**IDS 572 Assignment 2 – Models for investment decisions in LendingClub loans**

*Team Members: Gabriella Sussman, Souyma Vastrad, George Vlahos, Megan Petraitis*

**1. (a) Develop linear (glm) models to predict loan_status. Experiment with different parameter values, and identify which gives 'best' performance. 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.**

As RMSE (root mean square error) would be more applicable for regression and AUC (area under the curve) for classification with two classes, we decide to use AUC in order to determine the performance of our models around predicting charged off and fully paid loans.

| Inputs Used | AUC | Total # Variables | Top 5 Important Variables by Coef Value |
|---|---|---|---|
| alpha=1, lambda=lambda.1se | 0.6915161 | 19 | grade, sub_grade, home_ownership, verification_status, acc_open_past_24mths |
| alpha=1, lambda=lambda.min | 0.6931723 | 47 | Num_tl_120dpd_2m, collections_12_mths_ex_med, grade, num_tl_30dpd, pub_rec_bankruptcies |
| alpha=0.5, lambda=lambda.1se | 0.6912075 | 19 | Grade, home_ownership, int_rate, verification_status, acc_open_past_24mths |
| alpha=0.5, lambda=lambda.min | 0.6931959 | 48 | Num_tl_120dpd_2m, grade, collections_12_mths_ex_med, num_tl_30dpd, pub_rec_bankruptcies |
| alpha=0, lambda=lambda.1se | 0.689295 | 50 | Grade, num_tl_120dpd_2m, verification_status, pub_rec_bankruptcies, home_ownership |
| alpha=0, lambda=lambda.min | 0.6932065 | 50 | num_tl_120dpd_2m, grade, collections_12_mths_ex_med, num_tl_30dpd, pub_rec_bankruptcies |

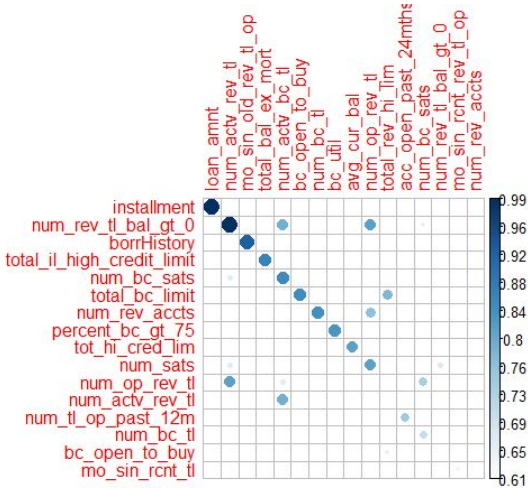
Since the AUCs for almost all models with different variations of parameters we tried are relatively similar, we decided to choose the cross validated glm model with Lasso regression (alpha=1) as it is better suited for variable selection compared to Ridge, which can only shrink coefficients close to zero, meaning we would still have many variables in our model and therefore it is not as effective in variable selection, since a simpler model is usually preferred. When comparing the use of lambda.min to lambda.1se in the iterations using Lasso, we noticed that the number of variables used in lambda.1se are much less, so using lambda.1se is likely a better choice as well to prevent overfitting and to achieve a simpler model.

***GLM Model (lasso + lambda.1se):***

loan_status~grade + sub_grade + home_ownership + verification_status + acc_open_past_24mths +

num_actv_bc_tl + dti + num_rev_tl_bal_gt_0 + mort_acc + emp_length + percent_bc_gt_75 + purpose + mths_since_recent_inq + total_bc_limit + tot_hi_cred_lim + bc_open_to_buy + total_rev_hi_lim

We decide to remove sub_grade from the variable list as we feel it may cause multicollinearity issues with the model since sub_grade is derived from grade. Multicollinearity makes it hard to interpret the coefficients, and it reduces the power of a model to identify independent variables that are statistically significant, and can also create redundant information, skewing the results in a regression model. Another reason for us to remove it as removing it helps reduce the run time for the step() function.




To reduce the possibility of multicollinearity in our model, we also tested the correlation between the variables and selected those with correlation threshold > 0.5. We cross referenced the variables found in the correlation plot with the variables from our glm model and found that:

- num_actv_bc_tl, num_rev_tl_bal_gt_0 = 0.7914067
- total_bc_limit, bc_open_to_buy = 0.8449024
- total_bc_limit , total_rev_hi_lim = 0.7825784

So we will remove these 5 variables as well before proceeding with the step function to prevent multicollinearity issues.

For the next step, we experiment with stepwise selection, using backwards and forwards selection to try to improve our model by reducing variables that are not important.

***GLM model before and after forwards selection:***

```
Start:  AIC=33737.79
lcdfTrn$loan_status ~ 1

                         Df Deviance   AIC
+ grade                   6    31864 31878
+ tot_hi_cred_lim         1    33338 33342
+ acc_open_past_24mths    1    33372 33376
+ dti                     1    33378 33382
+ mort_acc                1    33495 33499
+ home_ownership          2    33503 33509
+ verification_status     2    33546 33552
+ percent_bc_gt_75        1    33571 33575
+ mths_since_recent_inq   1    33602 33606
+ emp_length             11    33623 33647
+ purpose                10    33641 33663
<none>                         33736 33738
```

```
lcdfTrn$loan_status ~ grade + tot_hi_cred_lim + acc_open_past_24mths +
    dti + verification_status + emp_length + percent_bc_gt_75 +
    home_ownership + mths_since_recent_inq + mort_acc

          Df Deviance   AIC
+ purpose 10    31332 31408
<none>          31355 31411

Step:  AIC=31408.2
lcdfTrn$loan_status ~ grade + tot_hi_cred_lim + acc_open_past_24mths +
    dti + verification_status + emp_length + percent_bc_gt_75 +
    home_ownership + mths_since_recent_inq + mort_acc + purpose
```

After running the forwards selection step function, we find that all the variables are kept in the final model, and the AIC (Akaike Information Criterion) of our model at the start without any variables decreases from 33737.79 to 31408.2. The smaller the AIC, the better the fit of our model, so this suggests that all the variables we decided to keep help to improve the fit of our model.

```
Start:  AIC=31408.2
lcdfTrn$loan_status ~ grade + home_ownership + verification_status +
    acc_open_past_24mths + dti + mort_acc + emp_length + percent_bc_gt_75 +
    purpose + mths_since_recent_inq + tot_hi_cred_lim

                        Df Deviance   AIC
<none>                       31332 31408
- purpose               10    31355 31411
- mort_acc               1    31342 31416
- home_ownership         2    31344 31416
- mths_since_recent_inq  1    31343 31417
- percent_bc_gt_75       1    31351 31425
- verification_status    2    31355 31427
- emp_length            11    31376 31430
- tot_hi_cred_lim        1    31385 31459
- dti                    1    31399 31473
- acc_open_past_24mths   1    31441 31515
- grade                  6    32170 32234
lcdfTrn$loan_status ~ grade + home_ownership + verification_status +
    acc_open_past_24mths + dti + mort_acc + emp_length + percent_bc_gt_75 +
    purpose + mths_since_recent_inq + tot_hi_cred_lim
```

The results from running a backwards selection also show the same results, that none of the variables improve the AIC if dropped.

