Credit Risk Modeling using Logistic Regression in R

Vikash Singh - connect with me at https://in.linkedin.com/in/vikashsingh29 (https://in.linkedin.com/in/vikashsingh29)

19 May 2016

Introduction

Modeling credit risk for both personal and company loans is of major importance for banks and financial institutions. The probability that a debtor will default is a key component in getting to a measure for credit risk.

In this blog, we will be building one of the widely used machine learning techniques for solving classification problem, i.e, logistic regression.

About the Data

The original data used in this exercise comes from publicly available data from LendingClub.com (https://www.lendingclub.com/info/download-data.action (https://www.lendingclub.com/info/download-data.action)), a website that connects borrowers and investors over the Internet.

Variables used in the Data

Dependent Variable - 'not.fully.paid'

a binary variable indicating that the loan was not paid back in full, i.e, (the borrower either defaulted or the loan was "charged off," meaning the borrower was deemed unlikely to ever pay it back).

Independent Variables

1)credit.policy: 1 if the customer meets the credit underwriting criteria of LendingClub.com, and 0 otherwise.

2)purpose: The purpose of the loan (takes values "credit_card", "debt_consolidation", "educational", "major_purchase", "small_business", and "all_other").

3)int.rate: The interest rate of the loan, as a proportion (a rate of 11% would be stored as 0.11). Borrowers judged by LendingClub.com to be more risky are assigned higher interest rates.

4)installment: The monthly installments ($) owed by the borrower if the loan is funded.

5)annualincome: the self-reported annual income of the borrower.

6)log.annual.inc: The natural log of the self-reported annual income of the borrower.

7)dti: The debt-to-income ratio of the borrower (amount of debt divided by annual income).

8)fico: The FICO credit score of the borrower.

9)days.with.cr.line: The number of days the borrower has had a credit line.

10)revol.bal: The borrower's revolving balance (amount unpaid at the end of the credit card billing cycle).

11)revol.util: The borrower's revolving line utilization rate (the amount of the credit line used relative to total credit available).

12)inq.last.6mths: The borrower's number of inquiries by creditors in the last 6 months.

13)delinq.2yrs: The number of times the borrower had been 30+ days past due on a payment in the past 2 years.

14)pub.rec: The borrower's number of derogatory public records (bankruptcy filings, tax liens, or judgments).

