##Data Science for Business DECISION 520Q

##Section C Team 06

##Bryan Huang, Weiqi Li, Yashuo Wang (Gloria), Van Xu, Jose Zuart

###Term Project Report R Code

#Import the data

rawdata <- read.csv("Training Data.csv")

#Remove "ID"

rawdata <- rawdata[,-1]

#Set seed

set.seed(1)

#Split 50000 rows into holdout

holdout.indices <- sample(nrow(rawdata), 50000)

#Split the training data and holdout data

data.holdout <- rawdata[holdout.indices,]

data <- rawdata[-holdout.indices,]

#Import data analytic functions

source("DataAnalyticsFunctions.R")

source("PerformanceCurves.R")

#Just in case, redefine some important functions.

roc <- function(p,y, ...){

y <- factor(y)

n <- length(p)

p <- as.vector(p)

Q <- p > matrix(rep(seq(0,1,length=100),n),ncol=100,byrow=TRUE)

specificity <- colMeans(!Q[y==levels(y)[1],])

sensitivity <- colMeans(Q[y==levels(y)[2],])

plot(1-specificity, sensitivity, ylab="TP", xlab="FPR",type="l", main="ROC Curve", ...)

abline(a=0,b=1,lty=2,col=8)

ROCcurve <-as.data.frame( cbind( 1-specificity, sensitivity))

return (ROCcurve)

}

installpkg <- function(x){

if(x %in% rownames(installed.packages())==FALSE) {

if(x %in% rownames(available.packages())==FALSE) {

paste(x,"is not a valid package - please check again...")

} else {

install.packages(x)

}

} else {

paste(x,"package already installed...")

}

}

#Read packages

installpkg("tree")

library(tree)

installpkg("partykit")

library(partykit)

installpkg("randomForest")

library(randomForest)

installpkg("ggplot2")

installpkg("GGally")

library(ggplot2)

library(GGally)

installpkg("glmnet")

library(glmnet)

installpkg("pROC")

library(pROC)

installpkg("randomForest")

library(randomForest)

#K-mean

xdata <- model.matrix(Risk\_Flag ~ ., data=data)[,-1]

xdata <- scale(xdata)

#Try a k-mean cluster with 17 groups

FourCenters <- kmeans(xdata,17,nstart=30)

FourCenters

#PCA

#Compute the full PCA

pca.data <- prcomp(xdata, scale=TRUE)

### Lets plot the variance that each component explains

par(mar=c(4,4,4,4)+0.3)

plot(pca.data,main="PCA: Variance Explained by Factors")

mtext(side=1, "Factors", line=1, font=2)

loadings <- pca.data$rotation[,1:4]

#### Looking at which are large positive and large negative

v<-loadings[order(abs(loadings[,1]), decreasing=TRUE)[1:ncol(xdata)],1]

##Look at what compute the PC

loadingfit <- lapply(1:ncol(xdata), function(k) ( t(v[1:k])%\*%v[1:k] - 3/4 )^2)

v[1:which.min(loadingfit)]

v<-loadings[order(abs(loadings[,2]), decreasing=TRUE)[1:ncol(xdata)],2]

loadingfit <- lapply(1:ncol(xdata), function(k) ( t(v[1:k])%\*%v[1:k] - 3/4 )^2)

v[1:which.min(loadingfit)]

v<-loadings[order(abs(loadings[,3]), decreasing=TRUE)[1:ncol(xdata)],3]

loadingfit <- lapply(1:ncol(xdata), function(k) ( t(v[1:k])%\*%v[1:k] - 3/4 )^2)

v[1:which.min(loadingfit)]

v<-loadings[order(abs(loadings[,4]), decreasing=TRUE)[1:ncol(xdata)],3]

loadingfit <- lapply(1:ncol(xdata), function(k) ( t(v[1:k])%\*%v[1:k] - 3/4 )^2)

v[1:which.min(loadingfit)]

PerformanceMeasureOOSACC <- function(actual, prediction, threshold=.50) {

#1-mean( abs( (prediction>threshold) - actual ) )

#R2(y=actual, pred=prediction, family="binomial")

1-mean( abs( (prediction- actual) ) )

}

#Import Performance Measure function(OOS Accuracy)

PerformanceMeasureOOSACC <- function(actual, prediction, threshold=.50) {

#1-mean( abs( (prediction>threshold) - actual ) )

#R2(y=actual, pred=prediction, family="binomial")

