03 Classification Homework

Megha Pandit

February 5, 2019

6. Suppose we collect data for a group of students in a statistics class with variables X1 =hours studied, X2 =undergrad GPA, and Y = receive an A. We fit a logistic regression and produce estimated coefficient, $\hat{\beta}_0 = -6$, $\hat{\beta}_1 = 0.05$, $\hat{\beta}_2 = 1$.

(a) Estimate the probability that a student who studies for 40 h and has an undergrad GPA of 3.5 gets an A in the class.

The probability that a student who studies for 40h and has an undergrad GPA of 3.5 gets an A is given by:

$$P(Y=1) = \frac{e^{-6+0.05*40+1*3.5}}{1+e^{-6+0.05*40+1*3.5}} = \frac{e^{-0.5}}{1+e^{-0.5}} = \frac{0.6065}{1.6065} = 0.3775$$

The probability of the student getting an A is 37.75%.

(b) How many hours would the student in part (a) need to study to have a 50% chance of getting an A in the class?

We could write the logistic regression equation in terms of log odds as below:

$$log(\frac{P(Y=1)}{1-P(Y=1)}) = \hat{\beta}_0 + \hat{\beta}_1 X_1 + \hat{\beta}_2 X_2$$

$$log(\frac{0.5}{1-0.5}) = -6 + 0.05X_1 + 1*3.5$$

$$-6 + 0.05X_1 + 3.5 = 0$$

$$X_1 = \frac{2.5}{0.05} = 50$$

Therefore, the student would need to study for 50hrs to have a 50% chance of getting an A.

8. Suppose that we take a data set, divide it into equally-sized training and test sets, and then try out two different classification procedures. First we use logistic regression and get an error rate of 20% on the training data and 30% on the test data. Next we use 1-nearest neighbors (i.e. K = 1) and get an average error rate (averaged over both test and training data sets) of 18%. Based on these results, which method should we prefer to use for classification of new observations? Why?

Since the KNN is performed with K=1, it will yield a training error rate of zero while also overfitting the data. Since the error rate averaged over both training and test data is 18%, the test error rate must have been 36%. This is clearly higher than the test error rate that logistic regression produces. Therefore, I would choose logistic regression to classify these observations.

9. This problem has to do with odds.

(a) On average, what fraction of people with an odds of 0.37 of defaulting on their credit card payment will in fact default?

The fraction who will default is:

$$\frac{0.37}{1+0.37} = \frac{0.37}{1.37} = 0.27$$

or 27%.

(b) Suppose that an individual has a 16% chance of defaulting on her credit card payment. What are the odds that she will default?

The odds that the individual will default are:

$$\frac{0.16}{1-0.16} = \frac{0.16}{0.84} = 0.19$$

or 19%

10. This question should be answered using the Weekly data set, which is part of the ISLR 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.

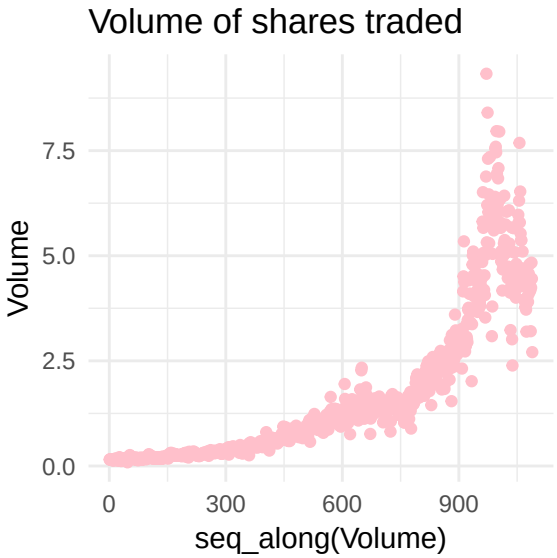
(a) Produce some numerical and graphical summaries of the Weekly data. Do there appear to be any patterns?

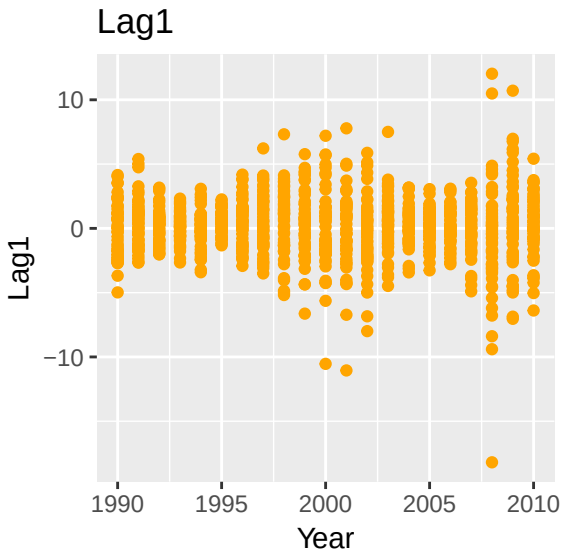
```
library(ISLR)
data("Weekly")

summary(Weekly)
```

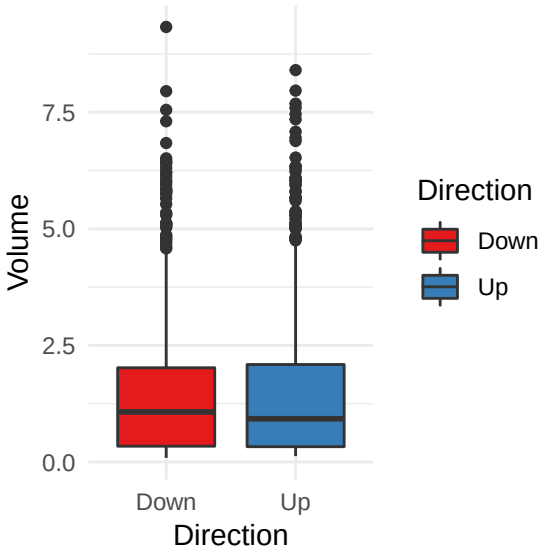
```
##       Year           Lag1               Lag2               Lag3
##  Min.   :1990   Min.   :-18.1950   Min.   :-18.1950   Min.   :-18.1950
##  1st Qu.:1995   1st Qu.: -1.1540   1st Qu.: -1.1540   1st Qu.: -1.1580
##  Median :2000   Median :  0.2410   Median :  0.2410   Median :  0.2410
##  Mean   :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
##  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
```

