Taiwan: Customer Defaults

Table of Contents

Project Note III

1. Problem Statement

A Taiwan-based credit card issuer to know about the potential customers who has been holding credit card relationship with bank, as well as identify the key drivers that determine this likelihood. This would inform the issuer's decisions on who to give a credit card to and what credit limit to provide. It would also help the issuer have a better understanding of their current and potential customers, which would inform their future strategy, including their planning of offering targeted credit products to their customers.

2. Recap from previous notes

- In the data set earlier we found it consists 30000 observations with 25 variables.
- The categorical data value like Sex , marital status and education has changed to numeric value .
- We have realized that 22.1 % percent defaulter and 77.9% are not default cases
- Default category whereas male customer is 9.6% and female category shows 12.5% leaning to defaulted
- University level - graduate or PG is more into default side
- Married customers somehow leaning to tend defaulter
- Average age of 25 to 30 is the highest risk .
- We have also checked the multicollinearity problems is existed in the data set , pay status categorical variables are dependent on each other and impact of REPAY_ SEP to REPAY_APR variables to default. Payment DEFAULT is high.
- We have also created some dummy variables like ratio of the payment for each month SEP to APR and balance amount month wise from SEP to APR .
- We have added some new variables like payment ratio , timely payment and found in September22%, August 14%, July 14%, June 11%, May 09 and April it is 10% customer are paid on time
- Performed FA and created final data set with new variables (enclosed train data set below)
- Performed data split with balancing of the samples in test and train data set

```
str(train_bank)

## 'data.frame':    9291 obs. of  14 variables:
##  $ LIMIT_BAL      : num  20000 450000 100000 30000 20000 20000 270000 50000 1
20000 200000 ...
##  $ SEX            : int  1 1 1 2 2 1 2 1 1 1 ...
##  $ EDUCATION      : int  2 1 2 2 2 2 2 3 2 2 ...
##  $ MARRIAGE       : int  2 1 1 2 1 1 2 1 2 1 ...
##  $ AGE            : int  33 45 30 22 24 31 26 53 28 32 ...
##  $ DEFAULT        : int  0 1 1 1 1 1 1 0 0 1 ...
##  $ BILLED._AMT    : num  -0.391 -0.492 0.455 -0.208 -0.647 ...
##  $ REPAY_STATUS.  : num  0.1827 0.3363 0.0201 -0.2731 2.5181 ...
##  $ PAID_AMT       : num  -0.7188 1.278 -0.0873 -0.4301 -0.5607 ...
##  $ TIMELY_PAID_AMT: num  0.543 0.523 0.431 -1.422 -1.695 ...
##  $ RATIO_PADI_AMT1: num  0.419 -0.546 -0.411 -0.44 -0.027 ...
##  $ RATIO_PADI_AMT2: num  0.37743 -2.66432 0.00851 -0.35402 -0.03372 ...
##  $ RATIO_PADI_AMT3: num  0.3778 1.6226 -0.1 -0.0637 0.1059 ...
##  $ RATIO_PADI_AMT4: num  0.62 0.416 0.325 0.922 0.141 ...
```

As per our the final data set of note II after doing FA ,we have been considered above variables for the final model building where we got 9291 observations and 14 variables .

3. Model Planning and Building

We are planning for three models as follows:

1. Logistic Regression Tree: A Prediction Model to predict the defaulters, and in turn understand the features with combination of other variable having played a significant role
2. Classification and Regression Tree
3. Random Forest: Another Prediction Model to predict the defaulters
4. We will also use these machine learning patterns
   - KNN
   - Naive Bayes
   - Bagging and boosting modeling procedures to create

4. Logistic Regression

Simple logit model on all variables

```
## Call:
## glm(formula = DEFAULT ~ ., family = binomial(link = "logit"),
##     data = train_bank)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.5842  -0.9796   0.2938   0.9860   2.4416
##
## Coefficients:
##                   Estimate Std. Error z value Pr(>|z|)
## (Intercept)      1.891e-01  1.828e-01   1.034 0.301045
## LIMIT_BAL       -8.208e-07  2.381e-07  -3.448 0.000566 ***
## SEX             -1.092e-01  4.743e-02  -2.303 0.021266 *
## EDUCATION       -3.273e-02  3.225e-02  -1.015 0.310172
## MARRIAGE        -1.460e-01  4.868e-02  -2.999 0.002706 **
## AGE              5.595e-03  2.780e-03   2.013 0.044138 *
## BILLED._AMT     -3.608e-02  2.500e-02  -1.443 0.148979
## REPAY_STATUS.    4.129e-01  2.412e-02  17.122  < 2e-16 ***
## PAID_AMT        -4.515e-01  3.196e-02 -14.126  < 2e-16 ***
## TIMELY_PAID_AMT -6.899e-01  2.354e-02 -29.306  < 2e-16 ***
## RATIO_PADI_AMT1 -9.910e-02  2.796e-02  -3.544 0.000394 ***
## RATIO_PADI_AMT2 -8.048e-02  2.608e-02  -3.086 0.002026 **
## RATIO_PADI_AMT3  8.423e-03  2.774e-02   0.304 0.761400
## RATIO_PADI_AMT4 -1.915e-01  2.616e-02  -7.322 2.44e-13 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 12880  on 9290  degrees of freedom
## Residual deviance: 10973  on 9277  degrees of freedom
## AIC: 11001
##
## Number of Fisher Scoring iterations: 4
```

Interpretation:

Following Features are having significant effect on making one defaulter:

1. REPAY_STATUS- Positive
2. PAID_AMT – Negative
3. TIMELY_PAID_AMT – Negative
4. RATIO_PADI_AMT1 – Negative
5. AGE & SEX – Positive
6. MARRIEGE – Postive

Let's remove the insignificant Variables and refine the initial built model.

