Homework 4

STAT 437 Spring 2022

Christopher Mims
(10436827)

April 17, 2022

Conceptual Exercises

Bayes Classifier

The following are answers to questions that pertain to the Bayes theorem and the Bayes classifier.

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

The 0-1 (zero-one) loss function is quite simple, as it assigns a loss value of zero for every correct classification and assigns a loss value of one for every incorrect classification of observations in a test set. This loss function is best used to find the accuracy of a classification model with the 0-1 weights. The 0-1 loss function can be used in multi-class classification problems as well. It can still calculate the loss of a classification algorithm by assigning the value one to every misclassified observation. Again, if the weights were changed to favor one class over another, then the 0-1 loss function would still measure the loss.

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 formula to obtain the posterior probability of $Y = k$ given $X = x$ is as follows:

$$\Pr(Y = k \mid X = x) = \frac{\pi_k f_k(x)}{\sum_{l=1}^{K} \pi_l f_l(x)}$$

Here we have the posterior probability of when Y is equal to k, a classification is from a specified class, given X is equal to x, when an observation is from a set of observations, is calculated by dividing the probability of x given the class k, times the probability of the given class within the set of classes, divided by the probability of the observation within the set of observations. In order to find the posterior probability, we must know the probability of the observation given the specified class. The Bayes classifier uses the posterior probabilities of an observation, x_0 from X, within each class of the set of classes, $Y \in \mathcal{G}$. The classifier then assigns the observation to the class with the largest posterior probability. The Bayes classifier has a linear decision boundary between each of the possible classes. This linear boundary is considered a hyperplane in high dimensional space. The Bayes classifier inherently minimizes the 0-1 loss due to the fact that it classifies based off of the highest probability for each class. Therefore, since correct classifications receive a score of zero, and the classification is based off of the highest probability, more classifications will be correct leading, to a minimized 0-1 score.

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.

The threshold value on that is used by the Bayes classifier for the determination of the class label for x_0 if $K = 2$ would be $\Pr(Y = 1 \mid X = x_0) > 0.5$. If, instead of using 0.5 as the determination value, another value was used to classify x_0, the classifier would no longer be a Bayes classifier, but rather a Linear Discriminant Analysis classifier. As stated above, the loss function would still be the 0-1 loss because it would still be assigning a value of zero to all correctly classified observations and one to all incorrectly classified observations. Though instead of trying to minimize the overall error rate, we would be trying to minimize the overall cost of the misclassified observations. As an example, if we wanted to classify students students into 'dog lovers' and 'cat lovers' but wanted to minimize the amount of students misclassified as 'dog lover' when they were really a 'cat lover' (since we know that people who love cats will let you know). Then we could change the value of our classifier to ensure more 'dog lovers' were misclassified as 'cat lovers', reducing the amount of times a cat lover has to correct us. (BTW: I like cats, I'm just super allergic, so by default I am a big dog lover!)

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.

```r
# `mtvnorm` package used to compute the multivariate Gaussian probabilities
library(mvtnorm)

# Create a matrix to hold posterior probability values
postProb.results <- matrix(0, ncol = 2, nrow = 1)
# Calculate posterior probabilities for x_0 = 1.5
# f_1(x) = ~ Gaussian(0,1) and f_2(x) ~ Gaussian(2,1)
cndp1 <- dmvnorm(x = 1.5, mean = c(0, 1), sigma = diag(2))
cndp2 <- dmvnorm(x =1.5, mean = c(2, 1), sigma = diag(2))
fmarginal <- 0.6 * cndp1 + 0.6 * cndp2
jp1 <- 0.6 * cndp1
jp2 <- 0.6 * cndp2
postp1 <- jp1/fmarginal
postp2 <- jp2/fmarginal
postProb.results[i,] <- c(postp1, postp2)

colnames(postProb.results) <- c('PPCL1', 'PPCL2')
postProb.results
```

k-NN Classifier

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.

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

With k-nearest neighbors classification, its decision boundary is overly flexible the lower the value of k. As the value of k increases, the decision boundary becomes less flexible. Figure 2.16 in *An Introduction to Statistical Learning* shows the difference between a model that uses $K = 1$ and then uses $K = 100$. When $K = 1$, the decision boundary is very jagged and chaotic, whereas the decision boundary for $K = 100$ is much smoother, and almost forms a straight line.

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?

The reason the training error of 1-NN classifier is 0, is due to the fact that the model is able to over-fit the data with an overly flexible decision boundary. This error rate does not transfer over to the test data. When looking at the test error rate, we can use $\frac{1}{K}$ to estimate the value of the test error rate. As $\frac{1}{K}$ increases, as $K \to 1$, the decision boundary becomes more flexible and a lower *training error rate* is obtained, but a higher *test error rate* is observed. But as $K \to \infty$, or K increases and $\frac{1}{K}$ decreases, the test and training error rates increase. Many times, the test error rate, when plotted, looks like a U. Therefore, it would not be true that a large value for K would result in a smaller test error value. If the correct value for K is chosen, a kl-NN classifier can have the same test error value as that of a Bayes classifier, as they both base their classification off assigning an observation to the class with the highest probability. When k is large and k/n is small, the k-NN classifier is approximately estimating true classification.

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

The k-NN classifier uses the Bayes rule and classifies the test observation x_0 to the class with the highest probability. For example, if $K = 3$ and there are 2 classes, if 2 out of the 3 neighbors belong to one class, then the probability that the test observation would belong to the class containing those observations would be $\frac{2}{3} = 66.67\%$.

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 for a k-NN classifier if the two units of measure vary greatly. If we were to use a measure of inches compared to miles, or minutes compared hours spend on homework, we would need to standardize the data. Also, if the varying units of measure rely on the context of the problem at hand. As the book gave the example of salary measured in thousands of dollars and age in years. We know my the context that an increase of $1,000 is way less impactful that an increase of 50 years in one's life. When we are standardizing the data, we want to make sure that we are standardizing the features and not the observations.

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 a 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.)

When a dataset has few observations in the training set, it would be wise to use cross-validation in order to choose an optimal k. This is due to the fact that when using cross-validation, we are able to supply the algorithm with different parts of the data, allowing for any adjustments due to the data's variance. If the data has widely variable variance, then the algorithm can become unstable. Therefore when running cross-validation, we can account for this and choose a k value that is best suited for the data at hand. As a dataset with few observations is not suited for k-NN models, using cross-validation will minimize the test errors.

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".)

