```
# ECO4444 Final
# authors: Cody Olivotto and Adeeb Salim
# dependencies: stargazer, fBasics, caret, ROCR, dplyr, boot

# import Libraries
library(caret)
library(ROCR)
library(dplyr)
library(fBasics) #generates summary statistics
library(boot)
library(stargazer) #generates pretty summary statistics for models

#import orignal Dataset
hmda_data <- read.csv("./rawdata/hmda_sw.txt", sep = '\t', header = TRUE)

# Convert NA variables for education and omit them (results in a .009% sample
 size loss)
hmda_data$school[hmda_data$school == 999999.4]<- NA
hmda_cleaned_data <- hmda_data[!is.na(hmda_data$school),]
attach(hmda_cleaned_data)



#---------------------------- A. create variables -----------------------------

####   Mandatory Variables and their respective names:
# Di_ratio = TotalDebtPmt_Income (from s46)
# Race = nonWhite (from S13)
# Self-employed = SelfEmp (from s27a)
# Marital Status = NotMarried (from s23a)
# education = Education_yrs (from school)

HouseExp_Income <- ifelse(s45>30.0,1,0)
TotalDebtPmt_Income <- s46 #di_ratio
netWealth <- netw
cCredHist <- s43
mCredHist <- s42
pubRec <- s44
Unemply <- uria
SelfEmp <- s27a
loanAppraised_Low <- ifelse(s6/s50<=.8,1,0)
loanAppraised_Medium <- ifelse((.95>s6/s50 & s6/s50>.80),1,0)
loanAppraised_High <- ifelse(s6/s50>.95, 1,0)
DeniedPMi <- s53
nonWhite <- ifelse(s13==5,0,1)
NotMarried <- ifelse(s23a=="M",0,1)
Education_yrs <- school
denied <-  ifelse((s7==3),1,0)

# Establish new data.frame w/ selected x, y (15 exogs)
data <- data.frame(denied,HouseExp_Income,TotalDebtPmt_Income,netWealth,
```

```
    cCredHist,mCredHist,pubRec,Unemply,SelfEmp, loanAppraised_Low,
    loanAppraised_Medium, loanAppraised_High, DeniedPMi, nonWhite, NotMarried,
    Education_yrs)



#----------------------- B. Summary Statistics & Graphs ----------------------

dataSummary <- basicStats(data)[c("Mean", "Median", "Mode", "Stdev",
    "Variance", "Minimum", "Maximum", "nobs"),]
dataSummary

#### Creating Graphics

# The dependent variable (Denied)
denied_perc <- t(prop.table(table(data$denied))) * 100
denied_perc # where Denied == 1 ; 87.95% of applicants are approved

# Race
nonWhite_perc <- t(prop.table(table(data$nonWhite))) * 100

# Education
edu_perc <- t(prop.table(table(data$Education_yrs))) * 100

# Denied PMI
DeniedPMI_perc <- t(prop.table(table(data$DeniedPMi))) * 100

# Married
NotMarried_perc <- t(prop.table(table(data$NotMarried))) * 100

### Run this all as one line of Code to Establish a matrix of graphs
par(mfrow=c(2,3))
barplot(denied_perc, main="Verdict of Applicants Loan Approval",ylab =
 "Percent",
        col = "orange", xlab = "Denied = 1")
barplot(nonWhite_perc, main = "Demographics of Applicants", ylab = "Percent",
        col = "purple", xlab = "White = 0 ",)
barplot(edu_perc, main = "Applicant Education Levels",ylab = "Percent",
        col = "gold", xlab = "Years of School",)
plot(netWealth, log="y", col = "green", main = "Distribution of Wealth by
 Applicant")
barplot(DeniedPMI_perc,main="Percent of Applicants denied PMI", ylab =
 "Percent",
        col = "red", xlab = "DeniedPMI = 1",)
barplot(NotMarried_perc, main="Percent of Applicants who are Unmarried",
        ylab = "Percent", col = "tan", xlab = "UnMarried = 1",)

#declutter
rm(denied_perc, nonWhite_perc, edu_perc, DeniedPMI_perc, NotMarried_perc)

#---------------- C. Logistic Regression and Model Evaluation ----------------
```

```
# Prep the Data
DF_C <- data.frame(denied, TotalDebtPmt_Income, nonWhite, SelfEmp, NotMarried,
 Education_yrs)
DF_C <-na.omit(DF_C) # Removed 1 additional N/A observation that was
 disrupting ROCR
set.seed(123)
r <- sample(nrow(DF_C))
DF_C <-DF_C[r,]

model_c <- glm(DF_C$denied~., data = DF_C, family=binomial)
pred_prob_c <- predict.glm(model_c, type = c("response"))

pred_c <- prediction(pred_prob_c,DF_C$denied)
perf_c <- performance(pred_c,measure = "tpr", x.measure = "fpr")

# Plot ROC curve
plot(perf_c, colorize=TRUE)
abline(a=0,b=1)

# Area under the Curve (AUC) for Full Sample model C
model_C_auc <-
 unlist(slot(performance(prediction(pred_prob_c,DF_C$denied),"auc"),"y
 .values"))

#### Confusion Matrix
# Cutoff
tableofDenial <- table(DF_C$denied) # to find tpr fpr etc.
