ISLR - Ch4 - Classification

Darshan Patel

1/24/2019

The following set of problems are from the applied exercises section in ISLR Chapter 4: Classification.

```
rm(list = ls())
library(MASS)
library(ISLR)
library(tidyverse)
```

```
## Warning: package 'tidyr' was built under R version 3.4.4
## Warning: package 'purrr' was built under R version 3.4.4
## Warning: package 'dplyr' was built under R version 3.4.4
```

```
library(gridExtra)
library(class)
```

Question 10: This question should be answered using the `Weekly` data set, which is part of the `ISLR` package. This data is similar to the `Smarket` data from this chapter's lab, except that it contains $1,089$ weekly returns for 21 years, from the beginning of 1990 to the end of 2010.

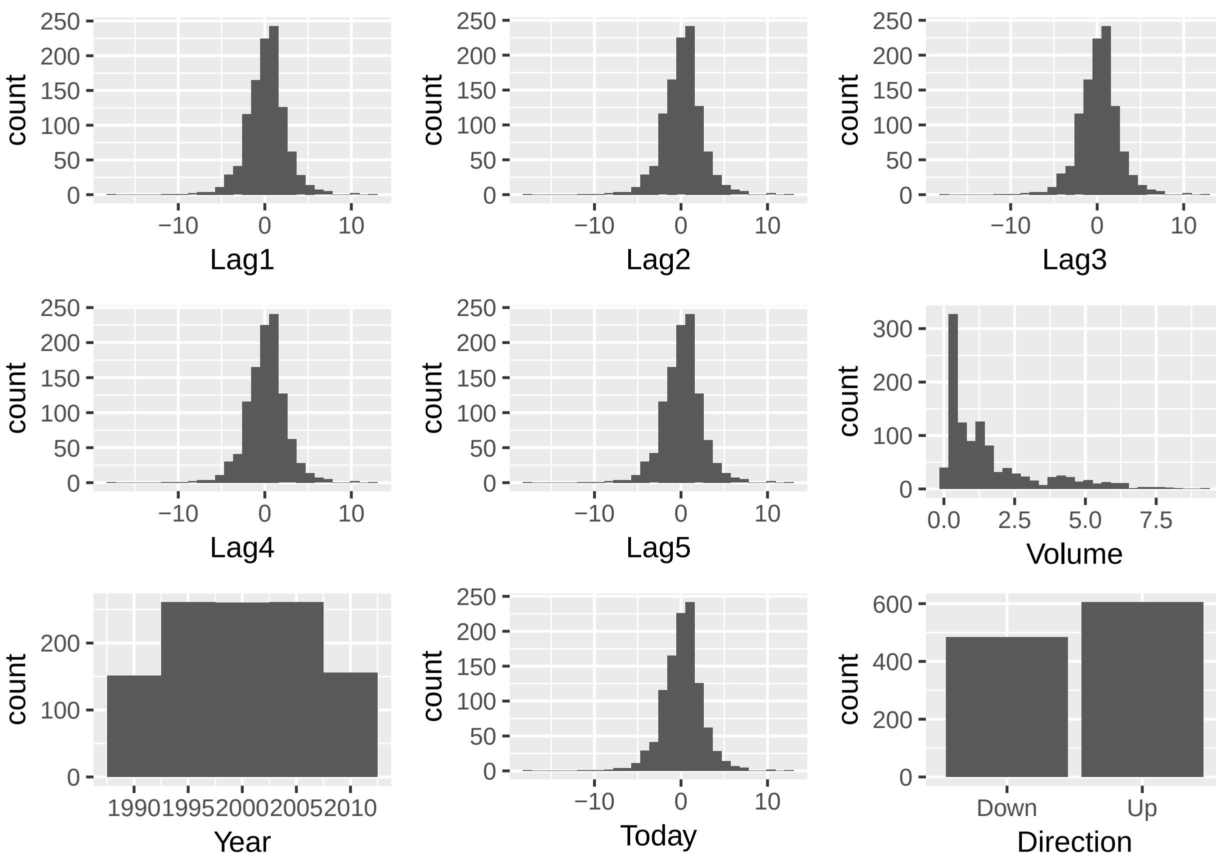
(a) Produce some numerical and graphical summaries of the `Weekly` data. Do there appear to be any patterns?

```
df = Weekly
summary(df)
```

```
##       Year           Lag1               Lag2               Lag3
##  Min.   :1990   Min.   :-18.1950   Min.   :-18.1950   Min.   :-18.1950
##  1st Qu.:1995   1st Qu.: -1.1540   1st Qu.: -1.1540   1st Qu.: -1.1580
##  Median :2000   Median :  0.2410   Median :  0.2410   Median :  0.2410
##  Mean   :2000   Mean   :  0.1506   Mean   :  0.1511   Mean   :  0.1472
##  3rd Qu.:2005   3rd Qu.:  1.4050   3rd Qu.:  1.4090   3rd Qu.:  1.4090
##  Max.   :2010   Max.   : 12.0260   Max.   : 12.0260   Max.   : 12.0260
##       Lag4               Lag5              Volume
##  Min.   :-18.1950   Min.   :-18.1950   Min.   :0.08747
##  1st Qu.: -1.1580   1st Qu.: -1.1660   1st Qu.:0.33202
##  Median :  0.2380   Median :  0.2340   Median :1.00268
##  Mean   :  0.1458   Mean   :  0.1399   Mean   :1.57462
##  3rd Qu.:  1.4090   3rd Qu.:  1.4050   3rd Qu.:2.05373
##  Max.   : 12.0260   Max.   : 12.0260   Max.   :9.32821
##      Today          Direction
##  Min.   :-18.1950   Down:484
##  1st Qu.: -1.1540   Up  :605
##  Median :  0.2410
##  Mean   :  0.1499
##  3rd Qu.:  1.4050
##  Max.   : 12.0260
```

```r
lag1 = ggplot(data = df, aes(Lag1)) + geom_histogram()
lag2 = ggplot(data = df, aes(Lag2)) + geom_histogram()
lag3 = ggplot(data = df, aes(Lag3)) + geom_histogram()
lag4 = ggplot(data = df, aes(Lag4)) + geom_histogram()
lag5 = ggplot(data = df, aes(Lag5)) + geom_histogram()
volume = ggplot(data = df, aes(Volume)) + geom_histogram()
year = ggplot(data = df, aes(Year)) + geom_histogram(binwidth = 5)
today = ggplot(data = df, aes(Today)) + geom_histogram()
direction = ggplot(data = df, aes(Direction)) + geom_bar()

grid.arrange(lag1, lag2, lag3, lag4, lag5, volume, year, today, direction, ncol=3)
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



All of the distributions of the lags appear to be slightly skewed right. `Volume` is heavily skewed right. `Year` is uniformly distributed.

(b) Use the full data set to perform a logistic regression with `Direction` as the response and the five lag variables plus `Volume` as predictors. Use the summary function to print the results. Do any of the predictors appear to be statistically significant? If so, which ones?

```r
model = glm(data = df, Direction~Lag1+Lag2+Lag3+Lag4+Lag5+Volume, family = binomial)
summary(model)
```

```
##
## Call:
## glm(formula = Direction ~ Lag1 + Lag2 + Lag3 + Lag4 + Lag5 +
##     Volume, family = binomial, data = df)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.6949  -1.2565   0.9913   1.0849   1.4579
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  0.26686    0.08593   3.106   0.0019 **
## Lag1        -0.04127    0.02641  -1.563   0.1181
## Lag2         0.05844    0.02686   2.175   0.0296 *
## Lag3        -0.01606    0.02666  -0.602   0.5469
## Lag4        -0.02779    0.02646  -1.050   0.2937
## Lag5        -0.01447    0.02638  -0.549   0.5833
## Volume      -0.02274    0.03690  -0.616   0.5377
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1496.2  on 1088  degrees of freedom
## Residual deviance: 1486.4  on 1082  degrees of freedom
## AIC: 1500.4
##
## Number of Fisher Scoring iterations: 4
```

The only predictor that appears to be statistically significant is `Lag2`.

(c) Compute the confusion matrix and overall fraction of correct predictions. Explain what the confusion matrix is telling about the types of mistakes made by logistic regression.

