**Dataset Selected:**

Blood Transfusion Service Center Data Set

URL: <http://archive.ics.uci.edu/ml/machine-learning-databases/blood-transfusion/transfusion.data>

Number of instances in dataset: 748

Number of attributes in dataset: 5 (Only 4 considered for training as one attribute is redundant)

How many fold cross-validation performed:10

|  |  |  |  |
| --- | --- | --- | --- |
| Classifier | Best Parameters Used | Accuracy | AUC |
| Decision Trees | Pruning Factor – 0.02 | 78.2 | 0.651 |
| Perceptron | Coeff: 0.2278, -0.0729, 0.1402, -0.0729 | 77.8 | 0.603 |
| Neural Net | Neurons - 6 | 72.8 | 0.71 |
| Deep Learning | Layers - 6 | 75.1 | 0.703 |
| SVM | Kernel - Radial | 78.1 | 0.62 |
| naïve Bayes | Tuning parameter 'fL' = 1, 3-fold, repeated 1 times | 73.8 | 0.643 |
| Logistic Regression | Kernel - Radial | 75.1 | 0.54 |
| k-Nearest Neighbors | Tuning parameter 'k' = 10, 3-fold, repeated 1 times | 78.2 | 0.611 |
| Bagging | Max Depth - 15 | 76.44 | 0.588 |
| Random Forests | ntree=20, mtry = 2 | 77.8 | 0.62 |
| AdaBoost | iterations =50, max depth for trees=30, comp=0.001 | 76 | 0.602 |
| Gradient Boosting | n.trees = 300, interaction.depth | 77.8 | 0.603 |

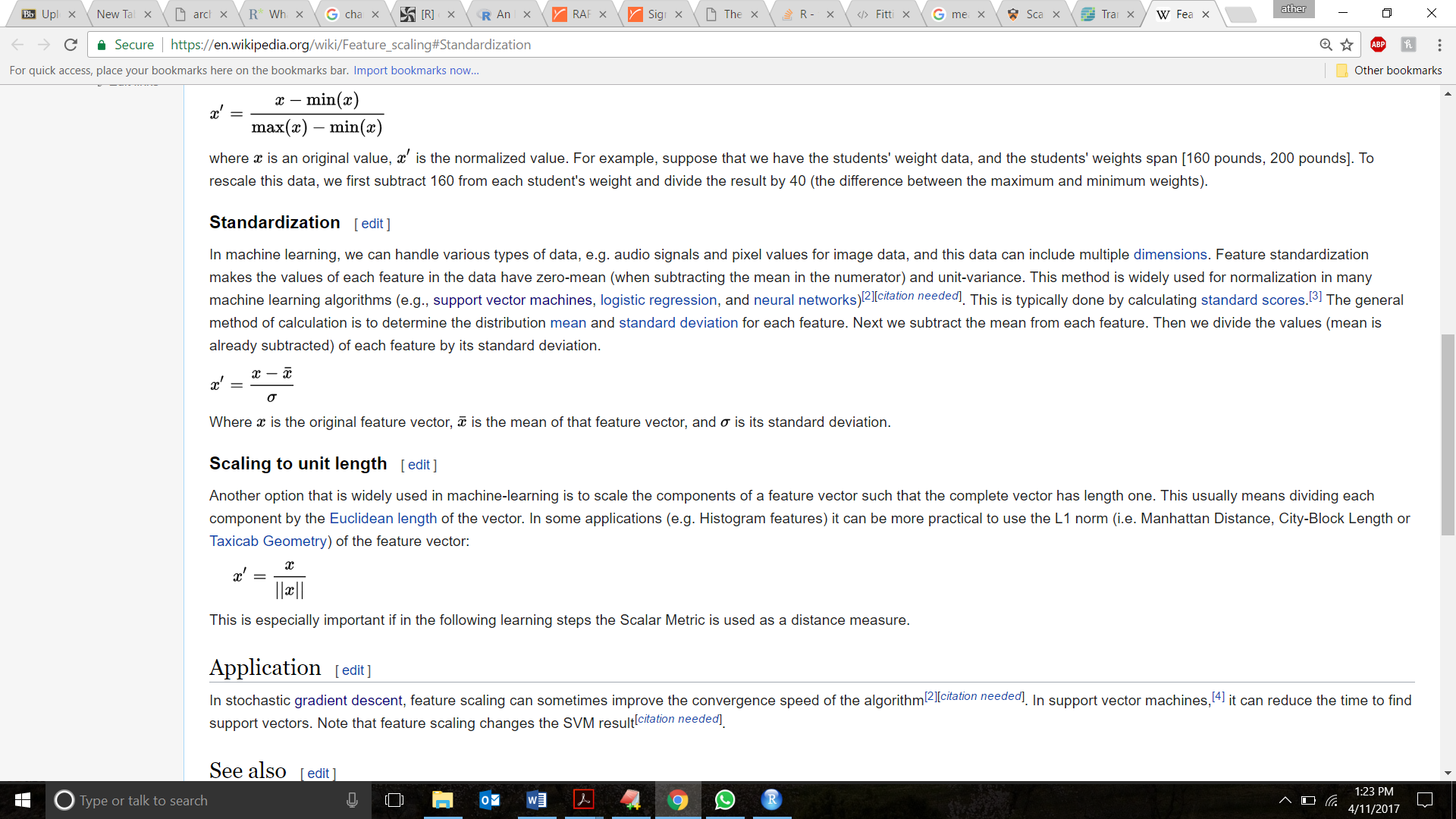
**Summary:**

The best classifiers observed for our data set are K NN(Tuning parameter 'k' = 10, 3-fold, repeated 1 times) and Deep Network (Layers - 6)

**Pre-Processing:**

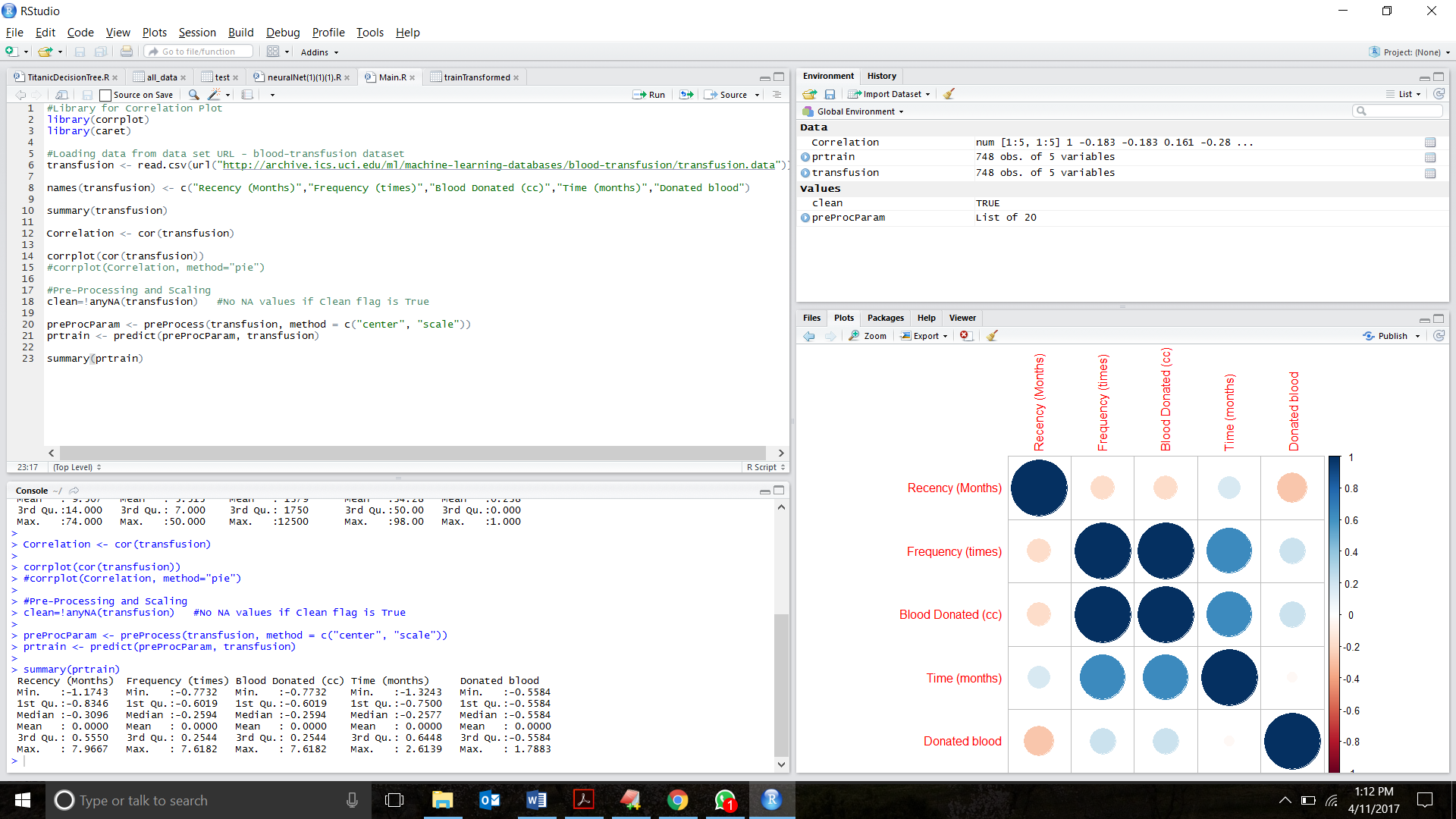
Using caret package to preprocess the data. The formula internally used for pre-processing the data is

