Autism traits in Toddlers

setwd("C:\\Users\\arina\\OneDrive\\Documents\\DSE\\ML\\Salini\\projects\\ML projects")  
df = read.csv('Toddler Autism.csv')  
df = as.data.frame(df)  
summary(df)

## Case\_No A1 A2 A3   
## Min. : 1.0 Min. :0.0000 Min. :0.0000 Min. :0.0000   
## 1st Qu.: 264.2 1st Qu.:0.0000 1st Qu.:0.0000 1st Qu.:0.0000   
## Median : 527.5 Median :1.0000 Median :0.0000 Median :0.0000   
## Mean : 527.5 Mean :0.5636 Mean :0.4488 Mean :0.4013   
## 3rd Qu.: 790.8 3rd Qu.:1.0000 3rd Qu.:1.0000 3rd Qu.:1.0000   
## Max. :1054.0 Max. :1.0000 Max. :1.0000 Max. :1.0000   
## A4 A5 A6 A7   
## Min. :0.0000 Min. :0.0000 Min. :0.0000 Min. :0.0000   
## 1st Qu.:0.0000 1st Qu.:0.0000 1st Qu.:0.0000 1st Qu.:0.0000   
## Median :1.0000 Median :1.0000 Median :1.0000 Median :1.0000   
## Mean :0.5123 Mean :0.5247 Mean :0.5769 Mean :0.6499   
## 3rd Qu.:1.0000 3rd Qu.:1.0000 3rd Qu.:1.0000 3rd Qu.:1.0000   
## Max. :1.0000 Max. :1.0000 Max. :1.0000 Max. :1.0000   
## A8 A9 A10 age   
## Min. :0.0000 Min. :0.0000 Min. :0.0000 Min. :12.00   
## 1st Qu.:0.0000 1st Qu.:0.0000 1st Qu.:0.0000 1st Qu.:23.00   
## Median :0.0000 Median :0.0000 Median :1.0000 Median :30.00   
## Mean :0.4592 Mean :0.4896 Mean :0.5863 Mean :27.87   
## 3rd Qu.:1.0000 3rd Qu.:1.0000 3rd Qu.:1.0000 3rd Qu.:36.00   
## Max. :1.0000 Max. :1.0000 Max. :1.0000 Max. :36.00   
## score sex Ethnicity Jaundice   
## Min. : 0.000 Length:1054 Length:1054 Length:1054   
## 1st Qu.: 3.000 Class :character Class :character Class :character   
## Median : 5.000 Mode :character Mode :character Mode :character   
## Mean : 5.213   
## 3rd Qu.: 8.000   
## Max. :10.000   
## Family\_mem\_with\_ASD Who.completed.the.test Class.ASD.Traits   
## Length:1054 Length:1054 Length:1054   
## Class :character Class :character Class :character   
## Mode :character Mode :character Mode :character   
##   
##   
##

#sapply(df, class)

ethnicity\_counts <- table(df$Ethnicity)  
print(ethnicity\_counts)

##   
## asian black Hispanic Latino middle eastern   
## 299 53 40 26 188   
## mixed Native Indian Others Pacifica south asian   
## 8 3 35 8 60   
## White European   
## 334

# Combine others  
df$Ethnicity[df$Ethnicity %in% c("mixed","Native Indian")] <- "Others"  
df$Ethnicity[df$Ethnicity %in% c("Pacifica","south asian")] <- "south asian"

# #   
# ethnicity\_counts <- table(df$Ethnicity)  
# print(ethnicity\_counts)

df$sex <- factor(df$sex)  
df$Ethnicity <- factor(df$Ethnicity)  
df$Jaundice <- factor(df$Jaundice)  
df$Family\_mem\_with\_ASD <- factor(df$Family\_mem\_with\_ASD)  
df$Who.completed.the.test <- factor(df$Who.completed.the.test)  
df$Class.ASD.Traits <- factor(df$Class.ASD.Traits)

factor\_cols <- sapply(df, is.factor)  
  
