Classification Homework

Tao He

1/30/2022

4.6.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.

sol.n: The logistic function is

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

Then, let X1 = 40, X2 = 3.5,

$$P(X) = \frac{\exp(-6 + 0.05 * X_1 + X_2)}{1 + \exp(-0.5)} = \frac{\exp(-6 + 0.05 * 40 + 3.5)}{1 + \exp(-6 + 0.05 * 40 + 3.5)} = \frac{\exp(-0.5)}{1 + \exp(-0.5)} = 0.38$$

(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?

sol.n:

The log odds is

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

; Then, we use the data from the former question,

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

;

$$X_1 = 50$$

;

Therefore, the student in part (a) need to study 50 hours to have a 50 % chance of getting an A in the class.

4.8. 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?

sol.n: We should prefer to use for classification of new observations by logistic regression method.

Since when k = 1, we have a training error rate of 0%. However, the average error rate is 18%, which implies the test error rate is 36%. The test error rate of KNN is greater than the test error rate of logistic regression in case of 36% >30%. Therefore, it is better to choose the logistic regression methods because of the lower test error rate.

4.9. 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?

sol.n: The odds ratio is

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

;

Then,

$$P(X) = \frac{0.37}{1+0.37} = 0.27$$

;

Therefore, On average, we have a fraction of 27% people with an odds of 0.37 of defaulting on their credit card payment.

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

sol.n: Since we have already known P(X) = 0.16;

The odds ratio is

$$\frac{P(X)}{1-P(X)} = \frac{0.16}{1-0.16} = 0.19$$

;

Therefore, she will default the the odds 0.19.

4.13. 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.

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

```
attach(Weekly)

# look at the feature of data set
summary(Weekly)
```

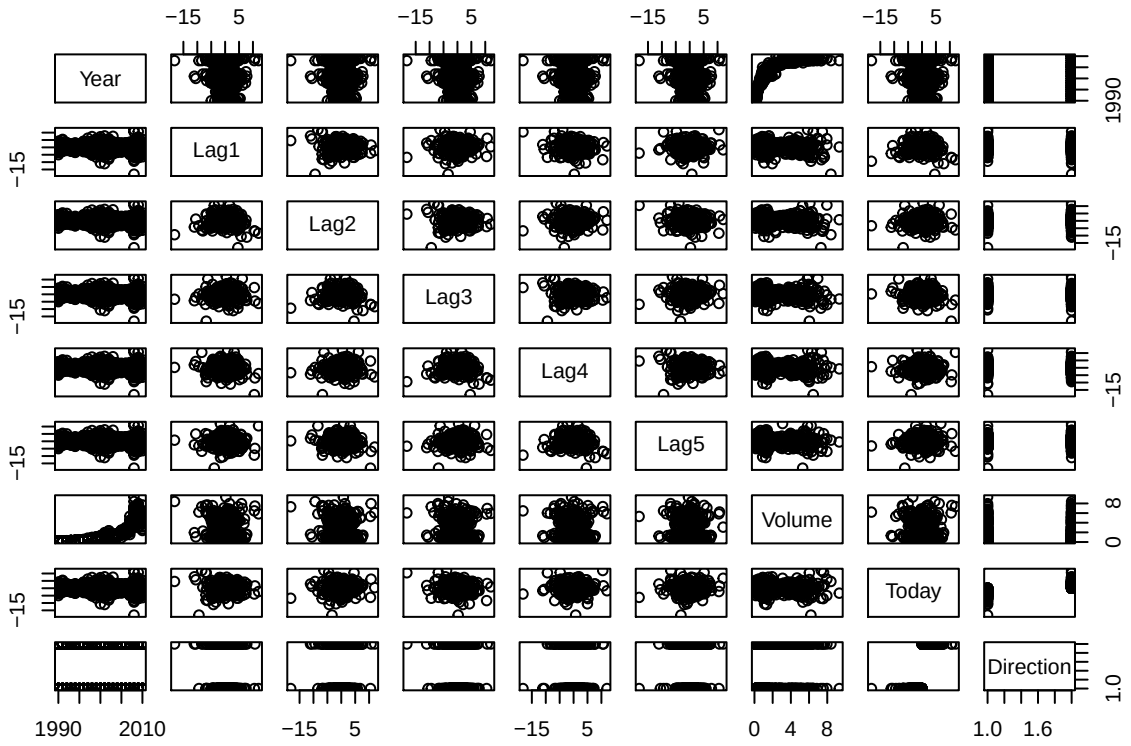
```
##       Year           Lag1               Lag2               Lag3
##  Min.   :1990   Min.   :-18.1950   Min.   :-18.1950   Min.   :-18.1950
##  1st Qu.:1995   1st Qu.: -1.1540   1st Qu.: -1.1540   1st Qu.: -1.1580
##  Median :2000   Median :  0.2410   Median :  0.2410   Median :  0.2410
##  Mean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

```
# look at the correlation of each variables
cor(Weekly[,-9])
```

```
##               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
```

```
pairs(Weekly)
```



Almost all the correlation is below 0.1.

Additionally, the correlation of "year" and "Volume" is higher than any other correlation values, even higher than 0.7, which illustrates that there exists high relation between "year" and "Volume".

(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 <- glm(Direction ~ .-Today - Year, data = Weekly, family = binomial)
summary(fit.glm)
```

```
##
## Call:
## glm(formula = Direction ~ . - Today - Year, 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
```

When we look at the estimate coefficients, only "Lag2" is significant with p-value which is below 0.05.

(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.

```
probs <- predict(fit.glm, type = "response")

# set down and up
pred.glm <- rep("Down", length(probs))
pred.glm[probs > 0.5] <- "Up"
table(pred.glm, Direction)
```

```
##         Direction
## pred.glm Down  Up
##     Down   54  48
##     Up    430 557
```

The overall fraction of correct predictions is $(54 + 557)/1089 = 0.5611$.

(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).

```
# from 1990 to 2008
train <- (Year < 2009)

Weekly.test <- Weekly[!train, ]
Direction.test <- Direction[!train]

fit.glm2 <- glm(Direction ~ Lag2, data = Weekly, family = binomial, subset = train)
summary(fit.glm2)
```

```
##
## Call:
## glm(formula = Direction ~ Lag2, family = binomial, data = Weekly,
##     subset = 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
```

```
probs2 <- predict(fit.glm2, Weekly.test, type = "response")

# set down and up
pred.glm2 <- rep("Down", length(probs2))
pred.glm2[probs2 > 0.5] <- "Up"
table(pred.glm2, Direction.test)
```

```
##          Direction.test
## pred.glm2 Down Up
##      Down    9  5
##      Up     34 56
```

The overall fraction of correct predictions is $(9 + 56)/(9 + 5 + 34 + 56) = 0.625$.

