McKinsey Sales Excellence Hackathon on 20 Jan 2018

Bala Kesavan

20 January 2018

# McKinsey Sales Excellence Hackathon

The business problem is to predict loan approval, given applicant details like income, debt and demographics.

# Exploratory Data Analysis

We start with EDA and list down some of the key observations made, while the basis for the same is displayed in the calls below.

## Key observations

Of ~70K records, only 1.5% of the two-class dependent variable, 'Approved', are in the minority class. Correcting this imbalance with SMOTE worked better than through ROSE and downSample.  
'DOB' and 'Lead\_Creation\_Date' were converted from date to elapsed time/age.  
The 'NA' in 'Loan\_Amount' and 'Loan\_Period' match. Inferred as no loan taken and set to zero. Similarly for 'EMI' and 'Interest\_Rate'.  
The raw 'Monthly Income' and 'Existing EMI' variables are not showing sufficient discerning power. Log transform seemed to improve this and was chosen over binning, bounding etc.  
Unlabelled levels found in 'Primary\_Bank\_Type', 'Employer\_Category1' and 'City\_Category'. Added label.  
Too many levels for 'Employer\_Code', 'City\_Code', 'Customer\_Existing\_Primary\_Bank\_Code', even after processing with drop levels. Exclude.

Chi Square tests of independence between each variable and loan approval were conducted, after data processing. Graphical tests were also performed to visualize the dependence. None of the variables analyzed could be excluded since the null hypothesis of independence have to be rejected due to the low p values. One sample bivariate analysis (Gender v/s Approved) and samples of a couple of other tests used are included below.

summary(McKTrain)

## ID Gender DOB   
## APPA10000905029: 1 Female:29764 11/01/82: 253   
## APPA10001412205: 1 Male :39949 04/03/71: 183   
## APPA10004965619: 1 03/03/71: 121   
## APPA10007179048: 1 03/03/91: 103   
## APPA10007532038: 1 03/03/81: 88   
## APPA10009948043: 1 03/03/87: 83   
## (Other) :69707 (Other) :68882   
## Lead\_Creation\_Date City\_Code City\_Category Employer\_Code   
## 02/09/16: 1838 C10001 :10007 : 814 : 4018   
## 22/09/16: 1629 C10002 : 8716 A:49885 COM0000002: 457   
## 29/09/16: 1038 C10003 : 8666 B: 7320 COM0000003: 324   
## 26/09/16: 1036 C10004 : 5843 C:11694 COM0000004: 262   
## 30/09/16: 1028 C10005 : 5564 COM0000005: 243   
## 28/09/16: 1001 C10006 : 4203 COM0000006: 195   
## (Other) :62143 (Other):26714 (Other) :64214   
## Employer\_Category1 Employer\_Category2 Monthly\_Income   
## : 4018 Min. :1.00 Min. : 0   
## A:33336 1st Qu.:4.00 1st Qu.: 1650   
## B:18056 Median :4.00 Median : 2500   
## C:14303 Mean :3.72 Mean : 5622   
## 3rd Qu.:4.00 3rd Qu.: 4000   
## Max. :4.00 Max. :38383838   
## NA's :4298   
## Customer\_Existing\_Primary\_Bank\_Code Primary\_Bank\_Type Contacted  
## B001 :14197 : 9391 N:24438   
## B002 :10880 G:20703 Y:45275   
## B003 : 9515 P:39619   
## : 9391   
## B004 : 7070   
## B005 : 1920   
## (Other):16740   
## Source Source\_Category Existing\_EMI Loan\_Amount   
## S122 :30941 A: 3 Min. : 0.0 Min. : 5000   
## S133 :23877 B:29812 1st Qu.: 0.0 1st Qu.: 20000   
## S159 : 4474 C:11374 Median : 0.0 Median : 30000   
## S143 : 3480 D: 497 Mean : 360.9 Mean : 39430   
## S127 : 1546 E: 1050 3rd Qu.: 350.0 3rd Qu.: 50000   
## S137 : 1400 F: 459 Max. :545436.5 Max. :300000   
## (Other): 3995 G:26518 NA's :51 NA's :27709   
## Loan\_Period Interest\_Rate EMI Var1   
## Min. :1.000 Min. :11.99 Min. : 118 Min. : 0.000   
## 1st Qu.:3.000 1st Qu.:15.25 1st Qu.: 649 1st Qu.: 0.000   
## Median :4.000 Median :18.00 Median : 941 Median : 2.000   
## Mean :3.891 Mean :19.21 Mean : 1101 Mean : 3.948   
## 3rd Qu.:5.000 3rd Qu.:20.00 3rd Qu.: 1295 3rd Qu.: 7.000   
## Max. :6.000 Max. :37.00 Max. :13556 Max. :10.000   
## NA's :27709 NA's :47437 NA's :47437   
## Approved   
## Min. :0.00000   
## 1st Qu.:0.00000   
## Median :0.00000   
## Mean :0.01463   
## 3rd Qu.:0.00000   
## Max. :1.00000   
##

str(McKTrain)