Explore the Data

```
str(loans)  #There are 5000 observations and  15 variables
```

```
## 'data.frame':    5000 obs. of  15 variables:
##  $ credit.policy    : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ purpose          : Factor w/ 7 levels "all_other","credit_card",..: 3 1 2 1 7
5 5 3 7 7 ...
##  $ int.rate         : num  0.15 0.111 0.134 0.106 0.15 ...
##  $ installment      : num  194 131.2 678.1 32.5 225.4 ...
##  $ log.annual.inc   : num  10.7 11 11.9 10.4 12.3 ...
##  $ dti              : num  4 11.08 10.15 14.47 6.45 ...
##  $ fico             : int  667 722 682 687 677 772 682 712 737 672 ...
##  $ days.with.cr.line: num  3180 5116 4210 1110 6240 ...
##  $ revol.bal        : int  3839 24220 41674 4485 56411 269 3408 6513 10296 2867
...
##  $ revol.util       : num  76.8 68.6 74.1 36.9 75.3 3.8 35.1 34.3 42.5 55.1 ...
##  $ inq.last.6mths   : int  0 0 0 1 0 3 1 3 0 3 ...
##  $ delinq.2yrs      : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ pub.rec          : int  1 0 0 0 0 0 0 1 0 0 ...
##  $ not.fully.paid   : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ annualincome     : int  45000 60000 145000 33990 213000 75000 32000 40000 960
00 15000 ...
```

```
summary(loans) # no missing values
```

```
##  credit.policy                   purpose        int.rate
##  Min.   :0.0000   all_other         :1227   Min.   :0.0600
##  1st Qu.:1.0000   credit_card       : 664   1st Qu.:0.1008
##  Median :1.0000   debt_consolidation:1957   Median :0.1218
##  Mean   :0.8962   educational       : 198   Mean   :0.1208
##  3rd Qu.:1.0000   home_improvement  : 354   3rd Qu.:0.1379
##  Max.   :1.0000   major_purchase    : 186   Max.   :0.2164
##                   small_business    : 414
##   installment     log.annual.inc        dti              fico
##  Min.   : 15.69   Min.   : 7.601   Min.   : 0.000   Min.   :617.0
##  1st Qu.:163.55   1st Qu.:10.545   1st Qu.: 7.067   1st Qu.:682.0
##  Median :260.64   Median :10.915   Median :12.300   Median :707.0
##  Mean   :308.33   Mean   :10.912   Mean   :12.309   Mean   :710.9
##  3rd Qu.:407.51   3rd Qu.:11.277   3rd Qu.:17.652   3rd Qu.:737.0
##  Max.   :926.83   Max.   :14.528   Max.   :29.960   Max.   :827.0
##
##  days.with.cr.line   revol.bal        revol.util    inq.last.6mths
##  Min.   :  180     Min.   :      0   Min.   :  0.0   Min.   : 0.000
##  1st Qu.: 2790     1st Qu.:   3328   1st Qu.: 22.3   1st Qu.: 0.000
##  Median : 4080     Median :   8605   Median : 45.7   Median : 1.000
##  Mean   : 4511     Mean   :  15872   Mean   : 46.4   Mean   : 1.407
##  3rd Qu.: 5640     3rd Qu.:  18155   3rd Qu.: 70.5   3rd Qu.: 2.000
##  Max.   :16259     Max.   :1207359   Max.   :106.5   Max.   :33.000
##
##   delinq.2yrs        pub.rec       not.fully.paid    annualincome
##  Min.   :0.0000   Min.   :0.0000   Min.   :0.0000   Min.   :   2000
##  1st Qu.:0.0000   1st Qu.:0.0000   1st Qu.:0.0000   1st Qu.:  38000
##  Median :0.0000   Median :0.0000   Median :0.0000   Median :  55000
##  Mean   :0.1614   Mean   :0.0668   Mean   :0.3066   Mean   :  66260
##  3rd Qu.:0.0000   3rd Qu.:0.0000   3rd Qu.:1.0000   3rd Qu.:  79000
##  Max.   :6.0000   Max.   :3.0000   Max.   :1.0000   Max.   :2039784
##
```

Convert credit.policy variable as categorical variable

```
loans$credit.policy = as.factor(loans$credit.policy)
str(loans)
```

```
## 'data.frame':    5000 obs. of  15 variables:
##  $ credit.policy    : Factor w/ 2 levels "0","1": 2 2 2 2 2 2 2 2 2 2 ...
##  $ purpose          : Factor w/ 7 levels "all_other","credit_card",..: 3 1 2 1 7
5 5 3 7 7 ...
##  $ int.rate         : num  0.15 0.111 0.134 0.106 0.15 ...
##  $ installment      : num  194 131.2 678.1 32.5 225.4 ...
##  $ log.annual.inc   : num  10.7 11 11.9 10.4 12.3 ...
##  $ dti              : num  4 11.08 10.15 14.47 6.45 ...
##  $ fico             : int  667 722 682 687 677 772 682 712 737 672 ...
##  $ days.with.cr.line: num  3180 5116 4210 1110 6240 ...
##  $ revol.bal        : int  3839 24220 41674 4485 56411 269 3408 6513 10296 2867
...
##  $ revol.util       : num  76.8 68.6 74.1 36.9 75.3 3.8 35.1 34.3 42.5 55.1 ...
##  $ inq.last.6mths   : int  0 0 0 1 0 3 1 3 0 3 ...
##  $ delinq.2yrs      : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ pub.rec          : int  1 0 0 0 0 0 0 1 0 0 ...
##  $ not.fully.paid   : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ annualincome     : int  45000 60000 145000 33990 213000 75000 32000 40000 960
00 15000 ...
```

Preparing the Dataset for Prediction

```
library(caTools)
```

```
## Warning: package 'caTools' was built under R version 3.2.4
```

```
set.seed(100)
spl = sample.split(loans$not.fully.paid, 0.7)
train = subset(loans, spl == TRUE)
test = subset(loans, spl == FALSE)
```