cutoff_ <- c("specificity","fpr","sensitivity","fnr","accuracy",
 "misclassification Rate") #names for data frame

## Confusion Matrix from base R (with loop to check 5 different cutoff levels
 without using CV)
confmats_c <- vector(mode= "list", length = 5)
cutoffSEQ <- seq(0.1,.9, 0.2)
for (i in 1:length(cutoffSEQ)){
  class_prediction_mc <- ifelse(pred_prob_c > cutoffSEQ[i], "positive_class",
   "negative_class")
  c_denied <- table(DF_C)
  confmats_c[[i]] <- table(class_prediction_mc, DF_C$denied)
}



#-------------- C.1 Building Data Frame For performance Results ---------------

# 10% Cutoff
c_10percentCutoff <-confmats_c[[1]]
#[1,1]
tpr <- c_10percentCutoff[2,2]/tableofDenial[[2]]  # "sensitivity"
fpr <- c_10percentCutoff[2,1]/tableofDenial[[1]]
```

```
tnr <- c_10percentCutoff[1,1]/tableofDenial[[1]]  # "specificity"
fnr <- c_10percentCutoff[1,2]/tableofDenial[[2]]
accuracy <- (c_10percentCutoff[2,2] +
 c_10percentCutoff[1,1])/length(DF_C$denied)
misclass_rate <- 1-accuracy
Ten_percent <- c(tpr,fpr,tnr,fnr,accuracy,misclass_rate)

# 30% Cutoff
c_30percentCutoff <-confmats_c[[2]]
tpr <- c_30percentCutoff[2,2]/tableofDenial[[2]]
fpr <- c_30percentCutoff[2,1]/tableofDenial[[1]]
tnr <- c_30percentCutoff[1,1]/tableofDenial[[1]]
fnr <- c_30percentCutoff[1,2]/tableofDenial[[2]]
accuracy <- (c_30percentCutoff[2,2] +
 c_30percentCutoff[1,1])/length(DF_C$denied)
misclass_rate <- 1-accuracy
Thirty_percent <- c(tpr,fpr,tnr,fnr,accuracy,misclass_rate)

# 50% Cutoff
c_50percentCutoff <-confmats_c[[3]]
tpr <- c_50percentCutoff[2,2]/tableofDenial[[2]]
fpr <- c_50percentCutoff[2,1]/tableofDenial[[1]]
tnr <- c_50percentCutoff[1,1]/tableofDenial[[1]]
fnr <- c_50percentCutoff[1,2]/tableofDenial[[2]]
accuracy <- (c_50percentCutoff[2,2] +
 c_50percentCutoff[1,1])/length(DF_C$denied)
misclass_rate <- 1-accuracy
Fifty_percent <- c(tpr,fpr,tnr,fnr,accuracy,misclass_rate)

# 70% Cutoff
c_70percentCutoff <-confmats_c[[4]]
tpr <- c_70percentCutoff[2,2]/tableofDenial[[2]]
fpr <- c_70percentCutoff[2,1]/tableofDenial[[1]]
tnr <- c_70percentCutoff[1,1]/tableofDenial[[1]]
fnr <- c_70percentCutoff[1,2]/tableofDenial[[2]]
accuracy <- (c_70percentCutoff[2,2] +
 c_70percentCutoff[1,1])/length(DF_C$denied)
misclass_rate <- 1-accuracy
seventy_percent <- c(tpr,fpr,tnr,fnr,accuracy,misclass_rate)

# 90% Cutoff
c_90percentCutoff <-confmats_c[[5]]
tpr <- c_90percentCutoff[2,2]/tableofDenial[[2]]
fpr <- c_90percentCutoff[2,1]/tableofDenial[[1]]
tnr <- c_90percentCutoff[1,1]/tableofDenial[[1]]
fnr <- c_90percentCutoff[1,2]/tableofDenial[[2]]
accuracy <- (c_90percentCutoff[2,2] +
 c_90percentCutoff[1,1])/length(DF_C$denied)
misclass_rate <- 1-accuracy
ninety_percent <- c(tpr,fpr,tnr,fnr,accuracy,misclass_rate)
```

```
#### c.1 Performance Results
model_C_performance <-data.frame(cutoff_,Ten_percent,Thirty_percent,
                                 Fifty_percent,seventy_percent,ninety_percent)
model_C_performance



#-------------------- C.2 Using A Carats confusion matrix ---------------------

# Confusion Matrix and Statistics # 10% Cutoff
c_10percentCutoff # hardcoded values to compare matricies
#confusionMatrix(actuals = DF_C$denied, predictedScores = pred_prob_c,
 threshold = .1)

# Confusion Matrix and Statistics # 30% Cutoff
c_30percentCutoff
#confusionMatrix(actuals = DF_C$denied, predictedScores = pred_prob_c,
 threshold = .3)

