CWRU DSCI351-451: ISLR4 Classification

Roger H. French, JiQi Liu

15 November, 2018

Contents

12.2.1.1 Chapter 4 Lab: Logistic Regression, LDA, QDA, and KNN

We'll be using the glm package for "Fitting Generalized Linear Models"

- glm is used to fit generalized linear models,
  - specified by giving a symbolic description of the linear predictor and
  - a description of the error distribution.

12.2.1.2 The Stock Market Data

Load the stock market data, thats in the ISLR package, and get familiar with it.

```
library(ISLR)
names(Smarket)
```

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

How big is it?

```
dim(Smarket)
```

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

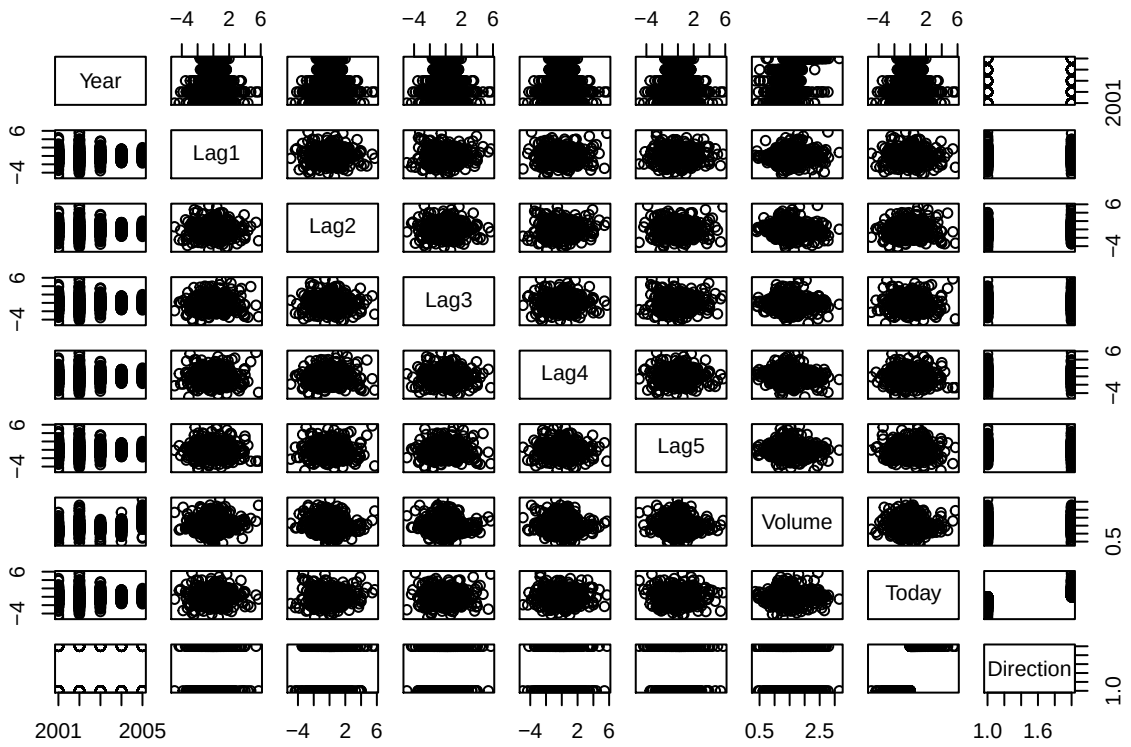
Get a descriptive statistics summary

```
summary(Smarket)
```