(e) Repeat (d) using LDA.

```
fit.lda <- lda(Direction ~ Lag2, data = Weekly, subset = train)
fit.lda
```

```
## Call:
## lda(Direction ~ Lag2, data = Weekly, subset = 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(fit.lda, Weekly.test)
table(pred.lda$class, Direction.test)
```

```
##       Direction.test
##        Down Up
##   Down    9  5
##   Up     34 56
```

The overall fraction of correct predictions is $(9 + 56)/(9 + 5 + 34 + 56) = 0.625$.

(f) Repeat (d) using QDA.

```
fit.qda <- qda(Direction ~ Lag2, data = Weekly, subset = train)
fit.qda
```

```
## Call:
```

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

```
pred.qda <- predict(fit.qda, Weekly.test)
table(pred.qda$class, Direction.test)
```

```
##       Direction.test
##        Down Up
##   Down    0  0
##   Up     43 61
```

The overall fraction of correct predictions is $(0 + 61)/(0 + 0 + 43 + 61) = 0.5865$.

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

```
train.X <- as.matrix(Lag2[train])
test.X <- as.matrix(Lag2[!train])
train.Direction <- Direction[train]

set.seed(679)
pred.knn <- knn(train.X, test.X, train.Direction, k = 1)
table(pred.knn, Direction.test)
```

```
##         Direction.test
## pred.knn Down Up
##     Down   21 30
##     Up     22 31
```

The overall fraction of correct predictions is $(21 + 30)/(21 + 30 + 22 + 31) = 0.4904$.

(h) Repeat (d) using naive Bayes.

```
fit.nb <- naiveBayes(Direction ~ Lag2, data = Weekly, subset = train)
fit.nb
```

```
##
## Naive Bayes Classifier for Discrete Predictors
##
## Call:
## naiveBayes.default(x = X, y = Y, laplace = laplace)
##
## A-priori probabilities:
## Y
##      Down        Up
## 0.4477157 0.5522843
##
## Conditional probabilities:
##       Lag2
## Y             [,1]     [,2]
##   Down -0.03568254 2.199504
```

```
##   Up    0.26036581 2.317485
```

```
pred.nb <- predict(fit.nb, Weekly.test)
table(pred.nb, Direction.test)
```

```
##        Direction.test
## pred.nb Down Up
##    Down    0  0
##    Up     43 61
```

The overall fraction of correct predictions is $(0 + 61)/(0 + 0 + 43 + 61) = 0.5865$.

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

sol.n: If we compare the test error rates, we see that logistic regression and LDA have the minimum error rates, followed by QDA, KNN and naive Bayes.

(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 with Lag2 + Lag1
fit.glm3 <- glm(Direction ~ Lag2 + Lag1, data = Weekly, family = binomial, subset = train)
probs3 <- predict(fit.glm3, Weekly.test, type = "response")
pred.glm3 <- rep("Down", length(probs3))
pred.glm3[probs3 > 0.5] = "Up"
table(pred.glm3, Direction.test)
```

```
##          Direction.test
## pred.glm3 Down Up
##      Down    7  8
##      Up     36 53
```

```
mean(pred.glm3 == Direction.test)
```

```
## [1] 0.5769231
```

```
# LDA with Lag2 + Lag1
fit.lda2 <- lda(Direction ~ Lag2 + Lag1, data = Weekly, subset = train)
pred.lda2 <- predict(fit.lda2, Weekly.test)
table(pred.lda2$class, Direction.test)
```

```
##       Direction.test
##        Down Up
##   Down    7  8
##   Up     36 53
```

```
mean(pred.lda2$class == Direction.test)
```

```
## [1] 0.5769231
```

```
# QDA with Lag2 + Lag1
fit.qda2 <- qda(Direction ~ Lag2 + Lag1, data = Weekly, subset = train)
pred.qda2 <- predict(fit.qda2, Weekly.test)
table(pred.qda2$class, Direction.test)
```

```
##       Direction.test
##        Down Up
##   Down    7 10
```

```
##   Up     36 51
```

```
mean(pred.qda2$class == Direction.test)
```

```
## [1] 0.5576923
```

```
# naive Bayes with Lag2 + Lag1
fit.nb2 <- naiveBayes(Direction ~ Lag2 + Lag1, data = Weekly, subset = train)
pred.nb2 <- predict(fit.nb2, Weekly.test)
table(pred.nb2, Direction.test)
```

```
##         Direction.test
## pred.nb2 Down Up
##     Down    3  8
##     Up     40 53
```

```
mean(pred.nb2 == Direction.test)
```

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

```
# KNN k =10
pred.knn2 <- knn(train.X, test.X, train.Direction, k = 10)
table(pred.knn2, Direction.test)
```

```
##          Direction.test
## pred.knn2 Down Up
##      Down   20 19
##      Up     23 42
```

```
mean(pred.knn2 == Direction.test)
```

```
## [1] 0.5961538
```

```
# KNN k =100
pred.knn3 <- knn(train.X, test.X, train.Direction, k = 100)
table(pred.knn3, Direction.test)
```

```
##          Direction.test
## pred.knn3 Down Up
##      Down    9 12
##      Up     34 49
```

```
mean(pred.knn3 == Direction.test)
```

```
## [1] 0.5576923
```

According to all the results, the KNN with k = 10 have the best performance in terms of test correct rates.

4.14. 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.

(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.

```
# create a binary variable "mpg01"
attach(Auto)
Auto$mpg01 = with(ifelse(mpg>median(mpg), 1, 0), data=Auto)
```

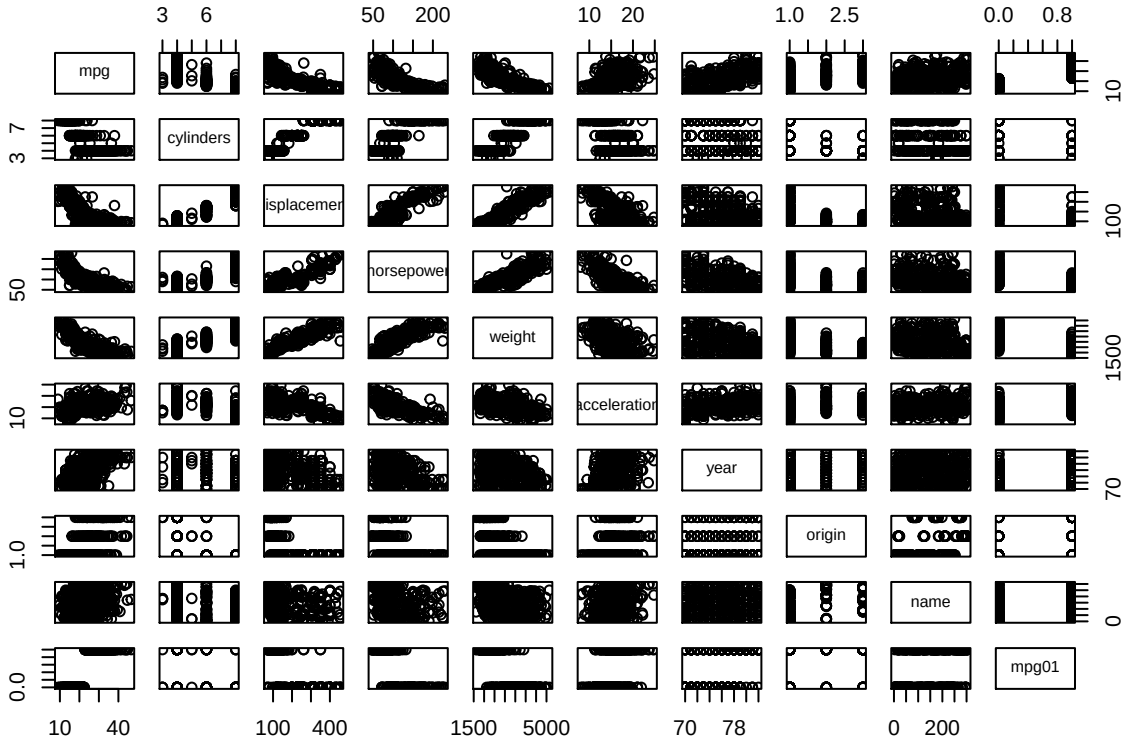
(b) Explore the data graphically in order to investigate the association between mpg01 and the other

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.

