Predicting Loan Defaults with Logistical Regression

Moko Sharma [ Mukund Raghav Sharma ]

## Executive Summary

This report provides the methodology, validation and reasons why the bank should deploy the statistical model that was created to improve the overall profit by using Statistical Modeling to successfully predict whether or not an applicant will default on his or her loan.

The overall process of statistical modeling involved exploring the data available to choose the appropriate criteria on the basis of which the model was created. The four chosen criteria are **percent savings per month** to give an idea about how much the prospective borrower saves per month, **employment and loan data** to shed light on the history of stable employment to generate income to payback the loan and a **credit risk score** to illuminate the borrowers worthiness to pay back the loan. Finally, **loan details** highlights how difficult it is to pay back on the specific type of loan; this variable is obtained by examining the loan grade and term.

Subsequent to discerning the criteria on the basis of which the statistical modelling will be based on, the model was created and evaluated. The evaluation process revealed that the best accuracy of predicting whether or not a borrower will default on his or her loan is **78.79%**. This benchmark is based on randomly selected sample data drawn from all the data available.

According to our testing, if the bank were to incorporate our model, it was ascertained that it could make a profit of upwards of **$10,902,292**. This value was derived by comparing the profit that computed by subtracting the total amount paid to the bank and the loan amount in dollars between using the model that predicts the bad loans and hypothesizes that the bank rejects them and not using the predictions of the model.

In conclusion, by incorporating this model, would overall help improve the bank drive the profits higher by accurately predicting whether or not a prospective lender will default on an issued loan. Additionally, as time goes by, more validation data can be used to improve this model even further.

## Introduction

The problem at hand involves using Logistical Regression to predict the proclivity of a loan applicant to default on his or her loan. The approach that will be taken to solve this problem is as follows:

1. Load the dataframe and do an initial smoke test by examining the dimensions of the data frame loaded.
2. Prepare the response variable based on the values of the **status** variable named as **LoanQuality** and add it as a separate column in the dataframe and then remove it along with the loanId as it will not add any benefit to our analysis.
3. Deduce the main criteria based on research and pondering over the most important elements of the independent variables. Involving:
   1. Do some feature engineering to create new variables that add more logical value and squash the dimensionality of the independent variables in an effort to prospectively simplify the model.
   2. Remove or impute missing values.
   3. Look out for skewness in the data and resolve that by some form of transformation. We want to remove all the skewness of our data in this step so that it gets to a digestible form that can be used in the later steps.
   4. Remove all the redudant columns that are related to that particular criteria.
4. Create a cleaned and transformed dataframe that can eventually be used in the second part of the project that contains variables that I deemed important and have thoroughly cleaned up.
5. Once the cleaned and transformed dataframe is obtained, the data frame is split into the training and test sets.
6. Using all the independent variables we engineered previously, a logistical regression model is created using the the Loan Quality as the dependent variable. This model is subsequently evaluated.
7. The prediction from the model is made and then the prediction power is evaluated using a confusion matrix and the accuracy is determined along with the AUC using an ROC curve.
8. Optimizing the threshold for **Accuracy** and **Profit**.
9. Summarizing the results.

### Loading in the Data

loanDf <- read.csv( './loans50k.csv' )  
loanDf.dim <- dim( loanDf )  
print( paste( "Dimensions for this Dataframe are:",   
 loanDf.dim[1], "rows",   
 loanDf.dim[2], "columns" ))

## [1] "Dimensions for this Dataframe are: 50000 rows 32 columns"

# The Min Max Function squashes the range to [ 0, 1 ]  
minMax <- function( x ) {  
 ( x - min( x )) / ( max( x ) - min( x ))  
}

## Preparing and Cleaning the Data for the Response Variable

A new column to the dataframe called **Loan Quality** that uses the status variable to discern whether or not a loan was considered good or bad will be added to represent the Response Variable. The loans that are charged off or default are considered as **Bad** and those that successfully paid will be marked as **Good**. All other statuses of loans such as late, current or in grace periods will be removed.

