ANISHA VIJAYAN (UIN: 662618335)
PREETHI SRINIVASAN (UIN: 663981973)

ASSIGNMENT 2

Models for investment decisions in Lending Club loans

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.

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?

Ans. We used xgBoost boosted tree model to make predictions about the loan status. We experimented with the below set of parameters to get to the final best model.

max_depth	eta	nrounds	Other parameters used	Error	AUC
5	0.01	500	early_stopping_rounds = 10, eval_metric = "error"	0.139580	0.669933
4	1	500	eval_metric = "auc"	0.140670032	0.671486
4	1	500	eval_metric = "error"	0.139544	0.668735
4	0.5	500	eval_metric = "error"	0.139544	0.676398
4	0.1	500	eval_metric = "error"	0.139580	0.665456
4	0.01	500	eval_metric = "error"	0.139580	0.664336
6	0.1	500	eval_metric = "auc"	0.139544	0.679362
6	0.01	1000	eval_metric = "auc"	0.139617	0.672747
6	0.1	1000	eval_metric = "auc", lambda=0.05	0.139617	0.678080
6	0.01	1000	eval_metric = "auc", lambda=0.05, subsample=0.7, colsample_bytree=0.5	0.139635	0.676123
6	0.01	1000	eval_metric = "auc", lambda=0.05, subsample=0.7, min_child_weight =1, gamma= 1	0.139562	0.682505
6	0.01	1000	eval_metric = "error", lambda=0.05, subsample=0.7, colsample_bytree=0.5, min_child_weight =1, gamma= 1	0.139635	0.671706

6	0.1	500	early_stopping_rounds = 10, eval_metric = "error"	0.139507	0.676537

When we tried to decrease the learning rate (eta) values, their evaluation errors increased and the evaluation auc values also decreased. This is not as per expected. When we tried to use the gamma (minimum loss reduction required to make a further partition) as 1, we got significant improvement in the auc value, but the error was high compared to other times. Therefore, after passing in all the possible values for the parameter max_depth and eta, we got a best model with evaluation error as 0.139507 and auc value as 0.676537 where the max_depth was 6 and eta 0.1.

Using the confusion matrix to evaluate the performance of the model, we get an accuracy of 86.13%.

	Reference	
Prediction	Fully Paid	Charged Off
Fully Paid	47267	7676
Charged Off	1	28

We get the below ROC curve for the xgboost model with an auc value of 0.6765. On the right-hand side is the lift curve for the xgboost model (Fig 1.1).

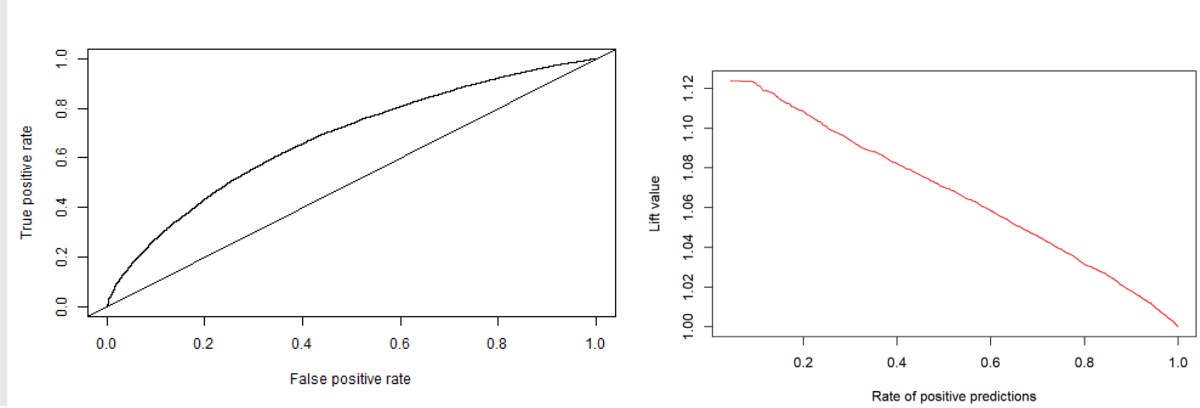

Fig.1.1: (Left) ROC curve for xgboost model (Right) Lift curve for xgboost model

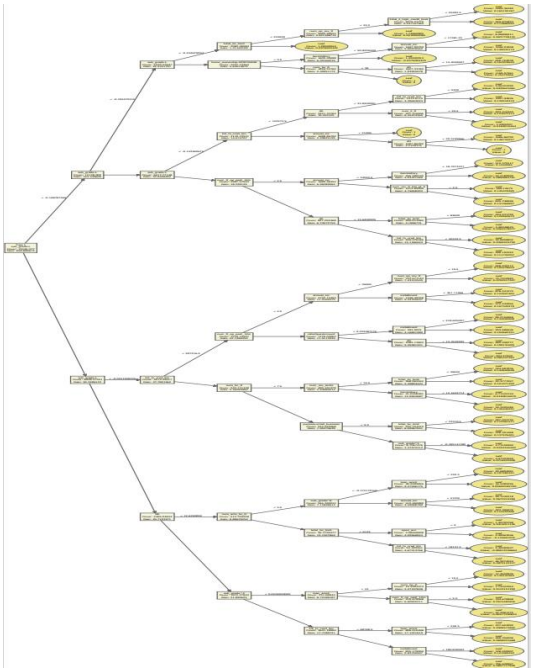

Fig 1.2: XGBoost Model for loan status prediction

With profit value as 35 and cost value as -42, we get the maximum profit for the xgboost model as $1,332,135 which is higher than the maximum profits gained from the decision tree model ($1,332,100) and random forest model ($1,332,345).

Table 1.1: Comparison of different models

Model Name	Cost Based Performance (Max Profit)	Confusion Matrix (Accuracy)	AUC Values
Decision Tree Model	$1,332,100	84.3%	0.619729
Random Forest Model	$1,332,345	86.1%	0.676998
XGBoost Model	$1,332,135	86.13%	0.6765

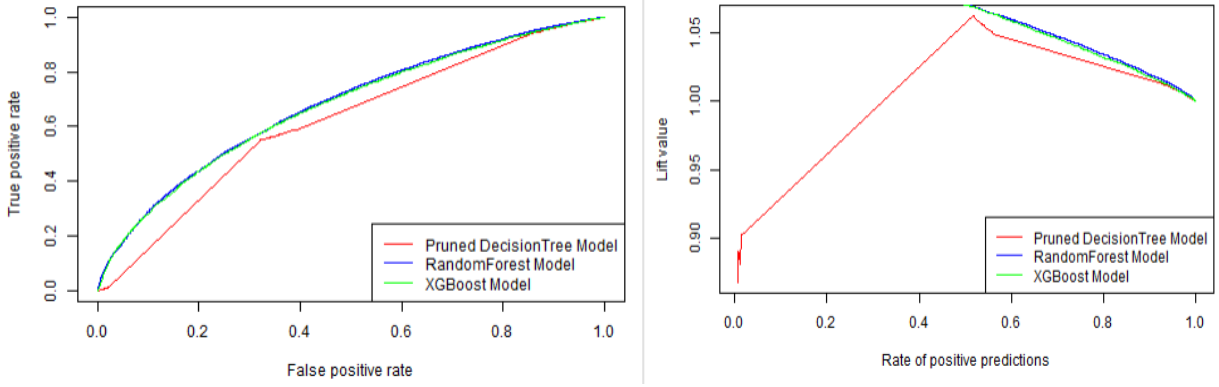

Fig 1.3: (Left)ROC Curves (Right) Lift curve for Decision Tree model, Random Forest Model and XGBoost model

The Fig1.3 shows the ROC Curves and Lift curves for the three different models. The curve for random forest model and the xgboost model is very similar. This is also evident by looking at their AUC (area under the curve) values from the Table1.1. From the Table1.1, we can see that both random forest model and xgboost model have almost the same efficiency. Their profit curves as shown in Fig1.4, also have almost overlapping curves for random forest model and xgboost model. We could take the best profit as 30,000 index value since post this value the profit curve seems to become non-linear and tends to fall flat.

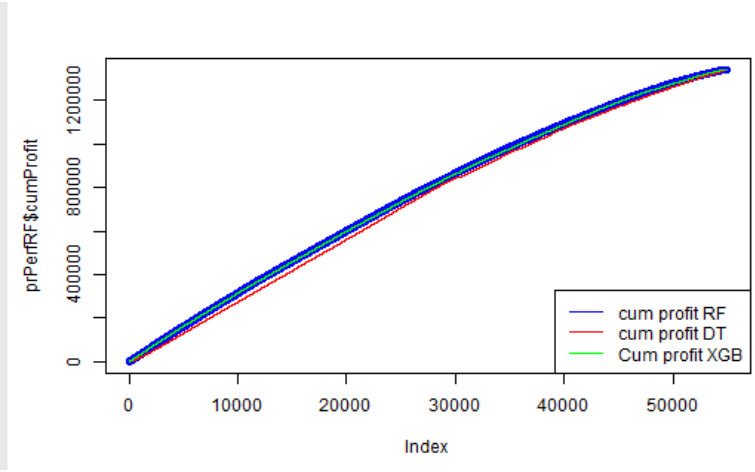

Fig 1.4: Profit Curve for Decision Tree, Random Forest and XGBoost model

decile	count	numDefaults	defaultRate	totA	totB	totC	totD	totE	totF	totG	cumDefaults	cumDefaultRate	cumDefaultLift
<int>	<int>	<int>	<dbl>	<int>	<int>	<int>	<int>	<int>	<int>	<int>	<int>	<dbl>	<dbl>
1	5498	193	0.03510	5498	0	0	0	0	0	0	193	0.03510	0.2510
2	5498	321	0.05838	4509	989	0	0	0	0	0	514	0.04674	0.3342
3	5497	468	0.08514	1811	3637	49	0	0	0	0	982	0.05954	0.4257
4	5497	536	0.09751	243	5033	206	15	0	0	0	1518	0.06903	0.4935
5	5497	668	0.12152	165	3613	1694	25	0	0	0	2186	0.07953	0.5686
6	5497	766	0.13935	96	3051	2120	230	0	0	0	2952	0.08950	0.6399
7	5497	914	0.16627	27	1348	3300	729	78	15	0	3866	0.10047	0.7183
8	5497	1091	0.19847	17	814	3646	856	136	26	2	4957	0.11272	0.8059
9	5497	1191	0.21666	13	373	2252	2517	265	71	6	6148	0.12426	0.8884
10	5497	1541	0.28033	27	101	1276	2434	1361	264	34	7689	0.13987	1.0000

Table 1.2: Defaults performance by Decile lifts for Xgboost model

decile	count	numDefaults	avgActRet	minRet	maxRet	avgTer	totA	totB	totC	totD	totE	totF	totG
<int>	<int>	<int>	<dbl>	<dbl>	<dbl>	<dbl>	<int>	<int>	<int>	<int>	<int>	<int>	<int>
1	5498	193	3.591	-32.32	11.51	2.224	5498	0	0	0	0	0	0
2	5498	321	4.259	-32.31	16.37	2.222	4509	989	0	0	0	0	0
3	5497	468	4.828	-33.33	21.78	2.237	1811	3637	49	0	0	0	0
4	5497	536	5.402	-33.33	25.11	2.222	243	5033	206	15	0	0	0
5	5497	668	5.786	-33.33	23.80	2.220	165	3613	1694	25	0	0	0
6	5497	766	5.656	-33.33	23.33	2.232	96	3051	2120	230	0	0	0
7	5497	914	6.023	-33.33	27.69	2.265	27	1348	3300	729	78	15	0
8	5497	1091	5.524	-33.33	23.58	2.300	17	814	3646	856	136	26	2
9	5497	1191	5.853	-33.33	31.90	2.294	13	373	2252	2517	265	71	6
10	5497	1541	5.261	-33.33	44.36	2.328	27	101	1276	2434	1361	264	34

Table 1.3: Returns performance by Decile lifts for Xgboost model

From Table1.3, we can see that after the 7th decile, the average actual return decreases. This may be because from the 7th decile there are a greater number of loans from lower grades than from upper grades thereby inducing risks of defaulting. This is even evident from the Table 1.2, where we can see that the lower deciles have higher default rates. The 8th decile also has the maximum return compared to the rest of the deciles hence it can be considered as the cutoff value. Also, in case of the decision tree model, the optimum decile was the 7th and for random forest it was the 8th. In this model as well, we can take the 8th decile as the cutoff.