```
# correlation
cor(Auto[, -9])
```

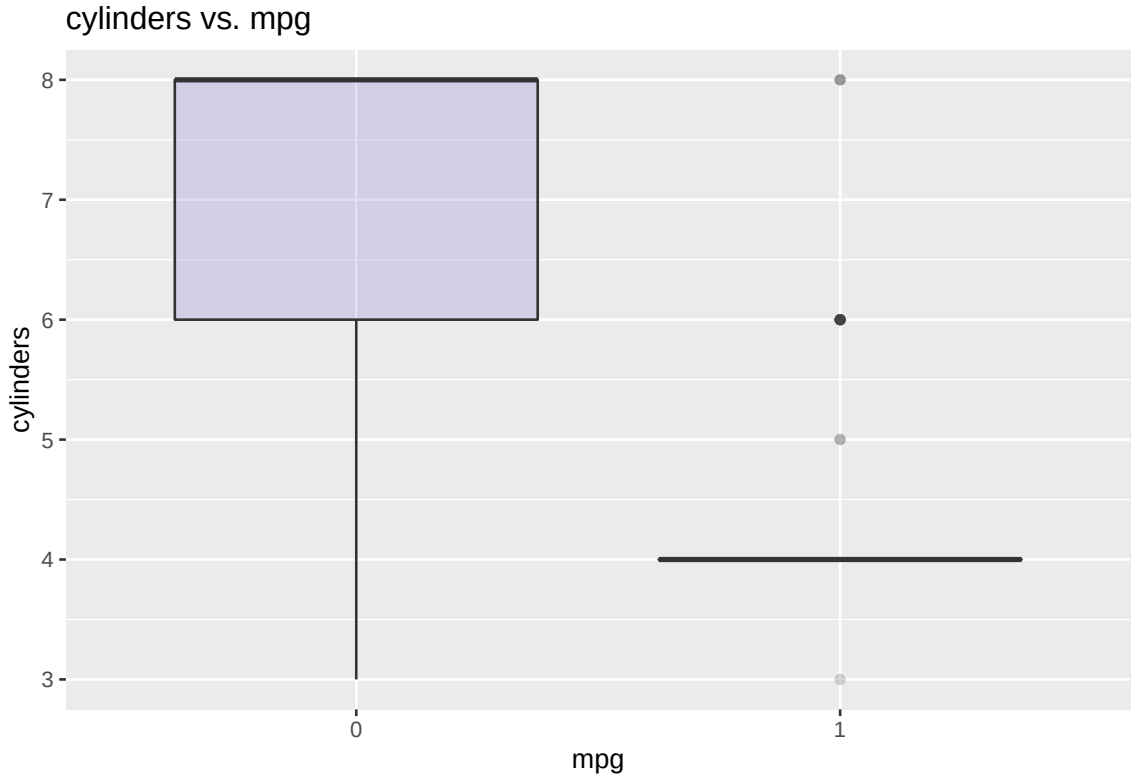
```
##                     mpg  cylinders displacement horsepower     weight
## mpg           1.0000000 -0.7776175   -0.8051269 -0.7784268 -0.8322442
## cylinders    -0.7776175  1.0000000    0.9508233  0.8429834  0.8975273
## displacement -0.8051269  0.9508233    1.0000000  0.8972570  0.9329944
## horsepower   -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
```

```
# correlation plot (scatter plot)
pairs(Auto)
```

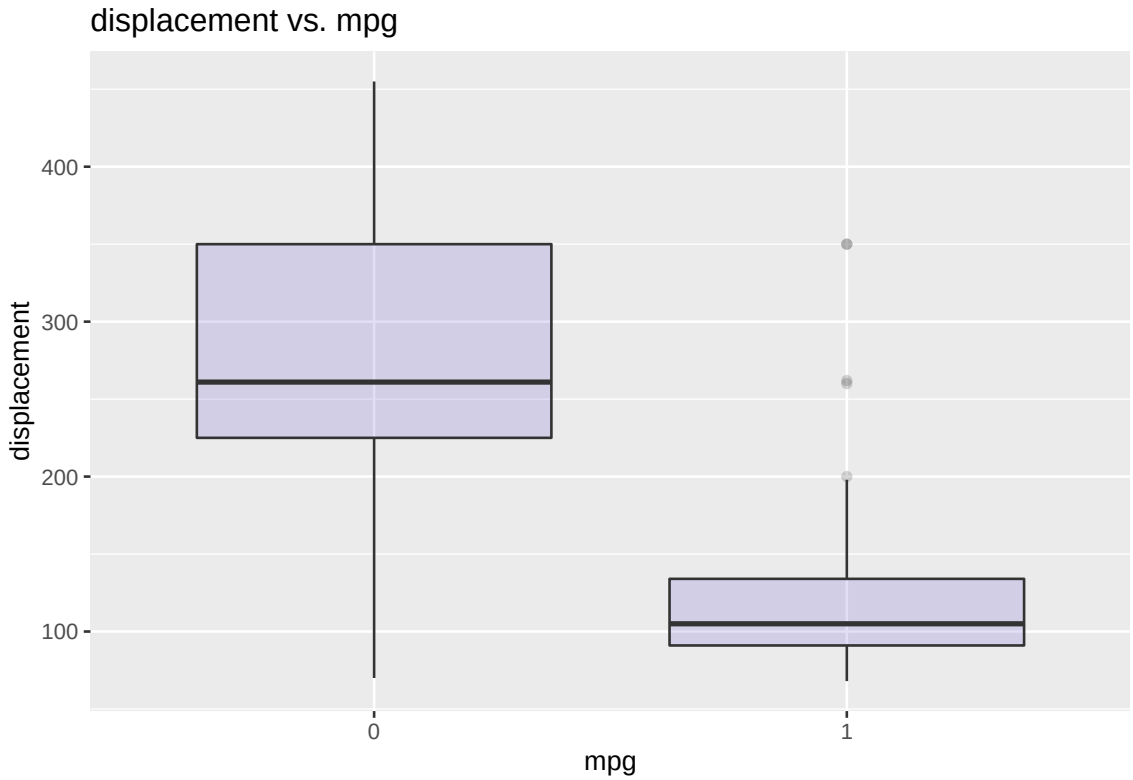



