Leticia Garcia, Danielle Strejc, Christopher Solis-Ocampo,

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

This is a continuation of the previous assignment where you developed decision tree based models to predict "Fully Paid" vs "Charged Off" loans in the Lending Club platform. In this second assignment, you will develop additional models – GBM, GLM (XGB) - to predict which loans are likely to be paid off and which will default. The previous assignment ended with the question on effective investment decisions based on your predictive models – we will examine this in more detail in the second assignment. We will also focus on parameter tuning, and reliable performance estimates through resampling and cross- validation.

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?

We experimented with different numbers of data sets to see the performance of the models. In the GBM Models the shrinkage and the number of trees were kept at a constant due the slowest of the process and the variables that were changing were the distribution and cross validation. We originally kept the same data set that we had in the previous assignment, others data sets that were considered were taking variables that were highly correlated with to see if it changed the accuracy, MSE value, and AUC value. We also in the end table we tested some data sets where the data was balanced to see if the performance had any affect.

Model Setup			
Model	**Data**	**Distribution**	**Number of Cross Validation**
Model 1	1[1]	Bernoulli	5
Model 2	1	Adaboost	5
Model 3	2[2]	Bernoulli	3
Model 4	2	Adaboost	5
Model 5	3[3]	Bernoulli	3
Model 6	3	Adaboost	5

[1] 1 represents data training set except for total payment, actual return, actual term, and annual return

[2] 2 represents the preprocessing data training

[3] 3 represents the data training where we take out all the highly correlated variables

Model 7	4[4]	Bernoulli	3
Model 8	4	Adaboost	5
Model 9	5[5]	Adaboost	5
Model 10	6 [6]	Adaboost	5

Evaluation Metrics				
Models	**Test Accuracy**	**MSE**	**Auc value**	**Overall CV Error**
Model 1	**0.8496**	**0.7772**	**0.68761**	**0.7961**
Model 2	**0.8494**	**0.6656**	**0.6868**	**0.6791**
Model 3	**0.8488**	**0.7772**	**0.6561**	**0.7946**
Model 4	**0.8412**	**0.6656**	**0.6723**	**0.6792**
Model 5	**0.8522**	**0.7950**	**0.68762**	**0.8063**
Model 6	**0.8509**	**0.6759**	**0.6757**	**0.6845**
Model 7	**0.8486**	**0.7902**	**0.6772**	**0.8067**
Model 8	**0.8258**	**0.6721**	**0.7105**	**0.6846**
Model 9	**0.5483**	**0.9253**	**0.9482**	**0.6826**
Model 10	**0.5498**	**0.9279**	**0.9495**	**0.94953**

Using test accuracy, MSE, RMSE, and AUC values we evaluated the performances of the models. Using the original data set from the previous assignment we see that using the bernoulli and cross validation of 3 we obtain performances of 0.8496 test accuracy and AUC value .68761 which was better compared to the second model which had distribution adaboost and cross validation of 5. The same can be said for our third data set model that had all the highly correlated variables taken. The models that had adaboost and high cross validation gave lower MSE value and also gave the lowest cv error as well.

[4] 4 represents data training that split 50/50
[5] represents df.rose (balancing the data)
[6] represents dfBOTH (balancing the data)

One thing to note is that I found that MSE value was higher than we were expecting. One explanation for this can be found in the confusion matrix in our results: we found a large portion of FN compared to TN that was a trend through for each model. Another thing that can be noted is that error went down when we used adaboost distribution and as well the number of cross validation increased as well. Since we had certain variables constant like the number of trees and shrinkage the error might have been better, but when dealing with a higher number of trees there were issues with being able to run with an acceptable amount of time. When dealing with shrinkage the model results didn't have much of an impact when the shrinkage was 0.001 and when putting in .1 for the shrinkage the model was worse.

The better model for came in terms of a trade off if you want a model with high test accuracy and AUC value the distribution you want the distribution to be Bernouilli and cross validation of 3. If you want the model with low cv error and MSE error the distribution will be with adaboost and increase of cross validation of 5. Model 6 and Model 8 had the best performance, Model 6 had the highest test accuracy and AUC value and Model 8 had the lowest cv error and MSE values. When doing the balancing data set for the models we see that test accuracy is poor at .5498 and MSE value is very high. The only thing the models provided was a good AUC value, but overall the models performed poorly.

One thing that might help the overall model would be increasing the cross validation as well increasing the number of trees. Other methods would be taking a look at the data and clearing up some of the variables to see if it makes a difference. The better model from this set has to be Model 6 and Model 8 based on the performance it gave back compared to the other models.

