# Data Mining - Assignment 4: Part 1

*Ryan Durfey*

*Friday, March 13, 2015*

### Data Preparation

```
sed<-8675309
set.seed(sed)

source("./clustreg.R")
source("./clustreg.predict.R")

## load back in the Training and Test/Holdout datasets from the Logistic Regression
## assignment
GC_train<-readRDS("C:/Misc Docs/UChicago/DataMining/Data_Mining/GC_train.rds")
GC_test<-readRDS("C:/Misc Docs/UChicago/DataMining/Data_Mining/GC_test.rds")

## subset to get numeric variables, with Amount in the first column
GC_train_num<-GC_train[,c(2:8)]
GC_train_num<-cbind(Amount=GC_train_num[,2],GC_train_num[,-2])
#str(GC_train_num)

GC_test_num<-GC_test[,c(2:8)]
GC_test_num<-cbind(Amount=GC_test_num[,2],GC_test_num[,-2])
#str(GC_test_num)
```

### Clusterwise Regression Models

```
## 1 cluster model
clustreg.1<-clustreg(GC_train_num,k=1,tries=24,sed=sed,niter=10)
clustreg.1$rsq.best
```

```
## [1] 0.5143132
```

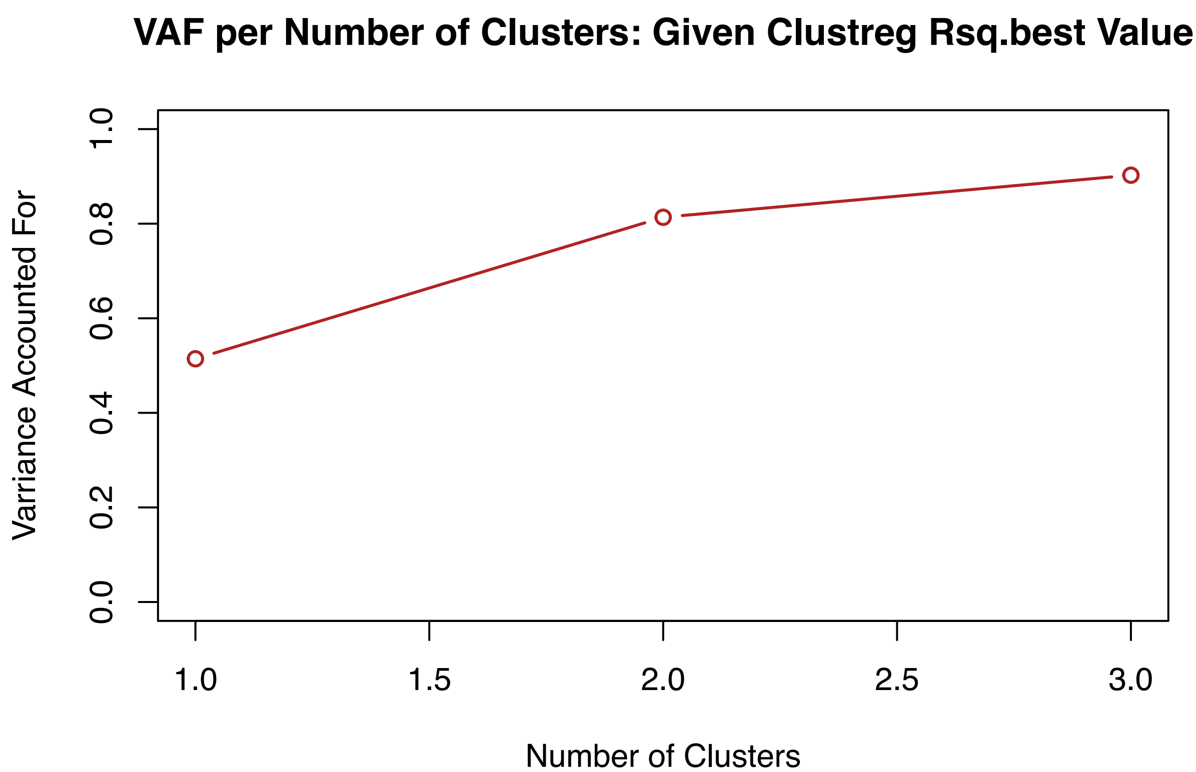
```
## 2 cluster model
clustreg.2<-clustreg(GC_train_num,k=2,tries=24,sed=sed,niter=10)
clustreg.2$rsq.best
```

```
## [1] 0.8135473
```

```
## 3 cluster model
clustreg.3<-clustreg(GC_train_num,k=3,tries=24,sed=sed,niter=10)
clustreg.3$rsq.best
```

```
## [1] 0.9025141
```

```r
## plot best R-squared values as function of number of clusters
plot(c(1,2,3),c(clustreg.1$rsq.best,clustreg.2$rsq.best,clustreg.3$rsq.best),type="b",
     ylim=c(0,1),lwd=1.5,col="firebrick",
     main="VAF per Number of Clusters: Given Clustreg Rsq.best Value",
     ylab="Varriance Accounted For",xlab="Number of Clusters")
```




```r
## NOTE: the rsq.best values returned by clustreg() seem to not coincide with the values
## actually observed in the model outputs. For thoroughness, we'll manually pull the
## maximum r.sqaured values from each model in each cluster solution.

## alternate plot, manually taking the maximum R-squared value from each cluster model
plot(c(1,2,3),c(clustreg.1$results[[1]]$r.squared,
                max(clustreg.2$results[[1]]$r.squared,clustreg.2$results[[2]]$r.squared),
                max(clustreg.3$results[[1]]$r.squared,clustreg.3$results[[2]]$r.squared,
                    clustreg.3$results[[3]]$r.squared)),type="b",ylim=c(0,1),lwd=1.5,
     col="firebrick",main="VAF per Number of Clusters: Manually Pulled from Model Results",
     ylab="Varriance Accounted For",xlab="Number of Clusters")
```


```
## NOTE: Even though the R-squared values each plot are indeed slightly different,
## the conclusion is the same. The 3-cluster model provides the highest R-squared values.


## proportion tables of the clusters from the 2 and 3 cluster models
table(clustreg.2$cluster)
```

```
##
##   1   2
## 162 538
```

```
round(prop.table(table(clustreg.2$cluster)),3)
```

```
##
##     1     2
## 0.231 0.769
```

```
table(clustreg.3$cluster)
```

```
##
##   1   2   3
## 271 357  72
```

```r
round(prop.table(table(clustreg.3$cluster)),3)
```

```
##
##     1     2     3
## 0.387 0.510 0.103
```