```
# boxplots
# cylinders
```

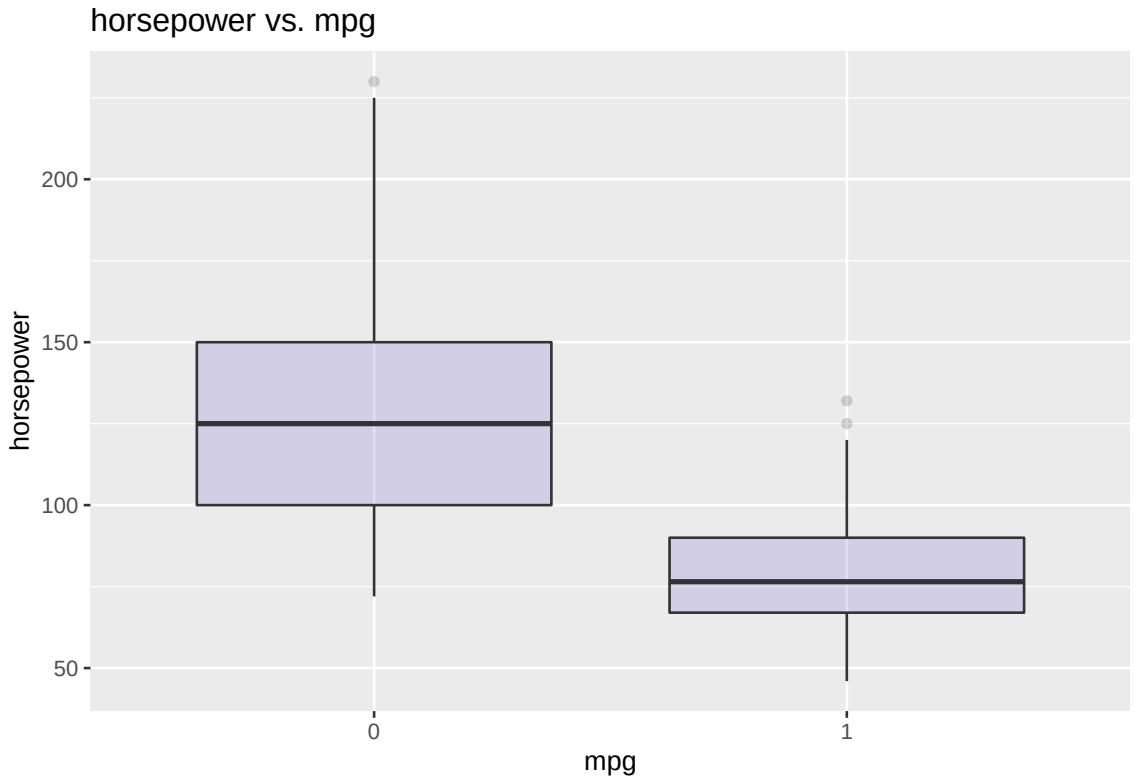
```
ggplot(Auto, aes(x=as.factor(mpg01), y=cylinders)) +
  geom_boxplot(fill="slateblue", alpha=0.2) +
  xlab("mpg") +
  ggtitle("cylinders vs. mpg")
```



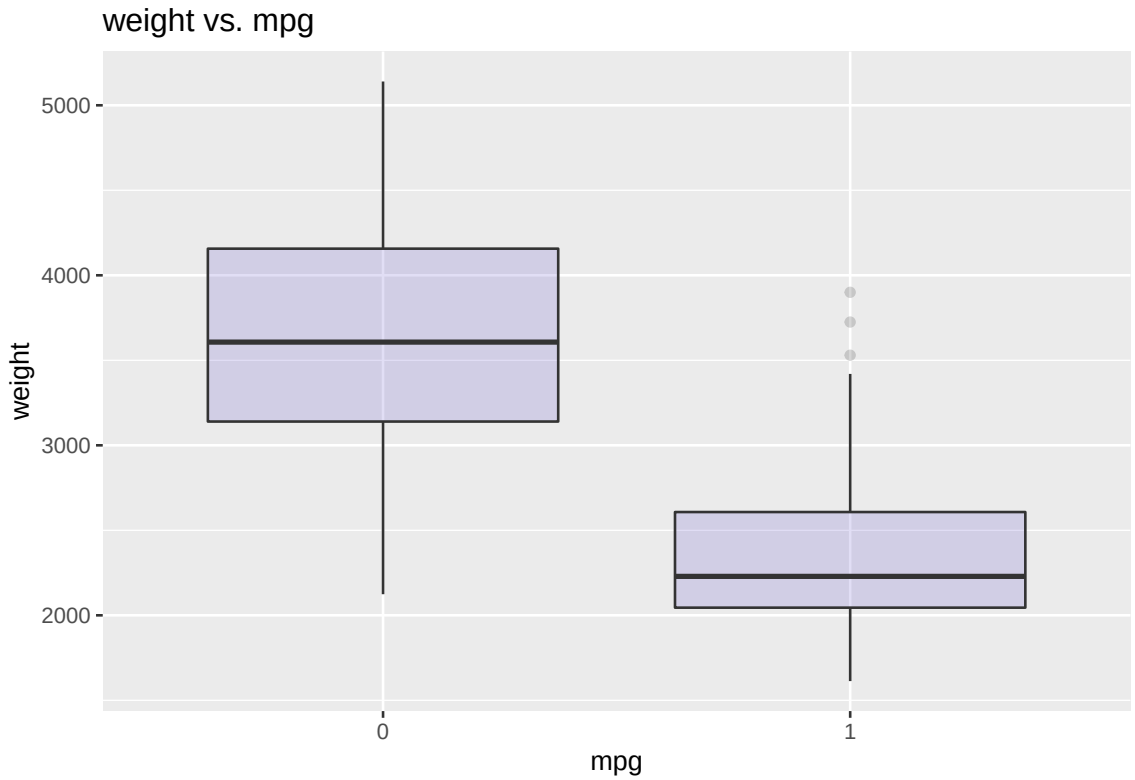