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

In the other document we see the plot of the loss function and also the corresponding gradient for each model. The loss function for the Adaboost:

Exponential loss: L(y, F) = exp $-y\,F\ -g\ x = y$ exp$(-y\,F(x))$.

In other distribution the loss function is

Loss function L(y, F(x)) = (y – F(x))2/2 (minimize squared error)).

The corresponding gradient is the overall cv error given in the model which can be seen in the table or in the corresponding in other documents.

(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.

To begin, we run very basic GLM models where we only adjust the dataframe used. This gives us an idea of the variables that may be significant as we look to build more advanced models. We look at AIC, AUC, Test error, and to a lesser extent precision and recall scores to identify the best performance between models.

Before we balance the data, we get a precision = 0.458, recall = 0.020, and F-statistic of 0.019 on the test set which are not very good, so we decide to try some balance codes. Models 4 and 6 both use a subset of variables that were left in after we took out those that were suspected of multicollinearity. We removed variables that were correlated with another x variable by more than 80%, and that df is referred to as "subset" or "sub".

Below is a table of the models we created in GLM:

Model Setup				Evaluation Metrics			
Model	**Data**	**PreProcess**	**Balance**	**AUC**	**Test Error**	**Precision & Recall**	**AIC**
Model 1	dfTrain	None	None	.737	.00003277	.865, .971	54211
Model 2 (M1BOTH	dfTrain	None	Both	.741	.00003277	.648, .141	54171
Model 3 (M1ROSE)	dfTrain	None	ROSE	.711	.00003277	.676, .141	89500
Model 4 (M2PP)	subset	BoxCox	None	.573	.00357	.850, 1	56059
Model 5 (M1PP)	dfTrain	Center	None	.737	.00003267	.930, .597	54211
Model 6 (m2)	subset	None	None	.690	.000065	.854, .989	56028

For AUC, our best models were the one where we balance the original data using "both" over and undersampling, and the original model and the pre-process original data model tied for second. When comparing AIC, the preprocess and "both" models are the lowest, which we prefer. All have low test error, and so we look to precision and recall scores, where the scores vary greatly. We choose to not look too much into the precision and recall scores. Overall we decide that our best models from GLM are Pre-Process, Both, and the original model without changes.

Our next set of models use cv.glmnet. We do lasso, ridge, and elasticnet = .5. We look at

AUC, mean cross validation error, and test accuracy for those models. We want cvm to be low, a high AUC and test accuracy.

We decide to look at our original df, the one where we process it with “center”, and the one where we balance the dataset to compare performances. We also do a pre-process on the balance set to see what that gives us.

Using GLMnet: note that the subset is a ridge-since it already removed correlated variables.

Model Setup				Evaluation Metrics		
Model	**Data**	**nFolds**	**type.measure**	**AUC**	**Test Accuracy**	**Mean cv error**
LASSO	DfTrnPP	3	auc	.730	84.83	.717
RIDGE	DfTrnPP	3	auc	.724	84.81	.697
Elastic	DfTrnPP	3	auc	.730	84.81	.717
Subset	sub Trn PP	3	auc	.692	85.09	.680
LASSO	dfTrn	5	auc	.729	84.8	.715
RIDGE	dfTrn	5	auc	.722	84.92	.695
Elastic	dfTrn	5	auc	.727	84.79	.716
Subset	sub trn PP	5	auc	.690	85.14	.686
LASSO	dfTrnBOTH-PP	5	auc	.731	.670	.719
RIDGE	dfTrnBOTH-PP	5	auc	.719	.663	.693
Elastic	dfTrnBOTH-PP	5	auc	.729	.669	.720
Subset	sub BOTH -PP	5	auc	.675	.638	.686
LASSO	dfTrn-PP	5	auc	.728	84.66	.717

RIDGE	dfTrn-PP	5	auc	.721	84.75	.696
Elastic	dfTrn-PP	5	auc	.728	84.67	.717
Subset	center PP	5	auc	.679	84.91	.684
LASSO	dfTrn-PP	5	mse	.723	84.73	.236
RIDGE	dfTrn-PP	5	mse	.717	84.75	.237
Elastic	dfTrn-PP	5	mse	.721	84.75	.236
Subset	boxcoxPP	5	mse	.678	84.92	.244

The subset models performed the worst in terms of AUC and about the same with testing accuracy compared with the other three that we looked at, across both fold numbers. We tried BoxCox and center, and there did not seem to be much difference. The highest AUC numbers all come from LASSO and ENET where there is some preprocessing applied to the models.

