Analysis and Study of UCI Heart Disease dataset

by Prateek Sarangi, Mon , Aug 17 2020

Part 1:- Getting the feature dataset from the actual data

Dataset details

Attribute information

- age
- sex
- chest pain type (4 values) -> cp
- resting blood pressure -> trestbps
- serum cholestoral in mg/dl-> chol
- fasting blood sugar > 120 mg/dl -> fbs
- resting electrocardiographic results (values 0,1,2) -> restecg
- maximum heart rate achieved -> thalach
- exercise induced angina -> exang
- oldpeak = ST depression induced by exercise relative to rest -> oldpeak
- the slope of the peak exercise ST segment -> slope
- number of major vessels (0-3) colored by flourosopy -> ca
- thal: 3 = normal; 6 = fixed defect; 7 = reversable defect

Program for exploratory analysis and preprocessing

Function to Normalize the data

```
normalize <- function(x) {
  return ((x - min(x)) / (max(x) - min(x)))
}
```

Import dataset and view it as a dataframe

```
heart <- read.csv("~/HeartDisease/heart.csv")
head(heart)
```

```
##   age sex cp trestbps chol fbs restecg thalach exang oldpeak slope ca thal
```

```
## 1  63   1  3       145  233   1       0      150      0      2.3      0  0    1
## 2  37   1  2       130  250   0       1      187      0      3.5      0  0    2
## 3  41   0  1       130  204   0       0      172      0      1.4      2  0    2
## 4  56   1  1       120  236   0       1      178      0      0.8      2  0    2
## 5  57   0  0       120  354   0       1      163      1      0.6      2  0    2
## 6  57   1  0       140  192   0       1      148      0      0.4      1  0    1
##   target
## 1      1
## 2      1
## 3      1
## 4      1
## 5      1
## 6      1
```

Normalize the age and separate it in various groups to show which age group is more prone to heart disease

It uses the normalize function defined above and then categorize the data into **four** groups.
After normalization the valuesaranges from **0-1**
It is grouped as follows
- **0.00 to 0.25** as **Group 1** giving it value **0.1**
- **0.25 to 0.50** as **Group 2** giving it value **0.4**
- **0.50 to 0.75** as **Group 3** giving it value **0.6**
- **0.75 to 1.00** as **Group 4** giving it value **0.9**

```
dfNorm <- as.data.frame(lapply(heart["age"], normalize))
heart["age"] <- dfNorm
heart["age"] <- as.data.frame(lapply(heart["age"], function(x){replace(x,between(x, 0.0, 0.25), 0.1)}))
heart["age"] <- as.data.frame(lapply(heart["age"], function(x){replace(x,between(x, 0.25, 0.6), 0.4)}))
heart["age"] <- as.data.frame(lapply(heart["age"], function(x){replace(x,between(x, 0.5, 0.75), 0.6)}))
heart["age"] <- as.data.frame(lapply(heart["age"], function(x){replace(x,between(x, 0.75, 1), 0.9)}))
```

Normalize the Rest Blood Pressure and separate it in various groups to show what blood pressure group is more prone to heart disease

It uses the normalize function defined above and then categorize the data into **three** groups.
After normalization the valuesaranges from **0-1**
It is grouped as follows
- **0.00 to 0.33** as **Group 1** giving it value **0.2**
- **0.33 to 0.67** as **Group 2** giving it value **0.6**
- **0.67 to 1.00** as **Group 3** giving it value **1.0**

```
dfNorm <- as.data.frame(lapply(heart["trestbps"], normalize))
heart["trestbps"] <- dfNorm
heart["trestbps"] <- as.data.frame(lapply(heart["trestbps"], function(x){replace(x, between(x, 0.0, 0.33
heart["trestbps"] <- as.data.frame(lapply(heart["trestbps"], function(x){replace(x, between(x, 0.33, 0.6
heart["trestbps"] <- as.data.frame(lapply(heart["trestbps"], function(x){replace(x, between(x, 0.67, 1)
```

Normalize the Cholestrole level and separate it in various groups to show what cholestrole level is more prone to heart disease

It uses the normalize function defined above and then categorize the data into **five** groups.
After normalization the valuesaranges from **0-1**
It is grouped as follows
- **0.00 to 0.20** as **Group 1** giving it value **0.1**

