MIS 510 Portfolio Project Option 1

Dan Amrine

September 4, 2019

setwd("C:/Users/dca80/Desktop/DATA/R\_DATA")  
#Read the data into the project  
credit<-as.data.frame(read.csv("GermanCredit.csv"))  
#view the strucutre of the dataset  
str(credit)

## 'data.frame': 1000 obs. of 32 variables:  
## $ OBS. : int 1 2 3 4 5 6 7 8 9 10 ...  
## $ CHK\_ACCT : int 0 1 3 0 0 3 3 1 3 1 ...  
## $ DURATION : int 6 48 12 42 24 36 24 36 12 30 ...  
## $ HISTORY : int 4 2 4 2 3 2 2 2 2 4 ...  
## $ NEW\_CAR : int 0 0 0 0 1 0 0 0 0 1 ...  
## $ USED\_CAR : int 0 0 0 0 0 0 0 1 0 0 ...  
## $ FURNITURE : int 0 0 0 1 0 0 1 0 0 0 ...  
## $ RADIO.TV : int 1 1 0 0 0 0 0 0 1 0 ...  
## $ EDUCATION : int 0 0 1 0 0 1 0 0 0 0 ...  
## $ RETRAINING : int 0 0 0 0 0 0 0 0 0 0 ...  
## $ AMOUNT : int 1169 5951 2096 7882 4870 9055 2835 6948 3059 5234 ...  
## $ SAV\_ACCT : int 4 0 0 0 0 4 2 0 3 0 ...  
## $ EMPLOYMENT : int 4 2 3 3 2 2 4 2 3 0 ...  
## $ INSTALL\_RATE : int 4 2 2 2 3 2 3 2 2 4 ...  
## $ MALE\_DIV : int 0 0 0 0 0 0 0 0 1 0 ...  
## $ MALE\_SINGLE : int 1 0 1 1 1 1 1 1 0 0 ...  
## $ MALE\_MAR\_or\_WID : int 0 0 0 0 0 0 0 0 0 1 ...  
## $ CO.APPLICANT : int 0 0 0 0 0 0 0 0 0 0 ...  
## $ GUARANTOR : int 0 0 0 1 0 0 0 0 0 0 ...  
## $ PRESENT\_RESIDENT: int 4 2 3 4 4 4 4 2 4 2 ...  
## $ REAL\_ESTATE : int 1 1 1 0 0 0 0 0 1 0 ...  
## $ PROP\_UNKN\_NONE : int 0 0 0 0 1 1 0 0 0 0 ...  
## $ AGE : int 67 22 49 45 53 35 53 35 61 28 ...  
## $ OTHER\_INSTALL : int 0 0 0 0 0 0 0 0 0 0 ...  
## $ RENT : int 0 0 0 0 0 0 0 1 0 0 ...  
## $ OWN\_RES : int 1 1 1 0 0 0 1 0 1 1 ...  
## $ NUM\_CREDITS : int 2 1 1 1 2 1 1 1 1 2 ...  
## $ JOB : int 2 2 1 2 2 1 2 3 1 3 ...  
## $ NUM\_DEPENDENTS : int 1 1 2 2 2 2 1 1 1 1 ...  
## $ TELEPHONE : int 1 0 0 0 0 1 0 1 0 0 ...  
## $ FOREIGN : int 0 0 0 0 0 0 0 0 0 0 ...  
## $ RESPONSE : int 1 0 1 1 0 1 1 1 1 0 ...

# generate 5 statistics to help understand the numeric data  
data.frame(mean=sapply(credit[,c(3,11,23)],mean),  
 sd = sapply(credit[,c(3,11,23)],sd),  
 max = sapply(credit[,c(3,11,23)],max),  
 min = sapply(credit[,c(3,11,23)],min),  
 median = sapply(credit[,c(3,11,23)],median))

## mean sd max min median  
## DURATION 20.903 12.05881 72 4 18.0  
## AMOUNT 3271.258 2822.73688 18424 250 2319.5  
## AGE 35.546 11.37547 75 19 33.0

#notice the mean duration of the loans are 20 months  
# the mean amount is 3,271 DM or $1,850 USD  
# the mean age of applicant is 35 years.  
  