From the plots, the Variance Accounted For is highest for the 3-cluster solution. Thus, we may suspect that it is the best candidate to choose. However, we still need to also consider the holdout validation.

## Holdout Validation

```r
## 1 cluster model
hold.clust1<-clustreg.predict(clustreg.1,GC_test_num)
hold.clust1$rsq
```

```
## [1] 0.4639696
```

```r
## note: 100% of the data will be in the 1 cluster, so a prop.table is not needed

## 2 cluster model
hold.clust2<-clustreg.predict(clustreg.2,GC_test_num)
hold.clust2$rsq
```

```
## [1] 0.8157452
```

```r
table(hold.clust2$cluster)
```

```
##
##   1   2
##  64 236
```

```r
round(prop.table(table(hold.clust2$cluster)),3)
```

```
##
##     1     2
## 0.213 0.787
```

```r
## 3 cluster model
hold.clust3<-clustreg.predict(clustreg.3,GC_test_num)
hold.clust3$rsq
```

```
## [1] 0.8795333
```

```r
table(hold.clust3$cluster)
```

```
##
##   1   2   3
##  99 163  38
```

```r
round(prop.table(table(hold.clust3$cluster)),3)
```

```
##
##     1     2     3
## 0.330 0.543 0.127
```

Holdout validation looks to perform fairly well in each of the models because we can see that the R-squared values as well as the cluster proportions are quite similar to those produced from the training set.

