Ch.4 Exercises: Classification

Conceptual

1.

$$
\begin{aligned}
P(x)(1+^{\beta_0+\beta_1 x}) &= e^{\beta_0+\beta_1 x} \\
\frac{P(x)}{\frac{1}{1+e^{\beta_0+\beta_1 x}}} &= e^{\beta_0+\beta_1 x} \\
\frac{P(x)}{1-\frac{e^{\beta_0+\beta_1 x}}{1+e^{\beta_0+\beta_1 x}}} &= e^{\beta_0+\beta_1 x} \\
\frac{P(x)}{1-P(x)} &= e^{\beta_0+\beta_1 x}
\end{aligned}
$$

2.

The terms from (4.12) that do not vary with k:

$$C = \frac{\frac{1}{\sqrt{2\pi}\sigma}\exp(-1/2\sigma^2 x^2)}{\sum_l \pi_l \frac{1}{\sqrt{2\pi}\sigma}\exp(-1/2\sigma^2(x-\mu_l)^2)}$$

Replacing C in (4.12):

$$P_k(x) = \pi_k C \exp(\frac{1}{2\sigma^2}(2\mu_k x - \mu_k^2))$$

Taking logs of both sides:

$$\log(P_k(x)) = \log(\pi_k) + \log(C) + \frac{1}{2\sigma^2}(2\mu_k x - \mu_k^2)$$

Rearranging and disregarding C:

$$\delta_k(x) = x\frac{\mu_k}{\sigma^2} - \frac{\mu_k^2}{2\sigma^2} + \log(\pi_k)$$

3.

Removing the assumption of shared variance terms across all K classes, the terms from (4.12) that do not vary with k:

$$C' = \frac{\frac{1}{\sqrt{2\pi}}}{\sum_l \pi_l \frac{1}{\sqrt{2\pi}\sigma_l}\exp(-1/2\sigma_l^2(x-\mu_l)^2)}$$

Replacing C' in (4.12) and taking logs:

$$P_k(x) = C' \frac{\pi_k}{\sigma_k} \exp(-\frac{1}{2\sigma^2}(x^2 - 2\mu_k x + \mu_k^2))$$

$$\log(P_k(x)) = -\frac{1}{2\sigma_k^2}x^2 + \frac{\mu_k x}{\sigma_k^2} - \frac{\mu_k^2}{2\sigma_k^2} + \log(\frac{\pi_k}{\sigma_k}) + \log(C')$$

As can be seen from the presence of x^2 in the final term, the discriminant is not linear.

4.

(a)

In a uniform distribution, all intervals of the same length are equally probable. Assuming $x \in [0.05, 0.95]$, then intervals: $[x - 0.05, x + 0.05]$, so $length = 0.1$. On average 10% of the observations would be available to make a prediction for the test observation.

(b)

Assuming $x \in [0.05, 0.95]$, $x1_{length} \times x2_{length} = 0.01$. Therefore, 1% of the available observations would be used to make a prediction.

(c)

When p=100; $0.1^p \times 100 = 0.1^{100} \times 100$ of the observations are available.

(d)

As the number of predictors increase, the fraction of observations available to make a prediction is reduced exponentially.

(e)

$Ifp = 1; d(length) = 0.1^{1/1} = 0.1$ $Ifp = 2; d(length) = 0.1^{1/2} = 0.32$ $Ifp = 100; d(length) = 0.1^{1/100} = 0.977$

As p increases the side length converges to 1, and this shows that the hypercube centered around the test observation with 10% of the test observation needs to be nearly the same size as the hypercube with all the observations. It also shows that observations are 'further' from a test observation as p increases; that is they are concentrated near the boundary of the hypercube.

5.

(a)

- LDA better on the test set.
- QDA better on the training set (more flexibility to better fit the data), but worse on the test set due to increased variance.

(b)

- QDA better on training and test sets.

(c)

- In general, QDA tends to be better than LDA when the sample size is large, and where there isn't a common covariance between the classes. As such I would expect QDA to provide a better fit and so provide better predictions.

(d)

- False : LDA will likely provide a better fit for a linear decision boundary than QDA, and so provide a better test error rate. QDA could provide an over-fitting model (due to higher flexibility) that performs well on the training set but worse on the test set(due to higher variance).

6.

