ST4061 – Computer Intensive Statistical Analytics II
ST6041 – Machine Learning and Statistical Analytics II
2022-23
In-class test 1

NAME AND SURNAME: Zach Buckley

STUDENT NUMBER: 118349121

PROGRAM: Mathematical Modelling and Machine Learning

INTRUCTIONS

- Provide your answers in this document, after each question.
- Paste the R code you used for each question item.
- **Save your files regularly.**

Question 1

Load the following libraries and dataset into your R session as follows:

```
library(MASS)
library(caret)
ca.train = read.csv("ca_train.csv",stringsAsFactors=TRUE)
ca.test = read.csv("ca_test.csv",stringsAsFactors=TRUE)
```

This dataset contains a sample of records from 500 young soccer players after their one-year stay at a soccer academy, with variables:

- y:	gets hired by a club on a trial contract as a professional player
- academy:	whether the player received academic training previously
- entry:	player score sheet after entry test
- exit:	player score sheet after exit test

Your goal is to build a prediction model in order to predict variable y.

(1) Fit an LDA model using all predictors to the training data (ca.train). Provide the corresponding confusion matrix obtained for the test data (ca.test).

Your answer:

```
            Reference
Prediction  No     Yes
     No     1909   64
     Yes     3     13
```

R code for (1):

```
#Question 1
#(1)
#View the data
par(mfrow=c(1,1))
plot(ca.train$entry,ca.train$exit,
pch=c(19,17)[ca.train$academy],      col=c(1,2)[ca.train$y],
cex=1, xlab = "Entry", ylab = "Exit")
legend(185,720,col=c(1,2),legend=levels(ca.train$y),pch=20,
bty='n', title = "Hired")
legend(150,720,pch=c(19,17),legend=levels(ca.train$academy)
, bty='n', title = "Academy")

#Fit LDA model
lda.o = lda(y~., data=ca.train)
lda.o

#Confusion matrix
lda.p = predict(lda.o, newdata=ca.test)$class
```

```
caret::confusionMatrix(lda.p,    ca.test$y,    positive    =
"Yes")$table
```

(2) Fit a QDA model using all predictors to the training data (ca.train) . Provide the corresponding confusion matrix obtained for the test data (ca.test).

Your answer:

```
              Reference
Prediction    No     Yes
    No        1907  59
    Yes        5     18
```

R code for (2):

```
#(2)
#Fit QDA model
qda.o = qda(y~., data=ca.train)
qda.o

#Confusion matrix
qda.p = predict(qda.o, newdata=ca.test)$class
caret::confusionMatrix(qda.p,    ca.test$y,    positive    =
"Yes")$table
```

(3) Evaluate and quote the specificities of both models. Comment on the values you obtain.

Your answer:
LDA specificity: 0.998431
QDA specificity: 0.9973849

Both models have a very high specificity value with the LDA model having a slightly higher value. This means this model gives very few false positive results - i.e. the models rarely predicts that a player will be hired when in fact they don't get hired. It is very good at correctly identifying cases where a player does not get hired. High specificity may come at a cost of low sensitivity

R code for (3):

```
#(3)
#Specificity  =  (True  negatives)/(True  negatives  +  False
positives)
#LDA specificity
caret::confusionMatrix(lda.p,    ca.test$y,    positive    =
"Yes")$byClass[2]
```

```
#0.998431

#QDA specificity
caret::confusionMatrix(qda.p,    ca.test$y,    positive    =
"Yes")$byClass[2]
#0.9973849
```

(4) Explain the difference in specificities, using relevant R output.

Your answer:
LDA sensitivity: 0.1688312
QDA sensitivity: 0.2337662

LDA may give a slightly higher specificity value due to having lower sensitivity than the QDA model. Lower sensitivity means the model will give many false negative results - i.e. the model will often predict a player won't get hired when in fact they do. Having a higher specificity often comes at a cost of lowering the sensitivity

The difference may also be due to the underlying structure of the data. LDA assumes that the covariance matrices for each predictor is the same while QDA doesn't.

```
#LDA Sensitivity
caret::confusionMatrix(lda.p,    ca.test$y,    positive    =
"Yes")$byClass[1:2]
#0.1688312
#QDA sensitivity
caret::confusionMatrix(qda.p,    ca.test$y,    positive    =
"Yes")$byClass[1:2]
#0.2337662
```

Question 2

No code is required for this question. A student is considering the dataset depicted in Figure 1, comprising of two predictors X_1 and X_2, and one categorical dependent variable Y with 2 categories.

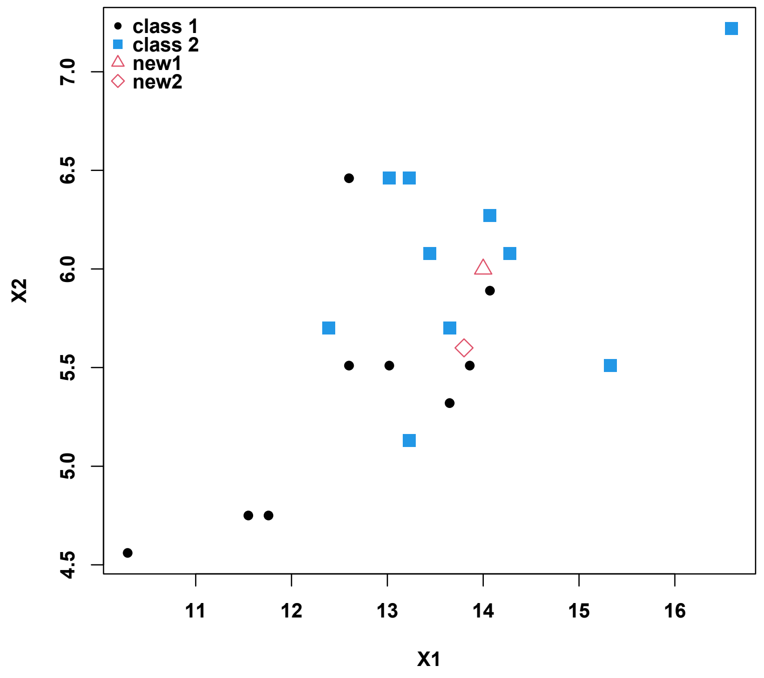

Figure 1 - Dataset for Question 2, with 2 test points new_1 and new_2.

