**BUSINESS DATA MINING**

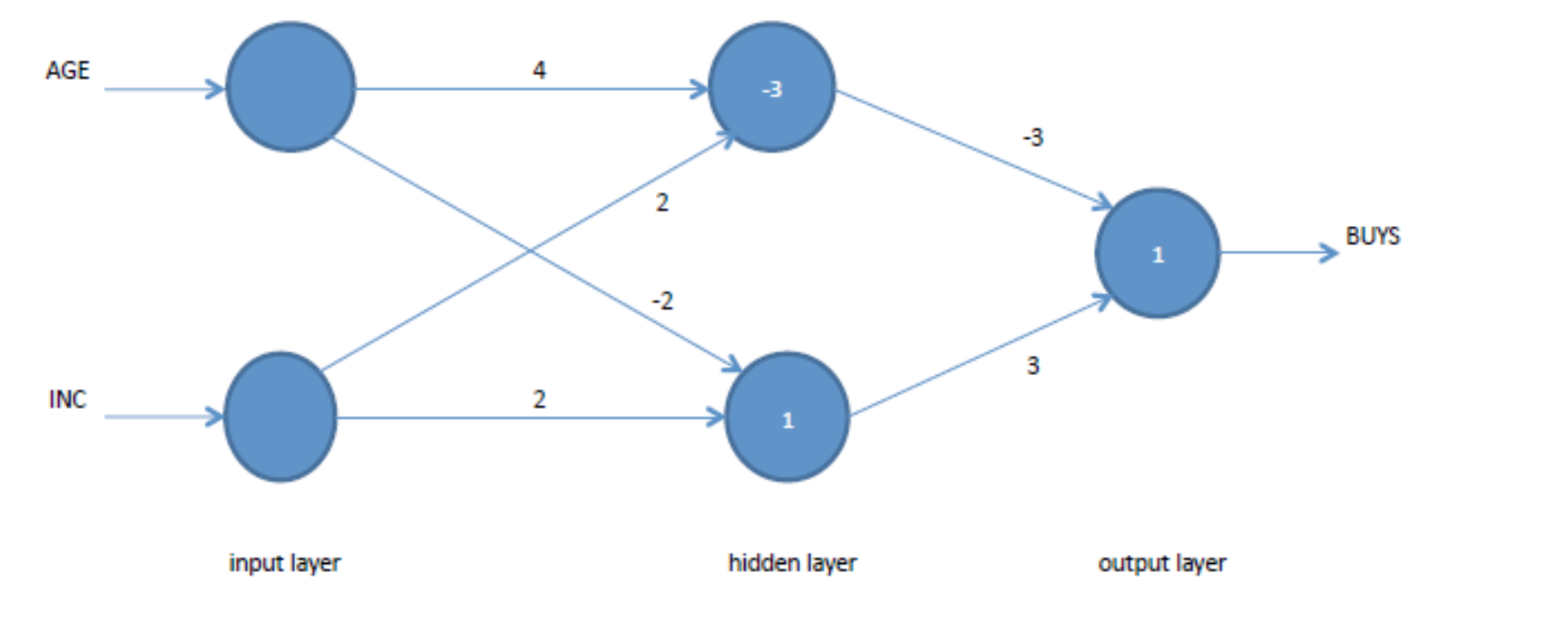
**(IDS 572)**

**Solutions to Homework 7**

**Group Members**

* Amey Pophali (apopha2@uic.edu)
* Karthik Varanasi (vvaran3@uic.edu)
* Mrinal Dhawan (mdhawa3@uic.edu)

Given Neural Net is:



Age range = 20 -80

INC range = 10K – 110 K

1. Normalized inputs:

For normalization using Min-Max function,

Input AGE – 70 Incomes - 50 K

Normalized Age = [ AGE – min(AGE) / max(AGE) -min(AGE) ]

= (70 – 20)/ (80-20)

= 50/60 = 0.8333334

Normalized INC = [ INC – min(INC) / max(INC) -min(INC) ]

= (50 – 10)/ (110-10)