The refined Regression Model

```
## Call:
## glm(formula = DEFAULT ~ ., family = binomial(link = "logit"),
##     data = logit_f1)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.1535  -0.6108  -0.5145  -0.3320   2.8482
##
## Coefficients:
##                   Estimate Std. Error z value Pr(>|z|)
## (Intercept)     -1.346e+00  1.516e-01  -8.881  < 2e-16 ***
## LIMIT_BAL       -1.199e-06  2.770e-07  -4.329 1.50e-05 ***
## SEX             -1.252e-01  5.754e-02  -2.176  0.02955 *
## AGE              7.655e-03  3.023e-03   2.532  0.01134 *
## REPAY_STATUS.    4.632e-01  2.707e-02  17.112  < 2e-16 ***
## PAID_AMT        -3.757e-01  4.169e-02  -9.011  < 2e-16 ***
## TIMELY_PAID_AMT -7.051e-01  2.598e-02 -27.136  < 2e-16 ***
## RATIO_PADI_AMT1 -9.419e-02  3.626e-02  -2.598  0.00939 **
## RATIO_PADI_AMT4 -1.508e-01  3.003e-02  -5.021 5.13e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 9504.6  on 8999  degrees of freedom
## Residual deviance: 7997.2  on 8991  degrees of freedom
## AIC: 8015.2
##
## Number of Fisher Scoring iterations: 5
```

Interpretation:

Same results as seen in initial Model build, and confirms that besides REPAY_STATUS all variables are has Negative Impact
In other words, we can see anti-incumbency implication

Check Multi-Collinearity Effect:

```
##       LIMIT_BAL            SEX            AGE   REPAY_STATUS.
##        1.256936       1.010270       1.031040       1.111527
##        PAID_AMT TIMELY_PAID_AMT RATIO_PADI_AMT1 RATIO_PADI_AMT4
##        1.096350       1.018324       1.038484       1.024806
```

No variables are having value more than five hence , there is no multi-collinearity but As can be seen, VIF is just slightly greater than 1, hence we can
Conclude that our variables are moderately correlated.

Performance Measures

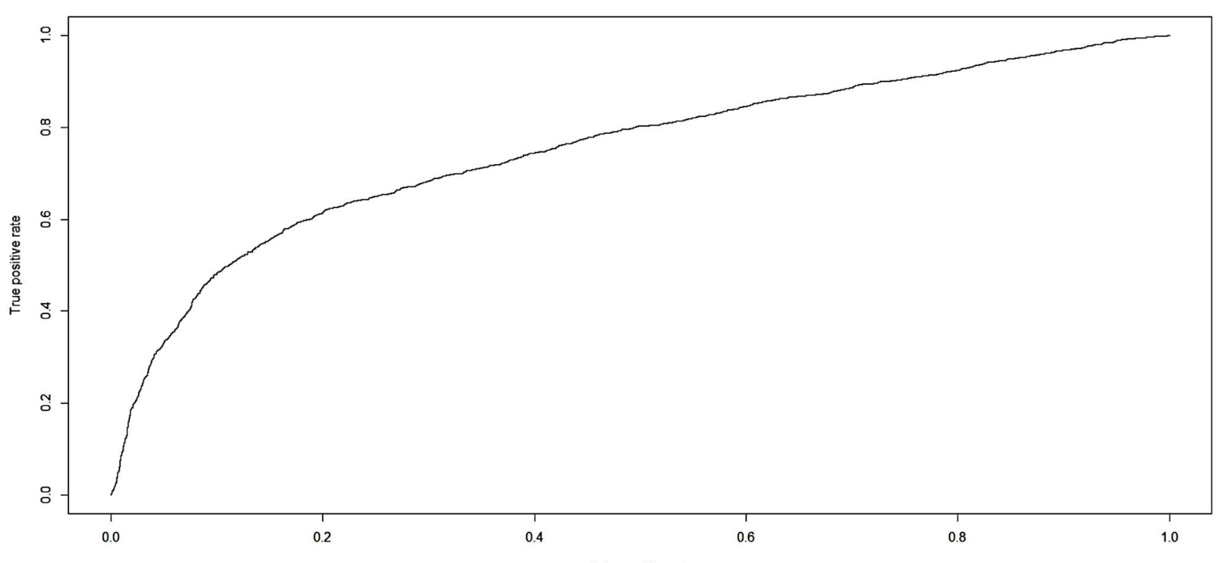
The following model performance measures will be calculated on entire data set to gauge the goodness of the model:

- KS Area Under Curve (AUC)
- Gini Coefficient
- Classification Error

Accuracy =(6672+646)/(6672+646+1342+340)= 82%
Classification Error Rate = 1- Accuracy = 18%
The lower the classification error rate, higher the model accuracy, resulting in a better model.

