

Lab 4

Chang Qu cq2203

October 23, 2018

Instructions

Make sure that you upload a knitted pdf or html file to the canvas page (this should have a .pdf or .html extension). Also upload the .Rmd file. Include output for each question in its own individual code chunk and don't print out any vector that has more than 20 elements.

Objectives: KNN Classification and Cross-Validation

Background

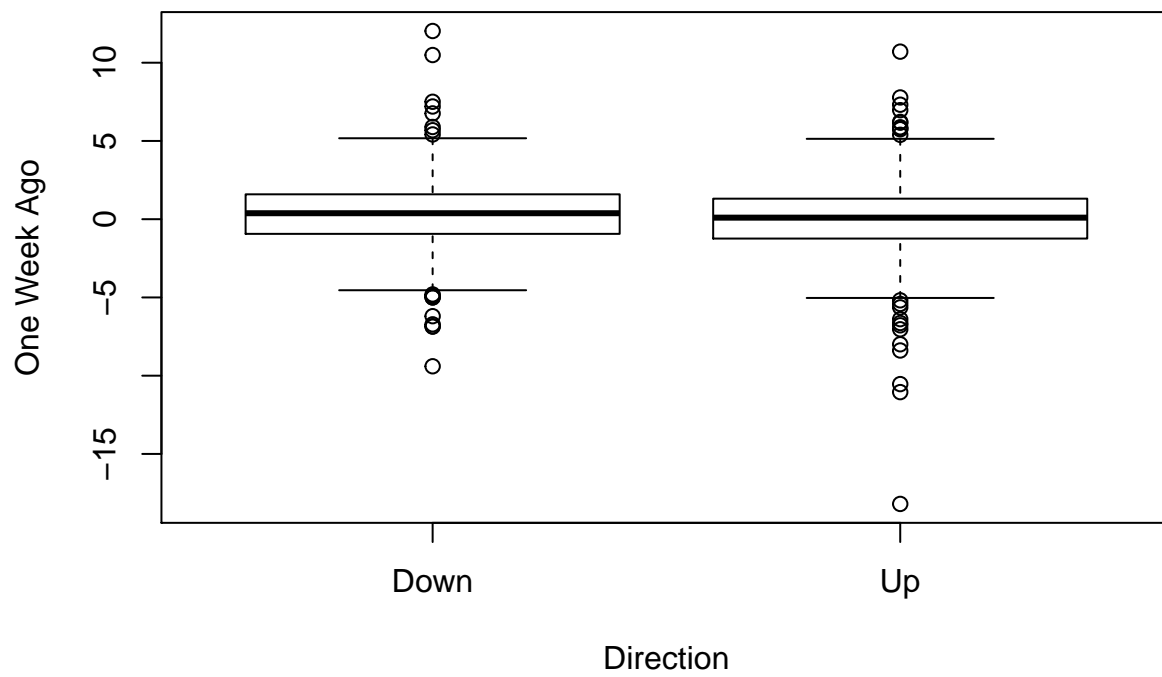
Today we'll be using the *Weekly* dataset from the *ISLR* package. This data is similar to the *Smarket* data from class. The dataset contains 1089 weekly returns from the beginning of 1990 to the end of 2010. Make sure that you have the *ISLR* package installed and loaded by running (without the code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

