LENDING CLUB – PREDICTIVE MODELS AND CASE STUDY ASSIGNMENT 2

IDS-572 – Assignment-2

Submitted By:
Archana Singh - 668528470
Nikita Bawane - 661069000
Ritu Gangwal - 670646774

Data preparation:

As it is a continuation of Assignment 1, we have done all the data cleaning part. We have arrived at 48 variables after doing all data leakage and removing all correlated variables. We have also replaced all the NAs values in various variables with meaningful values giving explanation.

Also, we have also shown all the calculations of annual returns, actual term and actual returns in our code. All the details about data preparation is attached in the appendix of this report for further reference.

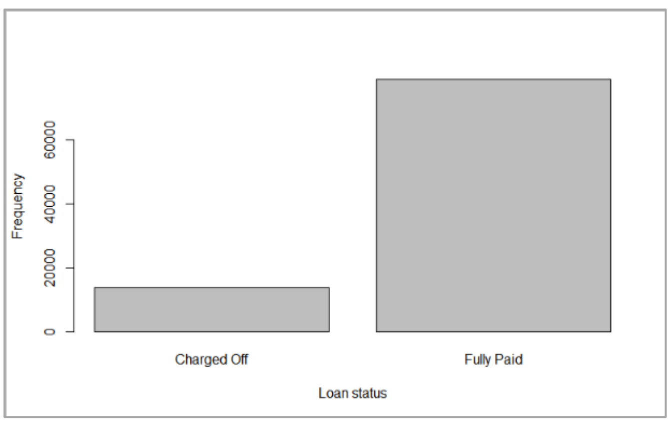
The final data now contains 48 variables/ attributes with 92624 rows/instances. We have already performed rpart, C50 and random forests on this data. In this assignment our primary focus will be on GBM, GLM and returns.

Proportion of defaults ('charged off' vs 'fully paid' loans) in the original data

The proportion of "Charged Off" vs "Fully Paid" is 14.7 : 85.3. This means approximately 15% of the total no. of loans are defaulted and rest 85% are fully paid. Total no. of charged off loans = 13652 and fully paid = 78972. Also, the data has only two type of loan_status i.e. 'Charged Off' and 'Fully Paid'.

Below is the graph showing the proportion of each type of loans:

By number:



Loan status	No. of loans
Fully Paid	78972
Charged Off	13652

Split of 'Charged Off' and 'Fully Paid' cases for each Loan-Grade

	A	B	C	D	E	F	G
Charged Off	1168	3367	4986	2833	1064	202	32
Fully Paid	21423	26156	20610	8238	2245	261	39

Split the data into training and validation sets (70:30 split):

It is always recommended to have greater values in training set in order to capture all the information and the aspects of the variables. By taking 70% as our training set, there is a high probability that we have captured almost all detailed information in our training set which is useful in making a good model.

On the other hand, if we take 50:50 split, we won't have much confidence to capture all the desired observations. Also, with smaller training set, model will simply replicate the training examples rather than generalizing the results. This might result in capturing the noise of training set and resulting in over fitting.

Hence considering the concerns that the lesser training data increases the parameter estimate variance and with lesser testing data, higher variance is expected in our performance statistics, we arrive at the conclusion to have the **70:30** data split.

<u>Balancing / resampling the data:</u>

Since the original data is highly unbalanced, we have performed balancing. In our data we have "Fully Paid" as majority class and "Charged Off" as minority class. The balancing is done only on the training data set while keeping the test data constant across all the models for better comparison of results.

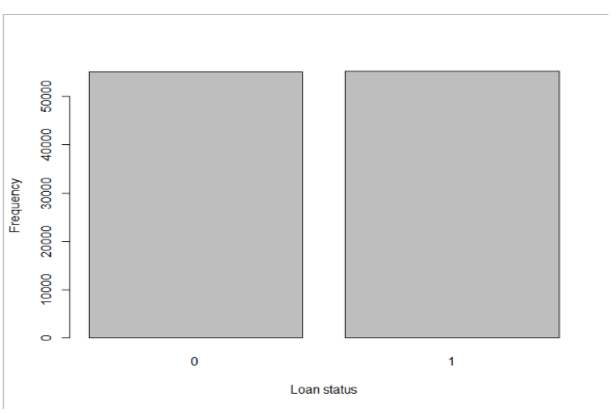
- **Over-sampling** the minority class - Oversampling duplicates examples from the minority class in the training dataset but, can result in overfitting for some models.
- **Under-sampling** the majority class – Undersampling deletes examples from the majority class and can result in losing information valuable to a model.
- **Both sampling** – It is basically mixture of both over and under sampling.

Below are the results on the training set after applying these three methods:

Data type	Loan status	No. of loans
Oversampled	Fully Paid	55225
	Charged Off	55112
Under-sampled	Fully Paid	9601
	Charged Off	9612
Both sampled	Fully Paid	32114
	Charged Off	32723

Below is the graph of the oversampled training data. We can clearly see that here both the classes have almost equal no. of observations so that the model is trained well even for unbalanced data.

Here, **1 = Fully paid and 0 = Charged Off** (We have changed the loan_status to binomial values to perform gbm and glm)



1. (a1) Develop gradient boosted models to predict loan_status. Experiment with different parameter values and identify which gives 'best' performance. How do you determine 'best' performance?

GBM Model:

Firstly, we train our model with GBM on the below defined parameters:

Input Parameters:

1. n.trees = It is a value specifying the total no. of trees used to fit. It is equivalent to the no. of iterations and the no. of basis functions in the additive expansion. Default is 100.
2. interaction.depth = It indicates the maximum depth of each tree. Default is 1.
3. n.minobsinnode = It specifies the minimum number of observations in the terminal nodes of the trees.
4. shrinkage parameter = it is applied to each tree in the expansion. Also known as the learning rate or step-size reduction; 0.001 to 0.1 usually work, but a smaller learning rate typically requires more trees. Default is 0.1.
5. bag.fraction = It is the fraction of the training set observations selected randomly to propose the next tree in the expansion. This introduces randomness into the model fit. Default is 0.5.
6. train.fraction = The first train.fraction * nrows(data) observations are used to fit the gbm and the remainder are used for computing out-of-sample estimates of the loss function.
7. cv.folds = Number of cross-validation folds to perform on the training data.
8. n.cores = It is the number of CPU cores to use.

Parameters to be considered during Performance Evaluation:

In order to estimate the best model, we have taken into consideration the values mentioned below:

1. Accuracy – It measures the overall accuracy of model classification
2. Sensitivity or Recall – It is a measure of true positive rate i.e. For "yes" how often it predicts "Yes".
3. Specificity – It is measure of true negative rate i.e. For "No", how often it predicts "No"
4. ROC – Receiver Operating Characteristics Curve
5. AUC – Area Under Curve

We evaluated the performance of the models using the confusion matrix, ROC curve and AUC values. The confusion matrix gives us the number of correct and incorrect predictions by the classification model in comparison to the actual target values in the data, which will help us understand the number of false positives and true positives. By this we can observe the accuracy of the model, which will assist us in comparing the performance of various models as well.

Below is the table of all the models of GBM we have executed with various parameter values. Here, we have used the concept of over sampling and the grid search to arrive at the best model.

Grid search - It is defined as the process of scanning the given data to configure the optimal parameters for a given model.

All the models are built with n.cores = 12, cv folds = 5, threshold = 0.5 and distribution = bernoulli

First two models are the vanilla models built with some random values of the input parameters to know the starting point:

Cases	Parameters	Training data				Test data			
		Accuracy	Specificity	Sensitivity	AUC	Accuracy	Specificity	Sensitivity	AUC
GBM model 1	Shrinkage = 0.001 interaction depth = 5 bag fraction = 0.5 Best tree = 500 min node size = 10	85.18%	100.00%	0.00%	0.69146	85.46%	100.00%	0.00%	0.68987
GBM model 2	Shrinkage = 0.05 interaction depth = 1 bag fraction = 0.5 Best tree = 1190 min node size = 30	85.25%	99.88%	1.16%	0.71325	85.45%	99.84%	0.96%	0.70239

Since the model is not working well for sensitivity, we would further like to resample our training data :

Cases	Parameters	Training data				Test data			
		Accuracy	Specificity	Sensitivity	AUC	Accuracy	Specificity	Sensitivity	AUC
GBM model 3 with oversampling	Shrinkage = 0.05 interaction depth = 2 bag fraction = 0.8 Best tree = 2000 min node size = 30	68.74%	65.27%	72.21%	0.756	64.38%	64.46%	63.96%	0.699

Here, we see that there is a drastic increase in sensitivity with bit decrease in accuracy, specificity and AUC values. Hence, in order to find optimal parameters for this model, we used grid search and found the below results:

Cases	Parameters	Training data				Test data			
		Accuracy	Specificity	Sensitivity	AUC	Accuracy	Specificity	Sensitivity	AUC
GBM model 4 with oversampling and grid search	Shrinkage = 0.1 interaction depth = 2 bag fraction = 0.5 Best tree = 1000 min node size = 5	68.62%	65.41%	71.82%	0.7552	68.81%	67.33%	70.42%	0.699

Best model – GBM model 4: On comparing all the observations of GBM, we have concluded that model 4 as our best model. This is because this models correctly trains our training data without capturing much noise, hence no over fitting. We have performed oversampling in order to predict the "charged off" category correctly. We can see that the final model gives us good accuracy, AUC, specificity and sensitivity on the test data i.e. all greater than 60%.