## Model Selection and Interpretation

In the end, we'll choose the 3-cluster model as the best and most appropriate. This is based on (1) the high R-squared values observed in each linear model within each cluster, (2) the holdout does well and shows similar proportion tables and best R-squared values across both training and test data, and (3) interpretability of clusters based on variable significance. To help explain that last reason, we will look at the results of the 3-cluster model.

```r
clustreg.3$results
```

```
## [[1]]
##
## Call:
## lm(formula = dat[c.best == i, ])
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -1340.0  -464.7  -121.9   387.9  1871.8
##
## Coefficients:
##                            Estimate Std. Error t value Pr(>|t|)
## (Intercept)               4209.1116   261.9784  16.067  < 2e-16 ***
## Duration                   146.7526     3.3931  43.251  < 2e-16 ***
## InstallmentRatePercentage -1025.5162    36.6011 -28.019  < 2e-16 ***
## ResidenceDuration         -134.6995    37.6576  -3.577 0.000413 ***
## Age                         -0.4568     3.4707  -0.132 0.895383
## NumberExistingCredits     -221.6068    73.3585  -3.021 0.002768 **
## NumberPeopleMaintenance    325.2883   110.0270   2.956 0.003394 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 653.5 on 264 degrees of freedom
## Multiple R-squared:  0.9193, Adjusted R-squared:  0.9174
## F-statistic:   501 on 6 and 264 DF,  p-value: < 2.2e-16
##
##
## [[2]]
##
## Call:
## lm(formula = dat[c.best == i, ])
##
## Residuals:
```

```
##     Min      1Q  Median      3Q     Max
## -1984.0  -339.7    18.6   356.3  1735.8
##
## Coefficients:
##                           Estimate Std. Error t value Pr(>|t|)
## (Intercept)               1699.655    173.853   9.776  < 2e-16 ***
## Duration                    90.539      3.020  29.982  < 2e-16 ***
## InstallmentRatePercentage -497.888     28.304 -17.590  < 2e-16 ***
## ResidenceDuration          -93.781     28.857  -3.250  0.00127 **
## Age                         -1.043      3.063  -0.340  0.73370
## NumberExistingCredits        7.543     57.568   0.131  0.89583
## NumberPeopleMaintenance    159.789     90.844   1.759  0.07946 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 588.2 on 350 degrees of freedom
## Multiple R-squared:  0.7438, Adjusted R-squared:  0.7394
## F-statistic: 169.4 on 6 and 350 DF,  p-value: < 2.2e-16
##
##
## [[3]]
##
## Call:
## lm(formula = dat[c.best == i, ])
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2365.0 -1193.6  -438.2   381.4  6347.3
##
## Coefficients:
##                           Estimate Std. Error t value Pr(>|t|)
## (Intercept)               10290.28    1364.39   7.542 1.90e-10 ***
## Duration                    186.99      16.93  11.045  < 2e-16 ***
## InstallmentRatePercentage -1532.49     216.31  -7.085 1.23e-09 ***
## ResidenceDuration            65.28     242.17   0.270 0.788355
## Age                         -19.55      21.25  -0.920 0.360857
## NumberExistingCredits     -1116.47     295.33  -3.780 0.000343 ***
## NumberPeopleMaintenance    -236.41     604.04  -0.391 0.696793
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1930 on 65 degrees of freedom
## Multiple R-squared:  0.768,  Adjusted R-squared:  0.7466
## F-statistic: 35.87 on 6 and 65 DF,  p-value: < 2.2e-16
```

These 3 clusters have some similarities. However, each cluster has certain some identifying qualities that can definte it. For instanced, cluster 1 is the only one with significance for NumberPeopleMaintenance, so it helps us interpret that model apart from the others. Similarly, based on significance level and estimate value, cluster 3 is more heavily influenced by NumberExistingCredits and InstallmentRatePercentage than the other clusters. Cluster 2 is a little trickier, but in comparing the intercepts of the clusters, we see that it is by far the lowest. This likely means that it comprises the lowest loan amounts. In contrast, cluster 3's intercept is over 6 times that of cluster 2.