(a)

$$P(X) = \frac{\exp(\hat{\beta}_0 + \hat{\beta}_1 X_1 + \hat{\beta}_2 X_2)}{1 + \exp(\hat{\beta}_0 + \hat{\beta}_1 X_1 + \hat{\beta}_2 X_2)}$$
$$P(X) = \frac{\exp(-0.5)}{1 + \exp(-0.5)} = 0.38$$

(b)

$$\log(\frac{P(X)}{1 - P(X)}) = \hat{\beta}_0 + \hat{\beta}_1 X_1 + \hat{\beta}_2 X_2$$
$$\log(\frac{0.5}{1 - 0.5}) = -6 + 0.05X1 + 3.5$$
$$X_1 = 50 hours.$$

7.

$$P(Y = yes|X = 4) = \frac{\pi_{yes} f_{yes}(x)}{\sum_{l=1}^{K} \pi_l f_l(x)} = \frac{\pi_{yes} \exp(-1/2\sigma^2 (x - \mu_{yes})^2)}{\sum_{l=1}^{K} \pi_l \exp(-1/2\sigma^2 (x - \mu_l)^2)}$$
$$P(Y = yes|X = 4) = \frac{0.8 \times \exp(-0.5)}{0.8 \times \exp(-0.5) + 0.2 \times \exp(-16/72)}$$
$$P(Y = yes|X = 4) = 0.75$$

8.

The KNN with K=1 model would fit the training set exactly and so the training error would be zero. This means the test error has to be 36% in order for the average of the errors to be 18%. As model selection is based on performance on the test set, we will choose logistic regression to classify new observations.

9.

(a)

$$Odds = \frac{P(X)}{1 - P(X)}$$
$$P(X) = \frac{0.37}{1.37} = 0.27$$

- 27% of people with odds of 0.37 will default.

(b)

$$Odds = \frac{0.16}{1 - 0.16} = 0.19$$

Applied

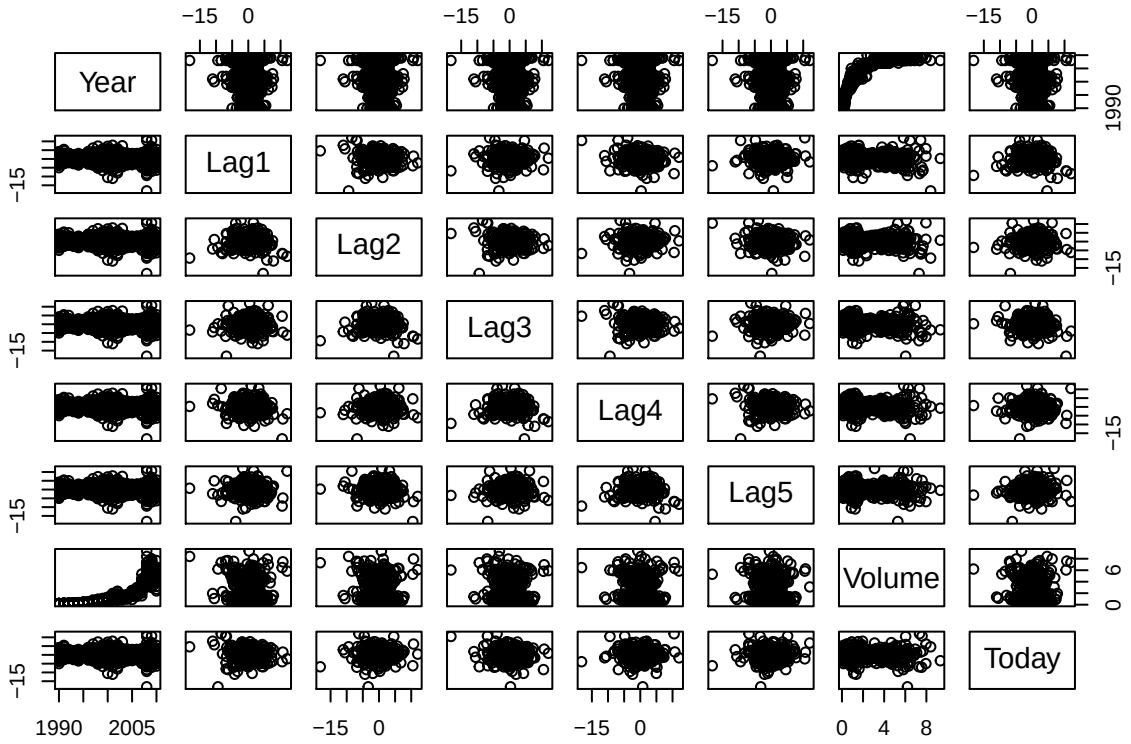
10.

