Loan Default Project

You Li

7/23/2019

## Introduction

Different loan lenders faced a common problem: how to distinguish the loan applicants who are likely to default on their loans. As data scientists, we could analysis the loan status variable, and build regression models to find out the correlation between the variables and the unqualified applicants.

## Part 1

knitr::opts\_chunk$set(message = FALSE, warning = FALSE, fig.height=4, fig.width=5.5)

## load data  
loan<-read\_csv("loans50k.csv")  
head(loan)

## # A tibble: 6 x 32  
## loanID amount term rate payment grade employment length home income  
## <int> <int> <chr> <dbl> <dbl> <chr> <chr> <chr> <chr> <dbl>  
## 1 188861 8000 36 m… 0.14 272. C Warehouse… 3 yea… RENT 49966  
## 2 517703 11000 36 m… 0.1 355. B Vice Pres… 10+ y… MORT… 50000  
## 3 268587 35000 36 m… 0.15 1220. D Owner/Att… 10+ y… RENT 360000  
## 4 579902 20000 60 m… 0.12 448. C Analyst 2 yea… MORT… 60000  
## 5 617630 12000 60 m… 0.12 267. B medical t… 10+ y… MORT… 64000  
## 6 29919 10000 36 m… 0.14 340. C Assistant… < 1 y… RENT 108000  
## # ... with 22 more variables: verified <chr>, status <chr>, reason <chr>,  
## # state <chr>, debtIncRat <dbl>, delinq2yr <int>, inq6mth <int>,  
## # openAcc <int>, pubRec <int>, revolRatio <dbl>, totalAcc <int>,  
## # totalPaid <dbl>, totalBal <int>, totalRevLim <int>, accOpen24 <int>,  
## # avgBal <int>, bcOpen <int>, bcRatio <dbl>, totalLim <int>,  
## # totalRevBal <int>, totalBcLim <int>, totalIlLim <int>

This dataset includes 30 variables for 50,000 loans.

Create a response variable based on loan statuses for logistic regression model.

loan<-  
 loan %>%  
 mutate(response=case\_when(loan$status=="Fully Paid" ~ "Great",  
 loan$status=="Charged Off"| loan$status=="Default" ~ "Bad"))  
   
newloan<-subset(loan,!is.na(loan$response))

Loan ID doesn’t affect loan statuses, so remove loan ID from the predictor variables. The response variable has been created based on “status”, “status” is not a predictor variable, so it should be removed too.“Payment” is a similar predictor variable as “amount”, so remove it from predictor variables.

drops2<-c("loanID","status","payment")  
newloan<-newloan[ , !(names(newloan) %in% drops2)]

Use sapply() function to count the number of each variable that contains ‘NA’. There are many missing values in “employment”,“bcOpen” and " bcRatio“.

sapply(newloan, function(x) sum(is.na(x)))

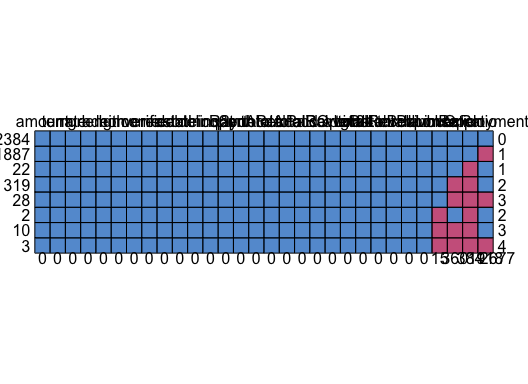
## amount term rate grade employment length   
## 0 0 0 0 1918 0   
## home income verified reason state debtIncRat   
## 0 0 0 0 0 0   
## delinq2yr inq6mth openAcc pubRec revolRatio totalAcc   
## 0 0 0 0 15 0   
## totalPaid totalBal totalRevLim accOpen24 avgBal bcOpen   
## 0 0 0 0 0 360   
## bcRatio totalLim totalRevBal totalBcLim totalIlLim response   
## 384 0 0 0 0 0

The number of per column is revealed below.

sapply(newloan, function(x) length(unique(x)))