# Confusion Matrix and Statistics # 50% Cutoff
c_50percentCutoff
#confusionMatrix(actuals = DF_C$denied, predictedScores = pred_prob_c,
 threshold = .5)

# Confusion Matrix and Statistics # 70% Cutoff
c_70percentCutoff
#confusionMatrix(actuals = DF_C$denied, predictedScores = pred_prob_c,
 threshold = .7)

# Confusion Matrix and Statistics # 90% Cutoff
c_90percentCutoff
#confusionMatrix(actuals = DF_C$denied, predictedScores = pred_prob_c,
 threshold = .9)

#cleanup
rm(confmats_c, accuracy, tpr, fpr, tnr, fnr, misclass_rate, Ten_percent,
   Thirty_percent, Fifty_percent, seventy_percent, ninety_percent, cutoff_,
   cutoffSEQ, i, r, tableofDenial, c_10percentCutoff, c_30percentCutoff,
   c_50percentCutoff, c_70percentCutoff, c_90percentCutoff,
   class_prediction_mc)



#-------------------- D. Variety of logistic regressions ---------------------

#omit the same NA observation like in section c.
data<-na.omit(data)

##Model with all variables, to view variable importance
fullSet <- data
fullBasicModel <- glm(fullSet$denied~., family = binomial, data = fullSet)
```

```
stargazer(fullBasicModel, type = 'text')
#found a high correlation between denied mortgage, so removed to see rest
basicModelNoDeniedPMI <- glm(fullSet$denied~ . - DeniedPMi, family = binomial,
 data=fullSet)
stargazer(basicModelNoDeniedPMI, type = 'text')

## Model 1 ---------------------------------------------------------------------
#Clean original data frame and new ones to be subdivided into test/train splits
data<-na.omit(data)
rm(HouseExp_Income,TotalDebtPmt_Income,netWealth,cCredHist,
   mCredHist,pubRec,Unemply,SelfEmp,loanAppraised_Low,loanAppraised_High,
   loanAppraised_Medium,DeniedPMi,nonWhite,NotMarried,Education_yrs,denied)
# data frames
DF_D1 <- data.frame(data$denied,I(data$HouseExp_Income*data$netWealth),
                    data$cCredHist, data$mCredHist,data$pubRec,data$Unemply,
                    data$SelfEmp,data$DeniedPMi,data$nonWhite, data$NotMarried,
                    data$Education_yrs,data$DeniedPMi,
                    I(data$Education_yrs*data$DeniedPMi))

# index
index <- sample(nrow(DF_D1),nrow(DF_D1)*.80)
# splits
train_DF_D1 = data[index,]
test_DF_D1 = data[-index,]

# define cost function
costfunc <- function(denied, pred_prob_train_D1) {
    weight1 <- 1
    weight0 <- 1
    c1 <- (denied==1)&(pred_prob_train_D1<optimal_cutoff)
    c0 <- (denied==0)&(pred_prob_train_D1>=optimal_cutoff)
    cost<- mean(weight1*c1+weight0*c0)
    return(cost)
}

# estimate model
model_D1 <-glm(denied~.,data = train_DF_D1,family = binomial)

###training
pred_prob_train_D1 <- predict.glm(model_D1,type = "response") #this is
 predprob for cost function
pred_train_D1 <- prediction(pred_prob_train_D1, train_DF_D1$denied)
perf_train_D1 <- performance(pred_train_D1, "tpr", "fpr")
plot(perf_train_D1,colorize=TRUE)
abline(a=0,b=1)
# AUC training set
d1_train_auc <- unlist(slot(performance(pred_train_D1,"auc",),"y.values"))
d1_train_auc

# Testing
```

```
pred_prob_test_d1 <- predict.glm(model_D1,newdata = test_DF_D1, type =
 "response")
pred_test_d1 <-prediction(pred_prob_test_d1,test_DF_D1$denied)
perf_test_d1 <- performance(pred_test_d1,"tpr","fpr")

plot(perf_test_d1,colorize=TRUE, add=TRUE)
# AUC Test set
d1_test_auc <- unlist(slot(performance(pred_test_d1,"auc"),"y.values"))
 #reduction in area under curve
