Computer Assignment 5 - Classification and Cross Validation

Machine Learning, Fall 2021

YOUR NAME

```r
library(mvtnorm)
library(ggplot2)
```

1994 Census Data

Let's walk through real example. First, load the `adults.csv` data (downloaded originally from here).

```r
adult = read.csv("adults.csv")
adult$X = NULL
```

Next, we will run the logistic regression model to predict the class `income`, which marks whether a given adult makes $\leqslant \$50k$ (coded as a 0) or $\geqslant \$50k$ (coded as a 1). To assess the performance of our model, we will only build the model on 75% of our data so that we can later use the remaining 25% as a testing data set.

```r
set.seed(13)
training_size <- round(.75 * nrow(adult))  # training set size
indices = sample(1:nrow(adult), training_size)
training_set <- adult[indices,]
testing_set <- adult[-(indices),]
m1 <- glm(income ~ ., data = training_set, family = binomial('logit'))
```

Question 1:

1. Examine the `summary` of our logistic regression model. Comment on the significance of each of our predictors.

2. Do any of the significant predictors surprise you?

3. Provide an interpretation of what the `Estimate` value is for the predictor `age`. Your answer should say something about this value's relation to the log-odds.

```r
YOUR CODE HERE
summary(m1)
```

ANSWERS:

1.
2.
3.

Question 2:

1. Now that you have created a model on the `training_data`, use the `predict` function in **R** to use your model to classify the data in the `testing_data`.

2. What proportion of values were classified incorrectly? Find the risk on the testing data.

```
YOUR CODE, AND ANALYSIS, HERE
```

ANSWERS:

1.
2.

Next we would like to fit an LDA model for comparison. Unfortunately, there is a slight problem: most of the predictors in the dataset are categorical. Recall that LDA assumes that the class conditional densities are Gaussian. Obviously this does not make sense for categorical variables. Luckily, **R** has a way around this. For a categorical variable that takes on d different values, **R** will automatically replace that variable with $d - 1$ binary variables representing the levels of the original categorical variable. When all of these binary variables are equal to 0, the implied category is the one that was left out. For example, in the `adults` dataset, the variable `sex` takes on two values: `Female` and `Male`. As described above, `sex` can be recoded as a single binary variable, `sexMale`, which is equal to 1 when `sex == Male` and 0 when `sex == Female`. With these new binary variables in place, we can run LDA. Thus, when you run LDA on the `adults` dataset, you will not see a binary variable for `Female`.

Question 3:

1. Build an LDA model on the `training_data`, and see how well it performs classifying the observations in the `testing_data`.
2. Compare the risk (percentage of missclassified observations) of the LDA and the logistic regression. Which method gives the best performance?

```
YOUR CODE, AND ANALYSIS, HERE
```

ANSWERS:

1.
2.

Not-So-Perfect Data for LDA

Recall on CA4, we performed LDA on a data set that was generated from the LDA model. For that dataset, the two groups (classified 0 and 1) were generated from two multinormal distributions with the same variance but different means. LDA performed very well on that data because *the data were generated from the underlying LDA model* that we were trying to fit.

