|  |
| --- |
| Photo displaying partial image of two pie charts on a canvas-textured page |
| Data Mining Assignment 4  Assignment Number - 4 |
| |  |  |  | | --- | --- | --- | | **Pooja Sheth** | **3/5/18** | **UW MSBA** | |

**Chapter 12**

For Exercises 12–14, use the data set churn. Normalize the numerical data, recode the categorical

variables, and deal with the correlated variables.

12. Generate a neural network model for classifying churn based on the other variables.

Describe the topology of the model.

## Scale data for neural network - normalize the numerical data, recode the categorical variables.

churn$Account.Length\_nm <- (churn$Account.Length - min(churn$Account.Length))/(max(churn$Account.Length)-min(churn$Account.Length))

churn$Vmail.Message\_nm <- (churn$Vmail.Message - min(churn$Vmail.Message))/(max(churn$Vmail.Message)-min(churn$Vmail.Message))

churn$Day.Mins\_nm <- (churn$Day.Mins- min(churn$Day.Mins))/(max(churn$Day.Mins)-min(churn$Day.Mins))

churn$Day.Calls\_nm <- (churn$Day.Calls- min(churn$Day.Calls))/(max(churn$Day.Calls)-min(churn$Day.Calls))

churn$Day.Charge\_nm <- (churn$Day.Charge- min(churn$Day.Charge))/(max(churn$Day.Charge)-min(churn$Day.Charge))

churn$EveMins\_nm <- (churn$EveMins- min(churn$EveMins))/(max(churn$EveMins)-min(churn$EveMins))

churn$Eve.Calls\_nm <- (churn$Eve.Calls- min(churn$Eve.Calls))/(max(churn$Eve.Calls)-min(churn$Eve.Calls))

churn$Eve.Charge\_nm <- (churn$Eve.Charge- min(churn$Eve.Charge))/(max(churn$Eve.Charge)-min(churn$Eve.Charge))

churn$Night.Mins\_nm <- (churn$Night.Mins- min(churn$Night.Mins))/(max(churn$Night.Mins)-min(churn$Night.Mins))

churn$Night.Calls\_nm <- (churn$Night.Calls- min(churn$Night.Calls))/(max(churn$Night.Calls)-min(churn$Night.Calls))

churn$Night.Charge\_nm <- (churn$Night.Charge- min(churn$Night.Charge))/(max(churn$Night.Charge)-min(churn$Night.Charge))

churn$Intl.Mins\_nm <- (churn$Intl.Mins- min(churn$Intl.Mins))/(max(churn$Intl.Mins)-min(churn$Intl.Mins))

churn$Intl.Calls\_nm <- (churn$Intl.Calls- min(churn$Intl.Calls))/(max(churn$Intl.Calls)-min(churn$Intl.Calls))

churn$Intl.Charge\_nm <- (churn$Intl.Charge- min(churn$Intl.Charge))/(max(churn$Intl.Charge)-min(churn$Intl.Charge))

churn$CustServ.Calls\_nm <- (churn$CustServ.Calls- min(churn$CustServ.Calls))/(max(churn$CustServ.Calls)-min(churn$CustServ.Calls))

#creating dataframe with selected variables

newdat <- as.data.frame(churn[, c(21, 26:39 , 44)])

# Random sampling

set.seed(222)

index <- sample( 1:nrow( newdat ), 3000)

# creating training and test set

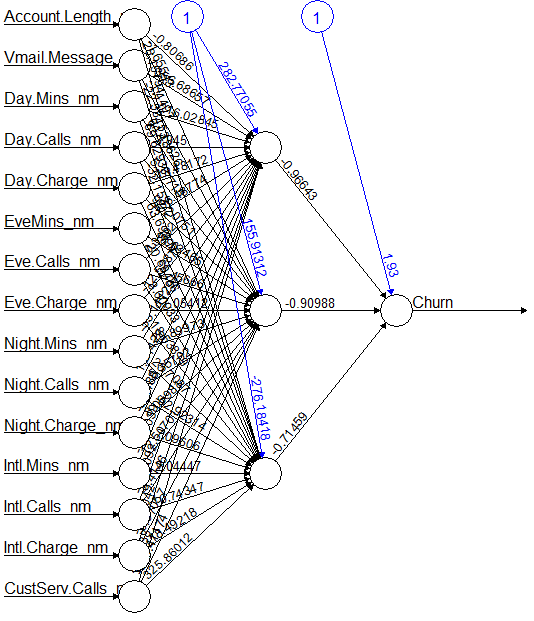
traindf <- newdat[index,]

testdf <-newdat [-index,]

