MATH 4322, Intro to Data Science & Machine Learning, Lab # 3.

Instructors: Cathy Poliak, Wendy Wang

DUE: Monday September 23 at 11:59.

1 The Stock Market Data

We will begin by examining some numerical and graphical summaries of the *Smarket* data, which is part of the *ISLR* library. This data set consists of percentage returns for the *S&P* 500 stock index over $n = 1250$ days, from the beginning of 2001 until the end of 2005. For each date, we have recorded

- $Lag1, \ldots, Lag5$ - the percentage returns for each of the five previous trading days,
- *Volume* (the number of shares traded on the previous day, in billions),
- *Today* (the percentage return on the date in question), and
- *Direction* (whether the market was *Up* or *Down* on this date).

```
> library(ISLR)
> names(Smarket)
[1] "Year"      "Lag1"      "Lag2"      "Lag3"      "Lag4"
[6] "Lag5"      "Volume"    "Today"     "Direction"
> dim(Smarket)
[1] 1250    9
> summary(Smarket)
```

Task #1: How many times was the market *Up*? *Down*?

The *cor()* function produces a matrix that contains all of the pairwise correlations among the quaNTitative predictors in a data set. The first command below gives an error message because the *Direction* variable is quaLitative.

```
> cor(Smarket)
Error in cor(Smarket) : 'x' must be numeric
> cor(Smarket[,-9])
         Year     Lag1     Lag2     Lag3     Lag4     Lag5
Year   1.0000  0.02970  0.03060  0.03319  0.03569  0.02979
Lag1   0.0297  1.00000 -0.02629 -0.01080 -0.00299 -0.00567
Lag2   0.0306 -0.02629  1.00000 -0.02590 -0.01085 -0.00356
Lag3   0.0332 -0.01080 -0.02590  1.00000 -0.02405 -0.01881
Lag4   0.0357 -0.00299 -0.01085 -0.02405  1.00000 -0.02708
Lag5   0.0298 -0.00567 -0.00356 -0.01881 -0.02708  1.00000
Volume 0.5390  0.04091 -0.04338 -0.04182 -0.04841 -0.02200
Today  0.0301 -0.02616 -0.01025 -0.00245 -0.00690 -0.03486
        Volume    Today
Year    0.5390  0.03010
Lag1    0.0409 -0.02616
Lag2   -0.0434 -0.01025
Lag3   -0.0418 -0.00245
Lag4   -0.0484 -0.00690
Lag5   -0.0220 -0.03486
Volume  1.0000  0.01459
Today   0.0146  1.00000
```

As one would expect, the correlations between the lag variables and todays returns are close to zero. In other words, there appears to be little correlation between todays returns and previous days returns (otherwise we wouldn't all be here today).

2 Logistic Regression.

Next, we will fit a logistic regression model in order to

- predict *Direction* (our response),
- using $Lag1, \dots Lag5$ and $Volume$ as our predictors.

The *glm()* function fits generalized linear models, one of which is logistic regression. The syntax of *glm()* is similar to that of *lm()*, except that we must pass in the argument $family = binomial$, which tells R to run a logistic regression in particular.

```
> 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

...
```

Task #2: The smallest p-value here is associated with $Lag1$. How do we interpret its coefficient estimate?

We can also use the *summary()* function to access particular aspects of the fitted model, such as the p-values for the coefficients. E.g. below we access

- the whole coefficient matrix, and
- a vector of p-values in particular

```
> summary(glm.fit)$coef
             Estimate  Std. Error  z value  Pr(>|z|)
(Intercept)  -0.12600     0.2407   -0.523     0.601
Lag1         -0.07307     0.0502   -1.457     0.145
Lag2         -0.04230     0.0501   -0.845     0.398
Lag3          0.01109     0.0499    0.222     0.824
Lag4          0.00936     0.0500    0.187     0.851
Lag5          0.01031     0.0495    0.208     0.835
Volume        0.13544     0.1584    0.855     0.392
> summary(glm.fit)$coef[,4]
(Intercept)        Lag1        Lag2        Lag3        Lag4
      0.601       0.145       0.398       0.824       0.851
       Lag5      Volume
      0.835       0.392
```

The *predict*() function can be used to predict the **probability** of market going up, given values of the predictors. The $type = "response"$ option tells R to output probabilities of the form $P(Y = 1|X)$, rather than other information such as the *logit* value $\log(\frac{P(Y=1|X)}{1-P(Y=1|X)})$. If no data set is supplied to the *predict*() function, then the probabilities are automatically computed for the training data that was used to fit the logistic regression model. The *contrasts*() function shows whether $Y = 1$ corresponds to value *Up* or *Down*.

```
> glm.probs=predict(glm.fit,type="response")
> glm.probs[1:10]
    1     2     3     4     5     6     7     8     9    10
0.507 0.481 0.481 0.515 0.511 0.507 0.493 0.509 0.518 0.489
```

```
> contrasts(Direction)
     Up
Down  0
Up    1
```

In order to predict for a particular 5-day dynamic and volume combination, we have to supply a *new.data* option to *predict*(). E.g. assume we'd like to predict for a "wavy" 5-day dynamic, like

- $Lag1 = Lag3 = Lag5 = 0.5$,
- $Lag2 = Lag4 = -0.5$

and a $Volume$ of 2 billion shares.

```
> plot(c(0.5,-0.5,0.5,-0.5,0.5), type='l') # "Catch" the wave.
> predict(glm.fit, type="response", newdata = data.frame(Lag1 =0.5, Lag2 = -0.5,
                                                         Lag3= 0.5, Lag4=-0.5,
                                                         Lag5= 0.5,
                                                         Volume=2))
...
```

Task #3: What is the predicted value? How do we interpret it?

