**Loan Default Prediction Classification**

**FRA Assignment**



Great lakes Institute of Management, Chennai

Pursuing PG in Business Analytics and Business Intelligence (PGBABI) 2015-2016

**Assignment Members**:

|  |  |  |
| --- | --- | --- |
| BABI- | Arun Venkatesan |  |
| BABI- | Arun Sundar |  |
| BABI- | Rajagopalan Kannan | Rajagopalan1979@Gmail.com |
| BABI- | Vijay Somnath |  |

Contents

e

[1. Assignement overview 3](#_Toc453499137)

[1.1 Problem Statement 3](#_Toc453499138)

[1.2 Data Dictionary 3](#_Toc453499139)

[2. Data Preparation 4](#_Toc453499140)

[2.1 Read data 4](#_Toc453499141)

[2.2 Data set structure 4](#_Toc453499142)

[3. Exploratory Data Analysis 5](#_Toc453499143)

[3.1 Check for missing values 5](#_Toc453499144)

[3.2 Check for unbalanced class distribution 7](#_Toc453499145)

[3.3 Plots 10](#_Toc453499146)

[4. Data Pre-Processing 12](#_Toc453499147)

[4.1 Missing values imputation 12](#_Toc453499148)

[4.2 Variable Transformation 13](#_Toc453499149)

[5. Assess model accuracy without resolving imbalance issue 14](#_Toc453499150)

[5.1 Logitistic Regression Model 14](#_Toc453499151)

[5.2 eXtreme Gradient Boosting (xGBTree) Model 17](#_Toc453499152)

[5.3 Random Forest Model 21](#_Toc453499153)

[5.4 Tree bag Model 25](#_Toc453499154)

[5.5 H2o deep learning Model 29](#_Toc453499155)

[5.6 Ensemble (gbm, rpart, treebag) Model 32](#_Toc453499156)

[5.7 Various other models (used weka) 34](#_Toc453499157)

[6. Address imbalanced issue 35](#_Toc453499158)

[6.1 Using ROSE (Random Over-Sampling Examples) technique: 35](#_Toc453499159)

[7. Build Models 42](#_Toc453499160)

[1. Results 43](#_Toc453499161)

# Assignement overview

## Problem Statement

FRA Group assignment: Default modeling in R

You are provided data on loan defaults. Your objective is to build default prediction models using R. You would be evaluated on the following tasks:

* Cleaning of data
* Handling unbalanced data
* Building model(s) on training data
  + How many models? More the merrier
* Evaluating model performance using test data

## Data Dictionary

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Variable Name** | **Description** | **Type** | **IV / DV** | **Attribute Type** |
| SeriousDlqin2yrs | Person experienced 90 days past due delinquency or worse | Y/N | DV | Categorical |
| RevolvingUtilizationOfUnsecuredLines | Total balance on credit cards and personal lines of credit except real estate and no installment debt like car loans divided by the sum of credit limits | percentage | IV | Continuous |
| DebtRatio | Monthly debt payments, alimony,living costs divided by monthy gross income | percentage | IV | Continuous |
| NumberOfOpenCreditLinesAndLoans | Number of Open loans (installment like car loan or mortgage) and Lines of credit (e.g. credit cards) | integer | IV | Continuous |
| NumberOfDependents | Number of dependents in family excluding themselves (spouse, children etc.) | integer | IV | Categorical |

IV – Independent variable

DV – Dependent variable

# Data Preparation

## Read data

setwd('c:/r/data/csv/')

train <- read.csv('fra\_train.csv', header=TRUE)

test <- read.csv('fra\_test.csv', header=TRUE)

test\_dat <- read.csv('fra\_test.csv', header=TRUE)

> dim(train)

[1] 5000 6

> dim(test)

[1] 1000 6

Training data set: 5000 records, 6 attributes

Test data set: 1000 records, 6 attributes

## Data set structure

> str(train)

'data.frame': 5000 obs. of 6 variables:

$ Casenum : int 1 2 3 4 5 6 7 8 9 10 ...

$ SeriousDlqin2yrs : Factor w/ 2 levels "0","1": 2 1 1 1 1 1 1 1 1 1 ...

$ RevolvingUtilizationOfUnsecuredLines: num -0.2664 -0.0438 -0.4183 -1.4532 -0.0973 ...

$ DebtRatio : num -0.219 -2.105 -2.464 -3.323 -3.692 ...

$ NumberOfOpenCreditLinesAndLoans : num 2.565 1.386 0.693 1.609 1.946 ...

$ NumberOfDependents : Factor w/ 9 levels "0","1","2","3",..: 3 2 1 1 1 2 1 1 1 3 ...

> str(test)

'data.frame': 1000 obs. of 6 variables:

$ SeriousDlqin2yrs : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 2 1 ...

$ RevolvingUtilizationOfUnsecuredLines: num -2.25 -1.03e-01 -4.55e-01 -4.57 -1.00e-07 ...

$ DebtRatio : num 7.728 -0.462 -1.71 7.963 -Inf ...

$ NumberOfOpenCreditLinesAndLoans : num 1.61 2.4 2.77 2.77 1.39 ...

$ NumberOfDependents : Factor w/ 8 levels "0","1","2","3",..: 1 1 1 1 1 1 1 2 1 1 ...

# Exploratory Data Analysis

## Check for missing values

Require(Amelia)

missmap(train, col=c('red', 'white'))

missmap(test, col=c('red', 'white'))

There is only one variable that has missing values on both training and testing data set and it is evident from the below plots below.

|  |
| --- |
| Training data set |
| test data set |

Let us check the number of missing values by attributes.

|  |
| --- |
| There are 142 records out of total 5000 records in the training data set have missing values on ‘NumberofDependents’ attribute  > sapply(train, function(x) sum(is.na(x)))  Casenum  0  SeriousDlqin2yrs  0  RevolvingUtilizationOfUnsecuredLines  0  DebtRatio  0  NumberOfOpenCreditLinesAndLoans  0  **NumberOfDependents**  **142** |
| There are 20 records out of total 1000 records in the test data set have missing values on ‘NumberofDependents’ attribute  > sapply(test, function(x) sum(is.na(x)))  Casenum  0  SeriousDlqin2yrs  0  RevolvingUtilizationOfUnsecuredLines  0  DebtRatio  0  NumberOfOpenCreditLinesAndLoans  0  **NumberOfDependents**  **20** |

We will later impute this ‘NumberofDependents’ attribute with the appropriate imputation mehtods that are available.

## Check for unbalanced class distribution

|  |
| --- |
| **Training data set**  > table(train$SeriousDlqin2yrs)  0 1  4695 305  We can see from this output that the outcome class is skewed. |
| Let us computute the percentage of class distribution.  > prop.table(table(train$SeriousDlqin2yrs))  0 1  0.939 0.061  We can see that the outcome type of ‘0’ constitute 94% of the total data set and the type ‘1’ constitute the remaining 6% fo the the training data set. This proves that the outcome class distribution is imbalanced and it might require some treatment prior to model building. |
| **Testing data set**  table(test$SeriousDlqin2yrs)  0 1  937 63  We can see from this output that the outcome class is skewed. |
| prop.table(table(test$SeriousDlqin2yrs)  0 1  0.937 0.063  We can see that the outcome type of ‘0’ constitute 94% of the total data set and the type ‘1’ constitute the remaining 6% fo the the testing data set. This proves that the outcome class distribution is imbalanced and it might require some treatment prior to model building. |

Let’s get a Visual representation of imbalanced class distribution on the training and testing data set