```r
predictions = predict(model, type = "response")
pred = rep("Down", nrow(df))
pred[predictions >.5]="Up"
table(pred,df$Direction)
```

```
##
## pred   Down  Up
##   Down   54  48
##   Up    430 557
```

The diagonal values tell how many directions were correctly classified as either `Up` or `Down`. On the other hand, the other two values tell how many values were misclassified. According to this logistic regression model, many misclassifications were made. The overall fraction of correct predictions is

```r
(54 + 557) / (nrow(df))
```

```
## [1] 0.5610652
```

(d) Now fit the logistic regression model using a training data period from 1990 to 2008, with `Lag2` as the only predictor. Compute the confusion matrix and the overall fraction of correct predictions for the held out data (that is, the data from 2009 and 2010).

```
train = subset(df, (Year >= 1990) & (Year <= 2008))
test = subset(df, Year >= 2009)

model2 = glm(data = train, Direction~Lag2, family = binomial)
summary(model2)
```

```
##
## Call:
## glm(formula = Direction ~ Lag2, family = binomial, data = train)
##
## Deviance Residuals:
##    Min      1Q  Median      3Q     Max
## -1.536  -1.264   1.021   1.091   1.368
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  0.20326    0.06428   3.162  0.00157 **
## Lag2         0.05810    0.02870   2.024  0.04298 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1354.7  on 984  degrees of freedom
## Residual deviance: 1350.5  on 983  degrees of freedom
## AIC: 1354.5
##
## Number of Fisher Scoring iterations: 4
```

According to the logistic regression model, `Lag2` is statistically significant at the $\alpha = 0.05$ level.

The confusion matrix is:

```
predictions = predict(model2, test, type = "response")
pred = rep("Down", nrow(test))
pred[predictions > 0.5]="Up"
table(pred, test$Direction)
```

```
##
## pred   Down Up
##   Down    9  5
##   Up     34 56
```

The overall fraction of correct predictions is:

```
(9 + 56) / nrow(test)
```

```
## [1] 0.625
```

(e) Repeat (d) using LDA.

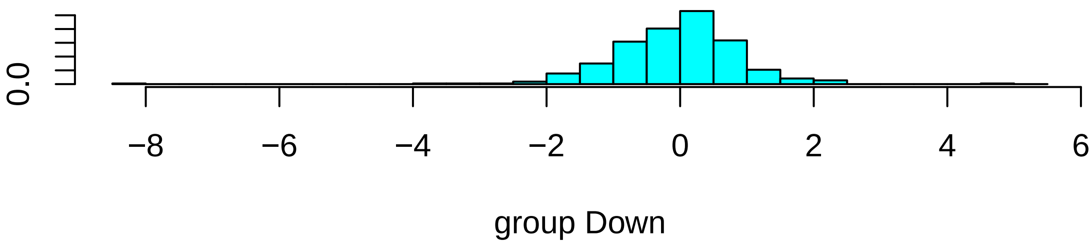
```
model3 = lda(data = train, Direction~Lag2)
model3
```

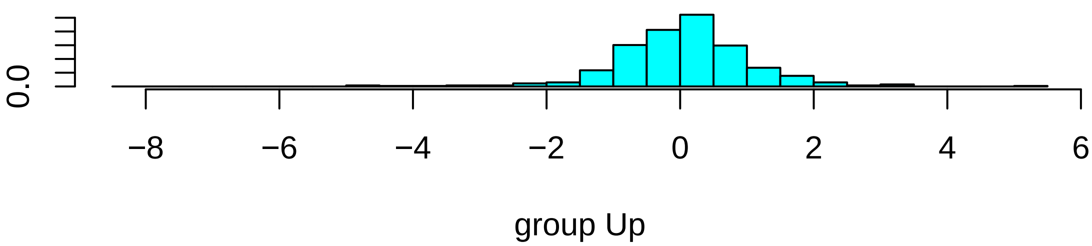
```
## Call:
## lda(Direction ~ Lag2, data = train)
##
```

```
## Prior probabilities of groups:
##      Down        Up
## 0.4477157 0.5522843
##
## Group means:
##             Lag2
## Down -0.03568254
## Up    0.26036581
##
## Coefficients of linear discriminants:
##            LD1
## Lag2 0.4414162
```

According to the LDA model, 44.77% of the observations had a `Down` `Direction` while 55.22% of the observations had an `Up` `Direction`. Furthermore, if $0.44 \times \text{Lag2}$ is large, then the model classifies `Direction` as `Up`, else otherwise.

```r
plot(model3)
```





This plot shows the linear discriminants of $0.44 \times \text{Lag2}$ for each of the `Direction`s.

The confusion matrix is:

```r
predictions = predict(model3, test)
table(predictions$class, test$Direction)
```

```
##
##        Down Up
##   Down    9  5
##   Up     34 56
```

The overall fraction of correct predictions is

```r
(9+56) / nrow(test)
```

```
## [1] 0.625
```

(f) Repeat (d) using QDA.

```r
model4 = qda(data = train, Direction~Lag2)
model4
```

```
## Call:
## qda(Direction ~ Lag2, data = train)
##
## Prior probabilities of groups:
##      Down        Up
## 0.4477157 0.5522843
##
## Group means:
##             Lag2
## Down -0.03568254
## Up    0.26036581
```

Since the QDA classifier involves quadratics, it does not contain linear terms.

The confusion matrix is:

```r
predictions = predict(model4, test)
table(predictions$class, test$Direction)
```

```
##
##        Down Up
##   Down    0  0
##   Up     43 61
```

The overall fraction of correct predictions is

```r
61 / nrow(test)
```

```
## [1] 0.5865385
```

(g) Repeat (d) using KNN with $K = 1$.

```r
set.seed(2019)
model5 = knn(data.frame(train$Lag2), data.frame(test$Lag2), train$Direction, k = 1)
summary(model5)
```

```
## Down   Up
##   50   54
```

According to this model, 50 observations were classified as `Down` while 54 were classified as `Up`.

The confusion matrix is:

```r
table(model5, test$Direction)
```

```
##
## model5 Down Up
##   Down   21 29
##   Up     22 32
```

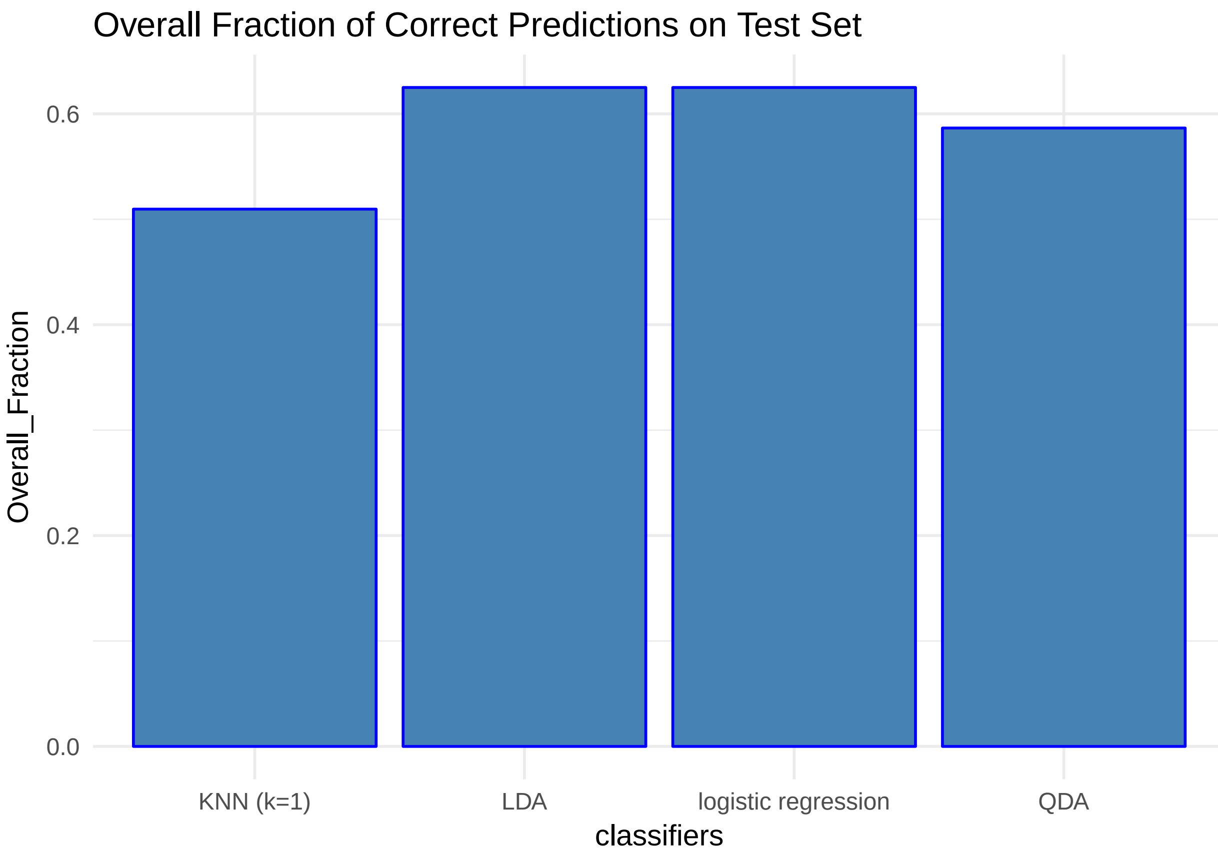
The overall fraction of correct predictions is

```r
(21 + 32) / nrow(test)
```

```
## [1] 0.5096154
```

(h) Which of these methods appear to provide the best results on this data? The best results were made by both the logistic regression model and the LDA model. A visual plot is made to show the comparison amongst all the classifiers' overall fraction of correct predictions.

```
ofcp = data.frame(classifiers = c("logistic regression", "LDA", "QDA", "KNN (k=1)"),
                  Overall_Fraction = c(0.625, 0.625, 0.5865385, 0.5096154))

ggplot(data = ofcp, aes(x = classifiers, y = Overall_Fraction)) +
  geom_col(color = "blue", fill = "steelblue") +
  ggtitle("Overall Fraction of Correct Predictions on Test Set") +
  theme_minimal()
```



(i) Experiment with different combinations of predictors, including possible transformations and interactions, for each of the methods. Report the variables, method, and associated confusion matrix that appears to provide the best results on the held out data. Note that you should also experiment with values for K in the KNN classifier.

```
df = Weekly
train = subset(df, (Year >= 1990) & (Year <= 2008))
test = subset(df, Year >= 2009)
```

Example 1: `Lag2` to the Power of 2

```
score = c()

lr1 = glm(data = train, Direction~Lag2^2, family = binomial)
predictions = predict(lr1, test, type = "response")
pred = rep("Down", nrow(test))
pred[predictions > 0.5] = "Up"
```

```r
print("Confusion Matrix for Logistic Regression")
```

```
## [1] "Confusion Matrix for Logistic Regression"
```

```r
table(pred, test$Direction)
```

```
##
## pred   Down Up
##   Down    9  5
##   Up     34 56
```

```r
score = c(score, sum(diag(table(pred, test$Direction))) / nrow(test))

lda1 = lda(data = train, Direction~Lag2^2)
predictions = predict(lda1, test)
print("Confusion Matrix for LDA")
```

```
## [1] "Confusion Matrix for LDA"
```

```r
table(predictions$class, test$Direction)
```

```
##
##        Down Up
##   Down    9  5
##   Up     34 56
```

```r
score = c(score, sum(diag(table(predictions$class, test$Direction))) / nrow(test))

qda1 = qda(data = train, Direction~Lag2^2)
predictions = predict(qda1, test)
print("Confusion Matrix for QDA")
```

```
## [1] "Confusion Matrix for QDA"
```

```r
table(predictions$class, test$Direction)
```

```
##
##        Down Up
##   Down    0  0
##   Up     43 61
```

```r
score = c(score, sum(diag(table(predictions$class, test$Direction))) / nrow(test))

knn1 = knn(data.frame(train$Lag2^2), data.frame(test$Lag2^2), train$Direction, k = 1)
print("Confusion Matrix for KNN where k = 1")
```

```
## [1] "Confusion Matrix for KNN where k = 1"
```

```r
table(knn1, test$Direction)
```

```
##
## knn1   Down Up
##   Down   24 32
##   Up     19 29
```

```r
score = c(score, sum(diag(table(knn1, test$Direction))) / nrow(test))

knn5 = knn(data.frame(train$Lag2^2), data.frame(test$Lag2^2), train$Direction, k = 5)
print("Confusion Matrix for KNN where k = 5")
```

```
## [1] "Confusion Matrix for KNN where k = 5"
```

```r
table(knn5, test$Direction)
```

```
##
## knn5   Down Up
##   Down   16 30
##   Up     27 31
```

```r
score = c(score, sum(diag(table(knn5, test$Direction))) / nrow(test))

knn10 = knn(data.frame(train$Lag2^2), data.frame(test$Lag2^2), train$Direction, k = 10)
print("Confusion Matrix for KNN where k = 10")
```

```
## [1] "Confusion Matrix for KNN where k = 10"
```

```r
table(knn10, test$Direction)
```

```
##
## knn10  Down Up
##   Down   16 23
##   Up     27 38
```

```r
score = c(score, sum(diag(table(knn10, test$Direction))) / nrow(test))

scores = data.frame(classifier = c("logistic regression", "LDA", "QDA",
                                   "KNN(k=1)", "KNN(k=5)", "KNN(k=10)"),
                    PercentCorrect = score)
scores
```

```
##            classifier PercentCorrect
## 1 logistic regression      0.6250000
## 2                 LDA      0.6250000
## 3                 QDA      0.5865385
## 4            KNN(k=1)      0.5096154
## 5            KNN(k=5)      0.4519231
## 6           KNN(k=10)      0.5192308
```

When using the square of `Lag2`, the logistic regression and LDA classifiers performed the best while KNN with $k = 5$ performed the worst.

Example 2: The log of `Volume`

```r
score = c()

lr1 = glm(data = train, Direction~log(Volume), family = binomial)
predictions = predict(lr1, test, type = "response")
pred = rep("Down", nrow(test))
pred[predictions > 0.5] = "Up"
print("Confusion Matrix for Logistic Regression")
```

```
## [1] "Confusion Matrix for Logistic Regression"
```

```r
table(pred, test$Direction)
```

```
##
## pred   Down Up
##   Down    1  1
##   Up     42 60
```

```r
score = c(score, sum(diag(table(pred, test$Direction))) / nrow(test))

lda1 = lda(data = train, Direction~log(Volume))
predictions = predict(lda1, test)
print("Confusion Matrix for LDA")
```

