PM 591 – Machine Learning for the Health Sciences

Assignment 2

Analysis

1. Brain weight data.

a. Using the function `KNN.reg` in the `FNN` package construct predictive models for brain weight using KNN regression for $K = 1, .., 15$ on a training set (use the exact same training/validation split–same seed and same split percentages–you used for linear regression). Hint: use a for loop to iterate over K.

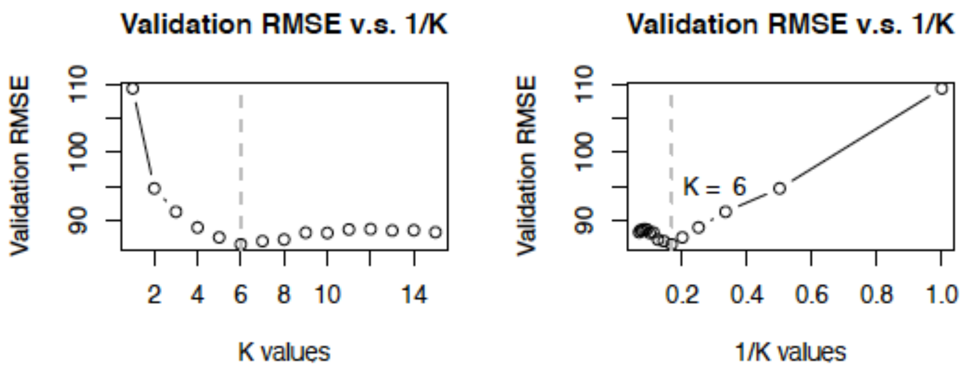
```r
suppressMessages(library(dplyr))
library(FNN)

set.seed(2022)
index <- sample(1:nrow(brain), floor(0.7 * nrow(brain)))
train <- brain[index, ]
validation <- brain[-index, ]
rmse <- function(observed, predicted) sqrt(mean((observed - predicted)^2))
upper <- 15
rmse_val_KNN <- rep(NA, upper)
for (k in 1:upper) {
    fit_KNN <- knn.reg(train = train[, -4, drop = FALSE], test = validation[,
        -4, drop = FALSE], y = train$Brain.weight, k = k)
    rmse_val_KNN[k] <- rmse(validation$Brain.weight, fit_KNN$pred)
}
```

b. Plot the validation RMSE as a function of K and select the best K.

```r
par(mfrow = c(1, 2))
plot(1:upper, rmse_val_KNN, type = "b", main = "Validation RMSE v.s. 1/K",
    xlab = "K values", ylab = "Validation RMSE")
abline(v = which.min(rmse_val_KNN), col = "grey", lwd = 2, lty = 2)

plot(1/(1:upper), rmse_val_KNN, type = "b", main = "Validation RMSE v.s. 1/K",
    xlab = "1/K values", ylab = "Validation RMSE")
text(0.3, 95, paste("K = ", which.min(rmse_val_KNN)))
abline(v = 1/(which.min(rmse_val_KNN)), col = "grey", lwd = 2,
    lty = 2)
```



```
print(paste("The minimum validation RMSE", round(min(rmse_val_KNN),
    2), "is obtained when K =", which.min(rmse_val_KNN)))
```

```
## [1] "The minimum validation RMSE 86.49 is obtained when K = 6"
```

c. Using the validation RMSE compare to the best linear regression model from homework 1. Is there an improvement in prediction performance? Interpret your results based on the bias-variance tradeoff.

- When performed on the exact same training and validation datasets, the validation RMSE (86.49) of KNN regression when K = 6 is still higher than the validation RMSE (74.99) of the best linear regression model fitted by sex, age, and head size.
- Using the KNN regression will result in the low bias but high variation in prediction, which explains the deterioration in the prediction performance.

2. The goal of this exercise is to fit several LDA classifiers and perform model selection using the Ischemic Stroke data set. For your convenience the code to pre-process/transform the data is provided below.