(a)

```
library(ISLR)
summary(Weekly)
```

```
##       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            Today         
##  Min.   :-18.1950   Min.   :-18.1950   Min.   :0.08747   Min.   :-18.1950  
##  1st Qu.: -1.1580   1st Qu.: -1.1660   1st Qu.:0.33202   1st Qu.: -1.1540  
##  Median :  0.2380   Median :  0.2340   Median :1.00268   Median :  0.2410  
##  Mean   :  0.1458   Mean   :  0.1399   Mean   :1.57462   Mean   :  0.1499  
##  3rd Qu.:  1.4090   3rd Qu.:  1.4050   3rd Qu.:2.05373   3rd Qu.:  1.4050  
##  Max.   : 12.0260   Max.   : 12.0260   Max.   :9.32821   Max.   : 12.0260  
##  Direction 
##  Down:484  
##  Up  :605  
##            
##            
##            
##
```

```
# Scatterplot matrix.
pairs(Weekly[,1:8])
```




```
# Correlation matrix.
cor(Weekly[,1:8])
```

```
##                Year         Lag1        Lag2        Lag3         Lag4
## Year    1.00000000 -0.032289274 -0.03339001 -0.03000649 -0.031127923
## Lag1   -0.03228927  1.000000000 -0.07485305  0.05863568 -0.071273876
## Lag2   -0.03339001 -0.074853051  1.00000000 -0.07572091  0.058381535
## Lag3   -0.03000649  0.058635682 -0.07572091  1.00000000 -0.075395865
## Lag4   -0.03112792 -0.071273876  0.05838153 -0.07539587  1.000000000
## Lag5   -0.03051910 -0.008183096 -0.07249948  0.06065717 -0.075675027
## Volume  0.84194162 -0.064951313 -0.08551314 -0.06928771 -0.061074617
## Today  -0.03245989 -0.075031842  0.05916672 -0.07124364 -0.007825873
##                Lag5      Volume        Today
## Year   -0.030519101  0.84194162 -0.032459894
## Lag1   -0.008183096 -0.06495131 -0.075031842
## Lag2   -0.072499482 -0.08551314  0.059166717
## Lag3    0.060657175 -0.06928771 -0.071243639
## Lag4   -0.075675027 -0.06107462 -0.007825873
## Lag5    1.000000000 -0.05851741  0.011012698
## Volume -0.058517414  1.00000000 -0.033077783
## Today   0.011012698 -0.03307778  1.000000000
```

- As can be seen on the scatterplot and correlation matrices, there appears to be a positive correlation between 'Year' and 'Volume' only. From the summary statistics, we can observe that the Lag variables are very similar to each other and 'Today'. There doesn't appear to to be any patterns except for an increase in volume from 1989 to 2001.

(b) (c)

```
logistic_fit = glm(Direction ~ Lag1+Lag2+Lag3+Lag4+Lag5+Volume, data=Weekly, family=binomial)
summary(logistic_fit)
```

```
##
## Call:
## glm(formula = Direction ~ Lag1 + Lag2 + Lag3 + Lag4 + Lag5 +
##     Volume, family = binomial, data = Weekly)
##
## 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
```

- Lag2 is statistically significant.

```
logistic_probs = predict(logistic_fit, type="response")
logistic_preds = rep("Down", 1089) # Vector of 1089 "Down" elements.
logistic_preds[logistic_probs>0.5] = "Up" # Change "Down" to up when probability > 0.5.

# Confusion matrix
attach(Weekly)
table(logistic_preds,Direction)
```

```
##               Direction
## logistic_preds Down  Up
##           Down   54  48
##           Up    430 557
```

- The fraction of days where the predictions are correct is $611/1089 = 56\%$. Therefore, the training error rate is 48%. Of the 987 "Up" predictions the model makes, it is correct $557/987 = 56.4\%$ of the time. Given that there were $605/1089 = 55.6\%$ "Up" days, the model's accuracy when predicting "Up" is only slightly better than random guessing.

(d)

```
# Training observations from 1990 to 2008.
train = (Year<2009)

# Test observations from 2009 to 2010.
Test = Weekly[!train ,]
Test_Direction= Direction[!train]

# Logistic regression on training set.
logistic_fit2 = glm(Direction ~ Lag2, data=Weekly, family=binomial, subset=train)

# Predictions on the test set.
logistic_probs2 = predict(logistic_fit2,Test, type="response")
logistic_preds2 = rep("Down", 104)
logistic_preds2[logistic_probs2>0.5] = "Up"

# Confusion matrix.
table(logistic_preds2,Test_Direction)
```

```
##                Test_Direction
## logistic_preds2 Down Up
##            Down    9  5
##            Up     34 56
```

- The model makes correct predictions on 65/104= 62.5% of the days.

(e)

```
# Using LDA.
library(MASS)
lda_fit = lda(Direction ~ Lag2, data=Weekly, subset=train)
#lda_fit

# Predictions on the test set.
lda_pred = predict(lda_fit,Test)
lda_class = lda_pred$class

# Confusion matrix.
table(lda_class,Test_Direction)
```

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

- The lda model makes correct predictions 65/104 = 62.5% of the days.

(f)

```
# Using QDA.
qda_fit = qda(Direction ~ Lag2, data=Weekly, subset=train)
qda_pred = predict(qda_fit,Test)
qda_class = qda_pred$class
table(qda_class,Test_Direction)
```

```
##          Test_Direction
## qda_class Down Up
##      Down    0  0
##      Up     43 61
```

- QDA model's TPR=1 and precision(correct predictions)=0.58, which is no better than guessing each day is "Up".