#look at the count of Response where 1 is good credit rating and 0 is bad.  
# there were 700 applicants with a good response and 300 with a bad response.

## Warning: package 'plyr' was built under R version 3.5.3

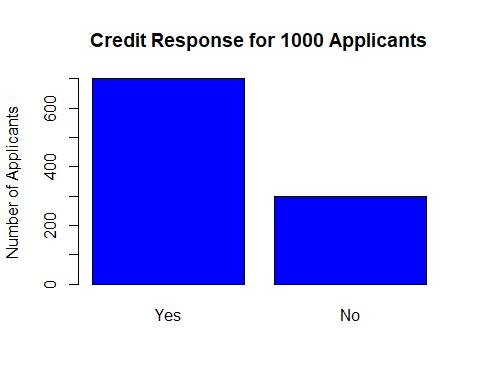


Fig.1 The breakdown of applicants with good(yes) and bad(no) Credit.

## in the next section we will be predicting the response based on 30   
## predictors.   
## we will use random forest and a nueral network  
  
# The first step is to seperate the data into a validation and training sets.  
set.seed(1)  
train.index<-sample(c(1:dim(credit)[1]), dim(credit)[1]\*0.6)  
traincrd.df<- credit[train.index,]  
validcrd.df<- credit[-train.index, ]  
  
#once the training and validations sets are defined we can run the random forest and neuralnet.  
#Load all necessary packages for confusion matrix and classifiation trees  
library(caret)

## Warning: package 'caret' was built under R version 3.5.3

## Loading required package: lattice

## Loading required package: ggplot2

## Warning: package 'ggplot2' was built under R version 3.5.3

library(dplyr)

## Warning: package 'dplyr' was built under R version 3.5.3

##   
## Attaching package: 'dplyr'

## The following objects are masked from 'package:plyr':  
##   
## arrange, count, desc, failwith, id, mutate, rename, summarise,  
## summarize

## The following objects are masked from 'package:stats':  
##   
## filter, lag

## The following objects are masked from 'package:base':  
##   
## intersect, setdiff, setequal, union

library(rpart)

## Warning: package 'rpart' was built under R version 3.5.3

library(rpart.plot)

## Warning: package 'rpart.plot' was built under R version 3.5.3

#convert all 0,1 in training and validation sets into no and yes  
traincrd.factor<-traincrd.df[ ,c("NEW\_CAR","USED\_CAR","FURNITURE","RADIO.TV","EDUCATION","RETRAINING","MALE\_DIV","MALE\_SINGLE","MALE\_MAR\_or\_WID","CO.APPLICANT","GUARANTOR","REAL\_ESTATE","PROP\_UNKN\_NONE","OTHER\_INSTALL","RENT","OWN\_RES","TELEPHONE","FOREIGN","RESPONSE")]  
#Convert "0" and "1" to "No" and "Yes"  
traincrd.factor.yes<-lapply(traincrd.factor,factor,levels= c(0,1), labels =c("No","Yes"))  
# for validation set  
validcrd.factor<-validcrd.df[ ,c("NEW\_CAR","USED\_CAR","FURNITURE","RADIO.TV","EDUCATION","RETRAINING","MALE\_DIV","MALE\_SINGLE","MALE\_MAR\_or\_WID","CO.APPLICANT","GUARANTOR","REAL\_ESTATE","PROP\_UNKN\_NONE","OTHER\_INSTALL","RENT","OWN\_RES","TELEPHONE","FOREIGN","RESPONSE")]  
#Convert "0" and "1" to "No" and "Yes"  
validcrd.factor.yes<-lapply(validcrd.factor,factor,levels= c(0,1), labels =c("No","Yes"))  
  
#create the training classification tree  
credit.ct<-rpart(RESPONSE ~ .,data = traincrd.factor.yes,method = "class",model=TRUE)  
#plot the training classification tree  
prp(credit.ct,type = 2, extra = "auto", under = TRUE, split.font = 1, varlen = -10, main = "Yes = Good Credit")