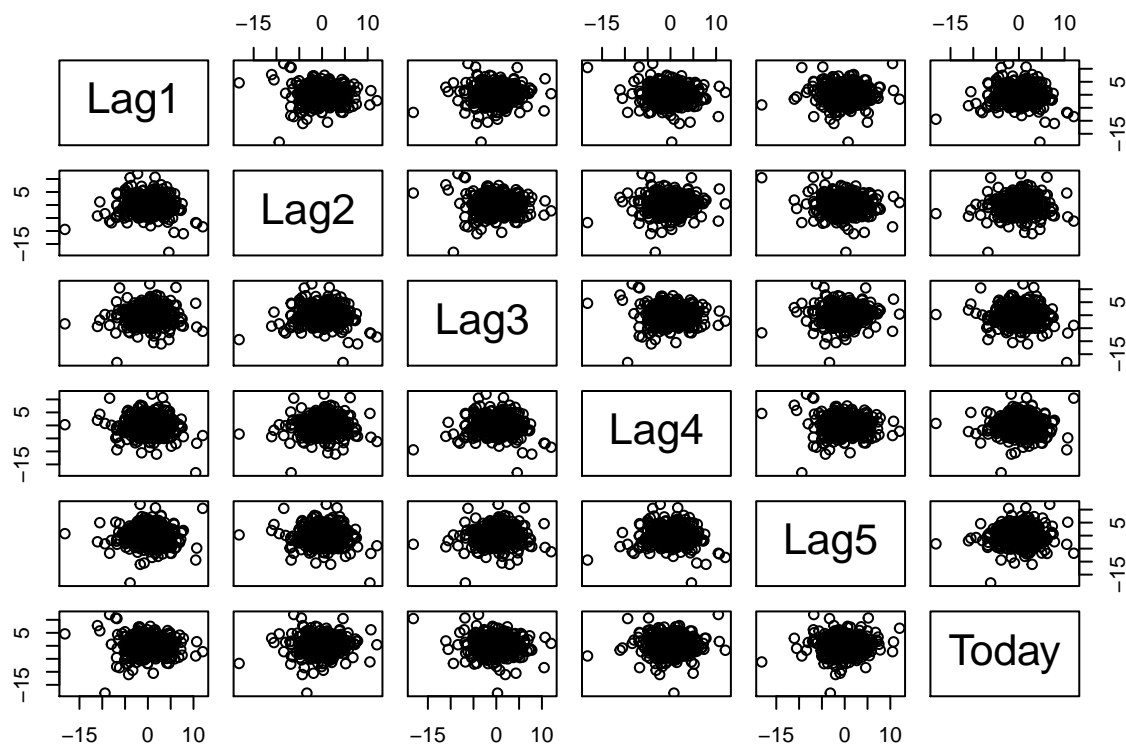
```
# install.packages("ISLR")  
library(ISLR)
```

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.

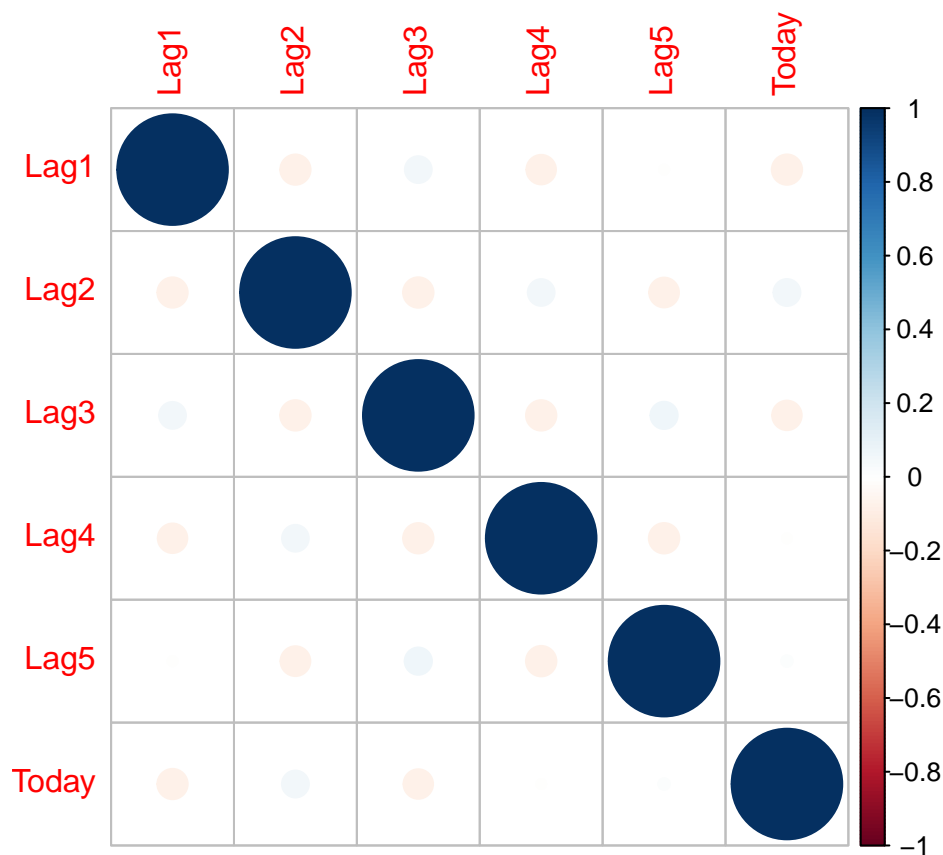
Part 1: Visualizing the relationship between this week's returns and the previous week's returns.

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*).





corplot 0.84 loaded



From the plot above, we see that there is no obvious relationship between a week's return and a previous

week's return. The scatterplot matrix shows that most of them have a slightly negative linear relationship with one another.

There is no strong correlation from the correlation plot.

When we plot this week's return against returns two weeks ago, we see a slightly negative linear relationship.

