Northeastern University

LOAN DEFAULT

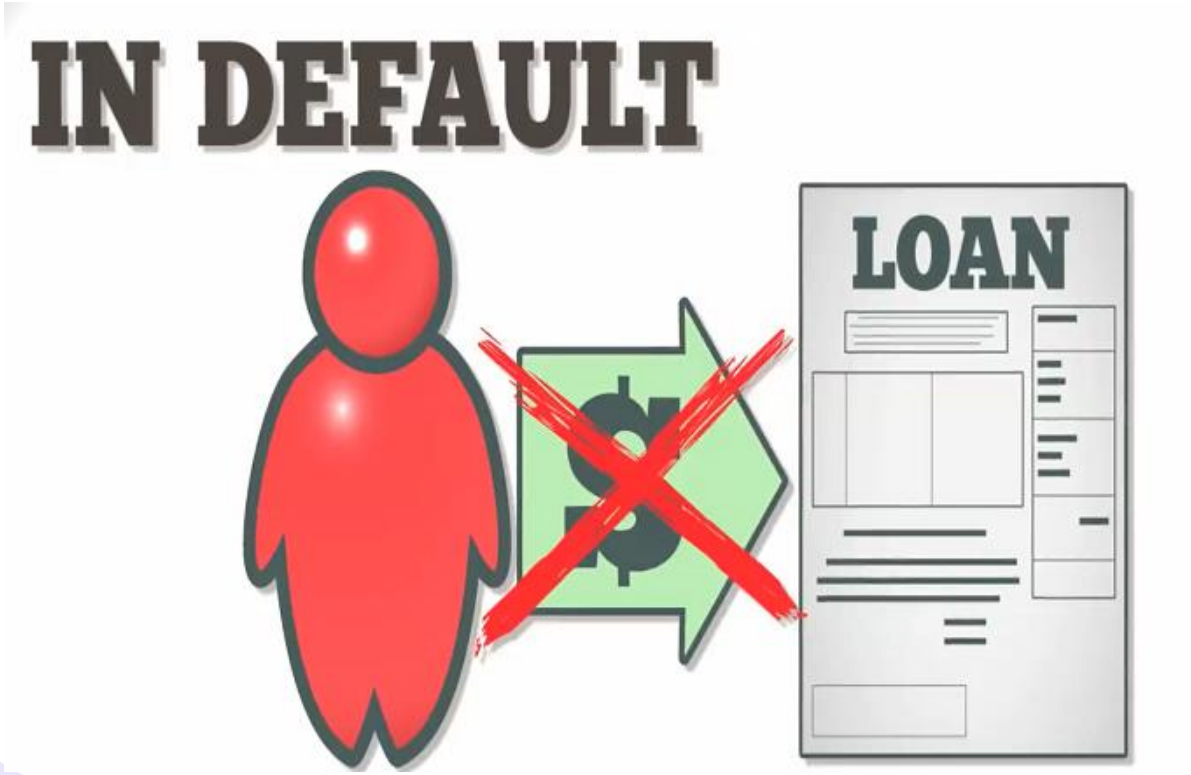
Outlier Group

Northeastern University - Toronto

ALY6040: Data Mining

INTRODUCTION

Business Context



Terminology

Default refers to the ability to repay.
Default: unable to pay back.
Non-default: able to repay.

Two scenarios of loss

Approve for default customers
Reject non-default customers

Research Questions

1. Given current customer information and historical data, would customer be able to repay the loan? – Classification problem
2. How to deal with imbalanced dataset

INTRODUCTION

Dataset at glimpse and project objective

Data set

- Information of current customers with Target variable (TARGET).
  TARGET 0: Non-default
  TARGET 1: Default
- 148,670 records and **34** variables

Project objective

- Master the data munging
- Deploy various methods of data mining for Classification problem
- Handle imbalanced class of target variable
- Familiarize ourselves with the target industry

INTRODUCTION

Overall process

Filter Selection
- Domain knowledge
- Statistic summary

Data preparation
- Exploration
- Transformation
- Validation

EDA
- Target variable analysis
- Correlation
- Explanatory predictors

Pre-model
- Define target metrics
- Feature selection for modeling

Modeling
- Logistic Regression
- Decision tree
- Gradient Boosting
- Random Forest

Handle imbalanced data

Improvement
- Data sampling
- Adjustment of model
- Adjustment of threshold, tunning hyperparameter

Comparison & Conclusion
- By performance metrics
- By domain knowledge

ANALYSIS

EDA

FINAL DATA SUMMARY

Final Data set

137,336 records x **24** variables

```
> str(df)
'data.frame':   137336 obs. of  24 variables:
 $ loan_limit                : Factor w/ 3 levels "cf","Missing",..: 1 1 1 1 1
 $ Gender                    : Factor w/ 4 levels "Female","Joint",..: 4 3 3 3
 $ approv_in_adv             : Factor w/ 3 levels "Missing","nopre",..: 2 2 3
 $ loan_type                 : Factor w/ 3 levels "type1","type2",..: 1 2 1 1
 $ loan_purpose              : Factor w/ 5 levels "Missing","p1",..: 2 2 2 5 2
 $ Credit_Worthiness         : Factor w/ 2 levels "l1","l2": 1 1 1 1 1 1 1 1 1
 $ open_credit               : Factor w/ 2 levels "nopc","opc": 1 1 1 1 1 1 1
 $ business_or_commercial    : Factor w/ 2 levels "b/c","nob/c": 2 1 2 2 2 2 2
 $ loan_amount               : int  116500 206500 406500 456500 696500 706500
 $ rate_of_interest          : num  3.99 3.99 4.56 4.25 4 ...
 $ term                      : num  360 360 360 360 360 360 360 360 360 360 ..
 $ interest_only             : Factor w/ 2 levels "int_only","not_int": 2 2 2
 $ lump_sum_payment          : Factor w/ 2 levels "lpsm","not_lpsm": 2 1 2 2 2
 $ property_value            : num  118000 418000 508000 658000 758000 ...
 $ occupancy_type            : Factor w/ 3 levels "ir","pr","sr": 2 2 2 2 2 2
 $ total_units               : num  1 1 1 1 1 1 1 1 1 1 ...
 $ income                    : num  1740 4980 9480 11880 10440 ...
 $ credit_type               : Factor w/ 4 levels "CIB","CRIF","EQUI",..: 4 3
 $ Credit_Score              : int  758 552 834 587 602 864 860 863 580 788 ..
 $ co.applicant_credit_type  : Factor w/ 2 levels "CIB","EXP": 1 2 1 1 2 2 2 1
 $ age                       : Factor w/ 7 levels "<25",">74","25-34",..: 3 6
 $ submission_of_application: Factor w/ 2 levels "not_inst","to_inst": 2 2 2
 $ Region                    : Factor w/ 4 levels "central","North",..: 4 2 4
 $ Status                    : int  1 1 0 0 0 0 0 0 0 0 ...
```