Accuracy alone is not a good measure of performance on unbalanced classes. Hence, it's usually better to look at the confusion matrix to better understand the model and look at metrics other than accuracy such as the sensitivity or AUC.

Performance evaluation on test data-

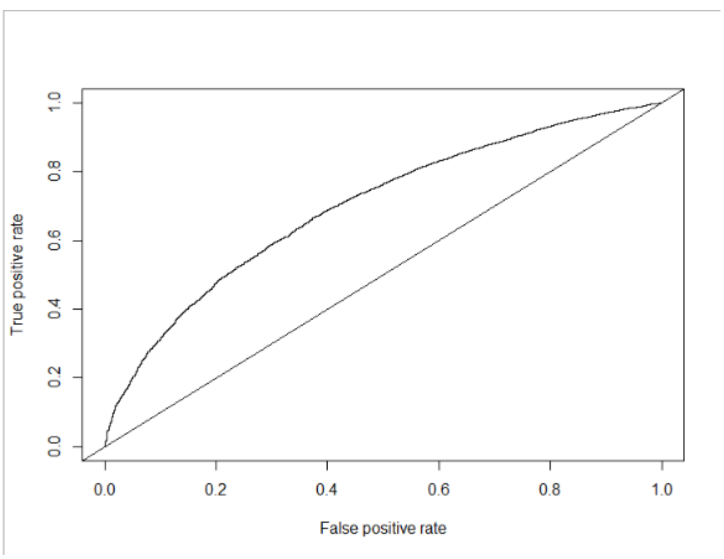
Confusion matrix for gbm model 4 on test data is explained below:

	References	
Prediction	0 (Charged Off)	1 (Fully paid)
0	2599	8448
1	1441	15299

Inference**:** Our decision tree on gbm model data predicts the following

- 2599 defaulters predicted as defaulters correctly
- 1441 defaulters predicted as fully paid incorrectly
- 8448 fully paid predicted as defaulters incorrectly and
- 15299 fully paid predicted as fully paid correctly

ROC Curve of test data:



Since ROC curve is in the N-W side of the ROC space, it means the model has better performance i.e. higher true positive rate and lower false positive rate. The AUC value i.e. area under the ROC curve = 0.699.

1.(a2) For the GBM model, what is the loss function, and gradient in the method you use? (Write the expression for these, and briefly describe).

Loss function for GBM Classification:

In the case of categorical response, the response variable *y i.e. loan_status* typically takes on binary values $y \in \{0, 1\}$, thus, if it comes from the Bernoulli distribution. The Fully paid status = 1 and Charged Odd = 0. **Hence, in our model the loss function is Bernoulli.**

Unlike in GLM, where users specify both a distribution family and a link for the loss function, in GBM distributions and loss functions are tightly coupled. In these algorithms, a loss function is specified using the distribution parameter. When specifying the distribution, the loss function is automatically selected as well.

The loss function, using *F (Model)* to predict *y(Loan Status)* is L (y,F).

Logistic loss: $L(y, F) = \ln(1 + \exp(-yF))$

Gradient: $-g(x_i) = y_i / (1 + \exp(y_i F(x_i))$

Note: logistic regression seeks f(x) to maximize likelihood of data which is equivalent to minimizing this log loss $\sum_i \ln(1 + \exp(-y_i F(x_i)))$.

1.(b1) Develop linear (glm) models to predict loan_status. Experiment with different parameter values and identify which gives 'best' performance. How do 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.

Methods of GLM:
Ridge regression: Variables with minor contribution have their coefficients close to zero. However, all the variables are incorporated in the model. This is useful when all variables need to be incorporated in the model according to domain knowledge.

Lasso regression: The coefficients of some less contributive variables are forced to be exactly zero. Only the most significant variables are kept in the final model.

Input Variables/ Parameter Values:

- x: matrix of predictor variables
- y: the response or outcome variable, which is a binary variable.
- family: the response type. We have used "binomial" for a binary outcome variable
- alpha: the elastic net mixing parameter. Allowed values include:
  "1": for lasso regression
  "0": for ridge regression
  a value between 0 and 1 (say 0.3) for elastic net regression.
- lambda: a numeric value defining the amount of shrinkage. In penalized regression, you need to specify a constant lambda to adjust the amount of the coefficient shrinkage. The best lambda for

your data, can be defined as the lambda that minimize the cross-validation prediction error rate. This can be determined automatically using the function cv.glmnet().

- Using lambda.1se, only 5 variables have non-zero coefficients. The coefficients of all other variables have been set to zero by the lasso algorithm, reducing the complexity of the model.
- *Setting lambda = lambda.1se produces a simpler model compared to lambda.min, but the model might be a little bit less accurate than the one obtained with lambda.min.*

(**Source**:http://www.sthda.com/english/articles/36-classification-methods-essentials/149-penalized-logistic-regression-essentials-in-r-ridge-lasso-and-elastic-net/)

In order to estimate the best model, we have taken into consideration the values mentioned below:

1. Accuracy – It measures the overall accuracy of model classification.
2. Sensitivity or Recall – It is a measure of true positive rate i.e. For "yes" how often it predicts "Yes". In our case, the positive class is 'Charged Off'.
3. ROC – Receiver Operating Characteristics Curve
4. AUC – Area Under Curve

GLM models on Imbalanced dataset:

As mentioned earlier, the original dataset is highly imbalanced. We have generated models on this dataset with a split of 70:30 –> 70% training data and 30% test data.

The 'Positive Class' for all the models is '0' (Charged Off).

Below **parameters** have been experimented with-

- *Alpha*: 1 (for Lasso) and 0 (for Ridge)
- *Lambda*: min.lambda and min.1se
- *Threshold*: 0.5
- *Sampling*: No Sampling, Imbalanced dataset.

Observation

- Accuracy of training and test set is in the range 84-85% both for Lasso and Ridge.
- AUC values are just above 50%.
- Sensitivity is below 10% (highlighted in Red), which is not ideal.

Hence, we can conclude that none of the models below are ideal.

Model	Input					Train Output			Test Output		
Sl.No.	Type	Sampling	Alpha	Predict Threshold	Lambda	Accuracy	Sensitivity	AUC	Accuracy	Sensitivity	AUC
1	Lasso	No Sampling	1	0.5	min.lambda = 0.0002298879	84.880%	**6.606%**	52.553%	84.892%	**6.460%**	52.348%
2	Ridge	No Sampling	0	0.5	min.lambda = 0.008071804	84.950%	**6.013%**	52.351%	85.000%	**5.866%**	52.165%
3	Lasso	No Sampling	1	0.5	min.1se = 0.0037466	84.936%	**4.453%**	51.699%	85.151%	**4.852%**	51.832%
4	Ridge	No Sampling	0	0.5	min.1se = 0.07527792	85.100%	**3.277%**	51.309%	85.295%	**3.069%**	51.177%

GLM models on Over Sampling

On sampling the data, we can see that the overall ***Accuracy of the models has reduced but the sensitivity and AUC has increased substantially.***

The test accuracy after over sampling has reduced to below 50%, as shown in the table below.

Model	Input				Train Output			Test Output		
Sl.No.	Type	Sampling	Alpha	Lambda	Accuracy	Sensitivity	AUC	Accuracy	Sensitivity	AUC
1	Lasso	Over Sampliing	1	min.lambda	62.82%	86.26%	62.85%	46.22%	85.37%	62.47%
2	Ridge	Over Sampliing	0	min.lambda	62.65%	86.26%	62.71%	46.22%	85.37%	62.47%
3	Lasso	Over Sampliing	1	min.1se	62.69%	86.22%	62.71%	46.06%	85.40%	62.38%
4	Ridge	Over Sampliing	0	min.1se	62.39%	86.54%	62.42%	46.06%	85.40%	62.38%

To improve the accuracy and get the best model, we perform *Grid Search and get the below model.*

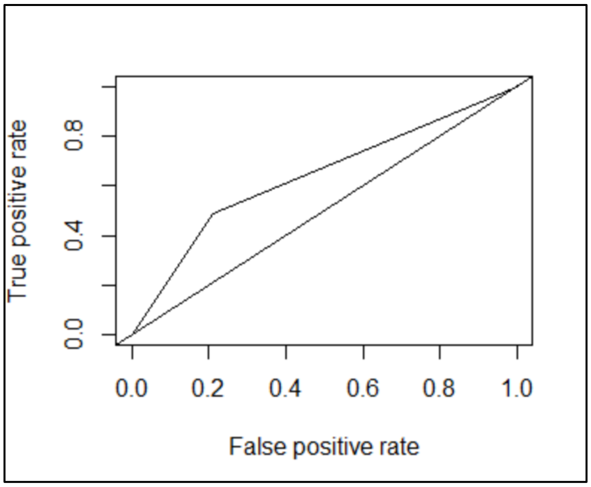
Best Model from Grid Search: The best model in GLM has the following parameter - Lasso (alpha = 1) with Over Sampling where lambda is 0.0002078539.