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.

In order to forecast the loan statues for different loans, we created a generalized linear model. To get the best performance with the best parameters, we used the cross validation for glmnet. The cross-validation function for glmnet did a 10-fold cross validation, produced a plot and returned values for lambda. We used 5 different models to get to the best set of parameters to fit our glm model. To get the best performance, we used the optimal value for lambda by comparing the minimum cross validation error from each model, which is the lambda.min value. The lambda.min minimizes out-of-sample loss in cross-validation. The lambda.1se is the one which is the largest lambda value within 1 standard error of lambda.min. Lambda.1se hedges against overfitting by selecting a larger value than the minimum. We prefer to use lambda.min although it gives higher bias, because it also gives less variance (by bias-variance tradeoff), thereby giving better performance.

Model 1: We did cross validation using the default parameters for the first model with family as binomial.

Lambda Min: 0.0001968
Lambda 1se: 0.008925
AUC: 0.6829

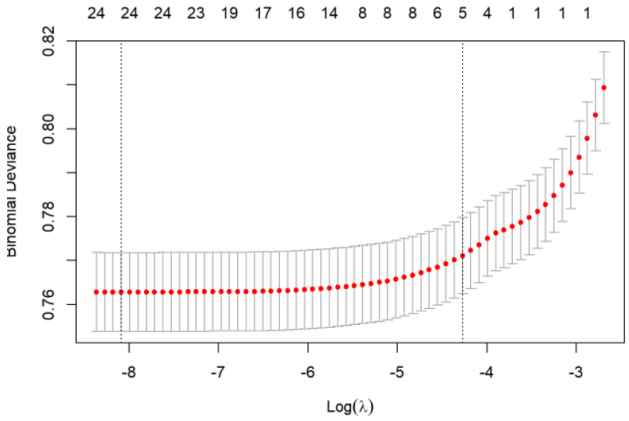

Fig 2.1: Plot for Model 1

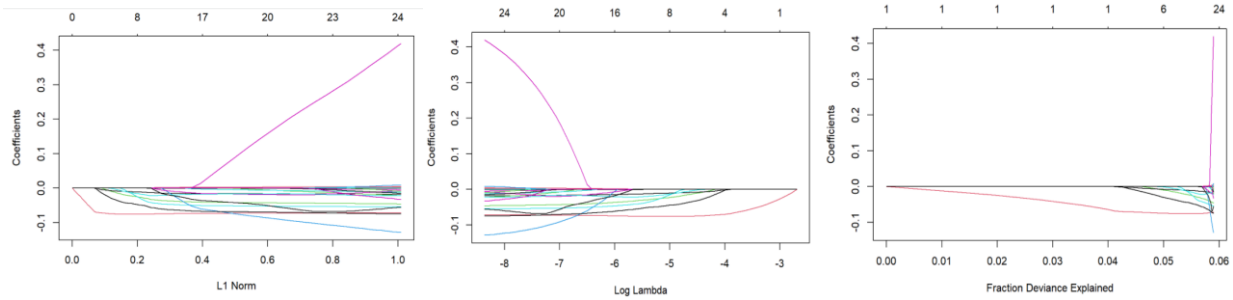

Fig 2.2: Model 1: 1. Coefficient of variables as lambda varies 2. Coefficient of variables as log of lambda varies 3. Coefficients of variables for different fraction of (null) deviance explained

Model 2: We did cross- validation with the type of measure as auc which gives the area under the ROC curve with family as binomial.

Lambda Min: 0.0003439
Lambda.1se: 0.006151
AUC: 0.6828

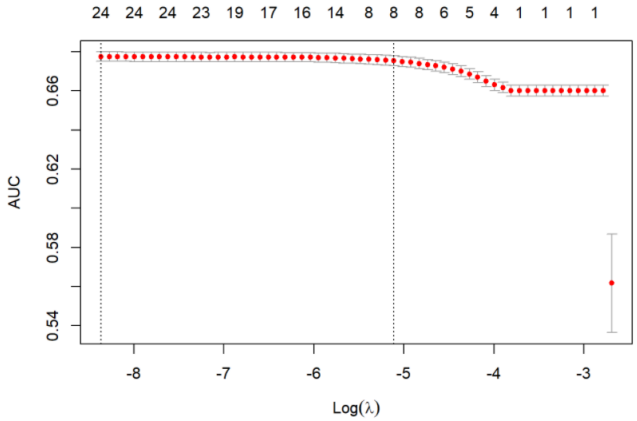

Fig 2.3: Plot for Model 2

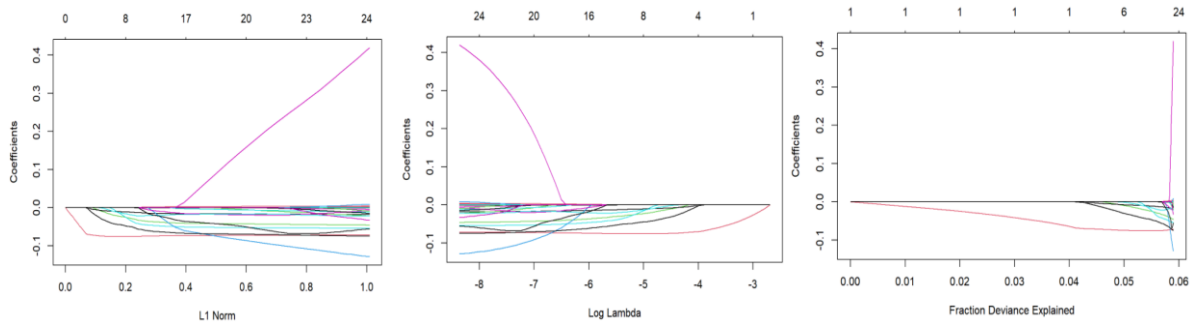

Fig 2.4: Model 2: 1. Coefficient of variables as lambda varies 2. Coefficient of variables as log of lambda varies 3. Coefficients of variables for different fraction of (null) deviance explained

When we used the lambda.min value to predict the values for this model, we got an auc of 0.6828 whereas when lambda.1se was used, we got an auc of 0.6793. The auc with lambda.min was more compared to auc of lambda.1se. It was the same with model1 as well. This further supports our assumption to use the lambda.min as a factor for measuring best performance.

Model 3: We did the cross validation with balanced data by providing the appropriate weights to the data in the model.

Lambda.min: 0.0004792
Lambda.1se: 0.01644
AUC: 0.6829

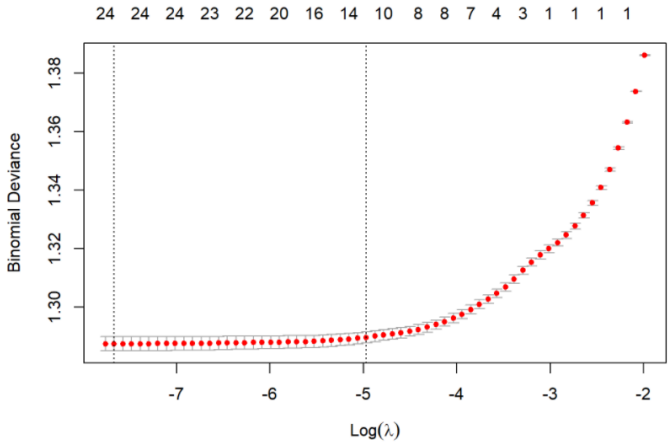

Fig 2.5: Plot for Model 3

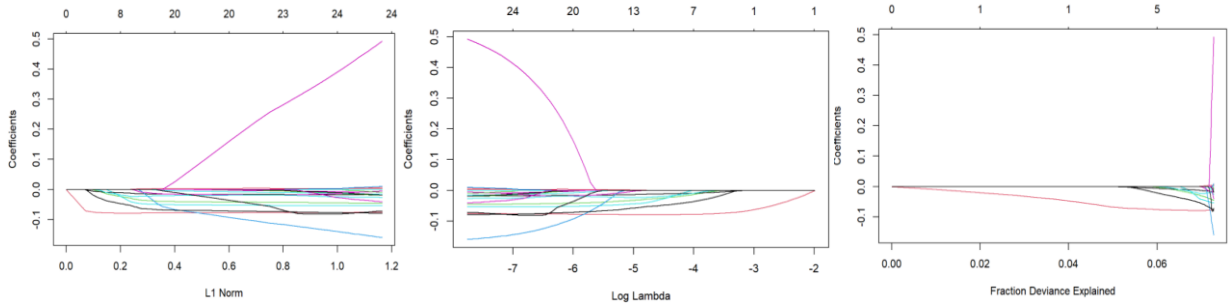

Fig 2.6: Model 3: 1. Coefficient of variables as lambda varies 2. Coefficient of variables as log of lambda varies 3. Coefficients of variables for different fraction of (null) deviance explained

Model 4: We did cross validation with ridge regression (L2 regularization) to minimize the complexity of the model. In ridge regression, a penalty parameter is added (lambda) which is equivalent to the square of the magnitude of the coefficients. This value of lambda is chosen using the cross validation. The main aim here is to make the fit small by making RSS (residual sum of squares) small by adding a shrinkage penalty. The coefficients that get too large are penalized by the shrinkage penalty. In ridge regression, the variables are not selected, instead it includes all the variables in the model. (alpha = 0)

Lambda.min: 0.00691
Lambda.1se: 0.1634
AUC: 0.6823

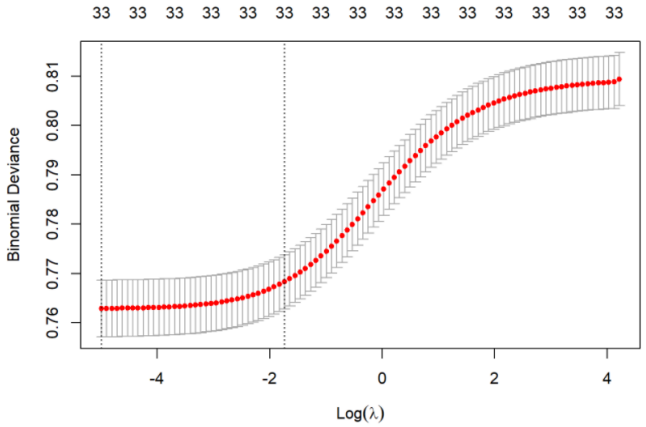

Fig 2.7: Plot for Model 4

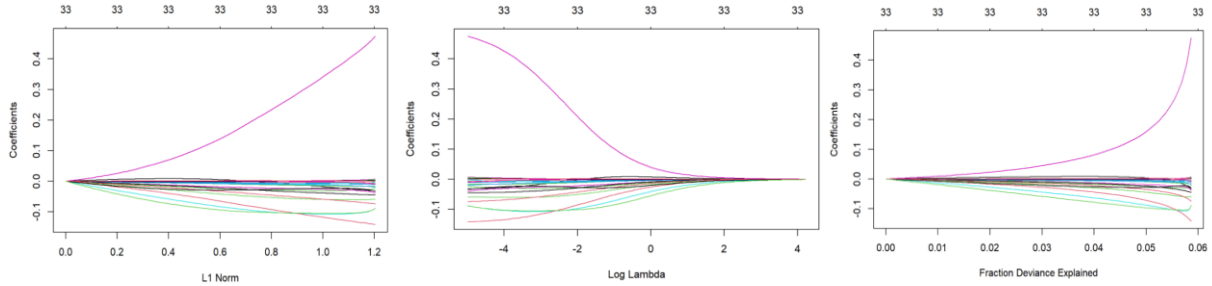

Fig 2.8: Model 4: 1. Coefficient of variables as lambda varies 2. Coefficient of variables as log of lambda varies 3. Coefficients of variables for different fraction of (null) deviance explained

