vol.36, issue 1 (2003): 61-68
- [32] Cnet *YouTube's AI is the puppet master over most of what you watch* Available at: <https://www.cnet.com/news/youtube-ces-2018-neal-mohan/> [Accessed on 03/04/2019]
- [33] M. AlGhatrif and J. Lindsay *A brief review: history to understand fundamentals of electrocardiography* J Community Hosp Intern Med Perspect (2012)
- [34] D. Anh, S. Krishnan, F. Bogun *Accuracy of electrocardiogram interpretation by cardiologists in the setting of incorrect computer analysis* Journal of Electrocardiol (2006)
- [35] Cardiomyopathy UK *What is cardiomyopathy?* Available at <https://www.cardiomyopathy.org/about-cardiomyopathy/what-is-cardiomyopathy> [Accessed on 03/04/2019]