Question 13

13. This question should be answered using the `Weekly` data set, which is part of the `ISLR2` package. This data is similar in nature 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.

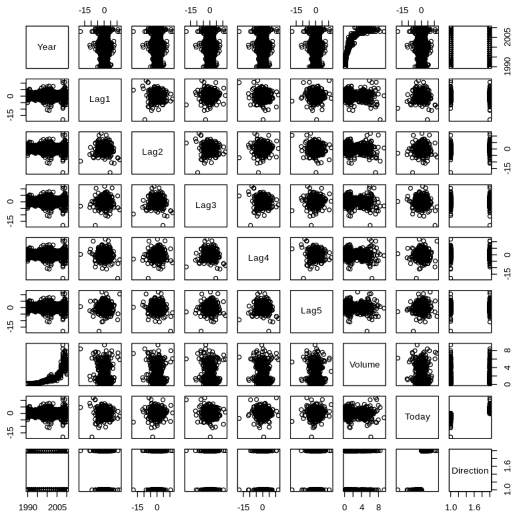
```
install.packages("ISLR2")       # An Introduction to Statistical Learning 2
install.packages("tidyverse")   # for data science
install.packages("caret")       # Classification And REgression Training
install.packages("modelr")      # for modeling
install.packages("corrplot")
```

```
Installing package into '/usr/local/lib/R/site-library'
(as 'lib' is unspecified)

Installing package into '/usr/local/lib/R/site-library'
(as 'lib' is unspecified)

Installing package into '/usr/local/lib/R/site-library'
(as 'lib' is unspecified)

Installing package into '/usr/local/lib/R/site-library'
(as 'lib' is unspecified)

Installing package into '/usr/local/lib/R/site-library'
(as 'lib' is unspecified)
```

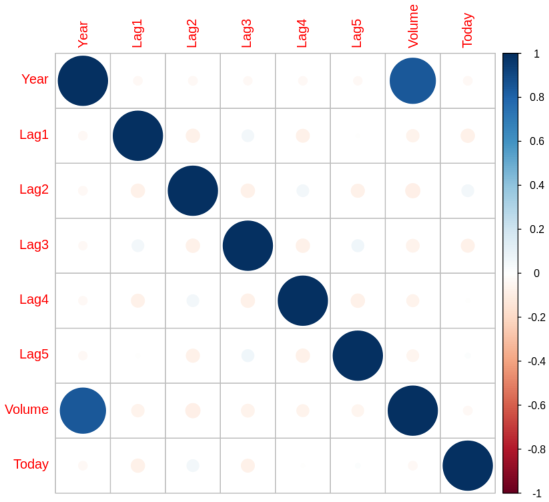
```
library(ISLR2)
library(tidyverse)
library(caret)
library(modelr)
library(corrplot)

lda <- MASS::lda
qda <- MASS::qda
```

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

```
plot(Weekly)
```




```
Weekly %>%                          # Assignment operator
  select_if(is.numeric) %>%
  cor() %>%
  corrplot::corrplot()
```



In principle, no, but it seems that over the years there has been an increase in volume, which may be associated with the more extreme values for the Today returns.

(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?

```
log_reg_weekly <-
  glm(Direction ~ . - Today, data = Weekly, family = "binomial")

summary(log_reg_weekly)
```

```
Call:
glm(formula = Direction ~ . - Today, family = "binomial", data = Weekly)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-1.7071  -1.2578   0.9941   1.0873   1.4665

Coefficients:
             Estimate Std. Error z value Pr(>|z|)
(Intercept) 17.225822  37.890522   0.455   0.6494
Year        -0.008500   0.018991  -0.448   0.6545
Lag1        -0.040688   0.026447  -1.538   0.1239
Lag2         0.059449   0.026970   2.204   0.0275 *
Lag3        -0.015478   0.026703  -0.580   0.5622
Lag4        -0.027316   0.026485  -1.031   0.3024
Lag5        -0.014022   0.026409  -0.531   0.5955
Volume       0.003256   0.068836   0.047   0.9623
---
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: 1486.2  on 1081  degrees of freedom
AIC: 1502.2

Number of Fisher Scoring iterations: 4
```

Just **Lag2** is statistically significant.

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

```r
actual_direction <- Weekly[["Direction"]]
predicted_direction <-
  ifelse(predict(log_reg_weekly, type = "response")  > 0.5,
         "Up", "Down") %>%
  factor(levels = c("Down", "Up"))

caret::confusionMatrix(data = predicted_direction,
                       reference = actual_direction)
```

```
Confusion Matrix and Statistics

               Reference
     Prediction Down  Up
           Down   56  47
           Up    428 558

                   Accuracy : 0.5638
                     95% CI : (0.5338, 0.5935)
        No Information Rate : 0.5556
        P-Value [Acc > NIR] : 0.3024

                      Kappa : 0.0413

      Mcnemar's Test P-Value : <2e-16

                Sensitivity : 0.11570
                Specificity : 0.92231
             Pos Pred Value : 0.54369
             Neg Pred Value : 0.56592
                 Prevalence : 0.44444
             Detection Rate : 0.05142
       Detection Prevalence : 0.09458
          Balanced Accuracy : 0.51901

           'Positive' Class : Down
```

The confusion matrix show us the Type I Error (positive class predicted, but the true condition is negative: 47), with the False Positive Rate of 7.7% (47÷[47+558]) and the Type II Error (negative class predicted, but the true condition is positive: 428) with the False Negative Rate of 88.4% (428÷[428+56]).