Model 5: We used cross validation with lasso regression (Least Absolute Shrinkage and Selection Operator) to minimize the complexity of the model by limiting the sum of the absolute values of the model coefficients (L1 regularization). In lasso, the penalty is the sum of absolute values of the coefficients. It sets the coefficient estimates to zero and thus eliminates non-useful variables. It also performs variable selection unlike ridge regression. In this case also, the lambda is chosen by the cross validation. As lambda increases, the shrinkage happens and so the variables that are close to zero are removed. Therefore, lasso is a combination of shrinkage and variable selection. (alpha = 1)

Lambda.min: 0.0001634
Lambda.1se: 0.01075
AUC: 0.6829

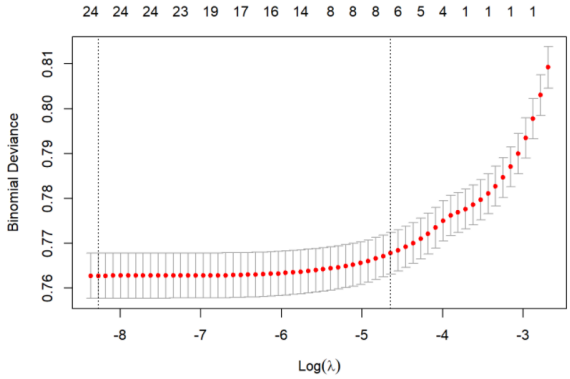

Fig 2.9: Plot for Model 5

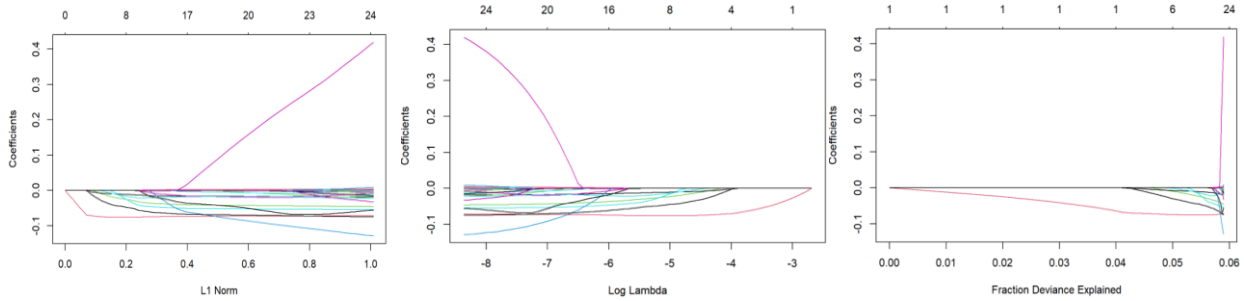

Fig 2.10: Model 5: 1. Coefficient of variables as lambda varies 2. Coefficient of variables as log of lambda varies 3. Coefficients of variables for different fraction of (null) deviance explained

When we compare the lambda.min values for the different models (Table 2.1), we see that the minimum is from the Model 5 (0.0001634), which is from lasso regression model.

Model1_Default <dbl>	Model2_AUC <dbl>	Model3_weight <dbl>	Model4_ridge <dbl>	Model5_lasso <dbl>
0.0001968	0.0003439	0.0004792	0.00691	0.0001634

Table 2.1: Lambda.min values for different models

We also tried to compare the cross-validation 'loss' at each lambda.min for the all the models (Table 2.2). We found that the minimum loss was with the Model 3 however, it had the maximum lambda.min value. There was an inverse relation between the cross-validation loss and the lambda.min values. Therefore, we considered the parameters from Model 5 as the best performing ones to build our glm model.

Model1_Default <dbl>	Model2_AUC <dbl>	Model3_weight <dbl>	Model4_ridge <dbl>	Model5_lasso <dbl>
0.7572	0.6814	0.3672	0.7577	0.7573

Table 2.2: Cross Validation loss at each lambda.min

We built a glm model with the alpha value as 1 and lambda value as the lambda.min from the lasso regression model (Model 5) for a family of binomial and type measure as deviance for our dataset. We got a lambda value of 0.0001634 same as model 5 and an auc of 0.6786.

Now that we had the best set of variables from the lasso regression model, we used them without the regularization to build our final model. The plots for the final model without the regularization is shown below.

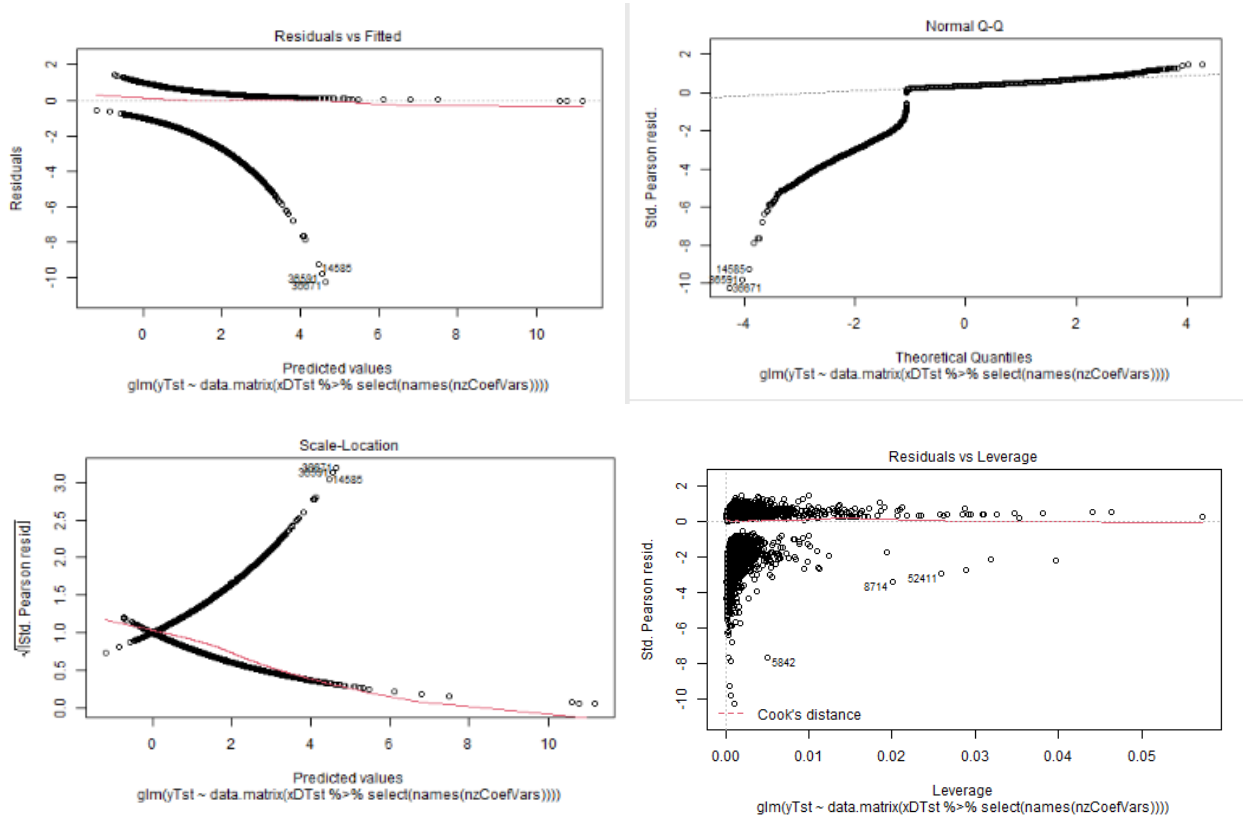

Fig 2.11: Plots from the non-regularized glm model

The fig 2.11 shows the plots from the non-regularized glm models. The top left graph is a plot of residuals vs fitted plot. Since the plot shows that the mean residuals don't change with the fitted values, but the spread of the residuals is almost funnel shaped which indicates heteroscedasticity. The top right graph is the Normal Q-Q plot which helps to detect if the residuals are normally distributed. The plot here indicates that it is bimodal. The bottom left graph is the Scale-Location plot which helps identify heteroscedasticity. Since the logistic regression models have non-constant variances in the error, they are heteroscedastic by nature. It is also evident from the almost funnel type shape of the graph. The bottom right is the residuals vs leverage plot which can help identify potential outliers. The red line indicates the Ordinary least square (OLS) fit for the data. The graph indicates that the dataset has low leverage and high standardized residual points.

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

Ans. Models are optimized by finding the optimal coefficients that minimizes the cost function for supervised learning. The cost function is basically the summation of all losses from each data point calculated with a loss function**.** The loss function for logistic regression is called **logistic loss function or Log Loss**. The log loss function is defined as:

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

where, y is the label in a labelled example and p(y) is the predicted probability of an example being y for all N examples.

The log function however heavily penalizes the samples that are far away from the desired value. Therefore, we need regularization because otherwise the model would drive the loss towards 0 for all examples along with driving the weights for each indicator feature to infinity thereby making the model completely overfit. With regularization we add a constraint on how big the coefficients can get in order to prevent overfitting. There are 2 commonly used regularization types, L1(Lasso) and L2(Ridge) which penalizes for model complexity using lambda as a regularization parameter. To get the best performance, we used the optimal value for lambda by comparing the minimum cross validation error from each model, which is the lambda.min value. The lambda.min minimizes out-of-sample loss in cross-validation. The lambda.1se is the one which is the largest lambda value within 1 standard error of lambda.min. L1 can do feature selection by making coefficients 0 for less important features and mitigate the issue of multicollinearity, while L2 also penalizes very large coefficients but doesn't make any to 0.

A link function is function of the mean of the response variable since we don't know the actual response variable. When the response variable is categorical, like in our case (loan status), we use logit function as the link function for our unknown response variable. The **logit function** is a natural log of odds.

The logit function is the default link function for the family of binomials, and we are using binomial here since we have a binary response variable – loan status, with values Charged Off or Fully Paid. The logit link function means we are doing a logistic regression for predicting a binary outcome from a set of continuous predictor variables. The below is the expression for logit link function where the beta0 is the intercept and beta1 to k are the coefficients for each feature X.

$$Ln\left(\frac{P}{1-P}\right) = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \ldots + \beta_k X_k$$

Sources:

https://www.theanalysisfactor.com/what-is-logit-function/
Understanding binary cross-entropy / log loss: a visual explanation | by Daniel Godoy | Towards Data Science
Loss Function (Part II): Logistic Regression | by Shuyu Luo | Towards Data Science

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

Ans.