## 'data.frame': 69713 obs. of 22 variables:  
## $ ID : Factor w/ 69713 levels "APPA10000905029",..: 7809 9062 12525 15528 17798 24053 28923 29974 33036 36522 ...  
## $ Gender : Factor w/ 2 levels "Female","Male": 1 2 2 2 2 1 2 1 2 1 ...  
## $ DOB : Factor w/ 10760 levels "","01/01/53",..: 8114 2595 3643 10258 6633 8267 9654 4343 860 4722 ...  
## $ Lead\_Creation\_Date : Factor w/ 92 levels "01/07/16","01/08/16",..: 43 10 55 25 58 1 4 7 4 4 ...  
## $ City\_Code : Factor w/ 679 levels "","C10001","C10002",..: 2 4 126 478 3 403 23 4 4 15 ...  
## $ City\_Category : Factor w/ 4 levels "","A","B","C": 2 2 4 4 2 4 3 2 2 3 ...  
## $ Employer\_Code : Factor w/ 36618 levels "","COM0000002",..: 29286 2 5128 4081 1777 7346 21787 14047 9675 28240 ...  
## $ Employer\_Category1 : Factor w/ 4 levels "","A","B","C": 2 4 4 2 2 2 3 2 2 2 ...  
## $ Employer\_Category2 : int 4 1 4 4 4 4 4 4 4 4 ...  
## $ Monthly\_Income : num 2000 3500 2250 3500 10000 7000 7500 3000 2500 2500 ...  
## $ Customer\_Existing\_Primary\_Bank\_Code: Factor w/ 58 levels "","B001","B002",..: 2 3 4 4 2 15 4 7 18 1 ...  
## $ Primary\_Bank\_Type : Factor w/ 3 levels "","G","P": 3 3 2 2 3 3 2 3 2 1 ...  
## $ Contacted : Factor w/ 2 levels "N","Y": 1 2 2 2 2 1 2 2 2 1 ...  
## $ Source : Factor w/ 29 levels "S122","S123",..: 1 1 16 16 8 7 1 7 7 1 ...  
## $ Source\_Category : Factor w/ 7 levels "A","B","C","D",..: 7 7 2 2 2 2 3 2 2 2 ...  
## $ Existing\_EMI : num 0 0 0 0 2500 0 0 0 0 0 ...  
## $ Loan\_Amount : int NA 20000 45000 92000 50000 NA 130000 30000 66000 NA ...  
## $ Loan\_Period : int NA 2 4 5 2 NA 5 3 5 NA ...  
## $ Interest\_Rate : num NA 13.2 NA NA NA ...  
## $ EMI : int NA 953 NA NA NA NA 3082 1088 1749 NA ...  
## $ Var1 : int 0 10 0 7 10 0 10 0 7 0 ...  
## $ Approved : int 0 0 0 0 0 0 0 0 0 0 ...

#Sample test for finding patterns in 'NA' (this one shows why the following situtaion was inferred as no loan/ zeroes)  
sum((is.na(McKTrain$Interest\_Rate)) & (is.na(McKTrain$EMI)))

## [1] 47437

# Data processing

Data processing/ munging has been done using external functions listed at the very end of this file.

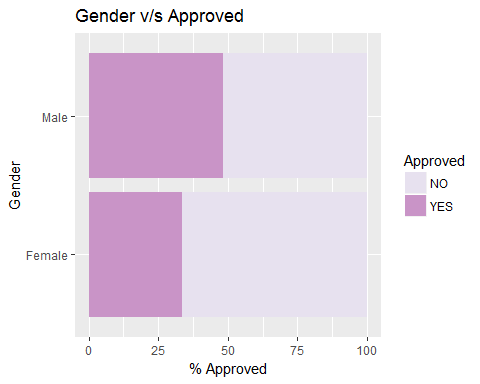
trainData\_WIP <- McK\_pre\_SMOTE\_common(McKTrain)  
trainData\_WIP <- McK\_pre\_proc\_train\_only(trainData\_WIP)  
#fixing imbalanced classes in 2 class dependent variable  
library(DMwR)  
set.seed(76)  
McKTrain\_SMOTE <- SMOTE(Approved~., data= trainData\_WIP)  
trainData\_WIP <- McK\_pre\_proc\_common(McKTrain\_SMOTE)  
  
testData\_WIP <- McK\_pre\_SMOTE\_common(McKTest)  
testData\_WIP <- McK\_pre\_proc\_common(testData\_WIP)

## Sample EDA tests on processed data

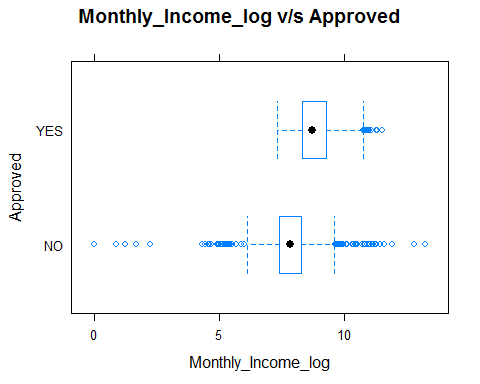
library(dplyr)  
library(ggplot2)  
library(caret)  
#Bivariate analysis: Gender v/s Approved  
chisq.test(trainData\_WIP$Gender, trainData\_WIP$Approved)

