Yodlee's Loan Prediction Challange using R

Dhanasekaran

May 5, 2016



# **Objective**:

The challange is to model loan applications and predict whether they will turn bad or not.

In the dataset, the first row contains the labels of the attributes. In the training dataset, there is an additional final column (named as *Loan Status*) which has the information on whether the loans will turn bad (0) or not (1).

Our task here is to split the given training dataset into train and test (i.e., hold-out sample) samples and build predictive models on the training sample using various algorithms including ensemble methods and test each of the models result against the test sample. Since the test sample is also drawn from the training dataset and also contains the response variable (Loan Status), we just make use of it to cross-validate our model accuracy.

Labels of attributes is shown below.

**Variables**

Loan Application Number | Dept to Income Ratio (a ratio calculated based on | borrower's monthly dept repayments to self reported monthly | income) Loan Amount | Borrower delinquency in last two years Loan duration | Date of borrower's first loan Interest rate | Number of months since the borrower's last delinquency EMI | The number of months since the borrower's credit record was | updated Borrower rating by bank | Number of times borrower has availed the loan from bank Borrower's duration of employment | Number of negative comments about the borrower in credit | history Home ownership of borrower | Total credit revolving balance Annual income of borrower | Percentage of credit the borrower is using relative to all | available revolving credit Borrower's verification status | Number of times the borrower has availed loan from all | banks Loan issue date | Late fees received to date Purpose of loan | Last month payment was received State | Loan amount received as payment **Loan Status** |

## Setting up the working directory in R

Load required R packages for this exercise

setwd("D:\\Kaggle\\Yodlee")  
  
library(tree)  
library(dplyr)  
library(adabag)  
library(glm2)

Read the dataset into R data frame object

loan\_data <- read.csv("Training.csv", sep = ",", header = T, stringsAsFactors = FALSE)

We will just view top 5 rows of the data frame as below.

head(loan\_data, row.names = FALSE, 5)

## Loan\_No Loan\_Amount Loan\_Duration Interest\_Rate EMI Bank\_Rating  
## 1 1 102426 47 months 11.88% 2736.001 CD  
## 2 2 105595 46 months 15.46% 3057.054 DC  
## 3 3 33098 41 months 13.17% 1006.782 CB  
## 4 4 197814 41 months 13.85% 6083.006 CE  
## 5 5 198063 38 months 13.38% 6422.936 CB  
## Employment Home\_Ownership Annual\_Income Verification\_Status  
## 1 3 years Rental 690268 Not Verified  
## 2 6 years Under Mortgage 777236 Not Verified  
## 3 3 years Under Mortgage 389424 Not Verified  
## 4 10 years and above Rental 555311 Verified  
## 5 10 years and above Rental 901968 Verified  
## Loan\_Date Loan\_Purpose State Debt\_to\_Income\_Ratio  
## 1 18-02-2008 Pay Credit Card Bill Jharkhand 20.49246  
## 2 29-12-2009 Others Rajasthan 21.01779  
## 3 22-11-2009 Marriage Meghalaya 22.89817  
## 4 24-06-2009 Pay-off Other Loans Tripura 13.22313  
## 5 27-02-2010 Others Uttar Pradesh 19.48961  
## Delinquency First\_Loan\_Date Last\_Delinquency Card\_Updated No\_of\_Loans  
## 1 0 9/12/1990 35 0 17  
## 2 0 29-04-1999 70 NA 4  
## 3 0 9/6/1995 NA 113 10  
## 4 0 15-11-1994 NA 110 9  
## 5 0 19-03-1997 NA NA 7  
## Negative\_Comments Credit\_Balance Percentage\_Used\_Credit  
## 1 0 186720 0.6536955  
## 2 0 374798 0.8896403  
## 3 1 142602 0.8042069  
## 4 1 67211 0.8218125  
## 5 0 50619 0.4797962  
## No\_of\_Loans\_Other\_banks Late\_Fees Last\_payment\_Date Last\_Amount  
## 1 32 0 1/11/2011 2769.771  
## 2 9 0 1/5/2013 6249.914  
## 3 19 0 1/2/2013 1049.132  
## 4 13 0 1/2/2013 266.200  
## 5 31 0 1/2/2011 152077.236  
## Loan\_Status  
## 1 1  
## 2 1  
## 3 1  
## 4 1  
## 5 1