Model Name	Accuracy	AUC Values	Sensitivity	Specificity
Random Forest Model	86.1%	0.676998	0.999979	0.000129
XGBoost Model	86.13%	0.6765	0.999597	0.0113445
GLM(lasso)	85.9%	0.6786	0.99877	0.00598
GLM(ridge)	85.9%	0.6773	0.99901	0.00507

From the above table since the accuracy for all the models is approximately the same, we can check for the area under the ROC curve values. The AUC value is slightly higher for generalized linear model with lasso regression. The percentage of positive cases caught, or the sensitivity is better in case of random forests than other models. Also, the specificity of the xgboost model is better. Therefore, since the maximum accuracy is obtained from the **XGBoost** model along with maximum specificity, we will consider that as the best model for the predicting the loan status.

(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. We used the library vip (variable importance plots) to get the variable importance of the different models. For the glm lasso regression model and the ridge regression model, the variable with highest importance is num_tl_120dpd_2m, followed by collections_12_mths_ex_med. In case of the lasso regression model, the subgrade has the third highest importance and in case of ridge regression, ratioOpenAccount has the third highest importance. For the random forest, installment has the highest importance followed by tot_hi_cred_lim and, whereas num_tl_120dpd_2m and collections_12_mths_ex_med has the least importance. In case of xgboost model and decision tree models, the subgrade has the maximum variable importance. As a conclusion, we could say that variables like num_tl_120dpd_2m, collections_12_mths_ex_med and subgrade are highly important for accurate prediction of the model.

(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. We divided our main dataset into training and test data with a proportion of 0.7 i.e, 70% of the data into training and remaining 30% into test dataset. Now when we performed under sampling on these datasets, 21624 out of the 76961 70% training data set was in the under-sampled training data set and 9261 out of the 32983 30% test data set was in the under-sampled test data set. Similarly, in case of over sampling, 132727 out of 76961 training data set was over sampled and 56726 out of 32983 test data was over sampled. When we applied both over and under sampling together, we got the same number of rows for both training data set and test data set as before the sampling.

Based on these under sampled, over sampled and both sampled data sets, we performed the lasso and ridge regression for the linear models. The table below shows the AUC values for the models based on these different balanced and unbalanced datasets.

Model	No Sampling	Under Sampling	Over Sampling	Both Under and Over Sampling
GLM (Lasso)	0.6786	0.6806	0.6782	0.68
GLM (Ridge)	0.6773	0.6808	0.6763	0.6781

The table below shows the Accuracy for the glm lasso and ridge regression models using unbalanced and balanced datasets.

Model	No Sampling	Under Sampling	Over Sampling	Both Under and Over Sampling
GLM (Lasso)	85.9%	63.6%	63.3%	63.2%
GLM (Ridge)	85.9%	63.3%	63.3%	63%

When we compare the AUC values for the different data sets, we see that there is an improvement in the values in case of the under-sampling data sets, but we cannot really differentiate between the two models since the AUC values for both the models are very similar. However, when we compare the accuracy of the models with these different datasets, we see that the accuracy was much better with the unbalanced data rather than with the balanced data. This makes sense as having greater training samples could produce better results. The specificity and sensitivity values are also higher for the unbalanced data rather than for the balanced data.

3. 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 – explain what performance criteria do you use, and why.

Ans. As per our analysis from the previous assignment, we calculated the actual return as the difference between the total payment and funded amount for an actual loan term multiplied by 100. The below is the summarized formula for the actual return calculation. This calculation does not include the Lending Club's service costs.

IF Fully Paid, then ((Total Payment – Funded amount)/ Funded amount) * 12/Actual Loan Term *100

IF Charged Off, then ((Total Payment – Funded amount)/ Funded amount) * 12/36 *100

We developed the generalized linear model with lasso and ridge regression and an xgboost model for the prediction of actual returns. And in order to evaluate their performances we used the Root mean square error (RMSE) which measure a model's prediction error using the below formula:

RMSE = mean ((observed - predicted) ^2) %>% sqrt()

The lower the RMSE, the better the model. Hence, we evaluated the RMSE for all our models.

GLM LASSO Regression model:

Ridge and Lasso regression are some of the simple techniques to reduce model complexity and prevent over-fitting which may result from simple linear regression.

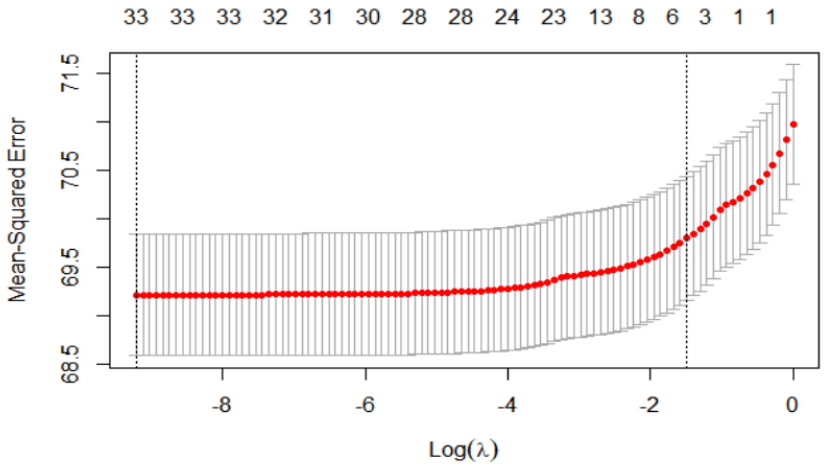

Fig: 3.1 Mean square error vs lambda plot for GLM lasso model

The Mean Squared Error is used as a default metric for evaluation of the performance of most regression algorithms. Here from the above graph it is clear that, as the log of lambda values increases mean squared error also increases from 68.76 to 70.74.

tile <int>	count <int>	avgpredRet <dbl>	numDefaults <int>	avgActRet <dbl>	minRet <dbl>	maxRet <dbl>	avgTer <dbl>	totA <int>	totB <int>	totC <int>	totD <int>	totE <int>	totF <int>	totG <int>
1	7697	7.769	1388	7.806	-32.14	44.36	2.220	17	885	2243	2859	1153	484	56
2	7696	6.650	1275	6.643	-33.33	34.26	2.223	46	2220	2910	1951	534	32	3
3	7696	6.085	1226	6.209	-33.33	33.57	2.200	177	2821	2813	1518	357	10	0
4	7696	5.650	1195	5.572	-33.33	34.20	2.232	442	3091	2741	1133	287	2	0
5	7696	5.274	1101	5.250	-33.33	34.85	2.256	975	3106	2586	865	163	1	0
6	7696	4.928	1053	4.937	-33.33	39.73	2.252	1639	3084	2356	524	93	0	0
7	7696	4.588	1020	4.384	-33.33	31.33	2.263	2340	3010	1989	302	55	0	0
8	7696	4.227	882	4.232	-32.34	26.55	2.271	3154	2916	1476	132	18	0	0
9	7696	3.809	833	3.750	-33.33	23.12	2.321	3871	2782	984	50	9	0	0
10	7696	3.088	803	3.285	-33.33	19.78	2.370	4781	2510	384	20	1	0	0

Table 3.1 Returns performance by Decile lifts for Training data for GLM lasso model

From Table 3.1, the average actual return and average predicted return amongst all is more for the first decile which is 7.769 and it is because of the three loan grades viz, loan grade “C” , “D”, and “E”. So in this case, one can think of investing in loan grades “C”, “D” and “E” as it would fetch good returns. The results obtained are for training data. Let’s have a look for the test data results.

tile <int>	count <int>	avgpredRet <dbl>	numDefaults <int>	avgActRet <dbl>	minRet <dbl>	maxRet <dbl>	avgTer <dbl>	totA <int>	totB <int>	totC <int>	totD <int>	totE <int>	totF <int>	totG <int>
1	3299	7.775	587	7.878	-33.33	31.97	2.210	7	407	936	1225	500	205	19
2	3299	6.655	537	6.647	-32.14	31.76	2.207	26	963	1223	852	216	16	3
3	3299	6.096	501	6.282	-33.33	34.24	2.204	65	1202	1195	682	151	3	1
4	3298	5.660	527	5.612	-33.33	29.17	2.214	192	1331	1140	521	114	0	0
5	3298	5.268	468	5.277	-32.18	31.10	2.247	430	1396	1066	343	63	0	0
6	3298	4.920	482	4.692	-32.20	31.98	2.265	723	1305	1001	226	43	0	0
7	3298	4.586	407	4.476	-31.14	23.71	2.257	1047	1249	830	150	22	0	0
8	3298	4.240	383	4.258	-30.37	19.06	2.284	1291	1250	678	74	5	0	0
9	3298	3.816	355	3.865	-32.31	24.81	2.276	1667	1191	415	22	3	0	0
10	3298	3.099	354	3.333	-33.33	19.61	2.368	1964	1146	179	6	3	0	0

Table 3.2 Returns performance by Decile lifts for Test data for GLM lasso model

From Table 3.2, the average actual return and average predicted return amongst all is more for the first decile which is 7.775 and it is because of the three loan grades viz, loan grade “C” , “D”, and “E”. The average term for these returns is found to be 2 years.

The root mean square error for this lasso regression comes to **8.291**.

Here further we would be performing ridge regression models in order to compare performance amongst different models.

GLM Ridge regression model (alpha =0):

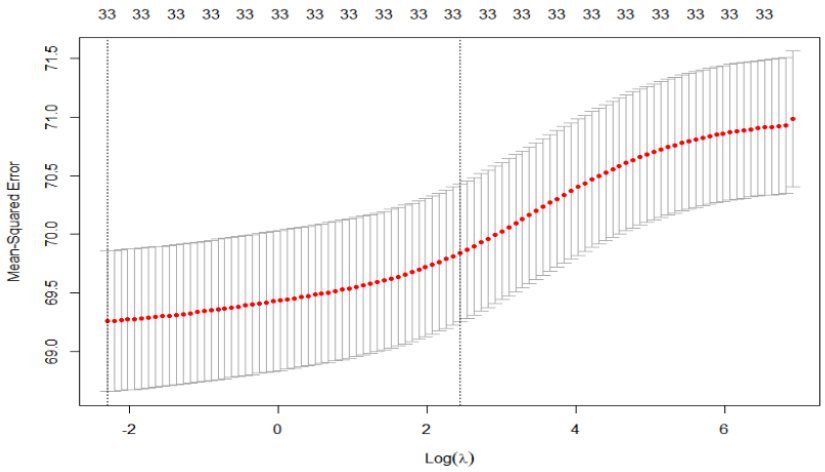

Fig: 3.2 Mean square error vs lambda plot for GLM ridge model

The estimator bias increases with λ and the estimator variance decreases with λ. The optimal level for λ is the one that minimizes the root mean squared error (RMSE). The mean squared error can be seen as gradually increasing from 69.25 to 70.75 (Fig 3.2). Here the optimal level for λ is around 0.09992 where the mean square error is 69.25.

tile <int>	count <int>	avgpredRet <dbl>	numDefaults <int>	avgActRet <dbl>	minRet <dbl>	maxRet <dbl>	avgTer <dbl>	totA <int>	totB <int>	totC <int>	totD <int>	totE <int>	totF <int>	totG <int>
1	3299	7.761	590	7.843	-33.33	31.97	2.215	7	406	926	1230	505	205	20
2	3299	6.651	535	6.696	-33.33	31.76	2.201	23	953	1228	858	218	17	2
3	3299	6.095	504	6.265	-33.33	34.24	2.203	65	1199	1207	674	152	1	1
4	3298	5.661	522	5.652	-33.33	29.17	2.215	185	1336	1141	521	114	1	0
5	3298	5.269	472	5.225	-32.18	31.10	2.251	435	1405	1063	338	57	0	0
6	3298	4.921	484	4.685	-32.20	31.98	2.256	716	1303	1005	231	43	0	0
7	3298	4.588	405	4.509	-31.14	23.71	2.259	1044	1251	829	152	22	0	0
8	3298	4.242	385	4.247	-30.37	18.59	2.287	1295	1253	677	69	4	0	0
9	3298	3.819	347	3.873	-32.31	24.81	2.274	1680	1185	408	23	2	0	0
10	3298	3.104	357	3.322	-33.33	19.61	2.372	1962	1149	179	5	3	0	0

Table 3.3 Returns performance by Decile lifts for Test data for GLM ridge model

From Table 3.3, the average predicted return is found to be more for the first decile which is 7.761 and is because of loan grades “C”,”D”, and “E”. The maximum return for this decile is 31.97 which is average term of 2 years.