Dataset notes: According to the Mayo Clinic, "ischemic stroke occurs when a blood clot blocks or narrows an artery leading to the brain. A blood clot often forms in arteries damaged by the buildup of plaques (atherosclerosis). It can occur in the carotid artery of the neck as well as other arteries. This is the most common type of stroke." (https://www.mayoclinic.org/diseases-conditions/stroke/symptoms-causes/syc-20350113#:~:text=Ischemic%20stroke%20occurs%20when%20a,most%20common%20type%20of%20stroke.)

a. When splitting the data into training, validation and testing or classification problems it is important to ensure each set retains approximately the same proportion of positive and negative examples as the full data. Split the data into training (70%), and validation (30%), but keeping the proportion of positive and negative examples roughly the same in the training and validation sets. This can be accomplished by sampling in a stratified manner, i.e. sampling 70/30 within the negative and the positive classes. Use the code below to perform stratified splitting.

```
# Code for reading in stroke data and converting categorical variables to factors
# To run chunk set options above to eval=TRUE

setwd("/Users/carrie/Downloads") #replace "/Users/yourpath/" with the path to the folder where 'stroke.csv' file
 is stored
stroke = read.csv("stroke (1).csv" )
str(stroke)
```

```
## 'data.frame':    126 obs. of  30 variables:
##  $ X                        : int  1 2 4 5 7 8 9 10 14 15 ...
##  $ Stroke                   : chr  "N" "N" "N" "Y" ...
##  $ NASCET                   : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ CALCVol                  : num  235.3 31.4 113.4 780.8 84.1 ...
##  $ CALCVolProp              : num  0.0704 0.0162 0.0381 0.2134 0.0414 ...
##  $ MATXVol                  : num  3157 3033 3835 3519 2990 ...
##  $ MATXVolProp              : num  0.76 0.813 0.783 0.761 0.75 ...
##  $ LRNCVol                  : num  225 369 321 141 293 ...
##  $ LRNCVolProp              : num  0.0911 0.134 0.083 0.0321 0.0754 ...
##  $ MaxCALCArea              : num  12.35 7.13 16.29 63.35 17.58 ...
##  $ MaxCALCAreaProp          : num  0.366 0.211 0.409 0.576 0.322 ...
##  $ MaxDilationByArea        : num  521 91.7 271 2270.5 95.2 ...
##  $ MaxMATXArea              : num  71.2 27.2 38.1 341.1 56.6 ...
##  $ MaxMATXAreaProp          : num  0.952 0.946 0.946 0.969 0.921 ...
##  $ MaxLRNCArea              : num  21.69 6.43 5.71 6.05 7.21 ...
##  $ MaxLRNCAreaProp          : num  0.43 0.282 0.155 0.187 0.217 ...
##  $ MaxMaxWallThickness      : num  2.41 2.54 3.71 6.12 3.98 ...
##  $ MaxRemodelingRatio       : num  5.7 1.74 2.83 15.65 1.91 ...
##  $ MaxStenosisByArea        : num  19 30.2 33.9 34.3 36.6 ...
##  $ MaxWallArea              : num  106.2 33.4 55.3 426.5 59.8 ...
##  $ WallVol                  : num  4192 3917 4935 4910 4045 ...
##  $ MaxStenosisByDiameter    : num  10.5 18.6 19.7 20.3 49.3 ...
##  $ age                      : int  72 76 72 61 65 64 82 83 85 56 ...
##  $ sex                      : int  1 1 0 1 1 1 0 0 0 0 ...
##  $ SmokingHistory           : int  1 1 0 1 0 1 1 0 1 1 ...
##  $ AtrialFibrillation       : int  0 0 0 0 0 0 0 1 0 0 ...
##  $ CoronaryArteryDisease    : int  0 0 0 0 0 1 1 1 1 0 ...
##  $ DiabetesHistory          : int  0 1 0 1 0 0 0 0 0 0 ...
##  $ HypercholesterolemiaHistory: int  0 1 0 1 0 1 0 0 1 0 ...
##  $ HypertensionHistory      : int  1 1 0 1 1 1 1 1 1 0 ...
```

```
stroke$Stroke                         <- factor(stroke$Stroke, levels=c('N', 'Y'), labels=c("No", "Yes"))
stroke$NASCET                         <- factor(stroke$NASCET, levels=0:1, labels=c("No", "Yes"))
stroke$sex                            <- factor(stroke$sex, levels=0:1, labels=c("Female", "Male"))
stroke$SmokingHistory                 <- factor(stroke$SmokingHistory, levels=0:1, labels=c("No", "Yes"))
stroke$AtrialFibrillation             <- factor(stroke$AtrialFibrillation, levels=0:1, labels=c("No", "Yes"))
stroke$CoronaryArteryDisease          <- factor(stroke$CoronaryArteryDisease, levels=0:1, labels=c("No", "Yes"))
stroke$DiabetesHistory                <- factor(stroke$DiabetesHistory, levels=0:1, labels=c("No", "Yes"))
stroke$HypercholesterolemiaHistory <- factor(stroke$HypercholesterolemiaHistory, levels=0:1, labels=c("No", "Yes"
))
stroke$HypertensionHistory            <- factor(stroke$HypertensionHistory, levels=0:1, labels=c("No", "Yes"))
```

```
# Code for splitting data into trainin and validation
# To run chunk set options above to eval=TRUE
set.seed(303)
n = nrow(stroke)
positives = (1:n)[stroke$Stroke=='Yes']
negatives = (1:n)[stroke$Stroke=='No']

positives_train = sample(positives, floor(0.7*length(positives)))
positives_val = setdiff(positives, positives_train)

negatives_train = sample(negatives, floor(0.7*length(negatives)))
negatives_val = setdiff(negatives, negatives_train)

stroke_train = stroke[c(positives_train, negatives_train), ]
stroke_val = stroke[c(positives_val, negatives_val), ]

ntrain = nrow(stroke_train); nval=nrow(stroke_val)

table(stroke_train$Stroke)
```

```
##
##  No Yes
##  43  44
```