(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).

```r
train_weekly <- Weekly %>%
  filter(between(Year, 1990, 2008))

test_weekly <- Weekly %>%
  filter(between(Year, 2009, 2010))

reg_upto2008 <-
  glm(Direction ~ Lag2, data = train_weekly, family = "binomial")

add_pred_direction <- function(df, model) {
  df %>%
  add_predictions(model, type = "response") %>%
  mutate(pred_direction = ifelse(
    pred > 0.5,
    "Up",
    "Down"
  ),
  pred_direction = factor(pred_direction, levels = c("Down", "Up")))
}

test_weekly_reg <-
  test_weekly %>%
  add_pred_direction(reg_upto2008)

caret::confusionMatrix(data = test_weekly_reg[["pred_direction"]],
                       reference = test_weekly_reg[["Direction"]])
```

```
Confusion Matrix and Statistics

          Reference
Prediction Down Up
      Down    9  5
      Up     34 56

               Accuracy : 0.625
                 95% CI : (0.5247, 0.718)
    No Information Rate : 0.5865
    P-Value [Acc > NIR] : 0.2439

                  Kappa : 0.1414

 Mcnemar's Test P-Value : 7.34e-06

            Sensitivity : 0.20930
            Specificity : 0.91803
         Pos Pred Value : 0.64286
         Neg Pred Value : 0.62222
             Prevalence : 0.41346
         Detection Rate : 0.08654
   Detection Prevalence : 0.13462
      Balanced Accuracy : 0.56367

       'Positive' Class : Down
```

(e) Repeat (d) using LDA.

```
lda_upto2008 <-
  MASS::lda(Direction ~ Lag2, data = train_weekly)

test_weekly_lda <-
  test_weekly %>%
  mutate(pred_direction =
           predict(lda_upto2008,
                   newdata = test_weekly,
                   type = "response")[["class"]])

caret::confusionMatrix(data = test_weekly_lda[["pred_direction"]],
                       reference = test_weekly_lda[["Direction"]])
```

```
Confusion Matrix and Statistics

          Reference
Prediction Down Up
      Down    9  5
      Up     34 56

               Accuracy : 0.625
                 95% CI : (0.5247, 0.718)
    No Information Rate : 0.5865
    P-Value [Acc > NIR] : 0.2439

                  Kappa : 0.1414

 Mcnemar's Test P-Value : 7.34e-06

            Sensitivity : 0.20930
            Specificity : 0.91803
         Pos Pred Value : 0.64286
         Neg Pred Value : 0.62222
             Prevalence : 0.41346
         Detection Rate : 0.08654
   Detection Prevalence : 0.13462
      Balanced Accuracy : 0.56367

       'Positive' Class : Down
```

(f) Repeat (d) using QDA.

```r
qda_upto2008 <-
  MASS::qda(Direction ~ Lag2, data = train_weekly)

test_weekly_qda <-
  test_weekly %>%
  mutate(pred_direction =
           predict(qda_upto2008,
                   newdata = test_weekly,
                   type = "response")[["class"]])

caret::confusionMatrix(data = test_weekly_qda[["pred_direction"]],
                       reference = test_weekly_qda[["Direction"]])
```

```
Confusion Matrix and Statistics

          Reference
Prediction Down Up
      Down    0  0
      Up     43 61

               Accuracy : 0.5865
                 95% CI : (0.4858, 0.6823)
    No Information Rate : 0.5865
    P-Value [Acc > NIR] : 0.5419

                  Kappa : 0

 Mcnemar's Test P-Value : 1.504e-10

            Sensitivity : 0.0000
            Specificity : 1.0000
         Pos Pred Value :    NaN
         Neg Pred Value : 0.5865
             Prevalence : 0.4135
         Detection Rate : 0.0000
   Detection Prevalence : 0.0000
      Balanced Accuracy : 0.5000

       'Positive' Class : Down
```

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

```r
train_x_weekly <-
  train_weekly %>%
  select(Lag2)

train_y_weekly <-
  train_weekly[["Direction"]]

test_x_weekly <-
  test_weekly %>%
  select(Lag2)

knn_upto2008 <- class::knn(
  train = train_x_weekly,
  test = test_x_weekly,
  cl = train_y_weekly,
  k = 1
)

caret::confusionMatrix(
  data = knn_upto2008,
  reference = test_weekly[["Direction"]]
)
```

```
Confusion Matrix and Statistics

          Reference
Prediction Down Up
      Down   21 30
      Up     22 31

               Accuracy : 0.5
                 95% CI : (0.4003, 0.5997)
    No Information Rate : 0.5865
    P-Value [Acc > NIR] : 0.9700

                  Kappa : -0.0033

 Mcnemar's Test P-Value : 0.3317

            Sensitivity : 0.4884
            Specificity : 0.5082
         Pos Pred Value : 0.4118
         Neg Pred Value : 0.5849
             Prevalence : 0.4135
         Detection Rate : 0.2019
   Detection Prevalence : 0.4904
      Balanced Accuracy : 0.4983

       'Positive' Class : Down
```

(h) Repeat (d) using naive Bayes.

```
library (e1071)

nb_upto2008 <-
  naiveBayes(Direction ~ Lag2, data = train_weekly)

test_weekly_nb <-
  test_weekly %>%
  mutate(pred_direction =
           predict(nb_upto2008, newdata = test_weekly, type = "class"))

caret::confusionMatrix(data = test_weekly_nb[["pred_direction"]],
                       reference = test_weekly_nb[["Direction"]])
```

```
Confusion Matrix and Statistics

          Reference
Prediction Down Up
      Down    0  0
      Up     43 61

               Accuracy : 0.5865
                 95% CI : (0.4858, 0.6823)
    No Information Rate : 0.5865
    P-Value [Acc > NIR] : 0.5419

                  Kappa : 0

 Mcnemar's Test P-Value : 1.504e-10

            Sensitivity : 0.0000
            Specificity : 1.0000
         Pos Pred Value :    NaN
         Neg Pred Value : 0.5865
             Prevalence : 0.4135
         Detection Rate : 0.0000
   Detection Prevalence : 0.0000
      Balanced Accuracy : 0.5000

       'Positive' Class : Down
```

(i) Which of these methods appears to provide the best results on this data?

LDA and the Logistic Regression has better results than QDA and KNN with K=1.

(j) 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.