d1_test_auc


prob_seq <- seq(0.01, 1, 0.01)

cv_cost1= rep(0,length(prob_seq))
for (i in 1:length(prob_seq)){
    optimal_cutoff = prob_seq[i]
    set.seed(123)
    cv_cost1[i] = cv.glm(data=train_DF_D1, glmfit = model_D1, cost=costfunc,
     K=10)$delta[2]
}

plot(prob_seq,cv_cost1)
optimal_cutoff_cv1 = prob_seq[which(cv_cost1==min(cv_cost1))]
optimal_cutoff_cv1 #.58

# train Classification
trainClass_D1 <-ifelse(pred_prob_train_D1>optimal_cutoff_cv1,1,0)
trainClass_D1 <- factor(trainClass_D1)
train_deny <- factor(train_DF_D1$denied)
D1train_confm <-confusionMatrix(actuals = train_DF_D1$denied,
                                predictedScores = pred_prob_train_D1,
                                threshold = optimal_cutoff_cv1)
D1train_confm

# test classification
testclass_d1 <- ifelse(pred_prob_test_d1>optimal_cutoff_cv1,1,0)
testclass_d1 <-factor(testclass_d1)
test_deny <- factor(test_DF_D1$denied)
D1test_confm <- confusionMatrix(actuals = test_DF_D1$denied,
                                predictedScores = pred_prob_test_d1,
                                threshold = optimal_cutoff_cv1)
D1test_confm

#misclassification error
d1_train_mce <- misClassError(actuals = train_DF_D1$denied,
                              predictedScores = pred_prob_train_D1,
                              threshold = optimal_cutoff_cv1)
d1_train_mce

## Model 2 ----------------------------
```

```
#Clean orignal data frame and new ones to be subdivided into test/train splits
data<-na.omit(data)
rm(HouseExp_Income,TotalDebtPmt_Income,netWealth,cCredHist,
   mCredHist,pubRec,Unemply,SelfEmp,loanAppraised_Low,loanAppraised_High,
   loanAppraised_Medium,DeniedPMi,nonWhite,NotMarried,Education_yrs,denied)
# data frames
DF_D2 <- data.frame(data$denied, I(data$HouseExp_Income+data$Education_yrs),
                    data$cCredHist, data$mCredHist, data$pubRec,
                    log(data$Unemply), data$SelfEmp, data$DeniedPMi,
                     data$nonWhite,
                    data$NotMarried, data$Education_yrs,
                    I(data$DeniedPMi+data$DeniedPMi), data$loanAppraised_High,
                    data$loanAppraised_Low,data$loanAppraised_Medium)

# index
index <- sample(nrow(DF_D2),nrow(DF_D2)*.80)
# splits
train_DF_D2 = data[index,]
test_DF_D2 = data[-index,]

# define cost function
costfunc <- function(denied, pred_prob_train_D2) {
    weight1 <- 1
    weight0 <- 1
    c1 <- (denied==1)&(pred_prob_train_D2<optimal_cutoff)
    c0 <- (denied==0)&(pred_prob_train_D2>=optimal_cutoff)
    cost<- mean(weight1*c1+weight0*c0)
    return(cost)
}

# estimate the model
model_D2 <-glm(denied~.,data = train_DF_D2,family = binomial)

###training
pred_prob_train_D2 <- predict.glm(model_D2,type = "response") #this is
 predprob for cost function
pred_train_D2 <- prediction(pred_prob_train_D2, train_DF_D2$denied)
perf_train_D2 <- performance(pred_train_D2, "tpr", "fpr")
plot(perf_train_D2,colorize=TRUE)
abline(a=0,b=1)
# AUC training set
d2_train_auc <- unlist(slot(performance(pred_train_D2,"auc",),"y.values"))
d2_train_auc

# Testing
pred_prob_test_d2 <- predict.glm(model_D2,newdata = test_DF_D2, type =
 "response")
pred_test_d2 <-prediction(pred_prob_test_d2,test_DF_D2$denied)
perf_test_d2 <- performance(pred_test_d2,"tpr","fpr")
plot(perf_test_d2,colorize=TRUE, add=TRUE)
# AUC Test set
```

```
