IDS 572 - Assignment 2

Pritha Ghosh, Anoop Gopalam, Tejaswi Cherukuri

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.

A: GLM Methods can be categorized into the following two types -

1. **Ridge Regression :** The alpha value is equal to zero. Penalty function is the sum of squares of the coefficients. It puts pressure on the coefficients to get low values.
2. **Lasso Regression:** The alpha value is equal to one. The penalty function is the sum of the absolute values of the coefficients. There is pressure to set the coefficients to zero if the variables are highly correlated, thereby affecting variable selection.

Parameter Values:

- **x:** Data matrix of the predictor variables
- **y:** The response variable
- **family:** The response type. In our models, we have put family = Binomial since we are predicting loan status, which is a binary response variable.
- **alpha:** Tuning parameter. For Ridge Regression, alpha =0. For Lasso Regression, alpha=1.
- **lambda:** Shrinkage penalty. The best lambda value is one that minimizes the cross-validation prediction error rate. It can be determined by the function cv.glmnet().

Using lambda.1se, only the following variables have coefficients which are not equal to zero.

row <chr>	column <chr>	value <dbl>
(Intercept)	1	3.029555e+00
int_rate	1	-2.060641e-02
grade	1	-4.539577e-02
sub_grade	1	-5.835295e-02
home_ownership	1	-1.283689e-02
dti	1	-7.310334e-03
acc_open_past_24mths	1	-2.279049e-02
mort_acc	1	6.183931e-04
num_rev_tl_bal_gt_0	1	-4.549401e-03
tot_hi_cred_lim	1	4.678089e-07

The coefficients of all the other variables have been set to zero by the Lasso Regression algorithm, thereby reducing the model complexity. However, although the model is less complex, there may be a trade-off on accuracy when we use lambda.1se instead of lambda.min.

We look at the below parameters while determining which model gives the best performance:

- Accuracy
- Sensitivity
- ROC curve
- AUC value

We first try experimenting with GLM models on our current imbalanced dataset. We then move on to experiment with weights as well as oversampling the dataset. We use a 70:30 split between the training and test sets.

The class that we are trying to predict is : Charged Off

We have experimented with the below mentioned parameters:

1. **Alpha**: 1 for Lasso, 0 fo Ridge
2. **Lambda:** lambda.min,lambda.1se
3. **Threshold:** 0.5
4. **Sampling:** We worked with the imbalanced dataset, then used weights as well as oversampling.

Observations:

Model	Sampling	Alpha	Threshold	Lambda	Train Output			Test Output		
					Accuracy	Sensitivity	AUC	Accuracy	Sensitivity	AUC
Lasso	No	1	0.5	lambda.min=0.0001131346	0.854	0.012	0.505	0.853	0.011	0.504
Lasso	No	1	0.5	lambda.1se=0.006782241	0.085	0.010	0.502	0.853	0.004	0.644
Lasso	Weights	1	0.5	lambda.min=0.0005594495	0.630	0.660	0.642	0.612	0.695	0.647
Lasso	Oversampling	1	0.5	lambda.1se=0.00226277	0.641	0.653	0.640	0.612	0.689	0.644
Ridge	No	0	0.5	lambda.min= 0.007618654	0.854	0.011	0.504	0.853	0.010	0.504
Ridge	Yes	0	0.5	lambda.1se=0.1385799	0.630	0.660	0.642	0.609	0.690	0.643

We can observe by the above results that - a model based on an unbalanced dataset has high accuracy, but low sensitivity.

If we apply weights or perform oversampling on the train data, even though the accuracy decreases on both the train as well as the test sets, the sensitivity increases and the AUC value also increases.

We would prefer a model with higher sensitivity over higher accuracy because we want our model to predict the Charged Off loans correctly.

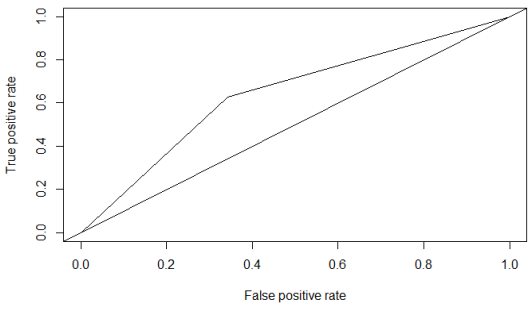
If we had to choose just one model out of the six models that we have designed, we would pick the Lasso Model with weights.