Logistic Regression

```
reg2_upto2008 <-
  glm(Direction ~ Lag1 + Lag2 + Volume,
      data = train_weekly, family = "binomial")

reg3_upto2008 <-
  glm(Direction ~ Lag1 + Lag2,
      data = train_weekly, family = "binomial")

reg4_upto2008 <-
  glm(Direction ~ Lag1 + Lag2 + Lag3 + Volume,
      data = train_weekly, family = "binomial")

reg5_upto2008 <-
  glm(Direction ~ Lag1 + Lag2 + Lag3 + Volume,
      data = train_weekly, family = "binomial")

reg6_upto2008 <-
  glm(Direction ~ Lag1 + Lag2 + Lag3 + Volume + I(Volume)^2,
      data = train_weekly, family = "binomial")

reg7_upto2008 <-
  glm(Direction ~ Lag1 + Lag2*Volume + Lag3,
      data = train_weekly, family = "binomial")

test_weekly_reg <-
  test_weekly_reg %>%
  add_predictions(reg2_upto2008, var = "pred_reg2", type = "response") %>%
  add_predictions(reg3_upto2008, var = "pred_reg3", type = "response") %>%
  add_predictions(reg4_upto2008, var = "pred_reg4", type = "response") %>%
  add_predictions(reg4_upto2008, var = "pred_reg5", type = "response") %>%
  add_predictions(reg4_upto2008, var = "pred_reg6", type = "response") %>%
  add_predictions(reg4_upto2008, var = "pred_reg7", type = "response") %>%
  mutate_at(vars(starts_with("pred_reg")),
            ~ factor(ifelse(. > 0.5,
                            "Up",
                            "Down"), levels = c("Down", "Up")))


caret::confusionMatrix(data = test_weekly_reg[["pred_reg2"]],
                       reference = test_weekly_reg[["Direction"]])

caret::confusionMatrix(data = test_weekly_reg[["pred_reg3"]],
                       reference = test_weekly_reg[["Direction"]])

caret::confusionMatrix(data = test_weekly_reg[["pred_reg4"]],
                       reference = test_weekly_reg[["Direction"]])

caret::confusionMatrix(data = test_weekly_reg[["pred_reg5"]],
                       reference = test_weekly_reg[["Direction"]])

caret::confusionMatrix(data = test_weekly_reg[["pred_reg6"]],
                       reference = test_weekly_reg[["Direction"]])

caret::confusionMatrix(data = test_weekly_reg[["pred_reg7"]],
                       reference = test_weekly_reg[["Direction"]])
```

```
Confusion Matrix and Statistics

          Reference
Prediction Down Up
      Down   27 33
      Up     16 28

               Accuracy : 0.5288
                 95% CI : (0.4285, 0.6275)
    No Information Rate : 0.5865
    P-Value [Acc > NIR] : 0.90168

                  Kappa : 0.0821

 Mcnemar's Test P-Value : 0.02227

            Sensitivity : 0.6279
            Specificity : 0.4590
         Pos Pred Value : 0.4500
         Neg Pred Value : 0.6364
             Prevalence : 0.4135
         Detection Rate : 0.2596
   Detection Prevalence : 0.5769
      Balanced Accuracy : 0.5435

       'Positive' Class : Down

Confusion Matrix and Statistics

          Reference
Prediction Down Up
      Down    7  8
      Up     36 53

               Accuracy : 0.5769
                 95% CI : (0.4761, 0.6732)
    No Information Rate : 0.5865
    P-Value [Acc > NIR] : 0.6193

                  Kappa : 0.035

 Mcnemar's Test P-Value : 4.693e-05

            Sensitivity : 0.16279
            Specificity : 0.86885
         Pos Pred Value : 0.46667
         Neg Pred Value : 0.59551
             Prevalence : 0.41346
         Detection Rate : 0.06731
   Detection Prevalence : 0.14423
      Balanced Accuracy : 0.51582

       'Positive' Class : Down

Confusion Matrix and Statistics

          Reference
```

LDA:

```r
lda2_upto2008 <-
  lda(Direction ~ Lag1 + Lag2 + Volume,
      data = train_weekly)

lda3_upto2008 <-
  lda(Direction ~ Lag1 + Lag2,
      data = train_weekly)

lda4_upto2008 <-
  lda(Direction ~ Lag1 + Lag2 + Lag3 + Volume,
      data = train_weekly)

lda5_upto2008 <-
  lda(Direction ~ Lag1 + Lag2 + Lag3 + Volume,
      data = train_weekly)

lda6_upto2008 <-
  lda(Direction ~ Lag1 + Lag2 + Lag3 + Volume + I(Volume)^2,
      data = train_weekly)

lda7_upto2008 <-
  lda(Direction ~ Lag1 + Lag2*Volume + Lag3,
      data = train_weekly)

conf_matrix_lda <- function(lda_model) {
  pred_class <-
    predict(lda_model, test_weekly)[["class"]]

  caret::confusionMatrix(data = pred_class,
                         reference = test_weekly[["Direction"]])
}

conf_matrix_lda(lda2_upto2008)

conf_matrix_lda(lda3_upto2008)

conf_matrix_lda(lda4_upto2008)

conf_matrix_lda(lda5_upto2008)

conf_matrix_lda(lda6_upto2008)

conf_matrix_lda(lda7_upto2008)
```

```
Warning message in lda.default(x, grouping, ...):
“variables are collinear”
Confusion Matrix and Statistics

          Reference
Prediction Down Up
      Down   27 33
      Up     16 28

               Accuracy : 0.5288
                 95% CI : (0.4285, 0.6275)
    No Information Rate : 0.5865
    P-Value [Acc > NIR] : 0.90168

                  Kappa : 0.0821

 Mcnemar's Test P-Value : 0.02227

            Sensitivity : 0.6279
            Specificity : 0.4590
         Pos Pred Value : 0.4500
         Neg Pred Value : 0.6364
             Prevalence : 0.4135
         Detection Rate : 0.2596
   Detection Prevalence : 0.5769
      Balanced Accuracy : 0.5435

       'Positive' Class : Down

Confusion Matrix and Statistics

          Reference
Prediction Down Up
      Down    7  8
      Up     36 53

               Accuracy : 0.5769
                 95% CI : (0.4761, 0.6732)
    No Information Rate : 0.5865
    P-Value [Acc > NIR] : 0.6193

                  Kappa : 0.035

 Mcnemar's Test P-Value : 4.693e-05

            Sensitivity : 0.16279
            Specificity : 0.86885
         Pos Pred Value : 0.46667
         Neg Pred Value : 0.59551
             Prevalence : 0.41346
         Detection Rate : 0.06731
   Detection Prevalence : 0.14423
      Balanced Accuracy : 0.51582

       'Positive' Class : Down

Confusion Matrix and Statistics
```