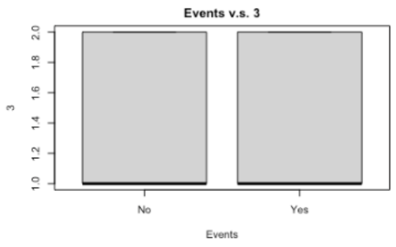
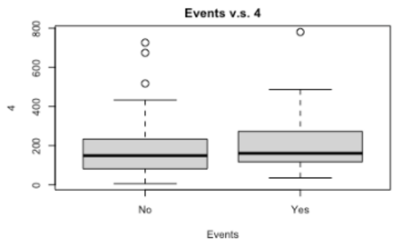
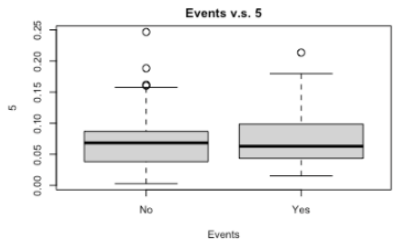
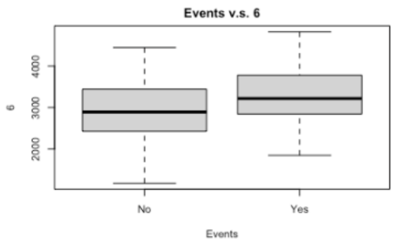
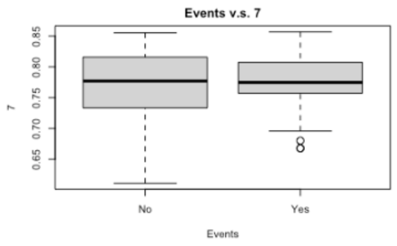
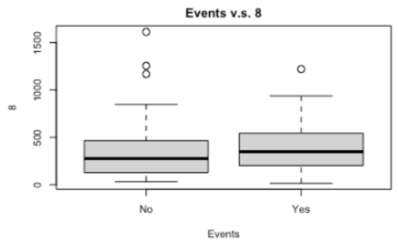
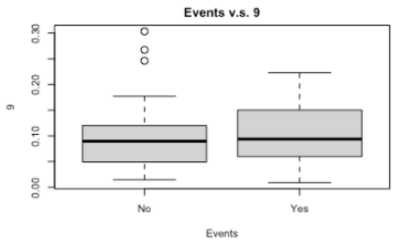
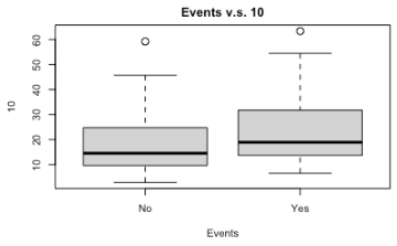
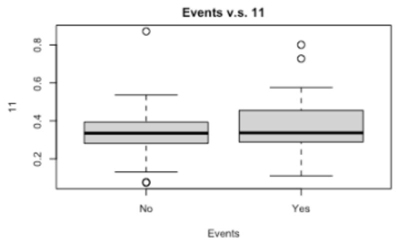