Lasso	Weights	1	0.5	lambda.min=0.0005594495	0.630	0.660	0.642	0.612	0.695	0.647

The Confusion Matrix for the above model on the Test set is below:

```
> confusionMatrix(as.factor(GLM1st_predclass), y1st)
Confusion Matrix and Statistics

          Reference
Prediction     0     1
         0  2479  8339
         1  1086 12402

               Accuracy : 0.6122
                 95% CI : (0.6061, 0.6184)
    No Information Rate : 0.8533
    P-Value [Acc > NIR] : 1

                  Kappa : 0.1592

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

            Sensitivity : 0.6954
            Specificity : 0.5979
         Pos Pred Value : 0.2292
         Neg Pred Value : 0.9195
             Prevalence : 0.1467
         Detection Rate : 0.1020
   Detection Prevalence : 0.4451
      Balanced Accuracy : 0.6467

       'Positive' Class : 0
```

ROC Curve:



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

A: In our linear (glm) models, we have used Binomial Regression.

Glm models are fit using the glm() function, which can be expressed as following -

glm(formula,family=family type(link=link function),data)

In the models we have built, we use family = binomial, therefore our link function = logit.

The logit link function converts the covariate values to a scale of probability between 0 and 1.

logit(p) = ln(p/[1-p]) = β0 + β1x

Loss function L = − y log p − (1−y) log (1−p) , where p = probability of 0 (Charged off) or 1 (fully paid)

The loss function is the negative of maximum likelihood function.

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

A:

Model	Train Output			Test Output		
	Accuracy	Sensitivity	AUC	Accuracy	Sensitivity	AUC
Lasso Best Model	0.630	0.660	0.642	0.612	0.695	0.647
RF Model	0.87	0.46	0.68	0.8543	0.4625	0.6652
XGBoost Model	0.854	0.71	0.68	0.856	0.702	0.68

The table above summarizes the results of the best Lasso Model that we have built now and compares it with the best random forest and xgBoost model that we built as part of Assignment 1.

The XGBoost model has the best performance. It has an accuracy of over 85% on both the train as well as the test sets, as well as the best sensitivity among all the models (above 70%). The AUC value is also ~0.68, which is considered good.

(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?

A: The below chart summarizes the variables that are found to be important by the best models from the different methods that we have experimented with so fat, namely - GLM, XGBoost, Random Forest and Decision Trees.

GLM Models

row <chr>	column <chr>	value <dbl>
(Intercept)	1	1.882261e+00
int_rate	1	-7.256564e-02
grade	1	-5.524253e-02
sub_grade	1	-2.578222e-02
emp_length	1	-3.621213e-03
home_ownership	1	-3.100669e-02
verification_status	1	-4.415386e-02
dti	1	-1.064473e-02
acc_open_past_24mths	1	-3.309843e-02
mort_acc	1	1.193035e-02

XGBoost

	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

Random Forest

```
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
```

Decision Tree

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

The top three most common variables across the models are : int_rate, dti, grade and sub_grade.

The variables which are unique to each individual model is as below -

A	B
GLM Models	**XGBoost**
home_ownership	installment
verification_status	
mort_acc	
Random Forest	**Decision Tree**
mo_sin_old_il_acc	total_bc_limit

The difference in the importance of variables is a driving factor as to why each of the models perform uniquely.

(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 ?

A: We have experimented with two splits -

1. 70:30 between the train and test sets
2. 50:50 between the train and test sets

The 70:30 split tends to give better accuracy than the 50:50 split.

On changing the seed value, and splitting the data again into test and train sets, we have observed that the 70:30 split gives similar results for any seed value, whereas the 50:50 split gives a much larger difference.

As a rule of thumb, it is recommended to have more data in the train set than the test set, and our experiments with the split corroborates this claim.

In building our models above, we have experimented by building models on the unbalanced dataset, putting weights on the training data as well as over-sampling the training data to balance the “Fully Paid” and “Charged Off” classes. We have observed that when we balance the training data across the classes, the models do a much better job at predicting the default loans, which is what our aim is.

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.

A: The difference between the Total Payment and the Funded Payment gives us the Annual Return on the loan.

Since the loan term is 3 years, we need to multiply the total returns by 12/36 to get the return for a year.

Annual Return = (Funded amount – total payment)/funded amount *12/36 * 100

The above formula is the unadjusted annual return as there are loans present in the dataset, which were closed before the end of the 3 year term.

Therefore, to calculate the adjusted Actual Annual Return, we need to apply the following formula:

Actual Annual Return = ((Total Payment – Funded Amount)/Funded Amount)/Actual Term,

where the Actual Term is the difference between the last payment date and the loan issue date.

Every time a borrower makes a payment, Lending Club takes a 1% service fee. Therefore, for every $100 payment, the service fee is $1. This is included in our calculations for annual returns, where we deal with Total Payment.

GLM Models to Predict Actual Returns

We have experimented with the lambda and alpha values.

For Lasso Regression, alpha=1 and for Ridge Regression, alpha=0.

For both Lasso and Ridge Regression, we have considered lambda.min and lambda.1se.

