LOAN DEFAULT PREDICTION AND INVESTMENT STRATEGIES IN ONLINE LENDING

Done By
Pritha Ghosh
Tejaswi Cherukuri
Anoop Gopalam

IDS - 572 | Data Mining for Business-Assignment 1
Loan Default prediction and Investment Strategies in Online Lending

IDS 572 - Assignment 1 - Part B

Pritha Ghosh, Anoop Gopalam, Tejaswi Cherukuri

5. Develop decision tree models to predict default.
(a) Split the data into training and validation sets. What proportions do you consider, why?

A: We consider a 70-30 split between the training and validation sets. We avoid a larger split because the proportion of charged-off loans is only 15% in comparison to the 85% of the fully paid loans.

(b) Train decision tree models (use both rpart, c50)
[If something looks too good, it may be due to leakage – make sure you address this]
What parameters do you experiment with, and what performance do you obtain (on training and validation sets)? Clearly tabulate your results and briefly describe your findings.
How do you evaluate performance – which measure do you consider, and why?

A: We have created the following decision tree models-

1. RPart tree: Complexity parameter = 0.00036, MinSplit=30, split = information
   Variables used in tree construction:

```
Variables actually used in tree construction:
 [1] avg_cur_bal                bc_open_to_buy
 [3] bc_util                    dti
 [5] earliest_cr_line           emp_length
 [7] grade                      loan_amnt
 [9] mo_sin_old_il_acct         mo_sin_old_rev_tl_op
[11] mo_sin_rcnt_rev_tl_op      mo_sin_rcnt_tl
[13] mths_since_recent_inq      num_actv_bc_tl
[15] num_il_tl                  num_op_rev_tl
[17] num_rev_tl_bal_gt_0        pct_tl_nvr_dlq
[19] percent_bc_gt_75           purpose
[21] sub_grade                  tax_liens
[23] tot_hi_cred_lim            total_bc_limit
[25] total_il_high_credit_limit total_rev_hi_lim
```

Performance:

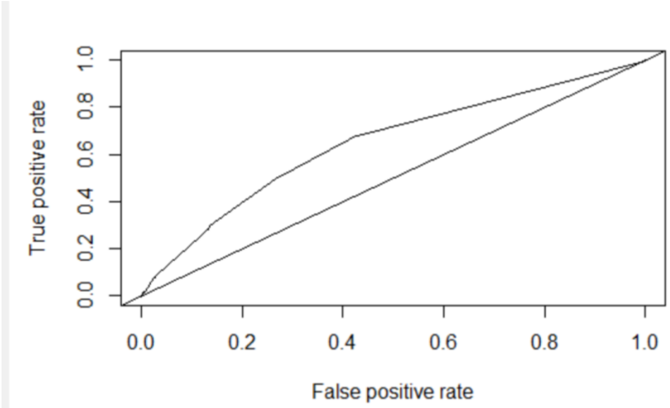
Train Set : Accuracy=85.73%

```
             true
pred          Fully Paid Charged Off
  Fully Paid       48324        7976
  Charged Off        116         299
> mean(predTrn3 == lcdfTrn$loan_status)
[1] 0.8573217
```

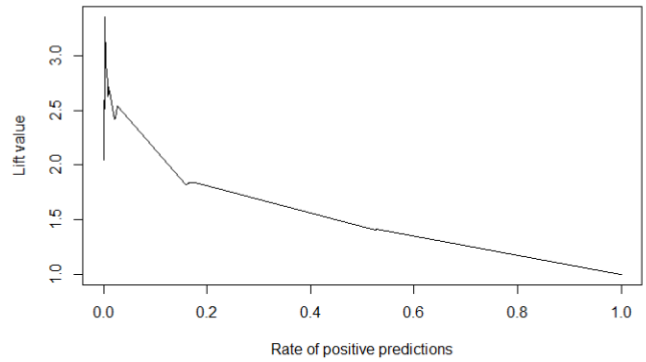
Test Set: Accuracy = 85.08%

```
s)
              true
pred           Fully Paid Charged Off
  Fully Paid       20618        3489
  Charged Off        137          63
> mean(predict(lcDT3,lcdfTst, type='class') ==l
[1] 0.8508249
```

