Student Name: Shashank B V

Student No: R00224414

For the module Data8001 as part of the

Master of Science in Data Science and Analytics, Department of Mathematics, 2022/23

Declaration of Authorship

I, Shashank B V, declare that the work submitted is my own.

- I declare that I have not obtained unfair assistance via use of the internet or a third party in the completion of this examination, including chatbots such as ChatGPT.
- I acknowledge that the Academic Department reserves the right to request me to present for oral examination as part of the assessment regime for this module.
- I confirm that I have read and understood the policy and procedures concerning academic honesty, plagiarism and infringements.
- I understand that where breaches of this declaration are detected, these will be reviewed under MTU (Cork) policy and procedures concerning academic honesty, plagiarism and infringements, as well as any other University regulations and policies which may apply to the case. I also understand that any breach of academic honesty is a serious issue and may incur penalties.
- EXAMINATION/ASSESSMENT MATERIAL MAY, AT THE DISCRETION OF THE INTERNAL EXAMINER, BE SUBMITTED TO THE UNIVERSITY'S PLAGIARISM DETECTION SOLUTION
- Where I have consulted the published work of others, this is always clearly attributed
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this work is entirely my own work
- I have acknowledged all main sources of help

Signed: shashank b v

Date: ____________

QUESTION – 1

Impact of GDPR on Digital Marketing

The General Data Protection Regulation (GDPR) has catalyzed a seismic shift in the realm of digital marketing since its 2018 inception. This regulation predominantly emphasizes explicit consent protocols for data utilization.

Opt-In Conundrums and Transparent Machinations

In the pre-GDPR era, the acquisition and exploitation of user data occurred with minimal oversight, often transcending the boundaries of informed consent. GDPR, however, mandates unequivocal and transparent assent from users prior to data aggregation. This requisite has engendered the ubiquity of consent mechanisms, such as cookies banners and opt-in dialogues, elaborating the scope and purpose of data collection.

Personalization Versus Data Protection

Data-driven personalization—a sine qua non in contemporary digital marketing—requires the voluminous collection of user-centric information. GDPR, however, compels marketers to reevaluate the magnitude of data they accrue, thereby inducing a potential diminution in personalization efficacy. This necessitates a judicious equilibrium between bespoke user engagement and regulatory adherence.

Data Economization and Its Concomitant Benefits

Another quintessential aspect of GDPR is the principle of data minimization; the mandate to restrict data collection to the bare essentials for the intended objective. Contrary to seeming restrictive, this edict enhances the quality of the accumulated data, rendering it more actionable and pertinent for marketers.

Reconfigurations in Digital Marketing Infrastructure

GDPR's stipulations have engendered modifications in ubiquitous marketing tools like Google Analytics. These alterations encompass functionalities that facilitate user data anonymization and expungement, thereby aligning these platforms with regulatory norms.

Escalated Accountability and Fiduciary Assurance

With the specter of punitive sanctions looming large for non-compliance, accountability has escalated manifold. This, paradoxically, augments consumer trust. Users demonstrate an increased willingness to divulge personal information, contingent upon the assurance of judicious and secure data stewardship.

The inception of GDPR has precipitated both challenges and opportunities in the digital marketing milieu. Marketers must now navigate an intricate labyrinth of ethical and regulatory stipulations, which, though ostensibly onerous, promise to engender a digital marketing paradigm that is both efficacious and ethical.

QUESTION – 2

In the milieu of data analytics education, the proliferation of AI conversational agents like ChatGPT merits cautious circumspection. Far from being a panacea, this technology could engender a cavalcade of pedagogical shortcomings.

Primarily, the expediency furnished by ChatGPT risks fostering epistemological lethargy. Rather than catalyzing Socratic dialogues or encouraging a nuanced foray into abstruse theories, the tool offers facile solutions. This is antithetical to pedagogical frameworks that posit struggle and active cognition as sine qua non for indelible learning.

Secondarily, ChatGPT's ontological constraints present an axiological quandary. As a generalist automaton, it lacks the domain-specific hermeneutics imperative for teaching the labyrinthine nuances of data analytics. Absent this specialized acumen, reliance upon such platforms could engender heuristic errors, an inexcusable transgression in a discipline where analytical veracity is non-negotiable. Numerous scholarly treatises underscore the indispensability of subject-matter expertise in pedagogy, a criterion in which ChatGPT is deficient.

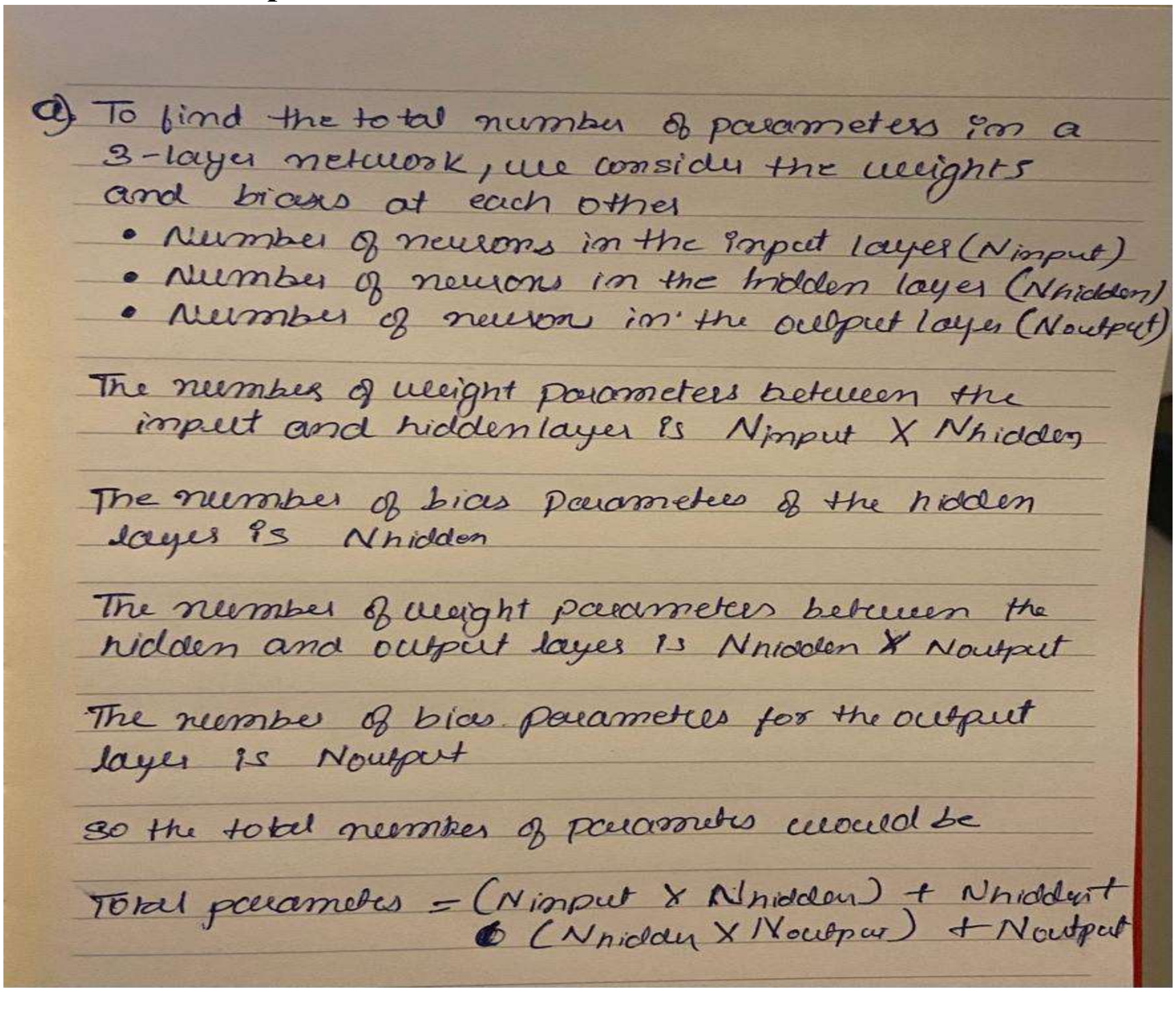
Thirdly, the absence of institutional oversight and scholastic rigor inherent in traditional learning ecosystems may catalyze epistemic malpractice when using ChatGPT. There's an ethical imperative to subject analytical methodologies to rigorous peer-review and scrupulous pedagogical oversight. Evading this systematized scrutiny jeopardizes the ontological integrity of analytics as a scholarly endeavor.

Lastly, the textual constraints of ChatGPT vitiate the learning process by obviating experiential immersion. Given the tactile nature of data analytics, which necessitates hands-on manipulation of databases and coding paradigms, a text-based pedagogical model is manifestly insufficient. Theories of experiential learning ennoble the corporeal act of 'doing' as a quintessential facet of skill acquisition—an aspect conspicuously absent in ChatGPT's educational arsenal.

To encapsulate, while ChatGPT's utilitarian virtues may seem tantalizing, they are anathema to the tenets of rigorous, in-depth pedagogy in the domain of data analytics. The platform presents an intellectual paradox: while ostensibly facilitating accessibility, it concomitantly engenders substantive deficits in educational quality.

