Lab 2: KNN, Logistic Regression and LDA - Solutions

```
#install.packages("ggplot")
#install.packages("class")
#install.packages("ISLR")

# load libraries
library(ggplot2)
library(class)

# clean environment
rm(list=ls())
# Ctrl+L to clear console
```

Question 1

In this question you will produce a picture like Figure 2.2 in the book Elements of Statistical Learning on pp.15, but for a different dataset. Try to understand first the code for this Figure by running the posted file mixture.R, where the generated dataset for Figure 2.2 and the code is given.

a. Generate a dataset

Generate a dataset consisting of 100 observations from the logit model in the lab2.pdf file.

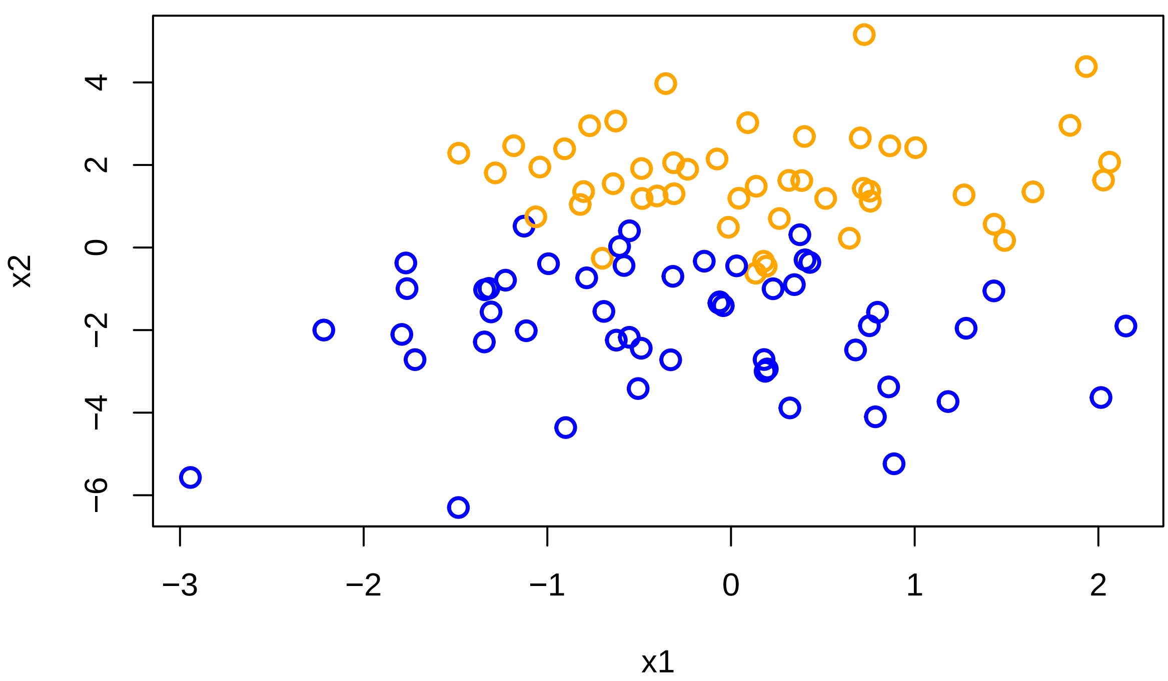
```
n<-100

# set sample size and seed (to ensure you get the same results when repeatedly running the code)
set.seed(666)

# generate X1, X2
x1 <- rnorm(n)
x2 <- 2*rnorm(n)

# define parameters
b1<-2
b2<-3

# linear combination (the term inside exponential in the question)
z = b1*x1 + b2*x2

# pass through an inv-logit function
pr = 1/(1+exp(-z))

# bernoulli response variable
# first imput is nr of observation, second is number of bernoulli trials
```

```
# and third one is probability of success in each trial
y = as.factor(rbinom(n,1,pr))
```

b. Plot the data

Plot the data in the two dimensions x_1 and x_2, using orange and blue circles for the two classes in y.

```
# set color pallete
col.list <- c("blue","orange")
palette(col.list)
# when you say col = y, it encolors observations according to their category
# with the default colors we set above
plot1<-plot(x1,x2, pch=1, col=y,cex=1.3, lwd=2)
```