Building the Logistic Model and checking the model summary

We have left out annualincome as it is already included in the variable ‘log.annual.inc’

```
modLog = glm(not.fully.paid ~. -annualincome, data=train, family="binomial")
summary(modLog)
```

```
##
## Call:
## glm(formula = not.fully.paid ~ . - annualincome, family = "binomial",
##     data = train)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -1.69976  -0.72158  -0.54723   0.00013   2.35322
##
## Coefficients:
##                             Estimate Std. Error z value Pr(>|z|)
## (Intercept)                2.064e+01  3.226e+02   0.064  0.94899
## credit.policy1            -1.902e+01  3.226e+02  -0.059  0.95300
## purposecredit_card        -4.983e-01  1.686e-01  -2.955  0.00313 **
## purposedebt_consolidation -2.668e-01  1.209e-01  -2.206  0.02737 *
## purposeeducational        -1.936e-02  2.348e-01  -0.082  0.93428
## purposehome_improvement   -6.254e-02  2.031e-01  -0.308  0.75807
## purposemajor_purchase     -1.666e-02  2.461e-01  -0.068  0.94602
## purposesmall_business      1.161e-01  1.797e-01   0.646  0.51817
## int.rate                   1.777e+01  3.092e+00   5.746 9.13e-09 ***
## installment                1.291e-03  2.829e-04   4.564 5.03e-06 ***
## log.annual.inc            -5.419e-01  1.001e-01  -5.413 6.20e-08 ***
## dti                        3.105e-03  7.445e-03   0.417  0.67662
## fico                       5.627e-05  2.194e-03   0.026  0.97954
## days.with.cr.line          3.618e-05  2.055e-05   1.760  0.07834 .
## revol.bal                  6.464e-06  3.370e-06   1.918  0.05508 .
## revol.util                 2.730e-03  2.080e-03   1.313  0.18934
## inq.last.6mths             1.328e-01  3.559e-02   3.731  0.00019 ***
## delinq.2yrs               -8.311e-02  9.695e-02  -0.857  0.39130
## pub.rec                    3.673e-01  1.656e-01   2.218  0.02657 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 4314.3  on 3499  degrees of freedom
## Residual deviance: 3127.3  on 3481  degrees of freedom
## AIC: 3165.3
##
## Number of Fisher Scoring iterations: 17
```

Significance Level of the Variables

Those variables which have atleast one star in the coefficients table are sigificant. Positive coefficient means higher the value of that variable, an increased risk of default, and vice versa.

The significant variales are Credit Card Purpose, Interest Rate, ‘inq.last.6mths’, and ‘pub.rec’.

Since some of the variables are not significant, we will rebuild the logistic regression with only the significant variables.

Revised Logistic Regression Model

```
modLog2 = glm(not.fully.paid ~ purpose + int.rate + installment + log.annual.inc +
inq.last.6mths + pub.rec, data=train, family="binomial")
summary(modLog2)
```

```
##
## Call:
## glm(formula = not.fully.paid ~ purpose + int.rate + installment +
##     log.annual.inc + inq.last.6mths + pub.rec, family = "binomial",
##     data = train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.1109  -0.8035  -0.5614   0.8903   2.4357
##
## Coefficients:
##                             Estimate Std. Error z value Pr(>|z|)
## (Intercept)               -0.3552949  0.8528953  -0.417 0.676989
## purposecredit_card        -0.3342296  0.1459913  -2.289 0.022057 *
## purposedebt_consolidation -0.2615709  0.1068706  -2.448 0.014383 *
## purposeeducational        -0.0968354  0.2150149  -0.450 0.652447
## purposehome_improvement   -0.1267309  0.1825735  -0.694 0.487596
## purposemajor_purchase     -0.2953788  0.2368597  -1.247 0.212375
## purposesmall_business     -0.0037807  0.1647683  -0.023 0.981694
## int.rate                  24.9879111  1.8469198  13.530  < 2e-16 ***
## installment                0.0009194  0.0002437   3.773 0.000161 ***
## log.annual.inc            -0.3909672  0.0783604  -4.989 6.06e-07 ***
## inq.last.6mths             0.3682961  0.0258440  14.251  < 2e-16 ***
## pub.rec                    0.3337165  0.1467921   2.273 0.023002 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 4314.3  on 3499  degrees of freedom
## Residual deviance: 3657.3  on 3488  degrees of freedom
## AIC: 3681.3
##
## Number of Fisher Scoring iterations: 4
```

All the variables are significant.