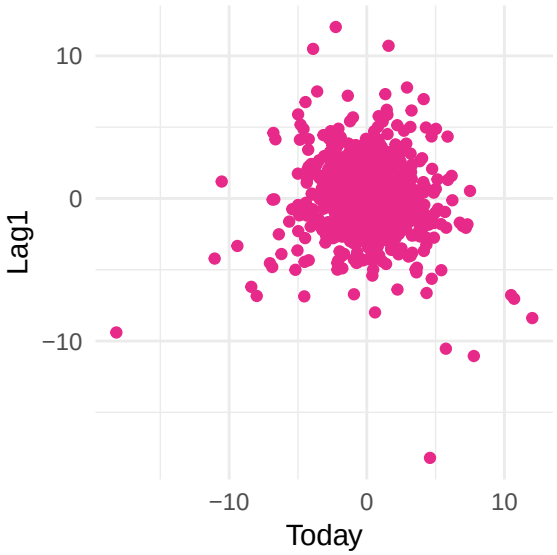
```
ggplot(data = Weekly) +
  aes(y = Volume, x = seq_along(Volume)) +
  geom_point(color = "pink") +
  theme_minimal()+
  ggtitle("Volume of shares traded")
```

```
ggplot(Weekly)+
  aes(x = Year, y = Lag1)+
  geom_point(color = "orange")+
  ggtitle("Lag1")

ggplot(data = Weekly) +
  aes(x = Direction, y = Volume, fill = Direction) +
  geom_boxplot() +
  scale_fill_brewer(palette = "Set1") +
  theme_minimal()

ggplot(data = Weekly) +
  aes(x = Today, y = Lag1) +
  geom_point(color = "#e7298a") +
  theme_minimal()
```









The mean values of all the Lags are similar and their minimum and maximum values are the same, although they

are percentage returns for different lengths of time. The volume of shares traded has been increasing over the years. There is no relationship between any of the lags and today's 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?

```
fit <- glm(Direction ~ Lag1 + Lag2 + Lag3 + Lag4 + Lag5 + Volume, data = Weekly,
           family = binomial(link = logit))
summary(fit)
```

```
##
## Call:
## glm(formula = Direction ~ Lag1 + Lag2 + Lag3 + Lag4 + Lag5 +
##     Volume, family = binomial(link = logit), 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 appears to be the only statistically significant predictor. All the others have small t-statistics and large p-values.

(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.

```
pred <- predict(fit, type = "response")
pred_glm <- rep("Down", length(pred))
pred_glm[pred > 0.5] <- "Up"

confusionMatrix(data = as.factor(pred_glm), reference = Weekly$Direction)
```

```
## Confusion Matrix and Statistics
##
```

```
##           Reference
## Prediction Down  Up
##       Down   54  48
##       Up    430 557
##
##                Accuracy : 0.5611
##                  95% CI : (0.531, 0.5908)
##     No Information Rate : 0.5556
##     P-Value [Acc > NIR] : 0.369
##
##                   Kappa : 0.035
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.11157
##             Specificity : 0.92066
##          Pos Pred Value : 0.52941
##          Neg Pred Value : 0.56434
##              Prevalence : 0.44444
##          Detection Rate : 0.04959
##    Detection Prevalence : 0.09366
##       Balanced Accuracy : 0.51612
##
##        'Positive' Class : Down
##
```

Overall, the model has an accuracy of 56.11%. "Down" is considered the positive class in this case. The positive predictive value is 0.529. This implies that 52.9% out of all the "Down" predictions made by the model are actually "Down". And 56.4% of all the "Up" predictions made by the model are actually "Up". Regarding the mistakes made by the model, the False Positive Rate or the percentage of "Down" predictions which are actually "Up" is 0.4705 or 47.05%. The False Negative Rate or the percentage of "Up" predictions which are actually "Down" is 43.56%.

(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).

```
train <- Weekly[Weekly$Year < 2009,]
test <- Weekly[Weekly$Year > 2008,]
fit1 <- glm(Direction ~ Lag2, data = train, family = binomial)
summary(fit1)
```

```
##
## 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
```

```
pred1 <- predict(fit1, newdata = test, type = "response")
pred1_glm <- rep("Down", length(pred1))
pred1_glm[pred1 > 0.5] <- "Up"
confusionMatrix(as.factor(pred1_glm), reference = test$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
##
```

The overall fraction of correct predictions for the held out data is $(9 + 56)/(9 + 5 + 34 + 56) = 65/104 = 0.625$, which is stated as the accuracy of the model, in the summary for the confusion matrix.

(e) Repeat (d) using LDA.

```
lda1 <- lda(Direction ~ Lag2, data = train)
lda1
```

```
## 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
```

```
pred.lda <- predict(lda1, newdata = test)
confusionMatrix(pred.lda$class, reference = test$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
##
```

The overall fraction of correct predictions through LDA is also 0.625 or 62.5%.

(f) Repeat (d) using QDA.

```
qda1 <- qda(Direction ~ Lag2, data = train)
qda1
```

```
## 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
```

```
pred.qda <- predict(qda1, newdata = test)
confusionMatrix(as.factor(pred.qda$class), reference = test$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
##
```

The overall fraction of correct predictions in the case od QDA is 0.5865 or 58.65%, which is lower than that for LDA and Logistic Regression.

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

```
train.x <- cbind(train[,3])
test.x <- cbind(test[,3])
train.direction <- train[,9]
test.direction <- test[,9]

set.seed(1)
knn1 <- knn(train.x, test.x, train.direction, k = 1)
table(knn1, test$Direction)
```

```
##
## knn1   Down Up
##   Down   21 30
##   Up     22 31
```

The overall fraction of correct predictions in the case of KNN with k = 1, is 52/104 = 50%.

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

Logistic regression and LDA, both give the best results among all the approaxches. The accuracy is 62.5%.

(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.

Out of all the models with different combinations of predictors and interactions, the one that has only Lag2 as a predictor gives the best results for logistic regression. Therefore, we proceed to try different combinations of predictors for the other methods: LDA, QDA and KNN.