## amount term rate grade employment length   
## 1203 2 25 7 15288 12   
## home income verified reason state debtIncRat   
## 3 3693 3 13 49 3845   
## delinq2yr inq6mth openAcc pubRec revolRatio totalAcc   
## 15 7 54 14 118 101   
## totalPaid totalBal totalRevLim accOpen24 avgBal bcOpen   
## 33783 32160 2382 33 19647 15837   
## bcRatio totalLim totalRevBal totalBcLim totalIlLim response   
## 1098 30402 29099 1617 24317 2

## check where the NAs locate  
md.pattern(newloan)



## amount term rate grade length home income verified reason state  
## 32384 1 1 1 1 1 1 1 1 1 1  
## 1887 1 1 1 1 1 1 1 1 1 1  
## 22 1 1 1 1 1 1 1 1 1 1  
## 319 1 1 1 1 1 1 1 1 1 1  
## 28 1 1 1 1 1 1 1 1 1 1  
## 2 1 1 1 1 1 1 1 1 1 1  
## 10 1 1 1 1 1 1 1 1 1 1  
## 3 1 1 1 1 1 1 1 1 1 1  
## 0 0 0 0 0 0 0 0 0 0  
## debtIncRat delinq2yr inq6mth openAcc pubRec totalAcc totalPaid  
## 32384 1 1 1 1 1 1 1  
## 1887 1 1 1 1 1 1 1  
## 22 1 1 1 1 1 1 1  
## 319 1 1 1 1 1 1 1  
## 28 1 1 1 1 1 1 1  
## 2 1 1 1 1 1 1 1  
## 10 1 1 1 1 1 1 1  
## 3 1 1 1 1 1 1 1  
## 0 0 0 0 0 0 0  
## totalBal totalRevLim accOpen24 avgBal totalLim totalRevBal  
## 32384 1 1 1 1 1 1  
## 1887 1 1 1 1 1 1  
## 22 1 1 1 1 1 1  
## 319 1 1 1 1 1 1  
## 28 1 1 1 1 1 1  
## 2 1 1 1 1 1 1  
## 10 1 1 1 1 1 1  
## 3 1 1 1 1 1 1  
## 0 0 0 0 0 0  
## totalBcLim totalIlLim response revolRatio bcOpen bcRatio employment  
## 32384 1 1 1 1 1 1 1  
## 1887 1 1 1 1 1 1 0  
## 22 1 1 1 1 1 0 1  
## 319 1 1 1 1 0 0 1  
## 28 1 1 1 1 0 0 0  
## 2 1 1 1 0 1 0 1  
## 10 1 1 1 0 0 0 1  
## 3 1 1 1 0 0 0 0  
## 0 0 0 15 360 384 1918  
##   
## 32384 0  
## 1887 1  
## 22 1  
## 319 2  
## 28 3  
## 2 2  
## 10 3  
## 3 4  
## 2677

Use mice package to impute missing numeric values.

Since employment is a factor variables with many missing values, it’s difficult to impute the missing values for employment appropriately. So the employment has to be removed.

employment<-as.data.frame(table(newloan2$employment))  
table(employment$Freq)

##   
## 1 2 3 4 5 6 7 8 9 10 11 12   
## 12572 1238 442 257 158 90 78 51 48 31 31 18   
## 13 14 15 16 17 18 19 20 21 22 23 24   
## 24 12 21 8 12 4 17 12 7 9 7 13   
## 25 26 27 28 29 30 31 32 33 34 35 36   
## 8 9 5 9 5 3 3 2 2 2 5 2   
## 38 39 40 41 42 45 46 47 49 50 53 56   
## 3 5 4 4 2 1 1 2 2 3 1 2   
## 59 61 64 68 72 74 75 76 81 87 89 91   
## 2 1 1 1 1 1 1 2 1 1 1 1   
## 92 94 95 98 99 102 108 113 115 118 126 128   
## 1 1 1 1 1 1 1 1 1 1 1 1   
## 130 137 144 147 152 201 210 214 232 268 271 274   
## 1 1 1 1 2 2 1 1 1 1 1 1   
## 583 627   
## 1 1

drops<-c("employment")  
newloan2<-newloan2[ , !(names(newloan2) %in% drops)]

