University of Cape Town

Department of Statistical Sciences

STA5076Z Supervised Learning

Assignment 3

EVRCHA001 - Charl Everts

19 July 2020

Department of Statistical Sciences

Plagiarism Declaration Form

A copy of this form completed and signed, to be attached to all coursework submissions to the Statistical Sciences Department. Submissions without this form will not be marked.

COURSE CODE:	STA5076Z
COURSE NAME:	Supervised Learning
STUDENT NAME:	Charl Everts
STUDENT NUM:	EVRCHA001

Plagiarism Declaration

1. I know that plagiarism is wrong. Plagiarism is to use another's work and pretend that it is one's own.
2. I have used a generally accepted citation and referencing style. Each contribution to, and quotation in, this tutorial/report/project from the work(s) of other people has been attributed and has been cited and referenced.
3. This tutorial/report/project is my own work.
4. I have not allowed, and will not allow, anyone to copy my work with the intention of passing it off as his or her own work.
5. I acknowledge that copying someone else's assignment or essay, or part of it, is wrong, and declare that this is my own work.

Note that agreement to this statement does not exonerate you from the University's plagiarism rules (http://www.uct.ac.za/uct/policies/plagiarism_students.pdf).

Signature: [illegible] Date: 19 July 2020

ABSTRACT

The purpose of this report is to evaluate the classification accuracy of support vector machines and artificial neural networks. The dataset investigated is the Framingham dataset based on coronary heart disease studies on patients in Framingham, Massachusetts.

A total of 15 predictor variables are used to predict the target variable “RiskofCHD”. The target variable predicts whether or not a patient is at risk of coronary heart disease within the next ten years, based on medical, demographic and behavioural metrics.

In total, 14 models were trained on an 80/20% train and test split from the original training dataset. The best SVM model (svm_7) had a test set classification accuracy of 87.69% and a run-time of 1 second. The best neural network model (nn_1) had a test set classification accuracy of 87.86% and a run-time of 55 seconds.

Table of Contents

List of Figures

List of Tables

Abbreviations/Symbols

CHD	coronary heart disease
BP	blood pressure
BMI	body mass index
HD	heart disease
SVM	support vector machine
NN	neural network
§	section
C	cost parameter

Chapter 1

Introduction

1.1 The “Framingham” Dataset

Cardiovascular risk prediction models play a vital role in the prevention and management of cardiovascular diseases. Models used in clinical practice aim to identify and treat high-risk patients.

The Framingham dataset stems from a cardiovascular study conducted on residents in Framingham, Massachusetts. The Framingham Risk Score estimates a patient’s ten-year risk of coronary heart disease (CHD). The data contains patient information spanning 3392 entries and 16 variables. Each attribute is either a demographic, behavioural or medical risk factor.

Demographic:
- Sex: male or female
- Age: Age of the patient
- Education: 1 = Some High School | 2 = High School | 3 = Some College | 4 = College

Behavioural:
- CurrentSmoker: if the patient is a smoker at the time of evaluation.
- CigsPerDay: average number of cigarettes smoked per day.

Medical:
- BPMeds: if the patient is on blood pressure medication.
- PrevalentStroke: if the patient had previously had a stroke.
- PrevalentHyp: if the patient suffers from hypertension.
- Diabetes: if the patient is diabetic.
- Tot Chol: total cholesterol level.
- SysBP: systolic blood pressure.
- DiaBP: diastolic blood pressure.
- BMI: Body Mass Index.
- HeartRate: heart rate.
- Glucose: glucose level.

The target variable is predicted as either “Yes” or “No” – whether the patient will suffer from CHD within the next ten years.

1.2 Exploratory Data Analysis

Plotting the missing values by feature indicates that glucose (Figure 1.1) has an abnormally high percentage of missing values. A value larger than 2% (turquoise) was considered problematic, and could not simply be omitted from the dataset. A reason for the missing values may be explained by the fact that doctors did not measure glucose levels at the time of evaluation, or did not have the appropriate test kits. The missing values were replaced by the median value for glucose.

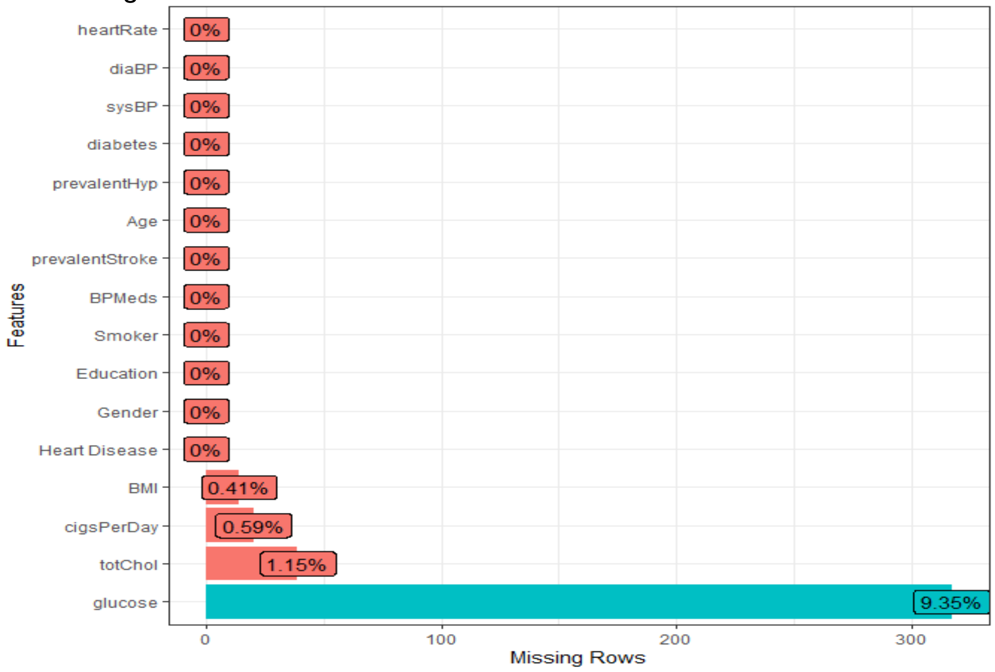


Figure 1.1: *Percentage of NA values by feature.*

Of the 3392 patients, 525 had some form of heart disease (Figure 1.2) at the time of their evaluation. Of the 1469 males in the dataset, 280 suffered from CHD (19.1%), while 245 of the 1923 females (12.7%) had suffered from CHD. This indicates that in the population, males were more likely to have suffered from CHD than females.

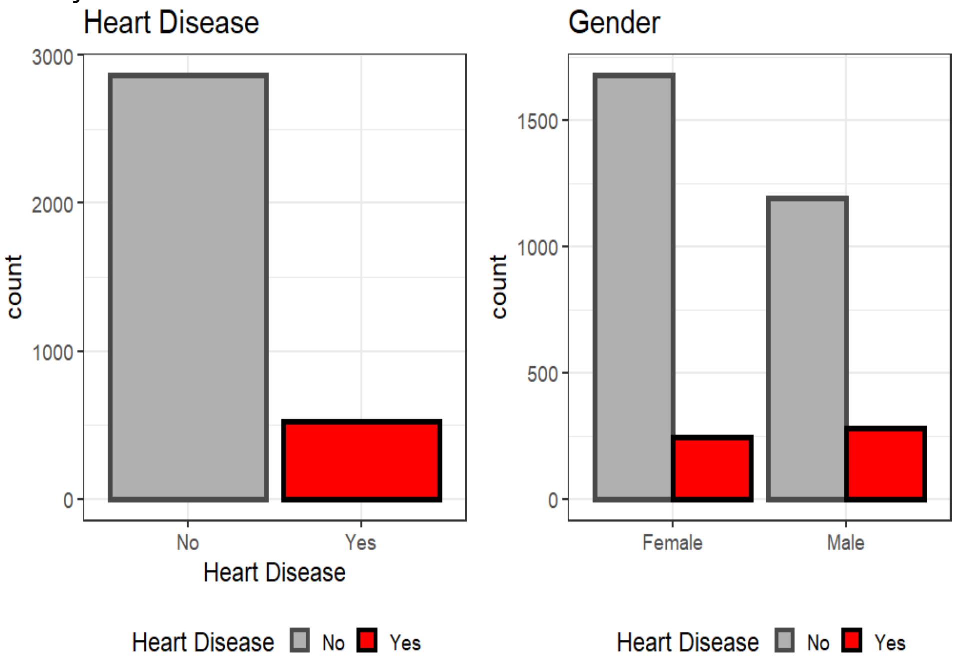


Figure 1.2: *Heart Disease by Gender.*

Evaluating heart disease by age (Figure 1.3) depicts the increase in heart disease as people get older. Patients aged 50 and older have a much higher propensity for heart disease than patients in the 30-39 age group category (Table 1.1).

AGE	HEART DISEASE
30-39	4.2%
40-49	10.8%
50-59	20.1%
60-69	35.8%

Table 1.1: *Patients with heart disease by age*.

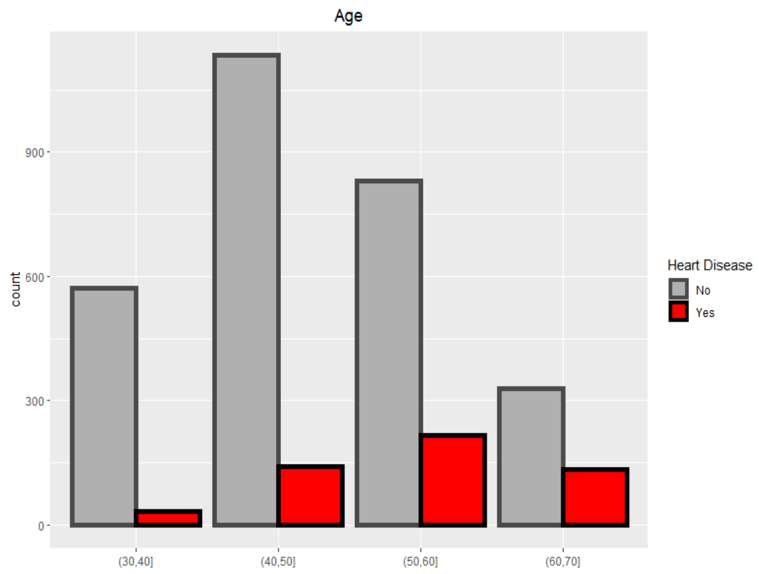


Figure 1.3: *Heart Disease by Age.*

Systolic and Diastolic blood pressure are two of the most important metrics when evaluating risk of CHD. There exists evidence (Figure 1.4) of a positive relationship between age and systolic BP, which is even more intriguing when compared to patients who have heart disease and those who do not. Patients without heart disease have a noticeably lower systolic and diastolic BP than those who do.

