Autism traits in Adults

library(foreign)  
library(ggplot2)  
library(dplyr)library(caret)

library(corrplot)

library(foreign)  
library(tidyr)  
library(likert)

library(sqldf)

library(gbm)

library(glmnet)

library(class)  
library(Metrics)

library(psych)

library(GPArotation)

## Warning: package 'GPArotation' was built under R version 4.2.3

##   
## Attaching package: 'GPArotation'

## The following objects are masked from 'package:psych':  
##   
## equamax, varimin

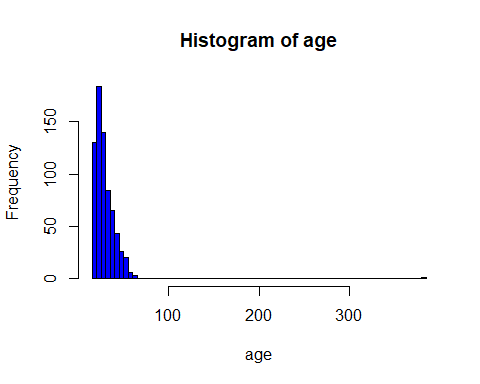
## A1\_Score A2\_Score A3\_Score A4\_Score A5\_Score A6\_Score A7\_Score A8\_Score  
## 0:196 0:385 0:382 0:355 0:353 0:504 0:410 0:247   
## 1:508 1:319 1:322 1:349 1:351 1:200 1:294 1:457   
##   
##   
##   
##   
##   
## A9\_Score A10\_Score age gender ethnicity jundice   
## 0:476 0:300 Min. : 17.0 f:337 White-European :233 no :635   
## 1:228 1:404 1st Qu.: 21.0 m:367 Asian :123 yes: 69   
## Median : 27.0 Middle Eastern : 92   
## Mean : 29.7 Black : 43   
## 3rd Qu.: 35.0 South Asian : 36   
## Max. :383.0 (Other) : 82   
## NA's :2 NA's : 95   
## austim contry\_of\_res used\_app\_before result   
## no :613 United States :113 no :692 Min. : 0.000   
## yes: 91 United Arab Emirates: 82 yes: 12 1st Qu.: 3.000   
## India : 81 Median : 4.000   
## New Zealand : 81 Mean : 4.875   
## United Kingdom : 77 3rd Qu.: 7.000   
## Jordan : 47 Max. :10.000   
## (Other) :223   
## age\_desc relation Class/ASD  
## 18 and more:704 Health care professional: 4 NO :515   
## Others : 5 YES:189   
## Parent : 50   
## Relative : 28   
## Self :522   
## NA's : 95   
##

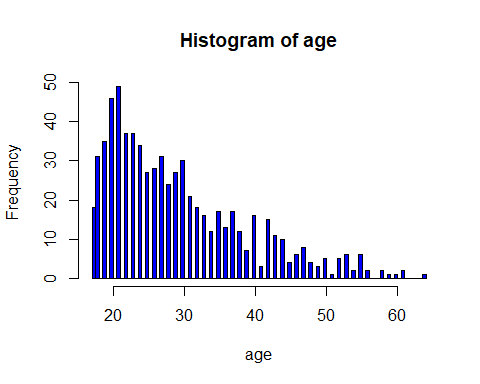
## [1] 2

## A1\_Score A2\_Score A3\_Score A4\_Score A5\_Score   
## 0 0 0 0 0   
## A6\_Score A7\_Score A8\_Score A9\_Score A10\_Score   
## 0 0 0 0 0   
## age gender ethnicity jundice austim   
## 2 0 95 0 0   
## contry\_of\_res used\_app\_before result age\_desc relation   
## 0 0 0 0 95   
## Class/ASD   
## 0

## A1\_Score A2\_Score A3\_Score A4\_Score A5\_Score   
## 0 0 0 0 0   
## A6\_Score A7\_Score A8\_Score A9\_Score A10\_Score   
## 0 0 0 0 0   
## age gender ethnicity jundice austim   
## 2 0 0 0 0   
## contry\_of\_res used\_app\_before result age\_desc relation   
## 0 0 0 0 0   
## Class/ASD   
## 0

## [1] "Asian" "Black" "Hispanic" "Latino"   
## [5] "Middle Eastern " "missing" "Others" "Pasifika"   
## [9] "South Asian" "Turkish" "White-European"

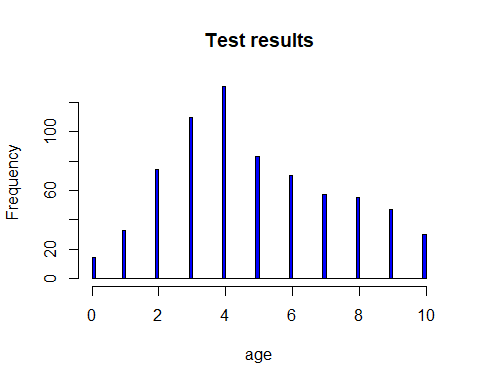




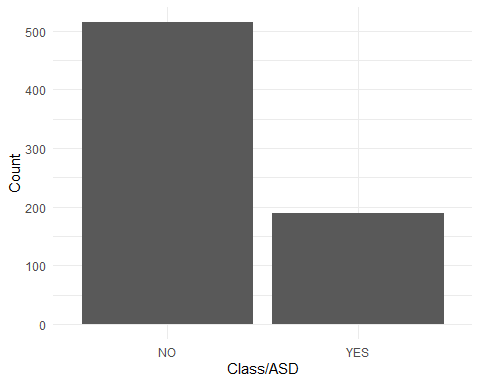
## 'data.frame': 704 obs. of 21 variables:  
## $ A1\_Score : Factor w/ 2 levels "0","1": 2 2 2 2 2 2 1 2 2 2 ...  
## $ A2\_Score : Factor w/ 2 levels "0","1": 2 2 2 2 1 2 2 2 2 2 ...  
## $ A3\_Score : Factor w/ 2 levels "0","1": 2 1 1 1 1 2 1 2 1 2 ...  
## $ A4\_Score : Factor w/ 2 levels "0","1": 2 2 2 2 1 2 1 2 1 2 ...  
## $ A5\_Score : Factor w/ 2 levels "0","1": 1 1 2 1 1 2 1 1 2 1 ...  
## $ A6\_Score : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 2 ...  
## $ A7\_Score : Factor w/ 2 levels "0","1": 2 1 2 2 1 2 1 1 1 2 ...  
## $ A8\_Score : Factor w/ 2 levels "0","1": 2 2 2 2 2 2 2 1 2 2 ...  
## $ A9\_Score : Factor w/ 2 levels "0","1": 1 1 2 1 1 2 1 2 2 2 ...  
## $ A10\_Score : Factor w/ 2 levels "0","1": 1 2 2 2 1 2 1 1 2 1 ...  
## $ age : num 26 24 27 35 40 36 17 64 29 17 ...  
## $ gender : Factor w/ 2 levels "f","m": 1 2 2 1 1 2 1 2 2 2 ...  
## $ ethnicity : Factor w/ 11 levels "Asian","Black",..: 11 4 4 11 6 7 2 11 11 1 ...  
## $ jundice : Factor w/ 2 levels "no","yes": 1 1 2 1 1 2 1 1 1 2 ...  
## $ austim : Factor w/ 2 levels "no","yes": 1 2 2 2 1 1 1 1 1 2 ...  
## $ contry\_of\_res : Factor w/ 67 levels "Afghanistan",..: 65 14 57 65 23 65 65 44 65 10 ...  
## $ used\_app\_before: Factor w/ 2 levels "no","yes": 1 1 1 1 1 1 1 1 1 1 ...  
## $ result : num 6 5 8 6 2 9 2 5 6 8 ...  
## $ age\_desc : Factor w/ 1 level "18 and more": 1 1 1 1 1 1 1 1 1 1 ...  
## $ relation : Factor w/ 6 levels "Health care professional",..: 6 6 4 6 2 6 6 4 6 1 ...  
## $ Class/ASD : Factor w/ 2 levels "NO","YES": 1 1 2 1 1 2 1 1 1 2 ...