QUESTION – 3

a) Total number of parameters



a) To find the total number of parameters in a 3-layer network, we consider the weights and biass at each other

- Number of neurons in the input layer (Ninput)
- Number of neurons in the hidden layer (Nhidden)
- Number of neuron in the output layer (Noutput)

The numbes of weight parameters between the input and hiddenlayer is Ninput X Nhidden

The number of bias parameters of the hidden layer is Nhidden

The number of weight parameters between the hidden and output layes is Nhidden X Noutput

The number of bias parameters for the output layer is Noutput

So the total number of parameters would be

Total parameters = (Ninput X Nhidden) + Nhidden + (Nhidden X Noutput) + Noutput

The total number of parameters in the network includes both weights and biases.

Parameters between input layer and hidden layer:

Number of weights = (Number of neurons in input layer) * (Number of neurons in hidden layer) = $5\times7=35$

Number of biases = Number of neurons in hidden layer = 7

Total = $35+7=42$

Parameters between hidden layer and output layer:

Number of weights = (Number of neurons in hidden layer) * (Number of neurons in output layer) = $7\times3=21$

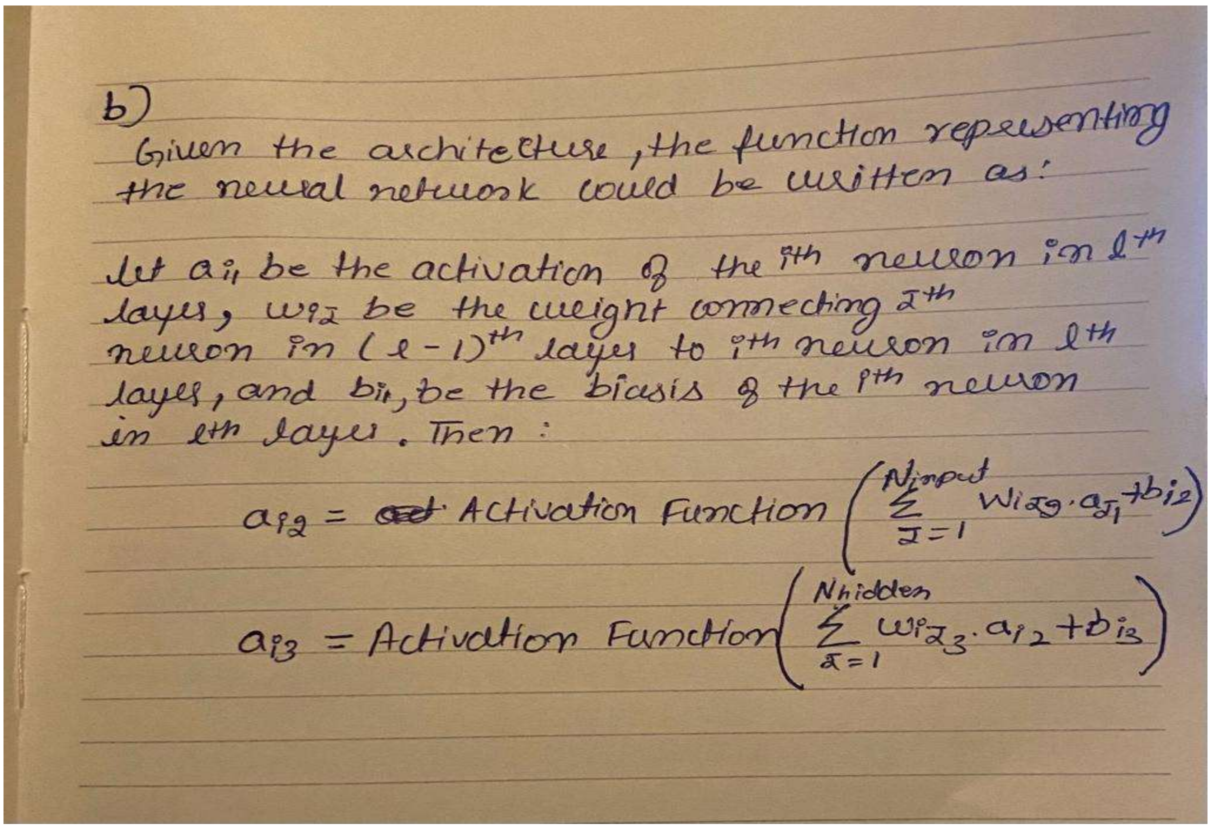
Number of biases = Number of neurons in output layer = 3

Total = $21+3=24$

Total number of parameters: $42+24=66$

So, there are a total of 66 parameters in this 3-layer neural network.

b)



b)

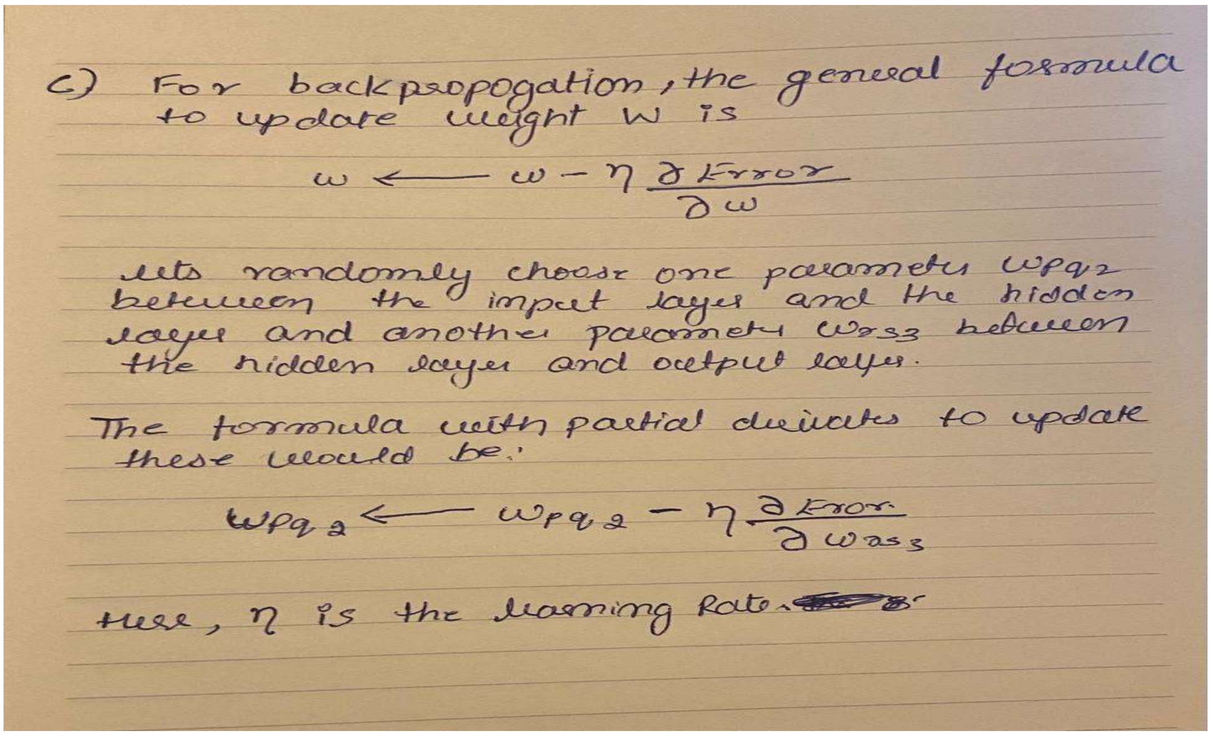
Given the architecture, the function representing the neural network could be written as:

Let a_{il} be the activation of the ith neuron in lth layer, w_{iI} be the weight connecting Ith neuron in $(l-1)$th layer to ith neuron in lth layer, and b_{il} be the biasis of the ith neuron in lth layer. Then:

$$a_{i2} = \text{~~act~~ Activation Function}\left(\sum_{I=1}^{N_{input}} w_{iI2} \cdot a_{I1} + b_{i2}\right)$$

$$a_{i3} = \text{Activation Function}\left(\sum_{I=1}^{N_{hidden}} w_{iI3} \cdot a_{i2} + b_{i3}\right)$$

c)



c) For backpropogation, the general formula to update weight W is

$$\omega \longleftarrow \omega - \eta \frac{\partial \text{Error}}{\partial \omega}$$

lets randomly choose one parameter ω_{pq2} between the input layer and the hidden layer and another parameter ω_{rs3} between the hidden layer and output layer.