##   
## Pearson's Chi-squared test with Yates' continuity correction  
##   
## data: trainData\_WIP$Gender and trainData\_WIP$Approved  
## X-squared = 145.15, df = 1, p-value < 2.2e-16

trainData\_WIP %>% group\_by(Gender, Approved) %>%   
 summarise(count=n()) %>%   
 mutate(perc=count/sum(count)) %>%  
 ggplot(aes(x=Gender, y=perc\*100, fill=Approved)) +  
 geom\_bar(stat="identity") +  
 geom\_col(show.legend = T, ) +   
 labs(title="Gender v/s Approved", y="% Approved") +  
 scale\_fill\_brewer(palette="PuRd") +  
 coord\_flip()



#Univariate analysis: Monthly\_Income\_log v/s Approved  
bwplot(Approved~Monthly\_Income\_log, data=trainData\_WIP,   
 xlab = 'Monthly\_Income\_log', ylab = 'Approved', main= 'Monthly\_Income\_log v/s Approved')



# Model training and evaluation

To pick good predictors out of the 18 remaining independent variables, a random forest model was fit first. From variable importance results of this model, these 9 predictors are picked: 'Monthly\_Income\_log', 'Lead\_Age', 'Existing\_EMI\_log', 'App\_Age', "Loan\_Amount", 'Var1', 'Loan\_Period', 'Interest\_Rate', 'EMI'.  
7 new models with different algorithms were fitted. No suprises in the Kaggle favorite, XGBoost, performing best.

## Predictor selection using random forest

#predictor selection using with RF  
Control <- trainControl(  
 #method = 'repeatedcv', number = 10, repeats = 3,   
 classProbs = TRUE, summaryFunction = twoClassSummary)  
model\_RF\_SMOTE <- train(Approved~., data = trainData\_WIP[,-which(names(trainData\_WIP) %in% c('ID'))],   
 method = 'rf', trControl = Control, metric = 'ROC')

## Viewing and picking top predictors

library(caret)  
varImp(model\_RF\_SMOTE, scale = FALSE)

## rf variable importance  
##   
## only 20 most important variables shown (out of 49)  
##   
## Overall  
## Monthly\_Income\_log 1087.02  
## Lead\_Age 615.76  
## Existing\_EMI\_log 296.09  
## App\_Age 289.21  
## Loan\_Amount 176.65  
## Var110 161.01  
## Loan\_Period 129.18  
## Interest\_Rate 96.63  
## EMI 95.77  
## SourceS133 39.54  
## Primary\_Bank\_TypeP 37.05  
## SourceS134 35.74  
## Var17 32.88  
## Employer\_Category1A 30.45  
## GenderMale 27.62  
## Primary\_Bank\_TypeG 26.97  
## Employer\_Category1B 26.58  
## Employer\_Category24 24.85  
## SourceS137 24.74  
## Source\_CategoryB 24.64

keeps <- c('Monthly\_Income\_log', 'Lead\_Age', 'Existing\_EMI\_log', 'App\_Age', "Loan\_Amount", 'Var1',   
 'Loan\_Period', 'Interest\_Rate', 'EMI', 'Approved', 'ID') #picking predictor columns  
trainData\_WIP\_Trimmed <- trainData\_WIP[,names(trainData\_WIP) %in% keeps] #keeping above columns

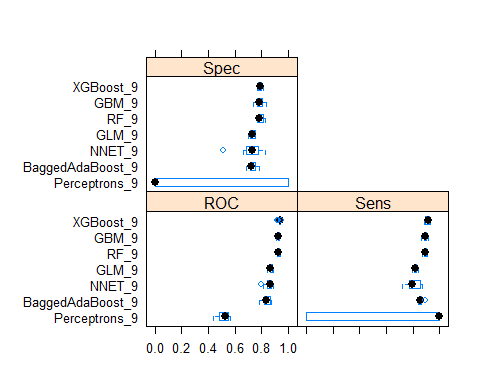
## Model training

Caret's ability to optimize parameters is leveraged in training models with 7 different algorithms.