Measure	Values
KS	41%
AUC	75%
GINI	14%
Classification Error	18%

```
The AUC value around 75% indicates the good performance of the model.
Graphical representation of the Area Under Curve is as follows:
```



5. **Classification and Regression Tree**

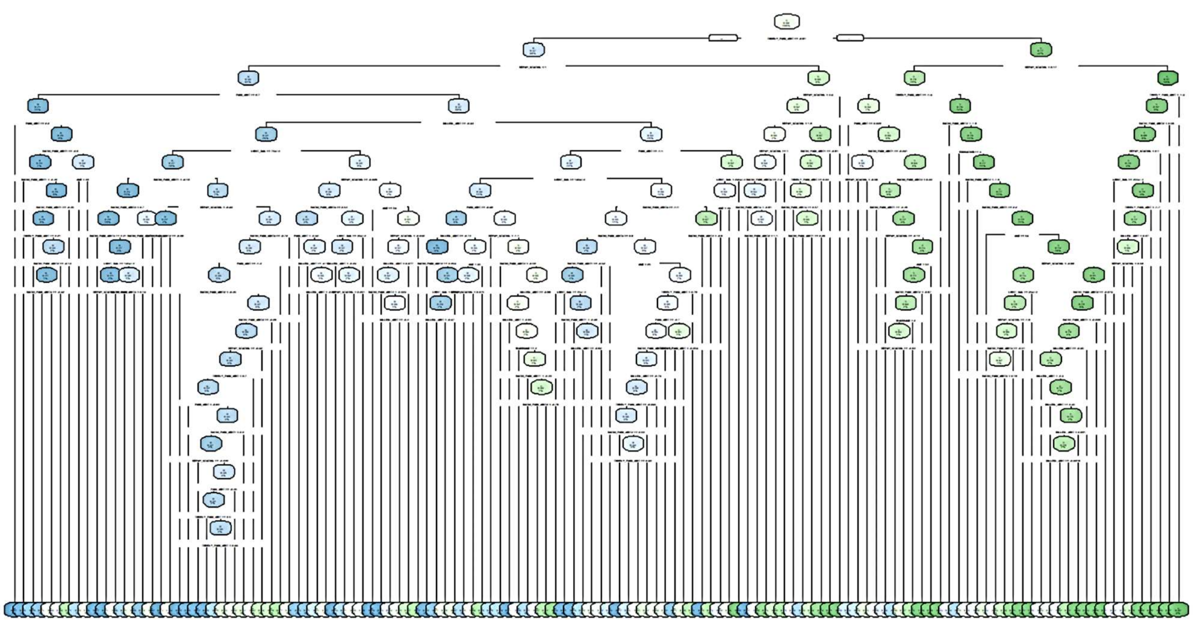
The initial CART Model is built by setting up Control Parameters as follows:

```
r.ctrl = rpart.control(minsplit = 100, minbucket = 10, cp = 0, xval = 10)
CT_model = rpart(DEFAULT ~ ., data = train_bank, method = "class", control = r.ctrl
)
CT_model
```

```
## n= 9291
##
## node), split, n, loss, yval, (yprob)
##       * denotes terminal node
##
##      1) root 9291 4643 1 (0.49973092 0.50026908)
##        2) TIMELY_PAID_AMT>=-0.6144733 6272 2319 0 (0.63026148 0.36973852)
##          4) REPAY_STATUS.< 1.000148 5686 1940 0 (0.65881112 0.34118888)
##            8) PAID_AMT>=0.7048767 1012  216 0 (0.78656126 0.21343874)
##             16) PAID_AMT>=2.234394 228   17 0 (0.92543860 0.07456140) *
##             17) PAID_AMT< 2.234394 784  199 0 (0.74617347 0.25382653)
##               34) RATIO_PADI_AMT1>=-0.8005472 671  155 0 (0.76900149 0.23099851)
##                 68) RATIO_PADI_AMT1< -0.3804398 186   27 0 (0.85483871 0.1451612
9) *
*
```

```
rpart.plot(CT_model)
```

```
## Warning: labs do not fit even at cex 0.15, there may be some overplotting
```