The formula with partial derivates to update these would be:

$$\omega_{pq2} \longleftarrow \omega_{pq2} - \eta \frac{\partial \text{Error}}{\partial \omega_{rs3}}$$

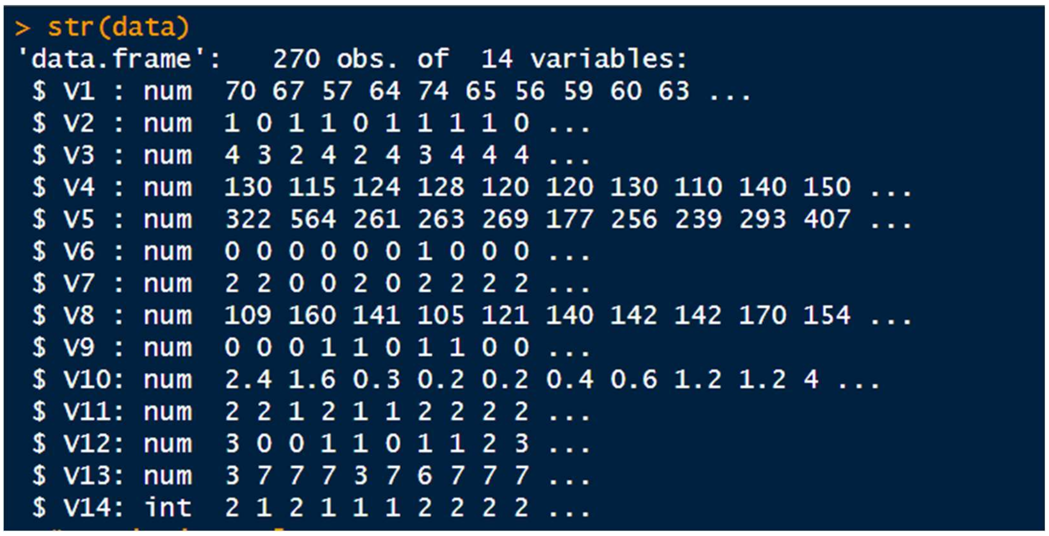
Here, η is the learning Rate.

QUESTION – 4

The main aim of the statlog heart dataset is to analyse and which is a medical dataset which focuses on various factors related to heart disease in individuals.

The dataset contains 270 observations and 14 variables which contains age, sex, chest pain type, resting blood pressure, serum cholesterol levels, and other important medical indicators. The main objective is to build a machine learning model that predicts the heart disease based on these features.

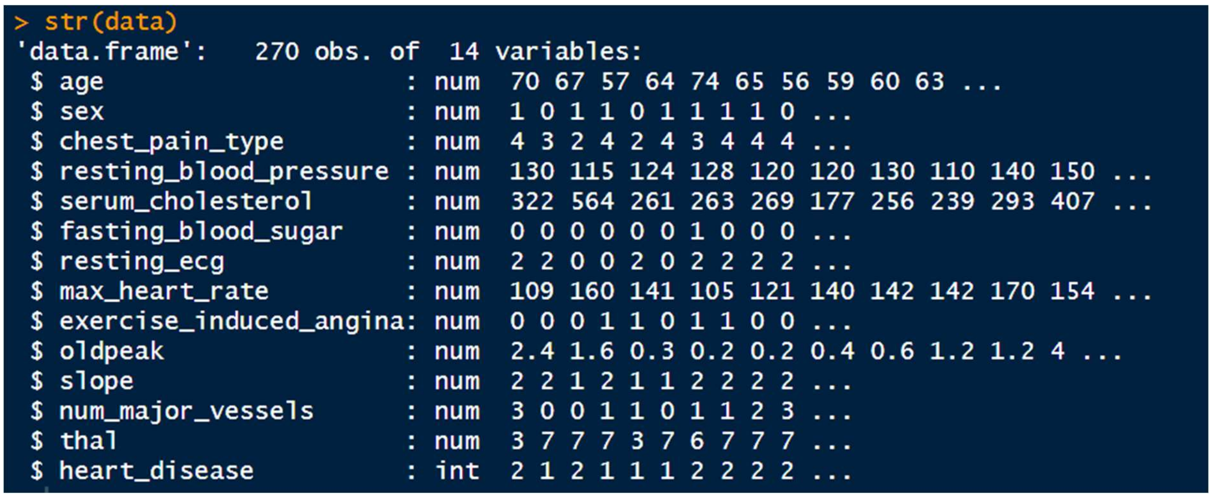
The dataset names “heart.dat” was read in r using read.table, initially data was not having the headers as we can see in the below figure.



```
> str(data)
'data.frame':	270 obs. of  14 variables:
 $ V1 : num  70 67 57 64 74 65 56 59 60 63 ...
 $ V2 : num  1 0 1 1 0 1 1 1 1 0 ...
 $ V3 : num  4 3 2 4 2 4 3 4 4 4 ...
 $ V4 : num  130 115 124 128 120 120 130 110 140 150 ...
 $ V5 : num  322 564 261 263 269 177 256 239 293 407 ...
 $ V6 : num  0 0 0 0 0 1 0 0 0 ...
 $ V7 : num  2 2 0 0 2 0 2 2 2 2 ...
 $ V8 : num  109 160 141 105 121 140 142 142 170 154 ...
 $ V9 : num  0 0 0 1 1 0 1 1 0 0 ...
 $ V10: num  2.4 1.6 0.3 0.2 0.2 0.4 0.6 1.2 1.2 4 ...
 $ V11: num  2 2 1 2 1 1 2 2 2 2 ...
 $ V12: num  3 0 0 1 1 0 1 1 2 3 ...
 $ V13: num  3 7 7 7 3 7 6 7 7 7 ...
 $ V14: int  2 1 2 1 1 1 2 2 2 2 ...
```

and then the data is copied into another variable called data1 to retain the original data because it’s easy to start again from the starting point. The dataset has 270 observations and 14 variables. Since the data doesn’t have column names, each variable is named according to the attribute it represents such as "age," "sex," "chest_pain_type,""resting_blood_pressure","serum_cholesterol","fasting_blood_sugar","resting_ecg","max_heart_rate", "exercise_induced_angina", "oldpeak","slope", "num_major_vessels", "thal", "heart_disease".

After renaming the column names below is the structure of the data.



```
> str(data)
'data.frame':   270 obs. of  14 variables:
 $ age                    : num  70 67 57 64 74 65 56 59 60 63 ...
 $ sex                    : num  1 0 1 1 0 1 1 1 1 0 ...
 $ chest_pain_type        : num  4 3 2 4 2 4 3 4 4 4 ...
 $ resting_blood_pressure : num  130 115 124 128 120 120 130 110 140 150 ...
 $ serum_cholesterol      : num  322 564 261 263 269 177 256 239 293 407 ...
 $ fasting_blood_sugar    : num  0 0 0 0 0 0 1 0 0 0 ...
 $ resting_ecg            : num  2 2 0 0 2 0 2 2 2 2 ...
 $ max_heart_rate         : num  109 160 141 105 121 140 142 142 170 154 ...
 $ exercise_induced_angina: num  0 0 0 1 1 0 1 1 0 0 ...
 $ oldpeak                : num  2.4 1.6 0.3 0.2 0.2 0.4 0.6 1.2 1.2 4 ...
 $ slope                  : num  2 2 1 2 1 1 2 2 2 2 ...
 $ num_major_vessels      : num  3 0 0 1 1 0 1 1 2 3 ...
 $ thal                   : num  3 7 7 7 3 7 6 7 7 7 ...
 $ heart_disease          : int  2 1 2 1 1 1 2 2 2 2 ...
```

Continuous Variables: There are 5 continuous variables and they are

age - Represents age of the person

resting_blood_pressure - Represents the resting blood pressure level.

serum_cholesterol - Represents serum cholesterol levels.

max_heart_rate - Represents the maximum heart rate achieved.

oldpeak - Represents the depression induced by exercise relative to rest.

Categorical Variables: it contains seven variable and they are

sex - Represents gender, typically Male = 1, Female = 0.

chest_pain_type - Represents different types of chest pain.

fasting_blood_sugar - Indicates whether fasting blood sugar is above a certain level (likely 1 for Yes, 0 for No).

resting_ecg - types of resting electrocardiographic results.

exercise_induced_angina - Binary indicator for exercise-induced angina (likely 1 for Yes, 0 for No).

slope - Represents the slope of the peak exercise ST segment.

num_major_vessels - Indicates the number of major vessels colored by fluoroscopy.

