679 HW2

Xinyi Wang

2/7/2019

Q6.

(a)

```
p <- function(x1,x2){ z <- exp(-6 + 0.05*x1 + 1*x2); return( round(z/(1+z),2))}
p(40,3.5)
```

```
## [1] 0.38
```

(b)

```
f <- function(x,y)  ((exp(-6+0.05*x+3.5)/(1+exp(-6+0.05*x+3.5)))-y)
uniroot(f,y=0.5, lower=0, upper=1,extendInt = "yes")$root
```

```
## [1] 50
```

To have 50% of chance, he needs to study at least 50 hours.

Q8.

The logistic regression. When K=1 for KNN approach, the training error is zero, therefore the test error for KNN was 36%. It was higher than logistic test error.

Q9.

(a)

```
print( 0.37/(1+0.37))
```

```
## [1] 0.270073
```

(b)

```
odds <- .16/(1-.16)
odds
```

```
## [1] 0.1904762
```

Q10.

(a)

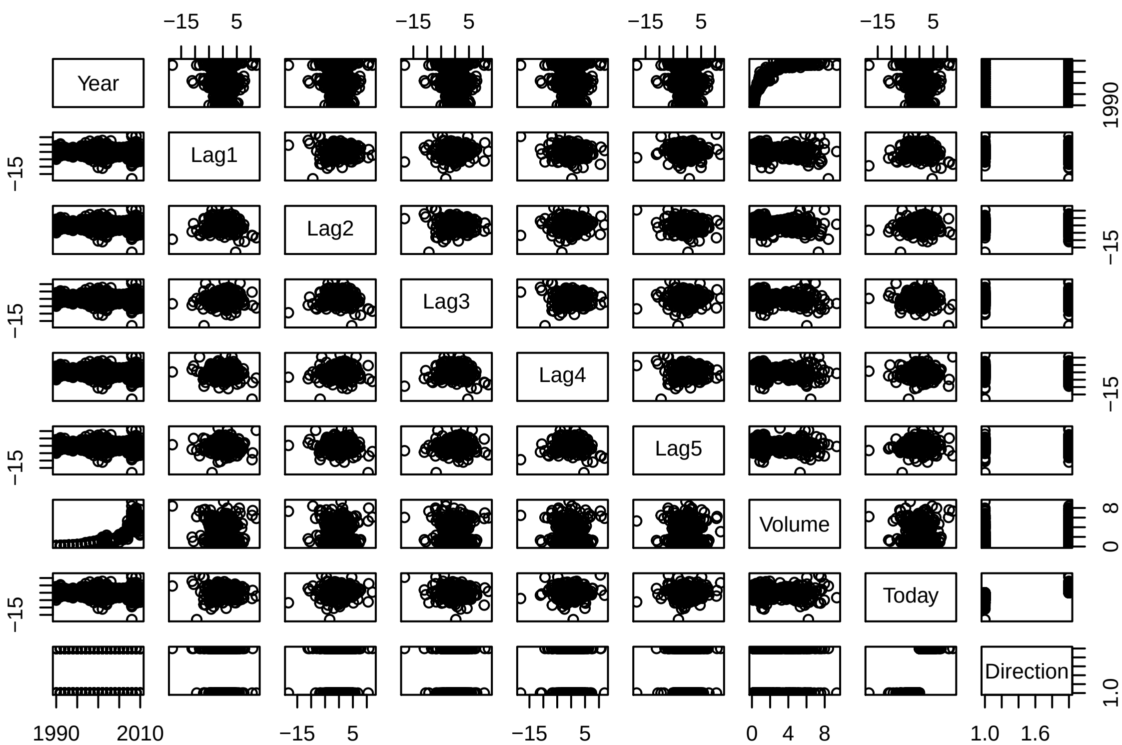
```
require(ISLR)
```

```
## Loading required package: ISLR
```

```r
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
```

```r
pairs(Weekly)
```



Seemingly, the only evidence is at Volume×Year, where shows a logarithmic pattern.

(b)

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

(c)

```r
library(caret)
```

```
## Loading required package: lattice
```

```
## Loading required package: ggplot2
```

```r
glm.probs <- predict(glm.fit, type="response")
predicted <- ifelse(glm.probs>.5, "Up", "Down")
predicted <- as.factor(predicted)
confusionMatrix(predicted,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
##
```

We may conclude that the percentage of correct predictions on the training data is (54+557)/1089 wich is equal to 56%. In other words 44% is the training error rate, which is often overly optimistic. We could also say that for weeks when the market goes up, the model is right 92% of the time (557/(48+557)). For weeks when the market goes down, the model is right only 11.1570248% of the time (54/(54+430)).

(d)

```r
trainset = (Weekly$Year<=2008)
testset = Weekly[!trainset,]

glm.fit.d <- glm(Direction ~ Lag2, data=Weekly, subset=trainset, family="binomial")
glm.probs.d <- predict(glm.fit.d, type="response", newdata=testset)
glm.preds.d <- ifelse(glm.probs.d>.5, "Up", "Down")
predicted2 <- as.factor(glm.preds.d)
confusionMatrix(predicted2,testset$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
##
```

Overall fraction of correct prediction is accuracy of ConfusionMatrix which is 0.625.

(e)

```
library(MASS)
lda.fit.e <- lda(Direction ~ Lag2, data=Weekly, subset=trainset)
predicted3 <- predict(lda.fit.e, newdata=testset)
confusionMatrix(predicted3$class,testset$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
##
```

Overall fraction of correct prediction is accuracy of ConfusionMatrix which is 0.625.

