Assignment 3

Group 4

1) Import the pre-processed dataset in R. Shuffle the records and split them into a training set (20,000 records), a validation set (8,000 records) and a test set (all remaining records).

```
require(C50)
require(gmodels)
require(ggplot2)
```

```
df<-read.csv("Loans_processed.csv")

set.seed(2017)
sampling <- sample(nrow(df))
df <- df[sampling,]

train <- df[1:20000,]
val <- df[20001:28000,]
test <- df[28001:nrow(df),]

train.x <- train[,1:7]
train.y <- train[,8]

val.x <- val[,1:7]
val.y <- val[,8]

test.x <- test[,1:7]
test.y <- test[,8]
```

2) Using a classification tree, try to predict with an accuracy greater than loans_repaid / (all loans). Do you manage to achieve this performance on the training set?

Expected performance if we predicted all loans to be repaid.

```
baseline <- sum(df$loan_status == "Fully Paid")/nrow(df)
```

Our baseline predicted accuracy is 0.8596128. This corresponds to us predicting all loans as repaid.

By using the default classification tree on the training data, we end up with a tree that has a single root node. We predict all the records to be Fully Paid loans. On our training set, we have an accuracy of .858, which is slightly below the baseline accuracy. However this is only due to small fluctuations in the proportion of charged off to fully paid loans in the training set compared to all of the data in total.

```
loanTree <- C5.0(x = train.x, y = train.y)
loanTreetrain.results <- predict(loanTree, train)

CrossTable(train$loan_status, loanTreetrain.results,
           prop.chisq = FALSE, prop.c = FALSE,
           prop.r = FALSE,
           dnn = c('actual default', 'predicted default'))
```

```
##
##
##    Cell Contents
## |-------------------------|
## |                       N |
## |         N / Table Total |
## |-------------------------|
##
##
## Total Observations in Table:  20000
##
##
##                    | loanTreetrain.results
## train$loan_status | Fully Paid |  Row Total |
## ------------------|------------|------------|
##       Charged Off |       2836 |       2836 |
##                   |      0.142 |            |
## ------------------|------------|------------|
##        Fully Paid |      17164 |      17164 |
##                   |      0.858 |            |
## ------------------|------------|------------|
##      Column Total |      20000 |      20000 |
## ------------------|------------|------------|
##
##
```

When applying our decision tree to the validation set, we have an accuracy of .865 which is slightly greater than the baseline accuracy of .8596. However, yet again this is only due to random fluctuations in the proportion of charged off to fully paid loans across the validation set compared to both the training set and the entire data as a whole. We have not improved on our accuracy performance at all in the validation set by predicting all loans as fully paid.

```
loanTreevalidate <- predict(loanTree, val)

CrossTable(val$loan_status, loanTreevalidate,
           prop.chisq = FALSE, prop.c = FALSE,
           prop.r = FALSE,
           dnn = c('actual default', 'predicted default'))
```

```
##
##
##    Cell Contents
## |-------------------------|
## |                       N |
## |         N / Table Total |
## |-------------------------|
##
##
## Total Observations in Table:  8000
##
##
##                  | loanTreevalidate
## val$loan_status | Fully Paid |  Row Total |
## ----------------|------------|------------|
##     Charged Off |       1149 |       1149 |
##                 |      0.144 |            |
## ----------------|------------|------------|
##      Fully Paid |       6851 |       6851 |
##                 |      0.856 |            |
## ----------------|------------|------------|
##    Column Total |       8000 |       8000 |
## ----------------|------------|------------|
##
##
```

3) Building a cost matrix parameter to measure recall and precision rather than accuracy.

We wish to minimize loans that we predicted to be paid but are actually charged off. Thus we apply a harsher penalty for loans that are in that category. We wish to minimize n_{12} while also retaining high accuracy. Therefore, we try each potential cost value on that cell of the classification matrix in order to meet our desired sensitivity. We set the costs for correctly classifying loans as 0 and iterate through various cost values for giving credit to someone who will default.

```
LoanTreeCost <- function (train.x, train.y, val.x, val.y, stepwise=0.1, Max=5) {
  s <- c()
  p <- c()
  for (i in seq(1,Max,stepwise)) {
    loanTree <- C5.0(x = train.x, y = train.y, costs = matrix(c(0,i,1,0),2,2))
    pred <- predict(loanTree,val.x)
    n11 <- sum(pred == "Charged Off" & val.y == "Charged Off")
    n12 <- sum(pred == "Fully Paid" & val.y == "Charged Off")
    sensitivity <- n11/(n11+n12)
    s <- append(s,sensitivity)

    n21 <- sum(pred == "Charged Off" & val.y == "Fully Paid")
    precision <- n11/(n11+n21)
    p <- append(p,precision)
  }
  return(list(s,p))
}

params <- LoanTreeCost(train.x,train.y,val.x,val.y,Max=10)
sensitivity <- params[[1]]
precision <- params[[2]]
```

```
params <- as.data.frame(params)
colnames(params) <- c("sensitivity","precision")
params$cost <- (as.numeric(rownames(params)) - 1)* 0.1 + 1
```