= 40/100

= 0.4

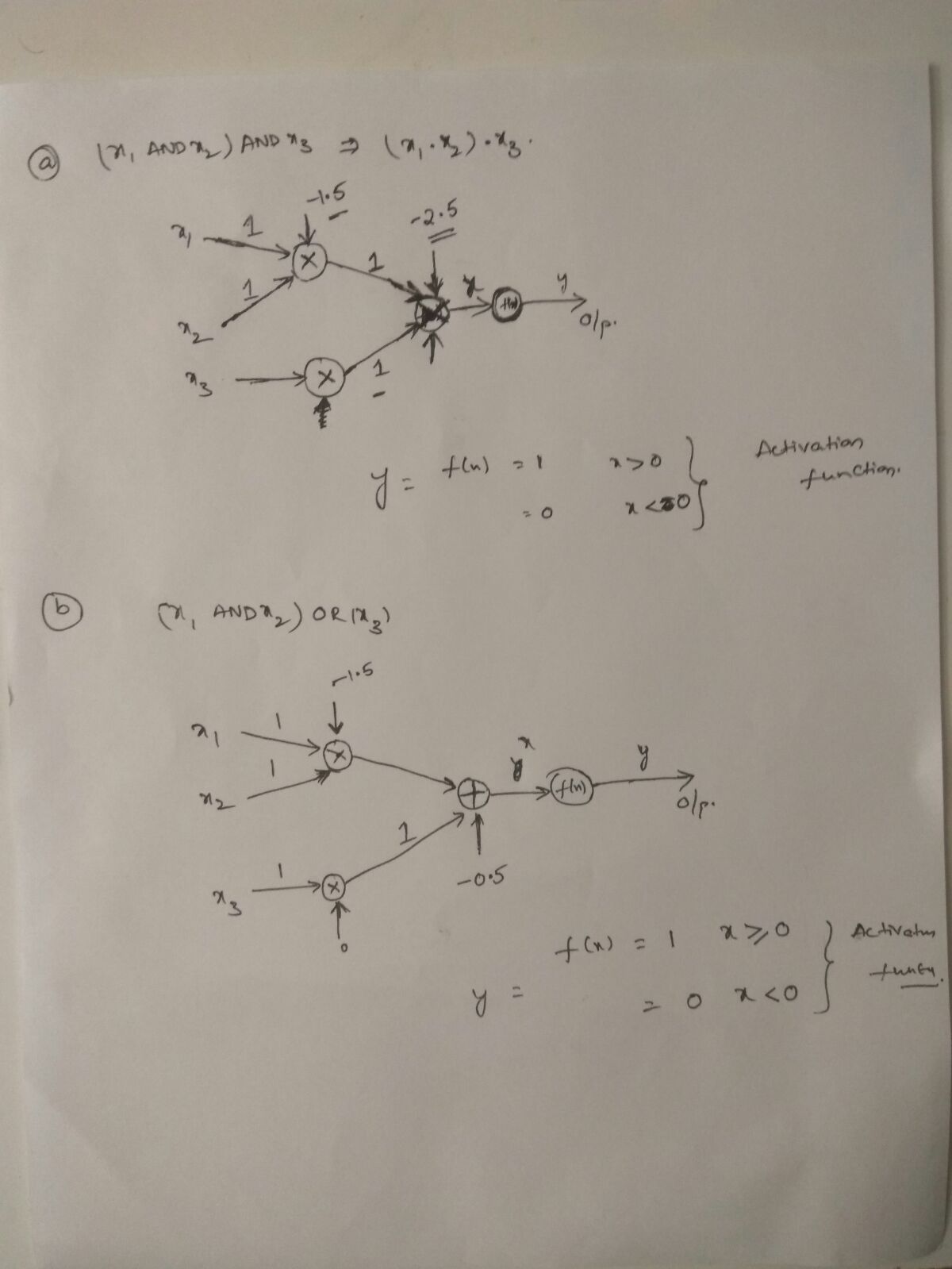
1. (X1 and X2) AND X3

|  |  |  |  |
| --- | --- | --- | --- |
| X1 | X2 | X3 | RESULT (Y) |
| 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 |
| 1 | 1 | 0 | 0 |
| 0 | 0 | 1 | 0 |
| 1 | 0 | 1 | 0 |
| 0 | 1 | 1 | 0 |
| 1 | 1 | 1 | 1 |

Optimal Weights:

Y= w1\*x1+w2\*x2+w3\*x3 + b where b= combined bias

Solving based on the perceptron algorithm, we get the network as

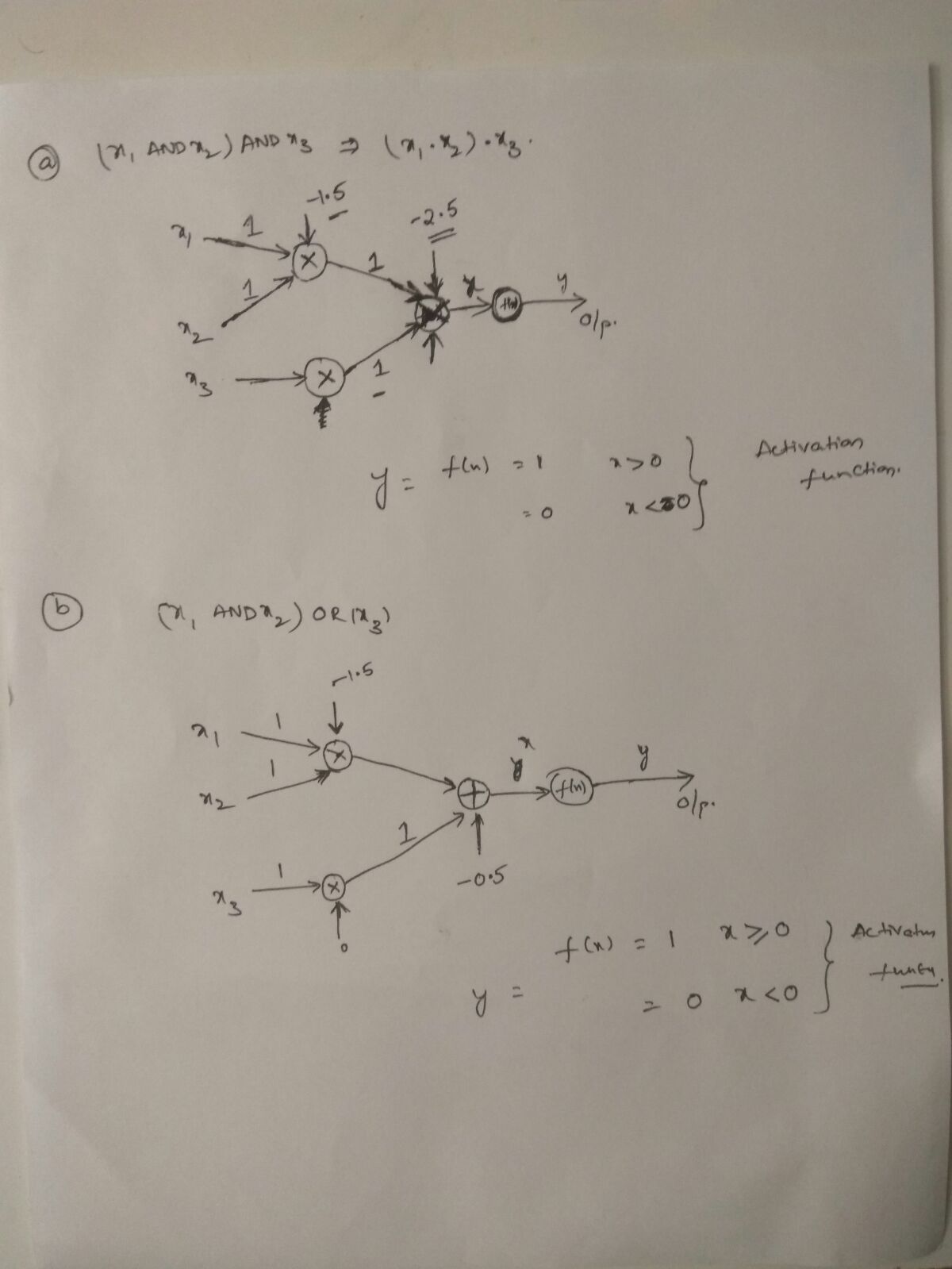


(X1 AND X2) OR X3

|  |  |  |  |
| --- | --- | --- | --- |
| X1 | X2 | X3 | RESULT (Y) |
| 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 |
| 1 | 1 | 0 | 1 |
| 0 | 0 | 1 | 1 |
| 1 | 0 | 1 | 1 |
| 0 | 1 | 1 | 1 |
| 1 | 1 | 1 | 1 |

Y= w1\*x1+w2\*x2+w3\*x3 + b where b= combined bias

Solving based on perceptron algorithm, we get the following neural net as out output.



