Machine Learning for Finance Coursework

Name: Thi Thao Van Ho
Student ID: 210039249

I. Introduction

LendingClub is the world's biggest P2P lending platform, which enables investors to lend money to borrowers for unsecured personal loans. However, Lending Club has the same risk of loan default as all lending organisations, which might have a substantial effect on both investors and borrowers.

This coursework provides analysis of random sample of loans made between 2007 and 2015 on the LendingClub platform. The objective is to create a machine learning model that can identify loans that are most likely to default based on data's payment information (train data, test data and variable description). The goal of this report is to assist the company in reducing the risk of losses from loan defaults while assisting investors in choosing which loans to participate in. We also intend to learn more about the factors that influence loan default, which could assist guide lending practices and regulations. Several significant factors that are connected to loan default were discovered via analysis of the dataset. Based on the analysis, recoveries, collection recovery fee, interest rate on the loan, loan grade, total received interest, inquiries last 6 months and revolving ultilisation are the most significant predictor of loan status. For instance, default rates are higher for loans with lower grades (grades E, D, and F) and higher interest rates (see the full code in appendix 10).

This report uses a range of approaches such as data cleaning and processing, data analysis, model selection and evaluation, and interpretation. Utilising these methods can help to develop and test machine learning models that can accurately recognise loan default and pinpoint the key contributing factors.

II. Methodologies

The first step is data cleaning and preprocessing, use machine learning methodology to clean data by handling missing values and converting loan_status to logical values, check categorical variables for both training and testing data. Then convert categorical variables to dummy variables to use some Matlab functions such as corr(), ridge() and lassolm(). After that, create predictor table for both the train and test table.

The second stage is to do covariance analysis to analyse the correlation between response variable (loan_status) and categorical variables. Move 'loan_status' column to the first column in 'new_trainTbl' then compute correlation coefficient between response variable and categorical variable using corr() function. The 'vnames' contains the variable names. After that, use 'imagesc' function to generate colormap of correlation coefficients at the 'set(gca)' to set the variable names as axis labels. The top 10 and bottom 10 correlated predictors are determined by descending the correlation coefficients. The sorted correlation coefficients are contained in the variable sortedCorr, and the sorted variable indices are contained in the variable idx (see the code in appendix preliminary analysis / covariance analysis).

The third step is to do model selection and evaluation. In this report, I use 7 different machine learning models below to recognise loan default and find out the most important factors that LendingClub should consider:

- Generalised Least Squares model (GLS) is used for regression analysis (please find formula in appendix 1). When the residuals are not independent and do not have a constant variance, the GLS model is used to estimate the parameters of a linear regression model. Contrary to OLS, GLS, on the other hand, provides more weight to data with smaller residual variance, which helps to reduce the influence of outliers on the regression line. One advantage of GLS model is that it is capable of properly handling data with heteroskedasticity, autocorrelation and serial correlation. OLS, in contrast, makes the assumption that the variance of the errors is constant for all observations, which can result in estimates that are inaccurate and inefficient when the actual variance structure is different. However, GLS models can use a lot of CPU power when working with huge datasets because they include matrix inversion and multiplication. Longer processing times may arise from this, and high-performance computing resources might be needed. Additionally, if the covariance structure is complicated, it can be difficult to describe the right model, which would add to the computational difficulty.

