Speed Dating Project

Aria Wang

setwd("~/Desktop/MSiA 420/speed-dating-project")  
data <- read.csv("Speed Dating Data.csv")  
library(caret)

# check NA  
na\_rate <- rep(1, dim(data)[2])  
# drop the variables that end with \_3   
for (i in 1:156){  
 na\_rate[i] <- sum(is.na(data[,i]))/nrow(data)  
}  
na\_columns <- colnames(data)[na\_rate > 0.5]

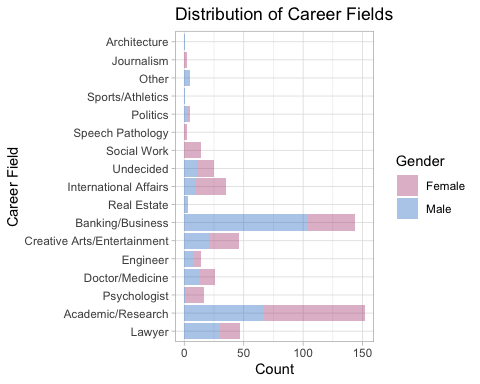
library(dplyr)  
data <- data %>%   
 select(-na\_columns)

data$gender <- as.factor(data$gender)  
data$career\_c <- as.factor(data$career\_c)  
data$samerace <- as.factor(data$samerace)  
data$race <- as.factor(data$race)  
data$dec <- as.factor(data$dec)  
data$date <- as.factor(data$date)

# scale the ratings  
data <- data %>%   
 mutate(pf\_sum\_o = pf\_o\_att + pf\_o\_sin + pf\_o\_int + pf\_o\_fun + pf\_o\_amb + pf\_o\_sha,  
 sum\_o = attr\_o + sinc\_o + intel\_o + fun\_o + amb\_o + shar\_o,  
 sum1\_1 = attr1\_1 + sinc1\_1 + intel1\_1 + fun1\_1 + amb1\_1 + shar1\_1,  
 sum4\_1 = attr4\_1 + sinc4\_1 + intel4\_1 + fun4\_1 + amb4\_1 + shar4\_1,  
 sum2\_1 = attr2\_1 + sinc2\_1 + intel2\_1 + fun2\_1 + amb2\_1 + shar2\_1,  
 sum3\_1 = attr3\_1 + sinc3\_1 + intel3\_1 + fun3\_1 + amb3\_1,  
 sum5\_1 = attr5\_1 + sinc5\_1 + intel5\_1 + fun5\_1 + amb5\_1,  
 sum1\_2 = attr1\_2 + sinc1\_2 + intel1\_2 + fun1\_2 + amb1\_2 + shar1\_2,  
 sum4\_2 = attr4\_2 + sinc4\_2 + intel4\_2 + fun4\_2 + amb4\_2 + shar4\_2,  
 sum2\_2 = attr2\_2 + sinc2\_2 + intel2\_2 + fun2\_2 + amb2\_2 + shar2\_2,  
 sum3\_2 = attr3\_2 + sinc3\_2 + intel3\_2 + fun3\_2 + amb3\_2,  
 sum5\_2 = attr5\_2 + sinc5\_2 + intel5\_2 + fun5\_2 + amb5\_2) %>%   
 mutate\_at(c("pf\_o\_att", "pf\_o\_sin", "pf\_o\_int", "pf\_o\_fun", "pf\_o\_amb", "pf\_o\_sha"),   
 funs(./pf\_sum\_o\*100)) %>%   
 mutate\_at(c("attr\_o", "sinc\_o", "intel\_o", "fun\_o", "amb\_o", "shar\_o"),   
 funs(./sum\_o\*100)) %>%   
 mutate\_at(c("attr1\_1", "sinc1\_1", "intel1\_1", "fun1\_1", "amb1\_1", "shar1\_1"),   
 funs(./sum1\_1\*100)) %>%   
 mutate\_at(c("attr4\_1", "sinc4\_1", "intel4\_1", "fun4\_1", "amb4\_1", "shar4\_1"),   
 funs(./sum4\_1\*100)) %>%   
 mutate\_at(c("attr2\_1", "sinc2\_1", "intel2\_1", "fun2\_1", "amb2\_1", "shar2\_1"),   
 funs(./sum2\_1\*100)) %>%  
 mutate\_at(c("attr3\_1", "sinc3\_1", "fun3\_1", "intel3\_1", "amb3\_1"),   
 funs(./sum3\_1\*100)) %>%  
 mutate\_at(c("attr5\_1", "sinc5\_1", "fun5\_1", "intel5\_1", "amb5\_1"),   
 funs(./sum5\_1\*100)) %>%  
 mutate\_at(c("attr1\_2", "sinc1\_2", "intel1\_2", "fun1\_2", "amb1\_2", "shar1\_2"),   
 funs(./sum1\_2\*100)) %>%  
 mutate\_at(c("attr4\_2", "sinc4\_2", "intel4\_2", "fun4\_2", "amb4\_2", "shar4\_2"),   
 funs(./sum4\_2\*100)) %>%  
 mutate\_at(c("attr2\_2", "sinc2\_2", "intel2\_2", "fun2\_2", "amb2\_2", "shar2\_2"),   
 funs(./sum2\_2\*100)) %>%  
 mutate\_at(c("attr3\_2", "sinc3\_2", "intel3\_2", "fun3\_2", "amb3\_2"),   
 funs(./sum3\_2\*100)) %>%  
 mutate\_at(c("attr5\_2", "sinc5\_2", "intel5\_2", "fun5\_2", "amb5\_2"),   
 funs(./sum5\_2\*100)) %>%  
 select(-c("pf\_sum\_o", "sum\_o", "sum1\_1", "sum4\_1", "sum2\_1", "sum3\_1",   
 "sum5\_1", "sum1\_2", "sum4\_2", "sum2\_2", "sum3\_2", "sum5\_2"))