The RMSE value for the above model comes to **8.295**.

GLM Model with alpha = 0.2:

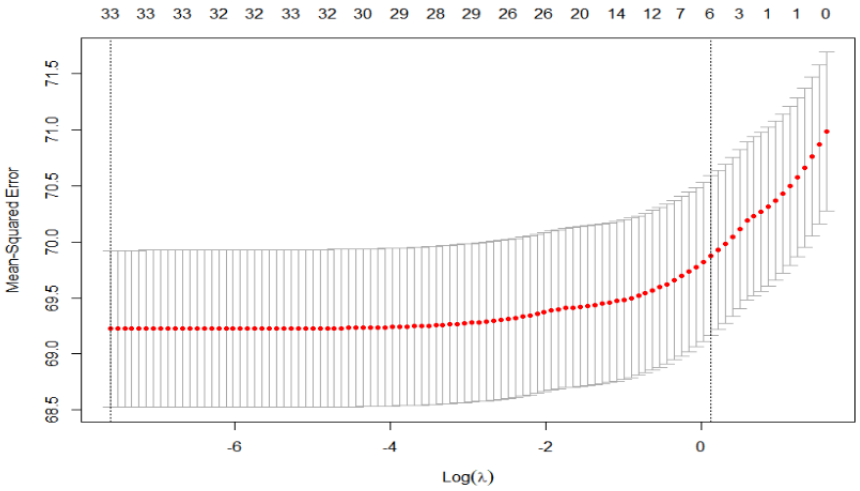

Fig: 3.3 Mean square error vs lambda plot for GLM model with alpha = 0.2

The graph in Fig 3.3 shows the mean square error starts from 69.25 and reaches to 70.25. The optimal level for lambda here is considered as lambda minimum which is 0.0004996.

tile <int>	count <int>	avgpredRet <dbl>	numDefaults <int>	avgActRet <dbl>	minRet <dbl>	maxRet <dbl>	avgTer <dbl>	totA <int>	totB <int>	totC <int>	totD <int>	totE <int>	totF <int>	totG <int>
1	3299	7.774	588	7.870	-33.33	31.97	2.211	7	407	934	1226	501	205	19
2	3299	6.654	536	6.656	-32.14	31.76	2.207	26	962	1224	852	216	16	3
3	3299	6.095	500	6.292	-33.33	34.24	2.202	66	1202	1195	682	150	3	1
4	3298	5.660	529	5.603	-33.33	29.17	2.216	191	1328	1143	521	115	0	0
5	3298	5.268	466	5.278	-32.18	31.10	2.249	429	1398	1066	342	63	0	0
6	3298	4.920	485	4.682	-32.20	31.98	2.265	724	1308	999	225	42	0	0
7	3298	4.586	405	4.488	-31.14	23.71	2.257	1047	1247	829	152	23	0	0
8	3298	4.240	383	4.252	-30.37	19.06	2.283	1290	1252	679	73	4	0	0
9	3298	3.816	354	3.875	-32.31	24.81	2.275	1668	1190	415	22	3	0	0
10	3298	3.099	355	3.324	-33.33	19.61	2.369	1964	1146	179	6	3	0	0

Table 3.4 Returns performance by Decile lifts for Test data for GLM model with alpha = 0.2

From Table 3.4, in this model, the average predicted return is found to be more for the first decile which is 7.774 and is because of loan grades “C”,”D”, and “E”. The maximum return for this particular decile is 31.97 which is average term of 2 years.

The RMSE value for the above model comes to **8.291**. So after slightly increasing the alpha value from 0 to 0.2 there is not much difference observed in the RMSE value.

GLM model with alpha = 0.5:

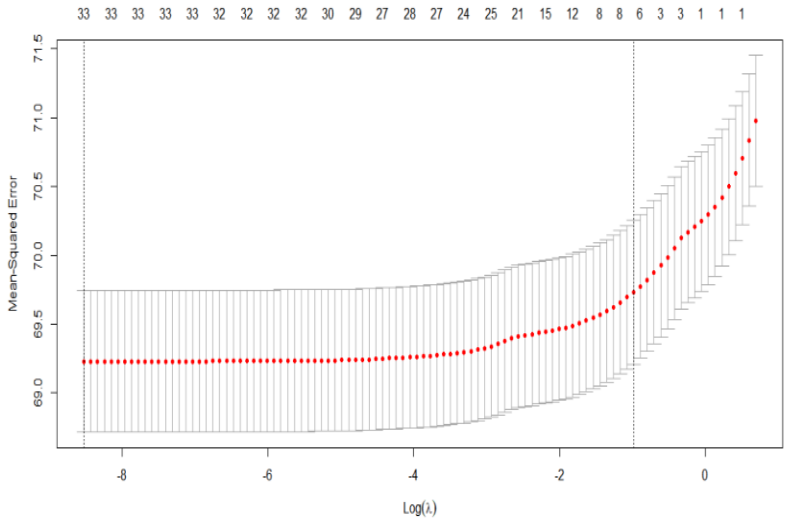

Fig: 3.4 Mean square error vs lambda plot for GLM model with alpha = 0.5

The graph in Fig 3.4 shows the mean square error starting from 69.25 and reaches to 70.95. The optimal level for lambda here is considered as lambda minimum which is 0.0001998.

tile <int>	count <int>	avgpredRet <dbl>	numDefaults <int>	avgActRet <dbl>	minRet <dbl>	maxRet <dbl>	avgTer <dbl>	totA <int>	totB <int>	totC <int>	totD <int>	totE <int>	totF <int>	totG <int>
1	3299	7.775	587	7.880	-33.33	31.97	2.211	7	406	936	1225	501	205	19
2	3299	6.655	537	6.645	-32.14	31.76	2.208	26	963	1223	853	215	16	3
3	3299	6.096	501	6.279	-33.33	34.24	2.203	65	1203	1194	681	152	3	1
4	3298	5.660	528	5.613	-33.33	29.17	2.214	192	1331	1141	521	113	0	0
5	3298	5.268	467	5.276	-32.18	31.10	2.249	430	1395	1067	343	63	0	0
6	3298	4.920	482	4.691	-32.20	31.98	2.265	722	1307	1001	225	43	0	0
7	3298	4.586	408	4.475	-31.14	23.71	2.257	1047	1248	829	151	23	0	0
8	3298	4.240	382	4.262	-30.37	19.06	2.283	1291	1251	678	74	4	0	0
9	3298	3.816	354	3.872	-32.31	24.81	2.276	1669	1189	415	22	3	0	0
10	3298	3.099	355	3.326	-33.33	19.61	2.368	1963	1147	179	6	3	0	0

Table 3.5 Returns performance by Decile lifts for Test data for GLM model with alpha = 0.5

From Table 3.5, in this model, the average predicted return is found to be more for the first decile which is 7.775 and is because of loan grades “C”,”D”, and “E”. The maximum return for this decile is 31.97 which is average term of 2 years.

The RMSE value for the above model comes to **8.291**. So after slightly increasing the alpha value from 0.2 to 0.5 there is not much difference observed in the RMSE value.

GLM Model with non-zero coefficients:

In this model, we tried to filter out our data set using the coefficients from ridge regression model which had non-zero values. We only included the non-zero coefficients from the ridge regression model to build this new model.

tile <int>	count <int>	avgpredRet <dbl>	numDefaults <int>	avgActRet <dbl>	minRet <dbl>	maxRet <dbl>	avgTer <dbl>	totA <int>	totB <int>	totC <int>	totD <int>	totE <int>	totF <int>	totG <int>
1	3299	7.775	587	7.880	-33.33	31.97	2.211	7	406	936	1225	501	205	19
2	3299	6.655	537	6.645	-32.14	31.76	2.208	26	963	1223	853	215	16	3
3	3299	6.096	501	6.279	-33.33	34.24	2.203	65	1203	1194	681	152	3	1
4	3298	5.660	528	5.613	-33.33	29.17	2.214	192	1331	1141	521	113	0	0
5	3298	5.268	467	5.276	-32.18	31.10	2.249	430	1395	1067	343	63	0	0
6	3298	4.920	482	4.691	-32.20	31.98	2.265	722	1307	1001	225	43	0	0
7	3298	4.586	408	4.475	-31.14	23.71	2.257	1047	1248	829	151	23	0	0
8	3298	4.240	382	4.262	-30.37	19.06	2.283	1291	1251	678	74	4	0	0
9	3298	3.816	354	3.872	-32.31	24.81	2.276	1669	1189	415	22	3	0	0
10	3298	3.099	355	3.326	-33.33	19.61	2.368	1963	1147	179	6	3	0	0

Table 3.6 Returns performance by Decile lifts for Test data for GLM model with non-zero coefficients

From the table 3.6, the average actual return and average predicted return is the highest in the top decile. The maximum number of loans from this decile are in the mid-grade ranges C, D and E. The maximum returns and average term of this decile is also good as compared to the other deciles.

Here the RMSE value is still = **8.291**.

Random Forest Model:

Considering random forest model to predict actual returns on training data, from the last assignment we got the best model to be random forest model 2 which has number of trees = 500. So here we have considered the same model to predict for best returns.

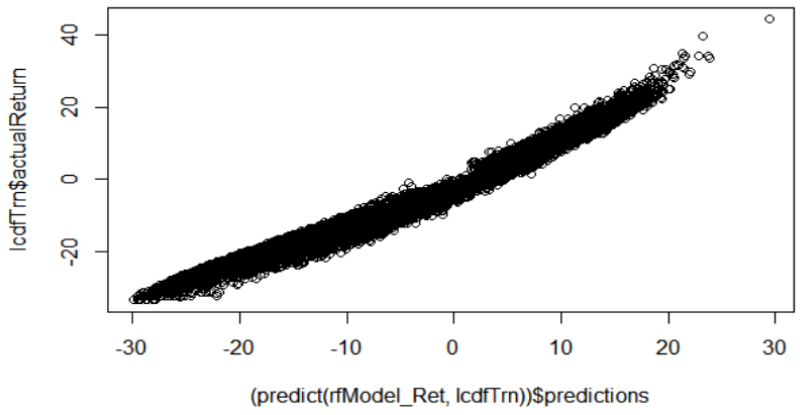

Fig: 3.6 Training data actual vs predicted returns for random forest model

From Fig 3.6, in the graph for training data, the predicted returns vs the actual returns is linearly proportional, as the actual returns increases the predicted returns also increases. This also shows that this is an ideal case.