Model	Input				Train Output			Test Output		
Sl.No.	Type	Sampling	Alpha	Lambda	Accuracy	Sensitivity	AUC	Accuracy	Sensitivity	AUC
1	Lasso	Over Sampliing	1	0.000207854	64.36%	80.24%	64.37%	53.33%	79.08%	63.02%

Confusion Matrix – Best Model

```
Confusion Matrix and Statistics

          Reference
Prediction     0     1
         0  3195 12123
         1   845 11624

               Accuracy : 0.5333
                 95% CI : (0.5274, 0.5392)
    No Information Rate : 0.8546
    P-Value [Acc > NIR] : 1

                  Kappa : 0.1299

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

            Sensitivity : 0.7908
            Specificity : 0.4895
         Pos Pred Value : 0.2086
         Neg Pred Value : 0.9322
             Prevalence : 0.1454
         Detection Rate : 0.1150
   Detection Prevalence : 0.5513
      Balanced Accuracy : 0.6402

       'Positive' Class : 0
```

ROC Curve



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

In our model for GLM, we have used Binomial regression.

The canonical link for the binomial family is the **logit function** (also known as log odds). Its inverse is the logistic function, which takes any real number and projects it onto the [0,1] range as desired to model the probability of belonging to a class. The model parameters are adjusted by minimizing the loss function using gradient descent.

Generalized linear models are fit using the glm( ) function. The form of the glm function is

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

The link function used is "logit". The logit link function is used to model the probability of 'success' as a function of covariates (e.g., logistic regression). The purpose of the logit link is to take a linear combination of the covariate values (which may take any value between ±∞) and convert those values to the scale of a probability, i.e., between 0 and 1. The logit link:

logit(p) = ln(p/[1-p]) = β0 + β1x

or in terms of theta, it can be written as:

$$logit\,(\theta_i) = ln\left(\frac{\theta_i}{1-\theta_i}\right) = \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \cdots + \beta_U x_{iU}$$

Loss function used in binomial for our model is basically the negative of maximum likelihood function.

Hence, loss function is:

Loss function L = – y log p – (1–y) log (1–p)

where p = probability of 0 (Charged off) or 1 (fully paid)

We have used the below R code to find the loss function:

```
# Method for default

logLoss(actual, predicted, distribution = "binomial", ...)

# Method for glm

logLoss(glmmodel5, ...)
```

<u>1.(c) Compare performance of models with that of random forests (which you did in your last assignment).</u>

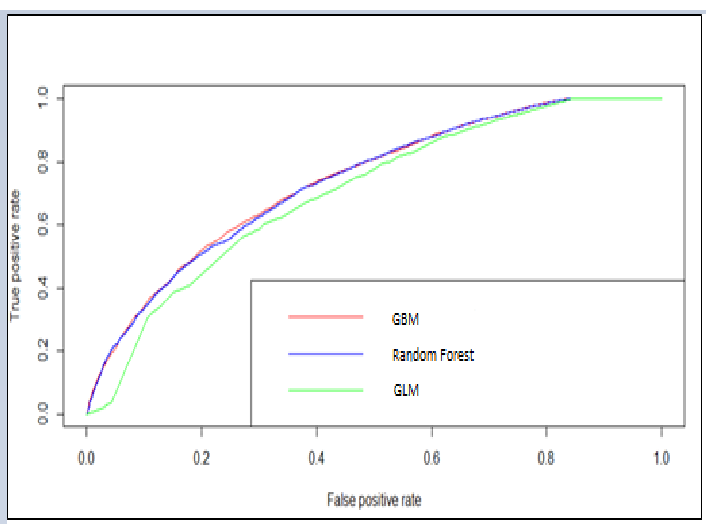
<u>Comparison between gbm model 4 and rf model 5:</u>

We chose random forest model 5 over rpart model 7 as it gives higher accuracy, AUC & sensitivity.

Models	Accuracy	Specificity	Sensitivity	AUC	Conclusion
rfmodel_5 with threshold = 0.25	68.35%	70.44%	56.06%	0.686	Based on Accuracy and AUC values, GBM is the best model
Gbm model 4	68.81%	65.41%	70.42%%	0.699	
GLM Best Model	53.33%	48.95%	79.08%	0.632	

We would prefer GBM over GLM and random forest because the idea behind GBM is boosting which will help to reduce bias unlike random forest.

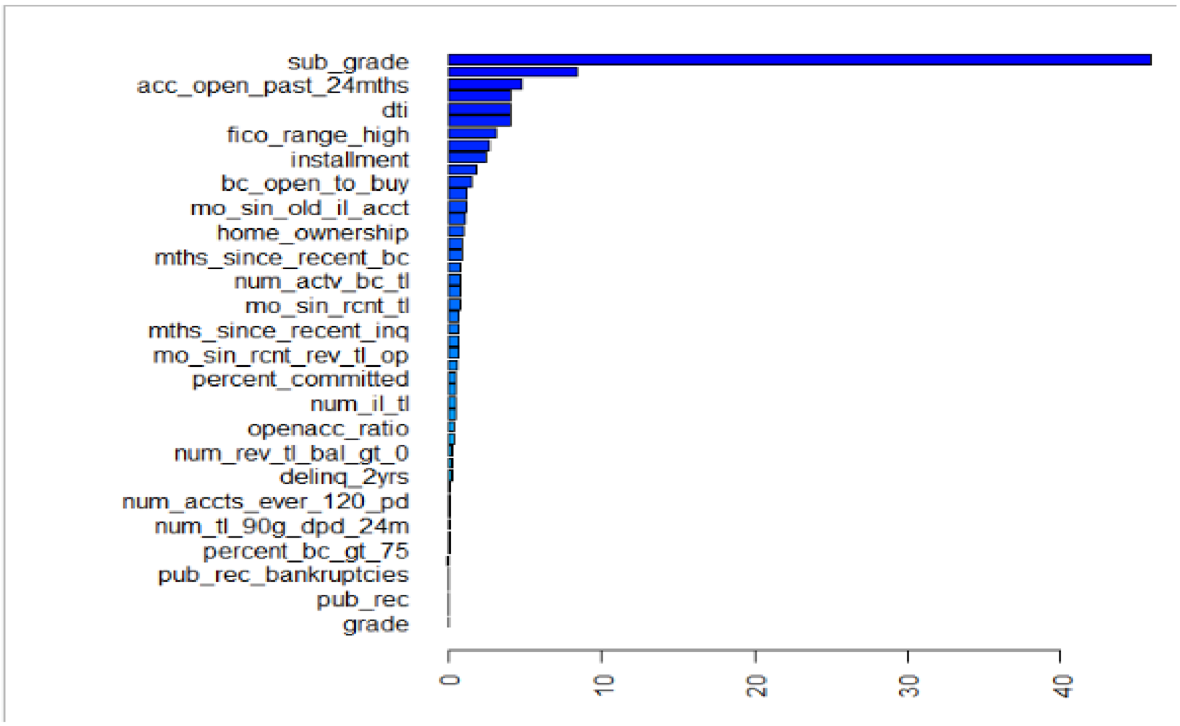
Below is the comparison of ROC curves for both the models:



Boosting is based on **weak** learners (high bias, low variance). In terms of decision trees, weak learners are shallow trees, sometimes even as small as decision stumps (trees with two leaves). Boosting reduces error mainly by reducing bias. On the other hand, Random Forest uses as you said **fully grown decision trees** (low bias, high variance). It tackles the error reduction task in the opposite way: by reducing variance. The trees are made uncorrelated to maximize the decrease in variance, but the algorithm cannot reduce bias. Hence, GBM is best with maximum ROC.

