Stat 437 HW4

Pat McCornack (11516300)

Conceptual exercises: I (Bayes classifier)

1. 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 a measure of the performance of a classifier that utilizes the sum of the number of misclassifications. It can be used for multi-class classification, and the definition is L(k, l) = 0 if k = l and L(k, l) = 1 if k $\neq$ l. In other words, if the predicted classification matches the true classification then the loss function is 0 for that observation and 1 if not. The Bayes classifier minimizes the total expected loss to draw a decision boundary.

1.2) Let Y be the random variable for the class label of a random vector X, such that $Y \in \mathcal{G} = \{1, \ldots, K\}$ where $K \geq 2$ is the number of classes. Let $\hat{Y}$ be the estimated class label for X. Given the prior $\Pr(Y = k) = \pi_k, k \in \mathcal{G}$ on Class k and the conditional density $f_k(x)$ of X when it comes from Class k. Provide the formula to obtain the posterior $\Pr(Y = k|X = x)$, which is essentially the Bayes theorem. What is the Bayes classifier and how does it classify a new observation x_0 from X? Is the decision boundary of the Bayes classifier linear or quadratic in X? 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.

The posterior probability is defined

$$Pr(Y = k|X = x) = \frac{f_k \pi_k}{\sum_{k=1}^{K} f_k \pi_k}$$

. The Bayes classifier assigns the class with the highest posterior probability to a new observation x_0.The Bayes decision boundary is quadratic in X. The Bayes classifier minimizes the 0-1 loss by assigning the most likely class based on the posterior probability - which is based on the population prior and conditional probabilities. Bayes is the unattainable gold standard because it uses these probabilities rather than estimates of these probabilities.

1.3) If $K = 2$ in subquestion 1.2), what is the threshold value on $\Pr(Y = 1|X = x_0)$ that is used by the Bayes classifier to determine the class label for x_0? Suppose you use a different threshold value on $\Pr(Y = 1|X = x_0)$ to classify x_0, 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 $K = 2$ the threshold value for class assignment is 50%. If another threshold value is used the classifier is no longer a Bayes classifier because it is not necessarily maximizing the probability. The corresponding loss function would still be the 0-1 loss though, because that metric is based on misclassifications and not associated with maximizing the posterior.

Using a different threshold may make sense depending on the priorities of the use case. For example, when trying to predict credit defaults it may make sense to decrease the threshold in order to increase the odds of obtaining true positives. This increases the odds of also obtaining false positives, but it may be more important to the company to identify defaulters than miclassify those who don't default as defaulters.

1.4) If $K = 2$ in subquestion 1.2), $\pi_1 = 0.6$, $f_1(x) \sim$ Gaussian$(0, 1)$ and $f_2(x) \sim$ Gaussian$(2, 1)$ and $x_0 = 1.5$. Compute $\Pr(Y = 1|X = x_0)$ and use the Bayes classifier to classify x_0.

Conceptual exercises: II (k-NN classifier)

2. Given the training set $\mathcal{T}$ of n observations $(x_1, y_1), \ldots, (x_n, y_n)$, where y_i is the class label of observation x_i and $y_i \in \mathcal{G} = \{1, \ldots, K\}$ for $K \geq 2$, consider k-NN classifier, where k is the neighborhood size.

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

As k increases, the decision boundary becomes more smooth and less convex.

2.2) Explain why the training error of 1-NN classifier is 0. 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 k leads to a k-NN classifier with smaller test error? Can the test error of a k-NN classifier be equal to the test error of the Bayes classifier? When k is large and k/n is small, what is a k-NN classifier approximately estimating?

When using a neighborhood size of 1, a new observation will simply assign the class of the point in the training set closest to the new observation. When inputting the training set the observation under consideration is itself in the training set, so it will assign its own label to itself. Therefore every observation in the training set will be assigned its own label, and the error will be 0.

The test error of a classifier can be used to estimate the expected loss, and is defined as the mean of the loss function over the test set, or: $\frac{1}{n}\sum_{i=1}^{n} L(y_i, \hat{y}_i)$ where y is the true label and $\hat{y}$ is the predicted label. The larger the size of the test sample, the closer this value approximates the expected loss. Since we often don't know the labels of a test set, what can be done to estimate the test error is to split the training data into a training set and 'test set'. The classifier is then trained using the training set, which is a subset of the entire training data, and the trained classifier is applied to the 'test set'. We know the labels of the observations in the 'test set' because they are in the training dataset, but they can still be used to estimate the true test error because the classifier was not trained using them. The process of splitting the training data into a training set and test set can be done m many times with different splits, and the mean test error can be calculated from the results as an estimate the of the true test error.

It is not true that a large k leads to a classifier with a smaller test error. The optimal neighborhood size is often an intermediate value.

The test error of a k-NN classifier may be equal to the error of the Bayes classifier. This is more likely for datasets that are easily separable.

When k is large and k/n is small the k-NN classifier is estimating the posterior probability that an observation belongs to a particular class, and is therefore similar to both logistic regression and the Bayes classifier.

2.3) When there are $K \geq 2$ classes, how does a k-NN classifier classify a test observation x_0?

When there are two or more classes, the k-NN classifier assigns the most common class among the test observations k nearest neighbors to the test observation.

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

Data should be standardized when differences among the units of the features will lead to large differences in the scales among features. Standardization is done to center and scale each feature.

2.5) Using your understanding of Example 3 in "LectureNotes4b_notes.pdf", provide a step-by-step guide on how to choose an optimal k for k-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 k? Explain your answer. (Hint: for the 2nd part, think about if having more observations helps better estimate test error.)