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

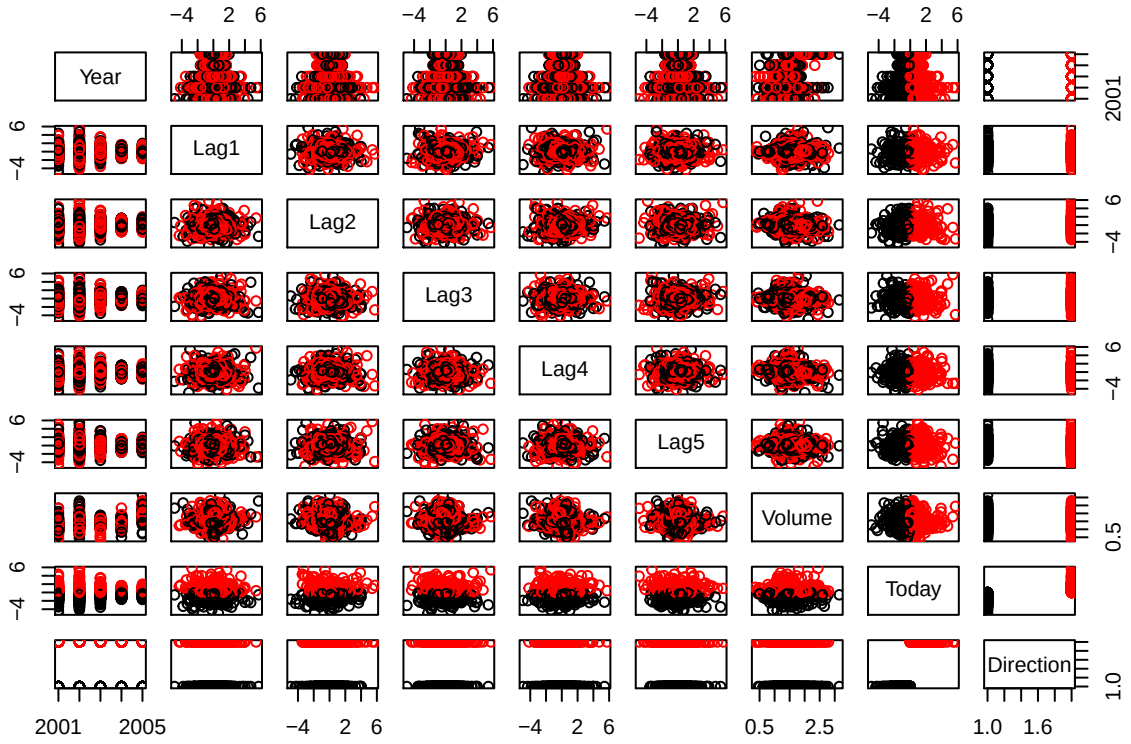
Explore the dataset with a pair-wise correlation plot

```
pairs(Smarket)
```



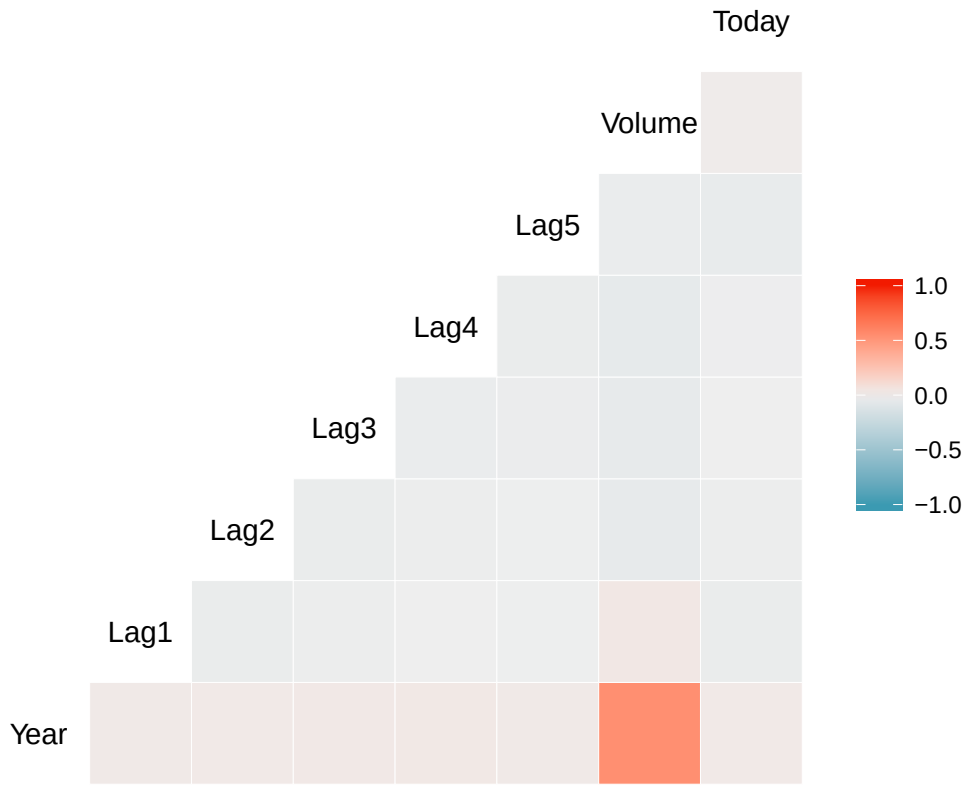
Get a better colored plot by the direction of the market from the prio day

```
pairs(Smarket, col = Smarket$Direction)
```



