Stat 437 HW4

Your Name (Your student ID)

##   
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':  
##   
## filter, lag

## The following objects are masked from 'package:base':  
##   
## intersect, setdiff, setequal, union

# Conceptual exercises: I (Bayes classifier)

. This exercise is on Bayes theorem and Bayes classifier.

1.1) State clearly the definition of the 0-1 loss function. Can this function be used in multi-class classification problems?

The 0-1 loss function is the loss of L is 0 if an observation is classified right, otherwise L is 1. The 0-1 loss can be used in multi-class classification problems.

1.2) Let be the random variable for the class label of a random vector , such that where is the number of classes. Let be the estimated class label for . Given the prior on Class and the conditional density of when it comes from Class . Provide the formula to obtain the posterior , which is essentially the Bayes theorem. What is the Bayes classifier and how does it classify a new observation from ? Is the decision boundary of the Bayes classifier linear or quadratic in ? Explain (but do not have to mathematically prove) why the Bayes classifier minimizes the expected 0-1 loss. Note the a proof of the fact that the Bayes classifier minimizes the expected 0-1 loss is given in “LectureNotes4\_notes.pdf”. You should not copy and paste the proof. Instead, please provide the explanation based on your understanding of the proof.

Posterior:

Bayes classifier uses prior, conditional, marginal, and posterior probabilities to create a Bayes model. It classifies a new observation by using to 0-1 loss. If the feature has a relationship with the classification, it can be used for more accurate predictions using the features.

The decision boundary is linear in .

Bayes classifier minimizes the expected 0-1 loss because it chooses the class that has the best posterior probability and penalizes the classes with misclassification.

1.3) If in subquestion 1.2), what is the threshold value on that is used by the Bayes classifier to determine the class label for ? Suppose you use a different threshold value on to classify , is the corresponding classifier still the Bayes classifier, and is the corresponding loss function still the 0-1 loss? Explain your answer. Provide a scenario where to classify an observation a different threshold value is more sensible than the threshold value used by the Bayes classifier.

When , the threshold value on is 0.5 on the average label to determine the class label for .

The corresponding classifier is still the Bayes classifier. The corresponding loss function is still the 0-1 loss.

1.4) If in subquestion 1.2), , and and . Compute and use the Bayes classifier to classify .

# Conceptual exercises: II (-NN classifier)

. Given the training set of observations , where is the class label of observation and for , consider -NN classifier, where is the neighborhood size.

2.1) Describe how the decision boundary (such as its smoothness and shape) of -NN classifier changes as changes.

While increases, is the average of considerably many ’s, whose corresponding ’s in can span a large subset of the feature space. When feature observations are not far apart, their neighborhood will not contain different observations and be “insufficiently flexible”.

2.2) Explain why the training error of -NN classifier is . Provide an estimator of the test error of a classifier and explain why it can be used as such an estimator. Is it true that a large leads to a -NN classifier with smaller test error? Can the test error of a -NN classifier be equal to the test error of the Bayes classifier? When is large and is small, what is a -NN classifier approximately estimating?

The training error of -NN classifier is because under certain conditions, the expected 0-1 loss can be well approximated by either the training error or the test error.

It is not true that a large leads to a -NN classifier with smaller test error.

The test error of a -NN classifier can be equal to the test error of the Bayes classifier?

When is large and is small, the -NN classifier is approximately estimating the posterior.

2.3) When there are classes, how does a -NN classifier classify a test observation ?

A -NN classifier classifies a test observation by basing its performance on smaller test errors.

2.4) When should data be standardized before applying a -NN classifier? When standardizing data, do we standardize each observation or each feature?

Data be standardized before applying a -NN classifier right before the data is being split into training and testing data.

We standardize each each feature of the data (i.e. column).

2.5) Using your understanding of Example 3 in “LectureNotes4b\_notes.pdf”, provide a step-by-step guide on how to choose an optimal for -NN classifier using cross-validation. You can provide such as guide in the form of “pseudocode” (see, e.g., <https://en.wikipedia.org/wiki/Pseudocode> for some details on pseudocode). Suppose the training set has few observations, can you still perform cross-validation in order to choose an optimal ? Explain your answer. (Hint: for the 2nd part, think about if having more observations helps better estimate test error.)

How to choose an optimal for -NN classifier using cross-validation:

1. Randomly split n observations into m folds of approximately equal size
2. Pick fold s as “test set”, set the remaining folds as “training set”, apply the kNN classifier, and obtain test error es
3. Do Item 2. for each s in {1, . . . , m}, and obtain m es’s
4. Compute sample mean µˆ(k, m) and sample standard deviation σˆ(k, m) for the m es’s
5. Use µˆ(k, m) as an estimate of the test error of the classifier
6. Pick a sequence C of b distinct values for k
7. Do Step 1 and Step 2 for each k in C, obtain b estimated test errors, µˆ(k, m), k ∈ C, for the b kNN classifiers
8. Set as the optimal for which µˆ(ˆk, m) is the smallest among µˆ(k, m), k ∈ C

You can still perform cross-validation in order to choose an optimal with few observations.