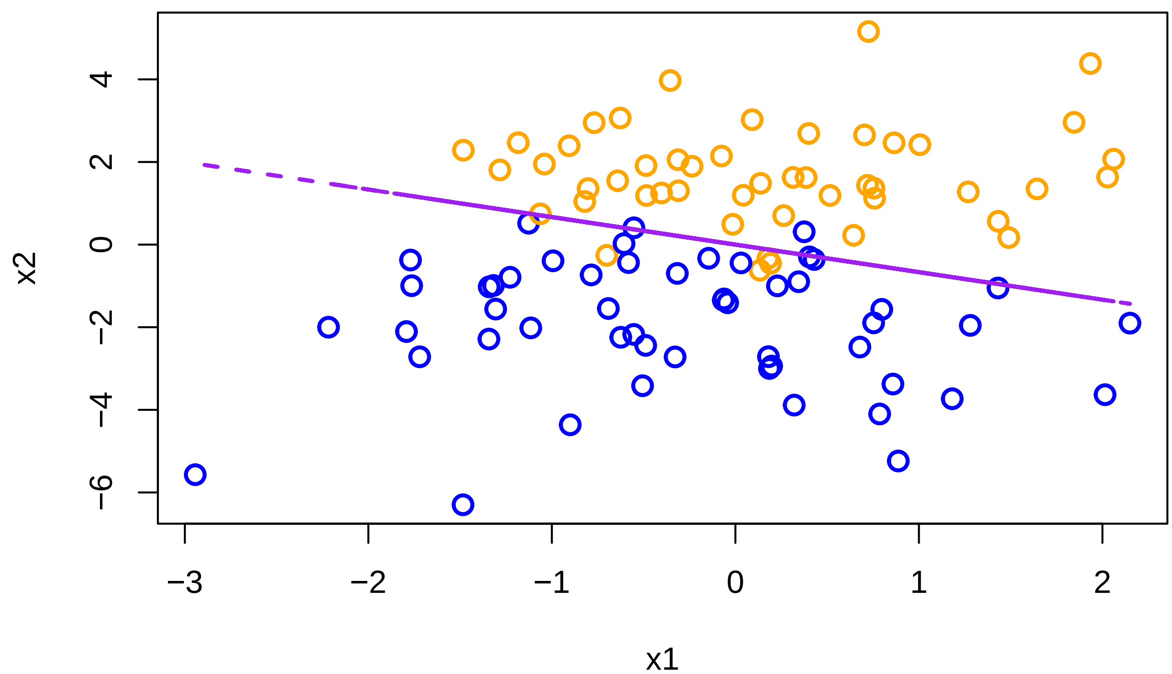


C. Bayes decision boundary

The Bayes decision boundary are the points x_1 and x_2 such that $Pr(y = 1|x) = 0.5$. For each x_1 in the simulated (or training) data, calculate x_2 such that $Pr(y = 1|x) = 0.5$ and add the Bayes decision boundary on the plot in b) using a dashed purple line. Is this boundary linear and can you find the exact formula for it? Explain.

Yes, it is linear, because $Pr(y = 1|x) = 0.5 = \frac{1}{1+exp(-\beta_1 x_1 - \beta_2 x_2)} = \frac{exp(z)}{1+exp(z)}$ means that the odds ratio is 1, or the log odds ratio is zero: $log(exp(z)) = 0$, so $z = \beta_1 x_1 + \beta_2 x_2 = 0$, from which $x_2 = -\frac{\beta_1}{\beta_2} x_1$, a line.

```
# create a pair of x1 and x2 as a function of x1
bayes= data.frame(x1,-b1*x1/b2)
plot1<-plot(x1,x2, pch=1, col=y,cex=1.3, lwd=2)

# adds the Bayes decision boundary
lines(bayes, type="l", lty=2, lwd=2, col="purple")
```



D. Test set

Construct a test set on a grid of $g = 50$ values for x_1 and x_2, ranging from their minimum to their maximum. Generate a test set from each combination of x_1 and x_2, and call it **test**. Gather the training set for x into a data frame called **train**.

```
g<-50
# create a sequence starting from minimum of x1 till maximum of x1 with 50 elements
x1test=seq(min(x1),max(x1), length.out=g)
x2test<-seq(min(x2),max(x2), length.out=g)

# train and test data
train<-data.frame(x1,x2)
test<- expand.grid(x1test, x2test)

# An equivalent way to construct the test sample:
# x1testg<-rep(x1test,g)
```

```
# x2testg<-kronecker(x2test,rep(1,g))
# test <-data.frame(x1testg,x2testg)
```