When comparing the LASSO model to the one where we removed the x variables where the correlation was at least 80%, we find that LASSO still performs better in all the comparisons above. Both Ridge and ElasticNet are not far behind. The test accuracies are all relatively close and high, which does have us wonder about overfit, especially since our AUC values are not extraordinary.

Lasso and elastic consistently have the best models, so we will be using them more later to evaluate models.

After first evaluating the original dataset at nfold=5 and type.measure= “mse” we decided to not use that same dataset for any other models, since it was a bit of an outlier with the cvm values. Our data needs to be pre-processed since there are many variables with wide ranges in values that vary. Once we do that, we see an improvement in the AUC scores, but we are not sure of why the testing errors and mean cv error are roughly the same. We suspect our models are overfit, even the subset one. We try to correct for this when we do the lasso and ridge models below.

We get the lambda value from the lambda.min function after doing the cross validation. We decided to try a few out here to help us determine which cv.glmnet model is the most suitable. All three of the df’s we look at here use auc as the type.measure, with 5 folds. None of these AUC’s are an improvement from the original GLM model, nor the ones in the previous tables. This tells us that perhaps our model is not tuned well enough, or maybe there is not much to be improved upon without more complex analysis.

Model	Data	nFolds	lambda	AUC	Test Accuracy	MSE
LASSO	dfTrn	5	.01143	.702	.849	.240
Elastic	dfTrn	5	.01143	.718	.849	.236
LASSO	DFPP	5	.0166	.688	.849	.242
Elastic	dfPP	5	.0166	.709	.849	.238
LASSO	sub PP	5	0.0079	.716	.849	.237
Elastic	sub PP	5	0.0079	.722	.849	.235

We tried to run LASSO and Ridge on the models that were balanced, but the evaluation indicated that these would not improve the models already in the table. The same was true for when we tried to Pre-Processed models used with the cross validated models. We include the latter at the end of the table to illustrate. The MSE is low compared to some of the models, but we are still cautious to call these models better than those before it.

We use cross validation to determine the best lambda to use for our models, based on the AUC, and MSE. We tried looking at “Class”, but it did not improve the models enough to include. It applies to logistic regression only, and looks at the misclassification error. Once we have those lambda values, we compare the models across nfold values of 3 and 5.

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

Anytime we apply regularization, we are saying the loss L(w) = sse+sum(w_i^2) , soif our ranges for the x vars are different our w’s will be different- so the x vals should be scaled to a common range.

We use ridge, elastic, and lasso, which have values of alpha = 0, .5, and 1, respectively. We want to find parameters that minimize the cost function. Commas indicate a subscript. This function is:

SIGMA (y,i – yhat,i)^2 = SIGMA(y,i – sigma(w,j * x,ij)^2 + lambda * SIGMA (w,j^2)

Where ‘w’ are what we want to optimize, and minimize the cost function. Lambda regularizes the coefficients and makes the models less prone to overfit. Lasso equation is the same, you just take the absolute value of |wj| instead of just (wj^2). Both help to lower the issue of multicollinearity in models.

For the link- we have a binary DV (logit), and we are using LS, so the link is that we are using two different types of models to analyze this specific loan status problem. Link refers to how the x variables explain the DV.

(c) Compare performance of models with that of random forests (which you did in your last assignment).

The linear (GLM) models were compared with a random forest model to establish what model better predicts the test data. The predicted variables correspond to loan status and the actual returns. The random forest model was applied toa 70/30 training/test data split. For the loan status prediction, the top three variables of importance include the loan amount, the funded amount, and the total amount of funding committed by investors at that point in time. Overall, this model underpredicted the defaulted (Charged off) loans while overpredicted the fully paid loans. The model had a mean squared error of 0.16. Moreover, the receiver operating characteristic curve area under the curve was 0.73. In the case of the annual return predictions, the top predicting variables were loan amount, funded amount, and the total amount of funding committed by investors at that point in time. The actual and predicted return scatterplot in the training and test mean squared errors were 9.21 and 20.06 respectively. Both scatter plots exhibited positive correlations between predicted and actual return. The mean squared errors in the training and test data are 9.21 and 20.96 respectively. Finally a positive correlation between risk score and annual return was predicted by the model.