# fit neural network

NN <-neuralnet(Churn ~ Account.Length\_nm + Vmail.Message\_nm+ Day.Mins\_nm + Day.Calls\_nm + Day.Charge\_nm + EveMins\_nm + Eve.Calls\_nm + Eve.Charge\_nm +Night.Mins\_nm + Night.Calls\_nm + Night.Charge\_nm + Intl.Mins\_nm + Intl.Calls\_nm + Intl.Charge\_nm + CustServ.Calls\_nm, traindf, hidden = 3 , linear.output = T)

plot(NN)



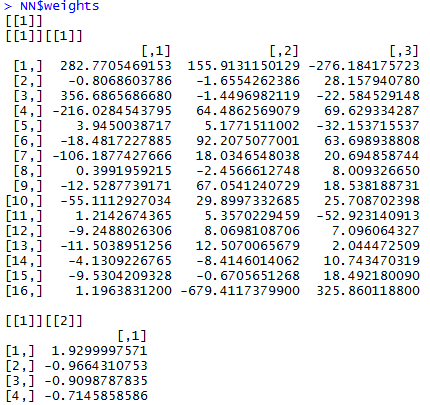
**Topology of the model:**

There are 15 input variables taken into account for the nerul network and they are :

Account.Length\_nm ,Vmail.Message\_nm, Day.Mins\_nm, Day.Calls\_nm, Day.Charge\_nm, EveMins\_nm, Eve.Calls\_nm, Eve.Charge\_nm, Night.Mins\_nm, Night.Calls\_nm, Night.Charge\_nm, Intl.Mins\_nm, Intl.Calls\_nm, Intl.Charge\_nm, CustServ.Calls\_nm.

There are 3 hidden nodes, and the training dataset is picked up for building the neural network.

The output variable is the tagret variable i.e. the churn variable and the associated weights are mentioned below.

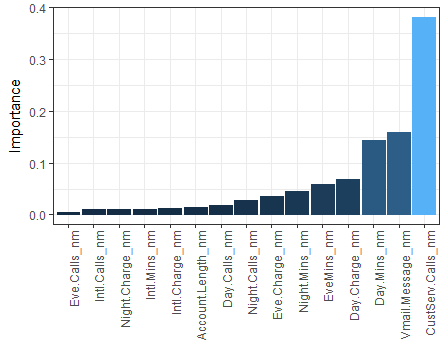


13. Which variables, in order of importance, are identified as most important for classifying

churn?

#NeuralNetTools

garson(NN)+theme(axis.text.x =element\_text(angle=90,hjust = 1))



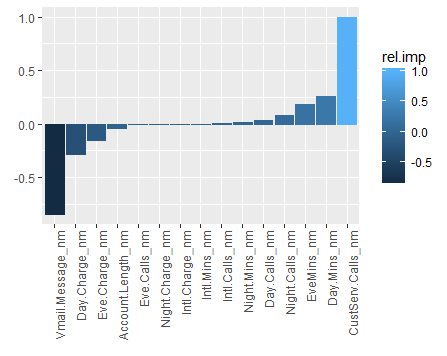
I have used two different methods to identify the important variables for the neural network model. Both give different outputs on the importance of the variables. The important variables from the above model are:

CustServ.Calls, Vmail.Message, Day.Mins, Day.Charge, EveMins, Night.Mins, Eve.Charge, Night.Calls, Day.Calls, Account.Length, Intl.Charge, Night.Charge, Intl.Calls, Eve.Calls.