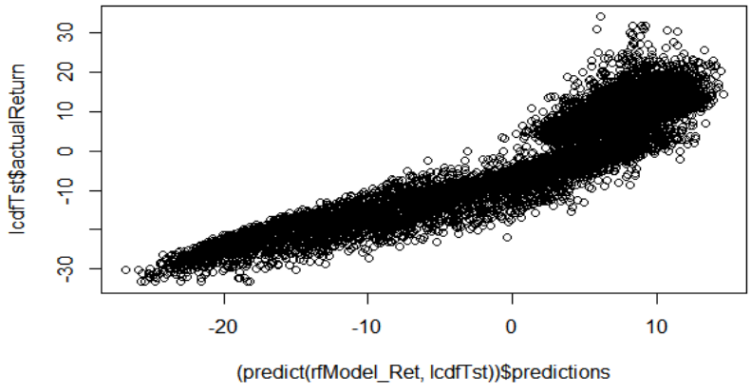

Fig: 3.7 Test data actual vs predicted returns for random forest model

From Fig 3.7, in the graph for test data, the predicted returns vs the actual returns is not exactly linearly proportional to each other as compared to the graph of training data, as the actual returns increases the predicted returns also increases slowly and steadily. There are no outliers found in test data.

tile <int>	count <int>	avgpredRet <dbl>	numDefaults <int>	avgActRet <dbl>	minRet <dbl>	maxRet <dbl>	avgTer <dbl>	totA <int>	totB <int>	totC <int>	totD <int>	totE <int>	totF <int>	totG <int>
1	7697	12.675	14	14.608	9.3577	44.3595	1.322	0	99	2201	3515	1494	352	36
2	7696	10.126	26	11.031	7.1142	16.7955	1.784	3	1202	3761	2342	378	9	1
3	7696	8.886	57	9.454	5.2904	14.2872	2.025	5	2453	4050	1135	48	5	0
4	7696	7.919	72	8.208	4.0587	13.0241	2.276	19	3535	3953	170	13	6	0
5	7696	7.083	127	7.243	1.6326	11.3898	2.416	215	4939	2458	58	20	5	1
6	7696	6.247	166	6.430	1.8110	10.1336	2.318	1211	5946	474	52	13	0	0
7	7696	5.321	212	5.491	0.7908	10.0856	2.232	3542	3954	128	55	14	3	0
8	7696	4.451	252	4.448	-1.5113	7.8259	2.491	6102	1417	107	54	14	1	1
9	7696	2.727	2154	1.827	-10.2259	7.5524	2.745	5659	789	699	382	134	28	5
10	7696	-13.557	7696	-16.675	-33.3333	-0.9183	3.000	686	2091	2651	1591	542	120	15

Table 3.7 Returns performance by Decile lifts for Training data for random forest model

From the Table 3.7, the average predicted return is observed to be more in the first decile which is 12.675 and the average actual return is 14.608 with an average term of 1 year which is better as compared to the previous model performance. These is because of the total number of loans in grades “C”, “D” and “E”. The number of defaults here is also less as compared to other deciles. The maximum number of defaults is found to be in the last decile where the returns are also low.

tile <int>	count <int>	avgpredRet <dbl>	numDefaults <int>	avgActRet <dbl>	minRet <dbl>	maxRet <dbl>	avgTer <dbl>	totA <int>	totB <int>	totC <int>	totD <int>	totE <int>	totF <int>	totG <int>
1	3299	10.763	106	12.477	-2.111	30.67	2.086	0	0	522	1974	658	136	9
2	3299	9.101	87	10.526	-1.551	31.98	2.073	0	150	2152	877	111	8	1
3	3299	8.223	113	9.312	-8.630	31.76	2.111	0	853	2122	268	41	13	2
4	3298	7.527	88	8.525	-8.019	24.81	2.141	0	1622	1566	96	11	3	0
5	3298	6.857	109	7.649	-7.329	24.68	2.201	0	2549	681	50	17	1	0
6	3298	6.056	160	6.699	-9.017	34.24	2.201	94	2874	274	41	12	2	1
7	3298	5.139	154	5.552	-13.124	23.00	2.169	1425	1751	85	28	7	2	0
8	3298	4.436	125	4.737	-15.343	18.16	2.229	2783	425	59	19	10	1	1
9	3298	3.525	382	3.170	-14.098	18.80	2.330	2811	282	109	68	25	3	0
10	3298	-9.826	3277	-16.330	-33.333	10.82	2.993	299	934	1093	680	228	55	9

Table 3.8 Returns performance by Decile lifts for Test data for random forest model

From table 3.8, the average predicted return is observed to be more in the first decile which is 10.763 and the average actual return is 12.477 with an average term of 2 year which is better as compared to the previous model performance but not better as compared to performance of training data in this model. These is because of the total number of loans in grades “C”, “D” and “E”. The average predicted return for second decile is found to be 9.101 whereas the average actual returns is 10.526. The number of defaults here is less as compared to other deciles. So the performance of this decile is also good as compared to first decile.

The RMSE value here is **10.28** which is high as compared to previous models.

XGBOOST Model:

Here using one-hot encoding we get the matrix for x and matrix for y. We made the data matrix from the above matrices. And the values are passed as hyper parameters with maximum depth as (2, 5, 6) and eta values (0.01, 0.001, 0.1).

To find the best parameters we used a for loop with nrounds = 50 and nfolds = 5, we get the parameters for best performance which is as follows:

	max_depth	eta	bestTree	RMSE_Error
1	2	0.010	50	8.855
4	2	0.001	50	9.536
7	2	0.100	50	8.321
2	5	0.010	50	8.839
5	5	0.001	50	9.534
8	5	0.100	47	8.308
3	6	0.010	50	8.836
6	6	0.001	50	9.533
9	6	0.100	43	8.315

From the above table it is clear that RMSE value is more when maximum depth = 2 and eta = 0.001. This won’t be a suitable parameter for best performance as lower the rmse value, best is the model. Thus neglecting this parameter and selecting the one which has low RMSE which is 8.308 where the maximum depth = 5 and eta = 0.1, we built the xgboost model.

tile <int>	count <int>	avgpredRet <dbl>	numDefaults <int>	avgActRet <dbl>	minRet <dbl>	maxRet <dbl>	avgTer <dbl>	totA <int>	totB <int>	totC <int>	totD <int>	totE <int>	totF <int>	totG <int>
1	7697	8.013	876	9.643	-31.01	44.36	2.073	0	566	2835	2885	1084	299	28
2	7696	6.609	1059	7.355	-33.33	34.85	2.158	0	1982	3545	1734	377	52	6
3	7696	6.058	958	6.810	-32.24	30.40	2.193	2	3219	3118	1076	253	27	1
4	7696	5.647	1170	5.789	-33.33	34.20	2.252	11	3631	2888	937	194	33	2
5	7696	5.256	1231	5.195	-33.33	31.30	2.299	223	4033	2479	771	175	13	2
6	7696	4.864	1184	4.833	-33.33	27.56	2.286	595	4175	2105	634	168	15	4
7	7696	4.447	1115	4.178	-33.33	28.46	2.303	1895	3736	1484	461	100	18	2
8	7696	4.103	900	3.751	-32.17	24.71	2.301	4271	2348	737	247	82	11	0
9	7696	3.779	776	3.400	-32.34	23.12	2.291	5475	1459	496	201	58	7	0
10	7696	3.142	1507	1.112	-33.33	22.20	2.452	4970	1276	795	408	179	54	14

Table 3.9 Returns performance by Decile lifts for Training data for xgboost model

From Table 3.9, the average predicted return is observed to be more in the first decile which is 8.013 and the average actual return is 9.643 with an average term of 2 years. These is because of the total number of loans in grades

"C", "D" and "E". The number of defaults here is the second most, less as compared to other deciles. The maximum number of defaults is found to be in the last decile where the returns are also low.

tile <int>	count <int>	avgpredRet <dbl>	numDefaults <int>	avgActRet <dbl>	minRet <dbl>	maxRet <dbl>	avgTer <dbl>	totA <int>	totB <int>	totC <int>	totD <int>	totE <int>	totF <int>	totG <int>
1	3299	7.976	497	8.345	-33.33	31.97	2.119	0	267	1157	1259	477	128	11
2	3299	6.615	512	6.779	-33.33	31.76	2.180	0	839	1509	762	162	26	1
3	3299	6.056	500	6.117	-32.20	31.98	2.220	1	1404	1260	528	92	12	2
4	3298	5.649	528	5.636	-32.24	27.15	2.254	7	1534	1215	437	90	14	1
5	3298	5.261	563	4.999	-33.33	30.67	2.296	79	1788	1040	315	69	6	1
6	3298	4.869	502	4.909	-32.16	25.67	2.275	233	1821	909	261	65	9	0
7	3298	4.463	451	4.448	-31.14	29.17	2.264	817	1586	664	181	45	4	1
8	3298	4.119	346	4.164	-32.24	34.24	2.261	1791	996	355	125	27	4	0
9	3298	3.797	290	3.812	-32.31	29.48	2.276	2319	662	204	80	29	2	2
10	3298	3.193	412	3.111	-33.33	23.89	2.388	2165	543	350	153	64	19	4

Table 3.10 Returns performance by Decile lifts for Test data for xgboost model

From Table 3.10, the average predicted return is observed to be more in the first decile which is 7.976 and the average actual return is 8.345 with an average term of 2 year which is better as compared to the previous model performance but not better as compared to performance of training data in this model. These is because of the total number of loans in grades "C", "D" and "E". The average predicted return for fifth decile is found to be 5.261 whereas the average actual returns is 4.999. The number of defaults here is more as compared to other deciles. But the returns are less as compared to first decile.

The RMSE value here is **8.268** which is the lowest value as compared to other previous models. And hence as per the fact that lower the RMSE value, best is the model performance, we can conclude that xgb model would provide best performance for this dataset.

Since the lowest RMSE is for the xgboost model, we consider that as the best model for performance.

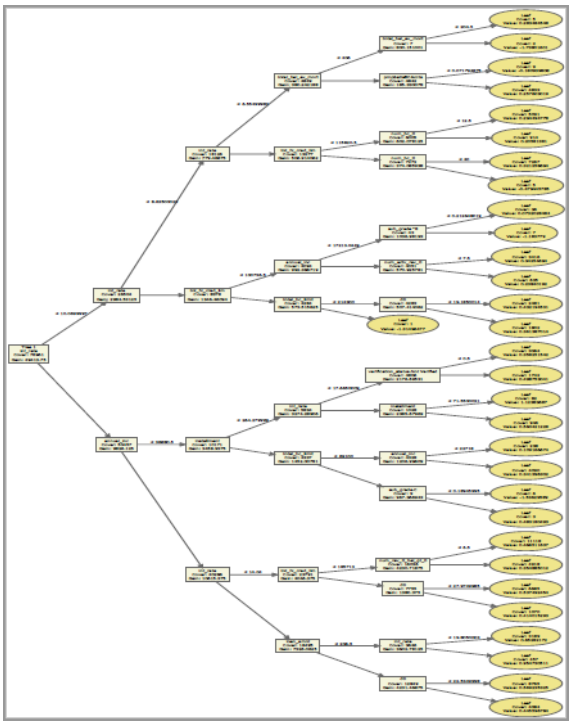

Fig 3.8: Xgboost model for actual returns

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.

