DA4

2025-06-22

ISLR Ch 4 Lab 4.6.1 The Stock Market Data

```r
library(ISLR)
library(ggplot2)
?Smarket
names(Smarket)
```

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

```r
model1 = Smarket
dim(Smarket)
```

```
## [1] 1250    9
```

this data set has 1250 rows and 9 columns

```r
summary(Smarket)
```

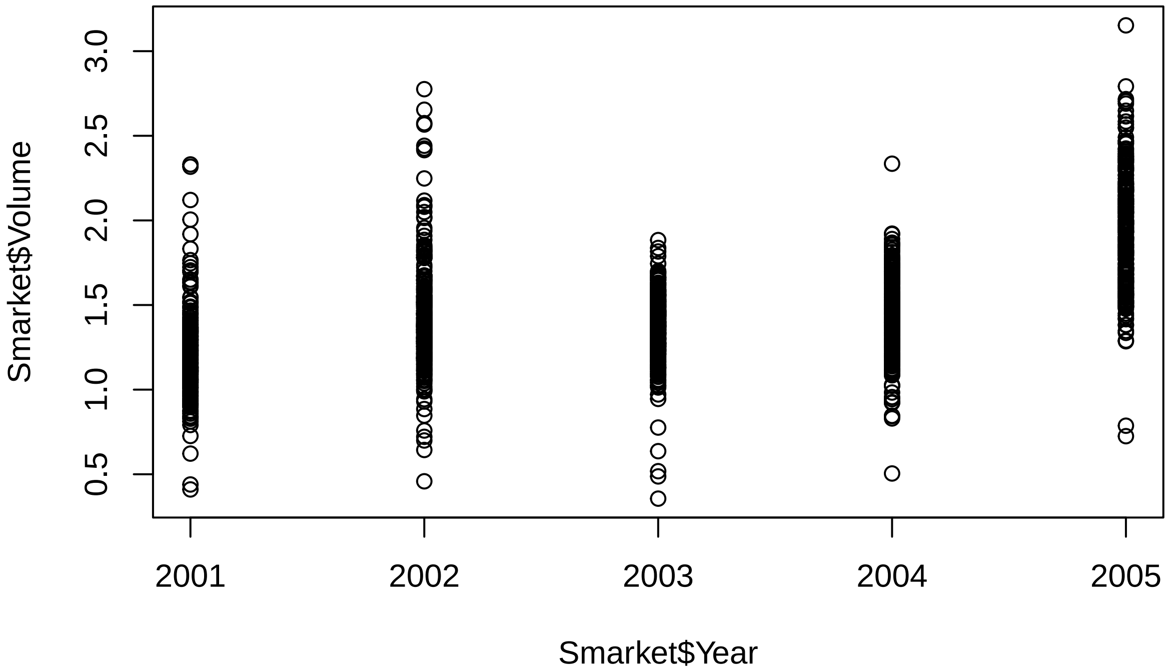
```
##       Year           Lag1                Lag2                Lag3          
##  Min.   :2001   Min.   :-4.922000   Min.   :-4.922000   Min.   :-4.922000  
##  1st Qu.:2002   1st Qu.:-0.639500   1st Qu.:-0.639500   1st Qu.:-0.640000  
##  Median :2003   Median : 0.039000   Median : 0.039000   Median : 0.038500  
##  Mean   :2003   Mean   : 0.003834   Mean   : 0.003919   Mean   : 0.001716  
##  3rd Qu.:2004   3rd Qu.: 0.596750   3rd Qu.: 0.596750   3rd Qu.: 0.596750  
##  Max.   :2005   Max.   : 5.733000   Max.   : 5.733000   Max.   : 5.733000  
##       Lag4                Lag5              Volume           Today          
##  Min.   :-4.922000   Min.   :-4.92200   Min.   :0.3561   Min.   :-4.922000  
##  1st Qu.:-0.640000   1st Qu.:-0.64000   1st Qu.:1.2574   1st Qu.:-0.639500  
##  Median : 0.038500   Median : 0.03850   Median :1.4229   Median : 0.038500  
##  Mean   : 0.001636   Mean   : 0.00561   Mean   :1.4783   Mean   : 0.003138  
##  3rd Qu.: 0.596750   3rd Qu.: 0.59700   3rd Qu.:1.6417   3rd Qu.: 0.596750  
##  Max.   : 5.733000   Max.   : 5.73300   Max.   :3.1525   Max.   : 5.733000  
##  Direction 
##  Down:602  
##  Up  :648  
## 
## 
## 
## 
```

```r
cor(Smarket[,-9])
```

