Homework 4

STAT 430, Spring 2017

Due: Friday, February 24 by 11:59 PM

Exercise 1

[10 points] We once again return to the data found in `auto-train.csv` and `auto-test.csv` which contain train and test data respectively. `auto.csv` is provided but not used. It is a modification of the `Auto` data from the `ISLR` package. Use `mpg` as the response. Train the following models:

- Additive Logistic Regression
- LDA
- QDA
- Naive Bayes

Report test and train accuracies for both. Indicate which model performs best.

You should consider coercing the response to be a factor variable.

Solution:

```
# import data
auto_train = read_csv("auto-train.csv")
auto_test = read_csv("auto-test.csv")
```

```
# coerce to factor
auto_train$mpg = as.factor(auto_train$mpg)
auto_test$mpg = as.factor(auto_test$mpg)
```

```
# load packages
library(MASS)
library(e1071)
library(caret)
```

```
# train models
auto_lr = glm(mpg ~ ., data = auto_train, family = "binomial")
auto_lda = lda(mpg ~ ., data = auto_train)
auto_qda = qda(mpg ~ ., data = auto_train)
auto_nb = naiveBayes(mpg ~ ., data = auto_train)
```

```
# get predictions

auto_lr_train_pred = ifelse(predict(auto_lr, auto_train) > 0, 1, 0)
auto_lr_test_pred = ifelse(predict(auto_lr, auto_test) > 0, 1, 0)

auto_lda_train_pred = predict(auto_lda, auto_train)$class
auto_lda_test_pred = predict(auto_lda, auto_test)$class

auto_qda_train_pred = predict(auto_qda, auto_train)$class
auto_qda_test_pred = predict(auto_qda, auto_test)$class

auto_nb_train_pred = predict(auto_nb, auto_train)
auto_nb_test_pred = predict(auto_nb, auto_test)
```

```r
# accuracy function
accuracy = function(actual, predicted) {
  mean(actual == predicted)
}
```

```r
# store results in data frame
auto_classifiers = c("Logistic", "LDA", "QDA", "Naive Bayes")

auto_train_acc = c(
  accuracy(predicted = auto_lr_train_pred, actual = auto_train$mpg),
  accuracy(predicted = auto_lda_train_pred, actual = auto_train$mpg),
  accuracy(predicted = auto_qda_train_pred, actual = auto_train$mpg),
  accuracy(predicted = auto_nb_train_pred, actual = auto_train$mpg)
)

auto_test_acc = c(
  accuracy(predicted = auto_lr_test_pred, actual = auto_test$mpg),
  accuracy(predicted = auto_lda_test_pred, actual = auto_test$mpg),
  accuracy(predicted = auto_qda_test_pred, actual = auto_test$mpg),
  accuracy(predicted = auto_nb_test_pred, actual = auto_test$mpg)
)

auto_results = data.frame(
  auto_classifiers,
  auto_train_acc,
  auto_test_acc
)

colnames(auto_results) = c("Method", "Train Accuracy", "Test Accuracy")
```

```r
# display data frame as table
knitr::kable(auto_results)
```

Method	Train Accuracy	Test Accuracy
Logistic	0.8966667	0.9239130
LDA	0.9133333	0.9239130
QDA	0.9000000	0.8913043
Naive Bayes	0.9033333	0.9021739

Interestingly, both logistic and LDA perform equally well, as they both obtain the best test accuracy!