d2_test_auc <- unlist(slot(performance(pred_test_d2,"auc"),"y.values"))
 #reduction in area under curve
d2_test_auc

# determine optimal cutoff
prob_seq <- seq(0.01, 1, 0.01)

cv_cost2= rep(0,length(prob_seq))
for (i in 1:length(prob_seq)){
    optimal_cutoff = prob_seq[i]
    set.seed(123)
    cv_cost2[i] = cv.glm(data=train_DF_D2, glmfit = model_D2, cost=costfunc,
     K=10)$delta[2]
}

plot(prob_seq,cv_cost2)
optimal_cutoff_cv2 = prob_seq[which(cv_cost2==min(cv_cost2))]
optimal_cutoff_cv2 #.55.. .51

# train Classification
trainClass_D2 <-ifelse(pred_prob_train_D2>optimal_cutoff_cv2,1,0)
trainClass_D2 <- factor(trainClass_D2)
train_deny <- factor(train_DF_D2$denied)
D2train_confm <- confusionMatrix(actuals = train_DF_D2$denied,
                                 predictedScores = pred_prob_train_D2,
                                 threshold = optimal_cutoff_cv2)
D2train_confm

# test classification
testclass_d2 <- ifelse(pred_prob_test_d2>optimal_cutoff_cv2,1,0)
testclass_d2 <-factor(testclass_d2)
test_deny <- factor(test_DF_D2$denied)
D2test_confm <- confusionMatrix(actuals = test_DF_D2$denied,
                                predictedScores = pred_prob_test_d2,
                                threshold = optimal_cutoff_cv2)
D2test_confm

#misclassification error
d2_train_mce <- misClassError(actuals = train_DF_D2$denied,
                              predictedScores = pred_prob_train_D2,
                              threshold = optimal_cutoff_cv2)
d2_train_mce


## Model 3 ----------------------------
data<-na.omit(data)
rm(HouseExp_Income,TotalDebtPmt_Income,netWealth,cCredHist,
   mCredHist,pubRec,Unemply,SelfEmp,loanAppraised_Low,loanAppraised_High,
   loanAppraised_Medium,DeniedPMi,nonWhite,NotMarried,Education_yrs,denied)
# data frames
```

```
DF_D3 <- data.frame(data$denied, data$HouseExp_Income,
 data$TotalDebtPmt_Income,
                     data$netWealth, data$cCredHist, data$mCredHist,
                      data$Unemply,
                     data$SelfEmp, data$loanAppraised_Low ,
                      data$loanAppraised_Medium,
                     data$nonWhite,
                      data$loanAppraised_High,data$DeniedPMi,data$nonWhite,
                     data$NotMarried, data$Education_yrs)
#index
index <- sample(nrow(DF_D3),nrow(DF_D3)*.80)
# splits
train_DF_D3 = data[index,]

# cost function
costfunc <- function(denied, pred_prob_train_D3) {
    weight1 <- 1
    weight0 <- 1
    c1 <- (denied==1)&(pred_prob_train_D3<optimal_cutoff)
    c0 <- (denied==0)&(pred_prob_train_D3>=optimal_cutoff)
    cost<- mean(weight1*c1+weight0*c0)
    return(cost)
}

# Determining which poly values to use
v1 <- numeric(4)
v2 <- numeric(4)
v3 <- numeric(4)
v4 <- numeric(4)
polys <-array(c(v1,v2,v3,v4), dim=c(4,4,4,4))

for (i in 1:4){
    for (j in 1:4){
        for (k in 1:4){
            for (l in 1:4){
                model_d3 <- glm(denied~ HouseExp_Income +
                 poly(TotalDebtPmt_Income,i,raw=TRUE)
                                + netWealth + poly(cCredHist,j,raw=TRUE) +
                                 mCredHist
                                + poly(Unemply,k,raw=TRUE) + SelfEmp +
                                 loanAppraised_Low
                                + loanAppraised_Medium + loanAppraised_High
                                + DeniedPMi  + nonWhite + NotMarried