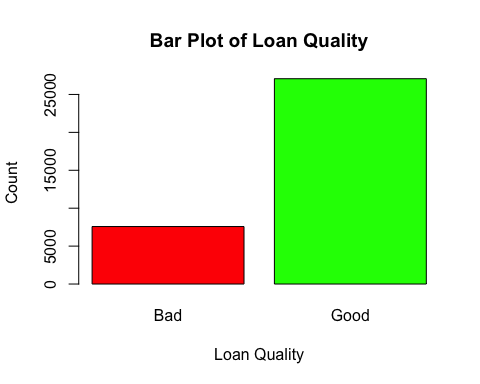
The redundant rows are sieved out by only keeping the Default, Fully Paid and Charged Off ones and check the dimensions of the new dataframe.

loanDf <- loanDf[ which( loanDf$status == 'Charged Off' |   
 loanDf$status == 'Default' |   
 loanDf$status == 'Fully Paid' ), ]  
dim( loanDf )

## [1] 34655 32

Now, the new **Loan Quality** column will be added to the dataframe and a barplot will be created for it.

loanQuality <- ifelse(( loanDf$status == 'Charged Off' | loanDf$status == 'Default' ),  
 yes = 'Bad',  
 no = 'Good' )  
  
loanDf$LoanQuality <- as.factor( loanQuality )  
loanQualityTable <- table( loanDf$LoanQuality )  
barplot( loanQualityTable,  
 main = 'Bar Plot of Loan Quality',  
 xlab = 'Loan Quality',  
 ylab = 'Count',  
 col = c( 'red', 'green' ))



Now that we have our response variable in place, the status column and the loanId column is dropped as it adds no valuable information to our prospective analysis.

loanDf$status <- NULL  
loanDf$loanID <- NULL

## Exploring and Transforming Data for Independent Variables

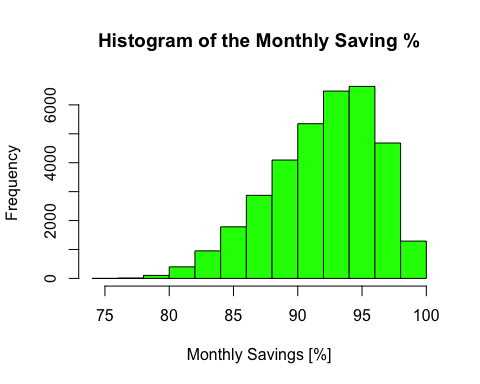
**1. Percent Savings Per Month**

The Percent Savings Per Month is important as it gives a sense of “how much does a borrower have to lose” partaking in a loan or how much money is saved per month after payment on the loan. The more the money saved by the lender, the less likely the lender will get into more debt and further perpetuate the cycle.

Our definition of the Percent Savings Per Month is given by:

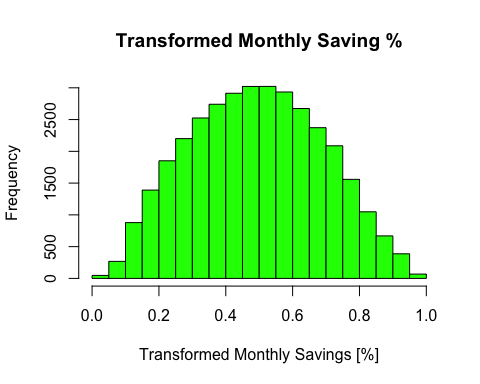
where:

monthlyIncome <- loanDf$income / 12  
monthlyPercentSavings <- ( monthlyIncome - loanDf$payment ) / ( monthlyIncome )  
monthlyPercentSavings <- monthlyPercentSavings \* 100   
  
hist( monthlyPercentSavings,  
 main = "Histogram of the Monthly Saving %",  
 xlab = "Monthly Savings [%]",  
 col = c( 'green' ))