(f)

```
qda.fit.f <- qda(Direction ~ Lag2, data=Weekly, subset=trainset)
predicted4 <- predict(qda.fit.f, newdata=testset)
confusionMatrix(predicted4$class,testset$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
##
```

Overall fraction of correct prediction is accuracy of ConfusionMatrix which is 0.5865.

(g)

```r
library(class)
set.seed(1)

train.g = Weekly[trainset, c("Lag2", "Direction")]
knn.pred = knn(train=data.frame(train.g$Lag2), test=data.frame(testset$Lag2), cl=train.g$Direction, k=1
confusionMatrix(knn.pred,testset$Direction)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction Down Up
##       Down   21 30
##       Up     22 31
##
##                Accuracy : 0.5
##                  95% CI : (0.4003, 0.5997)
##     No Information Rate : 0.5865
##     P-Value [Acc > NIR] : 0.9700
##
##                   Kappa : -0.0033
##  Mcnemar's Test P-Value : 0.3317
##
##             Sensitivity : 0.4884
##             Specificity : 0.5082
##          Pos Pred Value : 0.4118
##          Neg Pred Value : 0.5849
##              Prevalence : 0.4135
##          Detection Rate : 0.2019
##    Detection Prevalence : 0.4904
##       Balanced Accuracy : 0.4983
##
##        'Positive' Class : Down
##
```

Overall fraction of correct prediction is accuracy of ConfusionMatrix which is 0.5.

(h)

Logistic Regression and LDA.

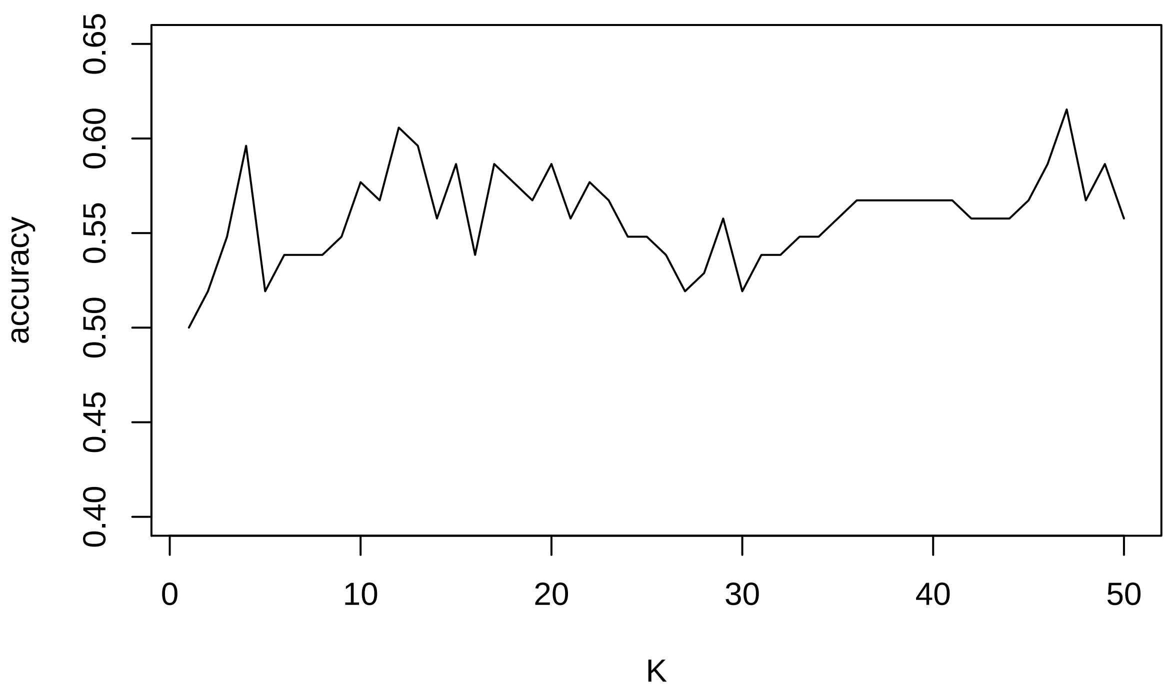
(i)

K-NN

```
set.seed(1)

results <- data.frame(k=1:50, acc=NA)
for(i in 1:50){
  knn.pred = knn(train=data.frame(train.g$Lag2), test=data.frame(testset$Lag2), cl=train.g$Direction, k
  cm <- table(testset$Direction, knn.pred)
  acc <- (cm["Down", "Down"] + cm["Up", "Up"])/sum(cm)
  results$acc[i] <- acc
}

plot(x=results$k, y=results$acc, type="l", xlab="K", ylab="accuracy", ylim=c(.4,.65))
```



The K doesn't seem to affect the accuracy values too much. Now, using a QDA model with all Lags predictors plus Volume. ##QDA

```
qda.fit <- qda(Direction ~ Lag1+Lag2+Lag3+Lag4+Lag5+Volume, data=Weekly, subset=trainset)
qda.preds <- predict(qda.fit, testset)
# show accuracy
print( sum(qda.preds$class==testset$Direction)/length(qda.preds$class))
```

```
## [1] 0.4326923
```

It had a worse performance than using only the Lag2 predictor shown in g. Again on QDA model, i try with interactive variables between all Lags predictors.

```r
qda.fit <- qda(Direction ~ Lag1*Lag2*Lag3*Lag4*Lag5 + Volume, data=Weekly, subset=trainset)
qda.preds <- predict(qda.fit, testset)
# show accuracy
print( sum(qda.preds$class==testset$Direction)/length(qda.preds$class))
```

```
## [1] 0.4134615
```

The accuracy was even worse than before.For last, i try the same predictors schema using LDA. ##LDA

```r
lda.fit <- lda(Direction ~ Lag1*Lag2*Lag3*Lag4*Lag5 + Volume, data=Weekly, subset=trainset)
lda.preds <- predict(lda.fit, testset)
# show accuracy
print( sum(lda.preds$class==testset$Direction)/length(lda.preds$class))
```

```
## [1] 0.4423077
```

The LDA performance kept similar of the QDA.