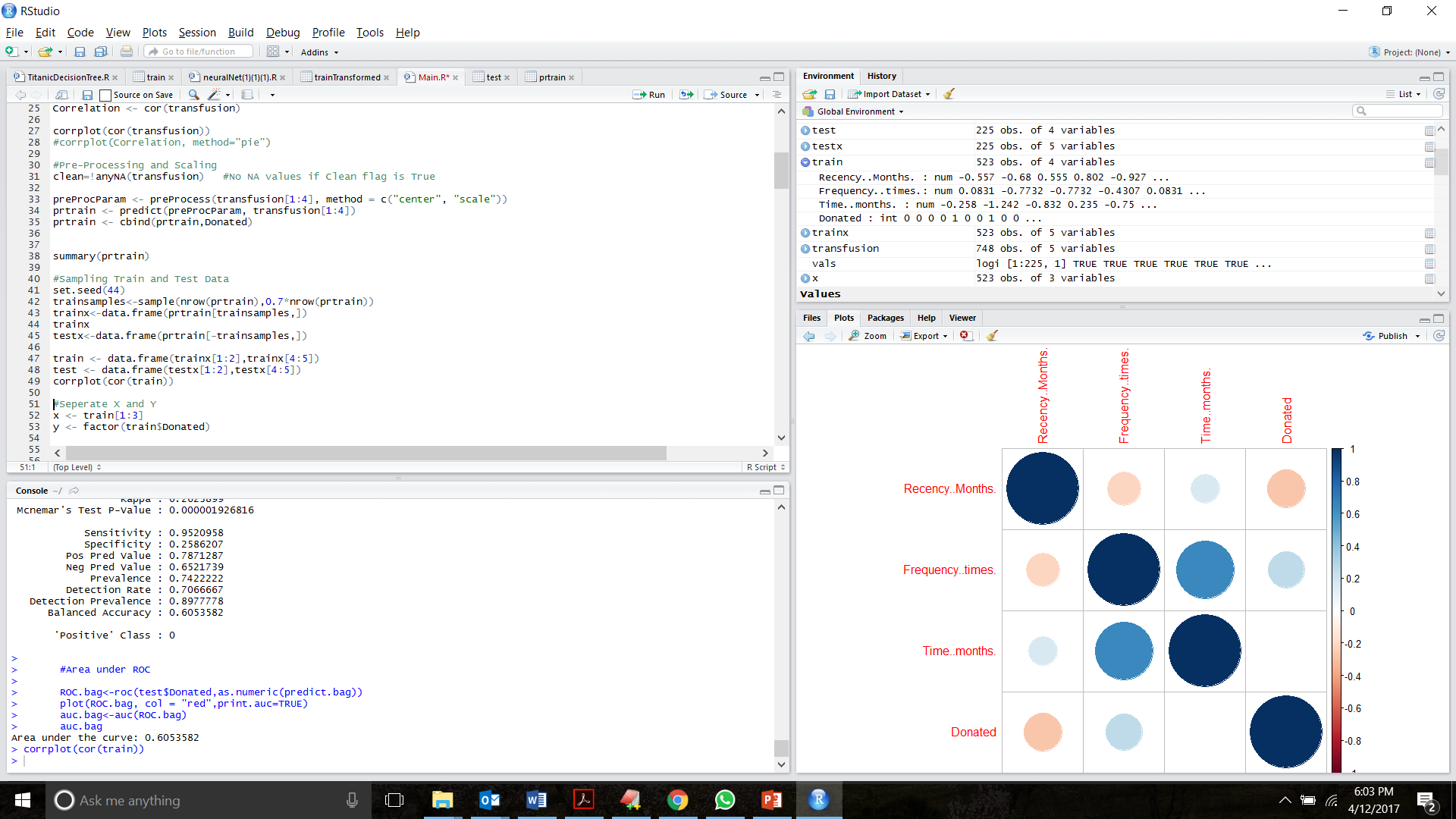
subtracting the mean from each of the values and dividing by the standard deviation.



As Blood donated is a redundant – Rank of Matrix was 4

Refer below corr snapshots for more details





After removing Blood Donated.

**Decision Tree**:

mytree<-rpart(train$Donated~., data=train, method = "class")

ptree<- prune(mytree, cp=0.02)

#Display Tree

fancyRpartPlot(mytree)

fancyRpartPlot(ptree)

# Make predictions on the test set

predict.dtree <- predict(mytree, newdata = test, type = "class")

predict.dtree.prune <- predict(ptree, newdata = test, type = "class")

#Use caret's confusion matrix

conTree<-confusionMatrix(predict.dtree,test$Donated)

conTreePrune<-confusionMatrix(predict.dtree.prune,test$Donated)

#Accuracy and Confusion Matrix

conTree

#Area under ROC

ROC.dtree<-roc(test$Donated,as.numeric(predict.dtree))

plot(ROC.dtree, col = "blue",print.auc=TRUE)

auc.dtree<-auc(ROC.dtree)

auc.dtree

#Accuracy and Confusion Matrix

conTreePrune

#Area under ROC

ROC.dtree.prune<-roc(test$Donated,as.numeric(predict.dtree.prune))

plot(ROC.dtree.prune, col = "blue",print.auc=TRUE)

auc.dtree.prune<-auc(ROC.dtree.prune)

auc.dtree.prune

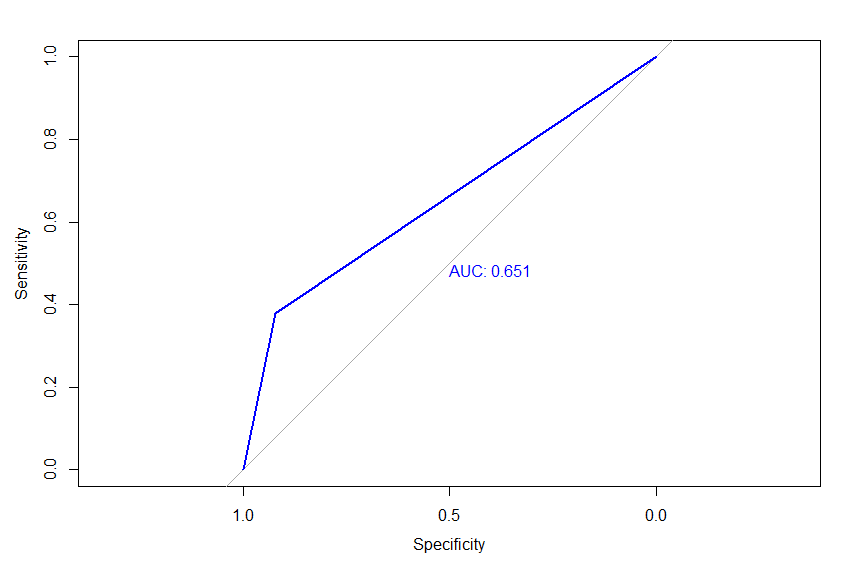
|  |  |  |
| --- | --- | --- |
| Prediction \ Reference | 0 | 1 |
| 0 | 154 | 36 |
| 1 | 13 | 22 |

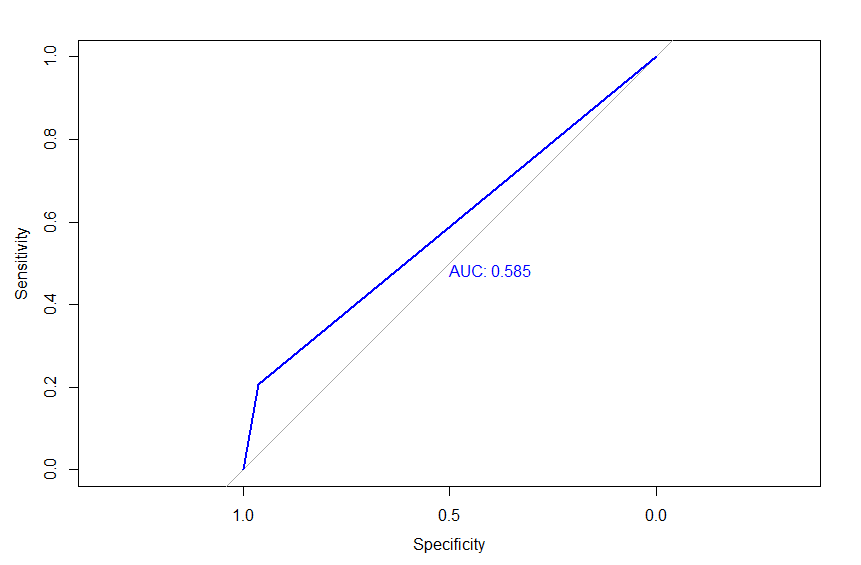
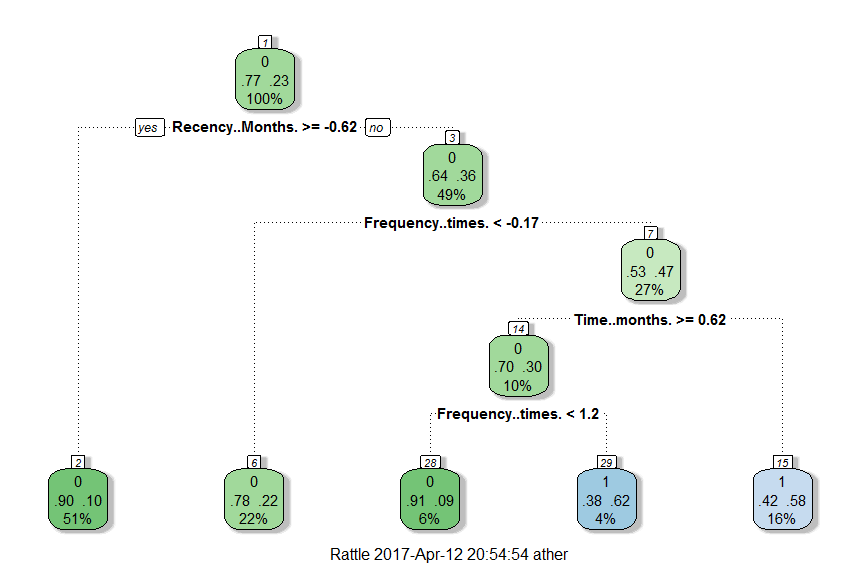
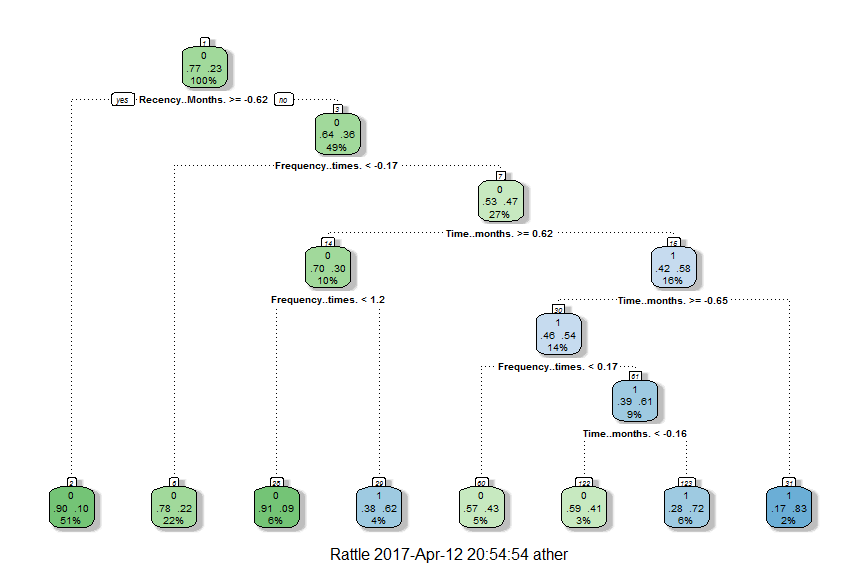
**Accuracy:** 78.2 %

