DSI-06 Homework 3: Chapter 4, pg 193

Julia Gallucci

2023-02-20

13. This question should be answered using the Weekly data set, which is part of the ISLR2 package.

```
#install.packages("ISLR2") install ISLR2 package if you haven't already done so
library(ISLR2)
df<- Weekly #save Weekly dataset to variable df.
head(df) #return the column names and first few rows of the dataset
```

```
##   Year   Lag1   Lag2   Lag3   Lag4   Lag5    Volume  Today Direction
## 1 1990  0.816  1.572 -3.936 -0.229 -3.484 0.1549760 -0.270      Down
## 2 1990 -0.270  0.816  1.572 -3.936 -0.229 0.1485740 -2.576      Down
## 3 1990 -2.576 -0.270  0.816  1.572 -3.936 0.1598375  3.514        Up
## 4 1990  3.514 -2.576 -0.270  0.816  1.572 0.1616300  0.712        Up
## 5 1990  0.712  3.514 -2.576 -0.270  0.816 0.1537280  1.178        Up
## 6 1990  1.178  0.712  3.514 -2.576 -0.270 0.1544440 -1.372      Down
```

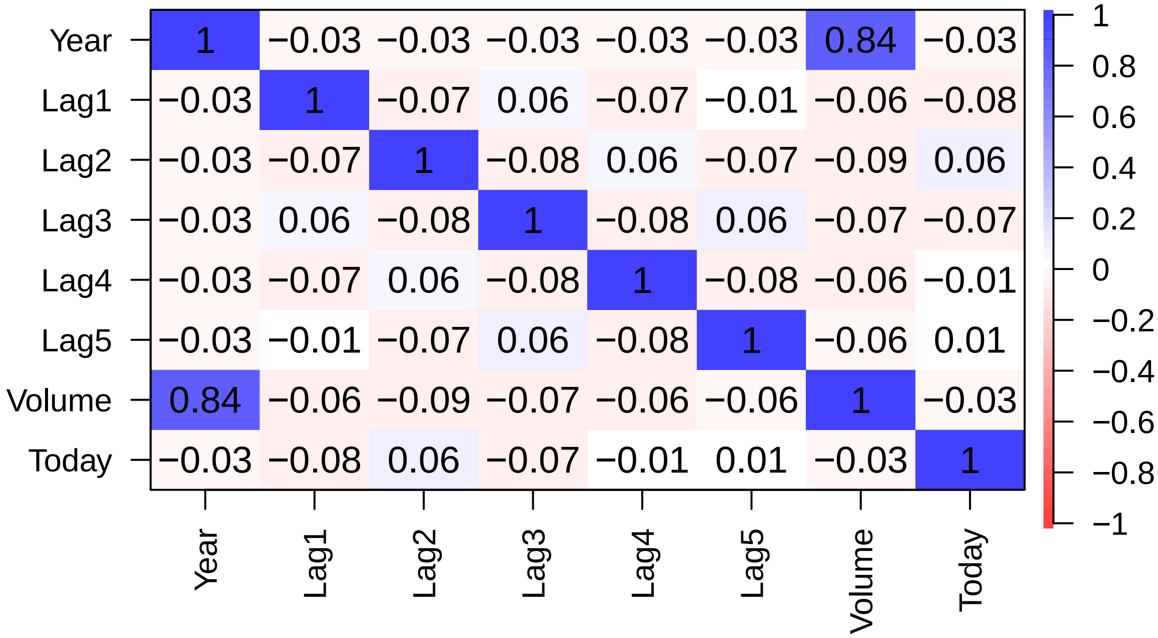
(a) Produce some numerical and graphical summaries of the Weekly data. Do there appear to be any patterns?

```
#summary of the dataframe
summary(df)
```

```
##       Year           Lag1               Lag2               Lag3
##  Min.   :1990   Min.   :-18.1950   Min.   :-18.1950   Min.   :-18.1950
##  1st Qu.:1995   1st Qu.: -1.1540   1st Qu.: -1.1540   1st Qu.: -1.1580
##  Median :2000   Median :  0.2410   Median :  0.2410   Median :  0.2410
##  Mean   :2000   Mean   :  0.1506   Mean   :  0.1511   Mean   :  0.1472
##  3rd Qu.:2005   3rd Qu.:  1.4050   3rd Qu.:  1.4090   3rd Qu.:  1.4090
##  Max.   :2010   Max.   : 12.0260   Max.   : 12.0260   Max.   : 12.0260
##       Lag4               Lag5              Volume            Today
##  Min.   :-18.1950   Min.   :-18.1950   Min.   :0.08747   Min.   :-18.1950
##  1st Qu.: -1.1580   1st Qu.: -1.1660   1st Qu.:0.33202   1st Qu.: -1.1540
##  Median :  0.2380   Median :  0.2340   Median :1.00268   Median :  0.2410
##  Mean   :  0.1458   Mean   :  0.1399   Mean   :1.57462   Mean   :  0.1499
##  3rd Qu.:  1.4090   3rd Qu.:  1.4050   3rd Qu.:2.05373   3rd Qu.:  1.4050
##  Max.   : 12.0260   Max.   : 12.0260   Max.   :9.32821   Max.   : 12.0260
##  Direction
##  Down:484
##  Up  :605
##
##
##
```

```
##
```

```
#plot correlations
#install.packages('psych') install psych package if you havent already done so
library(psych)
cor_Weekly <- cor(df[,-9]) #get correlation of all rows across all variables besides column 9 (not nume
cor.plot(cor_Weekly, xlas = 2) #xlas =2 rotates the variable labels for better readibility
```