- **0.20 to 0.40** as **Group 2** giving it value **0.3**
- **0.40 to 0.60** as **Group 3** giving it value **0.5**
- **0.60 to 0.80** as **Group 4** giving it value **0.7**
- **0.80 to 1.00** as **Group 5** giving it value **0.9**

```
dfNorm <- as.data.frame(lapply(heart["chol"], normalize))
heart["chol"] <- dfNorm
heart["chol"] <- as.data.frame(lapply(heart["chol"], function(x){replace(x, between(x, 0.0, 0.2), 0.1)}
heart["chol"] <- as.data.frame(lapply(heart["chol"], function(x){replace(x, between(x, 0.2, 0.4), 0.3)}
heart["chol"] <- as.data.frame(lapply(heart["chol"], function(x){replace(x, between(x, 0.4, 0.6), 0.5)}
heart["chol"] <- as.data.frame(lapply(heart["chol"], function(x){replace(x, between(x, 0.6, 0.8), 0.7)}
heart["chol"] <- as.data.frame(lapply(heart["chol"], function(x){replace(x, between(x, 0.8, 1), 0.9)}))
```

Separate the data in various groups to show which category of chest pain is more prone to heart disease

It replaces the values of chest pain into different groups.
- **Value 0**:- Typical angina(**Provided value 0.1**).
- **Value 1**:- Atypical angina(**Provided value 0.6**).
- **Value 2**:- Non-anginal pain(**Provided value 0.9**).
- **Value 3**:- Asymptomatic(**Provided value 0.01**).

```
heart["cp"] <- as.data.frame(lapply(heart["cp"], function(x){replace(x, x == 0, 0.1)}))
heart["cp"] <- as.data.frame(lapply(heart["cp"], function(x){replace(x, x == 1, 0.6)}))
heart["cp"] <- as.data.frame(lapply(heart["cp"], function(x){replace(x, x == 2, 0.9)}))
heart["cp"] <- as.data.frame(lapply(heart["cp"], function(x){replace(x, x == 3, 0.01)}))
```

Normailze the maximum heart rate of the patient

```
dfNorm <- as.data.frame(lapply(heart["thalach"], normalize))
heart["thalach"] <- dfNorm
```

Normalize the thal and separate it in various groups to show what thal is more prone to heart disease

It uses the normalize function defined above and then categorize the data into **three** groups.
Thal: 3 = normal; 6 = fixed defect; 7 = reversable defect
After normalization the valuesaranges from **0-1**
It is grouped as follows
- **0.00 to 0.25** as **Group 1** giving it value **0.5**
- **0.25 to 0.50** as **Group 2** giving it value **0.6**
- **0.50 to 0.75** as **Group 3** giving it value **0.9**
- **0.75 to 1.00** as **Group 4** giving it value **0.1**

```
dfNorm <- as.data.frame(lapply(heart["thal"], normalize))
heart["thal"] <- dfNorm
heart["thal"] <- as.data.frame(lapply(heart["thal"], function(x){replace(x, between(x, 0.0, 0.25), 0.5)
heart["thal"] <- as.data.frame(lapply(heart["thal"], function(x){replace(x, between(x, 0.25, 0.50), 0.6
heart["thal"] <- as.data.frame(lapply(heart["thal"], function(x){replace(x, between(x, 0.50, 0.75), 0.9
heart["thal"] <- as.data.frame(lapply(heart["thal"], function(x){replace(x, between(x, 0.75, 1.00), 0.1
```

Regularizing the rest ECG value of the patient

It replaces the values of Rest ECG in the following manner.
- **Value 0**:- Normal(**Provided value 0.3**).

- **Value 1**:- Having ST-T wave abnormality (T wave inversions and/or ST elevation or depression of > 0.05 mV)(**Provided value 0.9**).
- **Value 2**:- showing probable or definite left ventricular hypertrophy by Estes' criteria.(**Provided value 0.1**)

```
heart["restecg"] <- as.data.frame(lapply(heart["restecg"], function(x){replace(x, x == 0, 0.3)}))
heart["restecg"] <- as.data.frame(lapply(heart["restecg"], function(x){replace(x, x == 1, 0.9)}))
heart["restecg"] <- as.data.frame(lapply(heart["restecg"], function(x){replace(x, x == 2, 0.1)}))
```