Algorithm 1 Optimal k for k-NN Classification

```
1: Randomly split $n$ observations into $m$ folds of approximately equal size
2: Pick fold $s$ as *test* set
3: Set remaining folds as *training* set
4: for each $k$ in $\mathcal{C} = \{1, \dots, m\}$ do
5:     for each fold $s$ in *training* set do
6:         Apply the $k$-NN classifier
7:         Obtain each test error $e_s$
8:     end for
9: end for
10: for each value of $k$ do
11:     Compute sample mean $\hat{\mu}(k, m)$ of $e_s$
12:     Compute sample standard deviation $\hat{\sigma}(k, m)$ of $e_s$
13: end for
14: Obtain $k$ for the minimum value of $\hat{\mu}(k, m)$
```

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".)

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.)

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 $Y \in \mathcal{G} = \{1, \dots, K\}$ 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 $(x_1, y_1), \dots, (x_n, y_n)$ on (X, Y), where y_i is the class label of observation x_i, do the following:

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.

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?

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

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

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.

Applied Exercises

k-NN Classifier

For this section, we will be using the `nycflights13` data set we have used in previous homework assignments. We will use `set.seed(123)` for the entirety of this exercise. First we will select from `flights` for each of the 3 `carrier` "UA", "AA", and "DL" 500 random observations for the 3 features `dep_delay`, `arr_delay`, and `distance`. With this data, we will see if we can identify which carrier to which an observation belongs. **Note:** observations with `na`s will be extracted from the observations.

```
library(tidyr)
library(dplyr)
library(ggplot2)
library(nycflights13)
library(class)

# Subset `flights` data set
q7 <- select(flights, carrier, dep_delay, arr_delay, distance) %>%
          dplyr::filter(carrier %in% c("UA", "AA", "DL"))
# Set seed for reproducible results
set.seed(123)
# Randomly choose 500 samples
q7 <- sample_n(q7, 500)
# Remove `na` values from subset
q7 <- na.omit(q7)
```

After obtaining the features and observations, we will need to standardize the features, due to their varying scales, and then we will split the observations into a training set and test set. For the training set, we will use 70% of the observations, leaving the remaining observations for the test set.

```
# Standardize the features
q7.1 <- as.data.frame(scale(q7[,2:4]))
q7.1$carrier <- q7$carrier
q7.1 <- q7.1 %>% mutate(label = case_when(
                          carrier == "AA" ~ 1,
                          carrier == "DL" ~ 2,
                          carrier == "UA" ~ 3))
q7.1 <- subset(q7.1, select = -carrier)

# Set seed for reproducible results
set.seed(123)

# Subset data into training and test sets
r.train <- base::sample(1:nrow(q7.1), 0.7 * nrow(q7.1))
r.test <-  (1:nrow(q7.1))[-r.train]
q7.train <- q7.1[r.train,]
q7.test <- q7.1[r.test,]
```

If we consider the observations as forming 3 classes that are determined by `carrier`, we can apply 10-fold cross-validation to the k-NN classifier with features `arr_delay`, `dep_delay`, and `distance` to determine the optimal k from the values $\{1, \ldots, 15\}$. After the optimal value for k is found, the optimal k-NN model will be applied to the test set. A classification table, overall error rate, and a visualization of the classification results will be provided.

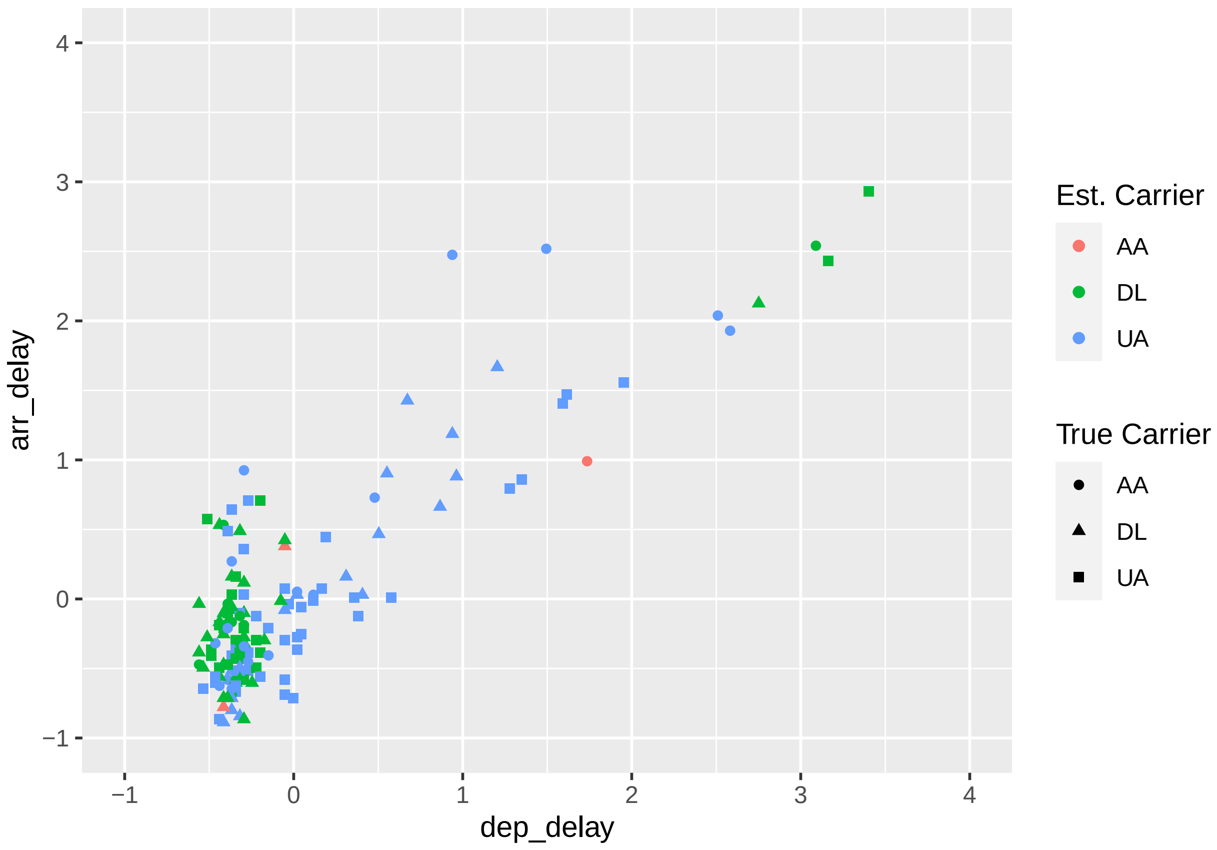
```r
# Number of folds
m <- 10

# Set seed for reproducible results
set.seed(123)

# Create folds
folds <- sample(1:m, nrow(q7.train), replace = TRUE)

# Create vector to hold test error values
test.errors <- as.data.frame(matrix(nrow = 15, ncol = 10))

# Apply cross validation for k in {1, ..., 15}
for (k in 1:15) {
  for (s in 1:m) {
    training.temp <- q7.train[folds != s,]
    test.temp <- q7.train[folds == s,]
    train.labs.temp <- training.temp[, 4]
    test.labs.temp <- test.temp[, 4]
    knnX <- knn(training.temp[, 1:3],
                test.temp[, 1:3],
                train.labs.temp, k)
    num.misclass.obs <- sum(1 - as.numeric(knnX == test.labs.temp))
    test.error <- num.misclass.obs / length(test.labs.temp)
    test.errors[k, s] <- test.error
  }
}

# Calculate mean and standard deviation of each value for k
test.errors$mean <- rowMeans(test.errors)
test.errors$sd <- apply(test.errors[1:10], 1, sd)

# Choose optimal k value (lowest test error)
optimal.k <- as.numeric(rownames(test.errors[which.min(test.errors$mean),]))

# Apply optimal k value to test set
q7.test$knn <- knn(q7.train[, 1:3],
                   q7.test[, 1:3],
                   q7.train[, 4],
                   optimal.k)

# Prepare for displaying results
carrierLab <- c("AA", "DL", "UA")
q7.test$`True Carrier` <- factor(q7.test$label, label = carrierLab)
q7.test$`Est. Carrier` <- factor(q7.test$knn, label = carrierLab)
# Calculate error rate
q7.error.rate <- sum(1 - as.numeric(q7.test$knn == q7.test$label)) /
                   length(q7.test$label)

# Create table of true vs estimated labels
`True Carrier` <- q7.test$`True Carrier`
```

```
`Est. Carrier` <- q7.test$`Est. Carrier`
table(`Est. Carrier`, `True Carrier`)
```

```
##             True Carrier
## Est. Carrier AA DL UA
##           AA  1  2  0
##           DL 10 32 24
##           UA 18 17 44
```