AUC Curve: 0.647



Lift Curve:



2. RPart tree: Complexity parameter = 0.00036, MinSplit=30, split = gini

Variables used in tree construction:

```
Variables actually used in tree construction:
 [1] acc_open_past_24mths        annual_inc
 [3] avg_cur_bal                 bc_open_to_buy
 [5] bc_util                     dti
 [7] earliest_cr_line            emp_length
 [9] installment                 int_rate
[11] loan_amnt                   mo_sin_old_il_acct
[13] mo_sin_old_rev_tl_op        mo_sin_rcnt_rev_tl_op
[15] mort_acc                    mths_since_recent_bc
[17] mths_since_recent_inq       num_accts_ever_120_pd
[19] num_actv_rev_tl             num_bc_sats
[21] num_il_tl                   num_sats
[23] num_tl_op_past_12m          pct_tl_nvr_dlq
[25] purpose                     sub_grade
[27] tot_hi_cred_lim             total_bal_ex_mort
[29] total_bc_limit              total_il_high_credit_limit
[31] total_rev_hi_lim
```

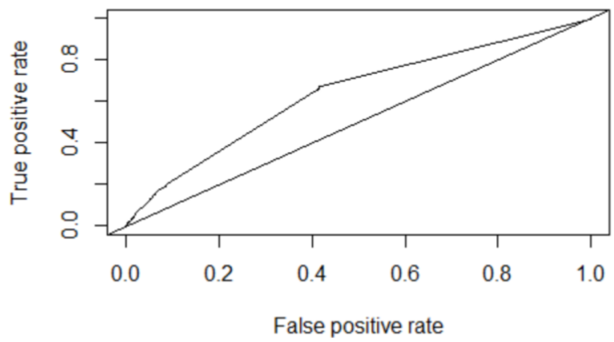
On Train Data: Accuracy = 85.92%

```
> table(pred = predTrn1, true=lcdfTrn$loan_s
               true
pred            Fully Paid Charged Off
  Fully Paid         48229        7803
  Charged Off          179         504
> mean(predTrn1 == lcdfTrn$loan_status)
[1] 0.8592612
```

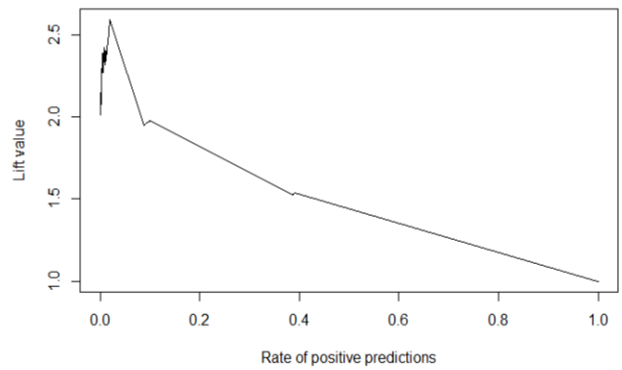
On Test Data: Accuracy = 85.16%

```
_status)
               true
pred            Fully Paid Charged Off
  Fully Paid         20599        3419
  Charged Off          188         101
> mean(predict(lcDT1,lcdfTst, type='cla
[1] 0.8516065
```

AUC Curve:0.639



Lift Curve:



3. C50 tree:trials=50

Number of predictors = 49

```
>
> print(ctree)

Call:
C5.0.formula(formula = as.factor(trainset$loan_status)
 ~ ., data = trainset, method = "class", trials = 50)

Classification Tree
Number of samples: 56715
Number of predictors: 49

Number of boosting iterations: 50
Average tree size: 157

Non-standard options: attempt to group attributes
```

Performance:

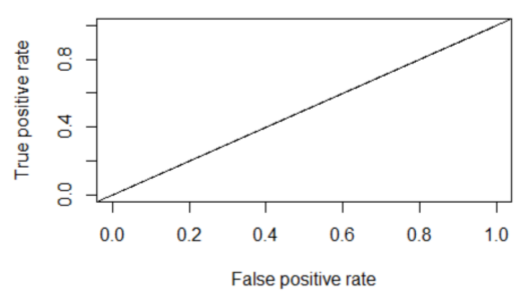
Train set : Accuracy = 86.54%

```
> table(pred = predTrain, true=trainset$loan_
              true
pred            Fully Paid Charged Off
  Fully Paid         48467        7616
  Charged Off           13         619
> mean(predTrain == trainset$loan_status)
[1] 0.8654853
> table(pred = predict(ctree,testset, type='
```

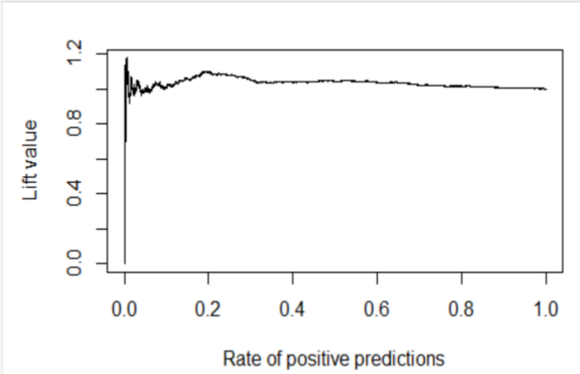
Test set: Accuracy:85.11%

```
_status)
             true
pred            Fully Paid Charged Off
  Fully Paid         20655        3557
  Charged Off           60          35
> mean(predict(ctree,testset, type='cl
[1] 0.8511951
>
```

AUC = 0.49



Lift Curve:



To evaluate performance, we see the accuracy in predicting the response variable correctly.
We also use the AUC and Lift curves as a test of performance.

(c) Identify the best tree model. Why do you consider it best? Describe this model – in terms of complexity (size). Examine variable importance. Briefly describe how variable importance is obtained in your best model.

A:For the first decision tree using rpart and information split, we got an accuracy of 85.73% on the training set and an accuracy of 85.08% on the test set. The AUC value that we got was 0.647.
For the second decision tree using rpart and gini split, we got an accuracy of 85.92% on the training set and an accuracy of 85.16% on the test set. The AUC value that we got was 0.639.
For the third decision tree using c50, we got an accuracy of 86.54% on the training set and an accuracy of 85.11% on the test set. The AUC value that we got was 0.49.
We reject the c50 tree because the AUC value is quite low.
Both the rpart trees perform considerably better. They both have AUC value of greater than 0.6 and perform well on both the train as well as the test sets.
If we would be forced to choose between one of them, we would choose the one with the gini split, because it performs slightly better, when we keep all the other parameter like cp and minsplit the same. The one with information split would require more processing time.
The lift curve of that model is also used as a measure of the effectiveness of a predictive model, calculated as the ratio between the results obtained with and without the predictive model. The greater the area between the lift curve and the baseline, the better the model. The life curve of the rpart tree with the gini split shows that the model is good.

This model uses 25 variables in the tree construction. This is far less than the number of variables used in the other trees, therefore we can be certain that our model will not cause overfit.

The variable importance of this model is listed below :

grade	1083.7846556
int_rate	1026.3511686
bc_open_to_buy	237.8606938
total_bc_limit	212.5593145
total_rev_hi_lim	174.1314679
sub_grade	152.3769180
annual_inc	148.1297711

6. Develop a random forest model. (Note the 'ranger' library can give faster computations) What parameters do you experiment with, and does this affect performance? Describe the best model in terms of number of trees, performance, variable importance. Compare the random forest and best decision tree model from Q 4 above. Do you find the importance of variables to be different? Which model would you prefer, and why ?