# Create a data frame with only the factor columns  
factor\_df <- df[, factor\_cols]  
dummy\_model <- dummyVars(~., data = factor\_df)  
encoded\_data <- predict(dummy\_model, newdata = factor\_df)  
df2 <- cbind(df[, !factor\_cols], encoded\_data)

grep("sex.m", colnames(df2))

## [1] 15

grep("Jaundice.no", colnames(df2))

## [1] 24

grep("score", colnames(df2))

## [1] 13

grep("Family\_mem\_with\_ASD.no", colnames(df2))

## [1] 26

grep("Ethnicity.White European", colnames(df2))

## [1] 23

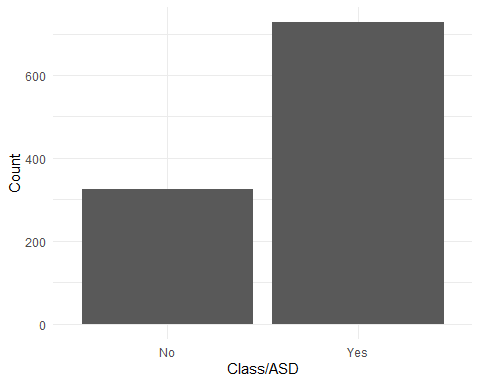
#grep("Ethnicity.Other", colnames(df2))  
grep("Ethnicity.Others", colnames(df2))

## [1] 21

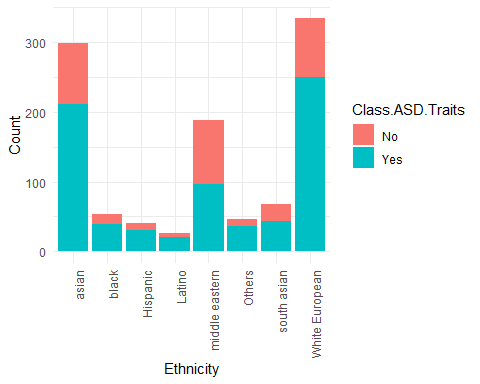
df3\_2 <- df2[,-c(1:11,13,15,24,26,21,28:33)]  
df3\_2$"age" = log(df3\_2$"age"+1)  
#View(df3\_2)  
colnames(df3\_2)

## [1] "age" "sex.f"   
## [3] "Ethnicity.asian" "Ethnicity.black"   
## [5] "Ethnicity.Hispanic" "Ethnicity.Latino"   
## [7] "Ethnicity.middle eastern" "Ethnicity.south asian"   
## [9] "Ethnicity.White European" "Jaundice.yes"   
## [11] "Family\_mem\_with\_ASD.yes" "Class.ASD.Traits.Yes"

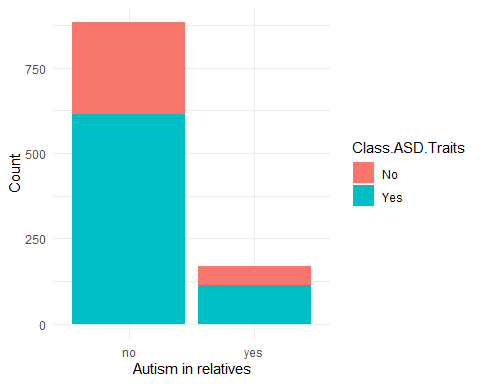
### EDA ###  
  
#yes vs. no autism  
ggplot(df, aes(x = `Class.ASD.Traits`)) +  
 geom\_bar(stat = "count") +  
 labs(x = "Class/ASD", y = "Count") +  
 theme\_minimal()



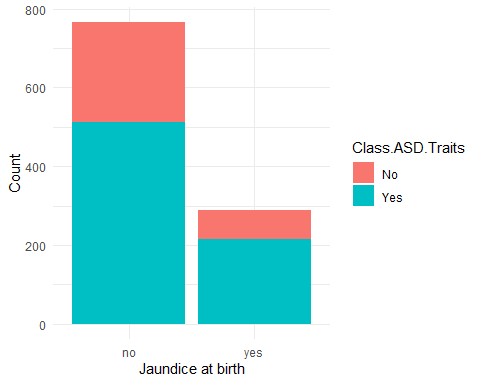
#ethnicities  
ggplot(df, aes(x = Ethnicity, fill = `Class.ASD.Traits`)) +  
 geom\_bar(stat = "count") +  
 labs(x = "Ethnicity", y = "Count") +  
 theme\_minimal()+ theme(axis.text.x = element\_text(angle = 90, hjust = 1))