The existence of these kinds of facets are important because they help us interpret our models. To further illustrate how it influenced our overall model selection, we can look at the 2-cluster model.

```r
clustreg.2$results
```

```
## [[1]]
##
## Call:
## lm(formula = dat[c.best == i, ])
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3018.5 -1024.6  -509.9   453.3  9663.9
##
## Coefficients:
##                           Estimate Std. Error t value Pr(>|t|)
## (Intercept)                5086.80     913.08   5.571 1.09e-07 ***
## Duration                    193.49      11.23  17.223  < 2e-16 ***
## InstallmentRatePercentage -1002.29     131.93  -7.597 2.68e-12 ***
## ResidenceDuration           259.85     141.80   1.832   0.0688 .
## Age                         -13.00      13.12  -0.990   0.3235
## NumberExistingCredits      -324.11     230.57  -1.406   0.1618
## NumberPeopleMaintenance    -251.03     368.15  -0.682   0.4963
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1873 on 155 degrees of freedom
## Multiple R-squared:  0.7104, Adjusted R-squared:  0.6992
## F-statistic: 63.38 on 6 and 155 DF,  p-value: < 2.2e-16
##
##
## [[2]]
##
## Call:
## lm(formula = dat[c.best == i, ])
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2683.5  -483.8    -8.6   475.4  3179.6
##
## Coefficients:
##                           Estimate Std. Error t value Pr(>|t|)
## (Intercept)               1419.233    207.908   6.826 2.39e-11 ***
## Duration                   107.894      3.378  31.941  < 2e-16 ***
## InstallmentRatePercentage -495.913     33.153 -14.958  < 2e-16 ***
## ResidenceDuration           29.694     34.030   0.873    0.383
## Age                          2.509      3.444   0.729    0.467
## NumberExistingCredits      -24.950     64.839  -0.385    0.701
## NumberPeopleMaintenance     86.142    107.495   0.801    0.423
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 840.7 on 531 degrees of freedom
## Multiple R-squared:  0.6846, Adjusted R-squared:  0.681
## F-statistic: 192.1 on 6 and 531 DF,  p-value: < 2.2e-16
```

In the 2-cluster model, interpreting the clusters is much more difficult because they are very few distinguishing characteristics between them. Each cluster has the same significant variables and there isn't as wide of a range between the intercepts as we observed with the 3-cluster model. Thus, it lends more confirmation to our choice of the 3-cluster model.

# Data Mining - Assignment 4: Part 2

*Ryan Durfey*

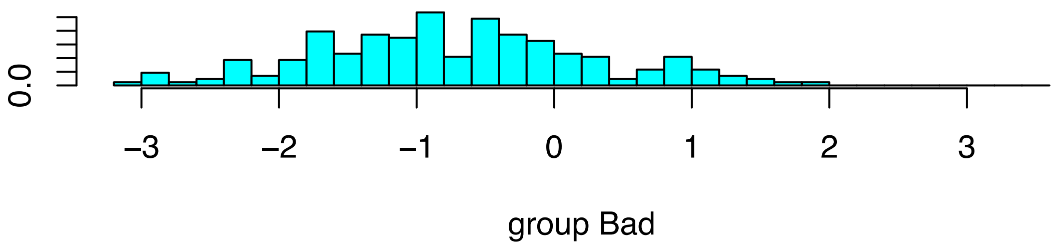
*Friday, March 13, 2015*

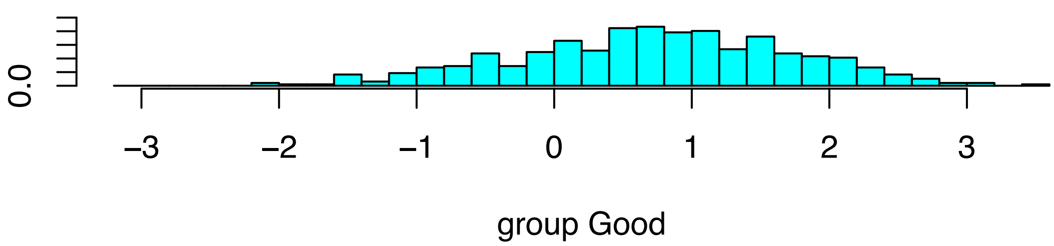
### The Data

```
set.seed(8675309)
library(MASS)
library(rpart)
library(functional)

## load back in the Training and Test/Holdout datasets from the Logistic Regression
## and Classification Tree assignments
GC_train<-readRDS("C:/Misc Docs/UChicago/DataMining/Data_Mining/GC_train.rds")
GC_test<-readRDS("C:/Misc Docs/UChicago/DataMining/Data_Mining/GC_test.rds")
```