# z.test(long$value, mu= 0.5442, sigma.x = 0.4980475)  
#   
# One-sample z-Test  
#   
# data: long$value  
# z = -25.82, p-value < 2.2e-16  
# alternative hypothesis: true mean is not equal to 0.5442  
# 95 percent confidence interval:  
# 0.4601855 0.4720405  
# sample estimates:  
# mean of x   
# 0.466113

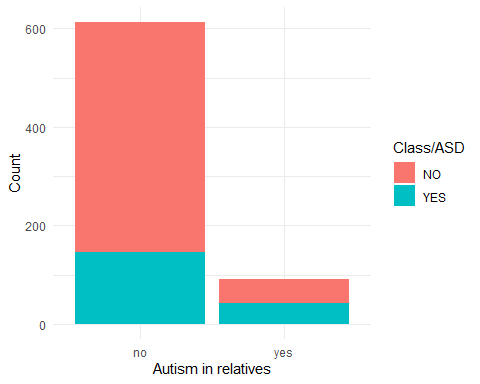
### EDA ###  
  
#distribution of test scores  
hist(df$result, breaks = 100, col = "blue", xlab = "age", ylab = "Frequency", main = "Test results")



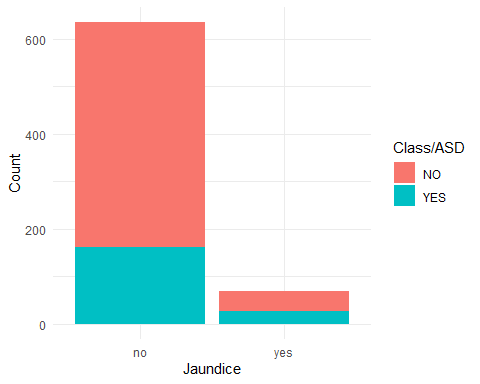
#yes vs. no autism  
ggplot(df, aes(x = `Class/ASD`)) +  
 geom\_bar(stat = "count") +  
 labs(x = "Class/ASD", y = "Count") +  
 theme\_minimal()



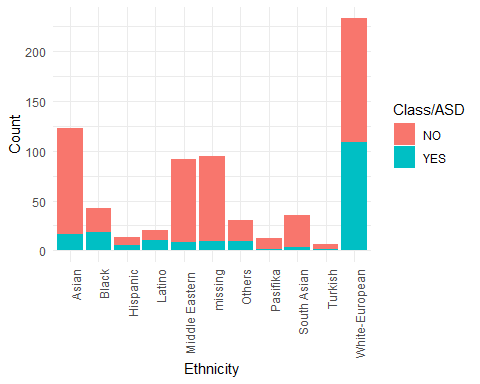
#yes vs. no autism in relatives  
ggplot(df, aes(x = `austim`, fill = `Class/ASD`)) +  
 geom\_bar(stat = "count") +  
 labs(x = "Autism in relatives", y = "Count") +  
 theme\_minimal()



#yes vs. no jaundice  
ggplot(df, aes(x =`jundice`, fill = `Class/ASD`)) +  
 geom\_bar(stat = "count") +  
 labs(x = "Jaundice", y = "Count") +  
 theme\_minimal()



#ethnicities  
ggplot(df, aes(x = ethnicity, fill = `Class/ASD`)) +  
 geom\_bar(stat = "count") +  
 labs(x = "Ethnicity", y = "Count") +  
 theme\_minimal() +  
 theme(axis.text.x = element\_text(angle = 90, hjust = 1))



#one hot encoding with dummy variables  
factor\_cols <- sapply(df, is.factor)  
encoded\_data <- model.matrix(~.-1, data = df[, factor\_cols])  
df2 <- cbind(df[, !factor\_cols], encoded\_data)  
df2 <- df2[,-c(3)]  
  
# #oversampling  
#   
# # Separate Target Classes  
# df\_1 <- df2[df2$"`Class/ASD`YES" == 0, ]  
# df\_2 <- df2[df2$"`Class/ASD`YES" == 1, ]  
#   
# # Upsample minority class  
# df\_2\_upsampled <- df\_2[sample(nrow(df\_2), 615, replace = TRUE), ]  
#   
# # Combine majority class with upsampled minority class  
# df2\_upsampled <- rbind(df\_1, df\_2\_upsampled)  
#   
# # Display new class counts  
# class\_counts <- table(df2\_upsampled$"`Class/ASD`YES")  
# barplot(class\_counts, main = "Class/ASD", xlab = "Class/ASD", ylab = "Count")

### Supervised learning ###  
  
#split into test and train, get rid of multicollinear variables  
grep("austimyes", colnames(df2))

## [1] 25

grep("relationmissing",colnames(df2))

## [1] 93

grep("`Class/ASD`YES", colnames(df2))

## [1] 98

grep("result", colnames(df2))

## [1] 2

#replicate gaussian distribution by logging result and age  
df2$"age" = log(df2$"age"+1)  
  
X <- df2[,-c(2,98)]  
y <- df2$"`Class/ASD`YES"  
   
train\_indices <- createDataPartition(y, p = 0.8, list = FALSE)  
X\_train <- X[train\_indices, ]  
y\_train <- y[train\_indices]  
X\_test <- X[-train\_indices, ]  
y\_test <- y[-train\_indices]

ncol(X) == qr(X)$rank

## [1] FALSE

calculate\_vif <- function(X) {  
 # Calculate VIF for each variable  
 vif\_values <- cor(X)^2  
 high\_vif\_vars <- names(vif\_values)[vif\_values > 0.5]  
 if (length(high\_vif\_vars) > 0) {  
 cat("Variables with high multicollinearity (VIF > 5):\n")  
 cat(high\_vif\_vars, sep = ", ")  
 cat("\n")  
 } else {  
 cat("No variables with high multicollinearity (VIF > 5) found.\n")  
 }  
 return(vif\_values)  
}  
  
vif\_results <- calculate\_vif(X)

## No variables with high multicollinearity (VIF > 5) found.

### Model 1: Linear regression - ALL ###  
  
model <- lm(y\_train ~ ., data = X\_train)  
summary(model)