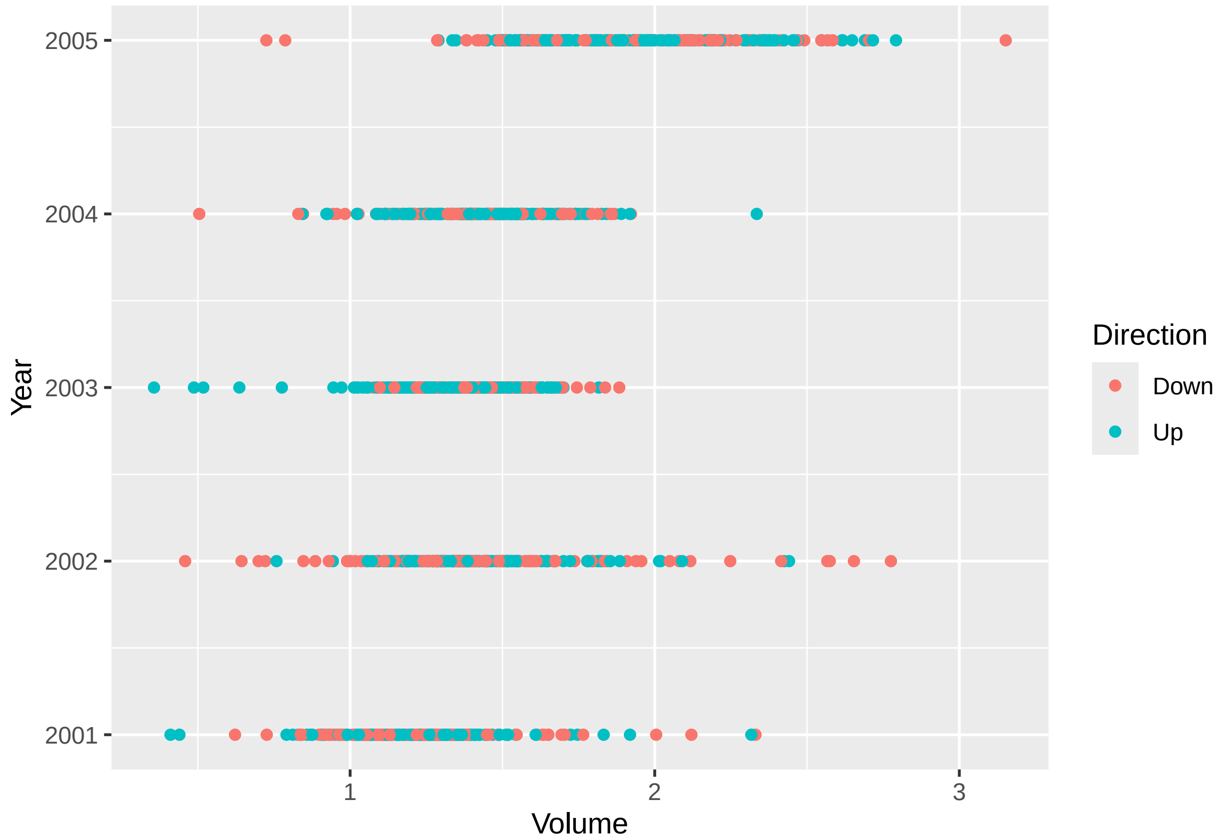
```
##              Year         Lag1         Lag2         Lag3         Lag4
## Year   1.00000000  0.029699649  0.030596422  0.033194581  0.035688718
## Lag1   0.02969965  1.000000000 -0.026294328 -0.010803402 -0.002985911
## Lag2   0.03059642 -0.026294328  1.000000000 -0.025896670 -0.010853533
## Lag3   0.03319458 -0.010803402 -0.025896670  1.000000000 -0.024051036
## Lag4   0.03568872 -0.002985911 -0.010853533 -0.024051036  1.000000000
## Lag5   0.02978799 -0.005674606 -0.003557949 -0.018808338 -0.027083641
## Volume 0.53900647  0.040909908 -0.043383215 -0.041823686 -0.048414246
## Today  0.03009523 -0.026155045 -0.010250033 -0.002447647 -0.006899527
##                Lag5      Volume        Today
## Year    0.029787995  0.53900647  0.030095229
## Lag1   -0.005674606  0.04090991 -0.026155045
## Lag2   -0.003557949 -0.04338321 -0.010250033
## Lag3   -0.018808338 -0.04182369 -0.002447647
## Lag4   -0.027083641 -0.04841425 -0.006899527
## Lag5    1.000000000 -0.02200231 -0.034860083
## Volume -0.022002315  1.00000000  0.014591823
## Today  -0.034860083  0.01459182  1.000000000
```