(g)

```
# Using KNN
library(class)
set.seed(1)
train_X = Weekly[train,3]
test_X = Weekly[!train,3]
train_direction = Direction[train]

# Changing from vector to matrix by adding dimensions
dim(train_X) = c(985,1)
dim(test_X) = c(104,1)

# Predictions for K=1
knn_pred = knn(train_X, test_X, train_direction, k=1)
table(knn_pred, Test_Direction)
```

```
##         Test_Direction
## knn_pred Down Up
##     Down   21 30
##     Up     22 31
```

- KNN with K=1 in correct in its predictions for 50% of the days.

(h)

- Logistic regression, LDA give the exact same confusion matrix. The TPR = 0.92, Precision = 0.62, TNR = 0.21 and NPV(Negative Predictive Value) = 0.64.
- For KNN with K=1, the TPR = 0.51, Precision = 0.58, TNR = 0.48 and FPV = 0.41.
- The logistic and LDA models provide the best results, particularly for predicting "Up" days.

(i)

```
# Using KNN and K=3
knn_pred2 = knn(train_X, test_X, train_direction, k=3)
table(knn_pred2, Test_Direction)
```

```
##          Test_Direction
## knn_pred2 Down Up
##      Down   16 19
##      Up     27 42
```

```
# Using KNN and K=9
knn_pred3 = knn(train_X, test_X, train_direction, k=6)
table(knn_pred3, Test_Direction)
```

```
##          Test_Direction
## knn_pred3 Down Up
##      Down   15 20
##      Up     28 41
```

- Higher K values shown an improvement in the overall correct predictions (59/104) made by a KNN model when using Lag2 as the only predictor.

```
# Using LDA with all Lag values
lda_fit2 = lda(Direction ~ Lag1+Lag2+Lag3+Lag4+Lag5, data=Weekly, subset=train)

# Predictions on the test set
lda_pred2 = predict(lda_fit2,Test)
lda_class2 = lda_pred2$class

# Confusion matrix
table(lda_class2,Test_Direction)
```

```
##           Test_Direction
## lda_class2 Down Up
##       Down    9 13
##       Up     34 48
```

- No real improvement using LDA when using all Lag variables.

```
# Using logistic with lag2 and lag2^2
logistic_fit3 = glm(Direction ~ Lag2 + I(Lag2^2), data=Weekly, family=binomial, subset=train)

# Predictions on the test set.
logistic_probs3 = predict(logistic_fit3,Test, type="response")
logistic_preds3 = rep("Down", 104)
logistic_preds3[logistic_probs3>0.5] = "Up"

# Confusion matrix.
table(logistic_preds3,Test_Direction)
```

```
##                Test_Direction
## logistic_preds3 Down Up
##            Down    8  4
##            Up     35 57
```

- Results are similar to Lag2 only.

```r
# Using logistic with lag2 and lag1^2
logistic_fit4 = glm(Direction ~ Lag2 + I(Lag1^2), data=Weekly, family=binomial, subset=train)

# Predictions on the test set.
logistic_probs4 = predict(logistic_fit4,Test, type="response")
logistic_preds4 = rep("Down", 104)
logistic_preds4[logistic_probs4>0.5] = "Up"

# Confusion matrix.
table(logistic_preds4,Test_Direction)
```

```
##                Test_Direction
## logistic_preds4 Down Up
##            Down    8  2
##            Up     35 59
```

- This model is correct on 64% of the days.

11. (a)

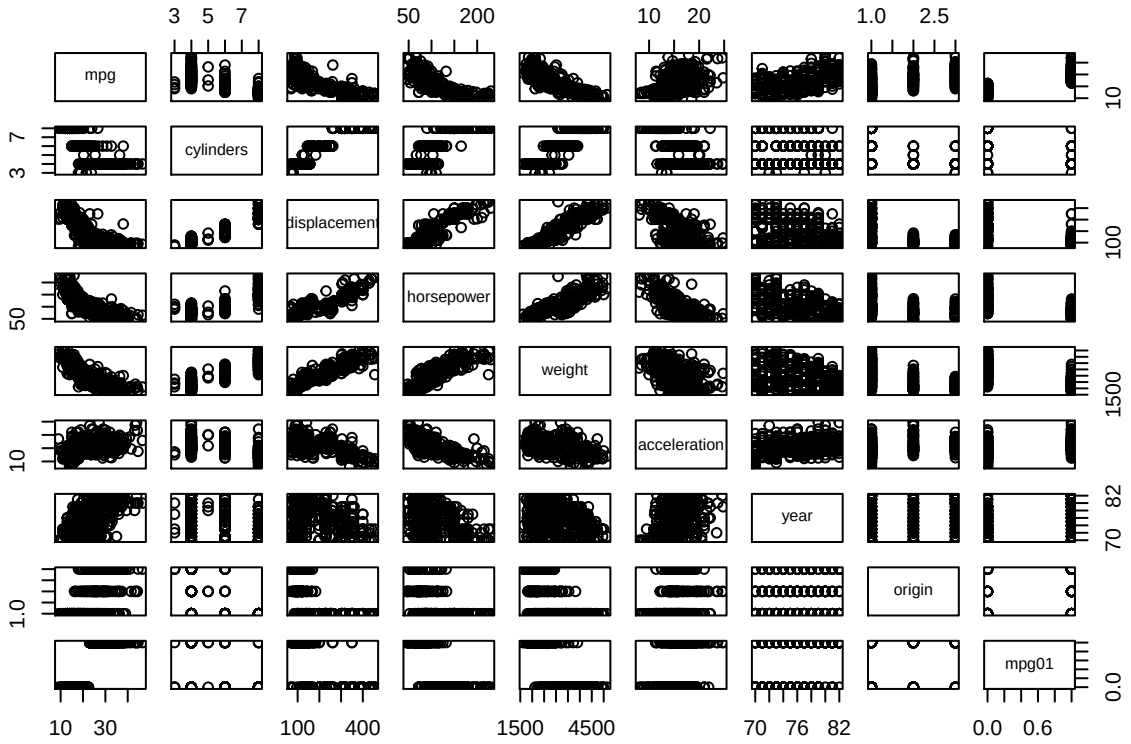
```r
# Dataframe with "Auto" data and empty "mpg01" column
df = Auto
df$mpg01 = NA
median_mpg = median(df$mpg)

# Loop
for(i in 1:dim(df)[1]){
  if (df$mpg[i] > median_mpg){
    df$mpg01[i] = 1
  }else{
    df$mpg01[i] = 0
  }
}
```

```r
# function to move a column to end of dataframe.
movetolast = function(data, move) {
  data[c(setdiff(names(data), move), move)]
}
```