Exercise 2

```r
library(MASS)

# setup parameters
num_obs = 1000

# means
mu_1 = c(10, 8.5)
mu_2 = c(20, 10)
```

```r
mu_3 = c(10, 15)
mu_4 = c(10, 20)

# sigmas
sigma_1 = matrix(c(10, -4, -4, 8), 2, 2)
sigma_2 = matrix(c(5, -3, -3, 5), 2, 2)
sigma_3 = matrix(c(8, 3, 3, 8), 2, 2)
sigma_4 = matrix(c(8, 6, 6, 8), 2, 2)

# control randomization
set.seed(42)

# make train data
hw04_train = data.frame(

  # create response
  as.factor(c(rep("A", num_obs / 2), rep("B", num_obs),
              rep("C", num_obs * 2), rep("D", num_obs))),

  # create predictors
  rbind(
    mvrnorm(n = num_obs / 2, mu = mu_1, Sigma = sigma_1),
    mvrnorm(n = num_obs, mu = mu_2, Sigma = sigma_2),
    mvrnorm(n = num_obs * 2, mu = mu_3, Sigma = sigma_3),
    mvrnorm(n = num_obs, mu = mu_4, Sigma = sigma_4)
  )

)
# label variables
colnames(hw04_train) = c("y", "x1", "x2")


# make test data
hw04_test = data.frame(

  # create response
  as.factor(c(rep("A", num_obs), rep("B", num_obs),
              rep("C", num_obs), rep("D", num_obs))),

  # create predictors
  rbind(
    mvrnorm(n = num_obs, mu = mu_1, Sigma = sigma_1),
    mvrnorm(n = num_obs, mu = mu_2, Sigma = sigma_2),
    mvrnorm(n = num_obs, mu = mu_3, Sigma = sigma_3),
    mvrnorm(n = num_obs, mu = mu_4, Sigma = sigma_4)
  )

)
# label variables
colnames(hw04_test) = c("y", "x1", "x2")

# write to files
readr::write_csv(hw04_train, "hw04-train.csv")
```

```
readr::write_csv(hw04_test, "hw04-test.csv")

# clear workspace
rm(list = ls())
```

[20 points] Use the data found in `hw04-train.csv` and `hw04-test.csv` which contain train and test data respectively. Use `y` as the response. Coerce `y` to be a factor after importing the data.

Create an ellipse plot then train the following models:

- Additive Logistic Regression
- LDA
- LDA with Flat Prior
- QDA
- QDA with Flat Prior
- Naive Bayes

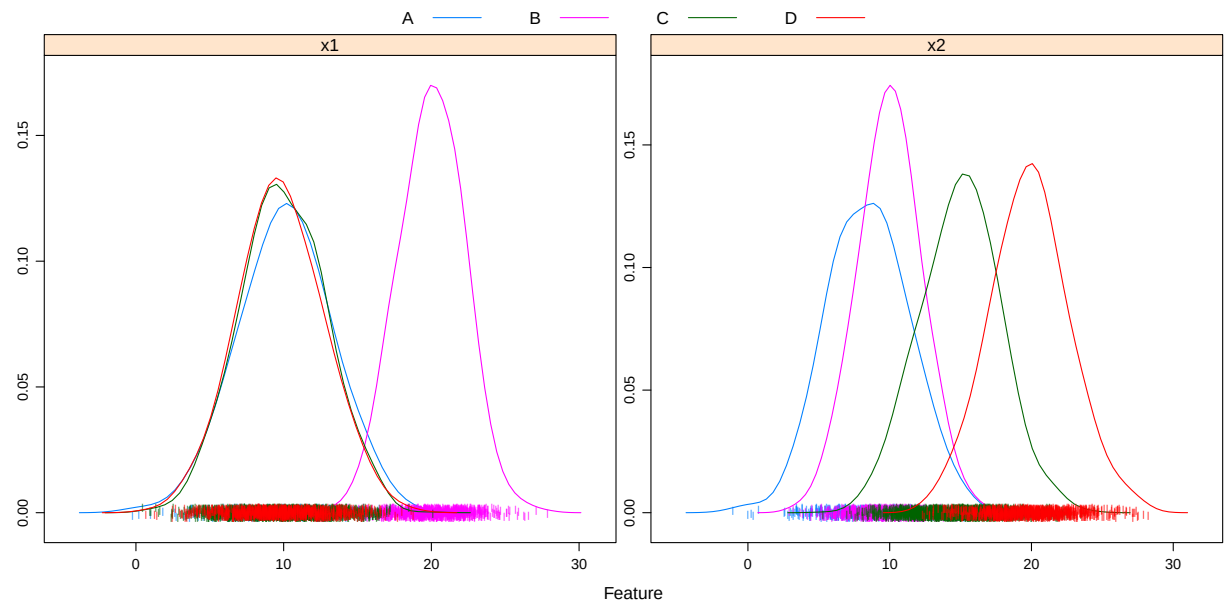
Report test and train accuracies for both. Indicate which model performs best. Do the results match the intution from the plot? Which class(es) is your best classifier classifying the best?