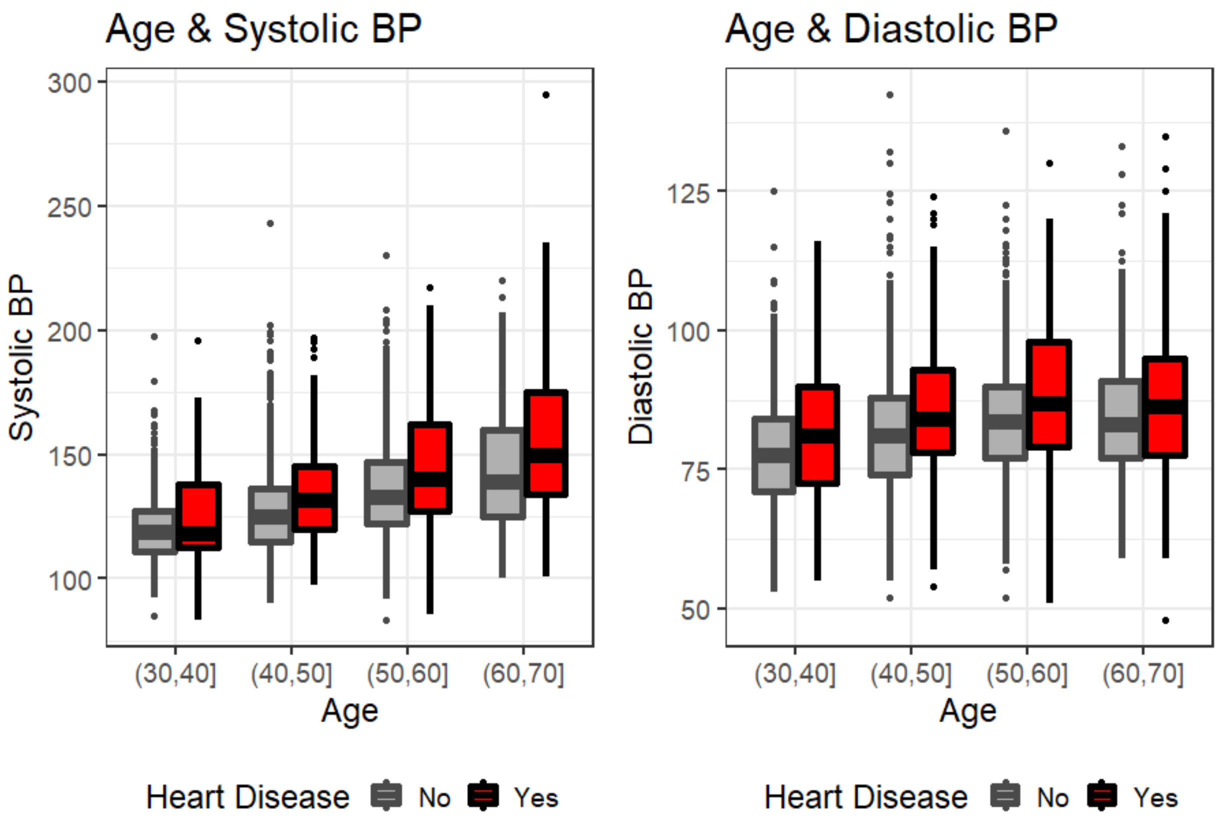


Figure 1.4: *Systolic and Diastolic BP by Age and Heart Disease.*

Figure 1.5 illustrates the relationship between hypertension and age, coloured by patients who suffer from heart disease. The majority of observations left of the dashed line indicating median age of the dataset (younger than 49 years) are not hypertensive and do not suffer from heart disease. While the majority of patients over the age of 49 years suffered from both hypertension and heart disease. This makes a strong case that hypertension is a precursor to heart disease in the over 49-year age category.

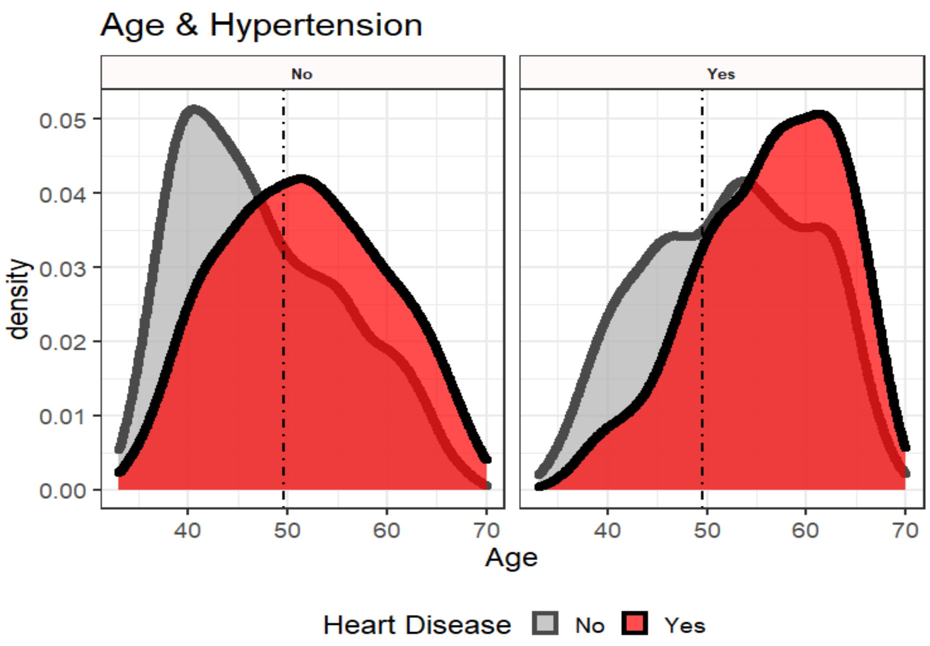


Figure 1.5: *Age and Hypertension by Heart Disease.*

The data is normally distributed, with a few cases of severe skewing (Figure 1.6). Glucose and cholesterol have particularly high levels of skewness to the right: 6.3744 and 0.9521. Tukey's test for outliers is conducted on both attributes, resulting in the boxplots in Figure 1.7.

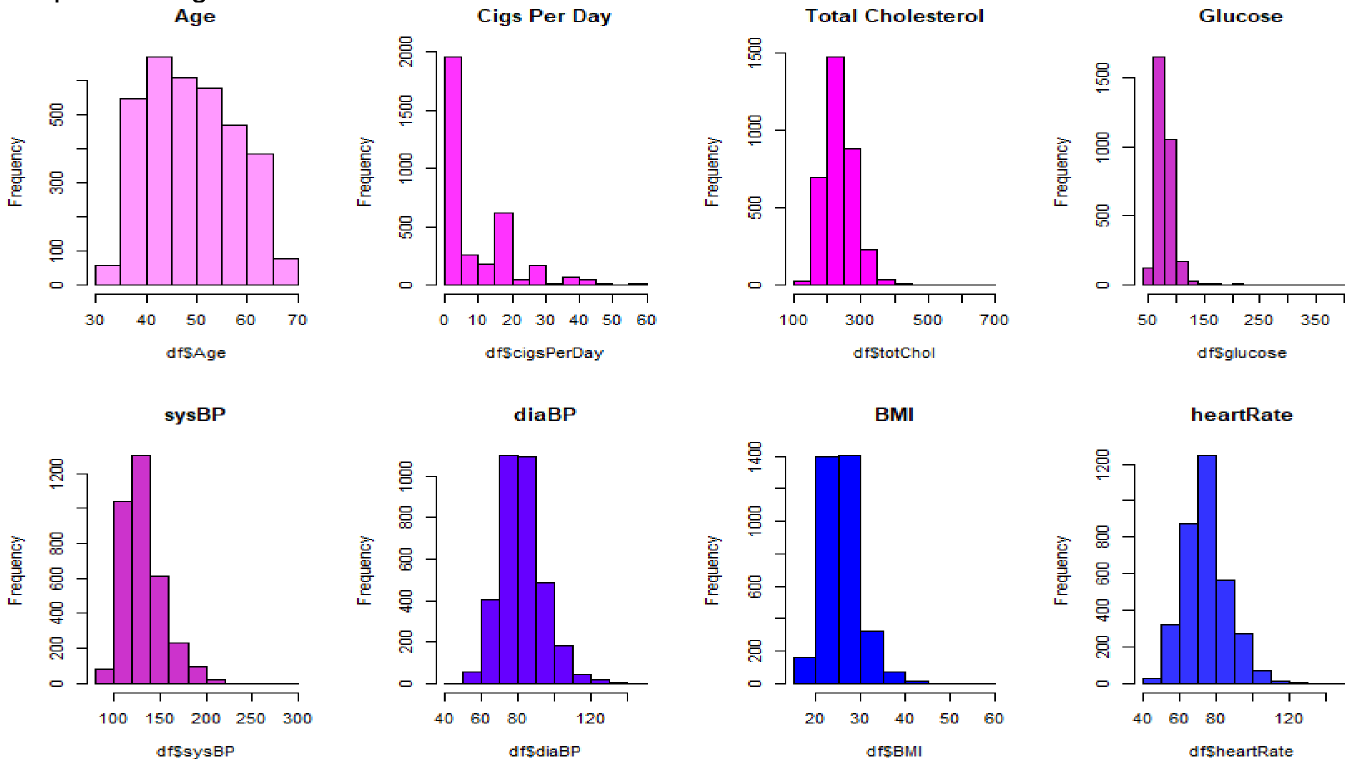


Figure 1.6: *Data distribution by feature.*

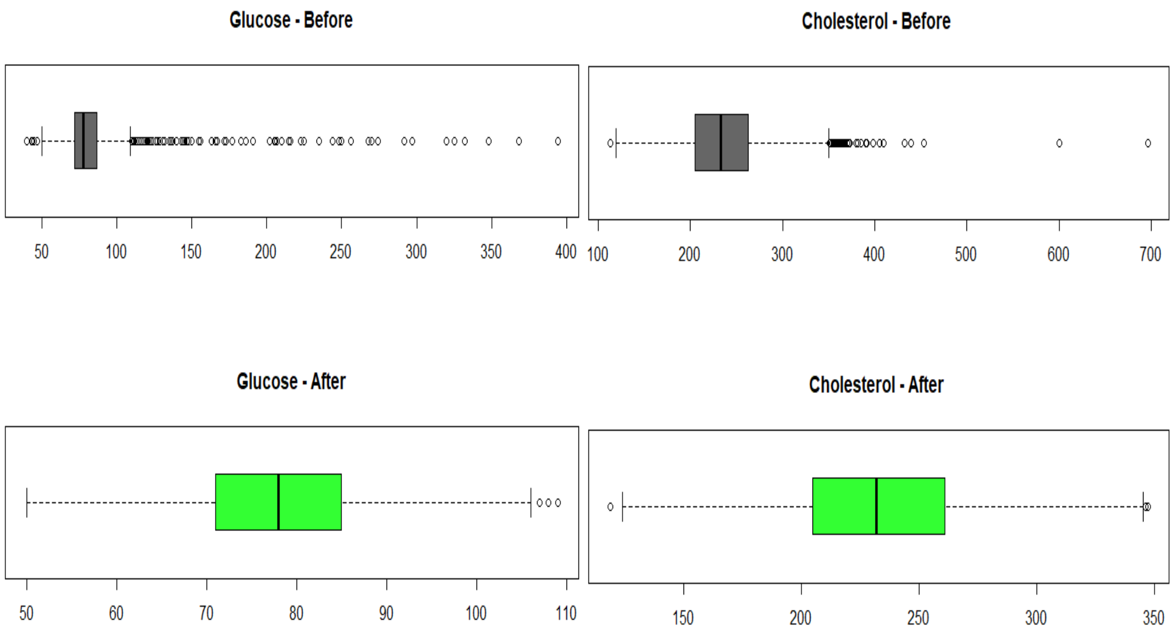


Figure 1.7: *Glucose and Cholesterol boxplots before and after outlier removal.*

The resulting correlation plots in Figure 1.8 depict only the most highly correlated variables in the dataset after the removal of outliers. The systolic BP and diastolic BP have the highest correlation. This is expected as they measure the same metric – blood pressure – which is most likely conducted in a similar fashion. The next strongest correlations exist between systolic BP, diastolic BP and prevalent hypertension. This makes sense, since high blood pressure readings are associated with people who have a history of hypertension.