```
## [1] "Confusion Matrix for LDA"
```

```r
table(predictions$class, test$Direction)
```

```
##
##        Down Up
##   Down    1  1
##   Up     42 60
```

```r
score = c(score, sum(diag(table(predictions$class, test$Direction))) / nrow(test))

qda1 = qda(data = train, Direction~log(Volume))
predictions = predict(qda1, test)
print("Confusion Matrix for QDA")
```

```
## [1] "Confusion Matrix for QDA"
```

```r
table(predictions$class, test$Direction)
```

```
##
##        Down Up
##   Down   16 21
##   Up     27 40
```

```r
score = c(score, sum(diag(table(predictions$class, test$Direction))) / nrow(test))

knn1 = knn(data.frame(log(train$Volume)), data.frame(log(test$Volume)),
           train$Direction, k = 1)
print("Confusion Matrix for KNN where k = 1")
```

```
## [1] "Confusion Matrix for KNN where k = 1"
```

```r
table(knn1, test$Direction)
```

```
##
## knn1   Down Up
##   Down   16 29
##   Up     27 32
```

```r
score = c(score, sum(diag(table(knn1, test$Direction))) / nrow(test))

knn5 = knn(data.frame(log(train$Volume)), data.frame(log(test$Volume)),
           train$Direction, k = 5)
print("Confusion Matrix for KNN where k = 5")
```

```
## [1] "Confusion Matrix for KNN where k = 5"
```

```r
table(knn5, test$Direction)
```

```
##
## knn5   Down Up
##   Down   26 40
##   Up     17 21
```

```r
score = c(score, sum(diag(table(knn5, test$Direction))) / nrow(test))

knn10 = knn(data.frame(log(train$Volume)), data.frame(log(test$Volume)),
            train$Direction, k = 10)
print("Confusion Matrix for KNN where k = 10")
```

```
## [1] "Confusion Matrix for KNN where k = 10"
```

```r
table(knn10, test$Direction)
```

```
##
## knn10  Down Up
##   Down   32 37
##   Up     11 24
```

```r
score = c(score, sum(diag(table(knn10, test$Direction))) / nrow(test))

scores = data.frame(classifier = c("logistic regression", "LDA", "QDA",
                                   "KNN(k=1)", "KNN(k=5)", "KNN(k=10)"),
                    PercentCorrect = score)
scores
```

```
##            classifier PercentCorrect
## 1 logistic regression      0.5865385
## 2                 LDA      0.5865385
## 3                 QDA      0.5384615
## 4            KNN(k=1)      0.4615385
## 5            KNN(k=5)      0.4519231
## 6           KNN(k=10)      0.5384615
```

This choice of function of `Volume` did not perform well on classifying `Direction`. The best classifiers were logistic regression and LDA.

Example 3: Interaction of `Lag4` and `Lag5`

```r
score = c()

lr1 = glm(data = train, Direction~Lag4*Lag5, family = binomial)
predictions = predict(lr1, test, type = "response")
pred = rep("Down", nrow(test))
pred[predictions > 0.5] = "Up"
print("Confusion Matrix for Logistic Regression")
```

```
## [1] "Confusion Matrix for Logistic Regression"
```

```r
table(pred, test$Direction)
```

```
##
## pred   Down Up
##   Down    0  3
##   Up     43 58
```

```r
score = c(score, sum(diag(table(pred, test$Direction))) / nrow(test))

lda1 = lda(data = train, Direction~Lag4*Lag5)
predictions = predict(lda1, test)
print("Confusion Matrix for LDA")
```

```
## [1] "Confusion Matrix for LDA"
```

```r
table(predictions$class, test$Direction)
```

```
##
##        Down Up
##   Down    0  3
##   Up     43 58
```

```r
score = c(score, sum(diag(table(predictions$class, test$Direction))) / nrow(test))

qda1 = qda(data = train, Direction~Lag4*Lag5)
predictions = predict(qda1, test)
print("Confusion Matrix for QDA")
```

```
## [1] "Confusion Matrix for QDA"
```

```r
table(predictions$class, test$Direction)
```

```
##
##        Down Up
##   Down    7 16
##   Up     36 45
```

```r
score = c(score, sum(diag(table(predictions$class, test$Direction))) / nrow(test))

knn1 = knn(data.frame(train$Lag4*train$Lag5), data.frame(test$Lag4*test$Lag5),
           train$Direction, k = 1)
print("Confusion Matrix for KNN where k = 1")
```

```
## [1] "Confusion Matrix for KNN where k = 1"
```

```r
table(knn1, test$Direction)
```

```
##
## knn1   Down Up
##   Down   23 28
##   Up     20 33
```

```r
score = c(score, sum(diag(table(knn1, test$Direction))) / nrow(test))

knn5 = knn(data.frame(train$Lag4*train$Lag5), data.frame(test$Lag4*test$Lag5),
           train$Direction, k = 5)
print("Confusion Matrix for KNN where k = 5")
```

```
## [1] "Confusion Matrix for KNN where k = 5"
```

```r
table(knn5, test$Direction)
```

```
##
## knn5   Down Up
##   Down   21 24
##   Up     22 37
```

```r
score = c(score, sum(diag(table(knn5, test$Direction))) / nrow(test))

knn10 = knn(data.frame(train$Lag4*train$Lag5), data.frame(test$Lag4*test$Lag5),
            train$Direction, k = 10)
print("Confusion Matrix for KNN where k = 10")
```

```
## [1] "Confusion Matrix for KNN where k = 10"
```

```r
table(knn10, test$Direction)
```

```
##
## knn10  Down Up
##   Down   20 22
##   Up     23 39
```

```r
score = c(score, sum(diag(table(knn10, test$Direction))) / nrow(test))

scores = data.frame(classifier = c("logistic regression", "LDA", "QDA",
                                   "KNN(k=1)", "KNN(k=5)", "KNN(k=10)"),
                    PercentCorrect = score)
scores
```

