Appendix B - R Code

```
library(ggplot2)
library(ggfortify)
library(dplyr)
library(class)
library(caret)
library(e1071)
library(ROCR)
library(tidyverse)
library(boot)
library(rpart)
library(rpart.plot)
library(randomForest)
library(bestglm)
require(caTools)
library(AMR)
library(ggfortify)
library(pROC)
install.packages("devtools", repo="http://cran.us.r-project.org")
library(devtools)
install_github("vqv/ggbiplot")
set.seed(1)
library(ggbiplot)


rm(list=ls())
importedData <- read.csv(file = "csv_result-Autism-Adult-Data.csv")

################################
# Cleaning
################################
# The data has some spelling errors in the column names, fix these
names(importedData)[names(importedData) == "contry_of_res"] <- "country_of_res"
names(importedData)[names(importedData) == "austim"] <- "autism"

# Remove rows with empty values
cleanData <- importedData[apply(importedData, 1, function(row) all(row != '?')), ]
# Remove age outlier
cleanData <- cleanData[!(cleanData$id %in% c(53)), ]
# Remove unnecessary columns
cleanData <- cleanData[ , !(names(cleanData) %in% c('id',  'used_app_before', 'age_desc', 'relation', '
# Age is a char, let's fix this
cleanData$age <- as.numeric(cleanData$age)
# Outcome is a char, change to a factor
cleanData$Class.ASD <- as.factor(cleanData$Class.ASD)

# Ethnicity has many categories. collapse these to mitigate the curse of dimensionality
```

```
fct_count(cleanData$ethnicity)
cleanData$ethnicity <- fct_collapse(cleanData$ethnicity,
                                    White = c("White-European"),
                                    Black = c("Black", "Turkish", "Middle Eastern "),
                                    Asian = c("Asian", "South Asian"),
                                    Other = c("others", "Others", "Pasifika", "Hispanic", "Latino"),
)
fct_count(cleanData$ethnicity)
f <- factor(cleanData$ethnicity, levels = c("White", "Asian", "Black", "Other"))
cleanData$ethnicity <- fct_relevel(f)
ggplot(data = cleanData, aes(x = ethnicity, fill = Class.ASD)) +
  geom_histogram(stat = "count")

################################
# PCA
################################
pca.data <- cleanData[ , 1:10]
for(i in 1:10) {
  pca.data[, i] <- as.integer(pca.data[, i])
}
pca.out <- prcomp(pca.data, scale = TRUE)
# plot is flipped, so flip it back
pca.out$rotation <- -pca.out$rotation
pca.out$x <- -pca.out$x
biplot(pca.out, scale = 0)
# variance explained by each PC
pca.var <- pca.out$sdev^2
# proportion of variance explained
pve <- pca.var / sum(pca.var)
# plot PVE
par(mfrow = c(1,2))
plot(pve, xlab = "Principal Component", ylab = "Proportion of Variance Explained", ylim = c(0,1), type =
plot(cumsum(pve), xlab = "Principal Component", ylab = "Cumulative Proportion of Variance Explained", yl

groupData <- rep("ASD", length(cleanData$Class.ASD))
groupData[as.integer(cleanData$Class.ASD) == 1] <- "Not ASD"
ggbiplot(pca.out, groups = as.factor(groupData), ellipse = TRUE, circle = FALSE)
################################
# Possible methods: Logistic Regression, Naive Bayes, Classification Tree, Random Forest
################################
n <- nrow(cleanData) -1
index <- sample(1:n, n*0.8, replace=FALSE)
trainDat <- cbind(pca.out$x[, 1:3], cleanData[, c(11:15, 17)])[index, ]
testDat <- cbind(pca.out$x[, 1:3], cleanData[, c(11:15, 17)])[-index, ]

# Cross fold validation
nFolds <- 5
folds <- createFolds(trainDat$Class.ASD, k = nFolds)



par(mfrow = c(1,4))
testResults <- c()
```

```r
##############################
# Logistic Regression
##############################
# Best subset
tempData <- trainDat
tempData$gender <- as.factor(trainDat$gender)
tempData$jundice <- as.factor(trainDat$jundice)
tempData$ethnicity <- as.factor(trainDat$ethnicity)