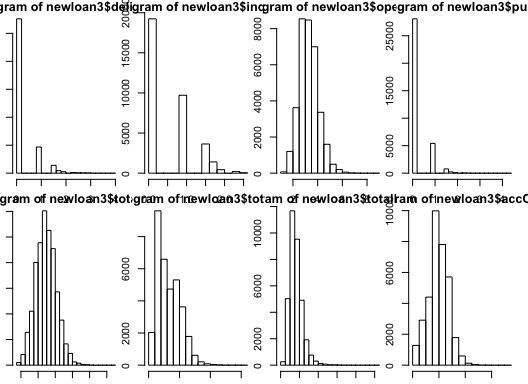
Now there is no missing values in the dataset.

sapply(newloan2, function(x) sum(is.na(x)))

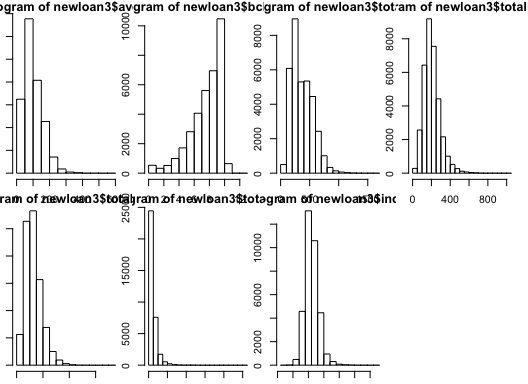
## amount term rate grade length home   
## 0 0 0 0 0 0   
## income verified reason state debtIncRat delinq2yr   
## 0 0 0 0 0 0   
## inq6mth openAcc pubRec revolRatio totalAcc totalPaid   
## 0 0 0 0 0 0   
## totalBal totalRevLim accOpen24 avgBal bcOpen bcRatio   
## 0 0 0 0 0 0   
## totalLim totalRevBal totalBcLim totalIlLim response   
## 0 0 0 0 0

We can see that there are still some variables can’t be transformed to an approximately normal distribution, such as “negative records”, “bcRatio”,“debtlncRat”.

par(mfrow=c(2,4),mar=c(1,1,1,1))  
hist(newloan3$delinq2yr)  
hist(newloan3$inq6mth)  
hist(newloan3$openAcc)  
hist(newloan3$pubRec)  
hist(newloan3$totalAcc)  
hist(newloan3$totalBal)  
hist(newloan3$totalRevLim)  
hist(newloan3$accOpen24)



hist(newloan3$avgBal)  
hist(newloan3$bcRatio)  
hist(newloan3$totalLim)  
hist(newloan3$totalRevBal)  
hist(newloan3$totalBcLim)  
hist(newloan3$totalIlLim)  
hist(newloan3$income)



## Part 2

Use 1 to represent “Great” and 0 to represent “Bad” in response variable.

data<-  
 newloan2 %>%  
 mutate(response=ifelse(response=="Great",1,0))

Seperate the datasets as trained and tested data.

set.seed(123)  
train <- sample(1:nrow(data), round(0.8\*nrow(data)))  
#length(train)  
data\_train<-data[train,]  
data\_test<-data[-train,]

totalPaid, cannot be used as a predictor variable because it is information that cannot be known before the loan is issued. so not to include it as a predictor in the models.

dropstotalpaid<-c("totalPaid")  
data\_train<-data\_train[ , !(names(data\_train) %in% dropstotalpaid)]

Using the generalized linear model, glm() function, make a logistic regression analysis using ‘response’ feature as outcome, with the rest of features in the training dataset as independent predictors. Specified binomial in the family argument will analyze the data using logistic regression.

full<-glm(response~.,data=data\_train,family=“binomial” ) summary(full)

Use the step function with backward selection to find a better model for predicting the response variable . null<-glm(response~1, data=data\_train,family=“binomial” ) full<-glm(response~.,data=data\_train,family=“binomial” ) stepAIC(null,scope=list(lower=null, upper=full),direction = “forward”)

Use the step function with backward selection to find a better model for predicting the response variable

full<-glm(response~.,data=data\_train,family=“binomial” ) stepAIC(full,direction = “backward”)

Compare AIC values of the two models produced by the backward and forward selection procedure.