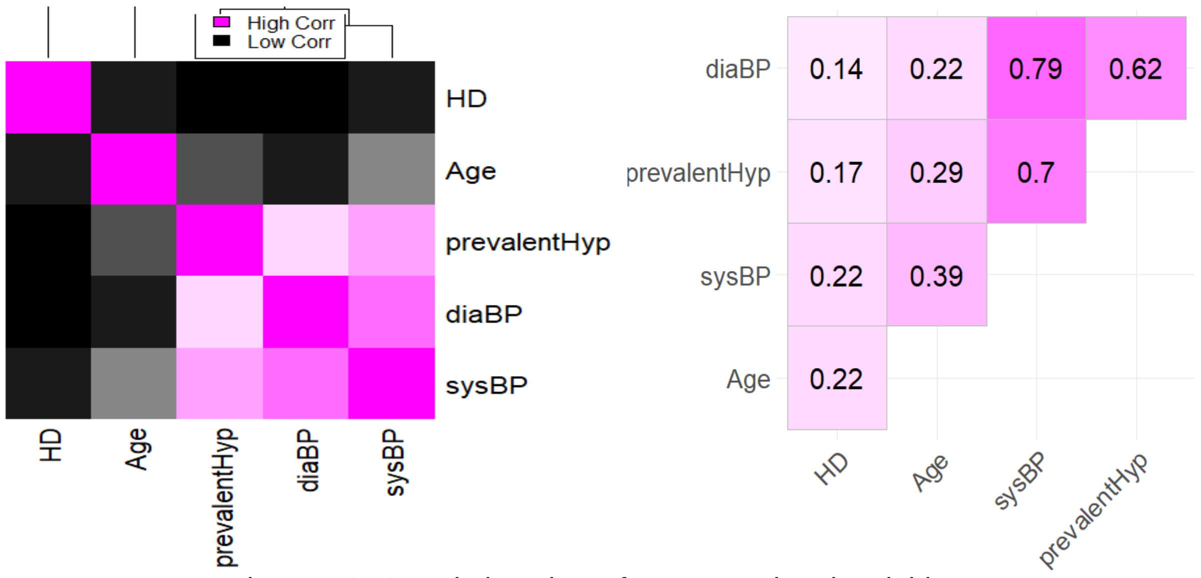


Figure 1.8: *Correlation plots of most correlated variables.*

1.3 Assignment Objectives

Using the training and test data provide, develop a Support Vector Machine (SVM), and a Neural Network (NN) model to predict the Framingham Cardiovascular Risk Score (TenYearCHD) using the variables available.

The dataset provided is comprised of 4240 observations of 17 variables. The traintest variable indicates which entries must be allocated to the training and testing datasets respectively. The best models must be used to make predictions on the final test set – those values where TenYearCHD are marked NA.

Models must be built using the following supervised learning techniques:

1. Support Vector Machines.
2. Neural Networks.

Model performance is based on the classification accuracy of the unseen test data. A description of each technique, a comparison of their performance, as well as motivation behind the choice of regularisation mechanism and hyper-parameters is discussed.

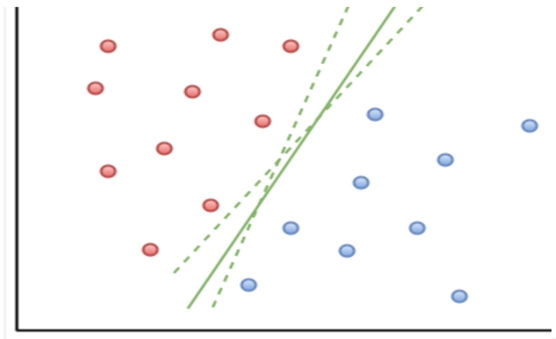
Chapter 2

Support Vector Machines

2.1 Methodology

An SVM attempts to find the optimal separation boundary between datapoints that have different classifications. The boundaries can take almost any shape; such as linear, radial or polynomial.

The most basic case where an SVM method can be implemented is in a linearly separable, binary classification problem (Figure 2.1). An infinite amount of possible separation lines exists. The goal of an SVM is to find the optimal separation boundary. An SVM finds this boundary by formulating a *maximal margin classifier (*Figure 2.2)*.* The algorithm attempts to maximise the distance between the closest point on each side of the separation boundary.



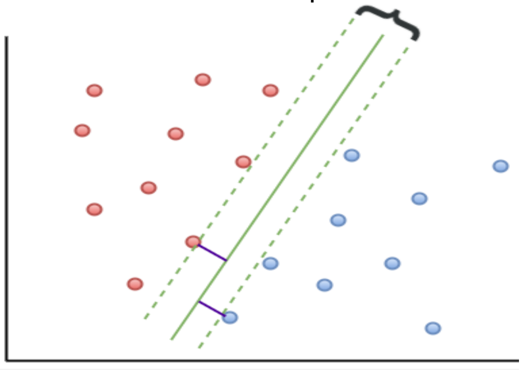
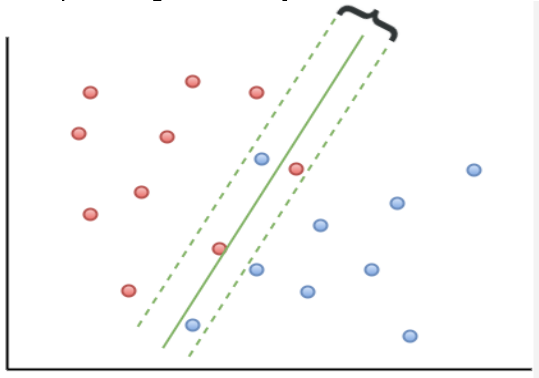


Figure 2.1: *Linearly separable data*. Figure 2.2: *Optimal separation boundary.*

The lines on either side of the boundary are called *support vectors.* There is a negative relationship between number of support vectors and the generalizability of the boundary. An increase in number of support vectors required to classify a dataset means that the dataset overlaps, and hence not perfectly separable.

Real world applications however rarely provide datasets that are perfectly linearly separable. When data points start overlapping, the *soft margin classification* (Figure 2.3) approach is implemented. This allows points with small deviations from the boundary to be misclassified.

The "cost" parameter (C) is a penalty associated with each misclassification. For large values of C, the algorithm will select a smaller-margin boundary if that boundary more accurately classifies the training data. Conversely, a small C will cause the optimizer to look for a larger-margin separating boundary, even if that boundary misclassifies more points.



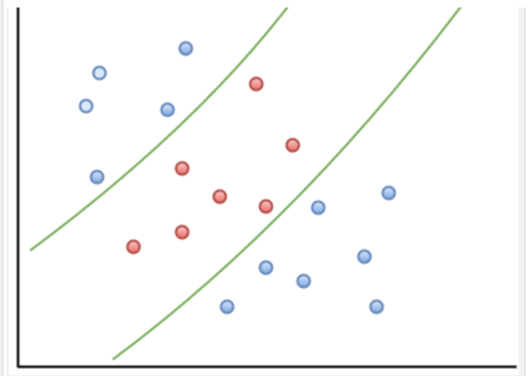


Figure 2.3: *Soft margin classifier.* Figure 2.4: *Non-linear separation boundary.*

A higher C results in more precise estimates and therefore a low bias, however overfitting which means high variance. A low C means less strict prediction constraints, resulting in higher bias.

Figure 2.4 depicts a non-linear separation boundary. A statistical computing program like R deals with this type of non-linear problem by using kernels. A kernel maps a classification problem to a metric space in which the datapoints are separable. The resulting decision boundary, which is linear in the enlarged feature space, will be nonlinear when transformed back onto the original feature space.

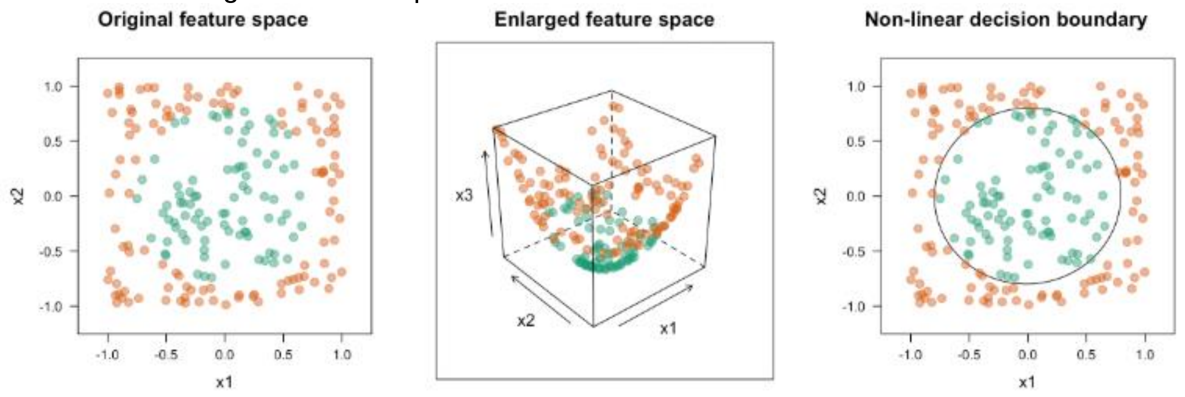


Figure 2.5: *Non-linear separation boundary in the original feature space (left pane and right pane). In 3D feature space (middle pane), data separable by a hyperplane.*

Utilising various kernel functions, SVMs are extremely flexible and well-suited to estimating complex nonlinear decision boundaries.

2.2 Results

Using the `kernlab` (models svm_1 – svm_4) and `e1071` (models svm_5 – svm_9) libraries, the SVM models were built using an array of methods using several combinations of variables, cost, and gamma values. The models are named svm_n, with n indicating the number of the model in the series.

Svm_1 is a radial SVM using all variables as predictors, which implemented a 10-fold cross-validation resampling method to find the optimal cost parameter and hence best accuracy. The model took 80 seconds to build.

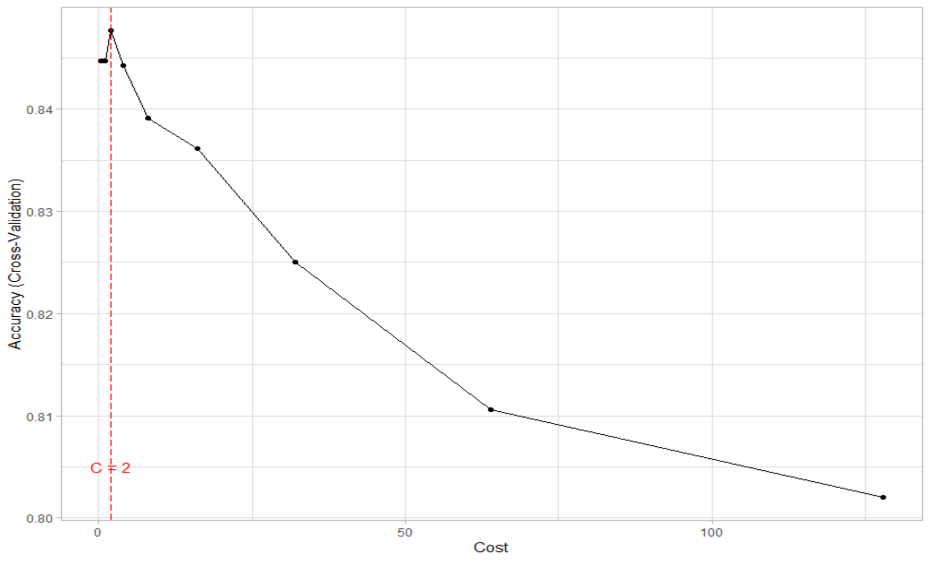


Figure 2.6: *10-fold cross-validation conducted to find optimal cost parameter (C=2).*

Using C = 2, svm_1 correctly classified the training set data on 84.77% of the observations. When used to predict the unseen test data, it correctly classified 84.76% of observations.