(b)

```r
df = movetolast(df, c("name"))
pairs(df[,1:9])
```




```
cor(df[,1:9])
```

```
##                     mpg  cylinders displacement horsepower     weight
## mpg           1.0000000 -0.7776175   -0.8051269 -0.7784268 -0.8322442
## cylinders    -0.7776175  1.0000000    0.9508233  0.8429834  0.8975273
## displacement -0.8051269  0.9508233    1.0000000  0.8972570  0.9329944
## horsepower   -0.7784268  0.8429834    0.8972570  1.0000000  0.8645377
## weight       -0.8322442  0.8975273    0.9329944  0.8645377  1.0000000
## acceleration  0.4233285 -0.5046834   -0.5438005 -0.6891955 -0.4168392
## year          0.5805410 -0.3456474   -0.3698552 -0.4163615 -0.3091199
## origin        0.5652088 -0.5689316   -0.6145351 -0.4551715 -0.5850054
## mpg01         0.8369392 -0.7591939   -0.7534766 -0.6670526 -0.7577566
##              acceleration       year     origin      mpg01
## mpg             0.4233285  0.5805410  0.5652088  0.8369392
## cylinders      -0.5046834 -0.3456474 -0.5689316 -0.7591939
## displacement   -0.5438005 -0.3698552 -0.6145351 -0.7534766
## horsepower     -0.6891955 -0.4163615 -0.4551715 -0.6670526
## weight         -0.4168392 -0.3091199 -0.5850054 -0.7577566
## acceleration    1.0000000  0.2903161  0.2127458  0.3468215
## year            0.2903161  1.0000000  0.1815277  0.4299042
## origin          0.2127458  0.1815277  1.0000000  0.5136984
## mpg01           0.3468215  0.4299042  0.5136984  1.0000000
```

- There is a strong positive correlation between `mpg` and `mpg01`, and a strong negative correlation between `cylinders`, `displacement`, `weight`, `horsepower` and `mpg01`.

- I will use these variables except `mpg.mpg` was used to separate observations into `mpg01` values and so using it can lead to perfectly separating test observations.

(c)

```
# Training and Test data
require(caTools)
```

```
## Loading required package: caTools
```

```
## Warning: package 'caTools' was built under R version 3.6.2
```

```
set.seed(123)
sample_data = sample.split(df$mpg, SplitRatio = 0.70)
train2 = subset(df, sample_data==TRUE)
test2 = subset(df, sample_data==FALSE)
```

(d)

```
# LDA model
lda_fit3 = lda(mpg01 ~ cylinders+displacement+horsepower+weight, data=train2)

# Predictions and confusion matrix
lda_pred3 = predict(lda_fit3,test2)
predictions = lda_pred3$class
actual = test2$mpg01
table(predictions,actual)
```

```
##            actual
## predictions  0  1
##           0 48  4
##           1  8 44
```

- The test error of this model is 11.5%.

(e)

```
# QDA model
qda_fit2 = qda(mpg01 ~ cylinders+displacement+horsepower+weight, data=train2)
qda_pred2 = predict(qda_fit2,test2)
predictions = qda_pred2$class
table(predictions,actual)
```

```
##            actual
## predictions  0  1
##           0 50  4
##           1  6 44
```

- The QDA model has a test error of 9.6%.