E. KNN

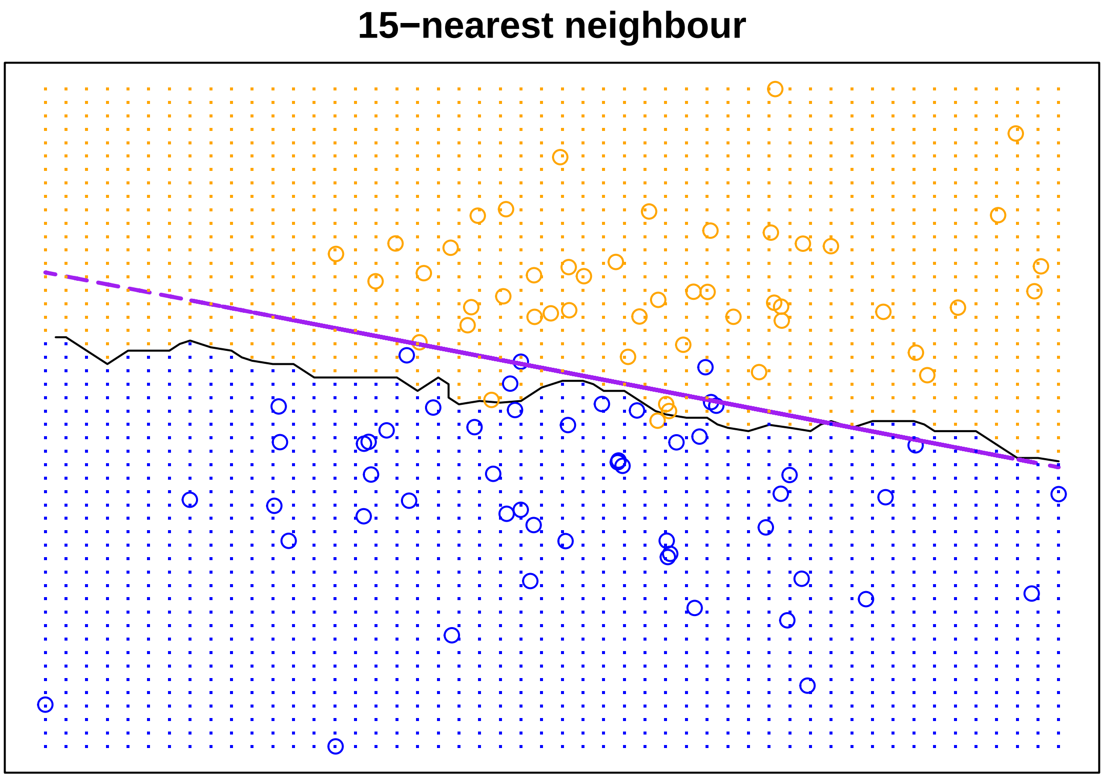
Run a KNN analysis with $k = 3$ nearest negihbors on the test data using the training data and the realizations of y. Use the command **knn()**.

```
# if you scale the data. Don't scale Y!
# we may scale them because knn works according to
# distance between points, since x2 has a larger variance
# it may be the case that they are in different units of measure
# to show an example with scaling, we scale here you may skip it
# if you want
s.train=scale(train)
s.test=scale(test)


mod15 <- knn(s.train, s.test,y, k=15, prob=TRUE)
# setting "prob" as "true", the proportion of the votes for the winning class are returned
# as attribute "prob". By default the "prob" is "false", and the attribute will be "null".
```

F. Plot the classes and the decision boundary

Following the **mixture.R** code, plot the training data with circles, the test data with dots, each with the color blue or orange according to which class they either belong to (in the training data) or to which class they were assigned to (in the test data). Add the KNN decision boundary to the plot using contour() and the Bayes decision boundary.

```
# retrieve the proportion of the votes for the winning class for each observation
prob15 <- attr(mod15, "prob")

# get the probability of being in class 1 for each observation
# fix potential label inversion problem (may not always occur)
prob15 <- ifelse(mod15=="1", prob15, 1-prob15)

# matrix of probabilities for test grid in two dimensions
probm15 <- matrix(prob15, g, g)

# set the margin parameter of the plot
par(mar=rep(2,4))

contour(x1test, x2test, probm15, levels=0.5, labels="", xlab="X1", ylab="X2", main=
          "15-nearest neighbour", axes=FALSE)
points(train, col=ifelse(y==1, "orange", "blue"))
lines(bayes, type="l", lty=2, lwd=2, col="purple")
points(test, pch=".", cex=1.2, col=ifelse(probm15>0.5, "orange", "blue"))
box()
```