ANALYSIS

EDA

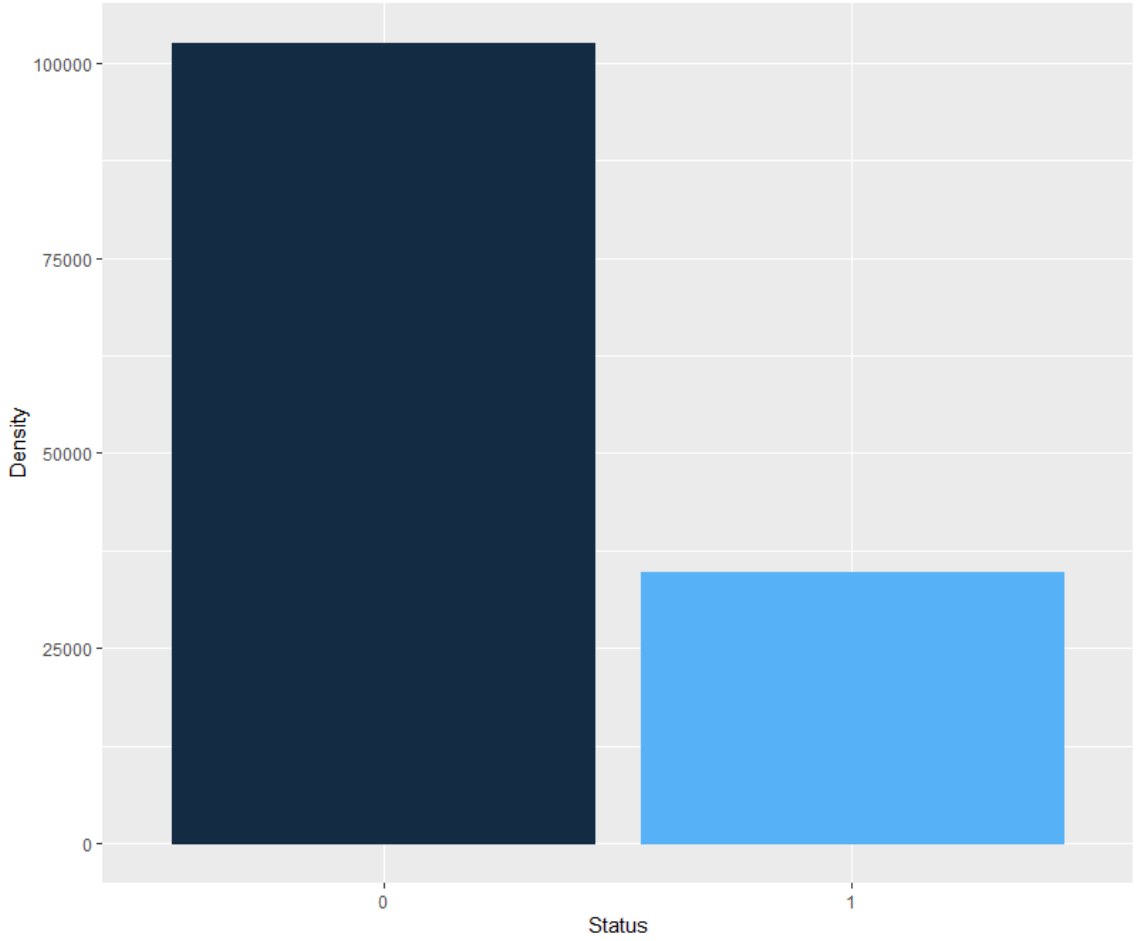
TARGET VARIABLE

Slightly imbalanced

75% non-default **25** default

Total Observations in Table: 137336

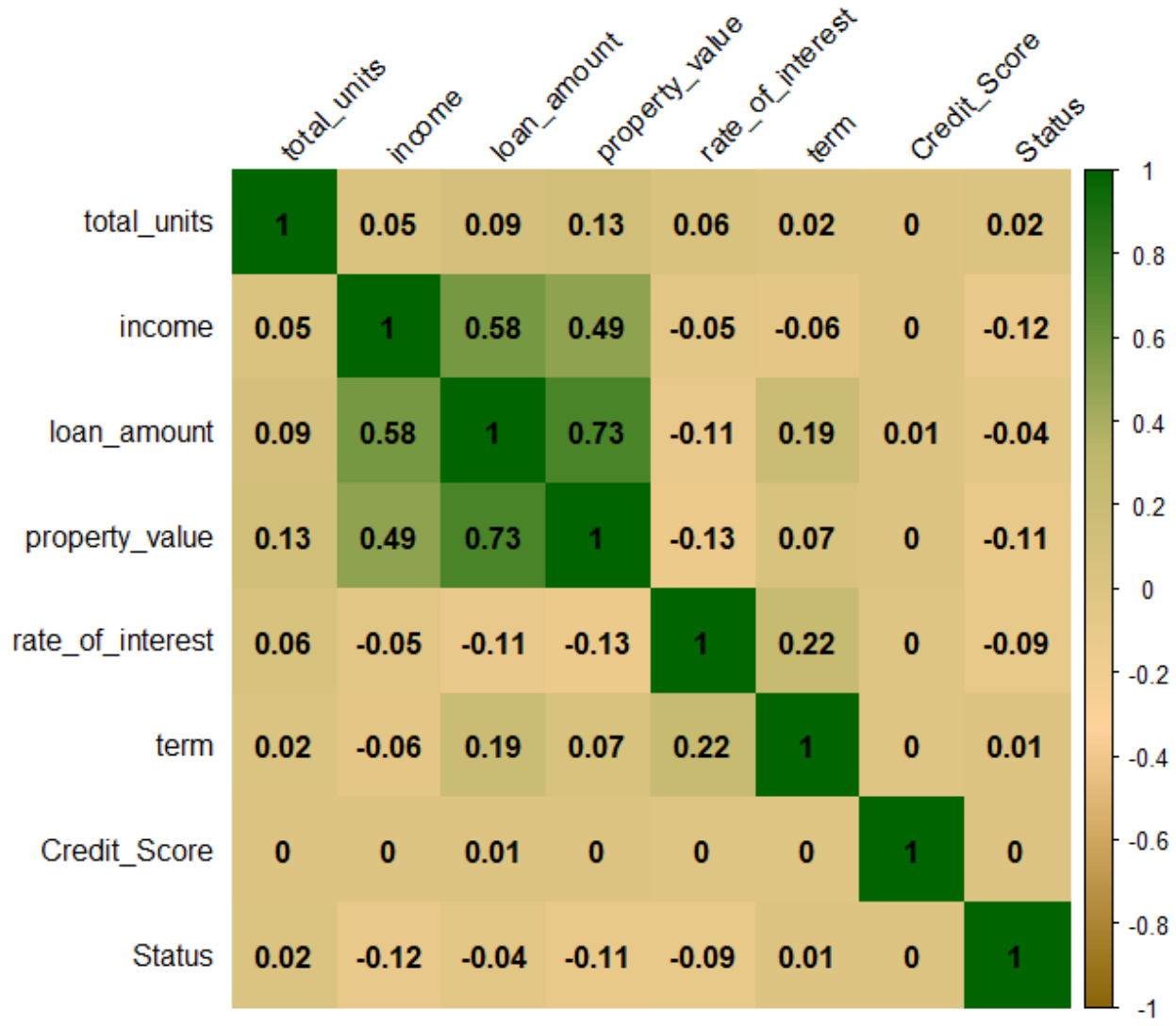
0	1
102551	34785
0.747	0.253



ANALYSIS

EDA

CORRELOGRAM



To Target variable (Status):

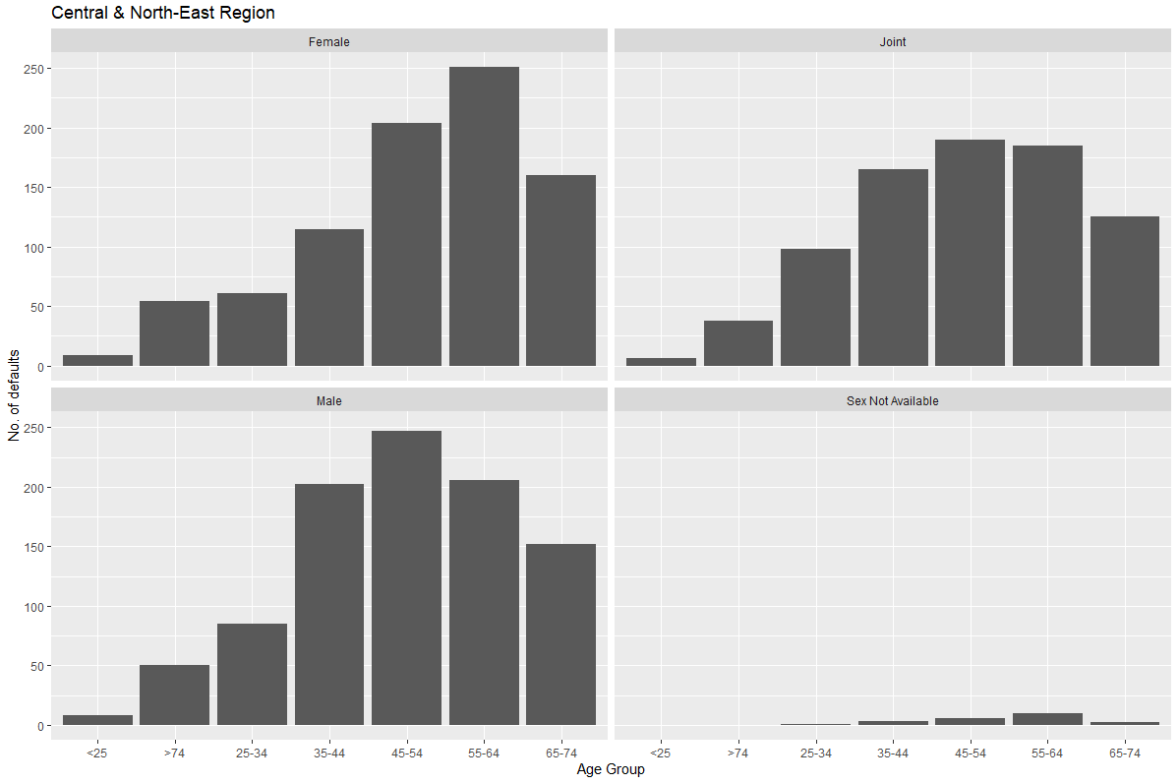
Mostly insignificant correlations

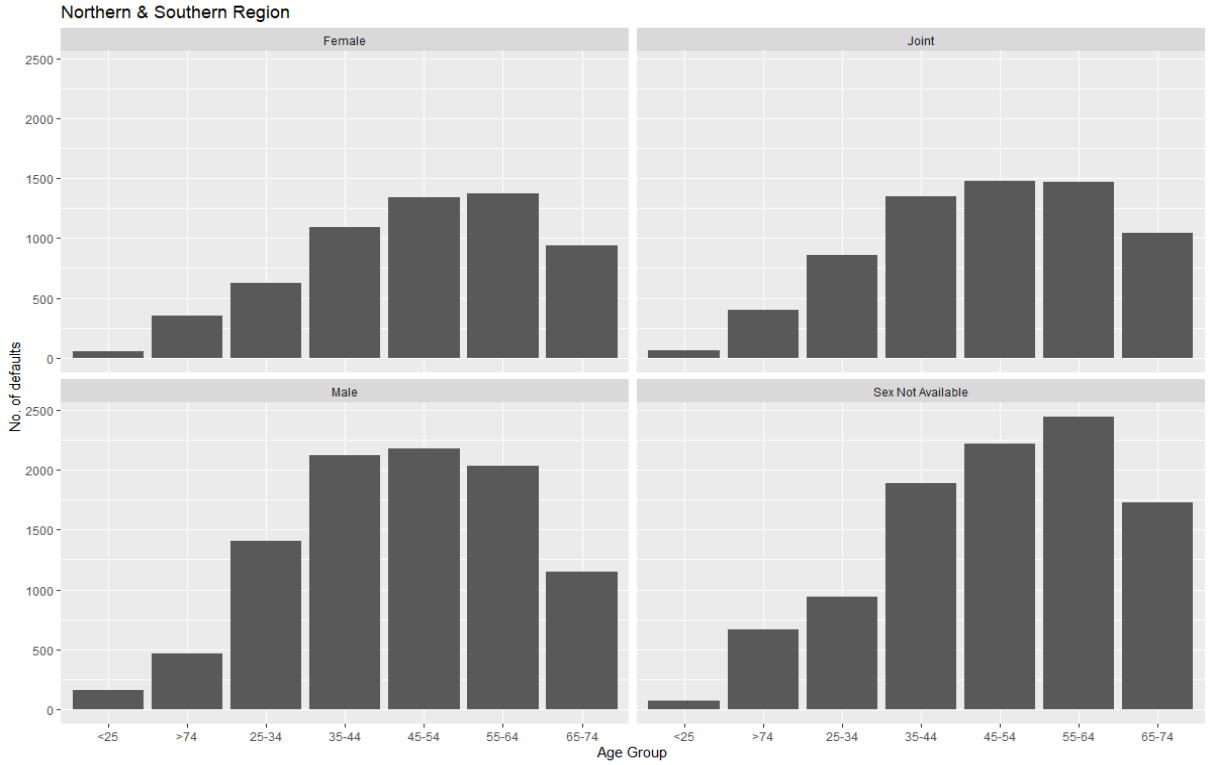
ANALYSIS

EDA

EXPLANATORY PREDICTORS

```
> table1

central   North North-East   south
   2271   15991        362   15926
```