**(b2) For the linear model, what is the loss function, and link function you use ? (Write the expression for these, and briefly describe).**

In our models we use Log loss (Binary Cross-entropy) as our loss function since we are dealing with binary classification. The cross-entropy loss will increase as the predicted probability diverges from the actual label. This means that predictions with a lower the predicted probability of being a true positive, will be penalized more. The expression for the Log Loss function is shown below:

$$-\frac{1}{N}\sum_{i=1}^{N} y_i.\log(p(y_i) + (1-y_i).\log(1-p(y_i))$$

In our GLMNET model, the family argument is a description of the error distribution and link function to be used in the model. We use family=”binomial” in our model since we are dealing with binary classification. The link function used for the binomial family is the logit function (or log odds).

$$\log(p/(1-p))$$

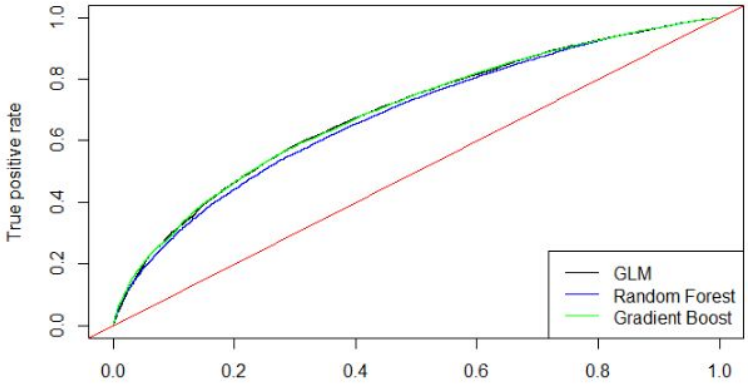
It can also be written as:

$$\log\left(\frac{\hat{y}}{1-\hat{y}}\right) = \log\left(\frac{Pr(y=1|x)}{Pr(y=0|x)}\right) = x^\top\beta + \beta_0.$$

This function creates a map of probability values from [0,1] to [-∞,+∞] and is the inverse of the sigmoid or logistic function, and transforms a continuous value (probability p) in the interval [0,1] to the real line, where it is usually the logarithm of the odds.

**(c) Compare performance of models with that of random forests and gradient boosted tree models (which you developed in your last assignment).**

| Model | AUC |
|---|---|
| Generalized Linear Model with Lasso (GLMNET) | 0.6915161 |
| Random Forest (RF) | 0.6798487 |
| Gradient Boosted (GBM) | 0.692036 |




Comparing the GLM (GLMNET) model we developed previously to a Random Forest Model (with num.trees =200, max.depth = 30) and a Gradient Boosted Model (using xGB), we found that our GLM and GBM models perform the slightly better compared to our Random Forest Model, and out of those two models, our GBM model seems to perform the best.

**(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?**

***RF Important Variables:***

| names <chr> | x <dbl> |
|---|---|
| tot_hi_cred_lim | 1.140473e-02 |
| total_rev_hi_lim | 9.459113e-03 |
| bc_open_to_buy | 8.896835e-03 |
| total_bc_limit | 8.680323e-03 |
| avg_cur_bal | 8.199353e-03 |
| borrHistory | 7.194153e-03 |
| earliest_cr_line | 6.925287e-03 |
| installment | 6.559146e-03 |
| loan_amnt | 5.824868e-03 |
| mo_sin_old_rev_tl_op | 5.799986e-03 |

***GBM Important Variables:***

| Feature <chr> | Gain <dbl> | Cover <dbl> | Frequency <dbl> |
|---|---|---|---|
| int_rate | 0.485015052 | 0.1702714972 | 0.083916084 |
| dti | 0.042099755 | 0.0594795360 | 0.048951049 |
| acc_open_past_24mths | 0.041236905 | 0.0587741715 | 0.041958042 |
| installment | 0.041189706 | 0.0776944636 | 0.076923077 |
| grade.A | 0.028672510 | 0.0265948147 | 0.013986014 |
| bc_open_to_buy | 0.025128585 | 0.0270613402 | 0.055944056 |
| tot_hi_cred_lim | 0.024035944 | 0.0471124880 | 0.055944056 |
| avg_cur_bal | 0.023110633 | 0.0285901810 | 0.041958042 |
| mort_acc | 0.020383335 | 0.0304274215 | 0.027972028 |
| annual_inc | 0.020076607 | 0.0457210983 | 0.041958042 |

The top variables for the RF model differ from those variables in the GBM model among the most significant but they do have some similarities as well. Tot_hi_cred_lim, bc_open_to_buy, and installment are the variables that appear significant in both models.

The most significant variable for the RF model was tot_hi_cred_lim with the second most important being total_rev_hi_lim. Both of these variables relate to the borrower's credit limit.

The most significant variable for the GBM model was int_rate which was about 12 times as much gain as the next most significant variable (dti), cover was almost 3 times as much and frequency was about 2 times difference. These measures indicate just how significant the interest rate (int_rate) is for the GBM model to predict loan_status.

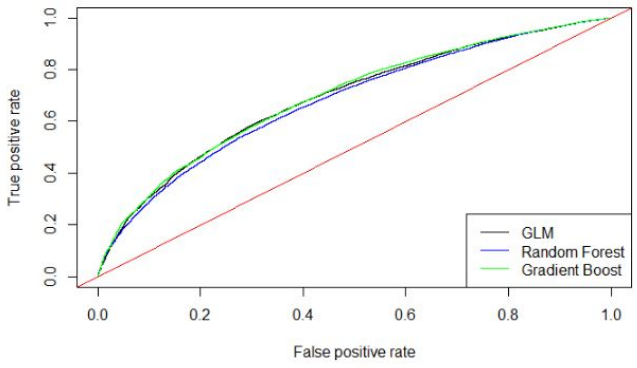
**(e) In developing models above, do you find larger training samples to give better models ? Do you find balancing the training data examples across classes to give better models ?**

When balancing the training and testing set ratios, it is important to find the sweet spot where there is enough data to accurately train the model with, not too much data in the training set that overfitting starts to become a problem. We decided to experiment with splitting the dataset into 70% for the training data and 30% for the testing data. We also try splitting the dataset into 80% for the training data and 20% for the testing data, which is a common split ratio used, stemming from the pareto principle (the law of the vital few).

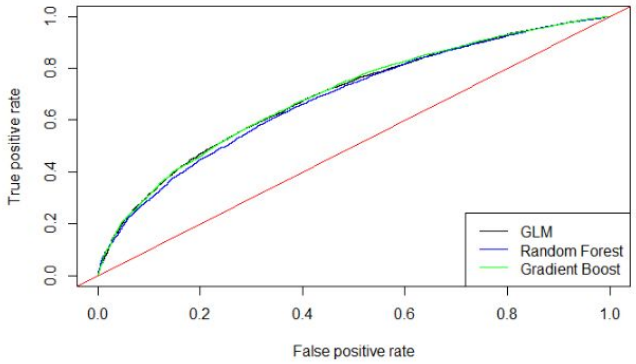
| Model | Data Samples | AUC |
|---|---|---|

| | | |
|---|---|---|
| Generalized Linear Model with Lasso (GLMNET) | 50% Training, 50% Testing | 0.6915161 |
| Generalized Linear Model with Lasso (GLMNET) | 70% Training, 30% Testing | 0.6930341 |
| Generalized Linear Model with Lasso (GLMNET) | 80% Training, 20% Testing | 0.6931981 |
| Random Forest (RF) | 50% Training, 50% Testing | 0.6798487 |
| Random Forest (RF) | 70% Training, 30% Testing | 0.6841938 |
| Random Forest (RF) | 80% Training, 20% Testing | 0.6818889 |
| Gradient Boosted (GBM) | 50% Training, 50% Testing | 0.692036 |
| Gradient Boosted (GBM) | 70% Training, 30% Testing | 0.6951292 |
| Gradient Boosted (GBM) | 80% Training, 20% Testing | 0.6933081 |

*50% Training 50% Testing ROC Curves:*




*70% Training 30% Testing ROC Curves:*

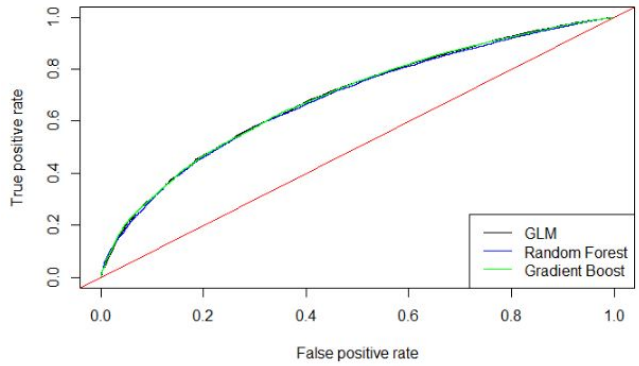



Comparing the models' performance using AUC, we can see that the AUC values don't seem to change by much between the different split ratios. However, we do observe that using 80% of the data in the training set compared to 70% seems to decrease the models' AUC values, which indicates that using a split ratio that is more than 7:3 might start leading to overfitting. From this experiment we find that using a split ratio of 7:3 for the Training and Testing data respectively gave us the most optimal models based on AUC, compared to the other split ratios.