```
# displacement
ggplot(Auto, aes(x=as.factor(mpg01), y=displacement)) +
  geom_boxplot(fill="slateblue", alpha=0.2) +
  xlab("mpg") +
  ggtitle("displacement vs. mpg")
```



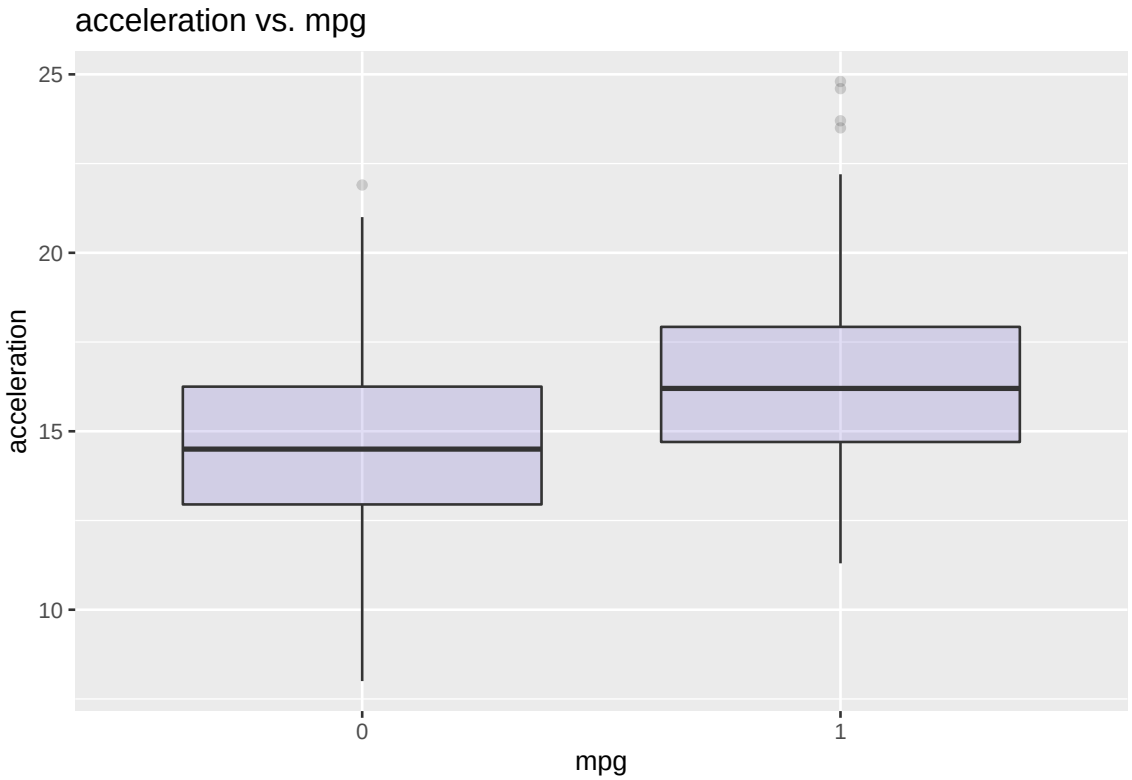

```
# horsepower
ggplot(Auto, aes(x=as.factor(mpg01), y=horsepower)) +
  geom_boxplot(fill="slateblue", alpha=0.2) +
  xlab("mpg") +
  ggtitle("horsepower vs. mpg")
```



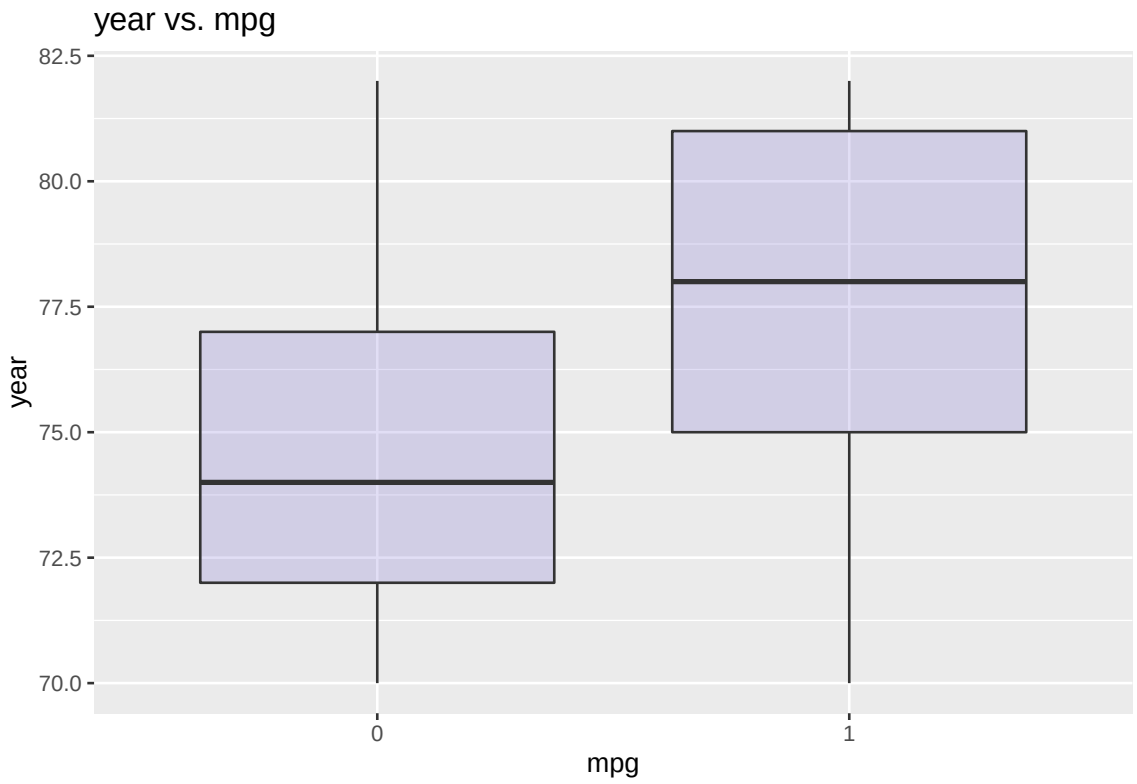

```
# weight
ggplot(Auto, aes(x=as.factor(mpg01), y=weight)) +
  geom_boxplot(fill="slateblue", alpha=0.2) +
  xlab("mpg") +
  ggtitle("weight vs. mpg")
```




```
#acceleration
ggplot(Auto, aes(x=as.factor(mpg01), y=acceleration)) +
  geom_boxplot(fill="slateblue", alpha=0.2) +
  xlab("mpg") +
  ggtitle("acceleration vs. mpg")
```




```
#year
ggplot(Auto, aes(x=as.factor(mpg01), y=year)) +
  geom_boxplot(fill="slateblue", alpha=0.2) +
  xlab("mpg") +
  ggtitle("year vs. mpg")
```



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