In the first model, we have kept the num.trees = 50, importance = permutation.
We got an accuracy of 85% on the test data and a prediction error of 15%.

```
Confusion Matrix and Statistics

              Charged Off Fully Paid
  Charged Off          31       3504
  Fully Paid           46      20726

               Accuracy : 0.854
                 95% CI : (0.8494, 0.8584)
    No Information Rate : 0.9968
    P-Value [Acc > NIR] : 1

                  Kappa : 0.011

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

            Sensitivity : 0.402597
            Specificity : 0.855386
         Pos Pred Value : 0.008769
         Neg Pred Value : 0.997785
             Prevalence : 0.003168
         Detection Rate : 0.001275
   Detection Prevalence : 0.145431
      Balanced Accuracy : 0.628992

       'Positive' Class : Charged Off
```

```
> rgModel2$prediction.error
[1] 0.1510888
```

In the second model, we experiment with parameters num.trees=50 and importance = impurity.

```
              Charged Off Fully Paid
  Charged Off          37       3498
  Fully Paid           43      20729

               Accuracy : 0.8543
                 95% CI : (0.8498, 0.8587)
    No Information Rate : 0.9967
    P-Value [Acc > NIR] : 1

                  Kappa : 0.0141

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

            Sensitivity : 0.462500
            Specificity : 0.855616
         Pos Pred Value : 0.010467
         Neg Pred Value : 0.997930
             Prevalence : 0.003291
         Detection Rate : 0.001522
   Detection Prevalence : 0.145431
      Balanced Accuracy : 0.659058
```

```
> rgModel2$prediction.error
[1] 0.1503659
```

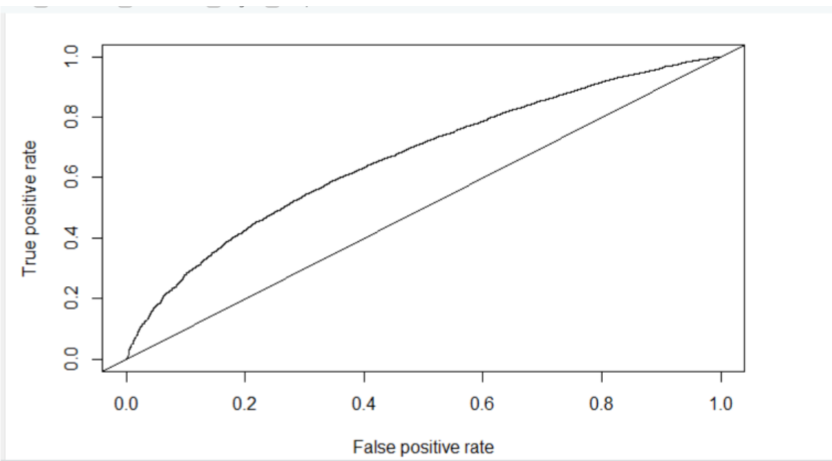
This model also gives us an accuracy of 85% on test data and a prediction error of 15%, but it does a slightly better job at predicting the "Charged Off" loans.
In the first model, the variable importance is as below -

```
> sort(rgModel1$variable.importance, decreasing = TRUE)
     tot_hi_cred_lim        total_bc_limit           avg_cur_bal
        1.753977e-02          1.359696e-02          1.245160e-02
      bc_open_to_buy       total_rev_hi_lim           installment
        1.236969e-02          1.100106e-02          8.754416e-03
             bc_util            annual_inc             loan_amnt
        8.704137e-03          8.166425e-03          7.976640e-03
```

In the second model, the variable importance is as below -

```
sort(rgModel2$variable.importance*1000, decreasing = TRUE)
                 dti              int_rate           avg_cur_bal
          568674.519            516728.081            508467.502
      bc_open_to_buy               bc_util  mo_sin_old_rev_tl_op
          493121.742            491758.229            489483.699
    total_rev_hi_lim       tot_hi_cred_lim      mo_sin_old_il_acct
          480629.130            475113.646            472828.301
```