# Conceptual exercises: III (Discriminant analysis)

. Exercise 2 of Section 4.7 of the Text, which starts with “It was stated in the text that classifying an observation to the class for which (4.12) is largest is equivalent to classifying an observation to the class for which (4.13) is largest. Prove that this is the case.” (Helpful information on how to prove this is contained in the lecture video on LDA and “LectureNotes5b\_notes.pdf”.)

. Exercise 3 of Section 4.7 of the Text, which starts with “This problem relates to the QDA model, in which the observations within each class are drawn from a normal distribution with a class specific mean vector and a class specific covariance matrix. We consider the simple case where p = 1; i.e. there is only one feature.” (Helpful information on how to prove this is contained in the lecture video on QDA and “LectureNotes5b\_notes.pdf”.)

Replace with for the posterior probability for Baye’s classifier equation and we get:

. Exercise 5 of Section 4.7 of the Text, which starts with “We now examine the differences between LDA and QDA.” (Hint: for this question, you may also use information from Figures 4.9, 4.10 and 4.11 in the Text.)

1. If the Bayes decision boundary is linear, do we expect LDA or QDA to perform better on the training set? On the test set?

QDA on training set - LDA on a on test set

1. If the Bayes decision boundary is non-linear, do we expect LDA or QDA to perform better on the training set? On the test set?

QDA will perform better both on the training and test sets.

1. In general, as the sample size n increases, do we expect the test prediction accuracy of QDA relative to LDA to improve, decline, or be unchanged? Why?

For QDA, if a sample size is small, QDA performs better since it depends on the variance of sample size. The performance of LDA and QDA depends on the Bayes’ decision boundary.

1. True or False: Even if the Bayes decision boundary for a given problem is linear, we will probably achieve a superior test error rate using QDA rather than LDA because QDA is flexible enough to model a linear decision boundary.

False

. Let be the random variable for the class label of a random vector (where is the number of features), such that and for Class with , where is the number of classes. Consider the Gaussian mixture model such that the conditional density of when it comes from Class is . Given the training set of observations on , where is the class label of observation , do the following:

6.1) Provide the MLEs of , and for each respectively for the case where all ’s are equal and for the case where not all ’s are equal. When , is the MLE of still accurate? If not, recommend a different estimator for estimating and provide details on this estimator.

6.2) Assume and and . For the density , what shape do its contours take, and how does control the shape of these contours? How do you check if the conditional density of given that it comes from Class is Gaussian?

6.3) Is it true that discriminant analysis will perform badly if the Gaussian assumption is violated? (Hint: for this question, you may also use the information provided by Figures 4.10 and 4.11 of the Text.) Let , i.e., up to are the feature variables. Can discriminant analysis be applied to observations of when some of is a discrete variable (such as a categorical variable)? Explain your answer.

Logistic regression can outperform LDA or QDA if the assumption on a Gaussian mixture model is violated. Discriminant analysis can be applied to observations of when some of is a discrete variable.

6.4) What is a ROC curve, and what is AUC? How is AUC used to gauge the performance of a classifier? If you apply the same classifier, say, LDA or QDA under the same Gaussian mixture model, to two data sets that are independently generated from the same data generating process, i.e., that are independently generated from for classification problems, and obtain two ROC curves, would the two ROC curves be quite different? Explain your answer. When there are 3 or more classes, are the codes provided in the lecture notes able to obtain ROC curves and their AUC’s for LDA and QDA?

A ROC curve is a receiver operating characteristics (ROC) curve and is a popular graphic for simultaneously displaying the two types of errors for all possible thresholds. An AUC is area under the (ROC) curve that is the overall performance of a classifier, summarized over all possible thresholds.

6.5) Describe the key similarities and differences, respectively, between LDA and logistic regression. Provide a situation where discriminant analysis can still be sensibly applied but logistic regression is not well-defined.

Similarities: Differences:

# Applied exercises: I (-NN classifier)

. Please refer to the NYC flight data nycflights13 that has been discussed in the lecture notes and whose manual can be found at <https://cran.r-project.org/web/packages/nycflights13/index.html>. We will use flights, a tibble from nycflights13.