Control <- trainControl(method = 'cv', number = 10, classProbs = TRUE, summaryFunction = twoClassSummary)  
  
#training with XGBoost  
set.seed(7)  
model\_xgbTree\_SMOTE\_Trimmed <- train(Approved~., data = trainData\_WIP\_Trimmed[,-which(names(trainData\_WIP\_Trimmed) %in% c('ID'))],   
 method="xgbTree", trControl=Control, verbose=FALSE, trace=FALSE)  
  
#training with Bagged AdaBoost  
set.seed(7)  
model\_BaggedAdaBoost\_SMOTE\_Trimmed <- train(Approved~., data = trainData\_WIP\_Trimmed[,-which(names(trainData\_WIP\_Trimmed) %in% c('ID'))],   
 method="AdaBag", trControl=Control, verbose=FALSE, trace=FALSE)  
  
#training with 'Multi-Layer Perceptron, multiple layers'  
set.seed(7)  
model\_MPML\_SMOTE\_Trimmed <- train(Approved~., data = trainData\_WIP\_Trimmed[,-which(names(trainData\_WIP\_Trimmed) %in% c('ID'))],   
 method="mlpWeightDecayML", trControl=Control, verbose=FALSE, trace=FALSE)  
  
#training with RF  
set.seed(7)  
model\_RF\_SMOTE\_Trimmed <- train(Approved~., data = trainData\_WIP\_Trimmed[,-which(names(trainData\_WIP\_Trimmed) %in% c('ID'))],   
 method = 'rf', trControl = Control, metric = 'ROC')  
  
#training with GBM  
set.seed(7)  
model\_GBM\_SMOTE\_Trimmed <- train(Approved~., data = trainData\_WIP\_Trimmed[,-which(names(trainData\_WIP\_Trimmed) %in% c('ID'))],   
 method="gbm", trControl=Control, verbose=FALSE)  
  
#training with GLM  
set.seed(7)  
model\_GLM\_SMOTE\_Trimmed <- train(Approved~., data = trainData\_WIP\_Trimmed[,-which(names(trainData\_WIP\_Trimmed) %in% c('ID'))],   
 method="glm", trControl=Control)  
  
  
#training with NNET  
set.seed(7)  
model\_NNET\_SMOTE\_Trimmed <- train(Approved~., data = trainData\_WIP\_Trimmed[,-which(names(trainData\_WIP\_Trimmed) %in% c('ID'))],   
 method="nnet", trControl=Control, verbose=FALSE, trace=FALSE)

# Trained models, performance summary

#collect resamples for models with top 9 predictors  
results\_9 <- resamples(list(XGBoost\_9=model\_xgbTree\_SMOTE\_Trimmed, BaggedAdaBoost\_9=model\_BaggedAdaBoost\_SMOTE\_Trimmed,   
 Perceptrons\_9 = model\_MPML\_SMOTE\_Trimmed, NNET\_9=model\_NNET\_SMOTE\_Trimmed, GLM\_9= model\_GLM\_SMOTE\_Trimmed,   
 GBM\_9 = model\_GBM\_SMOTE\_Trimmed, RF\_9 = model\_RF\_SMOTE\_Trimmed))   
# summarize the performance  
bwplot(results\_9)



summary(results\_9)

##   
## Call:  
## summary.resamples(object = results\_9)  
##   
## Models: XGBoost\_9, BaggedAdaBoost\_9, Perceptrons\_9, NNET\_9, GLM\_9, GBM\_9, RF\_9   
## Number of resamples: 10   
##   
## ROC   
## Min. 1st Qu. Median Mean 3rd Qu.  
## XGBoost\_9 0.9203912 0.9347907 0.9387655 0.9359886 0.9390259  
## BaggedAdaBoost\_9 0.7857955 0.8222258 0.8378949 0.8394788 0.8693872  
## Perceptrons\_9 0.4343642 0.4848035 0.5298403 0.5137784 0.5481706  
## NNET\_9 0.8004734 0.8446211 0.8666378 0.8548535 0.8724950  
## GLM\_9 0.8454761 0.8644672 0.8676390 0.8699210 0.8782399  
## GBM\_9 0.9132826 0.9245212 0.9289196 0.9281759 0.9335142  
## RF\_9 0.9176198 0.9255515 0.9289055 0.9303453 0.9366309  
## Max. NA's  
## XGBoost\_9 0.9466952 0  
## BaggedAdaBoost\_9 0.8747477 0  
## Perceptrons\_9 0.5651712 0  
## NNET\_9 0.8875112 0  
## GLM\_9 0.8873831 0  
## GBM\_9 0.9380367 0  
## RF\_9 0.9416090 0  
##   
## Sens   
## Min. 1st Qu. Median Mean 3rd Qu.  
## XGBoost\_9 0.8848039 0.8933824 0.9129902 0.9100490 0.9246324  
## BaggedAdaBoost\_9 0.8406863 0.8504902 0.8553922 0.8571078 0.8596814  
## Perceptrons\_9 0.0000000 0.0000000 1.0000000 0.6000000 1.0000000  
## NNET\_9 0.7230392 0.7787990 0.7953431 0.8066176 0.8517157  
## GLM\_9 0.7990196 0.8100490 0.8161765 0.8181373 0.8272059  
## GBM\_9 0.8651961 0.8866422 0.8933824 0.8931373 0.9013480  
## RF\_9 0.8725490 0.8872549 0.8933824 0.8926471 0.8995098  
## Max. NA's  
## XGBoost\_9 0.9338235 0  
## BaggedAdaBoost\_9 0.8921569 0  
## Perceptrons\_9 1.0000000 0  
## NNET\_9 0.8676471 0  
## GLM\_9 0.8382353 0  
## GBM\_9 0.9166667 0  
## RF\_9 0.9117647 0  
##   
## Spec   
## Min. 1st Qu. Median Mean 3rd Qu.  
## XGBoost\_9 0.7712418 0.7794118 0.7892157 0.7892157 0.7982026  
## BaggedAdaBoost\_9 0.6830065 0.7165033 0.7222222 0.7290850 0.7475490  
## Perceptrons\_9 0.0000000 0.0000000 0.0000000 0.4000000 1.0000000  
## NNET\_9 0.5163399 0.6846405 0.7287582 0.7186275 0.7777778  
## GLM\_9 0.6993464 0.7165033 0.7336601 0.7310458 0.7508170  
## GBM\_9 0.7418301 0.7704248 0.7859477 0.7866013 0.8071895  
## RF\_9 0.7679739 0.7769608 0.7875817 0.7944444 0.8129085  
## Max. NA's  
## XGBoost\_9 0.8104575 0  
## BaggedAdaBoost\_9 0.7810458 0  
## Perceptrons\_9 1.0000000 0  
## NNET\_9 0.8267974 0  
## GLM\_9 0.7549020 0  
## GBM\_9 0.8333333 0  
## RF\_9 0.8300654 0