Q11

(a)

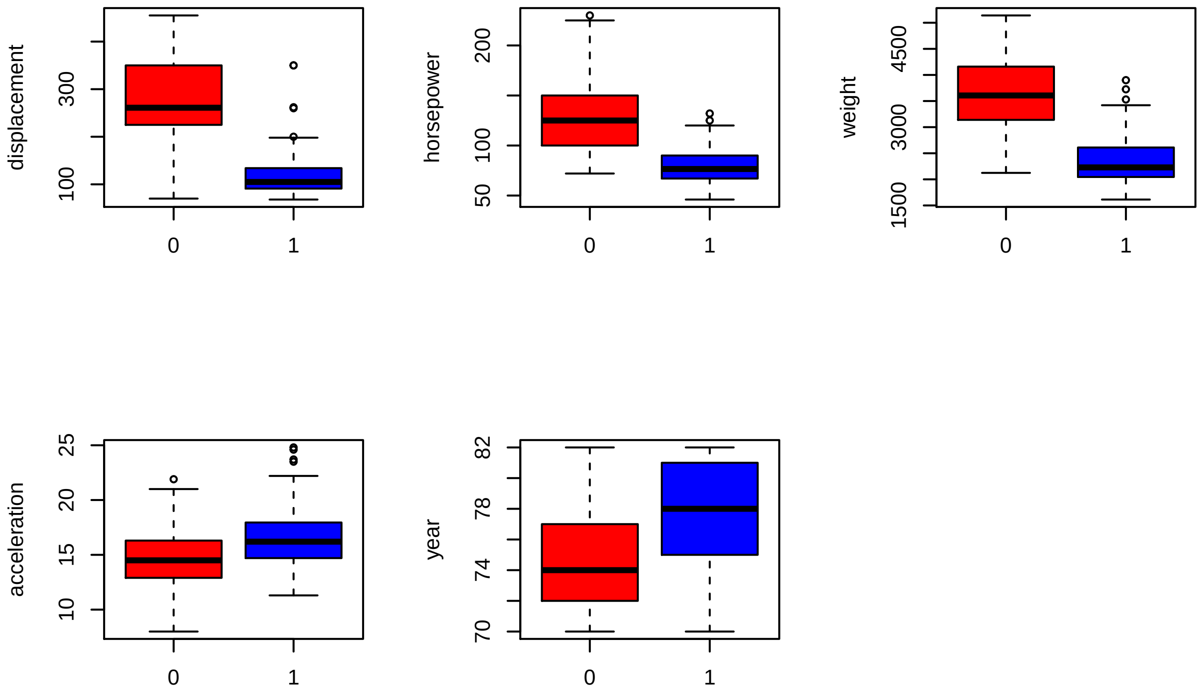
```r
# remove(list=ls())
data(Auto)
Auto$mpg01 <- with(ifelse(mpg>median(mpg), "1", "0"), data=Auto)
```

(b)

```r
attach(Auto)
```

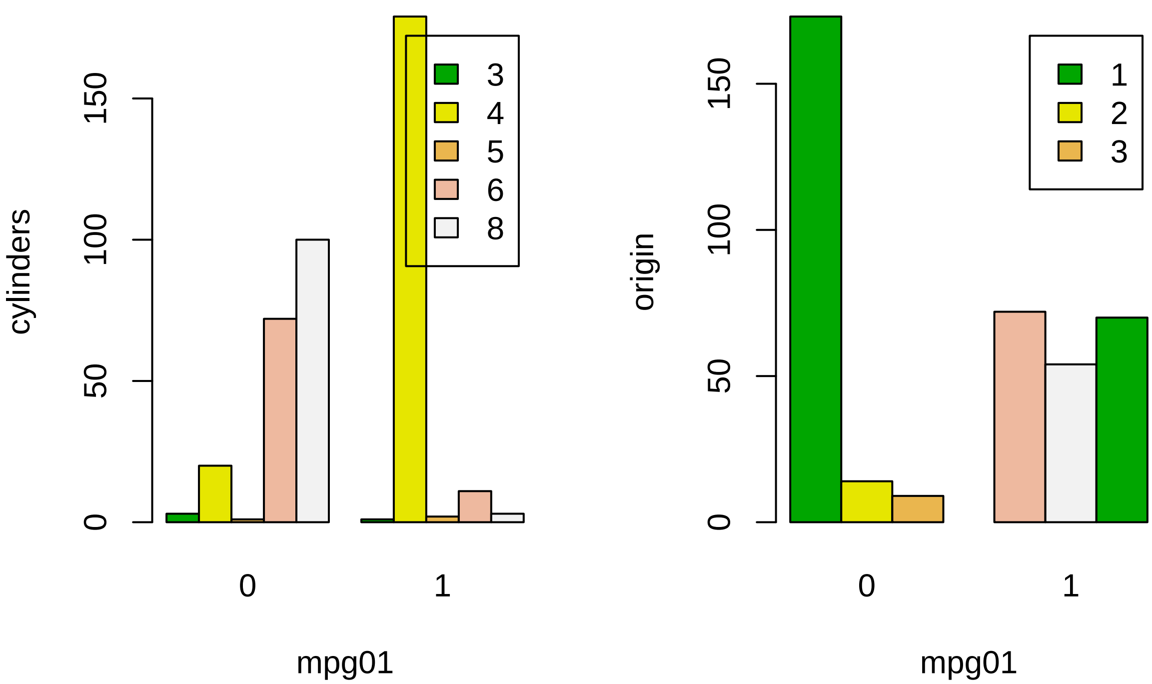
```
## The following object is masked from package:ggplot2:
##
##     mpg
```

```r
# Boxplots
par(mfrow=c(2,3))
for(i in names(Auto)){
  # excluding the own mpgs variables and others categorical variables
  if( grepl(i, pattern="^mpg|cylinders|origin|name")){ next }
  boxplot(eval(parse(text=i)) ~ mpg01, ylab=i, col=c("red", "blue"))
}
```