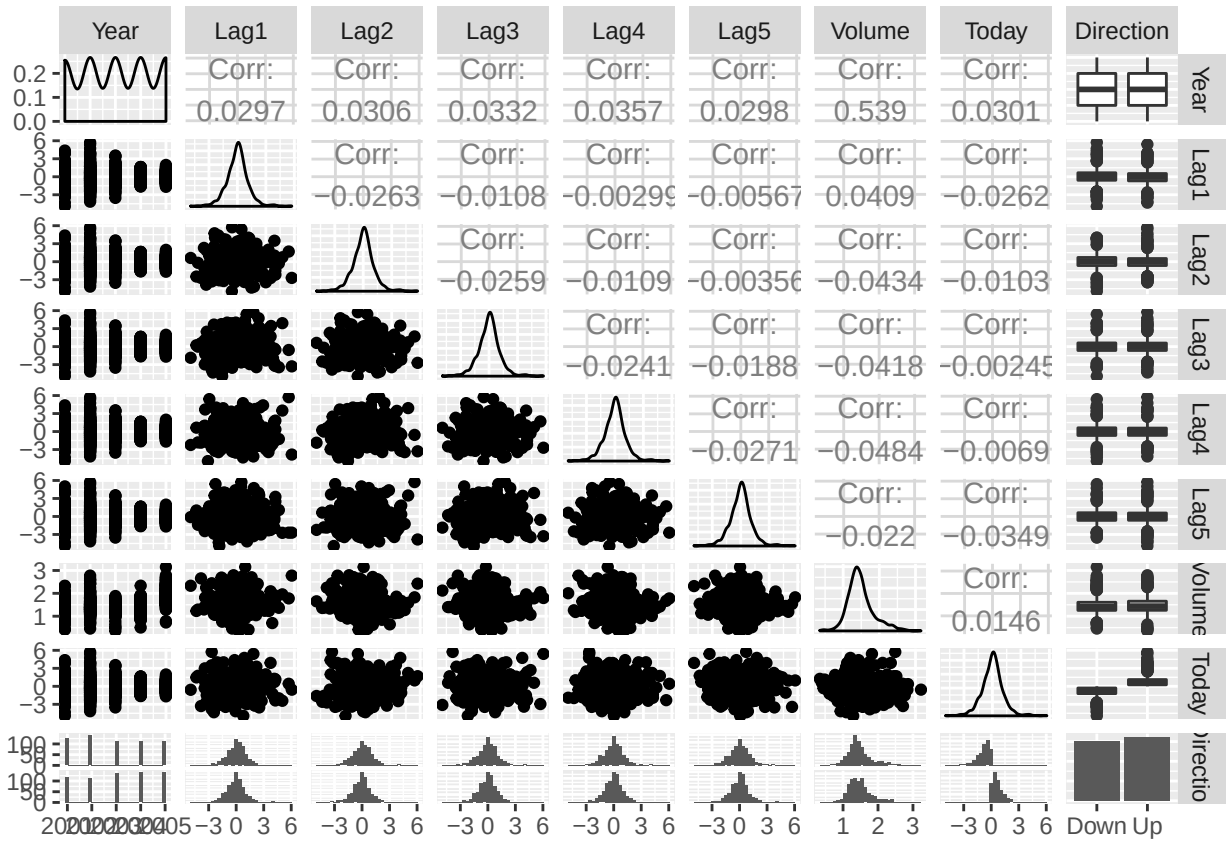

```
library(ggplot2)
library(GGally)
ggcorr(Smarket)
```

```
## Warning in ggcorr(Smarket): data in column(s) 'Direction' are not numeric
## and were ignored
```




```
ggpairs(Smarket)
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