#devtools

source\_gist('6206737' , filename = "gar\_fun.r")

gar.fun("predictorVars" , NN) +theme(axis.text.x =element\_text(angle=90,hjust = 1))



14. Compare the neural network model with the classification and regression tree (CART)

and C4.5 models for this task in Chapter 11. Describe the benefits and drawbacks of the

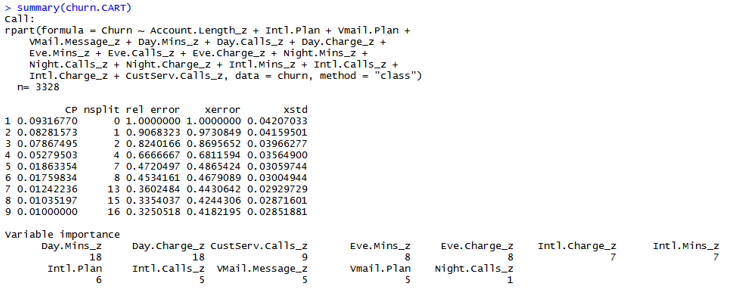
neural network model compared to the others. Is there convergence or divergence of results

among the models?

The way the variables are selected in all the three classification algorithms are different. The CART and the C5.0 are the decision tree algorithms and the Neural Network is a black box, it keeps on adjusting the weights in predicting the value for the target variable.

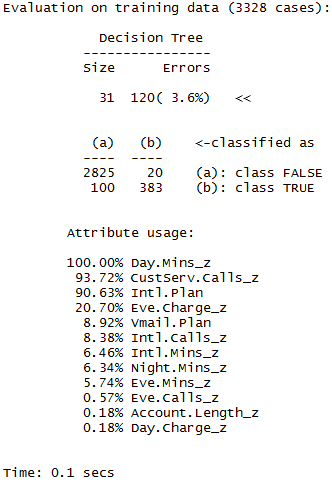
The CART on the one hand is the binary classification decision tree, whereas the C5.0 is not and can spilt into any number of nodes, determining the optimal nodes on splitting.

**CART method**



The CART method determines the variable importance to determine the optimal spilt on the root nodes, along with the length of the tree from the node.

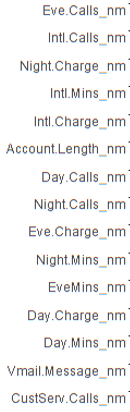
**C5.0 Method**



This method gives the variables importance in terms of the attribute usage. With Day.Mins being the root node with 100% usage and the attribute usage decreases so on.

**Neural Network**

The variables with their importance are categorized in the increasing order, with Eve.Calls\_nm of least importance and CustServ.Calls\_nm being the most important. It is based on the weights assigned to predict the target variable. As Nerual Netwrok is a black box the assignments of weights are what the algorithm self learns, by doing to and fro iterations couple of times. These weights help to decide which variables are more important.



**Chapter 15**

Use the churn data set at the book series website for the following exercises. Make sure that

the correlated variables have been accounted for.

27. Apply a CART model for predicting churn. Use default misclassification costs.

#Applied the default misclassification cost to the model – the model is build using the Train Data set.

costs<-list(loss=matrix(c(0,2,1,0), ncol=2, byrow=TRUE))

churn.CART <- rpart(Churn ~ Account.Length\_nm + Vmail.Message\_nm+ Day.Mins\_nm + Day.Calls\_nm + Day.Charge\_nm + EveMins\_nm + Eve.Calls\_nm + Eve.Charge\_nm +Night.Mins\_nm + Night.Calls\_nm + Night.Charge\_nm + Intl.Mins\_nm + Intl.Calls\_nm + Intl.Charge\_nm + CustServ.Calls\_nm, data = churn, method = "class")