In order to make actual predictions on whether the market will go *Up* or *Down* on a particular day, we must convert these predicted probabilities into class labels, *Up* or *Down*.

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

- The first command creates a vector of 1,250 *Down* elements.
- The second line transforms to *Up* all of the elements for which the predicted probability of a market increase exceeds 0.5 ($glm.probs > .5$)

Given these predictions, the *table*() function can be used to produce a **confusion matrix** in order to determine how many observations were correctly or incorrectly classified:

- the **diagonal** elements of the confusion matrix indicate **correct** predictions,
- while the **off-diagonals** represent **incorrect** predictions.

The *mean*() function can be used to compute the **fraction** of days for which the prediction was **correct** (if those lines don't work, use *attach*(*Smarket*) **call**)

```
> attach(Smarket)
```

```
> table(glm.pred,Direction)
```

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

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

```
[1] 0.5216
```

At first glance, it appears that the logistic regression model is working a little better than random guessing. However, this result is misleading because we **trained** and **tested** the model on the **same set of** $1,250$ **observations**. In other words, $100 - 52.2 = 47.8\%$ is the training error rate, which tends to be overly optimistic.

To better assess the accuracy of the logistic regression model in this setting, and yield a more realistic error rate, we can

- fit the model using part of the data (training data), and
- then examine how well it predicts the held out data.

To implement this strategy, we will first create a *Boolean* vector *train*, that takes on

- value *TRUE* if the observation occurred **before** 2005,
- value *FALSE* if the observation occurred **in** 2005

vector corresponding to the observations from 2001 through 2004. We will then use this vector to create a held out data set of observations from 2005.

```
> train=(Year<2005)
> Smarket.2005=Smarket[!train,]
> Direction.2005=Direction[!train]
```

Boolean vectors, like *train*, can be used to obtain a subset of the rows or columns of a matrix. For instance, the command *Smarket*[*train*,] would pick out a submatrix of the stock market data set, corresponding only to the dates **before** 2005, since those are the ones for which the elements of train are *TRUE*. .

The ! symbol can be used to reverse all of the elements of a Boolean vector: the $TRUE$ values become $FALSE$, and vice versa. Therefore, $Smarket[!train,]$ yields a submatrix of the stock market data containing only the observations for dates **in** 2005.

Now let's fit a logistic regression model using only the subset of the observations that correspond to dates **before** 2005, using the *subset* argument of $glm()$ function:

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

We then obtain predicted probabilities of the stock market going up for each of the days in our test set - that is, for the days **in** 2005.

```
glm.probs = predict(glm.fit, Smarket.2005 , type ="response")
```

Task #4: Print the first 10 predictions for test data (first 10 elements of *glm.probs*.

Notice that we have trained and tested our model on two completely separate data sets:

- training was performed using only the dates **before** 2005, and
- testing was performed using only the dates **in** 2005.

Finally, we compute the predictions for 2005 and compare them to the actual movements of the market over that time period.

```
glm.pred = rep("Down",252)
glm.pred[glm.probs>.5]="Up"
table(glm.pred,Direction.2005)
mean(glm.pred == Direction.2005)
```

Task #5: Print the resulting confusion matrix and the test error rate.

The results are rather disappointing: the rate of correct predictions is 48%, which is worse than random guessing!

We recall that the logistic regression model had very underwhelming p-values associated with all of the predictors, and that the smallest p-value, though not very small, corresponded to $Lag1$.

Perhaps by removing the variables that appear not to be helpful in predicting $Direction$, we can obtain a more effective model. After all, using predictors that have no relationship with the response tends to cause a deterioration in the test error rate (since such predictors cause an increase in variance without a corresponding decrease in bias), and so removing such predictors may in turn yield an improvement.

Below we have refit the logistic regression using just $Lag1$ and $Lag2$, which seemed to have the highest predictive power in the original logistic regression model.

```
> 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
> mean(glm.pred == Direction.2005)
[1] 0.5595238
```

Now the results appear to be a little better: 56% of the daily movements have been correctly predicted. It is worth noting that in this case, a much simpler strategy of predicting that the market will increase every day will also be correct 56% of the time! The confusion matrix shows that on days when logistic regression predicts an increase in the market, it has a 58% accuracy rate. This suggests a possible trading strategy of buying on days when the model predicts an increasing market, and avoiding trades on days when a decrease is predicted. Of course one would need to investigate more carefully whether this small improvement was real or just due to random chance.