Check the correlation coefficients in the data

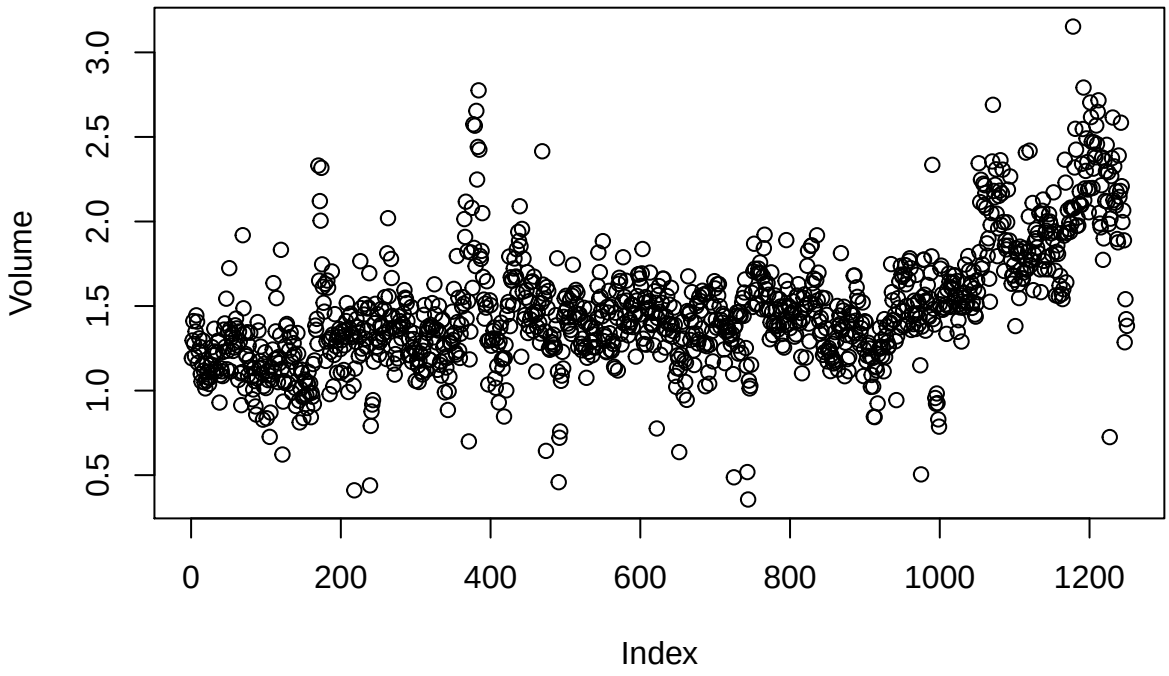
```
# cor(Smarket)
```

But column 9 is not numeric, its character string of market direction Up or Down, so remove it when calculating the correlation coefficients.