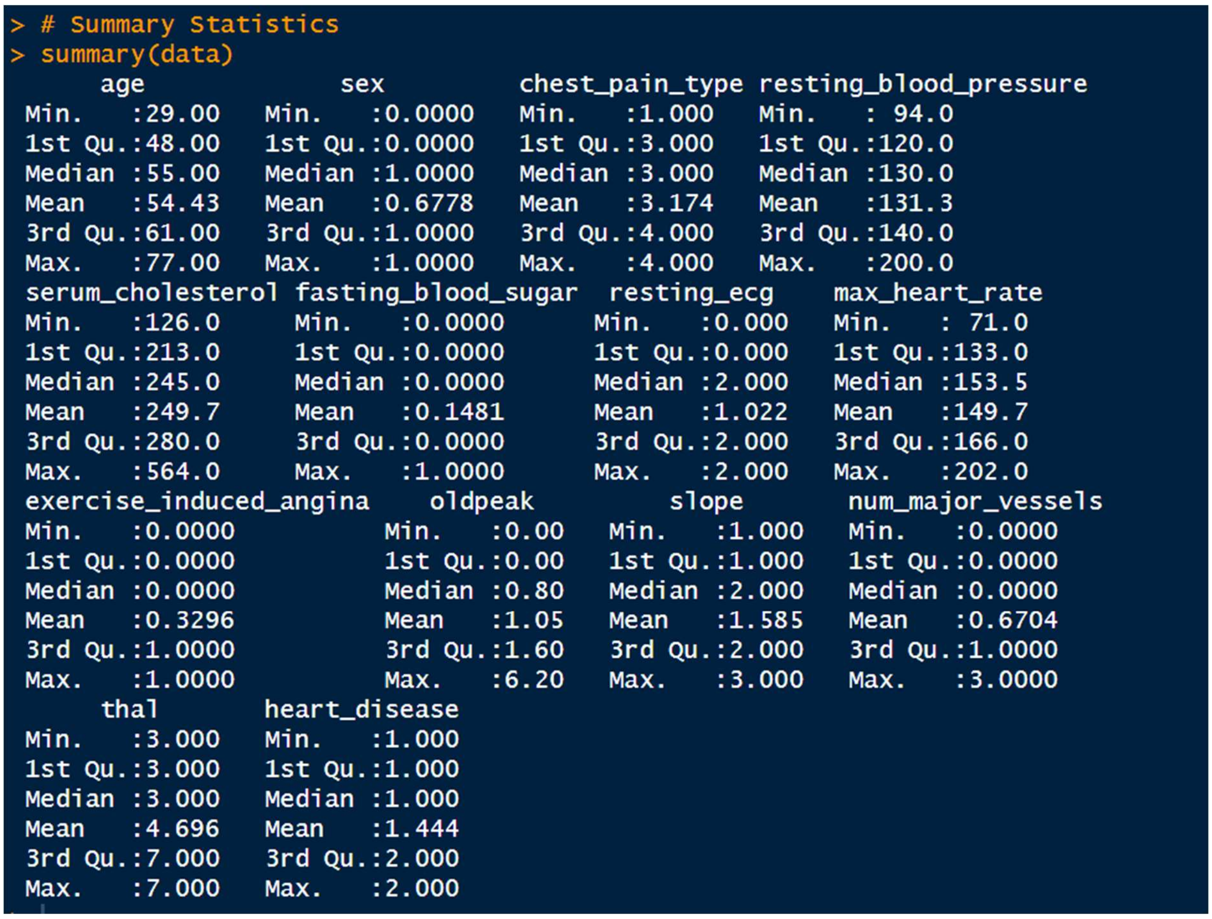
thal - Indicates different types of thalassemia.

Target Variable

heart_disease – This is the target variable, 1 is the presence and 2 is the absence of heart deasease.

a) Exploratory data analysis

Summary of the data



```
> # Summary Statistics
> summary(data)
      age             sex         chest_pain_type resting_blood_pressure
 Min.   :29.00   Min.   :0.0000   Min.   :1.000   Min.   : 94.0
 1st Qu.:48.00   1st Qu.:0.0000   1st Qu.:3.000   1st Qu.:120.0
 Median :55.00   Median :1.0000   Median :3.000   Median :130.0
 Mean   :54.43   Mean   :0.6778   Mean   :3.174   Mean   :131.3
 3rd Qu.:61.00   3rd Qu.:1.0000   3rd Qu.:4.000   3rd Qu.:140.0
 Max.   :77.00   Max.   :1.0000   Max.   :4.000   Max.   :200.0
 serum_cholesterol fasting_blood_sugar  resting_ecg     max_heart_rate
 Min.   :126.0     Min.   :0.0000      Min.   :0.000   Min.   : 71.0
 1st Qu.:213.0     1st Qu.:0.0000      1st Qu.:0.000   1st Qu.:133.0
 Median :245.0     Median :0.0000      Median :2.000   Median :153.5
 Mean   :249.7     Mean   :0.1481      Mean   :1.022   Mean   :149.7
 3rd Qu.:280.0     3rd Qu.:0.0000      3rd Qu.:2.000   3rd Qu.:166.0
 Max.   :564.0     Max.   :1.0000      Max.   :2.000   Max.   :202.0
 exercise_induced_angina    oldpeak         slope       num_major_vessels
 Min.   :0.0000          Min.   :0.00   Min.   :1.000   Min.   :0.0000
 1st Qu.:0.0000          1st Qu.:0.00   1st Qu.:1.000   1st Qu.:0.0000
 Median :0.0000          Median :0.80   Median :2.000   Median :0.0000
 Mean   :0.3296          Mean   :1.05   Mean   :1.585   Mean   :0.6704
 3rd Qu.:1.0000          3rd Qu.:1.60   3rd Qu.:2.000   3rd Qu.:1.0000
 Max.   :1.0000          Max.   :6.20   Max.   :3.000   Max.   :3.0000
      thal        heart_disease
 Min.   :3.000   Min.   :1.000
 1st Qu.:3.000   1st Qu.:1.000
 Median :3.000   Median :1.000
 Mean   :4.696   Mean   :1.444
 3rd Qu.:7.000   3rd Qu.:2.000
 Max.   :7.000   Max.   :2.000
```

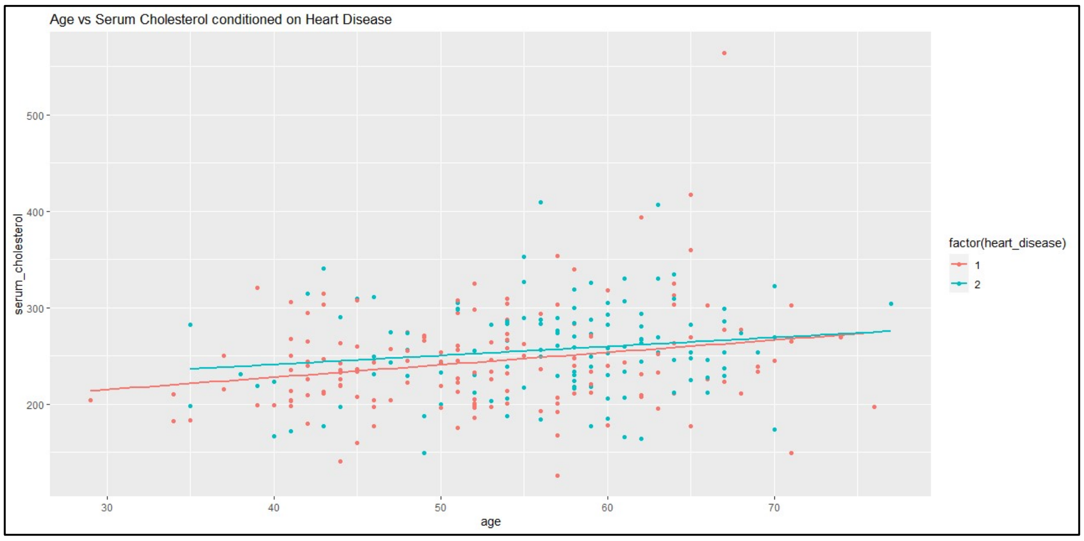
Based on the above summary of the data the variables Age, resting blood pressure, and serum cholesterol. The average age is 54.4 years, and the mean resting blood pressure and serum cholesterol are 131.3 mm Hg and 249.7 mg/dl.

Variables like sex, chest pain type, and fasting blood sugar are categorical and the dataset is male-dominant with a mean of 0.678 for the sex variable.

variables like max heart rate, old peak, and the number of major vessels are discrete.

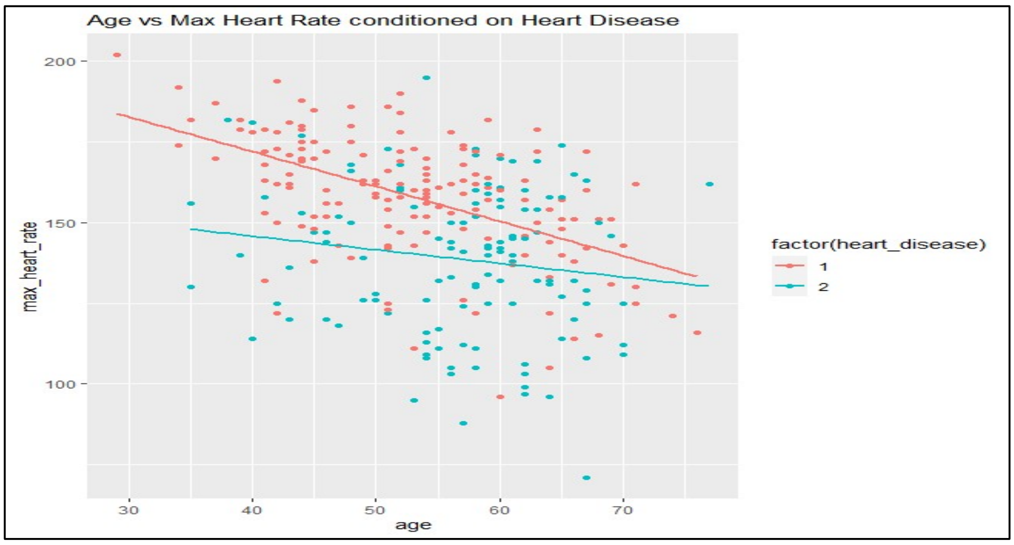
Heart disease is the target variable, with values mainly between 1 (likely no heart disease) and 2 (likely heart disease present). The mean is 1.444, indicating a balanced dataset

Based on the below scatterplot Age vs serum cholesterol conditioned on heart disease, we can see that the presence of heart disease increases with the age and serum cholesterol.