##   
## Call:  
## lm(formula = y\_train ~ ., data = X\_train)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -0.71870 -0.13072 0.00085 0.14060 0.64852   
##   
## Coefficients: (9 not defined because of singularities)  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) -0.309174 0.240653 -1.285 0.19951   
## age 0.057122 0.040888 1.397 0.16305   
## A1\_Score1 0.111061 0.026911 4.127 4.34e-05 \*\*\*  
## A2\_Score1 0.112731 0.024150 4.668 3.96e-06 \*\*\*  
## A3\_Score1 0.109292 0.025600 4.269 2.37e-05 \*\*\*  
## A4\_Score1 0.124429 0.025467 4.886 1.41e-06 \*\*\*  
## A5\_Score1 0.113009 0.026371 4.285 2.21e-05 \*\*\*  
## A6\_Score1 0.214286 0.029445 7.278 1.41e-12 \*\*\*  
## A7\_Score1 0.125661 0.023783 5.284 1.93e-07 \*\*\*  
## A8\_Score1 0.090386 0.023003 3.929 9.78e-05 \*\*\*  
## A9\_Score1 0.250724 0.028667 8.746 < 2e-16 \*\*\*  
## A10\_Score1 0.073793 0.023932 3.084 0.00216 \*\*   
## genderm -0.036261 0.022506 -1.611 0.10781   
## ethnicityBlack 0.105957 0.066074 1.604 0.10946   
## ethnicityHispanic 0.260456 0.105609 2.466 0.01401 \*   
## ethnicityLatino 0.317111 0.113337 2.798 0.00535 \*\*   
## `ethnicityMiddle Eastern ` 0.081439 0.058347 1.396 0.16343   
## ethnicitymissing -0.205984 0.181493 -1.135 0.25697   
## ethnicityOthers 0.056788 0.059993 0.947 0.34434   
## ethnicityPasifika -0.019663 0.095874 -0.205 0.83759   
## `ethnicitySouth Asian` 0.041652 0.053967 0.772 0.44062   
## ethnicityTurkish 0.164729 0.143066 1.151 0.25014   
## `ethnicityWhite-European` 0.137157 0.049999 2.743 0.00631 \*\*   
## jundiceyes 0.034378 0.038654 0.889 0.37425   
## austimyes 0.005825 0.033441 0.174 0.86179   
## contry\_of\_resAmericanSamoa 0.104131 0.190572 0.546 0.58504   
## contry\_of\_resAngola NA NA NA NA   
## contry\_of\_resArgentina 0.390852 0.257200 1.520 0.12927   
## contry\_of\_resArmenia -0.184562 0.209849 -0.879 0.37958   
## contry\_of\_resAruba 0.371822 0.258283 1.440 0.15064   
## contry\_of\_resAustralia -0.027556 0.101005 -0.273 0.78511   
## contry\_of\_resAustria 0.141807 0.163094 0.869 0.38502   
## contry\_of\_resAzerbaijan NA NA NA NA   
## contry\_of\_resBahamas -0.286430 0.210688 -1.359 0.17463   
## contry\_of\_resBangladesh 0.078387 0.166049 0.472 0.63709   
## contry\_of\_resBelgium 0.066827 0.165381 0.404 0.68634   
## contry\_of\_resBolivia -0.264006 0.282826 -0.933 0.35106   
## contry\_of\_resBrazil -0.046433 0.147758 -0.314 0.75347   
## contry\_of\_resBurundi NA NA NA NA   
## contry\_of\_resCanada 0.123595 0.109398 1.130 0.25914   
## contry\_of\_resChile -0.200790 0.276375 -0.727 0.46788   
## contry\_of\_resChina NA NA NA NA   
## `contry\_of\_resCosta Rica` -0.410568 0.278600 -1.474 0.14123   
## contry\_of\_resCyprus NA NA NA NA   
## `contry\_of\_resCzech Republic` NA NA NA NA   
## contry\_of\_resEcuador -0.513336 0.276018 -1.860 0.06353 .   
## contry\_of\_resEgypt 0.079191 0.188018 0.421 0.67381   
## contry\_of\_resEthiopia -0.074637 0.215153 -0.347 0.72882   
## contry\_of\_resFinland 0.182794 0.262394 0.697 0.48637   
## contry\_of\_resFrance 0.120822 0.116184 1.040 0.29891   
## contry\_of\_resGermany 0.141159 0.148983 0.947 0.34387   
## `contry\_of\_resHong Kong` -0.586473 0.256932 -2.283 0.02289 \*   
## contry\_of\_resIceland -0.516454 0.259227 -1.992 0.04691 \*   
## contry\_of\_resIndia 0.061847 0.094150 0.657 0.51156   
## contry\_of\_resIndonesia 0.063362 0.259575 0.244 0.80726   
## contry\_of\_resIran -0.150450 0.129013 -1.166 0.24413   
## contry\_of\_resIraq 0.230084 0.256013 0.899 0.36926   
## contry\_of\_resIreland 0.114637 0.150121 0.764 0.44547   
## contry\_of\_resItaly 0.136156 0.140770 0.967 0.33392   
## contry\_of\_resJapan 0.176900 0.256100 0.691 0.49006   
## contry\_of\_resJordan -0.033609 0.092937 -0.362 0.71779   
## contry\_of\_resKazakhstan -0.096296 0.198507 -0.485 0.62783   
## contry\_of\_resLebanon NA NA NA NA   
## contry\_of\_resMalaysia 0.408572 0.165569 2.468 0.01395 \*   
## contry\_of\_resMexico -0.068716 0.157482 -0.436 0.66279   
## contry\_of\_resNepal 0.496035 0.260858 1.902 0.05783 .   
## contry\_of\_resNetherlands -0.064061 0.120562 -0.531 0.59542   
## `contry\_of\_resNew Zealand` 0.074417 0.088524 0.841 0.40097   
## contry\_of\_resNicaragua NA NA NA NA   
## contry\_of\_resNiger -0.040602 0.261720 -0.155 0.87678   
## contry\_of\_resOman 0.247396 0.272265 0.909 0.36399   
## contry\_of\_resPakistan 0.161885 0.191645 0.845 0.39869   
## contry\_of\_resPhilippines -0.011564 0.150582 -0.077 0.93882   
## contry\_of\_resPortugal 0.118408 0.258842 0.457 0.64755   
## contry\_of\_resRomania -0.119089 0.166435 -0.716 0.47463   
## contry\_of\_resRussia 0.259693 0.121882 2.131 0.03363 \*   
## `contry\_of\_resSaudi Arabia` -0.101457 0.191812 -0.529 0.59709   
## contry\_of\_resSerbia -0.220821 0.258706 -0.854 0.39378   
## `contry\_of\_resSierra Leone` 0.571866 0.261239 2.189 0.02908 \*   
## `contry\_of\_resSouth Africa` -0.029146 0.193749 -0.150 0.88049   
## contry\_of\_resSpain 0.198143 0.169952 1.166 0.24425   
## `contry\_of\_resSri Lanka` -0.023914 0.126070 -0.190 0.84964   
## contry\_of\_resSweden 0.354334 0.192923 1.837 0.06688 .   
## contry\_of\_resTonga 0.135983 0.270535 0.503 0.61545   
## contry\_of\_resTurkey 0.058571 0.288259 0.203 0.83908   
## contry\_of\_resUkraine 0.039362 0.256874 0.153 0.87828   
## `contry\_of\_resUnited Arab Emirates` 0.015886 0.085269 0.186 0.85228   
## `contry\_of\_resUnited Kingdom` -0.037894 0.090984 -0.416 0.67724   
## `contry\_of\_resUnited States` 0.031483 0.089166 0.353 0.72418   
## contry\_of\_resUruguay -0.083457 0.258866 -0.322 0.74729   
## `contry\_of\_resViet Nam` 0.080780 0.149646 0.540 0.58958   
## used\_app\_beforeyes 0.075909 0.086951 0.873 0.38310   
## relationmissing NA NA NA NA   
## relationOthers -0.453739 0.217633 -2.085 0.03761 \*   
## relationParent -0.328559 0.181729 -1.808 0.07124 .   
## relationRelative -0.383498 0.182962 -2.096 0.03661 \*   
## relationSelf -0.309591 0.176666 -1.752 0.08035 .   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 0.239 on 476 degrees of freedom  
## Multiple R-squared: 0.7581, Adjusted R-squared: 0.7139   
## F-statistic: 17.15 on 87 and 476 DF, p-value: < 2.2e-16

#multicollinearity -> exclude country of residence  
df2\_2<-df2[, -c(26:97)]  
  
grep("Class/ASD", colnames(df2\_2))

## [1] 26

grep("result", colnames(df2\_2))

## [1] 2

X2 <- df2\_2[,-c(2,26)]  
y2 <- df2\_2$"`Class/ASD`YES"  
  
train\_indices2 <- createDataPartition(y2, p = 0.8, list = FALSE)  
X\_train2 <- X2[train\_indices, , drop = FALSE]  
y\_train2 <- y2[train\_indices]  
X\_test2 <- X2[-train\_indices, ]  
y\_test2 <- y2[-train\_indices]  
  
model2 <- lm(y\_train2 ~ ., data = X\_train2)  
summary(model2)