Clearly, this histogram is **left-skewed**. To remove this skewness, the distribution is raised to a power of 8, a transformation obtained through trial and error. Once the skewness is removed, the values are min-max normalized to get the values to a scale between 0 and 1.

monthlyPercentSavings <- minMax( monthlyPercentSavings \*\* 8 )   
hist( monthlyPercentSavings,  
 main = "Transformed Monthly Saving %",  
 xlab = "Transformed Monthly Savings [%]",  
 col = c( 'green' ))



loanDf$monthlyPercentSavings <- monthlyPercentSavings

**2. Employment Details**

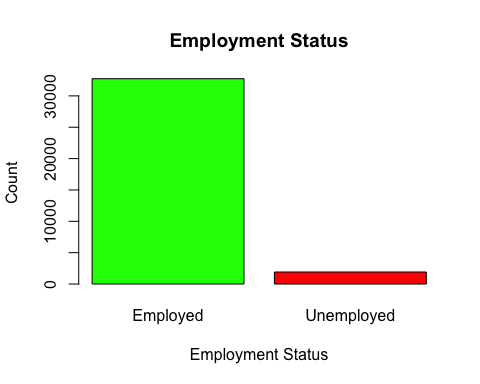
The Employment Details include the employment status, the length of continuous employment if applicable and verifiability of annual income comprise of this facet of our prediction variables. This criteria is important as having a steady and historically stable job is indicative of the ability to generate an income for the entire term of the loan.

The raw variables that will be considered from the data frame are:

1. Employment Title (**employment**) that will be converted into a qualitative variable that will represent the Employment Status called **employmentStatus** with two levels: Unemployed where the Employment Status is an empty string while for all other cases the values will be Employed as we truly don’t care about what the title is as long as it exists.

# Let us first extract the employment status   
# by checking for empty strings  
employmentStatus <- as.factor( ifelse( loanDf$employment == '',   
 yes = "Unemployed",  
 no = "Employed" ))  
  
# Show the results as a table  
employmentStatusTable <- table( employmentStatus )  
employmentStatusTable

## employmentStatus  
## Employed Unemployed   
## 32737 1918

# Add the new Employment Status Column to the Dataframe  
loanDf$employmentStatus <- employmentStatus  
  
# Graph a Bar plot   
barplot( employmentStatusTable,  
 main = "Employment Status",   
 xlab = "Employment Status",  
 ylab = "Count",   
 col = c( "green", "red" ))

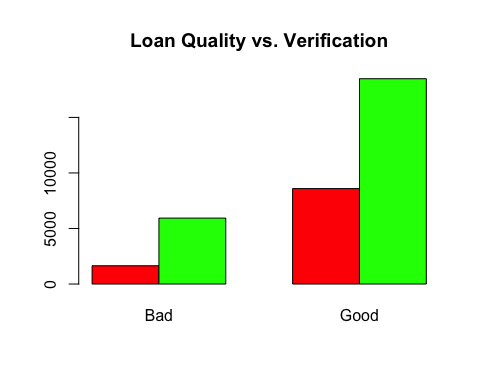
1. Length of Employment (**length**) that will be converted into a numeric score between 0 and 10 by removing the non-numeric values from the dataframe by making use of gsub and a regex string. All the Not-Available values will be automatically set to 0 to indicate unemployment. This newly transformed variable is named **employmentLength**.

# Remove the non-numeric and convert into a numeric value.  
employmentLength <- as.numeric( gsub( "[^0-9]", "", loanDf$length ))  
  
# Replace the NAs with 0s.   
employmentLength[ is.na( employmentLength )] <- 0  
  
# Add the Employment Length to the dataframe.  
loanDf$employmentLength <- employmentLength

1. Verification Status of Annual Income (**verified**): the qualitative variable will be cleaned up to only include 2 levels: Verified and Unverified. The verification of the annual income adds a depth of reliability to the borrower and hence is added to this criteria.