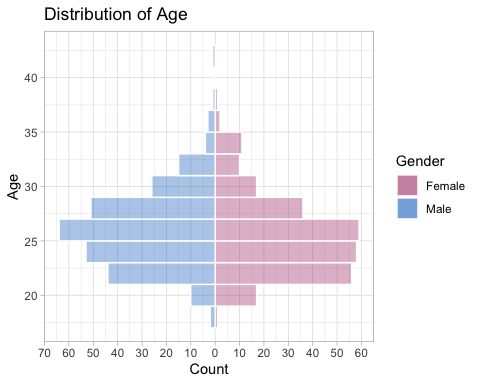
## Warning: funs() is soft deprecated as of dplyr 0.8.0  
## Please use a list of either functions or lambdas:   
##   
## # Simple named list:   
## list(mean = mean, median = median)  
##   
## # Auto named with `tibble::lst()`:   
## tibble::lst(mean, median)  
##   
## # Using lambdas  
## list(~ mean(., trim = .2), ~ median(., na.rm = TRUE))  
## This warning is displayed once per session.

library(ggplot2)  
# career distribution plot  
career\_label <- c("Lawyer", "Academic/Research", "Psychologist",   
 "Doctor/Medicine", "Engineer", "Creative Arts/Entertainment",   
 "Banking/Business", "Real Estate", "International Affairs",   
 "Undecided", "Social Work", "Speech Pathology", "Politics",   
 "Sports/Athletics", "Other", "Journalism", "Architecture")  
  
data %>%   
 filter(!is.na(career\_c)) %>%   
 select(iid, gender, career\_c) %>%   
 unique(by = iid) %>%   
 ggplot() +  
 geom\_bar(aes(career\_c, fill=gender)) +   
 scale\_x\_discrete(label = career\_label) + coord\_flip() +   
 labs(title = "Distribution of Career Fields", x = "Career Field", y = "Count") +   
 scale\_fill\_manual(values=c(rgb(0.7, 0.3, 0.5, 0.4), rgb(0.2, 0.5, 0.8, 0.4)),   
 "Gender", labels = c("Female", "Male")) + theme\_light()