```
# Split the data randomly
# Training and Test data
set.seed(679)
sample_data = sample.split(Auto$mpg, SplitRatio = 0.70)
Auto.train = subset(Auto, sample_data==TRUE)
Auto.test = subset(Auto, sample_data==FALSE)
```

(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 ~ cylinders + weight + displacement + horsepower, data = Auto, subset = train)
auto.lda
```

```
## Call:
## lda(mpg01 ~ cylinders + weight + displacement + horsepower, data = Auto,
##     subset = train)
##
## Prior probabilities of groups:
##   0   1
## 0.5 0.5
##
## Group means:
##   cylinders   weight displacement horsepower
## 0  6.765306 3620.403     273.1582  130.11224
## 1  4.178571 2334.765     115.6658   78.82653
##
## Coefficients of linear discriminants:
##                         LD1
## cylinders    -0.4708126926
```

```
## weight       -0.0009846242
## displacement -0.0014339977
## horsepower    0.0044241441
```

```
pred.lda3 <- predict(auto.lda, Auto.test)
table(pred.lda3$class, Auto.test$mpg01)
```

```
##
##      0  1
##   0 46  2
##   1 10 46
```

```
1 - mean(pred.lda3$class == Auto.test$mpg01)
```

```
## [1] 0.1153846
```

The test error of the model obtained by using LAD is 11.54%.

(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 ~ cylinders + weight + displacement + horsepower, data = Auto, subset = train)
auto.qda
```

```
## Call:
## qda(mpg01 ~ cylinders + weight + displacement + horsepower, data = Auto,
##     subset = train)
##
## Prior probabilities of groups:
##   0   1
## 0.5 0.5
##
## Group means:
##   cylinders   weight displacement horsepower
## 0  6.765306 3620.403     273.1582  130.11224
## 1  4.178571 2334.765     115.6658   78.82653
```

```
pred.qda3 <- predict(auto.qda, Auto.test)
table(pred.qda3$class, Auto.test$mpg01)
```

```
##
##      0  1
##   0 47  2
##   1  9 46
```

```
1 - mean(pred.qda3$class == Auto.test$mpg01)
```

```
## [1] 0.1057692
```

The test error of the model obtained by using QDA is 10.58%.

(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.glm <- glm(mpg01 ~ cylinders + weight + displacement + horsepower, data = Auto.train, family = bin
auto.probs <- predict(auto.glm, Auto.test, type = "response")
pred <- rep("0", length(auto.probs))
pred[auto.probs > 0.5] = "1"
table(pred, Auto.test$mpg01)
```

```
##
## pred  0  1
##    0 48  2
##    1  8 46
```

```
1 - mean(pred == Auto.test$mpg01)
```

```
## [1] 0.09615385
```

The test error of the model obtained by using the logistic regression is 9.615%.

(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 ~ cylinders + weight + displacement + horsepower, data = Auto.train)
pred.nb3 <- predict(auto.nb, Auto.test)
table(pred.nb3, Auto.test$mpg01)
```

```
##
## pred.nb3  0  1
##        0 47  2
##        1  9 46
```

```
1 - mean(pred.nb3 == Auto.test$mpg01)
```

```
## [1] 0.1057692
```

The test error of the model obtained by using naive Bayes is 10.58%.

(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, Test and response matrices.
train.matrix = data.matrix(Auto.train[,c("cylinders","displacement","weight","horsepower")])
test.matrix = data.matrix(Auto.test[,c("cylinders","displacement","weight","horsepower")])
train.y = data.matrix(Auto.train$mpg01)
test.y = data.matrix(Auto.test$mpg01)

# K=1 and predictions
auto.knn1 = knn(train.matrix, test.matrix, train.y, k=1)
table(auto.knn1, test.y)
```

```
##          test.y
## auto.knn1  0  1
##         0 47  8
##         1  9 40
```

```
1 - mean(auto.knn1 == Auto.test$mpg01)
```

```
## [1] 0.1634615
```

The test error of the model obtained by using KNN(k = 1) is 16.35%.

```
# K=10 and predictions
auto.knn2 = knn(train.matrix, test.matrix, train.y, k=10)
table(auto.knn2, test.y)
```

```
##          test.y
## auto.knn2  0  1
##         0 48  1
```

```
##         1  8 47
```

```
1 - mean(auto.knn2 == Auto.test$mpg01)
```

```
## [1] 0.08653846
```

The test error of the model obtained by using KNN(k = 10) is 8.654%.

```
# K=100 and predictions
auto.knn3 = knn(train.matrix, test.matrix, train.y, k=100)
table(auto.knn3, test.y)
```

```
##          test.y
## auto.knn3  0  1
##         0 45  3
##         1 11 45
```

```
1 - mean(auto.knn3 == Auto.test$mpg01)
```

```
## [1] 0.1346154
```

The test error of the model obtained by using KNN(k = 100) is 13.46%. When K is 10, this data set performs best.

4.15. 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 23 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(a,b){a^b}
Power(2,3)
```

```
## [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. 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 38, namely, 6, 561.

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

(c) Using the Power2() function that you just wrote, compute 10^3, 81^7, and 131^3.

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

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

```
Power2(81,7)
```

```
## [1] 2.287679e+13
```

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

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

(d) Now create a new function, Power3(), that actually returns the result x^a as an R object, rather than simply printing it to the screen. That is, if 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(result) The line above should be the last line in your function, before the } symbol.