Please use set.seed(123) for the whole of this exercise. Randomly select from flights for each of the 3 carrier “UA”, “AA” or “DL” 500 observations for the 3 features dep\_delay, arr\_delay and distance. Let us try to see if we can use the 3 features to identify if an observation belongs a specific carrier. The following tasks and questions are based on the extracted observations. Note that you need to remove rows with na’s from the extracted observations.

library(nycflights13)  
data(flights)  
  
f <- na.omit(flights)  
  
df1 <- filter(f, f$carrier == c("UA", "AA", "DL"))  
df2 <- df1[, c("carrier", "dep\_delay", "arr\_delay", "distance")]  
flight <- df2[sample(nrow(df2), 500), ]  
flight

## # A tibble: 500 x 4  
## carrier dep\_delay arr\_delay distance  
## <chr> <dbl> <dbl> <dbl>  
## 1 UA 5 20 404  
## 2 UA 2 -8 733  
## 3 DL 3 -6 1076  
## 4 UA -2 -13 2475  
## 5 AA -4 -32 2422  
## 6 UA -5 -41 997  
## 7 DL 81 89 1183  
## 8 UA -5 -27 719  
## 9 DL -2 -7 762  
## 10 DL 11 -8 1096  
## # ... with 490 more rows

7.1) First, you need to standardize the features since they are on very different scales. Then randomly split the observations into a training set that contains 70% of the observations and a test set that contains the remaining observations.

# classes  
flight$Class[flight$carrier == "UA"] <- 0  
flight$Class[flight$carrier == "AA"] <- 1  
flight$Class[flight$carrier == "DL"] <- 2  
  
# standardize  
fdf <- scale(flight[, 2:4])  
set.seed(123)  
  
# split data  
trainSet <- base::sample(1:nrow(fdf), 0.7 \* nrow(fdf))  
testSet <- (1:nrow(fdf))[-trainSet]  
train <- fdf[trainSet, ]  
test <- fdf[testSet, ]  
  
# create class labels  
trainLabels <- flight$Class[trainSet]  
testLabels <- flight$Class[testSet]

7.2) Consider the observations as forming 3 classes that are determined by carrier. To the training set, apply fold cross-validation to -NN classifier with features arr\_delay, dep\_delay, and distance to determine the optimal from the values . Apply the optimal -NN to the test set, provide the classification table and the overall error rate, and provide visualization of the classification results. Do you think the error rate is reasonable? Explain your answer. (Hint: you can follow the strategy provided by Example 3 in “LectureNotes4b\_notes.pdf”. )

m = 10  
set.seed(123)  
  
folds = sample(1:m, nrow(train), replace = TRUE)  
  
k = 2  
testError1 = double(m)  
  
library(MASS)

##   
## Attaching package: 'MASS'

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

library(class)  
  
# k min loop through s=1,...,m  
for (s in 1:m) {  
 trainingTmp = train[folds != s, ]  
 testTmp = train[folds == s, ]  
 trainingLabsTmp = trainLabels[folds != s]  
 testLabsTmp = trainLabels[folds == s]  
 knn2 = knn(trainingTmp, testTmp, trainingLabsTmp, k)  
 nOfMissObs = sum(1 - as.numeric(knn2 == testLabsTmp))  
 terror = nOfMissObs/length(testLabsTmp) # test error  
 testError1[s] = terror  
} # end of loop  
  
# k max  
kmax = 20 # m=10 fold cv  
testErrors = matrix(0, nrow = 2, ncol = kmax)  
for (k in 1:kmax) {  
 # loop through k  
 testError1 = double(m) # store test errors for each k  
 for (s in 1:m) {  
 # loop through s  
 trainingTmp = train[folds != s, ]  
 testTmp = train[folds == s, ]  
 trainingLabsTmp = trainLabels[folds != s]  
 testLabsTmp = trainLabels[folds == s]  
 knntmp = knn(trainingTmp, testTmp, trainingLabsTmp, k)  
 nOfMissObs = sum(1 - as.numeric(knntmp == testLabsTmp))  
 terror = nOfMissObs/length(testLabsTmp) # test error  
 testError1[s] = terror  
 } # loop in s ends  
 testErrors[, k] = c(mean(testError1), sd(testError1))  
} # loop in k ends  
  
# data frame for mean/sd  
colnames(testErrors) = paste("k=", 1:kmax, sep = "")  
rownames(testErrors) = c("mean(TestError)", "sd(TestError)")  
testErrors = as.matrix.data.frame(testErrors)  
as.numeric(testErrors[1, ])

## [1] 0.5518809 0.6065577 0.5429671 0.5351178 0.5420264 0.5132973 0.5294487  
## [8] 0.5069764 0.5451035 0.5126418 0.4923497 0.5140918 0.5038856 0.5034572  
## [15] 0.4892242 0.4962172 0.4867139 0.5113045 0.5177894 0.5268732

# optimal k  
hatk = which(testErrors[1, ] == min(testErrors[1, ]))  
hatk

## [1] 17

The error rate is reasonable because all means and standard deviations are relatively close to one another. The smallest test error rate among a family of kNN classifiers is when k = 9.