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



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



# #oversampling  
#   
# # Separate Target Classes  
# df\_1 <- df3\_2[df3\_2$"Class.ASD.Traits.Yes" == 0, ]  
# df\_2 <- df3\_2[df3\_2$"Class.ASD.Traits.Yes" == 1, ]  
#   
# # Upsample minority class  
# df\_1\_upsampled <- df\_1[sample(nrow(df\_1), 450, replace = TRUE), ]  
#   
# # Combine majority class with upsampled minority class  
# df3 <- rbind(df\_2, df\_1\_upsampled)  
#   
# # Display new class counts  
# class\_counts <- table(df3$"Class.ASD.Traits.Yes")  
# barplot(class\_counts, main = "ASD", xlab = "Class/ASD", ylab = "Count")

X <- df3\_2[,-c(12)]  
y <- df3\_2$Class.ASD.Traits.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]  
  
#y\_train <- factor(y\_train, levels = c(0, 1))  
#y\_test <- factor(y\_test, levels = c(0, 1))  
  
missing\_values\_sum <- colSums(is.na(df3\_2))  
print(missing\_values\_sum)

## age sex.f Ethnicity.asian   
## 0 0 0   
## Ethnicity.black Ethnicity.Hispanic Ethnicity.Latino   
## 0 0 0   
## Ethnicity.middle eastern Ethnicity.south asian Ethnicity.White European   
## 0 0 0   
## Jaundice.yes Family\_mem\_with\_ASD.yes Class.ASD.Traits.Yes   
## 0 0 0

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

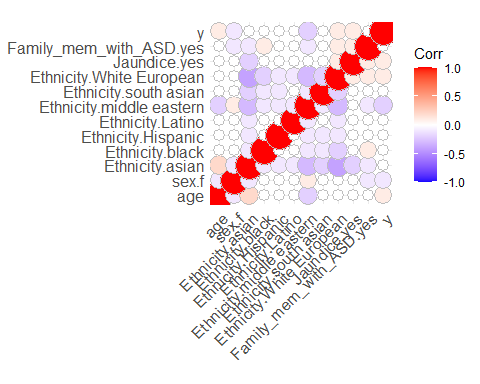
## [1] TRUE

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.

library(ggcorrplot)

data <- cbind(X, y)  
corr <- round(cor(data), 1)  
p.mat <- cor\_pmat(data)  
ggcorrplot(corr, method = "circle")



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

##   
## Call:  
## lm(formula = y\_train ~ ., data = X\_train)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -0.8532 -0.5393 0.2299 0.3063 0.6737   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) 0.5419072 0.1895869 2.858 0.004365 \*\*   
## age 0.0821433 0.0502444 1.635 0.102453   
## sex.f -0.1342151 0.0346815 -3.870 0.000117 \*  
## Ethnicity.asian -0.0938606 0.0797345 -1.177 0.239467   
## Ethnicity.black -0.0555422 0.1003108 -0.554 0.579932   
## Ethnicity.Hispanic -0.0238714 0.1123869 -0.212 0.831844   
## Ethnicity.Latino 0.0005953 0.1238228 0.005 0.996165   
## `Ethnicity.middle eastern` -0.2400030 0.0834716 -2.875 0.004140 \*\*   
## `Ethnicity.south asian` -0.0839756 0.0982069 -0.855 0.392749   
## `Ethnicity.White European` -0.0511970 0.0789478 -0.648 0.516846   
## Jaundice.yes 0.0659016 0.0357386 1.844 0.065540 \*   
## Family\_mem\_with\_ASD.yes -0.0691659 0.0436891 -1.583 0.113769   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 0.4532 on 832 degrees of freedom  
## Multiple R-squared: 0.05985, Adjusted R-squared: 0.04742   
## F-statistic: 4.815 on 11 and 832 DF, p-value: 3.189e-07

