# IDS 572 ASSIGNMENT 2

## Prediction and Investment Decisions - Lending Club

**Harika Kalyani Lakhinena - hlakhi2@uic.edu - UIN: 676218205**

**Roshan Dhanasiri - rdhana5@uic.edu - UIN: 676134844**

**Sathwik Reddy Maddi - smaddi9@uic.edu - UIN: 658204431**

**1. (a) Develop boosted tree models (using either gbm or xgBoost) to predict loan_status. Experiment with different parameters using a grid of parameter values. Use cross-validation. Explain the rationale for your experimentation. How does performance vary with parameters, and which parameter setting you use for the 'best' model?**

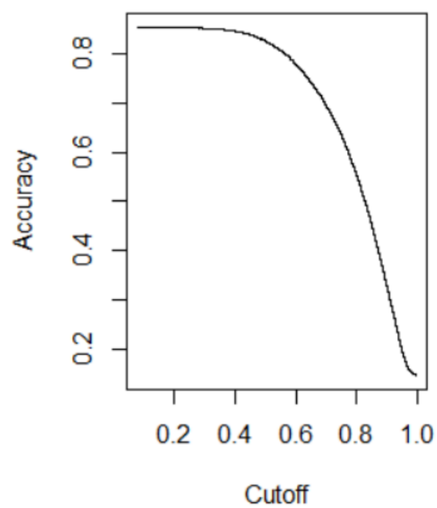
**Model performance should be evaluated through use of same set of criteria as for the earlier models - confusion matrix based, ROC analyses and AUC, cost-based performance.**

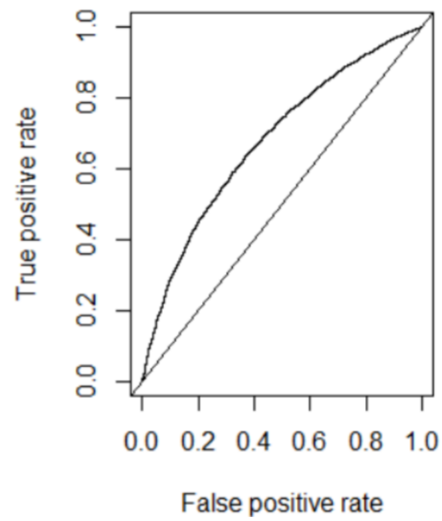
**Ans**. We're predicting loan_status using GBM.

GBM using Over-sampled data:

**Variable Importance:**

```
A gradient boosted model with bernoulli loss function.
1000 iterations were performed.
The best test-set iteration was 1000.
There were 42 predictors of which 40 had non-zero influence.
> summary(gbm_paramTune_os)
                                                      var     rel.inf
sub_grade                                       sub_grade 29.37019887
addr_state                                     addr_state 26.87944626
int_rate                                         int_rate 14.49181093
acc_open_past_24mths               acc_open_past_24mths  2.57238301
dti                                                   dti  2.10060938
avg_cur_bal                                   avg_cur_bal  1.96298874
tot_hi_cred_lim                           tot_hi_cred_lim  1.88380652
purpose                                           purpose  1.43977922
mo_sin_old_il_acct                     mo_sin_old_il_acct  1.36983110
mo_sin_old_rev_tl_op                 mo_sin_old_rev_tl_op  1.36323598
loan_amnt                                       loan_amnt  1.32584361
total_rev_hi_lim                         total_rev_hi_lim  1.18867531
mths_since_recent_bc                 mths_since_recent_bc  1.15180751
bc_util                                           bc_util  1.10525669
revol_bal                                       revol_bal  1.09663839
annual_inc                                     annual_inc  1.02533797
bc_open_to_buy                             bc_open_to_buy  1.01056551
total_bc_limit                             total_bc_limit  0.79145888
total_bal_ex_mort                       total_bal_ex_mort  0.69234696
pct_tl_nvr_dlq                             pct_tl_nvr_dlq  0.64339673
tot_cur_bal                                   tot_cur_bal  0.63983038
total_il_high_credit_limit total_il_high_credit_limit  0.63978395
revol_util                                     revol_util  0.63798531
total_acc                                       total_acc  0.52108200
num_actv_bc_tl                             num_actv_bc_tl  0.48595368
home_ownership                             home_ownership  0.41830992
num_rev_accts                               num_rev_accts  0.40732341
num_bc_tl                                       num_bc_tl  0.40030714
num_rev_tl_bal_gt_0                   num_rev_tl_bal_gt_0  0.33801129
num_actv_rev_tl                           num_actv_rev_tl  0.33110438
num_op_rev_tl                               num_op_rev_tl  0.28029623
mo_sin_rcnt_rev_tl_op               mo_sin_rcnt_rev_tl_op  0.26789317
```



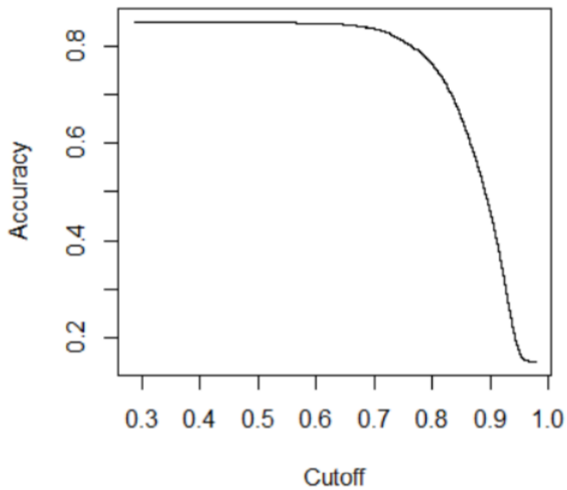

**Accuracy with Threshold (0.5) = 0.8125742**
**Area Under Curve = 0.6794854**

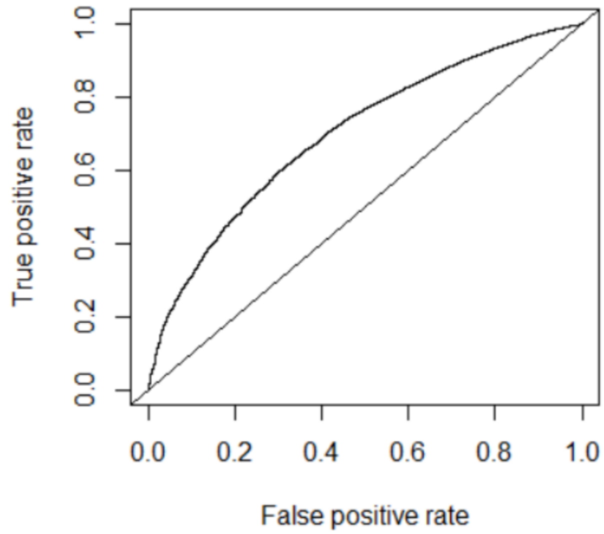
**GBM using Under-sampled data:**

**Variable Importance:**

```
A gradient boosted model with bernoulli loss function.
1000 iterations were performed.
The best test-set iteration was 87.
There were 36 predictors of which 31 had non-zero influence.
> summary(gbm_paramTune_us)
                                                var      rel.inf
sub_grade                                 sub_grade 31.91082033
addr_state                               addr_state 31.81100925
int_rate                                   int_rate  7.72148760
dti                                             dti  2.30672781
loan_amnt                                 loan_amnt  2.09651176
mo_sin_old_rev_tl_op           mo_sin_old_rev_tl_op  1.92037631
acc_open_past_24mths           acc_open_past_24mths  1.82493941
total_bc_limit                       total_bc_limit  1.51468447
total_bal_ex_mort                 total_bal_ex_mort  1.44928792
revol_bal                                 revol_bal  1.43803822
avg_cur_bal                             avg_cur_bal  1.40294668
total_rev_hi_lim                   total_rev_hi_lim  1.37367196
annual_inc                               annual_inc  1.35306004
tot_cur_bal                             tot_cur_bal  1.21878019
tot_hi_cred_lim                     tot_hi_cred_lim  1.17070015
total_il_high_credit_limit total_il_high_credit_limit  1.04021441
purpose                                     purpose  1.00434741
pct_tl_nvr_dlq                       pct_tl_nvr_dlq  0.73163135
mo_sin_rcnt_rev_tl_op           mo_sin_rcnt_rev_tl_op  0.72264838
total_acc                                 total_acc  0.70412777
num_rev_accts                         num_rev_accts  0.58682243
num_bc_tl                                 num_bc_tl  0.51648829
num_tl_op_past_12m               num_tl_op_past_12m  0.49943079
mort_acc                                   mort_acc  0.45875864
mo_sin_rcnt_tl                       mo_sin_rcnt_tl  0.43478643
num_rev_tl_bal_gt_0             num_rev_tl_bal_gt_0  0.40627906
num_bc_sats                             num_bc_sats  0.39660161
open_acc                                   open_acc  0.39582334
num_actv_bc_tl                       num_actv_bc_tl  0.33811967
num_actv_rev_tl                     num_actv_rev_tl  0.31001555
num_il_tl                                 num_il_tl  0.26820116
home_ownership                       home_ownership  0.23934229
num_op_rev_tl                         num_op_rev_tl  0.23447582
```