tempData$autism <- as.factor(trainDat$autism)
tempData$Class.ASD <- as.integer(trainDat$Class.ASD) - 1
best.LR <- bestglm(tempData, family = binomial(), IC="AIC")
best.LR$Subsets

# Testing with only the significant variables
LR.model <- glm(Class.ASD ~ PC1 + PC2 + PC3, family = binomial("logit"), trainDat, control = list(maxit
LR.prob <- predict(LR.model, testDat, type = "response")
LR.pred <- rep(0, dim(testDat)[1])
LR.pred[LR.prob > .5] <- 1
LR.test.table <- table(LR.pred, testDat$Class.ASD)
LR.test.acc <- (LR.test.table[1,1]+LR.test.table[2,2])/sum(LR.test.table)
LR.test.spec <- LR.test.table[1,1]/(LR.test.table[1,1] + LR.test.table[1,2])
LR.test.sens <- LR.test.table[2,2]/(LR.test.table[2,2] + LR.test.table[2,1])
LR.test.acc
LR.test.spec
LR.test.sens

# 5 fold CV
train.acc <- c()
test.acc <- c()
train.acc.temp <- c()
LR.test <- c()
for(i in 1:nFolds) {
  LR.model <- glm(Class.ASD ~ PC1 + PC2 + PC3 + age + jundice, family = binomial("logit"), trainDat[-fo
  trainPred <- predict(LR.model, trainDat[-folds[[i]], ], type = "response")
  LR.pred <- rep(0, dim(trainDat[-folds[[i]], ])[1])
  LR.pred[trainPred > .5] <- 1
  trainTable <- table(LR.pred, trainDat[-folds[[i]], ]$Class.ASD)


  testPred <- predict(LR.model, trainDat[folds[[i]], ], type = "response")
  LR.pred <- rep(0, dim(trainDat[folds[[i]], ])[1])
  LR.pred[testPred > .5] <- 1
  testTable <- table(LR.pred, trainDat[folds[[i]], ]$Class.ASD)

  train.acc.temp <- c(train.acc.temp, (trainTable[1,1]+trainTable[2,2])/sum(trainTable))
  LR.test <- c(LR.test, (testTable[1,1]+testTable[2,2])/sum(testTable))
}
train.acc <- cbind(train.acc, train.acc.temp)
test.acc <- cbind(test.acc, LR.test)
mean(LR.test)
testResults <- rbind(testResults, cbind(LR.test, rep("Logistic Regression", 5)))

# Testing with the testing set
```

```
LR.model <- glm(Class.ASD ~ PC1 + PC2 + PC3 + age + jundice, family = binomial("logit"), trainDat, conti
LR.prob <- predict(LR.model, testDat, type = "response")
LR.pred <- rep(0, dim(testDat)[1])
LR.pred[LR.prob > .5] <- 1
LR.test.table <- table(LR.pred, testDat$Class.ASD)
LR.test.acc <- (LR.test.table[1,1]+LR.test.table[2,2])/sum(LR.test.table)
LR.test.spec <- LR.test.table[1,1]/(LR.test.table[1,1] + LR.test.table[1,2])
LR.test.sens <- LR.test.table[2,2]/(LR.test.table[2,2] + LR.test.table[2,1])
LR.test.acc
LR.test.spec
LR.test.sens



##############################
# Naive Bayes
##############################
train.acc <- c()
test.acc <- c()
train.acc.temp <- c()
NB.test <- c()
for(i in 1:nFolds) {
  NB.model <- naiveBayes(Class.ASD ~ PC1 + PC2 + PC3 + age + jundice, data=trainDat[-folds[[i]], ])
  trainPred <- predict(NB.model, trainDat[-folds[[i]], ], type = "class")
  trainTable <- table(trainPred, trainDat[-folds[[i]], ]$Class.ASD)


  testPred <- predict(NB.model, trainDat[folds[[i]], ], type = "class")
  testTable <- table(testPred, trainDat[folds[[i]], ]$Class.ASD)

  train.acc.temp <- c(train.acc.temp, (trainTable[1,1]+trainTable[2,2])/sum(trainTable))
  NB.test <- c(NB.test, (testTable[1,1]+testTable[2,2])/sum(testTable))
}
train.acc <- cbind(train.acc, train.acc.temp)
test.acc <- cbind(test.acc, NB.test)
mean(NB.test)
testResults <- rbind(testResults, cbind(NB.test, rep("Naive Bayes", 5)))

NB.model <- naiveBayes(Class.ASD ~ PC1 + PC2 + PC3 + age + jundice, data=trainDat)
NB.pred <- predict(NB.model, testDat, type = "class")
NB.prob <- predict(NB.model, testDat, type = "raw")
NB.test.table <- table(NB.pred, testDat$Class.ASD)
NB.test.acc <- (NB.test.table[1,1]+NB.test.table[2,2])/sum(NB.test.table)
NB.test.spec <- NB.test.table[1,1]/(NB.test.table[1,1] + NB.test.table[1,2])
NB.test.sens <- NB.test.table[2,2]/(NB.test.table[2,2] + NB.test.table[2,1])
NB.test.acc
NB.test.spec
NB.test.sens