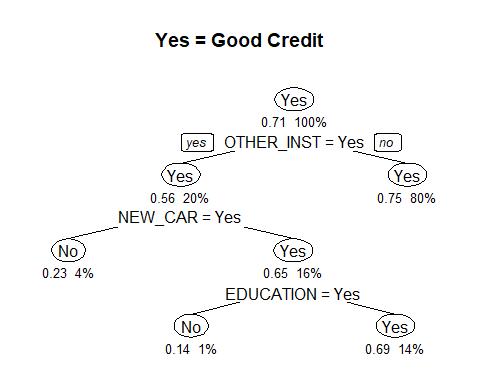


Fig.2 The classification tree on the training data set. If the applicant has a loan at the bank it impacts their credit rating.

#create the validation classification tree  
valcredit.ct<-rpart(RESPONSE ~ .,data = validcrd.factor.yes,method = "class",model=TRUE)  
#plot the validation classification tree  
prp(valcredit.ct,type = 2, extra = "auto", under = TRUE, split.font = 1, varlen = -10, main = "Yes = Good Credit")

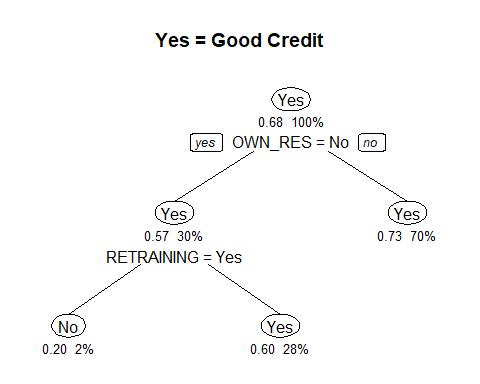


Fig.3 The classification tree on the validation data set.

# generate the predicted class membership for the training set  
credit.ct.point.pred.train<-predict(credit.ct,traincrd.factor.yes, type = "class")  
#Generate the training confusion matrix for accuracy  
confusionMatrix(credit.ct.point.pred.train,traincrd.factor.yes$RESPONSE)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction No Yes  
## No 26 7  
## Yes 146 421  
##   
## Accuracy : 0.745   
## 95% CI : (0.7081, 0.7794)  
## No Information Rate : 0.7133   
## P-Value [Acc > NIR] : 0.04623   
##   
## Kappa : 0.1778   
##   
## Mcnemar's Test P-Value : < 2e-16   
##   
## Sensitivity : 0.15116   
## Specificity : 0.98364   
## Pos Pred Value : 0.78788   
## Neg Pred Value : 0.74250   
## Prevalence : 0.28667   
## Detection Rate : 0.04333   
## Detection Prevalence : 0.05500   
## Balanced Accuracy : 0.56740   
##   
## 'Positive' Class : No   
##

# generate the predicted class membership for the validation set  
credit.ct.point.pred.val<-predict(valcredit.ct,validcrd.factor.yes, type = "class")  
#Generate the validation confusion matrix for accuracy  
confusionMatrix(credit.ct.point.pred.val,validcrd.factor.yes$RESPONSE)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction No Yes  
## No 8 2  
## Yes 120 270  
##   
## Accuracy : 0.695   
## 95% CI : (0.6473, 0.7398)  
## No Information Rate : 0.68   
## P-Value [Acc > NIR] : 0.2792   
##   
## Kappa : 0.0729   
##   
## Mcnemar's Test P-Value : <2e-16   
##   
## Sensitivity : 0.0625   
## Specificity : 0.9926   
## Pos Pred Value : 0.8000   
## Neg Pred Value : 0.6923   
## Prevalence : 0.3200   
## Detection Rate : 0.0200   
## Detection Prevalence : 0.0250   
## Balanced Accuracy : 0.5276   
##   
## 'Positive' Class : No   
##

#Will now compare using a random forest  
library(randomForest)