We can see a pattern! We have a significant linear relationship between Year and Volume. The correlational plot does not seem to illustrate that any other variables are significantly linearly related.

(b) Use the full data set to perform a logistic regression with Direction as the response and the five lag variables plus Volume as predictors. Use the summary function to print the results. Do any of the predictors appear to be statistically significant? If so,which ones?

```
#fit a logistic regression model to the entire dataset
log.fit_allData <- glm(Direction ~ Lag1 + Lag2 + Lag3 + Lag4 + Lag5 + Volume, family = binomial, data =
summary(log.fit_allData)
```

```
##
## Call:
## glm(formula = Direction ~ Lag1 + Lag2 + Lag3 + Lag4 + Lag5 +
##     Volume, family = binomial, data = df)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.6949  -1.2565   0.9913   1.0849   1.4579
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  0.26686    0.08593   3.106   0.0019 **
## Lag1        -0.04127    0.02641  -1.563   0.1181
## Lag2         0.05844    0.02686   2.175   0.0296 *
```

```
## Lag3        -0.01606    0.02666  -0.602   0.5469
## Lag4        -0.02779    0.02646  -1.050   0.2937
## Lag5        -0.01447    0.02638  -0.549   0.5833
## Volume      -0.02274    0.03690  -0.616   0.5377
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1496.2  on 1088  degrees of freedom
## Residual deviance: 1486.4  on 1082  degrees of freedom
## AIC: 1500.4
##
## Number of Fisher Scoring iterations: 4
```

The column labelled Pr(>|z|) gives the p-values associated with each variables. Recall that the p-values indicate whether or not to reject the null hypothesis that there is no association between the response and predictor variable. Lag 2 appear to be statistically significant!

(c) Compute the confusion matrix and overall fraction of correct predictions. Explain what the confusion matrix is telling you about the types of mistakes made by logistic regression

The predict() function can be used to predict the probability of the direction given the value of the predictors. We pass type = "response" into the predict() function in order to output the probabilities in the form $P(Y = 1 \mid X)$. Use the contrasts() function to determine how R has assigned the dummy variables to the qualitative predictor Direction.

```
contrasts(df$Direction)
```

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

```
#predict
log.prob <- predict(log.fit_allData, type = "response")
head(log.prob)
```

```
##         1         2         3         4         5         6
## 0.6086249 0.6010314 0.5875699 0.4816416 0.6169013 0.5684190
```

Using the probabilities, we can set a threshold and make a table of predictions. Let's use a threshold of 0.5 to start which means that the direction will be predicted as up if the probability is greater than 0.5.

```
#classify probabilities as predictions of "up" or "down
log.predict_class <- ifelse(log.prob > 0.5, "Up", "Down")
head(log.predict_class)
```

```
##      1      2      3      4      5      6
##   "Up"   "Up"   "Up" "Down"   "Up"   "Up"
```

We can make a confusion matrix from the predictions using table() in order to determine how many observations were correctly or incorrectly classified.

```
table(log.predict_class, df$Direction)
```

```
##
## log.predict_class Down  Up
```

```
##              Down   54  48
##              Up    430 557
```

The diagonal entries are the correct predictions and the off-diagonals are the incorrect predictions. The mean() function can be used to compute the fraction of weeks for which the direction prediction was correct. Accuracy can be computed as average of when our predicted direction is equal to the actual direction

```r
##accuracy
mean(log.predict_class == df$Direction)
```

```
## [1] 0.5610652
```

This illustrates that the model predicted the weekly trend correctly ~56.11% of the time.