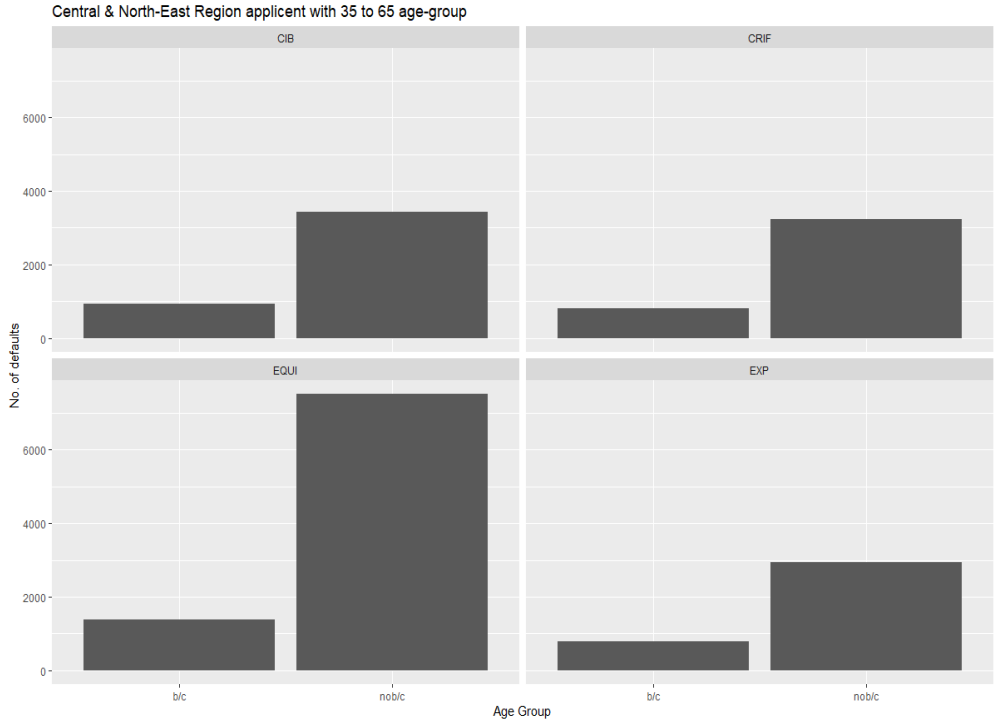


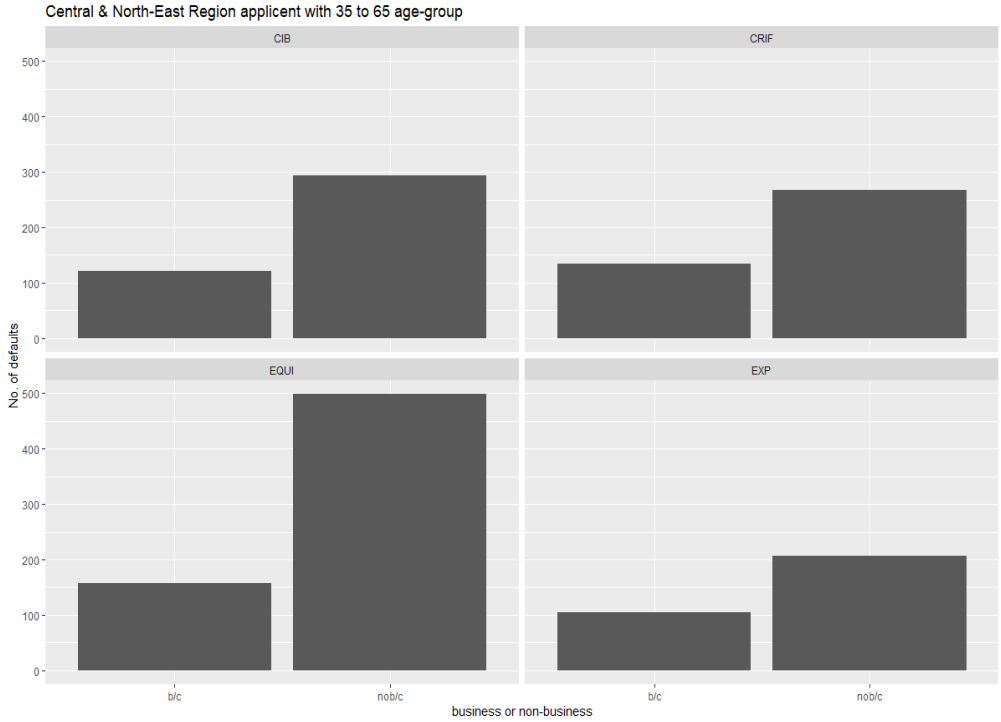


ANALYSIS

EDA

EXPLANATORY PREDICTORS






```
> table4
```

	CIB	CRIF	EQUI	EXP
b/c	930	815	1373	790
nob/c	3432	3226	7514	2927

```
> table5
```

	CIB	CRIF	EQUI	EXP
b/c	121	134	158	105
nob/c	294	267	499	206

MODELING

BACKGROUND RATIONALE

Class bias problem for ML

Risk for finance

Predict default customer (TARGET = 1) is more crucial than for
non-default customer (TARGET = 0).

False Negative (FN) is a dangerous error

So how to tackle this?

Data process	Modeling	Threshold
• Upsample • Downsample • SMOTE • ROSE	• Tune hyperparameters • Update model • Pruning	• Cut-off • Cp

MODELING

LOGISTICS REGRESSION Procedure

MODELING

LOGISTIC REGRESSION

Stepwise Logistics

Training set

```
> log.stepwise <- step(log.full, direction = "backward", trace = FALSE)
> summary(log.stepwise)

Call:
glm(formula = Status ~ loan_limit + Gender + approv_in_adv +
    loan_type + loan_purpose + Credit_Worthiness + open_credit +
    loan_amount + rate_of_interest + term + interest_only + lump_sum_payment +
    property_value + occupancy_type + total_units + income +
    credit_type + Credit_Score + co.applicant_credit_type + age +
    submission_of_application + Region, family = binomial(link = logit),
    data = training_set)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-4.0741  -0.5980  -0.4350   0.0033   3.2848
```

Full vs Stepwise

```
> # Compare two models
> anova(log.full,log.stepwise,test="LRT")
Analysis of Deviance Table

Model 1: Status ~ loan_limit + Gender + approv_in_adv + loan_type + loan_purpose +
    Credit_Worthiness + open_credit + business_or_commercial +
    loan_amount + rate_of_interest + term + interest_only + lump_sum_payment +
    property_value + occupancy_type + total_units + income +
    credit_type + Credit_Score + co.applicant_credit_type + age +
    submission_of_application + Region
Model 2: Status ~ loan_limit + Gender + approv_in_adv + loan_type + loan_purpose +
    Credit_Worthiness + open_credit + loan_amount + rate_of_interest +
    term + interest_only + lump_sum_payment + property_value +
    occupancy_type + total_units + income + credit_type + Credit_Score +
    co.applicant_credit_type + age + submission_of_application +
    Region
  Resid. Df Resid. Dev Df Deviance Pr(>Chi)
1    109827      77867
2    109827      77867  0        0
```

Test set

```
> confusionMatrix(t(conf_matrix),positive = "1")
Confusion Matrix and Statistics

          Actual
Pred         0     1
   0     20368  3589
   1       166  3345

               Accuracy : 0.8633
                 95% CI : (0.8592, 0.8673)
    No Information Rate : 0.7476
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.567

 Mcnemar's Test P-Value : < 2.2e-16

            Sensitivity : 0.4824
            Specificity : 0.9919
         Pos Pred Value : 0.9527
         Neg Pred Value : 0.8502
             Prevalence : 0.2524
         Detection Rate : 0.1218
   Detection Prevalence : 0.1278
      Balanced Accuracy : 0.7372

       'Positive' Class : 1
```

MODELING

LOGISTIC REGRESSION

Weighted Logistic

Create weight

```
> table(training_set$Status)

    0     1
82017 27851
> n<- length(training_set$Status)
> n0 <- table(training_set$Status)[1]
> n1 <- table(training_set$Status)[2]
> k<-2
> model_weights<- ifelse(training_set$Status == 0,n/(k*n0),n/(k*n1))
> table(model_weights)
model_weights
0.669787970786544  1.97242468852106
            82017             27851
```

Train with weight

```
> summary(log.weighted)

Call:
glm(formula = Status ~ loan_limit + Gender + approv_in_adv +
    loan_type + loan_purpose + Credit_Worthiness + open_credit +
    loan_amount + rate_of_interest + term + interest_only + lump_sum_payment +
    property_value + occupancy_type + total_units + income +
    credit_type + Credit_Score + co.applicant_credit_type + age +
    submission_of_application + Region, family = binomial(link = logit),
    data = training_set, weights = model_weights)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-3.5227  -0.7788  -0.5804   0.0027   4.0350

Coefficients:
                                Estimate Std. Error z value Pr(>|z|)
(Intercept)                    5.446e+00  3.718e-01  14.649  < 2e-16 ***
loan_limitMissing              2.112e-01  4.978e-02   4.243 2.21e-05 ***
loan_limitncf                  7.115e-01  3.177e-02  22.398  < 2e-16 ***
GenderJoint                    4.403e-02  3.170e-02   1.389 0.164934
GenderMale                     1.584e-01  2.265e-02   6.994 2.68e-12 ***
GenderSex Not Available        1.464e-02  2.898e-02   0.505 0.613569
approv_in_advnopre            -1.113e-01  9.299e-02  -1.197 0.231334
approv_in_advpre              -4.623e-01  9.490e-02  -4.871 1.11e-06 ***
loan_typetype2                 6.200e-01  2.263e-02  27.401  < 2e-16 ***
loan_typetype3                -1.217e-01  3.252e-02  -3.741 0.000183 ***
loan_purposep1                 4.428e-01  3.218e-01   1.376 0.168804
loan_purposep2                 1.162e+00  3.252e-01   3.572 0.000354 ***
loan_purposep3                 5.112e-01  3.218e-01   1.589 0.112096
loan_purposep4                 5.372e-01  3.218e-01   1.669 0.095072 .
Credit_Worthinessl2            5.272e-01  3.692e-02  14.278  < 2e-16 ***
open_creditopc                 8.452e-01  1.785e-01   4.736 2.18e-06 ***
loan_amount                    3.055e-06  1.007e-07  30.354  < 2e-16 ***
rate_of_interest              -9.390e-01  2.103e-02 -44.660  < 2e-16 ***
term                           1.218e-03  1.428e-04   8.531  < 2e-16 ***
interest_onlynot_int          -1.148e-01  3.683e-02  -3.116 0.001835 **
lump_sum_paymentnot_lpsm      -2.418e+00  5.971e-02 -40.498  < 2e-16 ***
```

Test performance

```
> confusionMatrix(t(conf_matrix2),positive = "1")
Confusion Matrix and Statistics

          Actual
Pred          0     1
   0      17997  2489
   1       2537  4445

               Accuracy : 0.817
                 95% CI : (0.8124, 0.8216)
    No Information Rate : 0.7476
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.5163

 Mcnemar's Test P-Value : 0.5074

            Sensitivity : 0.6410
            Specificity : 0.8764
         Pos Pred Value : 0.6366
         Neg Pred Value : 0.8785
             Prevalence : 0.2524
         Detection Rate : 0.1618
   Detection Prevalence : 0.2542
      Balanced Accuracy : 0.7587

       'Positive' Class : 1
```