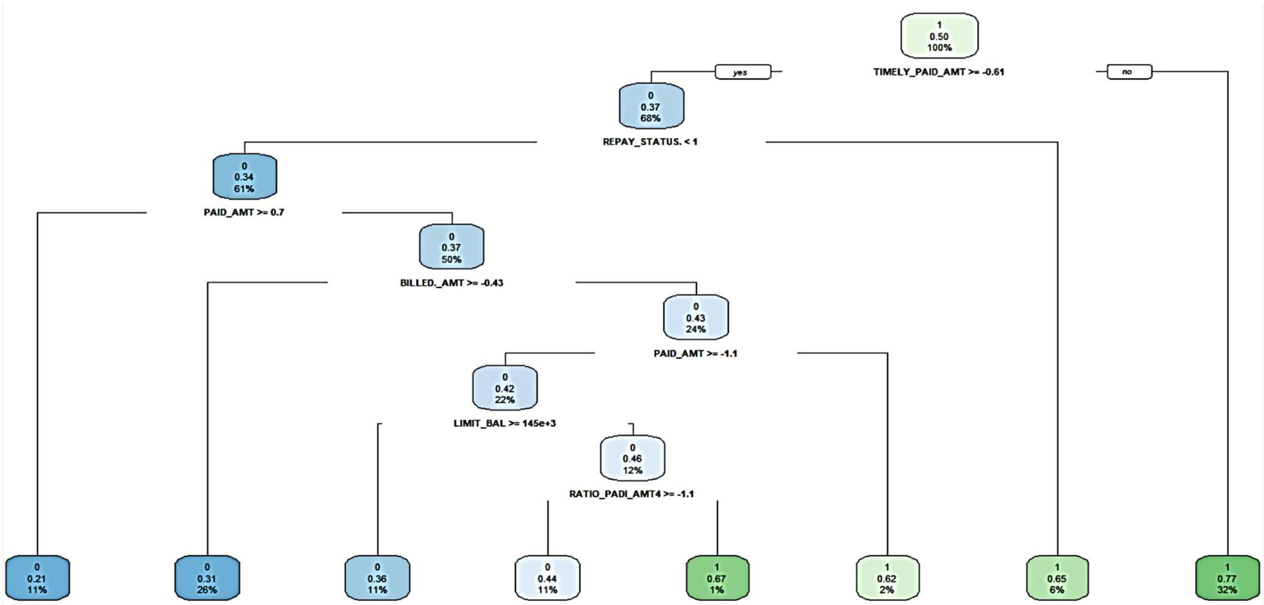


```
CT_model$cptable
```

```
##               CP nsplit rel error    xerror        xstd
## 1  0.3519276330      0 1.0000000 1.0230454 0.010377498
```

```
## 2   0.0370450140        1 0.6480724 0.6500108 0.009722263
## 3   0.0031588772        2 0.6110274 0.6164118 0.009584658
## 4   0.0026922249        7 0.5942279 0.6172733 0.009588369
```

Pruned Tree:



Interpretation:

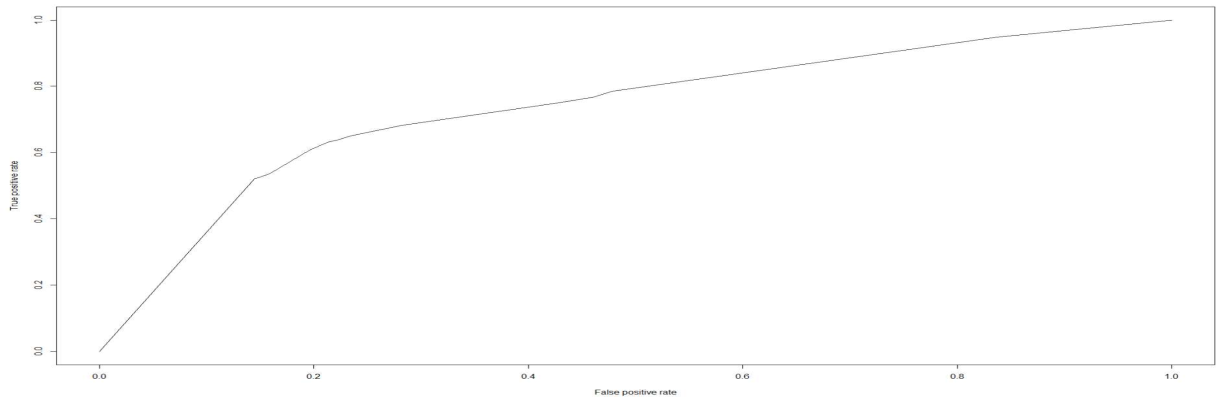
The Pruned Tree is using only seven Variables. TIMELY_PAID_AMT, REPAY_STATUS , PAID_AMT, BILLED._AMT , LIMIT_BAL, RATIO_PADI_AMT1

Variables Importance:

```
CT_model$variable.importance

## TIMELY_PAID_AMT   REPAY_STATUS.        PAID_AMT     BILLED._AMT
##      806.215602      382.899461      194.203815      164.608151
## RATIO_PADI_AMT4 RATIO_PADI_AMT2 RATIO_PADI_AMT1 RATIO_PADI_AMT3
##      152.255511      125.071627      109.634039       74.565264
##       LIMIT_BAL             AGE       EDUCATION        MARRIAGE
##       68.850349       30.739832        7.163986        5.436529
```

Performance Major :