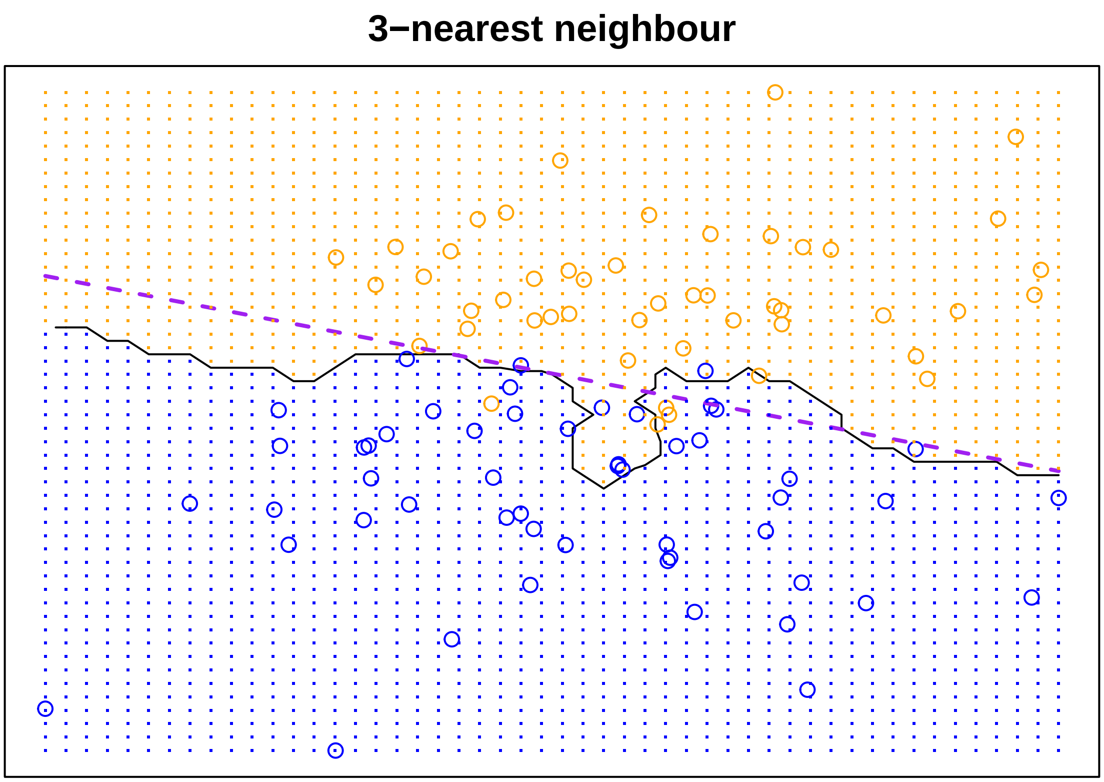
G. Choice of K and test error rate

Repeat e) and f) with $k = 3$ and $k = 10$. Calculate the test error rate for each of $k = 3, 10, 15$, and the Bayes decision boundary. Which k gets closest to the Bayes decision boundary? Explain why this makes sense or not.

K=10 gets closest to the Bayes decision boundary. The test error rates display a U-shape as K increases. This is due to the variance-bias trade-off. For small K: It is using nearest (most similar) neighbors. The bias is small. It has a good within-sample fit. The boundary is more zigzag. But the variance is large. It could overfit the training data and perform badly when predicting. For large K: It is including neighbors further away (more noise). The bias is high but the variance is low. When the bias is too large, the test error rate could again increase.

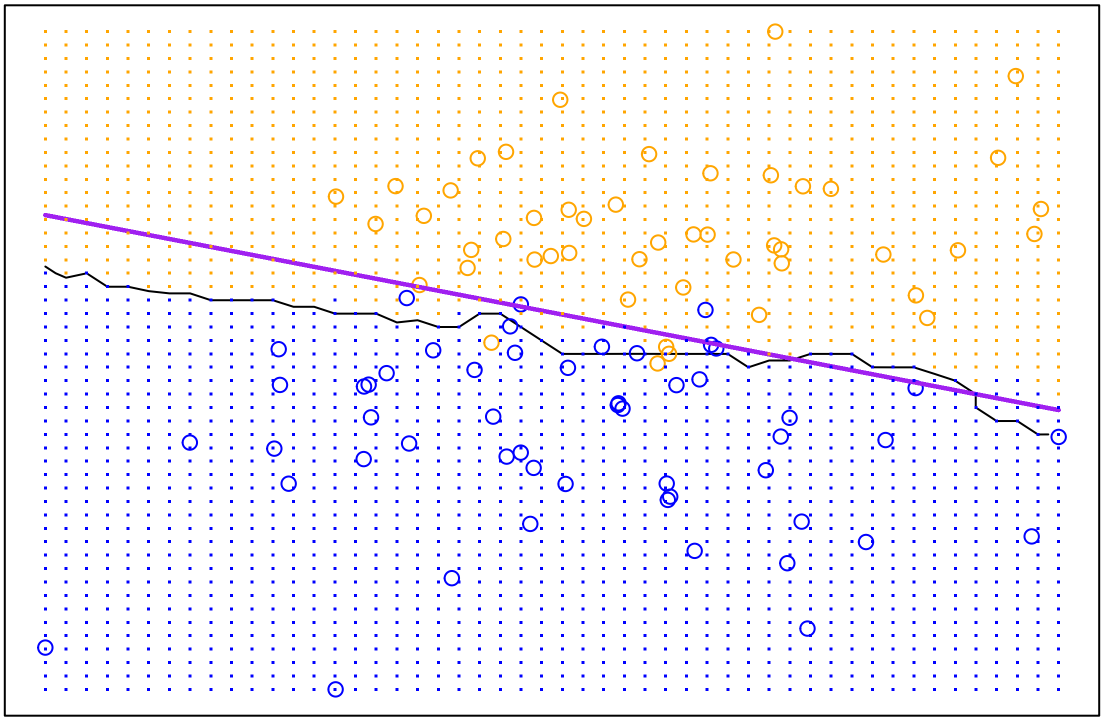
```
# k=3
mod3 <- knn(s.train, s.test,y, k=3, prob=TRUE)
prob3 <- attr(mod3, "prob")
prob3 <- ifelse(mod3=="1", prob3, 1-prob3)
probm3 <- matrix(prob3, g, g)
par(mar=rep(2,4))
contour(x1test, x2test, probm3, levels=0.5, labels="", xlab="X1", ylab="X2", main=
          "3-nearest neighbour", axes=FALSE)
points(train, col=ifelse(y==1, "orange", "blue"))

bayestest= data.frame(x1test,-b1*x1test/b2)
lines(bayestest, type="l", lty=2, lwd=2, col="purple")
```

```
# the baysian decision boundary is the same for train data or test data or a subset of the test data

points(test, pch=".", cex=1.2, col=ifelse(probm3>0.5, "orange", "blue"))
box()
```




```
# k=10
mod10 <- knn(s.train, s.test,y, k=10, prob=TRUE)
prob10 <- attr(mod10, "prob")
prob10 <- ifelse(mod10=="1", prob10, 1-prob10)
probm10 <- matrix(prob10, g, g)
par(mar=rep(2,4))
contour(x1test, x2test, probm10, levels=0.5, labels="", xlab="X1", ylab="X2", main=
          "10-nearest neighbour", axes=FALSE)
points(train, col=ifelse(y==1, "orange", "blue"))
bayestest2= data.frame(test[,1], -b1*test[,1]/b2)
lines(bayestest2, type="l", lty=2, lwd=2, col="purple")
# the same baysian decision boundary, just different ways of drawing

points(test, pch=".", cex=1.2, col=ifelse(probm10>0.5, "orange", "blue"))
box()
```