Overall error rate with value of 13 for k in k-NN model: 0.4797297

```
# Graph the estimated class labels against true class labels
q7.1.plot <- ggplot(q7.test, aes(x = dep_delay, y = arr_delay)) +
               geom_point(aes(color = `Est. Carrier`, shape = `True Carrier`)) +
               scale_x_continuous(limits = c(-1, 4)) +
               scale_y_continuous(limits = c(-1, 4))
q7.1.plot
```



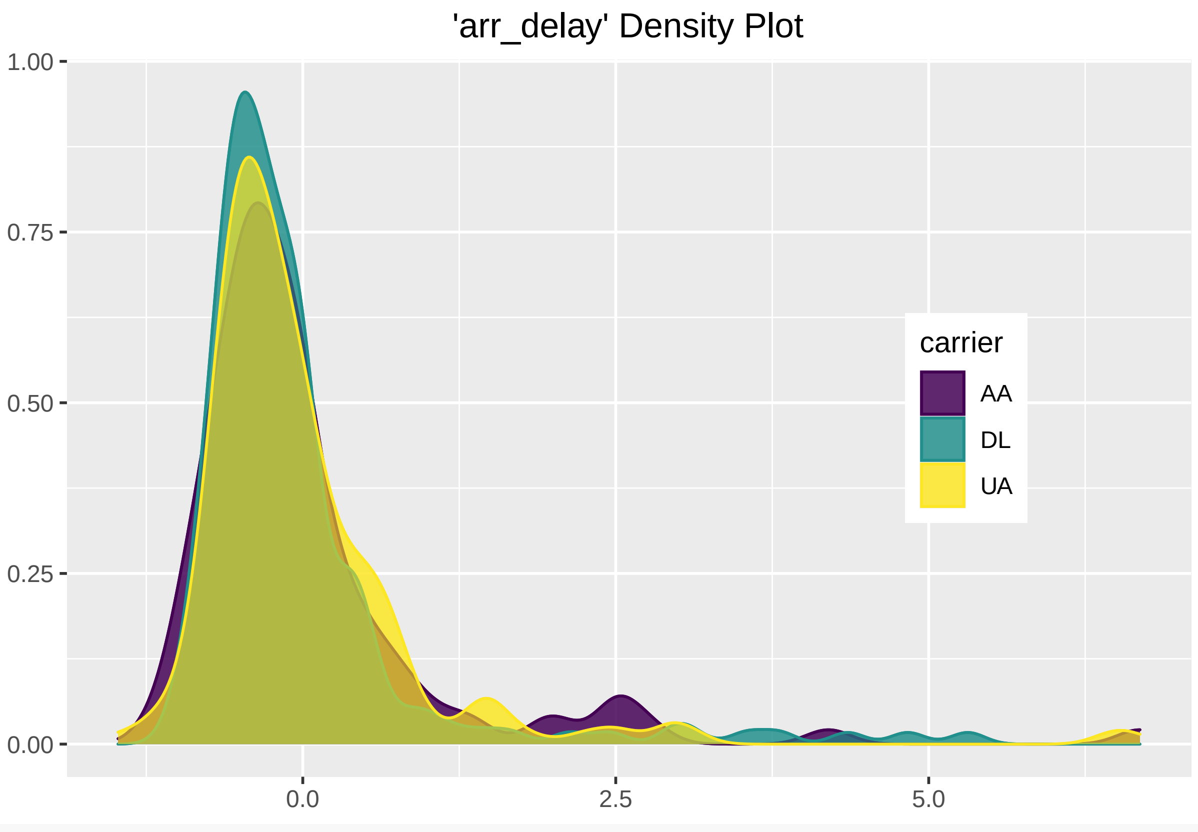
For this data, I believe that the error rate produced by the k-NN model is reasonable. From analysis of this data in previous homework assignments, there is very little difference between the data of each carrier. There were no clear observations to distinguish much between each carrier but rather between the time of year that `arr_delay` and `dep_delay` was effected.