Measure	Values
KS	41%
AUC	73%
GINI	14%
Classification Error	

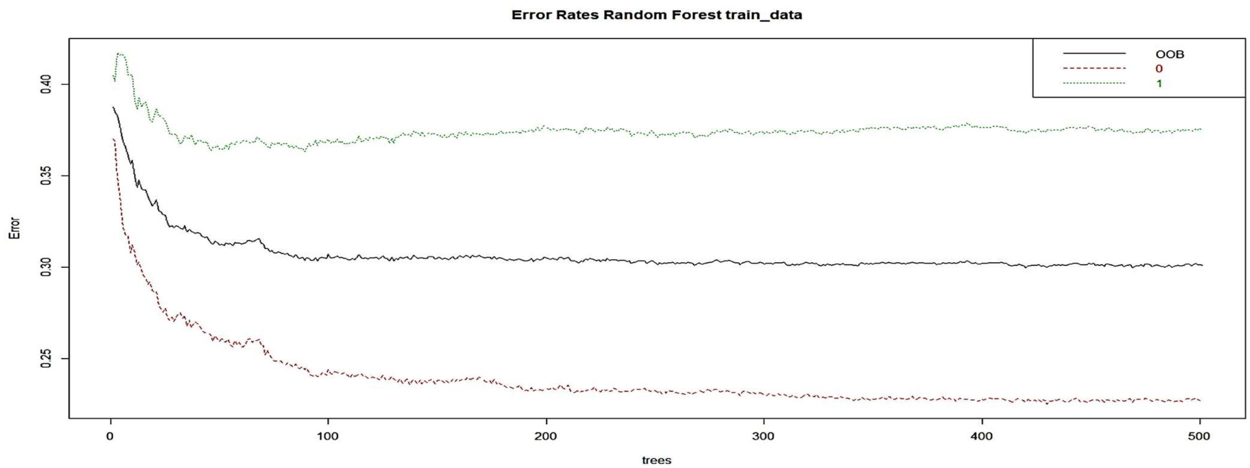
The model observed to perform as per expectations on majority of the model performance measures, indicating it's the model is still need to done better .

6. Random Forest

```
RF_model = randomForest(DEFAULT ~ ., data = train_bank, mtry = 3, nodesize =10, ntr
ee =501, importance = TRUE)
print(RF_model)

##
## Call:
##  randomForest(formula = DEFAULT ~ ., data = train_bank, mtry = 3,      nodesize
= 10, ntree = 501, importance = TRUE)
##                Type of random forest: classification
##                      Number of trees: 501
## No. of variables tried at each split: 3
##
##         OOB estimate of  error rate: 30.09%
## Confusion matrix:
##      0    1 class.error
## 0 3591 1052   0.2265776
 # 1 1744 2904   0.3752151
```

The Out-of-Bag Estimate of Error Rate for our given Random Forest in our case is 30.09% The graphical output for the OOB estimate of error rate is provided below

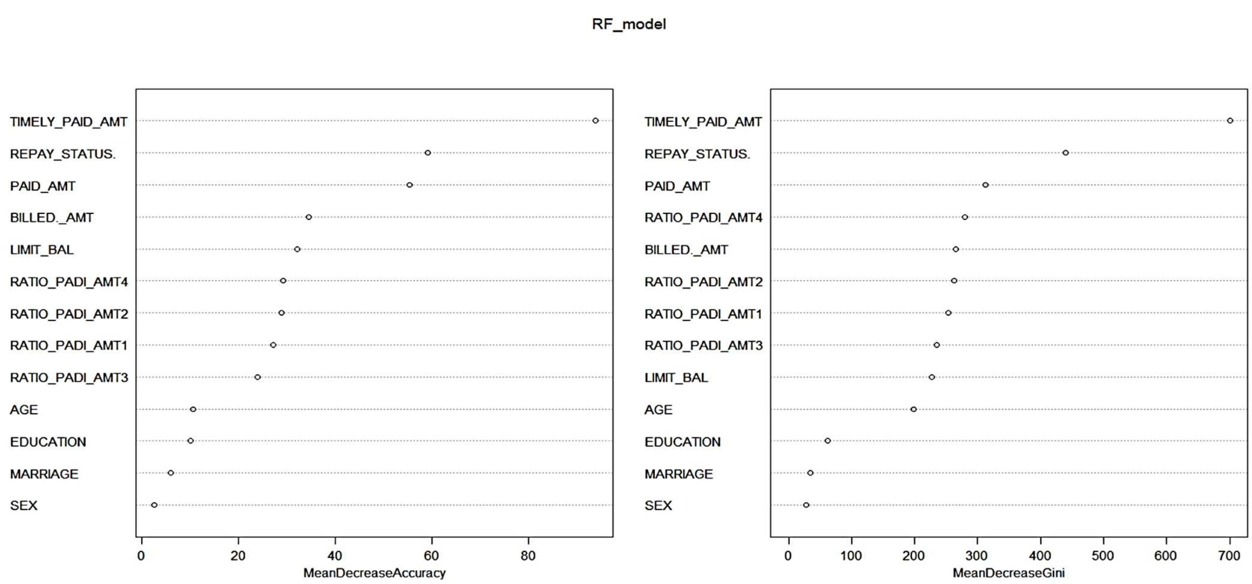


List the importance of the variables.