# library(logistf)  
#   
# # Fit the logistic regression model using Firth's method  
# model <- logistf(y\_train ~ ., data = X\_train)  
#   
# # Print the model summary  
# summary(model)  
  
log\_model <- glm(y\_train~., data = X\_train, family = binomial(link = "logit"))  
  
summary(log\_model)

##   
## Call:  
## glm(formula = y\_train ~ ., family = binomial(link = "logit"),   
## data = X\_train)  
##   
## Deviance Residuals:   
## Min 1Q Median 3Q Max   
## -1.9013 -1.2504 0.7181 0.8472 1.5507   
##   
## Coefficients:  
## Estimate Std. Error z value Pr(>|z|)   
## (Intercept) 0.19999 0.92711 0.216 0.829214   
## age 0.38311 0.24079 1.591 0.111597   
## sex.f -0.62638 0.16375 -3.825 0.000131 \*\*  
## Ethnicity.asian -0.51107 0.43052 -1.187 0.235184   
## Ethnicity.black -0.32381 0.52490 -0.617 0.537303   
## Ethnicity.Hispanic -0.13826 0.60065 -0.230 0.817945   
## Ethnicity.Latino -0.02099 0.66055 -0.032 0.974656   
## `Ethnicity.middle eastern` -1.13723 0.44038 -2.582 0.009813 \*\*   
## `Ethnicity.south asian` -0.46694 0.51183 -0.912 0.361617   
## `Ethnicity.White European` -0.29170 0.42848 -0.681 0.496014   
## Jaundice.yes 0.33660 0.18080 1.862 0.012648 \*   
## Family\_mem\_with\_ASD.yes -0.34362 0.20996 -1.637 0.101717   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## (Dispersion parameter for binomial family taken to be 1)  
##   
## Null deviance: 1050.4 on 843 degrees of freedom  
## Residual deviance: 1000.5 on 832 degrees of freedom  
## AIC: 1024.5  
##   
## Number of Fisher Scoring iterations: 4

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

## [1] 0.1435976

ridge <- glmnet(X\_train, y\_train, alpha = 0, lambda = cv$lambda.min)  
coef(ridge)

## 12 x 1 sparse Matrix of class "dgCMatrix"  
## s0  
## (Intercept) 0.482825356  
## age 0.072855058  
## sex.f -0.103789257  
## Ethnicity.asian -0.008150298  
## Ethnicity.black 0.017047976  
## Ethnicity.Hispanic 0.043692098  
## Ethnicity.Latino 0.060011579  
## Ethnicity.middle eastern -0.130549726  
## Ethnicity.south asian -0.002829829  
## Ethnicity.White European 0.024770342  
## Jaundice.yes 0.052928200  
## Family\_mem\_with\_ASD.yes -0.047213665

predictions <- ridge %>% predict(X\_test) %>% as.vector()  
data.frame(  
 RMSE = RMSE(predictions, y\_test),  
 Rsquare = R2(predictions, y\_test)  
)

## RMSE Rsquare  
## 1 0.4488392 0.02638249

### Lasso regression ###  
  
cv2 <- cv.glmnet(X\_train, y\_train, alpha = 1)  
cv2$lambda.min

## [1] 0.008608453

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

## 12 x 1 sparse Matrix of class "dgCMatrix"  
## s0  
## (Intercept) 0.556869861  
## age 0.057365662  
## sex.f -0.115871948  
## Ethnicity.asian -0.015580292  
## Ethnicity.black .   
## Ethnicity.Hispanic .   
## Ethnicity.Latino 0.006304614  
## Ethnicity.middle eastern -0.165192707  
## Ethnicity.south asian .   
## Ethnicity.White European .   
## Jaundice.yes 0.049998109  
## Family\_mem\_with\_ASD.yes -0.036789985