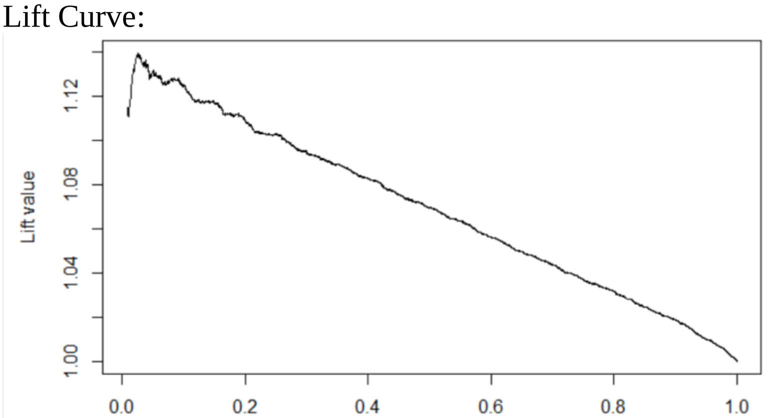
We choose the second model since it is comparatively better at predicting the Charged Off Loans
The AUC curve for the model is as below -




```
[1] 0.665284
```

The AUC value is : 0.66.

Lift Curve:



Since the model is not predicting the “Charged Off” cases correctly, we try oversampling the data.

```
> data_balanced_over <- ovun.sample(loan_status ~ ., data =lcdfTrn, method = "over",N =95000)$data
> table(data_balanced_over$loan_status)

Charged Off  Fully Paid
      46603       48397
```

After oversampling, the confusion matrix on the test set is as below-

```
             Charged Off Fully Paid
  Charged Off        117       3392
  Fully Paid         258      20540

               Accuracy : 0.8498
                 95% CI : (0.8453, 0.8543)
    No Information Rate : 0.9846
    P-Value [Acc > NIR] : 1

                  Kappa : 0.0333

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

            Sensitivity : 0.312000
            Specificity : 0.858265
         Pos Pred Value : 0.033343
         Neg Pred Value : 0.987595
             Prevalence : 0.015428
         Detection Rate : 0.004813
   Detection Prevalence : 0.144362
      Balanced Accuracy : 0.585133

       'Positive' Class : Charged Off
```

After balancing the dataset, the performance on predicting the “Charged Off” loans has improved by 2%.

Among the decision tree and random forest models, I would prefer the decision tree model with rpart() since it has the best overall accuracy and does a much better job at predicting the charged off loans.

Additional Model : XG Boost
Parameters: nrounds = 500, nfold=5, early_stopping_rounds = 10

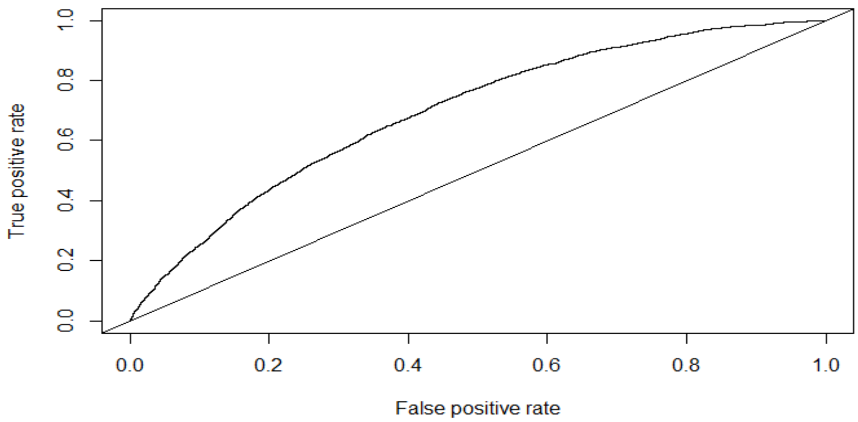
```
Stopping. Best iteration:
[75]    train-error:0.146663    train-auc:0.706721     eval-error:0.144362     eval-auc:0.681558

> xgb_lsM1$best_iteration
```