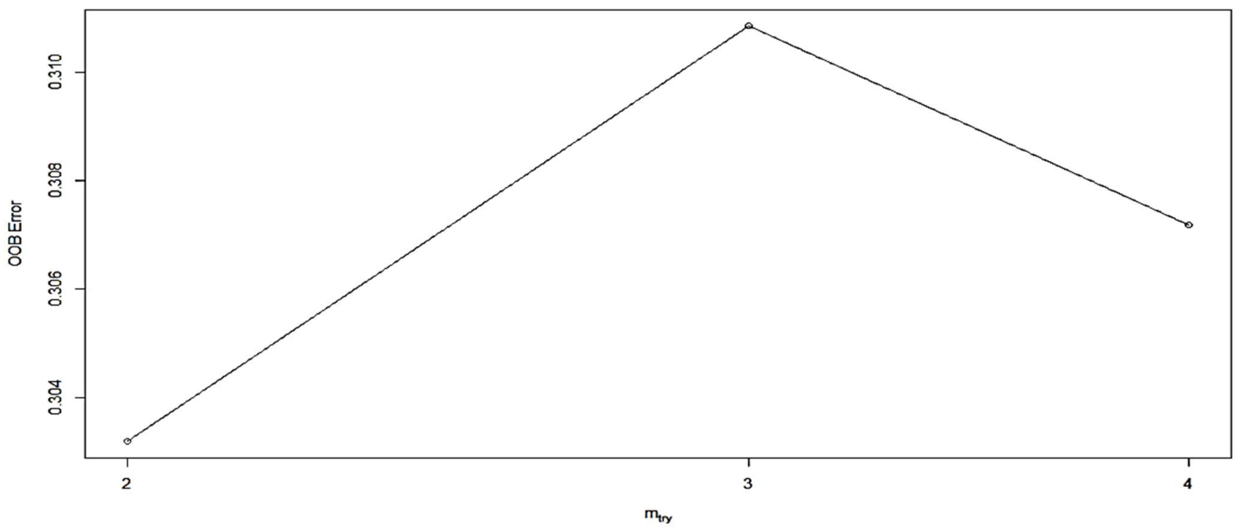
```
##                      0     1 MeanDecreaseAccuracy MeanDecreaseGini
## TIMELY_PAID_AMT 76.59 41.05                93.85           700.48
## REPAY_STATUS.   41.56 23.38                59.15           439.35
## PAID_AMT        18.16 42.96                55.36           312.82
## BILLED._AMT     16.83 20.24                34.56           265.57
## LIMIT_BAL       15.94 23.68                32.12           226.66
## RATIO_PADI_AMT4 16.31 15.72                29.22           280.35
## RATIO_PADI_AMT2 10.11 20.30                28.91           263.15
## RATIO_PADI_AMT1  2.46 29.72                27.23           254.00
## RATIO_PADI_AMT3  7.73 17.35                23.93           235.53
## AGE             13.06  0.56                10.64           198.71
## EDUCATION        6.01  7.93                10.06            62.33
## MARRIAGE         9.33 -2.24                 5.91            35.17
## SEX              1.98  1.38                 2.47            27.94
```

Variable Importance: Graphical representation

```
varImpPlot(RF_model)
```



Optimal mtry value



As can be seen, the optimum number of Variables is 4 to get the optimal Out of Bag Error of 30.09%.