Discriminant Analysis

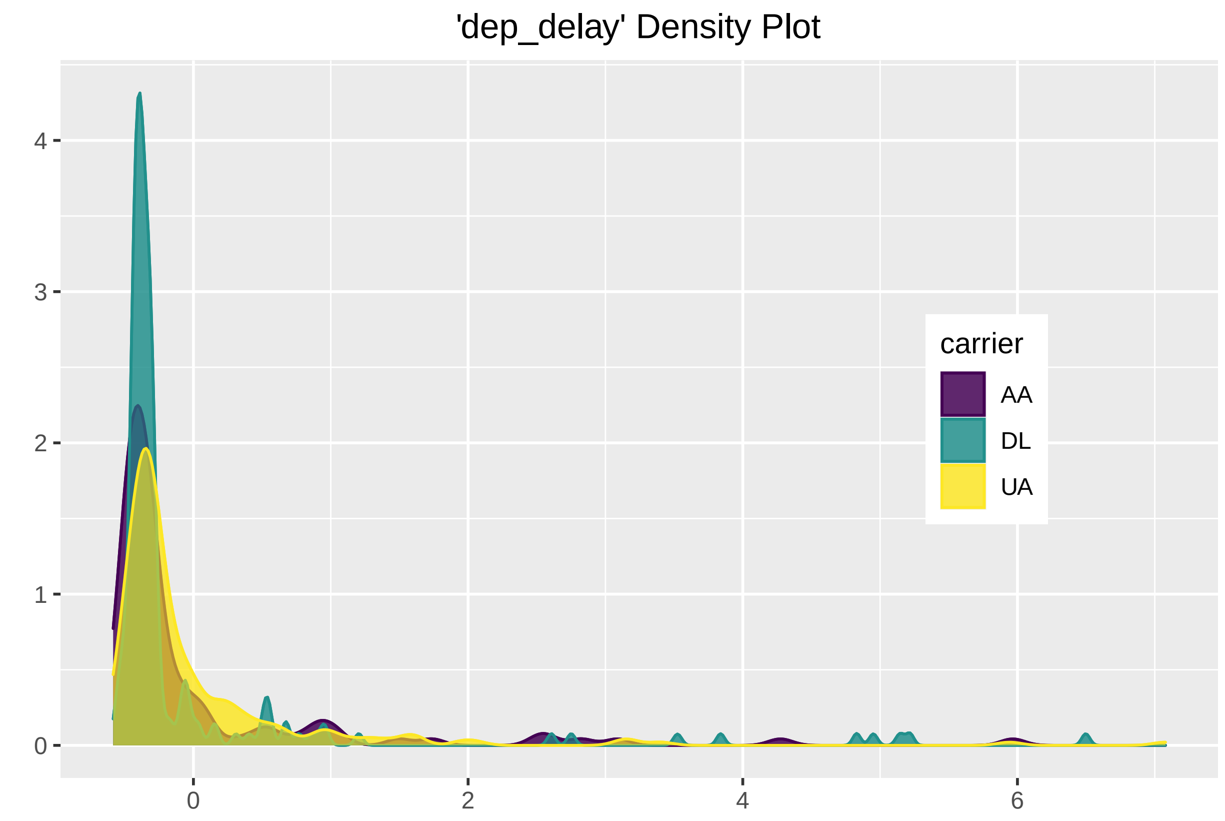
For the next part of the applied exercises we will apply a quadratic discriminant analysis model to this data in order to obtain classification. The results will be analyized and comparted to that of the k-NN model.

```
# Cool new libraries to use for plotting features (color themes)
library(hrbrthemes)
library(viridis)
library(gridExtra)

# Scale the data since I didn't create it correctly earlier
q7.3 <- as.data.frame(scale(q7[,2:4]))
q7.3$carrier <- q7$carrier
```

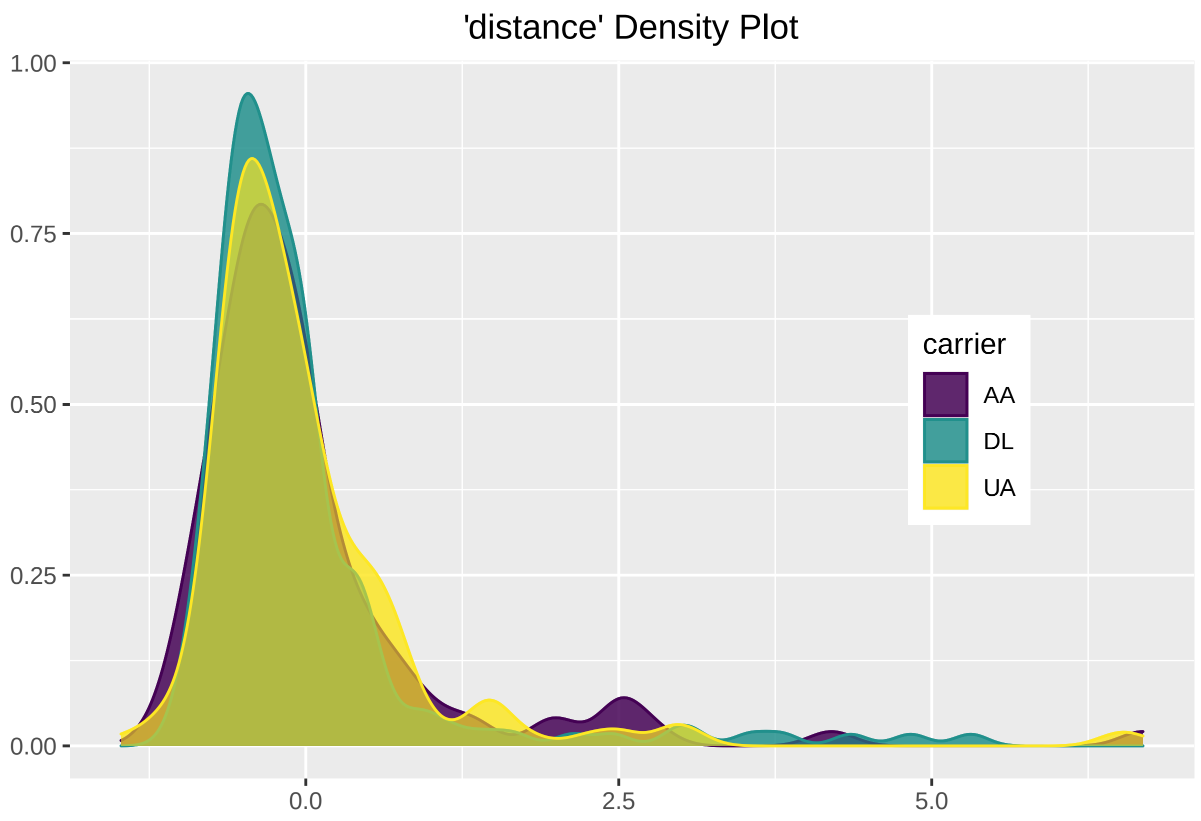
```
q7.arr_delay.plt <- ggplot(q7.3, aes(x = q7.3$arr_delay,
                                     color = carrier,
                                     fill = carrier)) +
                      geom_density(alpha = 0.6) +
                      geom_density(alpha = 0.6) +
                      scale_fill_viridis(discrete = TRUE) +
                      scale_color_viridis(discrete = TRUE) +
                      theme(legend.position = c(0.8, .5),
                            plot.title = element_text(hjust = 0.5)) +
                      ggtitle("'arr_delay' Density Plot") +
                      ylab("") + xlab("")
q7.arr_delay.plt
```



```
q7.dep_delay.plt <- ggplot(q7.3, aes(x = q7.3$dep_delay,
                                     color = carrier,
                                     fill = carrier)) +
                      geom_density(alpha = 0.6) +
                      geom_density(alpha = 0.6) +
                      scale_fill_viridis(discrete = TRUE) +
                      scale_color_viridis(discrete = TRUE) +
                      theme(legend.position = c(0.8, .5),
                            plot.title = element_text(hjust = 0.5)) +
                      ylab("") + xlab("") +
                      ggtitle("'dep_delay' Density Plot")
q7.dep_delay.plt
```