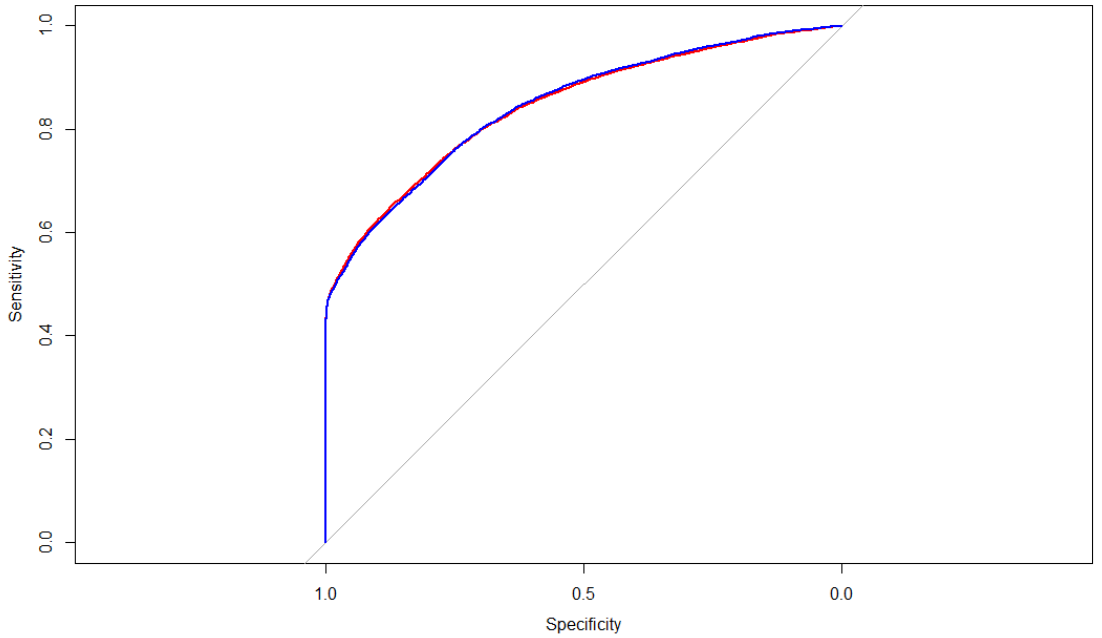
+18%

MODELING

LOGISTIC REGRESSION

Model recap

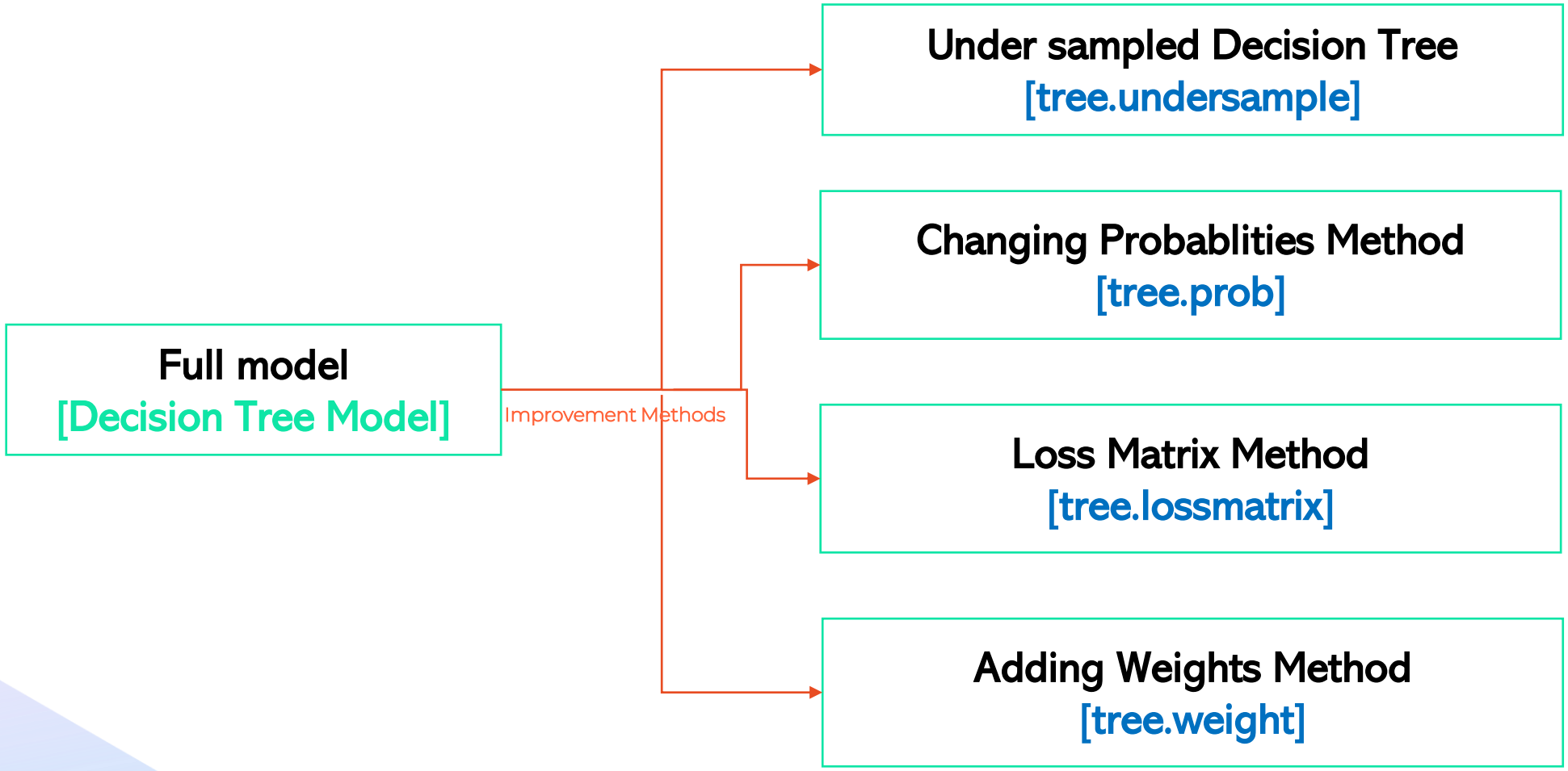
Model	Accuracy	Sensitivity	FN	F1	AUC
Log.stepwise	86.33%	48.24%	3589	0.6404	0.8453
Log.weighted	81.70%	64.10%	2489	0.6388	0.8461



MODELING

DECISION TREE

Procedure



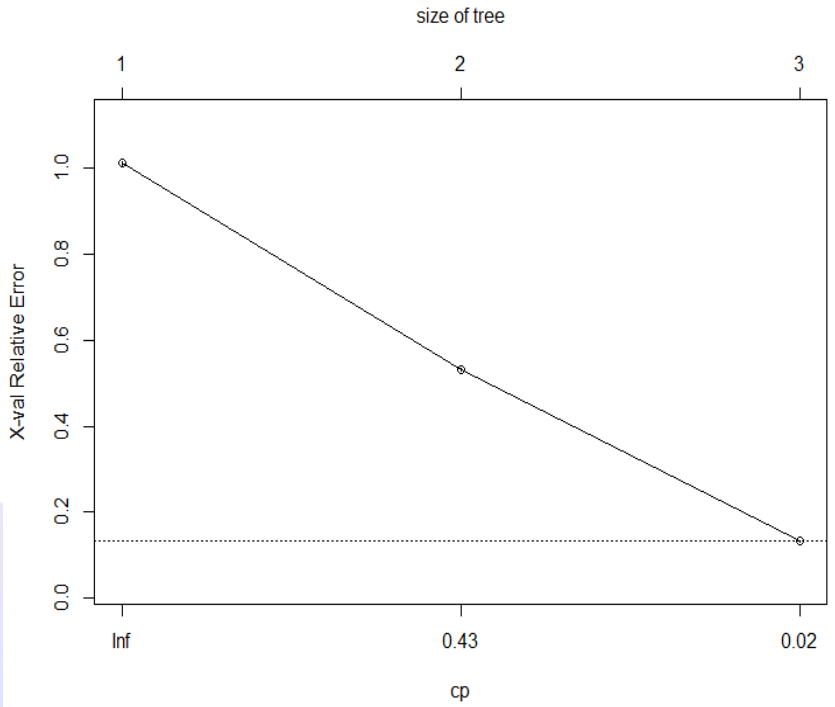
MODELING

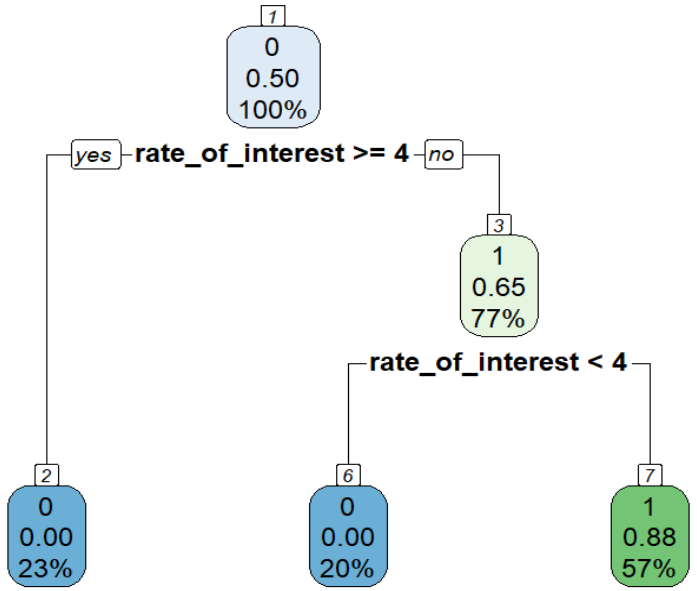
DECISION TREE

Under sampled Model

Training set

```
# Modeling fo decision tree
tree.undersample <- rpart(Status ~ ., method = "class",
                          data =  undersampled_training_set,
                          control = rpart.control(cp = 0.001))
```





Test set

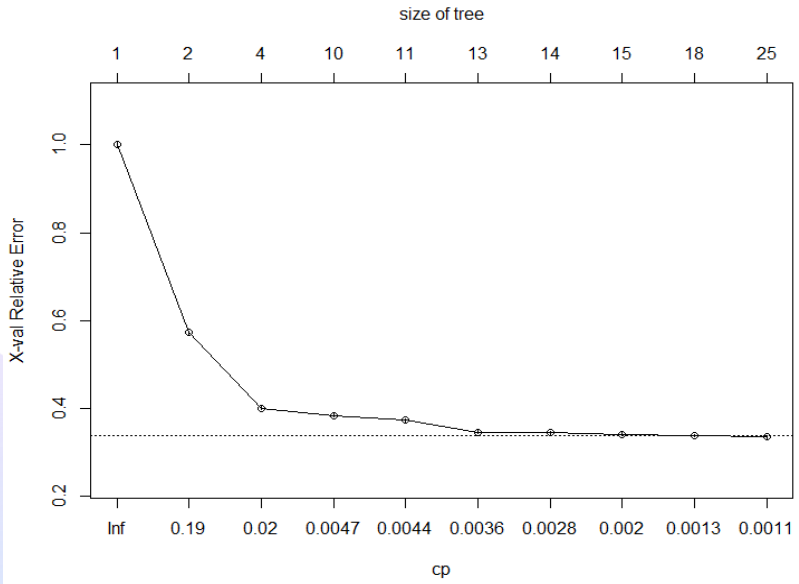
```
> confusionMatrix(t(confmat_undersample),positive = "1"
Confusion Matrix and Statistics

          Actual
Pred         0    1
   0     17754    0
   1      2780 6934

               Accuracy : 0.8988
                 95% CI : (0.8952, 0.9023)
    No Information Rate : 0.7476
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.7633

 Mcnemar's Test P-Value : < 2.2e-16

            Sensitivity : 1.0000
            Specificity : 0.8646
         Pos Pred Value : 0.7138
         Neg Pred Value : 1.0000
             Prevalence : 0.2524
         Detection Rate : 0.2524
   Detection Prevalence : 0.3536
      Balanced Accuracy : 0.9323

       'Positive' Class : 1
```

DECISION TREE

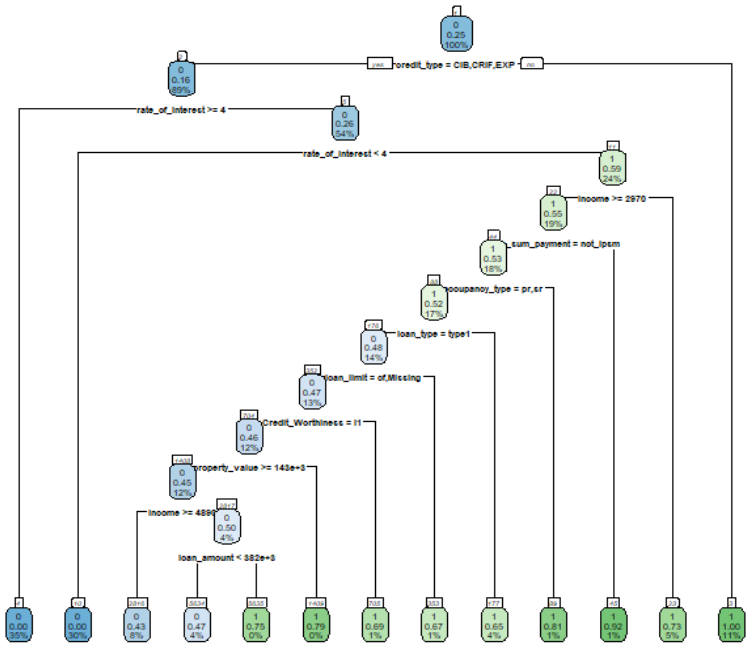
Changing Probabilities

```
# CHANGE PROBABILITY METHOD
tree.prob <- rpart(Status ~ ., method = "class",
                   data = training_set,
                   parms = list(prior = c(0.75, 0.25)),
                   control = rpart.control(cp = 0.001))
```