loanDf$verified <- as.factor( ifelse( loanDf$verified == "Not Verified",  
 yes = "Not Verified",   
 no = "Verified" ))  
qualityAndVerified <- table( loanDf$verified, loanDf$LoanQuality )  
qualityAndVerified

##   
## Bad Good  
## Not Verified 1645 8591  
## Verified 5936 18483

barplot( qualityAndVerified,  
 main = "Loan Quality vs. Verification",  
 col = c( 'red', 'green' ),   
 beside = TRUE )

**3. Credit Risk Score**

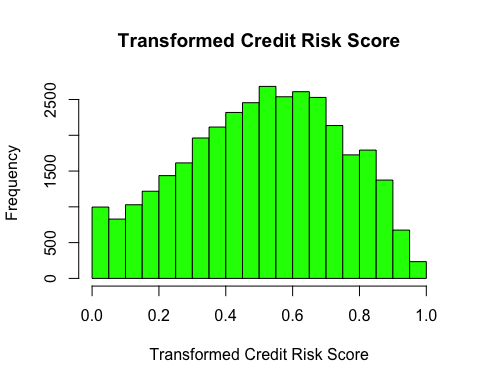
We calculate our own version of the Credit Risk Score that will be the result of engineering the features related to credit limits. This credit risk score will numerically highlight the credit risk of a borrower to default on a loan. The credit risk score is between 0 and 1 where 0 is least risky and 1 is extremely risky is mathematically given by:

DelinquencyPoints is the sum of number of 30+ day late payments in the last two years and number of derogatory public records. The rationale here is that the more the deliquency points, the higher the risk of defaulting on a loan. The number of credit checks is indicative of how desperately credit is needed and this is viewed as a negative. Finally, the Balance To Credit is the ratio of the total balance to the credit risk. The higher the balance, the more the inclination to default.

It is at this point that all the NA values are removed; the rationale for doing so was based on the fact that there were few enough of them in number to not resort to imputation.

# Delinquency  
delinquency <- loanDf$delinq2yr + loanDf$pubRec  
  
# Checks  
checks <- loanDf$inq6mth  
  
# Balance To Credit  
balanceToCredit <- ifelse( loanDf$totalRevLim == 0,   
 NA,   
 loanDf$totalRevBal / loanDf$totalRevLim )  
  
creditRiskScore <- delinquency + checks + balanceToCredit  
loanDf$creditRiskScore <- creditRiskScore  
loanDf <- na.omit( loanDf )  
loanDf$creditRiskScore <- 1 - minMax( loanDf$creditRiskScore )   
  
hist( loanDf$creditRiskScore,  
 main = 'Credit Risk Score',  
 xlab = 'Credit Risk Score')

The credit risk score seems to be extremely left skewed. A positive power transformation will be applied to remove the skew; the power used here will be 100 - this value was obtained via trial and error.

loanDf$creditRiskScore <- loanDf$creditRiskScore ^ 100   
hist( loanDf$creditRiskScore,  
 main = 'Transformed Credit Risk Score',  
 xlab = 'Transformed Credit Risk Score',  
 col = 'green' )

**4. Loan Details**

The Loan Details consist of Loan Grade and Loan Term. These two independent variables are chosen as they are indicative of the quality of the loan issued and can illuminate how challenging it will be for the borrower to pay back the loan.

1. Loan Grade (**grade**) will be kept as is after checking for any anomalous values.

summary( loanDf$grade )

## A B C D E F G   
## 0 5748 9268 9574 5640 2905 927 209

No anomolous values here hence, these values will be kept as is.

1. Loan Term (**term**) will be converted into a qualitative variable with the same name that consist of two levels: LongTerm and ShortTerm.

loanTerm <- ifelse( as.character( loanDf$term ) == " 36 months",  
 yes = "LongTerm",  
 no = "ShortTerm" )  
termTable <- table( loanTerm )  
termTable

## loanTerm  
## LongTerm ShortTerm   
## 25471 8800

loanDf$term <- as.factor( loanTerm )

All the other columns are removed since these are now redundant including the home variable that, has no effect on the prediction of defaulting on a loan barring the totalPaid column that will be used later.