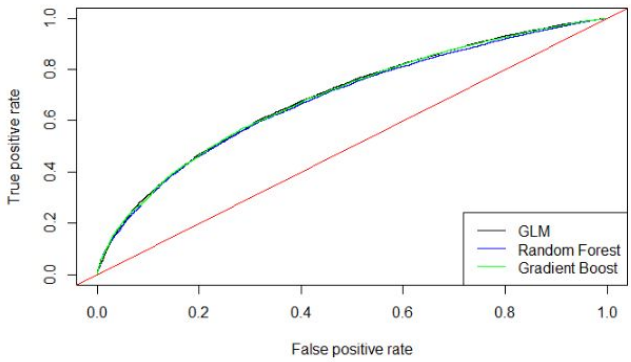
| **Model** | **Data Samples** | **AUC** |
|---|---|---|
| Generalized Linear Model with Lasso (GLMNET) | 50% Training, 50% Testing, with undersampled "Fully Paid" Cases | 0.6912131 |
| Generalized Linear Model with Lasso (GLMNET) | 50% Training, 50% Testing, with oversampled "Charged Off" Cases | 0.6934068 |
| Random Forest (RF) | 50% Training, 50% Testing, with undersampled "Fully Paid" Cases | 0.6860462 |

| | | |
|---|---|---|
| Random Forest (RF) | 50% Training, 50% Testing, with oversampled "Charged Off" Cases | 0.6846884 |
| Gradient Boosted (GBM) | 50% Training, 50% Testing, with undersampled "Fully Paid" Cases | 0.6914551 |
| Gradient Boosted (GBM) | 50% Training, 50% Testing, with oversampled "Charged Off" Cases | 0.692036 |

***Undersampled Data ROC Curves:***




***Oversampled Data ROC Curves:***




From the results, it also seems that oversampling and undersampling the training data does not make that big of an improvement when it comes to improving the AUC of our models.

We also experimented with oversampling "Charged Off" cases and undersampling "Fully Paid" cases to see if balancing our dataset across different classes might help improve our models.

```
set.seed(100)
us_lcdfTrn<-ovun.sample(loan_status~., data = as.data.frame(lcdfTrn), na.action = na.pass, method="under", p=0.5)$data
set.seed(100)
os_lcdfTrn<-ovun.sample(loan_status~., data = as.data.frame(lcdfTrn), na.action = na.pass, method="over", p=0.5)$data
```

***Ratios with Undersampling:***

| loan_status <fctr> | n <int> |
|---|---|
| Fully Paid | 5921 |
| Charged Off | 5930 |

***Ratios with Oversampling:***

| loan_status <fctr> | n <int> |
|---|---|
| Fully Paid | 34581 |
| Charged Off | 34349 |

For our xGB model, we use the "weights" argument when creating the training data matrix, and manually calculate the weights using the formula: Total # charged off loans/Total # Fully Paid loans which gives us 0.1714814, and then we assign every row their respective weight. So for undersampling, fully paid loans are given a 0.1714814 weight and charged off loans are given a weight of 1 and for oversampling, the equation is flipped so that the ratio we use to oversample charged off loans is derived from the Total # Fully Paid loans/Total # Charged Off loans, which gives us a weight of 5.831535 for Charged Off loans and 1 for Fully Paid loans.

```
#Undersampling "Fully Paid" cases
weightpos_us = (40511-sum(colcdfTrn_us$loan_status))/sum(colcdfTrn_us$loan_status)

#if Fully Paid, set column value to undersample weight (#Charged Off/#Fully Paid)
colcdfTrn_us$weights_us <- factor(if_else(colcdfTrn_us$loan_status == 1, weightpos_us , 1))

dxTrn_us<-xgb.DMatrix(subset(dxlcdfTrn_us), label=colcdfTrn_us$loan_status, weight=colcdfTrn_us$weights_us)
dxTst_us<-xgb.DMatrix(subset(dxlcdfTst_us), label=colcdfTst_us)
```

```
#oversampling "Charged Off" cases
weightneg_os = sum(colcdfTrn_os$loan_status)/(40511-sum(colcdfTrn_os$loan_status))

#if Fully Paid, set column value to undersample weight (#Charged Off/#Fully Paid)
weights_os <- factor(if_else(colcdfTrn_os$loan_status == 0, weightneg_os , 1))
dxTrn_os<-xgb.DMatrix(subset(dxlcdfTrn_os), label=colcdfTrn_os$loan_status, weight=colcdfTrn_os$weights_os)
dxTst_os<-xgb.DMatrix(subset(dxlcdfTst_os), label=colcdfTst_os)
```

**2. Develop models to identify loans *which provide the best returns*. Explain how you define returns? Does it include Lending Club's service costs? Develop glm, rf, gbm (xgb) models for this. Show how you systematically experiment with different parameters to find the best models. Compare model performance.**

The variable actual return is used to identify loans providing best returns. Actual returns is computed for loans with term > 0 as below

*Actual return = ((Total payment – Funded amount) / Funded Amount ) * (1 / Actual term) * 100*

This does not include the Lending Clubs' service costs.

Actual returns for loans are computed and Random Forest, GLM and XGB models are developed on the data.

A decile table is built for each model on the training and test data to summarize the model and to help make investment decisions based on loan returns.

a) *Random Forest model*

***Decile table for training dataset:***

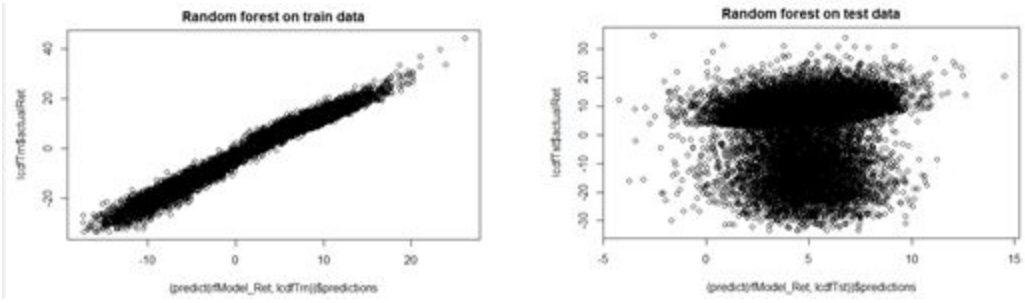
| | tile | count | avgpredRet | numDefaults | avgActRet | minRet | maxRet | avgTer | totA | totB | totC | totD | totE | totF |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 4052 | 11.305109 | 2 | 14.755891 | 9.409531 | 44.359486 | 0.9918577 | 2 | 177 | 1485 | 1580 | 637 | 152 |
| 2 | 2 | 4051 | 8.824818 | 6 | 11.002014 | 7.666301 | 18.170311 | 1.5975765 | 6 | 861 | 1898 | 980 | 284 | 18 |
| 3 | 3 | 4051 | 7.681484 | 20 | 9.308962 | 6.252734 | 16.585456 | 2.0354606 | 27 | 1284 | 1791 | 826 | 115 | 8 |
| 4 | 4 | 4051 | 6.821453 | 29 | 8.137168 | 3.315667 | 14.070252 | 2.2610435 | 156 | 1578 | 1852 | 422 | 40 | 3 |
| 5 | 5 | 4051 | 6.089533 | 47 | 7.234374 | 3.667151 | 12.735680 | 2.3102941 | 548 | 2001 | 1292 | 191 | 18 | 1 |
| 6 | 6 | 4051 | 5.410315 | 79 | 6.411777 | 2.540873 | 12.996053 | 2.2686904 | 1186 | 2073 | 698 | 84 | 8 | 2 |
| 7 | 7 | 4051 | 4.707178 | 91 | 5.368543 | 0.000000 | 10.609322 | 2.3458943 | 2188 | 1572 | 241 | 38 | 10 | 1 |
| 8 | 8 | 4051 | 4.027220 | 111 | 4.403023 | -1.644148 | 9.788733 | 2.6133325 | 3264 | 643 | 96 | 36 | 11 | 1 |
| 9 | 9 | 4051 | 2.271004 | 1486 | 1.411499 | -13.883000 | 8.622093 | 2.8537343 | 2531 | 568 | 515 | 318 | 96 | 22 |
| 10 | 10 | 4051 | -7.046371 | 4051 | -16.591478 | -33.333333 | -2.965819 | 3.0000000 | 378 | 844 | 1429 | 942 | 369 | 74 |