**Area under the curve:** 0.651

**Best Param: Pruning Factor =** 0.02

**ROC Curve – Decision Tree**



**Perceptron**:

perceptron <-lm(Donated~ ., data=prtrain)

perceptron <-cv.lm(data=prtrain, perceptron, m=5)

predicted\_perceptron= ifelse(predict(perceptron,prtest)>0.35,1,0)

confusionMatrix(predicted\_perceptron,Donated)

ROCp<-roc(Donated,predicted\_perceptron)

plot(ROCp)

aucp<-auc(ROCp)

aucp

plot(ROCp)

|  |  |  |
| --- | --- | --- |
| Prediction \ Reference | 0 | 1 |
| 0 | 161 | 44 |
| 1 | 6 | 14 |

**Accuracy :** 77.8 %

**Area under the curve:** 0.603

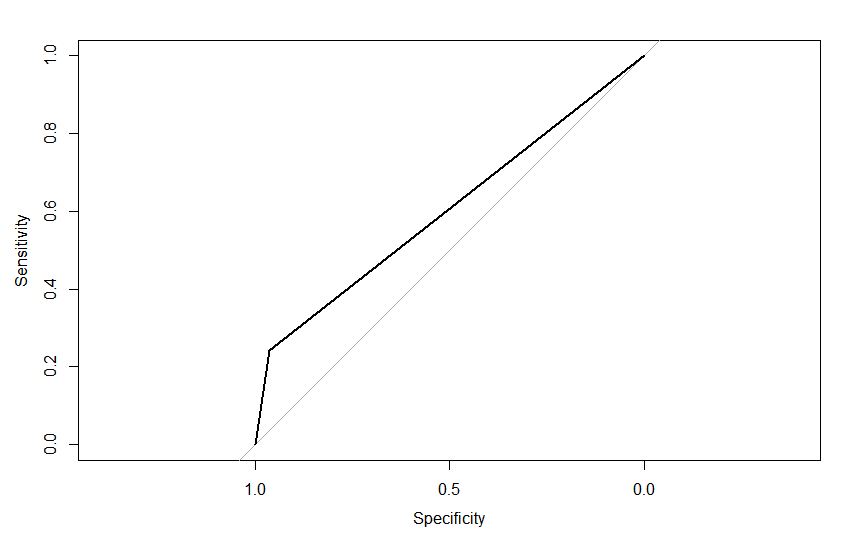
**Coefficients**:

(Intercept) Recency..Months. Frequency..times. Time..months.

0.2278 -0.0729 0.1402 -0.0729

Best parameter : Threshold =0.35

**ROC Curve - Perceptron**



**Neural Net**:

net <- neuralnet(f, data=train, hidden=6,linear.output=FALSE)

predict.nnet<-compute(net,test[1:3])

#extracting 1's and zeros

vals<- predict.nnet$net.result>0.2

predict.nnet.vals=ifelse(vals=="TRUE",1,0)

#Use caret's confusion matrix

conNeural<-confusionMatrix(predict.nnet.vals,test$Donated)

#Accuracy and Confusion Matrix

conNeural

#Area under ROC

ROC.nnet<-roc(test$Donated,as.numeric(predict.nnet.vals))

plot(ROC.nnet, col = "red",print.auc=TRUE)

auc.nnet<-auc(ROC.nnet)

auc.nnet

plot(net)

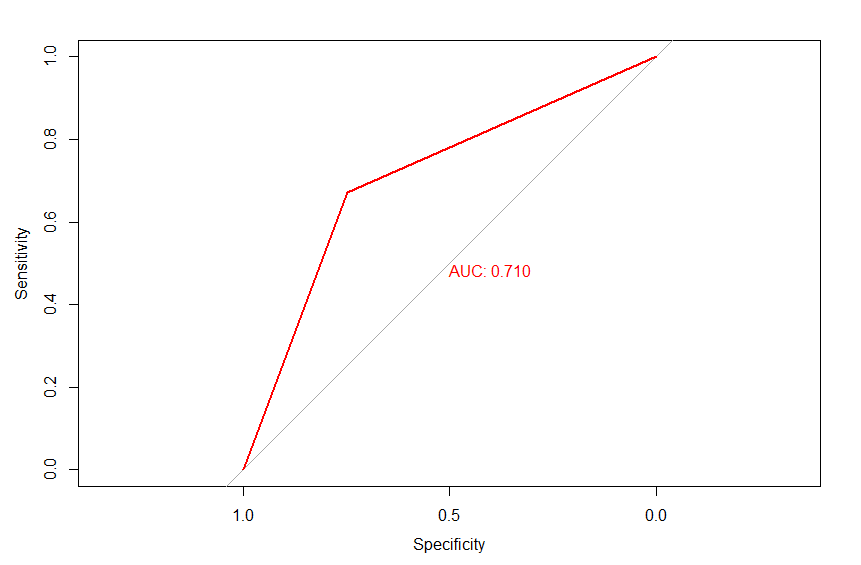
|  |  |  |
| --- | --- | --- |
| Prediction \ Reference | 0 | 1 |
| 0 | 125 | 19 |
| 1 | 42 | 39 |

**Accuracy:** 72.8 %

**Area under the curve:** 0.710

**Best Param Threshold(predict.nnet$net.result):** 0.2, **Number of Neurons**: 6

**ROC Curve - NeuralNet**



AUC change by changing threshold and number of neurons.