backward<-glm( response ~ grade + term + avgBal + debtIncRat +   
 accOpen24 + totalAcc + home + state + bcOpen + length + delinq2yr +   
 amount + inq6mth + revolRatio + totalBcLim + totalIlLim +   
 totalRevBal + totalRevLim + rate, family = "binomial", data = data\_train)  
  
forward<-glm( response ~ amount + term + rate + grade + length +   
 home + state + debtIncRat + delinq2yr + inq6mth + revolRatio +   
 totalAcc + totalRevLim + accOpen24 + avgBal + bcRatio + totalRevBal +   
 totalBcLim + totalIlLim, family = "binomial", data = data\_train)  
  
extractAIC(backward)

## [1] 83.00 26020.94

extractAIC(forward)

## [1] 83.00 26018.28

Forward model has smaller AIC value (26154.54), which means forward model is better. Use 0.5 as the cutoff probability to construct a classification table (also known as a confusion matrix) for the model. The accuracy rate from the table is 0.79.

fit<-glm( response ~ amount + term + rate + grade + length +   
 home + state + debtIncRat + delinq2yr + inq6mth + revolRatio +   
 totalAcc + totalRevLim + accOpen24 + avgBal + bcRatio + totalRevBal +   
 totalBcLim + totalIlLim, family = "binomial", data = data\_test)  
pre<-predict(fit,data=data\_test,type="response")  
fitted.results<-ifelse(pre>0.5,1,0)  
t<-table(data\_test$response,fitted.results)  
addmargins(t)

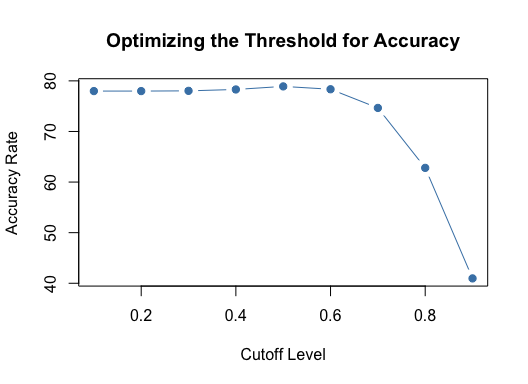
## fitted.results  
## 0 1 Sum  
## 0 216 1311 1527  
## 1 151 5253 5404  
## Sum 367 6564 6931

accuracy\_rate<-(209 +5283)/(359 +6572)  
accuracy\_rate

## [1] 0.7923821

Based on the plot Optimizing the Threshold for Accuracy, we can find that when the threhold is 0.5, the accuracy rate reach the highest. If the threhould is greater than 0.6, the acuuracy rate decreases.

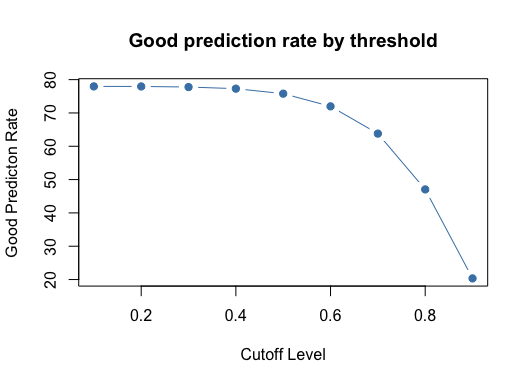
cutoffs <- seq(0.1,0.9,0.1)  
accuracy <- NULL  
for (i in seq(along = cutoffs)){  
 prediction <- ifelse(fit$fitted.values >= cutoffs[i], 1, 0) #Predicting for cut-off  
accuracy <- c(accuracy,length(which(data\_test$response ==prediction))/length(prediction)\*100)  
}  
  
plot(cutoffs, accuracy, pch =19,type='b',col= "steelblue",  
 main ="Optimizing the Threshold for Accuracy", xlab="Cutoff Level", ylab = "Accuracy Rate")

 totalPaid is information that only can be known after the loan is issued, so totalPaid can be used an index to define the loan is good or bad. Bamk get profit from the difference that totalPaid minus amout.

data\_test2<-  
 data\_test %>%  
 mutate(profit=totalPaid-amount)