| | tile | count | avgpredRet | numDefaults | avgActRet | minRet | maxRet | avgTer | totA | totB | totC | totD | totE | totF |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 4052 | 7.735604 | 644 | 7.383850 | -32.26256 | 34.12023 | 2.062880 | 0 | 417 | 1791 | 1326 | 407 | 96 |
| 2 | 2 | 4051 | 6.494930 | 657 | 6.318488 | -33.33333 | 30.85477 | 2.154275 | 0 | 1180 | 1796 | 846 | 205 | 21 |
| 3 | 3 | 4051 | 5.854286 | 629 | 5.934725 | -33.33333 | 24.61510 | 2.170722 | 16 | 1655 | 1557 | 646 | 151 | 23 |
| 4 | 4 | 4051 | 5.337186 | 618 | 5.533853 | -33.33333 | 30.76642 | 2.199518 | 223 | 1797 | 1356 | 519 | 141 | 13 |
| 5 | 5 | 4051 | 4.895363 | 568 | 5.283875 | -32.21050 | 27.92745 | 2.198546 | 799 | 1616 | 1087 | 412 | 121 | 13 |
| 6 | 6 | 4051 | 4.499716 | 504 | 4.763726 | -32.23344 | 25.77683 | 2.230519 | 1492 | 1295 | 827 | 337 | 85 | 11 |
| 7 | 7 | 4051 | 4.133310 | 419 | 4.653772 | -29.92276 | 22.03329 | 2.235633 | 2032 | 1000 | 675 | 259 | 75 | 8 |
| 8 | 8 | 4051 | 3.760158 | 458 | 4.261807 | -33.33333 | 27.11550 | 2.284131 | 2208 | 914 | 593 | 235 | 87 | 11 |
| 9 | 9 | 4051 | 3.276617 | 576 | 3.828311 | -32.21033 | 25.90782 | 2.339759 | 2021 | 947 | 655 | 294 | 111 | 22 |

***Decile table for test dataset:***

The predicted returns vary from the actual returns to a large extent in the training data. The highest average return rate predicted from the random forest model is found to be 11% in the training data and 7.7% in the test data. The maximum number of loans with the highest return is in Grade D in training data and Grade C in test data.

Below is a plot of the actual vs predicted values for the training and test datasets:




b) *GLM Model*

***Decile table for training dataset:***

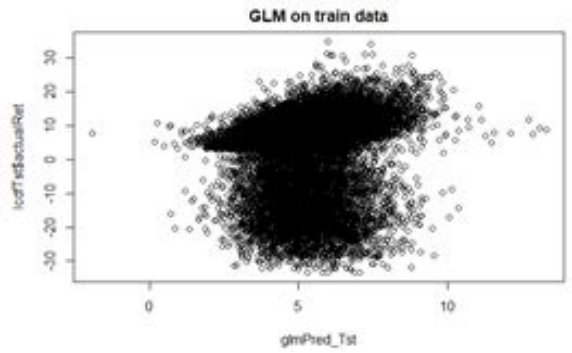
| | tile | count | avgpredRet | numDefaults | avgActRet | minRet | maxRet | avgTer | totA | totB | totC | totD | totE | totF |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 4052 | 7.297472 | 766 | 7.471211 | -32.21442 | 44.35949 | 2.060596 | 33 | 432 | 1142 | 1331 | 813 | 262 |
| 2 | 2 | 4051 | 6.325836 | 660 | 6.595149 | -33.33333 | 29.60077 | 2.095867 | 184 | 892 | 1535 | 1100 | 323 | 16 |
| 3 | 3 | 4051 | 5.881821 | 638 | 5.832656 | -32.18674 | 28.89645 | 2.175910 | 384 | 1122 | 1537 | 835 | 172 | 1 |
| 4 | 4 | 4051 | 5.531067 | 633 | 5.378757 | -32.22487 | 27.09527 | 2.195424 | 592 | 1215 | 1488 | 642 | 111 | 3 |
| 5 | 5 | 4051 | 5.228462 | 576 | 5.191611 | -32.20601 | 39.72585 | 2.192788 | 868 | 1322 | 1312 | 483 | 66 | 0 |
| 6 | 6 | 4051 | 4.943950 | 552 | 4.936330 | -32.31047 | 22.53789 | 2.244902 | 1059 | 1361 | 1221 | 357 | 53 | 0 |
| 7 | 7 | 4051 | 4.661196 | 551 | 4.589476 | -33.33333 | 24.76416 | 2.282744 | 1320 | 1369 | 1065 | 268 | 29 | 0 |
| 8 | 8 | 4051 | 4.350175 | 538 | 4.092797 | -32.25540 | 23.81077 | 2.304935 | 1593 | 1361 | 874 | 212 | 11 | 0 |
| 9 | 9 | 4051 | 3.967699 | 515 | 3.891765 | -30.26886 | 19.86238 | 2.318574 | 1881 | 1340 | 691 | 133 | 6 | 0 |
| 10 | 10 | 4051 | 3.255938 | 493 | 3.463821 | -32.30087 | 22.92906 | 2.405881 | 2372 | 1187 | 432 | 56 | 4 | 0 |

