Lab 4

YI CHEN,yc3356

October 10, 2017

Instructions

Before you leave lab today make sure that you upload a .pdf file to the canvas page (this should have a .pdf extension). This should be the PDF output after you have knitted the file, we don't need the .Rmd file (don't upload the one with the .Rmd extension). The file you upload to the Canvas page should be updated with commands you provide to answer each of the questions below. You can edit this file directly to produce your final solutions. Note, however, in the file you upload you should the above header to have the date, your name, and your UNI. Similarly, when you save the file you should replace **UNI** with your actualy UNI.

Background

Today we'll be using the *Weekly* dataset from the *ISLR* package. This data is similar to the *Smarket* data from class. The dataset contains 1089 weekly returns from the beginning of 1990 to the end of 2010. Make sure that you have the *ISLR* package installed and loaded by running (without the code commented out) the following:

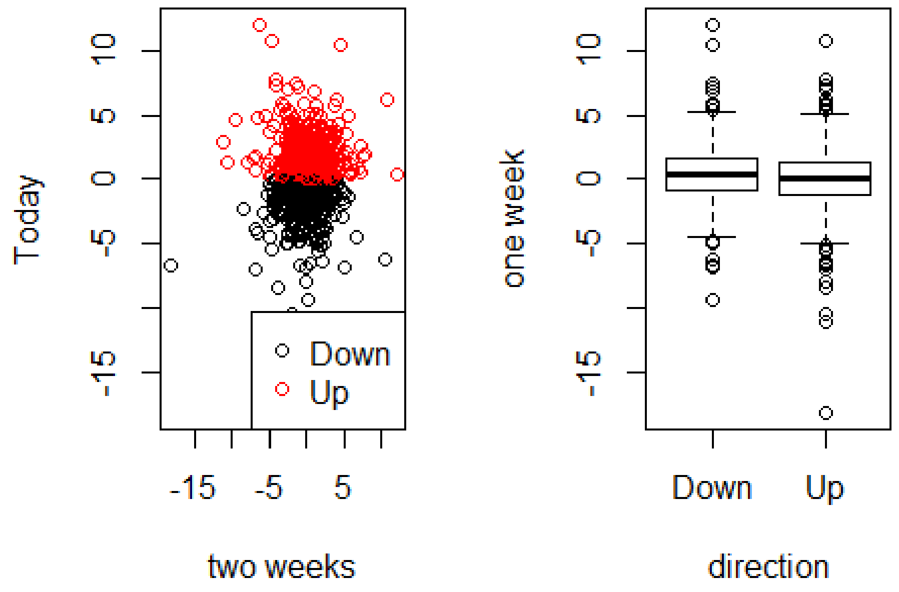
```r
# install.packages("ISLR")
library(ISLR)

## Warning: package 'ISLR' was built under R version 3.4.2
```

We'd like to see if we can accurately predict the direction of a week's return based on the returns over the last five weeks. *Today* gives the percentage return for the week considered and *Year* provides the year that the observation was recorded. *Lag1* - *Lag5* give the percentage return for 1 - 5 weeks previous and *Direction* is a factor variable indicating the direction ('UP' or 'DOWN') of the return for the week considered.

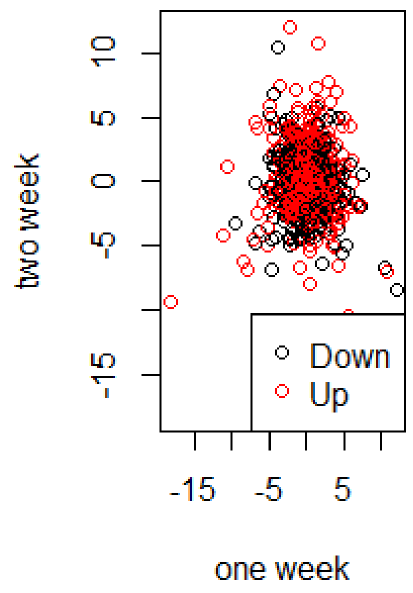
Part 1: Visualizing the relationship between this week's returns and the previous week's returns.

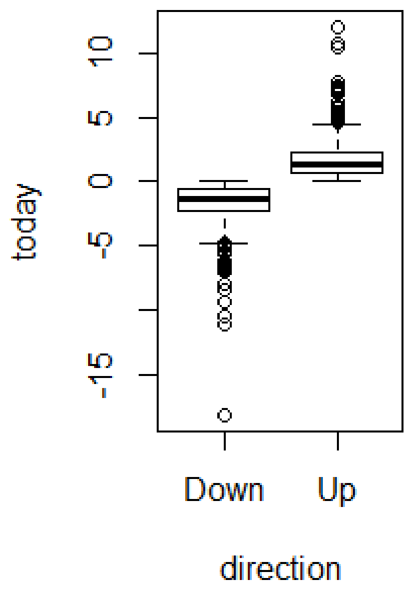
1. Explore the relationship between a week's return and the previous week's return. You should plot more graphs for yourself, but include in the lab write-up a scatterplot of the returns for the weeks considered (*Today*) vs the return from two weeks previous (*Lag2*), and side-by-side boxplots for the lag one week previous (*Lag1*) divided by the direction of this week's return (*Direction*).

```r
par(mfrow=c(1,2))
plot(Weekly$Today ~ Weekly$Lag2,col = Weekly$Direction,xlab='two weeks' ,
ylab='Today')
legend("bottomright", legend =
levels(Weekly$Direction),col=1:length(levels(Weekly$Direction)), pch=1)
boxplot(Weekly$Lag1~Weekly$Direction,ylab='one week',xlab='direction')
```



```r
par(mfrow=c(1,2))
plot(Weekly$Lag1, Weekly$Lag2, col = Weekly$Direction,xlab="one week",
ylab="two week")
legend("bottomright", legend =
levels(Weekly$Direction),col=1:length(levels(Weekly$Direction)), pch=1)
boxplot(Weekly$Today~Weekly$Direction,ylab='today',xlab='direction')
```