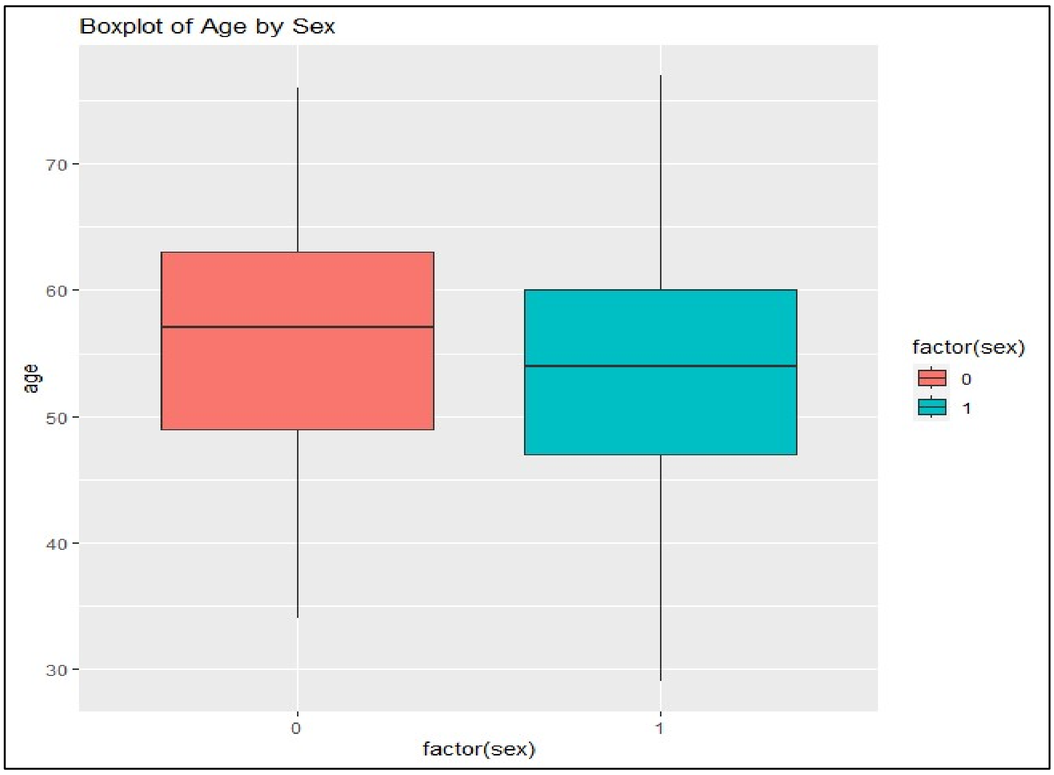


Age vs Serum cholesterol conditioned on heart disease

Based on the below scatterplot age vs heart rate on heart disease we can see that age in the 30s have a higher chances of heart attack and when the age increases there is a less chance of heart attack.



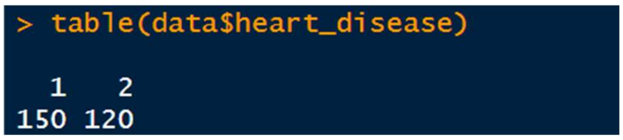
Age vs Heart rate on heart disease



Boxplot of age vs sex

Based on the above boxplot of age vs sex, we can see that the mean of the Females is higher than mean of males indicating higher chances of heart attack for males than females.

Based on the below snapshot we can say that the variable heart disease has two factors 1 and 2, where 1 represents the presence of heart disease and 2 represents the absence of heart disease.

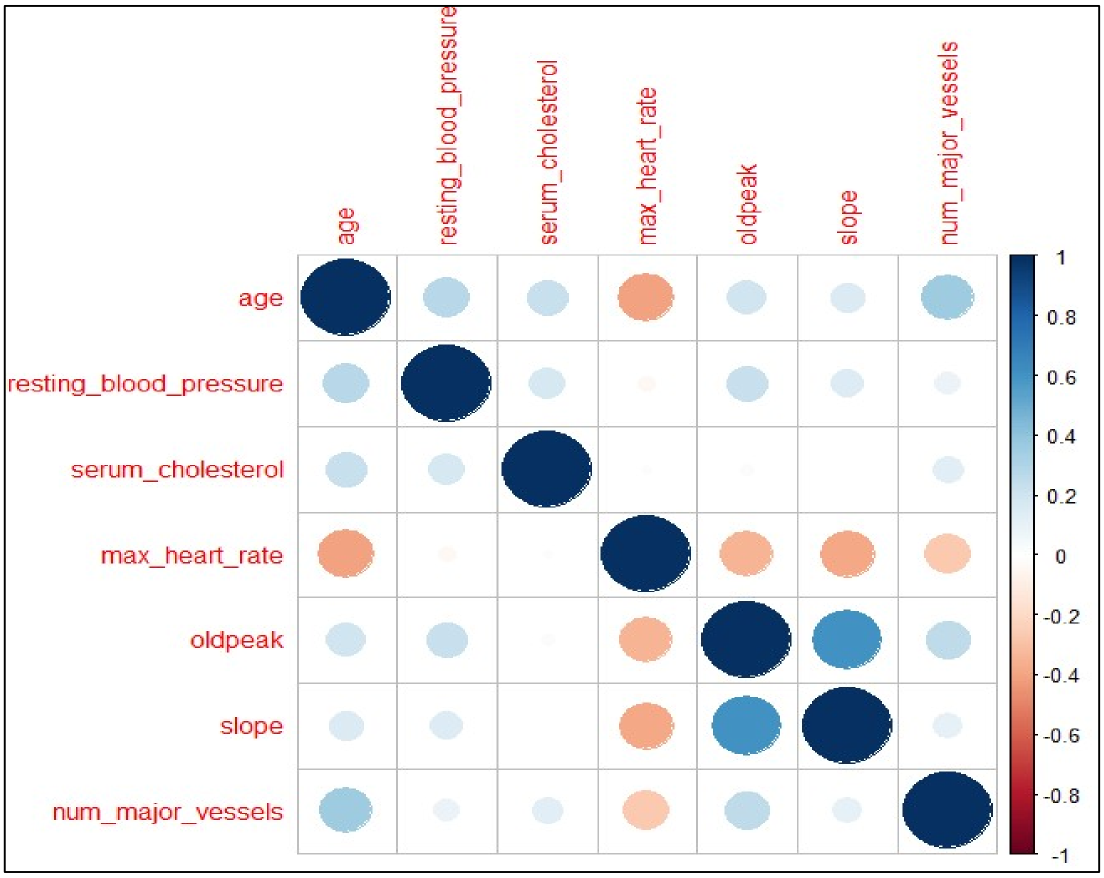


```
> table(data$heart_disease)

  1   2 
150 120 
```

The factor 1 says that having hearting disease have 150 individuals and the factor 2 says that 120 individuals have no heart disease.

Based on the below correlation graph we can say that old peak and slope have the positive relationship as the old peak increases the slope incresases, age and max_heart_rate has has the highest negative relationship indicating max_heart_rate increases with the age increases.



Correlation of heart dataset

Missing values: sum(is.na(data)) is the command to check for the missing values present in the dataset or not, based on executing there are zero missing values present in the dataset.

b) Building a neural network

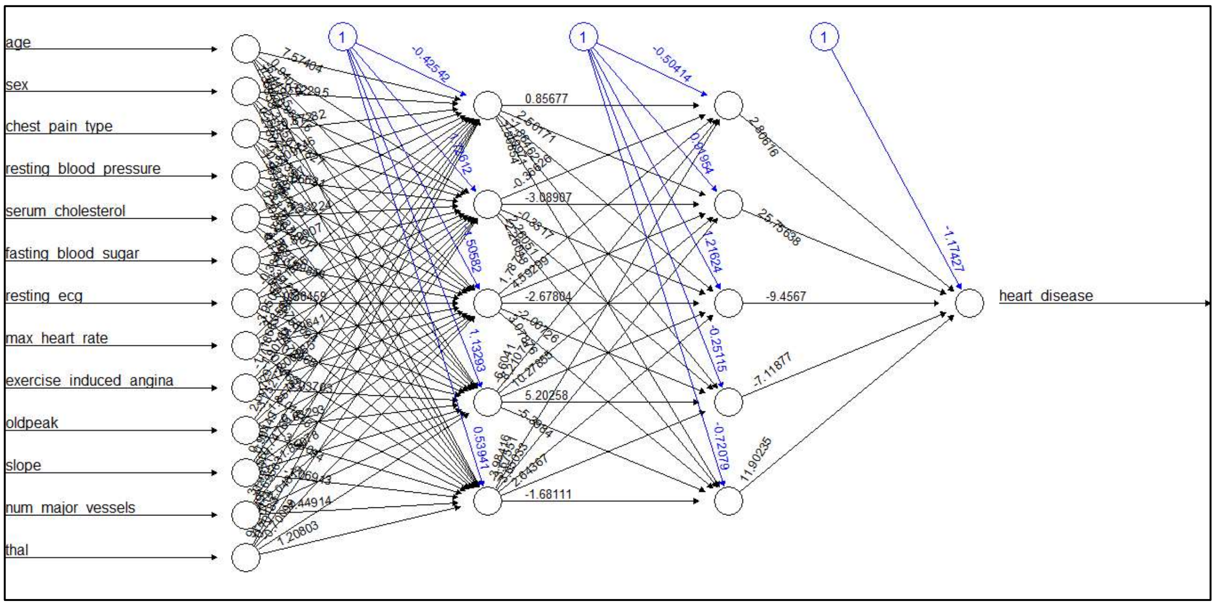
To build a neural network using the 'neural network' library in r for the heart data. The data is normalised using min – max scaling, for every column in the dataset minimum and maximum values are calculated using 'apply' function and these values are stored in the min and max vectors, then the scale function takes the minimum and maximum values to perform scaling on each column.