## XGBoost's performance

Being the best performing model, the XGBoost model is examined in detail below. The exhaustive tuning performed by caret is shown.

#taking a better look at the XGBoost model  
model\_xgbTree\_SMOTE\_Trimmed

## eXtreme Gradient Boosting   
##   
## 7140 samples  
## 9 predictor  
## 2 classes: 'NO', 'YES'   
##   
## No pre-processing  
## Resampling: Cross-Validated (10 fold)   
## Summary of sample sizes: 6426, 6426, 6426, 6426, 6426, 6426, ...   
## Resampling results across tuning parameters:  
##   
## eta max\_depth colsample\_bytree subsample nrounds ROC   
## 0.3 1 0.6 0.50 50 0.9059757  
## 0.3 1 0.6 0.50 100 0.9130847  
## 0.3 1 0.6 0.50 150 0.9164124  
## 0.3 1 0.6 0.75 50 0.9050594  
## 0.3 1 0.6 0.75 100 0.9124900  
## 0.3 1 0.6 0.75 150 0.9158853  
## 0.3 1 0.6 1.00 50 0.9042436  
## 0.3 1 0.6 1.00 100 0.9117475  
## 0.3 1 0.6 1.00 150 0.9152525  
## 0.3 1 0.8 0.50 50 0.9053801  
## 0.3 1 0.8 0.50 100 0.9140499  
## 0.3 1 0.8 0.50 150 0.9177436  
## 0.3 1 0.8 0.75 50 0.9058079  
## 0.3 1 0.8 0.75 100 0.9135281  
## 0.3 1 0.8 0.75 150 0.9168989  
## 0.3 1 0.8 1.00 50 0.9042151  
## 0.3 1 0.8 1.00 100 0.9122225  
## 0.3 1 0.8 1.00 150 0.9152838  
## 0.3 2 0.6 0.50 50 0.9256184  
## 0.3 2 0.6 0.50 100 0.9315628  
## 0.3 2 0.6 0.50 150 0.9335464  
## 0.3 2 0.6 0.75 50 0.9267850  
## 0.3 2 0.6 0.75 100 0.9338672  
## 0.3 2 0.6 0.75 150 0.9350843  
## 0.3 2 0.6 1.00 50 0.9256344  
## 0.3 2 0.6 1.00 100 0.9336077  
## 0.3 2 0.6 1.00 150 0.9352889  
## 0.3 2 0.8 0.50 50 0.9266128  
## 0.3 2 0.8 0.50 100 0.9315343  
## 0.3 2 0.8 0.50 150 0.9327839  
## 0.3 2 0.8 0.75 50 0.9263056  
## 0.3 2 0.8 0.75 100 0.9331775  
## 0.3 2 0.8 0.75 150 0.9344086  
## 0.3 2 0.8 1.00 50 0.9255583  
## 0.3 2 0.8 1.00 100 0.9338816  
## 0.3 2 0.8 1.00 150 0.9351832  
## 0.3 3 0.6 0.50 50 0.9306937  
## 0.3 3 0.6 0.50 100 0.9318764  
## 0.3 3 0.6 0.50 150 0.9323650  
## 0.3 3 0.6 0.75 50 0.9318567  
## 0.3 3 0.6 0.75 100 0.9346573  
## 0.3 3 0.6 0.75 150 0.9345949  
## 0.3 3 0.6 1.00 50 0.9325027  
## 0.3 3 0.6 1.00 100 0.9357747  
## 0.3 3 0.6 1.00 150 0.9355857  
## 0.3 3 0.8 0.50 50 0.9313353  
## 0.3 3 0.8 0.50 100 0.9332556  
## 0.3 3 0.8 0.50 150 0.9332789  
## 0.3 3 0.8 0.75 50 0.9330734  
## 0.3 3 0.8 0.75 100 0.9351099  
## 0.3 3 0.8 0.75 150 0.9346285  
## 0.3 3 0.8 1.00 50 0.9331463  
## 0.3 3 0.8 1.00 100 0.9346073  
## 0.3 3 0.8 1.00 150 0.9349905  
## 0.4 1 0.6 0.50 50 0.9091251  
## 0.4 1 0.6 0.50 100 0.9162193  
## 0.4 1 0.6 0.50 150 0.9191253  
## 0.4 1 0.6 0.75 50 0.9086301  
## 0.4 1 0.6 0.75 100 0.9154980  
## 0.4 1 0.6 0.75 150 0.9187544  
## 0.4 1 0.6 1.00 50 0.9079825  
## 0.4 1 0.6 1.00 100 0.9140403  
## 0.4 1 0.6 1.00 150 0.9172990  
## 0.4 1 0.8 0.50 50 0.9098928  
## 0.4 1 0.8 0.50 100 0.9154332  
## 0.4 1 0.8 0.50 150 0.9192182  
## 0.4 1 0.8 0.75 50 0.9082929  
## 0.4 1 0.8 0.75 100 0.9154131  
## 0.4 1 0.8 0.75 150 0.9185257  
## 0.4 1 0.8 1.00 50 0.9072128  
## 0.4 1 0.8 1.00 100 0.9141188  
## 0.4 1 0.8 1.00 150 0.9171725  
## 0.4 2 0.6 0.50 50 0.9256388  
## 0.4 2 0.6 0.50 100 0.9319364  
## 0.4 2 0.6 0.50 150 0.9331840  
## 0.4 2 0.6 0.75 50 0.9274838  
## 0.4 2 0.6 0.75 100 0.9329236  
## 0.4 2 0.6 0.75 150 0.9337418  
## 0.4 2 0.6 1.00 50 0.9297482  
## 0.4 2 0.6 1.00 100 0.9352637  
## 0.4 2 0.6 1.00 150 0.9359886  
## 0.4 2 0.8 0.50 50 0.9284166  
## 0.4 2 0.8 0.50 100 0.9321755  
## 0.4 2 0.8 0.50 150 0.9314759  
## 0.4 2 0.8 0.75 50 0.9282063  
## 0.4 2 0.8 0.75 100 0.9346205  
## 0.4 2 0.8 0.75 150 0.9357935  
## 0.4 2 0.8 1.00 50 0.9293873  
## 0.4 2 0.8 1.00 100 0.9340726  
## 0.4 2 0.8 1.00 150 0.9351043  
## 0.4 3 0.6 0.50 50 0.9295844  
## 0.4 3 0.6 