Area under the curve: 0.5941049

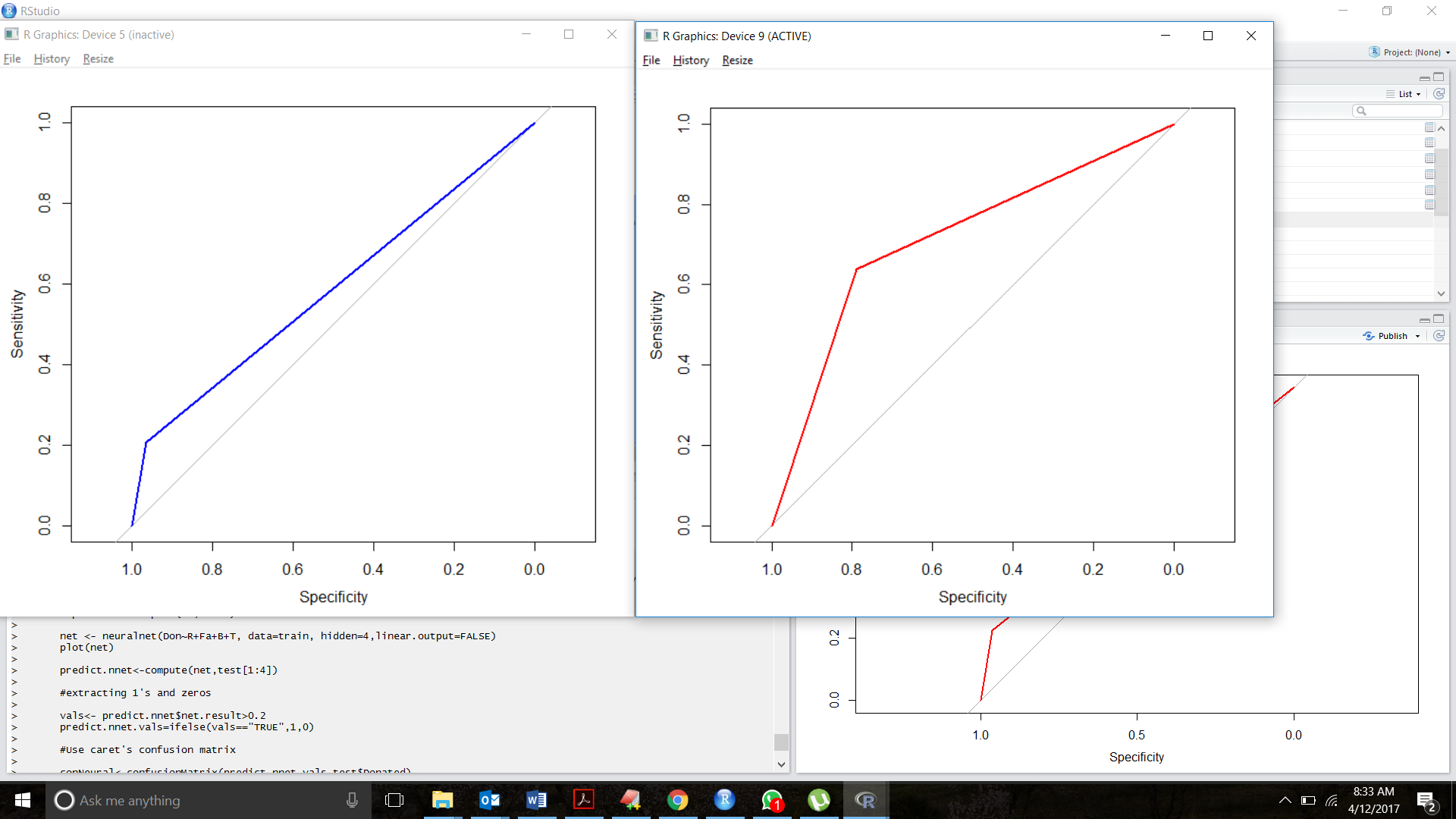
Area under the curve: 0.5626161

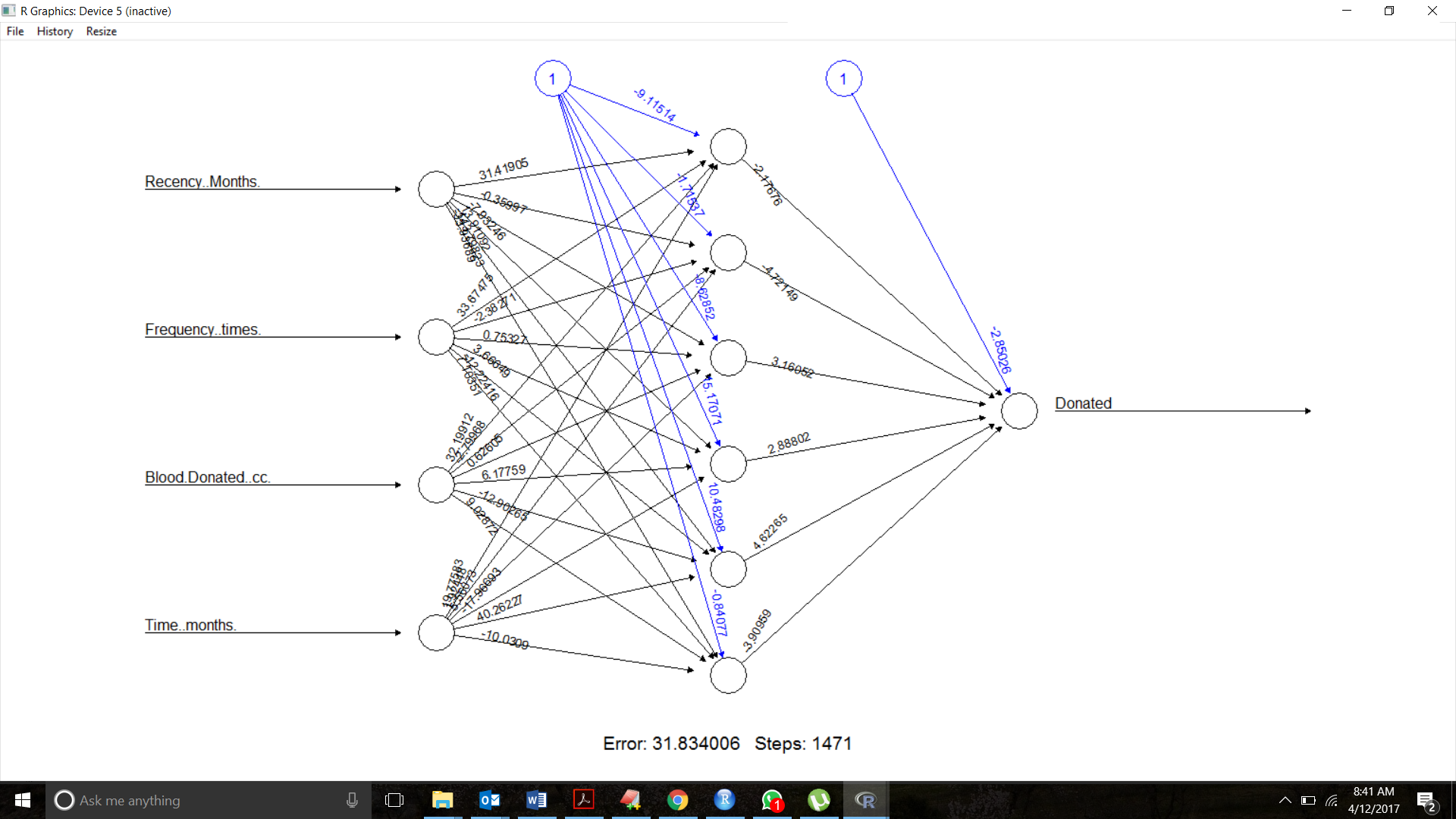
Area under the curve: 0.5967376

Area under the curve: 0.7040058

Area under the curve: 0.676337

Area under the curve: 0.7041751





**Deep Net**:

deep <- neuralnet(f, data=train, hidden=c(2,2,2,2,2,2),linear.output=FALSE)

predict.deep<-compute(deep,test[1:3])

#extracting 1's and zeros

vals<- predict.deep$net.result>0.2

predict.deep.vals=ifelse(vals=="TRUE",1,0)

#Use caret's confusion matrix

con.deep<-confusionMatrix(predict.deep.vals,test$Donated)

con.deep

#Area under ROC

ROC.deep<-roc(test$Donated,as.numeric(predict.deep.vals))

plot(ROC.deep, col = "red",print.auc=TRUE)

auc.deep<-auc(ROC.deep)

auc.deep

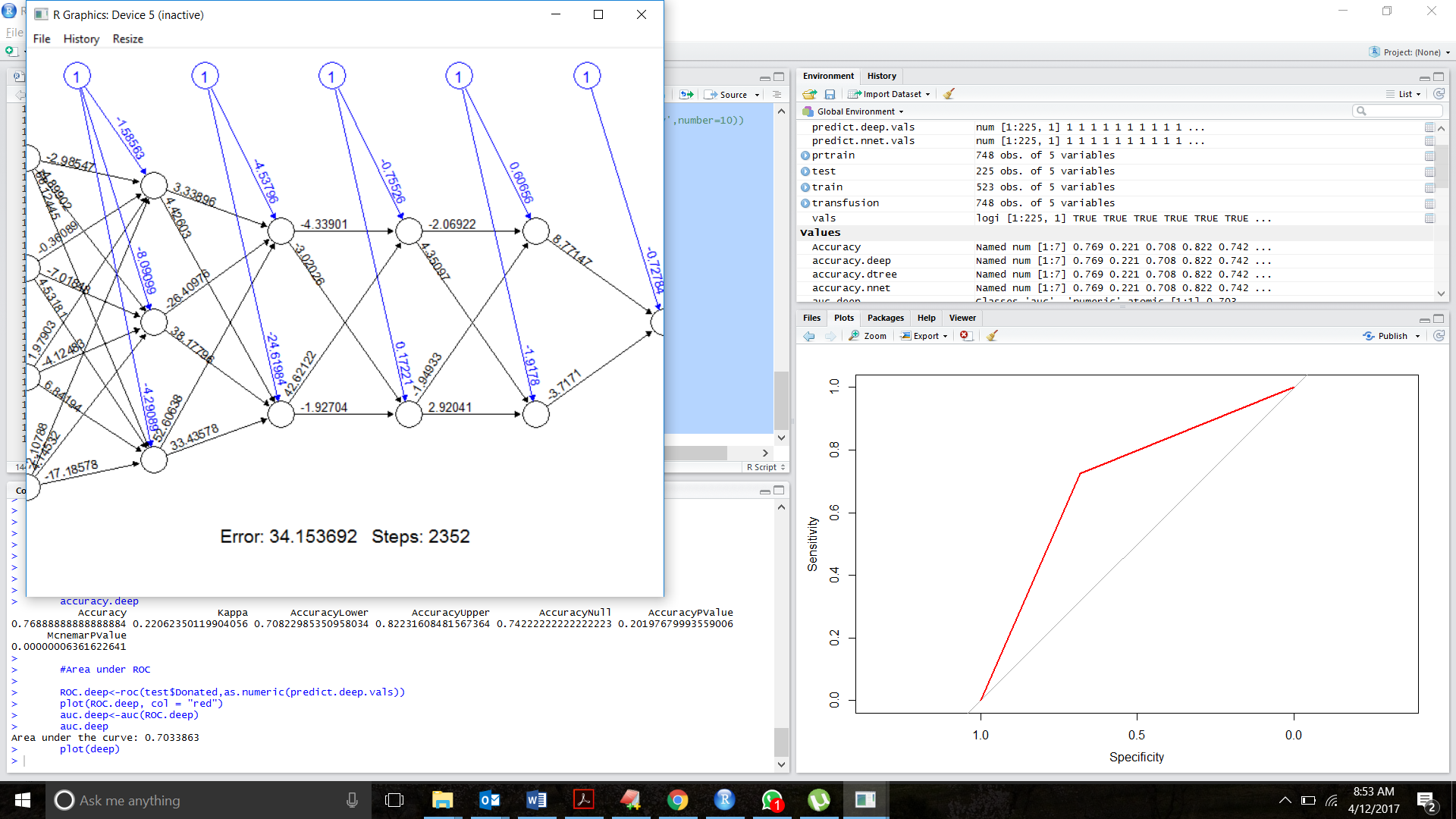
plot(deep)