2. This can be achieved by using the command VLOOKUP in MS Excel. The command enables us to look through the required excel columns and populate the values against them. The resultant data file is attached here.



*D <- read.csv("D:/UIC Fall/Data Mining/HW/7/D.csv")*

*#change relevant ones to Factor*

*D$OUTCOME = factor(D$OUTCOME)*

*D$First\_home = factor(D$First\_home)*

*D$UPB.Appraisal = factor(D$UPB.Appraisal)*

Start with removing the State code and Status variables from the data set as these variables are redundant. We do this by using the command -

*D = D[c(-11,-13)]*

Checking for outliers - We need to replace all the outliers for all the variables with the mean of the variables. We can do this using the commands provided in the text file.



1. We can get the summary of the data using the command ‘summary(D)’.

The summary is attached in the text file.



As we can see, the outliers have been removed from the data. Additionally, we can see that there are high number of ‘non-default’ values in the target variable OUTCOME. Going forward, we would have to handle this part of the data.

1. One of the ways of checking an insignificant variable is to check for the standard deviation of the variables. If the SD is less, it signifies the values are very close to each other and are not very significant. But by looking at the SDs of all the variables in the given dataset, there is no variable that we can replace with this method.

> sd(D$Orig\_LTV\_Ratio\_Pct)

[1] 8.853716

> sd(D$Credit\_score)

[1] 61.62902

> sd(D$First\_home)

[1] 0.4479125

> sd(D$median.income....)

[1] 5431.122

> sd(D$X..of.people.in.poverty)

[1] 2.265902

> sd(D$LoanValuetoAppraised)

[1] 3951.534

> sd(D$Tot\_mthly\_debt\_exp)

[1] 841.6633

> sd(D$Tot\_mthly\_incm)

[1] 1846.191

> sd(D$orig\_apprd\_val\_amt)

[1] 70681.73

> sd(D$pur\_prc\_amt)

[1] 67523.09

> sd(D$DTI.Ratio)

[1] 0.1554496

> sd(D$UPB.Appraisal)

[1] 0.4976226

Another way to verify if the variables in the data are significant or not is to check the correlation between the variables. If the correlation between any two variables is high, we can exclude one of them from our models. We can check the correlation between numerical models by removing the categorical variables then applying the command - COR.

*#Removing the categorical from the dataset*

*D2=D[c(-5,-11,-12)]*

*cor\_d=cor(D2)*

*View(cor\_d)*

Output -

**

Looking at the output, we can see that the co-relation between pur\_prc\_amt and orig\_apprvd\_val\_amt stands at 0.92, which is quite high. Hence we can exclude one of these variables from the dataset. We remove it by the following command -

*D = D[c(-9)]*

With the mentioned command, we excluded the variable ‘pur\_prc\_amt’ from the dataset.

1. Setting the seed to 2014 and splitting the data into 75% of training data and 25% of test data. This can be done using the commands -

*#Split data into 75% Training and 25% testing*

*E = sample(2,nrow(D),replace=T,prob=c(0.75,0.25))*

*Train = D[E==1,]*

*Test = D[E==2,]*

*table(Train$OUTCOME)*

*table(Test$OUTCOME)*

*str(Train)*

*str(Test)*

*Output of the above commands is attached here*

**

To make the data balanced, we under-sample the training data to obtain two training datasets.

The first one will have 30% default cases (TrgA) and the second one with 10% default cases (TrgB). This is done using the commands -

*#Install and Load libraries for Ovum*

*install.packages("ROSE")*

*library(ROSE)*

*#Sampling Traning data to contain 30% and 10% default 'OUTCOME' cases*

*TrgA <- ovun.sample(OUTCOME~., data = Train, method = "both", p=0.3, N=1000, seed = 2014)$data*

*table(TrgA$OUTCOME)*

*TrgB <- ovun.sample(OUTCOME~., data = Train, method = "both", p=0.1, N=1000, seed = 2014)$data*