The dimensions and summary of the finalized cleaned data frame is as follows:

dim( loanDf )

## [1] 34271 11

summary( loanDf )

## amount term grade income   
## Min. : 1000 LongTerm :25471 C :9574 Min. : 7067   
## 1st Qu.: 8000 ShortTerm: 8800 B :9268 1st Qu.: 45658   
## Median :12600 A :5748 Median : 65000   
## Mean :14647 D :5640 Mean : 75210   
## 3rd Qu.:20000 E :2905 3rd Qu.: 90000   
## Max. :35000 F : 927 Max. :7446395   
## (Other): 209   
## verified totalPaid LoanQuality monthlyPercentSavings  
## Not Verified:10112 Min. : 0 Bad : 7504 Min. :0.0000   
## Verified :24159 1st Qu.: 7106 Good:26767 1st Qu.:0.3398   
## Median :12300 Median :0.4905   
## Mean :14838 Mean :0.4918   
## 3rd Qu.:20490 3rd Qu.:0.6401   
## Max. :58697 Max. :1.0000   
##   
## employmentStatus employmentLength creditRiskScore   
## Employed :32384 Min. : 0.000 Min. :0.0000   
## Unemployed: 1887 1st Qu.: 2.000 1st Qu.:0.3372   
## Median : 6.000 Median :0.5225   
## Mean : 5.752 Mean :0.5076   
## 3rd Qu.:10.000 3rd Qu.:0.6871   
## Max. :10.000 Max. :1.0000   
##

Now that the dataframe is in right form, the model creation step is addressed.

## The Logistical Model

Logistical Regression is used to create the model as the response variable is categorical. This section is dedicated to creating and evaluating the model in addition to prediction and the subsequent evaluation of the prediction.

### Splitting the Training and Test Data

The cleaned dataframe is split into the training and test set with a 80 - 20 ratio.

set.seed( 123 ) # For reproducibility  
  
indexes <- sample( 1 : nrow( loanDf ), size = 0.8 \* nrow( loanDf ))  
  
# Creating the Training Set  
loanDf.train <- loanDf[ indexes, ]   
loanDf.train$totalPaid <- NULL  
loanDf.train$amount <- NULL  
  
# Creating the Testing Set  
loanDf.test <- loanDf[ -indexes, ]

### Creating the Model

The model is created on the training data using Logistical regression using the glm function on the training data set.

loanDf.model <- glm( LoanQuality ~ .,  
 data = loanDf.train,   
 family = binomial('logit'))  
loanDf.fitted <- fitted( loanDf.model )

### Evaluating the Model

The evaluating the model will involve:

1. Studying the results of the Wald Test from the coefficients of the Logistical Regression.
2. Obtaining the McFadden’s Pseudo R Square value for the model.
3. Conducting the Homer-Lemeshow Test to assess the Goodness of Fit of the Model.

summaryOfModel <- summary( loanDf.model )  
print( paste( "AIC:", summaryOfModel$aic ))

## [1] "AIC: 26324.7205416657"

summaryOfModel$coefficients  
## Pr(>|z|)  
## (Intercept) 1.895013e-81  
## termShortTerm 2.455032e-48  
## gradeB 3.497558e-21  
## gradeC 1.286860e-71  
## gradeD 7.059019e-97  
## gradeE 1.019137e-104  
## gradeF 2.627145e-82  
## gradeG 4.299596e-34  
## income 4.746013e-08  
## verifiedVerified 1.984872e-03  
## monthlyPercentSavings 3.529814e-11  
## employmentStatusUnemployed 1.332558e-07  
## employmentLength 3.566517e-01  
## creditRiskScore 2.429304e-13

1. From the summary of the model, it is clear that the Wald Test highlights statistical significance for all the chosen coefficients with a 5% level of significance. As a sidenote, the AIC of this model is **26325**.
2. McFadden’s Psuedo R Squared Value

library( pscl )

pR2( loanDf.model )[4]

## McFadden   
## 0.09024517

The 9% McFadden Pseudo R Squared value doesn’t highlight a strong predictive power of the model. This might cause us some concern, but still, we will try to improve the prediction by adjusting the probability threshold.