<u>1.(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?</u>

<u>Variable importance in GBM model:</u>




```
> summary(lcdf2_gbm4, las=2)
                                              var     rel.inf
sub_grade                               sub_grade 45.92316714
int_rate                                 int_rate  8.36255041
acc_open_past_24mths         acc_open_past_24mths  4.82855666
emp_length                             emp_length  4.11283715
dti                                           dti  4.06845764
curbal_open_acc                   curbal_open_acc  4.02382947
fico_range_high                   fico_range_high  3.14445287
revol_bal                               revol_bal  2.63422442
installment                           installment  2.52576192
borrHistory                           borrHistory  1.82914900
bc_open_to_buy                     bc_open_to_buy  1.50533726
annual_inc                             annual_inc  1.17974887
mo_sin_old_il_acct             mo_sin_old_il_acct  1.16144326
total_il_high_credit_limit total_il_high_credit_limit  1.09160867
home_ownership                     home_ownership  0.95141816
num_bc_tl                               num_bc_tl  0.92163837
mths_since_recent_bc         mths_since_recent_bc  0.91610205
```

Variable Importance for GLM

```
> coefs = coef(GLMmodelOS1)[,1]
> coefs = sort(abs(coefs), decreasing = T)
> coefs
                     int_rate
                 1.298961e+01
                  sub_gradeG3
                 1.654141e+00
                  sub_gradeG4
                 1.567156e+00
            num_tl_120dpd_2m
                 9.294932e-01
                  sub_gradeG1
                 8.334311e-01
      purposerenewable_energy
                 5.134919e-01
```

Below is the list of important variables for GBM and GLM best models, these variables are listed basis their rank for both the types of model:

Rank	GBM	GLM
1	sub_grade	int_rate
2	int_rate	sub_grade
3	acc_open_past_24mths	num_tl_120dpd_2m
4	emp_length	purpose
5	dti	emp_length
6	curbal_open_acc	home_ownership

S.No.	Common & Unique Variables	Variable_Name
1.	Top Ranked variables common to gbm & glm	int_rate, sub_grade,emp_length
2.	Top Ranked variables unique to gbm	acc_open_past_24mths, dti, curbal_open_acc
3.	Top Ranked variables unique to glm	num_tl_120dpd_2m, purpose, home_ownership

The above table clearly depicts the reason for both the model to perform differently, out of the top 6 ranked variables for both the models acc_open_past_24mths, dti, curbal_open_acc(GBM), num_tl_120dpd_2m, purpose, home_ownership(GLM) are unique to each model (**S.No. 2 & 3**).

1.(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?

Larger training samples (70:30 split) v/s equal training and testing samples (50:50):

Firstly, we consider 50:50 split for our data. Here we have taken two seed values to determine the consistency of our results. We have used gbm method to develop the below comparisons:

Seed Value	Accuracy (Training)	Accuracy (Test)	Difference
2204	90.36%	85.15%	5.21%
70	94.44%	80.07%	14.37%

Also, we have performed the same seed values 70:30 split.

Seed Value	Accuracy (Training)	Accuracy (Test)	Difference
2204	85.21%	85.37%	-0.16%
70	85.31%	85.13%	0.18%

The 50:50 split model is relatively unstable, because when we changed the seed value in training data, there is a change in the differences of accuracies. Whereas in 70:30 split model is more stable with mode average difference of 0.15%.

Also, it is always recommended to have greater values in training set in order to capture all the information and the aspects of the variables. By taking 70% as our training set, there is a high probability that we have captured almost all detailed information in our training set which is useful in making a good model.

On the other hand, if we take 50:50 split, we won't have much confidence to capture all the desired observations. Also, with smaller training set, model will simply replicate the training examples rather than generalizing the results. This might result in capturing the noise of training set and resulting in over fitting.

As we can see in the below above by taking 50:50 split, we are getting accuracy of training data as 94.44% which is remarkably high and is a case of overfitting.

Hence considering the concerns that the lesser training data increases the parameter estimate variance and with lesser testing data, higher variance is expected in our performance statistics, **we arrive at the conclusion to have the larger training sets for better performance.**

Balancing the training set:

Without balancing (GBM model 2) -

<table>
<tr><th rowspan="2">Cases</th><th rowspan="2">Parameters</th><th colspan="4">Training data</th><th colspan="4">Test data</th></tr>
<tr><th>Accuracy</th><th>Specificity</th><th>Sensitivity</th><th>AUC</th><th>Accuracy</th><th>Specificity</th><th>Sensitivity</th><th>AUC</th></tr>
<tr><td rowspan="5">GBM model 2</td><td>Shrinkage = 0.05</td><td rowspan="5">85.25%</td><td rowspan="5">99.88%</td><td rowspan="5">**1.16%**</td><td rowspan="5">0.71325</td><td rowspan="5">85.45%</td><td rowspan="5">99.84%</td><td rowspan="5">**0.96%**</td><td rowspan="5">0.70239</td></tr>
<tr><td>interaction depth = 1</td></tr>
<tr><td>bag fraction = 0.5</td></tr>
<tr><td>Best tree = 1190</td></tr>
<tr><td>min node size = 30</td></tr>
</table>

With balancing i.e. oversampling, Undersampling and both sampling -

Cases	Parameters	Training data				Test data			
		Accuracy	Specificity	Sensitivity	AUC	Accuracy	Specificity	Sensitivity	AUC
GBM model 4 with oversampling and grid search	Shrinkage = 0.1 interaction depth = 2 bag fraction = 0.5 Best tree = 1000 min node size = 5	68.62%	65.41%	**71.82%**	0.7552	68.81%	67.33%	**70.42%**	0.699
GBM model 5 with under-sampling and grid search	Shrinkage = 0.1 interaction depth = 2 bag fraction = 0.5 Best tree = 196 out of 1000 min node size = 5	66.15%	62.44%	69.86%	0.724	62.36%	61.41%	67.89%	0.7
GBM model 6 with both-sampling and grid search	Shrinkage = 0.1 interaction depth = 2 bag fraction = 0.5 Best tree = 1000 min node size = 5	69.86%	73.61%	66.05%	0.771	64.04%	64.00%	64.40%	0.6951

Balancing the training set for GLM:

Without Balancing:

Model	Input					Train Output			Test Output		
Sl.No.	Type	Sampling	Alpha	Predict Threshold	Lambda	Accuracy	Sensitivity	AUC	Accuracy	Sensitivity	AUC
1	Lasso	No Sampling	1	0.5	min.lambda = 0.0002298879	84.88%	**6.61%**	52.55%	84.89%	**6.46%**	52.35%

With balancing i.e. oversampling, under sampling and both sampling –

Model	Input				Train Output			Test Output		
Sl.No.	Type	Sampling	Alpha	Lambda	Accuracy	Sensitivity	AUC	Accuracy	Sensitivity	AUC
1	Lasso	Both	1	min.lambda	64.59%	80.08%	64.61%	53.61%	78.76%	64.05%
2	Lasso	Under Sampling	1	min.lambda	62.56%	86.92%	62.44%	45.11%	86.01%	62.09%
3	Lasso	Over Sampliing	1	0.000207854	64.36%	80.24%	64.37%	53.33%	79.08%	63.02%

Conclusion: We can clearly see that the performance of model increases drastically in terms of sensitivity when we balance out training data. Sensitivity of data without balancing was approximately zero and hence, even if it had a good accuracy, this model can't predict "Charged Off" class at all.

Therefore, **balanced training data gives better models.** Here we have chosen oversampling from over, under and both to retain all the actual training data.

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 models for this. Show how you systematically experiment with different parameters to find the best models. Compare model performance. Do you find larger training sets to give better models?

Calculate the annual return

Solution- The funded amount will give us the profit for each loan.

This difference when divided by difference of total payment and funded amount will give us the return of that loan.

As term for all loans is 36 months, so to be able to calculate Annual Return of each loan, we would need to pro-rate it to 12 months i.e. multiply the total return by 12/36.

Multiplication of annual return with 100 will yield the Annual Return Percentage.

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

Using the above approach, we can calculate the annual return percentage for all the loans. Mean of all these values will give us Average Annual Return Percentage, which happens to be **2.26%** for given data.

While installments of all loans is 36 months, but as per the given data some loans were closed before the expected period. For this, we need to calculate actual-term of the loan, which can be calculated by the difference of last payment date and loan issue date.

This in-turn will also impact the annual return and annualized percentage return i.e.

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

The above formula will give the **Actual Annual Return Percentage** when multiplied by 100, which can be used to calculate the average of actual annual return percentage, which happens to be **4.57%** for the given data.

Every time a borrower makes a payment Lending Club takes a **1% service fee.** This fee is rounded up or down to the nearest cent with a minimum fee of $0.01. This fee is a fixed rate and will be charged on any payment whether it is a regular payment, partial payment or a loan payoff. Hence, **service costs are included in our returns calculation where we have used total payment amount by borrower.**

Note: There will be no sampling performed in this question related to annual returns as it is a regression problem with continuous output variable (actual returns) So, we have performed our analysis on lcdf original and on oversampled one.

After calculating the annual returns, actual term and actual returns, we then started developing the models:

Random Forest:

Input parameters:

The parameters we experimented with in the random forest are

- the number of trees and
- mtry(No. of variables to choose at every split) and cv folds = 5

These two are the key parameters for random forest model apart from the common parameters for tree-based models e.g. depth, child node etc. The default value of mtry is sqrt(p) where p is the number of variables. The number of variables in our model is 48 and hence we tried mtry = 6 as well and found the performance to be similar.