After scaling the data is split into 80% training and 20% test dataset. In order to predict the 'heart disease' of a predictor variable on a set of predictor variables such as age, sex, type of chest pain, resting blood pressure, serum cholesterol level, fasting blood sugar, resting ECG results, maximum heart rate achieved, whether exercise induced angina, the oldpeak, the slope of the peak exercise ST segment, number of major vessels colored by fluoroscopy, and Thalassemia type.

The model uses two hidden layers, each containing 5 neurons. The activation function used is the logistic function, also known as the sigmoid function, which outputs values between 0 and 1. This is particularly useful for classification problems where the output can be interpreted as the probability of belonging to a particular class.

The neural network is trained on an 80% subset of the scaled data, Once the model is trained, its architecture and learned parameters can be visualized using a plot. This plot provides a way to examine the complexity of the model, including how each input feature contributes to the hidden layers and ultimately to the output.

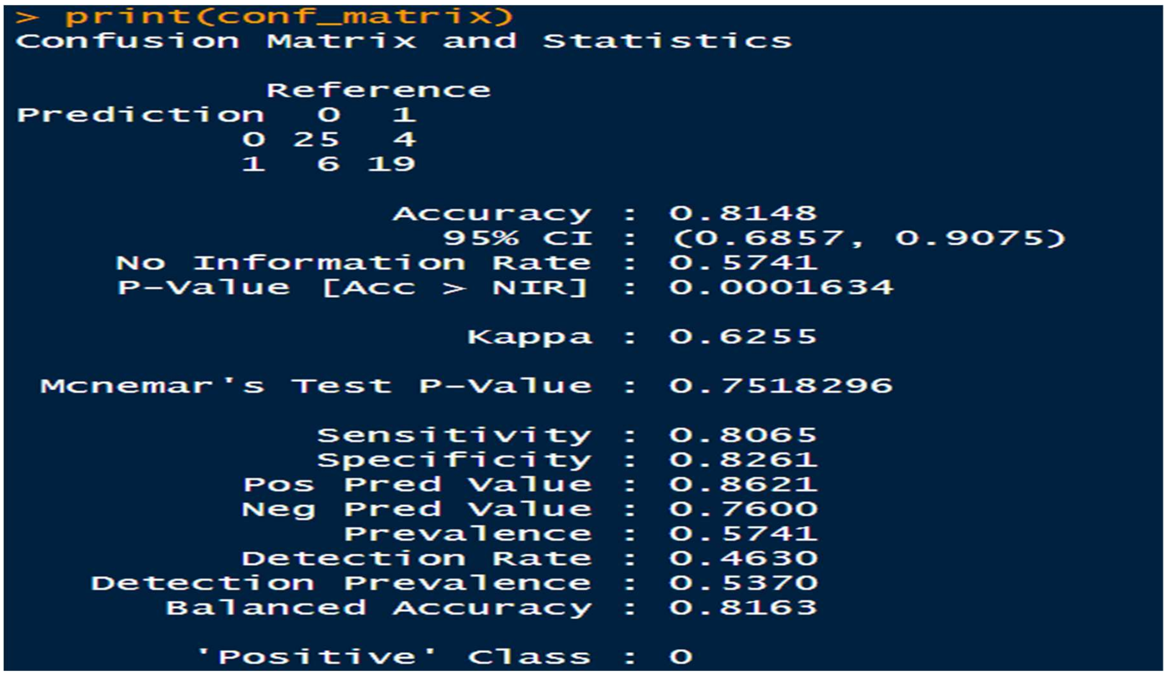
Below is the image of neural network trained to predict the presence and absence of heart disease



Training the neural network

c) Confusion Matrix

The confusion matrix helps to evaluate the performance of the model on the test, below is the confusion matrix which helps us to evaluate the model accuracy.



```
> print(conf_matrix)
Confusion Matrix and Statistics

          Reference
Prediction  0  1
         0 25  4
         1  6 19

               Accuracy : 0.8148
                 95% CI : (0.6857, 0.9075)
    No Information Rate : 0.5741
    P-Value [Acc > NIR] : 0.0001634

                  Kappa : 0.6255

 Mcnemar's Test P-Value : 0.7518296

            Sensitivity : 0.8065
            Specificity : 0.8261
         Pos Pred Value : 0.8621
         Neg Pred Value : 0.7600
             Prevalence : 0.5741
         Detection Rate : 0.4630
   Detection Prevalence : 0.5370
      Balanced Accuracy : 0.8163

       'Positive' Class : 0
```

Confusion matrix

Based on the above snapshot we can see that the model is 81.48% accurate, which has the ability to predict the heart disease presence or absence.

Kappa: The Kappa statistic is 0.6255, showing a moderate level of agreement between the prediction and the actual outcome, considering that agreement could occur by random chance.

Sensitivity (Recall): At 80.65% it shows how well the model identifies positives. It suggests that the model is good at identifying instances where heart disease is absent.

Specificity: At 82.61% it shows shows how well the model identifies negatives. It's good at identifying instances where heart disease is present.

Positive Predictive Value (PPV): The PPV is 86.21%, which is high and suggests that the likelihood of the model's positive predictions being actual positives is excellent.

Negative Predictive Value (NPV): The NPV is 76.00%, suggesting that the model's negative predictions are generally reliable but could be improved.

Overall, the model performs well, but there is still room for improvement, particularly in increasing sensitivity and negative predictive value.

d) Overfitting

Overfitting occurs when a machine learning model learns the training data too well, capturing not just the underlying pattern but also the noise and random fluctuations present. As a result, an overfitted model will perform well on the training data but poorly on new, unseen data, as it generalizes poorly. Essentially, it becomes too tailored to the training set and loses its ability to generalize to new data.

How to Address Overfitting in This Context:

Regularization: Neural networks, including the one built using the neuralnet library in R, often include a regularization term in the loss function. This term penalizes overly complex models, effectively limiting their ability to overfit. You can adjust the weight decay parameter in neuralnet to apply regularization.

Simplifying the Architecture: Reducing the number of hidden layers or neurons in the neural network can make the model simpler and less prone to overfitting.

Early Stopping: Monitor the model's performance on a validation set during training and stop when performance starts to degrade, rather than continuing until all epochs are completed.

Cross-Validation: Implement k-fold cross-validation to ensure that the model generalizes well to unseen data. This involves dividing the dataset into 'k' subsets and training the model k times, each time using a different subset as the validation set and the remaining data as the training set.

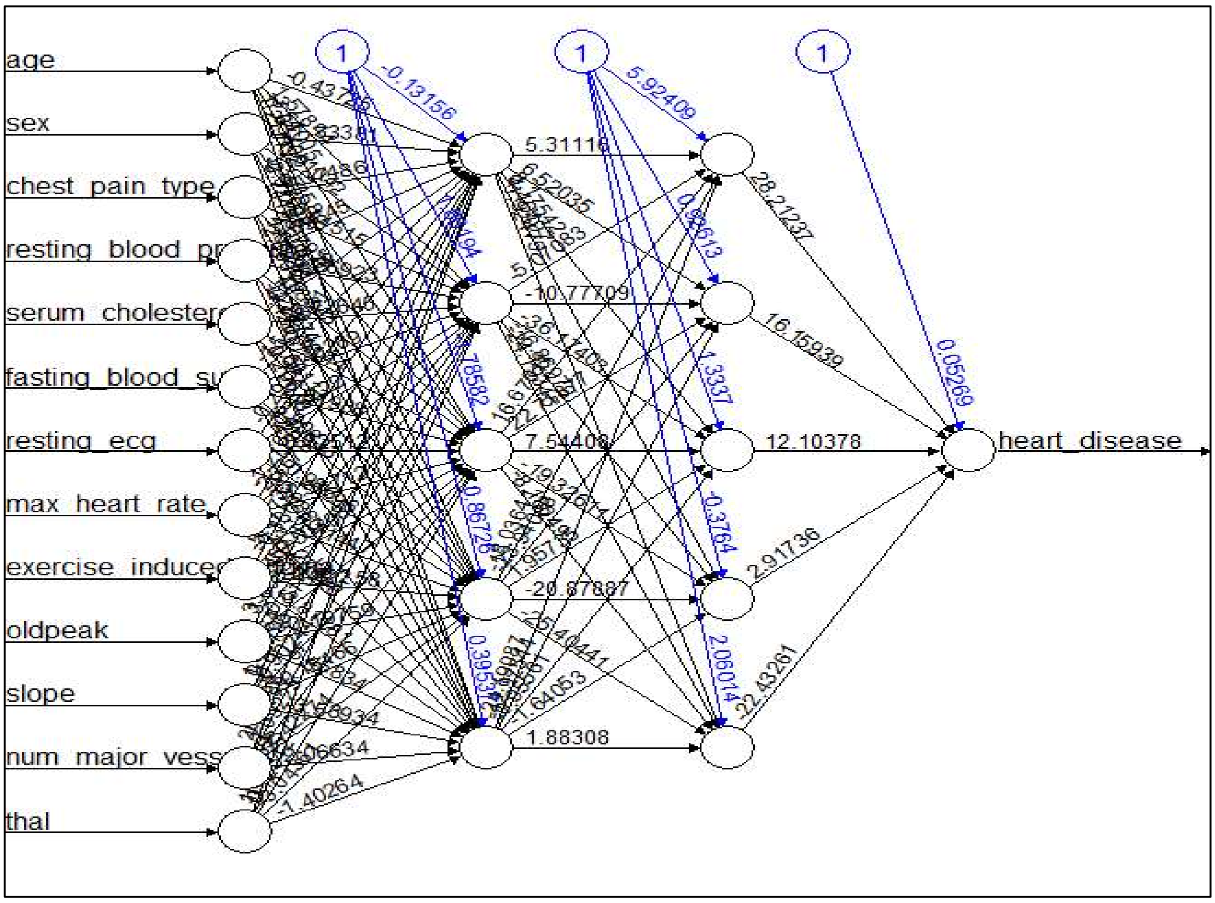
Prune the Model: After training, you can remove neurons or even entire layers that contribute little to the model's predictive power. This will result in a simpler and more interpretable model that is less likely to overfit.