```
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
```

```
attach(Smarket)
plot(Volume)
```



12.2.1.3 Logistic Regression

Lets do a logistic regression model of

- the Market Direction
  - (categorical variable, so its one class and has two values)
- versus the lag varaibles and volume.

And we say there is a binomial probability distribution.

12.2.1.3.1 The binomial distribution

In probability theory and statistics, the binomial distribution

- with parameters n and p is the discrete probability distribution
  - of the number of successes
  - in a sequence of n independent yes/no experiments,
  - each of which yields success with probability p.
- A success/failure experiment
  - is also called a Bernoulli experiment or Bernoulli trial;
- when n = 1, the binomial distribution is a Bernoulli distribution.
- The binomial distribution is the basis
  - for the popular binomial test of statistical significance.

The binomial distribution is frequently used to model

- the number of successes in a sample of size n drawn
- WITH replacement from a population of size N.

If the sampling is carried out WITHOUT replacement,

- the draws are not independent and
- so the resulting distribution is a hypergeometric distribution,
  - not a binomial one.

However, for N much larger than n,

- the binomial distribution is a good approximation, and widely used.

12.2.1.3.2 Back to our logistic regression modeling

So lets fit our logistic regression model,

- using the glm command
- and look at the summary.

```
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)
##
## Deviance Residuals:
##    Min      1Q  Median      3Q     Max
## -1.446  -1.203   1.065   1.145   1.326
##
## 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
```

We get

- the coefficents,
- the Std. Errors,
- the z-values and
- the p-values

You'll see that, from the p-values,

- none of the coefficients are significant
- This isn't suprising, the variables may be correlated
  - but the variables don't look correlated

We can look at the coefficents

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

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

```
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
```

```
summary(glm.fit)$coef[,4]
```

```
## (Intercept)        Lag1        Lag2        Lag3        Lag4        Lag5
##   0.6006983   0.1452272   0.3983491   0.8243333   0.8514445   0.8349974
##      Volume
##   0.3924004
```

We can make predictions from the model

- using the predict function.

We'll tell it the type is response,

0 and be looking for the probability of next day's market direction

```
glm.probs <- predict(glm.fit,type = "response")
```

The probabilities glm.probs

- come out close to 50% as we'd expect

We can break up the predictions into 10 grouping

And then classify them into

- Up if above 0.50
- or Down if below 0.50.

So that we can see the market direction behavior.

```
glm.probs[1:10]
```

```
##         1         2         3         4         5         6         7
## 0.5070841 0.4814679 0.4811388 0.5152224 0.5107812 0.5069565 0.4926509
##         8         9        10
## 0.5092292 0.5176135 0.4888378
```

```
contrasts(Direction)
```

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

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

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

So the table gives our correct predictions on the diagonal.

- And our wrong predictions on the off-diagonal.
- So the off-diagonal results are mistakes for our model.

And there are a lot of these wrong predictions by our model.

And we have gotten

```
(507 + 145)/1250
```

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

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

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

So you can see that we got 52.16% of the market predictions correct

- With our Logistic Prediction model.

12.2.1.3.3 Lets make a training and test datasets and try to fit better models

We want to have the training be the larger dataset

- And the test dataset is smaller in size.

Lets take the data before 2005 as our training set

- train is a vector of logicals,
- where for year<2005,
  - train is false

12.2.1.3.4 Now lets fit our glm model to the training data

And now we'll only fit our model to the subset = train

- and again using all 5 Lag predictor variables
  - subset=train is our training data
  - subset=!train, where ! means "NOT", is our test set

We'll check the size of the training set and inspect it a bit.

```
train <- (Year < 2005)
Smarket.2005 <- Smarket[!train,]
dim(Smarket.2005)
```

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

```
Direction.2005 <- Direction[!train]
glm.fit <- glm(Direction~Lag1 + Lag2 + Lag3 + Lag4 + Lag5 + Volume,data = Smarket,family = binomial,subs
```

So now we have fitted a new glm logistic regression model

- to the training data

Now we'll do prediction using this model

- on the testing "!train" data

Direction.2005 is the response variable

```
glm.probs <- predict(glm.fit,Smarket.2005,type = "response")
glm.pred <- rep("Down",252)
glm.pred[glm.probs > .5] = "Up"
table(glm.pred,Direction.2005)
```

```
##         Direction.2005
## glm.pred Down Up
##     Down   77 97
##     Up     34 44
```

```
mean(glm.pred == Direction.2005)
```

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

```
mean(glm.pred != Direction.2005)
```

```
## [1] 0.5198413
```

This time our predictive model is worse

- being correct only 48% of the time
- when tested on our test dataset

We probably are overfitting

- Model optimism is too high
- And our predictions are not sensible for the test dataset

Demonstrates the importance of training and testing

- So as to evaluate the predictive capability of a model

12.2.1.3.5 So lets try fitting a simpler model

Lets try fitting our glm model to only the Lag1 and Lag2 predictors

```
glm.fit <- glm(Direction~Lag1 + Lag2,data = Smarket,family = binomial,subset = train)
glm.probs <- predict(glm.fit,Smarket.2005,type = "response")
glm.pred <- rep("Down",252)
glm.pred[glm.probs > .5] = "Up"
table(glm.pred,Direction.2005)
```

```
##         Direction.2005
## glm.pred Down  Up
##     Down   35  35
##     Up     76 106
```

```r
mean(glm.pred == Direction.2005)
```

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

```r
(35 + 106)/(35 + 35 + 76 + 106)
```

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

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

```
##         1         2
## 0.4791462 0.4960939
```

Now our predictive model is best to date

- getting it right 55.95% of the time
- so the correct classification rate of 55.95

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