Model	lambda	alpha	RMSE on Train Set	RMSE on Test Set
GLM Ridge	lamda.min	0	8.342538	8.357387
GLM Ridge	lamda.1se	0	8.40277	8.410172
GLM Lasso	lambda.min	1	8.342457	8.35649
GLM Lasso	lambda.1se	1	8.391193	8.402655

As we can see, the difference in the Root mean squared error on the train set and test sets are quite similar for all the models.

However, if we had to choose just one model, we would choose the 3rd Model, which uses Lasso Regression as it does not consider the unimportant variables in building the model.

RF Models

We have experimented with parameters like num.trees and mtry.

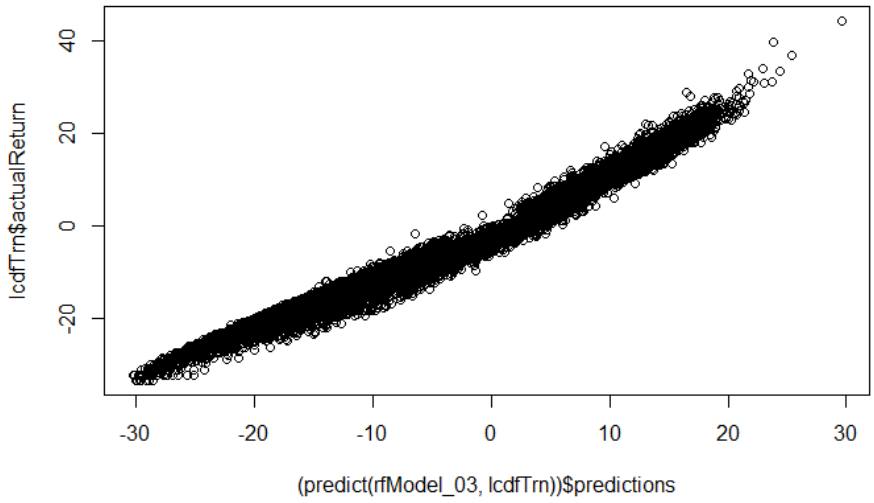
By default, mtry is the square of the number of variables in the model. Our model has about 50 variables, so by default the mtry value would be around 7.

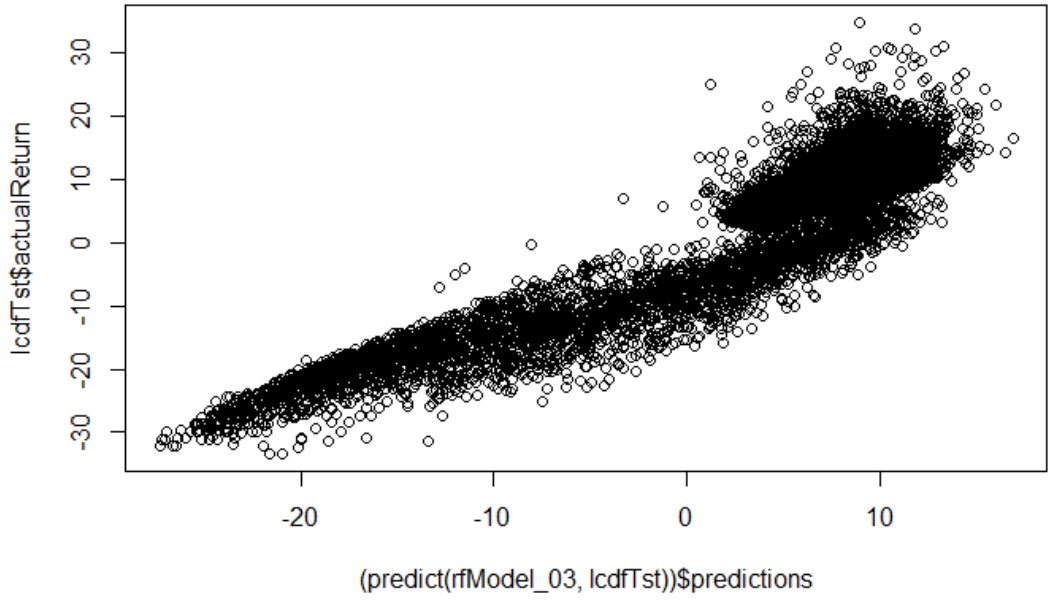
Model	num.trees	mtry	RMSE on Train Set	RMSE on Test Set
Random Forest	100	Default	1.865612	4.203787
Random Forest	200	7	1.848982	4.189607
Random Forest	200	10	1.4865	3.458336
Random Forest	500	Default	1.805467	4.130374

The table above summarizes the results of the three random forest models that we have built. The model performs very well on the train set, and the performance is average on the test set.

Out of the three models, the one with the best performance would be Model 3, with num.trees=200 and mtry=10 as it gives the least difference between the root mean squared errors of the train and test sets.