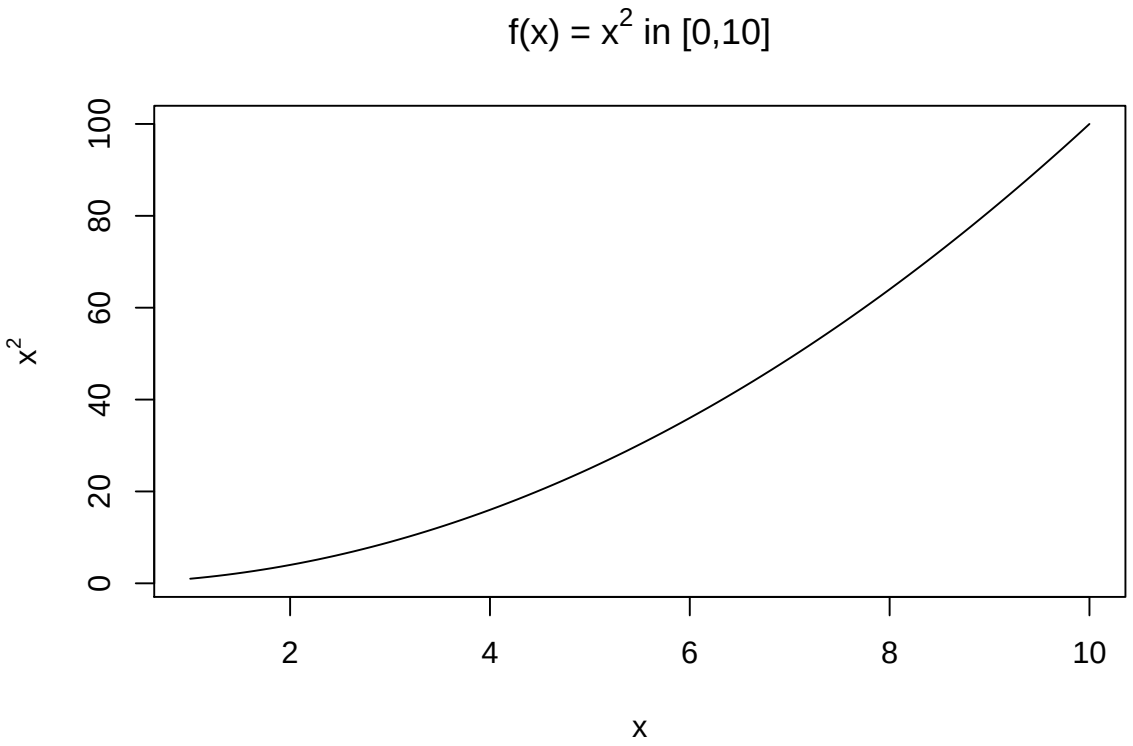
```
Power3 <- function(x, a) x^a
```

(e) Now using the Power3() function, create a plot of f(x) = x2. The x-axis should display a range of integers from 1 to 10, and the y-axis should display x2. 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 <- seq(from = 1, to = 10, length.out = 100)

plot(x, Power3(x, 2), type = 'l',
     ylab = expression(x^2),
     main = expression('f(x) = '*x^2*' in [0,10]'))
```



(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

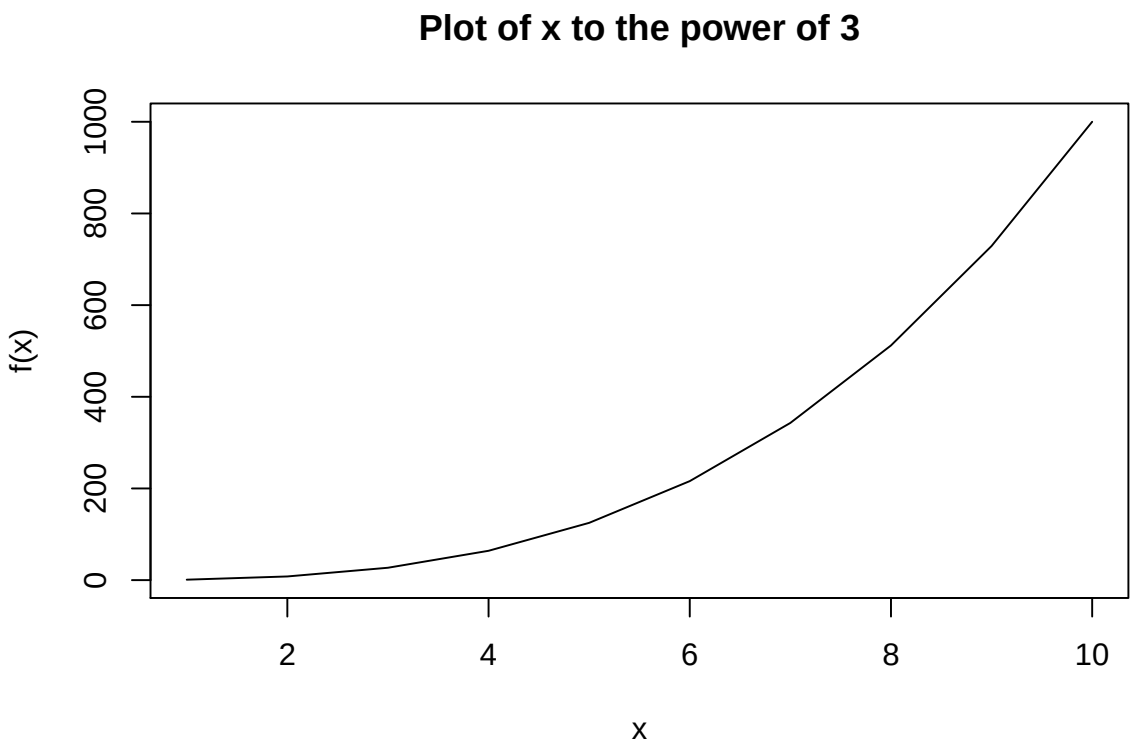
```
# PlotPower(1:10, 3)
```

then a plot should be created with an x-axis taking on values 1,2,…,10, and a y-axis taking on values 13,23,…,103.

```
PlotPower <- function(x, a) {

    plot(x, Power3(x, a), type = 'l', xlab = 'x', ylab = 'f(x)',
         main = paste('Plot of x to the power of', a))
}

PlotPower(1:10, 3)
```



4.16. 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.*

```
# Boston data set
library(MASS)
attach(Boston)
```

```
# fit classification models in order to predict whether a given census tract has a crime rate above or
# add 1 to a new column if crim > median, o otherwise.

Boston$crim01 = with(ifelse(crim>median(crim), 1, 0), data=Boston)

# check the correlation
cor(Boston)
```

```
##                crim          zn       indus         chas         nox
## crim     1.00000000 -0.20046922  0.40658341 -0.055891582  0.42097171
## zn      -0.20046922  1.00000000 -0.53382819 -0.042696719 -0.51660371
## indus    0.40658341 -0.53382819  1.00000000  0.062938027  0.76365145
## chas    -0.05589158 -0.04269672  0.06293803  1.000000000  0.09120281
## nox      0.42097171 -0.51660371  0.76365145  0.091202807  1.00000000
## rm      -0.21924670  0.31199059 -0.39167585  0.091251225 -0.30218819
## age      0.35273425 -0.56953734  0.64477851  0.086517774  0.73147010
## dis     -0.37967009  0.66440822 -0.70802699 -0.099175780 -0.76923011
## rad      0.62550515 -0.31194783  0.59512927 -0.007368241  0.61144056
## tax      0.58276431 -0.31456332  0.72076018 -0.035586518  0.66802320
## ptratio  0.28994558 -0.39167855  0.38324756 -0.121515174  0.18893268
## black   -0.38506394  0.17552032 -0.35697654  0.048788485 -0.38005064
## lstat    0.45562148 -0.41299457  0.60379972 -0.053929298  0.59087892
## medv    -0.38830461  0.36044534 -0.48372516  0.175260177 -0.42732077
```