##   
## Call:  
## lm(formula = y\_train2 ~ ., data = X\_train2)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -0.6874 -0.1474 0.0123 0.1554 0.6462   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) -0.5011678 0.1296148 -3.867 0.000124 \*\*\*  
## age 0.0405916 0.0366039 1.109 0.267950   
## A1\_Score1 0.1059494 0.0249620 4.244 2.58e-05 \*\*\*  
## A2\_Score1 0.0980469 0.0227209 4.315 1.90e-05 \*\*\*  
## A3\_Score1 0.1250493 0.0242021 5.167 3.36e-07 \*\*\*  
## A4\_Score1 0.1115384 0.0244798 4.556 6.44e-06 \*\*\*  
## A5\_Score1 0.1261612 0.0245043 5.149 3.69e-07 \*\*\*  
## A6\_Score1 0.1958953 0.0278330 7.038 5.95e-12 \*\*\*  
## A7\_Score1 0.1227147 0.0229769 5.341 1.37e-07 \*\*\*  
## A8\_Score1 0.1119644 0.0222313 5.036 6.48e-07 \*\*\*  
## A9\_Score1 0.2653288 0.0273487 9.702 < 2e-16 \*\*\*  
## A10\_Score1 0.0674837 0.0231518 2.915 0.003707 \*\*   
## genderm -0.0418295 0.0212640 -1.967 0.049678 \*   
## ethnicityBlack 0.1027053 0.0478973 2.144 0.032457 \*   
## ethnicityHispanic 0.2272675 0.0967102 2.350 0.019134 \*   
## ethnicityLatino 0.1568654 0.0652157 2.405 0.016494 \*   
## `ethnicityMiddle Eastern ` 0.0157965 0.0387685 0.407 0.683835   
## ethnicitymissing 0.0436020 0.0380417 1.146 0.252236   
## ethnicityOthers 0.0394409 0.0552354 0.714 0.475504   
## ethnicityPasifika -0.0029094 0.0820443 -0.035 0.971725   
## `ethnicitySouth Asian` 0.0418677 0.0531266 0.788 0.431000   
## ethnicityTurkish 0.0552265 0.1133020 0.487 0.626154   
## `ethnicityWhite-European` 0.0900872 0.0335044 2.689 0.007393 \*\*   
## jundiceyes 0.0317283 0.0364688 0.870 0.384682   
## austimyes 0.0000997 0.0316605 0.003 0.997488   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 0.2437 on 539 degrees of freedom  
## Multiple R-squared: 0.7153, Adjusted R-squared: 0.7026   
## F-statistic: 56.43 on 24 and 539 DF, p-value: < 2.2e-16

#sapply(X\_train, class)

# Lin reg w/o questions  
  
x3 <- df2\_2[,-c(2:12, 26)]  
y3 <- df2\_2$"`Class/ASD`YES"  
  
train\_indices3 <- createDataPartition(y3, p = 0.8, list = FALSE)  
X\_train3 <- x3[train\_indices, , drop = FALSE]  
y\_train3 <- y3[train\_indices]  
X\_test3 <- x3[-train\_indices, ]  
y\_test3 <- y3[-train\_indices]  
  
model3 <- lm(y\_train3 ~ ., data = X\_train3)  
summary(model3)

##   
## Call:  
## lm(formula = y\_train3 ~ ., data = X\_train3)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -0.72145 -0.27559 -0.07846 0.39582 0.96283   
##   
## Coefficients:  
 Estimate Std. Error t value Pr(>|t|)

(Intercept) 0.19776 0.20762 0.953 0.3412

age -0.01693 0.06069 -0.279 0.7804

genderm -0.03597 0.03460 -1.040 0.2990

ethnicityBlack 0.34504 0.08194 4.211 2.97e-05 \*\*\*

ethnicityHispanic 0.26172 0.11941 2.192 0.0288 \*

ethnicityLatino 0.34956 0.10812 3.233 0.0013 \*\*

`ethnicityMiddle Eastern ` -0.07588 0.06168 -1.230 0.2192

ethnicitymissing -0.03841 0.06125 -0.627 0.5308

ethnicityOthers 0.15780 0.08669 1.820 0.0693 .

ethnicityPasifika -0.03119 0.14054 -0.222 0.8244

`ethnicitySouth Asian` -0.06986 0.08375 -0.834 0.4046

ethnicityTurkish 0.05725 0.18485 0.310 0.7569

`ethnicityWhite-European` 0.28381 0.05291 5.364 1.20e-07 \*\*\*

jundiceyes 0.06257 0.05833 1.073 0.2839

austimyes 0.11221 0.05410 2.074 0.0385 \*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 0.4055 on 549 degrees of freedom  
## Multiple R-squared: 0.1967, Adjusted R-squared: 0.1762   
## F-statistic: 9.603 on 14 and 549 DF, p-value: < 2.2e-16

model4 <- glm(y\_train3 ~ ., data = X\_train3, family =binomial)

summary(model4)

Call:

glm(formula = y\_train3 ~ ., family = binomial, data = X\_train3)

Deviance Residuals:

Min 1Q Median 3Q Max

-1.4467 -0.7934 -0.4509 0.9224 2.4251

Coefficients:

Estimate Std. Error z value Pr(>|z|)

(Intercept) -1.5314 1.2590 -1.216 0.223851

age -0.0969 0.3658 -0.265 0.791074

genderm -0.2074 0.2168 -0.957 0.338789

ethnicityBlack 1.8109 0.4713 3.842 0.000122 \*\*\*

ethnicityHispanic 1.4926 0.6554 2.277 0.022765 \*

ethnicityLatino 1.7850 0.5937 3.007 0.002641 \*\*

`ethnicityMiddle Eastern ` -0.8436 0.5586 -1.510 0.131031

ethnicitymissing -0.3472 0.4866 -0.713 0.475575

ethnicityOthers 1.0142 0.5221 1.943 0.052041 .

ethnicityPasifika -0.2506 1.1110 -0.226 0.821525

`ethnicitySouth Asian` -0.7925 0.7974 -0.994 0.320333

ethnicityTurkish 0.4620 1.1658 0.396 0.691905

`ethnicityWhite-European` 1.5445 0.3525 4.382 1.18e-05 \*\*\*

jundiceyes 0.3577 0.3359 1.065 0.286950

austimyes 0.5744 0.3022 1.901 0.057337 .

---

Signif. codes: 0 ‘\*\*\*’ 0.001 ‘\*\*’ 0.01 ‘\*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

Null deviance: 638.67 on 563 degrees of freedom

Residual deviance: 539.51 on 549 degrees of freedom

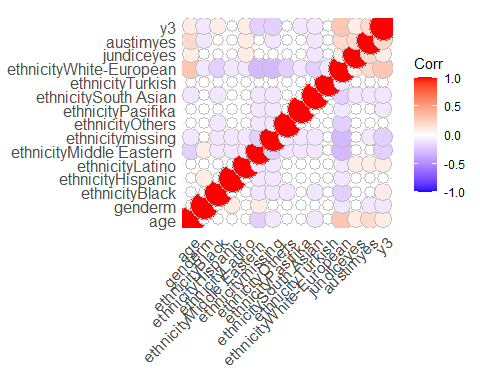
AIC: 569.51

Number of Fisher Scoring iterations: 5

library(ggcorrplot)

## Warning: package 'ggcorrplot' was built under R version 4.2.3

data3 <- cbind(x3, y3)  
corr <- round(cor(data3), 1)  
ggcorrplot(corr, method = "circle")



### Ridge regression ###  
X\_train3 <- as.matrix(X\_train3)  
X\_test3 <- as.matrix(X\_test3)  
  
cv <- cv.glmnet(X\_train3, y\_train3, alpha = 0)  
cv$lambda.min

## [1] 0.0487349

#[1] 0.03302436  
  
ridge <- glmnet(X\_train3, y\_train3, alpha = 0, lambda = cv$lambda.min)  
coef(ridge)