0.50 100 0.9311307  
## 0.4 3 0.6 0.50 150 0.9313381  
## 0.4 3 0.6 0.75 50 0.9335981  
## 0.4 3 0.6 0.75 100 0.9345768  
## 0.4 3 0.6 0.75 150 0.9343081  
## 0.4 3 0.6 1.00 50 0.9338988  
## 0.4 3 0.6 1.00 100 0.9348440  
## 0.4 3 0.6 1.00 150 0.9343762  
## 0.4 3 0.8 0.50 50 0.9316056  
## 0.4 3 0.8 0.50 100 0.9310089  
## 0.4 3 0.8 0.50 150 0.9284194  
## 0.4 3 0.8 0.75 50 0.9320806  
## 0.4 3 0.8 0.75 100 0.9330402  
## 0.4 3 0.8 0.75 150 0.9320998  
## 0.4 3 0.8 1.00 50 0.9339501  
## 0.4 3 0.8 1.00 100 0.9346429  
## 0.4 3 0.8 1.00 150 0.9336818  
## Sens Spec   
## 0.8450980 0.7774510  
## 0.8644608 0.7807190  
## 0.8725490 0.7764706  
## 0.8458333 0.7767974  
## 0.8651961 0.7803922  
## 0.8708333 0.7800654  
## 0.8443627 0.7705882  
## 0.8605392 0.7803922  
## 0.8691176 0.7764706  
## 0.8379902 0.7833333  
## 0.8656863 0.7741830  
## 0.8767157 0.7787582  
## 0.8465686 0.7761438  
## 0.8644608 0.7794118  
## 0.8696078 0.7790850  
## 0.8455882 0.7748366  
## 0.8602941 0.7807190  
## 0.8669118 0.7781046  
## 0.8835784 0.7839869  
## 0.9004902 0.7888889  
## 0.9051471 0.7941176  
## 0.8919118 0.7866013  
## 0.9039216 0.7892157  
## 0.9117647 0.7885621  
## 0.8889706 0.7758170  
## 0.9063725 0.7843137  
## 0.9102941 0.7879085  
## 0.8897059 0.7826797  
## 0.9017157 0.7836601  
## 0.9053922 0.7895425  
## 0.8938725 0.7807190  
## 0.9044118 0.7915033  
## 0.9066176 0.7892157  
## 0.8919118 0.7718954  
## 0.9061275 0.7856209  
## 0.9093137 0.7862745  
## 0.8997549 0.7905229  
## 0.8960784 0.7970588  
## 0.8980392 0.7895425  
## 0.8995098 0.7908497  
## 0.9093137 0.7957516  
## 0.9053922 0.7947712  
## 0.9026961 0.7830065  
## 0.9090686 0.7918301  
## 0.9073529 0.7921569  
## 0.9039216 0.7862745  
## 0.8982843 0.7931373  
## 0.8960784 0.7954248  
## 0.9085784 0.7836601  
## 0.9083333 0.7898693  
## 0.9056373 0.7924837  
## 0.9083333 0.7839869  
## 0.9100490 0.7892157  
## 0.9068627 0.7895425  
## 0.8509804 0.7839869  
## 0.8713235 0.7833333  
## 0.8752451 0.7813725  
## 0.8514706 0.7784314  
## 0.8671569 0.7807190  
## 0.8769608 0.7781046  
## 0.8517157 0.7774510  
## 0.8649510 0.7771242  
## 0.8720588 0.7777778  
## 0.8578431 0.7754902  
## 0.8696078 0.7774510  
## 0.8745098 0.7833333  
## 0.8492647 0.7866013  
## 0.8656863 0.7830065  
## 0.8750000 0.7826797  
## 0.8468137 0.7836601  
## 0.8634804 0.7781046  
## 0.8715686 0.7758170  
## 0.8901961 0.7790850  
## 0.9002451 0.7898693  
## 0.9002451 0.7908497  
## 0.8953431 0.7846405  
## 0.9019608 0.7898693  
## 0.9017157 0.7875817  
## 0.8977941 0.7879085  
## 0.9105392 0.7911765  
## 0.9100490 0.7892157  
## 0.8955882 0.7879085  
## 0.9029412 0.7901961  
## 0.8995098 0.7872549  
## 0.8997549 0.7800654  
## 0.9085784 0.7911765  
## 0.9075980 0.7921569  
## 0.8965686 0.7866013  
## 0.9051471 0.7908497  
## 0.9058824 0.7905229  
## 0.8955882 0.7836601  
## 0.8985294 0.7836601  
## 0.8928922 0.7934641  
## 0.9034314 0.7892157  
## 0.9066176 0.7954248  
## 0.9007353 0.7977124  
## 0.9071078 0.7924837  
## 0.9053922 0.7882353  
## 0.9026961 0.7915033  
## 0.8970588 0.7915033  
## 0.8946078 0.7918301  
## 0.8933824 0.7944444  
## 0.9061275 0.7911765  
## 0.9009804 0.7931373  
## 0.8990196 0.7915033  
## 0.9071078 0.7872549  
## 0.9031863 0.7934641  
## 0.9004902 0.7918301  
##   
## Tuning parameter 'gamma' was held constant at a value of 0  
##   
## Tuning parameter 'min\_child\_weight' was held constant at a value of 1  
## ROC was used to select the optimal model using the largest value.  
## The final values used for the model were nrounds = 150, max\_depth = 2,  
## eta = 0.4, gamma = 0, colsample\_bytree = 0.6, min\_child\_weight = 1  
## and subsample = 1.