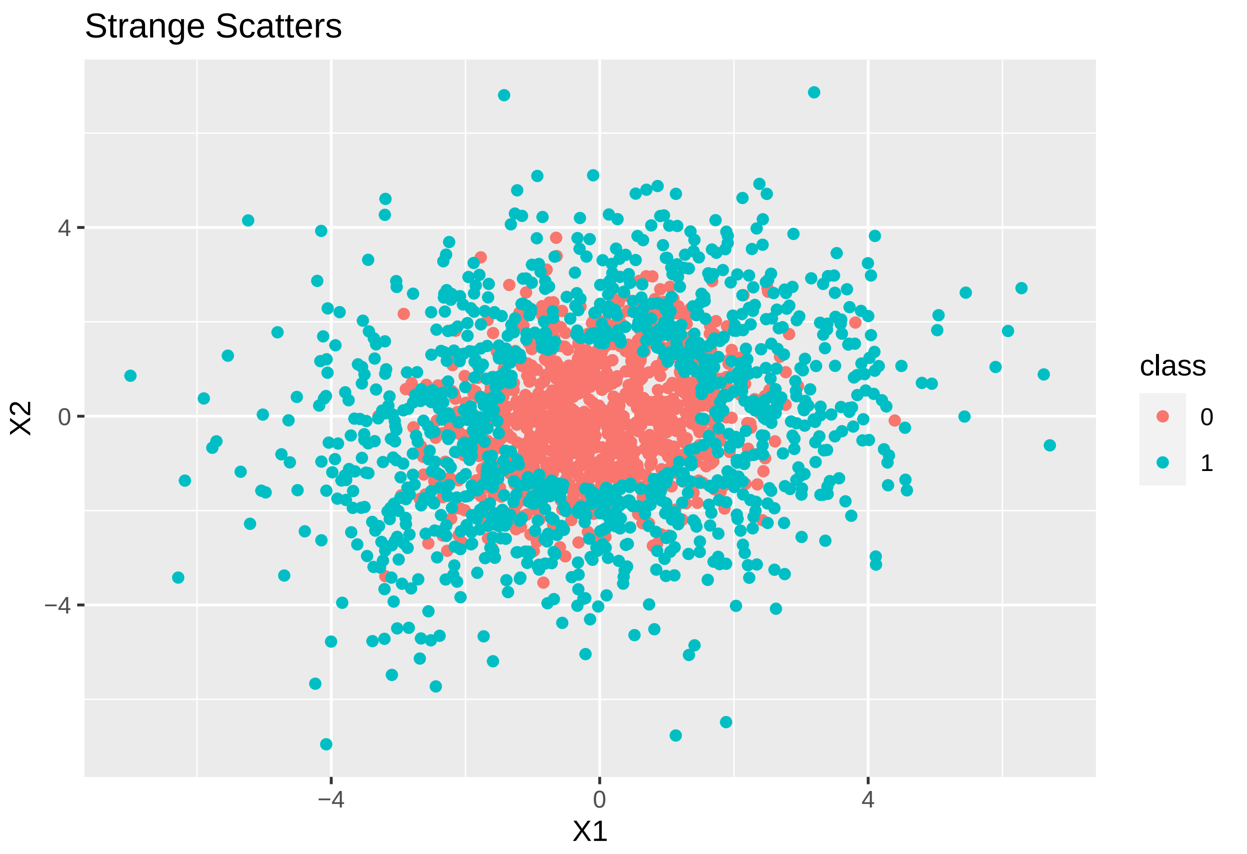
In this assignment, we will explore what happens when we apply our methods to a dataset that is generated from a distribution that is *very different* from a LDA model. We will start in two dimensions. Note that while we are still using the multinormal function to generate the following data, the variance of the marginals is different, the means are the same, and we manipulate `data_1` in such a way that it is no longer normal.

First we will generate the data.

```r
# Specify covariance matrices.
my_sigma_1 = t(matrix(c(2,0.2,0.2,2)*0.6, ncol=2)) %*% matrix(c(2,0.2,0.2,2)*0.6, ncol=2)
my_sigma_2 = t(matrix(c(2,0.2,0.2,2), ncol=2)) %*% matrix(c(2,0.2,0.2,2), ncol=2)
# Generate data.
data_0 = rmvnorm(1000, mean = rep(0, times = 2), sigma = my_sigma_1)
data_1 = rmvnorm(1500, mean = rep(0, times = 2), sigma = my_sigma_2)
# Remove central points from one of the classes.
central_points = ( sqrt( data_1[,1]**2 + data_1[,2]**2) < 1.5 )
data_1 = data_1[!central_points,]
```

Let's take a look at the data we get.

```r
temp_data = data.frame(rbind(data_0, data_1))
temp_data$class = as.factor(c(rep(0, times = 1000), rep(1, times = nrow(data_1))))
ggplot(temp_data, aes(x = X1, y = X2, color = class)) + geom_point() +
  ggtitle("Strange Scatters")
```



Question 4:

1. The way these data were created breaks *two* of the assumptions for LDA. What are they? Do you predict LDA or KNN will have better performance on this data? Defend your choice.
2. If you were to draw a decision boundary on this data, what shape would it be? How might such a decision boundary be difficult for an LDA model to capture?