Since we have '%' and 'months' attached with the values in EMI column and Loan Duration column and to convert them into factor variables, we will remove '%' and 'months' from the respective columns. '%>%' is an operator used within *dplyr* package.

loan\_data <- loan\_data %>%  
 mutate(Loan\_Duration = as.integer(sub('months', '', Loan\_Duration)),  
 Interest\_Rate = round(as.integer(sub('%', '', Interest\_Rate))\*100, 2),  
 Bank\_Rating = as.factor(Bank\_Rating),   
 Employment = as.factor(Employment),  
 Home\_Ownership = as.factor(Home\_Ownership),  
 Verification\_Status = as.factor(Verification\_Status),  
 Loan\_Purpose = as.factor(Loan\_Purpose),  
 State = as.factor(State),  
 Loan\_Status = as.factor(Loan\_Status))

# Feature Engineering

We cannot use *State* as one the predictors in our model due to the fact that it has more than 32 factors hence it is decided to compute the frequency of loan applications per State and use it as one of the predictors. Likewise the frequency of loans are computed by bank rating and used. This task is generaly known as **Feature Engineering** which helps to improve the prediction accuracy of the model.

l1 <- loan\_data %>%  
 group\_by(State) %>%   
 summarise(loans\_by\_state = n())  
  
l2 <- loan\_data %>%  
 group\_by(Bank\_Rating) %>%  
 summarise(loans\_by\_Rating = n())  
   
loan\_data <- loan\_data %>%  
 left\_join(l1, by = 'State') %>%  
 left\_join(l2, by = 'Bank\_Rating')

# Missing values replacement

Missing values are found in the following columns and removed using repective average.

* No\_of\_Loans (Number of times the borrower has availed the loan from the bank)
* Credit\_Balance (Total Credit Revolving balance)
* Percentage\_Used\_Credit (Percentage of Credit the Borrower is Using Relative to All Available Revolving Credit)
* No\_of\_Loans\_Other\_Banks (Number of Times the Borrower has Availed Loan from All Banks)

loan\_data <- loan\_data %>%  
 mutate(loan\_to\_income = round(Loan\_Amount/Annual\_Income, 2),  
 No\_of\_Loans = ifelse(is.na(No\_of\_Loans), mean(No\_of\_Loans, na.rm = TRUE),  
 No\_of\_Loans),  
 Credit\_Balance = ifelse(is.na(Credit\_Balance), mean(Credit\_Balance, na.rm = TRUE),  
 Credit\_Balance),  
 Percentage\_Used\_Credit = ifelse(is.na(Percentage\_Used\_Credit),  
 mean(Percentage\_Used\_Credit, na.rm = TRUE), Percentage\_Used\_Credit),  
 No\_of\_Loans\_Other\_banks = ifelse(is.na(No\_of\_Loans\_Other\_banks),  
 mean(No\_of\_Loans\_Other\_banks, na.rm = TRUE), No\_of\_Loans\_Other\_banks))

Exclude the identity variable *Loan No* (Loan Application Number) and other columns with date stamp.

loan\_data <- loan\_data[,-c(1, 2, 6, 9, 11, 13, 16, 17, 18, 25)]

# Cross validation samples

Here we split the data set into two portions as required (70% - Train; 30% - Test)

set.seed(2)  
sam.size <- floor(.7\*nrow(loan\_data))  
train\_ind <- sample(seq\_len(nrow(loan\_data)), size = sam.size)  
loan\_train <- loan\_data[train\_ind,]  
loan\_test <- loan\_data[-train\_ind,]  
  
loan\_Status\_test <- loan\_test[,17]   
loan\_test <- loan\_test[,-17]