```
train <- Weekly[Weekly$Year < 2009,]
test <- Weekly[Weekly$Year > 2008,]

lda2 <- lda(Direction ~ Lag1 + Lag2 + Lag2*Lag1, data = train)
lda2
```

```
## Call:
## lda(Direction ~ Lag1 + Lag2 + Lag2 * Lag1, data = train)
##
## Prior probabilities of groups:
##      Down        Up
## 0.4477157 0.5522843
##
## Group means:
##              Lag1        Lag2  Lag1:Lag2
## Down  0.289444444 -0.03568254 -0.8014495
## Up   -0.009213235  0.26036581 -0.1393632
##
## Coefficients of linear discriminants:
##                     LD1
## Lag1      -0.285484602
## Lag2       0.295080109
## Lag1:Lag2  0.009629381
```

```
pred.lda2 <- predict(lda2, newdata = test)
confusionMatrix(pred.lda2$class, reference = test$Direction)
```

```
## 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
##
```

The LDA model above gives an accuracy of 57.7%. Similar to logistic regression results, LDA results for the model with just Lag2 as a predictor were much better than with any other combinations of predictors and even with interactions.

```
qda2 <- qda(Direction ~ Lag2 + I(Lag2^2), data = train)
qda2
```

```
## Call:
## qda(Direction ~ Lag2 + I(Lag2^2), data = train)
##
## Prior probabilities of groups:
##      Down        Up
## 0.4477157 0.5522843
##
## Group means:
##             Lag2 I(Lag2^2)
## Down -0.03568254  4.828121
## Up    0.26036581  5.428657
```

```
pred.qda2 <- predict(qda2, newdata = test)
confusionMatrix(as.factor(pred.qda2$class), reference = test$Direction)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction Down Up
##       Down    7  3
##       Up     36 58
##
##                Accuracy : 0.625
##                  95% CI : (0.5247, 0.718)
##     No Information Rate : 0.5865
##     P-Value [Acc > NIR] : 0.2439
##
##                   Kappa : 0.1281
##  Mcnemar's Test P-Value : 2.99e-07
##
##             Sensitivity : 0.16279
##             Specificity : 0.95082
##          Pos Pred Value : 0.70000
##          Neg Pred Value : 0.61702
##              Prevalence : 0.41346
##          Detection Rate : 0.06731
```

```
##    Detection Prevalence : 0.09615
##       Balanced Accuracy : 0.55681
##
##        'Positive' Class : Down
##
```

Adding a quadratic term Lag2^2 improved the QDA result accuracy from 58.65% to 62.5%.

```
train.x <- cbind(train[,3])
test.x <- cbind(test[,3])
train.direction <- train[,9]
test.direction <- test[,9]

set.seed(1)
knn2 <- knn(train.x, test.x, train.direction, k = 5)
table(knn2, test$Direction)
```

```
##
## knn2   Down Up
##   Down   16 21
##   Up     27 40
```

```
mean(knn2 == test.direction)
```

```
## [1] 0.5384615
```

k = 5 improves the accuarcy of the KNN approach from 50% to 53.84%.

```
set.seed(1)
knn4 <- knn(train.x, test.x, train.direction, k = 10)
table(knn4, test$Direction)
```

```
##
## knn4   Down Up
##   Down   17 21
##   Up     26 40
```

```
mean(knn4 == test.direction)
```

```
## [1] 0.5480769
```

k = 10 has a marginal improvement over k = 5.

```
set.seed(1)
knn5 <- knn(train.x, test.x, train.direction, k = 20)
table(knn5, test$Direction)
```

```
##
## knn5   Down Up
##   Down   21 21
##   Up     22 40
```

```
mean(knn5 == test.direction)
```

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

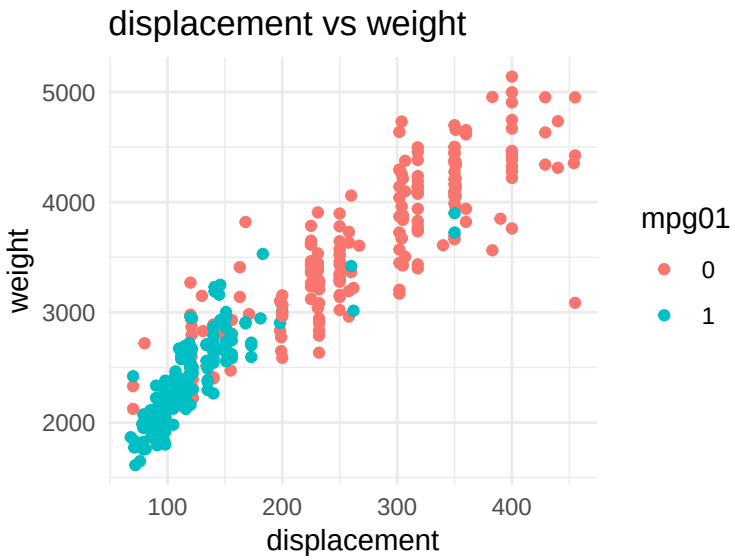
Larger values of k, i.e., around k = 20, seem to produce better results for the KNN approach with only Lag2 as a predictor.

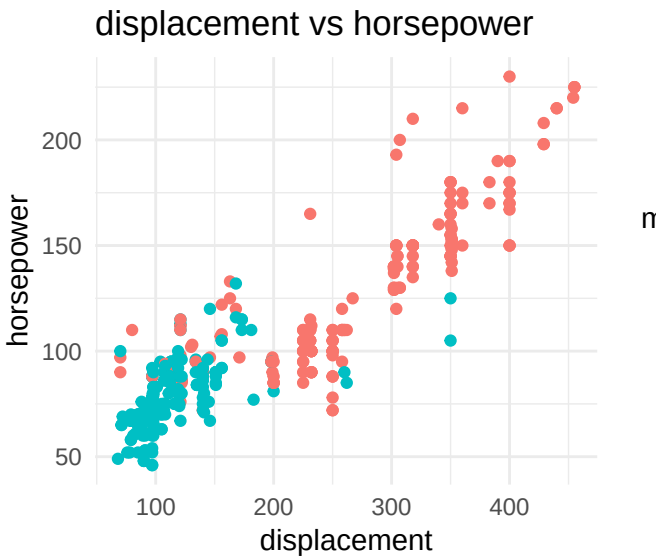
11.In this problem, you will develop a model to predict whether a given car gets high or low gas mileage based on the Auto data set. (a) Create a binary variable, mpg01, that contains a 1 if mpg contains a value above its median, and a 0 if mpg contains a value below its median. You can compute the median using the median() function. Note 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.

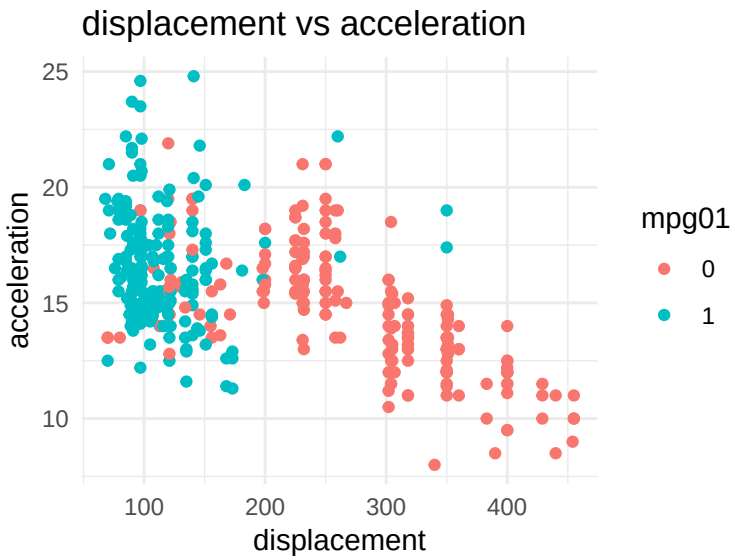
```
data("Auto")
Auto$mpg01 <- c()
for (i in 1:length(Auto$mpg)) {