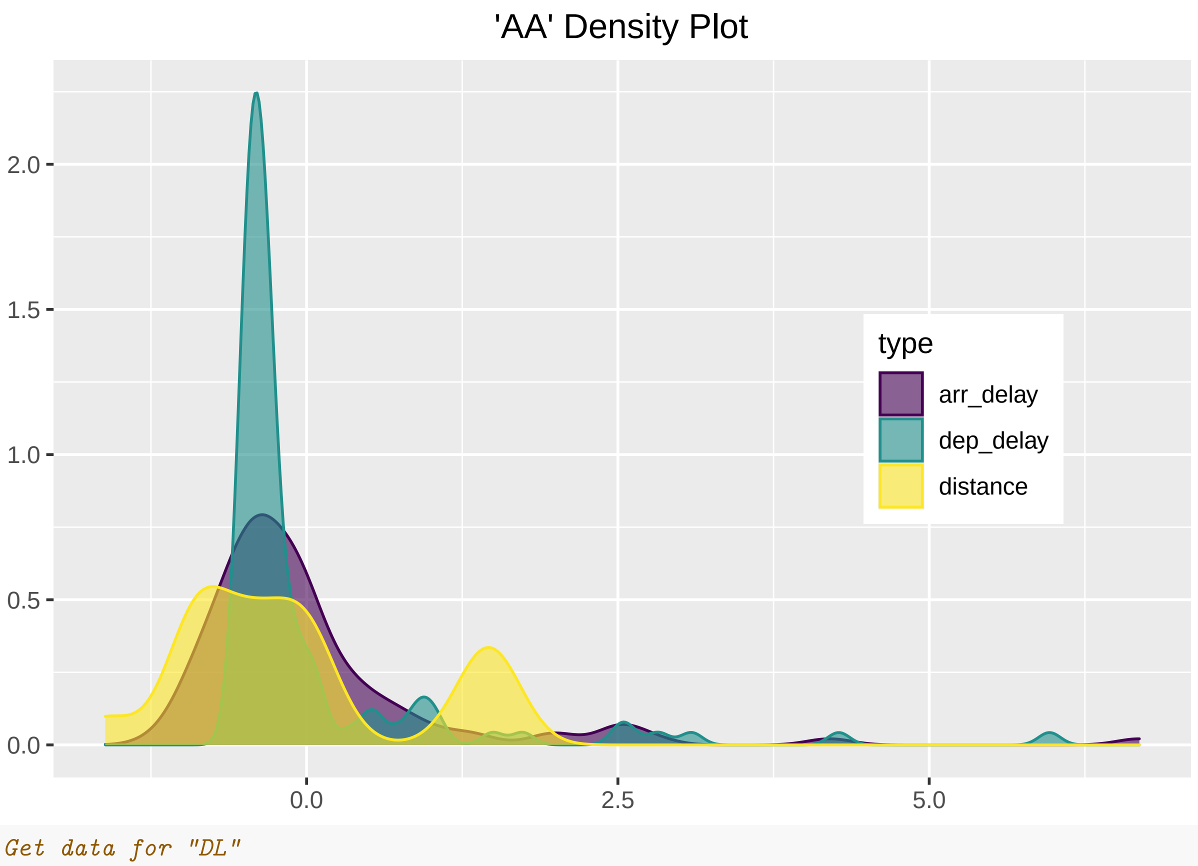


```
q7.distance.plt <- ggplot(q7.3, aes(x = q7.3$arr_delay,
                                    color = carrier,
                                    fill = carrier)) +
                    geom_density(alpha = 0.6) +
                    geom_density(alpha = 0.6) +
                    scale_fill_viridis(discrete = TRUE) +
                    scale_color_viridis(discrete = TRUE) +
                    theme(legend.position = c(0.8, .5),
                          plot.title = element_text(hjust = 0.5)) +
                    ylab("") + xlab("") +
                    ggtitle("'distance' Density Plot")
q7.distance.plt
```