Replace the peak exercise ST segment of the patient

It replaces the values of peak exercise ST segment into different groups.
- **Value 0**:- Upsloping(**Provided value 0.01**).
- **Value 1**:- Flat(**Provided value 0.2**).
- **Value 2**:- Downsloping(**Provided value 0.9**).

```
heart["slope"] <- as.data.frame(lapply(heart["slope"], function(x){replace(x, x == 0, 0.01)}))
heart["slope"] <- as.data.frame(lapply(heart["slope"], function(x){replace(x, x == 1, 0.2)}))
heart["slope"] <- as.data.frame(lapply(heart["slope"], function(x){replace(x, x == 2, 0.9)}))
```

Replace the major vessels (0-3) colored by flourosopy

It replaces the values of major vessels colored into different groups.
- **Value 0**:- Upsloping(**Provided value 0.09**).
- **Value 1**:- Flat(**Provided value 0.6**).
- **Value 2**:- Downsloping(**Provided value 0.45**).
- **Value 3**:- Downsloping(**Provided value 0.3**).
- **Value 4**:- Downsloping(**Provided value 0.1**).

```
heart["ca"] <- as.data.frame(lapply(heart["ca"], function(x){replace(x, x == 0, 0.9)}))
heart["ca"] <- as.data.frame(lapply(heart["ca"], function(x){replace(x, x == 1, 0.6)}))
heart["ca"] <- as.data.frame(lapply(heart["ca"], function(x){replace(x, x == 2, 0.45)}))
heart["ca"] <- as.data.frame(lapply(heart["ca"], function(x){replace(x, x == 3, 0.3)}))
heart["ca"] <- as.data.frame(lapply(heart["ca"], function(x){replace(x, x == 4, 0.1)}))
```

Replace the fasting bloog sugar

It replaces the values of fasting bloog sugar into different groups.
- **Value 0**:- Bloog sugar < 120 mg/dl(**Provided value 0.9**).
- **Value 1**:- Bloog sugar > 120 mg/dl(**Provided value 0.1**).

```
heart["fbs"] <- as.data.frame(lapply(heart["fbs"], function(x){replace(x, x == 0, 0.9)}))
heart["fbs"] <- as.data.frame(lapply(heart["fbs"], function(x){replace(x, x == 1, 0.1)}))
```

Replace the sex of the patient with values

It replaces the values of sex into different groups.
- **Value 1**:- Male, is replaced with **0.9**
- **Value 0**:- Female, is replaced with **0.1**

```
heart["sex"] <- as.data.frame(lapply(heart["sex"], function(x){replace(x, x == 0, 0.1)}))
heart["sex"] <- as.data.frame(lapply(heart["sex"], function(x){replace(x, x == 1, 0.9)}))
```

Replace the exercise induced angina

It replaces the values of peak exercise ST segment into different groups.
- **Value 0**:- No(**Provided value 0.9**).

- **Value 1**:- Yes(**Provided value 0.1**).

```
heart["exang"] <- as.data.frame(lapply(heart["exang"], function(x){replace(x, x == 0, 0.9)}))
heart["exang"] <- as.data.frame(lapply(heart["exang"], function(x){replace(x, x == 1, 0.1)}))
```

The dataset after the modifications are made.

```
##   age sex   cp trestbps chol fbs restecg   thalach exang oldpeak slope  ca thal
## 1 0.6 0.9 0.01      0.6  0.3 0.1     0.3 0.6030534   0.9     2.3  0.01 0.9  0.1
## 2 0.1 0.9 0.90      0.6  0.3 0.9     0.9 0.8854962   0.9     3.5  0.01 0.9  0.1
## 3 0.1 0.1 0.60      0.6  0.1 0.9     0.3 0.7709924   0.9     1.4  0.90 0.9  0.1
## 4 0.4 0.9 0.60      0.2  0.3 0.9     0.9 0.8167939   0.9     0.8  0.90 0.9  0.1
## 5 0.4 0.1 0.10      0.2  0.5 0.9     0.9 0.7022901   0.1     0.6  0.90 0.9  0.1
## 6 0.4 0.9 0.10      0.6  0.1 0.9     0.9 0.5877863   0.9     0.4  0.20 0.9  0.1
##   target
## 1      1
## 2      1
## 3      1
## 4      1
## 5      1
## 6      1
```

Splitting of training and testing data

We are using both randomly generated and sequentially choosen 75% of the data as training set and rest 25% as our test set.
train_ind_rand gives the indeces of the samples which are to be used as the training sample in the dataset.
trainrand gives the randomly choosen train dataset.
testrand given the randomly choosen test dataset.
trainseq gives the sequencially choosen train dataset.
testseq gives the sequentially choosen test dataset.

```
smp_size <- floor(0.75 * nrow(heart))
train_ind_rand <- sample(seq_len(nrow(heart)), size = smp_size)

trainrand <- heart[train_ind_rand, ]
testrand <- heart[-train_ind_rand, ]

trainseq <- heart[1:227, ]
testseq <- heart[227:303, ]
```

Writting the dataset into csv files so that it can be used in the Python program for Neural network classification.