                                + poly(Education_yrs,l,raw=TRUE),
                                data = train_DF_D3, family = binomial)
                pred_prob_train_D3 <- predict.glm(model_d3,type = "response")
                 #this is predprob for cost function
                pred_train_D3 <- prediction(pred_prob_train_D3,
                 train_DF_D3$denied)
                perf_train_D3 <- performance(pred_train_D3, "tpr", "fpr")
                # AUC training set
```

```
                polys[i,j,k,l] <-
                 unlist(slot(performance(pred_train_D3,"auc",),"y.values"))
            }
        }
    }
}
polySet <-  which(polys == max(polys),arr.ind=TRUE)
polySet
# MAX AUC .834 for the poly values 4 3 4 3
print(polys[polySet])

# Hardcoding a data frame with those poly values

DF_D3a <- data.frame(data$denied, data$HouseExp_Income,
 I(data$TotalDebtPmt_Income**4),
                     data$netWealth, I(data$cCredHist**3), data$mCredHist,
                     I(data$Unemply**4), data$SelfEmp, data$loanAppraised_Low,
                     data$loanAppraised_Medium, data$nonWhite,
                      data$loanAppraised_High,
                     data$DeniedPMi,data$nonWhite, data$NotMarried,
                      I(data$Education_yrs**4))

# Split new frame
train_DF_D3a = data[index,]
test_DF_D3a = data[-index,]

# model estimation
model_D3a <-glm(denied~.,data = train_DF_D3a,family = binomial)

###training
pred_prob_train_D3a <- predict.glm(model_D3a,type = "response") #this is
 predprob for cost function
pred_train_D3a <- prediction(pred_prob_train_D3a, train_DF_D3a$denied)
perf_train_D3a <- performance(pred_train_D3a, "tpr", "fpr")
plot(perf_train_D3a,colorize=TRUE)
abline(a=0,b=1)
# AUC training set
d3a_train_auc <- unlist(slot(performance(pred_train_D3a,"auc",),"y.values")) #
 .82865
d3a_train_auc

# Testing
pred_prob_test_d3a <- predict.glm(model_D3a,newdata = test_DF_D3a, type =
 "response")
pred_test_d3a <-prediction(pred_prob_test_d3a,test_DF_D3a$denied)
perf_test_d3a <- performance(pred_test_d3a,"tpr","fpr")
plot(perf_test_d3a,colorize=TRUE, add=TRUE)
# AUC Test set
d3a_test_auc <- unlist(slot(performance(pred_test_d3a,"auc"),"y.values"))
 #reduction in area under curve # .8434
d3a_test_auc
```

```r
# determine optimal cutoff
prob_seq <- seq(0.01, 1, 0.01)

cv_cost3a= rep(0,length(prob_seq))
for (i in 1:length(prob_seq)){
    optimal_cutoff = prob_seq[i]
    set.seed(123)
    cv_cost3a[i] = cv.glm(data=train_DF_D3a, glmfit = model_D3a,
     cost=costfunc, K=10)$delta[2]
}

## Optimal Cutoff Score
plot(prob_seq,cv_cost3a)
optimal_cutoff_cv3a = prob_seq[which(cv_cost3a==min(cv_cost3a))]
optimal_cutoff_cv3a #.41

# train Classification
trainClass_D3a <-ifelse(pred_prob_train_D3a>optimal_cutoff_cv3a,1,0)
trainClass_D3a <- factor(trainClass_D3a)
train_deny <- factor(train_DF_D3a$denied)
D3atrain_confm <- confusionMatrix(actuals = test_DF_D3a$denied,
                                  predictedScores = pred_prob_test_d3a,
                                  threshold = optimal_cutoff_cv3a)
D3atrain_confm

# test classification
testclass_d3a <- ifelse(pred_prob_test_d3a>optimal_cutoff_cv3a,1,0)
testclass_d3a <-factor(testclass_d3a)
test_deny <- factor(test_DF_D3a$denied)
D3atest_confm <- confusionMatrix(actuals = test_DF_D3a$denied,
                                 predictedScores = pred_prob_test_d3a,