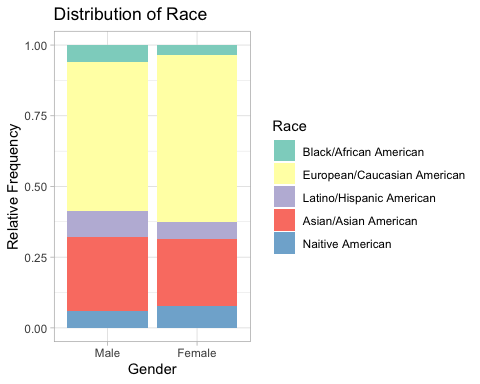


# scale\_fill\_brewer(palette="Set3", "Gender", labels = c("Female", "Male")) + theme\_light()

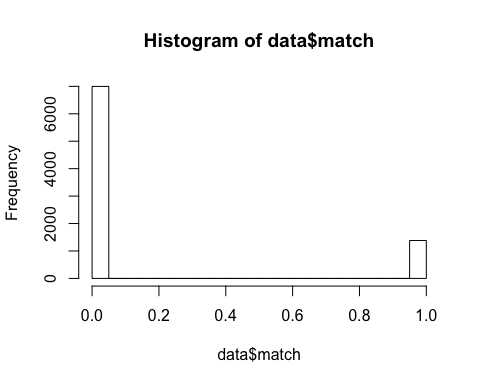
# age distribution plot  
temp\_age <- data %>%   
 filter(!is.na(age)) %>%   
 filter(age < max(age)) %>%   
 select(iid, gender, age) %>%   
 unique(by = iid)   
  
ggplot(data = temp\_age, aes(x = age,fill = gender)) + coord\_flip() +   
 geom\_histogram(data = subset(temp\_age, gender == "0"), binwidth = 2, color = "white") +   
 geom\_histogram(data = subset(temp\_age, gender == "1"),   
 aes(y = ..count.. \* (-1)), binwidth = 2, color = "white") +   
 scale\_y\_continuous(breaks = seq(-70, 70, 10), labels = abs(seq(-70, 70, 10)))+   
 scale\_x\_continuous(breaks = seq(10, 45, 5), labels = seq(10, 45,5)) +   
 labs(title = "Distribution of Age", x = "Age", y = "Count") +   
 scale\_fill\_manual(values=c(rgb(0.7, 0.3, 0.5, 0.4), rgb(0.2, 0.5, 0.8, 0.4)),   
 "Gender", labels = c("Female", "Male")) + theme\_light()



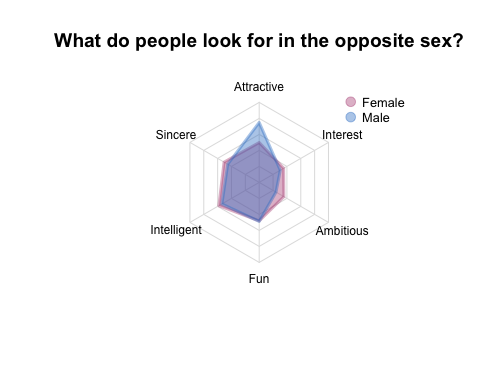
# race distribution plot  
race\_label <- c("Black/African American", "European/Caucasian American",   
 "Latino/Hispanic American", "Asian/Asian American",   
 "Naitive American", "Other")  
  
data %>%   
 filter(!is.na(race)) %>%   
 select(iid, gender, race) %>%   
 unique(by = iid) %>%   
 ggplot() +   
 geom\_bar(aes(x = gender, fill = race), position = "fill") +   
 labs(title = "Distribution of Race", x = "Gender", y = "Relative Frequency") +  
 scale\_fill\_brewer(palette="Set3", name="Race", labels = race\_label) +   
 scale\_x\_discrete(labels=c("0" = "Male", "1" = "Female")) + theme\_light()



hist(data$match)



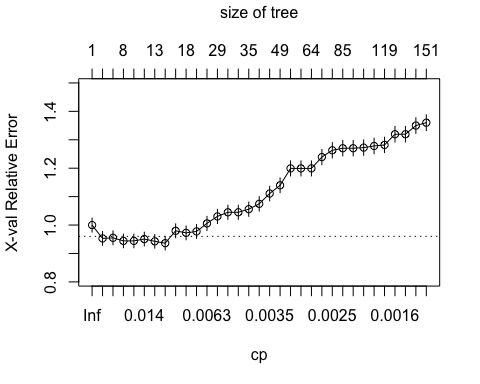
library(fmsb)  
# what do you look for in the opposite sex  
test1 <- data %>%   
 filter(!is.na(attr1\_1 + sinc1\_1 + intel1\_1 + fun1\_1 + amb1\_1 + shar1\_1)) %>%   
 select(iid, gender, attr1\_1:shar1\_1) %>%   
 unique(by = idd) %>%   
 group\_by(gender) %>%   
 summarise(Attractive = mean(attr1\_1), Sincere = mean(sinc1\_1),   
 Intelligent = mean(intel1\_1), Fun = mean(fun1\_1),   
 Ambitious = mean(amb1\_1), Interest = mean(shar1\_1))  
  