- **heart1.csv** contain the modified Heart Disease dataset.
- **trainrand.csv** conains the randomly chosen train dataset.
- **testrand.csv** conains the randomly chosen test dataset.
- **trainseq.csv** conains the sequencially choosen train dataset.
- **testseq.csv** conains the sequentially choosen test dataset.

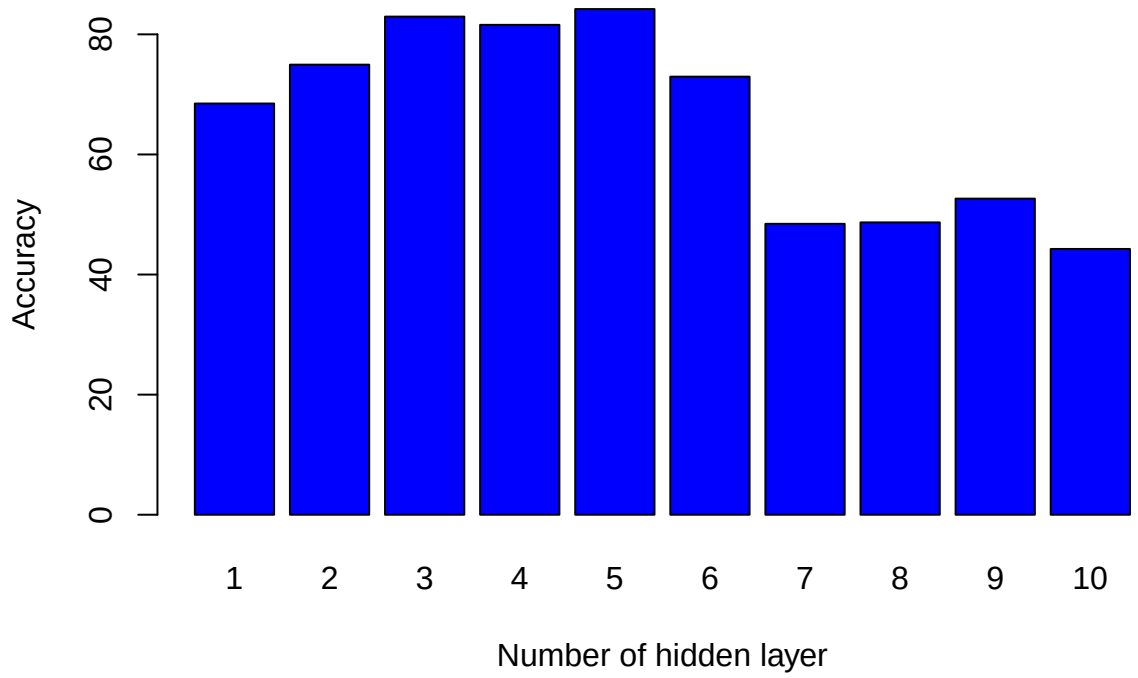
```
write.csv(heart, "~/HeartDisease/heart1.csv", row.names = FALSE)
write.csv(trainrand, "~/HeartDisease/trainrand.csv", row.names = FALSE)
write.csv(testrand, "~/HeartDisease/testrand.csv", row.names = FALSE)
write.csv(trainseq, "~/HeartDisease/trainseq.csv", row.names = FALSE)
write.csv(testseq, "~/HeartDisease/testseq.csv", row.names = FALSE)
```

Part 2:- Analysis of the neural network model

Plot for varying the number of hidden layers

Neural network with various number of hidden layers are tested and the results of the test accuracy are ploted.
Input layer contains **Thirteen** input neurons, each hidden layer has **Tweleve** neurons and the output layer has **One** neuron with **Two** classes.
- Class zero -> The patient is not suffering from heart disease.
- Class one -> The patient is suffering from heart disease.