The overall process of choosing the optimal k value using m fold cross validation is: For each k in some range, split the data into distinct training/test sets m times, train and test the classifier using each training/test set, and calculate the mean and standard deviation among the test errors for each of the m sets. Choose the value of k with the smallest mean test error and break ties by choosing the smallest standard deviation, then the smallest k if a further tie break is needed.

Cross-validation is not a good choice for a training set with few observations, as the 'test set' from the training data will be very small. The test error estimate best approximates the expected loss when there are a large number of observations.

Conceptual exercises: III (Discriminant analysis)

3. 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".)

4. 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".)

5. 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.)

a) If the Bayes decision boundary is linear, then linear discriminant analysis will likely perform better on both the training and test set.

b) If the Bayes decision boundary is non-linear, then quadratic discriminant analysis will perform best on both the training and test set.

c) As n increases the test prediction accuracy of QDA relative to LDA will improve. It is a more flexible classifier, but also more prone to bias. Increasing the sample size will reduce the bias and the higher flexibility means it will better approximate the Bayes classifier decision boundary.

6. Let Y be the random variable for the class label of a random vector $X \in \mathbb{R}^p$ (where p is the number of features), such that $Y \in \mathcal{G} = \{1, \ldots, K\}$ and $\Pr(Y = k) = \pi_k$ for Class k with $k \in \mathcal{G}$, where $K \geq 2$ is the number of classes. Consider the Gaussian mixture model such that the conditional density of X when it comes from Class k is $f_k(x) \sim$ Gaussian(μ_k, Σ_k). Given the training set $\mathcal{T}$ of n observations $(x_1, y_1), \ldots, (x_n, y_n)$ on (X, Y), where y_i is the class label of observation x_i, do the following:

6.1) Provide the MLEs of π_k, μ_k and Σ_k for each $k \in \mathcal{G}$ respectively for the case where all Σ_k's are equal and for the case where not all Σ_k's are equal. When $p > n$, is the MLE of Σ_k still accurate? If not, recommend a different estimator for estimating Σ_k and provide details on this estimator.

When all covariances are equal:
$\hat{\pi}_k = \frac{n_k}{n}$
$\hat{\mu}_k = \Sigma_{i:y_i=k} \frac{x_i}{n_k}$
$\hat{\Sigma} = \frac{1}{n-K} \sum_{k=1}^{K} \sum (x_i - \hat{\mu_k})(x_i - \hat{\mu_k})^T$

When covariances are unequal: $\hat{\pi}_k = \frac{n_k}{n}$
$\hat{\mu}_k = \Sigma_{i:y_i=k} \frac{x_i}{n_k}$
$\hat{\Sigma} = \frac{1}{n-K} \sum_{i:y_i=k} (x_i - \hat{\mu_k})(x_i - \hat{\mu_k})^T$

When $p > n$ the MLEs of Σ_k do not work well. An alternative in this case is to use the regularized estimate. This uses a pooled covariance matrix and is used for regularized discriminant analysis.

6.2) Assume $p = 2$ and $K = 2$ and $k = 1$. For the density $f_k(x) \sim$ Gaussian(μ_k, Σ_k), what shape do its contours take, and how does Σ_k control the shape of these contours? How do you check if the conditional density of X given that it comes from Class k is Gaussian?

The conditional density for k = 1 is a normal distribution whose center is defined by μ_k. The shape of the contours are controlled by Σ_k. The "steepness" of the distribution is higher when the eigenvalue is lower and smaller when the eigenvalue is larger.When features are uncorrelated the distribution is circular, and it becomes more elliptical with higher correlations.
To check that the conditional density is Gaussian one can check to see if the density plot is normally distributed.

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 $X = (X_1, \ldots, X_p)^P$, i.e., X_1 up to X_p are the feature variables. Can discriminant analysis be applied to observations of X when some of $X_j, j = 1 \ldots, p$ is a discrete variable (such as a categorical variable)? Explain your answer.

Discriminant analysis won't necessarily perform badly if the Gaussian assumption is violated, depending how accuracy is defined. For a highly unbalanced dataset it may accurately classify a very high percentage of observations, but may not correctly classify the cases that one is particularly interested in - like credit default.
Discriminant analysis may be applied to categorical variables by assigned a very tight Gaussian distribution to each particular variable. However, in the case of categorical variables it may be best to consider an option other than discriminant analysis.

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 (X, Y) 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?

The ROC curve is used to simultaneously display the rates of false positives and true positives for all classification thresholds. For a case like the unbalanced credit dataset it can be used to select a threshold value that minimimizes the rate of false positives while allowing an acceptable amount of false-negatives. The AUC is the area under the ROC curve, and it is a measure of the overall performance of a classifer.
Applying the same classifier to two data sets generated from the same underlying distribution should result in similar ROC curves. Since classifiers are attempting to find characteristics of the population, not the sample, that can be used to separate the data into classes they should have similar error rates - and therefore similar ROC curves. However, if there are very few samples in the training set then the classifiers may pick up bias from the set and perform somewhat differently.
The lecture codes are specifically for 2 class problems, and would not work for 3 or more classes.

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.

Both LDA and logistic regression calculates the probability that an observation belongs to each class, then assigns the label of the class with the higher probability. A key difference between LDA and logistic regression is that logistic regression does not use a Gaussian Mixture Model, and therefore doesn't take prior probabilities into account. Logistic regression just uses the conditional probabilities to classify, while LDA uses the posterior probability. They're also both parametric, unlike methods like KNN.
The MLEs of logistic regression are undefined when the classes can be perfectly separated by a hyperplane, but discriminant analysis may still be applied in this scenario.