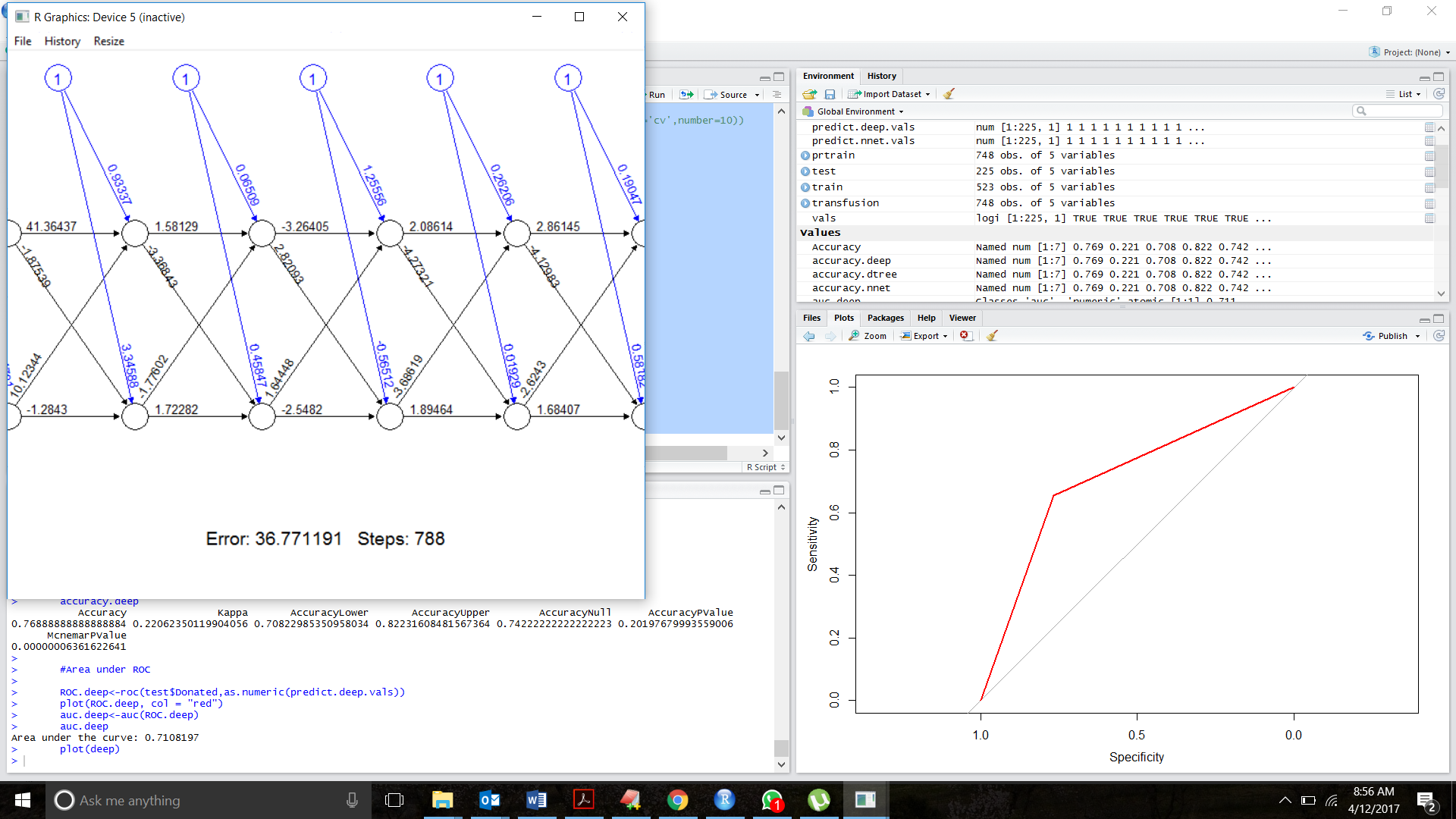
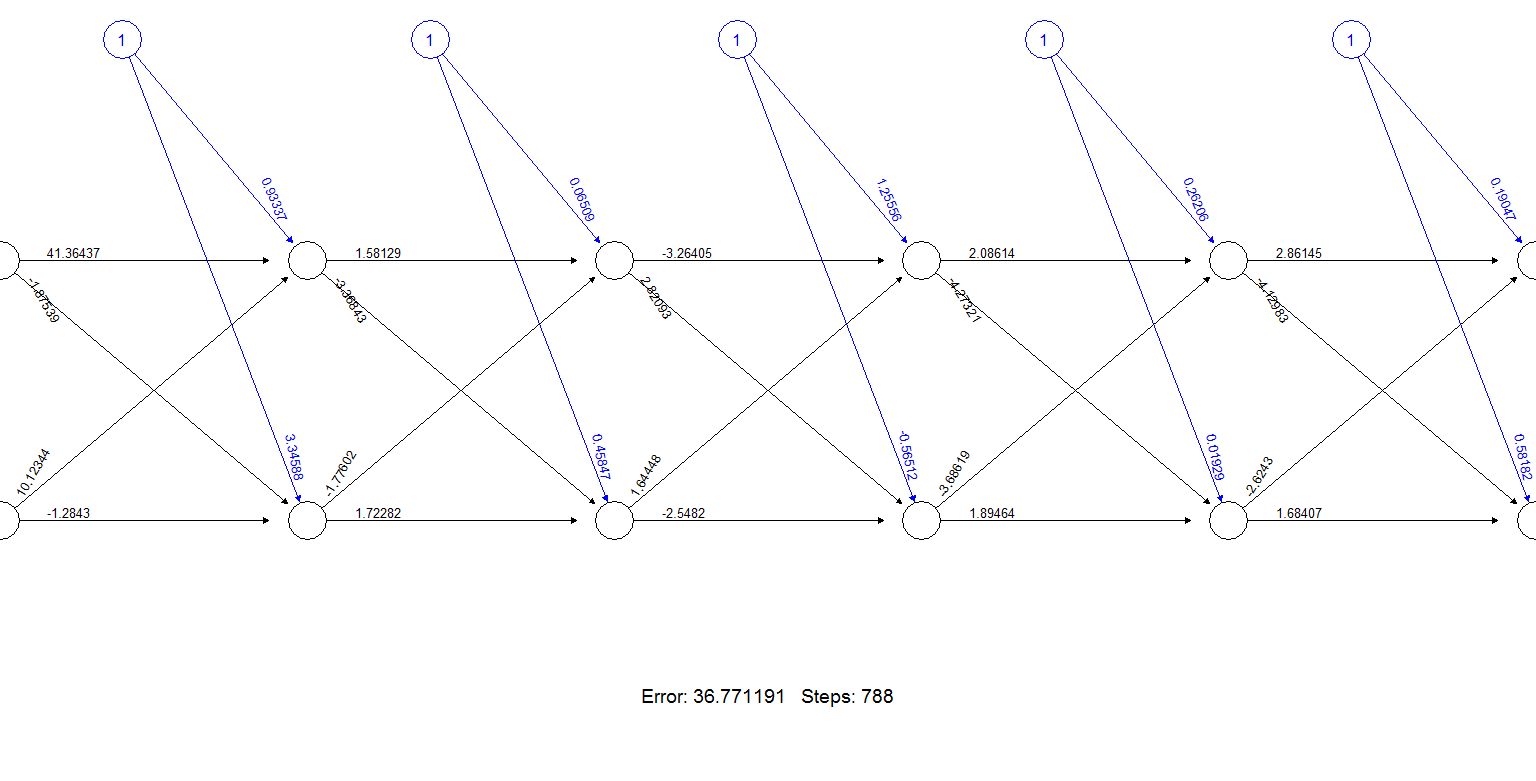
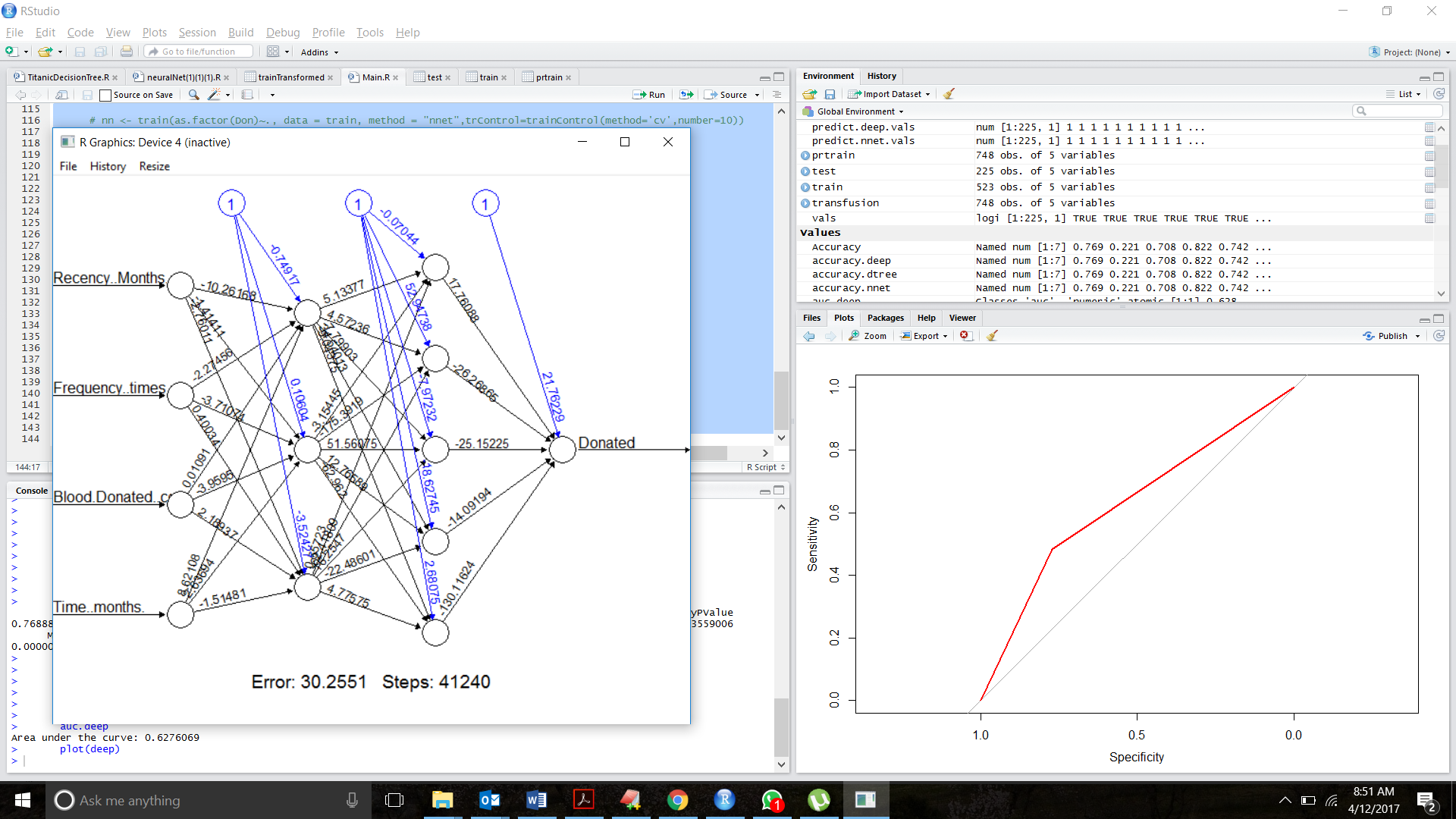
|  |  |  |
| --- | --- | --- |
| Prediction \ Reference | 0 | 1 |
| 0 | 137 | 26 |
| 1 | 30 | 32 |