##############################
# Classification Tree
```

```
##############################
train.acc <- c()
test.acc <- c()
train.acc.temp <- c()
CT.test <- c()
for(i in 1:nFolds) {
  tree.model <- rpart(Class.ASD ~ ., data = trainDat, method = "class")
  trainPred <- predict(tree.model, trainDat[-folds[[i]], ], type = "class")
  trainTable <- table(trainPred, trainDat[-folds[[i]], ]$Class.ASD)


  testPred <- predict(tree.model, trainDat[folds[[i]], ], type = "class")
  testTable <- table(testPred, trainDat[folds[[i]], ]$Class.ASD)

  train.acc.temp <- c(train.acc.temp, (trainTable[1,1]+trainTable[2,2])/sum(trainTable))
  CT.test <- c(CT.test, (testTable[1,1]+testTable[2,2])/sum(testTable))
}
train.acc <- cbind(train.acc, train.acc.temp)
test.acc <- cbind(test.acc, CT.test)
mean(CT.test)
testResults <- rbind(testResults, cbind(CT.test, rep("Classification Tree", 5)))

tree.model <- rpart(Class.ASD ~ ., data = trainDat, method = "class")
tree.pred <- predict(tree.model, testDat, type = "class")
tree.prob <- predict(tree.model, testDat, type = "prob")
tree.test.table <- table(tree.pred, testDat$Class.ASD)
tree.test.acc <- (tree.test.table[1,1]+tree.test.table[2,2])/sum(tree.test.table)
tree.test.spec <- tree.test.table[1,1]/(tree.test.table[1,1] + tree.test.table[1,2])
tree.test.sens <- tree.test.table[2,2]/(tree.test.table[2,2] + tree.test.table[2,1])
tree.test.acc
tree.test.spec
tree.test.sens
rpart.plot(tree.model, extra = 106)


##############################
# Random Forest
##############################
train.acc <- c()
test.acc <- c()
train.acc.temp <- c()
RF.test <- c()
for(i in 1:nFolds) {
  RF.model <- randomForest(Class.ASD ~ ., data=trainDat[-folds[[i]], ], proximity = TRUE)
  trainPred <- predict(RF.model, trainDat[-folds[[i]], ], type = "class")
  trainTable <- table(trainPred, trainDat[-folds[[i]], ]$Class.ASD)


  testPred <- predict(RF.model, trainDat[folds[[i]], ], type = "class")
  testTable <- table(testPred, trainDat[folds[[i]], ]$Class.ASD)

  train.acc.temp <- c(train.acc.temp, (trainTable[1,1]+trainTable[2,2])/sum(trainTable))
  RF.test <- c(RF.test, (testTable[1,1]+testTable[2,2])/sum(testTable))
```

```
}
train.acc <- cbind(train.acc, train.acc.temp)
test.acc <- cbind(test.acc, RF.test)
mean(RF.test)
testResults <- rbind(testResults, cbind(RF.test, rep("Random Forest", 5)))

RF.model <- randomForest(Class.ASD ~ ., data=trainDat, method = "class")
RF.pred <- predict(RF.model, data = testDat, method = "class")
RF.prob <- predict(RF.model, testDat, type = "prob")
RF.test.table <- table(RF.pred, trainDat$Class.ASD)
RF.test.acc <- (RF.test.table[1,1]+RF.test.table[2,2])/sum(RF.test.table)
RF.test.spec <- RF.test.table[1,1]/(RF.test.table[1,1] + RF.test.table[1,2])
RF.test.sens <- RF.test.table[2,2]/(RF.test.table[2,2] + RF.test.table[2,1])
RF.test.acc
RF.test.spec
RF.test.sens

par(mfrow = c(1,1))
colnames(testResults) <- c("Accuracy", "Model")
testResults <- as.data.frame(testResults)
testResults$Accuracy <- as.numeric(testResults$Accuracy)
ggplot(testResults, aes(x = Model, y = Accuracy)) +
  geom_boxplot() +
  xlab("Model") +
  ylab("5-fold Cross Validated Accuracy") +
  ggtitle("5-fold CV Accuracy Per Model", subtitle = waiver()) +
  theme(plot.title = element_text(hjust = 0.5))

plot(roc(Class.ASD ~ LR.prob, data = testDat), col = "green")
plot(roc(Class.ASD ~ NB.prob[, 1], data = testDat), add = TRUE, col = "blue")
plot(roc(Class.ASD ~ RF.prob[, 1], data = testDat), add = TRUE, col = "orange")
plot(roc(Class.ASD ~ tree.prob[, 1], data = testDat), add = TRUE, col = "purple")
legend("right", c("Logistic Regression", "Naive Bayes", "Random Forest", "Classification Tree"), lty=1,
       col = c("green", "blue", "orange", "purple"), bty="n", inset=c(0,150))
```