> ggplot(train, aes(NumberOfOpenCreditLinesAndLoans,

NumberOfDependents)) +

geom\_jitter(aes(colour=factor(SeriousDlqin2yrs)))

|  |
| --- |
|  |
|  |

|  |
| --- |
| **Imbalanced overview**:  Imbalanced data typically refers to a problem with classification problems where the classes are not represented equally.  In this case we may have a 2-class (binary) classification problem with 5000 instances (rows) in a training data set. A total of 4695 instances are labeled with Class-0 and the remaining 305 instances are labeled with Class-1.  This is an imbalanced dataset and the percentage ratio of Class-0 to Class-1 instances is 94% : 6% |

Recommended techniques to resolve imbalanced data issue and these will be studied in detail.

|  |  |  |
| --- | --- | --- |
| **#** | **Technique** | **Remarks** |
| 1 | Collect more data | NA. |
| 2 | Change performance metric | We will study and try this |
| 3 | Resampling data set | We will study and try this |
| 4 | Generate synthetic samples | We will study and try SMOTE on the data set |
| 5 | Try different algorithms | We will study and try this |
| 6 | Try Penalized models | We will study and try this |
| 7 | Anamoly / Change detection | We will study and try this |
| 8 | Down sampling using random Forest | We will study and try this |
| 9 | Random Oversampling Examples (ROSE) | We will study and try this |

## Plots

We will not continue our exploratory data analysis by drawing few plots and understand the data set little further.

|  |
| --- |
| > source('helper.R')  > hgraph(train) |
| We have drawn the histogram on all the attributes of the training data set and below is the output. It can be seen from the plot that   * SeriousDlqin2yrs attribute has unbalanced class distribution and this attribute is a dependent or outcome variable in this project. * Following discrete attributes are right skewed and these will be normalized through transformations for its use in the algorithms whereever it is required.   + RevolvingUtilizationOfUnsecuredLines   + DebtRatio   + NumberOfOpenCreditLinesAndLoans |

|  |
| --- |
| > hgraph(test)  Similar observations are noted in the testing data set as well. i.e.,   * SeriousDlqin2yrs attribute has unbalanced class distribution and this attribute is a dependent or outcome variable in this project. * Following discrete attributes are right skewed and these will be normalized through transformations for its use in the algorithms whereever it is required.   + RevolvingUtilizationOfUnsecuredLines   + DebtRatio   + NumberOfOpenCreditLinesAndLoans |
|  |

# Data Pre-Processing

## Missing values imputation

‘NumberofDependents’ is the only attribute that has missing values in both the training and testing data set, let’s impute this using a mode imputation method for now and later we will revisit other imputation method if this attribute turns out to be a signicant for prediction.

getmode <- function(v) {

uniqv <- unique(v)

uniqv[which.max(tabulate(match(v, uniqv)))]

}

train$NumberOfDependents[is.na(train$NumberOfDependents)] <- getmode(train$NumberOfDependents)

test$NumberOfDependents[is.na(test$NumberOfDependents)] <- getmode(test$NumberOfDependents)

we will try the following imputation methods later.

1. Bag imputation
2. kNN imputation
3. missForest Imputation
4. Hmisc imputation
5. Mi imputation
6. Mice imputation
7. Or any other suitable logic.

## Variable Transformation

Following variables will be normalized using log transformation, but this will be applied only on algorithms that require normalization requirements. Models like logistic (or) xgbtree will not require its predictors to be normally distributed.

train$RevolvingUtilizationOfUnsecuredLines <- log(train$RevolvingUtilizationOfUnsecuredLines)

train$DebtRatio <- log(train$DebtRatio)

train$NumberOfOpenCreditLinesAndLoans <- log(train$NumberOfOpenCreditLinesAndLoans)

test$RevolvingUtilizationOfUnsecuredLines <- log(test$RevolvingUtilizationOfUnsecuredLines)

test$DebtRatio <- log(test$DebtRatio)

test$NumberOfOpenCreditLinesAndLoans <- log(test$NumberOfOpenCreditLinesAndLoans)

# Assess model accuracy without resolving imbalance issue

But before applying the above mentioned methods, let us apply few alogorithms blindly without resolving the imbalancing issue and see how the algorithm performs, what the performance metrics are:

## Logitistic Regression Model

Below is a code that the algorithm has predicted most of the class 1 as class 0 because the class-0 distribution is more in the data set. Let’s try this in R.

#------------------------Read data set-------------------------------

train <- read.csv('fra\_train.csv', header=TRUE)

test <- read.csv('fra\_test.csv', header=TRUE)

train$SeriousDlqin2yrs <- factor(train$SeriousDlqin2yrs)

test$SeriousDlqin2yrs <- factor(test$SeriousDlqin2yrs)

#------------------------Data pre-processing-------------------------------

train$NumberOfDependents <- factor(train$NumberOfDependents)

test$NumberOfDependents <- factor(test$NumberOfDependents)

#------------------------Mode imputation for the missing values-------------------------------

# Mode Imputation

getmode <- function(v) {

uniqv <- unique(v)

uniqv[which.max(tabulate(match(v, uniqv)))]

}

train$NumberOfDependents[is.na(train$NumberOfDependents)] <- getmode(train$NumberOfDependents)

test$NumberOfDependents[is.na(test$NumberOfDependents)] <- getmode(test$NumberOfDependents)

#------------------------Split data set-------------------------------

# Remove Loan\_ID attribute

df <- train

df <- df[,-1]

test <- test[,-1]

# Split 80/20

trainIndex <- createDataPartition(df$SeriousDlqin2yrs, p=0.8,list=FALSE,times=1)

df\_train <- df[trainIndex,];df\_valid <- df[-trainIndex,]

# Training scheme - 10-fold CV

ctrl <- trainControl(method='repeatedcv', number=10,repeats=3)

#------------------------Build Logistic regression Model-------------------------------

logit\_fit <- glm(SeriousDlqin2yrs~.,

data = df\_train,

family = 'binomial')

summary(logit\_fit) # AIC: 1839

#------------------------ Logistic regression Model Results -------------------------------

> summary(logit\_fit)

Call:

glm(formula = SeriousDlqin2yrs ~ ., family = "binomial", data = df\_train)

Deviance Residuals:

Min 1Q Median 3Q Max

-0.5163 -0.3670 -0.3527 -0.3350 2.6464

Coefficients:

Estimate Std. Error z value Pr(>|z|)

(Intercept) -2.640e+00 1.331e-01 -19.839 <2e-16 \*\*\*

RevolvingUtilizationOfUnsecuredLines 1.044e-04 4.398e-04 0.237 0.8124

DebtRatio -9.483e-05 8.011e-05 -1.184 0.2365

NumberOfOpenCreditLinesAndLoans -1.765e-02 1.369e-02 -1.290 0.1971

NumberOfDependents 1.028e-01 5.709e-02 1.801 0.0717 .