## 15 x 1 sparse Matrix of class "dgCMatrix"  
## s0  
## (Intercept) 0.06366007  
## age 0.03421770  
## genderm -0.02508264  
## ethnicityBlack 0.18112134  
## ethnicityHispanic 0.35173131  
## ethnicityLatino 0.28496792  
## ethnicityMiddle Eastern -0.08305510  
## ethnicitymissing -0.09747335  
## ethnicityOthers 0.09280028  
## ethnicityPasifika -0.15868712  
## ethnicitySouth Asian -0.09142999  
## ethnicityTurkish 0.01031962  
## ethnicityWhite-European 0.24581975  
## jundiceyes 0.09280062  
## austimyes 0.14509777

predictions <- ridge %>% predict(X\_test3) %>% as.vector()  
# Model performance metrics  
data.frame(  
 RMSE = RMSE(predictions, y\_test3),  
 Rsquare = R2(predictions, y\_test3)  
)

## RMSE Rsquare  
## 1 0.4231675 0.05671038

#RMSE Rsquare  
# 0.2314875 0.7887824  
  
  
### Lasso regression ###  
  
cv2 <- cv.glmnet(X\_train3, y\_train3, alpha = 1)  
cv2$lambda.min

## [1] 0.004651983

lasso <- glmnet(X\_train3, y\_train3, alpha = 1, lambda = cv2$lambda.min)  
coef(lasso)

## 15 x 1 sparse Matrix of class "dgCMatrix"  
## s0  
## (Intercept) 0.08573314  
## age 0.01786231  
## genderm -0.01743918  
## ethnicityBlack 0.21284574  
## ethnicityHispanic 0.37579203  
## ethnicityLatino 0.31959360  
## ethnicityMiddle Eastern -0.05168673  
## ethnicitymissing -0.06674782  
## ethnicityOthers 0.10883232  
## ethnicityPasifika -0.11430295  
## ethnicitySouth Asian -0.05327698  
## ethnicityTurkish .   
## ethnicityWhite-European 0.28935093  
## jundiceyes 0.08574389  
## austimyes 0.14412889

predictions2 <- lasso %>% predict(X\_test3) %>% as.vector()  
# Model performance metrics  
data.frame(  
 RMSE = RMSE(predictions2, y\_test3),  
 Rsquare = R2(predictions2, y\_test3)  
)

## RMSE Rsquare  
## 1 0.4220618 0.06156525

#RMSE Rsquare  
#1 0.2305083 0.789022  
  
### Elastic net###  
y\_train\_f <- factor(y\_train3, levels = c(0, 1))  
y\_test\_f <- factor(y\_test3, levels = c(0, 1))  
  
el\_model <- train(X\_train3, y\_train\_f, method = "glmnet",  
 trControl = trainControl("cv", number = 10),  
 tuneLength = 10)  
  
print(el\_model)

## glmnet   
##   
## 564 samples  
## 14 predictor  
## 2 classes: '0', '1'   
##   
## No pre-processing  
## Resampling: Cross-Validated (10 fold)   
## Summary of sample sizes: 508, 508, 508, 507, 507, 508, ...   
## Resampling results across tuning parameters:  
##   
## alpha lambda Accuracy Kappa   
## 0.1 6.718216e-05 0.7375940 0.22483243  
## 0.1 1.551995e-04 0.7375940 0.22483243  
## 0.1 3.585310e-04 0.7375940 0.22483243  
## 0.1 8.282530e-04 0.7375940 0.22483243  
## 0.1 1.913372e-03 0.7375940 0.22483243  
## 0.1 4.420137e-03 0.7358396 0.21793817  
## 0.1 1.021109e-02 0.7358709 0.21493569  
## 0.1 2.358894e-02 0.7322995 0.19671314  
## 0.1 5.449352e-02 0.7305138 0.17684709  
## 0.1 1.258871e-01 0.7340852 0.10638770  
## 0.2 6.718216e-05 0.7375940 0.22483243  
## 0.2 1.551995e-04 0.7375940 0.22483243  
## 0.2 3.585310e-04 0.7375940 0.22483243  
## 0.2 8.282530e-04 0.7375940 0.22483243  
## 0.2 1.913372e-03 0.7375940 0.22483243  
## 0.2 4.420137e-03 0.7358396 0.21793817  
## 0.2 1.021109e-02 0.7340852 0.20796089  
## 0.2 2.358894e-02 0.7287594 0.18224415  
## 0.2 5.449352e-02 0.7393797 0.18245108  
## 0.2 1.258871e-01 0.7322995 0.04764244  
## 0.3 6.718216e-05 0.7375940 0.22483243  
## 0.3 1.551995e-04 0.7375940 0.22483243  
## 0.3 3.585310e-04 0.7375940 0.22483243  
## 0.3 8.282530e-04 0.7375940 0.22483243  
## 0.3 1.913372e-03 0.7375940 0.22483243  
## 0.3 4.420137e-03 0.7358396 0.21793817  
## 0.3 1.021109e-02 0.7305451 0.19338910  
## 0.3 2.358894e-02 0.7305451 0.18292131  
## 0.3 5.449352e-02 0.7358709 0.16265908  
## 0.3 1.258871e-01 0.7252193 0.00000000  
## 0.4 6.718216e-05 0.7375940 0.22483243  
## 0.4 1.551995e-04 0.7375940 0.22483243  
## 0.4 3.585310e-04 0.7375940 0.22483243  
## 0.4 8.282530e-04 0.7375940 0.22483243  
## 0.4 1.913372e-03 0.7358396 0.21793817  
## 0.4 4.420137e-03 0.7358396 0.21793817  
## 0.4 1.021109e-02 0.7305138 0.18937420  
## 0.4 2.358894e-02 0.7287594 0.17665488  
## 0.4 5.449352e-02 0.7322995 0.14170380  
## 0.4 1.258871e-01 0.7252193 0.00000000  
## 0.5 6.718216e-05 0.7375940 0.22483243  
## 0.5 1.551995e-04 0.7375940 0.22483243  
## 0.5 3.585310e-04 0.7375940 0.22483243  
## 0.5 8.282530e-04 0.7375940 0.22483243  
## 0.5 1.913372e-03 0.7358396 0.21793817  
## 0.5 4.420137e-03 0.7376253 0.22175219  
## 0.5 1.021109e-02 0.7305138 0.18937420  
## 0.5 2.358894e-02 0.7269737 0.17008504  
## 0.5 5.449352e-02 0.7322995 0.14170380  
## 0.5 1.258871e-01 0.7252193 0.00000000  
## 0.6 6.718216e-05 0.7375940 0.22483243  
## 0.6 1.551995e-04 0.7375940 0.22483243  
## 0.6 3.585310e-04 0.7375940 0.22483243  
## 0.6 8.282530e-04 0.7375940 0.22483243  
## 0.6 1.913372e-03 0.7358396 0.21793817  
## 0.6 4.420137e-03 0.7376253 0.22175219  
## 0.6 1.021109e-02 0.7305138 0.18937420  
## 0.6 2.358894e-02 0.7322682 0.17342529  
## 0.6 5.449352e-02 0.7304825 0.12150903  
## 0.6 1.258871e-01 0.7252193 0.00000000  
## 0.7 6.718216e-05 0.7375940 0.22483243  
## 0.7 1.551995e-04 0.7375940 0.22483243  
## 0.7 3.585310e-04 0.7375940 0.22483243  
## 0.7 8.282530e-04 0.7375940 0.22483243  
## 0.7 1.913372e-03 0.7358396 0.21793817  
## 0.7 4.420137e-03 0.7376253 0.22175219  
## 0.7 1.021109e-02 0.7305138 0.18937420  
## 0.7 2.358894e-02 0.7322368 0.16907744  
## 0.7 5.449352e-02 0.7269424 0.07486463  
## 0.7 1.258871e-01 0.7252193 0.00000000  
## 0.8 6.718216e-05 0.7375940 0.22483243  
## 0.8 1.551995e-04 0.7375940 0.22483243  
## 0.8 3.585310e-04 0.7375940 0.22483243  
## 0.8 8.282530e-04 0.7375940 0.22483243  
## 0.8 1.913372e-03 0.7358396 0.21793817  
## 0.8 4.420137e-03 0.7358709 0.21453334  
## 0.8 1.021109e-02 0.7287594 0.18224415  
## 0.8 2.358894e-02 0.7340539 0.16424795  
## 0.8 5.449352e-02 0.7234023 0.02838931  
## 0.8 1.258871e-01 0.7252193 0.00000000  
## 0.9 6.718216e-05 0.7375940 0.22483243  
## 0.9 1.551995e-04 0.7375940 0.22483243  
## 0.9 3.585310e-04 0.7375940 0.22483243  
## 0.9 8.282530e-04 0.7358396 0.21793817  
## 0.9 1.913372e-03 0.7358396 0.21793817  
## 0.9 4.420137e-03 0.7323308 0.20036390  
## 0.9 1.021109e-02 0.7305451 0.18511668  
## 0.9 2.358894e-02 0.7322995 0.15647557  
## 0.9 5.449352e-02 0.7252193 0.00000000  
## 0.9 1.258871e-01 0.7252193 0.00000000  
## 1.0 6.718216e-05 0.7375940 0.22483243  
## 1.0 1.551995e-04 0.7375940 0.22483243  
## 1.0 3.585310e-04 0.7375940 0.22483243  
## 1.0 8.282530e-04 0.7358396 0.21793817  
## 1.0 1.913372e-03 0.7358396 0.21793817  
## 1.0 4.420137e-03 0.7305451 0.19338910  
## 1.0 1.021109e-02 0.7305451 0.18511668  
## 1.0 2.358894e-02 0.7358709 0.16265908  
## 1.0 5.449352e-02 0.7252193 0.00000000  
## 1.0 1.258871e-01 0.7252193 0.00000000  
##   
## Accuracy was used to select the optimal model using the largest value.  
## The final values used for the model were alpha = 0.2 and lambda = 0.05449352.