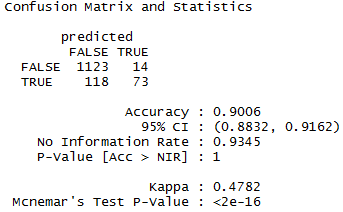
churn.CART.cost <- rpart(Churn ~ Account.Length\_nm +Vmail.Message\_nm + Day.Mins\_nm + Day.Calls\_nm + Day.Charge\_nm + EveMins\_nm + Eve.Calls\_nm +Eve.Charge\_nm + Night.Mins\_nm + Night.Calls\_nm + Night.Charge\_nm + Intl.Mins\_nm +Intl.Calls\_nm + Intl.Charge\_nm + CustServ.Calls\_nm, data = traindf, method = "class" ,parms = costs)

The train data set is used to predict the test data set.

predicted <- predict(churn.CART.cost , testdf , type = "class")

cm <- confusionMatrix(table(testdf$Churn , predicted))

The Confusion Matrix is constructed using the Predicted value and the Churn value of the test data set.



Construct a table containing the following measures:

1. Accuracy and overall error rate

The accuracy is calculated to be 0.9006.

> Error <- 1-Accuracy

> Error

[1] 0.0994

The accuracy of the model is quite high with 90.06% to classify the records as positive and negatives.

And the Error Rate being 9.94%.

> TN<-1123

> TP<-73

> FP<-14

> FN<-118

1. Sensitivity and false-positive rate

sensitivity<- TP/(TP+FN)

[1] 0.382199

The sensitivity of the model tells the ability to classify the record positively. Here the sensitivity of the model is 38.22%. So, it has a low percentage to classify the records positively that the customer would churn.

False\_Positive\_Rate<- 1- specificity

[1] 0.0123131

The False positive rate tells that the rate at which the model predicts the record to be true when it is false. Here it is 6.7% false positive rate which is low and good for the model.

1. Specificity and false-negative rate

specificity <- TN/(TN+FP)

0.9876869

The specificity is the ability to classify the record negatively. Here the specificity of the model is 98.76%. So, it has correctly predicted the customers that would not churn and do not churn.

False\_Negative\_Rate <- 1-sensitivity

0.617801

The False negative rate tells that the rate at which the model predicts the record to be false when it is true. Here it is 61.78% which is quite high, and it should be low. A higher value indicates that the model is incorrectly predicting the true values to be false. As predicting that they would not leave but they do, is a big risk.

1. Proportion of true positives and proportion of false positives

Proportion\_True\_Positives<- TP/(FP+TP)

0.8390805

Proportion of true positive is 83.9%, it is the probability that the customer is going to churn and does churn.

Proportion\_False\_Positives<-1- Proportion\_True\_Positives

0.1609195

Proportion of False Positives is 16.09%, it is the probability that the customer is not going to churn but we

predict that it would churn.

1. Proportion of true negatives and proportion of false negatives

Proportion\_True\_Negatives<- TN/(FN+TN)

0.9049154

Proportion of true negative is 90.49%, it is the probability that the customer is not going to churn, and the model does predict that it won’t churn.

Proportion\_False\_Negatives<- 1 -Proportion\_True\_Negatives

0.09508461

Proportion of False Negatives is 9.5%, it is the probability that the customer is going to churn but we

predict that it would not churn, which would be a big loss.

1. Overall model cost.

1123($0) +14($100) +118($300) +73($0) = $36,800

28. In a typical churn model, in which interceding with a potential churner is relatively cheap

but losing a customer is expensive, which error is costlier, a false negative or a false

positive (where positive =customer predicted to churn)? Explain.

A false positive error would be that the model classifies as the customer would churn but the customer doesn’t churn. This would not be a big loss. Rather, where a customer would churn but model predicts that the customer won’t churn would result in losing a customer. This would be the false negative classification of the model.

29. Based on your answer to the previous exercise, adjust the misclassification costs for your

CART model to reduce the prevalence of the costlier type of error. Rerun the CART

algorithm. Compare the false positive, false negative, sensitivity, specificity, and overall

error rate with the previous model. Discuss the trade-off between the various rates in terms

of cost for the company.