10−nearest neighbour



```r
# Generate the true classes for test data
z.test = b1*test[,1] + b2*test[,2]
pr.test = 1/(1+exp(-z.test))
y.test = as.factor(rbinom(length(pr.test),1,pr.test))

# Test error rates
table(mod3,y.test)
```

```
##      y.test
## mod3    0    1
##     0 1318   35
##     1  115 1032
```

```r
mean(mod3!=y.test)
```

```
## [1] 0.06
```

```r
table(mod10,y.test)
```

```
##       y.test
## mod10    0    1
##     0 1321   21
##     1  112 1046
```

```r
mean(mod10!=y.test)
```

```
## [1] 0.0532
```

```r
table(mod15,y.test)
```

```
##      y.test
## mod15    0    1
##     0 1310   21
##     1  123 1046
```

```r
mean(mod15!=y.test)
```

```
## [1] 0.0576
```

Question 2

a. Summary of the data

Pattern: Volume tends to increase over years.

```r
library(ISLR2)
View(Weekly)
names(Weekly)
```

```
## [1] "Year"      "Lag1"      "Lag2"      "Lag3"      "Lag4"      "Lag5"
## [7] "Volume"    "Today"     "Direction"
```

```r
dim(Weekly)
```

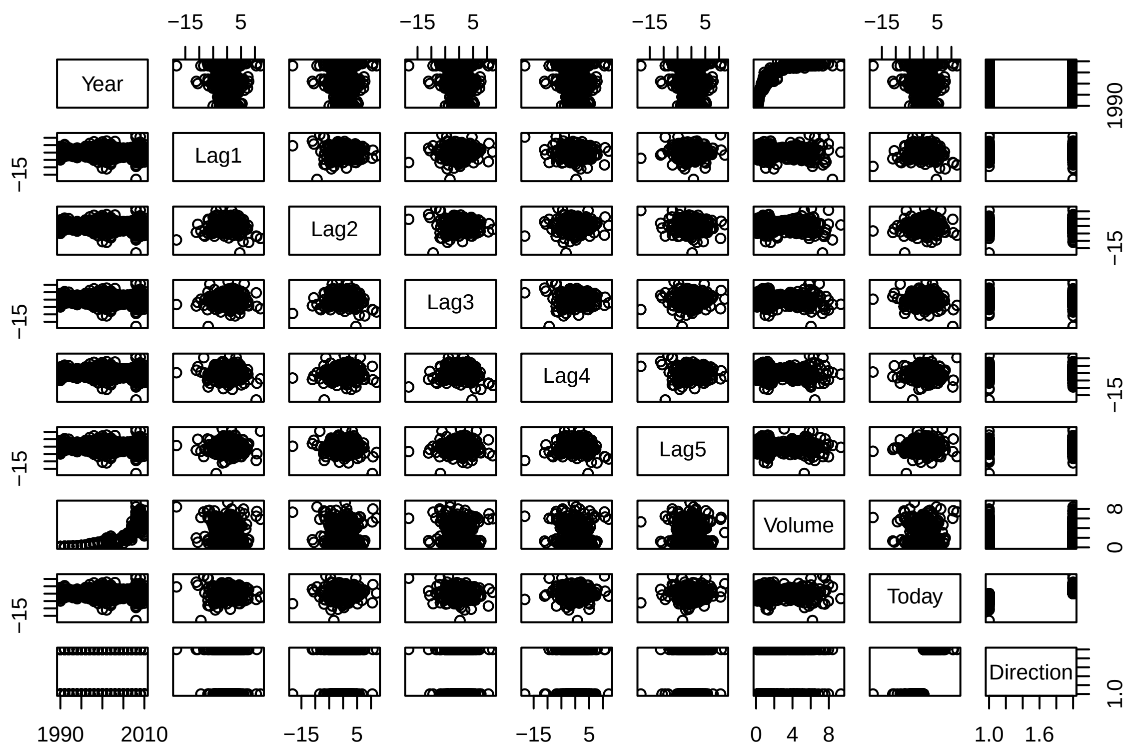
```
## [1] 1089    9
```

```r
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
##
##
##
##
```

```r
pairs(Weekly)
```