1-mean( abs( (prediction- actual) ) )

}

#Create X and Y vector from training data for Lasso

Mx<- model.matrix(Risk\_Flag ~ ., data=data)[,-1]

My<- data$Risk\_Flag == 1

#Run Lasso

lasso <- glmnet(Mx,My, family="binomial")

lassoCV <- cv.glmnet(Mx,My, family="binomial")

#Create data for post Lasso

features.min <- support(lasso$beta[,which.min(lassoCV$cvm)])

length(features.min)

data.min <- data.frame(Mx[,features.min],My)

#Create new x from holdout data

Test <- model.matrix(~. - Risk\_Flag, data=data.holdout)[,-1]

#Create empty frame for Cross Validation

n <- nrow(data)

nfold <- 10

OOS <- data.frame(m.lr=rep(NA,nfold), m.lr.l=rep(NA,nfold), m.lr.pl=rep(NA,nfold), m.tree=rep(NA,nfold), m.average=rep(NA,nfold))

foldid <- rep(1:nfold,each=ceiling(n/nfold))[sample(1:n)]

#Cross Validation(OSS Accuracy) for Logistic regression, Post Lasso, Lasso, and Classification Tree.

for(k in 1:nfold){

train <- which(foldid!=k) # train on all but fold `k'

### Logistic regression

m.lr <-glm(Risk\_Flag~., data=data, subset=train,family="binomial")

pred.lr <- predict(m.lr, newdata=data[-train,], type="response")

OOS$m.lr[k] <- PerformanceMeasureOOSACC(actual=My[-train], pred=pred.lr)

### the Post Lasso Estimates

m.lr.pl <- glm(My~., data=data.min, subset=train, family="binomial")

pred.lr.pl <- predict(m.lr.pl, newdata=data.min[-train,], type="response")

OOS$m.lr.pl[k] <- PerformanceMeasureOOSACC(actual=My[-train], prediction=pred.lr.pl)

### the Lasso estimates

m.lr.l <- glmnet(Mx[train,],My[train], family="binomial",lambda = lassoCV$lambda.min)

pred.lr.l <- predict(m.lr.l, newx=Mx[-train,], type="response")

OOS$m.lr.l[k] <- PerformanceMeasureOOSACC(actual=My[-train], prediction=pred.lr.l)

### the classification tree

m.tree <- tree(Risk\_Flag~ ., data=data, subset=train)

pred.tree <- predict(m.tree, newdata=data[-train,], type="vector")

#pred.tree <- pred.tree[,2]

OOS$m.tree[k] <- PerformanceMeasureOOSACC(actual=My[-train], prediction=pred.tree)

### the overall average

pred.m.average <- rowMeans(cbind(pred.tree, pred.lr.l, pred.lr.pl, pred.lr, pred.lr))

OOS$m.average[k] <- PerformanceMeasureOOSACC(actual=My[-train], prediction=pred.m.average)

print(paste("Iteration",k,"of",nfold,"completed"))

}

#Plot and compare

par(mar=c(7,5,.5,1)+0.3)

barplot(colMeans(OOS), las=2,xpd=FALSE , xlab="", ylim=c(0.9\*min(colMeans(OOS)),max(colMeans(OOS))), ylab = bquote( "Average Out of Sample Performance"))

###Import Performance Measure function(OOS R^2)

PerformanceMeasureOOSR2 <- function(actual, prediction, threshold=.50) {

#1-mean( abs( (prediction>threshold) - actual ) )

R2(y=actual, pred=prediction, family="binomial")

#1-mean( abs( (prediction- actual) ) )

}

#Create empty frame for Cross Validation

n <- nrow(data)

nfold <- 10

OOS <- data.frame(m.lr=rep(NA,nfold), m.lr.l=rep(NA,nfold), m.lr.pl=rep(NA,nfold), m.tree=rep(NA,nfold), m.average=rep(NA,nfold))

#names(OOS)<- c("Logistic Regression", "Lasso on LR with Interactions", "Post Lasso on LR with Interactions", "Classification Tree", "Average of Models")

foldid <- rep(1:nfold,each=ceiling(n/nfold))[sample(1:n)]

#Cross Validation(OSS R^2) for Logistic regression, Post Lasso, Lasso, and Classification Tree.

for(k in 1:nfold){

train <- which(foldid!=k) # train on all but fold `k'