# Logistic Regression Model

Initially two GLM models were studied and based on the results obtained, GLM3 (glm.fit3) is fit and finalised. Parameters with *P* values less than 0.05 were removed in steps and arrived to the final model.

set.seed(123)  
glm.fit1 <- glm(Loan\_Status ~ EMI + Verification\_Status + Negative\_Comments + No\_of\_Loans\_Other\_banks + Late\_Fees + Last\_Amount +  
 loans\_by\_state + loan\_to\_income, data = loan\_train, family = binomial)  
summary(glm.fit1)

##   
## Call:  
## glm(formula = Loan\_Status ~ EMI + Verification\_Status + Negative\_Comments +   
## No\_of\_Loans\_Other\_banks + Late\_Fees + Last\_Amount + loans\_by\_state +   
## loan\_to\_income, family = binomial, data = loan\_train)  
##   
## Deviance Residuals:   
## Min 1Q Median 3Q Max   
## -6.2664 0.0014 0.2788 0.6456 3.4934   
##   
## Coefficients:  
## Estimate Std. Error z value Pr(>|z|)  
## (Intercept) 2.127e+00 7.985e-02 26.637 < 2e-16  
## EMI -1.381e-04 1.560e-05 -8.850 < 2e-16  
## Verification\_StatusSource Verified -2.736e-01 6.024e-02 -4.541 5.60e-06  
## Verification\_StatusVerified -2.223e-01 6.014e-02 -3.696 0.000219  
## Negative\_Comments -5.475e-01 8.028e-02 -6.820 9.13e-12  
## No\_of\_Loans\_Other\_banks -3.861e-03 2.250e-03 -1.716 0.086230  
## Late\_Fees -3.576e-03 2.541e-04 -14.071 < 2e-16  
## Last\_Amount 1.557e-04 7.831e-06 19.888 < 2e-16  
## loans\_by\_state -6.540e-05 1.797e-05 -3.639 0.000273  
## loan\_to\_income -2.659e+00 2.476e-01 -10.738 < 2e-16  
##   
## (Intercept) \*\*\*  
## EMI \*\*\*  
## Verification\_StatusSource Verified \*\*\*  
## Verification\_StatusVerified \*\*\*  
## Negative\_Comments \*\*\*  
## No\_of\_Loans\_Other\_banks .   
## Late\_Fees \*\*\*  
## Last\_Amount \*\*\*  
## loans\_by\_state \*\*\*  
## loan\_to\_income \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## (Dispersion parameter for binomial family taken to be 1)  
##   
## Null deviance: 13819 on 16787 degrees of freedom  
## Residual deviance: 10914 on 16778 degrees of freedom  
## (12 observations deleted due to missingness)  
## AIC: 10934  
##   
## Number of Fisher Scoring iterations: 9

glm.fit2 <- glm(Loan\_Status ~ EMI + Late\_Fees + Last\_Amount +  
 loans\_by\_state + loan\_to\_income, data = loan\_train, family = binomial)  
summary(glm.fit2)

##   
## Call:  
## glm(formula = Loan\_Status ~ EMI + Late\_Fees + Last\_Amount + loans\_by\_state +   
## loan\_to\_income, family = binomial, data = loan\_train)  
##   
## Deviance Residuals:   
## Min 1Q Median 3Q Max   
## -6.2717 0.0014 0.2898 0.6665 3.5631   
##   
## Coefficients:  
## Estimate Std. Error z value Pr(>|z|)   
## (Intercept) 1.651e+00 5.256e-02 31.409 < 2e-16 \*\*\*  
## EMI -2.299e-04 1.218e-05 -18.871 < 2e-16 \*\*\*  
## Late\_Fees -3.372e-03 2.507e-04 -13.449 < 2e-16 \*\*\*  
## Last\_Amount 1.560e-04 7.754e-06 20.114 < 2e-16 \*\*\*  
## loans\_by\_state -5.249e-05 1.770e-05 -2.965 0.00302 \*\*   
## loan\_to\_income -8.685e-02 7.579e-02 -1.146 0.25185   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## (Dispersion parameter for binomial family taken to be 1)  
##   
## Null deviance: 13826 on 16799 degrees of freedom  
## Residual deviance: 11107 on 16794 degrees of freedom  
## AIC: 11119  
##   
## Number of Fisher Scoring iterations: 9