(f)

```r
# Logistic regression model
logistic_fit5 = glm(mpg01 ~ cylinders+displacement+horsepower+weight, data=train2, family=binomial)

logistic_probs5 = predict(logistic_fit5,test2, type="response")
logistic_preds5 = rep(0, length(test2$mpg01))
logistic_preds5[logistic_probs5>0.5] = 1

table(logistic_preds5,actual)
```

```
##                 actual
## logistic_preds5  0  1
##                0 50  4
##                1  6 44
```

- The logistic model has a 9.6% test error rate.

(g)

```r
# Train, Test and response matrices.
train2_matrix = data.matrix(train2[,c("cylinders","displacement","weight","horsepower")])
test2_matrix = data.matrix(test2[,c("cylinders","displacement","weight","horsepower")])
train2_y = data.matrix(train2$mpg01)
test2_y = data.matrix(test2$mpg01)

# K=1 and predictions
knn_pred4 = knn(train2_matrix, test2_matrix, train2_y, k=1)
table(knn_pred4, test2_y)
```

```
##          test2_y
## knn_pred4  0  1
##         0 45  9
##         1 11 39
```

- KNN with K=1 has a test error of 20%.

```r
# K=3 and predictions
knn_pred5 = knn(train2_matrix, test2_matrix, train2_y, k=3)
table(knn_pred5, test2_y)
```

```
##          test2_y
## knn_pred5  0  1
##         0 45  5
##         1 11 43
```

- KNN with K=3 is has a test error of 15%.

```r
# K=9 and predictions
knn_pred6 = knn(train2_matrix, test2_matrix, train2_y, k=10)
table(knn_pred6, test2_y)
```

```
##           test2_y
## knn_pred6  0  1
##          0 45  4
##          1 11 44
```

- K=10 leads to a slight improvement in test error(14.4%), with diminishing returns as K gets even higher.

12. (a) (b) (c)

```
Power2 = function(x,a){
  print(x^a)
}
Power2(3,8)
```

```
## [1] 6561
```

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

```
## [1] 1000
```

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

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

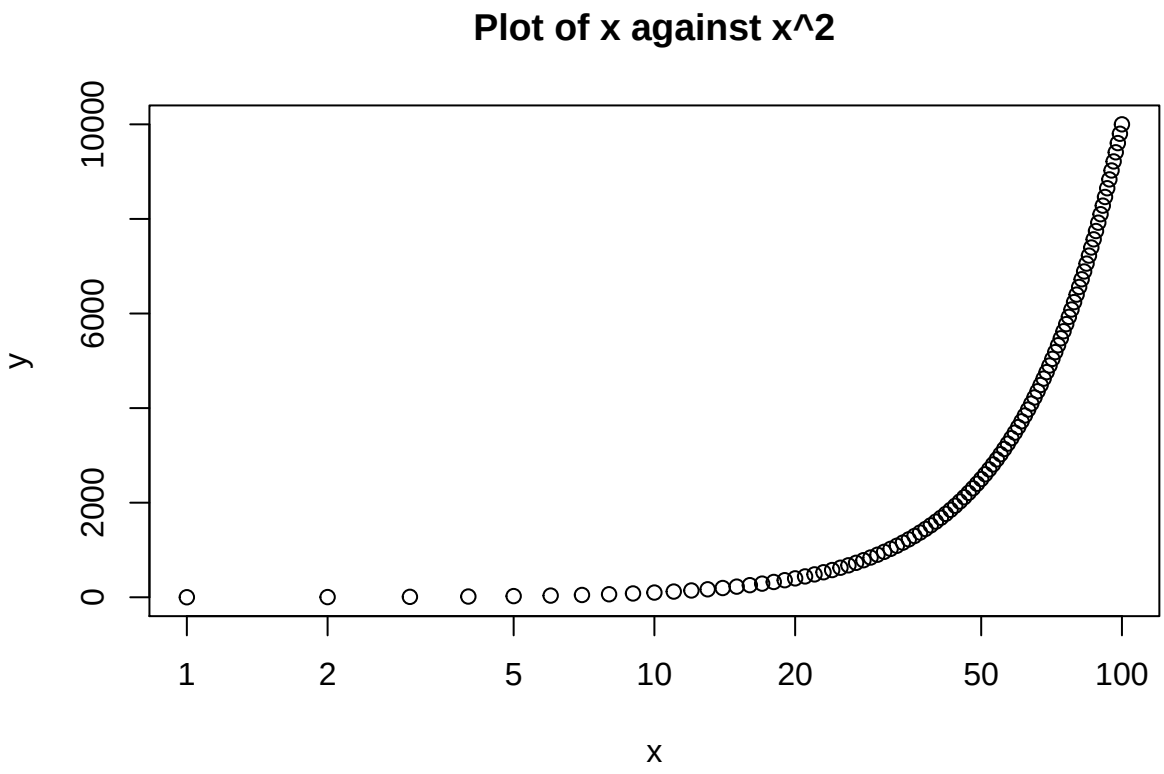
```
Power2(131,3)
```

```
## [1] 2248091
```

(d) (e)