  if (Auto$mpg[i] > median(Auto$mpg)){
   Auto$mpg01[i] <- 1
}
  else {
    Auto$mpg01[i] <- 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.

```
ggplot(data = Auto) +
  aes(x = displacement, y = weight) +
  geom_point(aes(color = factor(Auto$mpg01))) +
  theme_minimal()+
  labs(color = "mpg01")+
  ggtitle("displacement vs weight")

ggplot(data = Auto) +
  aes(x = displacement, y = horsepower) +
  geom_point(aes(color = factor(Auto$mpg01))) +
  theme_minimal()+
  labs(color = "mpg01")+
  ggtitle("displacement vs horsepower")

ggplot(data = Auto) +
  aes(x = displacement, y = acceleration) +
  geom_point(aes(color = factor(Auto$mpg01))) +
  theme_minimal()+
  labs(color = "mpg01")+
  ggtitle("displacement vs acceleration")
```





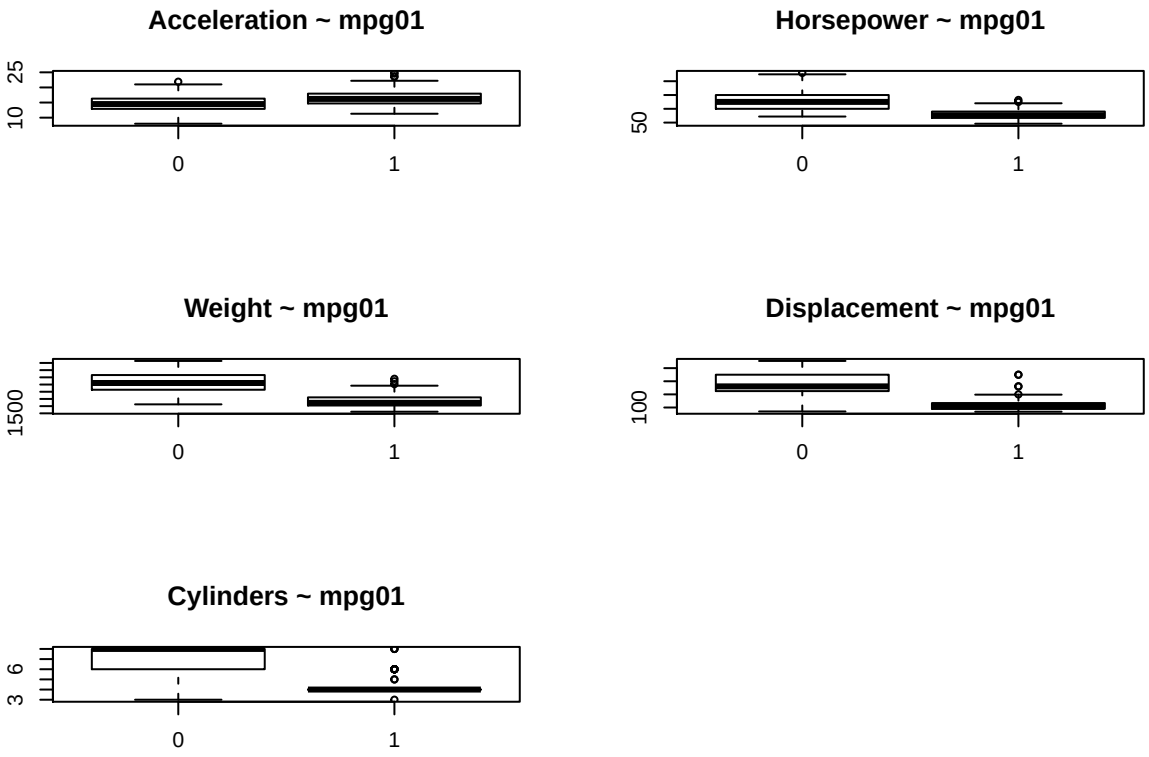


From the displacement vs weight plot, vehicles with mpg above the median mpg tend to have lower displacement and lower weight, and those with mpg lower than the median mpg have higher displacement and higher weight.