Compared with the other models, the random forest model has similar or better performance. For the GBM model the AUC performance was 0.71 (Model 8). For models using a pre-processing step, as the GLMnet LASSO, the AUC was 0.73. We concluded that the random forest model is comparable to the other models in predictive power while having more user-independent capacity.

(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?

For the GLM model without the 80% most correlated x variables, the most important variables were: DTI, total accounts, total current balance, accounts open in the past 24 months, number of bankcard accounts, total bankcard limit.

When we include all the variables in the GLM, home ownership, verification status, installment, mortgage account, some of the higher grades & subgrades, in addition to the ones mentioned above for the model without the 80% most correlated x variables. Seeing this, it makes sense why the most important factors are the same across models-they predict the loan status with the highest degree of accuracy.

Looking at the graphs provided on the other document, for the GBM models we see that from the first,second and last dataset there's high importance on interest rate compared to the other variables. The second and third highest influence on the model was sub grade and total received interest. Installment and grade were the following influence variables in the model.

The other data sets used in the GBM models specifically looking at the data had the high correlated variables taken out and 50/50 split with different variables being influenced. We see that subgrade had the most impact on the models as well average current balance was the second highest variable influence. Then we see that the account opened in the last 24 months had an impact on the model and then from there each model had different variables that influenced it.

Looking at the balance data set models we see that both had interest rate and subgrade for their highest variables influence. Then differ in other variables afterward in the graph.

Overall in all the GBM Models the first two variables had the highest influence in the model. The variables importance didn't differ much when changing the cross validation and changing the distribution of the model when overlooking all the models.

When looking at both GLM and GBM they both have similar variables between the two which are accounts open in the 24 months past ,installment, higher grades, and subgrades. Then both differ in other variables influencing the model after that. Since both GLM and GBM have different variables that may affect the performance in terms of test accuracy, AUC value, or test error.

(e) In developing models above, do you find larger training samples to give better models ?

We decided to try our best models from part (a) with a 50/50 split between test and training.

Model Setup				Evaluation Metrics			
Model	**Data**	**PreProcess**	**Balance**	**AUC**	**Test Error**	**Precision & Recall**	**AIC**
Model 1 m1h	dfTrain	None	None	.688	.0000197	.853, .989	40143
Model 2 (M1BOTH hh)	dfTrain	None	Both	.685	.000059	.723, .355	64481
Model 3 (ssm1h)	subset	BoxCox	None	.686	.0000197	.855, .987	40184
Model4 m1PPh	dfTrain	Center	None	.688	.0000197	~	40143

Compared to the preliminary models, these perform better when it comes to AIC, and in some cases also in precision and recall. However, the AUC is not as high. The model4 had issues running, so we left that row blank- but based on the other evaluation metrics it likely performed similar to the model1.

For the 50/50 split: We consider the original split to see clearly if a larger training set improves performance. We put in the Pre-Processed model with the hopes of actually getting a somewhat reliable model. It appears that there is not improvement in AUC for any of the models, and the cvm is also high. The test accuracy raises questions since the AUC is low and cvm is high, but we were not able to fully interpret why the scores look the way they do.

Model Setup				Evaluation Metrics		
Model	**Data**	**nFolds**	**type.measure**	**AUC**	**Test Accuracy**	**Mean cv error**
LASSO	dfTrnPP	5	auc	.689	.849	.677
RIDGE	dfTrnPP	5	auc	.688	.850	.679
Elastic	dfTrnPP	5	auc	.689	.849	.678
Subset	subTrnPP	5	auc	.685	.840	.683
LASSO	dfTrn	5	auc	.689	.849	.677
RIDGE	dfTrn	5	auc	.690	.850	.680
Elastic	dfTrn	5	auc	.690	.849	.678
Subset	subTrn	5	auc	.685	.848	.683

Model	Data	nFolds	lambdas	AUC	Test Accuracy	% deviance
LASSO	dfTrn-PP	5	.0077	.691	.849	6.1
Elastic	dfTrn-PP	5	.0077	.691	.849	6.27
LASSO	Trn- PP	5	.00064	.693	.849	6.5
Elastic	Trn- PP	5	.00064	.693	.849	6.48

We only look at these since they were the best ones from earlier analysis. We pick two lambda.min values and see if they help create models that are accurate/ significance. Here, with a low deviance % and low AUC, the test accuracy is still surprisingly high. So for GLM overall, there was not a large difference, but I would say the larger training set helped in some areas, but may have sacrificed in others, like AUC.