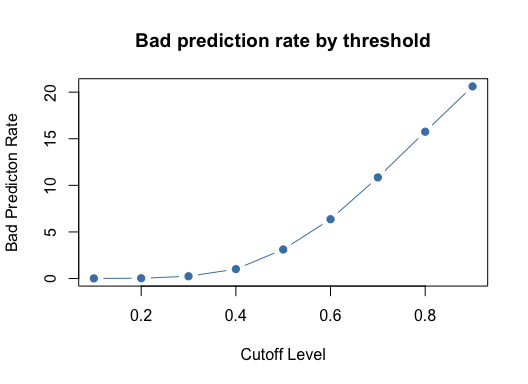
Create a plot for true positive rate and threshold. Based on the plot, we can find that when the threshold is 0.1, good loans prediction amount reach the highst.

cutoffs <- seq(0.1,0.9,0.1)  
accuracy\_possitive<- NULL  
for (i in seq(along = cutoffs)){  
 prediction <- ifelse(fit$fitted.values >= cutoffs[i], 1, 0) #Predicting for cut-off  
accuracy\_possitive <- c(accuracy\_possitive,length(which(prediction[which(data\_test$response==1)]==1))/length(prediction)\*100)  
}  
  
plot(cutoffs, accuracy\_possitive, pch =19,type='b',col= "steelblue",  
 main ="Good prediction rate by threshold", xlab="Cutoff Level", ylab = "Good Predicton Rate")

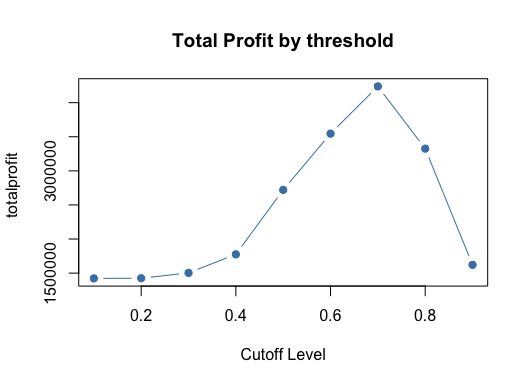


Create a plot for true negative rate,Based on the plot,we can find that when the threshold is 0.9, bdd loans prediction amount reach the highst.

cutoffs <- seq(0.1,0.9,0.1)  
accuracy\_negative<- NULL  
for (i in seq(along = cutoffs)){  
 prediction <- ifelse(fit$fitted.values >= cutoffs[i], 1, 0) #Predicting for cut-off  
accuracy\_negative <- c(accuracy\_negative,length(which(prediction[which(data\_test$response==0)]==0))/length(prediction)\*100)  
}  
  
plot(cutoffs, accuracy\_negative, pch =19,type='b',col= "steelblue",  
 main ="Bad prediction rate by threshold", xlab="Cutoff Level", ylab = "Bad Predicton Rate")



cutoffs <- seq(0.1,0.9,0.1)  
totalprofit<- NULL  
for (i in seq(along = cutoffs)){  
 prediction <- ifelse(fit$fitted.values >= cutoffs[i], 1, 0) #Predicting for cut-off  
totalprofit <- c(totalprofit,sum(data\_test2$profit[which(prediction==1)]))  
}  
  
plot(cutoffs, totalprofit , pch =19,type='b',col= "steelblue",  
 main ="Total Profit by threshold", xlab="Cutoff Level", ylab = "totalprofit ")



fitted.results2<-ifelse(pre>0.7,1,0)  
best<-sum(data\_test2$profit[which(fitted.results2==1)])  
best

## [1] 4238637

Summary:

From the study of the plots, we found that for any given threshold value (0~1), the accuracy of model (true positive+ true negative), the probability of true positive, and the total profit are fixed. The accuracy rate is generally consistent around 80% for threshold value ≤ 0.6. When threshold is greater than 0.6, the model accuracy quickly drops. In contrast, the total profit (from predicted “good” loans) increases with threshold until 0.7, followed by a sharp drop.

Hence, it is concluded that the bank should balance between model accuracy and total profit for decision making. In this case, looks like threshold value equals to 0.6 or 0.7, gives highest possible total profit. A quick check could be using “total profit” times “accuracy”. For threshold=0.6, it is 3545908. While threshold=0.7, it is 4249183. The take home message is to consider model accuracy together with final output result.