### Logistic regression

m.lr <-glm(Risk\_Flag~., data=data, subset=train,family="binomial")

pred.lr <- predict(m.lr, newdata=data[-train,], type="response")

OOS$m.lr[k] <- PerformanceMeasureOOSR2(actual=My[-train], pred=pred.lr)

### the Post Lasso Estimates

m.lr.pl <- glm(My~., data=data.min, subset=train, family="binomial")

pred.lr.pl <- predict(m.lr.pl, newdata=data.min[-train,], type="response")

OOS$m.lr.pl[k] <- PerformanceMeasureOOSR2(actual=My[-train], prediction=pred.lr.pl)

### the Lasso estimates

m.lr.l <- glmnet(Mx[train,],My[train], family="binomial",lambda = lassoCV$lambda.min)

pred.lr.l <- predict(m.lr.l, newx=Mx[-train,], type="response")

OOS$m.lr.l[k] <- PerformanceMeasureOOSR2(actual=My[-train], prediction=pred.lr.l)

### the classification tree

m.tree <- tree(Risk\_Flag~ ., data=data, subset=train)

pred.tree <- predict(m.tree, newdata=data[-train,], type="vector")

#pred.tree <- pred.tree[,2]

OOS$m.tree[k] <- PerformanceMeasureOOSR2(actual=My[-train], prediction=pred.tree)

pred.m.average <- rowMeans(cbind(pred.tree, pred.lr.l, pred.lr.pl, pred.lr, pred.lr))

OOS$m.average[k] <- PerformanceMeasureOOSR2(actual=My[-train], prediction=pred.m.average)

print(paste("Iteration",k,"of",nfold,"completed"))

}

#Plot and compare

par(mar=c(7,5,.5,1)+0.3)

barplot(colMeans(OOS), las=2,xpd=FALSE , xlab="", ylim=c(0.9\*min(colMeans(OOS)),max(colMeans(OOS))), ylab = bquote( "Average Out of Sample Performance"))

#Create x and y from the holdout data

MxH<- model.matrix(Risk\_Flag ~ ., data=data.holdout)[,-1]

MyH<- data.holdout$Risk\_Flag

##Run Lasso

m.lr.l <- glmnet(Mx,My, family="binomial",lambda = lassoCV$lambda.min)

#Predict on the holdout data

pred.lr.l <- predict(m.lr.l, newx=Test, type="response")

pred.lr.l

#Check the performance using OOS Accuracy, ROC and AUC.

PerformanceMeasureOOSACC(MyH, prediction=pred.lr.l)

auc(MyH, pred.lr.l)

lr.lroc <- roc(p=pred.lr.l,y=MyH)

##Run Post Lasso

data.minPL <- data.frame(MxH[,features.min],MyH)

m.lr.pl <- glm(My~., data=data.min, family="binomial")

#Predict on the holdout data

pred.lr.pl <- predict(m.lr.pl, newdata=data.minPL, type="response")

pred.lr.pl

#Check the performance using OOS Accuracy, ROC and AUC.

PerformanceMeasureOOSACC(MyH, prediction=pred.lr.pl)

auc(MyH, pred.lr.pl)

roc(p=pred.lr.pl,y=MyH)

#The Classification Tree

tree <- tree(Risk\_Flag~., data=data)

#Predict on the holdout data

pre.tree <- predict(tree, newdata=data.holdout)

pre.tree

#Check the performance using OOS Accuracy, ROC and AUC.

PerformanceMeasureOOSACC(MyH, prediction=pre.tree)

auc(MyH, pre.tree)

roc(p=pre.tree,y=MyH)

#Logistic Regression

m.lr <-glm(Risk\_Flag~., data=data,family="binomial")

#Predict on the holdout data

pred.lr <- predict(m.lr, newdata=data.holdout, type="response")

pred.lr

#Check the performance using OOS Accuracy, ROC and AUC.