test1forplot <- test1 %>%   
 select(-gender)  
   
maxmin <- data.frame(  
 Attractive = c(36, 0),  
 Sincere = c(36, 0),  
 Intelligent = c(36, 0),  
 Fun = c(36, 0),  
 Ambitious = c(36, 0),  
 Interest = c(36, 0))  
  
test11 <- rbind(maxmin, test1forplot)  
  
test11male <- test11[c(1,2,4),]  
test11female <- test11[c(1,2,3),]  
  
radarchart(test11,  
 pty = 32,  
 axistype = 0,  
 pcol = c(rgb(0.7, 0.3, 0.5, 0.4), rgb(0.2, 0.5, 0.8, 0.4)),  
 pfcol = c(rgb(0.7, 0.3, 0.5, 0.4), rgb(0.2, 0.5, 0.8, 0.4)),  
 plty = 1,  
 plwd = 3,  
 cglty = 1,  
 cglcol = "gray88",  
 centerzero = TRUE,  
 seg = 5,  
 vlcex = 0.75,  
 palcex = 0.75,  
 title = "What do people look for in the opposite sex?")  
legend(x = 1, y = 1.2, legend = c("Female", "Male"),  
 bty = "n", pch = 20 , col = c(rgb(0.7, 0.3, 0.5, 0.4), rgb(0.2, 0.5, 0.8, 0.4)),   
 text.col = "black", cex = 0.8, pt.cex = 2)



## Decision Tree

# drop 3 or s  
data <- data[, !grepl(".\*\_[3s]$",colnames(data))]  
  
# drop columns with >50% null values  
t = data.frame(colSums(is.na(data))/nrow(data))  
colnames(t)=c("nullrate")  
t <- t %>% subset(nullrate<0.5)  
data <- data[,rownames(t)]  
  
# other  
data <- data[, colnames(data)!='match\_es']  
data <- data[, colnames(data)!='pid']  
data <- data[, colnames(data)!='dec\_o']  
data <- data[, colnames(data)!='dec']  
data <- data[, colnames(data)!='like\_o']  
data <- data[, colnames(data)!='like']  
data <- data[, colnames(data)!='partner']  
# data <- data[, colnames(data)!='attr\_o']  
# data <- data[, colnames(data)!='attr']  
  
  
data$income <- as.numeric(data$income)  
data$tuition <- as.numeric(data$tuition)  
data <- data[, !(colnames(data) %in% c('zipcode','from','career','field','undergra','mn\_sat','attr5\_2'  
 ,'attr5\_1'))]

# fit inital model by cv  
library(rpart)  
set.seed(420)  
control <- rpart.control(minbucket = 5, cp = 0.001, maxsurrogate = 0, usesurrogate = 0, xval = 10)  
date.tr <- rpart(match ~.,data, method = "class", control = control)   
plotcp(date.tr) #plot of CV r^2 vs. size



date.tr1 <- prune(date.tr, cp=0.011)  
date.tr1$variable.importance

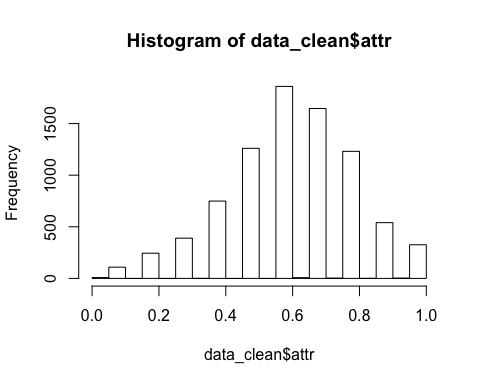
## fun prob\_o prob attr shar fun\_o attr\_o   
## 144.494863 106.295276 29.474435 27.760189 21.157164 16.199978 15.171390   
## intel\_o pf\_o\_int   
## 13.931836 8.071229