```
##
## Call:
## glm(formula = Direction ~ Lag1 + Lag2, family = binomial, data = Smarket,
##     subset = train)
##
## Deviance Residuals:
##    Min      1Q  Median      3Q     Max
## -1.345  -1.188   1.074   1.164   1.326
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  0.03222    0.06338   0.508    0.611
## Lag1        -0.05562    0.05171  -1.076    0.282
## Lag2        -0.04449    0.05166  -0.861    0.389
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1383.3  on 997  degrees of freedom
## Residual deviance: 1381.4  on 995  degrees of freedom
## AIC: 1387.4
##
## Number of Fisher Scoring iterations: 3
```

So still nothing became significant even with the simpler model

- all the p-values say not significant

12.2.1.4 Linear Discriminant Analysis

So lets use the ISLR and MASS pacakges,

- using the require (same as the library command)
- to load the ISLR and MASS packages

We'll do Linear Discriminant Analysis (LDA)

```r
require(MASS)
```

```
## Loading required package: MASS
```

```
?lda
```

```
## starting httpd help server ...
```

```
##  done
```

So lets use our stock market data (Smarket)

- the response will be the direction the market took on a day
- And the predictors will be the returns on the previous two days

The response is direction

We'll use the subset=train where train is the data before 2005

- And then we'll make predictions for the year 2005

```
lda.fit <- lda(Direction~Lag1 + Lag2,data = Smarket,subset = train)
lda.fit
```

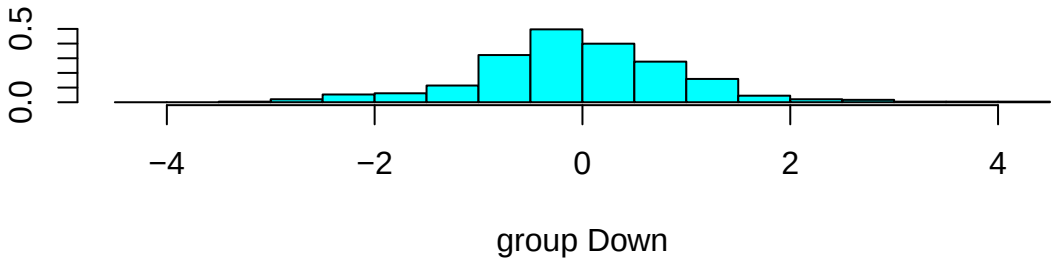
```
## Call:
## lda(Direction ~ Lag1 + Lag2, data = Smarket, subset = train)
##
## Prior probabilities of groups:
##     Down       Up
## 0.491984 0.508016
##
## Group means:
##             Lag1        Lag2
## Down  0.04279022  0.03389409
## Up   -0.03954635 -0.03132544
##
## Coefficients of linear discriminants:
##             LD1
## Lag1 -0.6420190
## Lag2 -0.5135293
```

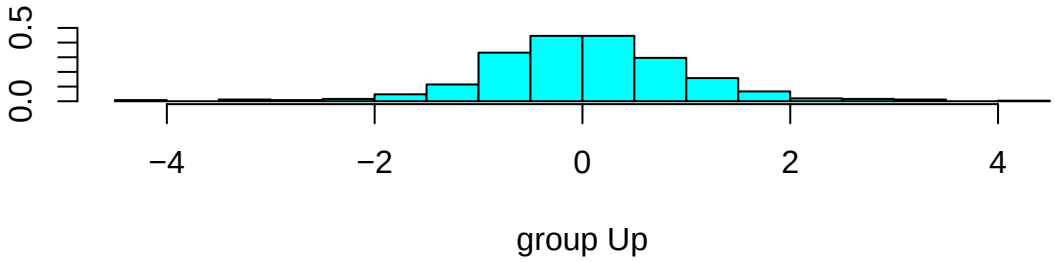
In the summary of lda.fit We see

- the formula used for the model
- a summary of the LDA
- We see roughly 50% ups and downs
- We see the group means for the ups and downs

Then the LDA coefficients, the linear function to separate the two groups

```
plot(lda.fit)
```





The two histograms of the ups and downs

- look pretty much the same

Next lets check our prediction for year 2005

```
lda.pred <- predict(lda.fit, Smarket.2005)
names(lda.pred)
```

```
## [1] "class"     "posterior" "x"
```

```
lda.class <- lda.pred$class
```

Now lets check our

- correct predictions (on the diagonal)
- incorrect predictions (the off diagonal results)

And see how good our prediction is.

We'll look at lda.class, to see

- the predicted up/down
- vs. the actual in the testing dataset.

When we compare the predictions to the actual up/down results

- we'll get true falses
- or 0 and 1's

And we can calculate the mean, to get our prediction accuracy

So we got 56% correct prediction

- 0.56 could be a useful edge.