```
##            classifier PercentCorrect
## 1 logistic regression      0.5576923
## 2                 LDA      0.5576923
## 3                 QDA      0.5000000
## 4            KNN(k=1)      0.5384615
## 5            KNN(k=5)      0.5576923
## 6           KNN(k=10)      0.5673077
```

Using an interaction term of `Lag4` $\times$ `Lag5` created interesting results. It is known that the logisic regression classifier and the LDA classifier will usually have the same percentage of correct predictions. In this example, the KNN classifier with $K = 5$ produced the same percentage. However, these classifiers were not the best classifiers. The KNN classifier with $K = 10$ outperformed the other models.

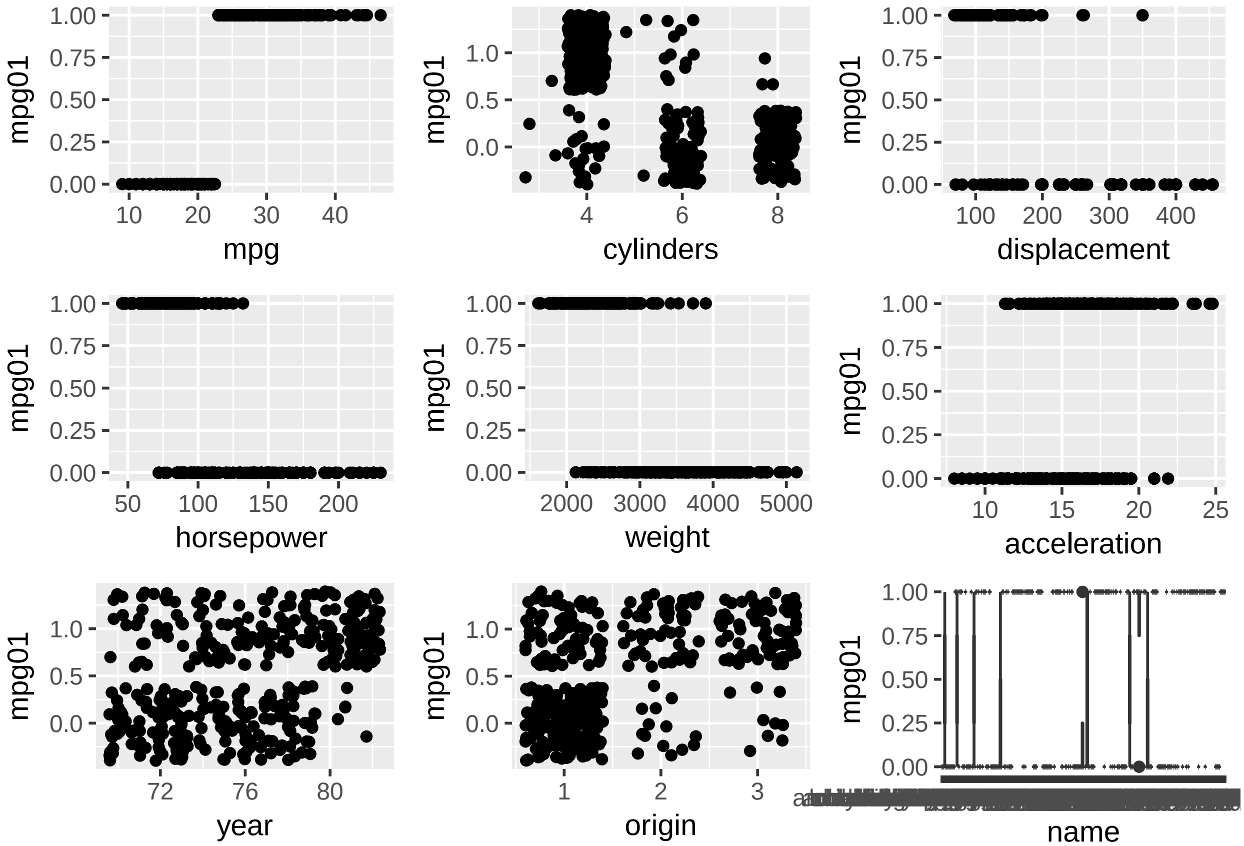
Question 11: In this problem, you will develop a model to predict whether a given car gets high or low gas mileage based on the `Auto` data set.

(a) Create a binary variable, `mpg01`, that contains a 1 if `mpg` contains a value above its median, and a 0 if `mpg` contains a value below its median. You can compute the median using the `median()` function. Note that you may find it easier to use the `data.frame()` function to create a single data set containing both `mpg01` and the other `Auto` variables.

```r
df = Auto
df$mpg01 = ifelse(df$mpg > median(df$mpg), 1, 0)
```

(b) Explore the data graphically in order to investigate the associated between `mpg01` and the other features. Which of the other features seem more likely to be useful in predicting `mpg01`? Scatterplots and boxplots may be useful tools to answer this question. Describe findings.

```r
g1 = ggplot(data = df, aes(x = mpg, y = mpg01)) + geom_point()
g2 = ggplot(data = df, aes(x = cylinders, y = mpg01)) + geom_jitter()
g3 = ggplot(data = df, aes(x = displacement, y = mpg01)) + geom_point()
g4 = ggplot(data = df, aes(x = horsepower, y = mpg01)) + geom_point()
g5 = ggplot(data = df, aes(x = weight, y = mpg01)) + geom_point()
g6 = ggplot(data = df, aes(x = acceleration, y = mpg01)) + geom_point()
g7 = ggplot(data = df, aes(x = year, y = mpg01, group = year)) + geom_jitter()
g8 = ggplot(data = df, aes(x = origin, y = mpg01)) + geom_jitter()
g9 = ggplot(data = df, aes(x = name, y = mpg01)) + geom_boxplot()

grid.arrange(g1, g2, g3, g4, g5, g6, g7, g8, g9, ncol = 3)
```



It appears that `mpg` itself will be a good predictor of `mpg01`; this makes sense since `mpg01` was derived from `mpg`. The following variables show some sort of relationship with `mpg01` that can be explained logistically: `horsepower`, `wweight`. The variable `displacement` does not make a clear relationship with `mpg01` nor does `year` and `origin`. The `cylinders` variable looks like a good predictor of `mpg01`.

Good predictors to use for following classifiers are: `horsepower` and `origin`.

(c) Split the data into a training set and a test set.

```
indexes = sample(1:nrow(df), size = round(0.9 * nrow(df)))
train = df[indexes,]
test = df[-indexes,]
```

(d) Perform LDA on the training data in order to predict `mpg01` using the variables that seemed most associated with `mpg01` in (b). What is the test error of the model obtained?

```
model1 = lda(data = train, mpg01~origin+horsepower)
predictions = predict(model1, test)
(table(predictions$class, test$mpg01)[1,2] +
    table(predictions$class, test$mpg01)[2,1])  / nrow(test)
```

```
## [1] 0.2051282
```

(e) Perform QDA on the training data in order to predicto `mpg01` using the variables that seemed most associated with `mpg01` in (b). What is the test error of the model obtained?

```
model2 = qda(data = train, mpg01~origin+horsepower)
predictions = predict(model2, test)
(table(predictions$class, test$mpg01)[1,2] +
    table(predictions$class, test$mpg01)[2,1]) / nrow(test)
```

```
## [1] 0.1538462
```

(f) Perform logistic regression on the training data in order to predict `mpg01` using the variables that seemed most associated with `mpg01` in (b). What is the test error of the model obtained?

```r
model3 = glm(data = train, mpg01~origin+horsepower, family = binomial)
predictions = predict(model3, test, type = "response")
pred = rep(0, nrow(test))
pred[predictions > 0.5] = 1
(table(pred, test$mpg01)[1,2] +
    table(pred, test$mpg01)[2,1]) / nrow(test)
```

```
## [1] 0.2051282
```

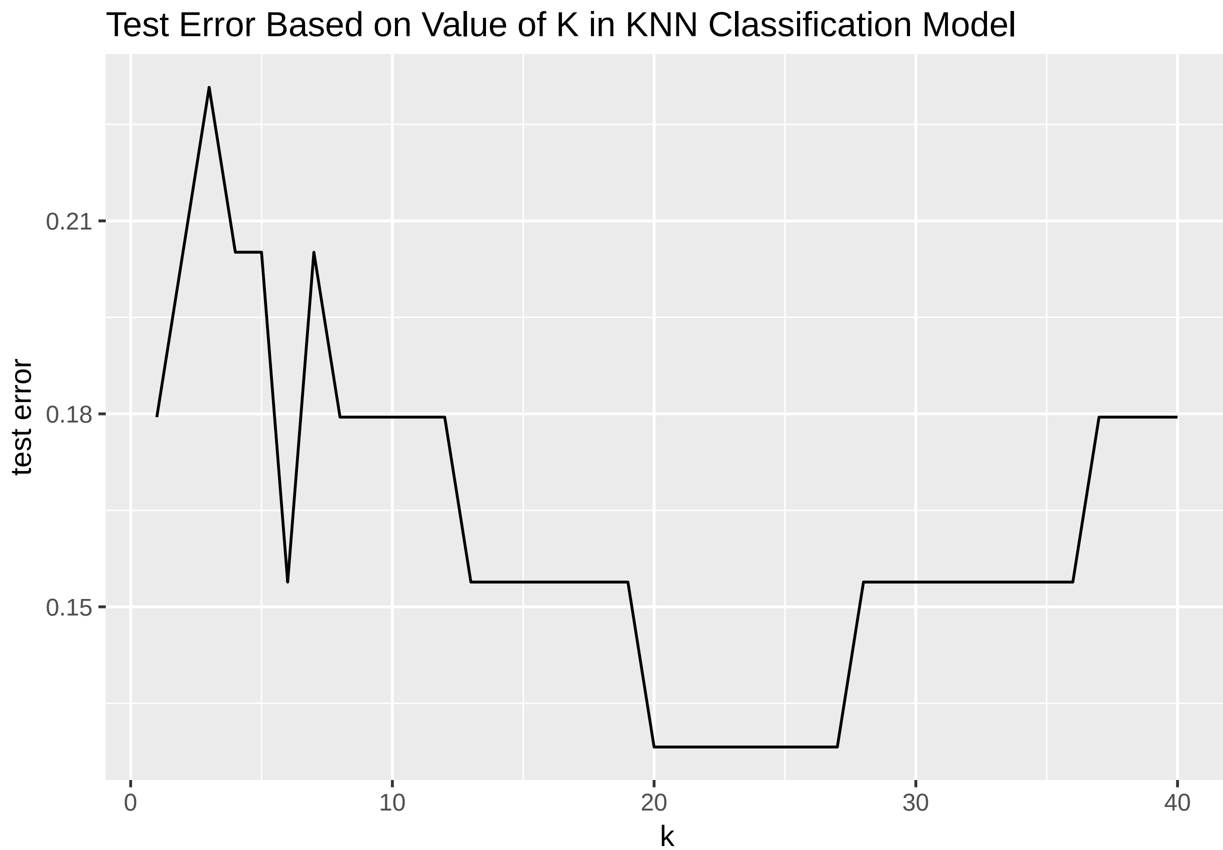
(g) Perform KNN on the training data, with several values of K, in order to predict `mpg01`. Use only the variables that seemed most associated with `mpg01` in (b). What test errors are obtained? Which value of K seems to perform the best on this data set?