## Warning: package 'randomForest' was built under R version 3.5.3

## randomForest 4.6-14

## Type rfNews() to see new features/changes/bug fixes.

##   
## Attaching package: 'randomForest'

## The following object is masked from 'package:dplyr':  
##   
## combine

## The following object is masked from 'package:ggplot2':  
##   
## margin

#Generate the random froest on the training set  
credrf<-randomForest(RESPONSE~.,data=traincrd.factor.yes, ntree = 500, mtry=4,nodesize = 5, importance=TRUE)  
#training set varaible importance plot on accuracy  
varImpPlot(credrf, type=1, n.var = 10, main = "Credit Response Random Forest (Accuracy)")

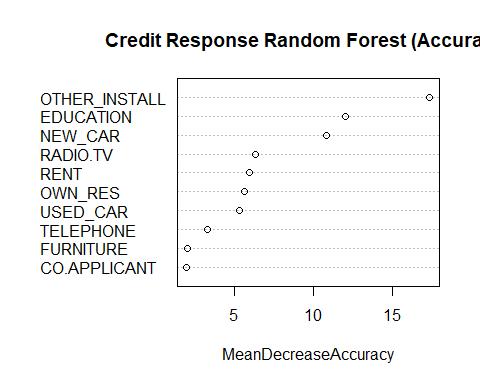


Fig.4 The top ten attributes by accuracy.

#training set varaible importance plot on node impurity  
varImpPlot(credrf, type=2, n.var = 10, main = "Credit Response Random Forest (Node Impurity)")

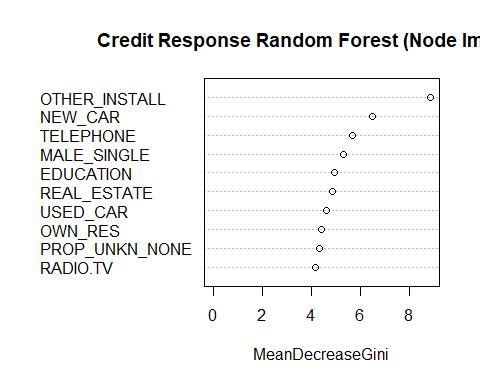


Fig.5 The node impurities.

# Generate the confusion matrix vs. the validation set  
credrf.pred<-predict(credrf,validcrd.factor.yes)  
confusionMatrix(credrf.pred,validcrd.factor.yes$RESPONSE)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction No Yes  
## No 11 18  
## Yes 117 254  
##   
## Accuracy : 0.6625   
## 95% CI : (0.6138, 0.7087)  
## No Information Rate : 0.68   
## P-Value [Acc > NIR] : 0.7899   
##   
## Kappa : 0.0248   
##   
## Mcnemar's Test P-Value : <2e-16   
##   
## Sensitivity : 0.08594   
## Specificity : 0.93382   
## Pos Pred Value : 0.37931   
## Neg Pred Value : 0.68464   
## Prevalence : 0.32000   
## Detection Rate : 0.02750   
## Detection Prevalence : 0.07250   
## Balanced Accuracy : 0.50988   
##   
## 'Positive' Class : No   
##

### The Neural Net  
#Load necesaary library for neural net.  
library(neuralnet)

## Warning: package 'neuralnet' was built under R version 3.5.3

##   
## Attaching package: 'neuralnet'

## The following object is masked from 'package:dplyr':  
##   
## compute

library(nnet)

## Warning: package 'nnet' was built under R version 3.5.3

# I attempted to think like a bank manager and chose a formula that would predict the amount, duration, and credit rating of the applicant based on key attributes.  
crednn<-neuralnet(RESPONSE~SAV\_ACCT+EMPLOYMENT+JOB+OTHER\_INSTALL+REAL\_ESTATE+RENT,data=traincrd.df, linear.output = F, hidden = 3)  
# display the weights  
crednn$weights

## [[1]]  
## [[1]][[1]]  
## [,1] [,2] [,3]  
## [1,] 5.01966768 6.448113 -3.0400808  
## [2,] 21.78636639 4.990925 0.1470659  
## [3,] -10.06645774 -6.977290 0.8629436  
## [4,] 17.60923144 -17.771373 -0.4058154  
## [5,] -0.09628575 -3.445006 -1.2999684  
## [6,] -15.08542991 40.805631 0.8876505  
## [7,] -7.77282270 8.421680 -0.2788558  
##   
## [[1]][[2]]  
## [,1]  
## [1,] -3.038871  
## [2,] 2.920612  
## [3,] 2.163136  
## [4,] 6.928635