- Stepwise regression is the process of building a regression model iteratively and step-by-step, stepwise entails choosing the independent variables that will be included in the final model. It includes adding or eliminating variables and testing for statistical significance after each iteration. The main benefit of stepwise regression is automation and simplicity. It is simpler to understand the model because of the method's identification of a subset of predictors that are most important for the response variable. Stepwise regression can reduce the chance of multicollinearity, which can happen when there are significant correlations between predictors, by only using the most pertinent predictors. One limitation of stepwise is the potential risk of overfitting when using best subset model. That means there is a higher chance of finding a model that fits the training data well but does not generalise well to testing data when there are a lot of predictors to pick. Another limitation of using stepwise model is that stepwise is based on assumption that there is a linear relationship between predictors and response variables. It's possible that there are non-linear correlations between the variables in real-world circumstances, contradicting this assumption. Therefore, predictions might be wrong.
- Least Absolute Shrinkage and Selection Operator (LASSO) is a type of linear regression using shrinkage (please see LASSO formula in appendix 2). When data values shrink towards a middle value, such as the mean, this is called shrinkage. One advantage of using LASSO is the capacity to choose only a limited number of potential predictors from a large number of predictors (referred to as "sparsity"), which OLS and Ridge are lacking. In other words, Lasso can assist in determining which predictors have the most influence on the result and efficiently ignore others that have a minimal impact. To do this, the coefficients of predictors with little influence are penalised, thereby "shrinking" to zero. However, there is bias in the LASSO estimator. The bias is caused by the LASSO penalty component, which generates a bias towards zero in the estimates of the predictor variable coefficients. When running LASSO regression in Matlab, I face a problem with scaling. To address that issue, I set a threshold of 0.5 to the coefficient after having been estimated by LASSO, any predictor is bigger than or equal to 0,5 will be classified as 1, otherwise, classified as 0.
- Elastic Net is a regularisation model used to address some of the drawbacks of both LASSO and Ridge regression by balancing feature selection and shrinkage (see more in appendix 3). Alpha and lambda are two turning variables for the elastic net model. The penalty term in Elastic Net regularisation consists of both L1 and L2 penalties. The alpha parameter regulates how these two penalties are balanced. Elastic Net is identical to ridge regression when alpha is set to 0 (only L2 penalty is applied), and to LASSO regression when alpha is set to 1 (only L1 penalty is used). Elastic Net's key benefit is that it handles highly correlated predictors better than Lasso, which has a propensity to choose just one out of a set of highly linked predictors and neglect the rest. When there are far more predictors than observations, Elastic Net can also manage the scenario. Elastic Net can be computationally expensive, though, and cross-

validation may be necessary in order to choose the best tuning settings. It can also depend on how the predictors are scaled.

- Bootstrap aggregation (Bagging) is a common ensemble learning method used in machine learning. The process of bagging involves training many models on arbitrary portions of the training data and combining their predictions to arrive at a final forecast. By randomly sampling the training data with replacement, a procedure known as bootstrapping produces the random subsets. A variety of model predictions are introduced since each model is trained using a different subset of data. One pros of using bagging is that bagging can lower the variance of the model by pooling the predictions of numerous models, making the model less sensitive to minute changes in the training data. Especially if the model has a tendency to overfit the data or if the data contains noise or outliers, this can increase the model's accuracy and robustness. Bagging, however, has some restrictions and drawbacks. It may be computationally expensive, particularly if many models are trained or if a sizable dataset is used. Furthermore, bagging doesn't necessarily make a model perform better; in fact, it may make a model perform worse if the models are poorly built or too similar to one another. Bagging also doesn't deal with the problem of a strong bias in the model, which may call for other techniques like boosting or model selection.
- Boosting is a machine learning model that combines several weak models to produce a stronger model (see more in appendix 4). The fundamental concept underlying boosting is to combine weak models that have been iteratively trained on various subsets of data to get a final model that is more accurate and reliable. The main benefit of boosting is that it frequently outperforms other machine learning methods in terms of accuracy, especially for challenging problems. Additionally, because it may be used with a variety of algorithms, such as decision trees, neural networks, and linear models, boosting is more flexible than other strategies. But there are also drawbacks to boosting. In comparison to other strategies, boosting may be more prone to overfitting, particularly if the weak models are overly complex or there are too many iterations. As it entails training numerous models on various subsets of the data, boosting can also be computationally expensive. Last but not least, boosting might be sensitive to outliers and noisy data because these can have an excessively negative impact on the final model.
- Random forest is a sort of ensemble learning technique mixing several decision trees to produce a forecast (appendix 5). Each decision tree is built from a random fraction of training data, which helps to decrease overfitting and boost model generalization. An advantage of random forest is that it can handle huge datasets with high-dimensional feature spaces, since it effectively handles a large number of input variables without the need for feature selection or dimensionality reduction, random forest is able to handle huge datasets with high-dimensional feature spaces. Besides, random forest help to reduce overfitting better than individual decision trees. Since each tree is trained on a separate subset of the data, less correlation exists between the trees, which helps avoid overfitting. In general, random forests can capture non-linear correlations between features and response variables, but their performance might not be as excellent as models made expressly for non-linear interactions, like neural networks, support vector machines, or gradient boosting models. Random forests may also not function effectively if the non-linear relationship is complicated or involves interactions between a number of features.

III. Main findings

Please find the table below to see the output of MSE of 7 different model constructed in Matlab.