Plot of test accuracy for different hidden layers



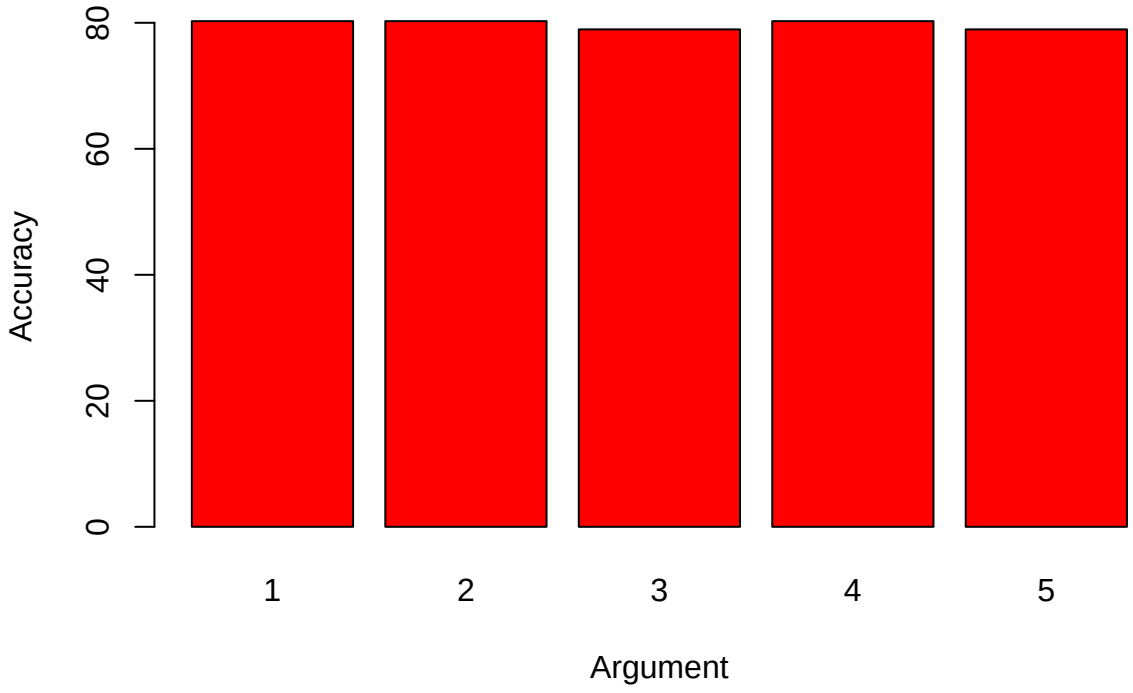
Conclusion from the plot

According to the test runs we can see that model with **Three**, **Four**, **Five** and **Six** hidden layers, having **Tweleve** neuron each perform better than the other choosen models.