Probabilities: Non-default to 75% and Defaults to 25%



Training set



Test set

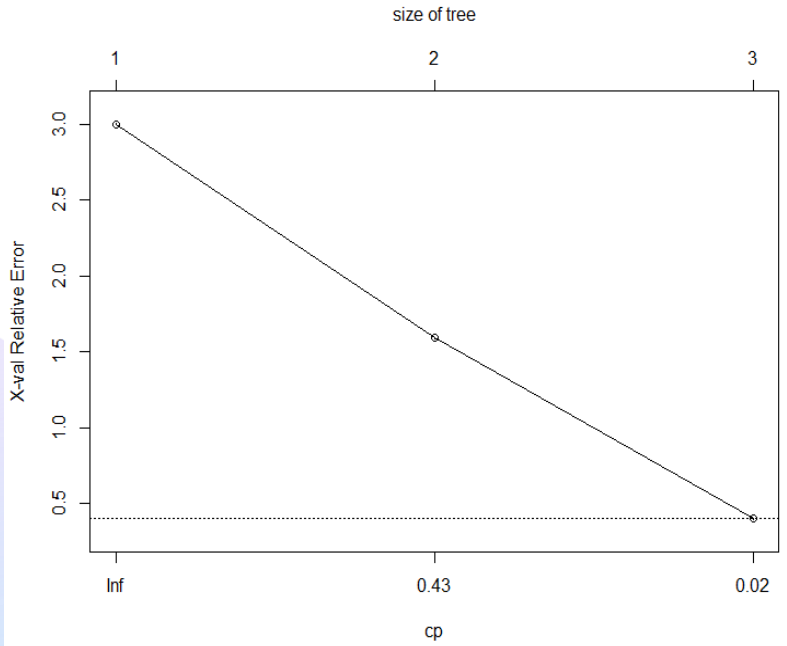
```
> confusionMatrix(t(confmat_prob),positive = "1")
Confusion Matrix and Statistics

          Actual
Pred         0    1
   0     19517 1377
   1      1017 5557

               Accuracy : 0.9128
                 95% CI : (0.9094, 0.9162)
    No Information Rate : 0.7476
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.765

 Mcnemar's Test P-Value : 2.181e-13

            Sensitivity : 0.8014
            Specificity : 0.9505
         Pos Pred Value : 0.8453
         Neg Pred Value : 0.9341
             Prevalence : 0.2524
         Detection Rate : 0.2023
   Detection Prevalence : 0.2393
      Balanced Accuracy : 0.8759

       'Positive' Class : 1
```

DECISION TREE

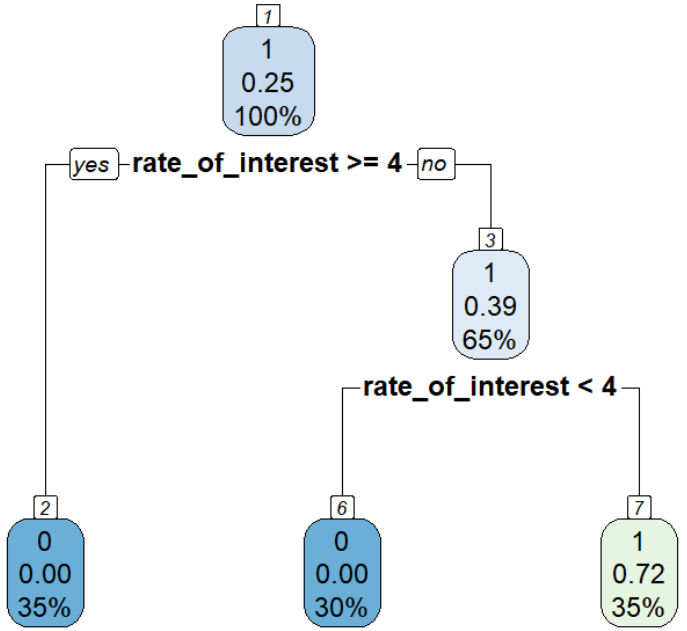
Loss Matrix Method

```
# USING LOSS MATRIX METHOD
set.seed(345)
tree.lossmatrix <- rpart(Status ~ ., method = "class",
                         data =  training_set,
                         parms = list(loss = matrix(c(0, 3, 1, 0), ncol=2)),
                         control = rpart.control(cp = 0.001))
```

Loss Matrix penalizes FN more strongly than FP



Training set



Test set

```
> confusionMatrix(t(confmat_loss_matrix),positive = "1")
Confusion Matrix and Statistics

    Actual
Pred     0     1
   0 17754     0
   1  2780  6934

               Accuracy : 0.8988
                 95% CI : (0.8952, 0.9023)
    No Information Rate : 0.7476
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.7633

 Mcnemar's Test P-Value : < 2.2e-16

            Sensitivity : 1.0000
            Specificity : 0.8646
         Pos Pred Value : 0.7138
         Neg Pred Value : 1.0000
             Prevalence : 0.2524
         Detection Rate : 0.2524
   Detection Prevalence : 0.3536
      Balanced Accuracy : 0.9323

       'Positive' Class : 1
```

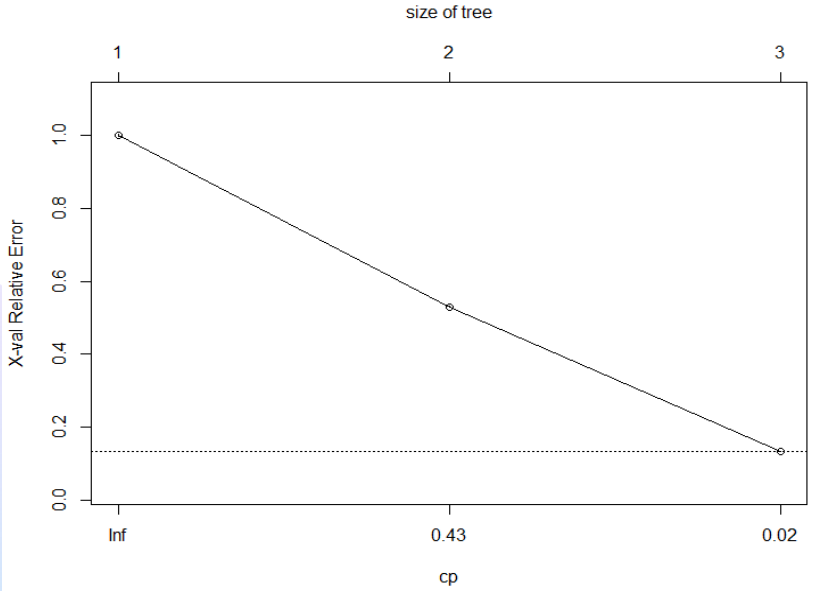
MODELING

DECISION TREE

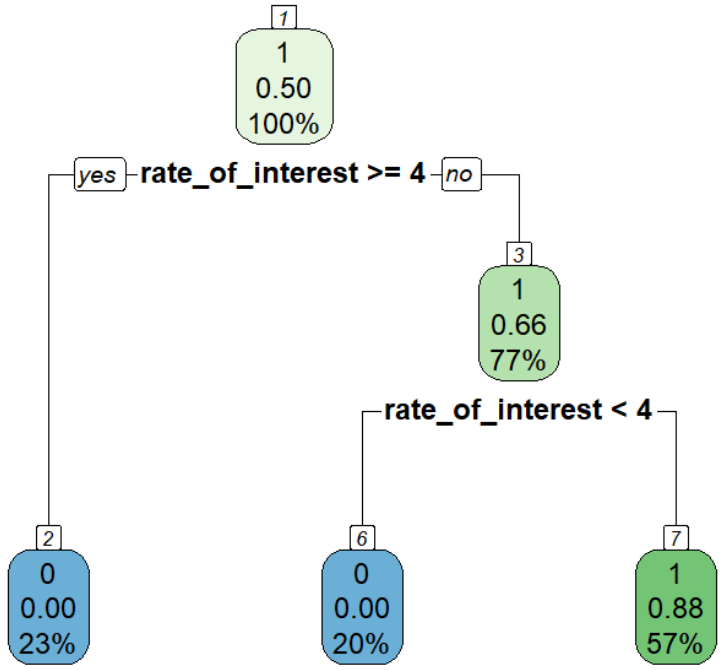
Adding Weights

```
# USING WEIGHT
# Create weight
case_weights_tree <- ifelse(training_set$Status== 1,3,1) # 3 for default, 1 for non-default
# run model
set.seed(345)
tree.weight <- rpart(Status ~ ., method = "class",
                     data = training_set,weights=case_weights_tree,
                     control = rpart.control(cp = 0.001))
```

This vector contains weights of 1 for the non-default and weights of 3 for defaults in the training set.



Training set



Test set

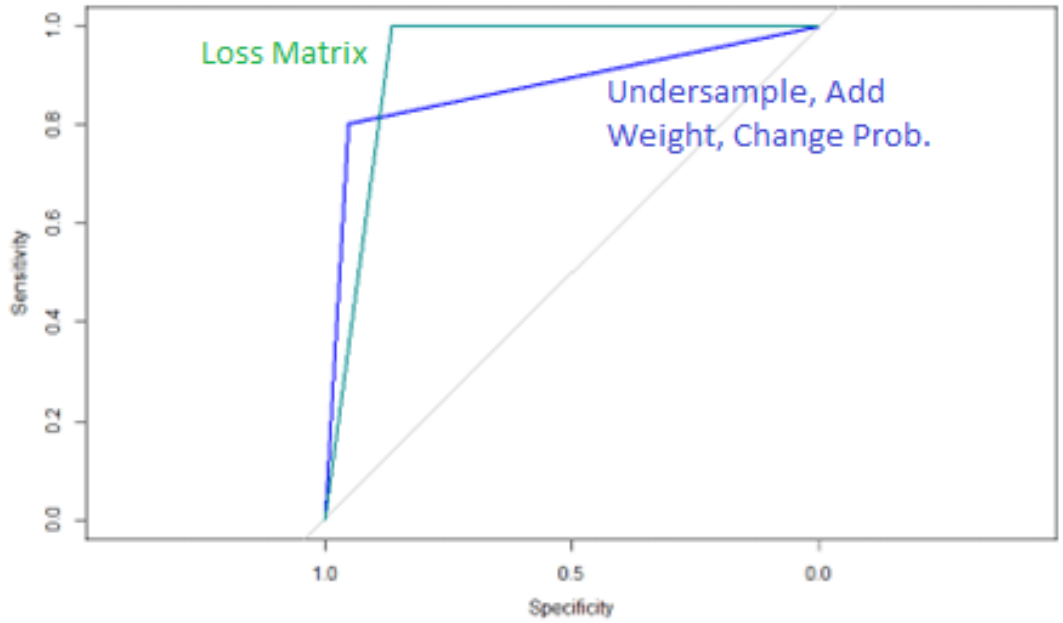
```
> confusionMatrix(t(confmat_weights),positive = "1")
Confusion Matrix and Statistics

          Actual
Pred         0     1
   0     17754     0
   1      2780  6934

               Accuracy : 0.8988
                 95% CI : (0.8952, 0.9023)
    No Information Rate : 0.7476
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.7633

 Mcnemar's Test P-Value : < 2.2e-16

            Sensitivity : 1.0000
            Specificity : 0.8646
         Pos Pred Value : 0.7138
         Neg Pred Value : 1.0000
             Prevalence : 0.2524
         Detection Rate : 0.2524
   Detection Prevalence : 0.3536
      Balanced Accuracy : 0.9323

       'Positive' Class : 1
```

MODELING

DECISION TREE

Model recap

Model	Accuracy	Sensitivity	FN	F1	AUC
Under sample	89.88%	100.00%	0	0.8330	0.9323
Change Prob	91.28%	80.14%	1377	0.8223	0.8759
Loss Matrix	89.88%	100.00%	0	0.8330	0.9323
Add Weights	89.88%	100.00%	0	0.8330	0.9323