(d) Now fit the logistic regression model using a training data period from 1990 to 2008, with Lag2 as the only predictor. Compute the confusion matrix and the overall fraction of correct predictions for the held out data (that is, the data from 2009 and 2010)

In order to run classification methods on this data set, it is a good idea to separate the data into training and testing sets. This allows us to get an idea of the accuracy of our classification models with both the training and testing error rates.

```r
#split into test and train data
train_obs <- (df$Year < 2009)
df.train <- df[train_obs,]
df.test <- df[!train_obs, ]
```

```r
#fit logistic regression of the training dataset
log.fit_trainData <- glm(Direction ~ Lag2, family = binomial, data = df, subset = train_obs)
summary(log.fit_trainData)
```

```
##
## Call:
## glm(formula = Direction ~ Lag2, family = binomial, data = df,
##     subset = train_obs)
##
## Deviance Residuals:
##    Min      1Q  Median      3Q     Max
## -1.536  -1.264   1.021   1.091   1.368
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  0.20326    0.06428   3.162  0.00157 **
## Lag2         0.05810    0.02870   2.024  0.04298 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1354.7  on 984  degrees of freedom
## Residual deviance: 1350.5  on 983  degrees of freedom
## AIC: 1354.5
##
## Number of Fisher Scoring iterations: 4
```

Now, we can use model fit to the train data to predict new test data

```r
#fit the model to new testing dataset
log.prob <- predict(log.fit_trainData, df.test, type = "response")
log.predict_class <- ifelse(log.prob > 0.5, "Up", "Down")
table(log.predict_class, df.test$Direction)
```

```
##
## log.predict_class Down Up
##              Down    9  5
##              Up     34 56
```

```r
mean(log.predict_class == df.test$Direction)
```

```
## [1] 0.625
```

Note that 100% - 62.5% = 37.5% is the testing error rate.

(e) Repeat (d) using LDA.

We will perform LDA on the Weekly data in order to predict Direction using Lag2. We can fit an LDA model using the lda() function which belongs to the MASS library.

```r
library(MASS) #load in library for LDA classifier
```

```
##
## Attaching package: 'MASS'

## The following object is masked from 'package:ISLR2':
##
##     Boston
```

```r
#fit lda model to the train dataset
lda.fit_trainData <- lda(Direction ~ Lag2, data = df, subset = train_obs)
lda.fit_trainData
```

```
## Call:
## lda(Direction ~ Lag2, data = df, subset = train_obs)
##
## Prior probabilities of groups:
##      Down        Up
## 0.4477157 0.5522843
##
## Group means:
##             Lag2
## Down -0.03568254
## Up    0.26036581
##
## Coefficients of linear discriminants:
##            LD1
## Lag2 0.4414162
```

We can use the predict() function to use our LDA model on our test data set.

```r
#predict using our fitted lda model on the test set
lda.pred <- predict(lda.fit_trainData, df.test)
lda.class <- lda.pred$class
table(lda.class, df.test$Direction)
```

```
##
## lda.class Down Up
```

```
##      Down    9  5
##      Up     34 56
```

```
mean(lda.class == df.test$Direction)
```

```
## [1] 0.625
```

Note, using Linear Discriminant Analysis to develop a classifying model yielded similar results as the logistic regression model created in D (accuracy of 62.5%, test error of 37.5%)

(f) Repeat (d) using QDA.

We will fit a QDA model to the Weekly data using the qda() function within the MASS library.

```
#fit QDA model on training dataset
qda.fit_trainData <- qda(Direction ~ Lag2, data = df, subset = train_obs)
qda.fit_trainData
```

```
## Call:
## qda(Direction ~ Lag2, data = df, subset = train_obs)
##
## Prior probabilities of groups:
##      Down        Up
## 0.4477157 0.5522843
##
## Group means:
##             Lag2
## Down -0.03568254
## Up    0.26036581
```

We can use the predict() function to use our QDA model on our test data set.

```
#prediction
qda.pred <- predict(qda.fit_trainData, df.test)
qda.class <- qda.pred$class
table(qda.class, df.test$Direction)
```

```
##
## qda.class Down Up
##      Down    0  0
##      Up     43 61
```

```
#accuracy
mean(qda.class == df.test$Direction)
```

```
## [1] 0.5865385
```

Note, the Quadratic Linear Analysis created a model with an accuracy of 58.65% (test error rate of the QDA model is 41.35%), which is lower than the previous methods.