Performance measure parameter:
RMSE or Root Mean Squared Error is the average deviation of the predictions from the observations. It is useful to get a gross idea of how well (or not) an algorithm is doing, in the units of the output variable. It is calculated as below:

RMSE = mean((observeds - predicteds)^2) %>% sqrt()

Below are the various observations:

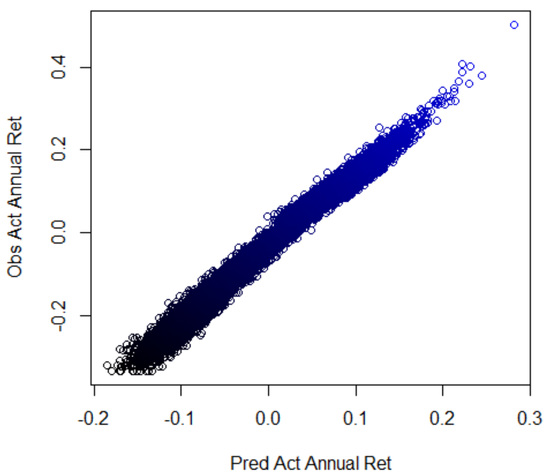
Models	Trees	mtry	Root Mean Square Error For Training Data	Root Mean Square Error For Testing Data
RF Model 1	100	6	0.03890813	0.08410515
RF Model 2	200	6	0.03866111	0.0839253
RF Model 3	500	Default	0.03851421	0.0838269
RF Model 4	**1000**	**Default**	**0.03846398**	**0.08378898**

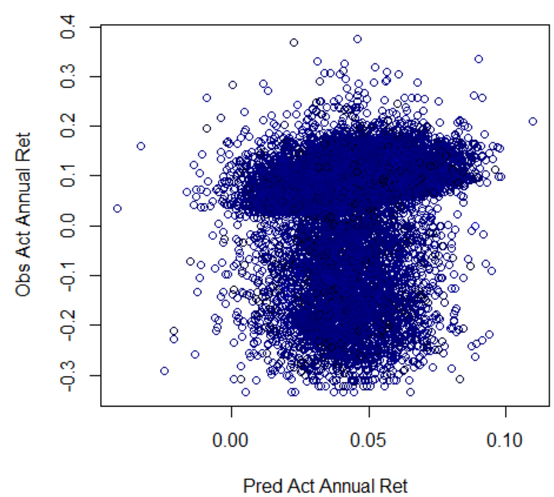
We have selected RF model 4 with 1000 trees and default mtry as our best model as it gives the minimum training and testing RMSE.

<u>The plot for the training and test predicted returns vs actual returns for rf model 4 is shown below:</u>

Training set

Test set





The plot shows that the training set has almost same predicted and observed annual returns with error of 0.038 whereas the test data has larger error of 0.0837 which is acceptable.

GBM Model:

Here we have tried experimenting with various values of interaction depth, no. of trees and shrinkage rate. We have kept bag fraction = 0.5, cv folds = 5 and no. of cores = 12 during our model development.

Models	Trees	Interaction Depth	Shrinkage	Bag Fraction	RMSE for Training Data	RMSE for Testing Data
GBM Model 1	1000	2	0.001	0.5	0.08471577	0.08414886
GBM Model 2	1000	2	0.01	0.5	0.0839576	0.08364711
GBM Model 3	2000	2	0.1	0.5	0.08397653	0.08364098
GBM Model 4	500	6	0.1	0.5	0.08338815	0.08363513
GBM Model 5	1000	6	0.1	0.5	0.08333519	0.08365349
GBM Model 6	**1000**	**2**	**0.1**	**0.5**	**0.08363214**	**0.08360752**

Best model is selected based on minimum RMSE for the test data. Hence, we achieved best model as GBM model 6 with 1000 trees and 0.1 shrinkage rate. Here, the RMSE on training data is 0.0836 and test data = 0.0836.

GLM Model:

Here we have experimented with lambda values and alpha values. CV folds = 5

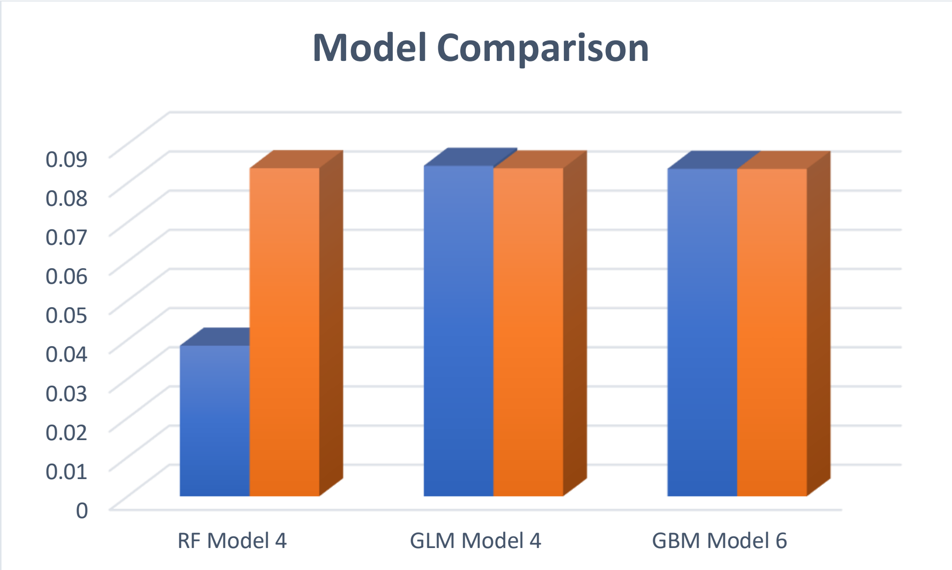
- lambda.min is the value of λ that gives minimum mean cross-validated error.
- lambda.1se gives the most regularized model such that error is within one standard error of the minimum.
- Alpha = 0 for ridge and 1 for lasso regression

Models	Type	Lambda	Alpha	RMSE For Training Data	RMSE For Testing Data
GLM Model 1	Ridge	lambda.1se	0	0.0850485	0.08447563
GLM Model 2	Ridge	lambda.min	0	0.08440592	0.08374996
GLM Model 3	Lasso	lambda.1se	1	0.08485152	0.08426033
GLM Model 4	**Lasso**	**lambda.min**	**1**	**0.08440388**	**0.08374595**

Best GLM Model is model 4 with minimum RMSE. Here, Ridge and Lasso are not significantly different. Hence, we have selected Lasso as our best model as Lasso method overcomes the disadvantage of Ridge regression by not considering the less important variables in the model.

<u>Comparison of best models for GBM, GLM and RF:</u>

Methods	Root Mean Square Error For Training Data	Root Mean Square Error For Testing Data	Difference
RF Model 4	0.03846398	0.08378898	-0.04533
GLM Model 4	0.08440388	0.08374595	0.00066
GBM Model 6	**0.08363214**	**0.08360752**	**0.00002**



Blue = Train

Orange = Test

We have selected GBM model as the best among RF, GLM and GBM as it gives the least difference between RMSE values of train and test sets. RF model is not a best one as it overfits for training data.

Also we have tried experimenting with 50:50 and 70:30 for our best model and found RMSE is large for 50:50 split for test data.

Models	Trees	Interaction Depth	Shrinkage	Bag Fraction	RMSE for Training Data	RMSE for Testing Data
GBM Model 70:30 split	1000	2	0.1	0.5	**0.08363**	**0.083607**
GBM Model 50:50 split	1000	2	0.1	0.5	0.08261	0.11467

Hence, we can conclude bigger training sets better models as we can capture all the information and the aspects of the variables. By taking 70% as our training set, there is a high probability that we have captured almost all detailed information in our training set which is useful in making a good model.

On the other hand, if we take 50:50 split, we won't have much confidence to capture all the desired observations. Also, with smaller training set, model will simply replicate the training examples rather than generalizing the results. This might result in capturing the noise of training set and resulting in over fitting.

So, larger training data gives better model.

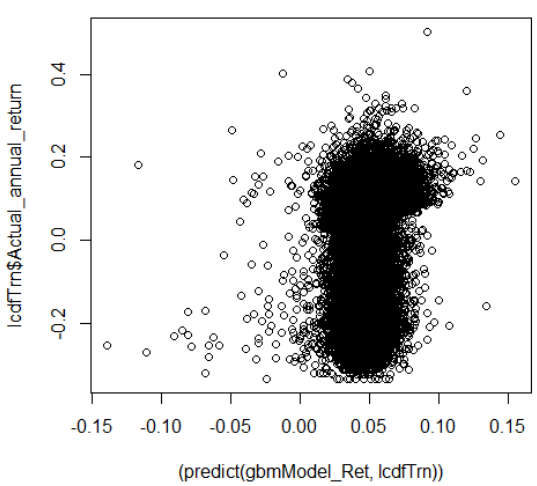
3. Considering results from Questions 1 and 2 above, how would you select loans for investment? Describe your approach and show performance?

As we have concluded from first and second that our best model is through GBM with the below parameters for estimating actual returns, we will now make our decile chart for this model.