As shown in the boxplot, all variables present some trend with mpg01.

```
# for the categorical variables i do barplots
par(mfrow=c(1,2))
for(i in c("cylinders", "origin")){
  aux <- table(eval(parse(text=i)), mpg01)
  cols <- terrain.colors(5)
  barplot(aux, xlab="mpg01", ylab=i, beside=T,  legend=rownames(aux), col=cols)
}
```



At the barplots, cylinders and origin also show relation with mpg01. For instance, on dataset cars of lower mpg are majoraty from origin 1, which is American.

(c)

```r
# splitting the train and test set into 75% and 25%
set.seed(1)
rows <- sample(x=nrow(Auto), size=.75*nrow(Auto))
trainset <- Auto[rows, ]
testset <- Auto[-rows, ]
```

(d)

```r
library(MASS)
lda.fit <- lda(mpg01 ~ displacement+horsepower+weight+acceleration+year+cylinders+origin, data=trainset)
lda.pred <- predict(lda.fit, testset)
testset$mpg01 <- as.factor(testset$mpg01)
confusionMatrix(lda.pred$class,testset$mpg01)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 41  0
##          1  5 52
##
##                Accuracy : 0.949
##                  95% CI : (0.8849, 0.9832)
##     No Information Rate : 0.5306
##     P-Value [Acc > NIR] : < 2e-16
##
##                   Kappa : 0.8969
##  Mcnemar's Test P-Value : 0.07364
##
##             Sensitivity : 0.8913
##             Specificity : 1.0000
##          Pos Pred Value : 1.0000
##          Neg Pred Value : 0.9123
##              Prevalence : 0.4694
##          Detection Rate : 0.4184
##    Detection Prevalence : 0.4184
##       Balanced Accuracy : 0.9457
##
##        'Positive' Class : 0
##
```

```r
# test-error
round(sum(lda.pred$class!=testset$mpg01)/nrow(testset)*100,2)
```

```
## [1] 5.1
```

(e)

```r
qda.fit <- qda(mpg01 ~ displacement+horsepower+weight+acceleration+year+cylinders+origin, data=trainset)
qda.pred <- predict(qda.fit, testset)
confusionMatrix(qda.pred$class,testset$mpg01)
```

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

```
## Prediction  0  1
##          0 43  1
##          1  3 51
##
##                Accuracy : 0.9592
##                  95% CI : (0.8988, 0.9888)
##     No Information Rate : 0.5306
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.9179
##  Mcnemar's Test P-Value : 0.6171
##
##             Sensitivity : 0.9348
##             Specificity : 0.9808
##          Pos Pred Value : 0.9773
##          Neg Pred Value : 0.9444
##              Prevalence : 0.4694
##          Detection Rate : 0.4388
##    Detection Prevalence : 0.4490
##       Balanced Accuracy : 0.9578
##
##        'Positive' Class : 0
##
```

```r
# test-error
round(sum(qda.pred$class!=testset$mpg01)/nrow(testset)*100,2)
```

```
## [1] 4.08
```

(f)

```r
lr.fit <- glm(as.factor(mpg01) ~ displacement+horsepower+weight+acceleration+year+cylinders+origin, data
lr.probs <- predict(lr.fit, testset, type="response")
lr.pred <- ifelse(lr.probs>0.5, "1", "0")
lr.pred <- as.factor(lr.pred)
confusionMatrix(lr.pred,testset$mpg01)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 41  1
##          1  5 51
##
##                Accuracy : 0.9388
##                  95% CI : (0.8715, 0.9772)
##     No Information Rate : 0.5306
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.8765
##  Mcnemar's Test P-Value : 0.2207
##
##             Sensitivity : 0.8913
##             Specificity : 0.9808
##          Pos Pred Value : 0.9762
##          Neg Pred Value : 0.9107
```

```
##               Prevalence : 0.4694
##           Detection Rate : 0.4184
##     Detection Prevalence : 0.4286
##        Balanced Accuracy : 0.9360
##
##         'Positive' Class : 0
##
```

```r
# test-error
round(sum(lr.pred!=testset$mpg01)/nrow(testset)*100,2)
```

```
## [1] 6.12
```