---

Signif. codes: 0 ‘\*\*\*’ 0.001 ‘\*\*’ 0.01 ‘\*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

Null deviance: 1837.7 on 3999 degrees of freedom

Residual deviance: 1830.9 on 3995 degrees of freedom

AIC: 1840.9

Number of Fisher Scoring iterations: 6

#------------------------in-sample (training) accuracy-------------------------------

> y\_pred <- predict(logit\_fit, df\_train, type='response')

> y\_pred <- floor(y\_pred+0.5)

> df\_train$ypred <- y\_pred

> sum(with(df\_train, table(ypred, SeriousDlqin2yrs)))

[1] 4000

> with(df\_train, table(ypred, SeriousDlqin2yrs))

SeriousDlqin2yrs

ypred 0 1

0 3756 244

> auc(df\_train$SeriousDlqin2yrs, df\_train$ypred)

Area under the curve: 0.5

#------------------------validation sample accuracy-------------------------------

> y\_pred <- predict(logit\_fit, df\_valid, type='response')

> y\_pred <- floor(y\_pred+0.5)

> df\_valid$ypred <- y\_pred

> sum(with(df\_valid, table(ypred, SeriousDlqin2yrs)))

[1] 1000

> with(df\_valid, table(ypred, SeriousDlqin2yrs))

SeriousDlqin2yrs

ypred 0 1

0 939 61

> auc(df\_valid$SeriousDlqin2yrs, df\_valid$ypred)

Area under the curve: 0.5

#------------------------Testing sample accuracy-------------------------------

> y\_pred <- predict(logit\_fit, test, type='response')

> y\_pred <- floor(y\_pred+0.5)

> test$ypred <- y\_pred

> sum(with(test, table(ypred, SeriousDlqin2yrs)))

[1] 1000

> with(test, table(ypred, SeriousDlqin2yrs))

SeriousDlqin2yrs

ypred 0 1

0 937 63

> auc(test$SeriousDlqin2yrs, test$ypred)

Area under the curve: 0.5

Let’s look at the confusion matrix generated for each sample. It is found that 244 records which are of class 1 are predicted as class-0 in the training data set. Similary 61 records which are are of class 1 but predicted as class-0, 63 records which are of class-1 are predicted as class-0 in the testing data set.

It is evident that the algorithm is categorizing the minority class (class-1) as class-0 because class-0 distribution is significant in the data set and this imbalance need to be resolved prior to applying the algorithms.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | Training | |  | Validation | |  | Testing | |
|  |  | Actual | |  | Actual | |  | Actual | |
|  |  | 0 | 1 |  | 0 | 1 |  | 0 | 1 |
| Predicted | 0 | 3756 | **244** | 0 | 939 | **61** | 0 | 937 | **63** |
| 1 |  |  | 1 |  |  | 1 | 0 | 0 |
| Accuracy |  | 0.939 | |  | 0.939 | |  | 0.937 | |
| AUC |  | 0.5 | |  | 0.5 | |  | 0.5 | |

**Note**: Accuracy measure is not good measure in this case though it looks good.

## eXtreme Gradient Boosting (xGBTree) Model

#\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

# Clean up

rm(list=ls())

gc()

#\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

# Load packages

require(caret);require(plyr);require(Hmisc);require(pROC);require(DMwR)

require(missForest);require(mi);require(mice);require(car);require(xgboost)

require(Amelia)

#\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

# Read files

setwd('c:/r/data/csv/')

train <- read.csv('fra\_train.csv', header=TRUE)

test <- read.csv('fra\_test.csv', header=TRUE)

test\_dat <- read.csv('fra\_test.csv', header=TRUE)

#\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

# Check the data set dimensions

dim(train)

dim(test)

#\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

# Check missing values

missmap(train, col=c('red', 'white'), legend=FALSE)

missmap(test, col=c('red', 'white'), legend=FALSE)

#\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

# Data Pre-Processing

train$SeriousDlqin2yrs <- factor(train$SeriousDlqin2yrs)

test$SeriousDlqin2yrs <- factor(test$SeriousDlqin2yrs)

train$NumberOfDependents <- factor(train$NumberOfDependents)

test$NumberOfDependents <- factor(test$NumberOfDependents)

#\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

# Mode Imputation

getmode <- function(v) {

uniqv <- unique(v)

uniqv[which.max(tabulate(match(v, uniqv)))]

}

train$NumberOfDependents[is.na(train$NumberOfDependents)] <- getmode(train$NumberOfDependents)

test$NumberOfDependents[is.na(test$NumberOfDependents)] <- getmode(test$NumberOfDependents)

#\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

# Now check missing values

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

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

#\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

# Split Data

# Remove Loan\_ID attribute

df <- train

df <- df[,-1]

test <- test[,-1]

# Split 80/20

trainIndex <- createDataPartition(df$SeriousDlqin2yrs, p=0.8,list=FALSE,times=1)

df\_train <- df[trainIndex,];df\_valid <- df[-trainIndex,]

# Training scheme - 10-fold CV

ctrl <- trainControl(method='repeatedcv', number=10,repeats=3)

#\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

# Preparation for xGB Algorithm

y <- df\_train$SeriousDlqin2yrs

y <- as.numeric(levels(y))[y]

df\_train$SeriousDlqin2yrs <- as.numeric(levels(df\_train$SeriousDlqin2yrs))[df\_train$SeriousDlqin2yrs]

df\_valid$SeriousDlqin2yrs <- as.numeric(levels(df\_valid$SeriousDlqin2yrs))[df\_valid$SeriousDlqin2yrs]

#\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

# Declare outcome and features

outcome\_name <- 'SeriousDlqin2yrs'

feature\_names <- setdiff(names(df\_train), outcome\_name)

#\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

# Train xGBTree classification model

xgb <- xgboost(data = data.matrix(df\_train[,feature\_names]),

label=df\_train$SeriousDlqin2yrs,

eta = 0.2, #0.1 #0.15 #0.2

max\_depth = 30, #20 #20

nround=25,

objective = "binary:logistic",

nthread = 3,

maximize = TRUE,

eval\_metric = 'auc')

#\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

# in-sample accuracy on training sample

y\_pred <- predict(xgb, data.matrix(df\_train[,feature\_names]))

y\_pred <- floor(y\_pred+0.5)

df\_train$ypred <- y\_pred

sum(with(df\_train, table(ypred, SeriousDlqin2yrs)))

with(df\_train, table(ypred, SeriousDlqin2yrs))

SeriousDlqin2yrs

ypred 0 1

0 3755 91

1 1 153

auc(df\_train$SeriousDlqin2yrs, df\_train$ypred)

Area under the curve: 0.8134

#\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

# validation sample accuracy

y\_pred <- predict(xgb, data.matrix(df\_valid[,feature\_names]))

y\_pred <- floor(y\_pred+0.5)

df\_valid$ypred <- y\_pred

sum(with(df\_valid, table(ypred, SeriousDlqin2yrs)))

with(df\_valid, table(ypred, SeriousDlqin2yrs))

SeriousDlqin2yrs

ypred 0 1

0 918 56

1 21 5

auc(df\_valid$SeriousDlqin2yrs, df\_valid$ypred)

Area under the curve: 0.5298

#\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

# predict on test sample

y\_pred <- predict(xgb, data.matrix(test[,feature\_names]))

y\_pred <- floor(y\_pred+0.5)

test$ypred <- y\_pred

sum(with(test, table(ypred, SeriousDlqin2yrs)))

with(test, table(ypred, SeriousDlqin2yrs))

SeriousDlqin2yrs

ypred 0 1

0 918 59

1 19 4

auc(test$SeriousDlqin2yrs, test$ypred)

Area under the curve: 0.5216

#\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

# unload packages

detach('package:caret')

detach('package:plyr')

detach('package:Hmisc')

detach('package:pROC')

detach('package:DMwR')

detach('package:missForest')

detach('package:mi')

detach('package:mice')

detach('package:car')

detach('package:xgboost')

detach('package:Amelia')

detach('package:ggplot2')

#\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

# Clean up

rm(list=ls())

gc()

It is evident from the confusion matrix that the algorithm has predicted significant portion of class 1 records as class 0 due to the imbalance issue.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | Training | |  | Validation | |  | Testing | |
|  |  | Actual | |  | Actual | |  | Actual | |
|  |  | 0 | 1 |  | 0 | 1 |  | 0 | 1 |
| Predicted | 0 | 3755 | **91** | 0 | 918 | **56** | 0 | 918 | **59** |
| 1 | 1 | 153 | 1 | 21 | 5 | 1 | 19 | 4 |
| Accuracy |  | 0.977 | |  | 0.923 | |  | 0.922 | |
| AUC |  | 0.8134 | |  | 0.5298 | |  | 0.5216 | |

**Note**: Accuracy measure is not good measure in this case though it looks good.

## Random Forest Model

Below is a code that the random Forest algorithm has predicted most of the class 1 as class 0 because the class-0 distribution is more in the data set. Let’s try this in R.