*table(TrgB$OUTCOME)*

The output of the commands -



DECISION TREE MODELS

We plot the decision trees on the training data TrgA and TrgB. Commands -

*#Plot RTree on TrgA*

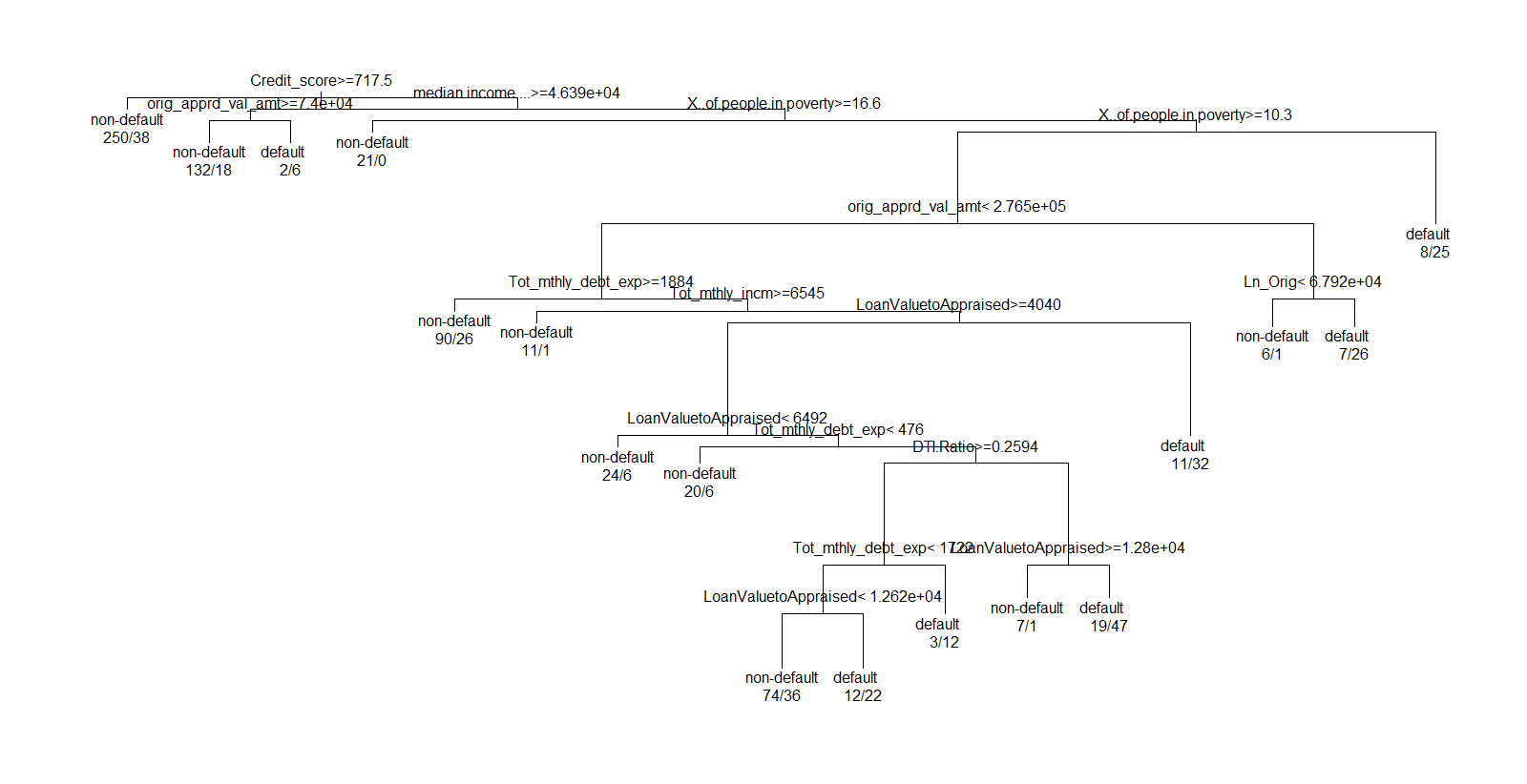
*r\_tree\_TrgA = rpart(TrgA$OUTCOME~.,data=TrgA,method="class")*

*print(r\_tree\_TrgA)*

*plot(r\_tree\_TrgA)*

*text(r\_tree\_TrgA, use.n = T, xpd= T)*

The tree -



*Confusion matrix of training data TrgA*

*#Predicts rtree on TrgA*

*predA = predict(r\_tree\_TrgA,type="class")*

*table(predA,TrgA$OUTCOME)*

The confusion matrix on of TrgA output

*table(predA,TrgA$OUTCOME)*

predA non-default default

non-default 635 133

default 62 170

Checking Output of TrgA decision tree on Test Data

*table(predict(r\_tree\_TrgA, type = "class", newdata = Test), Test$OUTCOME)*

default non-default

non-default 92 3330

default 20 429

Accurcacy of the TrgA deiciosn tree on Test Data

= (3330+20)/(3330+92+20+429)

= 3350/3871

**= 86.5%**

*#Plot RTree on TrgB*

*r\_tree\_TrgB = rpart(OUTCOME~.,data=TrgB,method="class")*

*print(r\_tree\_TrgB)*

*plot(r\_tree\_TrgB)*

*text(r\_tree\_TrgB, use.n = T, xpd= T)*

Confusion matrix of training data TrgB

*#Predicts rtree on TrgB*

*predB = predict(r\_tree\_TrgB,type="class")*

*table(predB,TrgB$OUTCOME)*

The confusion matrix on of TrgB output

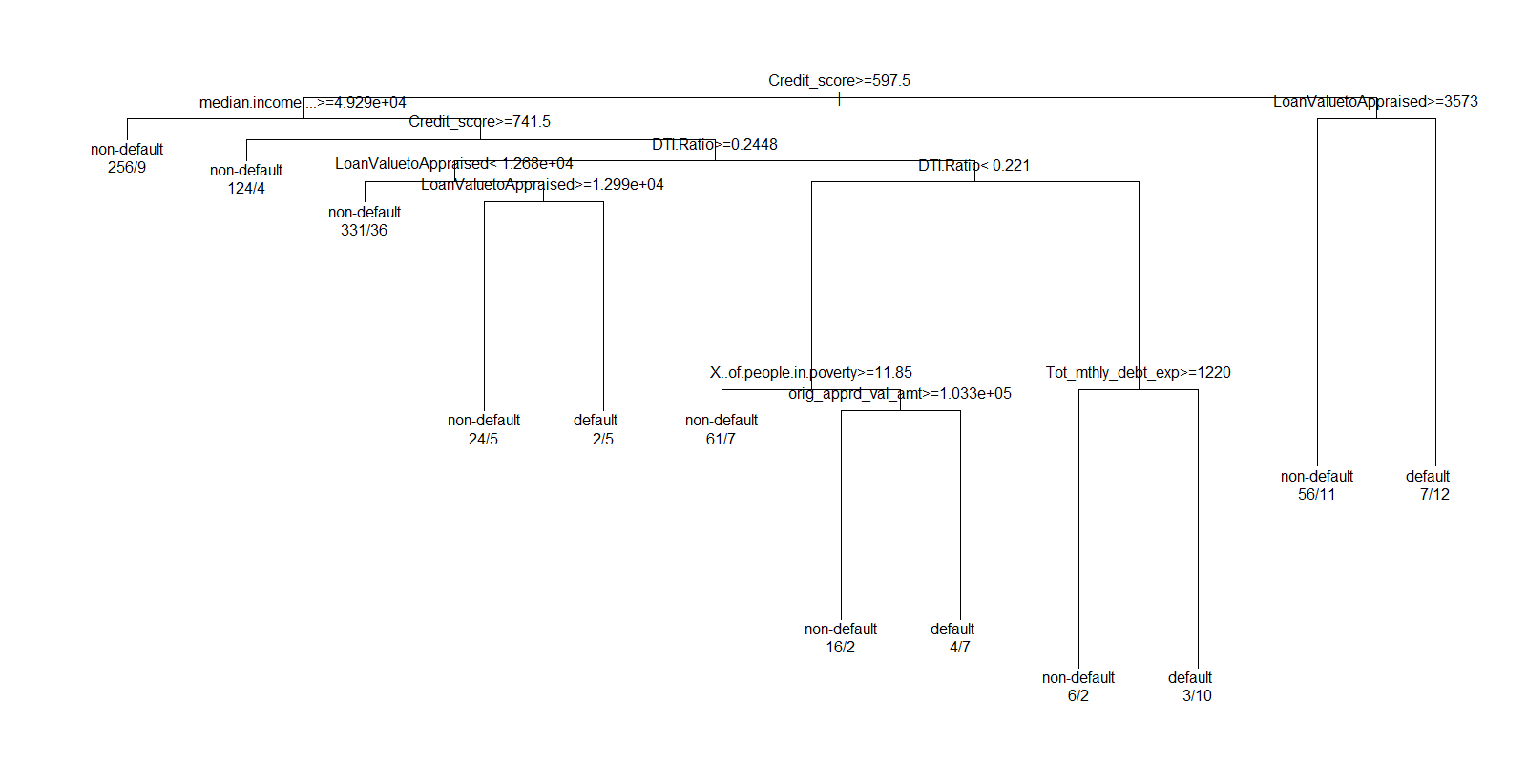
table(predB,TrgB$OUTCOME)