7.3) Note that your have standardized the features arr\_delay, dep\_delay, and distance. However, with the unstandardized features, you would surely know that none of them follows a Gaussian distribution no matter with respect to which class (i.e., carrier) you look at their observations since these features have non-negative values (whereas a Gaussian distribution can generate negative values). Again, the 3 classes are determined by carrier. So, you will apply QDA based on the 3 standardized the features to the training set to train the model, and then apply the trained model to the test set (that contains the 3 standardized the features) to classify its observations.

set.seed(123)  
  
t <- as.data.frame(train)  
dd <- as.data.frame(t$dep\_delay)  
ad <- as.data.frame(t$arr\_delay)  
dst <- as.data.frame(t$distance)  
  
qda.fit1 <- qda(trainLabels ~ ., data = dd)  
pred1 <- predict(qda.fit1)$class  
mtx1 <- as.data.frame(pred1)  
qda.fit2 <- qda(trainLabels ~ ., data = ad)  
pred2 <- predict(qda.fit2)$class  
mtx2 <- as.data.frame(pred2)  
qda.fit3 <- qda(trainLabels ~ ., data = dst)  
pred3 <- predict(qda.fit3)$class  
mtx3 <- as.data.frame(pred3)  
  
trainModel <- cbind(mtx1, mtx2, mtx3)  
  
# apply the trained model to the test set (that contains the  
# 3 standardized the features) to classify its observations  
knnOpt <- knn(trainModel, test, trainLabels, hatk)  
table(knnOpt, testLabels)

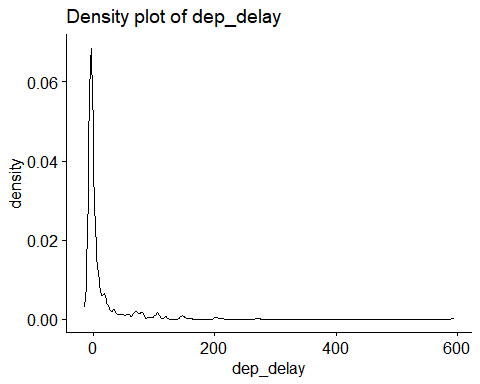
## testLabels  
## knnOpt 0 1 2  
## 0 44 20 42  
## 1 0 0 0  
## 2 25 7 12

(7.3a) First, check if the Gaussian assumption is satisfied. For this, note that if the standardized features arr\_delay, dep\_delay, and distance follow a trivariate Gaussian distribution for each individual class, then any pair among the 3 standardized features follows a bivariate Gaussian distribution for each individual class.

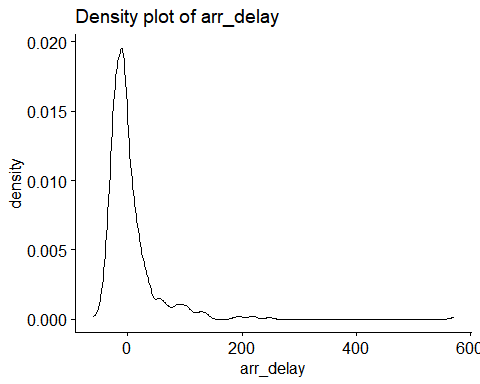
library("ggpubr")

## Loading required package: ggplot2

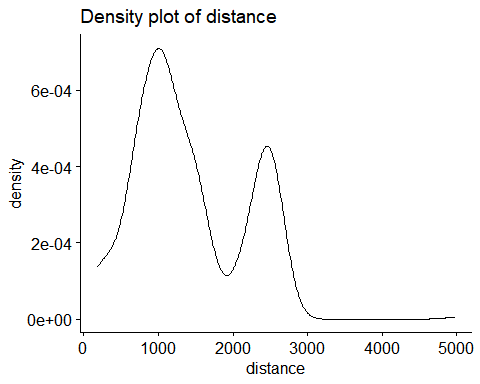
ggdensity(flight$dep\_delay, main = "Density plot of dep\_delay",   
 xlab = "dep\_delay")



ggdensity(flight$arr\_delay, main = "Density plot of arr\_delay",   
 xlab = "arr\_delay")



ggdensity(flight$distance, main = "Density plot of distance",   
 xlab = "distance")



(7.3b) Apply the estimated (i.e., trained) QDA model to the test set, provide the estimated mixing proportion and estimated mean vector for each class, and provide the classification table. If you randomly pick an observation on the 3 standardized features, is it approximately equally likely to belong to each of the 3 carriers? (You do not need to mathematically prove your answer. However, think along this line: we are testing the equality of 3 population proportions, and each estimated population proportion is based on around 350 observations, which approximately can be done via a z-test since the central limit theorem is in effect.) How is the performance of QDA on this test set? Explain your answers.