glm.fit3 <- glm(Loan\_Status ~ EMI + Late\_Fees + Last\_Amount +  
 loans\_by\_state, data = loan\_train, family = binomial)  
summary(glm.fit3)

##   
## Call:  
## glm(formula = Loan\_Status ~ EMI + Late\_Fees + Last\_Amount + loans\_by\_state,   
## family = binomial, data = loan\_train)  
##   
## Deviance Residuals:   
## Min 1Q Median 3Q Max   
## -6.2791 0.0014 0.2899 0.6666 3.5679   
##   
## Coefficients:  
## Estimate Std. Error z value Pr(>|z|)   
## (Intercept) 1.642e+00 5.201e-02 31.572 < 2e-16 \*\*\*  
## EMI -2.324e-04 1.197e-05 -19.414 < 2e-16 \*\*\*  
## Late\_Fees -3.370e-03 2.507e-04 -13.442 < 2e-16 \*\*\*  
## Last\_Amount 1.561e-04 7.755e-06 20.126 < 2e-16 \*\*\*  
## loans\_by\_state -5.245e-05 1.770e-05 -2.964 0.00303 \*\*   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## (Dispersion parameter for binomial family taken to be 1)  
##   
## Null deviance: 13826 on 16799 degrees of freedom  
## Residual deviance: 11110 on 16795 degrees of freedom  
## AIC: 11120  
##   
## Number of Fisher Scoring iterations: 9

GLM model was then fit for test observations. *predict* is a R function used to apply the model fit object on to the new dataset and the obtained probabilties were recoded into class labels.

glm.probs <- predict(glm.fit3, newdata = loan\_test, type = 'response')  
glm.pred <- ifelse(glm.probs>0.5, 1, 0)

# Confusion matrix - Logistic Regression

*Confusion matrix* is generally the measure to validate the classification results obtained from the model. We calculate the proportion of observations which have the predicted class as same as the observed class using *mean* function which provides us with the model accuracy.

table(glm.pred, loan\_Status\_test)

## loan\_Status\_test  
## glm.pred 0 1  
## 0 62 72  
## 1 1029 6037

mean(glm.pred == loan\_Status\_test)

## [1] 0.8470833

mean(glm.pred != loan\_Status\_test)

## [1] 0.1529167

# Decision Tree Model

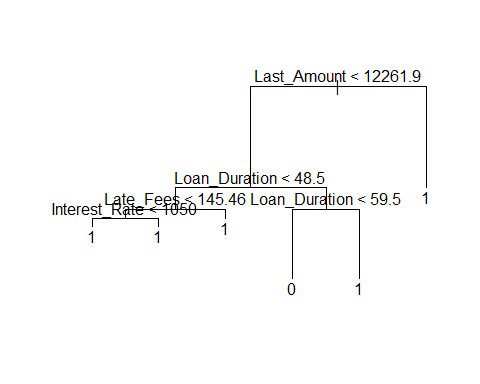
Decision tree model is built using tree package in R.

set.seed(123)  
train.tree <- tree(Loan\_Status ~., data = loan\_train)  
summary(train.tree)

##   
## Classification tree:  
## tree(formula = Loan\_Status ~ ., data = loan\_train)  
## Variables actually used in tree construction:  
## [1] "Last\_Amount" "Loan\_Duration" "Late\_Fees" "Interest\_Rate"  
## Number of terminal nodes: 6   
## Residual mean deviance: 0.5655 = 9491 / 16780   
## Misclassification error rate: 0.1059 = 1778 / 16788