Dropout: Randomly setting a fraction of the input units to 0 during training, which can help to improve generalization.

By applying one or more of these techniques and then re-evaluating the model using the test set and the confusion matrix, we can get a better sense of how well your adjustments have worked in combating overfitting

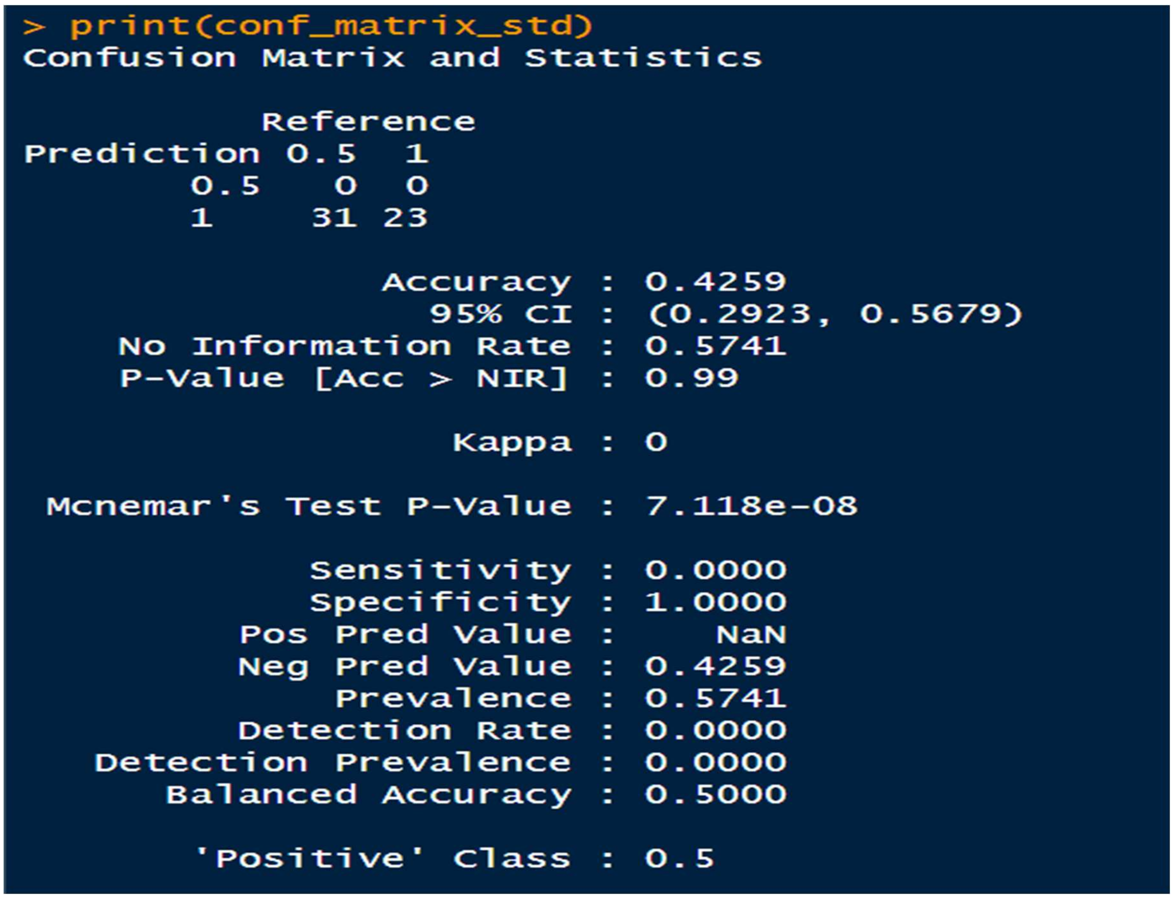
e) Improving the performance of the model by standardising the data

Before we have normalized the data by scaling between the minimum and maximum values for each column, now we will standardize the data by dividing each value by the maximum value for each column



Neural network after applying standardising the data

This is the neural network of predicting the heart disease on the predictor variables after standardising the data. Which helps the neural network to converge faster and reach a better accuracy. And the data is split into 80% training to train the data and 20% to test the data.



```
> print(conf_matrix_std)
Confusion Matrix and Statistics

          Reference
Prediction 0.5  1
       0.5   0  0
       1    31 23

               Accuracy : 0.4259
                 95% CI : (0.2923, 0.5679)
    No Information Rate : 0.5741
    P-Value [Acc > NIR] : 0.99

                  Kappa : 0

 Mcnemar's Test P-Value : 7.118e-08

            Sensitivity : 0.0000
            Specificity : 1.0000
         Pos Pred Value :    NaN
         Neg Pred Value : 0.4259
             Prevalence : 0.5741
         Detection Rate : 0.0000
   Detection Prevalence : 0.0000
      Balanced Accuracy : 0.5000

       'Positive' Class : 0.5
```

Confusion matrix

Based on the above confusion matrix we can say that the accuracy is 42.59% which is quite low previous model and it has given poor performance in predicting the presence and absence of the heat disease.

The model's performance may have been adversely affected by the standardization method employed. In this case, dividing each value by the max might not be the appropriate scaling strategy.

The neural network model may require re-tuning or architectural changes to better understand the standardized data.

f) Hyperparameter tuning strategy

Hyperparameter tuning is an important step in optimizing a neural network for better performance. Here are some methods to tune some of the key hyperparameters.

Methodology

Grid Search vs. Random Search: we can either use a grid search to explore the entire parameter space systematically or we can use a random search for faster but less comprehensive coverage. caret package in R provides excellent tools for this.

Cross-Validation: Employing k-fold cross-validation for a more robust evaluation of the hyperparameter combinations and we can also use Stratified k-fold if the classes are imbalanced.

Early Stopping: in order to prevent overfitting during these extensive experiments, implement early stopping and this will terminate training when performance on a validation set starts deteriorating.

Key Hyperparameters to Tune:

Number of Hidden Layers and Neurons: Starting with a single hidden layer and gradually increasing it while monitoring performance.

Learning Rate: Trying various learning rates to ensure that the model converges well. Too small a rate can lead to slow convergence, while too large can lead to overshooting.

Activation Functions: Experimenting with different activation functions like ReLU, Sigmoid, and Tanh to see which one performs better.

Batch Size: Depending on data size, experimenting with different batch sizes can significantly affect your model's performance and training time.

Regularization Techniques: Fine-tune regularization parameters like dropout rates and L1/L2 regularization terms.

Momentum/Adam Optimization: If using momentum or Adam optimizers, their parameters can also be tuned.

g) Maximum Likelihood Estimation

References

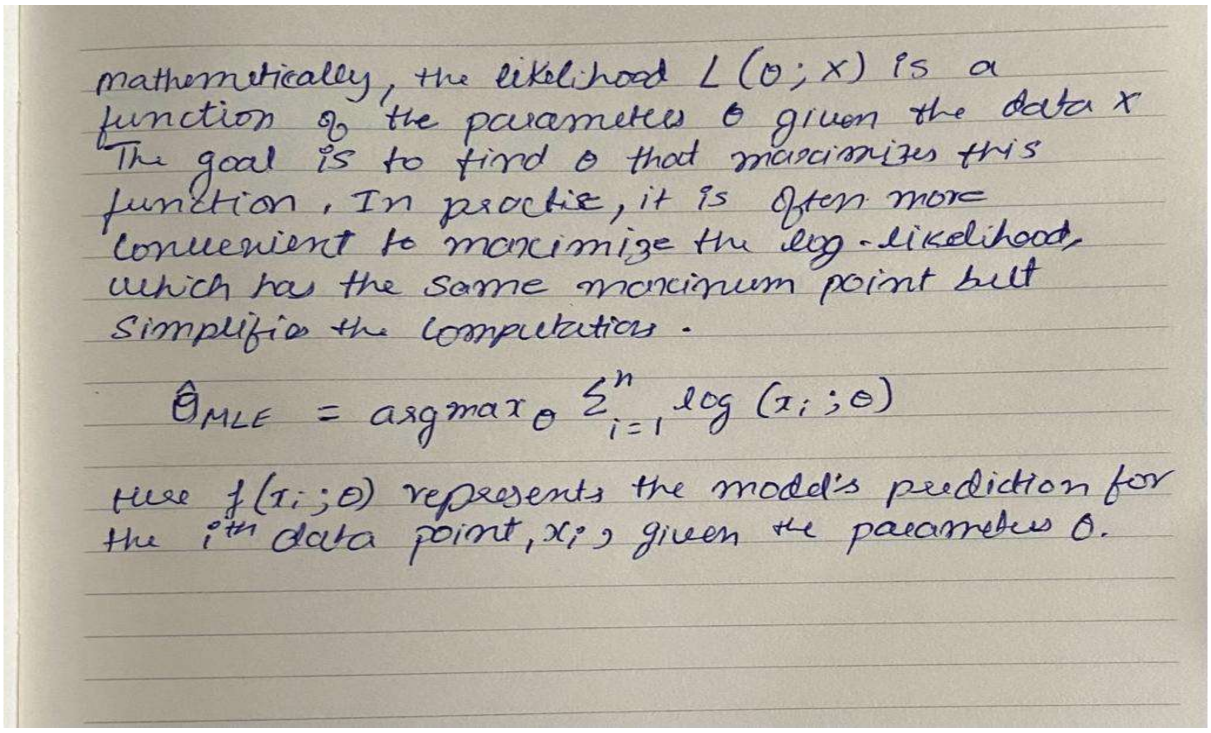
1.The Elements of Statistical Learning(https://hastie.su.domains/ElemStatLearn/)