The plot for Random Forest Model 3 for Actual vs Predicted Returns on both the train and test sets -



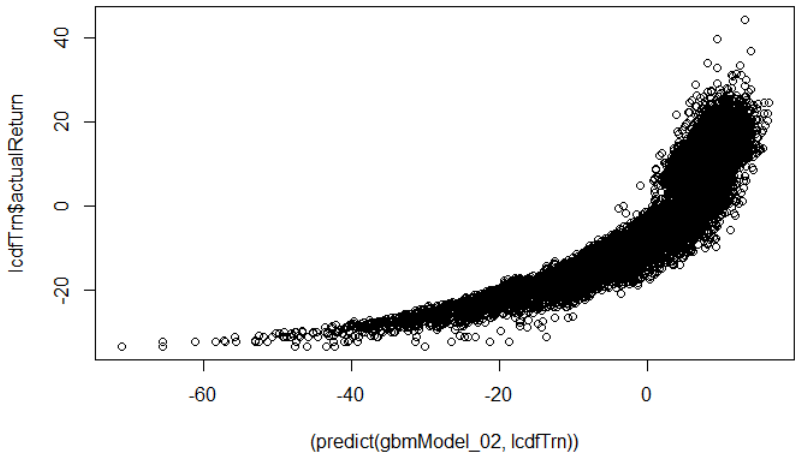


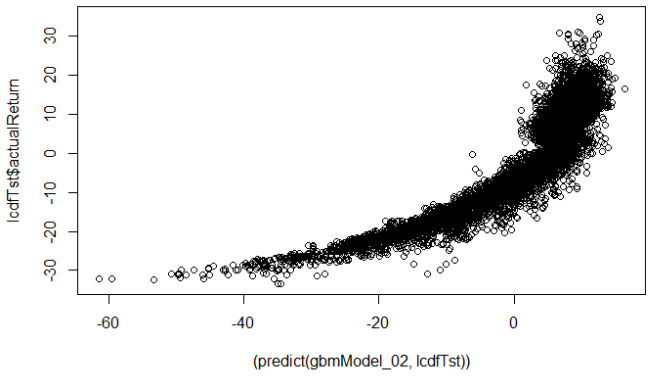
GBM Model

The parameters that we have experimented with are number of trees, interaction depth, shrinkage, bag function, cross validation folds and number of cores.

Model	n.trees	interaction.depth	shrinkage	bag.fraction	cv.folds	n.cores	RMSE on Train Set	RMSE on Test Set
GBM	100	2	0.1	0.5	5	NULL	3.942317	4.039113
GBM	200	2	0.1	0.5	5	NULL	3.67963	3.775727

Train Set and Test Set Plots for Actual vs Predicted Actual Annual Returns





Among the two GLM models, the one with number of trees 200 is a good model , with a low RMSE on both the train and test datasets.