library(MASS)  
  
# apply the estimated (i.e., trained) QDA model to the test  
# set provide the estimated mixing proportion and estimated  
# mean vector for each class  
qda.class1 <- predict(qda.fit1, dd)$class  
qda.fit1$prior

## 0 1 2   
## 0.4028571 0.2457143 0.3514286

qda.fit1$means

## `t$dep\_delay`  
## 0 -0.06801519  
## 1 -0.09552305  
## 2 0.17467422

qda.class2 <- predict(qda.fit2, ad)$class  
qda.fit2$prior

## 0 1 2   
## 0.4028571 0.2457143 0.3514286

qda.fit2$means

## `t$arr\_delay`  
## 0 -0.04298218  
## 1 -0.07054551  
## 2 0.14404830

qda.class3 <- predict(qda.fit3, dst)$class  
qda.fit3$prior

## 0 1 2   
## 0.4028571 0.2457143 0.3514286

qda.fit3$means

## `t$distance`  
## 0 0.26270912  
## 1 -0.03637665  
## 2 -0.31167381

tst <- as.data.frame(test)  
dd2 <- as.data.frame(tst$dep\_delay)  
ad2 <- as.data.frame(tst$arr\_delay)  
dst2 <- as.data.frame(tst$distance)  
  
# provide the classification table  
table(as.data.frame(qda.class1), dd2)

## dd2  
## c(-0.156844925078681, -0.203931363108906, -0.392277115229808, -0.32164745818447, -0.368733896214695, 2.52708204264417, -0.274561020154244, 0.361105893253799, 1.34992109188853, -0.06267204901823, -0.462906772275146, -0.298104239169357, -0.345190677199583, -0.415820334244921, 0.0785872650724462, -0.180388144093794, 1.16157533976763, -0.109758487048455, -0.251017801139132, -0.227474582124019, 0.314019455223573, 2.17393375741748, 1.42055074893387, -0.0155856109880046, 0.667167740450264, 0.00795760802710808, \n1.09094568272229, 2.29164985249304, 3.28046505112777, 0.102130484087559, 2.2210201954477, -0.133301706063568, 0.219846579163122, -0.486449991290259, 0.784883835525828, 2.26810663347793, 0.19630336014801, 3.49235402226379, 0.855513492571166, 0.596538083404926, -0.439363553260033, 0.172760141132897, -0.58062286735071, 0.149216922117784, 1.37346431090365, 0.926143149616504, 0.80842705454094)  
## c(1, 3) 0

table(as.data.frame(qda.class2), ad2)

## ad2  
## c(0.356852081748031, -0.20005993058383, -0.949749177953642, -0.649873479005717, -0.221479623365824, -0.285738701711808, -0.521355322313749, 1.81339119092367, 0.0783960755821003, 1.04228225077186, 1.6634533414497, -0.671293171787712, -0.0929614666738567, -0.607034093441728, -0.971168870735637, -0.692712864569706, -0.564194707877739, -0.499935629531755, -0.371417472839787, -0.778391635697685, 0.12123546114609, 0.892344401297896, -0.135800852237846, 1.02086255798986, -0.114381159455851, -0.628453786223722, \n-0.178640237801835, -0.585614400659733, -0.242899316147819, -0.264319008929814, -0.928329485171647, -0.542775015095744, -0.75697194291569, 0.164074846710079, 0.39969146731202, 0.463950545658004, 2.15610627543558, -0.0287023883278728, 0.185494539492074, 0.870924708515902, -0.328578087275798, 0.206914232274068, -0.349997780057792, 0.249753617838057, 0.0569763828001057, 1.10654132911784, 2.84153644445941, 0.0355566900181111, 2.52024105272949, -0.414256858403776, 0.142655153928084, 0.806665630169918, \n1.89906996205164, 0.335432388966036, -0.864070406825663, 0.0141369972361165, -0.392837165621782, 0.378271774530025, 1.53493518475774, 1.83481088370566, 0.635308087913961, 0.742406551823934, 2.69159859498545, -0.0501220811098674, -0.714132557351701, 1.32073825693779, -0.307158394493803, -1.20678549133758, 0.421111160094014, 0.849505015733907, 1.19222010024582, 0.592468702349972, 0.485370238439998, 0.228333925056063, 1.14938071468183, -0.735552250133696, -0.47851593674976)  
## c(1, 3) 0

table(as.data.frame(qda.class2), dst2)