**Accuracy VS Cutoff plot:**



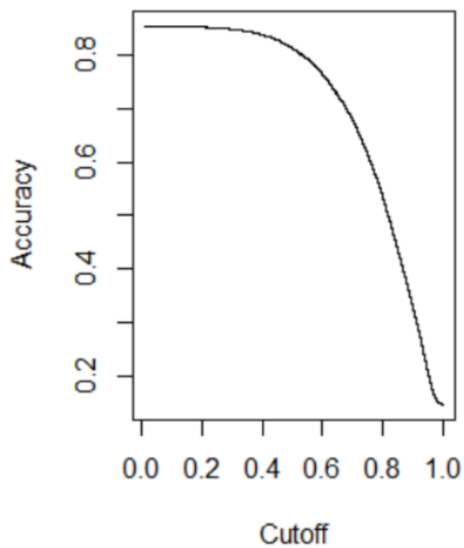

Accuracy with Threshold (0.5): 0.8321157

Area Under Curve = 0.7007469

**Variable Importance:**

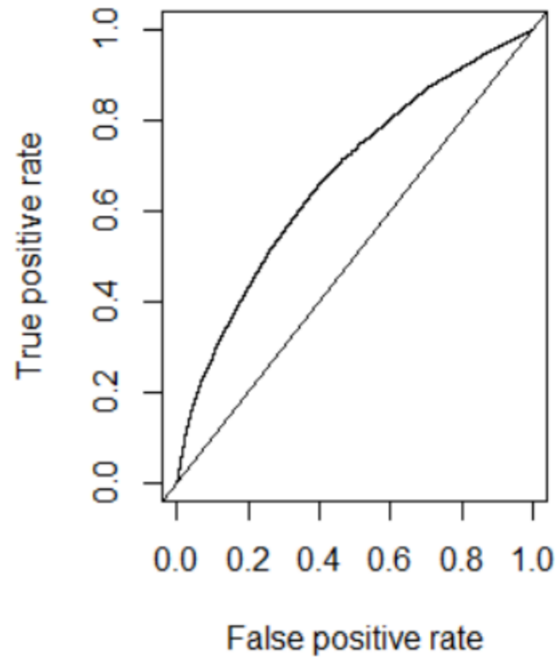
```
A gradient boosted model with bernoulli loss function.
1000 iterations were performed.
The best test-set iteration was 1000.
There were 42 predictors of which 40 had non-zero influence.
> summary(gbm_paramTune)
                                                  var     rel.inf
sub_grade                                   sub_grade 33.23943743
addr_state                                 addr_state 28.55039410
int_rate                                     int_rate  8.04243661
revol_util                                 revol_util  3.28600238
acc_open_past_24mths             acc_open_past_24mths  2.46693936
dti                                               dti  2.20867940
avg_cur_bal                               avg_cur_bal  1.87844915
tot_hi_cred_lim                       tot_hi_cred_lim  1.78856025
mo_sin_old_il_acct                 mo_sin_old_il_acct  1.45013325
bc_util                                       bc_util  1.37924574
annual_inc                                 annual_inc  1.20493254
mths_since_recent_bc             mths_since_recent_bc  1.13849389
loan_amnt                                   loan_amnt  1.10484985
total_rev_hi_lim                     total_rev_hi_lim  1.00333749
bc_open_to_buy                         bc_open_to_buy  0.95439609
mo_sin_old_rev_tl_op             mo_sin_old_rev_tl_op  0.90363841
purpose                                       purpose  0.81916582
home_ownership                         home_ownership  0.80627995
revol_bal                                   revol_bal  0.70868291
total_bal_ex_mort                   total_bal_ex_mort  0.62724410
total_bc_limit                         total_bc_limit  0.55805037
total_il_high_credit_limit total_il_high_credit_limit  0.54666759
num_actv_rev_tl                       num_actv_rev_tl  0.46559070
mo_sin_rcnt_tl                         mo_sin_rcnt_tl  0.45332756
pct_tl_nvr_dlq                         pct_tl_nvr_dlq  0.44066133
num_bc_tl                                   num_bc_tl  0.41815570
num_actv_bc_tl                         num_actv_bc_tl  0.40651696
num_il_tl                                   num_il_tl  0.38056084
num_rev_tl_bal_gt_0               num_rev_tl_bal_gt_0  0.36063540
num_rev_accts                           num_rev_accts  0.32503092
mort_acc                                     mort_acc  0.31442553
tot_cur_bal                               tot_cur_bal  0.31127919
mo_sin_rcnt_rev_tl_op           mo_sin_rcnt_rev_tl_op  0.26320288
total_acc                                   total_acc  0.23043274
```

**Accuracy vs Cutoff Plot:**




Accuracy with Threshold (0.5) = 0. 8125742

**ROC Curve:**

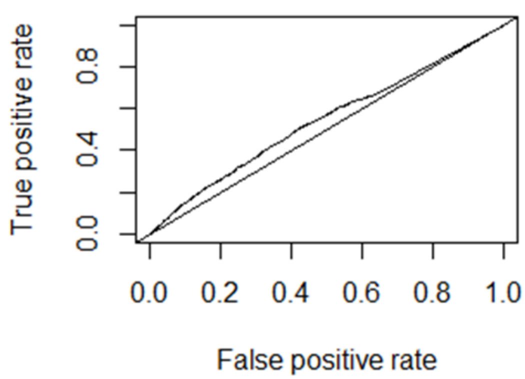



Area under curve = 0.6751129

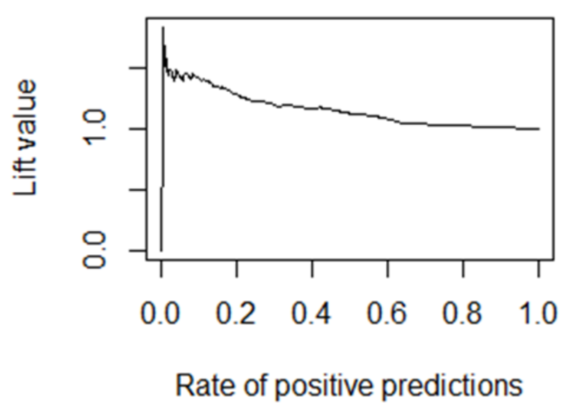
**Provide a table with comparative evaluation of all the best models from each method (decision trees and random forests from the earlier assignment and boosted trees); show their ROC curves in a combined plot. Also provide profit-curves and 'best' profit' and associated cutoff values. At the respective best cutoff levels, what are the accuracy values for the different models?**