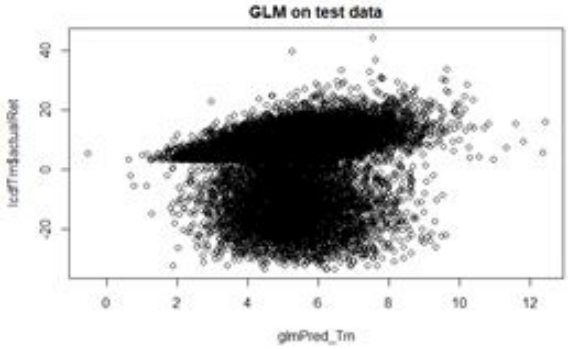
***Decile table for test dataset:***

| | tile | count | avgpredRet | numDefaults | avgActRet | minRet | maxRet | avgTer | totA | totB | totC | totD | totE | totF |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 4052 | 7.289334 | 784 | 7.016260 | -33.33333 | 34.12023 | 2.122515 | 46 | 418 | 1158 | 1329 | 807 | 246 |
| 2 | 2 | 4051 | 6.317799 | 672 | 6.532771 | -32.26256 | 30.76642 | 2.107067 | 175 | 916 | 1548 | 1042 | 350 | 18 |
| 3 | 3 | 4051 | 5.867482 | 615 | 6.077282 | -33.33333 | 34.95244 | 2.148182 | 344 | 1139 | 1563 | 806 | 191 | 8 |
| 4 | 4 | 4051 | 5.521626 | 622 | 5.472252 | -32.24389 | 28.35695 | 2.169769 | 585 | 1267 | 1452 | 647 | 98 | 1 |
| 5 | 5 | 4051 | 5.220061 | 586 | 5.162933 | -33.33333 | 22.26385 | 2.215918 | 861 | 1332 | 1310 | 482 | 63 | 3 |
| 6 | 6 | 4051 | 4.938043 | 558 | 4.901879 | -32.31881 | 24.79576 | 2.225859 | 1057 | 1349 | 1211 | 385 | 47 | 2 |
| 7 | 7 | 4051 | 4.656305 | 493 | 4.809120 | -30.18473 | 22.70519 | 2.250670 | 1311 | 1400 | 1029 | 288 | 23 | 0 |
| 8 | 8 | 4051 | 4.355831 | 532 | 4.298529 | -31.14155 | 22.52704 | 2.278405 | 1536 | 1426 | 866 | 208 | 15 | 0 |
| 9 | 9 | 4051 | 3.974393 | 514 | 3.992377 | -32.21835 | 20.95816 | 2.344768 | 1866 | 1331 | 722 | 124 | 8 | 0 |
| 10 | 10 | 4051 | 3.257356 | 529 | 3.321270 | -31.28430 | 23.04272 | 2.405323 | 2335 | 1220 | 421 | 74 | 1 | 0 |

The actual and predicted returns have very little variation in both datasets which indicates that the GLM might be better than the random forest model. The highest return rate is found in decile 1 with a return

rate of approximately 7% and maximum number of loans are from grade D.




c) XG Boosted model

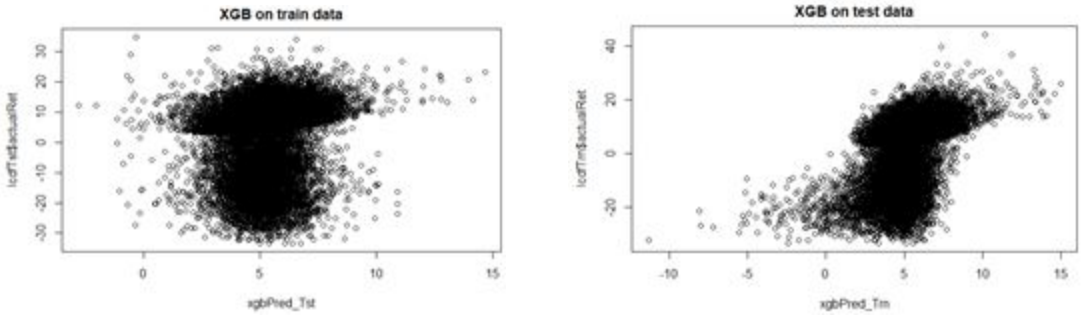
***Decile table on training dataset:***

| | tile | count | avgPredRet | numDefaults | avgActRet | minRet | maxRet | avgTer | totA | totB | totC | totD | totE | totF |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 4052 | 7.381930 | 569 | 8.139492 | -32.19706 | 36.73610 | 2.048681 | 0 | 528 | 1600 | 1215 | 545 | 146 |
| 2 | 2 | 4051 | 6.339847 | 594 | 6.712495 | -33.33333 | 32.90425 | 2.071587 | 0 | 1408 | 1802 | 671 | 143 | 24 |
| 3 | 3 | 4051 | 5.900443 | 685 | 5.904290 | -33.33333 | 27.47198 | 2.170596 | 11 | 1455 | 1654 | 738 | 172 | 18 |
| 4 | 4 | 4051 | 5.578973 | 701 | 5.601560 | -31.08733 | 39.72585 | 2.206048 | 16 | 1483 | 1713 | 691 | 138 | 9 |
| 5 | 5 | 4051 | 5.215938 | 650 | 5.234284 | -32.18389 | 28.85475 | 2.224296 | 149 | 1941 | 1338 | 505 | 99 | 13 |
| 6 | 6 | 4051 | 4.886029 | 587 | 5.070999 | -33.33333 | 24.33994 | 2.219511 | 901 | 1602 | 998 | 432 | 113 | 5 |
| 7 | 7 | 4051 | 4.570588 | 486 | 4.435118 | -31.16387 | 26.22534 | 2.267777 | 1799 | 1113 | 731 | 329 | 78 | 1 |
| 8 | 8 | 4051 | 4.293710 | 452 | 4.064860 | -32.26950 | 24.76416 | 2.263065 | 2468 | 739 | 500 | 276 | 61 | 6 |
| 9 | 9 | 4051 | 3.995256 | 441 | 3.726028 | -32.30087 | 18.02911 | 2.333907 | 2748 | 638 | 396 | 205 | 61 | 3 |
| 10 | 10 | 4051 | 3.237520 | 757 | 2.554282 | -33.33333 | 44.35949 | 2.472155 | 2194 | 694 | 565 | 355 | 178 | 57 |

***Decile table on test dataset:***

| | tile | count | avgPredRet | numDefaults | avgActRet | minRet | maxRet | avgTer | totA | totB | totC | totD | totE | totF |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 4052 | 7.372047 | 645 | 7.352612 | -33.33333 | 34.12023 | 2.095959 | 0 | 539 | 1650 | 1187 | 521 | 131 |
| 2 | 2 | 4051 | 6.335321 | 605 | 6.551642 | -32.26256 | 30.85477 | 2.122950 | 0 | 1403 | 1780 | 693 | 146 | 23 |
| 3 | 3 | 4051 | 5.892035 | 685 | 5.896173 | -33.33333 | 31.17493 | 2.177953 | 9 | 1527 | 1606 | 707 | 168 | 28 |
| 4 | 4 | 4051 | 5.574953 | 653 | 5.814841 | -32.24389 | 30.67082 | 2.207220 | 9 | 1569 | 1655 | 655 | 150 | 13 |
| 5 | 5 | 4051 | 5.215996 | 644 | 5.322622 | -32.31056 | 27.67063 | 2.215139 | 136 | 1952 | 1350 | 495 | 103 | 12 |
| 6 | 6 | 4051 | 4.902281 | 641 | 4.757894 | -32.21050 | 24.13817 | 2.237654 | 783 | 1624 | 1039 | 485 | 116 | 4 |
| 7 | 7 | 4051 | 4.589932 | 544 | 4.242346 | -31.28430 | 30.76642 | 2.233097 | 1778 | 1141 | 719 | 317 | 91 | 4 |
| 8 | 8 | 4051 | 4.304600 | 456 | 4.202359 | -32.23344 | 25.62618 | 2.260298 | 2357 | 785 | 537 | 287 | 83 | 2 |
| 9 | 9 | 4051 | 4.000936 | 392 | 3.969382 | -32.31881 | 31.35167 | 2.301226 | 2810 | 600 | 380 | 201 | 56 | 4 |
| 10 | 10 | 4051 | 3.246789 | 640 | 3.474717 | -32.21835 | 34.95244 | 2.416988 | 2234 | 658 | 564 | 358 | 169 | 57 |

The highest return rate is found to be 7% with the maximum number of loans from Grade C.




The decile table from the three models indicate that the loans with highest average returns are from lower Grades, that is, C and D. To make investment decisions it would be beneficial to look into loans from these grades.

The plots of the models contain outliers and hence cannot be used as a means to compare performance.

We use the Root Mean Squared Error(RMSE) to compare the performance of the models. RMSE provides an absolute measure to compare performance and lower value of RMSE indicates better fit.

Below are the RMSE values of the models on the test dataset:

```
[1] "RMSE for models"
[1] "Random forest= 8.35322737701568"
[1] "Glm= 8.32679684154131"
[1] "Boosted= 8.3039318313173"
```

The XG Boosted model has the least RMSE value and hence has better performance than the GLM and Random Forest models.

**3. 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?**

Since model 1 Predict loan_status are the loans that are less likely to default and model 2 Predict

returns are the loans with the highest predicted returns, combining these two models would provide predictions that are the most advantageous to the investor. This would show which loans have a higher return and low default likelihood. There are two methods we tested for combining these models.