Applied exercises: I (k-NN classifier)

7. 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(tidyverse)
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.2     v readr     2.1.4
## v forcats   1.0.0     v stringr   1.5.0
## v ggplot2   3.4.2     v tibble    3.2.1
## v lubridate 1.9.2     v tidyr     1.3.0
## v purrr     1.0.1
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to bec
```

```
library(nycflights13)
library(class)

set.seed(123)

df <- flights

# Subset the data
df <- df %>% filter(carrier %in% c('UA', 'AA', 'DL')) %>%
    dplyr::select('carrier', 'dep_delay', 'arr_delay', 'distance') %>%
    na.omit()

# Take 500 samples from the data
df.sample <- df[sample(nrow(df), 500), ]
df.sample$carrier <- as.factor(df.sample$carrier)
```

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.

```
# Standardize the data
df.sample[,2:4] <- scale(df.sample[,2:4])

# Split the data into a train and test set
train_ind <- sample(1:nrow(df.sample), nrow(df.sample) * 0.7)
```

```
test_ind <- (1:nrow(df.sample))[-train_ind]
train_set <- df.sample[train_ind, 2:4]
test_set <- df.sample[test_ind, 2:4]
train_labels <- df.sample$carrier[train_ind]
test_labels <- df.sample$carrier[test_ind]
```

7.2) Consider the observations as forming 3 classes that are determined by `carrier`. To the training set, apply 10 fold cross-validation to k-NN classifier with features `arr_delay`, `dep_delay`, and `distance` to determine the optimal k from the values $\{1, \ldots, 15\}$. Apply the optimal k-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".)

The optimal neighborhood size was $k = 3$ and the test error estimate using this neighborhood size was 50.6%. The resulting classification results of the test data set are displayed in the table below, and the test error rate was 53.3%. This is better than a trivial classifier that randomly assigns a classification to each classifier, which would have an approximate error rate of 33.3%, but still relatively unreliable. This does seem reasonable, as I don't suspect that the features under consideration are reliable predictors of the carrier of the flight based on the graph, although there is a trend of United Airlines flights having longer delays.

```
m <- 10
folds <- sample(1:m, nrow(train_set), replace = TRUE)

kmax = 15
test_errors <- matrix(0, nrow=2, ncol=kmax)

for (k in 1:kmax) {
  test_error_itr = double(m)
  for (s in 1:m) {
    train_itr <- train_set[folds != s, ]
    test_itr <- train_set[folds == s, ]
    train_itr_labs <- train_labels[folds != s]
    test_itr_labs <- train_labels[folds == s]
    knn_itr <- knn(train_itr, test_itr, train_itr_labs, k)
    n_missclass <- sum(1 - as.numeric(knn_itr == test_itr_labs))
    t_error <- n_missclass / length(test_itr_labs)
    test_error_itr[s] = t_error
  }
  test_errors[,k] = c(mean(test_error_itr), sd(test_error_itr))
}

colnames(test_errors) <- paste("k=", 1:kmax, sep="")
rownames(test_errors) = c("mean(test_error)", "sd(test_error)")
test_errors = as.data.frame(test_errors)
k_hat <- which(test_errors[1,] == min(test_errors[1,]))
k_hat
```

```
## [1] 3
```

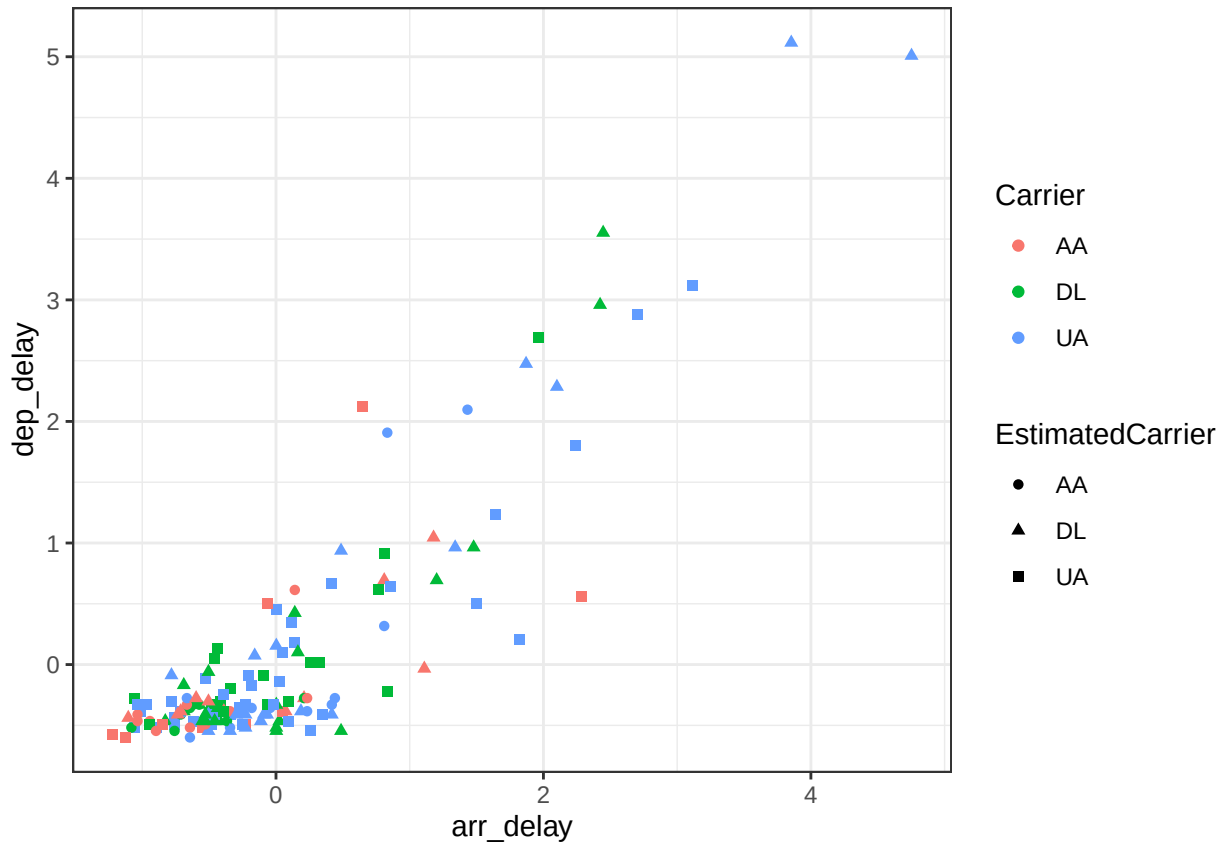
```
knn_test <- knn(train_set, test_set, train_labels, k_hat)
n_missclass_test <- sum(1 - as.numeric(knn_test == test_labels))
t_error_test <- n_missclass_test / length(test_labels)
t_error_test
```