From the displacement vs horsepower plot, it is seen that vehicles with higher than median mpg have lower displacement and horsepower.

From the displacement vs acceleration, vehicles with higher than median mpg have higher acceleration and lower displacement. And vehicles with mpg lower than the median mpg have lower acceleration and higher displacement.

```
par(mfrow = c(3,2))
boxplot(acceleration~mpg01, data = Auto, main = "Acceleration ~ mpg01")
boxplot(horsepower~mpg01, data = Auto, main = "Horsepower ~ mpg01")
boxplot(weight~mpg01, data = Auto, main = "Weight ~ mpg01")
boxplot(displacement~mpg01, data = Auto, main = "Displacement ~ mpg01")
boxplot(cylinders~mpg01, data = Auto, main = "Cylinders ~ mpg01")
```



From the above plots, cylinders, displacement, weight, and horsepower vary with respect to mpg01.

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

```
data.train <- Auto[Auto$year < 80,]
data.test <- Auto[Auto$year > 79,]
```

(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?

```
lda3 <- lda(mpg01 ~ cylinders + displacement + weight, data = data.train)
lda3
```

```
## Call:
## lda(mpg01 ~ cylinders + displacement + weight, data = data.train)
##
## Prior probabilities of groups:
##         0         1
## 0.6221498 0.3778502
##
## Group means:
##   cylinders displacement   weight
## 0  6.785340     274.4817 3630.592
## 1  4.146552     111.9095 2275.664
##
## Coefficients of linear discriminants:
```

```
##                       LD1
## cylinders    -0.401637959
## displacement  0.001083413
## weight       -0.001116384
```

```
pred.lda3 <- predict(lda3, newdata = data.test)
confusionMatrix(pred.lda3$class, factor(data.test$mpg01))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0  5 10
##          1  0 70
##
##                Accuracy : 0.8824
##                  95% CI : (0.7943, 0.9421)
##     No Information Rate : 0.9412
##     P-Value [Acc > NIR] : 0.988910
##
##                   Kappa : 0.4516
##  Mcnemar's Test P-Value : 0.004427
##
##             Sensitivity : 1.00000
##             Specificity : 0.87500
##          Pos Pred Value : 0.33333
##          Neg Pred Value : 1.00000
##              Prevalence : 0.05882
##          Detection Rate : 0.05882
##    Detection Prevalence : 0.17647
##       Balanced Accuracy : 0.93750
##
##        'Positive' Class : 0
##
```

```
mean(pred.lda3$class != data.test$mpg01)
```

```
## [1] 0.1176471
```

The test error rate of the model is 11.76%.

(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?

```
qda3 <- qda(mpg01 ~ cylinders + displacement + weight, data = data.train)
qda3
```

```
## Call:
## qda(mpg01 ~ cylinders + displacement + weight, data = data.train)
##
## Prior probabilities of groups:
##         0         1
## 0.6221498 0.3778502
##
## Group means:
##   cylinders displacement   weight
```

```
## 0  6.785340     274.4817 3630.592
## 1  4.146552     111.9095 2275.664
```

```
pred.qda3 <- predict(qda3, newdata = data.test)
confusionMatrix(pred.qda3$class, factor(data.test$mpg01))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0  5 11
##          1  0 69
##
##                Accuracy : 0.8706
##                  95% CI : (0.7802, 0.9336)
##     No Information Rate : 0.9412
##     P-Value [Acc > NIR] : 0.995921
##
##                   Kappa : 0.4246
##  Mcnemar's Test P-Value : 0.002569
##
##             Sensitivity : 1.00000
##             Specificity : 0.86250
##          Pos Pred Value : 0.31250
##          Neg Pred Value : 1.00000
##              Prevalence : 0.05882
##          Detection Rate : 0.05882
##    Detection Prevalence : 0.18824
##       Balanced Accuracy : 0.93125
##
##        'Positive' Class : 0
##
```

```
mean(pred.qda3$class != data.test$mpg01)
```

```
## [1] 0.1294118
```

The test error rate of the QDA model is 12.94%.

(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?

```
fit3 <- glm(mpg01 ~ cylinders + displacement + weight + horsepower, data = data.train, family = binomial
summary(fit3)
```

```
##
## Call:
## glm(formula = mpg01 ~ cylinders + displacement + weight + horsepower,
##     family = binomial, data = data.train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.2251  -0.2522  -0.0146   0.3038   3.6209
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
```

```
## (Intercept)  12.2264405  2.1360013   5.724 1.04e-08 ***
## cylinders     0.0018765  0.4655142   0.004  0.99678
## displacement -0.0116436  0.0111859  -1.041  0.29792
## weight       -0.0022942  0.0008673  -2.645  0.00817 **
## horsepower   -0.0461824  0.0176901  -2.611  0.00904 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 407.08  on 306  degrees of freedom
## Residual deviance: 147.47  on 302  degrees of freedom
## AIC: 157.47
##
## Number of Fisher Scoring iterations: 7
```

```
pred3 <- predict(fit3, newdata = data.test, type = "response")
pred3.glm <- rep(0, length(pred3))
pred3.glm[pred3 > 0.5] <- 1
confusionMatrix(as.factor(pred3.glm), as.factor(data.test$mpg01))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0  5 15
##          1  0 65
##
##                Accuracy : 0.8235
##                  95% CI : (0.7257, 0.8977)
##     No Information Rate : 0.9412
##     P-Value [Acc > NIR] : 0.9999672
##
##                   Kappa : 0.3377
##  Mcnemar's Test P-Value : 0.0003006
##
##             Sensitivity : 1.00000
##             Specificity : 0.81250
##          Pos Pred Value : 0.25000
##          Neg Pred Value : 1.00000
##              Prevalence : 0.05882
##          Detection Rate : 0.05882
##    Detection Prevalence : 0.23529
##       Balanced Accuracy : 0.90625
##
##        'Positive' Class : 0
##
```

Including horsepower as a predictor in logistic regression drastically improves the accuracy of the model. The test error rate of this model is 17.65%.