PerformanceMeasureOOSACC(MyH, prediction=pred.lr)

auc(MyH, pred.lr)

roc(p=pred.lr,y=MyH)

##Random Forest

model1 <- randomForest(data$Risk\_Flag~., data=data, nodesize=5, ntree = 500, mtry = 4)

model1

#Predict on the holdout data

pred.tree <- predict(model1,newdata = data.holdout,type="response")

#Check the performance using OOS Accuracy, ROC and AUC.

myRoc <- roc(p=pred.tree,y=MyH)

auc(MyH,pred.tree)

pred.tree

PerformanceMeasureOOSACC(MyH, prediction=pred.tree)

#Check the performance using FPR\_TPR

PL.performance1 <- FPR\_TPR(pred.tree>=0.1 , MyH)

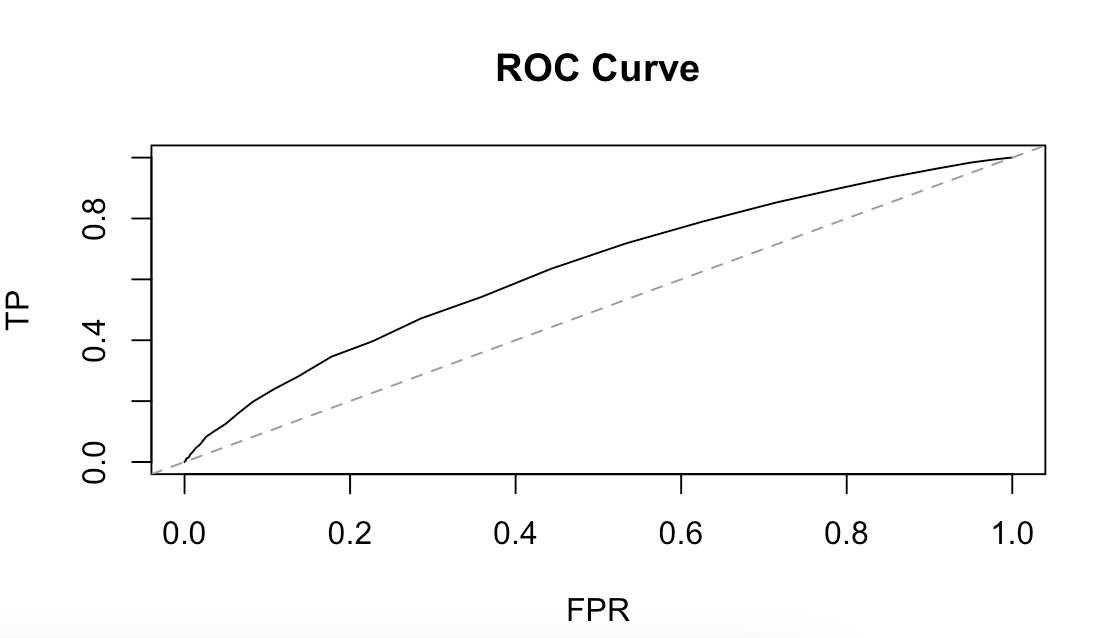
PL.performance1

Lasso

-PerformanceMeasureOOSACC(MyH, prediction=pred.lr.l)