(g)

```r
library(class)

sel.variables <- which(names(trainset)%in%c("mpg01", "displacement", "horsepower", "weight", "accelerat

set.seed(1)
accuracies <- data.frame("k"=1:10, acc=NA)
for(k in 1:10){
  knn.pred <- knn(train=trainset[, sel.variables], test=testset[, sel.variables], cl=trainset$mpg01, k=

  # test-error
  accuracies$acc[k]= round(sum(knn.pred!=testset$mpg01)/nrow(testset)*100,2)
}

accuracies
```

```
##     k   acc
## 1   1 16.33
## 2   2 20.41
## 3   3 14.29
## 4   4 14.29
## 5   5 13.27
## 6   6 13.27
## 7   7 12.24
## 8   8 14.29
## 9   9 14.29
## 10 10 13.27
```

The k=7 was the best response, outperformed all others.

Q12

(a)

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

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

(b)

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

```
}

Power2(3,8)
```

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

(c)

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

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

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

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

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

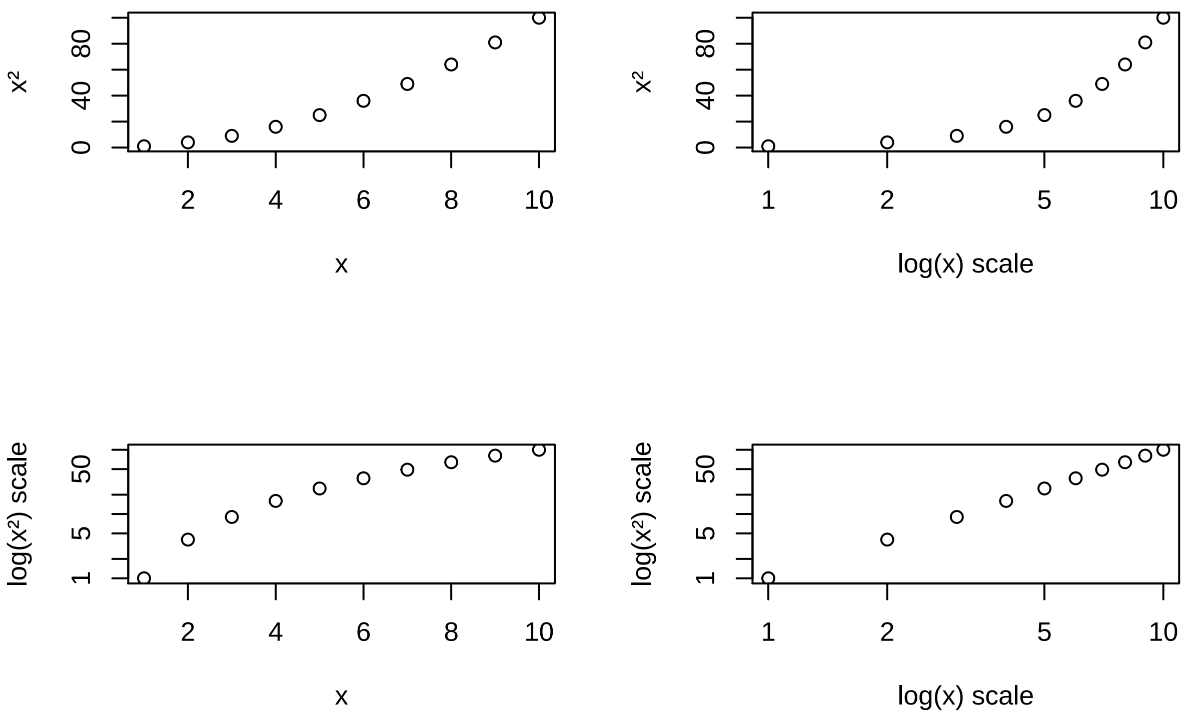
```
## [1] 2248091
```

(d)

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

(e)

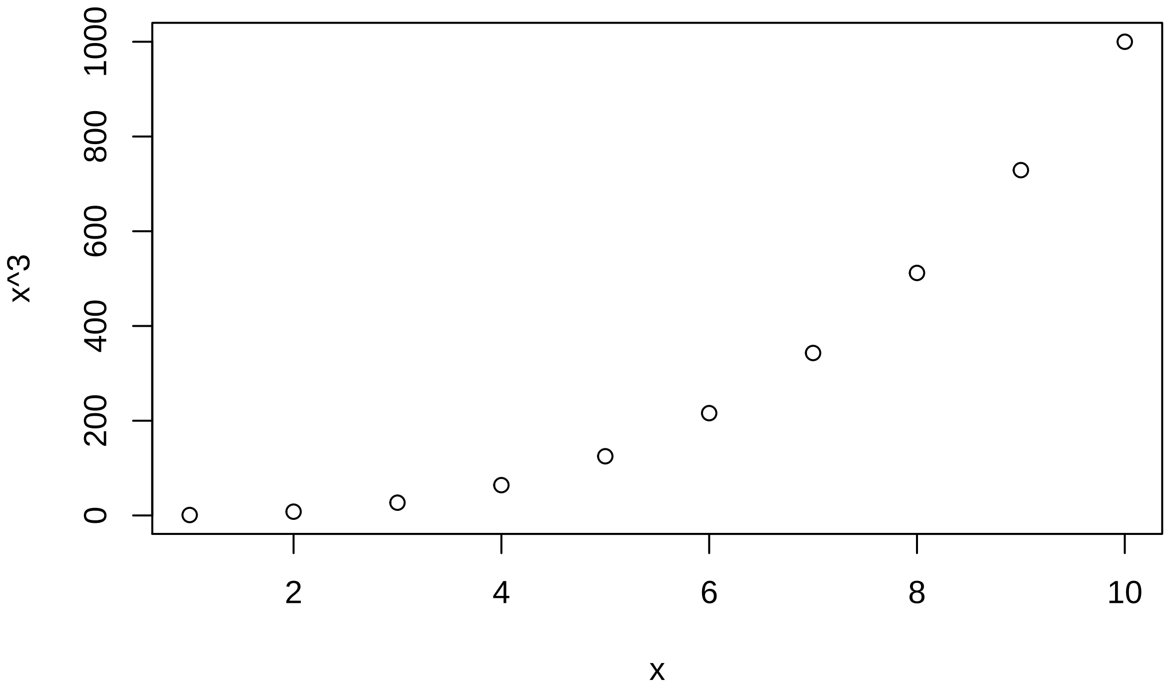
```
par(mfrow=c(2,2))
plot(x = x<-1:10, y= y<-Power3(x,2), xlab="x", ylab="x²")
plot(x,y,log="x", xlab="log(x) scale", ylab="x²")
plot(x,y,log="y", xlab="x", ylab="log(x²) scale")
plot(x,y,log="xy", xlab="log(x) scale", ylab="log(x²) scale")
```