#------------------------Read data set-------------------------------

train <- read.csv('fra\_train.csv', header=TRUE)

test <- read.csv('fra\_test.csv', header=TRUE)

#------------------------Data pre-processing-------------------------------

train$SeriousDlqin2yrs <- factor(train$SeriousDlqin2yrs)

test$SeriousDlqin2yrs <- factor(test$SeriousDlqin2yrs)

#------------------------Mode imputation for the missing values-------------------------------

# Mode Imputation

getmode <- function(v) {

uniqv <- unique(v)

uniqv[which.max(tabulate(match(v, uniqv)))]

}

train$NumberOfDependents[is.na(train$NumberOfDependents)] <- getmode(train$NumberOfDependents)

test$NumberOfDependents[is.na(test$NumberOfDependents)] <- getmode(test$NumberOfDependents)

#------------------------Split data set-------------------------------

# Remove Loan\_ID attribute

df <- train

df <- df[,-1]

test <- test[,-1]

# Split 80/20

trainIndex <- createDataPartition(df$SeriousDlqin2yrs, p=0.8,list=FALSE,times=1)

df\_train <- df[trainIndex,];df\_valid <- df[-trainIndex,]

# Training scheme - 10-fold CV

ctrl <- trainControl(method='repeatedcv', number=10,repeats=3)

#------------------------Build random Forest Model-------------------------------

tRF <- tuneRF(x = df\_train[,feature\_names],

y=df\_train$SeriousDlqin2yrs,

mtryStart = 3,

ntreeTry=100,

stepFactor = 1.5,

improve = 0.0001,

trace=TRUE,

plot = TRUE,

doBest = TRUE,

nodesize = 10,

importance=TRUE )

#------------------------in-sample (training) accuracy-------------------------------

df\_train$predict.class <- predict(tRF, df\_train, type="class")

df\_train$predict.score <- predict(tRF, df\_train, type='prob')

with(df\_train, table(predict.class,SeriousDlqin2yrs))

SeriousDlqin2yrs

predict.class 0 1

0 3751 **171**

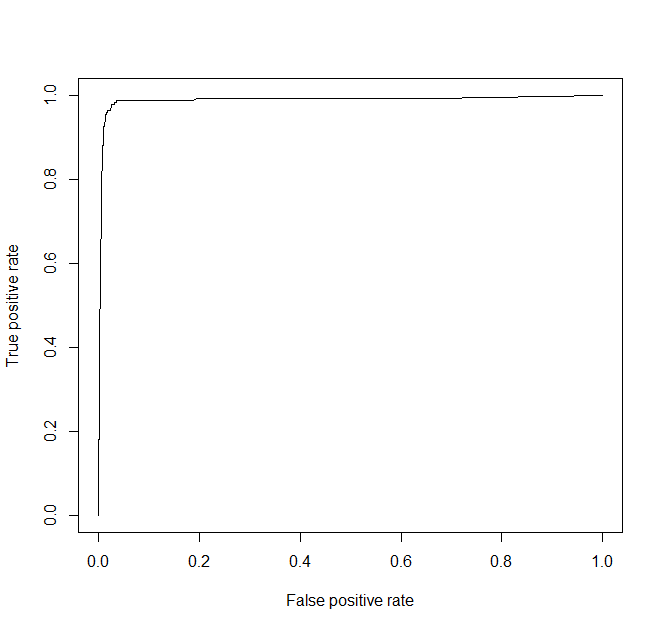
1 5 73

#AUC compute

pred <- prediction(df\_train$predict.score[,2], df\_train$SeriousDlqin2yrs)

perf <- performance(pred, "tpr", "fpr")

plot(perf)



KS <- max(attr(perf, 'y.values')[[1]]-attr(perf, 'x.values')[[1]])

auc <- performance(pred,"auc");

auc <- as.numeric(auc@y.values)

> auc

[1] 0.9886455

## Gini Compuation

library(ineq)

gini = ineq(df\_train$predict.score[,2], type="Gini")

## Printing the model performance statistics

> auc

[1] 0.9886455

> KS

[1] 0.9550141

> gini

[1] 0.816566

#------------------------validation sample accuracy-------------------------------

> df\_valid$predict.class <- predict(tRF, df\_valid, type="class")

> df\_valid$predict.score <- predict(tRF, df\_valid, type="prob")

> with(df\_valid, table(predict.class,SeriousDlqin2yrs))

SeriousDlqin2yrs

predict.class 0 1

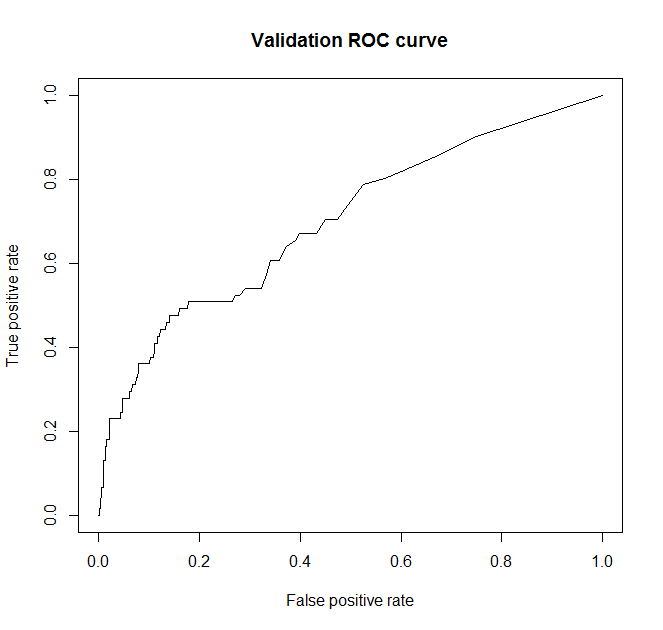
0 930 **56**

1 9 5

pred <- prediction(df\_valid$predict.score[,2], df\_valid$SeriousDlqin2yrs)

perf <- performance(pred, "tpr", "fpr")

plot(perf,main='Validatiaon ROC curve')



|  |
| --- |
| > auc <- performance(pred,"auc");  > auc <- as.numeric(auc@y.values)  > auc  [1] 0.6991917 |

#------------------------Testing sample accuracy-------------------------------

> test$predict.class <- predict(tRF, test, type="class")

> test$predict.score <- predict(tRF, test, type="prob")

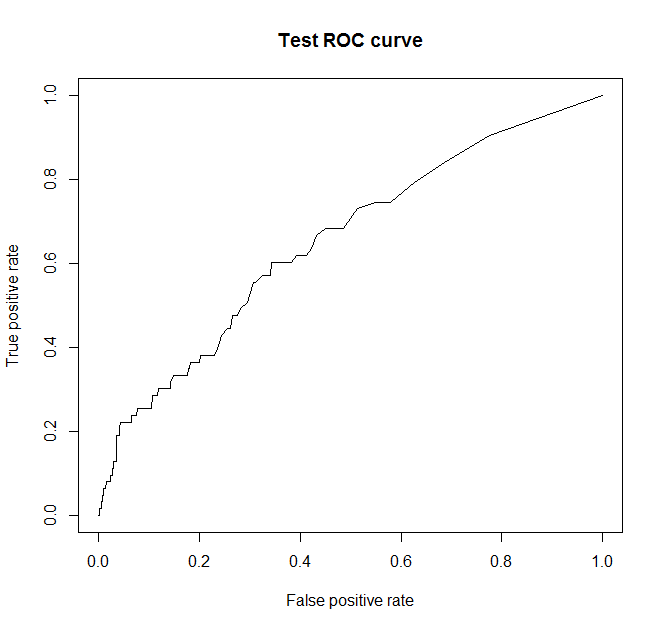
> with(test, table(predict.class,SeriousDlqin2yrs))

SeriousDlqin2yrs

predict.class 0 1

0 930 **60**

1 7 3



> auc <- performance(pred,"auc");

> auc <- as.numeric(auc@y.values)

> auc

[1] 0.6543935

Let’s look at the confusion matrix generated for each sample. It is found that 171 records which are of class 1 are predicted as class-0 in the training data set. Similary 56 records which are are of class 1 but predicted as class-0 in the validataion data set, 60 records which are of class-1 are predicted as class-0 in the testing data set.