Using the Revised Model for Making

Predictions on the Test Data Set

We will store the prediction values in a vector named ‘predicted.risk’ and add it in the test data set.

```
test$predicted.risk = predict(modLog2, newdata=test, type="response")
```

Measuring the accuracy of the model

```
table(test$not.fully.paid, as.numeric(test$predicted.risk >= 0.5))
```

```
##
##        0   1
##   0 970  70
##   1 306 154
```

Computing Accuracy of the Model

Overall Accuracy = (970 + 154)/ nrow(test) = 74.933 Sensitivity = 154/460 = 33.5 Specificity = 970 / 1040 = 93.3

Inference

We see that the model is doing far better in sensitivity as compared to specificity. We will come back to these concepts later, but before that, let us compare our model with the baseline accuracy.

Comparing Baseline Accuracy

```
table(test$not.fully.paid)
```

```
##
##    0    1
## 1040  460
```

```
1040/(1040+460)
```

```
## [1] 0.6933333
```

The baseline accuracy is 0.693. Hence, we see that the Revised Model beats the baseline model comfortably.

Test set Area Under the Curve (AUC)

```
library(ROCR)
```

```
## Loading required package: gplots
```

```
##
## Attaching package: 'gplots'
```

```
## The following object is masked from 'package:stats':
##
##     lowess
```

```
pred = prediction(test$predicted.risk, test$not.fully.paid)
as.numeric(performance(pred, "auc")@y.values)
```

```
## [1] 0.7318395
```

Area Under the Curve (AUC) comes out to be 0.73.

Although the model beats the Baseline Model, it does not do extremely good. The reason being that the model is low on Sensitivity. For a bank or financial institution, the misclassification cost of not predicting loan defaults correctly is much higher than misclassiffication cost of not predicting non-defaults correctly, hence it is important that the model should have a higher sensitivity.

This is controlled by the threshold value.