# data preprocessing  
selected = colnames(data) %in%  
 c("fun", "prob\_o", "prob", "attr", "shar", "fun\_o", "attr\_o", "intel\_o", "pf\_o\_int", "match")  
dataNN = data[,selected]  
dataNN[,1] = as.factor(dataNN[,1])  
dataNN[, -1] <- sapply(dataNN[,-1], function(x)   
 (x-min(x, na.rm = TRUE))/(max(x, na.rm=TRUE) - min(x, na.rm = TRUE)))  
data\_clean <- dataNN %>%   
 mutate\_at(vars(colnames(dataNN[-1])), ~ifelse(is.na(.), median(., na.rm = TRUE), .))  
write.csv(data\_clean, '~/Desktop/MSiA 420/speed-dating-project/data\_clean.csv')

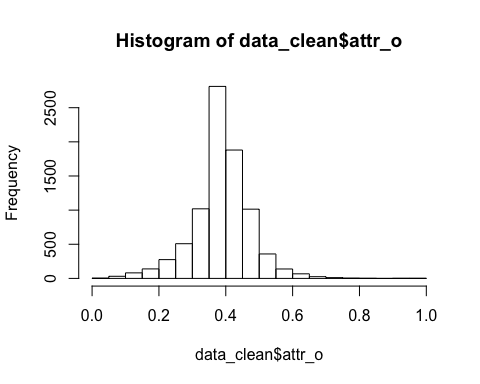
======================= end of data cleaning ========================

## Logistic regression

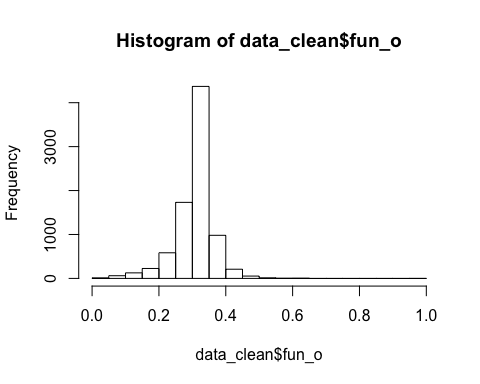
hist(data\_clean$attr)



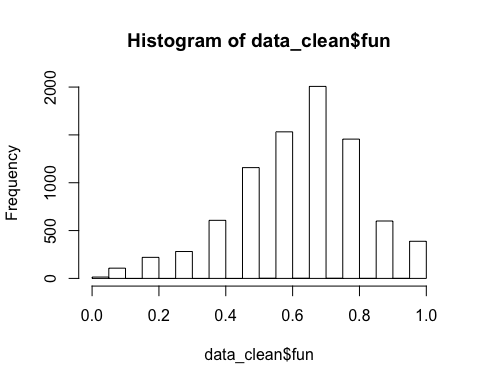
hist(data\_clean$attr\_o)



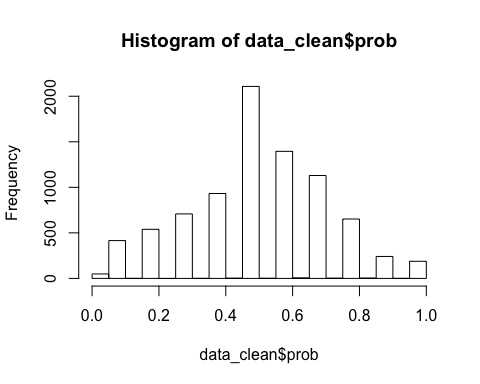
hist(data\_clean$fun\_o)



hist(data\_clean$fun)



hist(data\_clean$prob)



table(data\_clean$match)

##   
## 0 1   
## 6998 1380

logistic1 <- glm(match ~., data=data\_clean, family = 'binomial')  
summary(logistic1)