(f)

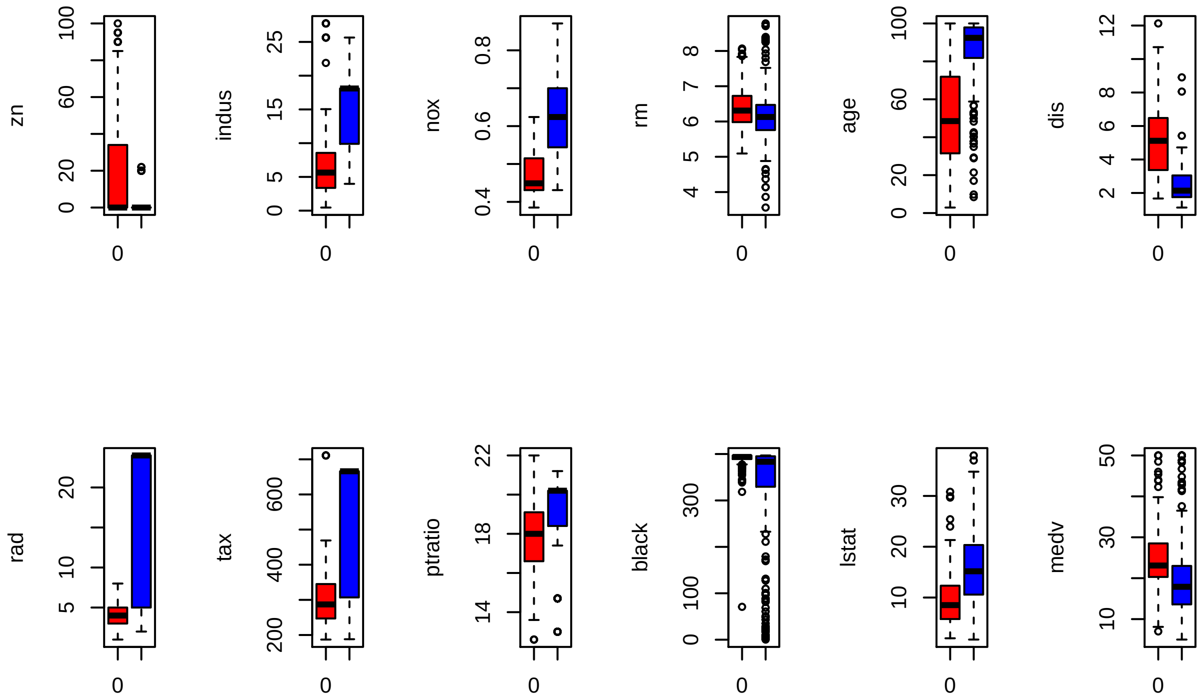
```r
par(mfrow=c(1,1))
PlotPower <- function(x,a){
  plot(x = x, y= y<-Power3(x,a), xlab="x", ylab=paste0("x^",a))
}

PlotPower(1:10,3)
```