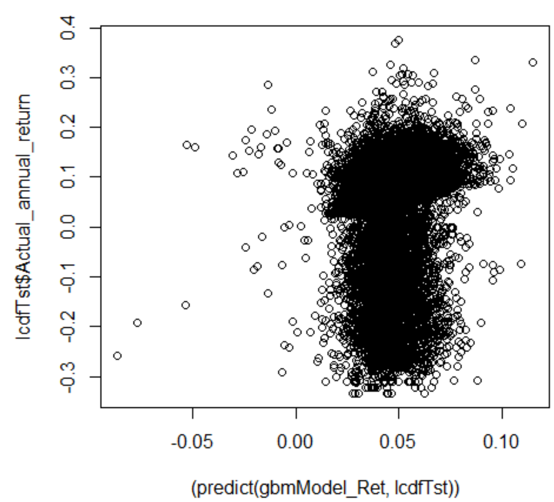
Model	Trees	Interaction Depth	Shrinkage	Bag Fraction	RMSE for Training Data	RMSE for Testing Data
GBM Model 6	**1000**	**2**	**0.1**	**0.5**	**0.08363214**	**0.08360752**

The plot for the training and test predicted returns vs actual returns for GBM model 6 is shown below:

Training set



Test set



Now, if we want to select the loans for investment, we should see how the predicted actual returns changes with deciles for both training and test data.

Train Data decile chart:

tile	count	avgpredRet	numDefaults	avgActRet	minRet	maxRet	avgTer	totA	totB	totC	totD	totE	totF
1	6484	0.067380704	935	0.07549326	-0.33333	0.501991	2.068186	0	844	2379	2669	501	72
2	6484	0.057098451	962	0.061757992	-0.33333	0.319054	2.15349	2	1978	2975	1211	304	14
3	6484	0.052362324	1054	0.053272759	-0.33333	0.328991	2.231731	30	2519	2851	851	220	13
4	6483	0.04872127	1021	0.049731955	-0.32222	0.407875	2.250651	306	2887	2400	682	192	16
5	6484	0.045679243	931	0.046007031	-0.33333	0.293408	2.261575	985	2848	1949	514	177	11
6	6484	0.043041129	906	0.04221989	-0.32283	0.255971	2.301298	1681	2817	1408	415	151	12
7	6483	0.040531703	864	0.038850023	-0.33333	0.366053	2.293776	2349	2414	1189	372	151	8
8	6484	0.037863649	870	0.035728513	-0.32177	0.38107	2.320474	3012	1958	1022	336	145	10
9	6484	0.034713961	803	0.032043317	-0.33333	0.388593	2.347058	3768	1429	853	302	112	20
10	6483	0.027673273	1251	0.019420961	-0.33333	0.402222	2.432081	3561	1106	873	410	353	145

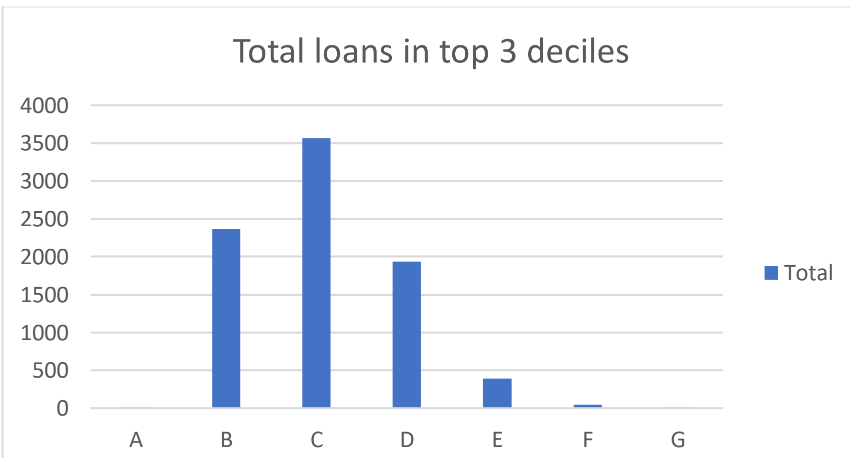
Test Data decile chart:

tile	count	avgpredRet	numDefaults	avgActRet	minRet	maxRet	avgTer	totA	totB	totC	totD	totE	totF
1	2779	0.067305961	461	0.06766436	-0.33333	0.334955	2.116999	0	325	1056	1140	224	31
2	2779	0.057139788	447	0.058523954	-0.32188	0.307129	2.165221	1	858	1257	540	110	13
3	2779	0.052309678	406	0.05779833	-0.31081	0.31735	2.196229	12	1050	1221	389	101	6
4	2778	0.048681934	442	0.050058177	-0.30966	0.374773	2.247309	145	1228	1039	275	85	6
5	2779	0.045587419	397	0.046147673	-0.33333	0.326493	2.264117	412	1222	830	236	70	9
6	2779	0.042931218	394	0.0423071	-0.32255	0.272346	2.292885	760	1117	634	184	80	4
7	2778	0.040402068	346	0.040729858	-0.32169	0.248743	2.314982	1030	1054	508	123	59	3
8	2779	0.037764385	370	0.03515947	-0.32314	0.31095	2.337009	1346	810	420	127	69	6
9	2779	0.034642943	328	0.035009886	-0.32244	0.223174	2.337437	1630	597	351	125	68	8
10	2778	0.027722453	464	0.02621374	-0.33333	0.286126	2.390907	1561	462	381	170	137	56

After analyzing the test data decile chart, we have concluded that an investor should invest in top 3 deciles of loans to gain maximum actual returns. The average prediction for these 3 top deciles for actual returns is more than 5% which is a good investment.

Also the average actual returns for these loans is more than 5.7% which is in line with our prediction made through our best model.

The top 3 deciles contains 8337 no. of loans with 1355 charged off and 6982 as fully paid.



- Here we have observed that loans with grades B and C are maximum in top 3 deciles.
- We have then examined these top deciles loans and found that loans from grade B to F have returns greater than 5%.
- Seeing loans with maximum actual returns, we see loans with grades D, E in top rows followed by C and F as below:

grade	sub_grade	loan_status	Actual_annual_return	actualTerm	int_rate	gbPredRet_tst	tile
E	E5	1	0.360384516	0.084931507	0.2099	0.094093015	1
E	E1	1	0.341771026	0.082191781	0.1825	0.063466536	1
D	D4	1	0.328991384	0.082191781	0.1757	0.05232653	3
D	D2	1	0.326493184	0.082191781	0.1655	0.051767074	3
D	D3	1	0.318847778	0.082191781	0.1699	0.069343668	1
D	D2	1	0.309844444	0.082191781	0.1655	0.058163379	2
D	D3	1	0.308451165	0.084931507	0.1699	0.065068817	1
F	F4	1	0.293659847	0.169863014	0.2499	0.059968693	2
D	D2	1	0.289252688	0.084931507	0.1655	0.056980368	2
D	D4	1	0.288058	0.082191781	0.1757	0.060329039	2
F	F4	1	0.279683743	0.334246575	0.2499	0.0680082	1
E	E4	1	0.276391771	0.167123288	0.1999	0.064098728	1
D	D1	1	0.272794302	0.084931507	0.1561	0.065060728	1
D	D4	1	0.271657333	0.082191781	0.1757	0.071135616	1
D	D2	1	0.266118182	0.082191781	0.1655	0.062204248	1
D	D2	1	0.265826458	0.082191781	0.1655	0.054538026	3
E	E2	1	0.258286559	0.169863014	0.1855	0.054801789	2
D	D4	1	0.256536129	0.084931507	0.1757	0.082332904	1
D	D1	1	0.256108333	0.082191781	0.1561	0.052410857	3
C	C2	1	0.254867333	0.082191781	0.1269	0.055976033	2

On further analyzing the default rate, grades D to F loans have default rate more than 20%. Hence, it is quite risky to invest in these loans for a risk averse person.