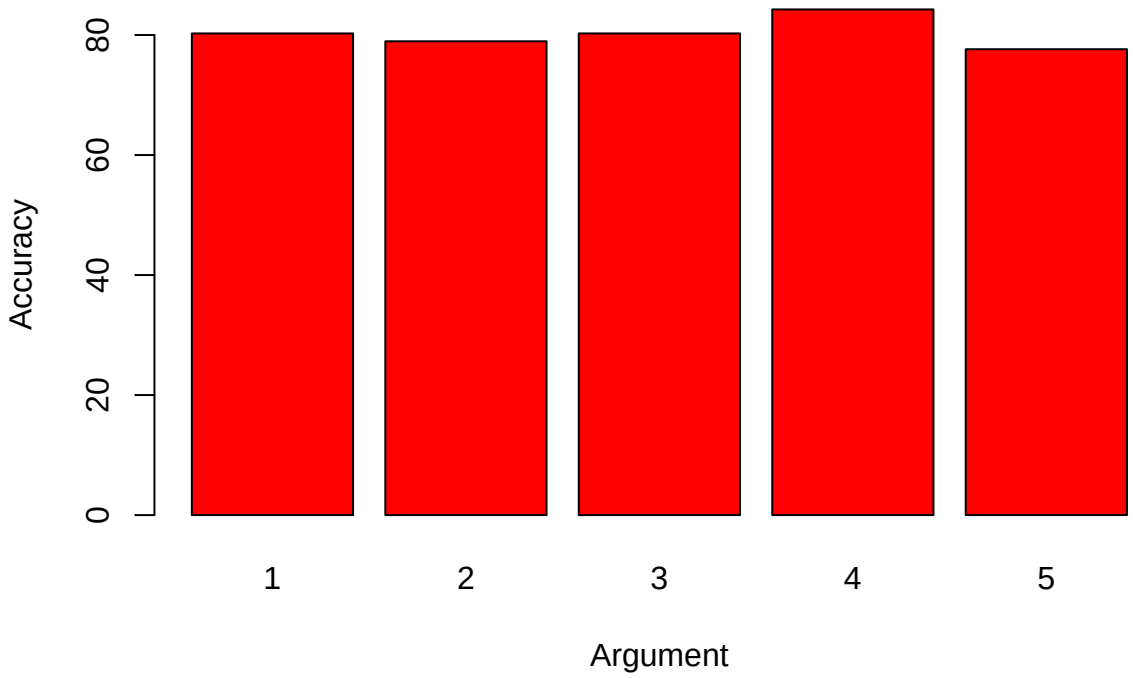
Plot for varying the number of neurons

For each hidden layer, five samples are taken for testing and it's accuracy is plotted below

For three hidden layers Argument numbers and the structure associated with it.
1. 13-8-8-8-1
2. 13-10-10-10-1
3. 13-8-12-12-1
4. 13-10-10-12-1
5. 12-10-12-12-1

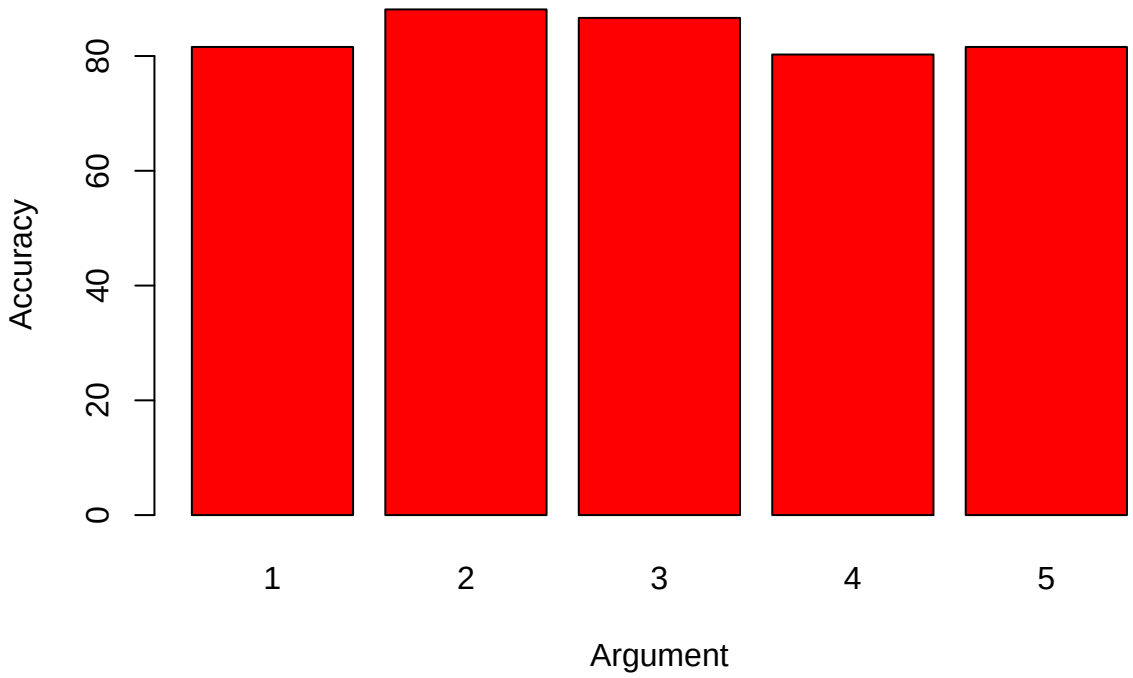


For four hidden layers Argument numbers and the structure associated with it.
1. 13-8-8-8-8-1
2. 13-12-12-12-12-1
3. 13-8-8-10-10-1
4. 13-12-12-8-8-1
5. 13-12-12-10-10-1



For five hidden layers Argument numbers and the structure associated with it.