For specifically the GBM models we found that when the data was split 50/50 and when the highly correlated variables were taken out had the best performances. Having a larger training data frame made the performance worse. We see that using the original data set with 70/30 there was higher MSE value compared to other models presented in the table and also had a low AUC value compared to other models. Overall both GLM and GBM models we found that 50/50 split worked best.

Do you find balancing the training data examples across classes to give better models?

We did balance in the form of "BOTH" and "ROSE". We found that ROSE did not help the models at all, however there were some improvements with BOTH. BOTH improved both the AUC and the AIC when compared to the control model with GLM.

When we did GLMNET, we found a slight improvement when incorporating the BOTH function. ROSE made the models worse, so we did not pursue that. When combining BOTH with PP we were optimistic, but the numbers ended up being off for training accuracy.

When balancing the training data doing the GBM models the performance of the model was worse compared to the other models with different data sets. Even though we see high AUC value, the MSE and over CV error are very high in the table. Test accuracy was also low for these models. Using different parameters didn't change the performances of the models if anything when doing the distribution for bernoulli the MSE error was higher.

2. Develop models to identify loans which provide the best returns. Explain how you define returns? Does it include Lending Club's service costs?

We define actual return as (total payment - funded amount) / the actual term of the loan in days*30* .99. The 30 makes it so we are looking at monthly actual return, and the 1% Lending Club service cost is also reflected in the equation. Actual term of the loan is the difference in days between the issue date and the last payment date.

Develop glm, rf, gbm (xgb) 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?

We found that 5 folds was optimal from looking at the models we performed earlier on.Below are the train and test predict returns when we use the 50/50 split set

GBM(XGB)

When evaluating the performance model beforehand we find that accuracy of it is at .85 and AUC value at 0.7360.

So we’re looking at the test data performance to determine the best model. We used a variety of variables to see the outcome of the training model. Our best performance was determined by train error, train AUC, eval error and eval AUC. Looking at the table below we see that when the max depth was 6 and the eta was around 0.1 it had the best performance through all the performance we were looking at.

Test Data Performance									
Objective	Max Depth	Eta	colsample _bytree	sub sample	lambda	Train Error	Train AUC	Eval Error	Eval AUC
binary:logistic	4	0.01	-	-	-	0.1505	0.7189	0.1503	0.7081
binary:logistic (Cross Validation)	3	0.11	-	-	-	0.1490	0.7672	0.1502	0.7465
binary:logistic	4	1	-	-	-	0.1440	0.7855	0.1512	0.7407
binary:logistic	6	0.1	-	-	-	0.1371	0.8487	0.1456	0.7631
binary:logistic	6	0.1	-	-	0.5	0.1376	0.8523	0.1466	0.7625
binary:logistic	6	0.1	0.5	0.7	0.5	0.1489	0.7787	0.1500	0.7384
binary:logistic	6	.01	0.5	0.7	-	0.1504	0.7539	0.1502	0.7308

nfolds were kept at 5 for all models

Model Setup								Evaluations	
Model	Objective	Eta	Max Depth	Alpha	Subsample	Min Child Weight	Col sample by tree	Train RMSE	Test RMSE
1	reg:squarederror	0.1	6	-	-	-	-	37.79	45.31
2	reg:squarederror (Boosting)	0.05	4	-	-	-	-	44.09	-

3	reg:squarederror (Boosting)	0.01	-	-	0.7	-	-	48.09	-
4	"reg:linear	0.05	4	-	-	-	-	43.86	45.53
5	"reg:linear" (Boosting) gblinear	0.3	-	0.0001	1	-	-	47.10	45.25
6	reg:linear (Boosting) gbtree	0.05	5	0.0001	-	1	0.6	49.22	-
7	reg:squarederror	0.1	6	-	-	-	-	50.49	55.71
8	reg:squarederror (Boosting)	0.05	4	-	-	-	-	55.69	-
9	reg:squarederror (Boosting)	0.01	-	-	0.7	-	-	63.13	-
10	reg:linear	0.05	4	-	-	-	-	55.54	56.67
11	"reg:linear" (Boosting) gblinear	0.3	-	0.0001	1	-	-	59.16	59.49
12	reg:linear (Boosting) gbtree	0.05	5	0.0001	-	1	0.6	64.47	-

For the xgboosting models we used the PP data set and subset dataframe for the models since it gave us the best results from the first question. There were a total of six different models for each data set using linear and square error and trying out different parameters with that setting. We used the RMSE and variables influence to evaluate the model. We tried different ways of finding other performance for the model but these two are the only way we were able to evaluate the performances. Besides we found when training the model the accuracy was .85 and AUC value was 0.7360. The numbers of rounds were kept the same for all the models since there was an issue in running the model in an appropriate amount of time.