QDA:

```r
qda2_upto2008 <-
  qda(Direction ~ Lag1 + Lag2 + Volume,
      data = train_weekly)

qda3_upto2008 <-
  qda(Direction ~ Lag1 + Lag2,
      data = train_weekly)

qda4_upto2008 <-
  qda(Direction ~ Lag1 + Lag2 + Lag3 + Volume,
      data = train_weekly)

qda5_upto2008 <-
  qda(Direction ~ Lag1 + Lag2 + Lag3 + Volume,
      data = train_weekly)

qda6_upto2008 <-
  qda(Direction ~ Lag1 + Lag2 + Lag3 + Lag4 + Volume ,
      data = train_weekly)

qda7_upto2008 <-
  qda(Direction ~ Lag1 + Lag2*Volume + Lag3,
      data = train_weekly)

map(
  list(
    qda2_upto2008,
    qda3_upto2008,
    qda4_upto2008,
    qda5_upto2008,
    qda6_upto2008,
    qda7_upto2008
  ),
  conf_matrix_lda
)
```

```
[[1]]
Confusion Matrix and Statistics

          Reference
Prediction Down Up
      Down   31 44
      Up     12 17

               Accuracy : 0.4615
                 95% CI : (0.3633, 0.562)
    No Information Rate : 0.5865
    P-Value [Acc > NIR] : 0.9962

                  Kappa : -3e-04

 Mcnemar's Test P-Value : 3.435e-05

            Sensitivity : 0.7209
            Specificity : 0.2787
         Pos Pred Value : 0.4133
         Neg Pred Value : 0.5862
             Prevalence : 0.4135
         Detection Rate : 0.2981
   Detection Prevalence : 0.7212
      Balanced Accuracy : 0.4998

       'Positive' Class : Down


[[2]]
Confusion Matrix and Statistics

          Reference
Prediction Down Up
      Down    7 10
      Up     36 51

               Accuracy : 0.5577
                 95% CI : (0.457, 0.655)
    No Information Rate : 0.5865
    P-Value [Acc > NIR] : 0.7579156

                  Kappa : -0.0013

 Mcnemar's Test P-Value : 0.0002278

            Sensitivity : 0.16279
            Specificity : 0.83607
         Pos Pred Value : 0.41176
         Neg Pred Value : 0.58621
             Prevalence : 0.41346
         Detection Rate : 0.06731
   Detection Prevalence : 0.16346
      Balanced Accuracy : 0.49943

       'Positive' Class : Down
```

KNN:

```
conf_matrix_knn <- function(k) {
  class::knn(
    train = train_x_weekly,
    test = test_x_weekly,
    cl = train_y_weekly,
    k = k
  ) %>%
    caret::confusionMatrix(data = .,
                           reference = test_weekly[["Direction"]])

}

map(2:15, conf_matrix_knn)
```

```
[[1]]
Confusion Matrix and Statistics

          Reference
Prediction Down Up
      Down   18 29
      Up     25 32

               Accuracy : 0.4808
                 95% CI : (0.3817, 0.5809)
    No Information Rate : 0.5865
    P-Value [Acc > NIR] : 0.9885

                  Kappa : -0.056

 Mcnemar's Test P-Value : 0.6831

            Sensitivity : 0.4186
            Specificity : 0.5246
         Pos Pred Value : 0.3830
         Neg Pred Value : 0.5614
             Prevalence : 0.4135
         Detection Rate : 0.1731
   Detection Prevalence : 0.4519
      Balanced Accuracy : 0.4716

       'Positive' Class : Down


[[2]]
Confusion Matrix and Statistics

          Reference
Prediction Down Up
      Down   15 19
      Up     28 42

               Accuracy : 0.5481
                 95% CI : (0.4474, 0.6459)
    No Information Rate : 0.5865
    P-Value [Acc > NIR] : 0.8152

                  Kappa : 0.0386

 Mcnemar's Test P-Value : 0.2432

            Sensitivity : 0.3488
            Specificity : 0.6885
         Pos Pred Value : 0.4412
         Neg Pred Value : 0.6000
             Prevalence : 0.4135
         Detection Rate : 0.1442
   Detection Prevalence : 0.3269
      Balanced Accuracy : 0.5187

       'Positive' Class : Down
```

Question 14

```
          Reference
```

14. 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.

```
                 95% CI : (0.4665, 0.6641)
```

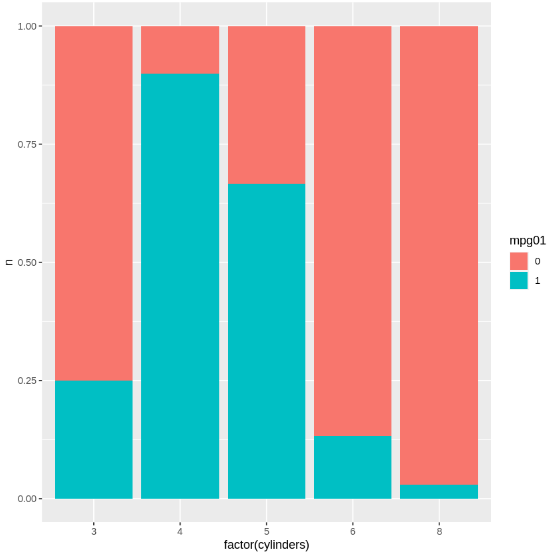
(a) Create a binary variable, `mpg01`, that contains a 1 if `mpg` contains a value above its median, and a 0 if `mpg` contains a value below its median. You can compute the median using the `median()` function. Note you may find it helpful to use the `data.frame()` function to create a single data set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

```
         Neg Pred Value : 0.6333
      Balanced Accuracy : 0.5181
```

```
Auto <- Auto %>%
  mutate(mpg01 = factor(ifelse(mpg > median(mpg),
                             1, 0)))
```