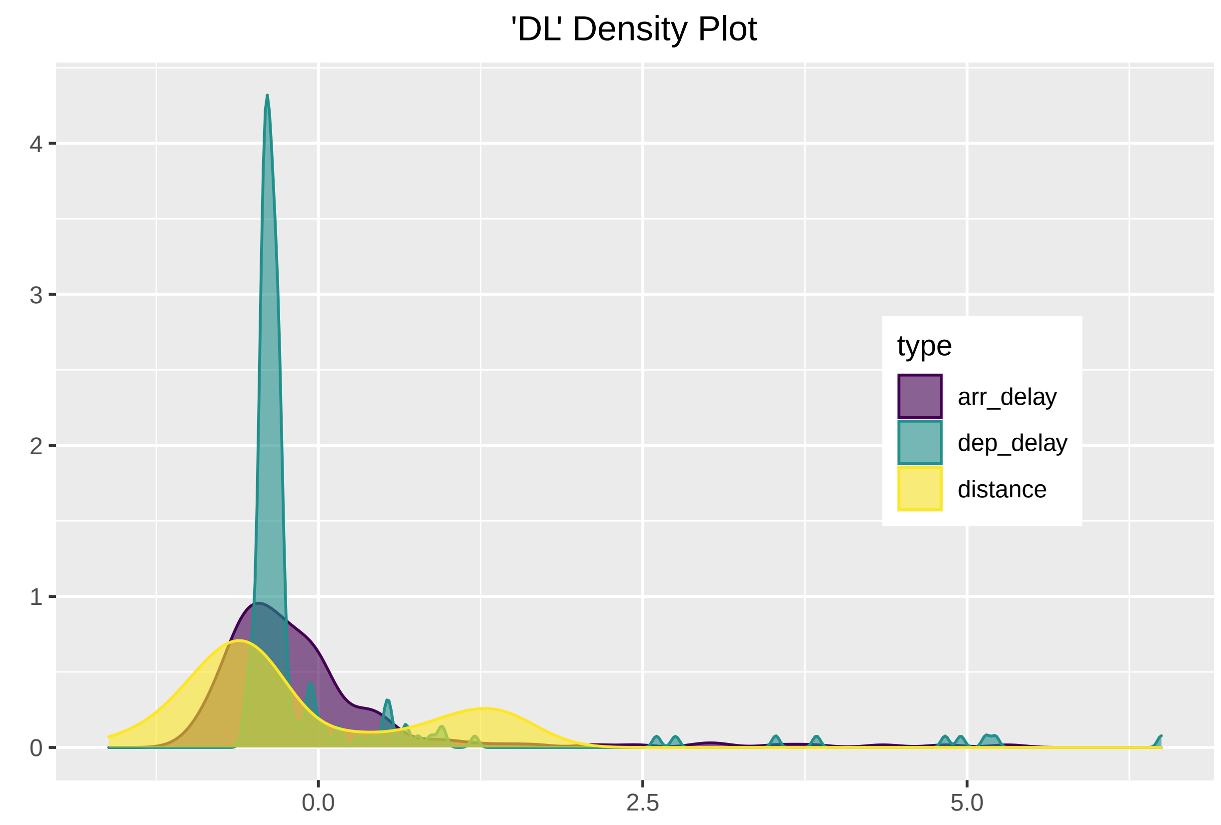
From the three previous plots, we can see three (3) distinct curves, one for `carrier`. This shows that the data for `distance` for all `carriers` follows a trivariate Gaussian distribution. Had all curves been exactly the same, we would not have accurate results from the QDA model.

```
# Get data for "AA"
aa.arr_delay <- as.data.frame(q7.3$arr_delay[q7.3$carrier == "AA"])
aa.arr_delay$type <- c(rep("arr_delay", length(aa.arr_delay)))
colnames(aa.arr_delay) <- c("value", "type")
aa.dep_delay <- as.data.frame(q7.3$dep_delay[q7.3$carrier == "AA"])
aa.dep_delay$type <- c(rep("dep_delay", length(aa.dep_delay)))
colnames(aa.dep_delay) <- c("value", "type")
aa.distance  <- as.data.frame(q7.3$distance[q7.3$carrier == "AA"])
aa.distance$type <- c(rep("distance", length(aa.distance)))
colnames(aa.distance) <- c("value", "type")

# Crate data.frame for values of "AA"
aa.data <- rbind(aa.arr_delay, aa.dep_delay, aa.distance)

# Create density plot for each feature `arr_delay`, `dep_delay`, and `distance`
q7.3a.aa.dens <- ggplot(aa.data, aes(x = value,
                                     color = type,
                                     fill = type)) +
                     geom_density(alpha = 0.6) +
                     scale_fill_viridis(discrete = TRUE) +
                     scale_color_viridis(discrete = TRUE) +
                     theme(legend.position = c(0.8, .5),
                           plot.title = element_text(hjust = 0.5)) +
                     ylab("") + xlab("") +
                      ggtitle("'AA' Density Plot")
q7.3a.aa.dens
```