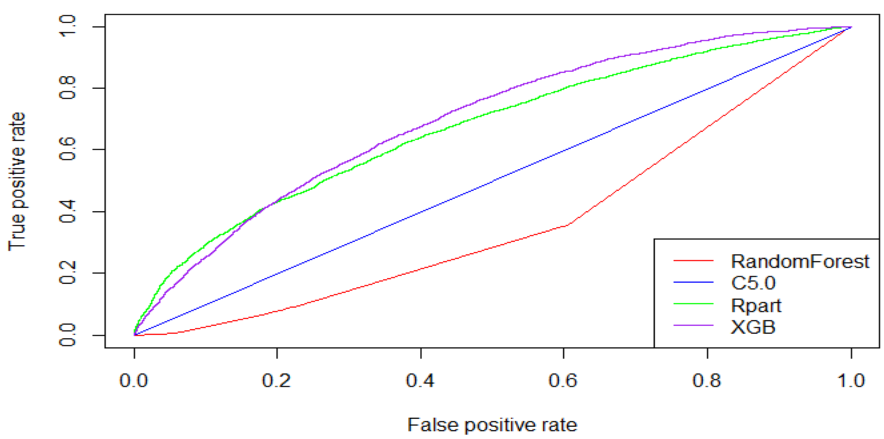
AUC value on test set: 0.68
Top 10 important variables -

	Feature	Gain	Cover	Frequency
1	int_rate	0.5465437685	1.976317e-01	0.1036525173
2	dti	0.0378911253	5.120722e-02	0.0523198421
3	installment	0.0370865508	5.768104e-02	0.0533070089
4	acc_open_past_24mths	0.0325252746	5.219352e-02	0.0365251728
5	avg_cur_bal	0.0325087844	4.197792e-02	0.0493583416
6	tot_hi_cred_lim	0.0316494806	7.171615e-02	0.0631786772
7	annual_inc	0.0199906575	4.195277e-02	0.0473840079
8	bc_util	0.0194640701	2.580485e-02	0.0434353406
9	mo_sin_old_rev_tl_op	0.0184014478	4.232077e-02	0.0444225074
10	emp_lengthn/a	0.0145163083	2.980452e-02	0.0187561698

AUC:



Consolidated ROC curves –



Model	**Parameters**	**Accuracy**	**AUC Value**
Rpart Decision Tree	Complexity parameter = 0.00036, MinSplit=30, split = gini	Train:85.92% Test:85.16%	0.639
C50 Decision Tree	trials=50	Train:86.54 Test: 85.11%	0.49
Random Forest	num.trees = 50, importance = impurity	Train:87% Test:85.4%	0.66
XGBoost	nrounds = 500, nfold=5, early_stopping_rounds = 10	Train:85.4% Test:85.6%	0.68

7. The average term for the Charged Off loans is 3 years, with an average interest rate of 13.8% and an average return rate of -11.69%.
The average term for the Fully Paid Off loans is 2.09 years, with an average interest rate of 11.56% and an average return rate of 8.03%.

loan_status <chr>	avgInt <dbl>	avgRet <dbl>	avgTerm <dbl>
Charged Off	13.88228	-11.697488	3.00000
Fully Paid	11.56970	8.031333	2.09523

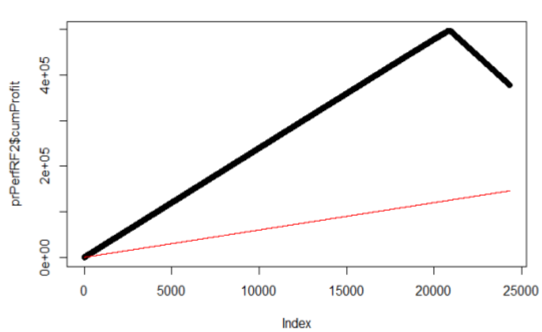
Therefore, for Fully Paid loans, on an investment of $100, there will be an average profit of $8 per year.