It is evident that the algorithm is categorizing the minority class (class-1) as class-0 because class-0 distribution is significant in the data set and this imbalance need to be resolved prior to applying the algorithms.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | Training | |  | Validation | |  | Testing | |
|  |  | Actual | |  | Actual | |  | Actual | |
|  |  | 0 | 1 |  | 0 | 1 |  | 0 | 1 |
| Predicted | 0 | 3751 | **171** | 0 | 930 | **56** | 0 | 930 | **60** |
| 1 | 5 | 73 | 1 | 9 | 5 | 1 | 7 | 3 |
| Accuracy |  | 0.956 | |  | 0.935 | |  | 0.933 | |
| AUC |  | 0.9886 | |  | 0.6991 | |  | 0.6543 | |

**Note**: Accuracy measure is not good measure in this case though it looks good.

## Tree bag Model

Below is a code that the Tree bag algorithm has predicted most of the class 1 as class 0 because the class-0 distribution is more in the data set. Let’s try this in R.

#------------------------Read data set-------------------------------

train <- read.csv('fra\_train.csv', header=TRUE)

test <- read.csv('fra\_test.csv', header=TRUE)

#------------------------Data pre-processing-------------------------------

train$SeriousDlqin2yrs <- factor(train$SeriousDlqin2yrs)

test$SeriousDlqin2yrs <- factor(test$SeriousDlqin2yrs)

#------------------------Mode imputation for the missing values-------------------------------

# Mode Imputation

getmode <- function(v) {

uniqv <- unique(v)

uniqv[which.max(tabulate(match(v, uniqv)))]

}

train$NumberOfDependents[is.na(train$NumberOfDependents)] <- getmode(train$NumberOfDependents)

test$NumberOfDependents[is.na(test$NumberOfDependents)] <- getmode(test$NumberOfDependents)

#------------------------Split data set-------------------------------

# Remove Loan\_ID attribute

df <- train

df <- df[,-1]

test <- test[,-1]

# Split 80/20

trainIndex <- createDataPartition(df$SeriousDlqin2yrs, p=0.8,list=FALSE,times=1)

df\_train <- df[trainIndex,];df\_valid <- df[-trainIndex,]

# Training scheme - 10-fold CV

ctrl <- trainControl(method='repeatedcv', number=10,repeats=3)

#------------------------Build Tree bag Model-------------------------------

tbmodel <- train(SeriousDlqin2yrs~.,

data=df\_train,

method='treebag')

#------------------------in-sample (training) accuracy-------------------------------

pred <- as.numeric(predict(tbmodel$finalModel, df\_train[,-1]))

df\_train$SeriousDlqin2yrs <- as.numeric(df\_train$SeriousDlqin2yrs)

table(pred,df\_train$SeriousDlqin2yrs)

pred 1 2

1 3755 11

2 1 233

(auc <- roc(df\_train$SeriousDlqin2yrs, pred))

Area under the curve: 0.9773

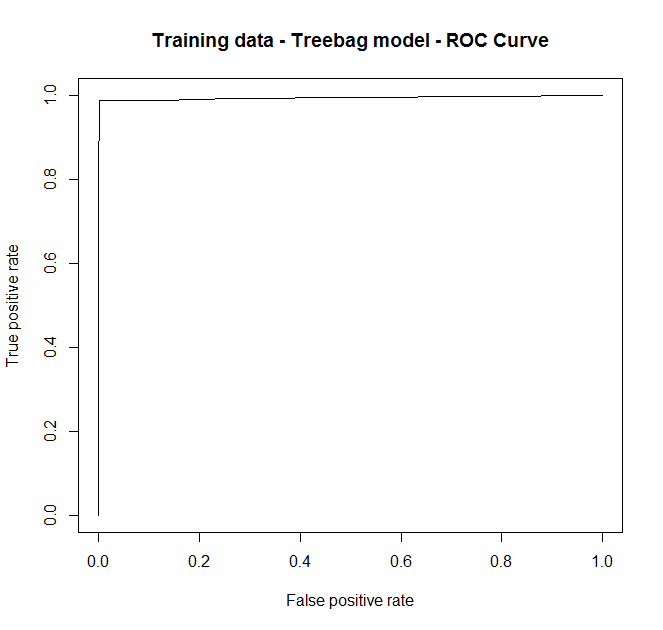
#ROC Curve

df\_train$predict.score <- predict(tRF, df\_train, type='prob')

pred <- prediction(df\_train$predict.score[,2], df\_train$SeriousDlqin2yrs)

perf <- performance(pred, "tpr", "fpr")

plot(perf)



#------------------------validation sample accuracy-------------------------------

pred <- as.numeric(predict(tbmodel$finalModel, df\_valid[,-1]))

table(pred,df\_valid$SeriousDlqin2yrs)

pred 0 1

1 925 55

2 14 6

(auc <- roc(df\_valid$SeriousDlqin2yrs, pred)) #0.5412

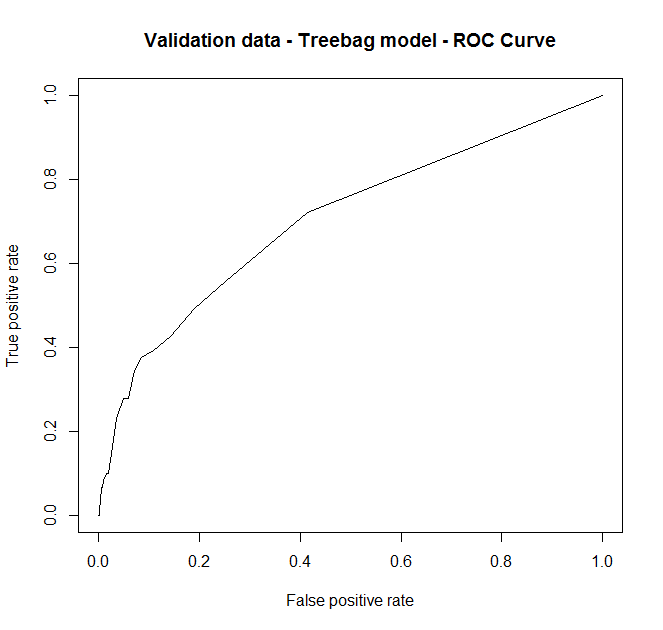
Area under the curve: 0.5417

df\_valid$predict.score <- predict(tbmodel, df\_valid, type='prob')

pred <- prediction(df\_valid$predict.score[,2], df\_valid$SeriousDlqin2yrs)

perf <- performance(pred, "tpr", "fpr")

plot(perf, main='Validation data - Treebag model - ROC Curve')