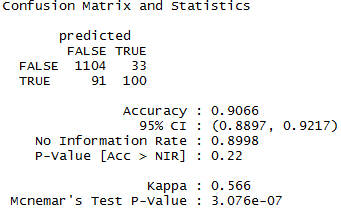
A cost matrix of 0,1,2,0 is used. Assigning cost of 0,0 to true negatives and true positives. And a cost of 1 to false negative and 2 to false positive.

costs<-list(loss=matrix(c(0,1,2,0), ncol=2, byrow=TRUE))

churn.CART.cost <- rpart(Churn ~ Account.Length\_nm +Vmail.Message\_nm + Day.Mins\_nm + Day.Calls\_nm + Day.Charge\_nm + EveMins\_nm + Eve.Calls\_nm +Eve.Charge\_nm + Night.Mins\_nm + Night.Calls\_nm + Night.Charge\_nm + Intl.Mins\_nm +Intl.Calls\_nm + Intl.Charge\_nm + CustServ.Calls\_nm, data = churn, method = "class" ,parms = costs)

churn$predicted3 <- predict(churn.CART.cost , data= churn , type = "class")

cm <- confusionMatrix(churn$predicted3 , churn$Churn)



1. Accuracy and overall error rate

The accuracy is calculated to be 0.9066

Error <- 1-Accuracy

0.0934

The accuracy of the model is quite high with 90.66% to classify the records as positive and negatives.

And the Error Rate being 9.34%.

The true negatives, true Positives, False negatives, False Positives are classified below.

TN<-1104

TP<-100

FP<-33

FN<-91

1. Sensitivity and false-positive rate

sensitivity<- TP/(TP+FN)

0.5235602

The sensitivity of the model tells the ability to classify the record positively. Here the sensitivity of the model is 52.35%. So, it has a low percentage to classify the records positively that the customer would churn, but still has improved from the previous model after applying the costs.

False\_Positive\_Rate<- 1- specificity

0.02902375

The False positive rate tells that the rate at which the model predicts the record to be true when it is false. Here it is 2.9% false positive rate which is low and good for the model.

1. Specificity and false-negative rate

specificity <- TN/(TN+FP)

0.9709763

The specificity is the ability to classify the record negatively. Here the specificity of the model is 97.09%. So, it has correctly predicted the customers that would not churn and do not churn.

False\_Negative\_Rate <- 1-sensitivity

0.4764398

The False negative rate tells that the rate at which the model predicts the record to be false when it is true. Here it is 47.64% which is not quite high, but it should be low, though it has decreased from the previous model. A higher value indicates that the model is incorrectly predicting the true values to be false. As predicting that they would not leave but they do, is a big risk.

1. Proportion of true positives and proportion of false positives

Proportion\_True\_Positives<- TP/(FP+TP)

0.7518797

Proportion of true positive is 75.18%, it is the probability that the customer is going to churn and does churn. It should be more high value.

Proportion\_False\_Positives<-1- Proportion\_True\_Positives

0.2481203

Proportion of False Positives is 24.81%, it is the probability that the customer is not going to churn but we

predict that it would churn.

1. Proportion of true negatives and proportion of false negatives

Proportion\_True\_Negatives<- TN/(FN+TN)

0.9238494

Proportion of true negative is 92.38%, it is the probability that the customer is not going to churn and does not churn, which is quite good.

Proportion\_False\_Negatives<- 1 -Proportion\_True\_Negatives

0.07615063

Proportion of False Negatives is 7.6%, it is the probability that the customer is going to churn but model

predicts that it would not churn, which would be a big risk.

1. Overall model cost.

1104($0) +33($100) +91($300) +100($0) = $30,600

The overall cost of the model is improved from $36,800 to $30,600. Thus, an improved in terms of costs of $6200

38. Construct a lift chart for the neural network model. What is the estimated lift at 20%?

33%? 40%? 50%?