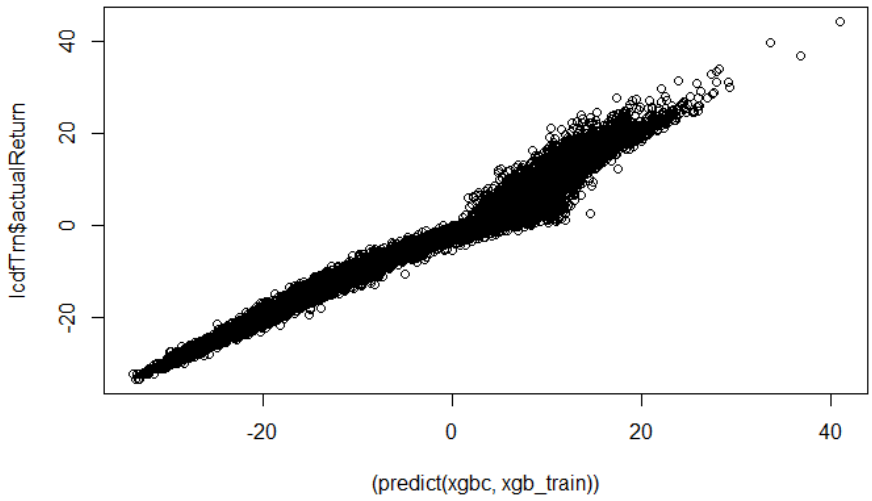
XG Boost

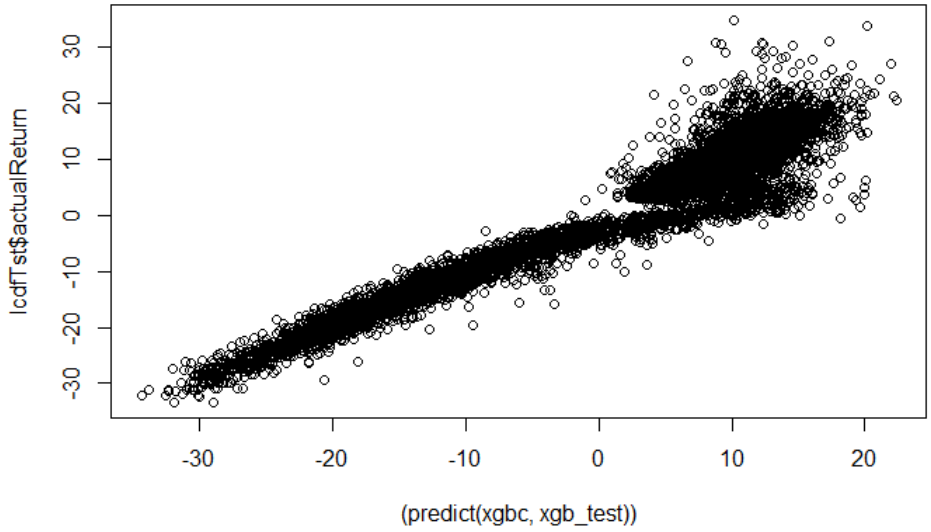
For XGBoost, we have experimented with parameters like max.depth and nrounds and the results are summarized in the table below -

Model	max.depth	nrounds	RMSE on Train Set	RMSE on Test Set
XGBoost	5	500	1.185852	2.063197
XGBoost	10	500	0.0303983	2.118649

We would choose the first model as the best model as it has a similar RMSE on both the train and test data sets.

Train and Test Set Plots for Actual vs Predicted Actual Returns





Model Performance Comparison

We compare the best models of each type - GLM, Random Forest and GBM.

We have plotted the Root Mean Square Errors on both the training as well as the test data set.

Although the RF model has the overall lowest RMSE on both the train and test sets, it cannot be considered as the best model as it overfits on the train data.

The XGBoost Model is our best performing model, with low RMSE on both the train and test sets, plus there is not much variation.

We have also tried experimenting with 50:50 split in our training and test datasets on the best XGBoost model to see how the performance varies in comparison to our standard 70:30 split.

Split	RMSE-Train	RMSE-Test	Difference
70:30	1.185852	2.063197	0.739843589
50:50	1.056983	2.101097	0.987824781

On using a 50:50 split, the difference in RMSE increases, therefore we would prefer to use a 70:30 split.

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?

A: We have concluded that the XGBoost model with the below parameters is our best model :

Model	max.depth	nrounds	RMSE on Train Set	RMSE on Test Set
XGBoost	5	500	1.185852	2.063197

We will now proceed on making the decile chart for this model. The objective of this is to see that how the predicted values of the Actual Returns changes with the decile for train and test sets.

We will consider the top deciles from the Actual Returns predictive model, ranked by the Loan Status model scores.

Train Data Decile Chart:

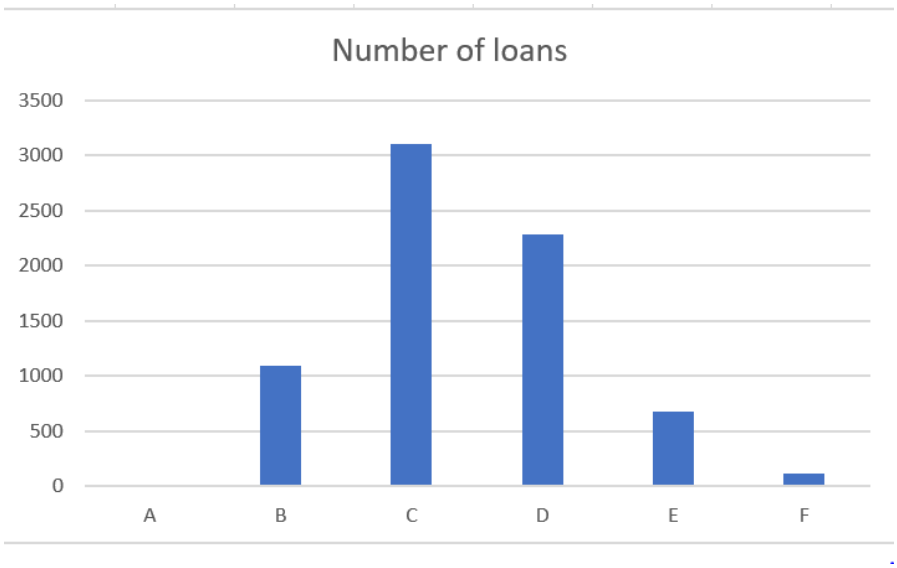
tile	count	avgpredRet	numDefaults	avgActRet	minRet	maxRet	avgTer	totA	totB	totC	totD	totE	totF
1	5672	14.001899	29	14.734215	1.1725556	44.359486	1.081146	2	84	1842	2369	1080	257
2	5672	10.799065	104	10.996680	0.4663656	21.024458	1.580216	4	982	2762	1493	424	5
3	5672	9.349333	157	9.301770	0.5439444	16.603558	2.008044	12	1788	2257	1587	25	3
4	5672	8.229324	180	8.088566	0.0000000	16.354429	2.295391	104	1931	3229	391	14	2
5	5672	7.326431	141	7.206489	-1.0254825	13.268773	2.378235	531	2615	2475	38	11	2
6	5672	6.469747	125	6.392843	-0.1970278	12.443671	2.288454	1369	3662	600	28	11	2
7	5671	5.500377	133	5.406516	-1.4951435	12.212310	2.270411	2580	2996	61	22	11	1
8	5671	4.561639	92	4.463017	-2.1543182	11.937784	2.542766	4917	693	32	16	9	2
9	5671	1.979623	1630	1.562948	-10.4626667	7.946291	2.835304	4188	429	548	369	113	20
10	5671	-16.652900	5671	-16.589316	-33.3333333	-4.846358	3.000000	539	1273	1940	1275	528	95