Since the average term for the loan is 2.09 years, the total average return on investment would be ~$16.
To compare equivalently to charged off loans, which have an average term of 3 years, let us assume that for the remainder of the 1 year term remaining for the Fully paid loans, the return is at a risk-free rate of 2%.
Therefore, we can assume that for the total term of 3 years, one would make a profit of about $18 on the Fully paid off loans.
This is a conservative assumption, for ease of our calculations. In a real-life scenario, the average return is computed monthly and not annually, therefore, an investor is easily able to re-invest the profits gained on a loan into another loan to make even higher profits.

decile <int>	count <int>	numDefaults <int>	avgActRet <dbl>	minRet <dbl>	maxRet <dbl>	avgTer <dbl>	totA <int>	totB <int>	totC <int>	totD <int>	totE <int>	totF <int>
1	2431	0	8.3180140	0.0000000	31.17493	1.677289	819	770	595	220	26	1
2	2431	0	8.2328534	0.0000000	27.28570	1.716773	794	825	573	196	41	2
3	2431	0	8.4899999	3.0791241	26.79482	1.716853	804	763	595	234	33	2
4	2431	0	8.2877414	0.0000000	29.60077	1.875930	805	717	628	230	44	7
5	2431	1	7.8313673	0.8051210	31.09951	2.188751	699	725	683	255	62	7
6	2431	3	7.7750295	-0.1458667	29.28361	2.254705	653	697	688	310	63	19
7	2431	4	7.7079562	-0.3334643	29.01263	2.456243	541	741	737	296	96	16
8	2430	18	7.6009974	-4.8463582	30.65945	2.565180	445	692	739	412	118	22
9	2430	1115	-0.6306885	-33.3333333	36.73610	2.812541	253	575	726	552	265	46
10	2430	2430	-12.0736247	-32.2695000	10.18747	3.000000	175	540	887	586	198	37

We first analyse the profits using the random forest method - This model is not ideal because it doesn't do a very good job at predicting the charged off loans correctly.
Score corresponding to maximum profit :

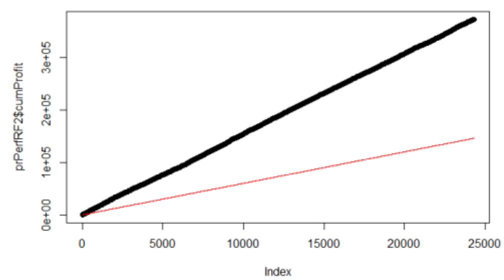
```
> which.max(prPerfRF2$cumProfit)
[1] 20841
```



7474	0.9960000	Charged Off	-35	226082

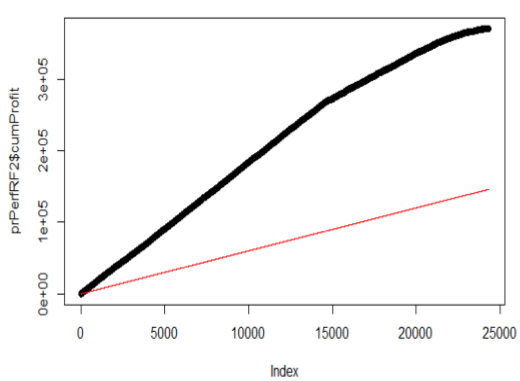
The threshold value is too high with the random forest model.

With C50 decision tree –



scoreRF	status	profit	cumProfit
0.1454465	Charged Off	-35	277

Next, we do the same analysis using the rpart decision tree - This is our best model as it does an equally good job of predicting both the Charged Off as well as the Fully Paid loans correctly.



	scoreRF	status	profit	cumProfit
1868	0.9090909	Fully Paid	24	24
3568	0.9090909	Fully Paid	24	48
5062	0.9090909	Charged Off	-35	13
12692	0.9090909	Fully Paid	24	37
15693	0.9090909	Fully Paid	24	61

For this model, the threshold is 0.909. We observe this in the graph as well - up until about the 15000th prediction, the graph is rising steadily upwards. The rate decreases when the probability of prediction drops to less than ~90%, which is the optimal threshold.