**Evaluation of Decision Trees**

**ROC Curve:**



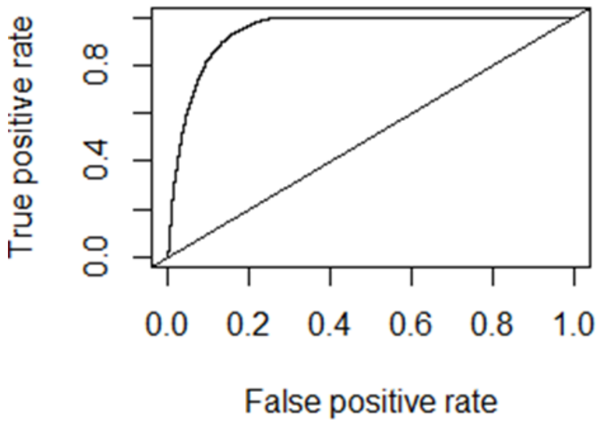

**LIFT Curve:**



AUC is 0.5440

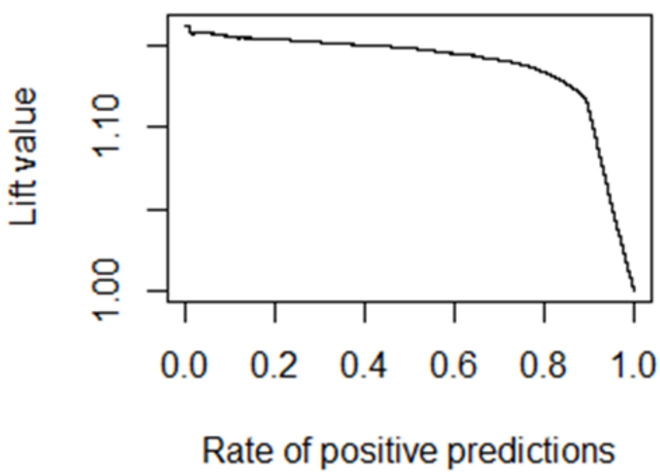
**Evaluation of Random Forests**

**ROC Curve:**

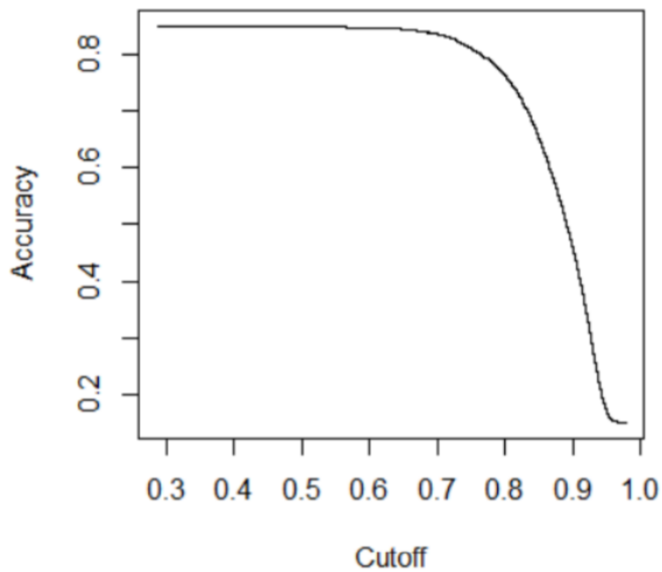



AUC: 0.9470

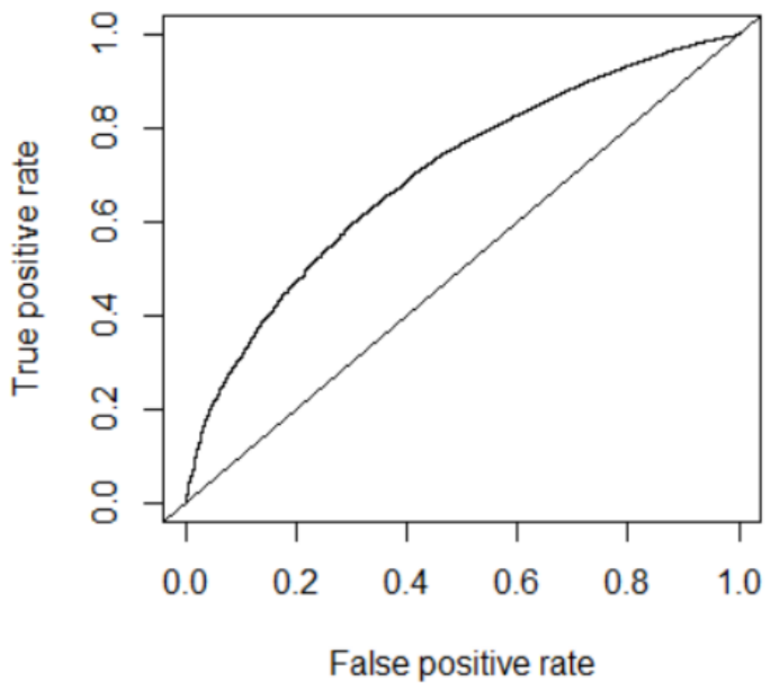
**Lift Curve:**




**Accuracy VS Cutoff plot:**




Accuracy with Threshold (0.5): 0.8321157




Area Under Curve = 0.7007469

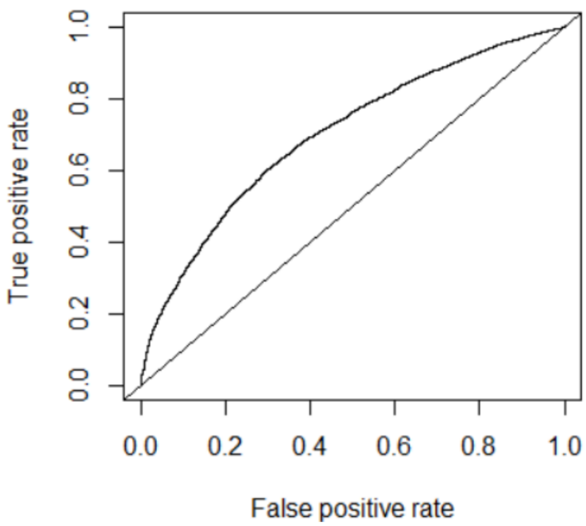
---

**2. (a) Develop linear (glm) models to predict loan_status. Experiment with different parameter values and identify which gives 'best' performance. Use cross-validation. Describe how you determine 'best' performance. How do you handle variable selection?**

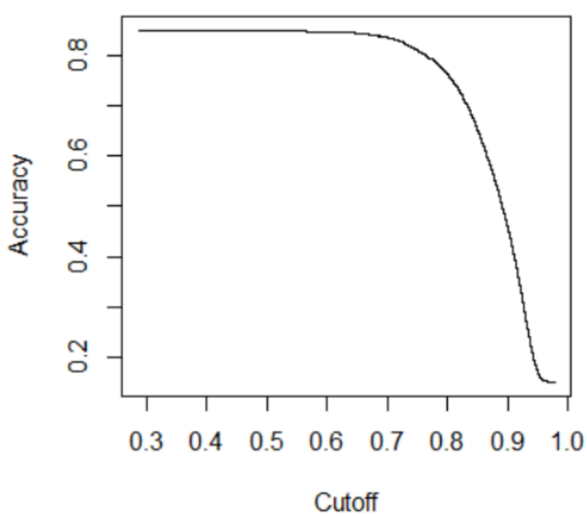
**Experiment with Ridge and Lasso, and show how you vary these parameters, and what performance is observed.**

**Ans.** The best model is determined by the model which has the highest accuracy and area under the curve. In GLM the model with best performance is GLM using Ridge function.