ANSWERS:

1.

2.

Question 5:

1. Split the data into a training and a testing set. Make sure your training set contains 70% of the data, and the testing set the remaining 30%.
2. Build an LDA model on the training data. Once you obtain this, use the function `predict` to find the predictions of your LDA model on the training data. What is the percentage of incorrectly classified observations from the training data?
3. Use again the function `predict` to find the predictions of the LDA model on the testing data. Find the percentage of missclassified observations.
4. Comment on your results. Does the model have poor performance? Is it any better than randomly guessing the value of Y with a cointoss (that will have an error of 50%)?

```
YOUR CODE AND ANALYSIS HERE
```

ANSWERS:

1.

2.

3.

4.

Question 6:

1. Generate a new split of the data into training and testing sets. Use the same 70% and 30% split.
2. Using the training data, and the function `knn`, find the KNN predictions for the training data, for $k = 1, 5, 11$. For each value of k, find the percentage of missclassified observations.
3. Using the training data, find the KNN predictions for the testing data, for $k = 1, 5, 11$. For each k, find the percentage of missclassified observations.
4. Is there a big difference between the percentages on the training and testing data? Which of the values of k would you use for predicting on new data?
5. Was your prediction in Q4 correct or incorrect?

```
YOUR CODE AND ANALYSIS HERE
```

ANSWERS:

1.

2.

3.

4.

5.

Error Rate for Increasing Data Sizes

We will now examine how the error rate stabilizes for larger and larger data sizes. Manipulate the variable `data_size` in the following code to simulate different sizes of the data we created for the last problem.

```r
data_size = 10
my_sigma_1 = t(matrix(c(2,0.2,0.2,2)*0.6, ncol=2)) %*% matrix(c(2,0.2,0.2,2)*0.6, ncol=2)
my_sigma_2 = t(matrix(c(2,0.2,0.2,2), ncol=2)) %*% matrix(c(2,0.2,0.2,2), ncol=2)
data_0 = data.frame(rmvnorm(data_size, mean = rep(0, times = 2), sigma = my_sigma_1))
data_1 = data.frame(X1 = NA, X2 = NA)
count = 0
while(count < data_size){
  new_draw = rmvnorm(1, mean = rep(0, times = 2), sigma = my_sigma_2)
  if( sqrt( new_draw[,1]**2 + new_draw[,2]**2) >= 1.5 ) {
    data_1 = rbind(data_1, data.frame(new_draw))
    count = count + 1
  }
}
data_1 = data_1[-1,]
my_data = data.frame(rbind(data_0, data_1))
my_data$class = as.factor(c(rep(0, times = data_size), rep(1, times = data_size)))
```

Question 7:

For `data_size` $\in \{25, 50, 75, 100, 125, 150, 175, 200, 225, 250, 275, 300, 1000\}$ do the following:

- Generate the "Strange Scatters" data using the given code.
- Split the data into testing and training data sets (say, 70/30 ratio).
- Build an LDA model and the KNN models for $k \in \{1, 5, 11\}$ on the training data, predict the labels on the training data, and calculate the training error rate.
- Build an LDA model and the KNN models for $k \in \{1, 5, 11\}$ on the training data, predict the labels on the testing data, and calculate the testing error rate.
- Save both error rates

Question 8:

1. Plot two scatterplots: one for each error type (training/testing errors) with the data sizes on the x-axis, the error rate on the y-axis, and color by which model type you used.
2. Do you notice any trends as your data size increases? Do the error rates "stabilize"?

```
YOUR CODE AND ANALYSIS HERE
```