Q13

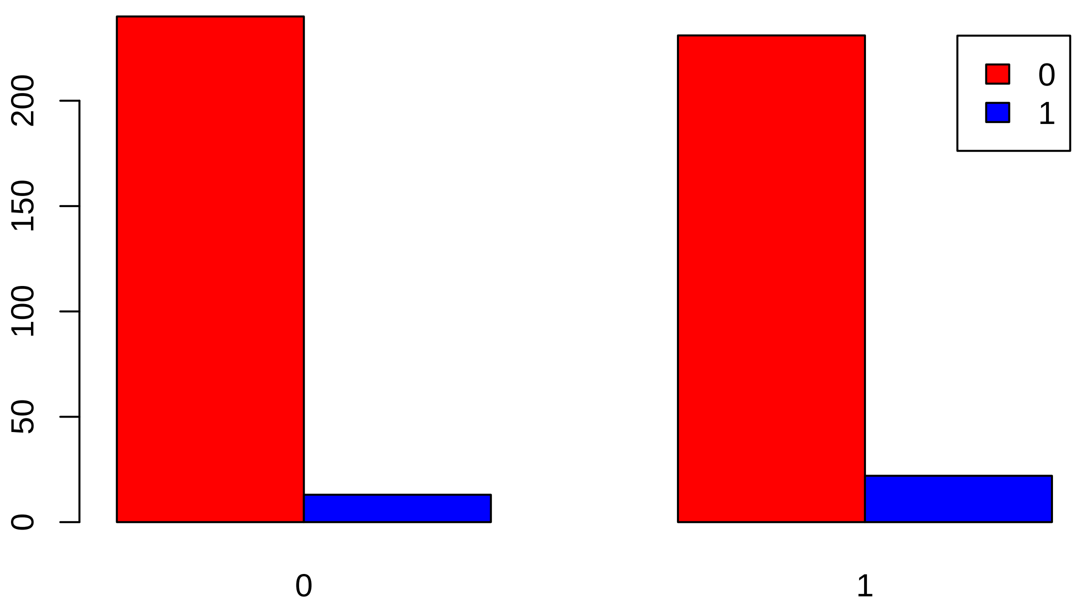
```r
data("Boston")
Boston$crim01 <- ifelse(Boston$crim > median(Boston$crim), "1", "0")
attach(Boston)
par(mfrow=c(2,6))
for(i in names(Boston)){
  # excluding the own crime variables and the chas variable
  if( grepl(i, pattern="^crim|^chas")){ next}
  boxplot(eval(parse(text=i)) ~ crim01, ylab=i, col=c("red", "blue"), varwidth=T)
}
```



All variable shows trend to crim01, exceptrm which has some difference among the crimes situation but its most population lies in the same range values.

For Chas variable, i do a barplot, it is a dummy variable to if the tract bounds the river.

```
par(mfrow=c(1,1))
aux <- table(chas, crim01)
barplot(aux, beside = T, legend=rownames(aux), col=c("red", "blue"))
```



The chas doesn't show much difference for crime situation.

Selecting the relevant variables, i use the: zn, indus, nox, age, dis, rad, tax, ptratio, black, lstat and medv.

```
set.seed(1)
vars = c("zn", "indus", "nox", "age", "dis", "rad", "tax", "ptratio", "black", "lstat", "medv", "crim01
rows = sample(x=nrow(Boston), size=.75*nrow(Boston))
```

```
trainset = Boston[rows, vars]
testset = Boston[-rows, vars]
```

Modeling Round 1

```
# LOGISTIC REGRESSION
lr.fit <- glm(as.factor(crim01) ~ ., data=trainset, family="binomial")
lr.probs <- predict(lr.fit, testset, type="response")
lr.pred <- ifelse(lr.probs>.5, "1","0")

test.err.lr <- mean(lr.pred!=testset$crim01)

# LINEAR DISCRIMINANT ANALYSIS
lda.fit <- lda(crim01 ~ ., data=trainset)
lda.pred <- predict(lda.fit, testset)
test.err.lda <- mean(lda.pred$class!=testset$crim01)

# QUADRATIC DISCRIMINANT ANALYSIS
qda.fit <- qda(crim01 ~ ., data=trainset)
qda.pred <- predict(qda.fit, testset)
test.err.qda <- mean(qda.pred$class!=testset$crim01)

# KNN-1
knn.pred <- knn(train=trainset, test=testset, cl=trainset$crim01, k=1)
test.err.knn_1 <- mean(knn.pred!=testset$crim01)

# KNN-CV
err.knn_cv <- rep(NA,9)
for(i in 2:10){
  knn.pred <- knn(train=trainset, test=testset, cl=trainset$crim01, k=i)
  err.knn_cv[i-1] <- mean(knn.pred!=testset$crim01)
}
test.err_knn_CV <- min(err.knn_cv)

round1 = data.frame("method"=c("LR", "LDA", "QDA", "KNN-1", "KNN-CV"), test.err=c(test.err.lr, test.err
round1
```

```
##   method   test.err
## 1     LR 0.08661417
## 2    LDA 0.14173228
## 3    QDA 0.13385827
## 4  KNN-1 0.08661417
## 5 KNN-CV 0.07874016
```

Both KNN methods outperforms the others, maybe it's related to the form of the data, which can be more non-linear and either differs more from a gaussian shape. The logistic regression performs better than LDA and QDA, that enhances the assumption of a non Gaussian distribution from the data. And as QDA performs better than LDA, i can imagine that the non-linear decision boundary helps this decision. So the non-parametric method presents the best results.

Doing a second round of modelling, this time choosing only the predictors which seemed more relevants by the logistic regression coefficients. Cheking the p-values:

```r
coefs <- summary(lr.fit)$coefficients
coefs[order(coefs[,"Pr(>|z|)"], decreasing=F),]
```

```
##                 Estimate  Std. Error    z value     Pr(>|z|)
## nox          46.268552442 8.367479623  5.5295686 3.210193e-08
## (Intercept) -34.234056447 7.272602370 -4.7072636 2.510642e-06
## rad           0.605101758 0.171668295  3.5248312 4.237528e-04
## age           0.040524724 0.012981423  3.1217475 1.797811e-03
## dis           0.720872986 0.253963152  2.8384944 4.532691e-03
## zn           -0.083465982 0.037217704 -2.2426419 2.491992e-02
## tax          -0.007250196 0.003264131 -2.2211721 2.633931e-02
## medv          0.096672482 0.048899905  1.9769462 4.804771e-02
## ptratio       0.216821863 0.128042186  1.6933627 9.038645e-02
## indus        -0.064414786 0.051403490 -1.2531209 2.101617e-01
## black        -0.006369013 0.005175739 -1.2305513 2.184907e-01
## lstat        -0.001571527 0.056064484 -0.0280307 9.776377e-01
```