[1] 0.7897643

Area under the curve: 0.6363

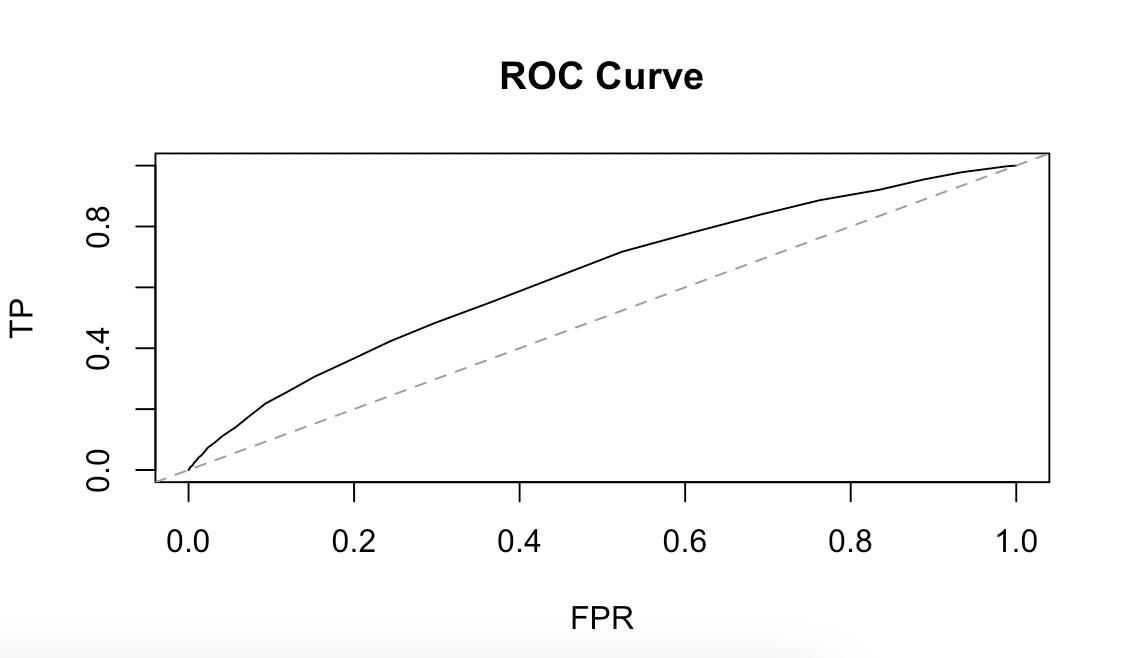


Post Lasso

-PerformanceMeasureOOSACC(MyH, prediction=pred.lr.pl)

[1] 0.7901111

Area under the curve: 0.6367

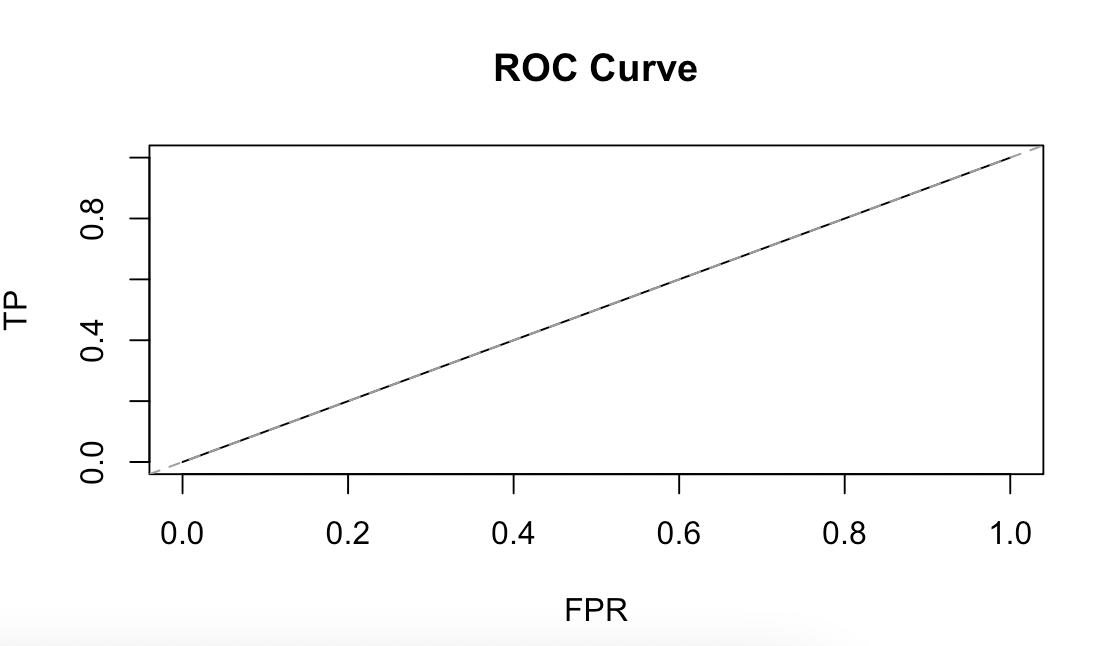


Classification tree

> PerformanceMeasureOOSACC(MyH, prediction=pre.tree)