Decision Tree Plot

plot(train.tree);text(train.tree, pretty = 0)



Decision tree model is then fit fot test observations and computed Confusion matrix.

tree.pred <- predict(train.tree, newdata = loan\_test, type = 'class')  
table(tree.pred,loan\_Status\_test)

## loan\_Status\_test  
## tree.pred 0 1  
## 0 335 68  
## 1 756 6041

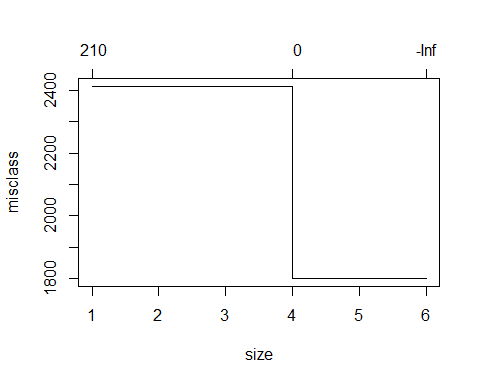
mean(tree.pred == loan\_Status\_test)

## [1] 0.8855556

# Tree Pruning using Cross Validation method

Prune the tree to select number of trees to end up with.

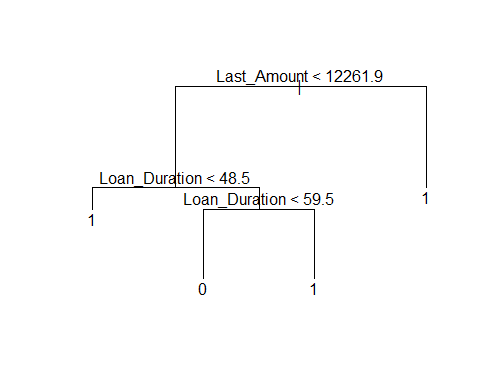
set.seed(123)  
cv.trees = cv.tree(train.tree, FUN = prune.misclass)  
plot(cv.trees)



## Application of pruned tree

The number of trees is found through pruning method and applied. O

set.seed(123)  
prune.tree <- prune.misclass(train.tree, best = 4)  
plot(prune.tree);text(prune.tree, pretty =0)



# Confusion Matrix - Decision Tree method

tree.pred <- predict(prune.tree, newdata = loan\_test, type = 'class')  
table(tree.pred, loan\_Status\_test)

## loan\_Status\_test  
## tree.pred 0 1  
## 0 335 68  
## 1 756 6041

mean(tree.pred == loan\_Status\_test)

## [1] 0.8855556

mean(tree.pred != loan\_Status\_test)

## [1] 0.1144444

# ADABOOST.M1 Algorithm

set.seed(123)  
adaboost.model <- boosting(Loan\_Status ~., data = loan\_train, boos = TRUE, coeflearn = 'Breiman')  
  
# adaboost.model$terms  
# adaboost.model$weights  
# adaboost.model$trees

Predict the boosting model on test observations and test errors are evaluated.

boost.pred <- predict.boosting(adaboost.model, newdata = loan\_test)  
predClass <- boost.pred$class

## Confusion matrix - **Adapative Boosting Technique**

table(predClass, loan\_Status\_test)

## loan\_Status\_test  
## predClass 0 1  
## 0 460 169  
## 1 631 5940

mean(predClass == loan\_Status\_test)

## [1] 0.8888889

mean(predClass != loan\_Status\_test)

## [1] 0.1111111

# Conclusion

In this paper, the R package *adabag* that implements the boosting algorithm is discussed and the error rate has substancially decreased from GLM model to AdaBoost model however the error rate acheived by using simple decision tree model is also similar as that of the adaboost model. And the overall accuracy has improved from 84% to 89%, which is beneficial. Therefore, boosting algorithm performs remarkably well over the other methods and this model can be further improved with the help of additional feature engineering of predictors.