As per our previous analysis, the best model for predicting the loan status was the **xgboost** model(model1). Also, in case of predicting the actual returns, the best model was the **xgboost** (model2) since it gave the least root mean square error compared to other models. Let's compare the Decile lift charts for both the models.

decile	count	numDefaults	avgActRet	minRet	maxRet	avgTer	totA	totB	totC	totD	totE	totF	totG
<int>	<int>	<int>	<dbl>	<dbl>	<dbl>	<dbl>	<int>	<int>	<int>	<int>	<int>	<int>	<int>
1	3299	119	3.583	-31.42	11.51	2.230	3299	0	0	0	0	0	0
2	3299	189	4.306	-32.31	16.37	2.220	2679	620	0	0	0	0	0
3	3299	284	4.815	-28.96	21.78	2.227	1081	2191	27	0	0	0	0
4	3298	317	5.353	-32.24	25.11	2.227	149	3026	116	7	0	0	0
5	3298	400	5.794	-30.01	23.80	2.239	105	2183	996	14	0	0	0
6	3298	456	5.679	-33.33	23.33	2.232	50	1854	1259	135	0	0	0
7	3298	557	6.074	-33.33	27.69	2.259	18	813	1962	445	49	11	0
8	3298	633	5.725	-33.33	23.58	2.270	9	477	2191	522	85	13	1
9	3298	719	5.781	-33.33	31.11	2.291	6	218	1329	1535	164	43	3
10	3298	927	5.212	-33.33	34.24	2.339	16	58	783	1443	822	157	19

Table 4.1: Returns performance by Decile lifts for Xgboost model for loan status prediction

tile	count	avgpredRet	numDefaults	avgActRet	minRet	maxRet	avgTer	totA	totB	totC	totD	totE	totF	totG
<int>	<int>	<dbl>	<int>	<dbl>	<dbl>	<dbl>	<dbl>	<int>	<int>	<int>	<int>	<int>	<int>	<int>
1	3299	7.964	508	8.346	-33.33	31.97	2.132	0	253	1112	1316	478	128	12
2	3299	6.624	519	6.599	-33.33	31.76	2.186	0	858	1479	781	153	26	2
3	3299	6.075	480	6.320	-30.91	29.92	2.223	1	1418	1275	505	86	13	1
4	3298	5.660	537	5.557	-33.33	31.98	2.244	4	1558	1235	393	95	12	1
5	3298	5.271	534	5.165	-33.33	30.67	2.274	90	1742	1074	314	68	8	2
6	3298	4.879	527	4.797	-31.42	29.17	2.285	239	1809	916	261	64	9	0
7	3298	4.472	443	4.377	-32.18	31.35	2.267	848	1580	642	187	37	4	0
8	3298	4.130	338	4.277	-30.98	25.67	2.267	1774	1012	348	126	35	2	1
9	3298	3.803	319	3.699	-32.31	29.97	2.276	2306	653	218	88	29	4	0
10	3298	3.175	396	3.183	-33.33	34.24	2.379	2150	557	364	130	75	18	4

Table 4.2: Returns performance by Decile lifts for Xgboost model for actual returns prediction

From the Table 4.1, the top deciles have lower number of loans defaulting as expected but it also has the lowest average actual return. The average actual return goes on increasing with each decile. The maximum returns also increase with the deciles. Also, with each decile, the loan in lower grades also increases. The top decile has all the loans from the highest-grade A; hence it has lower number of defaults and since their interest rates are also lower, the average actual return is also lowest. The actual return seems to be the highest in the 7th decile and it decreases after this decile. Therefore, the 7th decile could be taken as the suitable threshold and since it is dispersed among the loan grades from A to F, these loans can be considered as a safe bet for investment. In fact, the mid-range loan grades, grade B, C and D have the greatest number of loans in this decile, and they have lower interest rates than the lower grade loans.

From Table 4.2, the loans in the top decile have the highest predicted average returns as expected but they may be charged off loans as the number of defaults is high. They mostly comprise of the loans from mid-range grades like the B, C, D and E. Again, these loans have lower interest rates as compared to the lower grade loans. The default rate is not the least in the top decile, in fact it's one of the highest.

Since from both the decile charts, we see that we could get better results if we could combine both the models. Therefore, we build a third model which focuses on lower grade loans which have higher interest rates and those which have lower chances at defaulting. From the Table 4.2, we can see that there are few lower grade loan categories which may even provide higher returns. So not all the lower grade loans have lower returns. Thus, we will try to combine the loans with low default likelihood and have good returns from model1 and the higher return loans with low default likelihood from model2.

We tried to combine the model 2 based loans with higher predicted returns and sort these based on higher model 1 scores. The below table (Table 4.3) shows the decile lifts performance for this model.

tile2 <int>	count <int>	avgPredRet <dbl>	numDefaults <int>	avgActRet <dbl>	minRet <dbl>	maxRet <dbl>	avgTer <dbl>	totA <int>	totB <int>	totC <int>	totD <int>	totE <int>	totF <int>	totG <int>
1	165	7.498	12	7.715	-29.96	16.77	2.124	0	106	57	2	0	0	0
2	165	7.567	14	7.900	-27.71	16.13	1.994	0	62	97	6	0	0	0
3	165	7.522	16	7.756	-26.50	15.22	2.176	0	23	139	3	0	0	0
4	165	7.548	18	7.737	-26.89	19.23	2.040	0	29	104	32	0	0	0
5	165	7.797	15	8.651	-27.65	23.33	1.995	0	10	141	14	0	0	0
6	165	7.772	19	8.588	-25.52	19.55	2.125	0	14	91	57	2	1	0
7	165	8.093	28	7.499	-32.21	21.89	2.051	0	0	127	22	13	3	0
8	165	7.856	25	8.080	-33.33	18.95	2.079	0	2	92	63	7	1	0
9	165	7.948	20	9.111	-27.37	19.17	2.218	0	2	74	81	6	2	0
10	165	8.568	23	9.451	-22.72	22.01	2.121	0	2	13	133	14	3	0
11	165	7.981	24	9.143	-29.19	23.58	2.038	0	2	62	81	17	3	0
12	165	8.025	20	9.156	-29.77	23.00	2.038	0	1	41	96	23	4	0
13	165	8.364	26	8.591	-26.36	21.76	2.132	0	0	37	81	36	10	1
14	165	8.175	31	8.777	-33.33	26.79	2.090	0	0	11	117	33	4	0
15	165	7.722	31	7.880	-31.42	21.43	2.230	0	0	8	143	11	3	0
16	165	8.139	40	7.133	-27.36	30.42	2.333	0	0	4	108	40	12	1
17	165	7.962	32	8.199	-31.33	25.05	2.068	0	0	10	117	28	10	0
18	165	8.359	33	8.470	-26.20	26.48	2.220	0	0	3	62	78	19	3
19	165	8.274	37	8.451	-28.56	31.97	2.262	0	0	0	45	97	21	2
20	164	8.110	44	8.627	-30.82	27.83	2.316	0	0	1	53	73	32	5

Table 4.3: Returns performance by Decile lifts by combining top deciles from Model 2 ranked by model1 scores

From the Table 4.3, we see that the top decile has the lowest number of defaults, and the defaults increase with each decile. Also, the top decile has the most of their loans in the grade B and C which have lower interest rates but less chances of being charged off. Therefore, these loans are good to invest in although they don't have the maximum returns. The returns are higher as compared to Model1 and 2. Let's see another approach where we calculate the expected returns from a loan and sorting based on this value (Table 4.4).

tile2 <int>	count <int>	avgPredRet <dbl>	avgexpRet <dbl>	numDefaults <int>	avgActRet <dbl>	minRet <dbl>	maxRet <dbl>	avgTer <dbl>	totA <int>	totB <int>	totC <int>	totD <int>	totE <int>	totF <int>	totG <int>
1	165	11.310	7.249	25	10.083	-27.36	23.34	2.242	0	0	8	62	60	30	5
2	165	9.552	6.158	31	9.051	-27.14	31.97	2.040	0	0	30	78	38	17	2
3	165	8.930	5.780	25	8.881	-33.33	23.00	2.105	0	1	43	91	19	10	1
4	165	8.586	5.560	23	9.814	-32.21	24.43	2.037	0	2	48	73	36	6	0
5	165	8.313	5.407	25	8.523	-33.33	26.79	2.145	0	12	55	67	26	5	0
6	165	8.057	5.298	17	9.380	-29.18	30.42	2.026	0	18	74	55	16	1	1
7	165	7.931	5.209	17	8.890	-22.72	26.48	2.147	0	15	83	48	17	2	0
8	165	7.795	5.128	23	8.099	-29.77	19.89	2.031	0	24	67	58	12	4	0
9	165	7.719	5.063	24	7.824	-24.48	18.53	2.174	0	24	78	45	16	2	0
10	165	7.640	5.003	23	7.887	-29.96	22.02	2.128	0	23	67	58	13	4	0
11	165	7.526	4.940	25	6.933	-30.82	21.78	2.196	0	33	78	42	10	2	0
12	165	7.440	4.888	25	8.014	-27.65	19.04	2.127	0	22	78	55	8	2	0
13	165	7.364	4.847	17	8.788	-23.57	18.97	2.067	0	38	64	46	14	3	0
14	165	7.365	4.801	23	8.001	-24.01	25.76	2.141	0	20	74	53	13	5	0
15	165	7.329	4.759	21	8.554	-26.50	23.33	2.071	0	12	83	55	14	1	0
16	165	7.329	4.709	27	7.906	-28.08	20.66	2.117	0	3	76	60	20	6	0
17	165	7.278	4.651	22	8.317	-31.42	20.96	2.160	0	4	59	83	17	2	0
18	165	7.300	4.570	31	8.115	-27.26	19.84	2.192	0	2	28	103	27	5	0
19	165	7.264	4.471	41	6.849	-31.06	23.39	2.245	0	0	15	106	35	9	0
20	164	7.245	4.265	43	6.997	-28.24	27.83	2.257	0	0	4	78	67	12	3

Table 4.4: Returns performance by Decile lifts by calculating the expected returns from the loans and sorting on it

From Table 4.4, we see that the average predicted returns, average actual returns and the average expected returns is the highest in the top decile. The number of defaults is not the least in this case but based on the minimum and maximum returns for the top decile, it is better as compared to other deciles. Also, the top decile has most of the loans from grade D, E and F which are supposed to have higher interest rates. In fact, in the whole decile performance, we can

see that majority of the loans are distributed over the grades D, E and F and these have higher returns as compared to Model1 and Model2. Therefore, we can say these grades have good loan prospects for investment.

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. We created the random forest model for the lower grade loans ranging from C to G. On checking the decile performance of predictions on loan status for loans in the mid-range grades for the random forest model, we see that on training data they perform really well with no defaults at all for loans up to decile 7 with the maximum average return in the 8th decile (Table 5.1). On test data, however, it looks a bit more realistic with least number of defaults in the top decile and maximum return in the 8th decile. Grades C, D and E show the most set of loans for these deciles and hence are best for investment for good returns (Table 5.2). Fig 5.1 shows the plot for the random forest model based on the predictions for actual returns.

tile	count	avgSc	numDefaults	avgActRet	minRet	maxRet	avgTer	totA	totB	totC	totD	totE	totF	totG
<int>	<int>	<dbl>	<int>	<dbl>	<dbl>	<dbl>	<dbl>	<int>	<int>	<int>	<int>	<int>	<int>	<int>
1	3310	0.98158	0	9.229	4.735	20.272	2.440	0	0	2824	462	24	0	0
2	3310	0.96947	0	9.729	2.774	20.406	2.330	0	0	2547	680	82	1	0
3	3310	0.96023	0	10.043	5.048	23.007	2.257	0	0	2404	776	123	7	0
4	3310	0.95129	0	10.377	4.488	22.737	2.156	0	0	2259	903	136	12	0
5	3309	0.94154	0	10.706	4.059	23.238	2.100	0	0	2032	1043	219	15	0
6	3309	0.92917	0	11.103	0.000	34.846	2.000	0	0	1876	1053	331	46	3
7	3309	0.91122	0	11.878	0.000	31.902	1.829	0	0	1628	1150	442	87	2
8	3309	0.84997	218	12.411	-1.866	44.359	1.568	0	0	1401	1125	559	191	33
9	3309	0.31579	3309	-6.243	-32.211	14.396	3.000	0	0	1836	1011	368	85	9
10	3309	0.09259	3309	-19.258	-33.333	-3.576	3.000	0	0	1675	1151	386	85	12