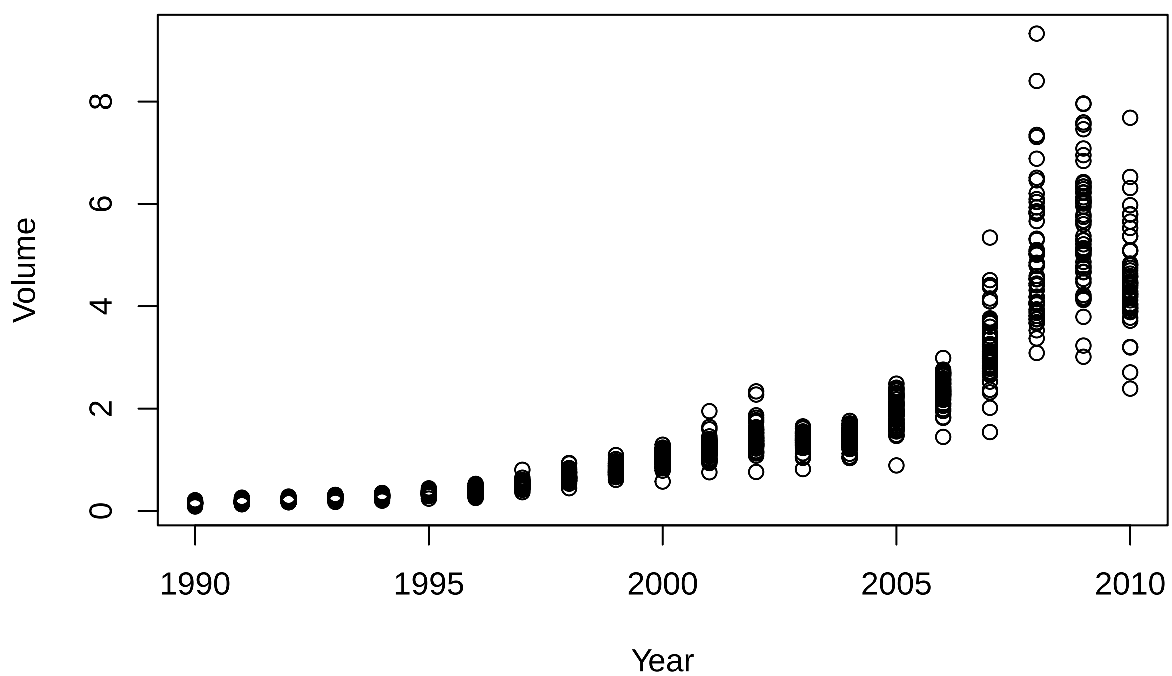



```r
cor(Weekly[, -9]) # the 9th column Direction is not numeric
```

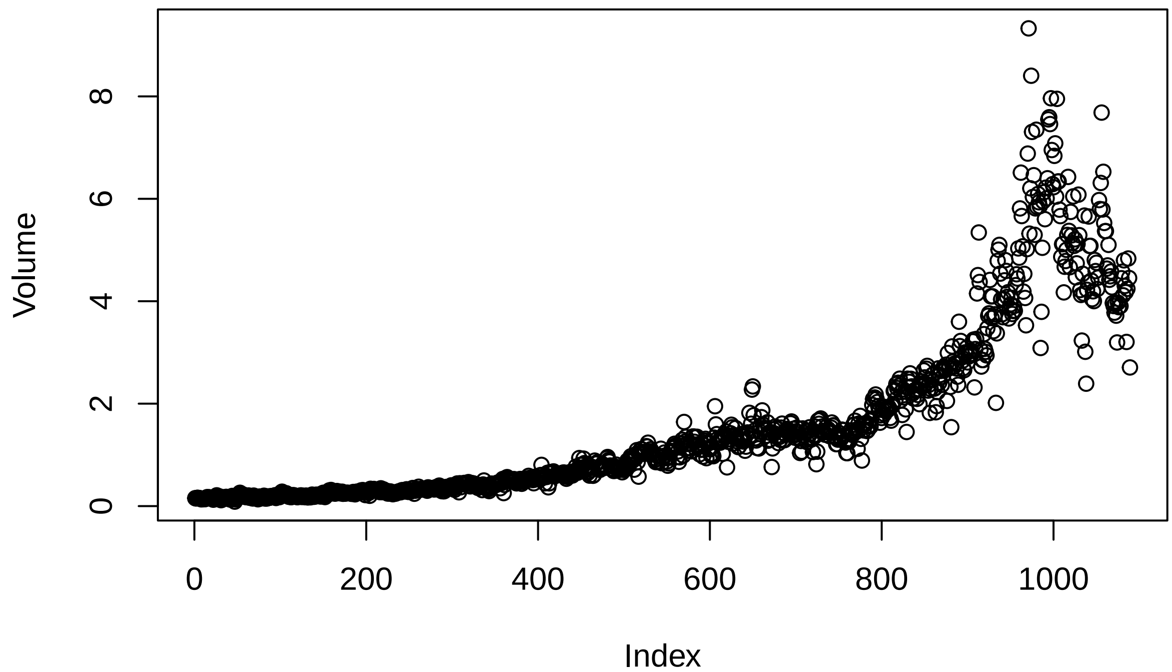
```
##               Year         Lag1        Lag2        Lag3         Lag4
## Year    1.00000000 -0.032289274 -0.03339001 -0.03000649 -0.031127923
## Lag1   -0.03228927  1.000000000 -0.07485305  0.05863568 -0.071273876
## Lag2   -0.03339001 -0.074853051  1.00000000 -0.07572091  0.058381535
## Lag3   -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
```

```
attach(Weekly)
plot(Year, Volume)
```




```
plot(Volume)
```




```r
# If not specifying X argument of plot(), by defult it plots the index of the observation against the V
# The indices of the observation are in chronological order.
```