```
## [1] 0.5333333
```

```
print(table(knn_test, test_labels))
```

```
##         test_labels
## knn_test AA DL UA
##       AA 10  9 13
##       DL 11 23 21
##       UA  9 17 37
```

```
test_set$Carrier = test_labels
test_set$EstimatedCarrier = knn_test
ggplot(test_set, aes(arr_delay, dep_delay)) +
  geom_point(aes(shape = EstimatedCarrier, color = Carrier)) +
  theme_bw()
```



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.

(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.

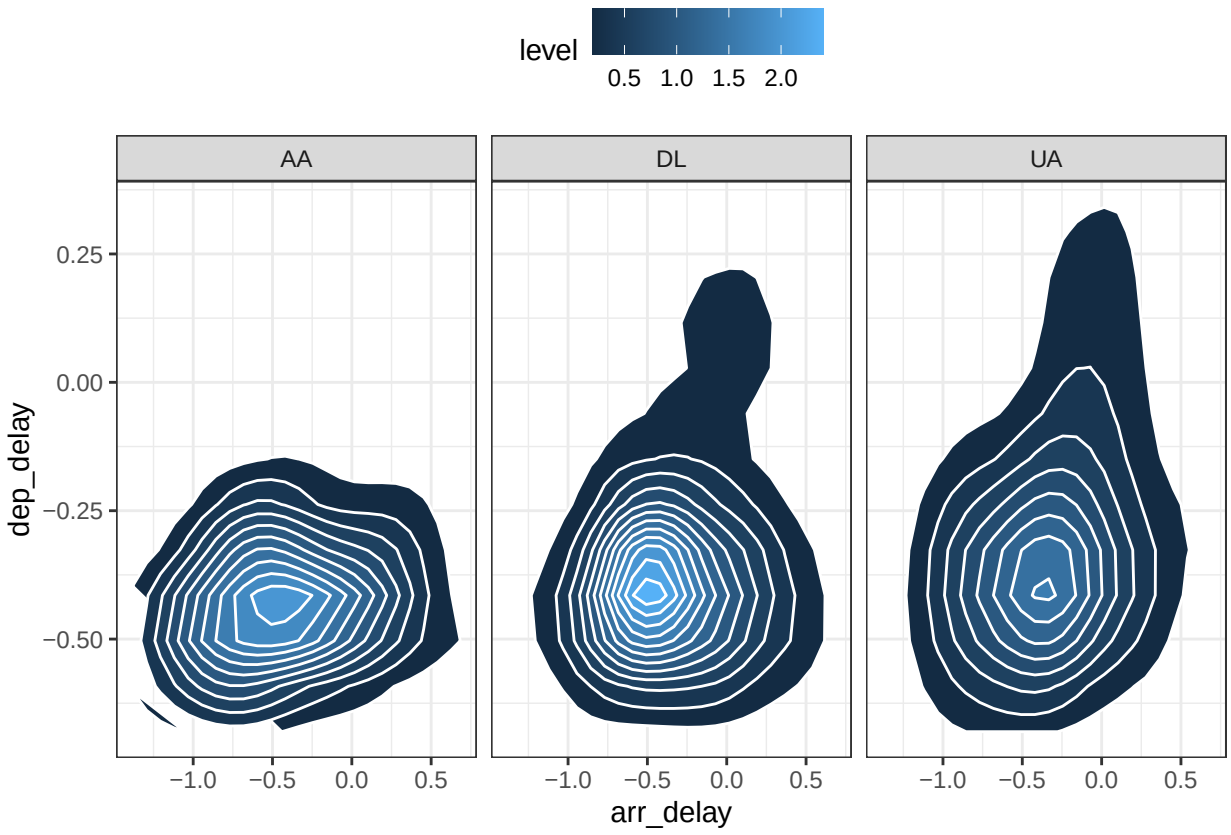
The contour density plots below show that the Gaussian assumption is violated for this dataset. The variable pairs arr_delay/distance and dep_delay/distance are both bimodal distributions for all three classes, violating the Gaussian assumption.