**Accuracy:** 75.1 %

**Area under the curve:** 0.703

**Best Param number of Neurons/layer=**2, **Number of Layers**: 6

**ROC Curve - DeepNet**



**SVM**:

l.svm <- train(x,y, method = "svmLinear", trControl=ctrl)

p.svm <- train(x,y, method = "svmRadial", trControl=ctrl)

predict.svm<-predict(l.svm, newdata=test[1:3])

predict.psvm<-predict(p.svm, newdata=test[1:3])

conSVM<-confusionMatrix(predict.svm,test$Donated)

conSVM

conpSVM<-confusionMatrix(predict.psvm,test$Donated)

conpSVM

ROC.svm<-roc(test$Donated,as.numeric(predict.svm))

plot(ROC.svm, col = "blue",print.auc=TRUE)

auc.svm<-auc(ROC.svm)

auc.svm

ROC.psvm<-roc(test$Donated,as.numeric(predict.psvm))

plot(ROC.psvm, col = "red",print.auc=TRUE)

auc.psvm<-auc(ROC.psvm)

auc.psvm

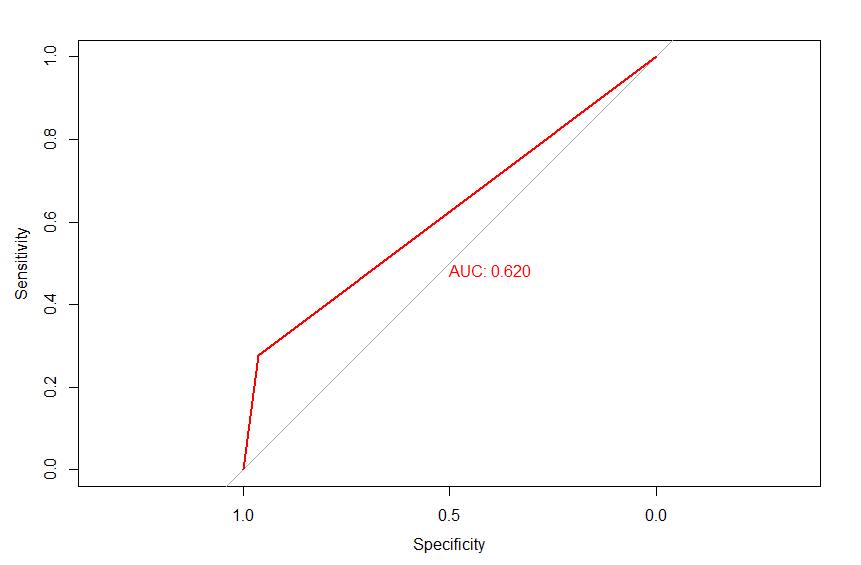
|  |  |  |
| --- | --- | --- |
| Prediction \ Reference | 0 | 1 |
| 0 | 161 | 42 |
| 1 | 6 | 16 |

**Accuracy:** 78.6 %

**Area under the curve:** 0.620

**Best Param: Kernel: Radial Kernel**

**ROC Curve – SVM**



**Naive Bayes:**

naive = train(as.factor(Donated) ~., data=prtrain, 'nb',trControl=trainControl(method='repeatedcv',number=3), tuneGrid=data.frame(fL= 1, usekernel=TRUE, adjust=TRUE))

print(naive)

predicted\_naive= predict(naive,prtest)

confusionMatrix(predicted\_naive,as.factor(Donated))

ROCn<-roc(Donated,as.numeric(predicted\_naive))

plot(ROCn)

aucn<-auc(ROCn)

aucn

plot(ROCn)

|  |  |  |
| --- | --- | --- |
| Prediction \ Reference | 0 | 1 |
| 0 | 140 | 32 |
| 1 | 27 | 26 |

**Accuracy:** 73.8%

**Area under the curve**: 0.643

**Best parameter:**

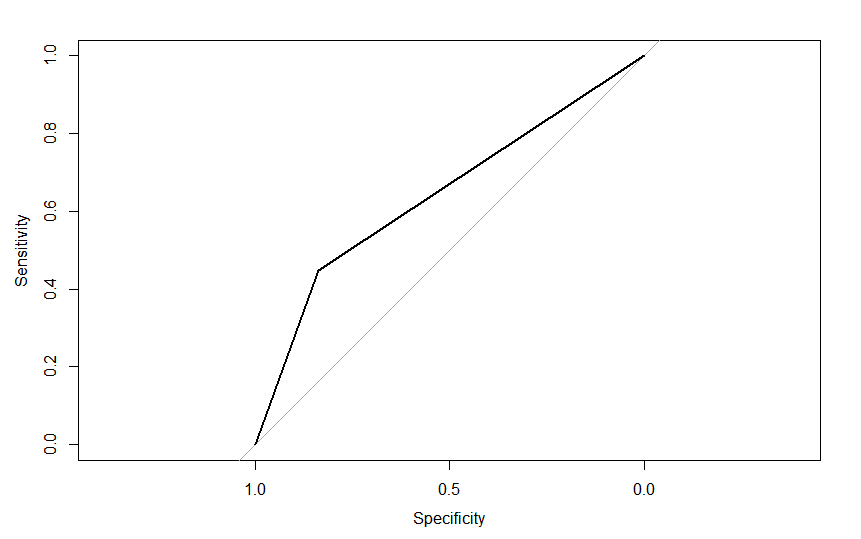
Tuning parameter 'fL' was held constant at a value of 1

Tuning parameter 'usekernel' was held constant at a value of TRUE

Tuning parameter 'adjust' was held constant at a value of TRUE

Cross-Validated (3 fold, repeated 1 times)

**ROC Curve – Naïve Bayes**



**Logistic Regression**:

glm <- train(x,y, method="glm",trControl=ctrl)

predict.glm<-predict(glm,test[1:3])

#Use caret's confusion matrix

conGLM<-confusionMatrix(predict.glm,test$Donated)

#Accuracy and Confusion Matrix

conGLM

#Area under ROC

ROC.glm<-roc(test$Donated,as.numeric(predict.glm))

plot(ROC.glm, col = "red",print.auc=TRUE)

auc.glm<-auc(ROC.glm)

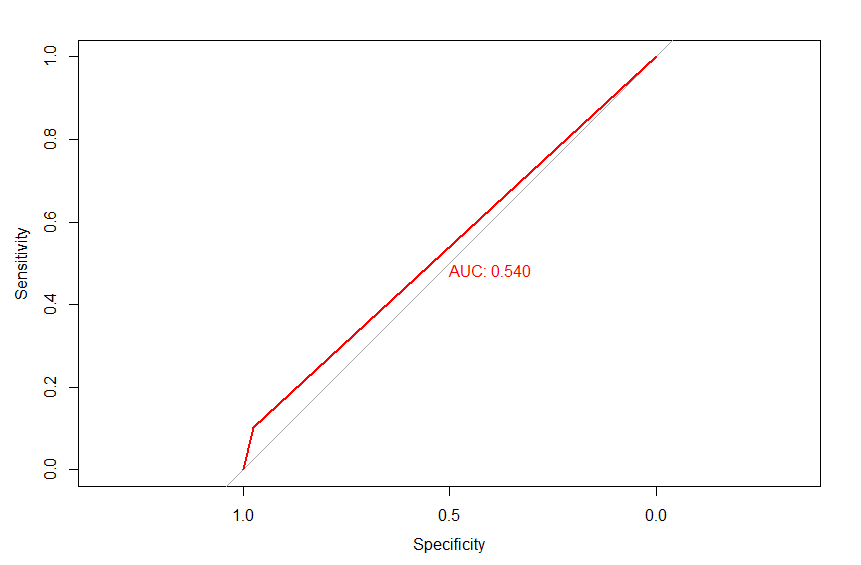
auc.glm

|  |  |  |
| --- | --- | --- |
| Prediction \ Reference | 0 | 1 |
| 0 | 163 | 52 |
| 1 | 4 | 6 |