we make -9 because the direction column consists text. The correlations between today's returns and previous days' returns (lag1-5) are really close to zero, so there's no relationship between previous returns and today's. The only significant correlation is between volume and year (0.539), so let's plot it

```r
plot(Smarket$Year, Smarket$Volume)
```




```r
ggplot(data = model1) +
  geom_point(mapping = aes(x = Volume, y = Year, color = Direction))
```



through this graphs we can see that volume is increasing as time passes by

4.6.2 Logistic Regression

```
attach(Smarket)
glm.fit = glm(Direction ~ Lag1 + Lag2 + Lag3 + Lag4 + Lag5 + Volume, data = Smarket, family=binomial)
summary(glm.fit)
```

```
##
## Call:
## glm(formula = Direction ~ Lag1 + Lag2 + Lag3 + Lag4 + Lag5 +
##     Volume, family = binomial, data = Smarket)
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept) -0.126000   0.240736  -0.523    0.601
## Lag1        -0.073074   0.050167  -1.457    0.145
## Lag2        -0.042301   0.050086  -0.845    0.398
## Lag3         0.011085   0.049939   0.222    0.824
## Lag4         0.009359   0.049974   0.187    0.851
## Lag5         0.010313   0.049511   0.208    0.835
## Volume       0.135441   0.158360   0.855    0.392
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1731.2  on 1249  degrees of freedom
## Residual deviance: 1727.6  on 1243  degrees of freedom
```

```
## AIC: 1741.6
##
## Number of Fisher Scoring iterations: 3
```

the lowest p value is for lag1 (0.145), but it is still >0.05, so we fail to reject the null hypothesis. We can't claim that there is a real relationship

```r
coef(glm.fit)
```

```
##  (Intercept)         Lag1         Lag2         Lag3         Lag4         Lag5
## -0.126000257 -0.073073746 -0.042301344  0.011085108  0.009358938  0.010313068
##       Volume
##  0.135440659
```

if we have an equation like y=a + b1x1 + b2x2 + … those coefficients are actually those b1, b2 values. Intercept is the a. But in order to see whether we can trust that, we need to have p value. here y=1 corresponds to "Up" and Y=0 to "Down"

```r
summary (glm.fit)$coef
```

```
##                 Estimate Std. Error    z value  Pr(>|z|)
## (Intercept) -0.126000257 0.24073574 -0.5233966 0.6006983
## Lag1        -0.073073746 0.05016739 -1.4565986 0.1452272
## Lag2        -0.042301344 0.05008605 -0.8445733 0.3983491
## Lag3         0.011085108 0.04993854  0.2219750 0.8243333
## Lag4         0.009358938 0.04997413  0.1872757 0.8514445
## Lag5         0.010313068 0.04951146  0.2082966 0.8349974
## Volume       0.135440659 0.15835970  0.8552723 0.3924004
```