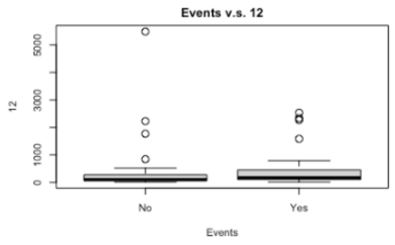
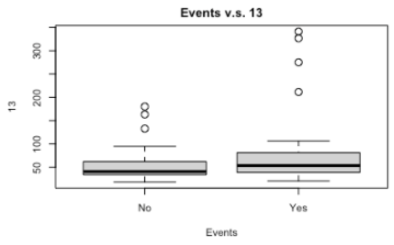
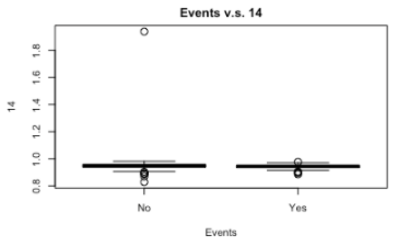
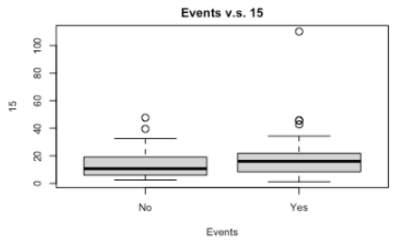
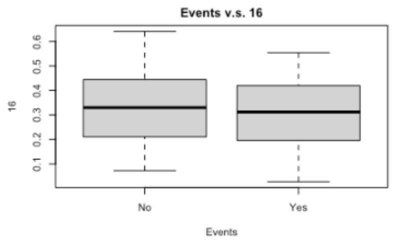
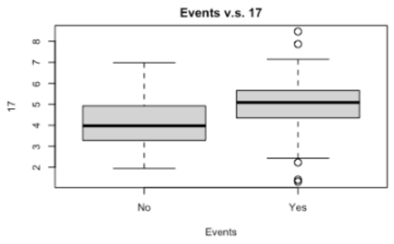
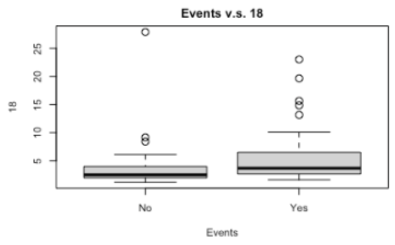
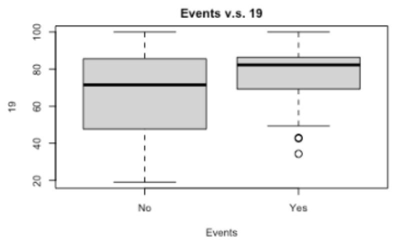
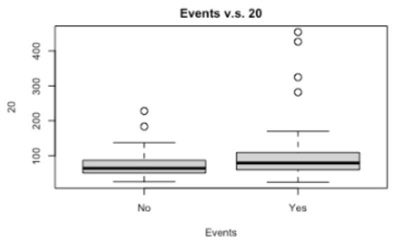
```
table(stroke_val$Stroke)
```