The first method sorts model 2 loans based on model 1's scores after which the investor can select the top percentage of these loans. In other words, we are taking the top decile (d=1) from model 2 and ranking it by model 1 scores. As shown below, the number of defaults is low and the average actual return is relatively high which is what we want to see. The Method 1 model shows a similar trend in numDefaults as our Random Forest model from question 2a). The number of defaults begins in the first decile as a small number relative to mean(numDeafults) and increases to its larger values of numDefaults in the last deciles. Unlike the RF model, the Method 1 model has a consistent avgPredRet value of about 7.5 throughout all deciles. When we look at the avgActRet let us focus on the highest values which are those above 8.0 which are represented in deciles: 1,2,3 and 8. These deciles show average actual return values above the average predicted returns and thus these would be loans to invest in. We knew prior that high grade loans would most likely be paid off and therefore they are safe investments. Now with seeing that decile 8 had more avgActRet than avgPredRet we may want to look closer at lower grade loans for possible investment opportunities.

**Method 1:**

A tibble: 20 x 15

| | tile2 <int> | count <int> | avgPredRet <dbl> | numDefaults <int> | defPerc <dbl> | avgActRet <dbl> | minRet <dbl> | maxRet <dbl> | avgTer <dbl> |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 203 | 7.698554 | 16 | 0.07881773 | 8.212396 | -28.90200 | 26.12436 | 1.866944 |
| 2 | 2 | 203 | 7.571213 | 12 | 0.05911330 | 8.027839 | -23.59789 | 19.39360 | 2.045134 |
| 3 | 3 | 203 | 7.482567 | 8 | 0.03940887 | 8.340481 | -23.04522 | 16.62608 | 2.047022 |
| 4 | 4 | 203 | 7.432906 | 18 | 0.08866995 | 7.753645 | -26.64333 | 25.01373 | 2.009616 |
| 5 | 5 | 203 | 7.607242 | 20 | 0.09852217 | 7.626745 | -33.33333 | 23.09597 | 2.028437 |
| 6 | 6 | 203 | 7.510078 | 18 | 0.08866995 | 7.963621 | -27.57013 | 16.80634 | 2.116109 |
| 7 | 7 | 203 | 7.486166 | 34 | 0.16748768 | 6.773886 | -25.52043 | 19.87501 | 1.964435 |
| 8 | 8 | 203 | 7.474895 | 22 | 0.10837438 | 8.020086 | -23.37621 | 16.95126 | 2.056524 |
| 9 | 9 | 203 | 7.518001 | 32 | 0.15763547 | 6.857447 | -32.19706 | 22.98571 | 2.227167 |
| 10 | 10 | 203 | 7.550907 | 30 | 0.14778325 | 7.550060 | -30.99250 | 25.89642 | 2.039625 |

1-10 of 20 rows | 1-10 of 15 columns Previous 1 2 Next

The second method is sorting the outcome of taking the probability of the loan being fully paid from model 1 and multiplying that by the predicted return from a loan from model 2 which gives the expected returns from a loan. As shown below, this method also provides a minimal amount of defaults and relatively high returns. Since this method is built from the calculation of (predicted Actual Return)*(prob of Fully Paid) it makes sense we would see higher return values for avgPredRet and avgActRet in the top decile. We notice a higher avgActRet in the first decile despite having more loans defaulted which tells us we can have lower grades loans that are less likely to default and can produce higher returns.

This Method 2 model is most similar to the GLM model since both have the first decile showing the highest avgActRet and showing the most numDefaults. Method 2 can also be compared to the XGB model in that a similar trend of the highest values appearing in decile 1 is present with the only difference being the numDefaults values. The avgPredRet for Method 2 has a consistently decreasing prediction throughout our deciles. When we compare this to the avgActRet we notice there are a few deciles (4,6,8,9, and 10) that have their avgActRet above their avgPredRet. This tells us that some lower grades we predicted to have lower returns will actually have a bigger return. Decile 6 specifically has a higher avgActRet than avgPredRet with the lowest number of defaults in our model. This adds to our understanding that some lower grades loans will yield higher returns despite investing in riskier

loans.

**Method 2:**

A tibble: 20 x 15

| | tile2 <int> | count <int> | avgPredRet <dbl> | numDefaults <int> | defPerc <dbl> | avgActRet <dbl> | minRet <dbl> | maxRet <dbl> | avgTer <dbl> |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 203 | 10.499455 | 33 | 0.16256158 | 9.376429 | -28.41507 | 31.12225 | 2.005884 |
| 2 | 2 | 203 | 8.845148 | 30 | 0.14778325 | 7.414482 | -26.40100 | 24.23726 | 2.185931 |
| 3 | 3 | 203 | 8.222139 | 25 | 0.12315271 | 7.567252 | -28.57415 | 19.53378 | 2.077472 |
| 4 | 4 | 203 | 7.969854 | 23 | 0.11330049 | 8.148814 | -32.19706 | 23.10076 | 2.084169 |
| 5 | 5 | 203 | 7.802109 | 25 | 0.12315271 | 7.799954 | -31.99226 | 25.01373 | 1.972355 |
| 6 | 6 | 203 | 7.581124 | 17 | 0.08374384 | 8.282987 | -28.90200 | 19.37669 | 1.953527 |
| 7 | 7 | 203 | 7.469896 | 27 | 0.13300493 | 6.527495 | -30.94880 | 16.81932 | 2.195247 |
| 8 | 8 | 203 | 7.312661 | 23 | 0.11330049 | 8.103115 | -22.19197 | 20.66970 | 2.015277 |
| 9 | 9 | 203 | 7.267184 | 20 | 0.09852217 | 7.723334 | -28.55691 | 19.36493 | 2.027372 |
| 10 | 10 | 203 | 7.246698 | 25 | 0.12315271 | 7.657183 | -26.69889 | 20.51371 | 2.061868 |

Comparing these two methods, method two has a higher number of defaults and therefore the investor is taking more of a risk but the returns are higher. Which method the investor would prefer would depend on their risk tolerance. These methods of combining compared to singular models do a better job at predicting a lower number of defaults and higher returns.

**4. 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. Compare performance of models from different methods (glm, gbm, rf).**

***Glm (on testing data):***

| | tile <int> | count <int> | avgpredRet <dbl> | numDefaults <int> | avgActRet <dbl> | minRet <dbl> | maxRet <dbl> | avgTer <dbl> | totA <int> |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 4052 | 7.418035 | 744 | 7.160875 | -33.33333 | 36.73610 | 2.084214 | 61 |
| 2 | 2 | 4051 | 6.389741 | 598 | 6.893916 | -33.33333 | 34.95244 | 2.096274 | 180 |
| 3 | 3 | 4051 | 5.921328 | 653 | 5.935866 | -32.22487 | 30.85477 | 2.155386 | 356 |
| 4 | 4 | 4051 | 5.560305 | 641 | 5.340004 | -33.33333 | 27.67063 | 2.215165 | 587 |
| 5 | 5 | 4051 | 5.240154 | 668 | 4.903753 | -31.15244 | 27.28570 | 2.201900 | 833 |
| 6 | 6 | 4051 | 4.943885 | 553 | 4.990010 | -31.09203 | 22.53789 | 2.252123 | 1009 |
| 7 | 7 | 4051 | 4.640008 | 526 | 4.634079 | -32.31047 | 24.79576 | 2.274456 | 1306 |
| 8 | 8 | 4051 | 4.299642 | 509 | 4.351996 | -33.33333 | 23.81077 | 2.286673 | 1623 |
| 9 | 9 | 4051 | 3.893032 | 518 | 3.920330 | -32.21033 | 20.04162 | 2.303087 | 1917 |
| 10 | 10 | 4051 | 3.111392 | 487 | 3.503776 | -32.30087 | 22.92906 | 2.372789 | 2346 |