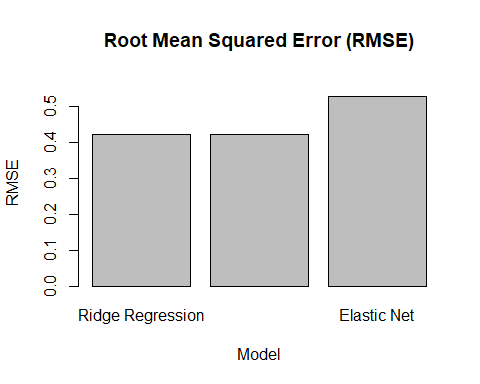
# Best tuning parameter  
el\_model$bestTune

## alpha lambda  
## 19 0.2 0.05449352

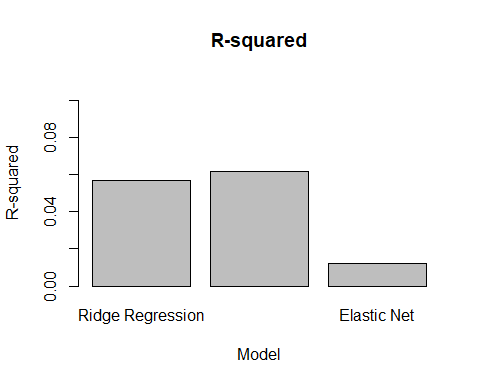
# alpha lambda  
#0.9 0.004389713  
  
coef\_el\_model <- coef(el\_model$finalModel, s = el\_model$bestTune$lambda)  
  
# Print the coefficients  
# Make predictions on the test set  
predictions3 <- predict(el\_model, newdata = X\_test3)  
  
predictions3 <- as.numeric(as.character(predictions3))  
y\_test\_f <- as.numeric(as.character(y\_test\_f))  
  
# Model performance metrics  
data.frame(  
 RMSE = RMSE(predictions3, y\_test\_f),  
 Rsquare = R2(predictions3, y\_test\_f)  
)

## RMSE Rsquare  
## 1 0.5277987 0.0118798

### Graphs ###  
# Calculate RMSE  
  
  
# Create a data frame to store the performance metrics  
model\_performance <- data.frame(  
 Model = c("Ridge Regression", "Lasso Regression", "Elastic Net"),  
 RMSE = c(RMSE(predictions, y\_test\_f), RMSE(predictions2, y\_test\_f), RMSE(predictions3, y\_test\_f)),  
 Rsquared = c(R2(predictions, y\_test\_f), R2(predictions2, y\_test\_f), R2(predictions3, y\_test\_f))  
)  
  
# Bar plot for RMSE  
barplot(model\_performance$RMSE, names.arg = model\_performance$Model,  
 ylim = c(0, max(model\_performance$RMSE) + 0.05),  
 xlab = "Model", ylab = "RMSE", main = "Root Mean Squared Error (RMSE)")



# Bar plot for R-squared  
barplot(model\_performance$Rsquared, names.arg = model\_performance$Model,  
 ylim = c(0, max(model\_performance$Rsquared) + 0.05),  
 xlab = "Model", ylab = "R-squared", main = "R-squared")



### K-NN ###  
# Set the number of neighbors (k)  
k <- 31  
knn\_model <- knn(train = X\_train3, test = X\_test3, cl = y\_train3, k = k)  
knn\_predictions <- as.factor(knn\_model)  
  
conf\_matrix <- table(knn\_predictions, y\_test3)  
accuracy <- function(x){sum(diag(x)/(sum(rowSums(x)))) \* 100}  
accuracy(conf\_matrix)

## [1] 72.85714

k <- 10  
knn\_model2 <- knn(train = X\_train3, test = X\_test3,cl = y\_train3, k = 10)  
knn\_predictions2 <- as.factor(knn\_model2)  
print(knn\_predictions2)

## [1] 1 0 0 0 0 0 0 0 1 0 0 0 0 1 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0  
## [38] 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 1 0 0 1 0 0 0 0 1 0 0 0 0 0 0 1  
## [75] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 1 0  
## [112] 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 0 0 0 1 0 0 0 0 0 1  
## Levels: 0 1

conf\_matrix2 <- table(knn\_predictions2, y\_test3)  
accuracy <- function(x){sum(diag(x)/(sum(rowSums(x)))) \* 100}  
accuracy(conf\_matrix2)

## [1] 70.71429

### Gradient boosting ###  
#   
# model\_gbm = gbm(y\_train ~.,  
# data = X\_train,  
# distribution = "adaboost",  
# cv.folds = 10,  
# shrinkage = .01,  
# n.minobsinnode = 10,  
# n.trees = 50)  
#   
# summary(model\_gbm)

Random forest

library(randomForest)

## Warning: package 'randomForest' was built under R version 4.2.3

## randomForest 4.7-1.1

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

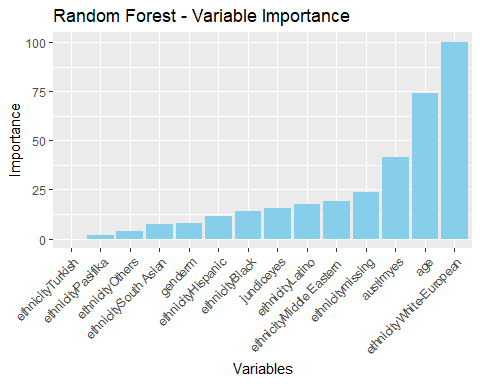
##   
## Attaching package: 'randomForest'

## The following object is masked from 'package:psych':  
##   
## outlier

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

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

library(caret)  
library(ggplot2)  
# Set the number of folds for cross-validation  
num\_folds <- 5  
  