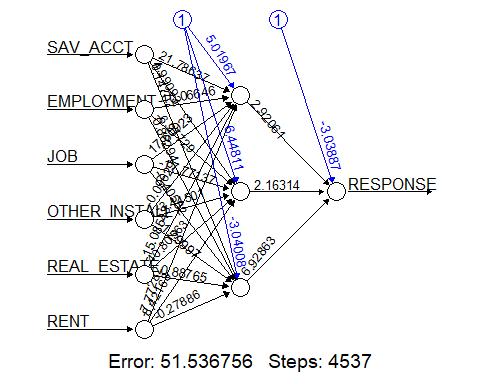


Fig.6 The first Neural Net without separating “yes” and “no”.

#predicting the training set  
crednn.predict<-compute(crednn,traincrd.df[ ,c("SAV\_ACCT","EMPLOYMENT","JOB","OTHER\_INSTALL","REAL\_ESTATE","RENT")])  
crednn.class <- crednn.predict$net.result  
crednn.class.fac<-ifelse(crednn.class>0.5,"1","0")  
confusionMatrix(as.factor(crednn.class.fac),as.factor(traincrd.df$RESPONSE))

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction 0 1  
## 0 37 28  
## 1 135 400  
##   
## Accuracy : 0.7283   
## 95% CI : (0.6908, 0.7636)  
## No Information Rate : 0.7133   
## P-Value [Acc > NIR] : 0.2222   
##   
## Kappa : 0.1839   
##   
## Mcnemar's Test P-Value : <2e-16   
##   
## Sensitivity : 0.21512   
## Specificity : 0.93458   
## Pos Pred Value : 0.56923   
## Neg Pred Value : 0.74766   
## Prevalence : 0.28667   
## Detection Rate : 0.06167   
## Detection Prevalence : 0.10833   
## Balanced Accuracy : 0.57485   
##   
## 'Positive' Class : 0   
##

#predicitng the validation set  
vcrednn.predict<-compute(crednn,validcrd.df[ ,c("SAV\_ACCT","EMPLOYMENT","JOB","OTHER\_INSTALL","REAL\_ESTATE","RENT")])  
vcrednn.class <- vcrednn.predict$net.result  
vcrednn.class.fac<-ifelse(vcrednn.class>0.5,"1","0")  
confusionMatrix(as.factor(vcrednn.class.fac),as.factor(validcrd.df$RESPONSE))

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction 0 1  
## 0 18 22  
## 1 110 250  
##   
## Accuracy : 0.67   
## 95% CI : (0.6215, 0.7159)  
## No Information Rate : 0.68   
## P-Value [Acc > NIR] : 0.6869   
##   
## Kappa : 0.073   
##   
## Mcnemar's Test P-Value : 3.665e-14   
##   
## Sensitivity : 0.1406   
## Specificity : 0.9191   
## Pos Pred Value : 0.4500   
## Neg Pred Value : 0.6944   
## Prevalence : 0.3200   
## Detection Rate : 0.0450   
## Detection Prevalence : 0.1000   
## Balanced Accuracy : 0.5299   
##   
## 'Positive' Class : 0   
##

#based on the high error this did not work properly and the #reason is the response is binary but it's predicting   
#numeric values from 0 to 1.   
  
#Will have to create a new data set and dummies to run the #network properly.   
  
