Assignment 5 Part II

Matthew Dunne

August 12, 2018

The Data

Use the same training and holdout data sets that you used for logistic regression.

```
setwd("C:/Users/mjdun/Desktop/Data Mining/Assignments")
#using csv which has it in factor variables
MyData <- read.csv(file="German.Credit.csv", header=TRUE, sep=",")
#select just the numeric variables, make Amount the first variable
library(caret)
```

```
## Loading required package: lattice
```

```
## Loading required package: ggplot2
```

```
#split data into train and test as you did in logistic regression assignment
set.seed(1234)
s1<-sample(1:nrow(MyData), nrow(MyData)*.7, replace=FALSE)
train<-MyData[s1, ]
test<-MyData[-s1, ]
```

LDA and QDA

Run LDA using all predictors and predict class (creditability) for train and holdout.

```
library(MASS)
lda.model<-lda(Creditability~., data=train)
#training predictions
lda.train.pred<-predict(lda.model)$class
#test predictions
lda.test.pred<-predict(lda.model, newdata = test)$class
```

Run QDA using all predictors and predict class (creditability) for train and holdout.

```
qda.model<-qda(Creditability~., data=train)
#training predictions
qda.train.pred<-predict(qda.model)$class
#test predictions
qda.test.pred<-predict(qda.model, newdata = test)$class
```

Compare Predictions from LDA and QDA

How often do LDA and QDA make the same predictions. Look at train and test.

First the training data:

```
#train data
table(lda.train.pred, qda.train.pred)
```

```
##                qda.train.pred
## lda.train.pred   0   1
##              0 141  18
```

```
##              1  81 460
```

For the training data, we see that LDA and QDA agree on 601 of the 700 observations. Of the observations where they do not agree the LDA seems to have more of a tendency towards 1. Because 0=Bad and 1=Good we would say that LDA has some tendency to classify people as having good credit compared to QDA.

Now the test data:

```
#test data
table(lda.test.pred, qda.test.pred)
```

```
##              qda.test.pred
## lda.test.pred   0   1
##             0  55  12
##             1  29 204
```

We see much the same dynamic.

Compare Predictions to Actual Values in Train and Holdout

Compare the confusion matrices for LDA (compare train to holdout).

```
round(prop.table(table(train[,1], lda.train.pred), 1),2)
```

```
##    lda.train.pred
##        0    1
##   0 0.51 0.49
##   1 0.11 0.89
```

```
round(prop.table(table(test[,1], lda.test.pred), 1),2)
```

```
##    lda.test.pred
##        0    1
##   0 0.44 0.56
##   1 0.13 0.87
```

There is a slight loss in performance from training to test, at least in terms of detecting Bad Credit. Detecting Good Credit holds up quite well.

Now compare the confusion matrices for QDA (compare train to holdout).

```
round(prop.table(table(train[,1], qda.train.pred), 1),2)
```

```
##    qda.train.pred
##        0    1
##   0 0.70 0.30
##   1 0.15 0.85
```

```
round(prop.table(table(test[,1], qda.test.pred), 1),2)
```

```
##    qda.test.pred
##        0    1
##   0 0.50 0.50
##   1 0.19 0.81
```

QDA suffers from the a steeper drop in performance in detecting Bad Credit. However it is falling from greater heights and does a better job of detecting Bad Credit in the test data relative to LDA.

Ensemble Model

Take the predictions from models you have previously built and apply the majority to rule to generate a another prediction. Include predictions from the Logistic Regression, Tree, LDA, and QDA models.

```
#recreate the predictions from your logistic regression model
LRData <- read.csv(file="German.Credit.csv", header=TRUE, sep=",")
columns<-c(1,2,4,5,7,8,9,10,11,12,13,15,16,17,18,19,20,21)
LRData[columns] <- lapply(LRData[columns], factor)
set.seed(1234)
s1<-sample(1:nrow(LRData), nrow(LRData)*.7, replace=FALSE)
chosen_model<-glm(formula = Creditability ~ Account.Balance + Duration.of.Credit..month. +
    Payment.Status.of.Previous.Credit + Purpose + Credit.Amount +
    Value.Savings.Stocks + Instalment.per.cent + Sex...Marital.Status +
    Guarantors + Duration.in.Current.address + Telephone, family = binomial(link = logit),
    data = LRData[s1, ])
#generate LR predictions for train data
LR.train.pred=chosen_model$fitted.values
LR.train.pred[LR.train.pred>=0.65]=1
LR.train.pred[LR.train.pred<0.65]=0
#generate LR predictions for test data
LR.test.pred<-predict(chosen_model, newdata=LRData[-s1, -1], type="response")
LR.test.pred[LR.test.pred>=0.65]=1
LR.test.pred[LR.test.pred<0.65]=0
#generate predictions from Tree Model, using your pruned tree
library(rpart)
pruned_tree=rpart(LRData[s1, ],control=rpart.control(cp=0.0126984,minsplit=30,xval=10, maxsurrogate=0))
#tree train predictions
tree.train.pred<-predict(pruned_tree,type="class")
#tree test predictions
tree.test.pred<-predict(pruned_tree, newdata=LRData[-s1, -1], type="class")
```

```
#convert predictions from Trees, LDA, and QDA from Factor to numeric so you can do ensembling
library(varhandle)
tree.train.pred<-unfactor(tree.train.pred)
tree.test.pred<-unfactor(tree.test.pred)
lda.train.pred<-unfactor(lda.train.pred)
lda.test.pred<-unfactor(lda.test.pred)
qda.train.pred<-unfactor(qda.train.pred)
qda.test.pred<-unfactor(qda.test.pred)
#create as data frames
train.models<-data.frame(LR.train.pred, tree.train.pred, lda.train.pred, qda.train.pred)
head(lda.test.pred)
```