[1] 0.7844849

-Area under the curve: 0.5

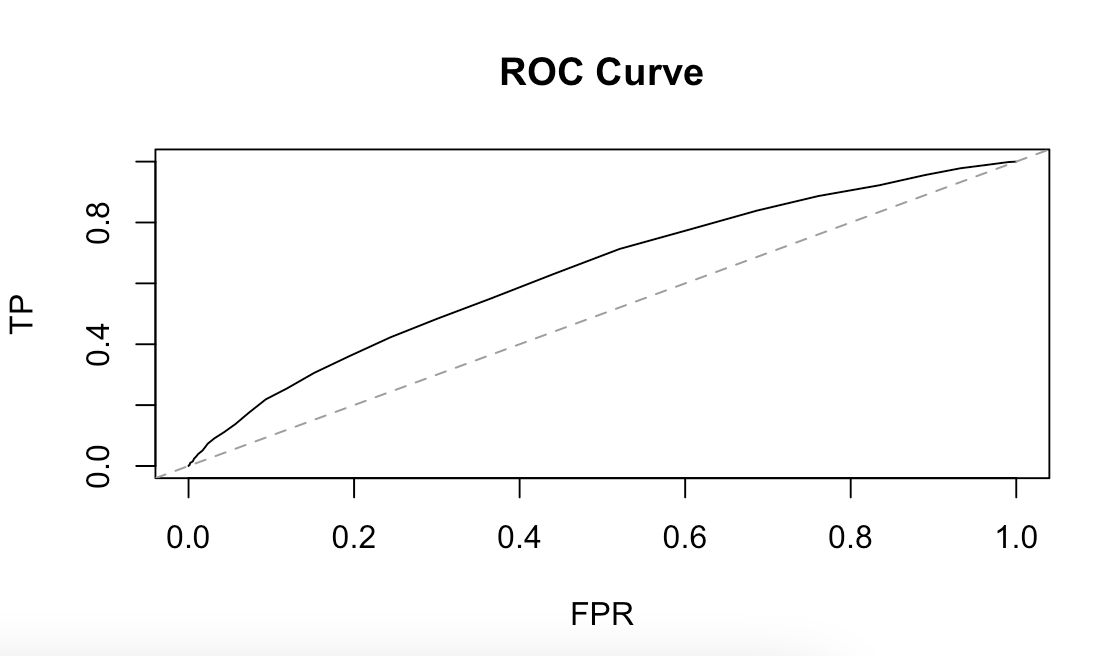


Logistic Regression

> PerformanceMeasureOOSACC(MyH, prediction=pred.lr)

[1] 0.790094

Area under the curve: 0.6367

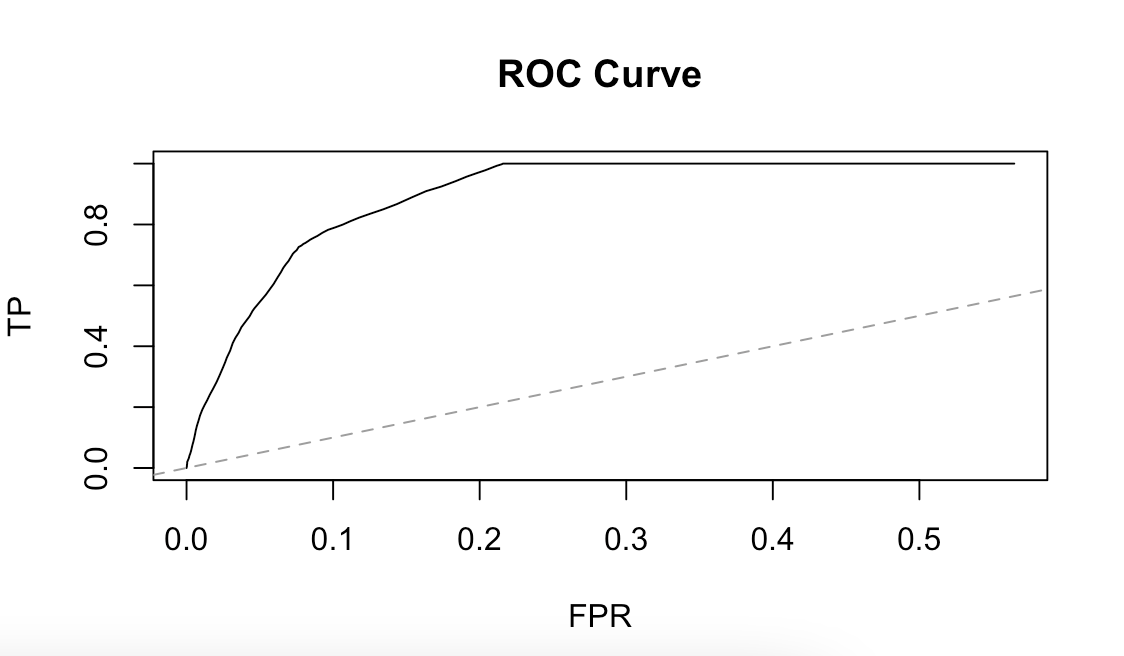


Random Forest

> PerformanceMeasureOOSACC(MyH, prediction=pred.tree)

[1] 0.875104

Area under the curve: 0.9387



threshold=0.1