```
## crim01   0.40939545 -0.43615103  0.60326017  0.070096774  0.72323480
##                 rm         age         dis          rad         tax    ptratio
## crim    -0.21924670  0.35273425 -0.37967009  0.625505145  0.58276431  0.2899456
## zn       0.31199059 -0.56953734  0.66440822 -0.311947826 -0.31456332 -0.3916785
## indus   -0.39167585  0.64477851 -0.70802699  0.595129275  0.72076018  0.3832476
## chas     0.09125123  0.08651777 -0.09917578 -0.007368241 -0.03558652 -0.1215152
## nox     -0.30218819  0.73147010 -0.76923011  0.611440563  0.66802320  0.1889327
## rm       1.00000000 -0.24026493  0.20524621 -0.209846668 -0.29204783 -0.3555015
## age     -0.24026493  1.00000000 -0.74788054  0.456022452  0.50645559  0.2615150
## dis      0.20524621 -0.74788054  1.00000000 -0.494587930 -0.53443158 -0.2324705
## rad     -0.20984667  0.45602245 -0.49458793  1.000000000  0.91022819  0.4647412
## tax     -0.29204783  0.50645559 -0.53443158  0.910228189  1.00000000  0.4608530
## ptratio -0.35550149  0.26151501 -0.23247054  0.464741179  0.46085304  1.0000000
## black    0.12806864 -0.27353398  0.29151167 -0.444412816 -0.44180801 -0.1773833
## lstat   -0.61380827  0.60233853 -0.49699583  0.488676335  0.54399341  0.3740443
## medv     0.69535995 -0.37695457  0.24992873 -0.381626231 -0.46853593 -0.5077867
## crim01  -0.15637178  0.61393992 -0.61634164  0.619786249  0.60874128  0.2535684
##               black      lstat       medv      crim01
## crim    -0.38506394  0.4556215 -0.3883046  0.40939545
## zn       0.17552032 -0.4129946  0.3604453 -0.43615103
## indus   -0.35697654  0.6037997 -0.4837252  0.60326017
## chas     0.04878848 -0.0539293  0.1752602  0.07009677
## nox     -0.38005064  0.5908789 -0.4273208  0.72323480
## rm       0.12806864 -0.6138083  0.6953599 -0.15637178
## age     -0.27353398  0.6023385 -0.3769546  0.61393992
## dis      0.29151167 -0.4969958  0.2499287 -0.61634164
## rad     -0.44441282  0.4886763 -0.3816262  0.61978625
## tax     -0.44180801  0.5439934 -0.4685359  0.60874128
## ptratio -0.17738330  0.3740443 -0.5077867  0.25356836
## black    1.00000000 -0.3660869  0.3334608 -0.35121093
## lstat   -0.36608690  1.0000000 -0.7376627  0.45326273
## medv     0.33346082 -0.7376627  1.0000000 -0.26301673
## crim01  -0.35121093  0.4532627 -0.2630167  1.00000000
```

```
# training and test set
set.seed(679)
Boston.sample = sample.split(Boston$crim01, SplitRatio = 0.80)
Boston.train = subset(Boston, Boston.sample==TRUE)
Boston.test = subset(Boston, Boston.sample==FALSE)
```

```
# Logistic regression
boston.glm <- glm(crim01 ~ indus + nox + age + rad + tax, data = Boston.train, family = binomial)
summary(boston.glm)
```

```
##
## Call:
## glm(formula = crim01 ~ indus + nox + age + rad + tax, family = binomial,
##     data = Boston.train)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -1.97856  -0.28123  -0.01511   0.01019   2.63169
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
```

```
## (Intercept) -19.890085   3.008556  -6.611 3.81e-11 ***
## indus        -0.057770   0.045214  -1.278   0.2014
## nox          34.142638   6.430808   5.309 1.10e-07 ***
## age           0.015108   0.009343   1.617   0.1059
## rad           0.548842   0.125235   4.383 1.17e-05 ***
## tax          -0.005666   0.002592  -2.186   0.0288 *
## ---
## 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: 204.39  on 398  degrees of freedom
## AIC: 216.39
##
## Number of Fisher Scoring iterations: 8
```

```
boston.probs <- predict(boston.glm, Boston.test, type = "response")
boston.pred <- rep("0", length(boston.probs))
boston.pred[boston.probs > 0.5] = "1"
table(boston.pred, Boston.test$crim01)
```

```
##
## boston.pred  0  1
##           0 48 10
##           1  3 41
```

```
1 - mean(boston.pred == Boston.test$crim01)
```

```
## [1] 0.127451
```

```
# LDA
boston.lda <- lda(crim01 ~ indus + nox + age + rad + tax, data = Boston.train)
boston.lda
```

```
## Call:
## lda(crim01 ~ indus + nox + age + rad + tax, data = Boston.train)
##
## Prior probabilities of groups:
##   0   1
## 0.5 0.5
##
## Group means:
##       indus       nox      age       rad      tax
## 0  6.901782 0.4712609 50.05990  4.163366 303.4505
## 1 15.345149 0.6385594 85.68465 14.722772 509.2921
##
## Coefficients of linear discriminants:
##                 LD1
## indus  0.0101219026
## nox    6.9277528735
## age    0.0142674375
## rad    0.0778546423
## tax   -0.0009042392
```

```
pred.blda <- predict(boston.lda, Boston.test)
table(pred.blda$class, Boston.test$crim01)
```

```
##
##      0  1
##   0 48 12
##   1  3 39
```

```
1 - mean(pred.blda$class == Boston.test$crim01)
```

```
## [1] 0.1470588
```

```
# Naive Bayes
boston.nb <- naiveBayes(crim01 ~ indus + nox + age + rad + tax, data = Boston.train)
pred.bnb <- predict(boston.nb, Boston.test)
table(pred.bnb, Boston.test$crim01)
```

```
##
## pred.bnb  0  1
##        0 48 15
##        1  3 36
```

```
1 - mean(pred.bnb == Boston.test$crim01)
```

```
## [1] 0.1764706
```

```
# KNN models
# "indus","nox","age","rad","tax"
# indus,nox,age,dis,rad,tax
train.matrix = data.matrix(Boston.train[,c("indus","nox","age","rad","tax")])
test.matrix = data.matrix(Boston.test[,c("indus","nox","age","rad","tax")])
train.y = data.matrix(Boston.train$crim01)
test.y = data.matrix(Boston.test$crim01)

# K=1 and predictions
boston.knn1 = knn(train.matrix, test.matrix, train.y, k=1)
table(boston.knn1, test.y)
```

```
##            test.y
## boston.knn1  0  1
##           0 44  4
##           1  7 47
```

```
1 - mean(boston.knn1 == Boston.test$crim01)
```

```
## [1] 0.1078431
```

```
# K=5 and predictions
boston.knn2 = knn(train.matrix, test.matrix, train.y, k=5)
table(boston.knn2, test.y)
```

```
##            test.y
## boston.knn2  0  1
##           0 42  5
##           1  9 46
```

```
1 - mean(boston.knn2 == Boston.test$crim01)
```

```
## [1] 0.1372549
```

The result shows that KNN(k = 1) performs the best among these methods with an test error rate around 10%.