b. Logistic regressions

Lag 2 appears to have some statistical significance with a Pr(>|z|) = 3%.

```r
attach(Weekly)
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
```

c. Confusion matrix

Overall correction rate: (54+557)/(54+557+48+430) = 56.1%. False positives (Type 1 error): the fraction that market is falsely predicted to be up when the market is actually down: 430/(54+430) = 88.8% False negatives (Type 2 error): the fraction that market is falsely predicted to be down when the market is actually up: 48/(48+557)=7.9%

The test error rate is too high (43.9%). This prediction tends to be overly optimistic (too often predict "Up").

```
glm.probs = predict(glm.fit, type = "response")
glm.pred = rep("Down", length(glm.probs))
glm.pred[glm.probs > 0.5] = "Up"
table(glm.pred, Direction)
```

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

```
mean(glm.pred == Direction)
```

```
## [1] 0.5610652
```

```
sum(glm.pred == "Up" & Direction == "Down") / sum(Direction == "Down")
```

```
## [1] 0.8884298
```

```
sum(glm.pred == "Down" & Direction == "Up") / sum(Direction == "Up")
```

```
## [1] 0.07933884
```

d. Logistic regressions with test sample

Slightly improved.

```
train = (Year < 2009)
Weekly.0910 = Weekly[!train, ]
glm.fit = glm(Direction ~ Lag2, data = Weekly, family = binomial, subset = train)
glm.probs = predict(glm.fit, Weekly.0910, type = "response")
glm.pred = rep("Down", length(glm.probs))
glm.pred[glm.probs > 0.5] = "Up"
Direction.0910 = Direction[!train]
table(glm.pred, Direction.0910)
```

```
##         Direction.0910
## glm.pred Down Up
##     Down    9  5
##     Up     34 56
```

```
# Overall correction rate:
mean(glm.pred == Direction.0910)
```

```
## [1] 0.625
```

```
# False positives
sum(glm.pred == "Up" & Direction.0910 == "Down") / sum(Direction.0910 == "Down")
```

```
## [1] 0.7906977
```

```
# False negatives
sum(glm.pred == "Down" & Direction.0910 == "Up") / sum(Direction.0910 == "Up")
```

```
## [1] 0.08196721
```

e. LDA

Almost the same as logistic regression.

```
library(MASS)
lda.fit = lda(Direction ~ Lag2, data = Weekly, subset = train)
lda.pred = predict(lda.fit, Weekly.0910)
table(lda.pred$class, Direction.0910)
```

```
##       Direction.0910
##        Down Up
##   Down    9  5
##   Up     34 56
```

```
# Overall correction rate:
mean(lda.pred$class == Direction.0910)
```

```
## [1] 0.625
```

```r
# False positives
sum(lda.pred$class == "Up" & Direction.0910 == "Down") / sum(Direction.0910 == "Down")
```

```
## [1] 0.7906977
```

```r
# False negatives
sum(lda.pred$class == "Down" & Direction.0910 == "Up") / sum(Direction.0910 == "Up")
```

```
## [1] 0.08196721
```

f. QDA

Predict "Up" all the time!

```r
qda.fit = qda(Direction ~ Lag2, data = Weekly, subset = train)
qda.class = predict(qda.fit, Weekly.0910)$class
table(qda.class, Direction.0910)
```

```
##          Direction.0910
## qda.class Down Up
##      Down    0  0
##      Up     43 61
```

```r
# Overall correction rate:
mean(qda.class == Direction.0910)
```