When looking at the boxplot, there is no obvious difference of the returns one week ago separated by direction. They have similar median and IQR. However, the Up group has a smaller minimum than the Down group.

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* $Lag1$, and $Lag2$ data given in *Smarket* $Lag2$. 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 three input values: (1) a new point which should be a vector of length 5, (2) a value for K , and (3) the Lag data which should be a data frame with five columns (and n rows).

```
KNN.decision <- function(test.lag, K = 5, Lag.data = data.frame(Weekly$Lag1,Weekly$Lag2,Weekly$Lag3,Weekly$Lag4,Weekly$Lag5)) {
  n <- nrow(Lag.data)
  stopifnot(length(test.lag) == 5, K <= n)

  dists      <- sqrt((test.lag[1]-Lag.data[[1]])^2 +
                    (test.lag[2]-Lag.data[[2]])^2 +
                    (test.lag[3]-Lag.data[[3]])^2 +
                    (test.lag[4]-Lag.data[[4]])^2 +
                    (test.lag[5]-Lag.data[[5]])^2)

  ord.indices <- order(dists)[1:K]
  neighb.dir  <- Lag.data[[6]][ord.indices]
  choice      <- names(which.max(table(neighb.dir)))
  return(choice)
}

KNN.decision(test.lag=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).

```

# Define test and training data
test <- Weekly[(Weekly$Year == 2009) | (Weekly$Year == 2010), ]
train <- Weekly[(Weekly$Year != 2009) & (Weekly$Year != 2010), ]

n.test <- nrow(test)
predictions <- rep(NA, nrow(test))

# Input for second argument
Lag.data <- data.frame(train$Lag1, train$Lag2, train$Lag3, train$Lag4, train$Lag5, train$Direction)

for (i in 1:n.test){

  # Input for first argument (needs to iterate)
  test.lag <- c(test$Lag1[i], test$Lag2[i], test$Lag3[i], test$Lag4[i], test$Lag5[i])

  # Predictions
  predictions[i] <- KNN.decision(test.lag=test.lag,
                                K = 5,
                                Lag.data = Lag.data)
}

test.error <- sum(predictions != test$Direction)/n.test
test.error

## [1] 0.4519231

```

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

```

predictions_k3 <- rep(NA, nrow(test))
for (i in 1:n.test){

  # Input for first argument (needs to iterate)
  test.lag <- c(test$Lag1[i], test$Lag2[i], test$Lag3[i], test$Lag4[i], test$Lag5[i])

  # Predictions
  predictions_k3[i] <- KNN.decision(test.lag=test.lag,
                                    K = 3,
                                    Lag.data = Lag.data)
}

test.error <- sum(predictions_k3 != test$Direction)/n.test
test.error

## [1] 0.4423077

```

$K = 3$ has a lower test error.

Part 3: Cross-validation

Ideally we'd like to use our model to predict future returns, but how do we know which value of K to choose? We could choose the best value of K by training with data from 1990 - 2008, testing with the 2009 - 2010 data, and selecting the model with the lowest test error as in the previous section. However, in order to build

the best model, we'd like to use ALL the data we have to train the model. In this case, we could use all of the *Weekly* data and choose the best model by comparing the training error, but unfortunately this isn't usually a good predictor of the test error.

In this section, we instead consider a class of methods that estimate the test error rate by holding out a (random) subset of the data to use as a test set, which is called *k*-fold cross validation. (Note this lower case *k* is different than the upper case *K* in KNN. They have nothing to do with each other, it just happens that the standard is to use the same letter in both.) This approach involves randomly dividing the set of observations into *k* groups, or folds, of equal size. The first fold is treated as a test set, and the model is fit on the remaining *k* - 1 folds. The error rate, ERR1, is then computed on the observations in the held-out fold. This procedure is repeated *k* times; each time, a different group of observations is treated as a test set. This process results in *k* estimates of the test error: ERR1, ERR2, ..., ERR*k*. The *k*-fold CV estimate of the test error is computed by averaging these values,

$$CV_{(k)} = \frac{1}{k} \sum_{i=1}^k ERR_k.$$

We'll run a 9-fold cross-validation in the following. Note that we have 1089 rows in the dataset, so each fold will have exactly 121 members.

5. Create a vector *fold* which has *n* elements, where *n* is the number of rows in *Weekly*. We'd like for the *fold* vector to take values in 1-9 which assign each corresponding row of the *Weekly* dataset to a fold. Do this in two steps: (1) create a vector using *rep()* with the values 1-9 each repeated 121 times (note $1089 = 121 \cdot 9$), and (2) use *sample()* to randomly reorder the vector you created in (1).