***xGB (on testing data):***

| | tile <int> | count <int> | avgPredRet <dbl> | numDefaults <int> | avgActRet <dbl> | minRet <dbl> | maxRet <dbl> | avgTer <dbl> | totA <int> |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 4052 | 7.560758 | 604 | 7.599430 | -33.33333 | 31.12225 | 2.070471 | 0 |
| 2 | 2 | 4051 | 6.444437 | 652 | 6.459968 | -33.33333 | 36.73610 | 2.125679 | 1 |
| 3 | 3 | 4051 | 5.989065 | 679 | 5.999518 | -33.33333 | 30.85477 | 2.167484 | 6 |
| 4 | 4 | 4051 | 5.605662 | 683 | 5.712764 | -32.18674 | 27.28570 | 2.217850 | 21 |
| 5 | 5 | 4051 | 5.203553 | 721 | 5.007994 | -31.21556 | 27.11550 | 2.227197 | 304 |
| 6 | 6 | 4051 | 4.835327 | 572 | 4.793697 | -32.31056 | 25.39731 | 2.239593 | 1155 |
| 7 | 7 | 4051 | 4.528929 | 579 | 4.388309 | -33.33333 | 27.09527 | 2.271418 | 1435 |
| 8 | 8 | 4051 | 4.214379 | 412 | 4.434105 | -32.26950 | 34.12023 | 2.238404 | 2195 |
| 9 | 9 | 4051 | 3.896435 | 369 | 3.904166 | -30.26886 | 33.38393 | 2.269204 | 2788 |
| 10 | 10 | 4051 | 3.139215 | 626 | 3.334543 | -32.31047 | 34.95244 | 2.414771 | 2313 |

***RF (on testing data using only lower graded loans, C to F):***

| | tile <int> | count <int> | avgSc <dbl> | numDefaults <int> | avgActRet <dbl> | minRet <dbl> | maxRet <dbl> | avgTer <dbl> | totC <int> |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 1870 | 0.8924224 | 246 | 6.862044 | -31.09203 | 23.87927 | 2.115837 | 1778 |
| 2 | 2 | 1870 | 0.8546655 | 285 | 6.665885 | -31.15244 | 24.44398 | 2.106700 | 1655 |
| 3 | 3 | 1870 | 0.8302278 | 317 | 6.161488 | -32.18389 | 25.01373 | 2.218094 | 1576 |
| 4 | 4 | 1870 | 0.8090743 | 391 | 5.627544 | -33.33333 | 30.67082 | 2.236691 | 1432 |
| 5 | 5 | 1870 | 0.7883069 | 355 | 6.038408 | -32.19706 | 30.78304 | 2.186470 | 1307 |
| 6 | 6 | 1870 | 0.7670935 | 402 | 5.478912 | -31.03119 | 27.09527 | 2.269651 | 1160 |
| 7 | 7 | 1870 | 0.7442009 | 419 | 5.728436 | -33.33333 | 23.33326 | 2.301892 | 932 |
| 8 | 8 | 1870 | 0.7171681 | 474 | 5.417732 | -33.33333 | 31.12225 | 2.263297 | 764 |
| 9 | 9 | 1869 | 0.6824630 | 517 | 5.387544 | -30.98357 | 34.12023 | 2.282740 | 545 |
| 10 | 10 | 1869 | 0.6083854 | 625 | 4.250520 | -33.33333 | 36.73610 | 2.382740 | 250 |

The highest maxRet for each model is 36.736. For the Glm model the highest value is in the top decile (d=1). In the XGB model the highest maxRet is in the 2nd decile. Considering these 2 models incorporate grades A and B loans this is to be expected that higher grade loans are more likely to be paid off so our return should be higher. When we look at the maxRet for the RF model we see the maximum value is 36.736 but this value resides in the lowest decile for the model. This model is excluding grade A and B loans which tells us that lower grade loans can yield high returns as well despite having a higher number of defaults. This however is due to the high interest rate of riskier, lower grade loans.

The avgActRet for the GLM and XGB models have ranges from just above 7 to about 3.3. The maximum avgActRet for the RF model is 6.86 with the lowest value being 4.25. Additionally, the number of defaults (numDefaults) is lower in the top half of deciles in the RF model for lower grade loans when comparing the number of defaults in the top deciles with our GLM and XGB models. This is what we wanted to see which is that riskier loans would have a higher number of defaults compared to higher grade loans while still showing a positive avgActRet. The count we have in each decile for the RF model is less than half of the count for the GLM and XGB models. This tells us that half of the loans come from grades A and B which is what we expected. These observations help us to see how much of an impact A and B loans have on the data set.

From the development of a data model using the lower grades it has proven beneficial in determining loans for investment. Before, we removed Grade A and B loans and we saw that higher grade loans yielded more actual returns, however this misrepresented the potential returns for lower grade loans in our results since higher grades loans have a lower interest rate. By focusing on lower grade loans we are able to see that they can yield the same maxRet as the higher grade loans because lower grade loans have a higher interest rate and many of the loans do not default. This shows that we can receive a higher avgActRet with lower grade loans than we initially perceived when comparing all the loan grades. Therefore, there are some effective investment opportunities with lower grade loans that we can see after focusing on lower grade loans which can potentially be increased if we are able to improve our model. Comparing the performance of our current models, it seems that investing in both high and lower graded loans seem to yield a better actual return rate compared to just investing in lower graded loans, but at the cost of a higher charged off rate.
Another approach we test is modelling our data on only grade C and D loans to see if that reduces the default rate and improves our average actual return rate.

***RF (lower graded loans C and D) on test data:***

| | tile | count | avgSc | numDefaults | defPerc | avgActRet | minRet | maxRet | avgTer |
|---|---|---|---|---|---|---|---|---|---|
| | <int> | <int> | <dbl> | <int> | <dbl> | <dbl> | <dbl> | <dbl> | <dbl> |
| 1 | 1 | 1683 | 0.8951063 | 196 | 0.1164587 | 7.179538 | -31.09203 | 23.87927 | 2.093197 |
| 2 | 2 | 1683 | 0.8594209 | 260 | 0.1544860 | 6.471347 | -32.18389 | 24.59900 | 2.129467 |
| 3 | 3 | 1683 | 0.8376714 | 305 | 0.1812240 | 6.200737 | -31.11136 | 25.89642 | 2.191546 |
| 4 | 4 | 1683 | 0.8177013 | 315 | 0.1871658 | 5.900818 | -31.07799 | 27.67063 | 2.233693 |
| 5 | 5 | 1683 | 0.7988084 | 315 | 0.1871658 | 5.969491 | -32.22487 | 26.12436 | 2.201358 |
| 6 | 6 | 1683 | 0.7791377 | 350 | 0.2079620 | 5.461804 | -33.33333 | 27.09527 | 2.275493 |
| 7 | 7 | 1683 | 0.7579740 | 377 | 0.2240048 | 5.545724 | -33.33333 | 24.76416 | 2.234860 |
| 8 | 8 | 1682 | 0.7340654 | 369 | 0.2193817 | 5.783678 | -32.23344 | 30.67082 | 2.229740 |
| 9 | 9 | 1682 | 0.7030694 | 468 | 0.2782402 | 4.392028 | -33.33333 | 28.35695 | 2.322731 |
| 10 | 10 | 1682 | 0.6394896 | 465 | 0.2764566 | 4.674449 | -33.33333 | 27.94457 | 2.369691 |

The results show that focusing on grade C and D loans compared to grade C, D, E and F loans seems to not only reduce the default rate, but also increase the average actual return rate as well, which is ideal.

