Predictive Modeling - Project 2

Eduardo Guzman

2022-04-26

Part 1

We first import the dataset and check to see if there are any missing values in any of the 6 columns. It turns out that only two of the columns have missing values.

```
#import the dataset
mydata <- read.table("project2.data", sep = ';', header = T)

#Observe that we have some missing values only in the X3 and Y columns
which(is.na(mydata$x3))
```

```
##  [1]  28  29  57  93 108 138 225 249 267 412
```

```
which(is.na(mydata$y))
```

```
##  [1]  86 204 232 336 376 424 436 459 474 496
```

Since there are missing values in X3 and Y, we gather the indices of the observations with missng values together, and remove them from the dataset. In this case, we find that there are 20 observations with missing data that we need to remove.

```
#Collect the indices of the observations with NA values.
removeindex <- NULL
removeindex <- append(removeindex, which(is.na(mydata$x3)))
removeindex <- append(removeindex, which(is.na(mydata$y)))

#Observe that there are 20 rows of data that we will need to remove
removeindex <- unique(removeindex)
length(removeindex)
```

```
## [1] 20
```

```
#Update the dataset by removeing those indices
mydata <- mydata[-removeindex,]
```

Part 2

Having removed all the missing values, we now want to partition our dataset so that we can train our models on 70% , and leave the other 30% for validation. One way to do this is to evenly distribute each of the n observations a random number between 1-10. The observations with numbers 1-7 become the training data, and the observations numbered 8-10 become the test data.

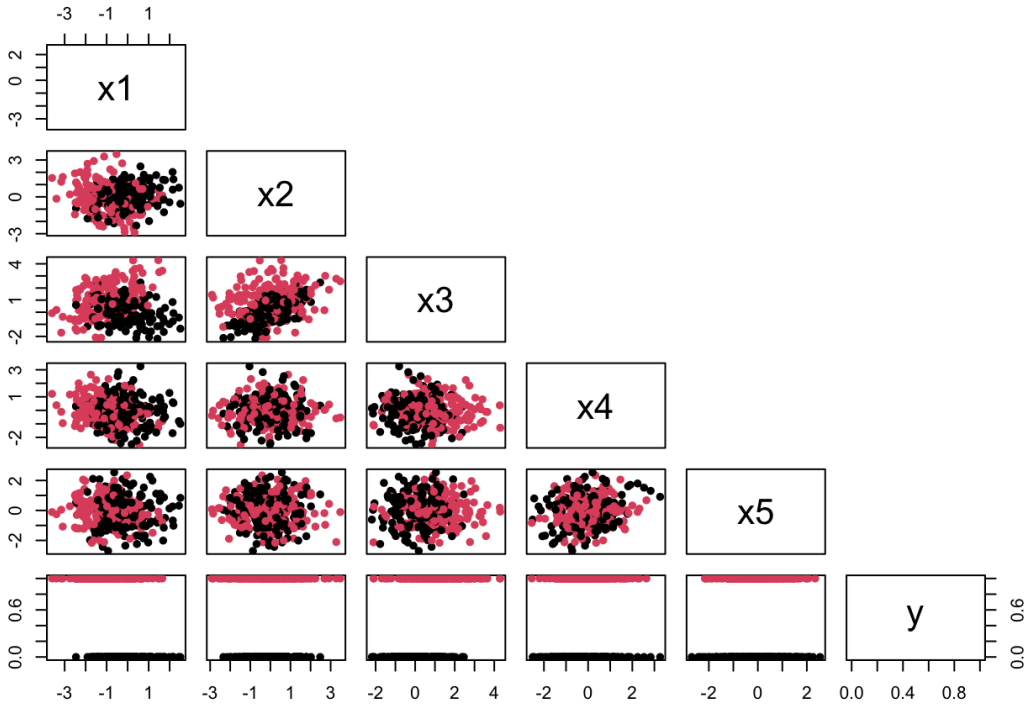
```
set.seed(2022)
n <- dim(mydata)[1]

#Create a 70-30 split between train and test data
id <- sample(rep(1:10, length=n))
train <- mydata[which(id<=7), ]
test <- mydata[which(id>=8), ]
```

Part 3

Using the training data that we constructed, we can create a quick scatter plot matrix to observe the relationships between all of the variables. In the plot below, the red observations correspond to the observations where y=1 (acount defaulted in last 3 months), and the black points are where y=0.

```
pairs(train, upper.panel = NULL, pch=16, col=train$y+1)
```