1. 13-8-8-8-8-8-1
2. 13-12-12-10-8-8-1
3. 13-12-10-10-10-8-1
4. 13-8-8-10-8-8-1
5. 13-8-8-12-8-8-1



Conclusion from the plot

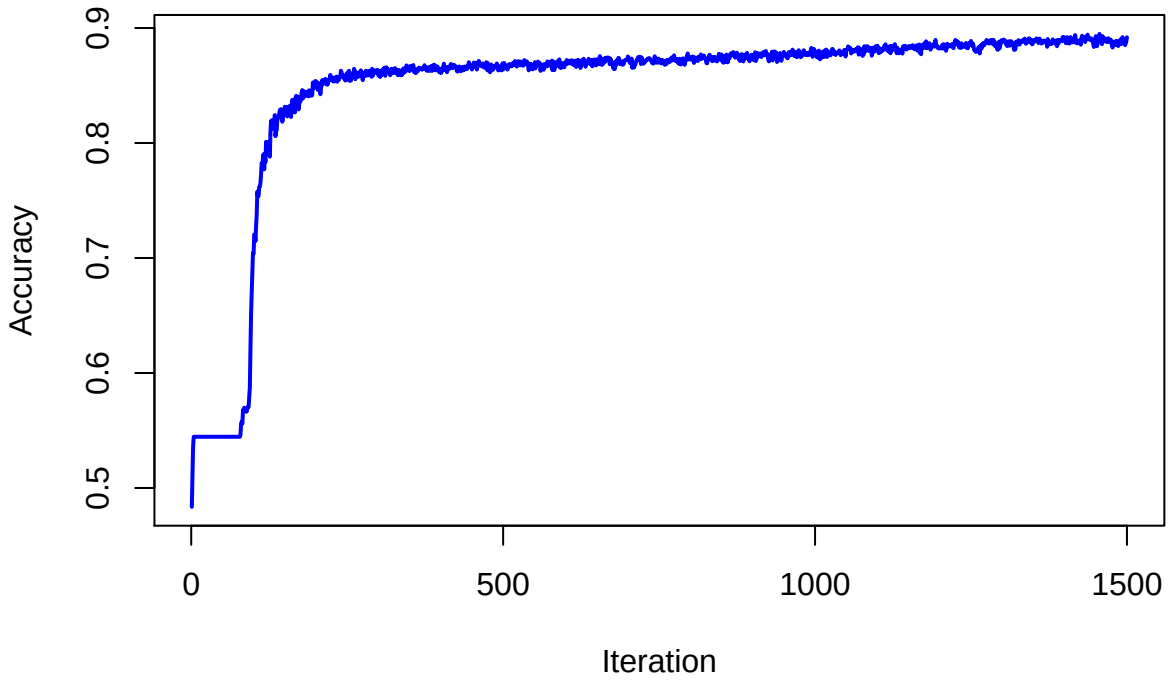
According to the test runs we can see that the models with structure - Input layer -> 13 input neuron
- First hidden layer -> 12 hidden neuraon
- Second hidden layer -> 12 hidden neuraon
- Third hidden layer -> 10 hidden neuraon
- Fourth hidden layer -> 8 hidden neuraon
- Fifth hidden layer -> 8 hidden neuraon
- Output layer -> 1 neuron, 2 classes

gives the best output and so is taken as the final structure of the neural network.