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

## RMSE Rsquare  
## 1 0.4499703 0.02310527

### Elastic net###  
y\_train\_f <- factor(y\_train, levels = c(0, 1))  
y\_test\_f <- factor(y\_test, levels = c(0, 1))  
  
el\_model <- train(X\_train, y\_train\_f, method = "glmnet",  
 trControl = trainControl("cv", number = 5),  
 tuneLength = 10)  
# Best tuning parameter  
el\_model$bestTune

## alpha lambda  
## 14 0.2 0.01302396

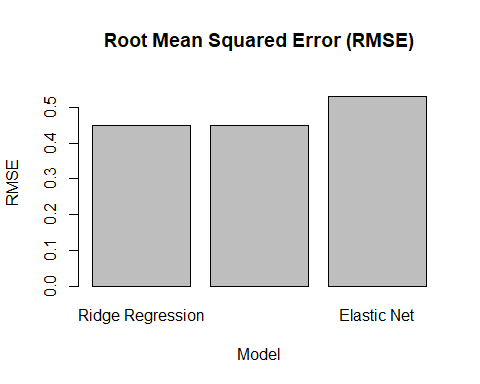
coef\_el\_model <- coef(el\_model$finalModel, s = el\_model$bestTune$lambda)  
  
# Print the coefficients  
print(coef\_el\_model)

## 12 x 1 sparse Matrix of class "dgCMatrix"  
## s1  
## (Intercept) -0.0095741228  
## age 0.3449649763  
## sex.f -0.5680099856  
## Ethnicity.asian -0.1633040681  
## Ethnicity.black .   
## Ethnicity.Hispanic 0.0856645322  
## Ethnicity.Latino 0.1831154602  
## Ethnicity.middle eastern -0.7793281050  
## Ethnicity.south asian -0.0966455956  
## Ethnicity.White European 0.0002703047  
## Jaundice.yes 0.2980643891  
## Family\_mem\_with\_ASD.yes -0.2712673086

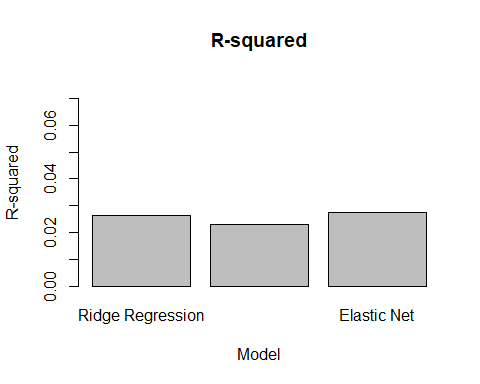
# Make predictions on the test set  
predictions3 <- predict(el\_model, newdata = X\_test)  
  
predictions3 <- as.numeric(as.character(predictions3))  
y\_test\_f <- as.numeric(as.character(y\_test\_f))  
  
# Model performance metrics  
data.frame(  
 RMSE = RMSE(predictions3, y\_test\_f),  
 Rsquare = R2(predictions3, y\_test\_f)  
)

## RMSE Rsquare  
## 1 0.5300494 0.02735646

### Graphs ###  
# Model 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 <- 31  
knn\_model <- knn(train = X\_train, test = X\_test, cl = y\_train, k = k)  
knn\_predictions <- as.factor(knn\_model)  
  
conf\_matrix <- table(knn\_predictions, y\_test)  
accuracy <- function(x){sum(diag(x)/(sum(rowSums(x)))) \* 100}  
accuracy(conf\_matrix)

## [1] 71.42857

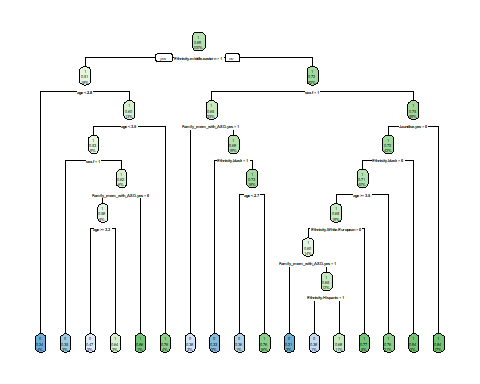
k <- 10  
knn\_model2 <- knn(train = X\_train, test = X\_test, cl = y\_train, k = k)  
knn\_predictions2 <- as.factor(knn\_model2)  
  
conf\_matrix2 <- table(knn\_predictions2, y\_test)  
accuracy <- function(x){sum(diag(x)/(sum(rowSums(x)))) \* 100}  
accuracy(conf\_matrix2)