```r
test_error = c()
for(i in 1:40){
  knn_model = knn(data.frame(train$origin+train$horsepower),
                  data.frame(test$origin+test$horsepower),
                  train$mpg01, k = i)
  test_error = c(test_error,
                 (table(knn_model, test$mpg01)[1,2] +
                   table(knn_model, test$mpg01)[2,1]) / nrow(test))
}
paste("The lowest test error was obtained when k =",
      which.min(test_error), "with an error of",
      test_error[which.min(test_error)])
```

```
## [1] "The lowest test error was obtained when k = 20 with an error of 0.128205128205128"
```

This is a plot of the test errors as k increased from 1 to 40.

```r
te = data.frame(index = 1:40, "test error" = test_error)
ggplot(te, aes(x = index, y = test_error)) + geom_path() +
  ggtitle("Test Error Based on Value of K in KNN Classification Model") +
  labs(x = "k", y = "test error")
```



The value of K that performed the best on this test set is $k = 20$. It is clear here that as k increased from 20 to 27, its performance did not change. After $k = 27$, test error started to increase.

Question 12: This problem involves writing functions.

(a) Write a function, `Power()`, that prints the result of raising 2 to the 3rd power. In other words, the function should compute 2^3 and print out the results. *Hint: Recall that* `x^a` *raises* `x` *to the power* `a`*. Use the* `print()` *function to output the result.*

```
Power = function(){print(2^3)}
```

(b) Create a new function, `Power2()`, that allows to pass *any* two numbers, `x` and `a`, and prints out the value of `x^a`.

```
Power2 = function(x, a){print(x^a)}
```

(c) Using the `Power2()` function, compute 10^3, 8^17 and 131^3.

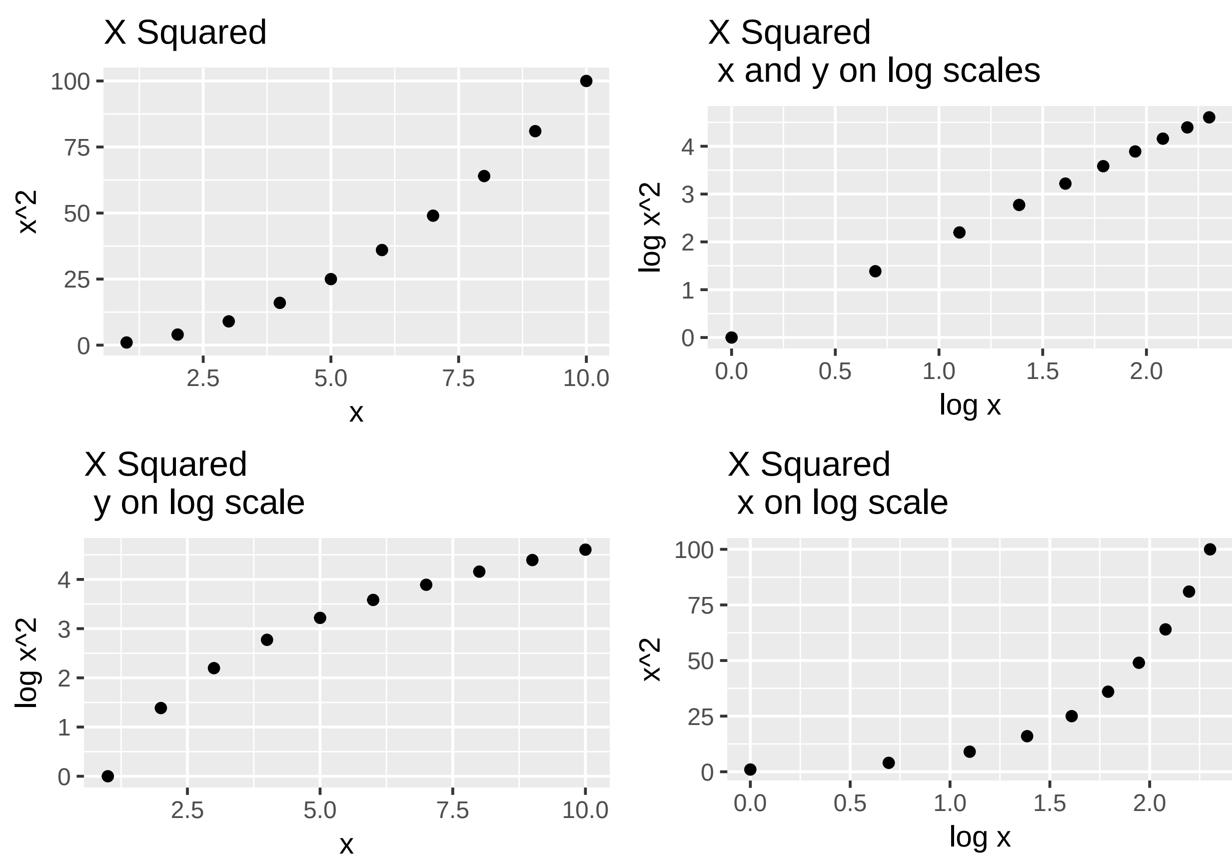
```
Power2(10, 3)

## [1] 1000

Power2(8, 17)

## [1] 2.2518e+15

Power2(131, 3)

## [1] 2248091
```

(d) Now create a new function, `Power3()`, that actually *returns* the result `x^a` as an `R` object, rather than simply printing it to the screen. That is, if the value `x^a` is stored in an object called `result` within the function, this result can be returned using `return()`.