Solution:

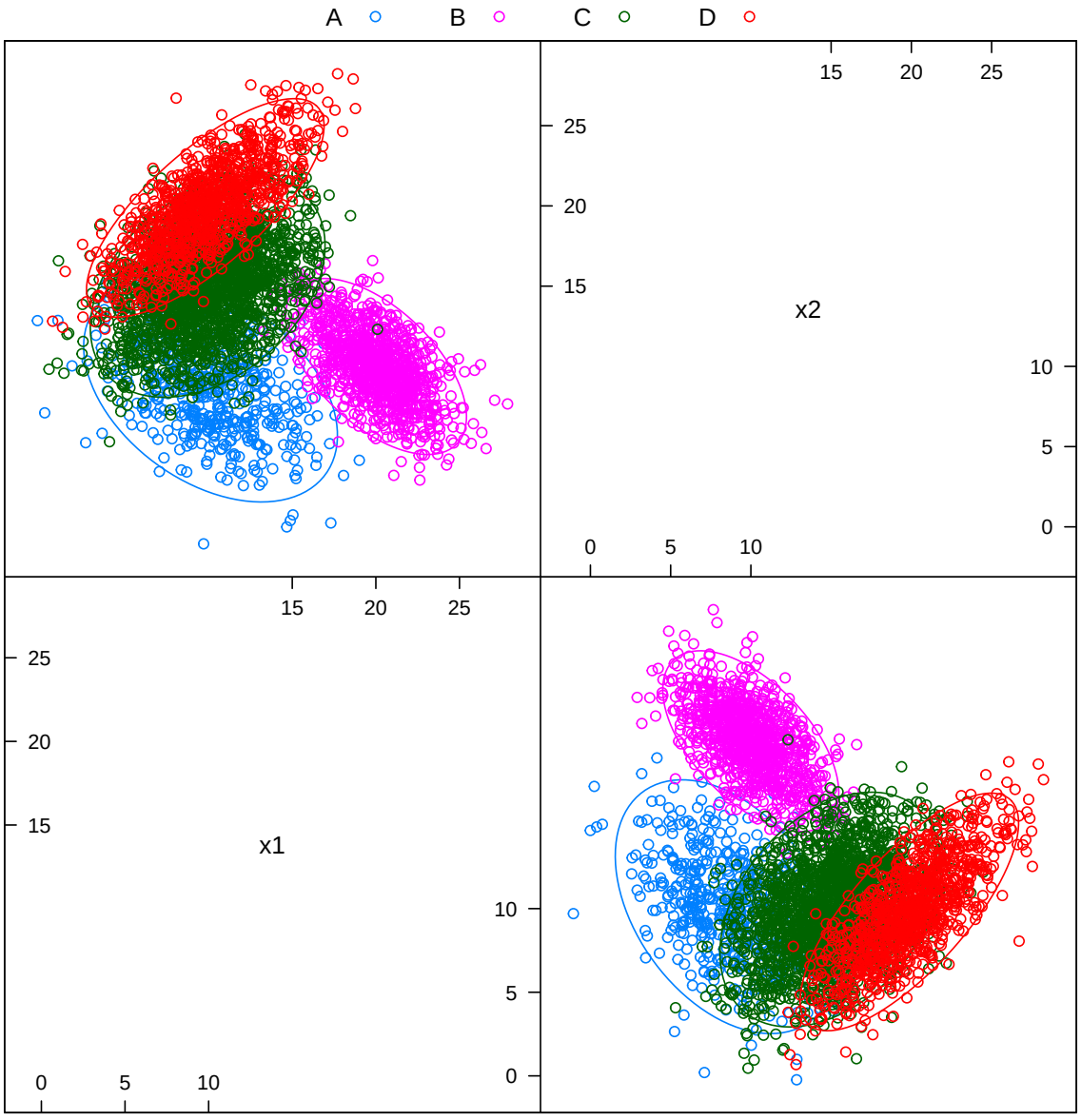
```
# read data
hw04_train = readr::read_csv("hw04-train.csv")
hw04_test = readr::read_csv("hw04-test.csv")

# coerce characters to factors
hw04_train$y = as.factor(hw04_train$y)
hw04_test$y = as.factor(hw04_test$y)
```

```
# load packages
library(MASS)
library(e1071)
library(caret)
library(nnet)
library(ellipse)
```

```
caret::featurePlot(x = hw04_train[, 2:3],
            y = hw04_train$y,
            plot = "density",
            scales = list(x = list(relation="free"),
                          y = list(relation="free")),
            adjust = 1.5,
            pch = "|",
            layout = c(2, 1),
            auto.key = list(columns = 4))
```




```
featurePlot(x = hw04_train[, 2:3],
            y = hw04_train$y,
            plot = "ellipse",
            auto.key = list(columns = 4))
```