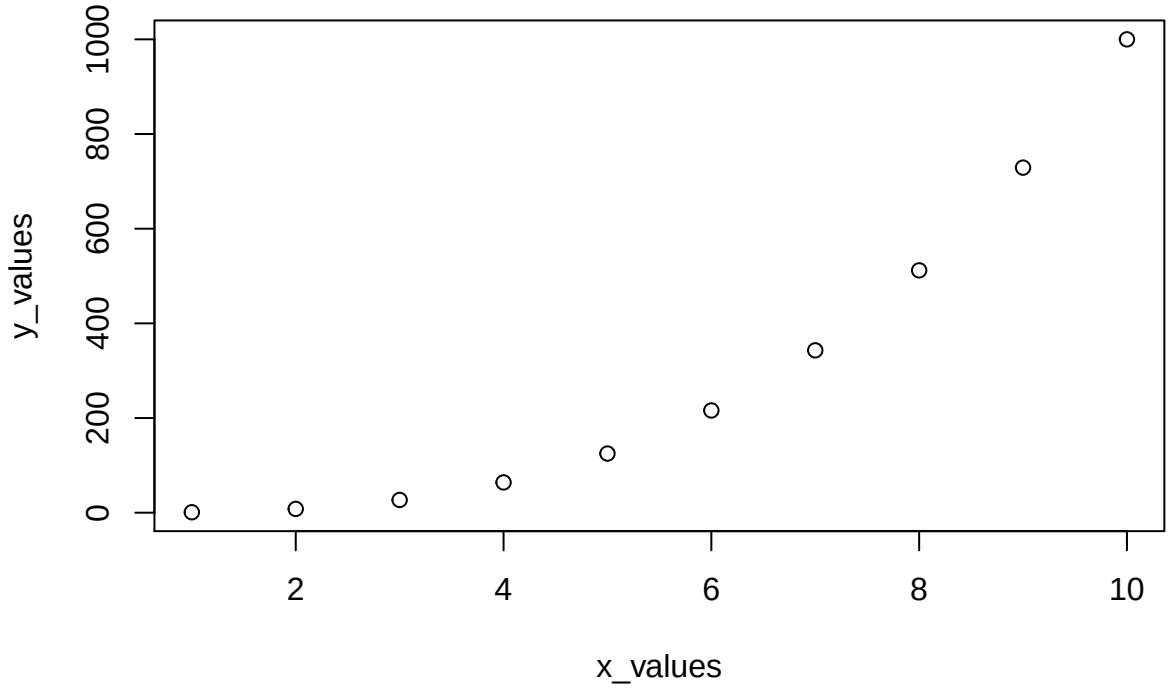
```
Power3 = function(x,a){
  result = x^a
  return(result)
}

# Plot f(x) = x^2
x = 1:100
y = Power3(x,2)
plot(x,y,log="x", main="Plot of x against x^2")
```



(f)

```
PlotPower = function(x,a){
  x_values = x
  y_values = x^a
  plot(x_values, y_values)
}
PlotPower(1:10,3)
```



13.

```
#library(ISLR)
#library(MASS)
#library(class)
boston_df = Boston

#Add 1 to column if CRIM > median and 0 otherwise
median_crim = median(Boston$crim)
boston_df$crim01 = with(ifelse(crim>median_crim, 1, 0), data=Boston)
```

```
#Correlation between crim01 and other variables.
cor(boston_df$crim01,boston_df)
```

```
##           crim        zn     indus       chas       nox         rm       age
## [1,] 0.4093955 -0.436151 0.6032602 0.07009677 0.7232348 -0.1563718 0.6139399
##             dis       rad       tax   ptratio      black     lstat       medv
## [1,] -0.6163416 0.6197862 0.6087413 0.2535684 -0.3512109 0.4532627 -0.2630167
##      crim01
## [1,]      1
```

```
#Training and Test sets
require(caTools)
set.seed(123)
boston_sample = sample.split(boston_df$crim01, SplitRatio = 0.80)
```

```r
boston_train = subset(boston_df, boston_sample==TRUE)
boston_test = subset(boston_df, boston_sample==FALSE)
```

```r
# Logistic regression using all variables except CHAS
# and crim(using crim will lead to perfect separation).
boston_lr = glm(crim01 ~.-chas-crim , data=boston_train, family=binomial)
summary(boston_lr)
```

```
##
## Call:
## glm(formula = crim01 ~ . - chas - crim, family = binomial, data = boston_train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.8978  -0.1705  -0.0002   0.0025   3.5340
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept) -39.431670   7.164710  -5.504 3.72e-08 ***
## zn           -0.102660   0.038683  -2.654 0.007958 **
## indus        -0.069107   0.047124  -1.467 0.142511
## nox          48.381844   8.123415   5.956 2.59e-09 ***
## rm            0.084088   0.798160   0.105 0.916096
## age           0.023671   0.013566   1.745 0.081014 .
## dis           0.820828   0.255752   3.209 0.001330 **
## rad           0.631930   0.164647   3.838 0.000124 ***
## tax          -0.005362   0.002838  -1.889 0.058826 .
## ptratio       0.334091   0.133126   2.510 0.012087 *
## black        -0.007506   0.005230  -1.435 0.151230
## lstat         0.076788   0.051698   1.485 0.137462
## medv          0.149509   0.077409   1.931 0.053430 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 560.06  on 403  degrees of freedom
## Residual deviance: 169.97  on 391  degrees of freedom
## AIC: 195.97
##
## Number of Fisher Scoring iterations: 9
```

```r
boston_probs = predict(boston_lr,boston_test, type="response")
boston_preds = rep(0, length(boston_test$crim01))
boston_preds[boston_probs>0.5] = 1
actual = boston_test$crim01

table(boston_preds, actual)
```

```
##             actual
## boston_preds  0  1
##            0 46  5
##            1  5 46
```