#------------------------Testing sample accuracy-------------------------------

pred <- as.numeric(predict(tbmodel$finalModel, test[,-1]))

table(pred,test$SeriousDlqin2yrs)

pred 0 1

1 923 59

2 14 4

(auc <- roc(test$SeriousDlqin2yrs, pred))

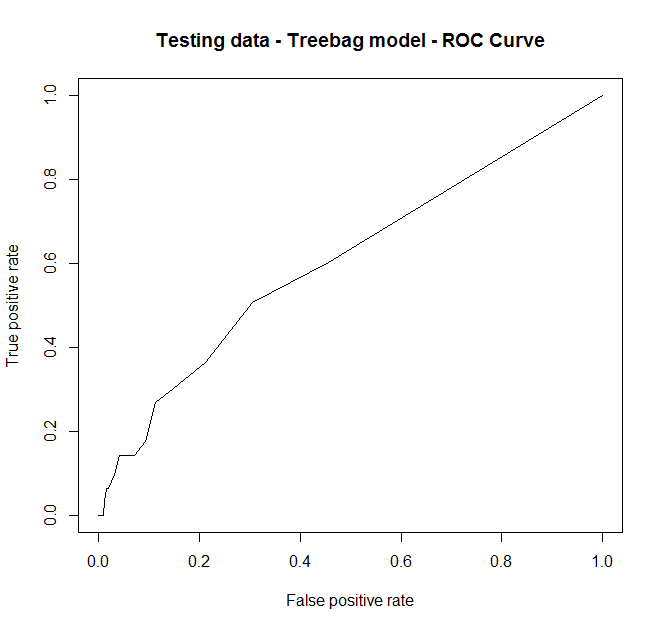
Area under the curve: 0.5243

test$predict.score <- predict(tbmodel, test, type='prob')

pred <- prediction(test$predict.score[,2], test$SeriousDlqin2yrs)

perf <- performance(pred, "tpr", "fpr")

plot(perf, main='Testing data - Treebag model - ROC Curve')



Let’s look at the confusion matrix generated for each data set. It is found that accuracy is good on the training data set. But where as 55records which are are of class 1 but predicted as class-0 in the validataion data set, 59 records which are of class-1 are predicted as class-0 in the testing data set.

Though this treebag model predicted pretty accurate on the training data set but did not do well on the validation and testing data set. It is evident that the algorithm is categorizing the minority class (class-1) as class-0 because class-0 distribution is significant in the data set and this imbalance need to be resolved prior to applying the algorithms.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | Training | |  | Validation | |  | Testing | |
|  |  | Actual | |  | Actual | |  | Actual | |
|  |  | 0 | 1 |  | 0 | 1 |  | 0 | 1 |
| Predicted | 0 | 3755 | **11** | 0 | 925 | **55** | 0 | 923 | **59** |
| 1 | 1 | 233 | 1 | 14 | 6 | 1 | 14 | 4 |
| Accuracy |  | 0.997 | |  | 0.931 | |  | 0.927 | |
| AUC |  | 0.9773 | |  | 0.5417 | |  | 0.5243 | |

**Note**: Accuracy measure is not good measure in this case though it looks good.

## H2o deep learning Model

Below is a code that the h2o deep learning algorithm has predicted most of the class 1 as class 0 because the class-0 distribution is more in the data set. Let’s try this in R.

#------------------------Load packages -------------------------------

require(caret)

require(pROC)

require(DMwR)

require(Amelia)

require(ROCR)

require(h2o)

#------------------------Read data set-------------------------------

train <- read.csv('fra\_train.csv', header=TRUE)

test <- read.csv('fra\_test.csv', header=TRUE)

#------------------------Data pre-processing-------------------------------

train$SeriousDlqin2yrs <- factor(train$SeriousDlqin2yrs)

test$SeriousDlqin2yrs <- factor(test$SeriousDlqin2yrs)

#------------------------Mode imputation for the missing values-------------------------------

# Mode Imputation

getmode <- function(v) {

uniqv <- unique(v)

uniqv[which.max(tabulate(match(v, uniqv)))]

}

train$NumberOfDependents[is.na(train$NumberOfDependents)] <- getmode(train$NumberOfDependents)

test$NumberOfDependents[is.na(test$NumberOfDependents)] <- getmode(test$NumberOfDependents)

#------------------------Split data set-------------------------------

# Remove Loan\_ID attribute

df <- train

df <- df[,-1]

test <- test[,-1]

# Split 80/20

trainIndex <- createDataPartition(df$SeriousDlqin2yrs, p=0.8,list=FALSE,times=1)

df\_train <- df[trainIndex,];df\_valid <- df[-trainIndex,]

# Training scheme - 10-fold CV

ctrl <- trainControl(method='repeatedcv', number=10,repeats=3)

#------------------------outcome, features-------------------------------

outcome <- 'SeriousDlqin2yrs'

features <- setdiff(names(df\_train), outcome)

#------------------------setup h2o-------------------------------

h2o.server <- h2o.init( nthreads= -1)

df\_train.hex <- as.h2o(df\_train)

df\_valid.hex <- as.h2o(df\_valid)

test.hex <- as.h2o(test)

#------------------------Build h2o deep learning Model-------------------------------

dl\_model\_1 = h2o.deeplearning( x=features,

y = outcome,

training\_frame =df\_train.hex ,

activation="Rectifier",

hidden=6,

epochs=60,

adaptive\_rate =F)

#\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_training sample accuracy

pred = as.data.frame(h2o.predict(dl\_model\_1, newdata = df\_train.hex) )

a <- pred$predict

df\_train$pp <- a

sum(with(df\_train, table(SeriousDlqin2yrs, pp)))

with(df\_train, table(SeriousDlqin2yrs, pp))

SeriousDlqin2yrs

pp 0 1

0 3039 158

1 717 86

#\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_validation sample accuracy

pred = as.data.frame(h2o.predict(dl\_model\_1, newdata = df\_valid.hex) )

a <- pred$predict

df\_valid$pp <- a

sum(with(df\_valid, table(SeriousDlqin2yrs, pp)))

with(df\_valid, table(SeriousDlqin2yrs, pp))

SeriousDlqin2yrs

pp 0 1

0 763 49

1 176 12

#\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Test sample accuracy

pred = as.data.frame(h2o.predict(dl\_model\_1, newdata = test.hex) )

a <- pred$predict

test$pp <- a

sum(with(test, table(SeriousDlqin2yrs, pp)))

with(test, table(SeriousDlqin2yrs, pp))

SeriousDlqin2yrs

pp 0 1

0 766 46

1 171 17

Let’s look at the confusion matrix generated for each data set. The result is interesting with the h2o model because we are not seeing the imbalance class prediction.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | Training | |  | Validation | |  | Testing | |
|  |  | Actual | |  | Actual | |  | Actual | |
|  |  | 0 | 1 |  | 0 | 1 |  | 0 | 1 |
| Predicted | 0 | 3039 | 158 | 0 | 763 | 49 | 0 | 766 | 46 |
| 1 | 717 | 86 | 1 | 176 | 12 | 1 | 171 | 17 |
| Accuracy |  | 0.78125 | |  | 0.775 | |  | 0.783 | |
| AUC |  | **??** | |  | **??** | |  | **??** | |

> prop.table(table(df\_train$SeriousDlqin2yrs))

0 1

0.939 0.061

> prop.table(table(df\_valid$SeriousDlqin2yrs))

0 1

0.939 0.061

> prop.table(table(test$SeriousDlqin2yrs))

0 1

0.937 0.063

## Ensemble (gbm, rpart, treebag) Model

Below is a code that the algorithm has predicted most of the class 1 as class 0 because the class-0 distribution is more in the data set. Let’s try this in R.