## [1] 68.57143

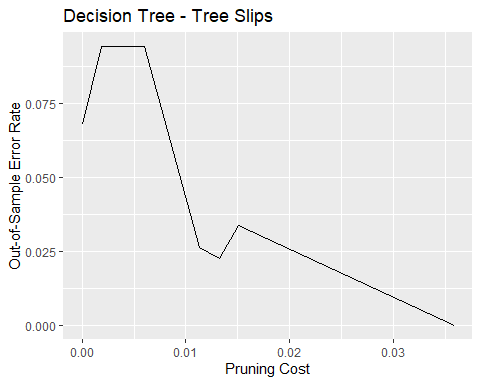
library(rpart)  
library(rpart.plot)

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

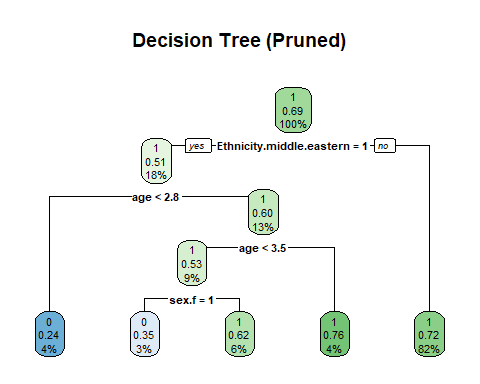
library(ggplot2)  
num\_folds <- 10  
  
# Build the decision tree model with cross-validation  
dt <- rpart(formula = y\_train~ ., data = data.frame(X\_train, y\_train),  
 control = rpart.control(cp = 0),  
 parms = list(split = "information"),  
 method = "class",  
 xval = num\_folds)  
  
options(repr.plot.width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



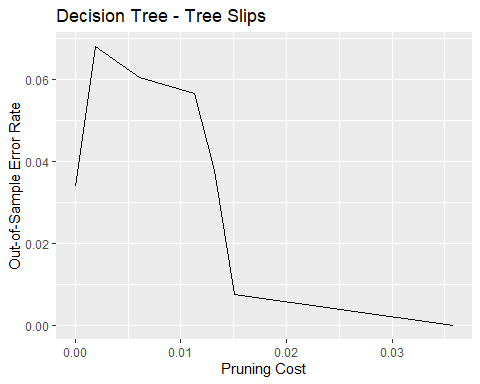
# Calculate the out-of-sample error rate  
oos\_error <- 1 - dt$cptable[, "xerror"]  
p <- ggplot(data.frame(Pruning\_Cost = dt$cptable[, "CP"], Out\_of\_Sample\_Error = oos\_error),  
 aes(x = Pruning\_Cost, y = Out\_of\_Sample\_Error)) +  
 geom\_line() +  
 xlab("Pruning Cost") +  
 ylab("Out-of-Sample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



# Model 2: Pruned  
num\_folds <- 10  
dt2 <- rpart(formula = y\_train~ ., data = data.frame(X\_train, y\_train),  
 control = rpart.control(cp = 0),  
 parms = list(split = "information"),  
 method = "class",  
 xval = num\_folds)  
  
# Prune the tree to the first four levels  
pruned\_dt <- prune(tree = dt2, cp = dt2$cptable[3, "CP"])  
options(repr.plot.width = 10, repr.plot.height = 6)   
rpart.plot(pruned\_dt, main = "Decision Tree (Pruned)", fallen.leaves = TRUE, cex.main = 1.2)