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

```r
# False positives
sum(qda.class == "Up" & Direction.0910 == "Down") / sum(Direction.0910 == "Down")
```

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

```r
# False negatives
sum(qda.class == "Down" & Direction.0910 == "Up") / sum(Direction.0910 == "Up")
```

```
## [1] 0
```

g. KNN (K=1)

```r
library(class)
# only one predictor, no need to scale
train.X = as.matrix(Lag2[train])
test.X = as.matrix(Lag2[!train])
train.Direction = Direction[train]
# set.seed(666)
knn.pred = knn(train.X, test.X, train.Direction, k = 1)
table(knn.pred, Direction.0910)
```

```
##         Direction.0910
## knn.pred Down Up
##     Down   21 29
##     Up     22 32
```

```
# Overall correction rate:
mean(knn.pred == Direction.0910)
```

```
## [1] 0.5096154
```

```
# False positives
sum(knn.pred == "Up" & Direction.0910 == "Down") / sum(Direction.0910 == "Down")
```

```
## [1] 0.5116279
```

```
# False negatives
sum(knn.pred == "Down" & Direction.0910 == "Up") / sum(Direction.0910 == "Up")
```

```
## [1] 0.4754098
```

h. Best performance

D) and E): Logistic regression and LDA with only one predictor "Lag2".

i. Experiments with predictors and K

Out of these permutations, the original LDA and logistic regression have better performance in terms of test error rate.

```
# Logistic regression with Lag2 interaction with Lag1
glm.fit = glm(Direction ~ Lag2:Lag1, data = Weekly, family = binomial, subset = train)
glm.probs = predict(glm.fit, Weekly.0910, type = "response")
glm.pred = rep("Down", length(glm.probs))
glm.pred[glm.probs > 0.5] = "Up"
Direction.0910 = Direction[!train]
table(glm.pred, Direction.0910)
```

```
##         Direction.0910
## glm.pred Down Up
##     Down    1  1
##     Up     42 60
```

```
mean(glm.pred == Direction.0910)
```

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

```
sum(glm.pred == "Up" & Direction.0910 == "Down") / sum(Direction.0910 == "Down")
```

```
## [1] 0.9767442
```

```r
sum(glm.pred == "Down" & Direction.0910 == "Up") / sum(Direction.0910 == "Up")
```

```
## [1] 0.01639344
```

```r
# LDA with Lag2 interaction with Lag1
lda.fit = lda(Direction ~ Lag2:Lag1, data = Weekly, subset = train)
lda.pred = predict(lda.fit, Weekly.0910)
table(lda.pred$class, Direction.0910)
```

```
##       Direction.0910
##        Down Up
##   Down    0  1
##   Up     43 60
```

```r
mean(lda.pred$class == Direction.0910)
```

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

```r
sum(lda.pred$class == "Up" & Direction.0910 == "Down") / sum(Direction.0910 == "Down")
```

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

```r
sum(lda.pred$class == "Down" & Direction.0910 == "Up") / sum(Direction.0910 == "Up")
```

```
## [1] 0.01639344
```

```r
# QDA with sqrt(abs(Lag2))
qda.fit = qda(Direction ~ Lag2 + sqrt(abs(Lag2)), data = Weekly, subset = train)
qda.class = predict(qda.fit, Weekly.0910)$class
table(qda.class, Direction.0910)
```

```
##          Direction.0910
## qda.class Down Up
##      Down   12 13
##      Up     31 48
```

```r
mean(qda.class == Direction.0910)
```

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

```r
sum(qda.class == "Up" & Direction.0910 == "Down") / sum(Direction.0910 == "Down")
```

```
## [1] 0.7209302
```

```r
sum(qda.class == "Down" & Direction.0910 == "Up") / sum(Direction.0910 == "Up")
```

```
## [1] 0.2131148
```

```r
# KNN k =10
knn.pred = knn(train.X, test.X, train.Direction, k = 10)
table(knn.pred, Direction.0910)
```

```
##          Direction.0910
## knn.pred Down Up
##     Down   16 19
##     Up     27 42
```

```r
mean(knn.pred == Direction.0910)
```

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

```r
sum(knn.pred == "Up" & Direction.0910 == "Down") / sum(Direction.0910 == "Down")
```

```
## [1] 0.627907
```

```r
sum(knn.pred == "Down" & Direction.0910 == "Up") / sum(Direction.0910 == "Up")
```

```
## [1] 0.3114754
```

```r
# KNN k = 100
knn.pred = knn(train.X, test.X, train.Direction, k = 100)
table(knn.pred, Direction.0910)
```

```
##          Direction.0910
## knn.pred Down Up
##     Down   10 12
##     Up     33 49
```

```r
mean(qda.class == Direction.0910)
```

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

```r
sum(qda.class == "Up" & Direction.0910 == "Down") / sum(Direction.0910 == "Down")
```

```
## [1] 0.7209302
```

```r
sum(qda.class == "Down" & Direction.0910 == "Up") / sum(Direction.0910 == "Up")
```

```
## [1] 0.2131148
```