#------------------------Read data set-------------------------------

train <- read.csv('fra\_train.csv', header=TRUE)

test <- read.csv('fra\_test.csv', header=TRUE)

train$SeriousDlqin2yrs <- factor(train$SeriousDlqin2yrs)

test$SeriousDlqin2yrs <- factor(test$SeriousDlqin2yrs)

#------------------------Data pre-processing-------------------------------

train$NumberOfDependents <- factor(train$NumberOfDependents)

test$NumberOfDependents <- factor(test$NumberOfDependents)

#------------------------Mode imputation for the missing values-------------------------------

# Mode Imputation

getmode <- function(v) {

uniqv <- unique(v)

uniqv[which.max(tabulate(match(v, uniqv)))]

}

train$NumberOfDependents[is.na(train$NumberOfDependents)] <- getmode(train$NumberOfDependents)

test$NumberOfDependents[is.na(test$NumberOfDependents)] <- getmode(test$NumberOfDependents)

#------------------------Split data set-------------------------------

# Remove Loan\_ID attribute

df <- train

df <- df[,-1]

test <- test[,-1]

# Split the training data set into 3 different data sets

df <- df[sample(nrow(df)),]

split <- floor(nrow(df)/3)

ensembleData <- df[0:split,]

blenderData <- df[(split+1):(split\*2),]

testingData <- df[(split\*2+1):nrow(df),]

dim(ensembleData) #1666 rows

dim(blenderData) #1666 rows

dim(testingData) #1668 rows

#------------------------Define CV Training scheme to be use din caret package-------------------------------

myControl <- trainControl(method='cv', number=3, returnResamp='none')

#------------------------Declare outcome and predictor variables-------------------------------

labelName <- 'SeriousDlqin2yrs'

predictors <- names(ensembleData)[names(ensembleData) != labelName]

#------------------------Build gBM Model – Bench Mark -------------------------------

# We run the data on a gbm model without any enembling to use as a comparative benchmark.

model <- train(blenderData[,predictors],

blenderData[,labelName],

method='gbm',

trControl=myControl)

preds <- as.numeric(predict(object=model, testingData[,predictors]))

library(pROC)

testingData[,labelName] <- as.numeric(testingData[,labelName])

auc <- roc(testingData[,labelName], preds)

print(auc$auc)

> 0.5

------------------------ Build Ensemble model with gbm, RPART, treebag-------------------------------

# build individual models as part of our ensemble model and train them with ensembleData.

model\_gbm <- train(ensembleData[,predictors], ensembleData[,labelName], method='gbm', trControl=myControl)

model\_rpart <- train(ensembleData[,predictors], ensembleData[,labelName], method='rpart', trControl=myControl)

model\_treebag <- train(ensembleData[,predictors], ensembleData[,labelName], method='treebag', trControl=myControl)

# Now, we predict the target variable on the other two data sets, i.e., blenderData and testingData.

blenderData$gbm\_PROB <- predict(object=model\_gbm, blenderData[,predictors])

blenderData$rf\_PROB <- predict(object=model\_rpart, blenderData[,predictors])

blenderData$treebag\_PROB <- predict(object=model\_treebag, blenderData[,predictors])

testingData$gbm\_PROB <- predict(object=model\_gbm, testingData[,predictors])

testingData$rf\_PROB <- predict(object=model\_rpart, testingData[,predictors])

testingData$treebag\_PROB <- predict(object=model\_treebag, testingData[,predictors])

# now let us train the final blending model

predictors <- names(blenderData)[names(blenderData) != labelName]

final\_blender\_model <- train(blenderData[,predictors], blenderData[,labelName], method='gbm', trControl=myControl)

# now we predict using final model on the testingData

preds <- as.numeric(predict(object=final\_blender\_model, testingData[,predictors]))

auc <- roc(testingData[,labelName], preds)

print(auc$auc)

> 0.5381

Though there is a bump of 0.0381 in AUC but the AUC has not increased drastically despite we used the ensembles techniques used.

|  |  |
| --- | --- |
| Here is the proportion of class 0 and 1 in the testing data  > prop.table(table(testingData$SeriousDlqin2yrs))  0 1  0.93465228 0.06534772 | Here is the confusion Matrix generated which shows that 100 records which are of class 1 are predicted as class 0, this due to the imbalance issue.  > table(preds, testingData[,labelName])    preds 0 1  0 1549 **100**  1 10 9 |

## Various other models (used weka)

Now let’s try this in Weka tool **on the training data set** (5000 rows) and check the confusion Matrix. We can see from the different algorithms that are tried out in weka, it is found that it is exhibiting the same behavior as what we got in R tool. So it is evident now that the statistical tools exhibit the same behavor and it is necessary now that we resolve this imbalance.

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  | zeroR | |  | oneR | |  | PART | |  | randomForest | |
|  |  | Actual | |  | Actual | |  | Actual | |  | Actual | |
|  |  | 0 | 1 |  | 0 | 1 |  | 0 | 1 |  | 0 | 1 |
| Predicted | 0 | 4695 | 305 | 0 | 4669 | **298** | 0 | 4679 | **302** | 0 | 4643 | **289** |
| 1 |  |  | 1 | 26 | 7 | 1 | 16 | 3 | 1 | 52 | 16 |
|  |  | jRIP | |  | naiveBayes | |  | PART | |  | REPTree | |
|  |  | Actual | |  | Actual | |  | Actual | |  | Actual | |
|  |  | 0 | 1 |  | 0 | 1 |  | 0 | 1 |  | 0 | 1 |
| Predicted | 0 | 4685 | **294** | 0 | 4694 | **305** | 0 | 4694 | **305** | 0 | 4691 | **301** |
| 1 | 10 | 11 | 1 | 1 | 0 | 1 | 1 | 0 | 1 | 4 | 4 |
|  |  | Nnet | |  | IBk(kNN) | |  | J48 (DT) | |  | logitBoost | |
|  |  | Actual | |  | Actual | |  | Actual | |  | Actual | |
|  |  | 0 | 1 |  | 0 | 1 |  | 0 | 1 |  | 0 | 1 |
| Predicted | 0 | 4695 | **305** | 0 | 4475 | **266** | 0 | 4695 | **305** | 0 | 4681 | **296** |
| 1 |  |  | 1 | 220 | 39 | 1 | 0 | 0 | 1 | 14 | 9 |

**Conclusion**:

Having tried all the models from basic to ensembles, it is clear that the imbalance class issue need to be resolved in order to get good accuracy on the validation and the test data set. We will adddress this issue in the coming sections

# Address imbalanced issue

## Using ROSE (Random Over-Sampling Examples) technique:

#------------------------Load packages -------------------------------

require(caret)

require(Amelia)

require(ROCR)

require(ROSE)

require(rpart)

#------------------------Read data set-------------------------------

train <- read.csv('fra\_train.csv', header=TRUE)

test <- read.csv('fra\_test.csv', header=TRUE)

#------------------------Data pre-processing-------------------------------

train$SeriousDlqin2yrs <- factor(train$SeriousDlqin2yrs)

test$SeriousDlqin2yrs <- factor(test$SeriousDlqin2yrs)

#------------------------Mode imputation for the missing values-------------------------------

# Mode Imputation

getmode <- function(v) {

uniqv <- unique(v)

uniqv[which.max(tabulate(match(v, uniqv)))]

}

train$NumberOfDependents[is.na(train$NumberOfDependents)] <- getmode(train$NumberOfDependents)

test$NumberOfDependents[is.na(test$NumberOfDependents)] <- getmode(test$NumberOfDependents)

#------------------------Split data set-------------------------------

# Remove Loan\_ID attribute

df <- train

df <- df[,-1]

test <- test[,-1]