1. Homer-Lemeshow Test

The null hypothesis in this case is that the selected logistical regression model fits while the alternative hypothesis highlights the contrary.

library( ResourceSelection )

## ResourceSelection 0.3-2 2017-02-28

hoslem.test( as.numeric( loanDf.train$LoanQuality ), loanDf.fitted, g = 10 )

##   
## Hosmer and Lemeshow goodness of fit (GOF) test  
##   
## data: as.numeric(loanDf.train$LoanQuality), loanDf.fitted  
## X-squared = 221270, df = 8, p-value < 2.2e-16

The null hypothesis is rejected with a statistical significance of 0.05 and 0.01 [ P < 2.2e-16 ]. This highlights that the selected model doesn’t fit well but is not a big concern as we are more interested in prediction rather than modeling.

### Prediction

Now that the model has been evaluated, the next step is to conduct the prediction and evaluate the predictive power.

**Predicting from the Model**

loanDf.predict <- predict( loanDf.model,   
 newdata = loanDf.test,   
 type = "response" )

**Evaluating the Prediction**

Now that predicted probabilities are obtained based on the model that was created, the accuracies will be evaluated by constructing a confusion matrix which is also known as the contingency table and extracting some important results from the confusion matrix.

library( caret )

threshhold <- 0.5   
loanDf.predicted <- factor( ifelse( loanDf.predict < threshhold , "Bad", "Good" ))   
loanDf.confusionMatrix.50 <- confusionMatrix( loanDf.test$LoanQuality,   
 loanDf.predicted,  
 positive = 'Good' )  
loanDf.confusionMatrix.50$overall['Accuracy']

## Accuracy   
## 0.7862874

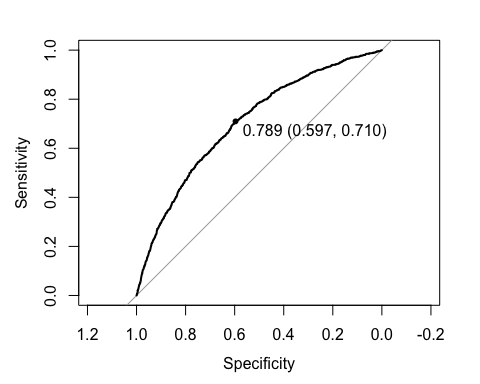
loanDf.confusionMatrix.50$byClass['Sensitivity']

## Sensitivity   
## 0.7945637

loanDf.confusionMatrix.50$byClass['Specificity']

## Specificity   
## 0.505102

Accuracy of the model, as given by the confusion matrix is **78.63%**. The Sensitivity or true positive rate, which in this context is the proportion of actual positives i.e. good loans amongst all loans that are identified as good, is **79.456%**. The Specificity or the true negative rate, which is the proportion of loans that are actual bad loans amongst all the loans that are identified as bad is: **50.510%**.

**Receiver Operator Characteristic Curve**

The Receiver Operator Characteristic Curve is a graphical plot that illustrates the diagnostic ability of a binary classifier system as its discrimination threshold is varied.

library( pROC )

loanDf.rocCurve <- roc( response = loanDf.test$LoanQuality,  
 predictor = loanDf.predict,   
 levels = rev( levels( loanDf$LoanQuality )))  
plot( loanDf.rocCurve, print.thres = "best" )

print( paste( "AUC: ", auc( loanDf.rocCurve )))

## [1] "AUC: 0.702886036732895"

The best threshold from the ROC Curve is **0.789** and the AUC [ Area Under Curve ] is **0.702** which is certainly higher than 0.5, which is typically used as a baseline of the predictive power of a model; AUC combines both the False Positive Rate and True Positive Rate into one metric. The higher the AUC, the better the predictive power of the model.