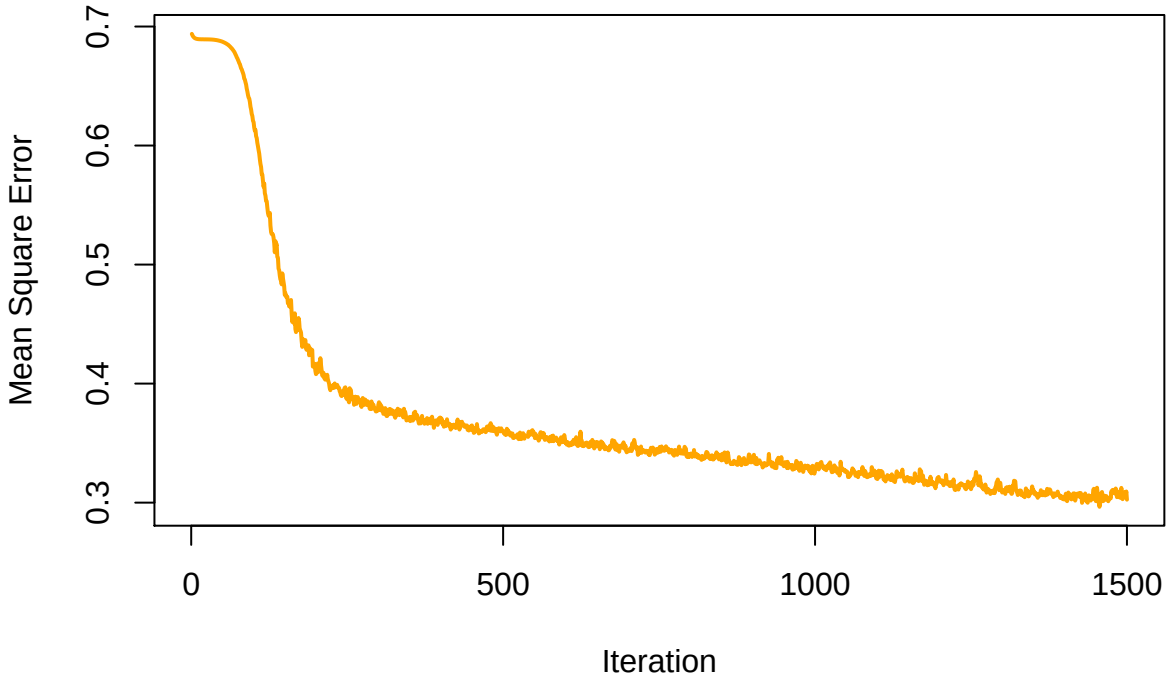
Plot for final accepted structure

The neural network with the best results is run **Twinty** times and the average mean square error, and accuracy was taken and plot in the following graph.

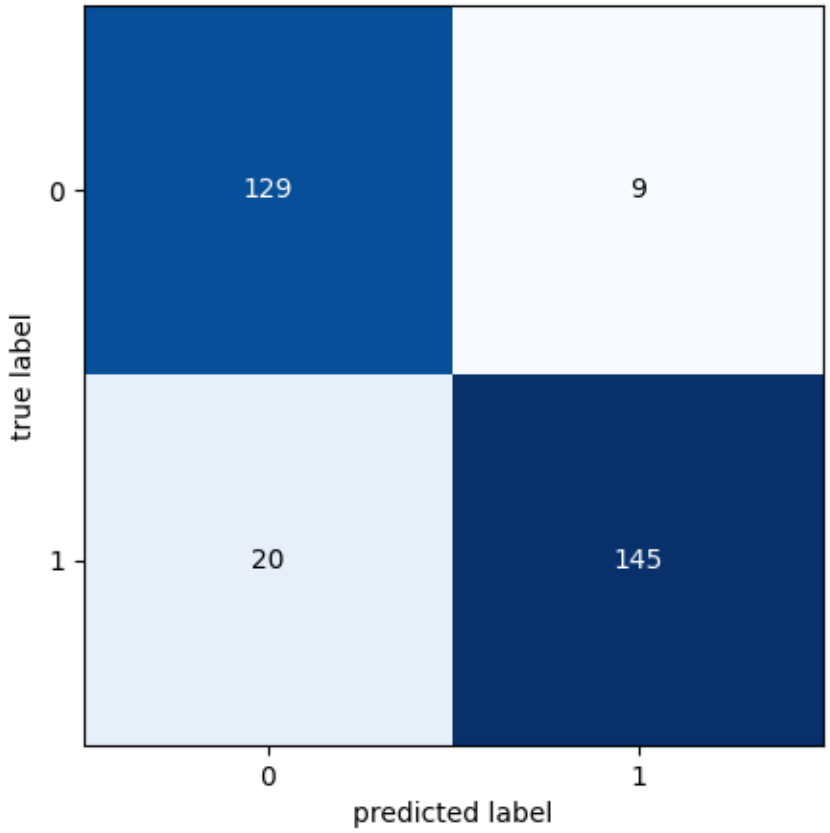
Plot for Accuracy of the model



Plot for Mean Square Error of the model



Confusion matrix and performance analysis



Performance matrix analysis

```
precision <- TP/(TP+FP)
recall <- TP/(TP+FN)
sensitivity <- TP/(TP+FN)
specificity <- TN/(TN+FP)
FNR <- FN/(FN+TP)
F_score <- (2*precision*recall)/(precision+recall)
MCC <- ((TP*TN)-(FP*FN)) / (sqrt((TP+FP)*(TP+FN)*(TN+FP)*(TN+FN)))
print(precision)
```

```
## [1] 0.9415584
```

```
print(recall)
```

```
## [1] 0.5291971
```

```
print(sensitivity)
```

```
## [1] 0.5291971
```

```
print(specificity)
```

```
## [1] 0.6896552
```

```
print(FNR)
```

```
## [1] 0.4708029
```

```
print(F_score)
```

```
## [1] 0.6775701
```

```
print(MCC)
```

```
## [1] 0.1287869
```