Row Labels	Charged Off - 0	Fully Paid - 1	Grand Total	% of defaults	Mean actual return	Mean of Predicted actual return
A	1	8	9	11.11%	0.025198032	0.051425948
B	234	2135	2369	9.88%	0.056286631	0.056366813
C	587	2979	3566	16.46%	0.05785537	0.058127301
D	403	1537	1940	20.77%	0.068973555	0.062150673
E	109	281	390	27.95%	0.062701941	0.062884652
F	15	34	49	30.61%	0.069131841	0.069530799
G	6	8	14	42.86%	-0.001845249	0.147529774
Grand Total	**1355**	**6982**	**8337**	**16.25%**		

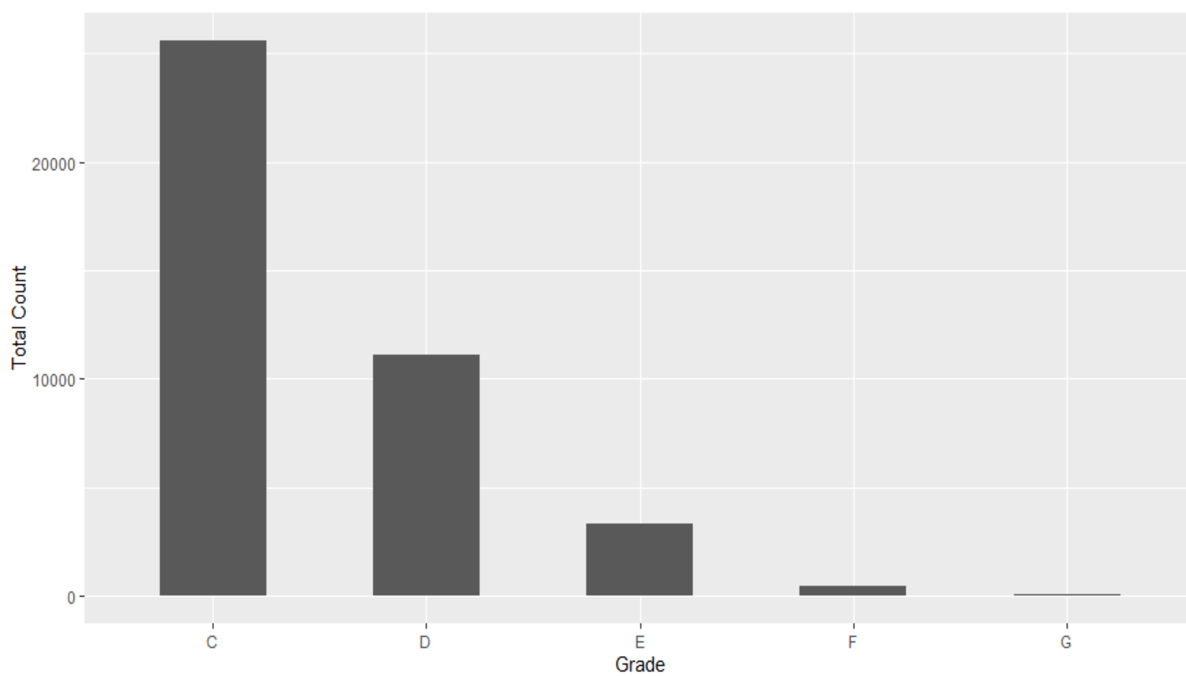
Conclusion: Going with the risk neutrality and top 3 deciles, its maximum no. of loans and less default rates, we choose to invest in grades B and C.

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.

Here, we have taken the new data set by removing grade A and grade B loans.

Below is the no. of loans according to the grades and its bar plot. We will be running our GBM, GLM and RF models on this new data set. The train : test split is 70:30.

```
> table(lcdf_grade$grade,lcdf_grade$loan_status)

        0     1
  A     0     0
  B     0     0
  C  4986 20610
  D  2833  8238
  E  1064  2245
  F   202   261
  G    32    39
```



Now, in order to check for effective investment approach, we again calculate the actual returns and develop regression models by RF, GLM and GBM.

The best models in each approach are highlighted in pink.

RF Model:

Models	Trees	Root Mean Square Error For Training Data	Root Mean Square Error For Testing Data
RF Model 1	100	0.04953938	0.1092291
RF Model 2	200	0.04913896	0.1090773
RF Model 3	500	0.0488903	0.1089316
RF Model 4	**1000**	**0.04880932**	**0.1088468**

GLM Model: Ridge and Lasso

Ridge			
Model	**s**	**Root Mean Square Error For Training Data**	**Root Mean Square Error For Testing Data**
GLM model 1	**lambda.min**	**0.1088252**	**0.108464**
GLM model 2	lambda.1se	0.1097481	0.1094467

For Lasso:

Lasso			
Model	**s**	**Root Mean Square Error for Training Data**	**Root Mean Square Error For Testing Data**
GLM model 3	**lambda.min**	**0.1088302**	**0.108473**
GLM model 4	lambda.1se	0.1095768	0.1092647

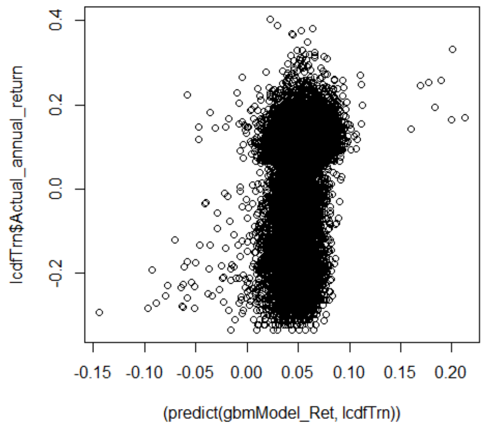
GBM model: cv = 5 and no. of cores = 12

Models	Trees	Interaction Depth	Shrinkage	Bag Fraction	RMSE for Training Data	RMSE for Testing Data
GBM Model 1	500	2	0.1	0.5	0.10833	0.1086459
GBM Model 2	1000	2	0.1	0.5	0.1075487	0.1086238
GBM Model 3	**2000**	**2**	**0.1**	**0.5**	**0.1074008**	**0.1085893**
GBM Model 4	500	6	0.1	0.5	0.1073693	0.1087685
GBM Model 5	1000	6	0.1	0.5	0.1067338	0.1088008
GBM Model 6	2000	6	0.1	0.5	0.1076024	0.1086267
GBM Model 7	2000	2	0.001	0.5	0.1090451	0.1088774
GBM Model 8	**2000**	**2**	**0.01**	**0.5**	**0.1074486**	**0.1085366**

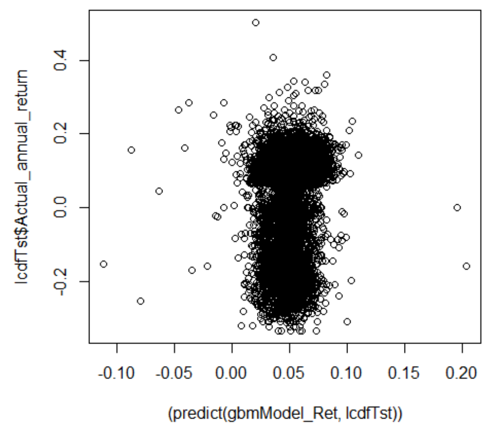
Here also we have selected GBM method as best among three because of the minimum RMSE for both training and testing data.

<u>The plot for the training and test predicted returns vs actual returns for GBM model 8 is shown below:</u>

Training set



Test set



Now, if we want to select the loans for investment, we should see how the predicted actual returns changes with deciles for both training and test data.

Train Data decile chart:

tile	count	avgpredRet	numDefaults	avgActRet	minRet	maxRet	avgTer	totA	totB	totC	totD	totE	totF
1	2836	0.074612393	346	0.089117195	-0.31325	0.331102	2.016816	0	0	1060	1436	276	48
2	2836	0.064817298	452	0.070935495	-0.33333	0.38107	2.121315	0	0	1613	988	212	23
3	2836	0.060059389	494	0.064953032	-0.30955	0.348224	2.174903	0	0	1789	840	194	11
4	2835	0.056376468	538	0.059276916	-0.33333	0.319054	2.215604	0	0	1918	711	192	11
5	2836	0.052941238	541	0.056951544	-0.32208	0.374773	2.248196	0	0	1947	712	166	10
6	2836	0.049658096	634	0.050163939	-0.33333	0.328991	2.309613	0	0	1947	689	190	10
7	2835	0.046394319	666	0.046895136	-0.32197	0.366053	2.351529	0	0	1958	664	202	9
8	2836	0.042528665	726	0.038634967	-0.33333	0.368352	2.374941	0	0	2005	591	217	19
9	2836	0.037408123	851	0.030280291	-0.32179	0.295885	2.392456	0	0	1943	618	247	25
10	2835	0.025653137	1134	0.003925584	-0.33333	0.402222	2.481999	0	0	1705	498	442	166

Test Data decile chart:

tile	count	avgpredRet	numDefaults	avgActRet	minRet	maxRet	avgTer	totA	totB	totC	totD	totE	totF
1	1216	0.074520167	193	0.075653	-0.33333	0.360385	2.104384	0	0	460	608	125	21
2	1215	0.064945495	190	0.068811	-0.30002	0.341771	2.14192	0	0	694	433	81	7
3	1215	0.060012762	224	0.061824	-0.33333	0.276392	2.210145	0	0	766	336	102	11
4	1216	0.056318071	240	0.055729	-0.32211	0.310894	2.270296	0	0	819	329	64	4
5	1215	0.052947635	255	0.051145	-0.32239	0.342408	2.296479	0	0	826	304	78	4
6	1215	0.049641751	294	0.044421	-0.33333	0.326493	2.29099	0	0	849	288	73	4
7	1216	0.046143264	268	0.04774	-0.31081	0.266725	2.329831	0	0	846	274	87	9
8	1215	0.042339268	317	0.038002	-0.33333	0.314169	2.378057	0	0	875	254	79	7
9	1215	0.037475124	317	0.040531	-0.33333	0.407875	2.376217	0	0	836	264	103	12
10	1215	0.026655837	437	0.021748	-0.32244	0.501991	2.471174	0	0	740	234	179	52

After analyzing the test data decile chart, we have concluded that an investor should invest in top 2 deciles of loans to gain maximum actual returns. The average prediction for these 2 top deciles for actual returns is more than 6.4% which is a good investment.