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

```
train.auto <- cbind(data.train[,c(2,3,4,5)])
test.auto <- cbind(data.test[,c(2,3,4,5)])
train.mpg <- data.train[,10]
test.mpg <- data.test[,10]

set.seed(1)
knn6 <- knn(train.auto, test.auto, train.mpg, k = 5)
table(knn6, data.test$mpg01)
```

```
##
## knn6  0  1
##    0  5 19
##    1  0 61
```

```
mean(knn6 != test.mpg)
```

```
## [1] 0.2235294
```

k = 5 yields a test error rate of 22.35%.

```
set.seed(1)
knn7 <- knn(train.auto, test.auto, train.mpg, k = 10)
table(knn7, data.test$mpg01)
```

```
##
## knn7  0  1
##    0  5 19
##    1  0 61
```

```
mean(knn7 != test.mpg)
```

```
## [1] 0.2235294
```

Surprisingly, k = 10 does not show any improvement over k = 5.

```
set.seed(1)
knn8 <- knn(train.auto, test.auto, train.mpg, k = 20)
table(knn8, data.test$mpg01)
```

```
##
## knn8  0  1
##    0  5 16
##    1  0 64
```

```
mean(knn8 != test.mpg)
```

```
## [1] 0.1882353
```

k = 20 yields a better result with the test error rate at 18.8%.

12. 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.

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

```
}
Power()
```

```
## [1] 8
```

(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: This should output the value of 3^8, namely, 6,561.

```
Power2 <- function (x,a){
  x^a
}
Power2 (3,8)
```

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

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

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

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

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

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

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

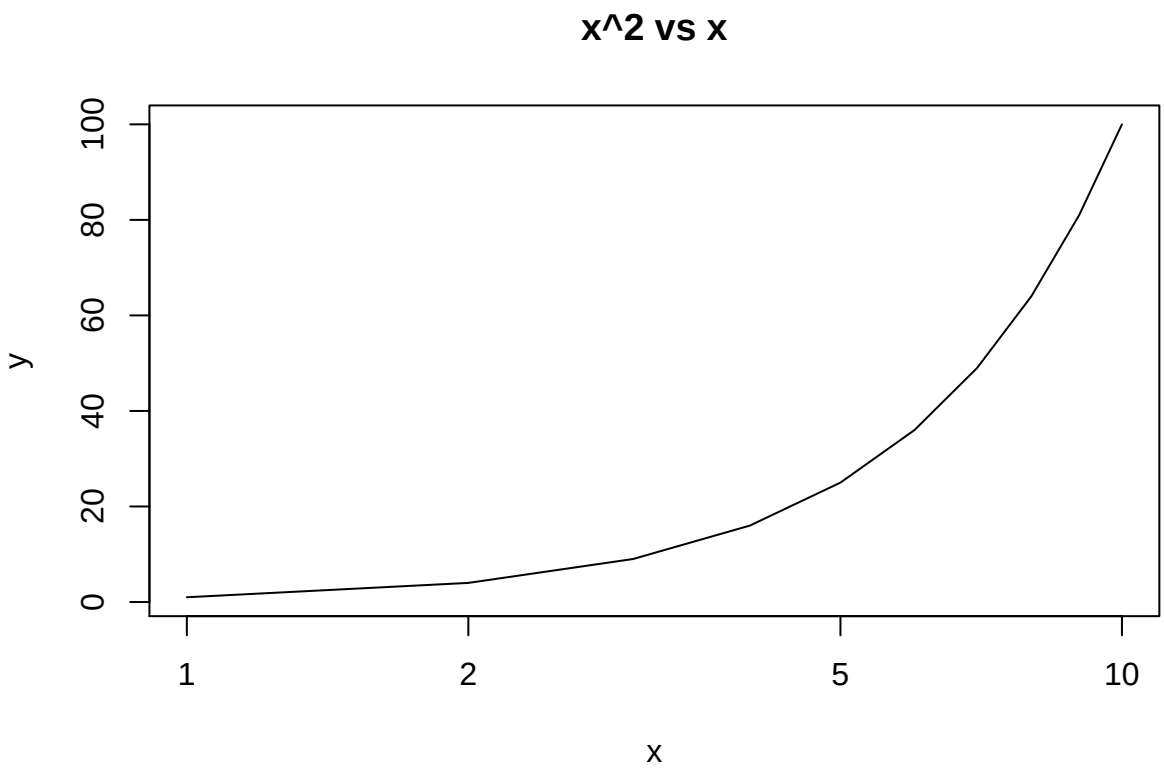
```
## [1] 2248091
```

(d) Now create a new function, Power3(), that actually returns the result x^a as an R object, rather than simply then you can simply return() this return() result, using the following line:

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

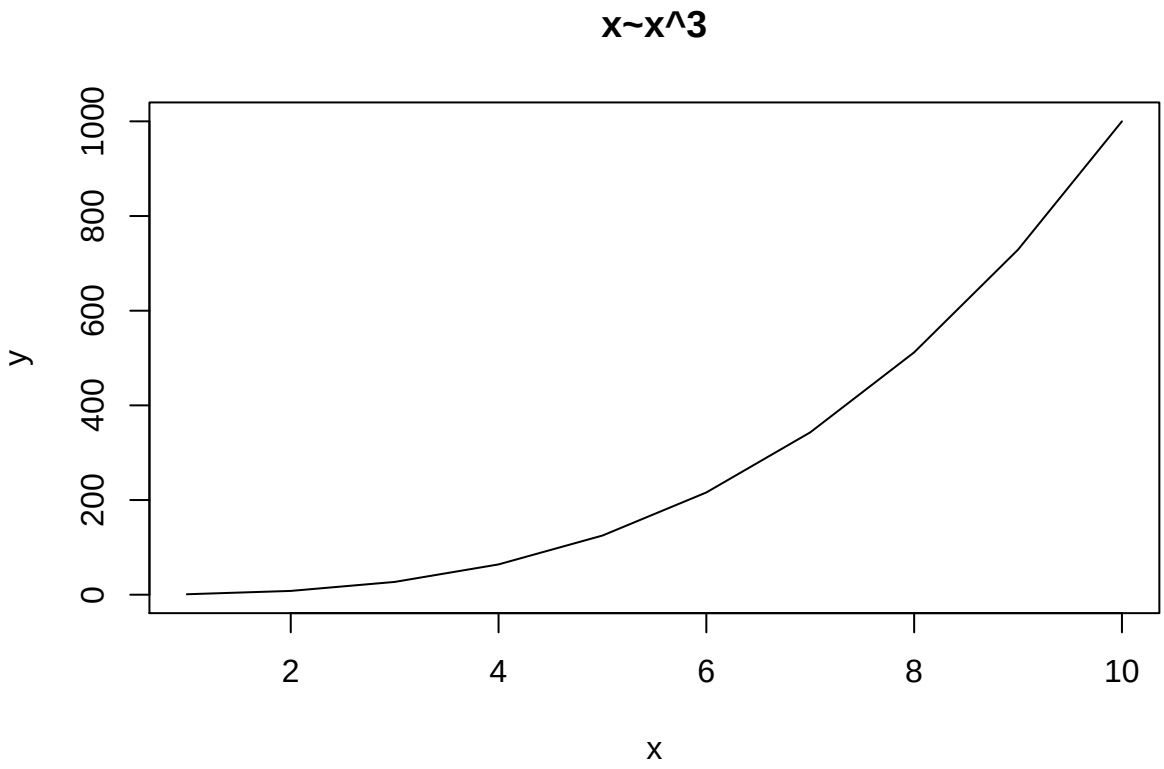
(e) Now using the Power3() function, create a plot of $f(x) = x^2$. The x-axis should display a range of integers from 1 to 10, and the y-axis should display x^2. Label the axes appropriately, and use an appropriate title for the figure. Consider displaying either the x-axis, the y-axis, or both on the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

```
x <- 1:10
y <- sapply(x, function(x)Power3(x,2))
plot(x, y, log = "x", type = "l", main = "x^2 vs x")
```



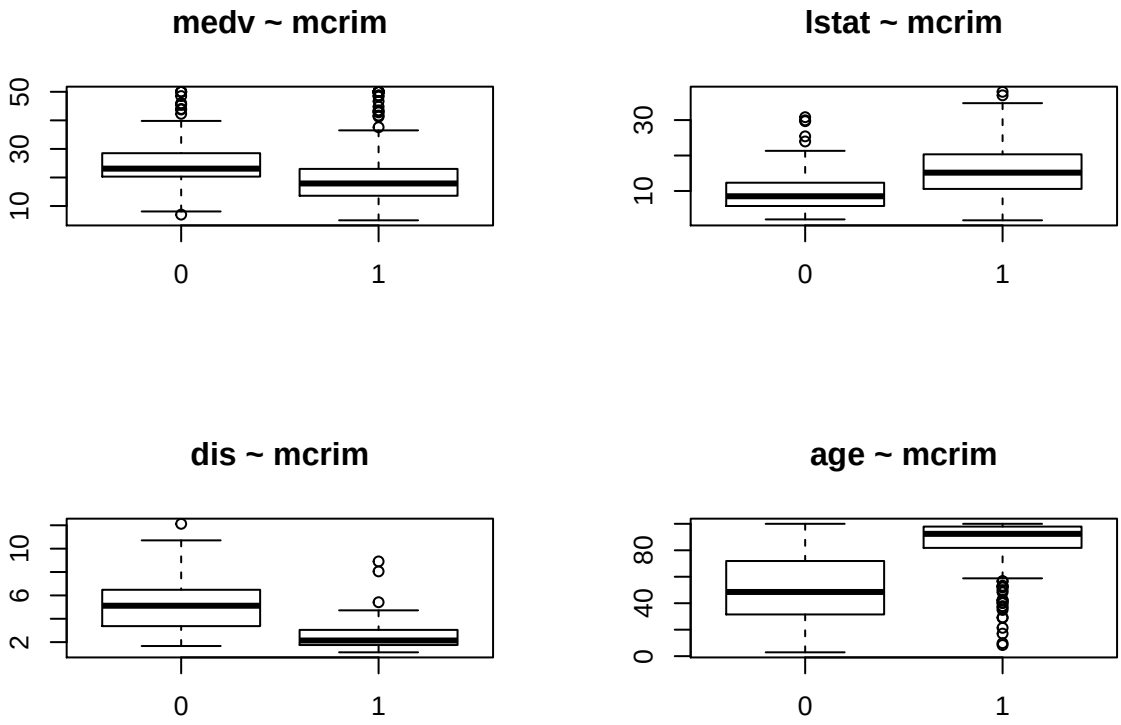
(f) Create a function, PlotPower(), that allows you to create a plot of x against xˆa for a fixed a and for a range of values of x. For instance, if you call PlotPower (1:10 ,3) then a plot should be created with an x-axis taking on values 1, 2, . . . , 10, and a y-axis taking on values 13, 23, . . . , 103.

```
PlotPower <- function(x,a){
  y <- sapply(x, function(x){x^a})
  plot(x,y, type = "l", main = paste('x~x^',a,sep = ''))
}
PlotPower(1:10,3)
```