predB non-default default

non-default 874 76

default 16 34

*The Rtree for TrgB -*



> table(predict(r\_tree\_TrgB, type = "class", newdata = Test), Test$OUTCOME)

default non-default

non-default 110 3712

default 2 47

Accuracy = 3712+2 / 110+2+3712+47

= 3714/3871

= 95.9 %

**The accuracy of the TrgA and TrgB are 86.5% and 95.9% respectively.**

LOGISTICS REGRESSION

On Training Data TrgA -

*#Commands for Logistics Regression on TrgA*

*Logis\_TrgA = glm(OUTCOME~., data = TrgA, family = "binomial")*

*summary(Logis\_TrgA)*

*exp(coef(Logis\_TrgA))*

*Output -*

**

As per the output, we can conclude that the p-value for the variables Ln\_orig, Orig\_LTV\_Ratio\_Pct, First\_homeY, orig\_apprd\_val\_amt is quite high. Hence these variables are insignificant.

We run the model on the test data using the following commands and make the confusion matrix

*#Predict Test data on the Model for TrgA*

*TestA = predict(Logis\_TrgA, newdata = Test, type = "response")*

*predA = rep('default',length(TestA))*

*predA[TestA>=0.5] <-'non-default'*

*confusionMatrix(predA, Test$OUTCOME, dnn = c("Predictions", "Actual Values"))*

Output

Confusion Matrix and Statistics

Actual Values

Predictions default non-default

default 93 3662

non-default 0 0

Accuracy : 0.0248

95% CI : (0.02, 0.0303)

No Information Rate : 0.9752

P-Value [Acc > NIR] : 1

Kappa : 0

Mcnemar's Test P-Value : <2e-16

Sensitivity : 1.00000

Specificity : 0.00000

Pos Pred Value : 0.02477

Neg Pred Value : NaN

Prevalence : 0.02477

Detection Rate : 0.02477

Detection Prevalence : 1.00000

Balanced Accuracy : 0.50000

'Positive' Class : default

As per the matrix, the accuracy of the model on the test data is 2%. As per the accuracy of the model.

Running Logistic Regression on TrgB data -

Commands -

*#Commands for Logistics Regression on TrgB*

*Logis\_TrgB = glm(OUTCOME~., data = TrgB, family = "binomial")*

*summary(Logis\_TrgB)*

*exp(coef(Logis\_TrgB))*

*Output -*

**

As per the output, we can conclude that the p-value for the variables LoanValuetoAppraised, Orig\_LTV\_Ratio\_Pct, orig\_apprd\_val\_amt is quite high. Hence these variables are insignificant.

Running the model on test data and creating the confusion matrix -

Commands -

*#Predict Test data on the Model for TrgB*

*TestB = predict(Logis\_TrgB, newdata = Test, type = "response")*

*predB = rep('default',length(TestB))*

*predB[TestB>=0.5] <-'non-default'*

*confusionMatrix(predB, Test$OUTCOME, dnn = c("Predictions", "Actual Values"))*

Output -

Confusion Matrix and Statistics

Actual Values

Predictions default non-default

default 93 3662

non-default 0 0

Accuracy : 0.0248

95% CI : (0.02, 0.0303)

No Information Rate : 0.9752

P-Value [Acc > NIR] : 1

Kappa : 0

Mcnemar's Test P-Value : <2e-16

Sensitivity : 1.00000

Specificity : 0.00000

Pos Pred Value : 0.02477

Neg Pred Value : NaN

Prevalence : 0.02477

Detection Rate : 0.02477

Detection Prevalence : 1.00000

Balanced Accuracy : 0.50000

'Positive' Class : default

We can see that the accuracy of the model on Test Data Is 2.48%. Hence we can summarize that this is not a good model.

NEURAL NETWORKS

Firstly, we have to normalize the numerical variables in the dataset to plot the neural networks.