Part 2: Building a classifier

Recall the KNN procedure. We classify a new point with the following steps:

-- Calculate the Euclidean distance between the new point and all other points.

-- Create the set $\mathcal{N}_{new}$ containing the K closest points (or, nearest neighbors) to the new point.

-- Determine the number of 'UPs' and 'DOWNs' in $\mathcal{N}_{new}$ and classify the new point according to the most frequent.

2. We'd like to perform KNN on the *Weekly* data, as we did with the *Smarket* data in class. In class we wrote the following function which takes as input a new point $(Lag1_{new}, Lag2_{new})$ and provides the KNN decision using as defaults $K = 5$, Lag1 data given in *Smarket$Lag1*, Lag2 data given in *Smarket$Lag2*, and Direction data given in *Smarket$Direction*. Update the function to calculate the KNN decision for weekly market direction using the *Weekly* dataset with *Lag1* - *Lag5* as predictors. Your function should have only four input values: (1) a new point which should be a vector of length 5, (2) a value for K (with default of 5), (3) the Lag data which should be a data frame with five columns (and n rows) and have as default the *Weekly* data and (4) the Direction data which should be vector of length n and have as default the *Weekly* data directions. Test your function on a point *c(-.5, .5, -.5, -.5, .5)*. The result should be 'UP'.

```
KNNclass <- function( point , K = 5, Lag1 = Weekly$Lag1, Lag2 = Weekly$Lag2,
Lag3 = Weekly$Lag3, Lag4 = Weekly$Lag4, Lag5 = Weekly$Lag5,  Direction =
Weekly$Direction) {
   # calculate the number of data
   n <- length(Lag1)

   stopifnot(length(Lag2) == n,length(Lag3) == n ,length(Lag4) ==
n,length(Lag5) == n)
   stopifnot(length(point) == 5)
   stopifnot(K <= n)

   dists      <- sqrt((Lag1-point[1])^2 + (Lag2-point[2])^2 + (Lag3-
point[3])^2 + (Lag4-point[4])^2 + (Lag5-point[5])^2)
   neighbors  <- order(dists)[1:K]
   neighb.dir <- Direction[neighbors]
   choice     <- names(which.max(table(neighb.dir)))
   return(choice)
}


KNNclass(point = c(-.5, .5, -.5, -.5, .5) )

## [1] "Up"
```

3. Now train your model using data from 1990 - 2008 and use the data from 2009-2010 as test data. To do this, divide the data into two data frames, *test* and *train*. Then write a loop that iterates over the test points in the test dataset calculating a prediction for each based on the training data with $K = 5$. Save these predictions in a vector. Finally, calculate your test error, which you should store as a variable named *test.error*. The test error calculates the proportion of your predictions which are incorrect (don't match the actual directions). HINT: Since the test data is stored in a dataframe, you may need to use an `as.numeric()` call when you pass it in as your new point. Or maybe not, depending on how you wrote your `KNNclass` function.

```
test <- Weekly[Weekly$Year == 2008, ]
train <- Weekly[Weekly$Year != 2008, ]
n.test <- nrow(test)
predictions <- rep(NA, n.test)

test_data <- test[2:6]

for (i in 1:n.test){
        point <- as.numeric(test_data[i,])
        predictions[i] <- KNNclass(point , K = 5, Lag1 = train$Lag1, Lag2 =
train$Lag2, Lag3 = train$Lag3, Lag4 = train$Lag4, Lag5 = train$Lag5)
}
test.error <- mean(predictions != test$Direction)
test.error

## [1] 0.5384615
```

4. Do the same thing as in question 3, but instead use $K = 3$. Which has a lower test error?

```
test <- Weekly[Weekly$Year == 2008, ]
train <- Weekly[Weekly$Year != 2008, ]
n.test <- nrow(test)
predictions <- rep(NA, n.test)

test_data <- test[2:6]

for (i in 1:n.test){
        point <- as.numeric(test_data[i,])
        predictions[i] <- KNNclass(point , K = 3, Lag1 = train$Lag1, Lag2 =
train$Lag2, Lag3 = train$Lag3, Lag4 = train$Lag4, Lag5 = train$Lag5)
}
test.error <- mean(predictions != test$Direction)
test.error

## [1] 0.4807692
```