```
library(MASS)
```

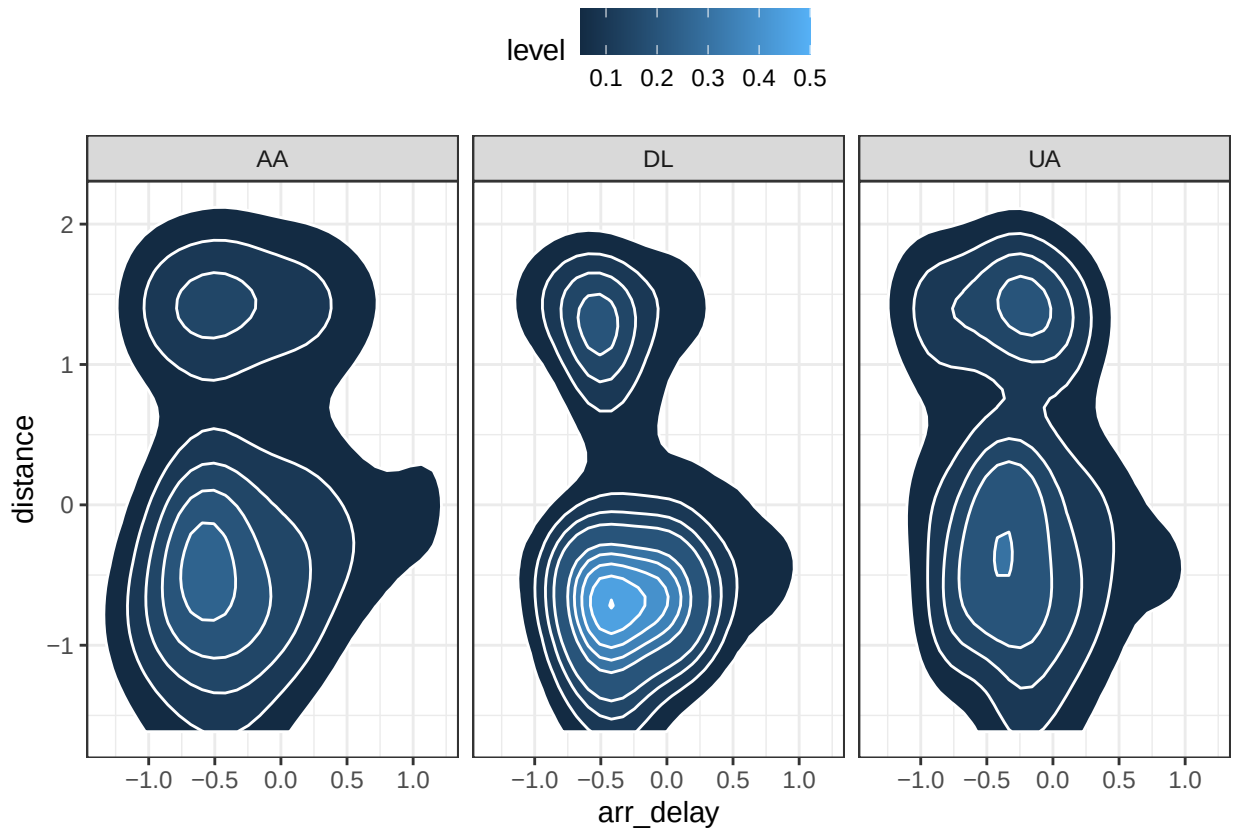
```
##
## Attaching package: 'MASS'
```

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

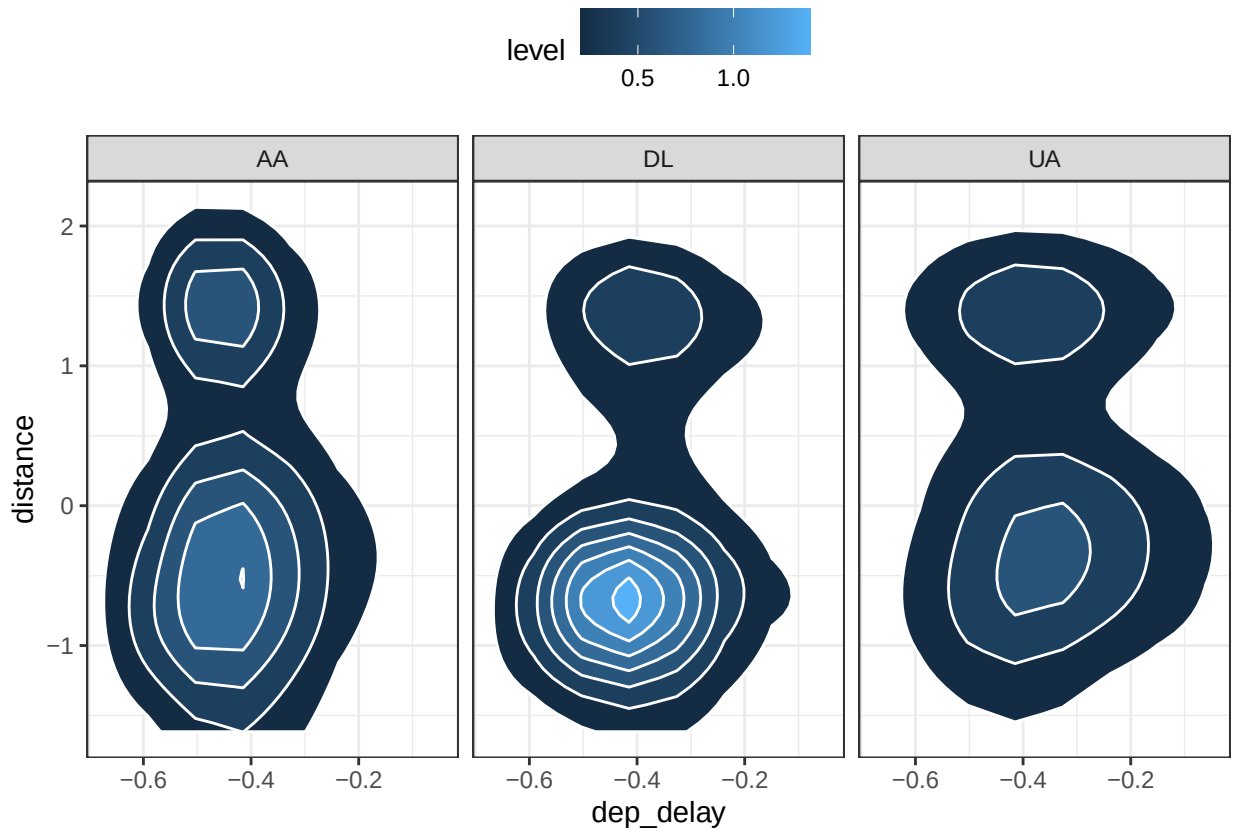
```
p1 <- ggplot(df.sample, aes(arr_delay, dep_delay)) +
  theme_bw() +
  stat_density_2d(aes(fill = after_stat(level)),
                  geom = "polygon", colour = "white") +
  theme(legend.position = "top", legend.direction = "horizontal")