Test Data Decile Chart

tile	count	avgpredRet	numDefaults	avgActRet	minRet	maxRet	avgTer	totA	totB	totC	totD	totE	totF
1	2431	13.374530	161	13.371335	-1.5474556	33.790354	1.364166	0	27	771	1001	512	105
2	2431	10.733354	110	10.889003	-0.8586809	34.952443	1.622012	0	357	1217	713	134	9
3	2431	9.423194	69	9.571933	-1.2529620	30.670824	1.948633	5	699	1134	556	34	3
4	2431	8.325029	45	8.365220	-4.5208333	30.854770	2.193966	43	924	1260	196	8	0
5	2431	7.386730	36	7.407013	-3.2066667	20.569043	2.328893	204	1180	1006	36	5	0
6	2431	6.483638	44	6.500796	-2.2037131	27.471984	2.252434	587	1523	300	15	5	1
7	2430	5.503341	42	5.494495	-2.6204545	17.251898	2.230339	1165	1194	60	8	3	0
8	2430	4.580575	30	4.557476	-4.0920537	21.673584	2.515059	2073	327	20	7	3	0
9	2430	2.361957	598	1.850118	-15.8023333	13.954612	2.802682	1860	191	201	128	40	8
10	2430	-16.676041	2430	-16.628303	-33.3333333	-2.862347	3.000000	219	524	862	554	221	45

After analyzing the test data decile chart, we would recommend investing in the loans in the top 3 deciles since it has a good average rate of return.

There are total 7293 loans in the top 3 deciles, out of which 339 are charged off and 6954 are fully paid.



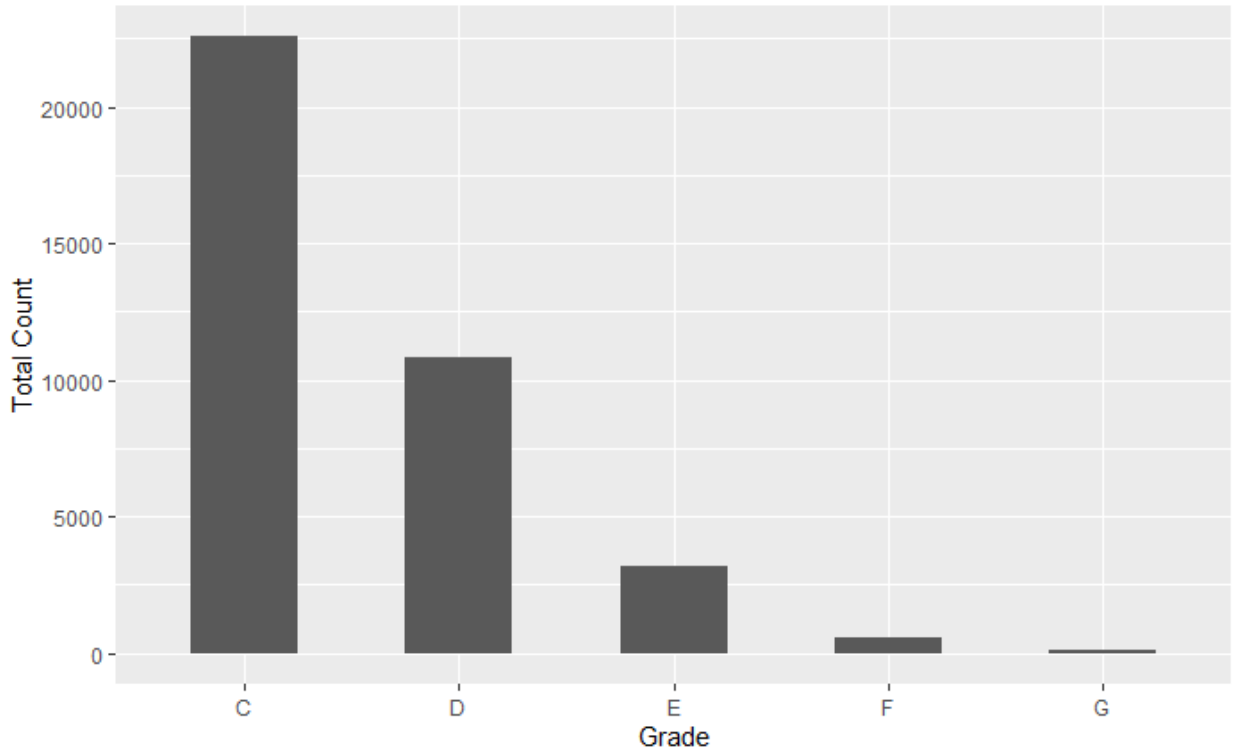
Most of the loans in the top 3 deciles are distributed across Grade C and Grade D. Therefore, we would choose to invest in the loans of Grade C or Grade D.