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

```
data("Boston")
mcrim <- ifelse(Boston$crim > median(Boston$crim), 1, 0)
df <- data.frame(Boston, mcrim)

par(mfrow = c(2,2))
boxplot(medv~mcrim, df, main='medv ~ mcrim')
boxplot(lstat~mcrim, df, main='lstat ~ mcrim')
boxplot(dis~mcrim, df, main='dis ~ mcrim')
boxplot(age~mcrim, df, main='age ~ mcrim')
```




```
train.df <- df[1:356,]
test.df <- df[357:506,]

fit4 <- glm(mcrim ~ dis + age + lstat, data = train.df, family = binomial)
summary(fit4)
```

```
##
## Call:
## glm(formula = mcrim ~ dis + age + lstat, family = binomial, data = train.df)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.8582  -0.6441  -0.2880   0.7199   3.2238
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept) -0.839291   0.869737  -0.965 0.334548
## dis         -0.520790   0.119393  -4.362 1.29e-05 ***
## age          0.031721   0.008541   3.714 0.000204 ***
## lstat        0.005805   0.027071   0.214 0.830206
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 452.27  on 355  degrees of freedom
```

```
## Residual deviance: 309.90  on 352  degrees of freedom
## AIC: 317.9
##
## Number of Fisher Scoring iterations: 5
```

```
pred4 <- predict(fit4, newdata = test.df, type = "response")
pred4.glm <- rep(0, length(pred4))
pred4.glm[pred4 > 0.5] <- 1
confusionMatrix(as.factor(pred4.glm), as.factor(test.df$mcrim))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0   2  13
##          1  13 122
##
##                Accuracy : 0.8267
##                  95% CI : (0.7564, 0.8835)
##     No Information Rate : 0.9
##     P-Value [Acc > NIR] : 0.9981
##
##                   Kappa : 0.037
##  Mcnemar's Test P-Value : 1.0000
##
##             Sensitivity : 0.13333
##             Specificity : 0.90370
##          Pos Pred Value : 0.13333
##          Neg Pred Value : 0.90370
##              Prevalence : 0.10000
##          Detection Rate : 0.01333
##    Detection Prevalence : 0.10000
##       Balanced Accuracy : 0.51852
##
##        'Positive' Class : 0
##
```

Logistic Regression gives an accuracy of 82.67%. Adding medv as a predictor does not change the accuracy. The test error rate is 17.33%.

```
lda4 <- lda(mcrim ~ dis + age + lstat + medv, data = train.df)
lda4
```

```
## Call:
## lda(mcrim ~ dis + age + lstat + medv, data = train.df)
##
## Prior probabilities of groups:
##         0         1
## 0.6685393 0.3314607
##
## Group means:
##        dis      age     lstat     medv
## 0 5.270933 49.58824  9.114664 25.40756
## 1 2.982705 82.00847 12.840763 24.25085
##
## Coefficients of linear discriminants:
```

```
##               LD1
## dis   -0.26498025
## age    0.02517994
## lstat  0.02434441
## medv   0.01804745
```

```
pred4.lda <- predict(lda4, newdata = test.df)
confusionMatrix(pred4.lda$class, as.factor(test.df$mcrim))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0   4  13
##          1  11 122
##
##                Accuracy : 0.84
##                  95% CI : (0.7714, 0.8947)
##     No Information Rate : 0.9
##     P-Value [Acc > NIR] : 0.9924
##
##                   Kappa : 0.1608
##  Mcnemar's Test P-Value : 0.8383
##
##             Sensitivity : 0.26667
##             Specificity : 0.90370
##          Pos Pred Value : 0.23529
##          Neg Pred Value : 0.91729
##              Prevalence : 0.10000
##          Detection Rate : 0.02667
##    Detection Prevalence : 0.11333
##       Balanced Accuracy : 0.58519
##
##        'Positive' Class : 0
##
```

Adding medv as a predictor in LDA improves the model slightly, giving an accuracy of 84%. The test error rate is 16%.

```
qda4 <- qda(mcrim ~ dis + age + lstat, data = train.df)
qda4
```

```
## Call:
## qda(mcrim ~ dis + age + lstat, data = train.df)
##
## Prior probabilities of groups:
##         0         1
## 0.6685393 0.3314607
##
## Group means:
##        dis      age     lstat
## 0 5.270933 49.58824  9.114664
## 1 2.982705 82.00847 12.840763
```

```
pred4.qda <- predict(qda4, newdata = test.df)
confusionMatrix(pred4.qda$class, as.factor(test.df$mcrim))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   0   1
##          0   3  12
##          1  12 123
##
##                Accuracy : 0.84
##                  95% CI : (0.7714, 0.8947)
##     No Information Rate : 0.9
##     P-Value [Acc > NIR] : 0.9924
##
##                   Kappa : 0.1111
##  Mcnemar's Test P-Value : 1.0000
##
##             Sensitivity : 0.2000
##             Specificity : 0.9111
##          Pos Pred Value : 0.2000
##          Neg Pred Value : 0.9111
##              Prevalence : 0.1000
##          Detection Rate : 0.0200
##    Detection Prevalence : 0.1000
##       Balanced Accuracy : 0.5556
##
##        'Positive' Class : 0
##
```

Including medv as a predictor in QDA gives an accuracy of around 82% whereas excluding it gives an accuracy of 84%. The test error rate for this model is 16%.

```
train.boston <- cbind(train.df[,c(7,8,13,14)])
test.boston <- cbind(test.df[,c(7,8,13,14)])
train.mcrim <- train.df[,15]
test.mcrim <- test.df[,15]

set.seed(1)
knn9 <- knn(train.boston, test.boston, train.mcrim, k = 5)
table(knn9, test.df$mcrim)
```

```
##
## knn9   0   1
##    0   9  31
##    1   6 104
```

```
mean(knn9 != test.mcrim)
```

```
## [1] 0.2466667
```

KNN with k = 5 yields a test error rate of 24.67%.

```
set.seed(1)
knn10 <- knn(train.boston, test.boston, train.mcrim, k = 10)
table(knn10, test.df$mcrim)
```

```
##
## knn10  0  1
##     0  9 36
```

```
##     1  6 99
```

```
mean(knn10 != test.mcrim)
```

```
## [1] 0.28
```

k = 10 yields a test error rate of 28%, higher than that for k = 5.

```
set.seed(1)
knn11 <- knn(train.boston, test.boston, train.mcrim, k = 20)
table(knn11, test.df$mcrim)
```

```
##
## knn11  0  1
##     0  9 39
##     1  6 96
```

```
mean(knn11 != test.mcrim)
```

```
## [1] 0.3
```

k = 20 yields an even higher test error rate of 30%.

```
set.seed(1)
knn12 <- knn(train.boston, test.boston, train.mcrim, k = 2)
table(knn12, test.df$mcrim)
```

```
##
## knn12  0  1
##     0  8 38
##     1  7 97
```

```
mean(knn12 != test.mcrim)
```

```
## [1] 0.3
```

k = 2 also yields a test error rate of 30%. Therefore, the best result could be for k = 5.