p2 <- ggplot(df.sample, aes(arr_delay, distance)) +
  theme_bw() +
  stat_density_2d(aes(fill = after_stat(level)),
                  geom = "polygon", colour = "white") +
  theme(legend.position = "top", legend.direction = "horizontal")


p3 <- ggplot(df.sample, aes(dep_delay, distance)) +
  theme_bw() +
  stat_density_2d(aes(fill = after_stat(level)),
                  geom = "polygon", colour = "white") +
  theme(legend.position = "top", legend.direction = "horizontal")

p1 + facet_wrap(vars(carrier))
```




```
p2 + facet_wrap(vars(carrier))
```




```
p3 + facet_wrap(vars(carrier))
```



(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.

The estimated mixing proportions and mean vectors for each class along with the classification table is provided below. If you were to randomly pick an observation from each of the three features, it is approximately likely to belong to each class because the prior probabilities for each class are the same. The QDA classifier accurately predicted only 40.7% of observations correctly, which is significantly worse than the 3-NN classifier.

```
# Redefine the train/test sets to include the labels
train_set$carrier <- train_labels
test_set$carrier <- test_labels

# Calculate the mixing proportion
n_train <- dim(train_set)[1]
n_AA <- dim(filter(train_set, carrier == "AA"))[1]
n_UA <- dim(filter(train_set, carrier == "AA"))[1]
```

```r
n_DL <- dim(filter(train_set, carrier == "AA"))[1]

pi_AA <- n_AA / n_train
pi_UA <- n_UA / n_train
pi_DL <- n_DL / n_train

# Calculate mean vectors for each class
mean_AA <- colMeans(filter(train_set, carrier == "AA")[1:3])
mean_UA <- colMeans(filter(train_set, carrier == "UA")[1:3])
mean_DL <- colMeans(filter(train_set, carrier == "DL")[1:3])

mean_vectors <- as.data.frame(rbind(mean_AA, mean_UA, mean_DL))

# Fit and test the QDA Model
qda_fit <- qda(carrier ~ dep_delay + arr_delay + distance, data = train_set)
qda_pred <- predict(qda_fit, test_set)
true_label <- test_set$carrier
qda_pred_label <- qda_pred$class
qda_table <- table(qda_pred_label, true_label)
qda_accuracy <- sum(diag(qda_table)) / length(qda_pred_label)

# Calculate the performance

# Print results
print("Mixing Proportions:")
```

```
## [1] "Mixing Proportions:"
```

```r
as.data.frame(rbind(pi_AA, pi_UA, pi_DL))
```

```
##              V1
## pi_AA 0.2028571
## pi_UA 0.2028571
## pi_DL 0.2028571
```

```r
print("Mean Vectors: ")
```

```
## [1] "Mean Vectors: "
```

```r
print(mean_vectors)
```

```
##            dep_delay    arr_delay    distance
## mean_AA -0.192810123 -0.198740333  0.06522608
## mean_UA  0.020773138  0.006349054  0.18196685
## mean_DL  0.001617649  0.063369552 -0.23528068
```

```r
print("QDA Results Table: ")
```

```
## [1] "QDA Results Table: "
```

```
qda_table
```

```
##                true_label
## qda_pred_label AA DL UA
##              AA  0  0  4
##              DL 17 30 36
##              UA 13 19 31
```

```
print("QDA Performance: ")
```

```
## [1] "QDA Performance: "
```

```
qda_accuracy
```

```
## [1] 0.4066667
```

(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.

The classier as applied to the 2 class case had a test accuracy of 52.5%. The ROC curve is provided below, and the AUC has a value of .526. A classifier that performs no better than chance is expected to have an AUC of 0.5, so this classifier performed marginally better than chance and had an overall poor performance. This is reflected in the ROC curve, which approximates a straight line through the center of the plot. This trend is what we would expect if the predictors weren't associated with the response which seems to be the case and is confirmed by the performance of the classifier.