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

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

A: Considering only the lower grade loans, that is C and below -

	Charged Off	Fully Paid
C	4116	18461
D	2647	8155
E	1045	2146
F	191	369
G	38	53



We will run our models on these loans, taking a split of 70:30 between the train and test data sets.

In order to see the best investment approach, we will develop RF, GLM and XGBoost models. The best models of each case are highlighted in green below.

GLM Models

Model	lambda	alpha	RMSE on Train Set	RMSE on Test Set
GLM Ridge	lamda.min	0	10.61133	10.59937
GLM Ridge	lamda.1se	0	10.69577	10.67488
GLM Lasso	lambda.min	1	10.6117	10.59913
GLM Lasso	lambda.1se	1	10.69577	10.67488

RF Models

Model	num.trees	mtry	RMSE on Train Set	RMSE on Test Set
Random Forest	100	Default	2.333121	5.410748
Random Forest	200	7	2.320915	5.397346
Random Forest	200	10	1.888512	4.487629
Random Forest	500	Default	2.295375	5.365719

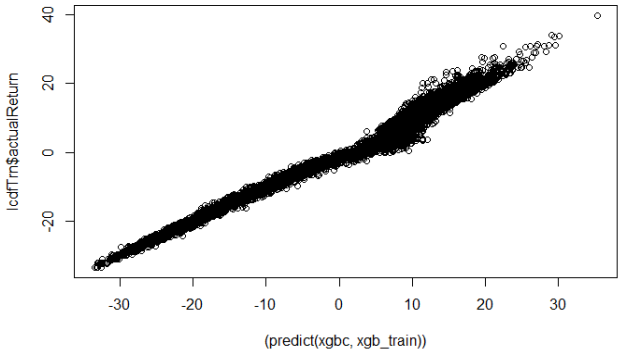
GBM Models

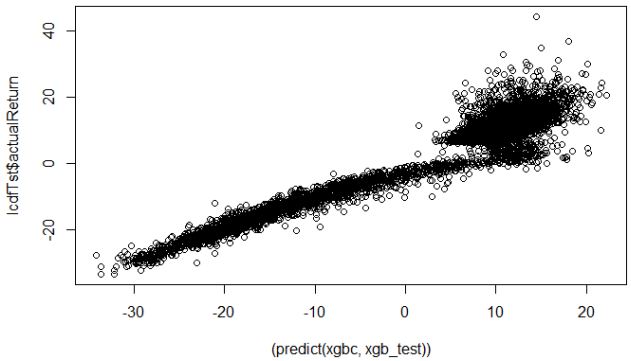
Model	n.trees	interaction.depth	shrinkage	bag.fraction	cv.folds	n.cores	RMSE on Train Set	RMSE on Test Set
GBM	100	2	0.1	0.5	5	NULL	4.98633	5.251917
GBM	200	2	0.1	0.5	5	NULL	4.788095	5.10318

XGBoost Models

Model	max.depth	nrounds	RMSE on Train Set	RMSE on Test Set
XGBoost	5	500	1.162666	2.815258
XGBoost	10	500	15.10282	14.80036

Train and Test Plots for the best model - XGBoost Model 1





We now see how the actual vs predicted results differ with deciles :