In general, from the evaluation of the model, this doesn’t seem like one with a goodness of fit nor with a high enough Pseudo R-Squared value. The prediction accuracy of the model, however, doesn’t seem as bad with an accuracy of ~**79%** but could most definitely be better. The ROC curve and its corresponding AUC highlights a semi-powerful model based on the bow shape of the curve and value of 0.702.

## Optimizing the Threshold for Accuracy

# Function that computes the accuracy of the prediction based off a threshold  
computeAccuracy <- function( t ) {  
 predicted <- factor( ifelse( loanDf.predict < t, 'Bad', 'Good' ))   
 cmat <- confusionMatrix( data = predicted,  
 reference = loanDf.test$LoanQuality,  
 positive = 'Good' )  
 cmat$overall['Accuracy']  
}  
  
# Granuality in our data is 0.005, we could go even more granualar but   
# this is a trade off between computation and precision.  
sequenceToComputeAccuraciesFor <- seq( from = 0.0, to = 1.0, by = 0.01 )   
accuracies <- lapply( sequenceToComputeAccuraciesFor, computeAccuracy )   
names( accuracies ) <- sequenceToComputeAccuraciesFor  
accuracies[ which.max( accuracies )]

## $`0.53`  
## Accuracy   
## 0.787892

The threshold with the highest accuracy is 0.53 which is **78.79%**.

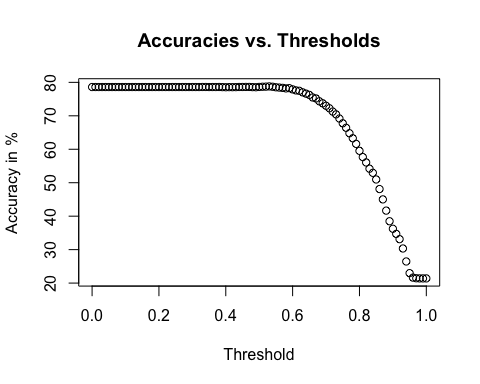
predicted <- factor( ifelse( loanDf.predict < 0.53, 'Bad', 'Good' ))   
cmat <- confusionMatrix( data = predicted,  
 reference = loanDf.test$LoanQuality,  
 positive = 'Good' )  
cmat$byClass['Sensitivity']

## Sensitivity   
## 0.9738307

cmat$byClass['Specificity']

## Specificity   
## 0.1049761

plot( y = unlist( accuracies ) \* 100,   
 x = sequenceToComputeAccuraciesFor,  
 main = "Accuracies vs. Thresholds",   
 ylab = "Accuracy in %",  
 xlab = "Threshold" )



## Optimizing the Threshold for Profit

### Profit For the Good Loans Based on the Model

This section involves applying the model we obtained earlier with a threshold of 0.5 on the test data to compute the profit given by summing up **totalPaid - amount**.

idxOfGoodLoans <-   
 which( as.character( loanDf.test$LoanQuality ) == as.character( "Good" ))  
loanDf.test.goodLoans <- loanDf.test[ idxOfGoodLoans, ]  
  
loanDf.model.profit <-   
 sum( loanDf.test.goodLoans$totalPaid - loanDf.test.goodLoans$amount )  
loanDf.model.profit

## [1] 12583350

The profit by applying the model to the test data with just the Good loans is **$12,583,350**.

### Change in Total Profit

The total profit is given by the overall sum of the totalPaid subtracted by the amount for the current model. And hence, the difference between the total profit with and without the bad loans using our own model is computed.