		True		**Accuracy_svm_1 (%)**
		0	1	
Predicted	0	490	4	**84.76**
	1	85	5	

Table 2.1: *Classification Accuracy for svm_1.*

Svm_2 was another radial SVM which selected its parameters based on maximizing the area under the curve (AUC) of the receiver operating characteristic (ROC) curve, using 10-fold cross-validation as training control. All variables were used as predictors. The model took 200 seconds to build. Using a cost = 16, svm_2 correctly classified the training set data on 84.56% of the observations. When used to predict the unseen test data, it correctly classified 84.42% of observations.

		True		**Accuracy_svm_2 (%)**
		0	1	
Predicted	0	492	2	**84.42%**
	1	89	1	

Table 2.2: *Classification Accuracy for svm_2.*

Figure 2.2 is a variable importance plot derived from svm_2. The findings agree with the conclusion made in §1.2: that age along with systolic and diastolic BP are the most significant factors in predicting a patient’s CHD risk. Variable exclusion in future models is therefore based on Figure 2.2 and §1.2.

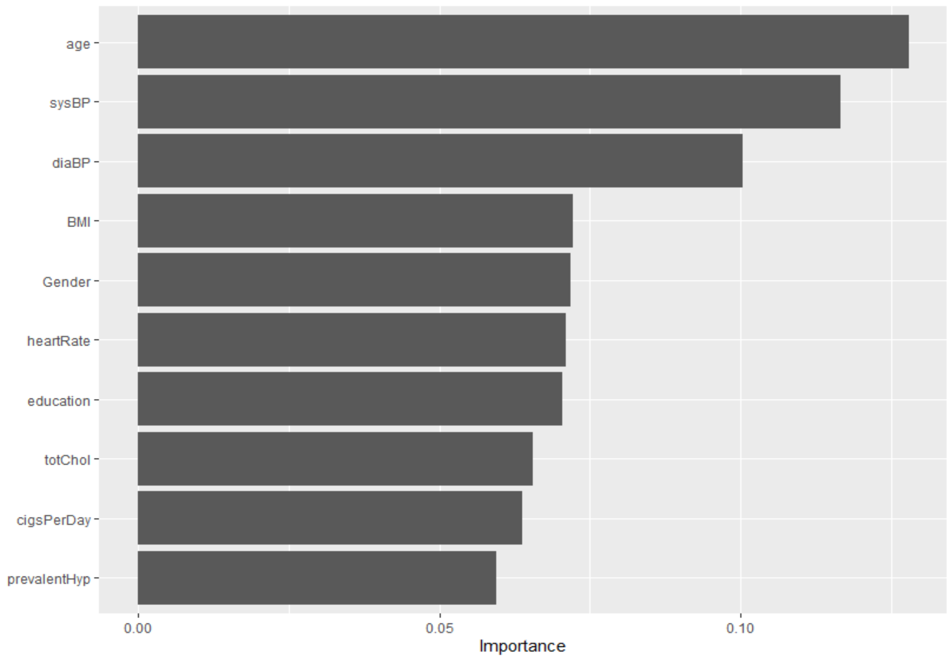


Figure 2.7: *Variable importance plot based on svm_2.*

The partial dependence plots in Figure 2.3 depicts the functional relationship between individual input variables and predictions of the target variable. It depicts how the prediction of CHD risk is affected by predictor variables.

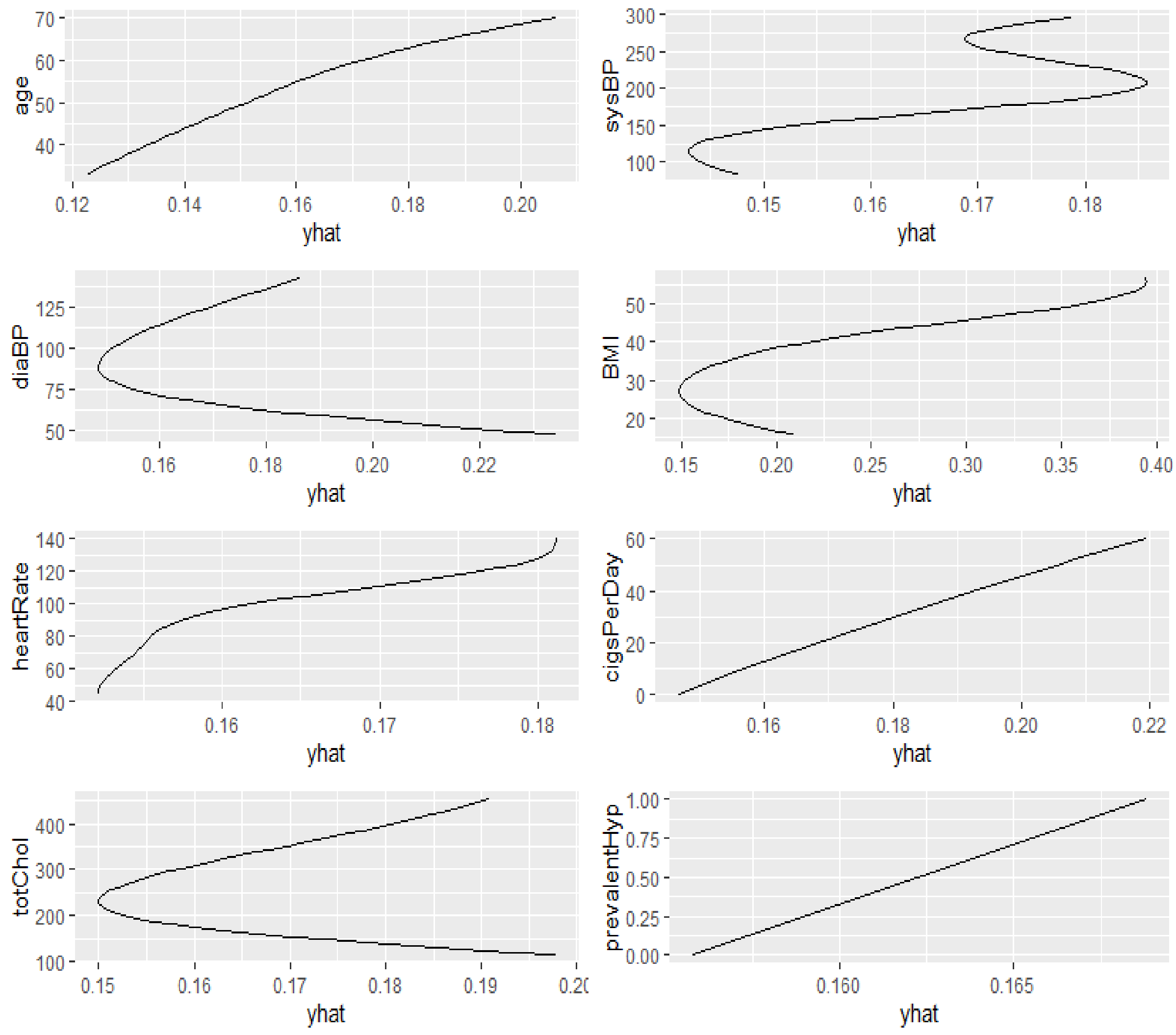


Figure 2.8: *Partial dependence features based on svm_2.*

It is evident that a near perfect positive linear relationship exists between the age, cigarettes per day, prevalent hypertension, and the target variable. The plots also illustrate the typical ranges for which risk of CHD is minimized for the respective variables. For example, a healthy individual should strive for a systolic BP in the range of 120-125 units.

Table 2.3 summarises the next two models based on the variables used and respective classification accuracies. Both were built using the radial method and optimal cost parameter found by 10-fold cross-validation.

Model	Variables	Optimal Cost	Training Accuracy (%)	Testing Accuracy (%)	Run-time (secs)
svm_3	. -prevalentHyp-cigsPerDay -totChol-education	1	84.64	84.59	110
svm_4	age+sysBP+diaBP +BMI+totChol	1	84.51	84.76	100

Table 2.3: *Classification Accuracy for svm_3/4.*

The next five models were built using the `e1071` using the `svm()` and `tune.svm()` for both linear and radial kernels. A variety of parameters and variables used to build the models.

Table 2.4 summarises the respective variables, parameters and classification accuracies for each model. The optimal values for the cost and gamma parameters are found by conducting 10-fold cross-validation. The combination of parameters resulting in the best classification accuracy on the test set are selected by the algorithm.

Model	**Type**	**Variables**	**Cost**	**Gamma**	**Training Accuracy (%)**	**Testing Accuracy (%)**	**Run-time (secs)**
svm_5	Linear - Untuned	all	1	0.1	83.78	87.35	1
svm_6	Linear - Tuned	all	1e-7	1e-7	83.78	87.35	50
svm_7	Radial - Untuned	all	1	0.1	85.15	87.69	1
svm_8	Radial - Tuned	all	10	0.001	83.91	87.69	130
svm_9	Radial - Tuned	age + sysBP	1e-3	1e-7	83.78	87.35	60

Table 2.4: *Classification Accuracy for svm_5/6/7/8/9.*

Chapter 3

Neural Networks

3.1 Methodology

Artificial neural networks are based on the structure and function of the human brain. A neural network is a computing system comprised of simple, highly interconnected nodes, referred to as ‘neurons’ or ‘perceptrons’. The perceptrons are organized in layers which process information using dynamic state responses to external inputs.

A perceptron takes inputs x_1, x_2, … , x_j and produces an output. Each input is assigned a weight w_1, w_2, … , w_j which expresses the importance of the respective inputs to the outputs. The neuron’s output is determined by whether the value of the weighted sum is greater than the threshold value used. A binary classification problem with 0 and 1 as outputs is formulated as follows:

$$\text{output} = \begin{cases} 0 & \text{if } \sum_j w_j x_j \leq \text{ threshold} \\ 1 & \text{if } \sum_j w_j x_j > \text{ threshold} \end{cases}$$

Perceptrons can be designed in multiple layers that make up a neural network (Figure 3.1). The neurons in the input layer relate to the variables used to train the model. In the Framingham dataset, 15 variables are used to predict the binary classification of 10-year CHD risk. The input layer therefore has 15 neurons, and the output has two (0-No & 1-Yes).

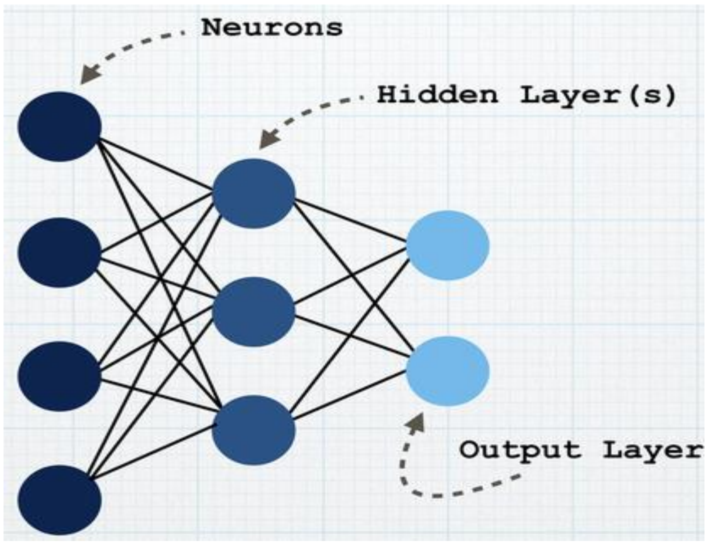


Figure 3.1: *Standard neural network model.*

The neural network models were built using the `neuralnet` package in R. The weights are initialised with random values drawn from a standard normal distribution. During the training of the model, the neural network calculates the output for given inputs and weights. The iterative process stops when the partial derivatives of the error function with respect to the weights are smaller than a given threshold. This is implemented using the back-propagation algorithm.