Decile Chart on Train Data:

	count	avgpredRet	numDefaults	avgActRet	minRet	maxRet	avgTer	totA	totB	totC	totD	totE	totF
1	2606	15.831122	2	16.660620	11.0856623	39.725847	0.8925788	0	0	421	1279	660	209
2	2606	12.704276	7	13.169852	3.6329340	21.062750	1.1802472	0	0	1373	894	300	39
3	2606	11.372543	21	11.609896	3.2615833	20.041621	1.5569281	0	0	1577	683	341	4
4	2606	10.377682	47	10.358033	3.3010658	15.988758	2.0267074	0	0	1402	1033	169	1
5	2606	9.517777	58	9.363809	0.5456592	14.276789	2.4001165	0	0	1473	1103	29	1
6	2605	8.706124	81	8.415936	1.1001445	13.606648	2.6770198	0	0	1935	654	14	2
7	2605	7.952286	95	7.671350	0.6418163	11.422179	2.8277261	0	0	2448	143	13	1
8	2605	7.062424	175	6.925359	0.0000000	9.683183	2.9234511	0	0	2529	56	17	2
9	2605	-5.698728	2541	-6.406562	-16.3183838	7.548403	2.9514144	0	0	1395	821	318	57
10	2605	-20.294008	2605	-20.238847	-33.3333333	-12.288140	3.0000000	0	0	1279	860	390	63

Decile Chart on Test Data:

	count	avgpredRet	numDefaults	avgActRet	minRet	maxRet	avgTer	totA	totB	totC	totD	totE	totF
1	1117	14.723135	77	14.739946	-1.69550183	44.35949	1.323560	0	0	136	488	374	102
2	1117	12.348139	60	12.488460	-0.14586667	27.94457	1.457608	0	0	476	470	159	12
3	1117	11.260242	50	11.554282	-0.92000000	32.90425	1.639891	0	0	612	422	76	7
4	1117	10.410370	41	10.613015	-0.06388889	27.92745	1.907519	0	0	654	416	42	5
5	1117	9.677512	36	9.870174	-1.90257143	27.09527	2.178474	0	0	716	372	28	1
6	1117	8.944113	14	9.067499	-0.86127619	28.09425	2.414803	0	0	883	223	10	1
7	1116	8.186161	15	8.245861	-1.06785714	21.77335	2.661734	0	0	1012	99	4	1
8	1116	7.188682	26	7.661006	-4.63444444	24.41964	2.735531	0	0	1054	54	8	0
9	1116	-4.829001	970	-5.908090	-20.32327485	21.06275	2.944074	0	0	641	345	103	24
10	1116	-20.371475	1116	-20.090290	-33.33333333	-10.90256	3.000000	0	0	561	387	136	28

After analyzing the test data decile chart, our conclusion is that -

The investors can invest in the top 3 deciles for maximum returns. The average return is more than 12%, which is very good.

The top 3 deciles contain 3351 loans, out of which 187 are Charged Off and 3164 are Fully Paid.

	Charged Off	Fully Paid
C	1271	5462
D	816	2443
E	312	673
F	55	104
G	14	16

Grade	Charged Off	Fully Paid	Default Rate
C	1271	5462	0.188772
D	816	2443	0.250384
E	312	673	0.316751
F	55	104	0.345912
G	14	16	0.466667

Running the best models of each type (GLM,RF,XGBoost) on the lower grades subset for predicting loan status , we get the following results -

Model	Train Output		Test Output	
	Accuracy	Sensitivity	Accuracy	Sensitivity
Lasso	0.6008	0.5532	0.5939	0.5401
Random Forest	0.85	0.45	0.7791	0.516129
XGBoost	0.8534	0.001	0.856	0.008

We can see from the above table that the performance of the Random Forest model is the best overall.

Looking at the test decile chart on this model, we can see that -

tile	count	avgSc	numDefaults	avgActRet	minRet	maxRet	avgTer	totA	totB	totC	totD	totE	totF
1	1117	0.8923581	149	6.912595	-31.11641	22.34097	2.086617	0	0	1043	73	1	0
2	1117	0.8540719	188	6.336827	-31.11136	24.59900	2.169022	0	0	974	141	2	0
3	1117	0.8294437	182	6.380623	-28.84886	26.22843	2.158124	0	0	930	176	11	0
4	1117	0.8079432	234	5.659956	-31.07799	30.67082	2.249277	0	0	823	284	9	1
5	1117	0.7873349	227	5.963794	-31.01948	23.65642	2.267738	0	0	789	302	22	4
6	1117	0.7672153	248	5.819790	-29.91854	25.01373	2.260171	0	0	685	381	45	5
7	1116	0.7438617	245	5.995411	-30.91513	30.78304	2.259602	0	0	574	462	70	10
8	1116	0.7168699	281	5.515901	-32.21033	36.73610	2.258905	0	0	464	511	122	17
9	1116	0.6810146	321	4.857930	-32.20601	31.09951	2.345790	0	0	325	511	232	41
10	1116	0.6062584	393	4.137249	-33.33333	31.35167	2.394078	0	0	126	418	471	81

The average rate of return on investing in loans of Grade C and D are above 6.5%, which is a good return rate.

We can see from the above table that the default rate of Grade C and D are relatively lower than the other grades.

Conclusion: Therefore, it is safe to invest in loans of Grade C and D, which have a high rate of return and low default rates.