We found that the best result came from max depth of 4 and 6 and eta ranging from 0.01 and 0.05, they gave the lowest RMSE value compared to the other models. One thing to note that may be affecting the performance can be the dataset or changing the parameters may improve the performance but since the running time was long it would hard to evaluate the models.

The variables' influence can be seen in the other documents for each model. We see that for the installment,interest rate and total received interest were high influence in the first dataset we had compared any other variables. In the second data set that we used annual income and some of the grades were the highest influence in the graph.

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

We looked at the 70/30 train test split and 50/50, and saw how the 50/50 seemed to be closer in terms of better performance.

B, C, D grade loans according to the 50/50 and 70/30 splits are selected loans that we would choose based on the actual returns table. See appendix for the table. We find that it is not entirely clear how to interpret the actual returns performance, since we have to normalize the data frame in order to get it to work. But it is clear that a majority of the highest returning loans were in one of those categories.

We run models for lasso, ridge, and elasticnet and all have very similar results. There is still the question of testing accuracy being high, but we suspect our model is overfit even though the actual returns table looks accurate.

When it comes to the GBM models we see high differences between the testing and training actual return especially for the second question and overall the results for the return tables for this model are not that great. GBM Actual Return for the first question looks good for the split for 50/50 both testing and training are very similar which could be from the model being overfit and having to look at the data set again.

That being said when selecting a loan for investment one important question we have to ask is on how much accuracy or how much error we want in the model. In terms of loan status we found that models that 50/50 performed the best over all measurements and when came to the actual return we found that preprocessing and 50/50 had the best results in terms of performances and loans specified above would be B,C, and D.

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).

Model Setup (GBM)				
	Data	Distribution	Shrinkage	Number of Cross Validations
Model 2	2	Adaboost	.01	5
Model 3	3	Bernoulli	.01	3
Model 4	4	Adaboost	.01	5

Evaluation Metrics (GBM)				
	Test Accuracy	**MSE**	**Overall CV Error**	**AUC Value**
Model 2	0.7780	0.7962	0.8923	0.6891
Model 3	0.7823	0.8153	0.9029	0.6037
Model 4	0.7798	0.8158	0.9032	0.6089

For the GBM models with lower grades we ended up using only data sets that showed high performance which are preprocessing, 50/50 and highly correlated variables sets in the model shown above. We also went with the best performing parameters in the first question as well.

We found that MSE and overall CV error was very high in the performances in every model. The AUC value was also lower compared to the AUC value in the first models in question 1. The test accuracy is also lower compared to the models from the first question. The most influential variables in the models were the subgrade, average current balance, installment, and account open in the past 24 months which slightly different not much compared to the other models done before.

The best model above was Model 2 with preprocessing data set with distribution of adaboost and cross validation of 3. Actual return tables also gave very different results when comparing the testing and training average predicted returns. Number of defaults are also very different compared to the two. We can determine from the results that tables are not well compared to the second question results. In terms of GBM models the performances it gave were worse compared to the previous question.

All the below are PreProcessed, since that was found to be necessary for interpretation.

Model Setup (GLMNET)				**Evaluation Metrics**			
Model	**Data**	**nFolds**	**Type of measure**	**AUC**	**Test Accuracy %**	**mean cv error**	**MAE/ y pred**
LASSO	dfTrnPP	5	auc	.656	76.96	.669	.850
RIDGE	dfTrnPP	5	auc	.684	77.11	.644	.844
Elastic	dfTrnPP	5	auc	.686	76.98	.665	.850
Subset	subsetPP	5	auc	.602	77.83	.600	.879
LASSO	dfTrn	5	auc	.687	76.98	.669	.848
RIDGE	dfTrn	5	auc	.652	77.07	.647	.845
Elastic	dfTrn	5	auc	.687	76.98	.667	.849
Subset	subset	5	auc	.606	77.64	.601	.879

Even though the test accuracy is not as high as it was in the other models that we looked at, we worry about overfit with the scores of the not- PP data. We remove total payment and interest rate, but other variables, such as those removed when we do the subset, may still be having a highly significant impact on the loan status.

Please submit a pdf file with answers to the assignment questions, and supporting analyses. Also include a single Rmd file with your R code (note – code needs to be adequately commented and divided into sections in the Rmd file to help readability and ease understanding by others).