### LDA and QDA Models

```
### LDA ####
model.LDA<-lda(Class~.,data=GC_train)

plot(model.LDA)
```




```
## We don't see a stark distinction between the two groups here and there is some overlap
## however, it isn't terrible

## LDA confusion matrix for training set
table(GC_train$Class,predict(model.LDA)$class)
```

```
##
##        Bad Good
##   Bad  119   96
##   Good  56  429
```

```
round(prop.table(table(Actual=GC_train$Class,Predicted=predict(model.LDA)$class),1),2)
```

```
##       Predicted
## Actual  Bad Good
##   Bad  0.55 0.45
##   Good 0.12 0.88
```

The LDA model seems to predict Good values fairly well, but struggles a bit with Bad ones. Accurately predicting Bad values 55% of the time isn't the worst in the world, but it's not the greatest either. Let's try QDA to see if we get better results there.

```r
#### QDA ####
## note: the variable #13, Purpose, causes issues with QDA, so it is removed for this model
model.QDA<-qda(Class~Duration+Amount+InstallmentRatePercentage+ResidenceDuration+Age
               +NumberExistingCredits+NumberPeopleMaintenance+Telephone+ForeignWorker
               +CheckingAccountStatus+CreditHistory+SavingsAccountsBonds+EmploymentDuration
               +PersonalStatus+OtherDebtorsGuarantors+Property+OtherInstallmentPlans
               +Housing+Job,data=GC_train)

## QDA confusion matrix for training set
table(GC_train$Class,predict(model.QDA)$class)
```

```
##
##        Bad Good
##   Bad  149   66
##   Good  67  418
```

```r
round(prop.table(table(Actual=GC_train$Class,Predicted=predict(model.QDA)$class),1),2)
```

```
##       Predicted
## Actual  Bad Good
##   Bad  0.69 0.31
##   Good 0.14 0.86
```

Excellent! The QDA model does indeed do a better job of predicting Bad values. Again, it's not perfect, but it isn't bad either. ## Even if we don't have a lot of faith in these models, we'll still run holdout validations on both of them.

### Holdout Validation on LDA and QDA Models

```r
## LDA holdout
round(prop.table(table(Actual=GC_test$Class,
                       Predicted=predict(model.LDA,newdata=GC_test[,-1])$class),1),2)
```

```
##       Predicted
## Actual  Bad Good
##   Bad  0.52 0.48
##   Good 0.15 0.85
```

```r
## QDA Holdout
round(prop.table(table(Actual=GC_test$Class,
                       Predicted=predict(model.QDA,newdata=GC_test[,-c(1)])$class),1),2)
```

```
##       Predicted
## Actual  Bad Good
##   Bad  0.56 0.44
##   Good 0.24 0.76
```

Our models perform decently enough in Holdout validation. The LDA model faired better, when compared to its training set. The QDA model shows a definite drop in proportions of both Good and Bad accurate predictions compared to its training set. Due to this, choosing between just these two models would have to include contextual considerations such as whether it is more important to accurately predict Good or Bad values.

## Ensemble Model: Training Set

Here, we aggregate our predictions from each of the four methods: Logistic Regression, Classification Tree, LDA, and QDA. These models were created in the previous assignment, and so some of the code is commented out to avoid verbosity.