To find a sensitivity of 25%, we see which trial cost matrix has a sensitivity closest to 25%.

```
sensitivity.25 <- which.min(abs(sensitivity-0.25))
cost.25 <- 1+0.1*(sensitivity.25 - 1)
sensitivity.25.value <- sensitivity[sensitivity.25]
precision.25.value <- precision[sensitivity.25]
```

This occurs when the cost parameter on n_{12} is set to 3. This means we penalize giving someone a loan that actually defaults 3 times as much missing out on a profitable loan. Such a cost parameter gives us a sensitivity of 0.2689295. The percentage of loans we would recommend to the bank for re-evaluation that were indeed charged off is 0.2727273.

To find a sensitivity of 40%, we see which trial cost matrix has a sensitivity closest to 40%.

```
sensitivity.40 <- which.min(abs(sensitivity-0.4))
cost.40 <- 1+0.1*(sensitivity.40 - 1)
sensitivity.40.value <- sensitivity[sensitivity.40]
precision.40.value <- precision[sensitivity.40]
```

This occurs when the cost parameter on n_{12} is set to 3.6. This means we penalize giving someone a loan that actually defaults 3.6 times as much missing out on a profitable loan. Such a cost parameter gives us a sensitivity of 0.4177546. The percentage of loans we would recommend to the bank for re-evaluation that were indeed charged off is 0.2302158.

To find a sensitivity of 50%, we see which trial cost matrix has a sensitivity closest to 50%.

```
sensitivity.50 <- which.min(abs(sensitivity-0.50))
cost.50 <- 1+0.1*(sensitivity.50 - 1)
sensitivity.50.value <- sensitivity[sensitivity.50]
precision.50.value <- precision[sensitivity.50]
```

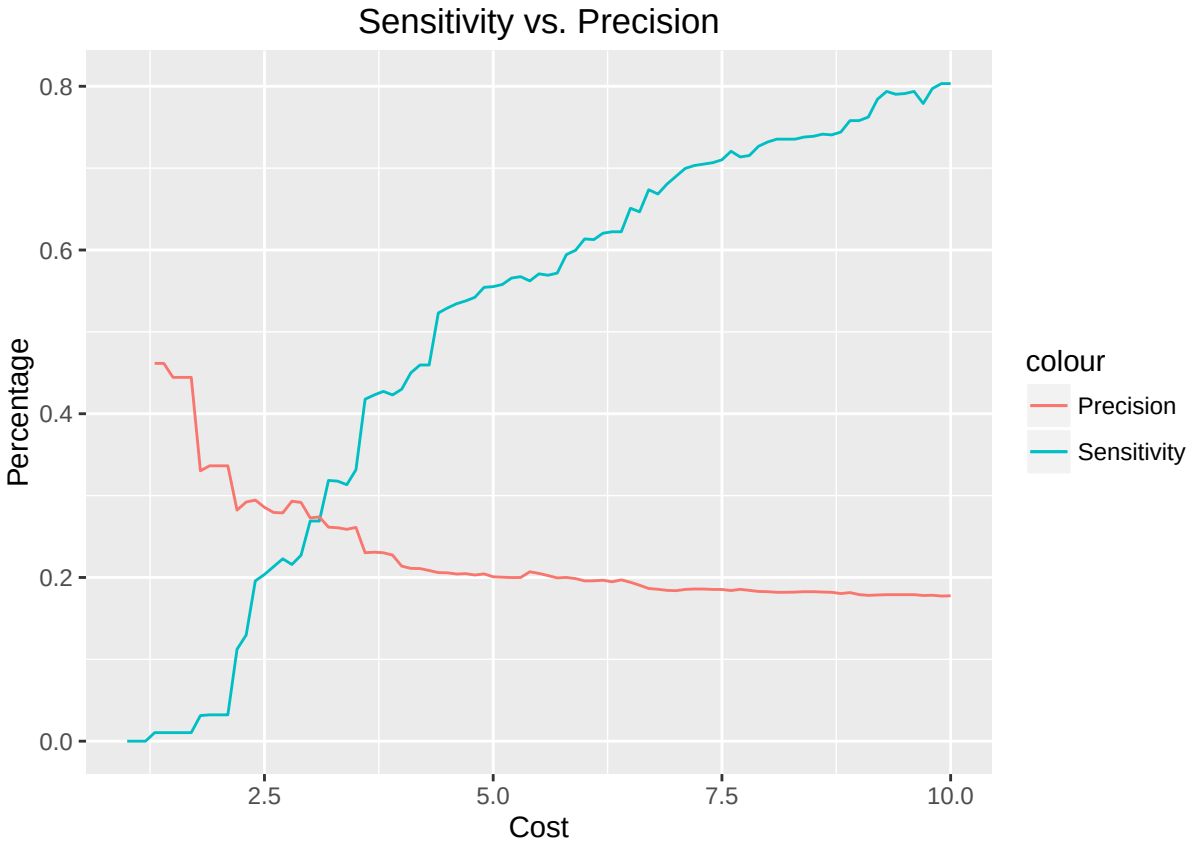
This occurs when the cost parameter on n_{12} is set to 4.4. This means we penalize giving someone a loan that actually defaults 4.4 times as much missing out on a profitable loan. Such a cost parameter gives us a sensitivity of 0.5230635. The percentage of loans we would recommend to the bank for re-evaluation that were indeed charged off is 0.205963.