                                 threshold = optimal_cutoff_cv3a)
D3atest_confm

#misclassification error
d3a_train_mce <- misClassError(actuals = train_DF_D3a$denied,
                               predictedScores = pred_prob_train_D3a,
                               threshold = optimal_cutoff_cv3a)
d3a_train_mce


# -------------------- E. Evaluating the superior model -----------------------
#fresh data
data<-na.omit(data)
rm(HouseExp_Income,TotalDebtPmt_Income,netWealth,cCredHist,
   mCredHist,pubRec,Unemply,SelfEmp,loanAppraised_Low,loanAppraised_High,
   loanAppraised_Medium,DeniedPMi,nonWhite,NotMarried,Education_yrs,denied)

#D2 was best model, so replicate with full data
#data from D2
```

```
bestData <- data.frame(data$denied,
 I(data$HouseExp_Income+data$Education_yrs),
                        data$cCredHist, data$mCredHist, data$pubRec,
                         log(data$Unemply),
                        data$SelfEmp, data$DeniedPMi, data$nonWhite,
                         data$NotMarried,
                        data$Education_yrs, I(data$DeniedPMi+data$DeniedPMi),
                        data$loanAppraised_High,data$loanAppraised_Low
                         ,data$loanAppraised_Medium)

optimal_cutoff <- optimal_cutoff_cv2

# estimate the model taken from D2
bestModel <- glm(data.denied~., data = bestData,family = binomial)
pred_prob_e <- predict.glm(bestModel, type = c("response"))

pred_e <- prediction(pred_prob_e,bestData$data.denied)
perf_e <- performance(pred_e,measure = "tpr", x.measure = "fpr")

# Plot ROC curve
plot(perf_e, colorize=TRUE)
abline(a=0,b=1)

# Area under the Curve (AUC) for Full Sample model
bestModel_auc <-
 unlist(slot(performance(prediction(pred_prob_e,bestData$data
 .denied),"auc"),"y.values"))

#### Confusion Matrix
tableofDenial <- table(bestData$data.denied) # to find tpr fpr etc.

## Confusion Matrix from base R for best model
class_prediction_mc <- ifelse(pred_prob_e > optimal_cutoff, "positive_class",
 "negative_class")
e_denied <- table(bestData)
confmat <- table(class_prediction_mc, bestData$data.denied)

tpr <- confmat[2,2]/tableofDenial[[2]]  # "sensitivity"
fpr <- confmat[2,1]/tableofDenial[[1]]
tnr <- confmat[1,1]/tableofDenial[[1]]  # "specificity"
fnr <- confmat[1,2]/tableofDenial[[2]]
accuracy <- (confmat[2,2] + confmat[1,1])/length(bestData$data.denied)
misclass_rate <- 1-accuracy
bestModelInfo <- c(tpr,fpr,tnr,fnr,accuracy,misclass_rate, bestModel_auc)

#### Confusion Matrix for C
tableofDenial <- table(DF_C$denied) # to find tpr fpr etc.

## Confusion Matrix from base R for model C to compare
class_prediction_mc <- ifelse(pred_prob_c > optimal_cutoff, "positive_class",
 "negative_class")
```

```
confmat <- table(class_prediction_mc, DF_C$denied)

tpr <- confmat[2,2]/tableofDenial[[2]]  # "sensitivity"
fpr <- confmat[2,1]/tableofDenial[[1]]
tnr <- confmat[1,1]/tableofDenial[[1]]  # "specificity"
fnr <- confmat[1,2]/tableofDenial[[2]]
accuracy <- (confmat[2,2] + confmat[1,1])/length(DF_C$denied)
misclass_rate <- 1-accuracy
modelCInfo <- c(tpr,fpr,tnr,fnr,accuracy,misclass_rate, model_C_auc)



#data types for summary output
stats <- c("specificity","fpr","sensitivity","fnr","accuracy",
 "misclassification Rate", "auc")
Summary <- data.frame(stats, bestModelInfo, modelCInfo)
stargazer(Summary, type = "text", summary = FALSE)

#cleanup
rm()
```