**Can this provide a useful approach for investment? Compare performance with that in Question 3.**

***Method 1:***

A tibble: 20 x 15

| | tile2 | count | avgPredRet | numDefaults | defPerc | avgActRet | minRet | maxRet | avgTer |
|---|---|---|---|---|---|---|---|---|---|
| | <int> | <int> | <dbl> | <int> | <dbl> | <dbl> | <dbl> | <dbl> | <dbl> |
| 1 | 1 | 203 | 7.698554 | 16 | 0.07881773 | 8.212396 | -28.90200 | 26.12436 | 1.866944 |
| 2 | 2 | 203 | 7.571213 | 12 | 0.05911330 | 8.027839 | -23.59789 | 19.39360 | 2.045134 |
| 3 | 3 | 203 | 7.482567 | 8 | 0.03940887 | 8.340481 | -23.04522 | 16.62608 | 2.047022 |
| 4 | 4 | 203 | 7.432906 | 18 | 0.08866995 | 7.753645 | -26.64333 | 25.01373 | 2.009616 |
| 5 | 5 | 203 | 7.607242 | 20 | 0.09852217 | 7.626745 | -33.33333 | 23.09597 | 2.028437 |
| 6 | 6 | 203 | 7.510078 | 18 | 0.08866995 | 7.963621 | -27.57013 | 16.80634 | 2.116109 |
| 7 | 7 | 203 | 7.486166 | 34 | 0.16748768 | 6.773886 | -25.52043 | 19.87501 | 1.964435 |
| 8 | 8 | 203 | 7.474895 | 22 | 0.10837438 | 8.020086 | -23.37621 | 16.95126 | 2.056524 |
| 9 | 9 | 203 | 7.518001 | 32 | 0.15763547 | 6.857447 | -32.19706 | 22.98571 | 2.227167 |
| 10 | 10 | 203 | 7.550907 | 30 | 0.14778325 | 7.550060 | -30.99250 | 25.89642 | 2.039625 |

1-10 of 20 rows | 1-10 of 15 columns Previous 1 2 Next

***Method 2:***

A tibble: 20 x 15

| | tile2 | count | avgPredRet | numDefaults | defPerc | avgActRet | minRet | maxRet | avgTer |
|---|---|---|---|---|---|---|---|---|---|
| | <int> | <int> | <dbl> | <int> | <dbl> | <dbl> | <dbl> | <dbl> | <dbl> |
| 1 | 1 | 203 | 10.499455 | 33 | 0.16256158 | 9.376429 | -28.41507 | 31.12225 | 2.005884 |
| 2 | 2 | 203 | 8.845148 | 30 | 0.14778325 | 7.414482 | -26.40100 | 24.23726 | 2.185931 |
| 3 | 3 | 203 | 8.222139 | 25 | 0.12315271 | 7.567252 | -28.57415 | 19.53378 | 2.077472 |
| 4 | 4 | 203 | 7.969854 | 23 | 0.11330049 | 8.148814 | -32.19706 | 23.10076 | 2.084169 |
| 5 | 5 | 203 | 7.802109 | 25 | 0.12315271 | 7.799954 | -31.99226 | 25.01373 | 1.972355 |
| 6 | 6 | 203 | 7.581124 | 17 | 0.08374384 | 8.282987 | -28.90200 | 19.37669 | 1.953527 |
| 7 | 7 | 203 | 7.469896 | 27 | 0.13300493 | 6.527495 | -30.94880 | 16.81932 | 2.195247 |
| 8 | 8 | 203 | 7.312661 | 23 | 0.11330049 | 8.103115 | -22.19197 | 20.66970 | 2.015277 |
| 9 | 9 | 203 | 7.267184 | 20 | 0.09852217 | 7.723334 | -28.55691 | 19.36493 | 2.027372 |
| 10 | 10 | 203 | 7.246698 | 25 | 0.12315271 | 7.657183 | -26.69889 | 20.51371 | 2.061868 |

1-10 of 20 rows | 1-10 of 15 columns Previous 1 2 Next

***RF (lower graded loans C and D) on test data:***

| | tile | count | avgSc | numDefaults | defPerc | avgActRet | minRet | maxRet | avgTer |
|---|---|---|---|---|---|---|---|---|---|
| | <int> | <int> | <dbl> | <int> | <dbl> | <dbl> | <dbl> | <dbl> | <dbl> |
| 1 | 1 | 1683 | 0.8951063 | 196 | 0.1164587 | 7.179538 | -31.09203 | 23.87927 | 2.093197 |
| 2 | 2 | 1683 | 0.8594209 | 260 | 0.1544860 | 6.471347 | -32.18389 | 24.59900 | 2.129467 |
| 3 | 3 | 1683 | 0.8376714 | 305 | 0.1812240 | 6.200737 | -31.11136 | 25.89642 | 2.191546 |
| 4 | 4 | 1683 | 0.8177013 | 315 | 0.1871658 | 5.900818 | -31.07799 | 27.67063 | 2.233693 |
| 5 | 5 | 1683 | 0.7988084 | 315 | 0.1871658 | 5.969491 | -32.22487 | 26.12436 | 2.201358 |
| 6 | 6 | 1683 | 0.7791377 | 350 | 0.2079620 | 5.461804 | -33.33333 | 27.09527 | 2.275493 |
| 7 | 7 | 1683 | 0.7579740 | 377 | 0.2240048 | 5.545724 | -33.33333 | 24.76416 | 2.234860 |
| 8 | 8 | 1682 | 0.7340654 | 369 | 0.2193817 | 5.783678 | -32.23344 | 30.67082 | 2.229740 |
| 9 | 9 | 1682 | 0.7030694 | 468 | 0.2782402 | 4.392028 | -33.33333 | 28.35695 | 2.322731 |
| 10 | 10 | 1682 | 0.6394896 | 465 | 0.2764566 | 4.674449 | -33.33333 | 27.94457 | 2.369691 |

Comparing our Random Forest Model using C and D grade loans, we can see that overall, our Method 1 model does much better than the RF models we generated on the lower graded loans. Method 1 also does much better compared to our single models, the xGB and GLM models, providing a much lower default rate and higher average actual return rate.

Using Method 2 provides the highest average actual return rate compared to our RF model and Method 2 model in the 1st decile, but that comes at the cost of a much higher default rate of 16.26%. So considering the existing models and methods we have, if the investor is looking to invest in loans based on a higher average actual return and is willing to take on the increased chance of the loans defaulting, it would probably be a wiser approach to invest in loans in the top decile of Method 2, which generally consist mostly of loans in grade C and D. With Method 2 we also see a high maxRet of 31.12 in the top decile while in the RF model we see higher maxRet values in the bottom deciles with values of 30.67, 28.35 and 27.94 in the last 3 deciles respectively. As loans become riskier, or lower in grade, the interest rate increases resulting in a higher maxRet.

Additionally, the default percentage (defPerc) shows higher percentages when moving down the deciles in our RF model than in Method's 1 or 2. Our RF model has default percentages ranging from 11% to 27%. In comparison the default percentages for our Method 1 and 2 models only go to about 16% at the maximum. Method 1 hits this maximum in decile 7 with one of the lowest avgActRet. This makes sense because Method 1 resulted with the top deciles having higher grade loans but lower interest rates meaning the loans were less risky. Method 2 results show the riskier loans. These loans have higher returns which is why we see a higher avgPredRet and avgActRet.

In conclusion, when comparing results from the combined models 1 and 2, the single models and the lower grade RF model, we see there is a significant investment opportunity when using Method 1, which consists of B, C, and D grade loans. Due to the high interest rates of lower loans, they can result in high returns even exceeding the returns from high grade loans. So while higher grade loans are safer and more likely to be Fully Paid, the return on investment will be minimal. Conversely, for lower grade loans, the risks are greater, but due to increased risk these loans have higher interest rates. From our results we observed that lower grade loans do have a higher chance of defaulting but there are still positive returns resulting from those lower grade loans. In particular, it seems that using method 1 and investing in the B, C and D grade loans in the 1st decile provides the best chance of receiving a higher return. We think spreading the investment out between a variety of "middle" grade loans from in method 1 is a safer and more effective approach compared to solely investing in higher graded A and B loans, or the lower graded C, D, E and F loans.