- Test error rate of 9.8%. Same accuracy when predicting 0(crime below median) or 1(crime above median).

```
# Logistic regression using zn, nox, dis, rad and ptratio.
# These variables were statistically significant in the previous model.
boston_lr2 = glm(crim01 ~ zn+nox+dis+rad+ptratio, data=boston_train, family=binomial)
boston_probs2 = predict(boston_lr2,boston_test, type="response")
boston_preds2 = rep(0, length(boston_test$crim01))
boston_preds2[boston_probs2>0.5] = 1
actual = boston_test$crim01

table(boston_preds2, actual)
```

```
##              actual
## boston_preds2  0  1
##             0 45  9
##             1  6 42
```

- Test error rises to 14.7% when using this subset.

```
# LDA
boston_lda = lda(crim01 ~.-chas-crim , data=boston_train)
boston_preds2 = predict(boston_lda, boston_test)
table(boston_preds2$class, actual)
```

```
##    actual
##      0  1
##   0 49 12
##   1  2 39
```

- Test error rate of 13.7%.

```
# QDA
boston_qda = qda(crim01 ~.-chas-crim , data=boston_train)
boston_preds3 = predict(boston_qda, boston_test)
table(boston_preds3$class, actual)
```

```
##    actual
##      0  1
##   0 50  9
##   1  1 42
```

- Test error rate of 9.8%. More accurate when predicting 0.

```
#KNN
#Training and Test sets without crim and chas
boston_train2 = data.matrix(subset(boston_train,select=-c(crim,chas)))
boston_test2 = data.matrix(subset(boston_test,select=-c(crim,chas)))

train2_y = data.matrix(boston_train[,15])
test2_y = data.matrix(boston_test[,15])
```

```r
# KNN-1 and predictions
boston_knn1 = knn(boston_train2, boston_test2, train2_y, k=1)
table(boston_knn1, test2_y)
```

```
##            test2_y
## boston_knn1  0  1
##           0 47  1
##           1  4 50
```

- Test error rate of 4.9%.

```r
# KNN-3 and predictions
boston_knn2 = knn(boston_train2, boston_test2, train2_y, k=3)
table(boston_knn2, test2_y)
```

```
##            test2_y
## boston_knn2  0  1
##           0 45  1
##           1  6 50
```

- Higher test error rate of 6.9%.

```r
# KNN-10 and predictions
boston_knn3 = knn(boston_train2, boston_test2, train2_y, k=10)
table(boston_knn3, test2_y)
```

```
##            test2_y
## boston_knn3  0  1
##           0 43  5
##           1  8 46
```

- Much higher test error rate of 11.7%.
- Higher K values result in the test error rate increasing. KNN-1 gives the best performance, therefore the Bayes decision boundary for the data set is likely non-linear.
- QDA and Logistic regression perform better than LDA but worse than KNN.

```r
#KNN-1 using indus, nox, age, dis, rad, tax (strongly correlated variables with crim01)
boston_train3 = data.matrix(subset(boston_train,select=c(indus,nox,age,dis,rad,tax)))
boston_test3 = data.matrix(subset(boston_test,select=c(indus,nox,age,dis,rad,tax)))

boston_knn4 = knn(boston_train3, boston_test3, train2_y, k=1)
table(boston_knn4, test2_y)
```

```
##            test2_y
## boston_knn4  0  1
##           0 42  1
##           1  9 50
```

- The test error is worse when using these variables.

```r
#KNN-2 using nox and rad - most statistically significant in the first logistic model.
boston_train4 = data.matrix(subset(boston_train,select=c(nox,rad)))
boston_test4 = data.matrix(subset(boston_test,select=c(nox,rad)))

boston_knn5 = knn(boston_train4, boston_test4, train2_y, k=2)
table(boston_knn5, test2_y)
```

```
##            test2_y
## boston_knn5  0  1
##           0 49  2
##           1  2 49
```

- Test error of 4%, which is the lowest among the tested models and subsets of variables. KNN with values of K=1,2 or 3 give the best results.