# Create the control parameters for cross-validation  
ctrl <- trainControl(method = "cv",   
 number = num\_folds,   
 savePredictions = TRUE,  
 classProbs = TRUE)  
levels(y\_train\_f)<- make.names(levels(y\_train\_f))  
  
  
# Build the random forest model with cross-validation  
rf <- train(y = y\_train\_f, x = X\_train3, method = "rf", trControl = ctrl)  
  
# Get the cross-validation results  
cv\_results <- rf$results  
importance <- varImp(rf)  
  
importance\_df <- data.frame(  
 Variables = row.names(importance$importance),  
 Importance = importance$importance[, 1] # MeanDecreaseGini importance measure  
)  
  
importance\_df <- importance\_df[order(importance\_df$Importance, decreasing = TRUE), ]  
  
p <- ggplot(importance\_df, aes(x = reorder(Variables, Importance), y = Importance)) +  
 geom\_bar(stat = "identity", fill = "skyblue") +  
 xlab("Variables") +  
 ylab("Importance") +  
 ggtitle("Random Forest - Variable Importance") +  
 theme(axis.text.x = element\_text(angle = 45, hjust = 1))  
print(p)

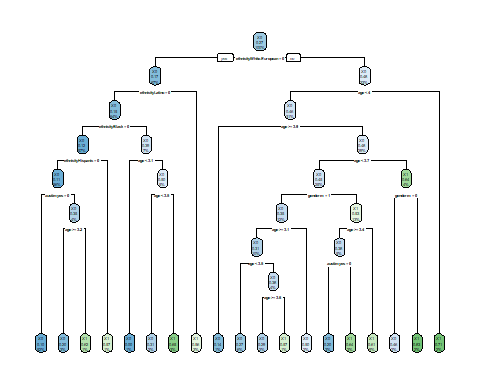


tree\_slips <- as.data.frame(rf$err.rate)

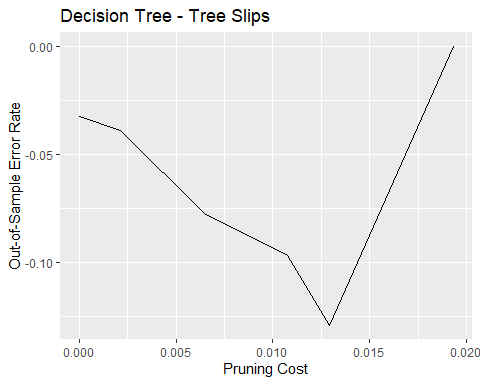
library(rpart)  
library(rpart.plot)

## Warning: package 'rpart.plot' was built under R version 4.2.3

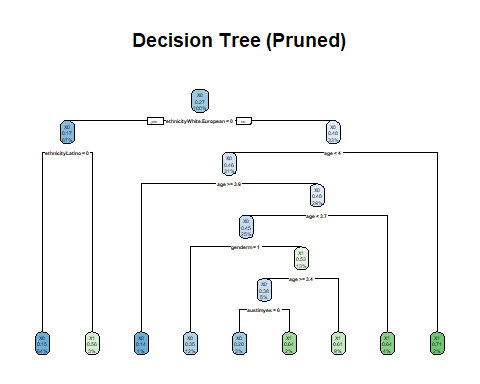
library(ggplot2)  
  
# Set the number of folds for cross-validation  
num\_folds <- 10  
  
# Build the decision tree model with cross-validation  
dt <- rpart(formula = y\_train\_f ~ ., data = data.frame(X\_train3),  
 control = rpart.control(cp = 0),  
 parms = list(split = "information"),  
 method = "class",  
 xval = num\_folds)  
  
options(repr.plot.width = 10, repr.plot.height = 80)  
rpart.plot(dt)



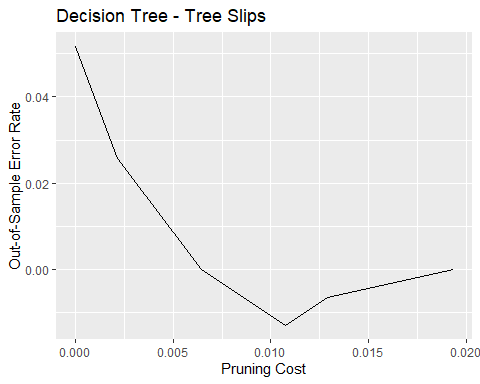
# Calculate the out-of-sample error rate  
oos\_error <- 1 - dt$cptable[, "xerror"]  
  
# Create the tree slips plot  
p <- ggplot(data.frame(Pruning\_Cost = dt$cptable[, "CP"], Out\_of\_Sample\_Error = oos\_error),  
 aes(x = Pruning\_Cost, y = Out\_of\_Sample\_Error)) +  
 geom\_line() +  
 xlab("Pruning Cost") +  
 ylab("Out-of-Sample Error Rate") +  
 ggtitle("Decision Tree - Tree Slips")  
  
print(p)



# Model 2  
num\_folds <- 10  
dt2 <- rpart(formula = y\_train\_f~ ., data = data.frame(X\_train3),  
 control = rpart.control(cp = 0),  
 parms = list(split = "information"),  
 method = "class",  
 xval = num\_folds)  
  
# Prune the tree to the first 3 levels  
pruned\_dt <- prune(tree = dt2, cp = dt2$cptable[3, "CP"])  
  
# Increase the size of the plot  
options(repr.plot.width = 10, repr.plot.height = 8) # Adjust the width and height as needed  
  
# Plot the pruned decision tree (first four levels)  
rpart.plot(pruned\_dt, main = "Decision Tree (Pruned)", fallen.leaves = TRUE, cex.main = 1.2)



oos\_error <- 1 - dt2$cptable[, "xerror"]  
  
# Create a data frame with Pruning\_Cost and Out\_of\_Sample\_Error  
data <- data.frame(Pruning\_Cost = dt2$cptable[, "CP"], Out\_of\_Sample\_Error = oos\_error)  
  
# Create the tree slips plot  
p2 <- ggplot(data, aes(x = Pruning\_Cost, y = Out\_of\_Sample\_Error)) +  
 geom\_line() +  
 xlab("Pruning Cost") +  
 ylab("Out-of-Sample Error Rate") +  
 ggtitle("Decision Tree - Tree Slips")  
  
# Print the tree slips plot  
print(p2)



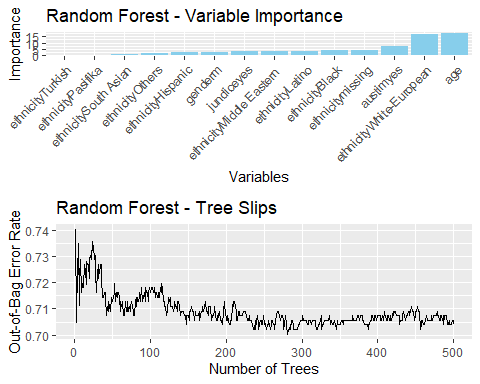
library(randomForest)  
library(ggplot2)  
  
# Set the number of folds for cross-validation  
num\_folds <- 10  
  
# Build the random forest model with cross-validation and keep.forest = TRUE  
rf <- randomForest(x = X\_train3, y = y\_train\_f, keep.forest = TRUE, cv.fold = num\_folds)  
  
# Get the variable importance  
importance <- importance(rf)  
  
importance\_df <- data.frame(  
 Variables = row.names(importance),  
 Importance = importance[, 1] # Mean Decrease Gini importance measure  
)  
importance\_df <- importance\_df[order(importance\_df$Importance, decreasing = TRUE), ]  
  
# Create the variable importance plot  
p1 <- ggplot(importance\_df, aes(x = reorder(Variables, Importance), y = Importance)) +  
 geom\_bar(stat = "identity", fill = "skyblue") +  
 xlab("Variables") +  
 ylab("Importance") +  
 ggtitle("Random Forest - Variable Importance") +  
 theme(axis.text.x = element\_text(angle = 45, hjust = 1))  
  