**Lift Chart for the Neural Network Model**

net.dat <-nnet(Churn ~., data = traindf , size=8)

# weights: 137

initial value 818.658272

final value 432.000000

converged

str(net.dat)

conf.net <- predict(net.dat , newdata = testdf)

conf.net

head(conf.net)

[,1]

10 0

17 0

25 0

30 0

39 0

48 0

head(testdf)

m.net <- data.frame(NoCost.net = conf.net)

head(m.net)

NoCost.net

10 0

17 0

25 0

30 0

39 0

48 0

mm.net <- data.frame(NoCost.net = conf.net)

head(mm.net)

NoCost.net

10 0

17 0

25 0

30 0

39 0

48 0

our.lift.net <- lift(as.factor(testdf$Churn)~NoCost.net , data = mm.net)

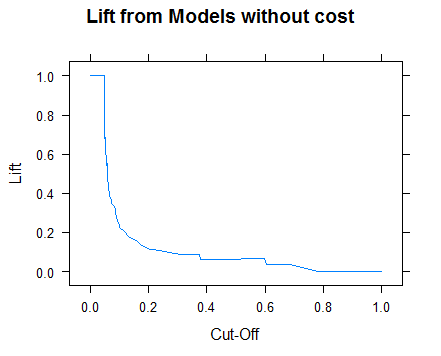
table(testdf$Churn, predict(net.dat , newdata = testdf))

0

FALSE 277

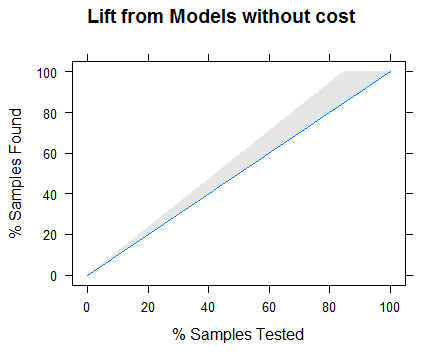
TRUE 51

xyplot(our.lift.net, plot = "lift" , auto.key = list(columns =2), main = "Lift from Models without cost")



The estimated lift at 20% approximately 0.15, at 33% it is approximately 0.1, at 40% and 50% it is approximately it is 0.5.

xyplot(our.lift.net, plot = "gain" , auto.key = list(columns =2), main = "Lift from Models without cost")



**Lift Chart using the tree Model**

conf=predict(churn.CART,newdata=traindf,type="prob")

conf.cost=predict(churn.CART.cost,newdata=traindf,type="prob")

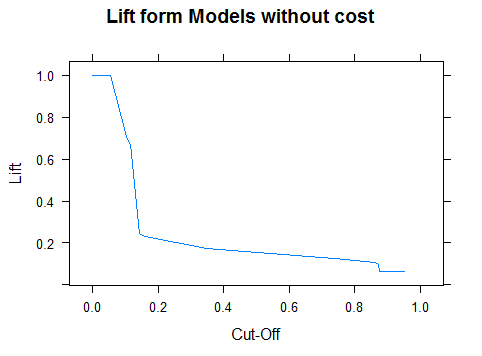
m2 <-data.frame(NoCost=conf[,2], Cost=conf.cost[,2])

m <-data.frame(NoCost=conf[,2])

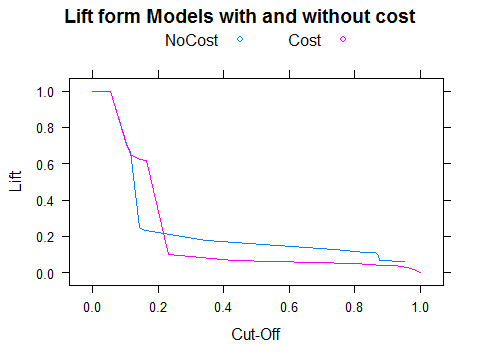
our.lift2<-lift(as.factor(traindf[,1])~NoCost+Cost,data=m2)

our.lift<-lift(as.factor(traindf[,1])~NoCost,data=m)

xyplot(our.lift,plot="lift", auto.key=list(columns=2), main="Lift form Models without cost")



xyplot(our.lift2,plot="lift", auto.key=list(columns=2), main="Lift form Models with and without cost")



xyplot(our.lift2,plot="gain", auto.key=list(columns=2), main="Gain for Models with and without cost")



xyplot(our.lift,plot="gain", auto.key=list(columns=1), main="Gain for Models without cost")