(g) Repeat (d) using KNN with K = 1.

We can fit a KNN model using the knn() function which belongs to the class library. Note, unlike the previous classification methods we have been running, this function fits the model and runs predictions in one step! Four inputs are required: • A data frame or matrix of the predictors in the training data. • A data frame or matrix of the predictors in the test data for which we want to make predictions on. • A vector containing the true classification of the training data. • An integer indicating the number of nearest neighbors to be considered by the classifier

```
library(class)
train.X <- as.data.frame(cbind(df$Lag2)[train_obs,]) #training data using lag2
test.X <- as.data.frame(cbind(df$Lag2)[!train_obs,]) #testing data using predictor lag2
train.Y <- df$Direction[train_obs] #training data using outcome direction
```

```
#fitting our model and testing it together!
set.seed(1) #set a random seed
knn.pred <- knn(train.X, test.X, train.Y, k = 1)
table(knn.pred, df.test$Direction)
```

```
##
## knn.pred Down Up
##     Down   21 30
##     Up     22 31
```

```
#average
mean(knn.pred == df.test$Direction)
```

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

Note, the K-Nearest neighbors resulted in an accuracy rate of 50%, which is equal to random chance!

(h) Repeat (d) using naive Bayes.

We will fit a naive Bayes model to the Weekly data using the naiveBayes() function which is part of the e1071 library.

```
#install.packages("e1071") install if you have not done so already
library(e1071)
```

```
#fit naiveB to training data
naiveB.fit_trainData <- naiveBayes(Direction ~ Lag2, data = df, subset = train_obs)
naiveB.fit_trainData
```

```
##
## Naive Bayes Classifier for Discrete Predictors
##
## Call:
## naiveBayes.default(x = X, y = Y, laplace = laplace)
##
## A-priori probabilities:
## Y
##      Down        Up
## 0.4477157 0.5522843
##
## Conditional probabilities:
##       Lag2
## Y             [,1]     [,2]
##   Down -0.03568254 2.199504
##   Up    0.26036581 2.317485
```

```
#predict using model on test data
naiveB.class <- predict(naiveB.fit_trainData, df.test)
table(naiveB.class, df.test$Direction)
```

```
##
## naiveB.class Down Up
```

```
##         Down    0  0
##         Up     43 61
```

```
#accuracy
mean(naiveB.class == df.test$Direction)
```

```
## [1] 0.5865385
```

Note, the naive Bayes created a model with an accuracy of 58.65% (test error rate of the model is 41.35%),

i) Which of these methods appears to provide the best results on this data?

Given the accuracy and test error rate, the Linear Discriminant Analysis and logistic regression model performed the best (accuracy of 62.5%, test error of 37.5%).

j) Experiment with different combinations of predictors, including possible transformations and interactions, for each of the methods. Report the variables, method, and associated confusion matrix that appears to provide the best results on the held out data. Note that you should also experiment with values for K in the KNN classifier.

examples:

Logistic regression with interaction

```
log.fit_interaction <- glm(Direction ~ Lag2*Lag4, family = binomial, data = df, subset = train_obs)
log.probs_interaction <- predict(log.fit_interaction, df.test, type = "response") #use model to predict
# Create a vector of length equal to the number of weeks in the test data set with each element as "Dow
log.pred <- ifelse(log.probs_interaction > 0.5, "Up","Down")
table(log.pred, df.test$Direction)
```

```
##
## log.pred Down Up
##     Down    4  4
##     Up     39 57
```

```
#accuracy
mean(log.pred == df.test$Direction)
```

```
## [1] 0.5865385
```

Linear Discriminant Analysis with interaction

```
lda.fit_interaction <- lda(Direction ~ Lag2*Lag4, data = df, subset = train_obs)
lda.pred_interaction <- predict(lda.fit_interaction, df.test)
lda.class_interaction <- lda.pred_interaction$class
table(lda.class_interaction, df.test$Direction)
```

```
##
## lda.class_interaction Down Up
##                  Down    4  4
##                  Up     39 57
```

```
#accuracy
mean(lda.class_interaction == df.test$Direction)
```

```
## [1] 0.5865385
```

KNN K=10

```
set.seed(1) #set a random seed
knn_10.pred <- knn(train.X, test.X, train.Y, k = 10)
table(knn_10.pred, df.test$Direction)
```

```
##
## knn_10.pred Down Up
##        Down   17 21
##        Up     26 40
```

```
#accuracy
mean(knn_10.pred == df.test$Direction)
```

```
## [1] 0.5480769
```