**Accuracy:** 75.1 %

**Area under the curve:** 0.540

**ROC Curve – SVM**



**KNN:**

knn = train(as.factor(Donated) ~., data=prtrain,method='knn',trControl=trainControl(method='repeatedcv',number=3),tuneGrid=data.frame(k=10))

print(knn)

predicted\_knn= predict(knn,prtest)

confusionMatrix(predicted\_knn,as.factor(Donated))

ROCknn<-roc(Donated,as.numeric(predicted\_knn))

plot(ROCknn)

aucknn<-auc(ROCknn)

aucknn

plot(ROCknn)

|  |  |  |
| --- | --- | --- |
| Prediction \ Reference | 0 | 1 |
| 0 | 161 | 43 |
| 1 | 6 | 15 |

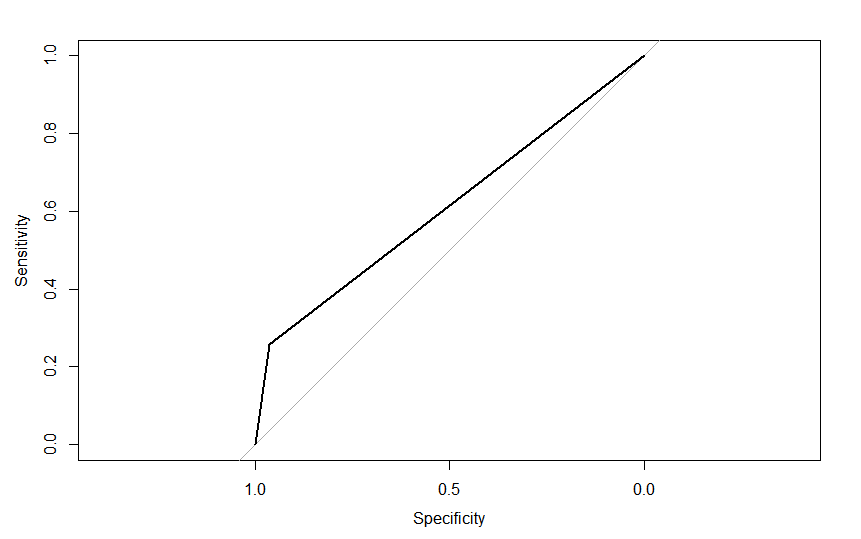
**Accuracy:** 78.2%

**Area under the curve**: 0.611

**Best parameter:**

Tuning parameter 'k' was held constant at a value of 10

Cross-Validated (3 fold, repeated 1 times)

**ROC Curve – KNN**

**Bagging**:

bag <- train(x,y, method="treebag",trControl=ctrl, control=rpart.control(maxdepth=25,cp=0.0010000,minsplit=20,xval=10),iter=100)

predict.bag<-predict(bag,test[1:3])

#Use caret's confusion matrix

conBAG<-confusionMatrix(predict.bag,test$Donated)

#Accuracy and Confusion Matrix

conBAG

#Area under ROC

ROC.bag<-roc(test$Donated,as.numeric(predict.bag))

plot(ROC.bag, col = "red",print.auc=TRUE)

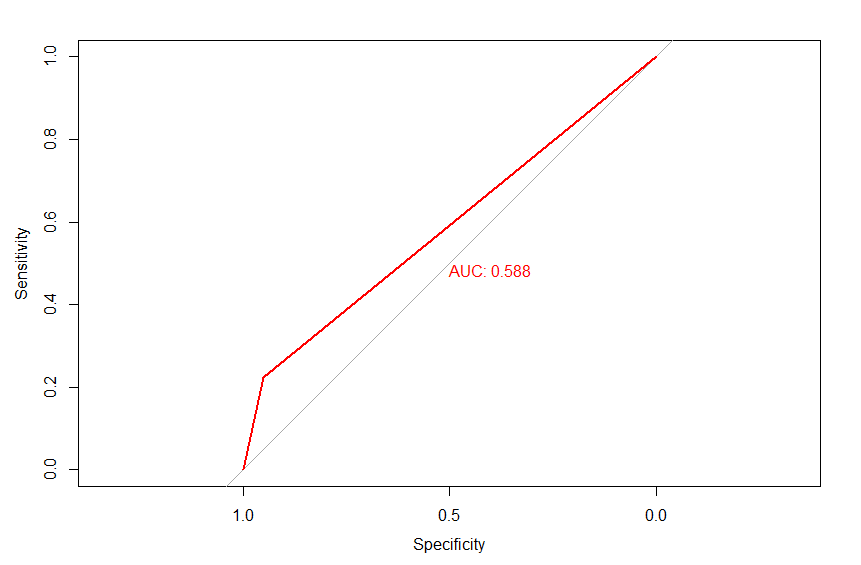
auc.bag<-auc(ROC.bag)

auc.bag

|  |  |  |
| --- | --- | --- |
| Prediction \ Reference | 0 | 1 |
| 0 | 159 | 45 |
| 1 | 8 | 13 |

**Accuracy:** 76.44 %

**Area under the curve:** 0.588

**ROC Curve – Bagging** 

**Random Forest:**

control <- trainControl(method="repeatedcv", number=10, repeats=3, search="random")

randomforest<- train(as.factor(Donated) ~., data=prtrain, method="rf", trControl=control, ntree=20)

print(randomforest)

predicted\_randomforest= (predict(randomforest,prtest))

confusionMatrix(predicted\_randomforest,as.factor(Donated))

ROCrf<-roc(Donated,as.numeric(predicted\_randomforest))

aucrf<-auc(ROCrf)

aucrf

plot(ROCrf)

|  |  |  |
| --- | --- | --- |
| Prediction \ Reference | 0 | 1 |
| 0 | 158 | 41 |
| 1 | 9 | 17 |

**Accuracy:** 77.8%

**Area under the curve**: 0.62

**Best parameter:**

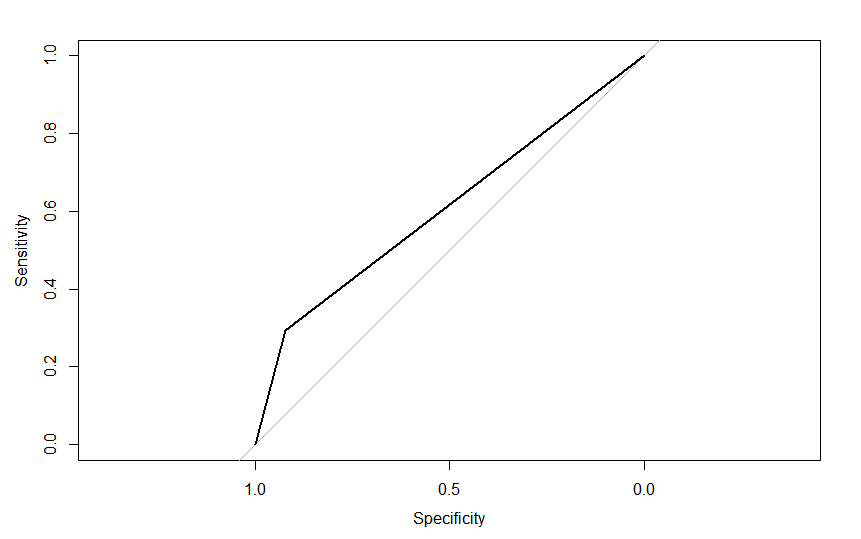
ntree=20

Accuracy was used to select the optimal model using the largest value.

The final value used for the model was mtry = 2

Cross-Validated (10 fold, repeated 3 times)

**ROC Curve – Random Forest**



**AdaBoost:**

adaModel <- ada(Donated ~ .,data=prtrain,control=rpart.control(maxdepth=30,cp=0.0010000,minsplit=20,xval=10),iter=50)

print(adaModel)

predicted\_ada= (predict(adaModel,prtest))

confusionMatrix(predicted\_ada,Donated)

ROCada<-roc(Donated,as.numeric(predicted\_ada))

aucada<-auc(ROCada)

aucada

plot(ROCada)

|  |  |  |
| --- | --- | --- |
| Prediction \ Reference | 0 | 1 |
| 0 | 155 | 42 |
| 1 | 12 | 16 |

**Accuracy:** 76%

**Area under the curve**: 0.602

**Best parameter:**

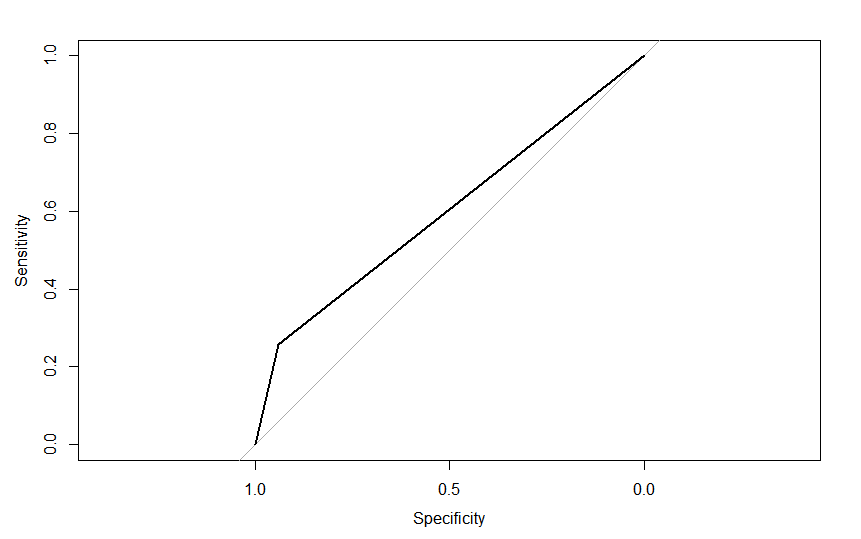
Number of boosting iterations =50

Maximum depth for trees=30

complexity=0.0010000

minimum split=20

k for cross validation =10

**ROC Curve – Adaboost**

**Gradient Boosting:**

control <- trainControl(method="repeatedcv", number=10, repeats=33)

gradientboost <- train(as.factor(Donated) ~ ., data = prtrain, method = "gbm", trControl = control,verbose = FALSE, tuneGrid=data.frame(n.trees = 300, interaction.depth = 1, shrinkage = 0.1 , n.minobsinnode = 10 ))

print(gradientboost)

predicted\_gb= predict(gradientboost,prtest)

confusionMatrix(predicted\_gb,Donated)