## dst2  
## c(-1.37616991696692, -0.446880395745309, -0.556127229341302, -0.940565453894289, -0.881101987506597, 1.64125528438063, -0.434434553943234, -0.0140416752953634, 0.288807141888465, 0.305401597624565, 1.38680696309376, 1.4877565688217, 1.17384478114714, 0.00116990912939507, -0.037550487588172, -0.903227928488064, 1.45871627128353, -1.65827566448062, -0.462091980170068, 1.43105884505669, -0.883867730129281, -0.419222969518476, -0.921205255535506, 0.023295850110862, -0.428903068697868, 1.61221498684245, \n0.324761795983348, 0.28465852795444, -0.524321189180444, -0.540915644916544, 0.817063982820987, -0.639099508021803, 1.41861300325462, 0.788023685282811, 1.04247200656968, -1.56285754399805, -1.67625299152806, -0.321039106413217, 1.41446438932059, -0.520172575246419, 0.313698825492615, -0.503578119510318, -0.53814990229386, 1.14480448360897, -1.24064852845543, -0.545064258850569, 0.547404077109359, -0.621122180974361, -0.629419408842411, -0.456560494924701, 0.274978428775048, 1.44765330079279)  
## c(1, 3) 0

If you randomly pick an observation on the 3 standardized features, it is approximately equally likely to belong to each of the 3 carriers. How is the performance of QDA on this test set? Explain your answers.

(7.3c) Extract observations that are for “UA” or “DL” from the training set and the test set, respectively, to form a new training set and a new subset, so that there are now 2 classes “UA” and “DL”. Apply QDA to the new training set and then apply the trained model to the new test set. Report the overall error rate on the test set, provide the ROC curve, and calculate the AUC. How is the performance of QDA on this test set? Explain your answer.

flight\_new <- filter(flight, flight$carrier == c("UA", "DL"))  
  
# standardize  
fdf\_new <- scale(flight\_new[, 2:4])  
set.seed(123)  
  
# split data  
trainSet\_new <- base::sample(1:nrow(fdf\_new), 0.7 \* nrow(fdf\_new))  
testSet\_new <- (1:nrow(fdf\_new))[-trainSet\_new]  
train\_new <- fdf[trainSet\_new, ]  
test\_new <- fdf[testSet\_new, ]  
  
# create class labels  
trainLabels\_new <- flight\_new$Class[trainSet\_new]  
testLabels\_new <- flight\_new$Class[testSet\_new]

t\_new <- as.data.frame(train\_new)  
dd\_new <- as.data.frame(t\_new$dep\_delay)  
ad\_new <- as.data.frame(t\_new$arr\_delay)  
dst\_new <- as.data.frame(t\_new$distance)  
  
set.seed(123)  
  
qda.fit1\_new <- qda(trainLabels\_new ~ ., data = dd\_new)  
pred1\_new <- predict(qda.fit1\_new)$class  
mtx1\_new <- as.data.frame(pred1\_new)  
qda.fit2\_new <- qda(trainLabels\_new ~ ., data = ad\_new)  
pred2\_new <- predict(qda.fit2\_new)$class  
mtx2\_new <- as.data.frame(pred2\_new)  
qda.fit3\_new <- qda(trainLabels\_new ~ ., data = dst\_new)  
pred3\_new <- predict(qda.fit3\_new)$class  
mtx3\_new <- as.data.frame(pred3\_new)  
  
trainModel\_new <- cbind(mtx1\_new, mtx2\_new, mtx3\_new)  
  
# apply the trained model to the test set (that contains the  
# 3 standardized the features) to classify its observations  
knnOpt\_new <- knn(trainModel\_new, test\_new, trainLabels\_new,   
 hatk)  
table(knnOpt\_new, testLabels\_new)

## testLabels\_new  
## knnOpt\_new 0 2  
## 0 33 27  
## 2 0 0

qda.class1\_new <- predict(qda.fit1\_new, dd\_new)$class  
qda.fit1\_new$prior

## 0 2   
## 0.5434783 0.4565217

qda.fit1\_new$means

## `t\_new$dep\_delay`  
## 0 -0.08339008  
## 2 0.29682917

qda.class2\_new <- predict(qda.fit2\_new, ad\_new)$class  
qda.fit2\_new$prior

## 0 2   
## 0.5434783 0.4565217

qda.fit2\_new$means

## `t\_new$arr\_delay`  
## 0 -0.1160947  
## 2 0.2320739

qda.class3\_new <- predict(qda.fit3\_new, dst\_new)$class  
qda.fit3\_new$prior

## 0 2   
## 0.5434783 0.4565217

qda.fit3\_new$means

## `t\_new$distance`  
## 0 -0.02243109  
## 2 -0.04143570

tst\_new <- as.data.frame(test\_new)  
dd2\_new <- as.data.frame(tst\_new$dep\_delay)  
ad2\_new <- as.data.frame(tst\_new$arr\_delay)  
dst2\_new <- as.data.frame(tst\_new$distance)  
  