Scatter Plot Matrix

```
# accuracy function
accuracy = function(actual, predicted) {
  mean(actual == predicted)
}
```

```
# store results in data frame
hw04_classifiers = c("Logistic", "LDA", "LDA, Flat Prior", "QDA", "QDA, Flat Prior", "Naive Bayes")

hw04_train_acc = c(
  accuracy(hw04_train$y, predict(multinom(y ~ ., data = hw04_train, trace = FALSE), hw04_train)),
  accuracy(hw04_train$y, predict(lda(y ~ ., data = hw04_train), hw04_train)$class),
  accuracy(hw04_train$y, predict(lda(y ~ ., data = hw04_train, prior = c(1, 1, 1, 1) / 4), hw04_train)$
  accuracy(hw04_train$y, predict(qda(y ~ ., data = hw04_train), hw04_train)$class),
  accuracy(hw04_train$y, predict(qda(y ~ ., data = hw04_train, prior = c(1, 1, 1, 1) / 4), hw04_train)$
```

```
  accuracy(hw04_train$y, predict(naiveBayes(y ~ ., data = hw04_train), hw04_train))
)

hw04_test_acc = c(
  accuracy(hw04_test$y, predict(multinom(y ~ ., data = hw04_train, trace = FALSE), hw04_test)),
  accuracy(hw04_test$y, predict(lda(y ~ ., data = hw04_train), hw04_test)$class),
  accuracy(hw04_test$y, predict(lda(y ~ ., data = hw04_train, prior = c(1, 1, 1, 1) / 4), hw04_test)$cla
  accuracy(hw04_test$y, predict(qda(y ~ ., data = hw04_train), hw04_test)$class),
  accuracy(hw04_test$y, predict(qda(y ~ ., data = hw04_train, prior = c(1, 1, 1, 1) / 4), hw04_test)$cla
  accuracy(hw04_test$y, predict(naiveBayes(y ~ ., data = hw04_train), hw04_test))
)

hw04_results = data.frame(
  hw04_classifiers,
  hw04_train_acc,
  hw04_test_acc
)

colnames(hw04_results) = c("Method", "Train Accuracy", "Test Accuracy")
```

```
# display data frame as table
knitr::kable(hw04_results)
```

Method	Train Accuracy	Test Accuracy
Logistic	0.8517778	0.82575
LDA	0.8380000	0.80175
LDA, Flat Prior	0.8093333	0.83125
QDA	0.8522222	0.83075
QDA, Flat Prior	0.8208889	0.86000
Naive Bayes	0.8266667	0.80000

```
# class proportions in train data
table(hw04_train$y) /  length(hw04_train$y)
```

```
##
##         A         B         C         D
## 0.1111111 0.2222222 0.4444444 0.2222222
```

```
# class proportions in test data
table(hw04_test$y) /  length(hw04_test$y)
```

```
##
##    A    B    C    D
## 0.25 0.25 0.25 0.25
```

```
# confusion matrix
table(predicted = predict(qda(y ~ ., data = hw04_train), hw04_test)$class,
      actual    = hw04_test$y)
```

```
##          actual
## predicted   A   B   C   D
##         A 703   3  33   0
##         B   8 976  11   0
##         C 279  21 849 205
```

```
##         D  10   0 107 795
```

- We see that the QDA with a Flat Prior performs the best.
- The fact that the Flat Prior works best doesn't have any intuition here, since there is no context. It just so happens that the proportion of classes in the test data is uniform. (See the data generation code.)
- The plot does offer intuition for QDA $>$ LDA $>$ NB. First, NB performs the worst because there is clearly significant correlation between `x1` and `x2`. (See the data generation code as well.) Between LDA and QDA it is clear that QDA is better as the Σ_k appear to be very different for different classes.
- From the confusion matrix, we see that QDA with Flat Prior is predicting best inside of class `B`. It has by far the fewest results off the diagonal. This is unsurprising as we could see from the pairs plot that the `B` class had the least overlap with the other classes. This is mostly due to its values of `x1`.

Note: when using the older `read.csv()` strings are *automatically* imported as factors by deafult. This would seem useful here, but a terrible idea in general. It is better to import as a character, then later explicitly coerce to a factor if desired. For this reason, `read_csv()` does not even provide an option to import characters as a factor. (At least not one this instructor is aware of.)