The back-propagation algorithm (Figure 3.2) adjusts weights in the neural network in order to find the local minimum of the error function. The gradient descent method calculates the partial derivative of the error function with respect to the weight: *∂E/∂w.* If the partial derivative is negative, the weight is increased (left pane), while if the partial derivative is positive, the weight is decreased (right pane). The weights are adjusted until the error function is minimized.

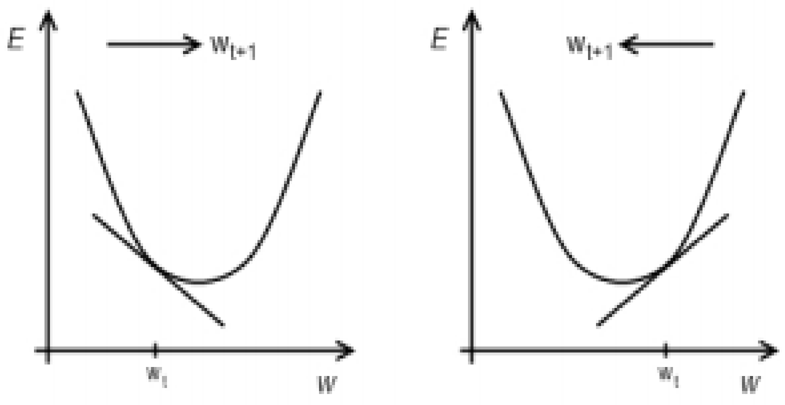


Figure 3.2: *Backpropagation for a univariate error function E(w).*

3.2 Results

The models built in the `neuralnet` package used the same error and activation functions. Cross-entropy serves as the error function, which predicts each probability and compares it to the to the actual class output value (0 or 1). A score is calculated that penalizes the probability based on the distance from the expected value. The cross-entropy loss is minimised, hence smaller values represent more accurate models. The logistic activation function is used throughout the models. The logistic activation function creates complex mappings between the network's inputs and outputs. This ensures the output will always be in the range (0,1).

Table 3.1 summarises structure, predictive accuracy and efficiency of the four neural network models built. The models were trained and tested on the 80/20 train/test split as per requirements. Model performance is evaluated on testing accuracy and run-time metrics.

Model	**Variables**	**# Hidden Layers**	**# Neurons**	**Training Accuracy (%)**	**Testing Accuracy (%)**	**Run-time (secs)**
nn_1	all	1	2	85.66	87.86	55
nn_2	all	1	5	86.38	85.81	125
nn_3	all	1	10	89.07	82.05	270
nn_4	age + sysBP + diaBP + diabetes	2	2,2	84.04	87.69	150

Table 3.1: *Summary of all four neural network models.*

The nn_1 model outperformed the other three models on both fronts. Using 1 hidden layer and 2 neurons in the hidden layer (Figure 3.3), the model achieved an accuracy of 87.86% on the test data. This is slightly better than the other models, however the fast run-time sets it apart from the rest. Model nn_1 is therefore the best performing neural network model (Table 3.2).

Interestingly, in the single hidden layer models, nn_1 (2 neurons) out performed nn_3 (10 neurons) with respect to testing accuracy. However, nn_3 was better at classifying the training data. This would lead one to believe that using more neurons tends to overfit the test data, leading to high variance.

In building nn_4, variables were excluded based on the variable importance plot in §2.2.

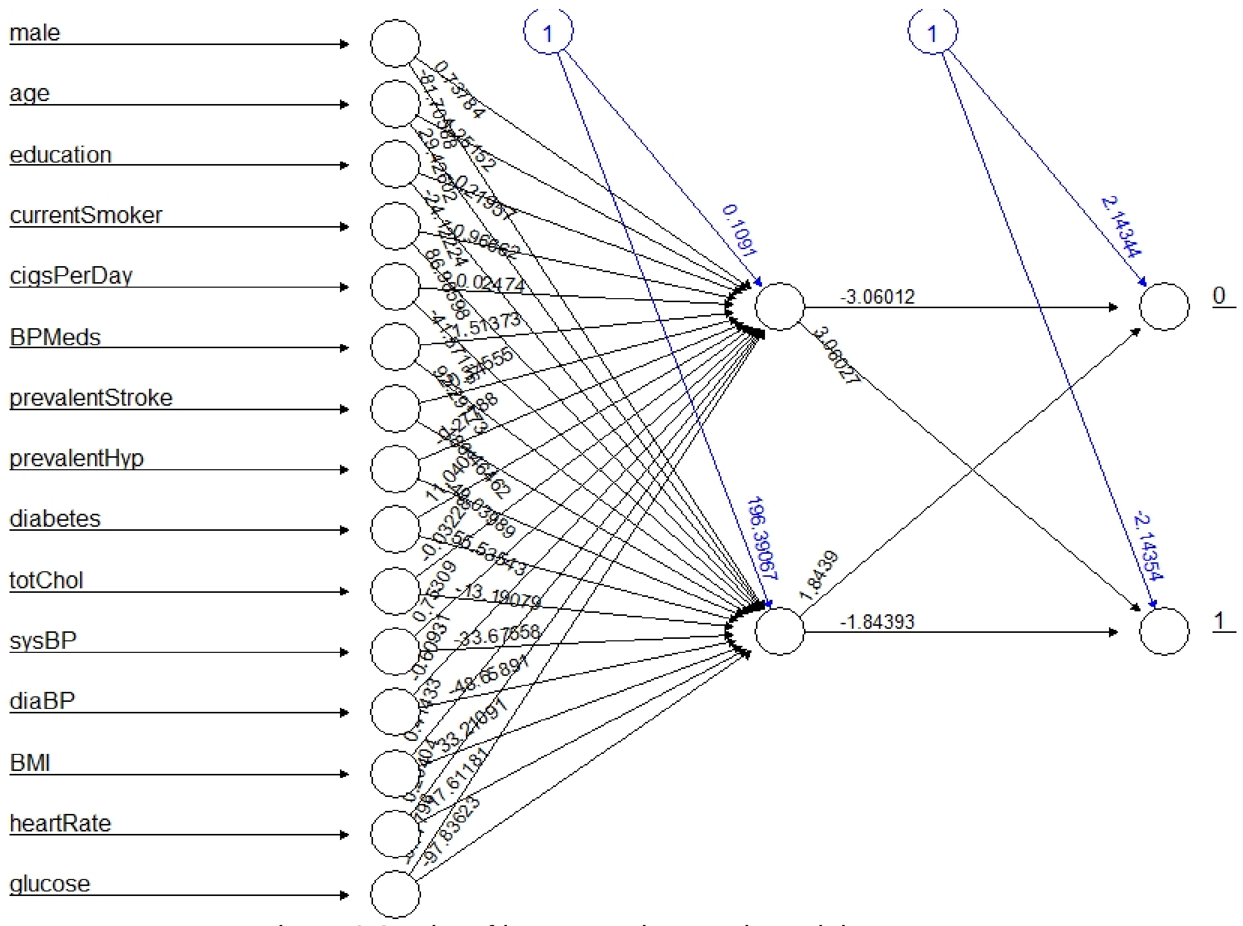


Figure 3.3: *Plot of best neural network model: nn_1.*

Chapter 4

Conclusion

This report evaluated the performance of 13 supervised learning models using support vector machines and neural networks. The deciding factors for best performers are classification accuracy on the test set and time taken to build the model.

Out of the SVM models, svm_7 and svm_8 had the same testing accuracy, however svm_7 was much more efficient in run-time.

Model	**Type**	**Variables**	**Cost**	**Gamma**	**Training Accuracy (%)**	**Testing Accuracy (%)**	**Run-time (secs)**
svm_7	Radial – Untuned	all	1	0.1	85.15	87.69	1

Table 4.1*: Results for svm_7.*

Out of the NN models, nn_1 had the best testing classification accuracy, closely followed by nn_4. The nn_1 model however was far more efficient in run time.

Model	**Variables**	**# Hidden Layers**	**# Neurons**	**Training Accuracy (%)**	**Testing Accuracy (%)**	**Run-time (secs)**
nn_1	all	1	2	85.66	87.86	55

Table 4.2: *Results for nn_1.*

In Table 4.3, it is evident that the svm_7 model had a propensity to incorrectly classify observations as '1'. It correctly classified a '0' observation 99.6% of the time. However, it incorrectly classified a '0' as '1' 12.3% of the time. Conversely, svm_7 did not once incorrectly classify a '1' as '0'. This is a cause for concern and questions the validity of the model. All the SVM models failed to correctly classify the vast majority of '1' observations. This suggests that an SVM may not be the best supervised learning method to implement in this context.

The nn_1 model had a better classification accuracy, as well as a better distribution of incorrect classifications, unlike the severely skewed incorrect classifications noted in svm_7. Therefore, nn_1 is the best model. It is used to predict the 10-year risk of coronary heart disease on the unseen test data in EVRCHA001.csv.

svm_7		True	
		0	1
Predicted	0	511	0
	1	72	2
nn_1		True	
		0	1
Predicted	0	497	57
	1	14	17

Table 4.3: *Classification accuracy on the test set for svm_ and nn_1.*

References

Brownlee, J., 2020. *Loss And Loss Functions For Training Deep Learning Neural Networks*. [online] Machine Learning Mastery. Available at: <https://machinelearningmastery.com/loss-and-loss-functions-for-training-deep-learning-neural-networks> [Accessed 19 July 2020].

DataCamp Community. 2020. *Tutorials - Online Data Analysis & Interpretation | Datacamp*. [online] Available at: <https://www.datacamp.com/community/tutorials/support-vector-machines-r James Le August 22nd, 2018> [Accessed 13 July 2020].

Dwu, L., 2019. *Machine Learning - Heart Disease Framingham*. [online] Kaggle. Available at: <https://www.kaggle.com/lauriandwu/machine-learning-heart-disease-framingham> [Accessed 16 July 2020].

Eight to Late. 2020. *A Gentle Introduction To Support Vector Machines Using R*. [online] Available at: <https://eight2late.wordpress.com/2017/02/07/a-gentle-introduction-to-support-vector-machines-using-r/> [Accessed 15 July 2020].

Gunther, F. and Fritsch, S., 1993. neuralnet: Training of Neural Networks. *The R Journal*, [online] Vol. 2/1, pp.30-38. Available at: <https://journal.r-project.org/archive/2010/RJ-2010-006/RJ-2010-006.pdf> [Accessed 17 July 2020].

Inzaugarat, E., 2020. *How Do Neural Networks Work? And How Can They Be Used To Help With COVID-19 Pandemic?*. [online] Medium. Available at: <https://becominghuman.ai/how-do-neural-networks-work-and-how-can-they-be-used-to-help-with-covid-19-pandemic-875ca8515f73> [Accessed 19 July 2020].

MissingLink.ai. 2018. *7 Types Of Activation Functions In Neural Networks: How To Choose?*. [online] Available at: <https://missinglink.ai/guides/neural-network-concepts/7-types-neural-network-activation-functions-right/> [Accessed 18 July 2020].

Nicholson, C., 2020. *A Beginner's Guide To Neural Networks And Deep Learning*. [online] Pathmind. Available at: <https://pathmind.com/wiki/neural-network> [Accessed 10 July 2020].