**GLM:**

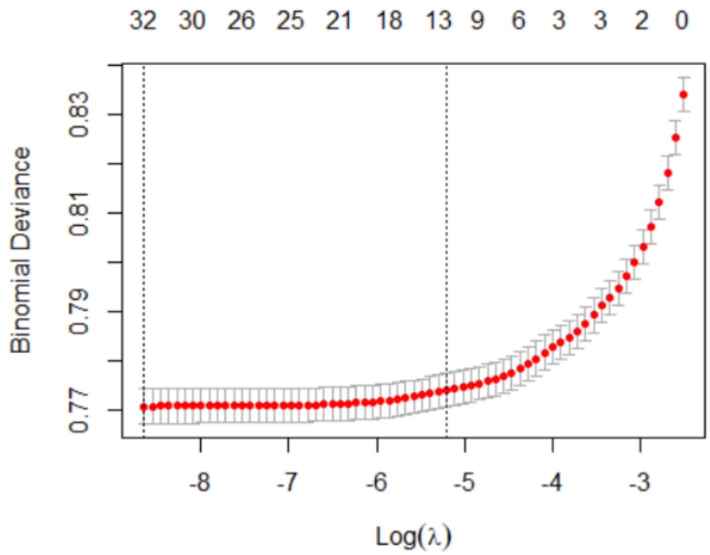



Area Under Curve = 0.6955761

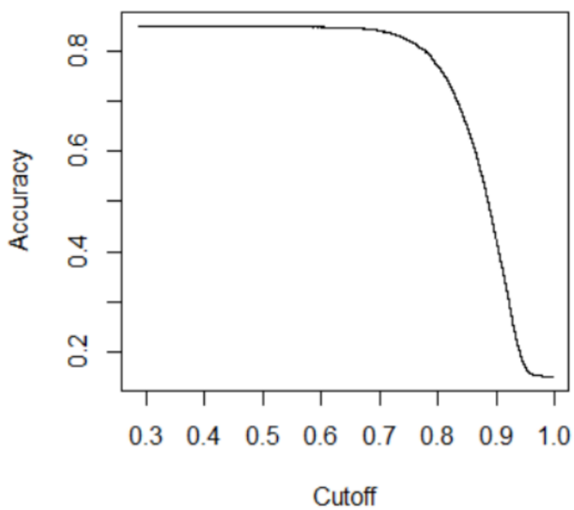


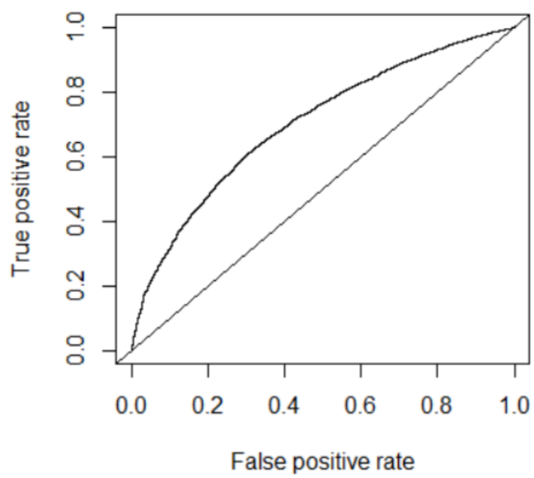

Accuracy= 0.850937488753734

**Plot of GLMNET:**



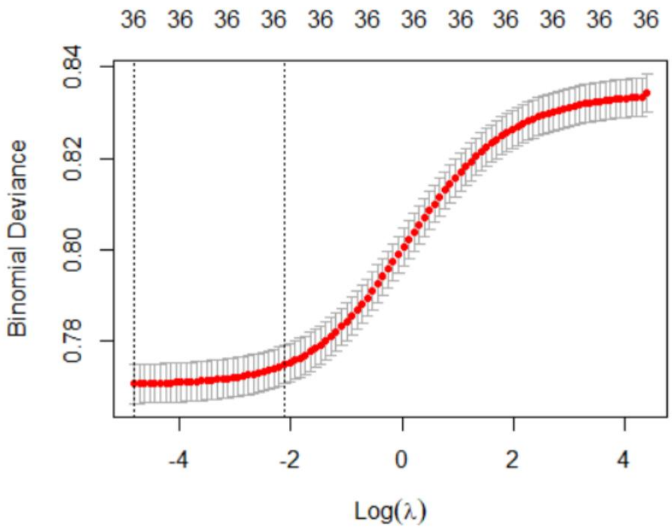

GLM using Ridge:



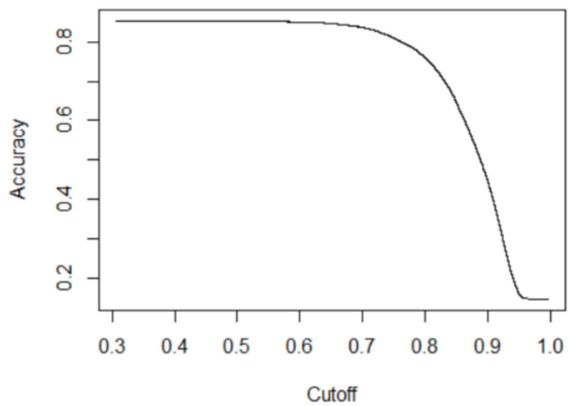



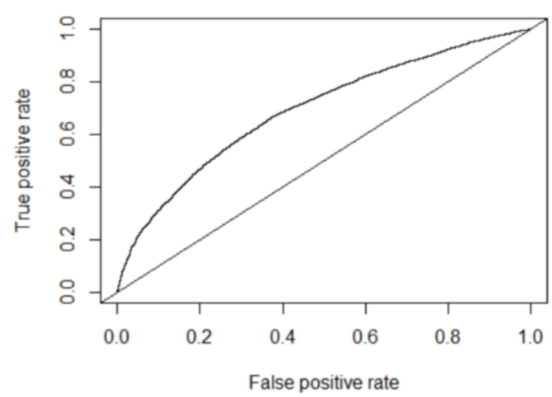

Accuracy = 0.851117429013567
Area under curve: 0.6968744

Plot of GLMNET using Ridge:



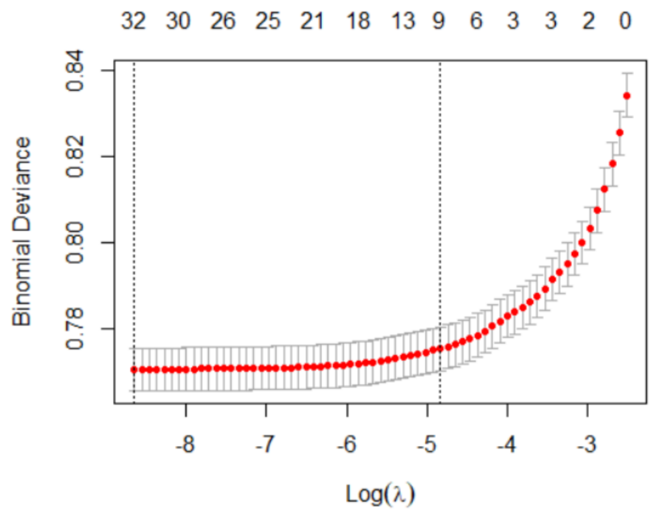

GLMNET Using LASSO:




Accuracy = 0.851009464857667
AUC = 0.6938379

Plot of GLMNET using LASSO:



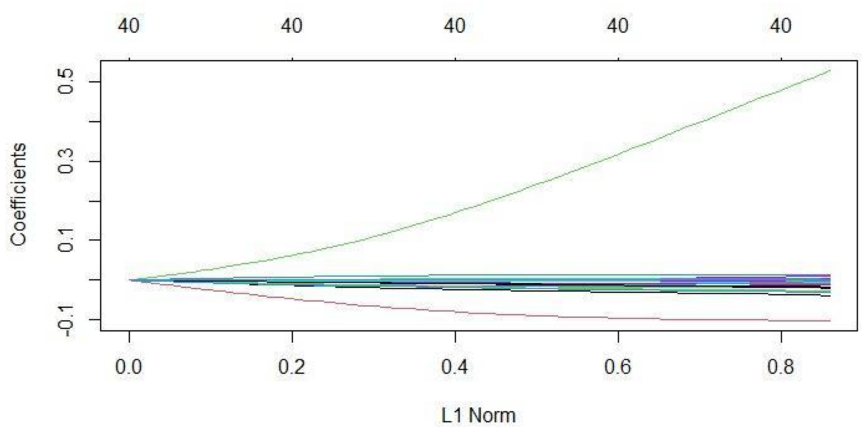

Variable selection can be handled by penalizing the magnitude of coefficients of features along with minimizing the error between predicted and actual observations. This is done by using Ridge and Lasso.

For Ridge Regression alpha value = 0,

glmDefault_cv_lasso<- cv.glmnet(data.matrix(xD), lcdfTrn$loan_status, family="binomial", alpha = 0)

For Lasso regression alpha value = 1,

glmDefault_cv_lasso<- cv.glmnet(data.matrix(xD), lcdfTrn$loan_status, family="binomial", alpha = 1)




**(b) For the linear model, what is the loss function, and link function you use? (Write the expression for these, and briefly describe).**

**Ans.**

<u>RIDGE</u>:

*Loss function* = OLS + alpha * summation (squared coefficient values)

loss function is sum of squared errors

L2 regularization

We can automate this task of finding the optimal lambda value using the cv.glmnet() function. The optimal lambda value comes out to be -2.1 and will be used to build the ridge regression model.

<u>LASSO:</u>

*Loss function* = OLS + alpha * summation (absolute values of the magnitude of the coefficients)

L1 norm -- can give sparse models

In the above function, alpha is the penalty parameter we need to select. Using an l1-norm constraint forces some weight values to zero to allow other coefficients to take non-zero values.

The first step to build a lasso model is to find the optimal lambda value using the code below. For lasso regression, the alpha value is 1. The output is the best cross-validated lambda, which comes out to be -4.9.

The loss function defined in the 2-class classification problem is known as Cross-Entropy loss function.

$$y^t \in 0, 1 \qquad y^t \sim Ber(y^t; \mu^t) \qquad 0 \leq \mu^t \leq 1$$

$$p(y|X, w) = \prod_{t=1}^{N} (\mu^t)^{y^t} (1 - \mu^t)^{(1-y^t)}$$

$$L = -\log p(y|X, w) = \sum_t -y^t \log \mu^t - (1 - y^t) \log(1 - \mu^t)$$

Cross-Entropy Loss Function

**Link function** = ln(μi/(1−μi))

---

**(c) Compare performance of models with that of random forests (from last assignment) and gradient boosted tree models.**

**Ans.** By comparing the performance of models with random forests, GBM performed better in terms of accuracy and area under curve.

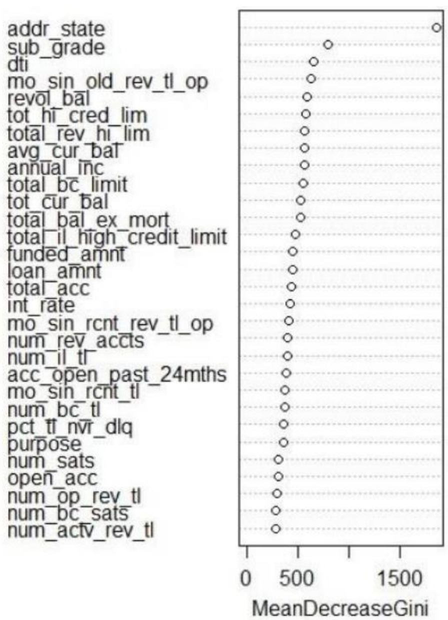
---

**(d) Examine which variables are found to be important by the best models from the different methods, and comment on similarities, difference. What do you conclude?**

**Ans.** For all the best models with the best different methods, most of the variables which are important are the same. But, the order of their importance differed.