none of the variables' p values are small enough to say that these coefficients are accurate.

```r
probs = predict(glm.fit, type="response")
probs[1:15]
```

```
##         1         2         3         4         5         6         7         8
## 0.5070841 0.4814679 0.4811388 0.5152224 0.5107812 0.5069565 0.4926509 0.5092292
##         9        10        11        12        13        14        15
## 0.5176135 0.4888378 0.4965211 0.5197834 0.5183031 0.4963852 0.4864892
```

```r
contrasts(Direction)
```

```
##      Up
## Down  0
## Up    1
```

this predicts the possibility that the market will go "up". Because it's type is "response", R uses the logistic regression. By default it would use linear one. Here most of them will go UP, because R created a dummy variable which is 1 at UP

```
glm.pred = rep("Down", 1250)
glm.pred[probs >.5]="Up"
table(glm.pred, Direction)
```

```
##         Direction
## glm.pred Down  Up
##     Down  145 141
##     Up    457 507
```

here we made 1250 values of "Down" as a default. Then to those whose probs are >0.5 we gade "Up". Then we constructed table which serves as a confusion matrix. Those TP and TN values are 507+145= 652.

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

```
## [1] 0.5216
```

this shows for how many of days the model predicted direction correctly. For 52% of days. It might seem +- good, but actually this is the training set error. In order to actually measure the effectiveness of log regression model we need to separate data set into training and test sets.

We will create training set from 2001-2004 years and testing will be 2005.

```
train = Smarket[Year != 2005,]
test = Smarket[Year == 2005,]
dim(test)
```

```
## [1] 252   9
```

```
directions = Smarket[Year == 2005, "Direction"]
```

in directions the directions of actual test data set are stored, so we could compare them later.

```
fit2 = glm(Direction ~ Lag1 + Lag2 + Lag3 + Lag4 + Lag5 + Volume, data = train, family = binomial)
fit2.probs = predict(fit2, test, type="response")
fit2.pred = rep("Down", 252)
fit2.pred[fit2.probs > 0.5] = "Up"

table(fit2.pred, directions)
```

```
##          directions
## fit2.pred Down Up
##      Down   77 97
##      Up     34 44
```

```
mean(fit2.pred == directions)
```

```
## [1] 0.4801587
```

from this we can infer that 77+44 were identified right, but it is actually only 48% of time to be correct. It is worse than random guessing

```r
glm.fits=glm(Direction ~ Lag1+Lag2, data=Smarket, family=binomial)
glm.probs=predict (glm.fits, test, type="response")
glm.pred=rep("Down", 252)
glm.pred[glm.probs >.5]="Up"
table(glm.pred, directions)
```

```
##         directions
## glm.pred Down  Up
##     Down    9   9
##     Up    102 132
```

```r
mean(glm.pred== directions)
```

```
## [1] 0.5595238
```

We can say that when less predictors are used, the predicting algorithm is more effective. Probably because there is no correlation between today's return and previous days, they create a lot of unnecessary "noise" for the model.

```r
predict(glm.fits, newdata = data.frame(Lag1=c(1.2 ,1.5), Lag2=c(1.1,-0.8)),type="response")
```

```
##         1         2
## 0.4848777 0.5006454
```

we want to make new data frame if we want to predict for specific values

4.6.5 K-Nearest Neighbors

```r
library(class)
train.X = cbind(Lag1, Lag2)[Year != 2005,]
test.X= cbind(Lag1, Lag2)[Year == 2005,]
train.Direction = Direction[Year != 2005]
```

we need a train matrix and a test matrix and then the directions for train

```r
set.seed(1)
knn.pred = knn(train.X, test.X, train.Direction, k=1)

table(knn.pred, directions)
```

```
##         directions
## knn.pred Down Up
##     Down   43 58
##     Up     68 83
```

```r
mean(knn.pred == directions)
```

```
## [1] 0.5
```