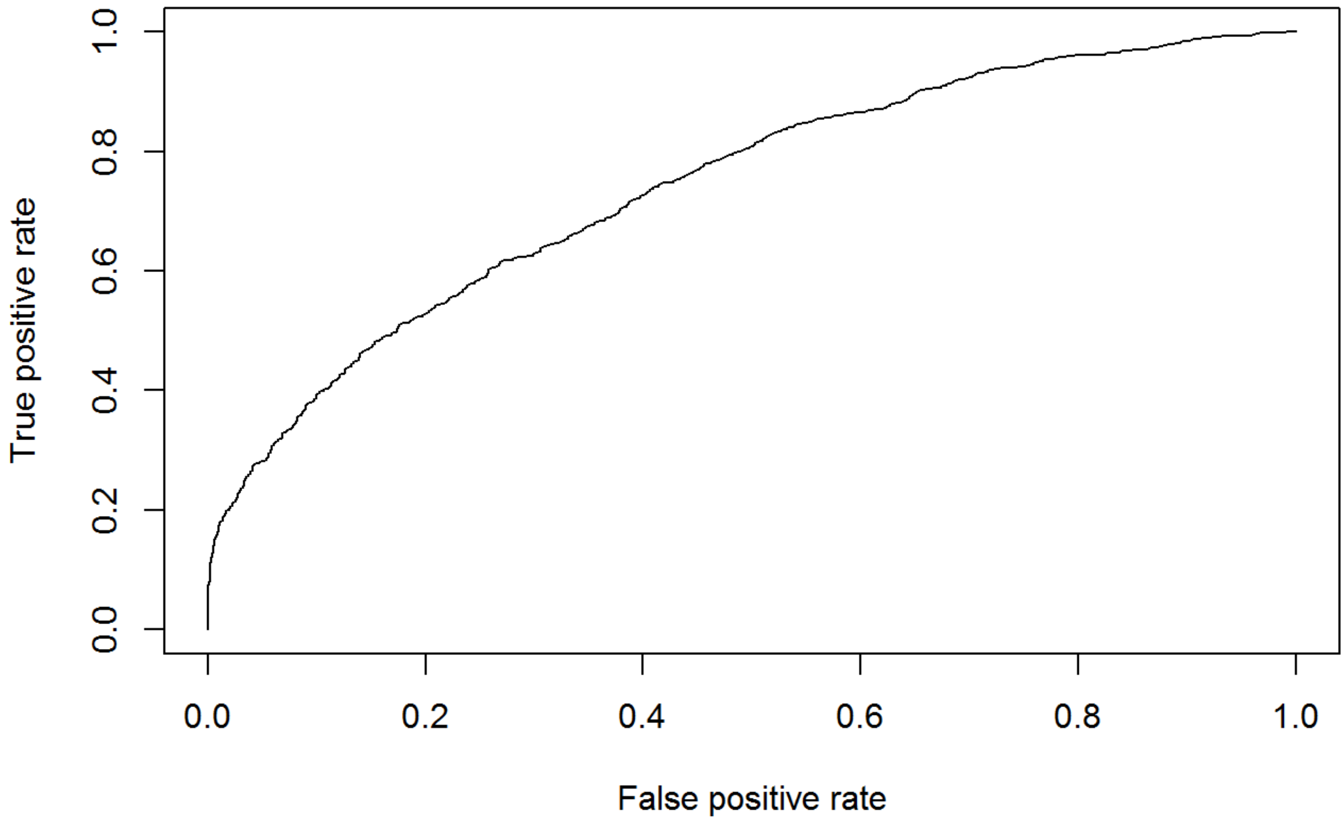
If we increase the threshold value from 0.5 to 0.7, the sensitivity will decrease from 33% to 17%, even though overall acuuracy will increase to 74% from 73% earlier. However, that is not aceptable from the bank's point of view which is more interested in classifying defaults.

On the other hand, if we reduce the threshold level from 0.5 to 0.3, our sensitivity incraeses drastically from 33% to 60%. But the overall model acuracy comes down to 68%, which is less than the baseline model accuracy.