##   
## Call:  
## glm(formula = match ~ ., family = "binomial", data = data\_clean)  
##   
## Deviance Residuals:   
## Min 1Q Median 3Q Max   
## -2.7641 -0.5755 -0.3610 -0.1728 3.0049   
##   
## Coefficients:  
## Estimate Std. Error z value Pr(>|z|)   
## (Intercept) -9.9489 0.5819 -17.098 < 2e-16 \*\*\*  
## pf\_o\_int 0.8436 0.2522 3.344 0.000824 \*\*\*  
## attr\_o 3.9463 0.4582 8.612 < 2e-16 \*\*\*  
## intel\_o -2.7648 0.7780 -3.554 0.000380 \*\*\*  
## fun\_o 3.9869 0.6961 5.727 1.02e-08 \*\*\*  
## prob\_o 2.9293 0.1815 16.143 < 2e-16 \*\*\*  
## attr 2.5224 0.2317 10.885 < 2e-16 \*\*\*  
## fun 1.7454 0.2636 6.620 3.59e-11 \*\*\*  
## shar 1.0923 0.2269 4.814 1.48e-06 \*\*\*  
## prob 1.6896 0.1888 8.951 < 2e-16 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## (Dispersion parameter for binomial family taken to be 1)  
##   
## Null deviance: 7496.8 on 8377 degrees of freedom  
## Residual deviance: 5898.0 on 8368 degrees of freedom  
## AIC: 5918  
##   
## Number of Fisher Scoring iterations: 6

library(car)

## Loading required package: carData

##   
## Attaching package: 'car'

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

vif(logistic1) # no multicollinearity

## pf\_o\_int attr\_o intel\_o fun\_o prob\_o attr fun shar   
## 1.011868 1.180550 1.249889 1.144091 1.069263 1.349581 1.578344 1.476627   
## prob   
## 1.178891

# cv function  
CVInd <- function(n,K) { # n is sample size; K is number of parts; returns K-length list of   
 # indices for each part   
 m <- floor(n/K) #approximate size of each part   
 r <-n - m\*K   
 I <- sample(n,n) #random reordering of the indices   
 Ind <- list() #will be list of indices for all K parts   
 length(Ind) <- K   
 for (k in 1:K) {   
 if (k <= r) kpart <- ((m+1)\*(k-1)+1):((m+1)\*k)   
 else kpart <- ((m+1)\*r+m\*(k-r-1)+1):((m+1)\*r+m\*(k-r))   
 Ind[[k]] <- I[kpart] #indices for kth part of data   
 }   
 Ind  
}  
  
# misclass rate (return the best rate threshold)  
misclass <- function(rate = seq(0,1,0.01), true\_class, predicted\_value) {  
 misclass\_ratio <- c()  
 model\_class <- c()  
 result\_table <- data.frame()  
 for (i in rate) {  
 predicted\_class <- as.character(ifelse(predicted\_value > i, 1, 0))  
 model\_class <- c(model\_class, predicted\_class)  
 temp <- mean(predicted\_class != as.character(true\_class))  
 misclass\_ratio <- c(misclass\_ratio, temp)  
 }  
 # index <- which(misclass\_ratio == min(misclass\_ratio))  
 return(c(min(misclass\_ratio), rate[which(misclass\_ratio == min(misclass\_ratio))]))  
}

K <- 5 # K-fold CV on each replicate  
n <- nrow(data\_clean)  
Ind <- CVInd(n, K)  
y <- data\_clean$match  
  
yhat <- as.numeric(y)  
yhat\_class <- as.character(y)  
misclassCV <- c()  
  
set.seed(2020)  
for (k in 1:K) {   
 out <- glm(match ~., family = 'binomial', data = data\_clean[-Ind[[k]],])  
 yhat[Ind[[k]]] <- predict(out, newdata = data\_clean[Ind[[k]],-1],   
 type = 'response')  
 temp <- misclass(true\_class = y[Ind[[k]]], predicted\_value = yhat[Ind[[k]]])  
 mc <- temp[1]  
 threshold <- temp[2]  
 yhat\_class[Ind[[k]]] <- as.character(ifelse(yhat[Ind[[k]]] > threshold, 1, 0))  
 misclassCV <- c(misclassCV, mc)  
 }  
misclass\_logistic <- mean(misclassCV)  
misclass\_logistic # 0.1471695