```
Power3 = function(x, a){return(x^a)}
```

(e) Now using the `Power3()` function, create a plot of $f(x) = x^2$. The x-axis should display a range of integers from 1 to 10, and the y-axis should display x^2. Label the axes appropriately and use an appropriate title for the figure. Consider displaying either the x-axis, the y-axis, or both on the log-scale.

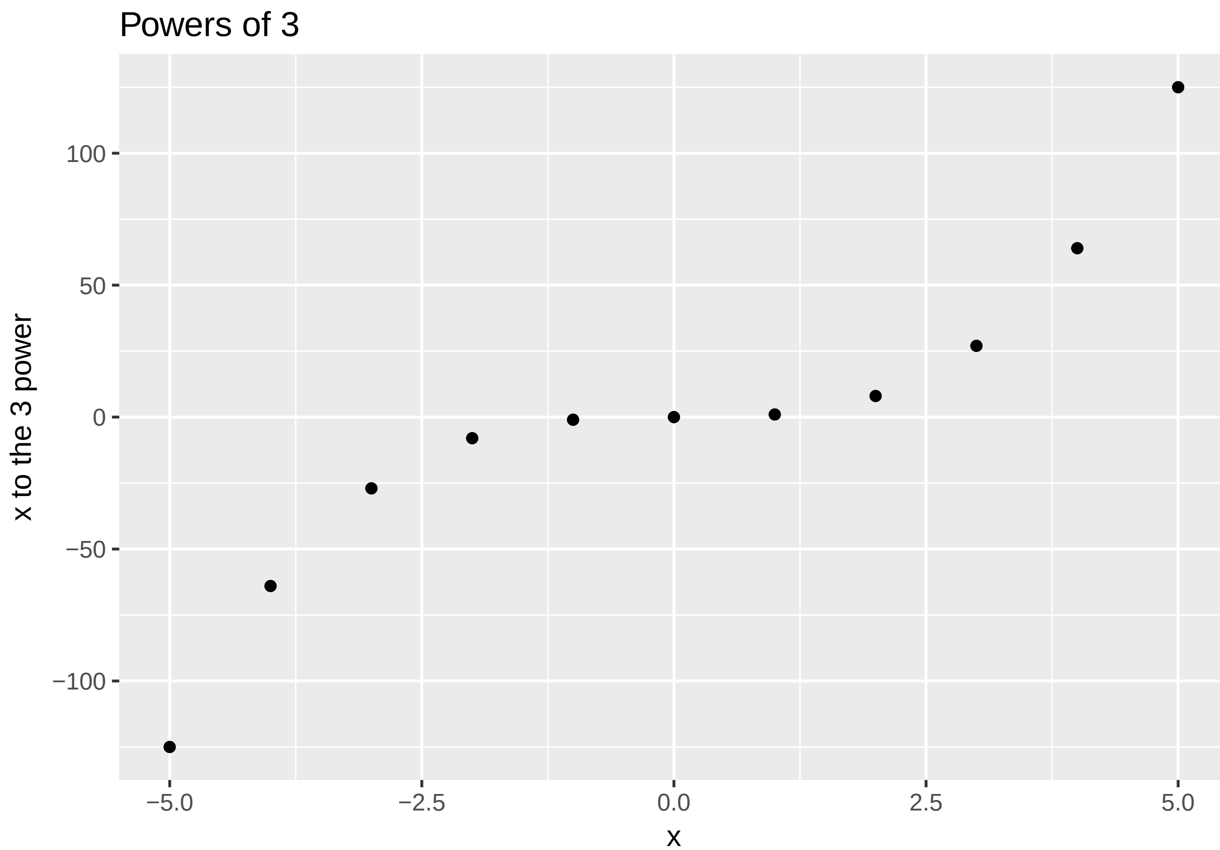
```
df = data.frame(x = c(1:10), y = Power3(c(1:10), 2))
xy = ggplot(data = df, aes(x,y)) + geom_point() +
  labs(x = "x", y = "x^2") + ggtitle("X Squared")
logxlogy = ggplot(data = df, aes(log(x), log(y))) + geom_point() +
  labs(x = "log x", y = "log x^2") + ggtitle("X Squared \n x and y on log scales")
xlogy = ggplot(data = df, aes(x, log(y))) + geom_point() +
  labs(x = "x", y = "log x^2") + ggtitle("X Squared \n y on log scale")
logxy = ggplot(data = df, aes(log(x), y)) + geom_point() +
  labs(x = "log x", y = "x^2") + ggtitle("X Squared \n x on log scale")

grid.arrange(xy, logxlogy, xlogy, logxy, ncol = 2)
```



(f) Create a function, `PlotPower()`, that allows one to create a plot of `x` against `x^a` for a fixed `a` and for a range of values of `x`. For instance, if `PlotPower(1:10,3)` is called, then a plot should be created with an x-axis taking on values $1, 2, \ldots, 10$ and a y-axis taking on values $1^3, 2^3, \ldots, 10^3$.

```
PlotPower = function(x, a){
  df = data.frame(x = x, y = x^a)
  ggplot(data = df, aes(x,y)) + geom_point() +
```

```r
    ggtitle(paste("Powers of", a)) +
    labs(y = paste("x to the", a, "power"))
}
PlotPower(-5:5, 3)
```



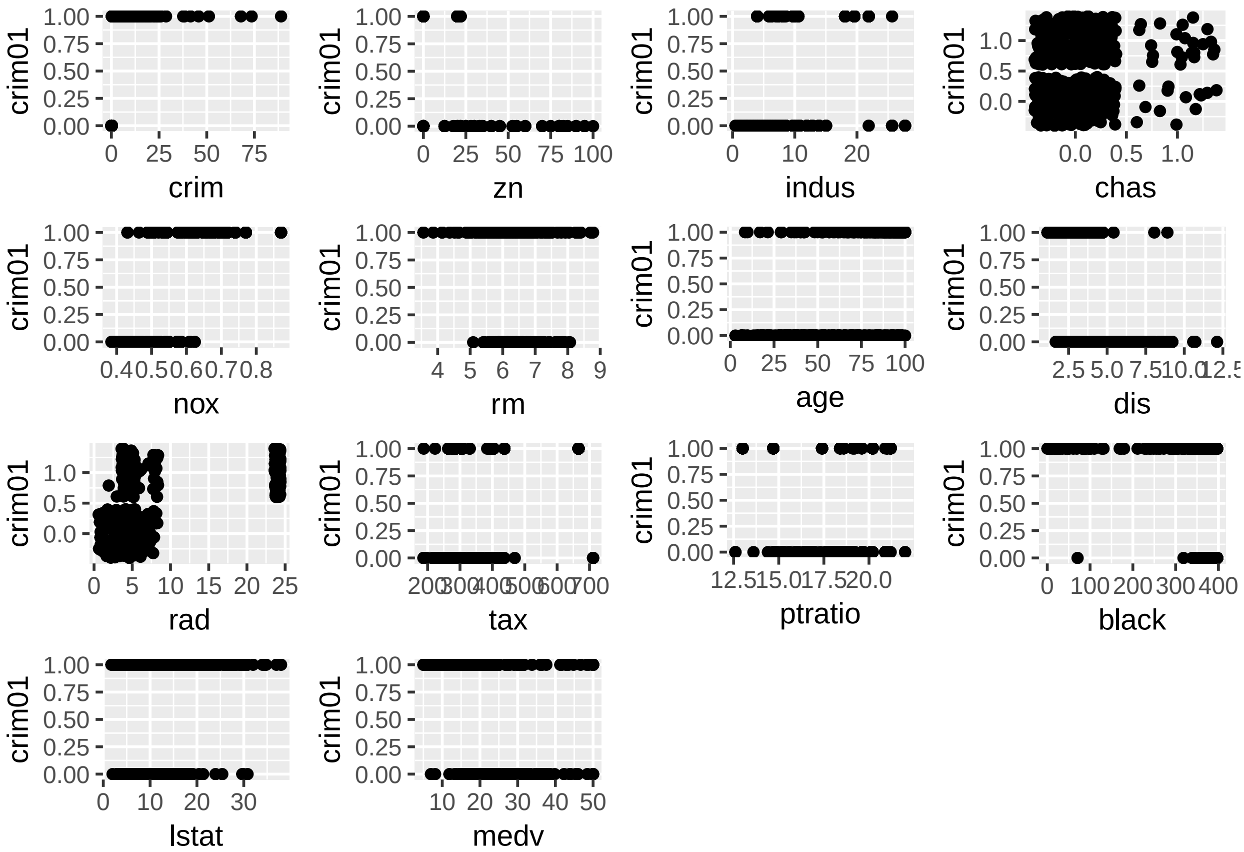
Question 13: Using the `Boston` data set, fit classification models in order to predict whether a given suburn has a crime rate above or below the median. Explore logistic regression, LDA and KNN models given variable subsets of the predictors. Describe findings.