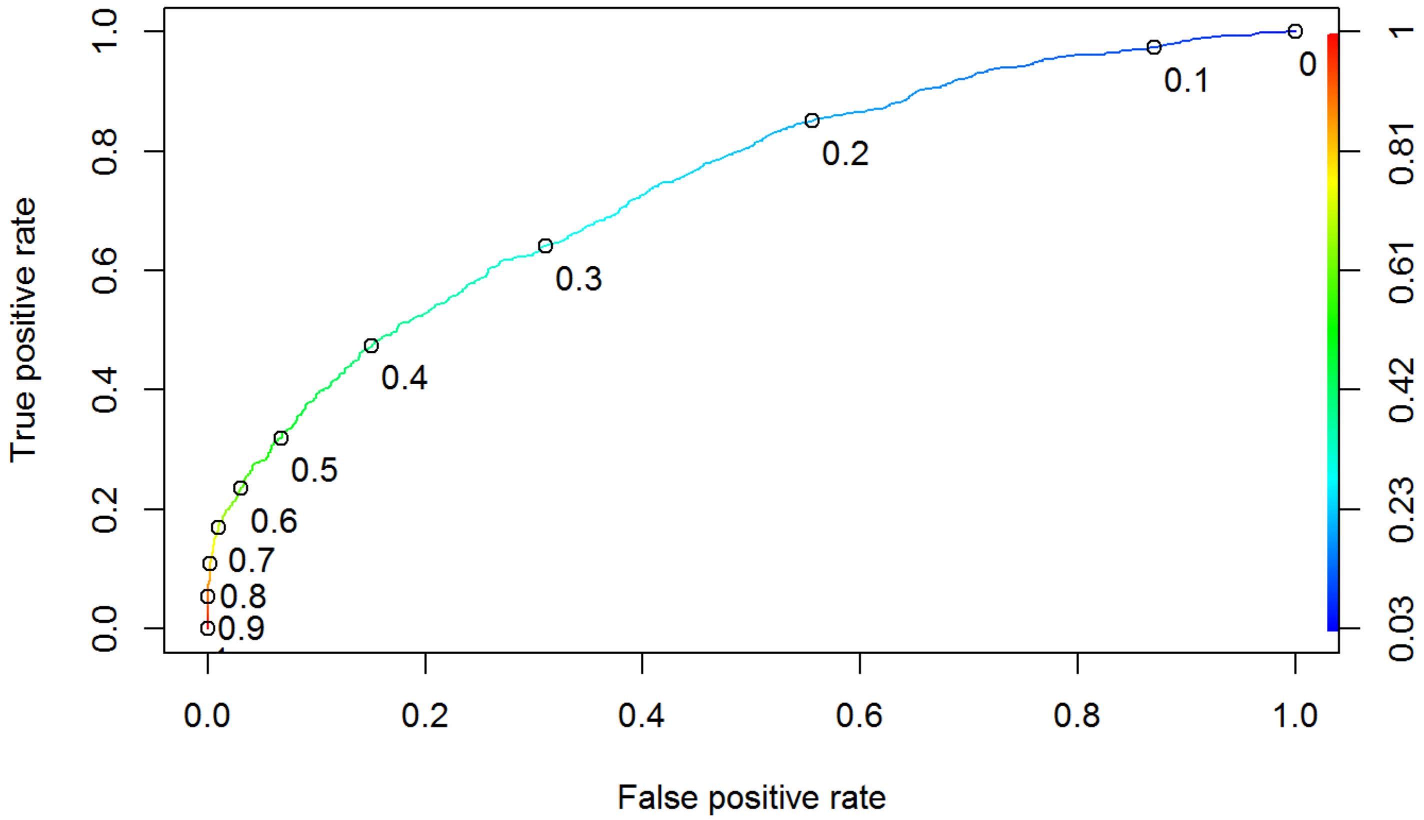
So, what is the ideal trade-off between sensitivity and specificity? Luckily, r has a package to help us understand the threshold cutoff.

This can be done using the following syntax

```
library(ROCR)

# Make predictions on training set
predictTrain = predict(modLog2, type="response")

# Prediction function
ROCRpred = prediction(predictTrain, train$not.fully.paid)

# Performance function
ROCRperf = performance(ROCRpred, "tpr", "fpr")

# Plot ROC curve
plot(ROCRperf)
```




```
# Add colors
plot(ROCRperf, colorize=TRUE)

# Add threshold labels
plot(ROCRperf, colorize=TRUE, print.cutoffs.at=seq(0,1,by=0.1), text.adj=c(-0.2,1.7
))
```



We see that probably the cutoff value of 0.35 is giving us a better tradeoff between sensitivity and specificity. Let's use this threshold to see how our model performs in the test data set.

Testing the new threshold on the test data set and call it vector t1

```
t1 = table(test$not.fully.paid, as.numeric(test$predicted.risk >= 0.35))
t1
```

```
##
##       0   1
##   0 828 212
##   1 219 241
```

```
#Overall Accuracy = (828 + 241) / nrow(test) = 0.71
#Sensitivity = 241 / 460 = 0.52
#Specificity = 828/ 1040 = 0.80
```

Interpretation

At a cutoff of 0.35, the sensitivity sees a drastic improvement from 33% in the original model to 52%. And the Overall Accuracy of the model is also not compromised much as it is still considerably better at 71% than the baseline accuracy of 69%.

Conclusion

Banks and Financial Institutions can use this model to create a Loan Acceptance Strategy for every Loan Applications and minimise the Bad Loan Error Rate from their portfolio.

In future blogs, I will be covering Decision Trees and Random Forests for carrying out the same exercise. Please note that this is just an illustartion of one technique, there are other methods as well, which may perform better than this model.

However, logistic regression remains one of the most widely used classification technique in Credit Risk Modeling.