4) Pick a cost parameter matrix that you assess as the most appropriate for identifying loan applications that deserve further examination.

In general, we would want a cost matrix that has a high degree of precision (few false positives) and also a high degree of sensitivity (few false negatives). Unfortunately, as we increase the cost of false negatives, we have more false positives. Therefore, our optimal level for the cost parameter on false negatives depends on our preferred risk tolerance and whether we would want more false positives or false negatives. Based on the plot below, there exist two potential avenues for the bank to place its cost parameter at. First, the bank could place its cost parameter where its sensitivity is greater than 50%. At this value, the sensitivity has been increasing very fast before slowing in growth. At this point the precision is still relatively high. This option is preferable to the bank if it wants a high precision without compromizing its sensitivity. The cost parameter at this value is 4.4. In this case, the bank would really hate false positives where it predicted paid loans that were in reality charged off.

Alternatively, we notice that the precision value decreases much slower than the sensitivity value increases. If the bank values a very high sensitivity and can afford to be less precise (have some customers that are good be filtered out), then the optimal cost parameter would be very high. We would only accept customers who we are extremely sure about being paid customers. This avenue works because the precision value decreases very slowly as we increase cost. However, the sensitivity parameter increases much faster, so we can improve our sensitivity without greatly compromising our precision. The cost parameter at this value is very high. We could use 10, or even a higher number to sensure that we have very few false negatives. In this case, the bank would really hate false negatives, where it would hate to predict a charged off customer that in reality would have been a source of revenue through a paid loan.

```
ggplot(params, aes(x = cost)) +
  geom_line(aes(y = sensitivity, color = "Sensitivity")) +
  geom_line(aes(y = precision, color = "Precision")) +
  ggtitle("Sensitivity vs. Precision") +
  ylab("Percentage") +
  xlab("Cost")
```



Therefore, based on our analysis above and the graph displayed above, an appropriate cost parameter matrix that would be appropriate to identify loan applications that deserve further examination would be the [[1,4.4],[1,1]] matrix. This would allow us to have a good sensitivity while not compromising precision either. Our sensitivity on the validation set is around 50% and the precision is around 20%. We assess a penalty of 4.4 times to loans that we classify as good but later default. Loans we predict would default but in reality will be paid should be looked at more carefully by a loan officer.

5) Evaluate the performance of your cost parameter matrix on the test set.

The resulting decision tree is as follows.

```
loanTree50 <- C5.0(x = train.x, y = train.y, costs = matrix(c(1,cost.50,1,1),2,2))

predict50 <- predict(loanTree50,test)

CrossTable(test$loan_status, predict50,
           prop.chisq = FALSE, prop.c = FALSE,
           prop.r = FALSE,
           dnn = c('actual default', 'predicted default'))
```

```
##
##
##    Cell Contents
## |-------------------------|
## |                       N |
## |         N / Table Total |
## |-------------------------|
##
##
## Total Observations in Table:  10693
##
##
##                | predicted default
## actual default | Charged Off |  Fully Paid |   Row Total |
## ---------------|-------------|-------------|-------------|
##    Charged Off |         786 |         661 |        1447 |
##                |       0.074 |       0.062 |             |
## ---------------|-------------|-------------|-------------|
##     Fully Paid |        3037 |        6209 |        9246 |
##                |       0.284 |       0.581 |             |
## ---------------|-------------|-------------|-------------|
##   Column Total |        3823 |        6870 |       10693 |
## ---------------|-------------|-------------|-------------|
##
##
```

When evaluating the performance of the cost parameter matrix on the test data, we have an accuracy of 0.6541663, which is equivalent to a misclassification rate of 0.3458337. While our misclassification rate has increased when compared to our original classifier, our false negatives are greatly reduced. We give out much fewer loans that will later be charged off. By penalizing our predicted paid but in reality charged off customers, we decrease the number of cases where we predicted no default where the customer actually defaulted. To us, having a defaulting customer is considered 4.4 times more expensive than missing out on a potential new loan. To more accurately tune this parameter, the bank would have to run some internal analysis comparing the true cost of a loan given out that is later defaulted on versus the missed profit from not extending a good loan.