# Split 80/20

trainIndex <- createDataPartition(df$SeriousDlqin2yrs, p=0.8,list=FALSE,times=1)

df\_train <- df[trainIndex,];df\_valid <- df[-trainIndex,]

# Training scheme - 10-fold CV

ctrl <- trainControl(method='repeatedcv', number=10,repeats=3)

#------------------------Let’s do Over sampling and balance the data-------------------------------

data\_balanced\_over <- ovun.sample(SeriousDlqin2yrs ~ .,

data = df\_train,

method = "over",

N = 7512)$data

table(data\_balanced\_over$SeriousDlqin2yrs)

0 1

3756 3756

#------------------------Let’s do Under sampling and balance the data-------------------------------

data\_balanced\_under <- ovun.sample(SeriousDlqin2yrs ~ .,

data = df\_train,

method = "under",

N = 488)$data

table(data\_balanced\_under$SeriousDlqin2yrs)

0 1

244 244

#------------------------Let’s do both now-------------------------------

#Now the data set is balanced. But, you see that we've lost significant information from the sample. Let's do both undersampling and oversampling on this imbalanced data. This can be achieved using method = "both". In this case, the minority class is oversampled with replacement and majority class is undersampled without replacement.

data\_balanced\_both <- ovun.sample(SeriousDlqin2yrs ~ .,

data = df\_train,

method = 'both',

p = 0.5, N=4000)$data

table(data\_balanced\_both$SeriousDlqin2yrs)

0 1

2015 1985

#------------------------Let’s do ROSE-------------------------------

data.rose <- ROSE(SeriousDlqin2yrs~.,

data = df\_train)$data

table(data.rose$SeriousDlqin2yrs)

0 1

2013 1987

#------------------------Now build models-------------------------------

tree.rose <- rpart(SeriousDlqin2yrs ~ ., data = data.rose)

tree.over <- rpart(SeriousDlqin2yrs ~ ., data = data\_balanced\_over)

tree.under <- rpart(SeriousDlqin2yrs ~ ., data = data\_balanced\_under)

tree.both <- rpart(SeriousDlqin2yrs ~ ., data = data\_balanced\_both)

#------------------------training data set accuracy-------------------------------

#make predictions on unseen data

pred.tree.rose <- predict(tree.rose, newdata = df\_train)

pred.tree.over <- predict(tree.over, newdata = df\_train)

pred.tree.under <- predict(tree.under, newdata = df\_train)

pred.tree.both <- predict(tree.both, newdata = df\_train)

#AUC ROSE, Oversampling, Undersampling, and both

roc.curve(df\_train$SeriousDlqin2yrs, pred.tree.rose[,2])

0.515

roc.curve(df\_train$SeriousDlqin2yrs, pred.tree.over[,2])

0.735

roc.curve(df\_train$SeriousDlqin2yrs, pred.tree.under[,2])

0.751

roc.curve(df\_train$SeriousDlqin2yrs, pred.tree.both[,2])

0.753

#------------------------Validation data set accuracy-------------------------------

pred.tree.rose <- predict(tree.rose, newdata = df\_valid)

pred.tree.over <- predict(tree.over, newdata = df\_valid)

pred.tree.under <- predict(tree.under, newdata = df\_valid)

pred.tree.both <- predict(tree.both, newdata = df\_valid)

#AUC ROSE, Oversampling, Undersampling, and both

roc.curve(df\_valid$SeriousDlqin2yrs, pred.tree.rose[,2])

0.524

roc.curve(df\_valid$SeriousDlqin2yrs, pred.tree.over[,2])

0.699

roc.curve(df\_valid$SeriousDlqin2yrs, pred.tree.under[,2])

0.746

roc.curve(df\_valid$SeriousDlqin2yrs, pred.tree.both[,2])

0.701

#------------------------Test data set accuracy-------------------------------

pred.tree.rose <- predict(tree.rose, newdata = test)

pred.tree.over <- predict(tree.over, newdata = test)

pred.tree.under <- predict(tree.under, newdata = test)

pred.tree.both <- predict(tree.both, newdata = test)

#AUC ROSE, Oversampling, Undersampling, and both

roc.curve(test$SeriousDlqin2yrs, pred.tree.rose[,2])

0.508

roc.curve(test$SeriousDlqin2yrs, pred.tree.over[,2])

0.684

roc.curve(test$SeriousDlqin2yrs, pred.tree.under[,2])

0.731

roc.curve(test$SeriousDlqin2yrs, pred.tree.both[,2])

0.701

#------------------------Unload the packages-------------------------------

detach('package:caret')

detach('package:Amelia')

detach('package:ggplot2')

detach('package:ROCR')

detach('package:ROSE')

detach('package:rpart')

#------------------------Clean up R environment-------------------------------

rm(list=ls())

gc()

Let’s try tree bag model on the data set **with the imbalance present** and understand the performance metrics.

#\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Clean up R environment

rm(list=ls())

gc()

#\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Load R Packages

require(caret)

require(pROC)

require(DMwR)

require(Amelia)

require(ROCR)

#\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Read data set

setwd('c:/r/data/csv/')

train <- read.csv('fra\_train.csv', header=TRUE)

test <- read.csv('fra\_test.csv', header=TRUE)

#\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Data set dimensions

> dim(train)

[1] 5000 6

> dim(test)

[1] 1000 6

#\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Data pre-processing

train$SeriousDlqin2yrs <- factor(train$SeriousDlqin2yrs)

test$SeriousDlqin2yrs <- factor(test$SeriousDlqin2yrs)

#\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Mode Imputation

getmode <- function(v) {

uniqv <- unique(v)

uniqv[which.max(tabulate(match(v, uniqv)))]

}

train$NumberOfDependents[is.na(train$NumberOfDependents)] <- getmode(train$NumberOfDependents)

test$NumberOfDependents[is.na(test$NumberOfDependents)] <- getmode(test$NumberOfDependents)

train$NumberOfDependents <- as.numeric(train$NumberOfDependents)

test$NumberOfDependents <- as.numeric(test$NumberOfDependents)

#\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Now check for missing values after imputation

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

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

|  |  |
| --- | --- |
| > sapply(train, function(x) sum(is.na(x)))  Casenum  0  SeriousDlqin2yrs  0  RevolvingUtilizationOfUnsecuredLines  0  DebtRatio  0  NumberOfOpenCreditLinesAndLoans  0  NumberOfDependents  0 | > sapply(test, function(x) sum(is.na(x)))  Casenum  0  SeriousDlqin2yrs  0  RevolvingUtilizationOfUnsecuredLines  0  DebtRatio  0  NumberOfOpenCreditLinesAndLoans  0  NumberOfDependents  0 |

#\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Split the data

# Remove Loan\_ID attribute

df <- train

df <- df[,-1]

test <- test[,-1]

# Split 80/20

trainIndex <- createDataPartition(df$SeriousDlqin2yrs, p=0.8,list=FALSE,times=1)

df\_train <- df[trainIndex,];df\_valid <- df[-trainIndex,]

# Training scheme - 10-fold CV

ctrl <- trainControl(method='repeatedcv', number=10,repeats=3)

#\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_SMOTE the data

|  |  |
| --- | --- |
| Before SMOTE | After SMOTE |
| Training data set |  |
| Validataion data set |  |
| Test data set |  |

# Build Models

# Results

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Case** | **Algorithm** | **Pre-Processing** | **Training** | **Validation** | **Test** |
| **1** |  |  |  |  |  |
| **2** |  |  |  |  |  |
| **3** |  |  |  |  |  |
| **4** |  |  |  |  |  |
| **5** |  |  |  |  |  |
| **6** |  |  |  |  |  |
| **7** |  |  |  |  |  |
| **8** |  |  |  |  |  |
| **9** |  |  |  |  |  |