```
table(lda.class,Direction.2005)
```

```
##          Direction.2005
## lda.class Down  Up
##      Down   35  35
##      Up     76 106
```

```
mean(lda.class == Direction.2005)
```

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

```
sum(lda.pred$posterior[,1] >= .5)
```

```
## [1] 70
```

```
sum(lda.pred$posterior[,1] < .5)
```

```
## [1] 182
```

```
lda.pred$posterior[1:20,1]
```

```
##       999      1000      1001      1002      1003      1004      1005
## 0.4901792 0.4792185 0.4668185 0.4740011 0.4927877 0.4938562 0.4951016
##      1006      1007      1008      1009      1010      1011      1012
## 0.4872861 0.4907013 0.4844026 0.4906963 0.5119988 0.4895152 0.4706761
##      1013      1014      1015      1016      1017      1018
## 0.4744593 0.4799583 0.4935775 0.5030894 0.4978806 0.4886331
```

```
lda.class[1:20]
```

```
##  [1] Up   Up   Up   Up   Up   Up   Up   Up   Up   Up   Up   Down Up   Up
## [15] Up   Up   Up   Down Up   Up
## Levels: Down Up
```

```
sum(lda.pred$posterior[,1] > .9)
```

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

12.2.1.5 Quadratic Discriminant Analysis

Here we can do the same Discriminant Analysis

But using a Quadratic DA, instead of the LDA.

- Using the gda function

```
qda.fit <- qda(Direction~Lag1 + Lag2,data = Smarket,subset = train)
qda.fit
```

```
## Call:
## qda(Direction ~ Lag1 + Lag2, data = Smarket, subset = train)
##
## Prior probabilities of groups:
##     Down       Up
## 0.491984 0.508016
##
## Group means:
##             Lag1        Lag2
## Down  0.04279022  0.03389409
```

```
## Up   -0.03954635 -0.03132544
```

```
qda.class <- predict(qda.fit,Smarket.2005)$class
table(qda.class,Direction.2005)
```

```
##          Direction.2005
## qda.class Down  Up
##      Down   30  20
##      Up     81 121
```

```
mean(qda.class == Direction.2005)
```

```
## [1] 0.5992063
```

Now we are able to predict correctly 59% of the Up/Down categories

- Of the market
- from the Lag1 and Lag2 data of prior years

So QDA worked better in this case.

12.2.1.5.1 Splitting Data into Training/Testing datasets

Is very important

- So as we try different statistical learning methods
- We can evaluate their performance
- And Identify the best method
- For use with the data, and the specific problem.

12.2.1.6 K-Nearest Neighbors

So now lets try K-nearest neighbors method on our stock market data

12.3 How does knn compare to kmeans?

```
library(class)
?kmeans
?knn
```

knn is different from the simpler k-means we used in class before.

knn vs. k-means

These are completely different methods.

- The fact that they both have the letter K in their name is a coincidence.

K-means is a clustering algorithm

- that tries to partition a set of points
  - into K sets (clusters)
- such that the points in each cluster
  - tend to be near each other.

It is an unsupervised classification method

- because the points have no external classification.

K-nearest neighbors is a classification (or regression) algorithm

- that in order to determine the classification of a point,
- combines the classification of the K nearest points.

It is supervised because

- you are trying to classify a point
- based on the known classification of other points.

Ahh, so a big difference

- k-means is unsupervised
- knn is supervised

So we want to keep our understanding of the types of approaches we can use.

knn is very simple, but works quite well much of the time

You should always try knn

12.3.0.0.1 Now on to trying knn

Notice that there is no formula given for knn

Setup the training and testing datasets

set the seed, so as to get reproducible results

knn wants to know

- the training, testing datasets
  - and variables and the number k
- the k = 1 says we want 1 nearest neighbor classification
- to classify a new observation,
  - use the closest (in euclidian distance) point
- and classify the same

We'll use Lag1 and Lag2 like the last model

```
train.X <- cbind(Lag1,Lag2)[train,]
test.X <- cbind(Lag1,Lag2)[!train,]
train.Direction <- Direction[train]
set.seed(1)
knn.pred <- knn(train.X,test.X,train.Direction,k = 1)
table(knn.pred,Direction.2005)
```