Creating Neural Network on training data TrgA -

*#Neural Networks*

*install.packages("ISLR")*

*library(ISLR)*

*# Create vector of column Max and Min values*

*maxs = apply(TrgA[,c(1:4, 6:9, 12:14)], 2, max)*

*mins = apply(TrgA[,c(1:4, 6:9, 12:14)], 2, min)*

*# Use scale() and convert the resulting matrix to a data frame*

*scaled.data = as.data.frame(scale(TrgA[,c(1:4, 6:9, 12:14)], center = mins, scale = maxs - mins))*

*head(scaled.data)*

*summary(scaled.data)*

*#Adding Outcome to the scaled data set*

*OUTCOME = TrgA$OUTCOME*

*data = data.frame(OUTCOME,scaled.data)*

*head(data)*

*The normalized data output is attached here -*

**

*Post normalization of the data, we go ahead and construct the neural network using TrgA.*

Commands used are -

*#Constructing neural network*

*library(nnet)*

*nn = nnet(OUTCOME ~ ., data=TrgA, linout=F, size=10, decay=0.01, maxit=1000)*

*#Neural network model out put*

*summary(nn)*

*#To plot the neural network using nnet we need to use devtools*

*library(devtools)*

*source\_url('https://gist.githubusercontent.com/fawda123/7471137/raw/466c1474d0a505ff044412703516c34f1a4684a5/nnet\_plot\_update.r')*

*plot.nnet(nn)*

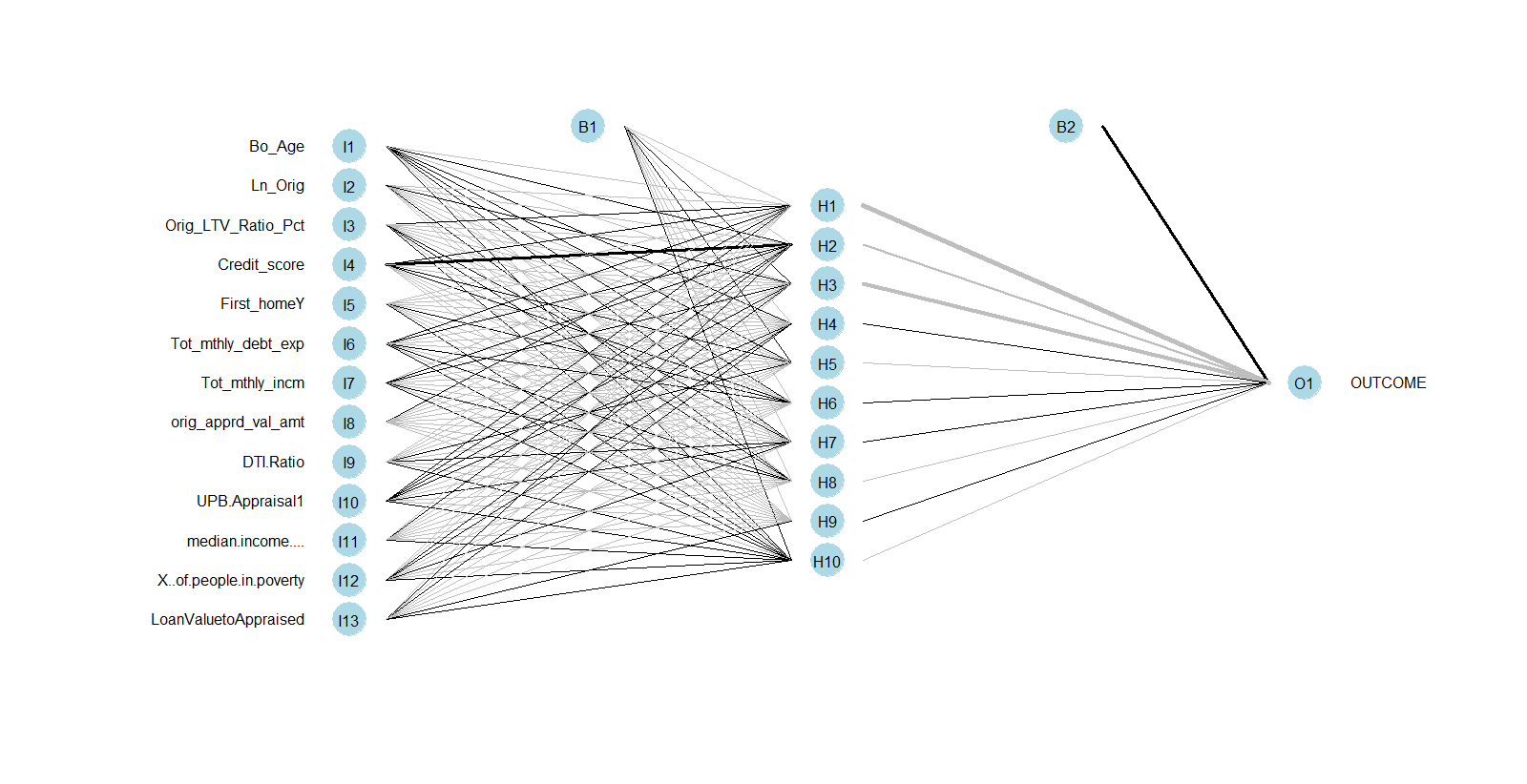
*#To check the weight values*

*nn$wts*

*nn$fitted.values*

Output of the above commands is attached here -

**

*NN Plot* 

*We predict the performance of the neural network on the Test data ‘Test’.*

*# Using nnet model on the test data*

*nn.preds = predict(nn, Test, type = "class")*

*table(Test$OUTCOME, nn.preds)*

*Output -*

> table(Test$OUTCOME, nn.preds)

nn.preds

default non-default

default 13 89

non-default 480 3167

Accuracy = (13+3167)/(13+3167+480+89)

= 3180/3749 = 0.84

Hence Accuracy of the TrgA NN on test data is **84%.**

Plotting the Lift chart for TrgA model

install.packages("ROCR")

library(ROCR)