(83+43) is only a half of the cases that were predicted correctly. We should change the k value.

```r
set.seed(1)
knn.pred = knn(train.X, test.X, train.Direction, k=2)

table(knn.pred, directions)
```

```
##          directions
## knn.pred Down Up
##     Down   43 59
##     Up     68 82
```

```r
mean(knn.pred == directions)
```

```
## [1] 0.4960317
```

2 is worse

```r
set.seed(1)
knn.pred = knn(train.X, test.X, train.Direction, k=3)

table(knn.pred, directions)
```

```
##          directions
## knn.pred Down Up
##     Down   48 55
##     Up     63 86
```

```r
mean(knn.pred == directions)
```

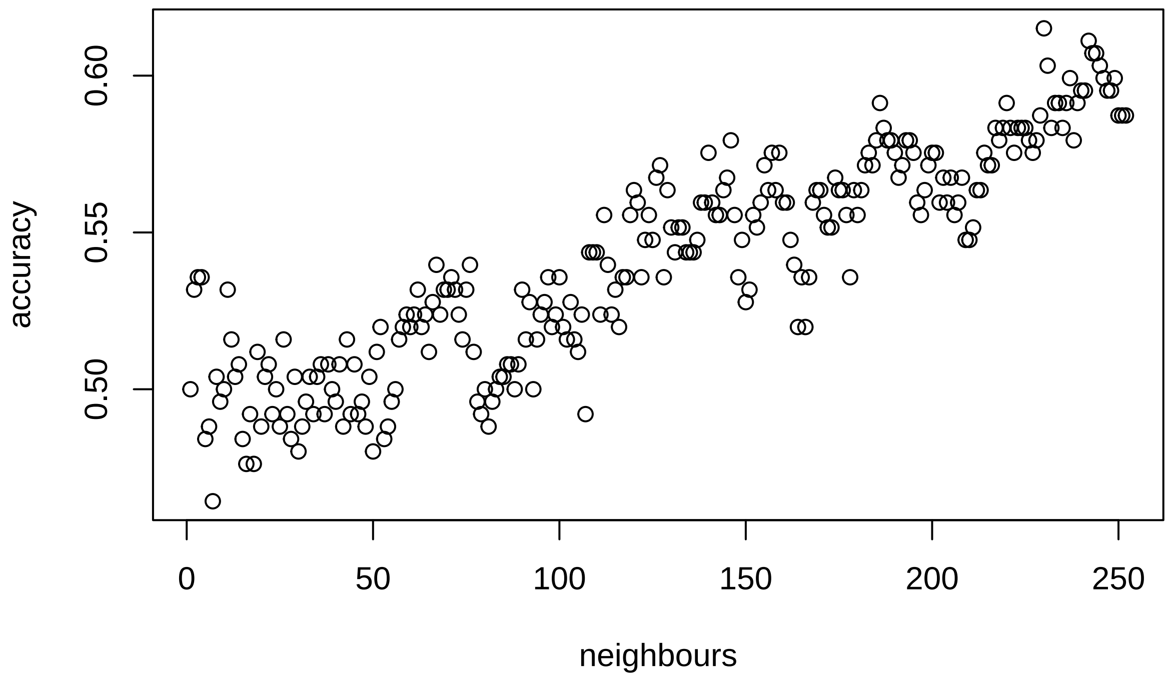
```
## [1] 0.531746
```

when k is 3 it is better, 53% of time it is right

```r
neighbours = c()
accuracy = c()

for (i in 1:252) {
  knn.pred = knn(train.X, test.X, train.Direction, k=i)
  accuracy[i] = mean(knn.pred == directions)
  neighbours[i] = i
}

plot(neighbours, accuracy)
```



we can see that actually the best accuracy is given by 200-250 neighbours