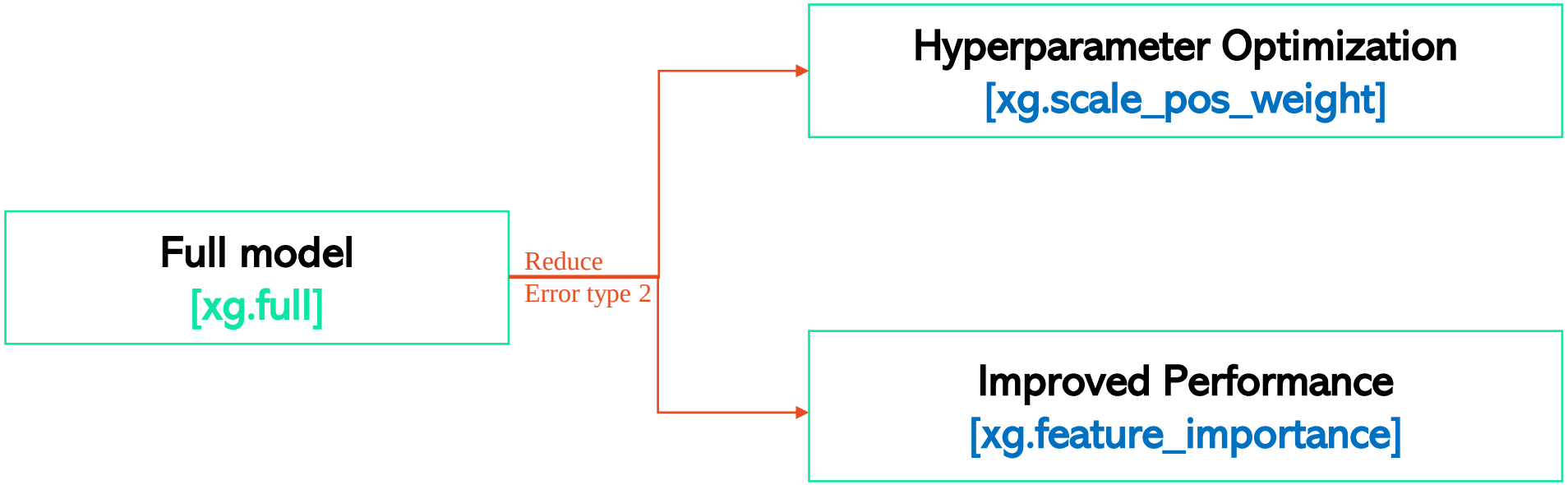


- 1st, 3rd and 4th are the same, while the 2nd model is somewhat different. Thus, in the final comparison, author may use only 2 models from the table to evaluate each method performance.
- When taking account to overall performance (AUC) and Sensitivity, Loss Matrix Method outperforms with AUC of 0.9323

MODELING

XG BOOSTING

Procedure



Northeastern University

MODELING

XG BOOSTING

Default – Full model

```
# Matrix One Hot Encoding
trainm <- sparse.model.matrix(Status ~ .-1, data = trainset)
head(trainm)

train_label <- trainset[,'Status']

train_matrix <- xgb.DMatrix(data = as.matrix(trainm),label = train_labe
```

```
> xgb_model
##### xgb.Booster
raw: 813.4 Kb
call:
  xgb.train(params = xgb_params, data = train_matrix, nrounds = 100,
    watchlist = watchlist)
params (as set within xgb.train):
  objective = "multi:softprob", eval_metric = "mlogloss", num_class = "2",
alidate_parameters = "TRUE"
xgb.attributes:
  niter
callbacks:
  cb.print.evaluation(period = print_every_n)
  cb.evaluation.log()
# of features: 42
niter: 100
nfeatures : 42
evaluation_log:
    iter train_mlogloss test_mlogloss
       1       0.494340      0.494301
       2       0.384170      0.384138
---
      99       0.114238      0.135762
     100       0.113959      0.135785
```

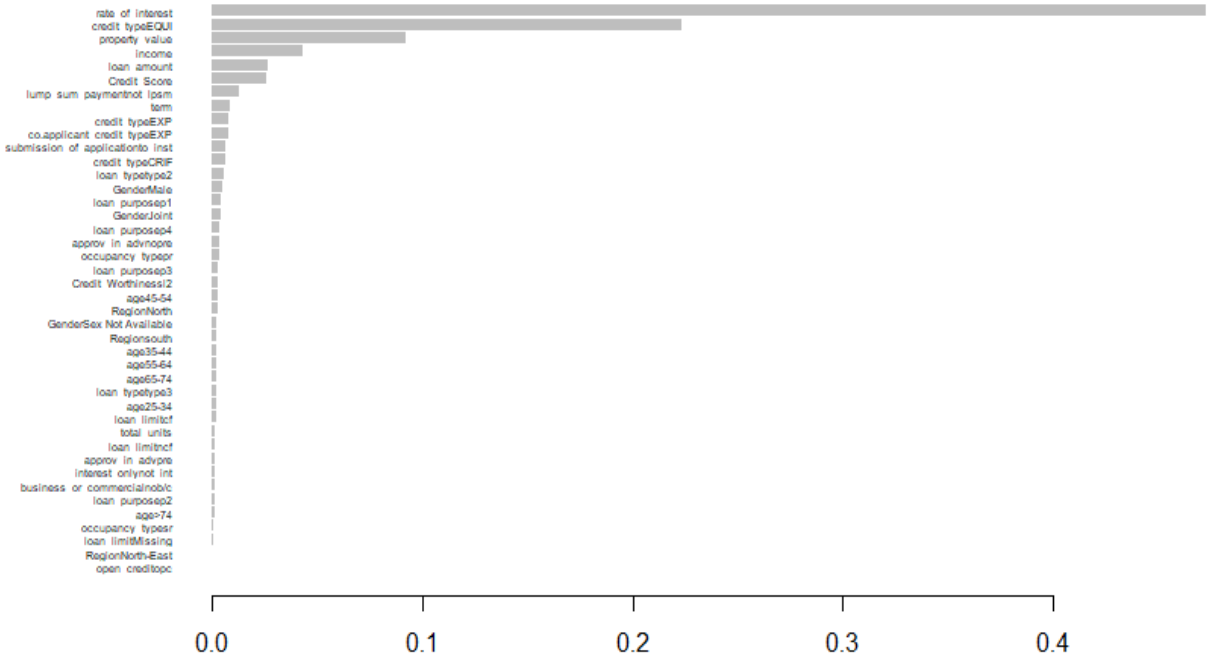
```
> err[errStest_mlogloss == min(errStest_mlogloss),]
    iter train_mlogloss test_mlogloss
66    66       0.246145       0.26591
```

```
Confusion Matrix and Statistics

          Reference
Prediction     0     1
         0 20114  2108
         1   327  4784

               Accuracy : 0.9109
                 95% CI : (0.9075, 0.9143)
    No Information Rate : 0.7479
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.7417

 Mcnemar's Test P-Value : < 2.2e-16

            Sensitivity : 0.6941
            Specificity : 0.9840
         Pos Pred Value : 0.9360
         Neg Pred Value : 0.9051
             Prevalence : 0.2521
         Detection Rate : 0.1750
   Detection Prevalence : 0.1870
      Balanced Accuracy : 0.8391

       'Positive' Class : 1
```

- ✓ One-Hot encoding performed.
- ✓ Full-Model – All features were selected.
- ✓ Sensitivity for the model is 69.4%.
- ✓ Minimal test error at Iteration 66.
- ✓ Overfitting needs to be addressed.

MODELING

XG BOOSTING



Default – Hyperparameter Tuning

```
> # Feature Importance
> feat_imp <- xgb.importance(colnames(train_matrix),model = xgb_model)
> feat_imp
                                Feature          Gain         Cover
 1:                  rate_of_interest 0.4281560671 0.0680954844
 2:                   credit_typeEQUI 0.2354519833 0.0459696883
 3:                    property_value 0.1040538091 0.0965805785
 4:                            income 0.0400439743 0.1132670056
 5:                      Credit_Score 0.0357485215 0.0839445554
 6:                       loan_amount 0.0303445304 0.1056444167
 7:          lump_sum_paymentnot_lpsm 0.0138323287 0.0300740218
 8:                     credit_typeEXP 0.0091163524 0.0126709181
 9:                               term 0.0090800566 0.0322754752
10:                    credit_typeCRIF 0.0072846171 0.0121248303
11: submission_of_applicationto_inst 0.0066846194 0.0362462541
12:       co.applicant_credit_typeEXP 0.0065434217 0.0232617458
```

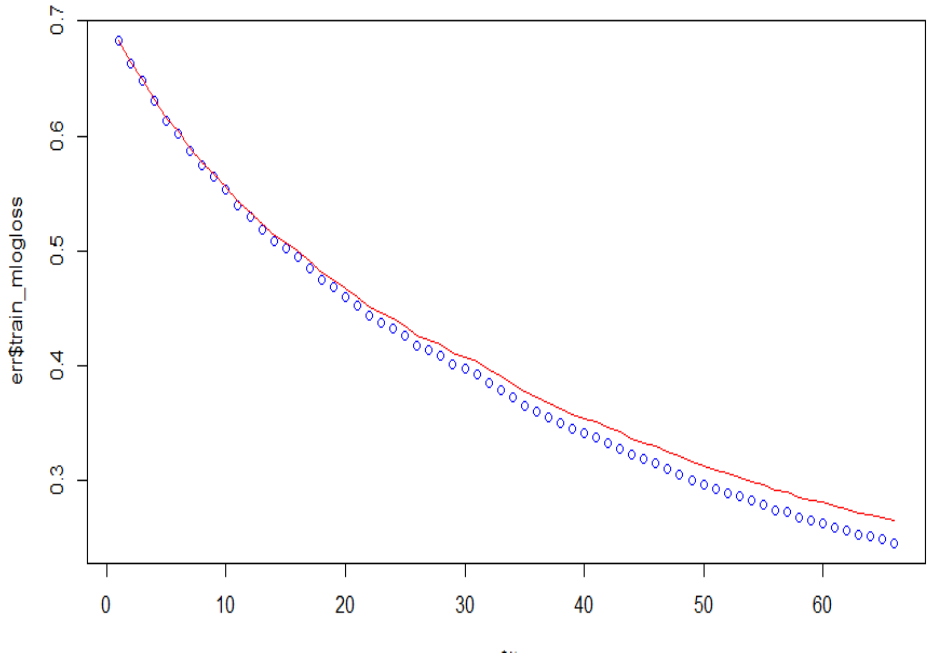
- ✓ Feature Importance – Rate Of Interest, Credit Type, Property value, Income.
- ✓ Hyperparameter tuning – eta, max_depth.

```
> xgb_model
##### xgb.Booster
raw: 18.1 Mb
call:
  xgb.train(params = xgb_params, data = train_matrix, nrounds = 66,
    watchlist = watchlist, eta = 0.025, max_depth = 15, subsample = 0.5,
    colsample_bytree = 0.5, set.seed = 567)
params (as set within xgb.train):
```

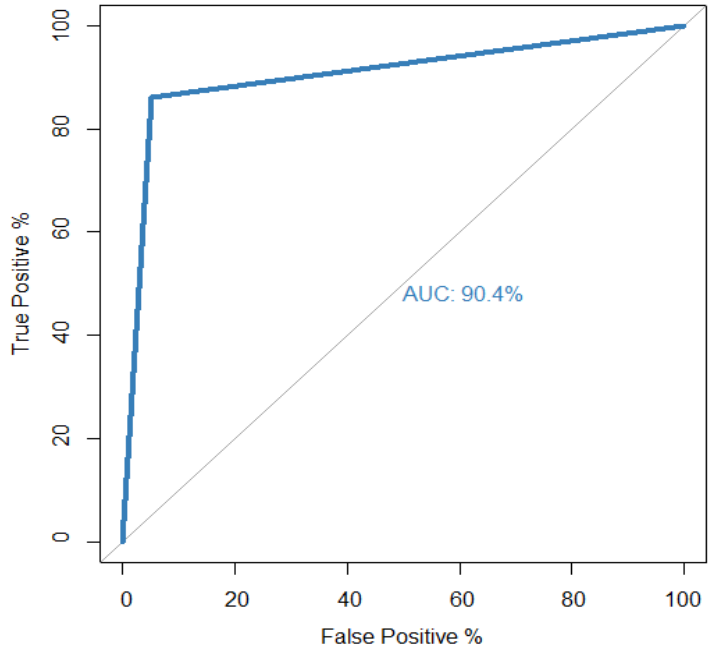
MODELING

XG BOOSTING

Default – Hyperparameter Tuning




```
> caret::confusionMatrix(factor(pred$max_prob
 '1')
Confusion Matrix and Statistics

          Reference
Prediction     0     1
         0 19388   962
         1  1053  5930

               Accuracy : 0.9263
                 95% CI : (0.9231, 0.9294)
    No Information Rate : 0.7479
    P-Value [Acc > NIR] : < 2e-16

                  Kappa : 0.8054

 Mcnemar's Test P-Value : 0.04497

            Sensitivity : 0.8604
            Specificity : 0.9485
         Pos Pred Value : 0.8492
         Neg Pred Value : 0.9527
             Prevalence : 0.2521
         Detection Rate : 0.2170
   Detection Prevalence : 0.2555
      Balanced Accuracy : 0.9045

       'Positive' Class : 1

>
```