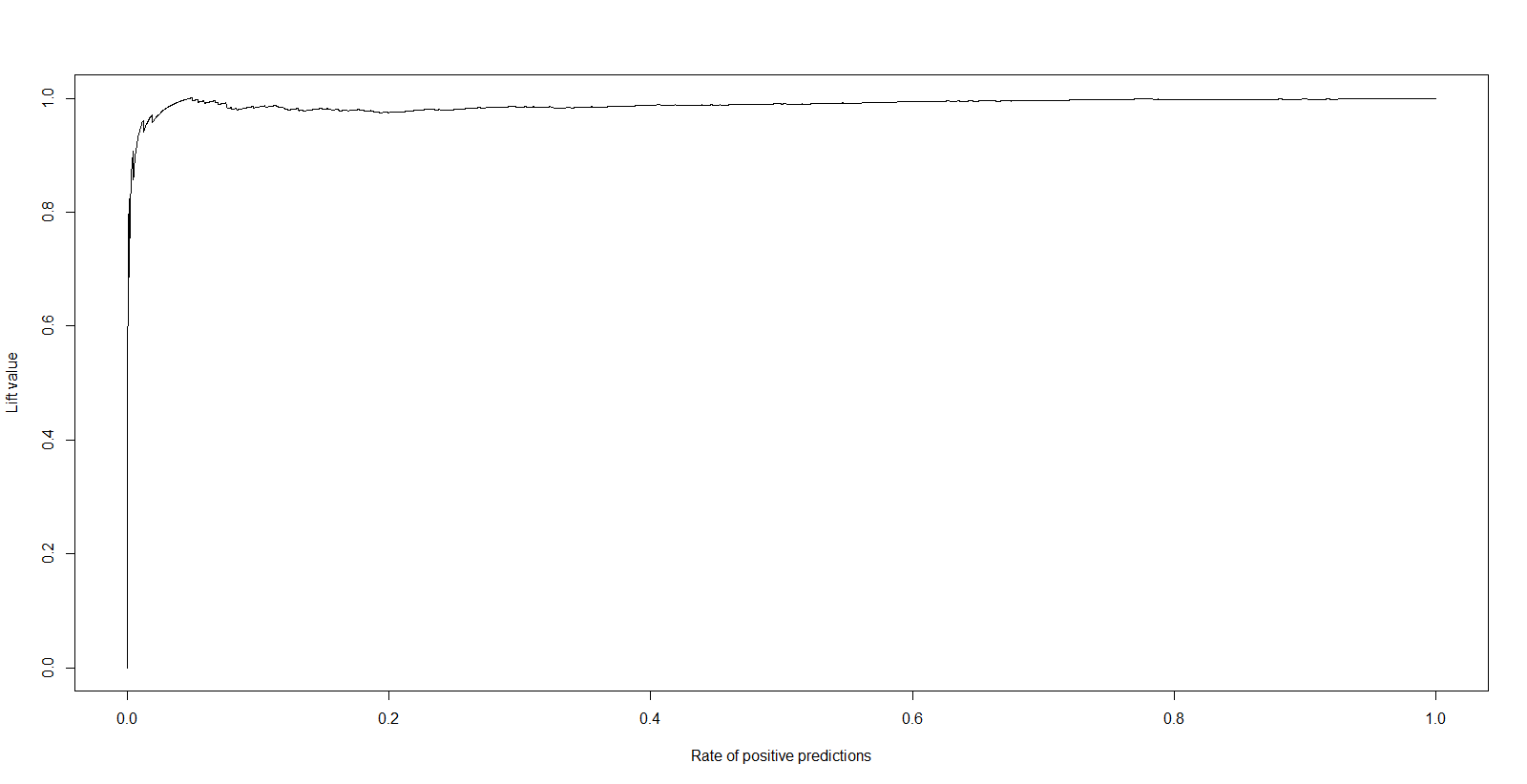
predLA = predict(nn,newdata=Test,type="raw")

predLiftA = prediction(predLA, Test$OUTCOME)

perfLA = performance(predLiftA,"lift","rpp")

plot(perfLA)

Lift Chart



Normalizing the data and Creating Neural Network on Trg B

*# Create vector of column Max and Min values*

*maxs = apply(TrgB[,c(1:4, 6:9, 12:14)], 2, max)*

*mins = apply(TrgB[,c(1:4, 6:9, 12:14)], 2, min)*

*# Use scale() and convert the resulting matrix to a data frame*

*scaled.data = as.data.frame(scale(TrgB[,c(1:4, 6:9, 12:14)], center = mins, scale = maxs - mins))*