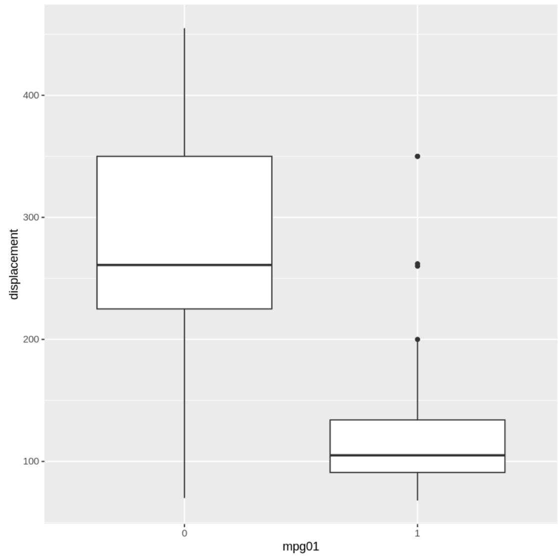
(b) Explore the data graphically in order to investigate the association between `mpg01` and the other features. Which of the other features seem most likely to be useful in predicting `mpg01`? Scatterplots and boxplots may be useful tools to answer this question. Describe your findings.

```
Auto %>%
  count(cylinders, mpg01) %>%
  ggplot(aes(factor(cylinders), n, fill = mpg01)) +
  geom_col(position = position_fill())
```




```
Mcnemar's Test P-Value : 0.5677
```

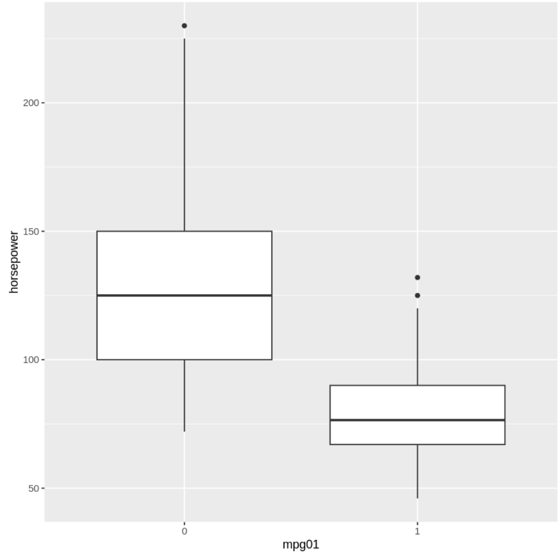
```
ggplot(Auto, aes(mpg01, displacement)) +
  geom_boxplot()
```




```
Mcnemar's Test P-Value : 0.3123
```

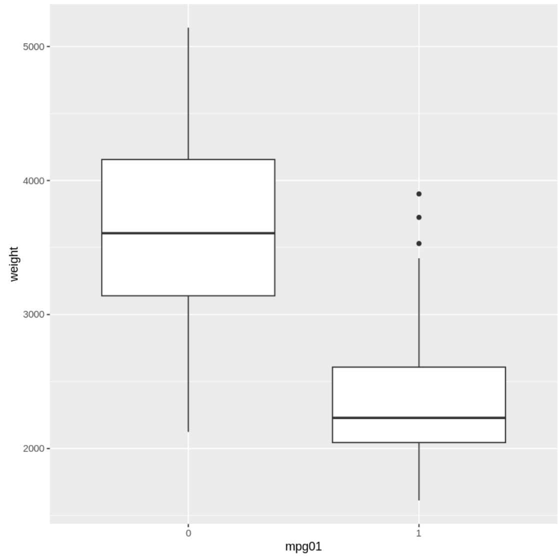
```
qplot(mpg01, horsepower, data = Auto, geom = "boxplot")
```

```
Warning message:
“`qplot()` was deprecated in ggplot2 3.4.0.”
```



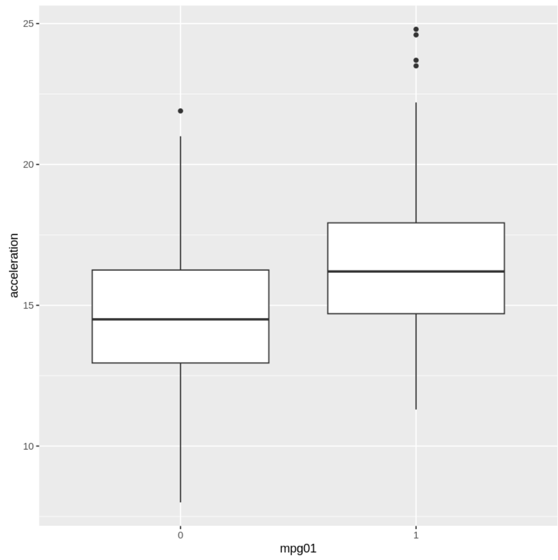
Down 17 20

```
qplot(mpg01, weight, data = Auto, geom = "boxplot")
```



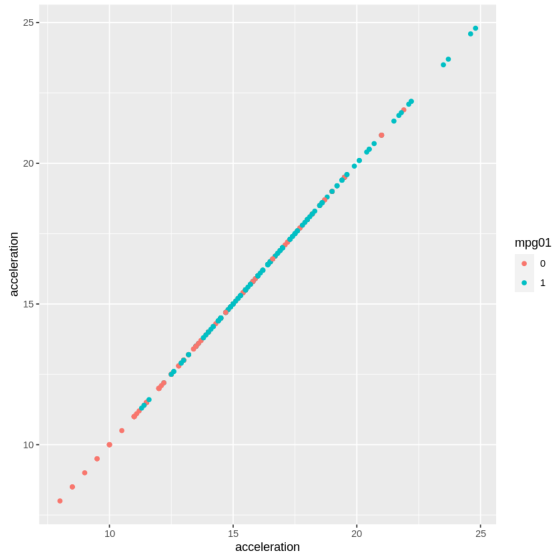
Prediction Down Up

```
qplot(mpg01, acceleration, data = Auto, geom = "boxplot")
```