2.Deep Learning Book by1 Ian Goodfellow, Yoshua Bengio, and Aaron Courville(https://www.deeplearningbook.org/)

3.StatQuest: Maximum Likelihood, clearly explained!!! (https://www.youtube.com/watch?v=XepXtl9YKwc&ab_channel=StatQuestwithJoshStarmer)

3.Maximum Likelihood Estimation for Neural Network from scratch - an article on Towards Data Science

Maximum Likelihood Estimation (MLE) is a statistical method used for estimating the parameters of a model. In the context of neural networks, MLE aims to find the set of parameters (weights and biases) that maximize the likelihood of the observed data given the model. Essentially, it tries to adjust the network's parameters so that the predicted outcomes are as close as possible to the actual observations.



mathemetically, the likelihood $L(\theta; x)$ is a function of the parameters θ given the data x The goal is to find θ that maximizes this function, In practice, it is often more convenient to maximize the log-likelihood, which has the same maximum point but simplifies the computation.

$$\hat{\theta}_{MLE} = \arg\max_{\theta} \sum_{i=1}^{n} \log (x_i ; \theta)$$

Here $f(x_i;\theta)$ represents the model's prediction for the i^{th} data point, x_i, given the parameters θ.

How MLEs Are Used in Neural Networks:

Objective Function: In the context of a neural network trained for classification, the commonly used loss function corresponding to MLE is the categorical cross-entropy loss. For regression tasks, the equivalent is often the mean squared error (MSE). These loss functions can be seen as negative log-likelihood functions, and minimizing them is equivalent to maximizing the likelihood.

Backpropagation and Optimization: During the training process, the backpropagation algorithm adjusts the network's weights and biases to minimize the loss function. Optimization algorithms like Gradient Descent are used to find the parameter values that minimize the negative log-likelihood, effectively maximizing the likelihood.

Regularization: Regularization techniques like L1 or L2 regularization can also be incorporated into the MLE framework by adding a penalty term to the likelihood function.

Evaluation: Once trained, the quality of the model's parameters can be assessed by evaluating them on a validation set, providing an unbiased assessment of the model's performance and ensuring that the parameters are optimal not just for the training data but also for new, unseen data.

By finding the parameters that maximize the likelihood, you are essentially fitting the neural network in a way that makes the observed data most probable under the model.

Code:

```
setwd("C:\\Users\\shash\\OneDrive\\Desktop\\Data Science and Analytics")

######################################## Question 2 ###########################

####### a) to calculate total number of parameters

# Load the neuralnet library

#install.packages("neuralnet")
```

```
library(neuralnet)

set.seed(414)

# sample number of neurons for the input and hidden layers

layer_neurons_input_hidden <- sample(3:10, 2)

input_neurons <- layer_neurons_input_hidden[1]

hidden_neurons <- layer_neurons_input_hidden[2]

# sample number of neurons for the output layer

output_neurons <- sample(2:4, 1)

# Print out the sampled number of neurons for each layer

cat("Number of neurons in input layer:", input_neurons, "\n")

cat("Number of neurons in hidden layer:", hidden_neurons, "\n")

cat("Number of neurons in output layer:", output_neurons, "\n")

#################################### question 3 ###########################

# read.dat file with headers

data <- read.table("heart.dat", sep=" ", header=FALSE)

# copying the data into another variable

data1 = data

# structure of the data

str(data)

# Assigning column names

colnames(data) <- c(

  "age", "sex", "chest_pain_type", "resting_blood_pressure",

  "serum_cholesterol", "fasting_blood_sugar", "resting_ecg",
  "max_heart_rate", "exercise_induced_angina", "oldpeak",
  "slope", "num_major_vessels", "thal", "heart_disease"
)
# Checking the structure
str(data)
###################### EDA(EXPLOROtary DATA ANALYSIS) ######################
# Load necessary libraries
library(ggplot2)
library(dplyr)
library(corrplot)
# Check the structure of the data
str(data)
# Summary Statistics
summary(data)
# Data Visualization
# Histograms for numerical variables
num_vars <- c('age', 'resting_blood_pressure', 'serum_cholesterol', 'max_heart_rate',
'oldpeak', 'slope', 'num_major_vessels')
ggplot(data, aes(x=age, y=serum_cholesterol, color=factor(heart_disease))) +
  geom_point() +
  geom_smooth(method='lm', se=FALSE, aes(group=factor(heart_disease),
color=factor(heart_disease))) +
  ggtitle("Age vs Serum Cholesterol conditioned on Heart Disease")
```

```
ggplot(data, aes(x=age, y=max_heart_rate, color=factor(heart_disease))) +
  geom_point() +
  geom_smooth(method='lm', se=FALSE, aes(group=factor(heart_disease),
color=factor(heart_disease))) +
  ggtitle("Age vs Max Heart Rate conditioned on Heart Disease")
# Boxplots for Age against different categorical variables
ggplot(data, aes(x=factor(sex), y=age, fill=factor(sex))) +
  geom_boxplot() +
  ggtitle("Boxplot of Age by Sex")
ggplot(data, aes(x=factor(chest_pain_type), y=age, fill=factor(chest_pain_type))) +
  geom_boxplot() +
  ggtitle("Boxplot of Age by Chest Pain Type")
# Correlation Analysis
cor_matrix <- cor(data[, num_vars], method='pearson')
corrplot(cor_matrix, method='circle')
# Class Distribution
table(data$heart_disease)
# Check for missing values
sum(is.na(data))
##################### b ###############################
# Normalize data
maxs <- apply(data, 2, max)
mins <- apply(data, 2, min)
```

```
scaled_data <- as.data.frame(scale(data, center = mins, scale = maxs - mins))

# Split the data into training (80%) and testing (20%) sets

set.seed(414)

sample <- sample(1:nrow(scaled_data), size = 0.8 * nrow(scaled_data))

train_data <- scaled_data[sample, ]

test_data <- scaled_data[-sample, ]

# Install and load the neuralnet library

install.packages("neuralnet")

library(neuralnet)

# Define the formula for predicting 'heart_disease' based on other variables

formula <- as.formula("heart_disease ~ age + sex + chest_pain_type + resting_blood_pressure
+ serum_cholesterol + fasting_blood_sugar + resting_ecg + max_heart_rate +
exercise_induced_angina + oldpeak + slope + num_major_vessels + thal")

# Train the neural network

nn <- neuralnet(formula, data = train_data, hidden = c(5, 5), linear.output = FALSE, act.fct =
"logistic")

# Plot the neural network

plot(nn)

################################# c ####################

# Compute the neural network predictions

nn_predictions <- predict(nn, test_data[,1:13])

# Convert these probabilities to binary outcomes (assuming threshold = 0.5)

nn_pred_class <- ifelse(nn_predictions > 0.5, 1, 0)
```

```
# Create a confusion matrix

library(caret) # Load the 'caret' package for confusionMatrix function

conf_matrix <- confusionMatrix(as.factor(nn_pred_class), as.factor(test_data$heart_disease))

# Print the confusion matrix

print(conf_matrix)

############################## e #################################

# Standardizing the data by dividing each value by the max value of each column

std_data <- as.data.frame(lapply(data, function(x) x / max(x, na.rm = TRUE)))

# Splitting the data into training (80%) and test (20%) sets

set.seed(414)

sample <- sample(1:nrow(std_data), size = 0.8 * nrow(std_data))

train_data_std <- std_data[sample, ]

test_data_std <- std_data[-sample, ]

# Train the neural network on standardized data

nn_std <- neuralnet(formula, data = train_data_std, hidden = c(5, 5), linear.output = FALSE, act.fct = "logistic")

plot(nn_std)

# Make predictions on the test set

nn_predictions_std <- predict(nn_std, test_data_std[,1:13])

nn_pred_class_std <- ifelse(nn_predictions_std > 0.5, 1, 0)

# Create a confusion matrix for the standardized model

conf_matrix_std <- confusionMatrix(as.factor(nn_pred_class_std), as.factor(test_data_std$heart_disease))
```

```
print(conf_matrix_std)
```