Nielsen, M., 2020. *Neural Networks And Deep Learning*. [online] Neuralnetworksanddeeplearning.com. Available at: <http://neuralnetworksanddeeplearning.com/chap1.html> [Accessed 14 July 2020].

Selvarajah, S., Kaur, G., Haniff, J., Cheong, K., Hiong, T., van der Graaf, Y. and Bots, M., 2014. Comparison of the Framingham Risk Score, SCORE and WHO/ISH cardiovascular risk prediction models in an Asian population. *International Journal of Cardiology*, 176(1), pp.211-218.

Shull, M., 2020. *R - Heart Disease - Logreg Tree Forest SVM Bayes*. [online] Kaggle. Available at: <https://www.kaggle.com/micahshull/r-heart-disease-logreg-tree-forest-svm-bayes> [Accessed 16 July 2020].

A-0: Exploratory Data Analysis

```
rm(list=ls())
setwd("C:/Users/user/Desktop/FinTech 2020/Supervised Learning/Assignment
3")
getwd()

library(e1071)
load("C:/Users/user/Desktop/FinTech 2020/Supervised Learning/
     Assignment 3/framingham.RData")
full <- framingham.assignment
str(full)
head(full)
full$TenYearCHD
summary(full)

# NA values in the TenYearCHD column are assigned to finaltest
x <- which(is.na(full$TenYearCHD))
x
finaltest <- full[x,]
finaltest
row.names(finaltest) <- 1:nrow(finaltest)
finaltest$traintest <- NULL
str(finaltest)

#fulltrain built from all entries with known TenYearCHD column values
fulltrain <- full[-x,]
fulltrain
row.names(fulltrain) <- 1:nrow(fulltrain)
fulltrain$traintest <- NULL
str(fulltrain)

df <- fulltrain
library(tidyverse)
library(ggcorrplot)
library(lattice)
library(psych)
library(DataExplorer)
library(reshape2)
library(car)
library(caret)
library(data.table)
library(e1071)
library(scales)
library(stringr)
library(gridGraphics)
library(gridExtra)
library(cowplot)
library(lmtest)
library(gvlma)
library(RColorBrewer)
library(packHV)
library(caTools)
library(DataExplorer)
library(ggcorrplot)

## Target variable as.factor/as.numeric
# Heart disease as factor with levels "No" & "Yes"

df$TenYearCHD <- as.factor(df$TenYearCHD)
```

```
df <- df %>% mutate(HD = TenYearCHD)
df$TenYearCHD = factor(df$TenYearCHD,
                       labels = c('No', 'Yes'))

# rename columns
df <- df %>% rename("Heart Disease" = TenYearCHD,
                    Gender = male,
                    Age = age,
                    Education = education,
                    Smoker = currentSmoker)

## Data Types

df$Gender = factor(df$Gender,
                   labels = c("Female", "Male"))

# replace missing values with numeric
df$Education <- df$Education %>% replace_na(5)

# 1 = Some High School
# 2 = High School or GED
# 3 = Some College or Vocational School
# 4 = College
# 5 = Unknown

df$Education = factor(df$Education,
                      labels = c("Some High School",
                                 "High School or GED",
                                 "Some College/Vocational School",
                                 "College", "Unknown"))

df$Smoker = factor(df$Smoker, labels = c("Smoker", "Non-Smoker"))

df$BPMeds <- df$BPMeds %>% replace_na(0)
df$BPMeds = factor(df$BPMeds, labels = c("No Medication", "Medication"))

df$prevalentStroke = factor(df$prevalentStroke,
                            labels = c("No", "Yes"))

df$prevalentHyp = factor(df$prevalentHyp,
                         labels = c("No", "Yes"))

df$diabetes = factor(df$diabetes, labels = c("No", "Yes"))

# Age as a categorical data type

#hist(df$Age, col = "grey40")
b = seq(min(df$Age)-3, max(df$Age)+8, 10);
df$A <- cut(df$Age, breaks = b, right = T)
table(df$A)

# Organize dataset with factors first
df <- df %>% select(HD,`Heart Disease`, A, Gender, Education, Smoker,
                    BPMeds, prevalentStroke, everything())

# Missing values
plot_missing(df[,-c(1,3)]) + theme_bw()

# glucose is the only variable with more than 2% missing values
# replace all NA's with the median of glucose
df$glucose = replace_na(df$glucose, median(df$glucose, na.rm = T))
```

```
df$glucose

## Barplots of Categorical Data

# Heart Disease counts
a = ggplot(df, aes(`Heart Disease`, fill = `Heart Disease`,
                   color = `Heart Disease`)) +
  geom_bar(stat = "count", lwd = 2) +
  scale_fill_manual(values=c("grey69", "red")) +
  scale_color_manual(values=c('grey29', "black")) +
  labs(title = "Heart Disease") +  theme_bw(base_size = 18) +
  theme(legend.position="bottom")

# Gender
b = ggplot(df, aes(Gender, fill = `Heart Disease`,
                   color = `Heart Disease`)) +
  geom_bar(stat = "count", position = "dodge", lwd = 2) +
  scale_fill_manual(values=c("grey69", "red")) +
  scale_color_manual(values=c('grey29', "black")) +
  labs(title = "Gender", x = "") +  theme_bw(base_size = 18) +
  theme(legend.position="bottom")

options(repr.plot.width=16, repr.plot.height=7)
plot_grid(a,b, ncol = 2, nrow = 1)

# Age
c = ggplot(df, aes(A, fill = `Heart Disease`,
                   color = `Heart Disease`)) +
  geom_bar(stat = "count", position = "dodge", lwd = 2) +
  scale_fill_manual(values=c("grey69", "red")) +
  scale_color_manual(values=c('grey29', "black")) +
  ggtitle("Age") +  theme(plot.title = element_text(hjust = 0.5))
  theme_bw(base_size = 18) + theme(legend.position="bottom")

options(repr.plot.width=16, repr.plot.height=7)
plot_grid(c, ncol = 1, nrow = 1)

## Boxplots by Age & Heart Disease


# Systolic Blood Pressure
a = ggplot(df, aes(A, sysBP, fill = `Heart Disease`,
                   color = `Heart Disease`)) +
  geom_boxplot(lwd = 2) + labs(title = "Age & Systolic BP",
                               y = "Systolic BP", x = "Age") +
  scale_fill_manual(values=c("grey69", "red")) +
  scale_color_manual(values=c('grey29', "black")) +
  theme_bw(base_size = 18) + theme(legend.position="bottom")

# Diastolic Blood Pressure
b = ggplot(df, aes(A, diaBP, fill = `Heart Disease`,
                   color = `Heart Disease`)) +
  geom_boxplot(lwd = 2) + labs(title = "Age & Diastolic BP",
                               y = "Diastolic BP", x = "Age") +
  scale_fill_manual(values=c("grey69", "red")) +
  scale_color_manual(values=c('grey29', "black")) +
  theme_bw(base_size = 18) + theme(legend.position="bottom")

options(repr.plot.width=16, repr.plot.height=7)
plot_grid(a,b, ncol = 2, nrow = 1)
```

```
# Age by Total Cholesterol
a = ggplot(df, aes(A, totChol , fill = `Heart Disease`,
                   color = `Heart Disease`)) +
  geom_boxplot(lwd = 2) + labs(title = "Age & Total Cholesterol",
                               y = "Total Cholesterol", x = "Age") +
  scale_fill_manual(values=c("grey69", "red")) +
  scale_color_manual(values=c('grey29', "black")) +
  theme_bw(base_size = 18) + theme(legend.position="bottom")

# Age by BMI
b = ggplot(df, aes(A, BMI, fill = `Heart Disease`,
                   color = `Heart Disease`)) +
  geom_boxplot(lwd = 2) + labs(title = "Age & Body Mass Index",
                               y = "BMI", x = "Age") +
  scale_fill_manual(values=c("grey69", "red")) +
  scale_color_manual(values=c('grey29', "black")) +
  theme_bw(base_size = 18) + theme(legend.position="bottom")

options(repr.plot.width=16, repr.plot.height=7)
plot_grid(a,b, ncol = 2, nrow = 1)

## Visualize the Data Distributions/Interactions

# Age & Heart Disease
a = ggplot(df,aes(Age, fill = `Heart Disease`,
                  color = `Heart Disease`)) +
  geom_density(lwd = 3, show.legend = T, alpha = 0.7) +
  labs(title = "Age & Hypertension") + facet_grid(~prevalentHyp) +
  scale_fill_manual(values=c("grey69", "red")) +
  scale_color_manual(values=c('grey29', "black")) +
  theme_bw(base_size = 18) + theme(legend.position="bottom") +
  theme(strip.background = element_rect(fill="snow1")) +
  theme(strip.text = element_text(face = "bold", size = 10)) +
  geom_vline(xintercept = mean(df$Age), lwd = 1,
             color = "black", linetype = 4)

options(repr.plot.width = 16, repr.plot.height = 7)
plot_grid(a, ncol = 1, nrow = 1)

## skewness of distributions
par(mfrow=c(2,4))
a = hist(df$Age, main = "Age", col = "#FF99FF");
b = hist(df$cigsPerDay, main = "Cigs Per Day", col = "#FF33FF");
c = hist(df$totChol, main = "Total Cholesterol", col = "#FF00FF");
d = hist(df$glucose, main = "Glucose", col = "#CC33CC");
a = hist(df$sysBP, main = "sysBP", col = "#CC33CC");
b = hist(df$diaBP, main = "diaBP",col = "#6600FF");
c = hist(df$BMI, main = "BMI", col = "#0000FF");
d = hist(df$heartRate, main = "heartRate", col = "#3333FF");

skewness(na.omit(df$cigsPerDay))
skewness(na.omit(df$totChol))
skewness(na.omit(df$sysBP))
skewness(na.omit(df$diaBP))
skewness(na.omit(df$BMI))
skewness(na.omit(df$heartRate))
skewness(na.omit(df$glucose)) #requires evaluating

## removing outliers

# Glucose
# 50% of the data falls between the quantiles
Q <- quantile(df$glucose, probs=c(.25, .75), na.rm = T);
iqr <- IQR(df$glucose, na.rm = T);

# as per Tukey's test
# remove the outlier beyond 1.5 * iqr for Glucose
df2 <- df %>% filter(glucose > (Q[1] - 1.5*iqr) &
                       glucose < (Q[2] + 1.5*iqr))

# visualize the new dataset without outliers

par(mfrow=c(2,1))
options(repr.plot.width=10, repr.plot.height=10)
boxplot(df$glucose, col = "grey40", horizontal = T,
        main = "Glucose - Before")
boxplot(df2$glucose, col = "#33FF33", horizontal = T,
        main = "Glucose - After")

# Cholesterol
# 50% of the data falls between the quantiles
Q <- quantile(df2$totChol, probs=c(.25, .75), na.rm = T);
iqr <- IQR(df2$totChol, na.rm = T);

# as per Tukey's test
# remove the outlier beyond 1.5 * iqr for Glucose
df2 <- df2 %>% filter(totChol > (Q[1] - 1.5*iqr) &
                        totChol < (Q[2] + 1.5*iqr))

# visualize the new dataset without outliers
par(mfrow=c(2,1))
boxplot(df$totChol, col = "grey40", horizontal = T,
        main = "Cholesterol - Before ")
boxplot(df2$totChol, col = "#33FF33", horizontal = T,
        main = "Cholesterol - After ")
par(mfrow=c(1,1))

## Correlations (outliers removed)

c2 <- df2 %>% select(everything(), - `Heart Disease`)
#c2 %>% select_if(is.factor) %>% names()
cols = c("HD",
         "Gender",
         "Education",
         "Smoker",
         "BPMeds",
         "prevalentStroke",
         "prevalentHyp",
         "diabetes",
         "A")
c2[, cols] <- c2 %>% select(all_of(cols)) %>% lapply(as.numeric)

cor <- cor(c2)
cols = c("HD",
         "Gender",
         "Age",
         "Education",
         "Smoker",
         "cigsPerDay",
         "BPMeds",
         "prevalentStroke",
         "prevalentHyp",
         "diabetes",
         "totChol",
         "sysBP",
         "diaBP",
         "BMI",
         "heartRate",
         "glucose",
         )

cor <- as_tibble(reshape2::melt(cor, id = cols))
colnames(cor) <- c("Target", "Variable", "Correlation")

C <- cor[which(cor$Target == "HD"),]
C <- C[order(- abs(C$Correlation)), ]
C <- subset(C, abs(C$Correlation) > 0.10)

# Select & Order by correlation
cr2 <- c2 %>% select("HD",
                     "Age",
                     "sysBP",
                     "prevalentHyp",
                     "diaBP")

##  Correlation Plot & Pairs Panel

par(mfrow=c(1,1))
col <- colorRampPalette(c("black", "white", "magenta"))(20)
heatmap(x = corr, col = col, symm = TRUE, margins = c(10, 10))
legend(x = "top", legend = c("High Corr", "Low Corr"),
       fill = colorRampPalette(c("magenta", "black"))(2))

library(ggcorrplot)
corr <- round(cor(cr2, use="complete.obs"), 2)
ggcorrplot(corr, lab = TRUE, colors =
             c("magenta", "white", "magenta"),
           show.legend = F, outline.color = "gray",
           type = "upper", #hc.order = T,
           tl.cex = 20, lab_size = 8, sig.level = .2) +
  labs(fill = "Correlation")
```