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

```
head(qda.test.pred)
```

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

```
test.models<-data.frame(LR.test.pred, tree.test.pred, lda.test.pred, qda.test.pred)
```

Create the Ensemble Prediction on the training data.

```
#set value of Ensemble prediction to 0 as default
train.models$Ensemble<-0
#for loop to calculate majority result for train models
```

```
for (i in 1:nrow(train.models)){
  if(sum(train.models[i,])==3 || sum(train.models[i,])==4)
    {train.models$Ensemble[i]<-1}
  if(sum(train.models[i,]==2))
    {train.models$Ensemble[i]=sample(0:1,1)}
}
head(train.models)
```

```
##     LR.train.pred tree.train.pred lda.train.pred qda.train.pred Ensemble
## 114             1               1              1              1        1
## 622             1               1              1              0        1
## 609             0               1              0              0        0
## 999             1               1              1              0        1
## 858             0               0              0              0        0
## 638             1               1              1              1        1
```

And create the Ensemble Prediction on the test data.

```
#set value of Ensemble prediction to 0 as default
test.models$Ensemble<-0
#for loop to calculate majority result for train models
for (i in 1:nrow(test.models)){
  if(sum(test.models[i,])==3 || sum(test.models[i,])==4)
    {test.models$Ensemble[i]<-1}
  if(sum(test.models[i,]==2))
    {test.models$Ensemble[i]=sample(0:1,1)}
}
head(test.models)
```

```
##    LR.test.pred tree.test.pred lda.test.pred qda.test.pred Ensemble
## 4             0              1             1             1        1
## 5             0              1             1             1        1
## 7             1              1             1             1        1
## 8             1              1             1             1        1
## 11            0              1             1             1        1
## 15            1              1             1             1        1
```

Confusion Matrices Comparing Ensemble Model Across Train and Holdout

First look at how well the Ensemble Model predicts the actual values in the train data:

```
round(prop.table(table(train[,1], train.models$Ensemble), 1),2)
```

```
##
##        0    1
##   0 0.70 0.30
##   1 0.14 0.86
```

Then see how the Ensemble Model predicts actual values on the test data:

```
round(prop.table(table(test[,1], test.models$Ensemble), 1),2)
```

```
##
##        0    1
##   0 0.52 0.48
##   1 0.19 0.81
```

We see that just for the Ensemble Model the prediction rate holds up fairly well for Good Credit (1) but not very well for Bad Credit (0). The percentage for the latter falls from 70% to 52%.

Comparing the Ensemble Model with Other Models

For comparison, here are the confustion matrices for the Linear Regression:

```
round(prop.table(table(train[,1], LR.train.pred), 1),2)
```

```
##    LR.train.pred
##        0    1
##   0 0.73 0.27
##   1 0.20 0.80
```

```
round(prop.table(table(test[,1], LR.test.pred), 1),2)
```

```
##    LR.test.pred
##        0    1
##   0 0.61 0.39
##   1 0.25 0.75
```

and for the Tree Model:

```
round(prop.table(table(train[,1], tree.train.pred), 1),2)
```

```
##    tree.train.pred
##        0    1
##   0 0.54 0.46
##   1 0.09 0.91
```

```
round(prop.table(table(test[,1], tree.test.pred), 1),2)
```

```
##    tree.test.pred
##        0    1
##   0 0.48 0.52
##   1 0.14 0.86
```

For correctly detecting Bad Ratings, the Ensemble Model performs almost exactly the same as QDA in the holdout (see p. 2). The Tree Model was inferior to QDA, though slightly better than LDA.

The Logistic Regression Model was superior to all other models in both train and test data in detecting Bad Ratings, yet worse overall in detecting Good Ratings. This is almost certainly for one simple reason, I calibrated the threshold in the Logistic Regression Model to have have a bias toward Bad Ratings, i.e. I set the probability threshold to classify as 1 at 0.65 instead of 0.5. The other models did not include this calibration and so did not perform as well here.

So an Ensemble Model of LDA, QDA, and Tree might make sense as they might compensate for each other's weaknesses, but it would not be wise to dilute the predictive power of the Logistic Regression model by averaging it with other models that are, at least in this instance, inferior.