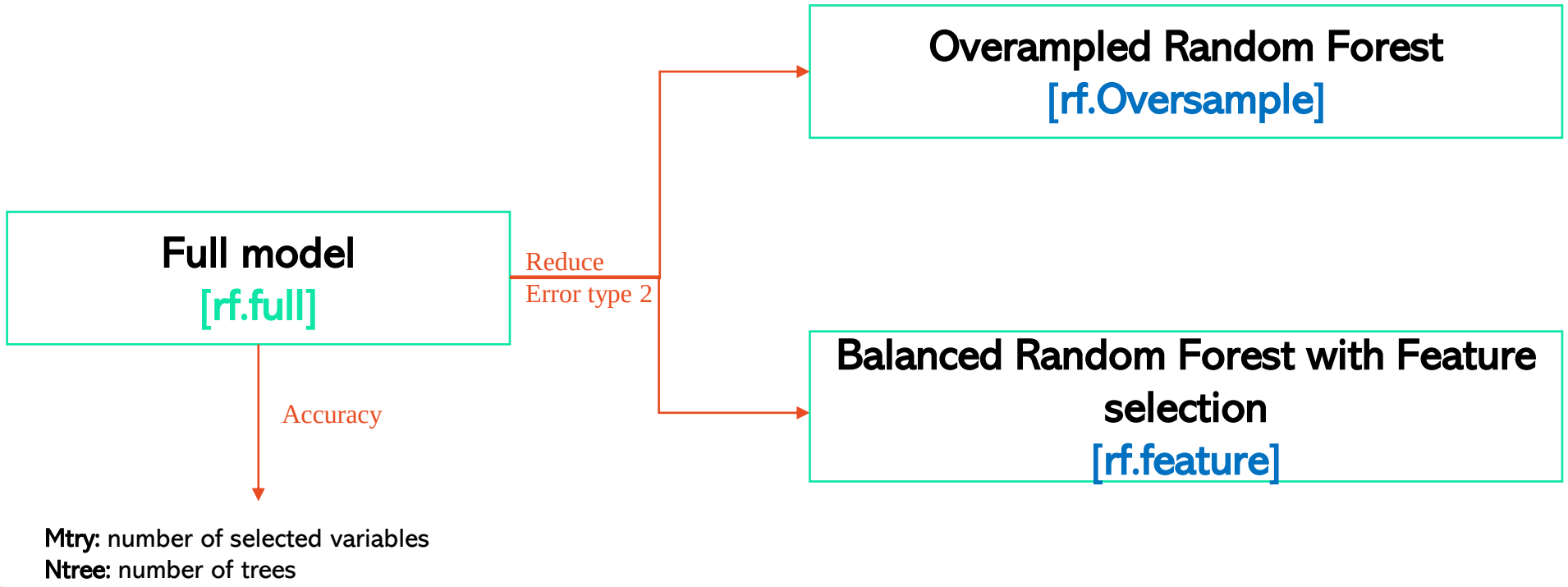


- ✓ Comparing the Train error and test error.
- ✓ Model's accuracy is 92.63 percent
- ✓ AUC is 90.4 percent.

MODELING

RANDOM FOREST

Procedure



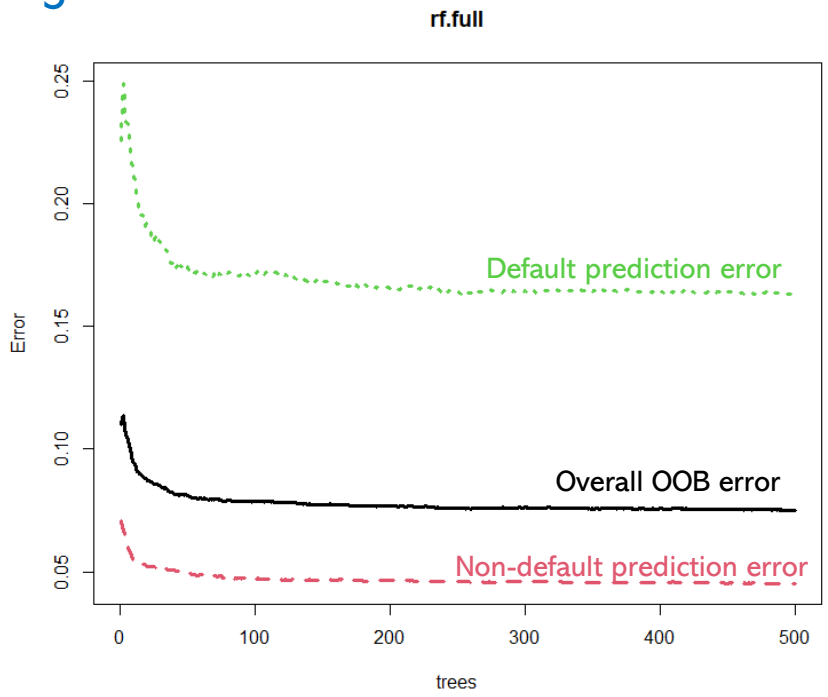
MODELING

RANDOM FOREST

Default – Full model

Training set

```
> rf.full <- randomForest(Status~.,
+                         data = training_set)
> rf.full

Call:
 randomForest(formula = Status ~ ., data = training_set)
               Type of random forest: classification
                     Number of trees: 500
No. of variables tried at each split: 4

        OOB estimate of  error rate: 7.51%
Confusion matrix:
      0     1 class.error
0 78309  3708  0.04521014
1  4540 23311  0.16301030
```



Test set

```
> confusionMatrix(t(rf.full.matrix),positive = "1")
Confusion Matrix and Statistics

    Actual
Pred     0     1
   0 19538  1141
   1   996  5793

               Accuracy : 0.9222
                 95% CI : (0.919, 0.9253)
    No Information Rate : 0.7476
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.7924

 Mcnemar's Test P-Value : 0.001839

            Sensitivity : 0.8354
            Specificity : 0.9515
         Pos Pred Value : 0.8533
         Neg Pred Value : 0.9448
             Prevalence : 0.2524
         Detection Rate : 0.2109
   Detection Prevalence : 0.2472
      Balanced Accuracy : 0.8935

       'Positive' Class : 1
```

FN is **4 times** higher FP

MODELING

RANDOM FOREST

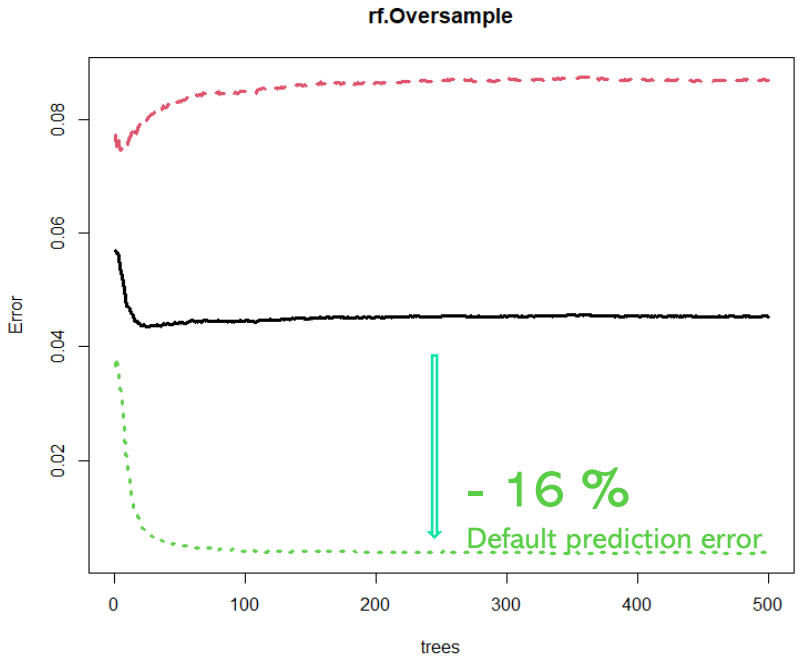
Oversampled model

Oversample: to balance the set

```
> table(oversampled_training_set$Status)

    0     1
82017 81997
```

Training set

```
> rf.Oversample = randomForest(Status~., data = oversampled_training_set,
+                              ntree = 500,
+                              mtry = 5)
> rf.Oversample

Call:
 randomForest(formula = Status ~ ., data = oversampled_training_set,
 ntree = 500, mtry = 5)
               Type of random forest: classification
                     Number of trees: 500
No. of variables tried at each split: 5

        OOB estimate of  error rate: 4.53%
Confusion matrix:
      0     1 class.error
0 74884  7133 0.086969775
1   304 81693 0.003707453
```



Test set

```
> confusionMatrix(t(rf.Oversample.matrix),p
Confusion Matrix and Statistics

                    Actual
rf.Oversample.pred.test     0     1
                      0 18701   440
                      1  1833  6494

               Accuracy : 0.9172
                 95% CI : (0.9139, 0.9205)
    No Information Rate : 0.7476
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.7944

 Mcnemar's Test P-Value : < 2.2e-16

            Sensitivity : 0.9365
            Specificity : 0.9107
         Pos Pred Value : 0.7799
         Neg Pred Value : 0.9770
             Prevalence : 0.2524
         Detection Rate : 0.2364
   Detection Prevalence : 0.3032
      Balanced Accuracy : 0.9236

       'Positive' Class : 1
```

+10%

MODELING

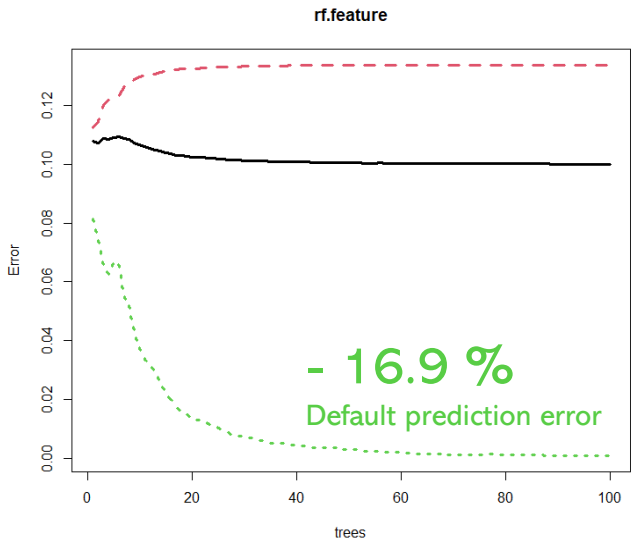
RANDOM FOREST

Balanced model with feature selection

Training set

```
# Run model
set.seed(300)
rf.feature = randomForest(Status~credit_type + rate_of_interest+property_value+
    lump_sum_payment+income+co.applicant_credit_type+
    submission_of_application+ loan_type +
    business_or_commercial+Gender,
  data = training_set,
  ntree = 100,
  mtry = 4,
  strata = training_set$Status,
  sampsize = rep(sum(training_set$Status==1),2))
```

```
> rf.feature

Call:
 randomForest(formula = Status ~ credit_type + rate_of_inter
est +      property_value + lump_sum_payment + income + co.a
pplicant_credit_type +      submission_of_application + loan
_type + business_or_commercial +      Gender, data = trainin
g_set, ntree = 100, mtry = 4, strata = training_set$Status,
     sampsize = rep(sum(training_set$Status == 1), 2))
               Type of random forest: classification
                     Number of trees: 100
No. of variables tried at each split: 4

        OOB estimate of  error rate: 10%
Confusion matrix:
      0     1  class.error
0 71055 10962 0.1336552178
1    26 27825 0.0009335392
```