```
library(ROCR)

rocplot = function(pred, truth, ...){
  predob = prediction(pred, truth)
  perf = performance(predob, "tpr", "fpr")
  plot(perf, colorize = TRUE, ...)
}

# Create the subsetted training/testing sets
train_set_2 <- train_set %>% filter(carrier %in% c("UA", "DL"))
test_set_2 <- test_set %>% filter(carrier %in% c("UA", "DL"))

train_set_2$carrier <- as.character(train_set_2$carrier)
test_set_2$carrier <- as.character(test_set_2$carrier)

qda_fit <- qda(carrier ~ dep_delay + arr_delay + distance, data = train_set_2)
qda_pred <- predict(qda_fit, test_set_2)
true_label <- test_set_2$carrier
qda_pred_label <- qda_pred$class
qda_table <- table(qda_pred_label, true_label)
```

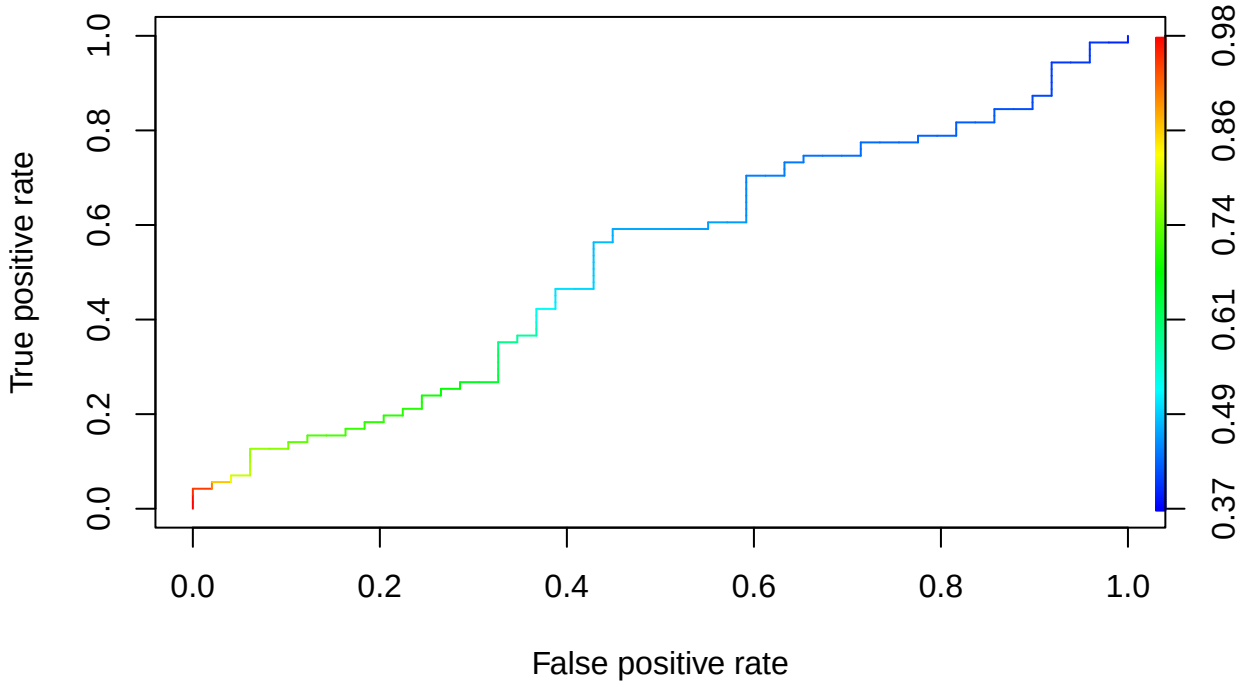
```
qda_accuracy <- sum(diag(qda_table)) / length(qda_pred_label)

qda_table
```

```
##                true_label
## qda_pred_label DL UA
##              DL 30 38
##              UA 19 33
```

```
qda_accuracy
```

```
## [1] 0.525
```

```
rocplot(qda_pred$posterior[,2], true_label)
```




```
aucStdThr <- ROCR::prediction(qda_pred$posterior[,2], true_label) %>%
  ROCR:: performance(measure = "auc") %>%
  .@y.values
as.numeric(aucStdThr)
```