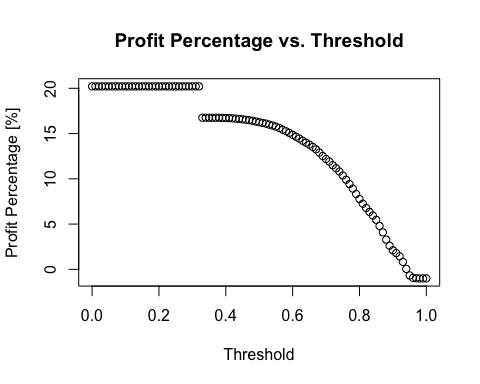
totalProfit <- sum( loanDf.test$totalPaid - loanDf.test$amount )  
loanDf.model.profit - totalProfit

## [1] 10902292

The difference between the total profit including the bad loans and those without the bad loans is: **$10,902,292**.

### Optimizing the Threshold for Total Profit

computeProfitPercentage <- function( t ) {  
 predicted <- factor( ifelse( loanDf.predict < t, 'Bad', 'Good' ))   
 idxOfGood <- predicted[ which( as.character( predicted ) == 'Good' )]   
 predictedGood <- loanDf.test[ idxOfGood, ]  
 model.predictedTotalProfit <- sum( predictedGood$totalPaid - predictedGood$amount )  
 notModel.totalProfit <- sum( loanDf.test$totalPaid - loanDf.test$amount )  
 ( model.predictedTotalProfit - notModel.totalProfit ) / notModel.totalProfit  
}  
  
sequenceToComputeProfitFor <- seq( from = 0.0, to = 1.0, by = 0.01 )   
compareProfitPercentage <- lapply( sequenceToComputeProfitFor, computeProfitPercentage )   
names( compareProfitPercentage ) <- sequenceToComputeProfitFor   
compareProfitPercentage[ which.max( compareProfitPercentage )]

## $`0`  
## [1] 20.21133

The threshold at which the increase in the profit percent using our model is the highest between **0** and **0.31** and at an increase of **20.21** times the original profit.

plot( y = unlist( compareProfitPercentage ),  
 x = sequenceToComputeProfitFor,   
 main = 'Profit Percentage vs. Threshold',  
 ylab = 'Profit Percentage [%]',  
 xlab = 'Threshold' )

### Comparisons Against a Perfect Model

The perfect model would reject all the bad loans.

idxOfGood <-   
 which( as.character( loanDf.test$LoanQuality ) == 'Good' )   
perfectGoodLoans <- loanDf.test[ idxOfGood, ]  
perfectGoodLoanProfit <- sum( perfectGoodLoans$totalPaid - perfectGoodLoans$amount )  
totalProfit <- sum( loanDf.test$totalPaid - loanDf.test$amount )  
( perfectGoodLoanProfit - totalProfit ) / ( totalProfit )

## [1] 6.485375

The profit by a perfect model is **648.5%** higher than not using the model at all.

### Best Profit Threshold Accuracy and Other Metrics

Our previously obtained optimal accuracy threshold used is **0.53**. This threshold is used to compute the overall accuracy and the corresponding sensitivity.

predicted <- factor( ifelse( loanDf.predict < 0.53, 'Bad', 'Good' ))   
cmat <- confusionMatrix( data = predicted,  
 reference = loanDf.test$LoanQuality,  
 positive = 'Good' )  
cmat$overall['Accuracy']

## Accuracy   
## 0.787892

cmat$byClass['Sensitivity']

## Sensitivity   
## 0.9738307

cmat$byClass['Specificity']

## Specificity   
## 0.1049761

The accuracy from this model is: **78.78%**. The sensitivity is **97.3%** and the specificity is **10.5%**. The maximum profit threshold does not coincide with the maximum accuracy threshold.

## Summary

The model that was created and analyzed in the previous sections has a maximum accuracy at a threshold point of **0.53** with an accuracy of **78.79%**; the Sensitivity or true positive rate is **97.38%** while the Specificity or the true negative rate is: **10.5%**.

The optimizied profit threshold point is **0.31** with an overall increase of **20.21** times the original profit, an accuracy of **78.6%**, sensitivity of **100%** and the specificity is **0%**.

In conclusion, the use of the model results in an increase in profits of **$10,902,292** in comparison to not using the model at all.