# provide the classification table  
table(as.data.frame(qda.class1), dd2\_new)

## dd2\_new  
## c(-0.227474582124019, -0.203931363108906, -0.0155856109880046, -0.368733896214695, -0.274561020154244, 0.361105893253799, -0.439363553260033, 1.34992109188853, -0.392277115229808, -0.156844925078681, -0.557079648335597, -0.415820334244921, 0.149216922117784, -0.462906772275146, 0.0785872650724462, 0.879056711586278, -0.180388144093794, -0.298104239169357, -0.109758487048455, -0.345190677199583, -0.32164745818447, -0.251017801139132, -0.06267204901823, -0.509993210305371, 0.502365207344475, 2.55062526165928, \n0.314019455223573, 3.16274895605221, -0.486449991290259)  
## c(1, 3) 0

table(as.data.frame(qda.class2), ad2\_new)

## ad2\_new  
## c(-0.242899316147819, -0.20005993058383, -0.328578087275798, -0.521355322313749, 0.0783960755821003, 1.04228225077186, -0.607034093441728, 1.6634533414497, -0.671293171787712, -0.564194707877739, -0.971168870735637, -0.435676551185771, -0.499935629531755, -0.649873479005717, -0.135800852237846, -0.79981132847968, 0.892344401297896, 0.720986859041939, 0.506789931221993, -0.714132557351701, -0.371417472839787, -0.0929614666738567, -0.114381159455851, 0.0355566900181111, -0.821231021261674, 0.292593003402047, \n-0.349997780057792, 0.271173310620052, -0.285738701711808, -0.178640237801835, 0.12123546114609, -0.585614400659733, -0.221479623365824, 2.30604412490954, 0.39969146731202, 2.90579552280539, -0.414256858403776, 0.421111160094014, 0.185494539492074, 0.249753617838057, -0.157220545019841)  
## c(1, 3) 0

table(as.data.frame(qda.class2), dst2\_new)

## dst2\_new  
## c(-0.921205255535506, -0.446880395745309, -0.419222969518476, -0.520172575246419, -0.434434553943234, 0.288807141888465, -0.940565453894289, 0.305401597624565, 1.41446438932059, 1.17384478114714, -0.456560494924701, 0.00116990912939507, -0.037550487588172, -1.65827566448062, -0.639099508021803, 1.43105884505669, 0.274978428775048, -1.26000872681422, 1.64125528438063, 1.45871627128353, -0.29891316543175, -0.428903068697868, 1.41861300325462, -1.24064852845543, -0.903227928488064, 1.4877565688217, \n-0.540915644916544, -0.881101987506597, -0.0140416752953634, 1.61221498684245, 1.04247200656968, -0.53814990229386, -0.883867730129281, 0.168497337801738, -0.321039106413217, -0.621122180974361)  
## c(1, 3) 0

# Applied exercises: II (Discriminant analysis)

. The following is on software commands:

(8.1) What is the main cause of the message “Warning in lda.default(x, grouping, …): variables are collinear”? What is the main cause of the message “Error in qda.default(x, grouping, …) : some group is too small for ‘qda’”?

Warning in lda.default(x, grouping, …): variables are collinear: the predictors are correlated, which leads to inclusive results if the predictors are too similar.

Error in qda.default(x, grouping, …) : some group is too small for ‘qda’: there are factors that are being made to predict and they all need to have the same level of observations in order for it to work.

(8.2) Provide details on the list that predict{MASS} returns.

The list includes a class which is the map classification. It also includes posterior which is the posterior probabilites of the classes.

(8.3) The arguments gamma and lambda of rda{klaR} are usually determined by cross-validation. Can they be set manually?

Yes

. We will use the human cancer microarray data that were discussed in the lectures and are provided by the R library ElemStatLearn (available at <https://cran.r-project.org/src/contrib/Archive/ElemStatLearn/>). Pick 3 cancer types “MELANOMA”, “OVARIAN” and “RENAL”, and randomly select the same set of genes for each cancer type. Please use set.seed(123) for the whole of this exercise. Your analysis will be based on observations for these genes and cancer types.