For Random Forest, based on mean decrease Gini:

Addr_state was most important, followed by sub grade, dti, mo_sin_old_rev_tl_op, revol bal, tot_hi_cred_lim, total_rev_hi_lim, avg_cur_bal, annual_inc, total_bc_limit, tot_cur_bal, total_bal_ex_mort and so on upto num_actv_rev_tl.




| var | rel.inf |
|---|---|
| sub_grade | 31.91082033 |
| addr_state | 31.81100925 |
| int_rate | 7.72148760 |
| dti | 2.30672781 |
| loan_amnt | 2.09651176 |
| mo_sin_old_rev_tl_op | 1.92037631 |
| acc_open_past_24mths | 1.82493941 |
| total_bc_limit | 1.51468447 |
| total_bal_ex_mort | 1.44928792 |
| revol_bal | 1.43803822 |
| avg_cur_bal | 1.40294668 |
| total_rev_hi_lim | 1.37367196 |
| annual_inc | 1.35306004 |
| tot_cur_bal | 1.21878019 |
| tot_hi_cred_lim | 1.17070015 |
| total_il_high_credit_limit | 1.04021441 |
| purpose | 1.00434741 |
| pct_tl_nvr_dlq | 0.73163135 |
| mo_sin_rcnt_rev_tl_op | 0.72264838 |
| total_acc | 0.70412777 |
| num_rev_accts | 0.58682243 |
| num_bc_tl | 0.51648829 |
| num_tl_op_past_12m | 0.49943079 |
| mort_acc | 0.45875864 |
| mo_sin_rcnt_tl | 0.43478643 |
| num_rev_tl_bal_gt_0 | 0.40627906 |
| num_bc_sats | 0.39660161 |
| open_acc | 0.39582334 |
| num_actv_bc_tl | 0.33811967 |
| num_actv_rev_tl | 0.31001555 |
| num_il_tl | 0.26820116 |
| home_ownership | 0.23934229 |
| num_op_rev_tl | 0.23447582 |

**RANDOM FOREST** **GBM Using Under Sampling Data**

| Label | Value |
|---|---|
| (Intercept) | 2.9448240000 |
| mort_acc | 0.0160469700 |
| mo_sin_rcnt_tl | 0.0033564710 |
| mo_sin_rcnt_rev_tl_op | 0.0014692550 |
| pct_tl_nvr_dlq | 0.0011798440 |
| open_acc | 0.0011718050 |
| num_sats | 0.0010358740 |
| addr_state | 0.0008879191 |
| num_rev_accts | 0.0006525662 |
| total_acc | 0.0005702526 |
| purpose | 0.0005104390 |
| mo_sin_old_rev_tl_op | 0.0002690454 |
| total_bc_limit | 0.0000025714 |
| avg_cur_bal | 0.0000015161 |
| total_rev_hi_lim | 0.0000010336 |
| annual_inc | 0.0000002686 |
| tot_hi_cred_lim | 0.0000002091 |
| tot_cur_bal | 0.0000001845 |
| total_bal_ex_mort | 0.0000001284 |
| revol_bal | 0.0000000278 |
| loan_amnt | -0.0000026857 |
| funded_amnt | -0.0000026873 |
| num_il_tl | -0.0001966692 |
| num_op_rev_tl | -0.0012443990 |
| num_bc_tl | -0.0014236930 |
| num_bc_sats | -0.0022205840 |
| num_actv_rev_tl | -0.0073991720 |
| dti | -0.0080257270 |
| num_rev_tl_bal_gt_0 | -0.0090145210 |
| num_actv_bc_tl | -0.0108421700 |
| num_tl_op_past_12m | -0.0244462200 |
| sub_grade | -0.0245549400 |
| acc_open_past_24mths | -0.0260111800 |
| int_rate | -0.0365062900 |
| home_ownership | -0.0500243200 |
| grade | -0.1127102000 |

### GLM Using Ridge

*Similarities and Differences:*

Addr_state, sub grade, dti, mo_sin_old_rev_tl_op, revol bal, tot_hi_cred_lim, total_rev_hi_lim, avg_cur_bal, annual_inc, total_bc_limit, tot_cur_bal, total_bal_ex_mort and so on upto num_actv_rev_tl are important variables in Random Forest and GBM. But, In GLM mort_acc, mo_sin_rcnt_tl, mo_sin_rcnt_rev_tl_op, pct_tl_nvr_dlq, open_acc, num_sats, addr_state

In GLM, loan amount and interest rate are penalized. Whereas, those were two of the most important variables in GBM and Random Forest.

**(e) In developing models above, do you find larger training samples to give better models? Do you find balancing the training data examples across classes to give better models?**

**Ans.** The performance of the model was presented in the table below for all GLM, GLM ridge, GLM lasso, GLM elastic models. The best performance was determined based on the values of accuracy, AUC of ROC curves.

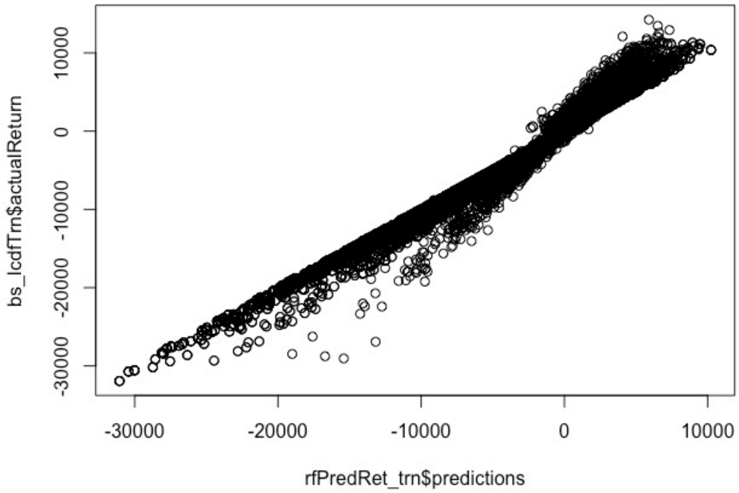
Models which were built using under sampling data performed better when compared to all the other models. So, the models with larger training samples do not always give better models. The results obtained are pertaining to the dataset we have and it varies with the objective of our model. Yes, Balancing training data across classes gives better models. Imbalanced data affects random forests more than GBM and GLMs.

---

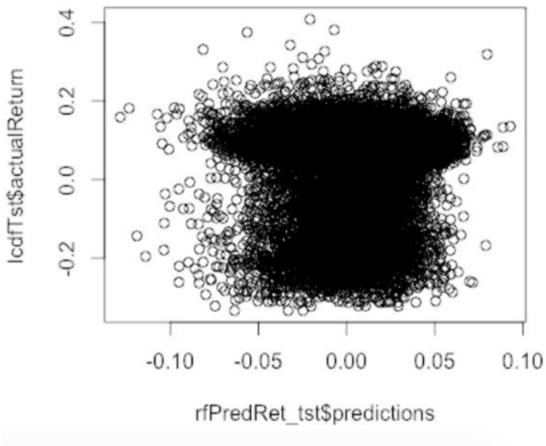
**3. Develop models to identify loans which provide the best returns. Explain how you define returns? Does it include Lending Club's service costs? 2 Develop glm, rf, gbm/xgb models for this. Show how you systematically experiment with different parameters to find the best models. Compare model performance – explain what performance criteria do you use, and why.**

**Ans**.Out of all the models whose performance was studied, it was found that the GLMNET performed best with lowest error rate for training data. Below is the model wise performance for each.