Performance Measures

Rank order

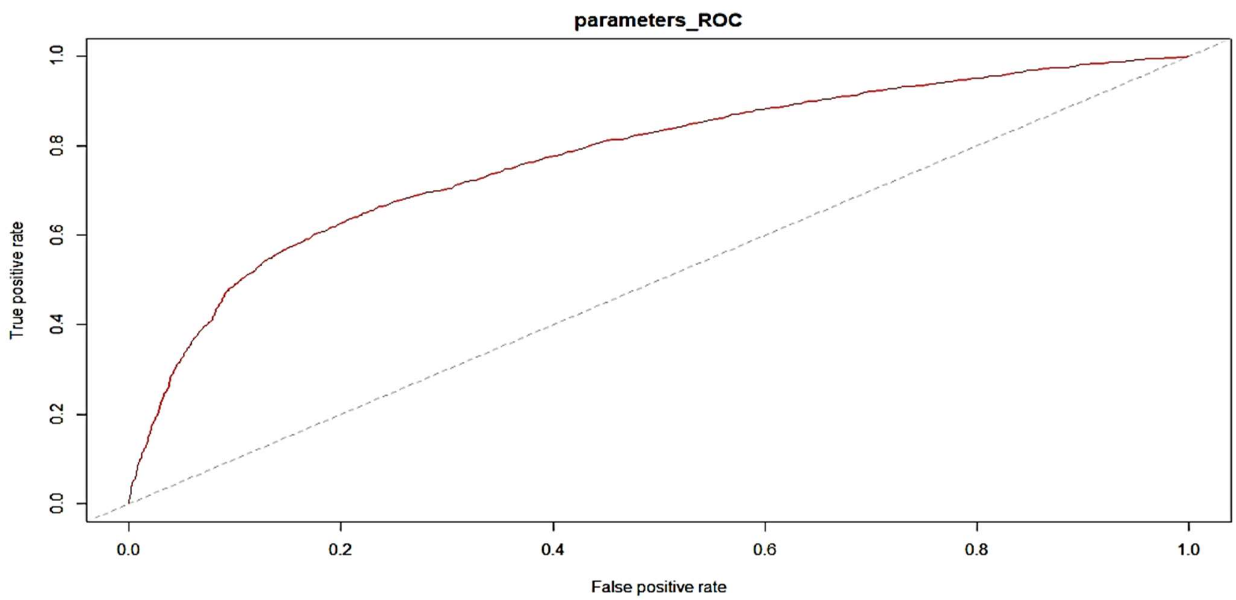
	deciles	cnt	cnt_resp	cnt_non_resp	rrate	cum_resp	cum_non_resp	cum_rel_resp	cum_rel_non_resp	ks
1	1	885	49	836	5.5%	1988	7012	100.0%	100.0%	0.0000
2	9	894	417	477	46.6%	1017	783	51.2%	11.2%	0.3999
3	8	909	243	666	26.7%	1260	1449	63.4%	20.7%	0.4272
4	7	898	161	737	17.9%	1421	2186	71.5%	31.2%	0.4030
5	10	906	600	306	66.2%	600	306	30.2%	4.4%	0.2582
6	6	929	152	777	16.4%	1573	2963	79.1%	42.3%	0.3686
7	5	904	119	785	13.2%	1692	3748	85.1%	53.4%	0.3166
8	4	885	98	787	11.1%	1790	4535	90.0%	64.7%	0.2537
9	3	902	79	823	8.8%	1869	5358	94.0%	76.4%	0.1760
10	2	888	70	818	7.9%	1939	6176	97.5%	88.1%	0.0946

Interpretation:

The baseline Response Rate is 5%, whereas the response rate in top two deciles is 31%, the highest KS is 42%, indicating it to be not a good model.

Measure	Values
AUC	77%
KS	42%
Gini	30%
Accuracy	75%

Here we have seen classification error is 24% comparing to the previous model is little high.



7. Machine learning approach with Ensemble Methods

7.1 KNN Classifier

```
knn_fit<- knn(train = train_bank[,-6], test = test_bank[,-c(6,15)], cl=train_bank$D
EFAULT,k =5,prob=TRUE)
knn_chk= table(test_bank$DEFAULT,knn_fit)
knn_chk

##    knn_fit
##        0    1
##   0 4786 2226
##   1  805 1183

accuracy.knn = sum(diag(knn_chk))/sum(knn_chk)
accuracy.knn

## [1] 0.6632222
```

KNN Algorithm accuracy print It prints accuracy of our knn model. Here our accuracy is 63%. That’s fine we have to check other models .

7.2 Naive Bayes

```
NB = naiveBayes(x =train_bank[-6], y =train_bank$DEFAULT)
pred.NB = predict(NB, newdata =test_bank[-6])
pred.NB

tab.NB =table(test_bank[,6], pred.NB)
tab.NB

##    pred.NB
##        0    1
##   0 5370 1642
##   1  729 1259

accuracy.NB = sum(diag(tab.NB))/sum(tab.NB)
accuracy.NB

## [1] 0.7365556
```

To check the efficiency of the model, we are now going to run the testing data set on the model, after which we will evaluate the accuracy of the model by using a Confusion matrix.

7.2.1 Confusion Matrix and Statistics

```
          Reference
Prediction    0    1
         0 5370  729
         1 1642 1259

               Accuracy : 0.7366
                 95% CI : (0.7273, 0.7456)
    No Information Rate : 0.7791
    P-Value [Acc > NIR] : 1

                  Kappa : 0.3427

 Mcnemar's Test P-Value : <2e-16

            Sensitivity : 0.7658
            Specificity : 0.6333
         Pos Pred Value : 0.8805
         Neg Pred Value : 0.4340
             Prevalence : 0.7791
         Detection Rate : 0.5967
   Detection Prevalence : 0.6777
      Balanced Accuracy : 0.6996

       'Positive' Class : 0
```

The final output shows that we built a Naive Bayes classifier that can predict whether cus0tmer can be defaulted with an accuracy of approximately 70%.

The model observed to perform decent on majority of the model performance measures, indicating it to be a good model.