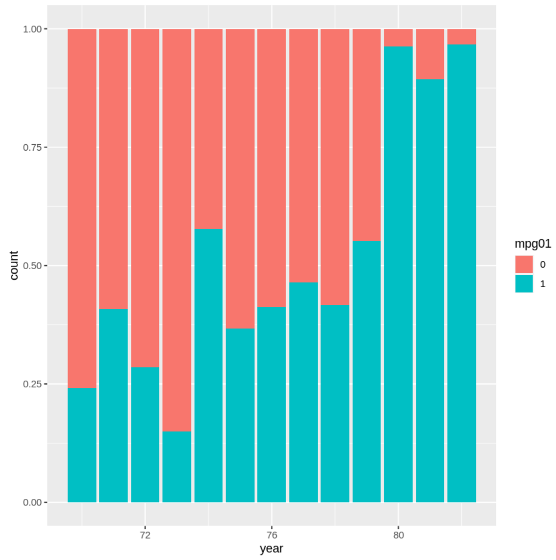
Reference

```
ggplot(Auto, aes(acceleration, acceleration, color = mpg01)) +
  geom_point()
```



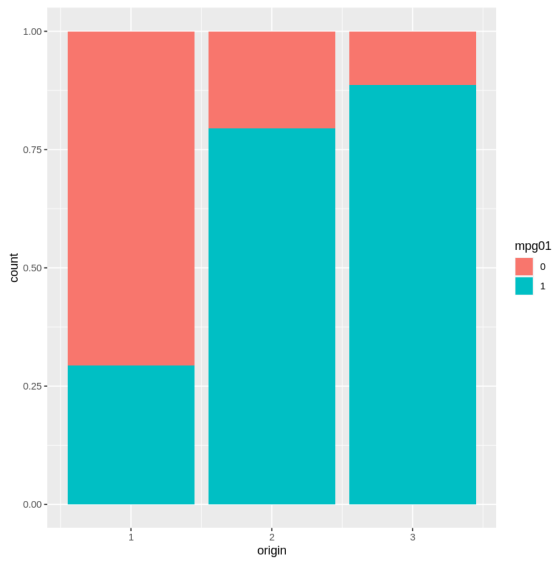
Up 25 41

```
ggplot(Auto, aes(year, fill = mpg01)) +
  geom_bar(position = position_fill())
```

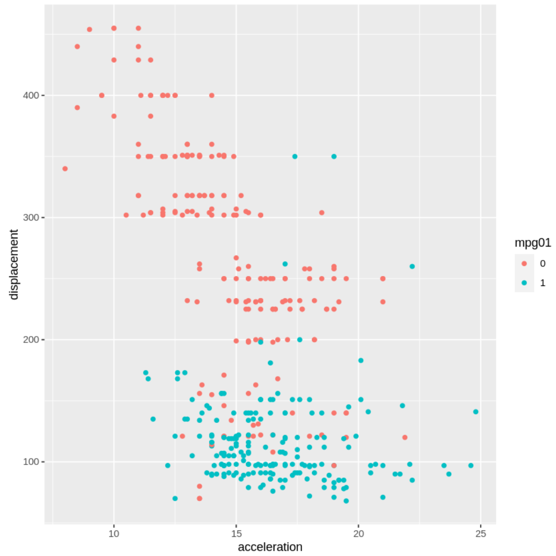


Up 25 41

```
ggplot(Auto, aes(origin, fill = mpg01)) +
  geom_bar(position = position_fill())
```




```
ggplot(Auto, aes(acceleration, displacement, color = mpg01)) +
  geom_point()
```




```
No Information Rate : 0.5865
```

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

```
train_auto <- Auto %>%
  sample_frac(size = 0.5)

test_auto <- Auto %>%
  anti_join(train_auto)
```

```
Joining with `by = join_by(mpg, cylinders, displacement, horsepower, weight,
acceleration, year, origin, name, mpg01)`
```

(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?

```
auto_lda <-
  lda(mpg01 ~ year + acceleration + displacement + weight + horsepower + origin + cylinders,
      data = train_auto)

predictions_auto_lda <-
  predict(auto_lda, newdata = test_auto)[["class"]]

caret::confusionMatrix(data = predictions_auto_lda,
                       reference = test_auto[["mpg01"]])
```

```
Confusion Matrix and Statistics

          Reference
Prediction  0  1
         0 88  3
         1 13 92

               Accuracy : 0.9184
                 95% CI : (0.8708, 0.9526)
    No Information Rate : 0.5153
    P-Value [Acc > NIR] : < 2e-16

                  Kappa : 0.8371

 Mcnemar's Test P-Value : 0.02445

            Sensitivity : 0.8713
            Specificity : 0.9684
         Pos Pred Value : 0.9670
         Neg Pred Value : 0.8762
             Prevalence : 0.5153
         Detection Rate : 0.4490
   Detection Prevalence : 0.4643
      Balanced Accuracy : 0.9199

       'Positive' Class : 0
```

Error rate ~ 8.2%

(e) Perform QDA 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?

```
auto_qda <-
  qda(mpg01 ~ year + acceleration + displacement + weight + horsepower + origin + cylinders,
      data = train_auto)

predictions_auto_qda <-
  predict(auto_qda, newdata = test_auto)[["class"]]

caret::confusionMatrix(data = predictions_auto_qda,
                       reference = test_auto[["mpg01"]])
```

```
Confusion Matrix and Statistics

          Reference
Prediction  0  1
         0 91  9
         1 10 86

               Accuracy : 0.9031
                 95% CI : (0.8528, 0.9406)
    No Information Rate : 0.5153
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.806

 Mcnemar's Test P-Value : 1

            Sensitivity : 0.9010
            Specificity : 0.9053
         Pos Pred Value : 0.9100
         Neg Pred Value : 0.8958
             Prevalence : 0.5153
         Detection Rate : 0.4643
   Detection Prevalence : 0.5102
      Balanced Accuracy : 0.9031

       'Positive' Class : 0
```