### Logistic Model

```
model.full<-glm(GC_train,family=binomial(link=logit))
#drop1(model.full)
## take out appropriate variables found from drop1()
sig.names<-c("ResidenceDuration","Age","NumberPeopleMaintenance","Telephone",
             "ForeignWorker","SavingsAccountsBonds","Property",
             "OtherInstallmentPlans","Housing","Job")
model.logistic<-glm(GC_train[,!names(GC_train)%in%sig.names],family=binomial(link=logit))
## retrieve predicted values using chosen probability threshold
prob.thresh<-0.5
logistic.train.pred<-predict(model.logistic,type="response")
logistic.train.pred[logistic.train.pred>=prob.thresh]<-"Good"
logistic.train.pred[logistic.train.pred<prob.thresh]<-"Bad"
```

### RPART Classification Tree

```
tree.full<-rpart(GC_train,control=rpart.control(cp=0,minsplit=30,xval=10,maxsurrogate=0))
#printcp(tree.full)
#plotcp(tree.full)
## recreate tree with cp=0.0139535
model.tree<-rpart(GC_train,control=rpart.control(cp=0.0139535,minsplit=30,xval=10,
                                                 maxsurrogate=0))
#plot(model.tree,col=3,compress=TRUE,branch=0.2,uniform=TRUE)
#text(model.tree,cex=.6,col=4,use.n=TRUE,fancy=TRUE,fwidth=.4,fheight=.4,bg=c(5))
## retrieve predicted values
tree.train.pred<-predict(model.tree,type="class")
```

### LDA

```
## LDA - use model from above
LDA.train.pred<-predict(model.LDA)$class
```

### QDA

```
## QDA - use model from above
QDA.train.pred<-predict(model.QDA)$class
```

**Overall Predicted Output Based on Majority**

```
## combine them together into one data frame
GC_train_outputs<-data.frame(Log.Pred=logistic.train.pred,Tree.Pred=tree.train.pred,
                             LDA.pred=LDA.train.pred,QDA.Pred=QDA.train.pred)

## applying majority rule to get a final prediction decision
###### function to use majority rule & random selection for ties ######
majority.rule<-function(x){
        majority<-apply(x,1,Compose(table,function(i) i==max(i),which,names))
        majority<-lapply(majority,function(x){ ifelse(length(x)>1,
                                               sample(c("Bad","Good"),1),x)})

        majority<-factor(unlist(majority))
}

Majority.Pred<-majority.rule(GC_train_outputs)

## confusion matrices for training data & predicted values from ensemble model
table(GC_train$Class,Majority.Pred)
```

```
##       Majority.Pred
##        Bad Good
##   Bad  125   90
##   Good  51  434
```

```
round(prop.table(table(Actual=GC_train$Class,Predicted=Majority.Pred),1),2)
```

```
##       Predicted
## Actual  Bad Good
##   Bad  0.58 0.42
##   Good 0.11 0.89
```

The confusion matrices for the Ensemble Model's peformance on the training set shows us that it is very good at predicting Good output values, but only has a 58% chance at accurately predicting Bad values. This is an interesting finding, because most of the individual models were at a similar level of predicting these values. Not by much, mind you, but it at least appears that using the Ensemble Model approach does not automatically guarantee us better results than using just one individual model.

## Holdout Validation

```
##logistic holdout pred
log.ho<-predict(model.logistic,newdata=GC_test[,-1],type="response")
log.ho[log.ho>=prob.thresh]<-"Good"
log.ho[log.ho<prob.thresh]<-"Bad"

## tree holdout pred
tree.ho<-predict(model.tree,newdata=GC_test[,-1],type="class")
```

```
## LDA holdout pred
LDA.ho<-predict(model.LDA,newdata=GC_test[,-1])$class

## QDA holdout pred
QDA.ho<-predict(model.QDA,newdata=GC_test[,-1])$class

## put them together & apply majority rule
GC_test_outputs<-data.frame(Log.Pred=log.ho,Tree.Pred=tree.ho,LDA.pred=LDA.ho,
                            QDA.Pred=QDA.ho)
Majority.HO.Pred<-majority.rule(GC_test_outputs)

## confusion matrix for holdout validation
table(GC_test$Class,Majority.HO.Pred)
```

```
##       Majority.HO.Pred
##         Bad Good
##   Bad    43   42
##   Good   35  180
```

```
round(prop.table(table(Actual=GC_test$Class,Predicted=Majority.HO.Pred),1),2)
```

```
##       Predicted
## Actual  Bad Good
##   Bad  0.51 0.49
##   Good 0.16 0.84
```

Over all, our Ensemble Model performs about as well as most of the individual models/methods that are incorporated in it. However, due to the higher complexity and coding that goes into the Ensemble Model, this is a rather unsatisfying conclusion. By using one of the individual models, we could essentially achieve the same level of prediction accuracy with a much lower cost of time and effort.