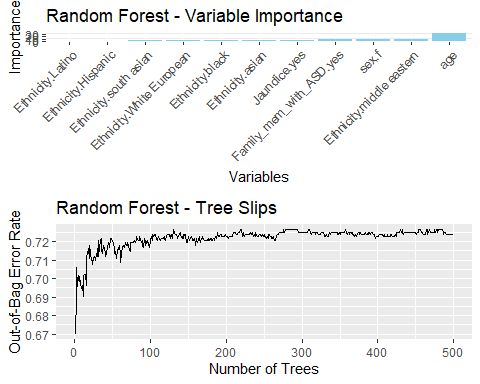


oos\_error <- 1 - dt2$cptable[, "xerror"]  
  
# Create a data frame with Pruning\_Cost and Out\_of\_Sample\_Error  
data <- data.frame(Pruning\_Cost = dt2$cptable[, "CP"], Out\_of\_Sample\_Error = oos\_error)  
  
# Create the tree slips plot  
p2 <- ggplot(data, aes(x = Pruning\_Cost, y = Out\_of\_Sample\_Error))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



num\_folds <- 10  
  
# Build the random forest model with cross-validation and keep.forest = TRUE  
rf <- randomForest(x = X\_train, y = y\_train\_f, keep.forest = TRUE, cv.fold = num\_folds)  
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")  
  
  
library(gridExtra)

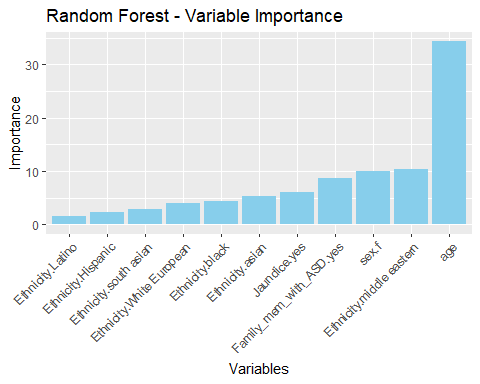
combined\_plot <- grid.arrange(p1, p2, nrow = 2)



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]

print(p1)



### 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\_train)

## Kaiser-Meyer-Olkin factor adequacy  
## Call: KMO(r = X\_train)  
## Overall MSA = 0.09  
## MSA for each item =   
## age sex.f Ethnicity.asian   
## 0.63 0.71 0.12   
## Ethnicity.black Ethnicity.Hispanic Ethnicity.Latino   
## 0.03 0.03 0.03   
## Ethnicity.middle eastern Ethnicity.south asian Ethnicity.White European   
## 0.09 0.03 0.12   
## Jaundice.yes Family\_mem\_with\_ASD.yes   
## 0.65 0.76

print(KMO(X\_train))

## Kaiser-Meyer-Olkin factor adequacy  
## Call: KMO(r = X\_train)  
## Overall MSA = 0.09  
## MSA for each item =   
## age sex.f Ethnicity.asian   
## 0.63 0.71 0.12   
## Ethnicity.black Ethnicity.Hispanic Ethnicity.Latino   
## 0.03 0.03 0.03   
## Ethnicity.middle eastern Ethnicity.south asian Ethnicity.White European   
## 0.09 0.03 0.12   
## Jaundice.yes Family\_mem\_with\_ASD.yes   
## 0.65 0.76

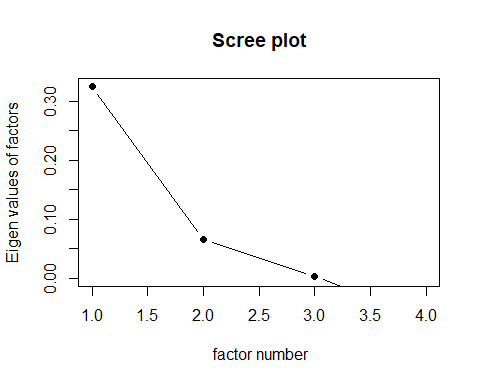
factor\_df <- X\_train[, KMO(X\_train)$MSAi>0.50]  
round( KMO(factor\_df)$MSA, 1 )