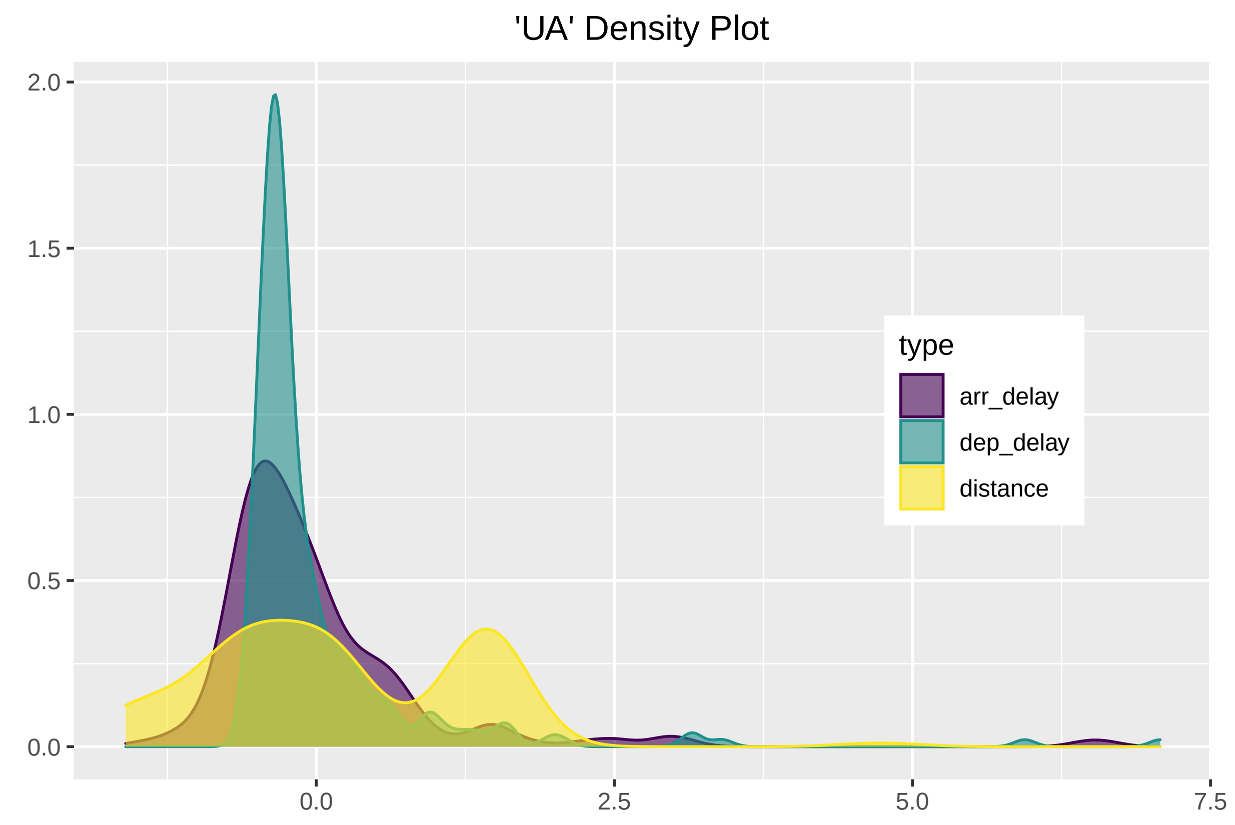


```
# Get data for "DL"
dl.arr_delay <- as.data.frame(q7.3$arr_delay[q7.3$carrier == "DL"])
dl.arr_delay$type <- c(rep("arr_delay", length(dl.arr_delay)))
colnames(dl.arr_delay) <- c("value", "type")
dl.dep_delay <- as.data.frame(q7.3$dep_delay[q7.3$carrier == "DL"])
dl.dep_delay$type <- c(rep("dep_delay", length(dl.dep_delay)))
colnames(dl.dep_delay) <- c("value", "type")
dl.distance  <- as.data.frame(q7.3$distance[q7.3$carrier == "DL"])
dl.distance$type <- c(rep("distance", length(dl.distance)))
colnames(dl.distance) <- c("value", "type")

# Crate data.frame for values of "DL"
dl.data <- rbind(dl.arr_delay, dl.dep_delay, dl.distance)

# Create density plot for each feature `arr_delay`, `dep_delay`, and `distance`
q7.3a.dl.densty <- ggplot(dl.data, aes(x = value,
                                  color = type,
                                  fill = type)) +
                      geom_density(alpha = 0.6) +
                      scale_fill_viridis(discrete = TRUE) +
                      scale_color_viridis(discrete = TRUE) +
                      theme(legend.position = c(0.8, .5),
                            plot.title = element_text(hjust = 0.5)) +
                      ylab("") + xlab("") +
                      ggtitle("'DL' Density Plot")
```

```
q7.3a.dl.densty
```



```
# Get data for "UA"
ua.arr_delay <- as.data.frame(q7.3$arr_delay[q7.3$carrier == "UA"])
ua.arr_delay$type <- c(rep("arr_delay", length(ua.arr_delay)))
colnames(ua.arr_delay) <- c("value", "type")
ua.dep_delay <- as.data.frame(q7.3$dep_delay[q7.3$carrier == "UA"])
ua.dep_delay$type <- c(rep("dep_delay", length(ua.dep_delay)))
colnames(ua.dep_delay) <- c("value", "type")
ua.distance  <- as.data.frame(q7.3$distance[q7.3$carrier == "UA"])
ua.distance$type <- c(rep("distance", length(ua.distance)))
colnames(ua.distance) <- c("value", "type")

# Crate data.frame for values of "DL"
ua.data <- rbind(ua.arr_delay, ua.dep_delay, ua.distance)

# Create density plot for each feature `arr_delay`, `dep_delay`, and `distance`
q7.3a.ua.densty <- ggplot(ua.data, aes(x = value,
                                    color = type,
                                    fill = type)) +
                     geom_density(alpha = 0.6) +
                     scale_fill_viridis(discrete = TRUE) +
                     scale_color_viridis(discrete = TRUE) +
                     theme(legend.position = c(0.8, .5),
                           plot.title = element_text(hjust = 0.5)) +
                     ylab("") + xlab("") +
                     ggtitle("'UA' Density Plot")
q7.3a.ua.densty
```



As we can see for each of these plots, that are separated out by `carrier`, any pair of feature curves would follow a bivariate Gaussian distribution, as shown by the different curves of each data type. We now know that the data follows a Gaussian distribution within each class and within each feature.

Source: https://r-graph-gallery.com/135-stacked-density-graph.html

```
# Import MASS library for QDA model
library(MASS)

# Set seed for reproducible results
set.seed(123)

# Copy training and test sets for use with QDA model
q7.3b.train <- q7.train[, 1:4]
q7.3b.test <- q7.test[, 1:4]

# Build and train QDA model
q7.3b.fit <- qda(factor(label) ~ ., data = q7.3b.train)

# Obtain predictions on the test set
q7.3b.preds <- predict(q7.3b.fit, q7.3b.test[1:3])

# Calculate estimated mixing proportion
est.mix <- t(as.data.frame(table(q7.3b.train$label)/length(q7.3b.train$label)))
colnames(est.mix) <- c("AA", "DL", "UA")
est.mix <- as.data.frame(est.mix)
est.mix <- est.mix[2,]
rownames(est.mix) <- c("mean")
```

```r
# Calculate mean vector for each class
q7.3b.test$pred <- q7.3b.preds$class
q7.3b.test$correct <- c(rep(0, length(q7.3b.test$pred)))
r.correct <- which(q7.3b.test$label == q7.3b.test$pred)
q7.3b.test$correct[r.correct] = 1
est.mean.df <- q7.3b.test %>% group_by(label, correct) %>%
                  summarise(n = n()) %>% mutate(mean = n/sum(n))
est.mean <- est.mean.df$mean[est.mean.df$correct == 1]
est.mean <- as.data.frame(est.mean)
rownames(est.mean) <- c("AA", "DL", "UA")


# Create classification table
`True Class` <- q7.3b.test$label
`Predicted Class` <- q7.3b.preds$class
q7.3b.pred.tbl <- table(`Predicted Class`, `True Class`)
colnames(q7.3b.pred.tbl) <- c("AA", "DL", "UA")
rownames(q7.3b.pred.tbl) <- c("AA", "DL", "UA")
q7.3b.pred.tbl
```

```
##                True Class
## Predicted Class AA DL UA
##              AA  1  0  1
##              DL 13 18 25
##              UA 15 33 42
```

Our estimated mixing proportions are the proportions of each class in the training set (since we don't know how many are in the test set). They are as follows:

0.2244898, 0.3527697, 0.4227405

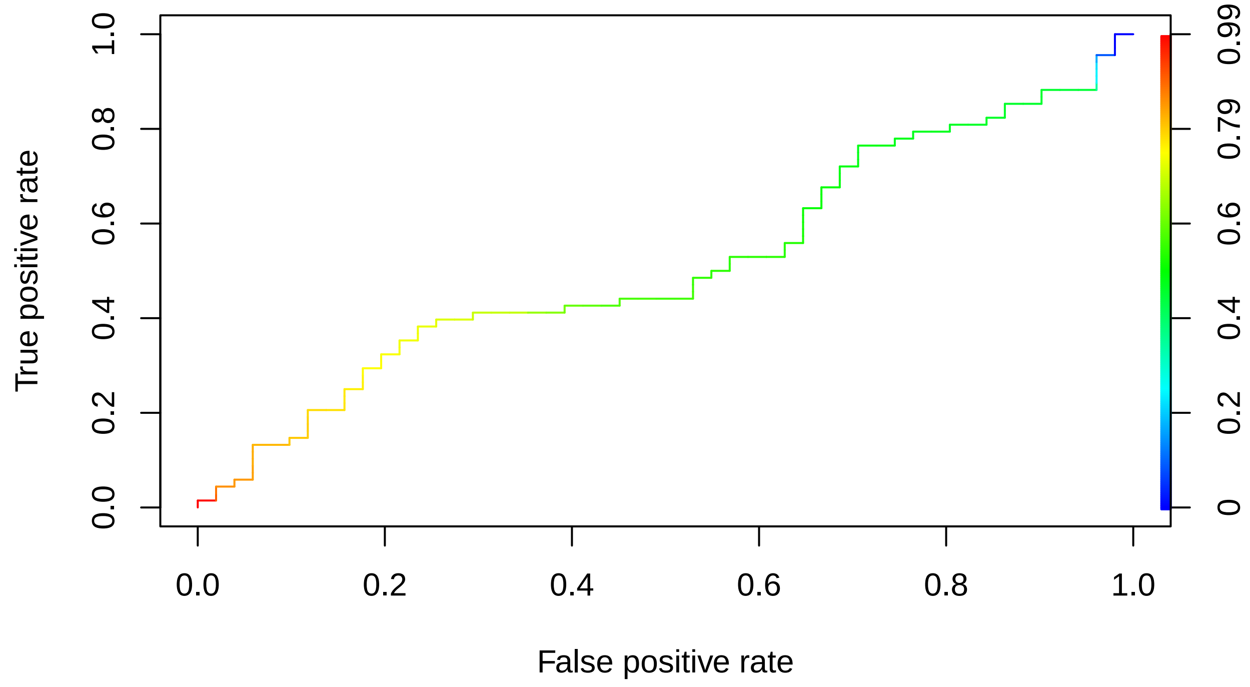
Our estimated mean vector for each class is as follows:

c(0.0344827586206897, 0.352941176470588, 0.617647058823529)

If a random observation was chosen on the three standardized features, it would equally be likely to belong to each of the three carriers. This is due to the fact that we have approximately an equal mixture of each carrier in our data set. Since we are dealing with normal distributions, all three carriers are equally likely to be chosen. This logic does not carry over to the QDA model that we have trained on this data set. This model is very unlikely to choose "AA" as a carrier and favors "UA" as a carrier. As we can see from our classification table the model only chose one observation to belong to "AA" while it chose 90 observations to be "UA".

```r
# Set seed for reproducible results
set.seed(123)

# Copy training and test sets for use with QDA model
q7.3c.train <- q7.train[, 1:4]
q7.3c.test <- q7.test[, 1:4]

# Filter training and test sets to only include carriers "DL" and "UA"
q7.3c.train <- q7.3c.train %>% filter(label %in% c(2, 3))
q7.3c.test  <- q7.3c.test %>% filter(label %in% c(2, 3))

# Build and train QDA model
q7.3c.fit <- qda(factor(label) ~ ., data = q7.3c.train)
```

```r
# Obtain predictions on the test set
q7.3c.preds <- predict(q7.3c.fit, q7.3c.test[1:3])

# Overall error rate
q7.3c.true.labs <- q7.3c.test$label
q7.3c.pred.labs <- q7.3c.preds$class
q7.3c.error.rate <- round(sum(((as.numeric(q7.3c.true.labs == q7.3c.pred.labs))
                               / length(q7.3c.true.labs)) * 100), 2)

# Create an ROC curve
library(ROCR)
rocplot <- function(pred, truth) {
  predob <- prediction(pred, truth)
  perf <- performance(predob, "tpr", "fpr")
  plot(perf, colorize = TRUE)
}
q7.3c.roc.plot <- rocplot(q7.3c.preds$posterior[, 2], q7.3c.test$label)
```



```r
# Calculate AUC
q7.3c.auc <- ROCR::prediction(q7.3c.preds$posterior[, 2], q7.3c.test$label) %>%
  ROCR::performance(measure = "auc") %>% .@y.values
q7.3c.auc <- as.numeric(q7.3c.auc)
```

After training the QDA model on the new data set and applying the model to the test set, we see that we still have a very poor performing model. From this ROC plot we can calculate the AUC value of 0.5147059. Here we can see that the model is as good as if were were to randomly flip a coin to choose which carrier an observation would belong.

Discriminant Analysis

What is the main cause of the message "Warning in `lda.default`(x, grouping, . . .): variables are collinear"?

With this warning message, the algorithm is letting us know that some of our predictors are correlated. When we have predictors that are correlated, then it makes the model less accurate due to the fact that if there were a change in one predictor, which could then be compensated by the value of another predictor, and the effect of either value's effect on the prediction would be hard to determine. This is why we check for correlation and remove any columns (one of the pair) that are highly correlated.

What is the main cause of the message "Error in `qda.default`(x, grouping, . . .) : some group is too small for 'qda' "?

This error occurs when you have groups, or factors, that are too small or contain 'na' values. When there are not enough observations in a group or factor, then there is not enough information supplied to the algorithm to make an accurate determination of how to classify that group or factor.

Provide details on the `list` that `predict{MASS}` returns.

When using `predict{MASS}` for predicting classification of a test set, the function will return a `list` of the predicted labels for each of the observations in the test set. This list can then be compared to the true labels of the test set to calculate the overall accuracy of the model.

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

Yes, the `gamma` and `lambda` arguements of `rda{klaR}` can be set manually, though it would not be wise. As these arguments are determined by the minimized estimated error rate of cross-validation if they are not manually set. Therefore, unless you have calculated the estimated error rate (which would imply that you have done cross-validation), these arguments should be left unspecified.

Human Cancer Microarray Data

For this part of the applied exercises, we will use the human cancer microarray data that has been discussed in lectrures and is provided in the `ElemStatLearn` R library. With this data we will choose 3 cancer types `MELANOMA`, `OVARIAN`, and `RENAL`. Fore each of these cancer types we will randomly select 60 genes to use for our analysis and guide our observatons. Note: We will use `set.seed(123)` in order to ensure reproducible results.