ROCgb<-roc(Donated,as.numeric(predicted\_gb))

aucgb<-auc(ROCgb)

aucgb

plot(ROCgb)

|  |  |  |
| --- | --- | --- |
| Prediction \ Reference | 0 | 1 |
| 0 | 161 | 44 |
| 1 | 6 | 14 |

**Accuracy:** 77.8%

**Area under the curve**: 0.603

**Best parameter:**

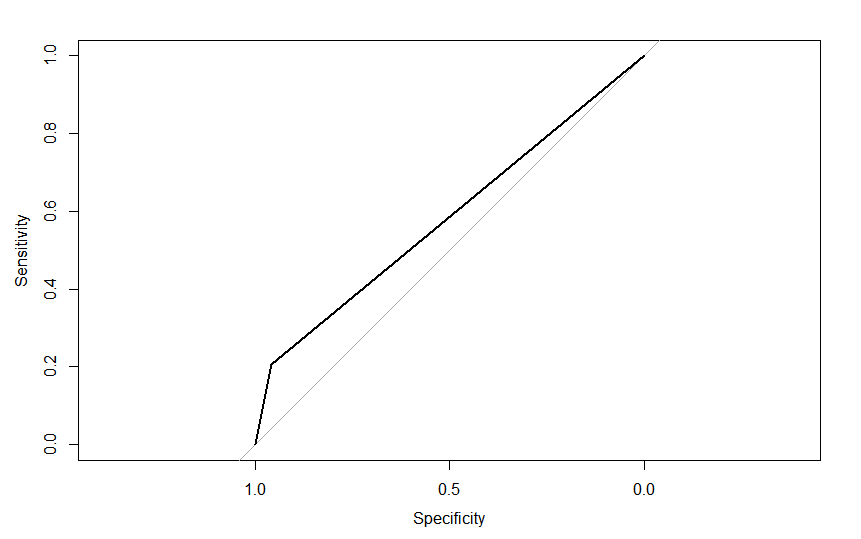
Tuning parameter 'n.trees' was held constant at a value of 300

Tuning parameter 'interaction.depth' was held constant at a value of 1

Tuning parameter 'shrinkage' was held constant at a value of 0.1

Tuning parameter 'n.minobsinnode' was held constant at a value of 10

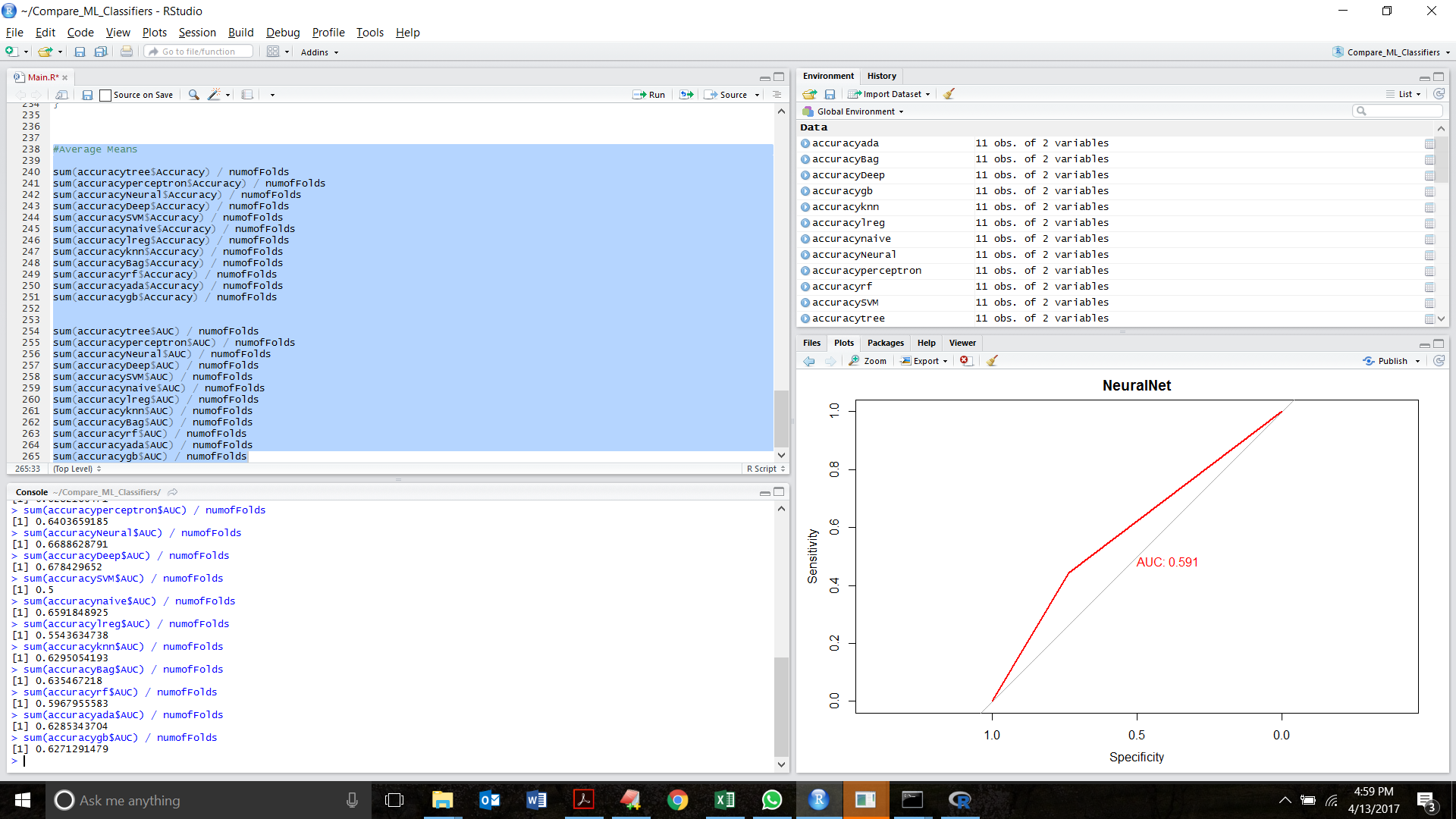
Cross-Validated (10 fold, repeated 3 times)

**ROC Curve – Gradient Boosting**

|  |  |  |
| --- | --- | --- |
| Classifier | Avg Accuracy on 10 Folds | AUC on 10 Folds |
| Decision Trees | 77.81441441 | 0.628216047 |
| Perceptron | 77.81981982 | 0.640365919 |
| Neural Net | 64.84684685 | 0.668862879 |
| Deep Learning | 67.65225225 | 0.678429652 |
| SVM | 76.21081081 | 0.5 |
| naïve Bayes | 75.67387387 | 0.659184893 |
| Logistic Regression | 77.54954955 | 0.554363474 |
| k-Nearest Neighbors | 79.15315315 | 0.629505419 |
| Bagging | 78.88108108 | 0.635467218 |
| Random Forests | 76.07567568 | 0.596795558 |
| AdaBoost | 77.14234234 | 0.62853437 |
| Gradient Boosting | 78.74954955 | 0.627129148 |

**Cross Validation Results:**

Average results after running 10-fold cross validation on all the classifiers simultaneously.



**Results:**

> #Average Means

>

> sum(accuracytree$Accuracy) / numofFolds

[1] 77.81441441

> sum(accuracyperceptron$Accuracy) / numofFolds

[1] 77.81981982

> sum(accuracyNeural$Accuracy) / numofFolds

[1] 64.84684685

> sum(accuracyDeep$Accuracy) / numofFolds

[1] 67.65225225

> sum(accuracySVM$Accuracy) / numofFolds

[1] 76.21081081

> sum(accuracynaive$Accuracy) / numofFolds

[1] 75.67387387

> sum(accuracylreg$Accuracy) / numofFolds

[1] 77.54954955

> sum(accuracyknn$Accuracy) / numofFolds

[1] 79.15315315

> sum(accuracyBag$Accuracy) / numofFolds

[1] 78.88108108

> sum(accuracyrf$Accuracy) / numofFolds

[1] 76.07567568

> sum(accuracyada$Accuracy) / numofFolds

[1] 77.14234234

> sum(accuracygb$Accuracy) / numofFolds

[1] 78.74954955

>

>

> sum(accuracytree$AUC) / numofFolds

[1] 0.6282160471

> sum(accuracyperceptron$AUC) / numofFolds

[1] 0.6403659185

> sum(accuracyNeural$AUC) / numofFolds

[1] 0.6688628791

> sum(accuracyDeep$AUC) / numofFolds

[1] 0.678429652

> sum(accuracySVM$AUC) / numofFolds

[1] 0.5

> sum(accuracynaive$AUC) / numofFolds

[1] 0.6591848925

> sum(accuracylreg$AUC) / numofFolds

[1] 0.5543634738

> sum(accuracyknn$AUC) / numofFolds

[1] 0.6295054193

> sum(accuracyBag$AUC) / numofFolds

[1] 0.635467218

> sum(accuracyrf$AUC) / numofFolds

[1] 0.5967955583

> sum(accuracyada$AUC) / numofFolds

[1] 0.6285343704

> sum(accuracygb$AUC) / numofFolds

[1] 0.6271291479