The student is considering using a kNN classifier with k=3 to generate predictions for test points new_1 and new_2 in Figure 1. Indicate in the table below what these predictions would be.

Your answer:

kNN with k=3	**Prediction**
Test point new_1	Class 2
Test point new2	Class 1

Question 3

For this question you are required to use the following packages:

```
library(ISLR) # for the data
library(gbm)
library(randomForest)
```

Do **not** use the **caret** package, or any other package except for the above packages, for this question.

Start by creating the following dataset:

```
x.train = Khan$xtrain
x.test = Khan$xtest
y.train = as.factor(Khan$ytrain)
y.test = as.factor(Khan$ytest)
```

a) Provide a tabular summary of the distribution of values in the train and test samples of observations **y.train** and **y.test**. Comment briefly on these distributions.

b) Is this a regression or a classification problem? Justify your answer.

c) Fit a random forest to the training data defined above, using default hyperparameter values from the randomForest package. Quote the corresponding confusion matrix. Run the instruction set.seed(4061) before you run the R code for this question.

d) Generate predictions for the test data from the random forest model fit in (c) and quote the test set prediction accuracy.

e) Name the features in this dataset whose variable importance is strictly greater than 0.4, according to the fit from (c).

f) Provide an interpretation for the measure of variable importance used in (e).

g) Fit a gradient boosting model to the training data, using default hyperparameter values from the gbm package. Run the instruction set.seed(4061) before you run the R code for this question. Generate predictions for the test set from the GBM fit from (g) (do not provide these predictions in your answer) and quote the corresponding prediction accuracy.

Your answer:

Question item	**Answer**
(a)	y.train 1 2 3 4 8 23 12 20 y.test 1 2 3 4 3 6 6 5 The test and training data appear to have similar distributions. This is important as it ensures that the model's performance on the test data is a good representation of its performance on training data. Both distributions are also not sparse. i.e. several samples in each level.
(b)	This is a classification problem. The target variable (y) can take a number of fixed classes: 1, 2, 3 and 4. The goal of regression is to predict a continuous value while the goal of classification is to predict a categorical value.

(c)	`           y.train`
	`rf.yhat   1  2  3  4`
	`        1  8  0  0  0`
	`        2  0 23  0  0`
	`        3  0  0 12  0`
	`        4  0  0  0 20`

(d)	95% accuracy
(e)	V509, V545, V566, V742, V1003, V1319, V1389, V1708, V1954, V2046, V2050
(f)	The Gini index was used as a measure of variable importance. It is a measure of impurity in a tree. The Gini index of a node is the probability of incorrectly classifying a random element in the node. Variables that result in the largest decrease of Gini index are considered more important.
(g)	100% accuracy

R code for Question 3

```
#Check dimensions
dim(x.train)
length(y.train)

#(a)
#Table for y.train
table(y.train)
#Frequency
table(y.train)/length(y.train)

#Table for y.test
table(y.test)
#Frequency
table(y.test)/length(y.test)


#(b)This is a classification problem. The target variable (y) can
 take a number of fixed classes
#1, 2, 3, 4. The goal of regression is to predict a continuous
 value and the goal of
#classification is to predict a categorical value.

#(c)
set.seed(4061)
#Grow random forest
rf.out = randomForest(y.train~., data= x.train)
#Find models predictions for training data
rf.yhat = predict(rf.out,  x.train, type="class")
#Generate confusion matrix
(tb.rf = table(rf.yhat, y.train))

#(d)
#Generate predictions for test set
rf.pred = predict(rf.out,  x.test, type="class")
#Check confusion matrix
(tb.rf = table(rf.pred, y.test))

#Find prediction accuracy
sum(diag(tb.rf)) / sum(tb.rf)
#95% accuracy

#(e)
#Plot importance
par(mfrow=c(1,1))
varImpPlot(rf.out, pch=15, main="Variable Importance Plot")
abline(v = 0.4,col = "blue")

#Find importance
imp = rf.out$importance
idx <- which(imp > 0.4) # row numbers
cbind(idx,imp[idx])
```

```
#509  545  566  742 1003 1319 1389 1708 1954 2046 2050

#(f) #The Gini index was used as a measure of variable
 importance. It is a measure of impurity in a tree.
# The Gini index of a node is the probability of incorrectly
 classifying a random element in the node.
# Variables that result in the largest decrease of Gini index are
 considered more important.

#(g)
set.seed(4061)
#Create data frames
CS.x.train = data.frame(x.train)
CS.x.test = data.frame(x.test)

#Create GB model
gbm.out = gbm(y.train~.,data = CS.x.train)

#Find predictions for test set
gbm.p = predict(gbm.out, newdata=CS.x.test,type = "response")

#Find class predictions by taking the highest probability
class_names = colnames(gbm.p)[apply(gbm.p, 1, which.max)]
result = data.frame(y.test, class_names)

#Create confusion matrix
(tb.gbm= table(result))
#Find accuracy
(acc.gbm = sum(diag(tb.gbm)) / sum(tb.gbm))
#100% accuracy on test data
```