9.1) Pick 2 features and visualize the observations using the 2 features. Do you think it is hard to classify the observations based on the amount of overlap among the 3 neighborhoods of observations in each of the 3 classes? Here “a neighborhood of observations in a class” is a “open disk that contains the observations in the class”.

library(ElemStatLearn)  
data("nci")  
n0 = dim(nci)[2]  
p0 = dim(nci)[1]  
  
set.seed(123)  
  
rSel = sample(1:p0, size = 60, replace = FALSE)  
chk = colnames(nci) %in% c("MELANOMA", "OVARIAN", "RENAL")  
cSel = which(chk == TRUE)  
ncia = nci[rSel, cSel]  
colnames(ncia) = colnames(nci)[cSel]  
  
tmp = data.frame(t(ncia))  
tmp$Class = colnames(ncia)  
table(tmp$Class)

##   
## MELANOMA OVARIAN RENAL   
## 8 6 9

tmp[1:2, (ncol(tmp) - 2):ncol(tmp)]

## X59 X60 Class  
## RENAL -0.03 3.50 RENAL  
## RENAL.1 -0.04 0.43 RENAL

It is not hard to classify the observations based on the amount of overlap among the 3 neighborhoods of observations in each of the 3 classes when viewed in their ambient space R^2.

9.2) Apply LDA and report the classwise error rate for each cancer type.

lda.fit = lda(Class ~ ., data = tmp)  
lda.pred = predict(lda.fit, tmp)  
TrueClassLabel = tmp$Class  
LDAEstimtedClassLabel = lda.pred$class  
table(LDAEstimtedClassLabel, TrueClassLabel)

## TrueClassLabel  
## LDAEstimtedClassLabel MELANOMA OVARIAN RENAL  
## MELANOMA 6 0 0  
## OVARIAN 0 5 0  
## RENAL 2 1 9

# classwise errors

MELANOMA: 6/6 = 1 RENAL: 5/5 = 1 OVARIAN: 9/12 = 0.75

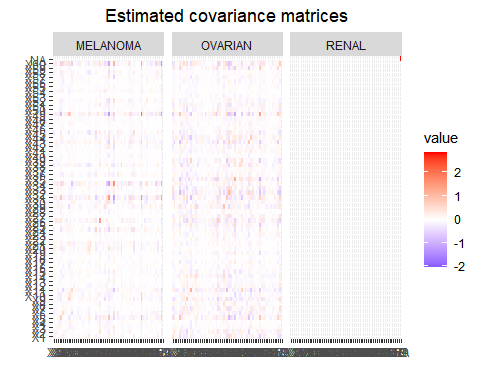
9.3) Use the library klaR, and apply regularized discriminant analysis (RDA) by setting the arguments gamma and lambda of rda{klaR} manually so that the resulting classwise error rate for each cancer type is zero.

library(klaR)  
  
rda.fit = rda(Class ~ ., data = tmp, gamma = 0.05, lambda = 0.2)  
rda.pred = predict(rda.fit, tmp) # classify obs in `tmp`  
TrueClassLabel = tmp$Class  
rDAEstimtedClassLabel = rda.pred$class  
table(rDAEstimtedClassLabel, TrueClassLabel)

## TrueClassLabel  
## rDAEstimtedClassLabel MELANOMA OVARIAN RENAL  
## MELANOMA 8 0 0  
## OVARIAN 0 6 0  
## RENAL 0 0 9

9.4) Obtain the estimated covariance matrices from the RDA and visualize them using the same strategy in Example 3 in “LectureNotes5c\_notes.pdf”. What can you say about the degree of dependence among these genes for each of the three cancer types? (Hint and caution: the class labels “MELANOMA”, “OVARIAN” and “RENAL” will be ordered sigmabetically by R. So, you need to keep track on which estimated covariance matrix is for which class. Otherwise, you will get wrong visualization.)

hatSigma1 = rda.fit$covariances[, , 1]  
hatSigma2 = rda.fit$covariances[, , 2]  
hatSigma3 = rda.fit$covariances[, , 3]  
  
library(reshape2)  
  
melted\_hatSigma1 = melt(hatSigma1)  
melted\_hatSigma2 = melt(hatSigma2)  
melted\_hatSigma3 = melt(hatSigma3)  
EstSigmaAll = rbind(melted\_hatSigma1, melted\_hatSigma2, +melted\_hatSigma3)  
EstSigmaAll$Cancer = rep(c("MELANOMA", "OVARIAN", "RENAL"), each = nrow(melted\_hatSigma1))  
EstSigmaAll$Cancer = factor(EstSigmaAll$Cancer)  
  
library(ggplot2)  
  
ggplot(data = EstSigmaAll, aes(x = Var1, y = Var2, fill = value)) +   
 geom\_tile() + scale\_fill\_gradient2(low = "blue", high = "red") +   
 facet\_grid(~Cancer) + xlab("") + ylab("") + ggtitle("Estimated covariance matrices") +   
 theme(plot.title = element\_text(hjust = 0.5))

 There is much less degree of dependence among genes for “RENAL” cancer versus thes genes for other two cancer types.