**Training Plot:**



**Test Plot:**

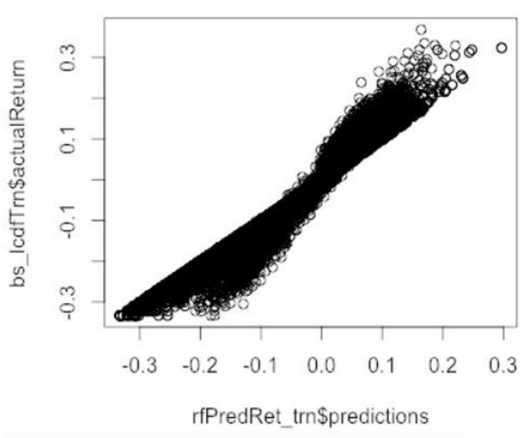



Number of Trees = 100

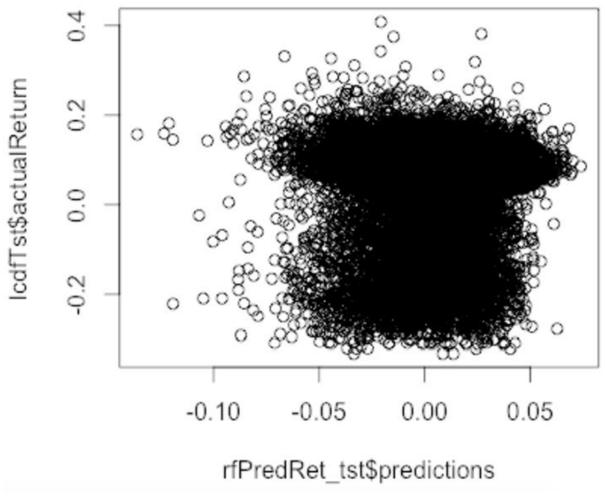
Training error =0.02182912

Test Error = 0.09452704

**Training plot:**



**Test Plot:**

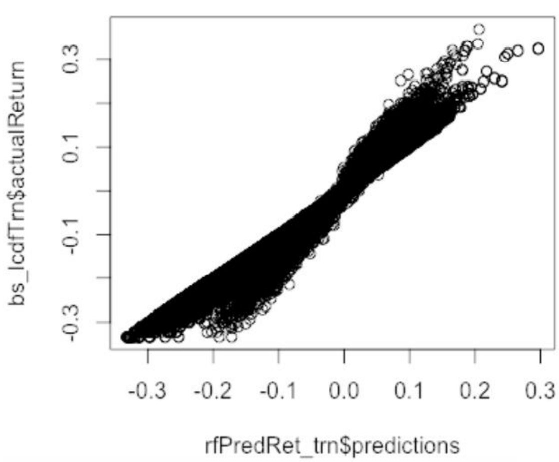



Number of trees = 200

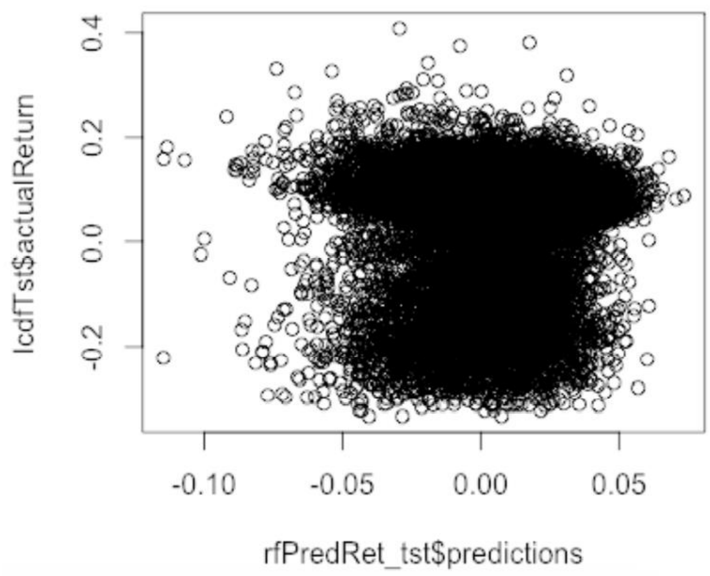
Training error = 0.021434

Test Error = 0.09426395

**Training Plot:**



**Test Plot:**




We conclude that the RM model with 200 trees to be has least test error and highest accuracy.

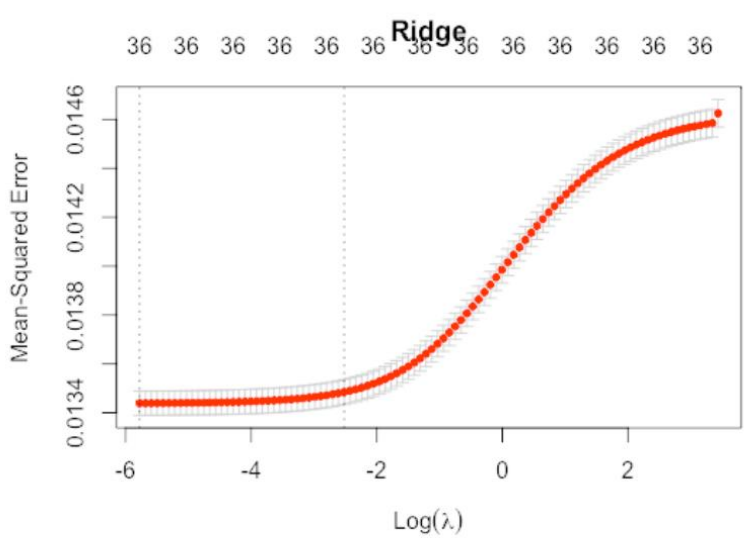
## GLMNET

We tried to build generalized linear model with trying out different values of “alpha”

Starting out with alpha = 0 i.e. Ridge regression

Mean square error = 0.01202288

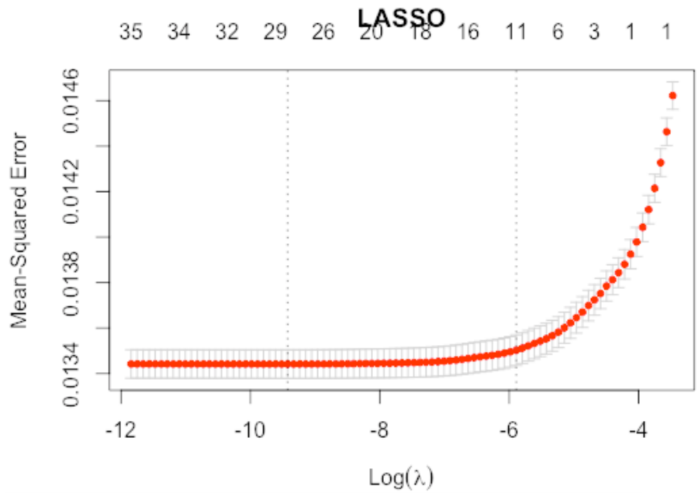
*Plot of mean square error is:*




For alpha = 0.5 i.e. Elastic net Mean square error = 0.01209347

For alpha = 1 i.e. Lasso        Mean square error = 0.01209347

**Plot of mean squared error : 0.01211678**




**So my highest accuracy model is “Ridge” with least MSE.**

**<u>Gradient boosting method</u>**

**We performed a grid search for hyper parameter tuning with 500 trees.**

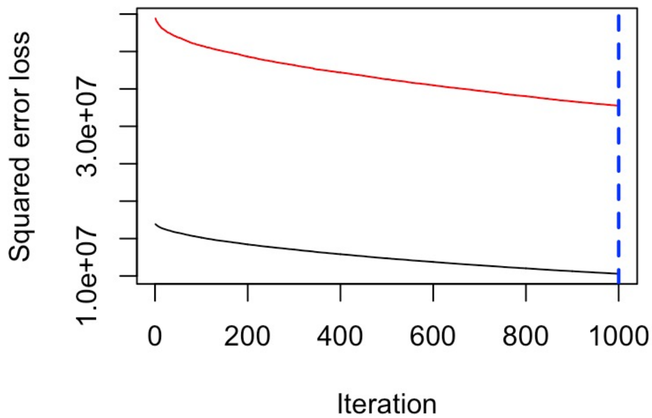
**Training error for the model was :**

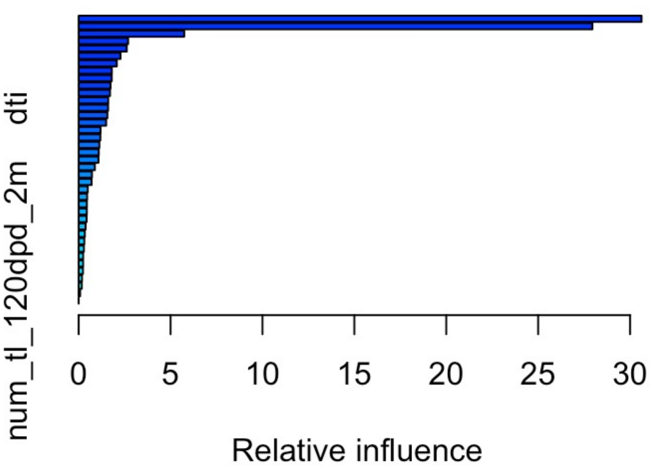
```
error <- sqrt(mean((gbPredRet_trn- bs_lcdfTrn$actualReturn)^2))
> error
[1] 0.1137398
```