Error rate ~ 9.7%

(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?

```
auto_reg <-
  glm(mpg01 ~ year + acceleration + displacement + weight + horsepower + origin + cylinders,
      data = train_auto,
      family = "binomial")

predictions_auto_reg <-
  predict(auto_reg, newdata = test_auto, type = "response") %>%
  map_int(~ifelse(. > 0.5, 1L, 0L)) %>%
  factor()

caret::confusionMatrix(data = predictions_auto_reg,
                       reference = test_auto[["mpg01"]])
```

```
Confusion Matrix and Statistics

          Reference
Prediction  0  1
         0 87  5
         1 14 90

               Accuracy : 0.9031
                 95% CI : (0.8528, 0.9406)
    No Information Rate : 0.5153
    P-Value [Acc > NIR] : < 2e-16

                  Kappa : 0.8065

 Mcnemar's Test P-Value : 0.06646

            Sensitivity : 0.8614
            Specificity : 0.9474
         Pos Pred Value : 0.9457
         Neg Pred Value : 0.8654
             Prevalence : 0.5153
         Detection Rate : 0.4439
   Detection Prevalence : 0.4694
      Balanced Accuracy : 0.9044

       'Positive' Class : 0
```

Error rate ~ 9.7%

(g) Perform naive Bayes 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?

```
auto_nb <-
  naiveBayes(mpg01 ~ year + acceleration + displacement + weight + horsepower + origin + cylinders,
      data = train_auto)

predictions_auto_nb <-
  predict(auto_nb, newdata = test_auto, type = "class")

caret::confusionMatrix(data = predictions_auto_nb,
                       reference = test_auto[["mpg01"]])
```

```
Confusion Matrix and Statistics

          Reference
Prediction  0  1
         0 90  7
         1 11 88

               Accuracy : 0.9082
                 95% CI : (0.8587, 0.9447)
    No Information Rate : 0.5153
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.8164

 Mcnemar's Test P-Value : 0.4795

            Sensitivity : 0.8911
            Specificity : 0.9263
         Pos Pred Value : 0.9278
         Neg Pred Value : 0.8889
             Prevalence : 0.5153
         Detection Rate : 0.4592
   Detection Prevalence : 0.4949
      Balanced Accuracy : 0.9087

       'Positive' Class : 0
```

Error rate ~ 9.2%

(h) 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 do you obtain? Which value of K seems to perform the best on this data set?

```
train_x_auto <-
  train_auto %>%
  select(year,
         acceleration,
         displacement,
         weight,
         horsepower,
         origin,
         cylinders) %>%
  mutate_all(scale)

test_x_auto <-
  test_auto %>%
  select(year,
         acceleration,
         displacement,
         weight,
         horsepower,
         origin,
         cylinders) %>%
  mutate_all(scale)

train_y_auto <- train_auto[["mpg01"]]

knn_auto <- function(k) {
  class::knn(
    train = train_x_auto,
    test = test_x_auto,
    cl = train_y_auto,
    k = k
  ) %>%

  caret::confusionMatrix(data = .,
                       reference = test_auto[["mpg01"]])

}

map(1:15, knn_auto)
```

```
[[1]]
Confusion Matrix and Statistics

          Reference
Prediction  0  1
         0 95  8
         1  6 87

               Accuracy : 0.9286
                 95% CI : (0.8831, 0.9604)
    No Information Rate : 0.5153
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.8569

 Mcnemar's Test P-Value : 0.7893

            Sensitivity : 0.9406
            Specificity : 0.9158
         Pos Pred Value : 0.9223
         Neg Pred Value : 0.9355
             Prevalence : 0.5153
         Detection Rate : 0.4847
   Detection Prevalence : 0.5255
      Balanced Accuracy : 0.9282

       'Positive' Class : 0


[[2]]
Confusion Matrix and Statistics

          Reference
Prediction  0  1
         0 93  7
         1  8 88

               Accuracy : 0.9235
                 95% CI : (0.8769, 0.9565)
    No Information Rate : 0.5153
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.8468

 Mcnemar's Test P-Value : 1

            Sensitivity : 0.9208
            Specificity : 0.9263
         Pos Pred Value : 0.9300
         Neg Pred Value : 0.9167
             Prevalence : 0.5153
         Detection Rate : 0.4745
   Detection Prevalence : 0.5102
      Balanced Accuracy : 0.9236

       'Positive' Class : 0
```

Errors: [1]: ~ 7.1% [2]: ~ 7.6% [3]: ~ 5.6% [4]: ~ 7.6% [5]: ~ 7.6% [6]: ~ 8.6% [7]: ~ 7.1% [8]: ~ 8.1% [9]: ~ 7.1% [10]: ~ 7.6% [11]: ~ 8.1% [12]: ~ 8.6% [13]: ~ 8.1% [14]: ~ 8.1% [15]: ~ 7.6%

```
          Reference
```

Question 15

```
               Accuracy : 0.9439
```

15. This problem involves writing functions.

(a) Write a function, `Power()`, that prints out the result of raising 2 to the 3rd power. In other words, your 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.*

```
         Detection Rate : 0.4796
```

```
Power <- function() {
  print(2^3)
}

Power()
```

```
[1] 8
```

Reference

(b) Create a new function, `Power2()`, that allows you to pass *any* two numbers, `x` and `a`, and prints out the value of `x^a`. You can do this by beginning your function with the line

```
> Power2 <- function(x, a) {
```

You should be able to call your function by entering, for instance,

```
> Power2(3, 8)
```

on the command line. This should output the value of 3^8, namely, $6,561$.