Test set

```
> confusionMatrix(t(rf.feature.matrix),posi
Confusion Matrix and Statistics

                    Actual
rf.feature.pred.test     0     1
                   0 17756     0
                   1  2778  6934

               Accuracy : 0.8989
                 95% CI : (0.8952, 0.9024)
    No Information Rate : 0.7476
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.7634

 Mcnemar's Test P-Value : < 2.2e-16

            Sensitivity : 1.0000
            Specificity : 0.8647
         Pos Pred Value : 0.7140
         Neg Pred Value : 1.0000
             Prevalence : 0.2524
         Detection Rate : 0.2524
   Detection Prevalence : 0.3536
      Balanced Accuracy : 0.9324

       'Positive' Class : 1
```

+17%

Feature selection: only powerful predictors
-> more precise

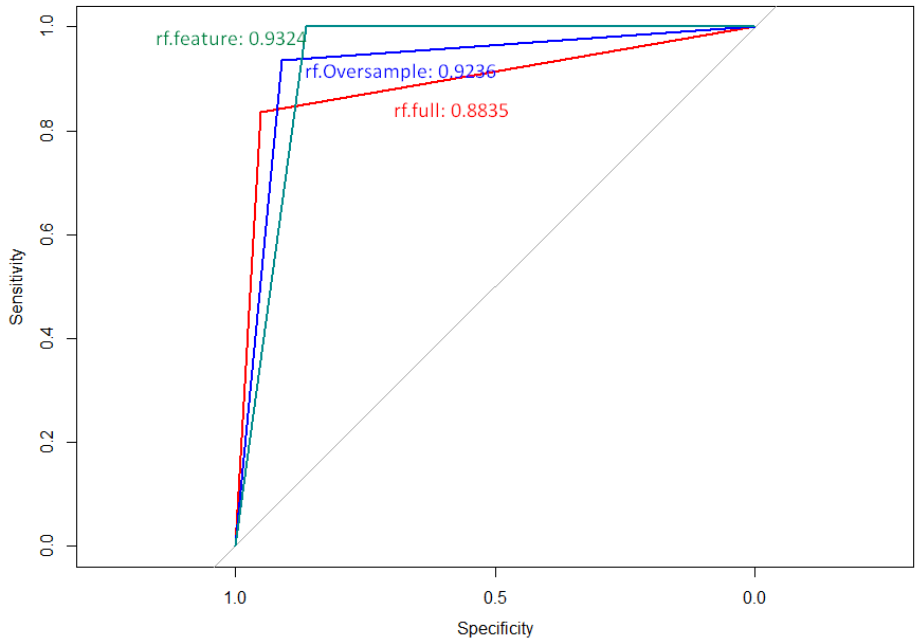
Strata: Used for stratified sampling. **Sampsize**: Sample size.
-> enable the balanced bootstrapping

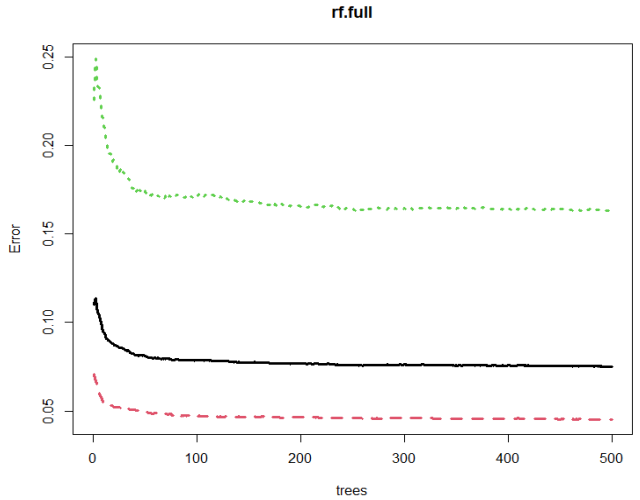
MODELING

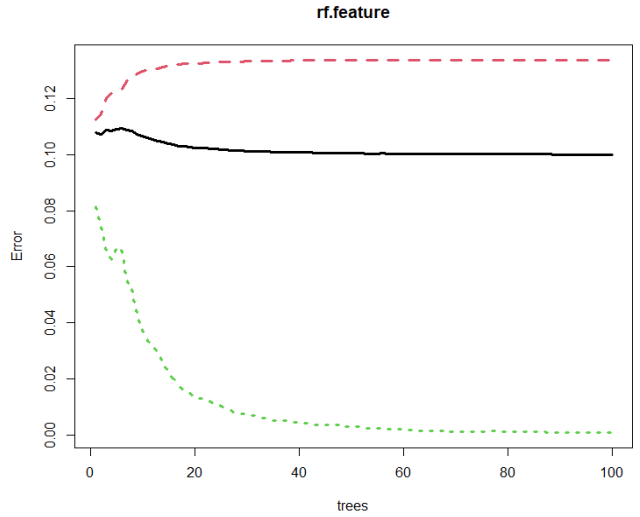
RANDOM FOREST

Model recap

Model	Accuracy	Sensitivity	FN	F1	AUC
Rf.full	92.22%	83.54%	1141	0.8442	0.8935
Rf.Oversample	91.72%	93.65%	440	0.8510	0.9236
Rf.feature	89.89%	100.00%	0	0.8331	0.9324



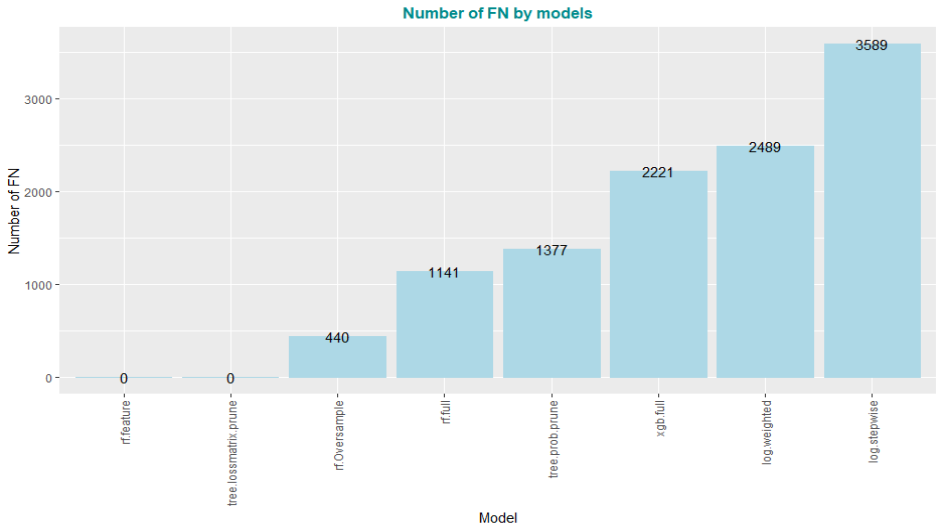




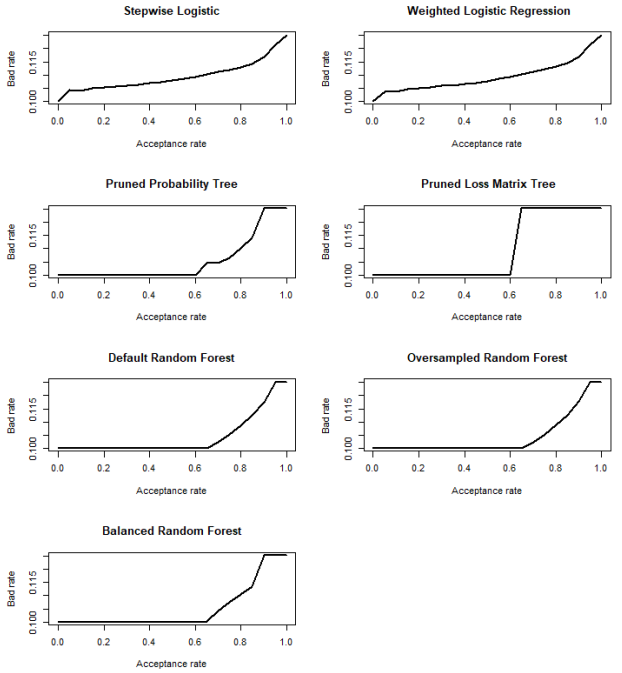
COMPARISON - SELECTION

Selection by Model Performance metrics

Type	Model	Accuracy	Sensitivity	FN	F1	AUC
Logistic Regression	log.stepwise	86.33%	48.24%	3589	0.640	0.845
Logistic Regression	log.weighted	81.70%	64.10%	2489	0.639	0.846
Decision Tree	tree.prob.prune	91.28%	80.14%	1377	0.822	0.876
Decision Tree	tree.lossmatrix.prune	89.88%	**100.00%**	**0**	0.833	**0.932**
XGBoost	xgb.full	91.09%	69.41%	2108	0.850	0.839
XGBoost	xgb.feature	92.63%	86.04%	962	0.874	0.904
Random Forest	rf.full	**92.22%**	83.54%	1141	0.844	0.894
Random Forest	rf.Oversample	91.72%	93.65%	440	**0.851**	0.924
Random Forest	rf.feature	89.89%	**100.00%**	**0**	0.833	**0.932**



Selection by Domain Knowledge: Bad rate strategy




```
> Model_Selection(0.9)
  accept_rate bad_rate                Model
1         0.9   0.1169          log.stepwise
2         0.9   0.1169          log.weighted
3         0.9   0.1176               rf.full
4         0.9   0.1178         rf.Oversample
5         0.9   0.1252         tree.prob.prune
6         0.9   0.1252 tree.lossmatrix.prune
7         0.9   0.1252           rf.feature
>
> Model_Selection(0.8)
  accept_rate bad_rate                Model
1         0.8   0.1087               rf.full
2         0.8   0.1089         rf.Oversample
3         0.8   0.1100         tree.prob.prune
4         0.8   0.1104           rf.feature
5         0.8   0.1130          log.stepwise
6         0.8   0.1132          log.weighted
7         0.8   0.1252 tree.lossmatrix.prune
>
> Model_Selection(0.5)
  accept_rate bad_rate                Model
1        0.45   0.1000         tree.prob.prune
2        0.45   0.1000 tree.lossmatrix.prune
3        0.45   0.1000         rf.Oversample
4        0.45   0.1000           rf.feature
5        0.45   0.1000               rf.full
6        0.45   0.1071          log.weighted
7        0.45   0.1074          log.stepwise
>
```

CONCLUSION

1. **Accuracy** is not always the prioritized metrics

2. **F1** score is also **misleading**, the right metrics is **Sensitivity**

3. Consider **various solutions** when dealing with Imbalanced data set

Data process	Modeling	Threshold
• Upsample • Downsample • SMOTE • ROSE	• Tune hyperparameters • Update model • Pruning	• Cut-off • Cp

4. Beside metrics, **domain knowledge** is also another imperative choice when comparing models

5. For this data set, overall, **Random Forest** turns out to be the most powerful models as it uses boosting methods

LOAN DEFAULT

Group Project - OUTLIERS

THANK YOU