**Test error for the model was:**

```
error_tst <- sqrt(mean((gbPredRet_tst- lcdfTst$actualReturn)^2))
> error_tst
[1] 0.08973837
```

**Performance plot for GBM:**



```
> summary(gbm_paramTune)
                                                var     rel.inf
addr_state                               addr_state 30.62544037
sub_grade                                 sub_grade 27.95179956
annual_inc                               annual_inc  5.74254655
acc_open_past_24mths           acc_open_past_24mths  2.69086877
revol_bal                                 revol_bal  2.61770478
total_bal_ex_mort                 total_bal_ex_mort  2.27467993
bc_open_to_buy                       bc_open_to_buy  2.07526714
tot_hi_cred_lim                     tot_hi_cred_lim  1.80513180
total_bc_limit                       total_bc_limit  1.80026566
total_rev_hi_lim                   total_rev_hi_lim  1.73236672
purpose                                     purpose  1.70694157
mo_sin_old_rev_tl_op           mo_sin_old_rev_tl_op  1.60879852
dti                                             dti  1.60180613
revol_util                               revol_util  1.56046142
total_il_high_credit_limit total_il_high_credit_limit  1.49611721
avg_cur_bal                             avg_cur_bal  1.19131663
mths_since_recent_bc           mths_since_recent_bc  1.17916052
bc_util                                     bc_util  1.12540049
total_acc                                 total_acc  1.09275728
tot_cur_bal                             tot_cur_bal  1.07742995
int_rate                                   int_rate  0.88232392
num_bc_tl                                 num_bc_tl  0.72359706
num_il_tl                                 num_il_tl  0.70294777
num_rev_accts                         num_rev_accts  0.49450210
mo_sin_rcnt_rev_tl_op         mo_sin_rcnt_rev_tl_op  0.46094296
num_tl_op_past_12m               num_tl_op_past_12m  0.44649566
pct_tl_nvr_dlq                       pct_tl_nvr_dlq  0.44310162
mo_sin_rcnt_tl                       mo_sin_rcnt_tl  0.43216962
mort_acc                                   mort_acc  0.37704537
open_acc                                   open_acc  0.33499090
num_sats                                   num_sats  0.30742389
num_actv_rev_tl                     num_actv_rev_tl  0.28493975
num_op_rev_tl                         num_op_rev_tl  0.24469882
num_actv_bc_tl                       num_actv_bc_tl  0.24274424
home_ownership                       home_ownership  0.23408040
num_bc_sats                             num_bc_sats  0.18177685
num_rev_tl_bal_gt_0             num_rev_tl_bal_gt_0  0.16980075
grade                                         grade  0.08015736
num_tl_120dpd_2m                   num_tl_120dpd_2m  0.00000000
```

---

**4. Considering results from Questions 1 and 2 above – that is, considering the best model for predicting loan-status and that for predicting loan returns -- how would you select loans for investment? There can be multiple approaches for combining information from the two models - describe your approach and show performance. How does performance here compare with use of single models?**

**Ans.**

Decile performances by each model

### 1. Decile performance in RANDOM FOREST

# A tibble: 10 x 14

| | tile | count | avgpredRet | numDefaults | avgActRet | minRet | maxRet | avgTer | totA | totB | totC | totD | totE | totF |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | <int> | <int> | <dbl> | <int> | <dbl> | <dbl> | <dbl> | <dbl> | <int> | <int> | <int> | <int> | <int> | <int> |
| 1 | 1 | 2779 | 0.0394 | 221 | 0.0556 | -0.322 | 0.257 | 2.13 | 738 | 1249 | 600 | 167 | 24 | 1 |
| 2 | 2 | 2779 | 0.0297 | 237 | 0.0475 | -0.313 | 0.260 | 2.21 | 1160 | 1083 | 392 | 120 | 23 | 1 |
| 3 | 3 | 2779 | 0.0244 | 283 | 0.0449 | -0.300 | 0.221 | 2.27 | 1162 | 1023 | 438 | 136 | 19 | 0 |
| 4 | 4 | 2778 | 0.0195 | 279 | 0.0478 | -0.313 | 0.256 | 2.25 | 1040 | 990 | 565 | 152 | 30 | 1 |
| 5 | 5 | 2779 | 0.0145 | 332 | 0.0458 | -0.333 | 0.381 | 2.28 | 887 | 976 | 664 | 210 | 38 | 4 |
| 6 | 6 | 2779 | 0.00925 | 363 | 0.0459 | -0.312 | 0.283 | 2.28 | 708 | 1015 | 727 | 265 | 61 | 3 |
| 7 | 7 | 2778 | 0.00323 | 429 | 0.0463 | -0.310 | 0.319 | 2.29 | 518 | 933 | 917 | 344 | 59 | 5 |
| 8 | 8 | 2779 | -0.00409 | 544 | 0.0439 | -0.322 | 0.286 | 2.28 | 327 | 767 | 1107 | 442 | 124 | 11 |
| 9 | 9 | 2779 | -0.0136 | 637 | 0.0419 | -0.322 | 0.288 | 2.32 | 172 | 570 | 1213 | 618 | 179 | 22 |
| 10 | 10 | 2778 | -0.0337 | 802 | 0.0367 | -0.333 | 0.408 | 2.34 | 61 | 248 | 1022 | 898 | 425 | 108 |

We observe that in the first 7 deciles the average predicted return is positive and in the top 4 deciles the difference between the average prediction and average actual is less. The number of defaults are not significantly differing as we move down the decile and seems like grade A and B have good investment opportunities as expected and up to some extent grade C as well

### 2. Decile performance with GLMNET

# A tibble: 10 x 14

| | tile | count | avgpredRet | numDefaults | avgActRet | minRet | maxRet | avgTer | totA | totB | totC | totD | totE | totF |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | <int> | <int> | <dbl> | <int> | <dbl> | <dbl> | <dbl> | <dbl> | <int> | <int> | <int> | <int> | <int> | <int> |
| 1 | 1 | 2779 | 0.0259 | 95 | 0.0402 | -0.281 | 0.205 | 2.25 | 2209 | 511 | 53 | 6 | 0 | 0 |
| 2 | 2 | 2779 | 0.00873 | 155 | 0.0438 | -0.283 | 0.216 | 2.28 | 1677 | 942 | 153 | 6 | 1 | 0 |
| 3 | 3 | 2779 | -0.000202 | 218 | 0.0448 | -0.333 | 0.233 | 2.22 | 1189 | 1290 | 276 | 21 | 3 | 0 |
| 4 | 4 | 2778 | -0.00768 | 279 | 0.0472 | -0.309 | 0.232 | 2.25 | 812 | 1410 | 509 | 43 | 4 | 0 |
| 5 | 5 | 2779 | -0.0146 | 310 | 0.0509 | -0.322 | 0.272 | 2.24 | 480 | 1430 | 765 | 96 | 7 | 1 |
| 6 | 6 | 2779 | -0.0211 | 355 | 0.0542 | -0.333 | 0.308 | 2.21 | 244 | 1240 | 1096 | 188 | 11 | 0 |
| 7 | 7 | 2778 | -0.0282 | 467 | 0.0514 | -0.333 | 0.246 | 2.25 | 121 | 978 | 1337 | 311 | 30 | 1 |
| 8 | 8 | 2779 | -0.0361 | 596 | 0.0458 | -0.333 | 0.289 | 2.28 | 43 | 629 | 1494 | 541 | 68 | 4 |
| 9 | 9 | 2779 | -0.0462 | 694 | 0.0449 | -0.323 | 0.319 | 2.32 | 5 | 301 | 1330 | 929 | 204 | 10 |
| 10 | 10 | 2778 | -0.0661 | 914 | 0.0378 | -0.333 | 0.502 | 2.32 | 5 | 62 | 679 | 1181 | 706 | 127 |

If we go GLMNET for decile prediction we see that only top 2 deciles give the positive average predicted return with low number of defaults with majority chunk of grade A in top 2 deciles, so I would invest in them