Neg Pred Value : 0.9082

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

Power2(3, 8)
```

```
[1] 6561
```

(c) Using the `Power2()` function that you just wrote, compute 10^3, 8^{17}, and 131^3.

1 10 90

```
Power2(10, 3)
```

```
[1] 1000
```

P-Value [Acc > NIR] : <2e-16

```
Power2(8, 17)
```

```
[1] 2.2518e+15
```

```
Power2(131, 3)
```

```
[1] 2248091
```

Prevalence : 0.5153

(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 you store the value `x^a` in an object called `result` within your function, then you can simply `return()` this result, using the following line:

`return()`

```
return(result)
```

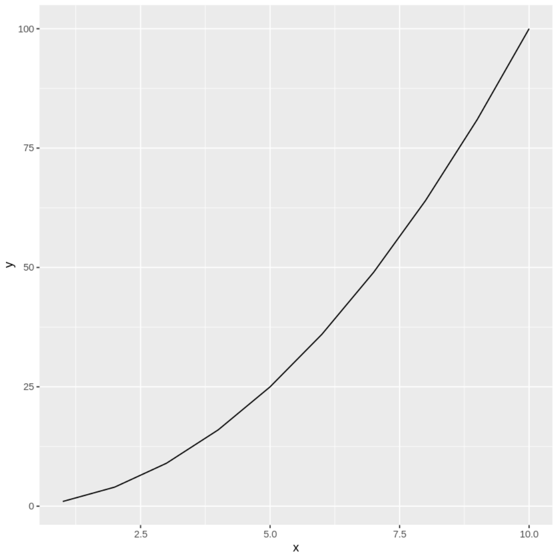
The line above should be the last line in your function, before the } symbol.

```
Power3 <- function(x, a) {
  result <- x^a

  result
}
```

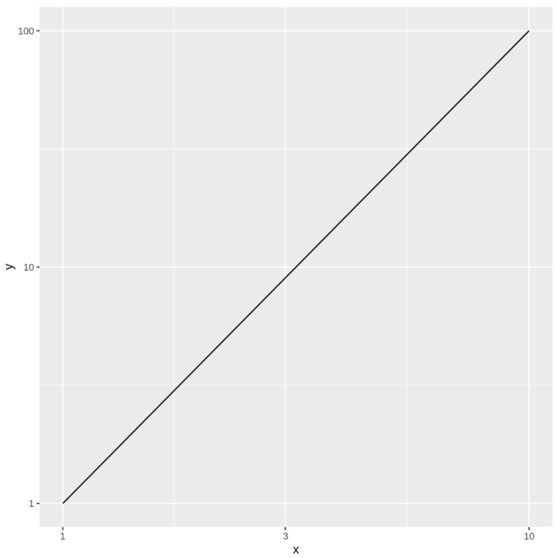
(e) Now using the `Power3()` function, create a plot of $f(x) = x^2$. The x-axis should display a range of integers from 1 to 10, and the y-axis should display x^2. Label the axes appropriately, and use an appropriate title for the figure. Consider displaying either the x-axis, the y-axis, or both on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

Positive Class : 0

```r
plot_data <-
  tibble(
    x = 1:10,
    y = Power3(x, 2)
  )

ggplot(plot_data, aes(x, y)) +
  geom_line()
```




```r
ggplot(plot_data, aes(x, y)) +
  geom_line() +
  scale_y_log10() +
  scale_x_log10()
```

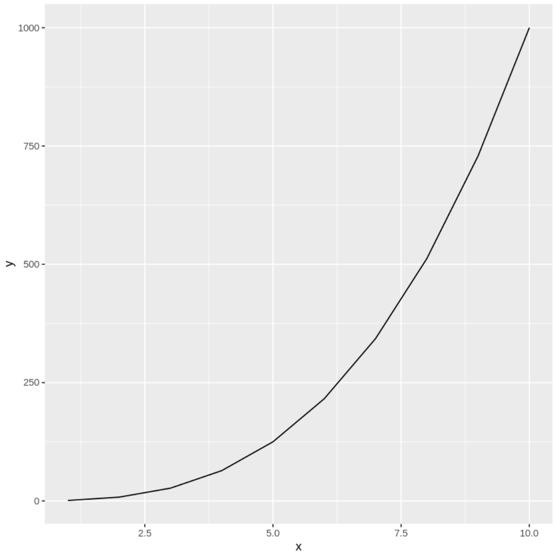


(f) Create a function, `PlotPower()`, that allows you to create a plot of `x` against `x^a` for a fixed `a` and for a range of values of `x`. For instance, if you call

```
> PlotPower(1:10, 3)
```

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) {
  plot_data <-
  tibble(
    x = x,
    y = x^a
  )

ggplot(plot_data, aes(x, y)) +
  geom_line()
}

PlotPower(1:10, 3)
```



Neg Pred Value : 0.8835

Question 16

16. Using the `Boston` data set, fit classification models in order to predict whether a given census tract has a crime rate above or below the median. Explore logistic regression, LDA, naive Bayes, and KNN models using various subsets of the predictors. Describe your findings.

Hint: You will have to create the response variable yourself, using the variables that are contained in the `Boston` *data set.*

Accuracy : 0.9133

```
Boston <- MASS::Boston %>%
   as_tibble() %>%
   mutate(crim01 = ifelse(crim > median(crim),
                           1, 0))

Boston_train <- Boston %>%
  sample_frac(size = 0.5)

Boston_test <- Boston %>%
  anti_join(Boston_train)
```

```
Joining with `by = join_by(crim, zn, indus, chas, nox, rm, age, dis, rad, tax,
ptratio, black, lstat, medv, crim01)`
```

Balanced Accuracy : 0.9142

```
Boston
```

A data.frame: 506 × 13

	crim	zn	indus	chas	nox	rm	age	dis	rad	tax	ptratio
	<dbl>	<dbl>	<dbl>	<int>	<dbl>	<dbl>	<dbl>	<dbl>	<int>	<dbl>	<dbl>
1	0.00632	18.0	2.31	0	0.538	6.575	65.2	4.0900	1	296	15.3
2	0.02731	0.0	7.07	0	0.469	6.421	78.9	4.9671	2	242	17.8
3	0.02729	0.0	7.07	0	0.469	7.185	61.1	4.9671	2	242	17.8
4	0.03237	0.0	2.18	0	0.458	6.998	45.8	6.0622	3	222	18.7
5	0.06905	0.0	2.18	0	0.458	7.147	54.2	6.0622	3	222	18.7
6	0.02985	0.0	2.18	0	0.458	6.430	58.7	6.0622	3	222	18.7
7	0.08829	12.5	7.87	0	0.524	6.012	66.6	5.5605	5	311	15.2