A-1: Support Vector Machines

```
rm(list=ls())
setwd("C:/Users/user/Desktop/FinTech 2020/Supervised Learning/Assignment 
3")
getwd()

# kernlab does not require data standardisation
# before building SVM models 1-4
# Data only standardised for models 5-9 (e1071)

library(kernlab)
load("C:/Users/user/Desktop/FinTech 2020/Supervised Learning/
     Assignment 3/framingham.RData")
full <- framingham.assignment
str(full)
head(full)
full$TenYearCHD
summary(full)

x <- which(is.na(full$TenYearCHD))
x
finaltest <- full[x,]
finaltest
row.names(finaltest) <- 1:nrow(finaltest)
finaltest$traintest <- NULL
str(finaltest)

fulltrain <- full[-x,]
fulltrain
row.names(fulltrain) <- 1:nrow(fulltrain)
fulltrain$traintest <- NULL
str(fulltrain)

#set 0 and 1 to female and male
for (i in 1:nrow(fulltrain)) {
  if (fulltrain$male[i] == 1) {
    fulltrain$male[i] <- "Male"
  } else {
    fulltrain$male[i] <- "Female"
  }
}
fulltrain[,1]
fulltrain

#change column name: Male <- Gender
colnames(fulltrain)[1] <- "Gender"
colnames(fulltrain)

#set 0 and 1 to no and yes
for (i in 1:nrow(fulltrain)) {
  if (fulltrain$TenYearCHD[i] == 0) {
    fulltrain$TenYearCHD[i] <- "No"
  } else {
    fulltrain$TenYearCHD[i] <- "Yes"
  }
}
fulltrain[,16]
fulltrain

#change column name: TenYearCHD <- RiskOfCHD
colnames(fulltrain)[16] <- "RiskOfCHD"
colnames(fulltrain)

#set 0 and 1 to female and male
for (i in 1:nrow(finaltest)) {
  if (finaltest$male[i] == 1) {
    finaltest$male[i] <- "Male"
  } else {
    finaltest$male[i] <- "Female"
  }
}
finaltest[,1]
finaltest

#change column name: TenYearCHD <- RiskOfCHD
colnames(fulltrain)[16] <- "RiskOfCHD"
colnames(fulltrain)

#change column name: TenYearCHD <- RiskOfCHD
colnames(finaltest)[16] <- "RiskOfCHD"
colnames(finaltest)

#omit NA's
fulltrain
fulltrain <- na.omit(fulltrain)
row.names(fulltrain) <- 1:nrow(fulltrain)
str(fulltrain) # 2928 observations

finaltest
finaltest$RiskOfCHD <- 0
finaltest$RiskOfCHD
finaltest #since entire column made of NA's, all will be deleted
finaltest <- na.omit(finaltest)
row.names(finaltest) <- 1:nrow(finaltest)
str(finaltest) # 730 observations

#80/20 train/test split on fulltrain
library(caret)
set.seed(23)
samp <- createDataPartition(fulltrain[,"RiskOfCHD"],1,.8)[[1]]
samp

#80% train split (2344 observations)
train = fulltrain[samp,]
train$RiskOfCHD <- as.factor(train$RiskOfCHD)
head(train)
str(train)

#20% test split (584 observationa)
test = fulltrain[-samp,]
test$RiskOfCHD <- as.factor(test$RiskOfCHD)
head(test)
```

```
str(test)

# Helper packages
library(dplyr)    # for data wrangling
library(ggplot2)  # for awesome graphics
library(rsample)  # for data splitting

# Modeling packages
library(caret)    # for classification and regression training
library(kernlab)  # for fitting SVMs

# Model interpretability packages
library(pdp)      # for partial dependence plots, etc.
library(vip)      # for variable importance plots

# Data standardisation not required when using kernlab
# Tune an SVM with radial basis kernel using 10-fold CV
set.seed(23)
# takes approx 60 secs
svm_1 <- train(
  RiskOfCHD ~ .,
  data = train,
  method = "svmRadial",
  preProcess = c("center", "scale"),
  trControl = trainControl(method = "cv", number = 10),
  tuneLength = 10)

svm_1
ggplot(svm_1) + theme_light() +
  geom_vline(xintercept = 2, col = "red", lty = 5) +
  annotate(geom = 'text', label = 'C = 2', x = 2, y = 0.805, col = "red")

svm_1$results
svm_1$bestTune
confusionMatrix(svm_1)

# Predicting the Test set results
y_pred_1 = predict(svm_1, newdata = test)
y_pred_1

# Making the Confusion Matrix
CM_1 = table(test$RiskOfCHD, y_pred_1)
CM_1

# Control params for SVM
ctrl <- trainControl(
  method = "cv",
  number = 10,
  classProbs = TRUE,
  summaryFunction = twoClassSummary  # also needed for AUC/ROC
)

# Tune a Radial SVM determined when AUC maximised
set.seed(23)

svm_2 <- train(
  RiskOfCHD ~ .,
  data = train,
  method = "svmRadial",
  preProcess = c("center", "scale"),
  metric = "ROC",  # area under ROC curve (AUC)
  trControl = ctrl,
  tuneLength = 10)
svm_2

# Print results
svm_2$results
confusionMatrix(svm_2)

# Predicting the Test set results
y_pred_2 = predict(svm_2, newdata = test)
y_pred_2

# Confusion Matrix
CM_2 = table(test$RiskOfCHD, y_pred_2)
CM_2

# create a funtion for vip
prob_yes <- function(object, newdata) {
  predict(object, newdata = newdata, type = "prob")[, "Yes"]
}

# variable importance plot
set.seed(23)  # for reproducibility
vip(svm_2, method = "permute", nsim = 5, train = train,
    target = "RiskOfCHD", metric = "auc", reference_class = "Yes",
    pred_wrapper = prob_yes)

features <- c("age","sysBP","diaBP", "BMI",
              "heartRate","cigsPerDay",
              "totChol", "prevalentHyp")

# create a partial dependance plot function
pdps <- lapply(features, function(x) {
  partial(svm_2, pred.var = x, which.class = 2,
          prob = TRUE, plot = TRUE, plot.engine = "ggplot2") +
    coord_flip()
})
grid.arrange(grobs = pdps,ncol = 2)

# SVM model 3 - Radial excl prevalentHyp, cigsPerDay, education, totChol
set.seed(23)
# takes approx 60 secs
svm_3 <- train(
  RiskOfCHD ~ .-prevalentHyp-cigsPerDay-totChol-education,
  data = train,
  method = "svmRadial",
  preProcess = c("center", "scale"),
  trControl = trainControl(method = "cv", number = 10),
  tuneLength = 10)
svm_3

svm_3$results
svm_3$bestTune
```

```
confusionMatrix(svm_3)

# Predicting the Test set results
y_pred_3 = predict(svm_3, newdata = test)
y_pred_3

# Making the Confusion Matrix
CM_3 = table(test$RiskOfCHD, y_pred_3)
CM_3

# SVM model 4 - Radial with age + sysBP + diaBP + BMI + totChol
set.seed(23)
# takes approx 60 secs
svm_4 <- train(
  RiskOfCHD ~ age + sysBP + diaBP + BMI + totChol,
  data = train,
  method = "svmRadial",
  preProcess = c("center", "scale"),
  trControl = trainControl(method = "cv", number = 10),
  tuneLength = 10)
svm_4

svm_4$results
svm_4$bestTune
confusionMatrix(svm_4)

# Predicting the Test set results
y_pred_4 = predict(svm_4, newdata = test)
y_pred_4

# Making the Confusion Matrix
CM_4 = table(test$RiskOfCHD, y_pred_4)
CM_4

# Reload data and apply data normalisation for svm_5 - svm_9
# e1071 requires data standardisation

library(e1071)
load("C:/Users/user/Desktop/FinTech 2020/Supervised Learning/Assignment
3/framingham.RData")
full <- framingham.assignment
str(full)
head(full)
full$TenYearCHD
summary(full)

x <- which(is.na(full$TenYearCHD))
x
finaltest <- full[x,]
finaltest
row.names(finaltest) <- 1:nrow(finaltest)
finaltest$traintest <- NULL
str(finaltest)

fulltrain <- full[-x,]
fulltrain
row.names(fulltrain) <- 1:nrow(fulltrain)
```

```
fulltrain$traintest <- NULL
str(fulltrain)

fulltrain[c(2,5,10,11,12,13,14,15)] <-
  lapply(fulltrain[c(2,5,10,11,12,13,14,15)],
  function(x) c(scale(x)))
str(fulltrain)

#change column name: TenYearCHD <- RiskOfCHD
colnames(fulltrain)[16] <- "RiskOfCHD"
colnames(fulltrain)

#change column name: TenYearCHD <- RiskOfCHD
colnames(finaltest)[16] <- "RiskOfCHD"
colnames(finaltest)

#omit NA's
fulltrain
fulltrain <- na.omit(fulltrain)
row.names(fulltrain) <- 1:nrow(fulltrain)
str(fulltrain) # 2928 observations

finaltest
finaltest$RiskOfCHD <- 0
finaltest$RiskOfCHD
finaltest #since entire column made of NA's, all will be deleted
finaltest <- na.omit(finaltest)
row.names(finaltest) <- 1:nrow(finaltest)
str(finaltest) # 730 observations

#80/20 train/test split on fulltrain
library(caret)
set.seed(23)
samp <- createDataPartition(fulltrain[,"RiskOfCHD"],1,.8)[[1]]
samp

#80% train split (2343 observations)
train = fulltrain[samp,]
train$RiskOfCHD <- as.factor(train$RiskOfCHD)
row.names(train) <- 1:nrow(train)
head(train)
str(train)

#20% test split (585 observationa)
test = fulltrain[-samp,]
test$RiskOfCHD <- as.factor(test$RiskOfCHD)
row.names(test) <- 1:nrow(test)
head(test)
str(test)

# untuned linear kernal using all variables
svm_5 = svm(as.factor(RiskOfCHD) ~ .,
            data = train,
            type = 'C-classification',
            kernel = "linear",
            scale = FALSE,
            cost = 1, gamma = 0.1)
```

```
svm_5

train_pred_5 = predict(svm_5, train)
mean(train_pred_5 == as.factor(train$RiskOfCHD))

y_pred_5 = predict(svm_5, test)
mean(y_pred_5 == as.factor(test$RiskOfCHD))

# Making the Confusion Matrix
CM_5 = table(test$RiskOfCHD, y_pred_5)
CM_5

# tuned linear kernal using all variables
set.seed(23)
svm_6 = tune.svm(as.factor(RiskOfCHD) ~ .,
                 data = train,
                 type = 'C-classification',
                 kernel = "linear",
                 scale = FALSE,
                 cost = c(1e-7, 1e-3, 1, 10),
                 gamma = c(1e-7, 1e-3, 1))

svm_6$best.performance #lowest error
svm_6$best.model
svm_6$best.parameters

y_pred_6 = predict(svm_6$best.model, test)
mean(y_pred_6 == as.factor(test$RiskOfCHD))

# Making the Confusion Matrix
CM_6 = table(test$RiskOfCHD, y_pred_6)
CM_6

# untuned radial kernal using all variables
svm_7 = svm(as.factor(RiskOfCHD) ~ .,
            data = train,
            type = 'C-classification',
            kernel = "radial",
            scale = FALSE,
            cost = 1, gamma = 0.1)
svm_7

train_pred_7 = predict(svm_7, train)
mean(train_pred_7 == as.factor(train$RiskOfCHD))

y_pred_7 = predict(svm_7, test)
mean(y_pred_7 == as.factor(test$RiskOfCHD))

# Making the Confusion Matrix
CM_7 = table(test$RiskOfCHD, y_pred_7)
CM_7

# tuned radial kernal using all variables
set.seed(23)
svm_8 = tune.svm(as.factor(RiskOfCHD) ~ .,
                 data = train,
                 type = 'C-classification',
                 kernel = "radial",
                 scale = FALSE,
                 cost = c(1e-3, 1, 5, 10, 15),
                 gamma = c(1e-7, 1e-3, 1))
summary(svm_8)
svm_8$best.performance #lowest error
svm_8$best.model
svm_8$best.parameters

train_pred_8 = predict(svm_8$best.model, train)
mean(train_pred_8 == as.factor(train$RiskOfCHD))

y_pred_8 = predict(svm_8$best.model, test)
mean(y_pred_8 == as.factor(test$RiskOfCHD))

# Making the Confusion Matrix
CM_8 = table(test$RiskOfCHD, y_pred_8)
CM_8

# Tune radial kernal using only age + sysBP
set.seed(23)
svm_9 = tune.svm(as.factor(RiskOfCHD) ~ age + sysBP,
                  data = train,
                  type = 'C-classification',
                  kernel = "radial",
                  scale = FALSE,
                  cost = c(1e-3, 1, 5),
                  gamma = c(1e-7, 1e-3, 1))
summary(svm_9)
svm_9$best.performance
svm_9$best.model
svm_9$best.parameters

train_pred_9 = predict(svm_9$best.model, train)
mean(train_pred_9 == as.factor(train$RiskOfCHD))

y_pred_9 = predict(svm_9$best.model, test)
mean(y_pred_9 == as.factor(test$RiskOfCHD))

# Making the Confusion Matrix
CM_9 = table(test$RiskOfCHD, y_pred_9)
CM_9
```