## [1] 0.5

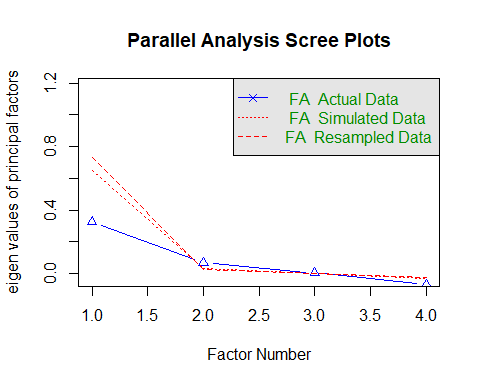
ev <- eigen(cor(factor\_df)) # get eigenvalues  
ev$values

## [1] 1.1791536 1.0419692 0.9232455 0.8556318

scree(factor\_df, pc=FALSE)



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



matrix <- as.matrix(df3\_2)  
  
# Convert the matrix to a transactions object  
transactions <- as(matrix, "transactions")

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

# Apply the Apriori algorithm  
rules <- apriori(transactions, parameter = list(support = 0.1, confidence = 0.5, minlen = 2))

## Apriori  
##   
## Parameter specification:  
## confidence minval smax arem aval originalSupport maxtime support minlen  
## 0.5 0.1 1 none FALSE TRUE 5 0.1 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: 105   
##   
## set item appearances ...[0 item(s)] done [0.00s].  
## set transactions ...[12 item(s), 1054 transaction(s)] done [0.00s].  
## sorting and recoding items ... [8 item(s)] done [0.00s].  
## creating transaction tree ... done [0.00s].  
## checking subsets of size 1 2 3 done [0.00s].  
## writing ... [24 rule(s)] done [0.00s].  
## creating S4 object ... done [0.00s].

# Filter rules to include only those with rhs => {Class.ASD.Traits.Yes}  
filtered\_rules <- subset(rules, subset = rhs %in% "Class.ASD.Traits.Yes")  
  
# Show the generated rules  
inspect(filtered\_rules)

## lhs rhs support confidence coverage lift count  
## [1] {Family\_mem\_with\_ASD.yes} => {Class.ASD.Traits.Yes} 0.1091082 0.6764706 0.1612903 0.9793956 115  
## [2] {Ethnicity.asian} => {Class.ASD.Traits.Yes} 0.2011385 0.7090301 0.2836812 1.0265353 212  
## [3] {Jaundice.yes} => {Class.ASD.Traits.Yes} 0.2039848 0.7465278 0.2732448 1.0808246 215  
## [4] {Ethnicity.White European} => {Class.ASD.Traits.Yes} 0.2371917 0.7485030 0.3168880 1.0836843 250  
## [5] {sex.f} => {Class.ASD.Traits.Yes} 0.1840607 0.6081505 0.3026565 0.8804816 194  
## [6] {age} => {Class.ASD.Traits.Yes} 0.6907021 0.6907021 1.0000000 1.0000000 728  
## [7] {age,   
## Family\_mem\_with\_ASD.yes} => {Class.ASD.Traits.Yes} 0.1091082 0.6764706 0.1612903 0.9793956 115  
## [8] {age,   
## Ethnicity.asian} => {Class.ASD.Traits.Yes} 0.2011385 0.7090301 0.2836812 1.0265353 212  
## [9] {age,   
## Jaundice.yes} => {Class.ASD.Traits.Yes} 0.2039848 0.7465278 0.2732448 1.0808246 215  
## [10] {age,   
## Ethnicity.White European} => {Class.ASD.Traits.Yes} 0.2371917 0.7485030 0.3168880 1.0836843 250  
## [11] {age,   
## sex.f} => {Class.ASD.Traits.Yes} 0.1840607 0.6081505 0.3026565 0.8804816 194

# ### PCA  
#   
# scaled\_X <-scale(X)  
# pca\_result <- prcomp(scaled\_X)  
# variance\_ratio <- pca\_result$sdev^2 / sum(pca\_result$sdev^2)  
# # Scree plot  
# plot(variance\_ratio, type = "b", xlab = "Principal Component", ylab = "Explained Variance Ratio")  
#   
# # Biplot  
# biplot(pca\_result)  
#   
#