7.3 Naive Bayes classifier

Create a classification task

```
library(mlr)
```

```
## Call:
## naiveBayes.default(x = X, y = Y, laplace = laplace)
##
## A-priori probabilities:
## Y
##         0         1
## 0.4997309 0.5002691
##
## Conditional probabilities:
##    LIMIT_BAL
## Y       [,1]     [,2]
##   0 174127.9 126738.5
##   1 131340.4 114400.6
##
##    SEX
## Y       [,1]      [,2]
##   0 1.606935 0.4884836
##   1 1.568417 0.4953504
##
##    EDUCATION
## Y       [,1]      [,2]
##   0 1.835236 0.7937926
##   1 1.903614 0.7253323
##
##    MARRIAGE
## Y       [,1]      [,2]
##   0 1.563429 0.5205949
##   1 1.521945 0.5256006
##
##    AGE
## Y       [,1]     [,2]
##   0 35.46866 9.093063
##   1 35.80443 9.723742
##
##    BILLED._AMT
## Y           [,1]      [,2]
##   0  0.008420614 0.9801492
##   1 -0.042941510 0.9762284
##
##    REPAY_STATUS.
## Y          [,1]      [,2]
##   0 -0.08402265 0.7870855
##   1  0.40146671 1.3678757
##
##    PAID_AMT
## Y          [,1]      [,2]
##   0  0.07222019 0.9913057
##   1 -0.22903501 0.6413568
```

```
##
##    TIMELY_PAID_AMT
## Y         [,1]      [,2]
##   0  0.1593017 0.8065326
##   1 -0.5536048 1.1702067
##
##    RATIO_PADI_AMT1
## Y          [,1]      [,2]
##   0  0.02844571 0.9316682
##   1 -0.07652734 0.7439935
##
##    RATIO_PADI_AMT2
## Y           [,1]      [,2]
##   0  0.006558947 0.9229444
##   1 -0.009970515 0.9391812
##
##    RATIO_PADI_AMT3
## Y           [,1]      [,2]
##   0 -0.014470130 0.9407554
##   1  0.005906947 0.7912172
##
##    RATIO_PADI_AMT4
## Y          [,1]      [,2]
##   0  0.04260285 0.8371578
##   1 -0.10099714 1.0038388

##Confusion matrix to check accuracy
table(predictions_mlr[,1],test_bank$DEFAULT)

##
##        0    1
##   0 7012    0
##   1    0 1988

This model is clasified with corretly
```

Confusion matrix to check accuracy for Naive classification task

	0	1	
0	5370	729	6099
1	1642	1259	2901
	7012	1988	

As we see, the predictions are around same 7012 false values . The only way to improve is to have more features or more data. We may arrive at a better model using Naive Bayes

7.4 Bagging

```
bank_bagging <- bagging(DEFAULT ~.,data=train_bank,
                        control=rpart.control(maxdepth=5, minsplit=4))

pred_class <- predict(bank_bagging, test_bank)
```

```
tab.bg <- table(test_bank$DEFAULT,pred_class)
accuracy.bg = sum(diag(tab.bg))/sum(tab.bg)
accuracy.bg

## [1] 0.7672222
```

	0	1	
0	5715	1297	7012
1	804	1184	1988
	6519	2481	

In Bagging also, we are able predict little closure false values ,which shows models full of accuracy but the true values are not correctly classified .

7.5 XGBoost

```
classifier = xgboost(data = as.matrix(train_bank[,-6]), label = train_bank$DEFAULT,
nrounds = 10)

## [1]  train-rmse:0.843655
## [2]  train-rmse:0.666488
## [3]  train-rmse:0.557688
## [4]  train-rmse:0.492921
## [5]  train-rmse:0.456142
## [6]  train-rmse:0.436011
## [7]  train-rmse:0.423342
## [8]  train-rmse:0.415939
## [9]  train-rmse:0.411865
## [10] train-rmse:0.408762

# Making the Confusion Matrix
cm = table(test_bank$DEFAULT, y_pred)
accuracy.bs = sum(diag(cm))/sum(cm)
accuracy.bs

## [1] 0.7791111
```

7.6 K-Fold Cross Validation

```
folds_bank = createFolds(train_bank$DEFAULT, k =12)
cv = lapply(folds_bank, function(x) {
tr_fold = train_bank[-x, ]
tt_fold = test_bank[x, ]
classifier = xgboost(data = as.matrix(train_bank[-6]), label = train_bank$DEFAULT,
nrounds = 10)
y_pred = predict(classifier, newdata = as.matrix(tt_fold[-c(6,15)]))
y_pred = (y_pred >= 0.5)
cmx= table(tt_fold[,6], y_pred)

accuracy = mean(as.numeric(cv))

accuracy

[1] 0.5
```

8. Communicating Results – Conclusion

Summary of comparison :

Models	KS	AUC	GINI	Accuracy	Classification Error
Logistic Regression	41%	75%	14%	87%	18%
CART	41%	73%	14%	76%	19%
Random Forest	42%	77%	30%	76%	24%

Accuracy. KNN	66%
Accuracy. NB	73%
Accuracy. Bagging	76%
Accuracy. Boosting	77%
Accuracy_ Kflod	50%

Conclusion

1) The best models are **Logistic Regression** classifier, we can predict with **87 % accuracy**, whether a customer is likely to default next month.whereas **XGBOOST** method can **77% accuacy .**

The strongest predictors of default are the **PAY_X (ie the repayment status in previous months), the LIMIT_BAL & the PAY_AMTX (amount paid in previous months)**.

9. Appendix

TCD-Project-note.pdf