For comparison, here are the confusion matrices for each individual model on the training and holdout datasets.

```
## logistic regression model
round(prop.table(table(GC_train$Class,logistic.train.pred),1),2)
```

```
##       logistic.train.pred
##         Bad Good
##   Bad  0.54 0.46
##   Good 0.10 0.90
```

```
round(prop.table(table(GC_test$Class,log.ho),1),2)
```

```
##       log.ho
##         Bad Good
##   Bad  0.53 0.47
##   Good 0.18 0.82
```

```r
## rpart tree classification model
round(prop.table(table(GC_train$Class,tree.train.pred),1),2)
```

```
##       tree.train.pred
##         Bad Good
##   Bad  0.52 0.48
##   Good 0.08 0.92
```

```r
round(prop.table(table(GC_test$Class,tree.ho),1),2)
```

```
##       tree.ho
##         Bad Good
##   Bad  0.40 0.60
##   Good 0.19 0.81
```

```r
## LDA model
round(prop.table(table(GC_train$Class,LDA.train.pred),1),2)
```

```
##       LDA.train.pred
##         Bad Good
##   Bad  0.55 0.45
##   Good 0.12 0.88
```

```r
round(prop.table(table(GC_test$Class,LDA.ho),1),2)
```

```
##       LDA.ho
##         Bad Good
##   Bad  0.52 0.48
##   Good 0.15 0.85
```

```r
## QDA model
round(prop.table(table(GC_train$Class,QDA.train.pred),1),2)
```

```
##       QDA.train.pred
##         Bad Good
##   Bad  0.69 0.31
##   Good 0.14 0.86
```

```r
round(prop.table(table(GC_test$Class,QDA.ho),1),2)
```

```
##       QDA.ho
##         Bad Good
##   Bad  0.56 0.44
##   Good 0.24 0.76
```

Based on the above confusion matrices, the only model that does noticeably worse than the Ensemble Model is the Classification Tree with the holdout data. Therefore, it may be a wise decision to ultimately choose the Logistic Regression Model, LDA Model, or QDA Model over the Ensemble Model if we are dealing with similar data in the future.

### Additional Remarks about Logistic Regression

Similar to when we created the Logistic Regression Model in the previous assignment, we can again modify the probability threshold when assigning Good and Bad values to the predicted values.

```
prob.thresh2<-0.75

logistic.train.pred2<-predict(model.logistic,type="response")
logistic.train.pred2[logistic.train.pred2>=prob.thresh2]<-"Good"
logistic.train.pred2[logistic.train.pred2<prob.thresh2]<-"Bad"

log.ho2<-predict(model.logistic,newdata=GC_test[,-1],type="response")
log.ho2[log.ho2>=prob.thresh2]<-"Good"
log.ho2[log.ho2<prob.thresh2]<-"Bad"

round(prop.table(table(GC_train$Class,logistic.train.pred2),1),2)
```

```
##       logistic.train.pred2
##         Bad Good
##   Bad  0.81 0.19
##   Good 0.34 0.66
```

```
round(prop.table(table(GC_test$Class,log.ho2),1),2)
```

```
##       log.ho2
##         Bad Good
##   Bad  0.73 0.27
##   Good 0.36 0.64
```

Since the Logistic Regression Model gives us the freedom to change the probability threshold of assigning Good and Bad values, we can obtain better results than what we otherwise have seen in other models. Here, with a 0.75 threshold, we now have a much better prediction accuracy of Bad output values. Due to this additional aspect of the Logistic Regression Model, if I had to select just one, this would be my model of choice.