# Calculate the OOB error rate  
oob\_error <- data.frame(  
 Number\_of\_Trees = seq(1, rf$ntree),  
 OOB\_Error\_Rate = 1 - rf$err.rate[, "OOB"]  
)  
  
# Create the tree slips plot  
p2 <- ggplot(oob\_error, aes(x = Number\_of\_Trees, y = OOB\_Error\_Rate)) +  
 geom\_line() +  
 xlab("Number of Trees") +  
 ylab("Out-of-Bag Error Rate") +  
 ggtitle("Random Forest - Tree Slips")  
  
# Combine the plots  
library(gridExtra)

##   
## Attaching package: 'gridExtra'

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

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

combined\_plot <- grid.arrange(p1, p2, nrow = 2)



print(combined\_plot)

## TableGrob (2 x 1) "arrange": 2 grobs  
## z cells name grob  
## 1 1 (1-1,1-1) arrange gtable[layout]  
## 2 2 (2-2,1-1) arrange gtable[layout]

### Factor analysis = UNSUPERVISED ###  
# \* 0.00 to 0.49 unacceptable  
# \* 0.50 to 0.59 miserable  
# \* 0.60 to 0.69 mediocre  
# \* 0.70 to 0.79 middling  
# \* 0.80 to 0.89 meritorious  
# \* 0.90 to 1.00 marvelous  
  
KMO(X\_train3)

## Kaiser-Meyer-Olkin factor adequacy  
## Call: KMO(r = X\_train3)  
## Overall MSA = 0.19  
## MSA for each item =   
## age genderm ethnicityBlack   
## 0.74 0.60 0.10   
## ethnicityHispanic ethnicityLatino ethnicityMiddle Eastern   
## 0.08 0.11 0.14   
## ethnicitymissing ethnicityOthers ethnicityPasifika   
## 0.14 0.09 0.09   
## ethnicitySouth Asian ethnicityTurkish ethnicityWhite-European   
## 0.11 0.09 0.23   
## jundiceyes austimyes   
## 0.63 0.56

print(KMO(X\_train3))

## Kaiser-Meyer-Olkin factor adequacy  
## Call: KMO(r = X\_train3)  
## Overall MSA = 0.19  
## MSA for each item =   
## age genderm ethnicityBlack   
## 0.74 0.60 0.10   
## ethnicityHispanic ethnicityLatino ethnicityMiddle Eastern   
## 0.08 0.11 0.14   
## ethnicitymissing ethnicityOthers ethnicityPasifika   
## 0.14 0.09 0.09   
## ethnicitySouth Asian ethnicityTurkish ethnicityWhite-European   
## 0.11 0.09 0.23   
## jundiceyes austimyes   
## 0.63 0.56

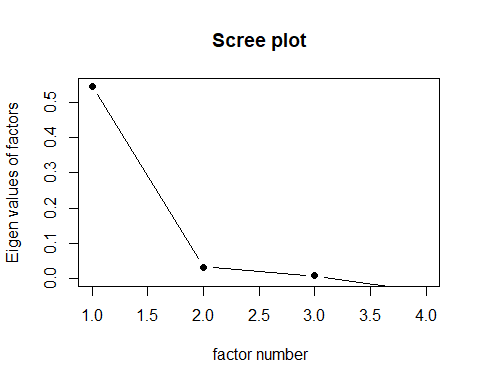
factor\_df <- X\_train3[, KMO(X\_train3)$MSAi>0.50]  
round( KMO(factor\_df)$MSA, 1 )

## [1] 0.6

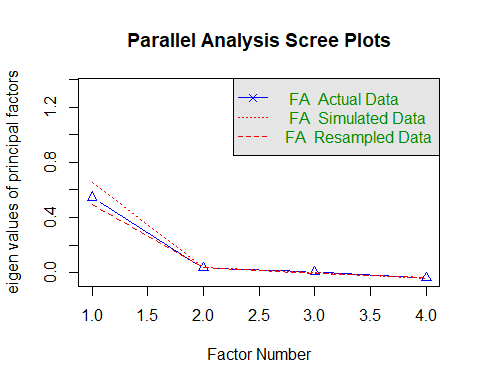
ev <- eigen(cor(factor\_df)) # get eigenvalues  
ev$values

## [1] 1.3519376 0.9907409 0.8684812 0.7888402

scree(factor\_df, pc=FALSE)



fa.parallel(factor\_df, fa="fa")



## Parallel analysis suggests that the number of factors = 0 and the number of components = NA

#Parallel analysis suggests that the number of factors = 3 and the number of components = NA   
# trying 2 factors  
# facs <- 2  
# fit <- factanal(factor\_df, facs, rotation="promax")  
# print(fit, digits=2, cutoff=0.3, sort=TRUE)  
#   
# ## Correlation are low -> tryong oblique  
# facs <- 2  
# fit <- factanal(factor\_df, facs, rotation="oblimin")  
# print(fit, digits=2, cutoff=0.3, sort=TRUE)

library(arules)

## Warning: package 'arules' was built under R version 4.2.3

##   
## Attaching package: 'arules'

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

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

## The following objects are masked from 'package:base':  
##   
## abbreviate, write

matrix4 <- as.matrix(data3)  
transactions <- as(matrix4, "transactions")

## Warning in asMethod(object): matrix contains values other than 0 and 1! Setting  
## all entries != 0 to 1.

rules <- apriori(transactions, parameter = list(support = 0.01, confidence = 0.5, minlen = 2), appearance = list(rhs = "y3"))

## Apriori  
##   
## Parameter specification:  
## confidence minval smax arem aval originalSupport maxtime support minlen  
## 0.5 0.1 1 none FALSE TRUE 5 0.01 2  
## maxlen target ext  
## 10 rules TRUE  
##   
## Algorithmic control:  
## filter tree heap memopt load sort verbose  
## 0.1 TRUE TRUE FALSE TRUE 2 TRUE  
##   
## Absolute minimum support count: 7   
##   
## set item appearances ...[1 item(s)] done [0.00s].  
## set transactions ...[15 item(s), 704 transaction(s)] done [0.00s].  
## sorting and recoding items ... [14 item(s)] done [0.00s].  
## creating transaction tree ... done [0.00s].  
## checking subsets of size 1 2 3 4 5 done [0.00s].  
## writing ... [10 rule(s)] done [0.00s].  
## creating S4 object ... done [0.00s].

# Show the generated rules  
inspect(rules)

## lhs rhs support confidence coverage lift count  
## [1] {ethnicityLatino} => {y3} 0.01420455 0.5000 0.02840909 1.862434 10  
## [2] {age,   
## ethnicityLatino} => {y3} 0.01420455 0.5000 0.02840909 1.862434 10  
## [3] {jundiceyes,   
## austimyes} => {y3} 0.01562500 0.5500 0.02840909 2.048677 11  
## [4] {ethnicityWhite-European,   
## jundiceyes} => {y3} 0.02556818 0.5625 0.04545455 2.095238 18  
## [5] {ethnicityWhite-European,   
## austimyes} => {y3} 0.03835227 0.5400 0.07102273 2.011429 27  
## [6] {age,   
## jundiceyes,   
## austimyes} => {y3} 0.01562500 0.5500 0.02840909 2.048677 11  
## [7] {age,   
## ethnicityWhite-European,   
## jundiceyes} => {y3} 0.02556818 0.5625 0.04545455 2.095238 18  
## [8] {genderm,   
## ethnicityWhite-European,   
## austimyes} => {y3} 0.01278409 0.5625 0.02272727 2.095238 9  
## [9] {age,   
## ethnicityWhite-European,   
## austimyes} => {y3} 0.03835227 0.5400 0.07102273 2.011429 27  
## [10] {age,   
## genderm,   
## ethnicityWhite-European,   
## austimyes} => {y3} 0.01278409 0.5625 0.02272727 2.095238 9