```r
df = Boston
df$crim01 = ifelse(df$crim > median(df$crim), 1, 0)
```

First determine which variables correlate with `crim01`.

```r
g1 = ggplot(data = df, aes(x = crim, y = crim01)) + geom_point()
g2 = ggplot(data = df, aes(x = zn, y = crim01)) + geom_point()
g3 = ggplot(data = df, aes(x = indus, y = crim01)) + geom_point()
g4 = ggplot(data = df, aes(x = chas, y = crim01)) + geom_jitter()
g5 = ggplot(data = df, aes(x = nox, y = crim01)) + geom_point()
g6 = ggplot(data = df, aes(x = rm, y = crim01)) + geom_point()
g7 = ggplot(data = df, aes(x = age, y = crim01)) + geom_point()
g8 = ggplot(data = df, aes(x = dis, y = crim01)) + geom_point()
g9 = ggplot(data = df, aes(x = rad, y = crim01)) + geom_jitter()
g10 = ggplot(data = df, aes(x = tax, y = crim01)) + geom_point()
```

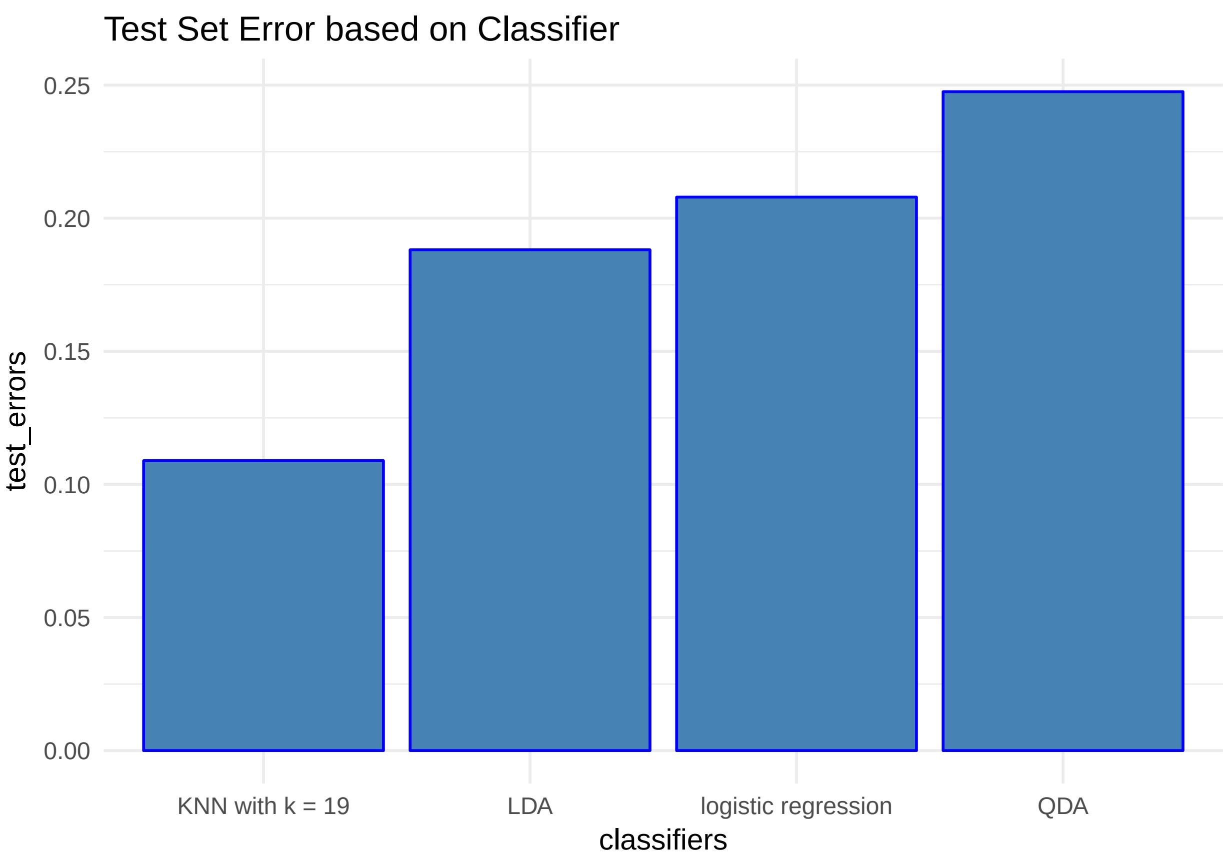
```
g11 = ggplot(data = df, aes(x = ptratio, y =crim01)) + geom_point()
g12 = ggplot(data = df, aes(x = black, y = crim01)) + geom_point()
g13 = ggplot(data = df, aes(x = lstat, y = crim01)) + geom_point()
g14 = ggplot(data = df, aes(x = medv, y = crim01)) + geom_point()

grid.arrange(g1, g2, g3, g4, g5, g6, g7, g8, g9, g10, g11, g12, g13, g14, nrow = 4)
```



The variables that look most correlated with `crim01` are: `zn`, `dis`, and `rad`.

```
set.seed(101)
indexes = sample(1:nrow(df), size = round(0.8 * nrow(df)))
train = df[indexes,]
test = df[-indexes,]
te = c()
model1 = glm(data = train, crim01~zn+dis+rad, family = binomial)
predictions = predict(model1, test, type = "response")
pred = rep(0, nrow(test))
pred[predictions > 0.5] = 1
te = c(te, (table(pred, test$crim01)[1,2] +
              table(pred, test$crim01)[2,1]) / nrow(test))
model2 = lda(data = train, crim01~zn+dis+rad)
predictions = predict(model2, test)
te = c(te, (table(predictions$class, test$crim01)[1,2] +
              table(predictions$class, test$crim01)[2,1]) / nrow(test))
model3 = qda(data = train, crim01~zn+dis+rad)
predictions = predict(model3, test)
te = c(te, (table(predictions$class, test$crim01)[1,2] +
```

```
              table(predictions$class, test$crim01)[2,1]) / nrow(test))
test_error = c()
for(i in 1:40){
  knn_model = knn(data.frame(train$zn+train$dis+train$rad),
                  data.frame(test$zn+test$dis+test$rad),
                  train$crim01, k = i)
  test_error = c(test_error,
                 (table(knn_model, test$crim01)[1,2] +
                   table(knn_model, test$crim01)[2,1]) / nrow(test))
}
te = c(te, test_error[which.min(test_error)])
crim_te = data.frame(classifiers = c("logistic regression", "LDA", "QDA",
                                     paste("KNN with k =", which.min(test_error))),
                  test_errors = te)
ggplot(data = crim_te, aes(x = classifiers, y = test_errors)) +
  geom_col(color = "blue", fill = "steelblue") +
  ggtitle("Test Set Error based on Classifier") + theme_minimal()
```



The classifier that best classified whether crime rate was above or below the median was the KNN classifier, with $k = 19$. The worst performing classifier was the QDA classifier which created about 15% more error.

All of the practice applied exercises in this document are taken from "An Introduction to Statistical Learning, with applications in R" (Springer, 2013) with permission from the authors: G. James, D. Witten, T. Hastie and R. Tibshirani.