*head(scaled.data)*

*summary(scaled.data)*

*#Adding Outcome to the scaled data set*

*OUTCOME = TrgB$OUTCOME*

*data = data.frame(OUTCOME,scaled.data)*

*head(data)*

*#Constructing neural network*

*library(nnet)*

*nn = nnet(OUTCOME ~ ., data=TrgB, linout=F, size=10, decay=0.01, maxit=1000)*

Output -



Plotting the Neural Network -

*#Constructing neural network*

*library(nnet)*

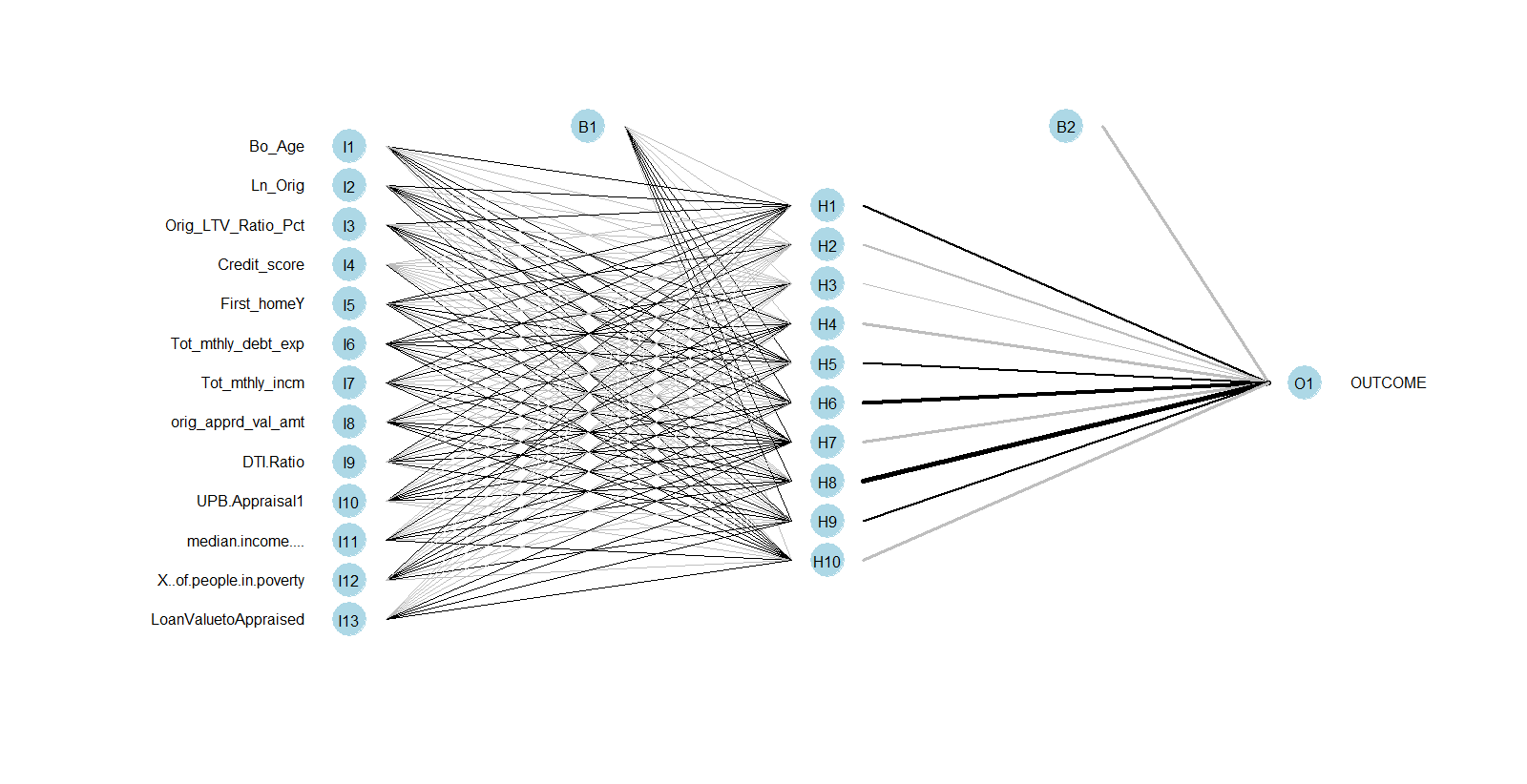
*nn = nnet(OUTCOME ~ ., data=TrgB, linout=F, size=10, decay=0.01, maxit=1000)*

*#Neural network model out put*

*summary(nn)*

Summary -





Predicting output of NN of TrgB on Test Data

*# Using nnet model on the test data:*

*nnB.preds = predict(nnB, Test, type = "class")*

Confusion Matrix

table(Test$OUTCOME, nnB.preds)

nnB.preds

default non-default

default 0 102

non-default 12 3635

Accuracy = 3635/(102+12+3635) = **96%**

Plotting the Lift Chart

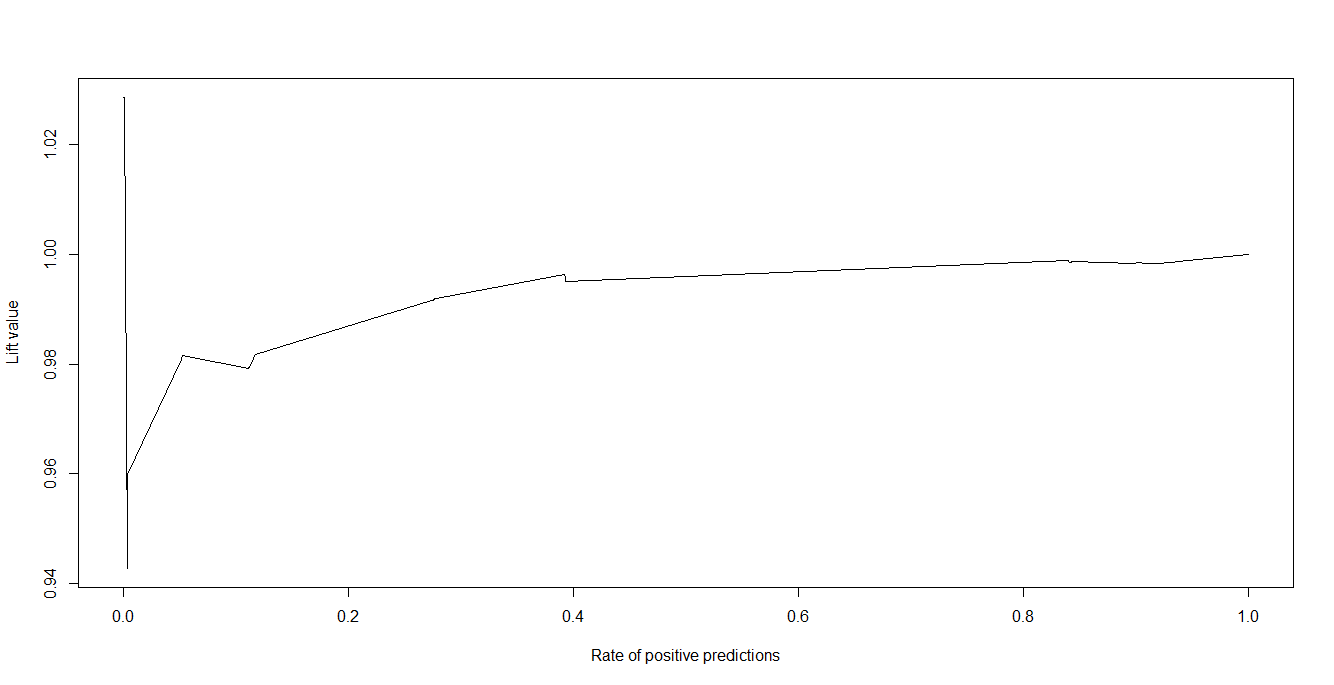
predLB = predict(nnB,newdata=Test,type="raw")

predLiftB = prediction(predLB, Test$OUTCOME)

perfLB = performance(predLiftB,"lift","rpp")

plot(perfLB)

Output -



**Comparing the Neural Networks -**

TrgA Neural Network has an accuracy rate of 84%. But this is a good model as it predicts some of the default values correctly.

TrgB Neural Network has an accuracy of 96%. But it is to be noted that it has predicted none of the default values correctly. Hence this model is not a good choice even with a high accuracy.

**Conclusion -**

Looking at the results, we can conclude that the best models that could be used to predict is

The Neural Network created with Traning Data TrgA. We reject the TrgB neural network because it doesn’t predict the default values correctly.