# I reduced it to response, savings account, employment, and job.  
  
#### The next Neural Net  
  
#new partition   
set.seed(2)  
Ctraining = sample(row.names(credit), dim(credit)[1]\*0.6)  
Cvalidation = setdiff(row.names(credit), Ctraining)  
  
#list of variables for new data from  
vars=c("RESPONSE","EMPLOYMENT","JOB")  
  
#dummify columns with multiple classes.  
#Remember that if the response is 0 then the new class is 1  
#and if response is 1 then in that dummy class it is also 1.   
#This process reduces all the classes into binary values using columns  
CtrainData<-cbind(credit[Ctraining,c(vars)],  
 class.ind(credit[Ctraining,]$RESPONSE),  
 class.ind(credit[Ctraining,]$EMPLOYMENT),  
 class.ind(credit[Ctraining,]$JOB))  
names(CtrainData)=c(vars,paste("RESPONSE\_",c(0,1), sep = ""),  
 paste("EMPLOYMENT\_",c(0,1,2,3,4), sep = ""),  
 paste("JOB\_",c(0,1,2,3), sep = ""))  
#same process for the validation data  
CvalData<-cbind(credit[Cvalidation,c(vars)],  
 class.ind(credit[Cvalidation,]$RESPONSE),  
 class.ind(credit[Cvalidation,]$EMPLOYMENT),  
 class.ind(credit[Cvalidation,]$JOB))  
names(CvalData)=c(vars,paste("RESPONSE\_",c(0,1), sep = ""),  
 paste("EMPLOYMENT\_",c(0,1,2,3,4), sep = ""),  
 paste("JOB\_",c(0,1,2,3), sep = ""))  
  
#created a neural net with 2 hidden nodes  
# used hidden= to specify nodes in each layer  
Cnn<-neuralnet(RESPONSE\_1~EMPLOYMENT\_0+EMPLOYMENT\_1+EMPLOYMENT\_2+EMPLOYMENT\_3+EMPLOYMENT\_4+JOB\_0+JOB\_1+JOB\_2+JOB\_3, data=CtrainData, hidden = 3,act.fct = "logistic",linear.output = FALSE)  
Cnn$weights

## [[1]]  
## [[1]][[1]]  
## [,1] [,2] [,3]  
## [1,] -0.3909426 0.5793971 -0.653908190  
## [2,] -2.1016153 2.9393674 -1.584254062  
## [3,] 3.7988065 10.8404550 -1.182345045  
## [4,] 2.9203809 -1.0919689 0.007058307  
## [5,] -1.0986842 -14.0079918 13.733694075  
## [6,] -4.8669236 -1.5201030 -0.666866966  
## [7,] -3.5712523 11.3899986 2.508621910  
## [8,] -3.2361885 -4.5987778 0.480319751  
## [9,] 5.9265175 0.1678562 4.309590808  
## [10,] -5.9447153 12.0598020 -11.409440345  
##   
## [[1]][[2]]  
## [,1]  
## [1,] 0.8906605  
## [2,] -1.1671631  
## [3,] -0.5187573  
## [4,] 1.2957682

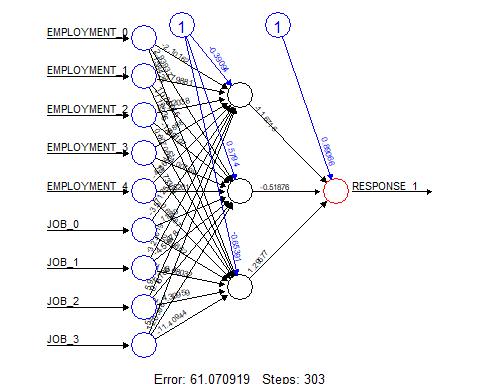


Fig.7 the neural net predicting only the “yes” response base on the applicant’s job and employment length only.

#I had to convert it from a scales of -1 - 1 to 0's and 1's factor class. I used 0.5   
#for the cutoff between a "1" and "0"  
#Hopefully i am not missing something about the Neural Net reults that makes this invalid.  
  
# confusion matrix for the 2 training results of the model res1 = 0 or bad credit, and res2 = 1 or good credit  
Ctraining.prediction = compute(Cnn,CtrainData[ ,-5])  
Ctraining.class <- Ctraining.prediction$net.result  
Ctraining.class.fac<-ifelse(Ctraining.class>0.5,"1","0")  
confusionMatrix(as.factor(Ctraining.class.fac),as.factor(credit[Ctraining,]$RESPONSE))

## Warning in confusionMatrix.default(as.factor(Ctraining.class.fac),  
## as.factor(credit[Ctraining, : Levels are not in the same order for  
## reference and data. Refactoring data to match.