Upon inspection, there are some features where the two subgroups are more clearly separated - for example, the x1 vs x3 plot has the red and black points fairly well separated. This suggests that a model with both x1 and x3 would help in terms of predictive power. on the other hand, notice on the x5 vs y plot that there is a lot of overlap between the two groups. While x5 might contain some valuable information, it may not be the best for making accurate classifications given that there is so much overlap between the two groups.

```
#Calculate the mean of each varaible for each subgroup
aggregate(train$x1, list(train$y), FUN=mean)  # 0: -0.1217   1: -1.0299
aggregate(train$x2, list(train$y), FUN=mean)  # 0:  0.0030   1: -0.0240
aggregate(train$x3, list(train$y), FUN=mean)  # 0:  0.1182   1:  1.0631
aggregate(train$x4, list(train$y), FUN=mean)  # 0: -0.0920   1: -0.0920
aggregate(train$x5, list(train$y), FUN=mean)  # 0: -0.0246   1: -0.0220


#Standard deviations for each variable where y=1 (account defaults)
sd(train$x1[train$y==1]) #1.0013
sd(train$x2[train$y==1]) #1.2103
sd(train$x3[train$y==1]) #1.2234
sd(train$x4[train$y==1]) #0.9198
sd(train$x5[train$y==1]) #0.9106


#Standard deviations for each variable where y=0 (non-default)
sd(train$x1[train$y==0]) #0.9706
sd(train$x2[train$y==0]) #0.9327
sd(train$x3[train$y==0]) #0.9442
sd(train$x4[train$y==0]) #1.0249
sd(train$x5[train$y==0]) #1.0572
```

Note that both subgroups have features with mean's around 0 and a standard deviation of 1. This suggests that we do not need to rescale any of the columns.

Group A1

For this section we consider two prediction models – a logistic regression model with linear terms only and a logistic regression model with linear and quadratic terms - and then use 8-Fold Cross Validation to determine which of the two models performs better. Below is the code to my implementation of 8-fold cross validation used to compare the two models. Notice that at each of the 8 iterations, we train two different models on their respective datasets, and record the number of misclassifications that result from using that model on the test set.

Also notice that for the logistic model with linear terms only, we only use x1,x2 and x3 as predictors. And for the logistic model with quadratic terms, we use a model with x1 as the quadratic term. I tried a few different models by slightly adjusting the code and found that both of these performed fairly well in comparison to other comparable models - though they are not necessarily the best selection.

```
#Split the dataset into 8 folds
n2 <- dim(train)[1]
folds <- sample(rep(1:8, length=n2))

#Run 8-Fold Cross validation on the models
Err.mod1 <- NULL
Err.mod2 <- NULL
for (k in 1:8){
  #Define the test/train sets for this iteration
  cvtest <- train[folds==k,]
  cvtrain <- train[folds!=k,]

  #Train a Logistic Regression Model with linear terms only (drop x4 and x5)
  mod1 <- glm(y~x1+x2+x3, data=cvtrain, family="binomial")
  mod2 <- glm(formula = y ~ poly(x1, degree = 2)+x2+x3, data = cvtrain, family = "binomial")

  #Get the predicted probabilities and classification for the test set
  mod1.probs <- predict.glm(mod1, newdata = cvtest, type = "response")
  mod1.preds <- as.numeric(mod1.probs >= 0.5)

  mod2_probs <- predict.glm(mod2, newdata = cvtest, type = "response")
  mod2_predict <- as.numeric(mod2_probs >= 0.5)

  #Count the number of occurrences where the predicted class does not match the observed class
  len <- length(mod1.probs)
  mod1.errors = 0
  mod2.errors = 0
  for(i in 1:len){
    if(mod1.preds[i] != cvtest$y[i]){
      mod1.errors = mod1.errors + 1
    }
    if(mod2_predict[i] != cvtest$y[i]){
      mod2.errors = mod2.errors + 1
    }
  }

  #Record the misclassification rates of each model for this iteration
  Err.mod1[k] <- (mod1.errors / len)
  Err.mod2[k] <- (mod2.errors / len)
}
```

We can then compute the CV score for each of the models to determine which is better for making predictions (lower is better since this means we had fewer overall misclassifications)

```
#Get the CV for the Linear Regression Model with linear terms
nk <- 42
n <- 336

#For the logistic model with only linear terms
CV.mod1 <-  sum((nk/n) * Err.mod1)
CV.mod1 #0.25

#For the logistic model with linear and quadratic terms
CV.mod2 <- sum((nk/n) * Err.mod2)
CV.mod2 #0.217
```

Observe that the model with the quadratic term yielded the lower CV score.This suggests that this model is better at making predictions compared to the other model. We can train and test a similar model on the full training/test sets defined in Part 2.

Part 5

```
#Fit the selected model on the full training data
mod <- glm(y ~ poly(x1, degree = 2)+x2+x3, data = train, family = "binomial")

#Use this model to evaluate the predicted values in the validation set
probs <- predict.glm(mod, newdata = test, type = "response")
preds <- as.numeric(probs >= 0.5)

table(preds, test$y)
```

```
##
## preds  0  1
##     0 55  8
##     1 21 60
```

Furthermore, using the classification matrix above, we can compute the overall misclassification, as well as the false-positive and false-negative rates.

```
#Overall Misclassification Rate
(8 + 21) / 144  #0.2013889

#False Positive Rate
21 / (21 + 55) #0.2763

#False Negative Rate
8 / (8 + 60) #0.1176
```

Note that a False-Positive value of 0.27 may be something that the bank wants to consider. This means that of all of the people who did not default in the last 3 months,we are wrongly predicting their status 30% of the time - that their account defaulted (even though it did not). Depending on how important it is for the bank to correctly classify, we may want to try to consider slight variations of other models considered.

Likewise, a False-negative value of 0.11 means that of the people who actually did default, we are wrongly prediciting their status about 10% of the time - that their account did not default (even though in this case it did). Again, depending on how much such a misclassification would cost the bank, we may want to spend more time developing another model.