## Predictions

Code used to make and save predictions to submit to the hackathon.

XGB\_results <- predict(model\_xgbTree\_SMOTE\_Trimmed, newdata = testData\_WIP)  
submission <- cbind(as.character(McKTest$ID), XGB\_results)  
submission <- as.data.frame(submission)  
names(submission) <- c('ID', 'Approved')  
levels(submission$Approved)[levels(submission$Approved) == 1] <- 0  
levels(submission$Approved)[levels(submission$Approved) == 2] <- 1  
write.csv(submission, file='submission\_XGB.csv', row.names = FALSE)  
  
#similarly for all other models

## Using functions

To maintain high reuse and repeatability, data pre-processing/ munging has been placed into functions.

McK\_pre\_SMOTE\_common <- function (McKSalesData) {  
 McKSalesDataTrimmed <- McKSalesData  
 #dropping less informative columns  
 drops <- c('Employer\_Code', 'City\_Code', 'Customer\_Existing\_Primary\_Bank\_Code') #picking less informative columns  
 McKSalesDataTrimmed <- McKSalesDataTrimmed[,!names(McKSalesDataTrimmed) %in% drops] #dropping above columns  
   
 McKSalesDataTrimmed$Employer\_Category2[is.na(McKSalesDataTrimmed$Employer\_Category2)] <- 'EC2' #fixing NA in a variable before making it categorical  
   
 #converting appropriate columns to categorical  
 change\_to\_fact <- c('Var1','Employer\_Category2')  
 McKSalesDataTrimmed[change\_to\_fact] <- lapply(McKSalesDataTrimmed[change\_to\_fact], factor)  
   