A-2: Neural Networks

```
## neural net package
rm(list=ls())
library(neuralnet)
load("C:/Users/user/Desktop/FinTech 2020/Supervised Learning/Assignment
3/framingham.RData")
full <- framingham.assignment
str(full)
head(full)
full$TenYearCHD
summary(full)

x <- which(is.na(full$TenYearCHD))
x
finaltest <- full[x,]
finaltest
row.names(finaltest) <- 1:nrow(finaltest)
finaltest$traintest <- NULL
str(finaltest)

finaltest[c(2,5,10,11,12,13,14,15)] <-
  lapply(finaltest[c(2,5,10,11,12,13,14,15)],
         function(x) c(scale(x)))
str(finaltest)

fulltrain <- full[-x,]
fulltrain
row.names(fulltrain) <- 1:nrow(fulltrain)
fulltrain$traintest <- NULL
str(fulltrain)

fulltrain[c(2,5,10,11,12,13,14,15)] <-
  lapply(fulltrain[c(2,5,10,11,12,13,14,15)],
         function(x) c(scale(x)))
str(fulltrain)

#change column name: TenYearCHD <- RiskOfCHD
colnames(fulltrain)[16] <- "RiskOfCHD"
colnames(fulltrain)

#change column name: TenYearCHD <- RiskOfCHD
colnames(finaltest)[16] <- "RiskOfCHD"
colnames(finaltest)

#omit NA's
fulltrain
fulltrain <- na.omit(fulltrain)
row.names(fulltrain) <- 1:nrow(fulltrain)
str(fulltrain) # 2928 observations

finaltest
finaltest$RiskOfCHD <- 0
finaltest$RiskOfCHD
finaltest #since entire column made of NA's, all will be deleted
finaltest <- na.omit(finaltest)
```

```
row.names(finaltest) <- 1:nrow(finaltest)
str(finaltest) # 730 observations

#80/20 train/test split on fulltrain
library(caret)
set.seed(23)
samp <- createDataPartition(fulltrain[,"RiskOfCHD"],1,.8)[[1]]
samp

#80% train split (2343 observations)
train = fulltrain[samp,]
train$RiskOfCHD <- as.factor(train$RiskOfCHD)
row.names(train) <- 1:nrow(train)
head(train)
str(train)

#20% test split (585 observationa)
test = fulltrain[-samp,]
test$RiskOfCHD <- as.factor(test$RiskOfCHD)
row.names(test) <- 1:nrow(test)
head(test)
str(test)

## Start of NN code

## model 1 - hidden = c(2)
set.seed(23)
neuralnetmodel_1 = neuralnet(RiskOfCHD ~ .,
                           data = train,
                           linear.output = FALSE,
                           act.fct = "logistic",
                           hidden = c(2),
                           err.fct = "ce")
plot(neuralnetmodel_1)
neuralnetmodel_1$weights
neuralnetmodel_1$err.fct
neuralnetmodel_1$act.fct

train.predict_1 = predict(neuralnetmodel_1, train)
head(round(train.predict_1,3), 6)

maxprobability <- apply(train.predict_1, 1, which.max)
maxprobability

train.predict.class_1 <- c(0, 1)[maxprobability]
head(train.predict.class_1)
train.predict.class_1

table(train.predict.class_1, train$RiskOfCHD)
mean(train.predict.class_1 == train$RiskOfCHD)*100

# classification accuracy on the test set

test.predict_1 = predict(neuralnetmodel_1, test)
head(test.predict_1)

maxprobability <- apply(test.predict_1, 1, which.max)
```

```
test.predict.class_1 <- c(0,1)[maxprobability]
head(test.predict.class_1)
table(test.predict.class_1, test$RiskOfCHD)
mean(test.predict.class_1 == test$RiskOfCHD)*100

# final prediction on unseen test set (730 observations)
# (best model to be written to .csv)

test.predict_1_csv = predict(neuralnetmodel_1, finaltest)
head(test.predict_1_csv)

maxprobability <- apply(test.predict_1_csv, 1, which.max)
test.predict.class_1_csv <- c(0,1)[maxprobability]
head(test.predict.class_1_csv)
test.predict.class_1_csv

# write to .csv with student number
write.csv(test.predict.class_1_csv,
          "C:/Users/user/Desktop/FinTech 2020/Supervised
Learning/Assignment 3\\EVRCHA001.csv",
          row.names = TRUE)

## model 2 - hidden = c(5)
set.seed(23)
neuralnetmodel_2 = neuralnet(RiskOfCHD ~ .,
                             data = train,
                             linear.output = FALSE,
                             act.fct = "logistic",
                             hidden = c(5),
                             err.fct = "ce",
                             stepmax = 10^6)
plot(neuralnetmodel_2)
neuralnetmodel_2$weights

train.predict_2 = predict(neuralnetmodel_2, train)
head(round(train.predict_2,3), 6)

maxprobability <- apply(train.predict_2, 1, which.max)
maxprobability

train.predict.class_2 <- c(0, 1)[maxprobability]
head(train.predict.class_2)
train.predict.class_2

table(train.predict.class_2, train$RiskOfCHD)
mean(train.predict.class_2 == train$RiskOfCHD)*100

# classification accuracy on the test set

test.predict_2 = predict(neuralnetmodel_2, test)
head(test.predict_2)

maxprobability <- apply(test.predict_2, 1, which.max)
test.predict.class_2 <- c(0,1)[maxprobability]
head(test.predict.class_2)
table(test.predict.class_2, test$RiskOfCHD)
mean(test.predict.class_2 == test$RiskOfCHD)*100
```

```
## model 3 - hidden = c(10)
set.seed(23)
neuralnetmodel_3 = neuralnet(RiskOfCHD ~ .,
                             data = train,
                             linear.output = FALSE,
                             act.fct = "logistic",
                             hidden = c(10),
                             err.fct = "ce")
plot(neuralnetmodel_3)
neuralnetmodel_3$weights

train.predict_3 = predict(neuralnetmodel_3, train)
head(round(train.predict_3,3), 6)

maxprobability <- apply(train.predict_3, 1, which.max)
maxprobability

train.predict.class_3 <- c(0, 1)[maxprobability]
head(train.predict.class_3)
train.predict.class_3

table(train.predict.class_3, train$RiskOfCHD)
mean(train.predict.class_3 == train$RiskOfCHD)*100

# classification accuracy on the test set

test.predict_3 = predict(neuralnetmodel_3, test)
head(test.predict_3)

maxprobability <- apply(test.predict_3, 1, which.max)
test.predict.class_3 <- c(0,1)[maxprobability]
head(test.predict.class_3)
table(test.predict.class_3, test$RiskOfCHD)
mean(test.predict.class_3 == test$RiskOfCHD)*100

## model 4 - hidden = c(2,2)
set.seed(23)
neuralnetmodel_4 = neuralnet(RiskOfCHD ~ age + sysBP + diaBP + diabetes,
                             data = train,
                             linear.output = FALSE,
                             act.fct = "logistic",
                             hidden = c(2,2),
                             err.fct = "ce",
                             stepmax = 10^5)
plot(neuralnetmodel_4)
neuralnetmodel_4$weights

train.predict_4 = predict(neuralnetmodel_4, train)
head(round(train.predict_4,3), 6)

maxprobability <- apply(train.predict_4, 1, which.max)
maxprobability

train.predict.class_4 <- c(0, 1)[maxprobability]
head(train.predict.class_4)
train.predict.class_4
```

```
table(train.predict.class_4, train$RiskOfCHD)
mean(train.predict.class_4 == train$RiskOfCHD)*100

# classification accuracy on the test set

test.predict_4 = predict(neuralnetmodel_4, test)
head(test.predict_4)

maxprobability <- apply(test.predict_4, 1, which.max)
test.predict.class_4 <- c(0,1)[maxprobability]
head(test.predict.class_4)
table(test.predict.class_4, test$RiskOfCHD)
mean(test.predict.class_4 == test$RiskOfCHD)*100
```