## [1] 0.1481263

confusionMatrix(as.factor(yhat\_class), y)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction 0 1  
## 0 6776 1019  
## 1 222 361  
##   
## Accuracy : 0.8519   
## 95% CI : (0.8441, 0.8594)  
## No Information Rate : 0.8353   
## P-Value [Acc > NIR] : 1.787e-05   
##   
## Kappa : 0.2992   
##   
## Mcnemar's Test P-Value : < 2.2e-16   
##   
## Sensitivity : 0.9683   
## Specificity : 0.2616   
## Pos Pred Value : 0.8693   
## Neg Pred Value : 0.6192   
## Prevalence : 0.8353   
## Detection Rate : 0.8088   
## Detection Prevalence : 0.9304   
## Balanced Accuracy : 0.6149   
##   
## 'Positive' Class : 0   
##

## Ridge

selected\_ridge <- c('match','fun','prob\_o','attr','attr\_o','fun\_o','prob','shar','amb\_o',  
 'shar\_o','age\_o','sinc\_o','intel\_o','pf\_o\_int','tuition',  
 'attr2\_2','attr3\_1','shopping','fun1\_1','sinc4\_1','sinc2\_2','pf\_o\_fun',   
 'income','fun3\_2','sinc5\_1','sinc','theater','pf\_o\_sin','pf\_o\_amb','intel1\_1')  
data\_ridge = data[, selected\_ridge]  
data\_ridge[,1] = as.factor(data\_ridge[,1])  
data\_ridge[, -1] <- sapply(data\_ridge[,-1], function(x)   
 (x-min(x, na.rm = TRUE))/(max(x, na.rm=TRUE) - min(x, na.rm = TRUE)))  
data\_ridge <- data\_ridge %>%   
 mutate\_at(vars(colnames(data\_ridge[-1])), ~ifelse(is.na(.), median(., na.rm = TRUE), .))  
  
library(glmnet)

## Loading required package: Matrix

## Loaded glmnet 3.0-1

x <- model.matrix(match~., data\_ridge)[,-1]  
y <- data\_ridge$match  
cv.ridge <- cv.glmnet(x, y, alpha = 0, family = "binomial")  
ridge1 <- glmnet(x, y, alpha = 0, family = "binomial",  
 lambda = cv.ridge$lambda.min)  
cv.ridge$lambda.min

## [1] 0.01002687

coef(ridge1)

## 30 x 1 sparse Matrix of class "dgCMatrix"  
## s0  
## (Intercept) -6.972393839  
## fun 1.782331216  
## prob\_o 2.465521694  
## attr 2.267925439  
## attr\_o 2.508546501  
## fun\_o 2.297859623  
## prob 1.674172996  
## shar 1.088681665  
## amb\_o -2.726511357  
## shar\_o 1.528721515  
## age\_o -0.664971212  
## sinc\_o -2.440517071  
## intel\_o -2.207143313  
## pf\_o\_int 0.800729816  
## tuition -0.081638440  
## attr2\_2 -0.696859934  
## attr3\_1 -0.194644761  
## shopping -0.280235923  
## fun1\_1 0.605202152  
## sinc4\_1 -0.469174686  
## sinc2\_2 -1.046376039  
## pf\_o\_fun 0.365593839  
## income 0.229045250  
## fun3\_2 0.144550031  
## sinc5\_1 0.341061831  
## sinc 0.001714128  
## theater -0.104959328  
## pf\_o\_sin 0.244974745  
## pf\_o\_amb -0.187747735  
## intel1\_1 0.888335088

K <- 5 # K-fold CV on each replicate  
n <- nrow(data\_clean)  
Ind <- CVInd(n, K)  
  
yhat <- as.numeric(y)  
misclassCV <- c()  
  
set.seed(2020)  
for (k in 1:K) {   
 out <- glmnet(x[-Ind[[k]],], y[-Ind[[k]]], alpha = 0, family = "binomial",  
 lambda = cv.ridge$lambda.min)  
 yhat[Ind[[k]]] <- predict(out, newx = x[Ind[[k]],], type = 'response')  
 temp <- misclass(true\_class = y[Ind[[k]]], predicted\_value = yhat[Ind[[k]]])  
 misclassCV <- c(misclassCV, temp)  
 }  
misclass\_ridge <- mean(misclassCV)  
misclass\_ridge # 0.1466937

## [1] 0.3709679