```
rep.vec <- rep(1:9,121)
head(rep.vec,36)
```

```
## [1] 1 2 3 4 5 6 7 8 9 1 2 3 4 5 6 7 8 9 1 2 3 4 5 6 7 8 9 1 2 3 4 5 6 7 8
## [36] 9
```

```
re.order <- sample(rep.vec,replace=F)
head(re.order,36)
```

```
## [1] 2 9 2 6 3 2 7 4 6 4 7 2 2 9 9 4 6 2 3 4 1 2 3 8 6 6 6 2 5 1 7 5 9 8 9
## [36] 3
```

6. Iterate over the 9 folds, treating a different fold as the test set and all others the training set in each iteration. Using a KNN classifier with $K = 5$ calculate the test error for each fold. Then calculate the cross-validation approximation to the test error which is the average of ERR1, ERR2, ..., ERR9.

```
test.error <- NULL
for (k in 1:9) {

  # Choose test and training data from kth fold
  test <- Weekly[re.order==k, ]
  train <- Weekly[re.order!=k, ]

  n.test <- nrow(test)
  predictions <- rep(NA, n.test)

  # Input for second argument
  Lag.data <- data.frame(train$Lag1,train$Lag2,train$Lag3,train$Lag4,train$Lag5,train$Direction)

  for (i in 1:n.test){

    # Input for first argument
```

```

    test.lag <- c(test$Lag1[i],test$Lag2[i],test$Lag3[i],test$Lag4[i],test$Lag5[i])

    predictions[i] <- KNN.decision(test.lag=test.lag,
                                   K = 5,
                                   Lag.data = Lag.data)
  }

  test.error[k] <- sum(predictions != test$Direction)/n.test
}

mean(test.error)

```

```
## [1] 0.4784206
```

7. Repeat step (6) for $K = 1$, $K = 3$, and $K = 7$. For which set is the cross-validation approximation to the test error the lowest?

```

# K=1

test.error <- NULL
for (k in 1:9) {

  # Choose test and training data from kth fold
  test <- Weekly[re.order==k, ]
  train <- Weekly[re.order!=k, ]

  n.test <- nrow(test)
  predictions <- rep(NA, n.test)

  # Input for second argument
  Lag.data <- data.frame(train$Lag1,train$Lag2,train$Lag3,train$Lag4,train$Lag5,train$Direction)

  for (i in 1:n.test){

    # Input for first argument
    test.lag <- c(test$Lag1[i],test$Lag2[i],test$Lag3[i],test$Lag4[i],test$Lag5[i])

    predictions[i] <- KNN.decision(test.lag=test.lag,
                                    K = 1,
                                    Lag.data = Lag.data)
  }

  test.error[k] <- sum(predictions != test$Direction)/n.test
}

mean(test.error)

```

```
## [1] 0.4949495
```

```
# K=3
```

```

test.error <- NULL
for (k in 1:9) {

```

```

# Choose test and training data from kth fold
test <- Weekly[re.order==k, ]
train <- Weekly[re.order!=k, ]

n.test <- nrow(test)
predictions <- rep(NA, n.test)

# Input for second argument
Lag.data <- data.frame(train$Lag1,train$Lag2,train$Lag3,train$Lag4,train$Lag5,train$Direction)

for (i in 1:n.test){

  # Input for first argument
  test.lag <- c(test$Lag1[i],test$Lag2[i],test$Lag3[i],test$Lag4[i],test$Lag5[i])

  predictions[i] <- KNN.decision(test.lag=test.lag,
                                K = 3,
                                Lag.data = Lag.data)
}

test.error[k] <- sum(predictions != test$Direction)/n.test
}

mean(test.error)

```

```
## [1] 0.5022957
```

```
# K=7
```

```

test.error <- NULL
for (k in 1:9) {

  # Choose test and training data from kth fold
  test <- Weekly[re.order==k, ]
  train <- Weekly[re.order!=k, ]

  n.test <- nrow(test)
  predictions <- rep(NA, n.test)

  # Input for second argument
  Lag.data <- data.frame(train$Lag1,train$Lag2,train$Lag3,train$Lag4,train$Lag5,train$Direction)

  for (i in 1:n.test){

    # Input for first argument
    test.lag <- c(test$Lag1[i],test$Lag2[i],test$Lag3[i],test$Lag4[i],test$Lag5[i])

    predictions[i] <- KNN.decision(test.lag=test.lag,
                                    K = 7,
                                    Lag.data = Lag.data)
  }

  test.error[k] <- sum(predictions != test$Direction)/n.test
}

```



```
}
```

```
mean(test.error)
```

```
## [1] 0.4830119
```

$k = 5$ has the lowest test error