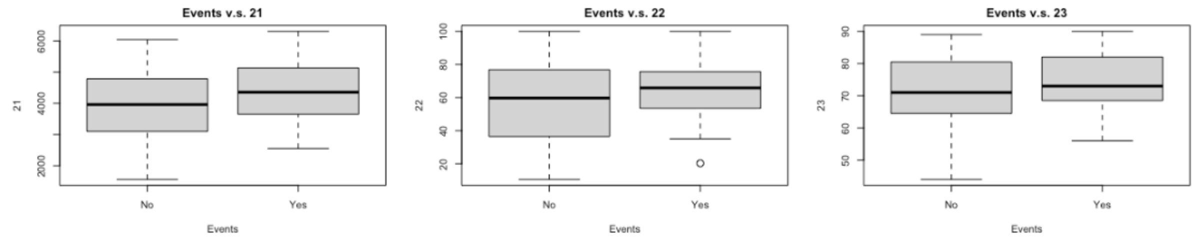
```
##
##  No Yes
##  19  20
```

Note: Because of the moderate sample size we will not have a separate test set – we will learn later in the course about cross-validation, which will allow us to split the data into training and testing only and still perform model selection.

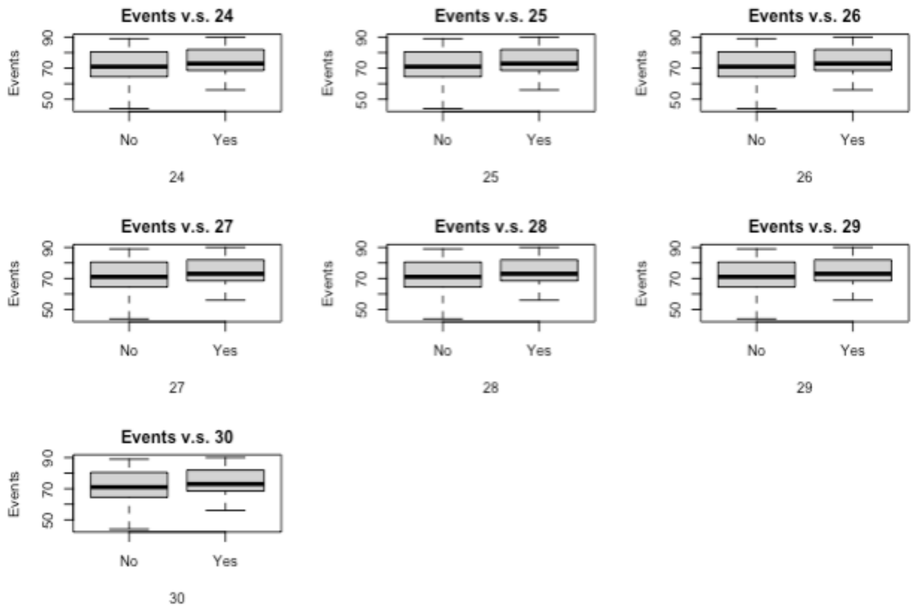
a. Read in the data and convert all categorical variables to factors (use code below). Split the data into a training (70%) and validation (30%) using stratified dampling (use code below). Using the training data, graphically assess each of the predictors using a boxplot for quantitative predictors and a mosaic plot for a categorical predictors. Note: you can use plot to get these graphs. Use for example `boxplot(your_predictor ~ Stroke, data=stroke_train)` to get a boxplot for a quantitative predictor and `plot(Stroke, your_predictor, data=stroke_train)` for a categorical predictor to get a mosaic plot. Visually determine the 3 most most predictive **imaging features**, i.e. the imaging features that best separate the stroke=YES vs. stroke='No' classes. (This is an informal procedure since a visual assessment is inherently subjective, in a future class we will learn how to do feature selection in a systematic way).

```
#boxplot for continuous variables
par(mfrow=c(3,3))
cont=c(3:23)
for(i in cont){
boxplot(stroke_train[,i] ~ stroke_train$Stroke, ylab=i, cex=1.5, xlab="Events", main=paste("Events
v.s.", i))
}
```