I choose nox, rad, ptratio, black and medv.

```r
vars <- c("nox", "rad", "ptratio", "black", "medv", "dis", "crim01")
trainset = Boston[rows, vars]
testset = Boston[-rows, vars]
```

Modeling Round 2

```r
# LOGISTIC REGRESSION
lr.fit <- glm(as.factor(crim01) ~ ., data=trainset, family="binomial")
lr.probs <- predict(lr.fit, testset, type="response")
lr.pred <- ifelse(lr.probs>.5, "1","0")

test.err.lr <- mean(lr.pred!=testset$crim01)

# LINEAR DISCRIMINANT ANALYSIS
lda.fit <- lda(crim01 ~ ., data=trainset)
lda.pred <- predict(lda.fit, testset)
test.err.lda <- mean(lda.pred$class!=testset$crim01)

# QUADRATIC DISCRIMINANT ANALYSIS
qda.fit <- qda(crim01 ~ ., data=trainset)
qda.pred <- predict(qda.fit, testset)
test.err.qda <- mean(qda.pred$class!=testset$crim01)

# KNN-1
knn.pred <- knn(train=trainset, test=testset, cl=trainset$crim01, k=1)
test.err.knn_1 <- mean(knn.pred!=testset$crim01)

# KNN-CV
err.knn_cv <- rep(NA,9)
for(i in 2:10){
  knn.pred <- knn(train=trainset, test=testset, cl=trainset$crim01, k=i)
  err.knn_cv[i-1] <- mean(knn.pred!=testset$crim01)
}
test.err_knn_CV <- min(err.knn_cv)
```

```
round2 = data.frame("method"=c("LR", "LDA", "QDA", "KNN-1", "KNN-CV"), test.err=c(test.err.lr, test.err
round2
```

```
##   method   test.err
## 1     LR 0.07874016
## 2    LDA 0.13385827
## 3    QDA 0.15748031
## 4  KNN-1 0.08661417
## 5 KNN-CV 0.10236220
```

On round 2, the general performance was worse for all approachs, so probably there are relevent information in the excluded variables.

Now, i try again, using the most 6 variable that seemed, in my observation from the graphs shown before, more associated with crime index. They are zn, indus, nox, dis, rad and tax.

```
vars <- c("zn","indus", "nox", "dis", "rad", "tax", "crim01")
trainset = Boston[rows, vars]
testset = Boston[-rows, vars]
```

Modeling Round 2

```
# LOGISTIC REGRESSION
lr.fit <- glm(as.factor(crim01) ~ ., data=trainset, family="binomial")
lr.probs <- predict(lr.fit, testset, type="response")
lr.pred <- ifelse(lr.probs>.5, "1","0")

test.err.lr <- mean(lr.pred!=testset$crim01)

# LINEAR DISCRIMINANT ANALYSIS(LDA)
lda.fit <- lda(crim01 ~ ., data=trainset)
lda.pred <- predict(lda.fit, testset)
test.err.lda <- mean(lda.pred$class!=testset$crim01)

# QUADRATIC DISCRIMINANT ANALYSIS(QDA)
qda.fit <- qda(crim01 ~ ., data=trainset)
qda.pred <- predict(qda.fit, testset)
test.err.qda <- mean(qda.pred$class!=testset$crim01)

# KNN-1
knn.pred <- knn(train=trainset, test=testset, cl=trainset$crim01, k=1)
test.err.knn_1 <- mean(knn.pred!=testset$crim01)

# KNN-CV
err.knn_cv <- rep(NA,9)
for(i in 2:10){
  knn.pred <- knn(train=trainset, test=testset, cl=trainset$crim01, k=i)
  err.knn_cv[i-1] <- mean(knn.pred!=testset$crim01)
}
test.err_knn_CV <- min(err.knn_cv)

round3 = data.frame("method"=c("LR", "LDA", "QDA", "KNN-1", "KNN-CV"), test.err=c(test.err.lr, test.err
round3
```

```
##   method   test.err
## 1     LR 0.09448819
## 2    LDA 0.14960630
## 3    QDA 0.08661417
## 4  KNN-1 0.00000000
## 5 KNN-CV 0.04724409
```

Surprisingly, the third round of my chosen variable, based on the boxplot, had the greatest performance of the previous rounds. Mainly the QDA and KNNs approachs. KNN-1 had showed test error of 0.7%. The linear approachs were very bad.

When i eliminate some variables, it helped for the non-linear approachs did better models. Seeing the three rounds on the graph bellow.

```
performances <- rbind(cbind(round="round_1", round1), cbind(round="round_2", round2), cbind(round="round

library(reshape2)
dcast(data=performances, method ~ round, value.var="test.err")
```

```
##   method    round_1    round_2    round_3
## 1  KNN-1 0.08661417 0.08661417 0.00000000
## 2 KNN-CV 0.07874016 0.10236220 0.04724409
## 3    LDA 0.14173228 0.13385827 0.14960630
## 4     LR 0.08661417 0.07874016 0.09448819
## 5    QDA 0.13385827 0.15748031 0.08661417
```

```
library(ggplot2)
ggplot(data=performances, aes(x=method,y=test.err)) + geom_bar(stat="identity", aes(fill=method)) + coo
```