Table 5.1: Returns performance for the random forest model on training data

tile	count	avgSc	numDefaults	avgActRet	minRet	maxRet	avgTer	totA	totB	totC	totD	totE	totF	totG
<int>	<int>	<dbl>	<int>	<dbl>	<dbl>	<dbl>	<dbl>	<int>	<int>	<int>	<int>	<int>	<int>	<int>
1	1414	0.9642	15	9.133	0.4869	16.663	2.455	0	0	1243	165	6	0	0
2	1413	0.9433	17	9.567	3.1157	18.297	2.367	0	0	1103	277	32	1	0
3	1413	0.9276	16	10.024	1.3452	18.677	2.246	0	0	1025	341	47	0	0
4	1413	0.9126	27	10.270	0.5449	27.694	2.174	0	0	964	387	59	3	0
5	1413	0.8963	33	10.536	-0.6256	21.698	2.132	0	0	840	487	80	5	1
6	1413	0.8766	37	10.833	0.0000	30.671	2.032	0	0	808	483	108	14	0
7	1413	0.8481	68	11.268	-4.8778	31.102	1.899	0	0	697	490	193	32	1
8	1413	0.7963	171	11.433	-8.8971	31.111	1.693	0	0	600	526	215	65	7
9	1413	0.5377	1111	-3.627	-29.9727	34.241	2.587	0	0	701	440	207	60	5
10	1413	0.1695	1413	-19.131	-33.3333	-5.889	3.000	0	0	682	505	173	44	9

Table 5.2: Returns performance for the random forest model on test data

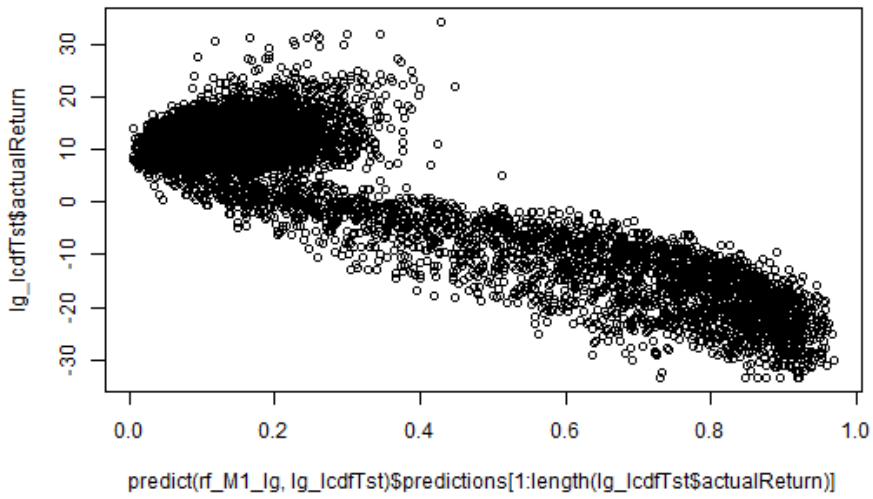

Fig 5.1: Plot of random forest model on new predictions

We created xgboost model for both prediction of loans status and actual returns for the lower grade loans. The Table 5.3 shows the returns performance of the xgboost model based on the predictions for the actual returns and Fig 5.2 shows the graph for the predictions on loan status. The 9th decile has the least number of defaults and the maximum average actual returns. This decile has most of the loans in grades C, D and E. When we look at the Table 5.4 which shows the returns performance for the xgboost model based on the predictions for the loan status, we see that bottom most decile has the least number of defaults with the maximum average actual returns. The maximum number of loans from these deciles fall under the C and D grades. The Fig 5.3 shows the graph for the predictions on actual returns.

tile	count	avgpredRet	numDefaults	avgActRet	minRet	maxRet	avgTer	totA	totB	totC	totD	totE	totF	totG
<int>	<int>	<dbl>	<int>	<dbl>	<dbl>	<dbl>	<dbl>	<int>	<int>	<int>	<int>	<int>	<int>	<int>
1	1417	0.4833	432	4.984	-33.33	30.42	2.392	0	0	4	914	394	92	13
2	1417	0.4775	357	6.279	-33.33	34.20	2.325	0	0	239	678	360	129	11
3	1417	0.4752	340	5.265	-32.17	29.92	2.254	0	0	943	407	67	0	0
4	1417	0.4734	314	6.205	-32.21	30.40	2.255	0	0	552	671	194	0	0
5	1417	0.4699	284	5.603	-33.33	34.26	2.218	0	0	1008	322	87	0	0
6	1417	0.4682	248	6.297	-31.11	23.67	2.343	0	0	1206	206	5	0	0
7	1417	0.4674	269	5.694	-30.97	24.91	2.316	0	0	1184	233	0	0	0
8	1416	0.4665	264	6.112	-29.19	34.85	2.227	0	0	1230	186	0	0	0
9	1416	0.4635	210	6.908	-33.33	23.80	2.237	0	0	1035	301	80	0	0
10	1416	0.4617	214	6.535	-28.50	23.00	2.153	0	0	1295	95	7	19	0

Table 5.3: Returns performance for the xgboost model based on predictions for actual returns

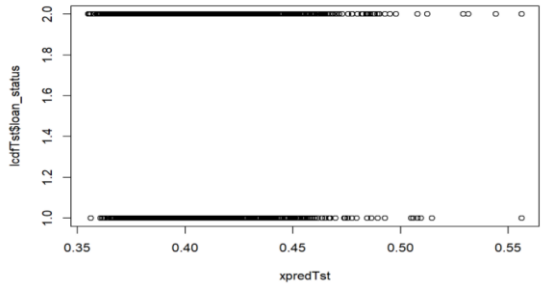

Fig 5.2: Plot of xgboost model for loan status predictions

tile	count	numDefaults	avgActRet	minRet	maxRet	avgTer	totA	totB	totC	totD	totE	totF	totG
<int>	<int>	<int>	<dbl>	<dbl>	<dbl>	<dbl>	<int>	<int>	<int>	<int>	<int>	<int>	<int>
1	1417	450	4.968	-33.33	31.97	2.408	0	0	61	552	632	150	22
2	1417	378	5.345	-33.33	31.98	2.325	0	0	278	923	180	35	1
3	1417	344	5.548	-30.89	26.63	2.272	0	0	703	601	95	18	0
4	1417	336	5.345	-31.42	34.26	2.319	0	0	827	483	85	22	0
5	1417	295	5.980	-32.16	34.20	2.262	0	0	905	429	76	6	1
6	1417	275	6.085	-33.33	23.80	2.272	0	0	1054	305	55	3	0
7	1417	251	6.358	-33.33	24.91	2.274	0	0	1128	255	32	2	0
8	1416	216	6.560	-31.40	34.85	2.256	0	0	1167	225	21	3	0
9	1416	196	6.749	-28.91	21.89	2.201	0	0	1264	136	15	1	0
10	1416	191	6.944	-29.32	23.80	2.132	0	0	1309	104	3	0	0

Table 5.4: Returns performance for the xgboost model based on predictions for loan status

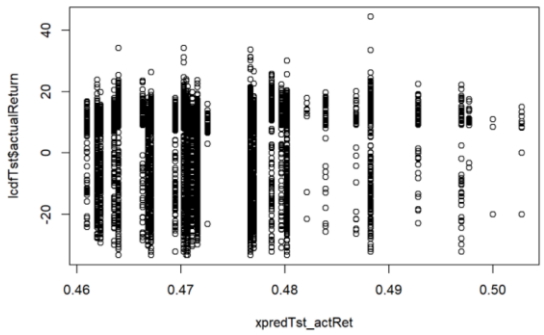

Fig 5.3: Plot of xgboost model for actual returns predictions

When we combined both the xgboost model predictions into one, we get the table 5.5. From Table 5.5, we see that until the 13th tile, there is a decrease in the number of defaults. The minimum number of defaults being in the 13th decile with the maximum average actual returns. Almost all the loans for this tile falls in the loan grade D hence being the optimal grade for a good investment.

From our analysis in the question 4, we had a set of loans with loan grades D, E and F as the optimum choice. However, from this model we can make clear decisions that loans with loan grades D are better choice for investment with good returns and lower risk of defaulting.

tile2 <int>	count <int>	avgPredRet <dbl>	avgexpRet <dbl>	numDefaults <int>	avgActRet <dbl>	minRet <dbl>	maxRet <dbl>	avgTer <dbl>	totA <int>	totB <int>	totC <int>	totD <int>	totE <int>	totF <int>	totG <int>
1	71	0.5018	0.2450	26	4.538	-30.74	24.75	2.429	0	0	0	3	34	32	2
2	71	0.4940	0.2157	27	4.206	-28.19	30.42	2.405	0	0	0	0	40	26	5
3	71	0.4921	0.2028	27	4.879	-29.38	26.56	2.428	0	0	0	4	47	15	5
4	71	0.4854	0.1902	24	6.337	-27.42	26.20	2.242	0	0	0	16	43	11	1
5	71	0.4818	0.1827	28	1.778	-32.16	19.24	2.551	0	0	0	26	44	1	0
6	71	0.4816	0.1794	31	2.958	-33.33	20.76	2.418	0	0	0	23	46	2	0
7	71	0.4817	0.1765	21	5.885	-22.89	20.66	2.381	0	0	0	34	35	2	0
8	71	0.4810	0.1740	23	3.883	-29.03	21.40	2.547	0	0	0	40	30	1	0
9	71	0.4800	0.1719	23	4.087	-28.42	19.07	2.349	0	0	0	48	23	0	0
10	71	0.4811	0.1701	27	2.930	-26.91	17.89	2.551	0	0	0	61	10	0	0
11	71	0.4803	0.1685	17	5.564	-26.25	19.12	2.411	0	0	0	62	8	1	0
12	71	0.4800	0.1668	18	5.653	-25.56	22.40	2.204	0	0	0	61	10	0	0
13	71	0.4803	0.1655	14	7.640	-32.18	20.30	2.261	0	0	0	65	6	0	0
14	71	0.4804	0.1643	21	5.099	-26.15	17.02	2.448	0	0	0	66	5	0	0
15	71	0.4800	0.1628	19	5.241	-22.93	18.96	2.528	0	0	0	69	2	0	0
16	71	0.4816	0.1613	17	6.229	-30.21	22.15	2.375	0	0	1	68	2	0	0
17	71	0.4808	0.1597	19	5.772	-29.08	18.72	2.280	0	0	1	70	0	0	0
18	70	0.4802	0.1578	15	5.323	-31.42	18.42	2.341	0	0	0	67	2	1	0
19	70	0.4802	0.1546	18	6.425	-23.40	20.28	2.259	0	0	0	67	3	0	0
20	70	0.4809	0.1458	17	5.287	-32.15	19.29	2.424	0	0	2	64	4	0	0

Table 5.5: Returns performance of combined model

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

Ans:

From the above prediction models generalized linear models, decision tree, random forest, and Xgboost, the best performance and best accuracy is obtained in Xgboost model. This can be seen through the return's performance chart of each model for training and test data of the xgboost model. The RMSE value should be as small as possible in order to obtain best model performance. And so, the RMSE value amongst all the models is low for xgboost model which is 8.268 and hence this model is recommended to use for investment prediction of loans. Also according to the above analysis, it can found that the loan grades "C","D" and "E" have the maximum predicted and actual returns, also the number of defaults is found to be nominal and hence it would be good to go ahead investing in these loan grades as compared to other loan grades. Lower grade loans after "E" have much chances of being charged off because of high interest rates, though in some cases if fully paid then the investor would achieve maximum profit but such loans are bit risky. While it can be observed that higher grade loans have comparatively lower interest rates, though the number of fully paid loans are more, but it cannot be assured that those grade loans would be always safe to invest. Hence on the basis of analysis with loan status and actual returns, loan grades "C","D","E" are recommended for investing in LC loans.