**3. Decile performance of Gradient boosting Method (GBM):**

# A tibble: 10 x 14

| | tile | count | avgpredRet | numDefaults | avgActRet | minRet | maxRet | avgTer | totA | totB | totC | totD | totE | totF |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | <int> | <int> | <dbl> | <int> | <dbl> | <dbl> | <dbl> | <dbl> | <int> | <int> | <int> | <int> | <int> | <int> |
| 1 | 1 | 2779 | 0.0233 | 170 | 0.0493 | -0.313 | 0.248 | 2.13 | 1348 | 999 | 357 | 75 | 0 | 0 |
| 2 | 2 | 2779 | 0.0226 | 270 | 0.0472 | -0.313 | 0.272 | 2.22 | 999 | 1139 | 481 | 158 | 1 | 0 |
| 3 | 3 | 2779 | 0.0212 | 280 | 0.0473 | -0.333 | 0.246 | 2.20 | 1023 | 1016 | 522 | 193 | 25 | 0 |
| 4 | 4 | 2778 | 0.0206 | 238 | 0.0439 | -0.323 | 0.255 | 2.28 | 1127 | 1406 | 210 | 27 | 7 | 1 |
| 5 | 5 | 2779 | 0.0201 | 281 | 0.0431 | -0.323 | 0.233 | 2.30 | 999 | 1376 | 314 | 77 | 13 | 0 |
| 6 | 6 | 2779 | 0.0196 | 358 | 0.0425 | -0.314 | 0.287 | 2.33 | 759 | 1352 | 523 | 124 | 20 | 0 |
| 7 | 7 | 2778 | 0.0188 | 458 | 0.0437 | -0.323 | 0.335 | 2.30 | 405 | 929 | 1073 | 325 | 41 | 4 |
| 8 | 8 | 2779 | 0.0175 | 563 | 0.0439 | -0.322 | 0.305 | 2.27 | 210 | 554 | 1408 | 540 | 58 | 7 |
| 9 | 9 | 2779 | 0.0152 | 706 | 0.0454 | -0.333 | 0.375 | 2.29 | 0 | 0 | 1652 | 890 | 218 | 16 |
| 10 | 10 | 2778 | 0.0109 | 863 | 0.0408 | -0.333 | 0.502 | 2.35 | 0 | 0 | 1178 | 883 | 588 | 111 |

The most stable and proportionate decile performance is given by GBM. Other than obvious investing in grade A and grade B we can also go with grade C in the 6th decile. As always we observe that the maximum returns are for the bottom decile because of high risk as they are more prone to default and hence high risk with high returns.

*So, I would choose to go with GBM decile performance as it gives me more proportionate observations to choose my investment from and each decile has positive average prediction.*

---

**5. As seen in data summaries and your work in the first assignment, higher grade loans are less likely to default, but also carry lower interest rates; many lower grad loans are fully paid, and these can yield higher returns. One approach may be to focus on lower grade loans (C and below) and try to identify those which are likely to be paid off. Develop models from the data on lower grade loans and check if this can provide an effective investment approach – for this, you can use one of the methods (glm, rf, or gbm/xgb) which you find to give superior performance from earlier questions. Can this provide a useful approach for investment? Compare performance with that in Question 4.**

**Ans.**

Grade A and Grade B definitely are safe investments with low risk.

We trained our model on the lower loan grades, i.e. all the loans graded lower than B.

I checked the decile performance from a **Random Forest** trained model and found the results shown below:

# A tibble: 10 x 14

| | tile | count | avgpredRet | numDefaults | avgActRet | minRet | maxRet | avgTer | totA | totB | totC | totD | totE | totF |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | <int> | <int> | <dbl> | <int> | <dbl> | <dbl> | <dbl> | <dbl> | <int> | <int> | <int> | <int> | <int> | <int> |
| 1 | 1 | 6484 | 0.0970 | 159 | 0.122 | 0.0744 | 0.331 | 1.43 | 45 | 914 | 2633 | 2043 | 716 | 101 |
| 2 | 2 | 6484 | 0.0653 | 273 | 0.0855 | 0.0563 | 0.196 | 2.04 | 343 | 2168 | 2896 | 858 | 195 | 23 |
| 3 | 3 | 6484 | 0.0522 | 346 | 0.0696 | 0.0435 | 0.154 | 2.24 | 1073 | 3106 | 1766 | 454 | 80 | 4 |
| 4 | 4 | 6483 | 0.0423 | 411 | 0.0566 | 0.0329 | 0.132 | 2.46 | 2207 | 2861 | 1112 | 245 | 56 | 2 |
| 5 | 5 | 6484 | 0.0329 | 612 | 0.0441 | 0.0164 | 0.123 | 2.67 | 3919 | 1696 | 636 | 168 | 63 | 2 |
| 6 | 6 | 6484 | 0.00234 | 4763 | 0.00448 | -0.0914 | 0.115 | 2.87 | 1689 | 1679 | 1851 | 893 | 311 | 50 |
| 7 | 7 | 6483 | -0.0608 | 6483 | -0.0680 | -0.206 | -0.0257 | 3 | 812 | 1872 | 2201 | 1135 | 379 | 71 |
| 8 | 8 | 6484 | -0.120 | 6484 | -0.133 | -0.288 | -0.0909 | 3 | 730 | 1580 | 2385 | 1225 | 459 | 100 |
| 9 | 9 | 6484 | -0.174 | 6484 | -0.185 | -0.313 | -0.148 | 3 | 486 | 1472 | 2416 | 1467 | 541 | 91 |
| 10 | 10 | 6483 | -0.237 | 6483 | -0.246 | -0.333 | -0.201 | 3 | 364 | 1353 | 2306 | 1650 | 608 | 162 |

We see that the bottom 3 deciles are almost all defaulters and we observe that the average prediction is more than the average of actual. Most C and D grade are loans are pushed to the top decile which was expected and the major relative proportion of grade F are in top decile as well so maybe we can focus on these people more for high reward.

But when I checked the decile performance using GLMNET I could not see major difference as all the top loans consisted of Grade A and Grade B.

# A tibble: 10 x 14

| | tile | count | avgpredRet | numDefaults | avgActRet | minRet | maxRet | avgTer | totA | totB | totC | totD | totE | totF |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | <int> | <int> | <dbl> | <int> | <dbl> | <dbl> | <dbl> | <dbl> | <int> | <int> | <int> | <int> | <int> | <int> |
| 1 | 1 | 6484 | 0.0205 | 1238 | 0.0175 | -0.323 | 0.190 | 2.37 | 4730 | 1520 | 223 | 11 | 0 | 0 |
| 2 | 2 | 6484 | 0.00331 | 2046 | 0.00343 | -0.313 | 0.208 | 2.45 | 3091 | 2820 | 525 | 48 | 0 | 0 |
| 3 | 3 | 6484 | -0.00627 | 2349 | -0.00236 | -0.333 | 0.249 | 2.48 | 1972 | 3319 | 1067 | 126 | 0 | 0 |
| 4 | 4 | 6483 | -0.0141 | 2816 | -0.0112 | -0.322 | 0.238 | 2.51 | 1012 | 3357 | 1864 | 230 | 20 | 0 |
| 5 | 5 | 6484 | -0.0212 | 3173 | -0.0179 | -0.313 | 0.234 | 2.55 | 495 | 2779 | 2706 | 479 | 25 | 0 |
| 6 | 6 | 6484 | -0.0282 | 3496 | -0.0251 | -0.333 | 0.277 | 2.60 | 219 | 2135 | 3317 | 759 | 49 | 5 |
| 7 | 7 | 6483 | -0.0352 | 3882 | -0.0394 | -0.333 | 0.318 | 2.63 | 115 | 1463 | 3450 | 1298 | 149 | 8 |
| 8 | 8 | 6484 | -0.0430 | 4120 | -0.0462 | -0.333 | 0.270 | 2.66 | 27 | 922 | 3296 | 1860 | 368 | 11 |
| 9 | 9 | 6484 | -0.0528 | 4466 | -0.0576 | -0.333 | 0.317 | 2.71 | 1 | 312 | 2676 | 2607 | 845 | 40 |
| 10 | 10 | 6483 | -0.0724 | 4912 | -0.0707 | -0.333 | 0.331 | 2.75 | 6 | 74 | 1078 | 2720 | 1952 | 542 |

**6. Considering all your results, which approach(s) would you recommend for investing in LC loans? Explain your rationale.**

**Ans.**

From the standpoint of an investor, we can see that they are limited by both money and time. We should employ alternative models that suit their risk appetite based on their risk aversion.

As a result of the above analysis,

We've noticed that GBM appears to perform better, although it demands a lot of computing power.

Based on their risk-return appetite, there are three sorts of investors: Low, Medium, and High.

In terms of Low,

The GBM loan status model will be used to recommend outcomes. Because it provides us with the best loans possible.

There's a good chance you'll get paid. Despite the lower returns on these loans, the investors in this group are confident.

Higher returns aren't an issue.

In terms of Medium,

We will suggest the combination model that we developed in response to question 5. The default rate increases somewhat, but the returns increase dramatically. This combination will benefit this type of investment the most.

of potential investors

In terms of High,

We shall suggest a model based on lower-quality loans. We've noticed that the loans in the top tile of this one yields the best results, when compared to the other ways. This category's investors prioritize high returns, therefore this model is ideal for them.