```
##         Direction.2005
## knn.pred Down Up
##     Down   43 58
##     Up     68 83
```

```
(83 + 43)/252
```

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

So you see that with k=1, we got 0.5,

- so the knn was useless
- no better than flipping a coin
- we need to use more nearby points to get some sample information

12.3.0.0.2 So lets try k = 3

So lets try checking 3 nearbye points as we classify a new point

```
knn.pred <- knn(train.X,test.X,train.Direction,k = 3)
table(knn.pred,Direction.2005)
```

```
##         Direction.2005
## knn.pred Down Up
##     Down   48 54
##     Up     63 87
```

```
mean(knn.pred == Direction.2005)
```

```
## [1] 0.5357143
```

Not bad, now we got 53.6% classification right

12.3.0.0.3 What you would do in a real knn classificatiton problem

Try to fit knn models

- with an automated set of different k values

And evaluate for the set of knn models

- To determine what the best peforming k value, you should use, is.

12.3.0.1 An Application to Caravan Insurance Data

So here is an unannotated set of working code.

For the Caravan dataset

- The data contains 5822 real customer records.
- Each record consists of 86 variables,
  - containing sociodemographic data (variables 1-43)
  - and product ownership (variables 44-86).
- The sociodemographic data is derived from zip codes.
- All customers living in areas with the same zip code
  - have the same sociodemographic attributes.
  - Variable 86 (Purchase) indicates whether the customer
  - purchased a caravan insurance policy.
- Further information on the individual variables can be obtained at

And its modeld using knn and glm

```
dim(Caravan)
```

```
## [1] 5822   86
```

```
?Caravan
attach(Caravan)
summary(Purchase)
```

```
##   No  Yes
## 5474  348
```

```
348/5822
```

```
## [1] 0.05977327
```

```
standardized.X <- scale(Caravan[,-86])
var(Caravan[,1])
```

```
## [1] 165.0378
```

```
var(Caravan[,2])
```

```
## [1] 0.1647078
```

```
var(standardized.X[,1])
```

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

```
var(standardized.X[,2])
```

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

```
test <- 1:1000
train.X <- standardized.X[-test,]
test.X <- standardized.X[test,]
train.Y <- Purchase[-test]
test.Y <- Purchase[test]
set.seed(1)
knn.pred <- knn(train.X,test.X,train.Y,k = 1)
mean(test.Y != knn.pred)
```

```
## [1] 0.118
```

```
mean(test.Y != "No")
```

```
## [1] 0.059
```

```
table(knn.pred,test.Y)
```

```
##         test.Y
## knn.pred  No Yes
##      No  873  50
##      Yes  68   9
```

```
9/(68 + 9)
```

```
## [1] 0.1168831
```

```
knn.pred <- knn(train.X,test.X,train.Y,k = 3)
table(knn.pred,test.Y)
```

```
##         test.Y
## knn.pred  No Yes
##      No  920  54
##      Yes  21   5
```

```
5/26
```

```
## [1] 0.1923077
```

```
knn.pred <- knn(train.X,test.X,train.Y,k = 5)
table(knn.pred,test.Y)
```

```
##         test.Y
## knn.pred  No Yes
##      No  930  55
##      Yes  11   4
```

```
4/15
```

```
## [1] 0.2666667
```

```
glm.fit <- glm(Purchase~.,data = Caravan,family = binomial,subset = -test)
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
glm.probs <- predict(glm.fit,Caravan[test,],type = "response")
glm.pred <- rep("No",1000)
glm.pred[glm.probs > .5] = "Yes"
table(glm.pred,test.Y)
```

```
##         test.Y
## glm.pred  No Yes
##      No  934  59
##      Yes   7   0
```

```
glm.pred <- rep("No",1000)
glm.pred[glm.probs > .25] = "Yes"
table(glm.pred,test.Y)
```

```
##         test.Y
## glm.pred  No Yes
##      No  919  48
##      Yes  22  11
```

```
11/(22 + 11)
```

```
## [1] 0.3333333
```

12.3.0.2 Cites

- Gareth James, Daniela Witten, Trevor Hastie, and Robert Tibshirani. An Introduction to Statistical Learning: With Applications in R. 1st ed. 2013, Corr. 5th printing 2015 edition. Springer Texts in Statistics. New York: Springer, 2013.