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction 0 1  
## 0 0 0  
## 1 183 417  
##   
## Accuracy : 0.695   
## 95% CI : (0.6564, 0.7316)  
## No Information Rate : 0.695   
## P-Value [Acc > NIR] : 0.52   
##   
## Kappa : 0   
##   
## Mcnemar's Test P-Value : <2e-16   
##   
## Sensitivity : 0.000   
## Specificity : 1.000   
## Pos Pred Value : NaN   
## Neg Pred Value : 0.695   
## Prevalence : 0.305   
## Detection Rate : 0.000   
## Detection Prevalence : 0.000   
## Balanced Accuracy : 0.500   
##   
## 'Positive' Class : 0   
##

# confusion matrix for the 2 validation results of the model res1 = 0 or bad credit, and res2 = 1 or good credit  
Cval.prediction = compute(Cnn,CvalData[ ,-5])  
Cval.class <- Cval.prediction$net.result  
Cval.class.fac<-ifelse(Cval.class>0.5,"1","0")  
confusionMatrix(as.factor(Cval.class.fac),as.factor(credit[Cvalidation,]$RESPONSE))

## Warning in confusionMatrix.default(as.factor(Cval.class.fac),  
## as.factor(credit[Cvalidation, : Levels are not in the same order for  
## reference and data. Refactoring data to match.

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction 0 1  
## 0 0 0  
## 1 117 283  
##   
## Accuracy : 0.7075   
## 95% CI : (0.6602, 0.7517)  
## No Information Rate : 0.7075   
## P-Value [Acc > NIR] : 0.5249   
##   
## Kappa : 0   
##   
## Mcnemar's Test P-Value : <2e-16   
##   
## Sensitivity : 0.0000   
## Specificity : 1.0000   
## Pos Pred Value : NaN   
## Neg Pred Value : 0.7075   
## Prevalence : 0.2925   
## Detection Rate : 0.0000   
## Detection Prevalence : 0.0000   
## Balanced Accuracy : 0.5000   
##   
## 'Positive' Class : 0   
##

#The Overall model derived from this neural can predict the applicants with good credit with 70% accuracy.

This Project began with a data set containing 30 variables and 1000 records. Each record represented an applicant for a loan and the bank used a field called *RESPONSE* to denote if the applicant had good or bad credit. The first step was to understand the data by looking at the structure. Immediately the *str* function tells us that all the fields in the dataset are integer and no factor.

The goal of the project is to run a classification tree and neural net on this dataset. The fact that the data was all integer means it had to be converted to factor to use in both the classification tree and neural net. This was the first step before attempting the tree.

We ran two classification trees to predict the values of “yes” and “no” in the *RESPONSE* field. The first was a basic tree without learning and utilizing a small number of leaves. This showed us the most important fields that impacted the response were if the applicant had another loan with the bank, they were shopping for a new car, and it was for education. These attributes had the greatest impact on a positive response for good credit. The second tree, a random forest, also confirmed this.

The first classification tree was able to predict the response with 70% accuracy on the validation set and 75% accuracy on the training set. The random forest was considerably less accurate at 66% but it was able to rank all the attribute’s impact by accuracy.

The final part of the project was using Neural Nets to find the probability of each attribute’s impact. Rather than run a net on all 30 attributes I chose some specific attributes that logically would have the most impact. The first net used the amount they had in savings, how long they were employed, their job, if the applicant has another loan at the bank, if they own a home or rent. The result from the neural net had to be converted from numeric values between -1 and 1 to a binary factor of “0” and “1”. I used a cutoff of 0.5 for this. The model predicted the response with an accuracy of 73%.

The final net took a deeper dive and used the individual values rather than columns from the data frame. This was considerably more difficult and the result accuracy of 70% was close enough to tell me that the initial neural net would be sufficient to classify the credit of new applicants coming in.

Overall this process shows us that the database we’ve collected, and these attributes can be used to determine if a new applicant has good or bad credit. These models will save us time in the future and help us have a more efficient application process.