The top 2 deciles contains 2431 no. of loans with 398 charged off and 2033 as fully paid.

Grades	Charged off - 0	Fully paid - 1	Grand Total	Default rate
C	134	1041	1175	11.40%
D	197	823	1020	19.31%
E	57	148	205	27.80%
F	8	15	23	34.78%
G	2	6	8	25.00%
Grand Total	**398**	**2033**	**2431**	**16.37%**

- Here we have observed that loans with grades C and D are maximum in top 2 deciles.
- Seeing loans with maximum actual returns, we see loans with grades D, G in top rows followed by C and F as below:

grade	sub_grade	loan_status	Actual_annual_return	actualTerm	int_rate	gbPredRet_tst
D	D3	1	0.334955093	0.082191781	0.1699	0.078912381
F	F4	1	0.293659847	0.169863014	0.2499	0.062877044
D	D2	1	0.266118182	0.082191781	0.1655	0.077344453
D	D4	1	0.262440046	0.084931507	0.1757	0.067234632
D	D4	1	0.256536129	0.084931507	0.1757	0.081158965
G	G3	1	0.248743464	1.084931507	0.2788	0.069105076
D	D3	1	0.247783978	0.084931507	0.1699	0.071309757
G	G4	1	0.244551265	0.835616438	0.2849	0.130249849
D	D3	1	0.239482319	0.082191781	0.1699	0.082968573
E	E5	1	0.233852019	0.419178082	0.2099	0.093595623
C	C3	1	0.232859908	0.084931507	0.1333	0.062835205
C	C5	1	0.23232908	0.084931507	0.1465	0.063812188
F	F1	1	0.231663345	0.419178082	0.2199	0.072718723
E	E2	1	0.231516731	0.252054795	0.1855	0.067500135
D	D2	1	0.230283041	0.169863014	0.1655	0.075053627
D	D1	1	0.223173566	0.167123288	0.1561	0.067580112
C	C3	1	0.222698667	0.082191781	0.1333	0.063277239
F	F1	1	0.222314529	0.504109589	0.2199	0.063310882
C	C2	1	0.221643306	0.084931507	0.1269	0.062647026
D	D3	1	0.221453279	0.167123288	0.1699	0.084279808
E	E5	1	0.219715686	0.419178082	0.2099	0.100144944

On further analyzing the default rate, grades E to F loans have default rate more than 20%. Hence, it is quite risky to invest in these loans for a risk averse person.

Conclusion: Going with the risk neutrality and top 2 deciles, its maximum no. of loans and less default rates, we choose to invest in grades C and D. This is a better investment approach than ques.3 as it considers the lower grade loans with high returns, less default rates and average risk involved.

APPENDIX

Generate some new derived attributes which you think may be useful for predicting default

a. Proportion of Satisfactory bankcard accounts – Number of Satisfactory Bankcard Account is a subset of Number of Bankcard Account, so these two can be clubbed together to form a derived variable i.e. PropSatisBankcardAccts = 0, if num_bc_tl=0, else (num_bc_sats /num_bc_tl)

b. Ratio of total open accounts to total accounts – Number of Total Open Account is a subset of Number of Total Accounts, so these two can be merged to form a derived variable i.e. openacc_ratio = open_acc/total_acc (In the given data total_acc != 0)

c. Ratio of Funded Amount Invested to Loan Amount – Funded Amount Invested can either be equal to or lesser than Loan Amount and this relationship can be used to form another derived variable i.e. the percentage amount that an investor has committed to the loan borrower percent_committed = funded_amnt_inv/loan_amnt

d. Ratio of Total Current Balance of All Accounts to Number of Open Credit Lines in the borrower's credit file – The former parameter is a subset of the later and this relationship can be used to form another derived product curbal_open_acc = tot_cur_bal/open_acc

e. Ratio of funded amount to installment – The ratio of funded amount with installment gives the duration in which the total amount will get paid i.e. installmentamount = funded_amnt/installment

Treatment of NA values:

There are missing values in the given data. Once we removed the variables with all records as 'NA'(49 variables), we were left with **108** (157-49) variables out of total 157 (150 + ActualTerm + 6 derived) variables. After removing variables with more than 60% of missing data(list of those missing columns is mentioned below), we are left with 98 variables.

Variables with all 'NA' **Count of variables removed = 49**	id, member_id, url, desc, next_pymnt_d, annual_inc_joint, dti_joint, verification_status_joint, open_acc_6m, open_act_il, open_il_12m, open_il_24m, mths_since_rcnt_il, total_bal_il, il_util, open_rv_12m, open_rv_24m, max_bal_bc, all_util, inq_fi, total_cu_tl, inq_last_12m, revol_bal_joint, sec_app_fico_range_low, sec_app_fico_range_high, sec_app_earliest_cr_line, sec_app_inq_last_6mths, sec_app_mort_acc, sec_app_open_acc, sec_app_revol_util, sec_app_open_act_il, sec_app_num_rev_accts, sec_app_chargeoff_within_12_mths, sec_app_collections_12_mths_ex_med, sec_app_mths_since_last_major_derog, hardship_type, hardship_reason, hardship_status, deferral_term, hardship_amount, hardship_start_date, hardship_end_date, payment_plan_start_date, hardship_length, hardship_dpd, hardship_loan_status, orig_projected_additional_accrued_interest, hardship_payoff_balance_amount, hardship_last_payment_amount

Number of variables left = 151+6-49=108

Columns with more than 60% 'NA'	mths_since_last_record, mths_since_last_major_derog, mths_since_recent_bc_dlq, mths_since_recent_revol_delinq, debt_settlement_flag_date, settlement_status, settlement_date, settlement_amount, settlement_percentage, settlement_term

Number of variables left = 108-10=98

The below list of 12 variables are the ones which has missing values. We have replaced the missing variables for these columns either with mean, median or zero basis nature of the attributes.

Missing values replaced with mean, median or zero for these variables	emp_title, mths_since_last_delinq, revol_util, last_pymnt_d, bc_open_to_buy, bc_util, mo_sin_old_il_acct, mths_since_recent_bc, mths_since_recent_inq, num_tl_120dpd_2m, percent_bc_gt_75, actualTerm, Actual_annual_return

The table below indicates logic used for replacement of NA for the above variables:

S.No.	Column Name	Logic Used for replacement of Missing Values
1	mths_since_last_delinq	This column has 48% missing values which is because of no delinquency, so we can replace it by max value (170) or higher, we will experiment this replacement in lcx dataset
2	revol_util	Replaced by median
3	bc_open_to_buy	Replaced by median
4	bc_util	For this column, mean of data is 63 whereas third and max quartile is 84.5 and 202 respectively which indicates presence of outlier, hence replaced by median
5	mo_sin_old_il_acct	This column is for months since oldest bank account was opened, hence makes sense to replace with zero
6	mths_since_recent_bc	This column is for months since recent bankcard account was opened, replacing missing values with zero for it as well
7	#percent_bc_gt_75	As percentage of missing values for this column is just 1%, so we can replace the missing values by median for it
8	num_tl_120dpd_2m	Replacing with zero
9	mths_since_recent_inq	This column gives us months since recent inquiry was done, missing values indicate that this person may not have applied for a loan before, hence replacing NA with Zero
10	actualTerm and Actual_annual_return	Replacing by Zero
11	emp_title & last_payment_d	Removed these two columns

Data Leakage and Correlation variables:

Variables removed to avoid data leakage Count of variables removed = **27**	funded_amnt_inv, term, pymnt_plan, title, zip_code, addr_state, open_acc, initial_list_status, out_prncp, out_prncp_inv, total_pymnt, total_pymnt_inv, total_rec_prncp, total_rec_int, total_rec_late_fee, recoveries, collection_recovery_fee, last_pymnt_amnt, last_credit_pull_d, last_fico_range_high, last_fico_range_low, collections_12_mths_ex_med, policy_code, application_type, hardship_flag, debt_settlement_flag, no.ofinstallments

Variables removed due to more than 80% correlation Count of variables removed = **19**	open_acc, num_sats, num_op_rev_tl, num_rev_accts, total_bc_limit, num_bc_sats, num_actv_rev_tl, tot_hi_cred_lim, tot_cur_bal, total_rev_hi_lim, total_bal_ex_mort, loan_amnt, funded_amnt, avg_cur_bal, fico_range_low, mo_sin_old_rev_tl_op, revol_util, bc_util, num_tl_30dpd