Events v.s. 3
3
No
Yes
Events

Events v.s. 4
4
No
Yes
Events

Events v.s. 5
5
No
Yes
Events

Events v.s. 6
6
No
Yes
Events

Events v.s. 7
7
No
Yes
Events

Events v.s. 8
8
No
Yes
Events

Events v.s. 9
9
No
Yes
Events

Events v.s. 10
10
No
Yes
Events

Events v.s. 11
11
No
Yes
Events

Events v.s. 12
12
No
Yes
Events

Events v.s. 13
13
No
Yes
Events

Events v.s. 14
14
No
Yes
Events

Events v.s. 15
15
No
Yes
Events

Events v.s. 16
16
No
Yes
Events

Events v.s. 17
17
No
Yes
Events

Events v.s. 18
18
No
Yes
Events

Events v.s. 19
19
No
Yes
Events

Events v.s. 20
20
No
Yes
Events



```
#boxplot for categorical variables
par(mfrow=c(3,3))
cat=c(24:30)
for (j in cat){
boxplot(stroke_train[,i] ~ stroke_train$Stroke, ylab="Events", xlab=j, main=paste("Events v.s.",
j))
}
```



b. Build LDA classifiers of increasing complexity by including: i) age, sex, and smoking history; ii) all the previous features + the clinical variables AtrialFibrillation, CoronaryArteryDisease, DiabetesHistory, HypercholesterolemiaHistory, and HypertensionHistory; iii) all the previous features + the most predictive imaging feature based on part b; and iv) all the previous features + the next 2 most predictive imaging features.

c. Write an R function `classificationError` to compute the overall misclassification error, specificity, and sensitivity of a classifier. The function should take a confusion matrix as its input (which you can create using `table` as shown in the lecture) and return a vector with the overall misclassication error, specificity and sensitivity. (Hint: separately compute the three quantities `error`, `spec`, and `sens` inside the body of the function and then put them together in a vector using `c(error=error, sensitivity=sens, specificity=spec)` in the last line of the body of the function before the closing `}` – the last line is by default what a function returns. The returned object can be any R object including a siggle number, a vector, a data.frame or even another function!)

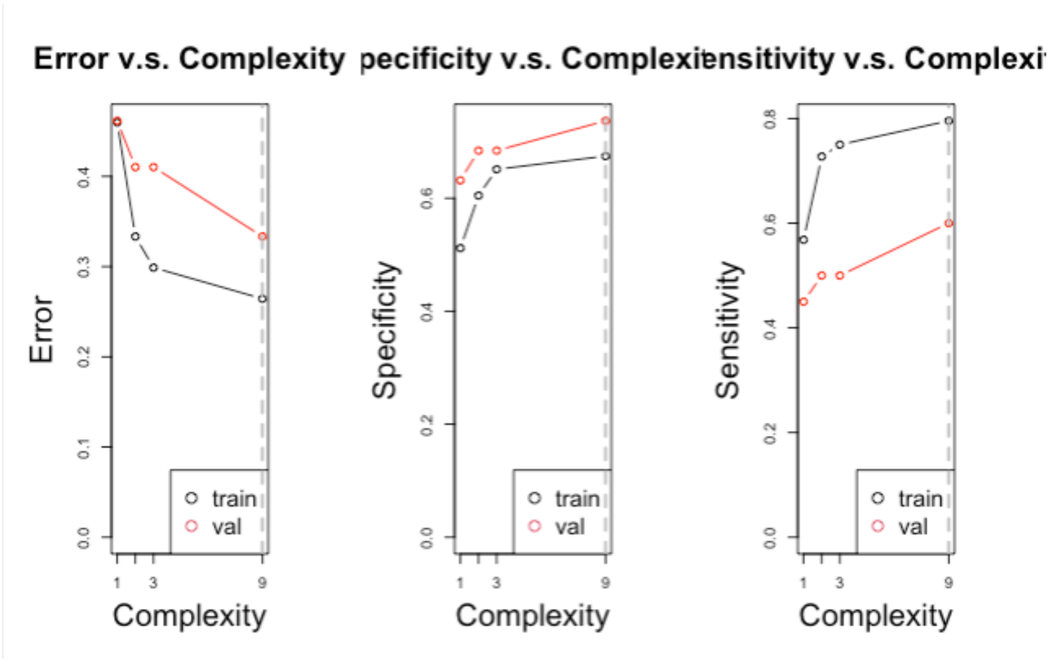
d. Compute the training and test errors for each of the classifiers in e. Which classifier would you choose?