 McKSalesDataTrimmed  
}  
  
McK\_pre\_proc\_train\_only <- function (McKSalesData) {  
 McKSalesDataTrimmed <- McKSalesData  
 #converting appropriate columns to categorical  
 change\_to\_fact <- c('Approved')  
 McKSalesDataTrimmed[change\_to\_fact] <- lapply(McKSalesDataTrimmed[change\_to\_fact], factor)  
   
 levels(McKSalesDataTrimmed$Approved)[levels(McKSalesDataTrimmed$Approved) == '0'] <- 'NO'  
 levels(McKSalesDataTrimmed$Approved)[levels(McKSalesDataTrimmed$Approved) == '1'] <- 'YES'  
   
   
 McKSalesDataTrimmed  
}  
  
McK\_pre\_proc\_test\_only <- function (McKSalesData) {  
 McKSalesData$Existing\_EMI[is.na(McKSalesData$Existing\_EMI)] <- "" ## dummy value for NA rows in test data  
 McKSalesData  
}  
  
calc\_age <- function(birthDate, refDate = Sys.Date()) { #function that calculates elapsed time since a given date   
 #to current/ system date (modified code found on github)  
 require(lubridate)  
 duration <- as.duration(interval(birthDate, refDate))  
 cbind((duration/(60\*60\*24\*365)), (duration/(60\*60\*24)))  
}  
  
McK\_pre\_proc\_common <- function (McKSalesData) {  
 McKSalesDataTrimmed <- McKSalesData  
 #Unlabelled levels fix  
 levels(McKSalesDataTrimmed$Primary\_Bank\_Type)[levels(McKSalesDataTrimmed$Primary\_Bank\_Type) == ''] <- 'PBT'  
 levels(McKSalesDataTrimmed$Employer\_Category1)[levels(McKSalesDataTrimmed$Employer\_Category1) == ''] <- 'EC1'  
 levels(McKSalesDataTrimmed$City\_Category)[levels(McKSalesDataTrimmed$City\_Category) == ''] <- 'CC'  
   
   
 #fixes for missing data in continuous variable columns  
 McKSalesDataTrimmed$Existing\_EMI[is.na(McKSalesDataTrimmed$Existing\_EMI)] <- 0  
 McKSalesDataTrimmed$Loan\_Amount[is.na(McKSalesDataTrimmed$Loan\_Amount)] <- 0  
 McKSalesDataTrimmed$Loan\_Period[is.na(McKSalesDataTrimmed$Loan\_Period)] <- 0  
 McKSalesDataTrimmed$Interest\_Rate[is.na(McKSalesDataTrimmed$Interest\_Rate)] <- 0  
 McKSalesDataTrimmed$EMI[is.na(McKSalesDataTrimmed$EMI)] <- 0  
   
 #transforming continous variables (other transformations like upper bound, exp and sqrt tested and rejected)  
 #the +1 is because of zeros in the data and we need to prevent the invalid log of zero calculation  
 McKSalesDataTrimmed$Monthly\_Income\_log <- log(McKSalesDataTrimmed$Monthly\_Income+1)  
 McKSalesDataTrimmed$Existing\_EMI\_log <- log(McKSalesDataTrimmed$Existing\_EMI+1)  
 drops\_trfm <- c('Existing\_EMI', 'Monthly\_Income')  
 McKSalesDataTrimmed <- McKSalesDataTrimmed[,!names(McKSalesDataTrimmed) %in% drops\_trfm] #dropping the original/ untransformed columns  
   
 #converting dates to ages  
 McKSalesDataTrimmed$Lead\_Age <- (calc\_age(as.Date(McKSalesDataTrimmed$Lead\_Creation\_Date, format = '%d/%m/%y')))[,2]  
 McKSalesDataTrimmed <- McKSalesDataTrimmed[,!names(McKSalesDataTrimmed) %in% c("Lead\_Creation\_Date")] #dropping 'Lead\_Creation\_Date' column  
 #several ages are negative (POSIX standards issue with converting yy to YYYY format) and need to be fixed.  
 levels(McKSalesDataTrimmed$DOB)[levels(McKSalesDataTrimmed$DOB) == ''] <- '01/01/2000' ## dummy value for blank rows in data  
 McKSalesDataTrimmed$App\_Age <- (calc\_age(as.Date(McKSalesDataTrimmed$DOB, format = '%d/%m/%y')))[,1]  
 neg\_age\_index <- as.Date(McKSalesDataTrimmed$DOB, format = '%d/%m/%y') > Sys.Date()  
 McKSalesDataTrimmed$App\_Age[neg\_age\_index] <- McKSalesDataTrimmed$App\_Age[neg\_age\_index]+100  
 McKSalesDataTrimmed <- McKSalesDataTrimmed[,!names(McKSalesDataTrimmed) %in% c("DOB")] #dropping 'DOB' column  
  
  
 #dropping unused levels in categorical variables  
 McKSalesDataTrimmed$Source <- droplevels((McKSalesDataTrimmed$Source))  
   
 McKSalesDataTrimmed  
   
}