```
## [1] 0.5265881
```

Applied exercises: II (Discriminant analysis)

8. 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' "?

The LDA warning that variables are collinear is indicating that some subset of variables are strongly related to each other. That is, as one variable changes, the other(s) does too.

The QDA error is indicating that one or more of the classes have too few observations relative to the number of features to define the boundary. This is especially prominent in high dimensional cases.

(8.2) Provide details on the `list` that `predict{MASS}` returns. This function returns the classication results and the posterior probabilities for each class. For each observation, the classification result is class for which the posterior probability is maximized. The posterior probabilities are returned in the form of a matrix of the observations and classes, where the values are the posterior probability of an observation belonging to a class.

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

While cross-validation is typically used to determine the optimal value of gamma and lambda, rda{klaR} does allow the user to specify them or leave them to their default values.

9. 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 60 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.

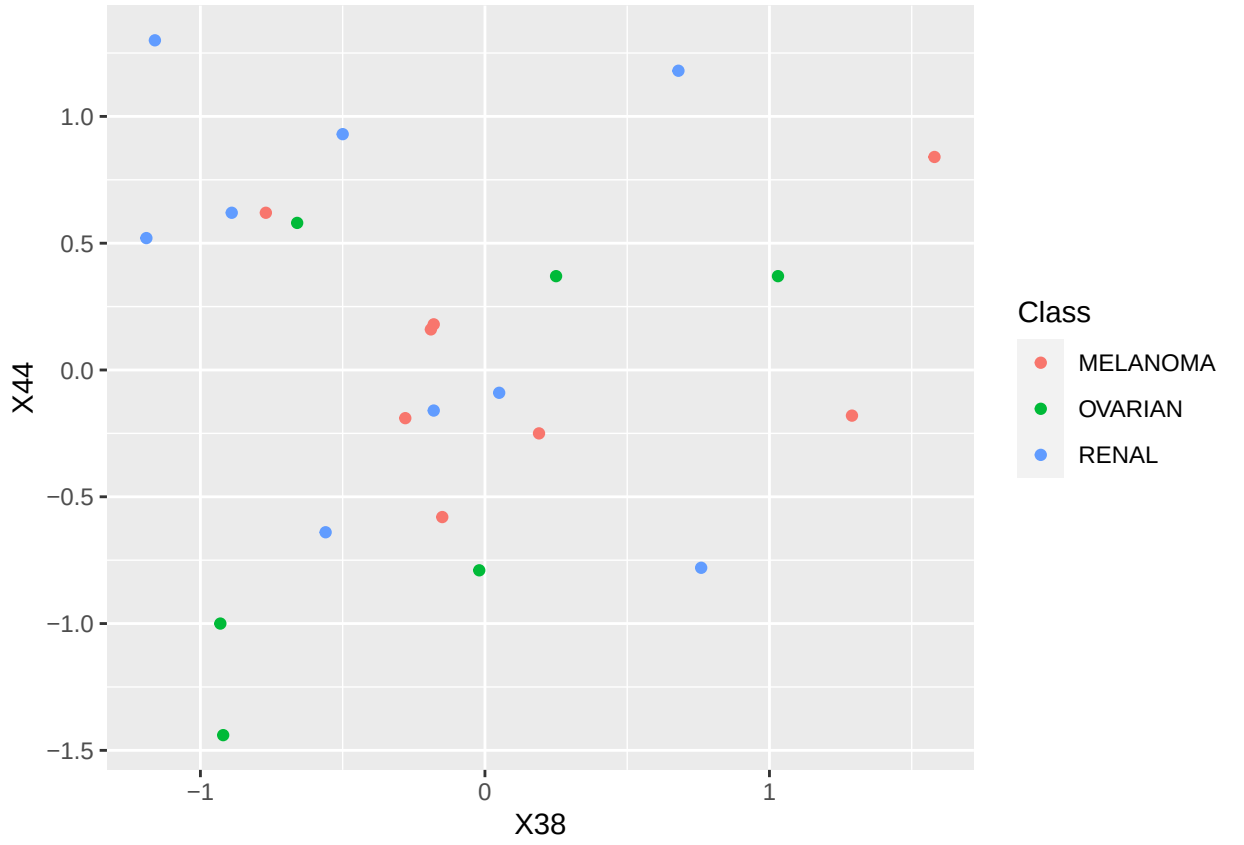
```
#install.packages("C:/Users/patri/Desktop/nci_data/ElemStatLearn_2015.6.26.2.tar.gz", type = "
library(ElemStatLearn)
data(nci)
set.seed(123)

# Subset the data
r_sel <- sample(1:nrow(nci), 60)
check <- colnames(nci) %in% c("MELANOMA", "OVARIAN", "RENAL")
c_sel = which(check == TRUE)
cancer_df <- nci[r_sel, c_sel]

# Transpose the data
tmp <- data.frame(t(cancer_df))
tmp$Class = colnames(cancer_df)
rownames(tmp) = NULL
cancer_df <- tmp
```

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".

There is significant overlap among the classes in this 2D representation, but it's important to note

that this dataset has 60 dimensions. While classification would be very difficult using the 2 features under consideration, it may be that in the high dimensional space the neighborhoods are isolated enough to make classification possible.

```
ggplot(cancer_df, aes(X38, X44, color = Class)) +
  geom_point()
```



9.2) Apply LDA and report the classwise error rate for each cancer type. The classwise accuracies as shown below are as follows: Melanoma: 75% Ovarian: 83.3% Renal: 100%

Note that these are training errors.

```
lda_fit <- lda(Class ~ ., data = cancer_df)
```

```
## Warning in lda.default(x, grouping, ...): variables are collinear
```

```
lda_pred <- predict(lda_fit, cancer_df)
true_label <- cancer_df$Class
pred_label <- lda_pred$class
table(pred_label, true_label)
```

```
##            true_label
## pred_label MELANOMA OVARIAN RENAL
##   MELANOMA        6       0     0
##   OVARIAN         0       5     0
##   RENAL           2       1     9
```

```
mel_error <- 6/8
ovarian_error <- 5/6
renal_error <- 1

as.matrix(rbind(mel_error, ovarian_error, renal_error))
```

```
##                   [,1]
## mel_error     0.7500000
## ovarian_error 0.8333333
## renal_error   1.0000000
```

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 = cancer_df, gamma = 0.05, lambda = 0.2)
rda_pred <- predict(rda_fit, cancer_df)
rda_true_label <- cancer_df$Class
rda_pred_label <- rda_pred$class
table(rda_pred_label, rda_true_label)
```

```
##               rda_true_label
## rda_pred_label 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 alphabetically by R. So, you need to keep track on which estimated covariance matrix is for which class. Otherwise, you will get wrong visualization.)

For all cancer types there is a great deal of interdependence among genes, but Leukemia and Renal in particular share a set of very strongly interdependent genes. Genes X60, X49, and X34 are related to most other genes for all three cancer types as well.

```
library(reshape2)
```

```
##
## Attaching package: 'reshape2'
```

```
## The following object is masked from 'package:tidyr':
##
##     smiths
```

```
hatSigma1 <- rda_fit$covariances[,,1]
hatSigma2 <- rda_fit$covariances[,,2]
hatSigma3 <- rda_fit$covariances[,,3]
```

```r
melted_hatSigma1 <- melt(hatSigma1)
melted_hatSigma2 <- melt(hatSigma2)
melted_hatSigma3 <- melt(hatSigma3)
EstSigmaAll <- rbind(melted_hatSigma1, melted_hatSigma2, melted_hatSigma3)
EstSigmaAll$Cancer <- rep(c("1. MELANOMA", "2. LEUKEMIA", "3. RENAL"),
                          each = nrow(melted_hatSigma1))
EstSigmaAll$Cancer <- factor(EstSigmaAll$Cancer)

ggplot(data = EstSigmaAll, aes(Var1, 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))
```