e. Plot in the same graph the training and test misclassification error as a function of classifier complexity. Comment/interpret the plots.

```
suppressMessages(require(MASS))
# 1)
train1_lda <- lda(Stroke ~ age+sex+SmokingHistory, data = stroke_train)
# 2)
train2_lda <- lda(Stroke ~ age+sex+SmokingHistory+AtrialFibrillation+CoronaryArteryDisease
+DiabetesHistory+HypercholesterolemiaHistory+HypertensionHistory, data = stroke_train)
# 3)
train3_lda <- lda(Stroke ~ age+sex+SmokingHistory+AtrialFibrillation+CoronaryArteryDisease
+DiabetesHistory+HypercholesterolemiaHistory+HypertensionHistory+MATXVol, data = stroke_train)
# 4)
train9_lda <- lda(Stroke ~ age+sex+SmokingHistory+AtrialFibrillation+CoronaryArteryDisease
+DiabetesHistory+HypercholesterolemiaHistory+HypertensionHistory+MaxStenosisByArea+MATXVol
+MaxCALCArea, data = stroke_train)


classificationError <- function(model, data) {
       pred_lda = predict(model, data)
       confMatrix = table(true = data$Stroke, predicted = pred_lda$class)
       return(c(Error = (confMatrix[1, 2] + confMatrix[2, 1])/sum(confMatrix),
       Specificity = confMatrix[1, 1]/(confMatrix[1, 2] + confMatrix[1,
       1]), Sensitivity = confMatrix[2, 2]/(confMatrix[2,
       1] + confMatrix[2, 2])))
}


PlotYX <- function(y, method) {
train <- c(classificationError(get(paste("train1_", method,
sep = "")), stroke_train)[y], classificationError(get(paste("train2_",
method, sep = "")), stroke_train)[y], classificationError(get(paste("train3_",
method, sep = "")), stroke_train)[y], classificationError(get(paste("train9_",
method, sep = "")), stroke_train)[y])
val <- c(classificationError(get(paste("train1_", method,
sep = "")), stroke_val)[y], classificationError(get(paste("train2_",
method, sep = "")), stroke_val)[y], classificationError(get(paste("train3_",
method, sep = "")), stroke_val)[y], classificationError(get(paste("train9_",
method, sep = "")), stroke_val)[y])
Complexity <- c(1, 2, 3, 9)
plot(Complexity, train, type = "b", ylim = c(0, max(c(train,
val))), main = paste(y, "v.s. Complexity"), cex.main = 2,
cex.lab = 2, ylab = paste(y), xaxt = "n")
axis(side = 1, at = Complexity, labels = Complexity)
lines(Complexity, val, type = "b", col = "red")
legend("bottomright", c("train", "val"), pch = 1, col = 1:2,
cex = 1.5)
abline(v = ifelse(y == "Error", Complexity[which.min(val)],
Complexity[which.max(val)]), col = "grey", lwd = 2,
lty = 2)
}
```



As the model complexity increased from model 1 to model 4, both the training and test errors decreased. We would choose model 4 with 11 predictors as the best classifier due to the lowest misclassification errors. It is important to select variables with more discriminant information rather than noise. In these classification problems we have different features: some of the discriminant variables contribute to the classification task; some the redundant variables that correlated with the discriminant variables; and the noise variables which bring no additional information when adding into the models. Thus, variable selection in discriminant analysis is of great importance for getting a reliable and parsimonious classifier. In addition, we could evaluate the sensitivity and specificity of different models. Those are also measures of the accuracy of the classification task. There is a trade-off between sensitivity and specificity that is dependent on the cut-off level chosen for a positive threshold.