Model	MSE using Training sample	MSE using Testing sample
GLS	0.0474	0.0484
Stepwise	0.0389	0.0399
Lasso	0.0378	0.0388
Elastic Net	0.0392	0.0403
Bagging	0.0363	0.0376
Boosting	0.0363	0.0375
Random forest	0.0358	0.0372

Mean squared error (MSE), is a commonly used metric to assess how well regression models work. The target variable's average squared difference between actual and anticipated values is measured. A lower MSE signifies improved model performance in terms of correctly predicting the target variable. According to the MSE values table above, it appears that the random forest model has the lowest MSE on both the training and testing samples, suggesting that it may have the greatest predictive performance out of all the regression models. Bagging and boosting are also alternate models that have low MSE values.

In terms of the most important variables, it appears that recoveries, and collection recoveries and interest rate are the three variables that have the greatest impact on predicting the response variable. Other elements like the interest rate, the grades E and F, and the total amount of late fees appear to be very important in the model (see more in appendix 6, 7 and 9).

IV. Conclusion

Inconclusion, Random forest and boosting models have the lowest MSE and, as a result, performed the best in terms of loan default prediction, according to the results of the various models on the training and testing datasets. While GLS and Elastic Net (appendix 8) models performed considerably poorly, Bagging and Lasso models (appendix 7) also performed well.

In addition, it was discovered that the recoveries, collection recovery costs, total principal receive to date, interest rate, grades E and F, total late fees collected, grade D, enquiries in the previous six months, and revolving utilisation rate were the most important variables in predicting loan defaults. Therefore, LendingClub should keep in mind about these significant predictors to predict loan defaults.

V. Appendix

1. GLS use the form:

$$p(X) = \frac{e^{\beta_0+\beta_1 X}}{1+e^{\beta_0+\beta_1 X}}.$$

p(X) is restricted between 0 and 1 with any values of $\beta_0 and\ \beta_1$and X

$$\begin{aligned}\hat{\beta}_{LASSO}(\lambda) &= \arg\min_{\beta}\left[\frac{1}{2T}\sum_{t=1}^{T}(y_t-\beta' x_t)^2+\lambda\sum_{j=1}^{p}|\beta_j|\right]\\ &= \arg\min_{\beta}\left(\frac{1}{2T}\|y-X\beta\|_2^2+\lambda\|\beta\|_1\right).\end{aligned}$$

2. LASSO:
3. Elastic Net estimator:

$$\hat{\beta}_{EN} = \arg\min_{\beta}\left[\|y-X\beta\|_2^2+\alpha\lambda\|\beta\|_1+(1-\alpha)\lambda\|\beta\|_2^2\right],$$

where $0<\alpha<1$

4. Boosting:

$$y_t = f(x_t)+u_t = \sum_{m=1}^{M} f_m(x_t)+u_t,$$

where:

- ★ y_t: response variable
- ★ $x_t \in R^p$: vector of predictors / explanatory variables
- ★ u_t: random error
- ★ $f_m(x_t)$: basis function or weak learner

The goal is to estimate optimal function f^*:

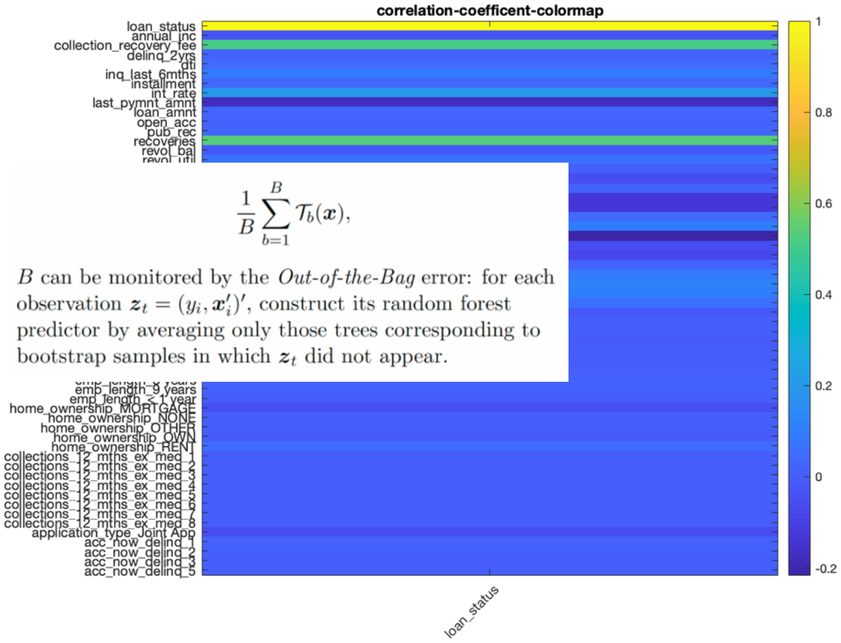
$$f^* = \arg\min_{f} \mathrm{E}\{L(y, f(X))\},$$

where:

- ▸ $y=(y_1,\cdots,y_T)'$
- ▸ $X=(x_1,\cdots,x_p)$ and $X_i=(x_{i1},\cdots,x_{iT})'$
- ▸ $L()$ is a loss function.

5. Random forest:

6. Correlation colormap:

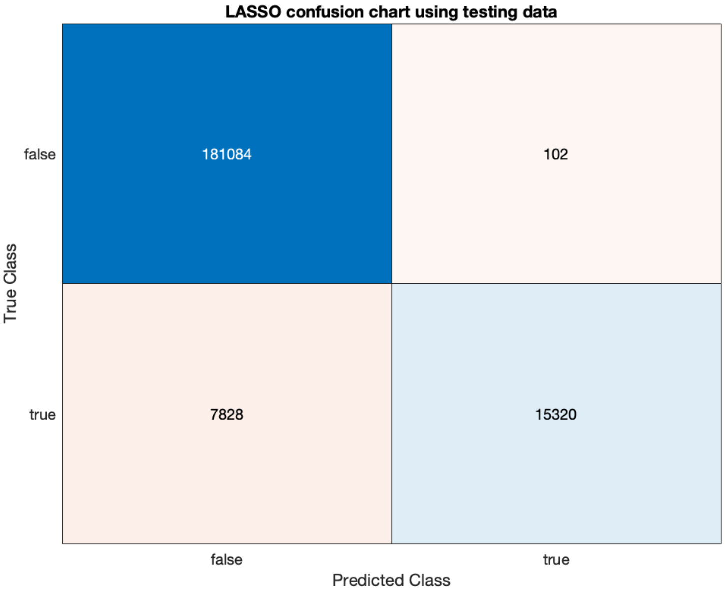


$$\frac{1}{B}\sum_{b=1}^{B} \mathcal{T}_b(\boldsymbol{x}),$$

B can be monitored by the *Out-of-the-Bag* error: for each observation $\boldsymbol{z}_t = (y_i, \boldsymbol{x}_i')'$, construct its random forest predictor by averaging only those trees corresponding to bootstrap samples in which $\boldsymbol{z}_t$ did not appear.

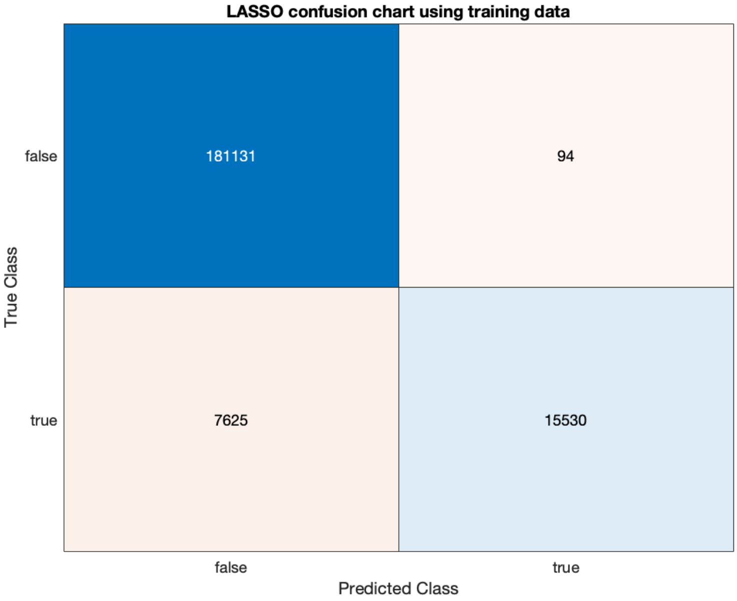
The colormap above illustrate the correlation coefficient of response variable ‘loan_status’ and predictors. As can be seen from the picture above, recoveries, collection recovery costs, interest rate are highly correlated with variable loan_status, meaning that those are 3 significant factors when considering loan defaults.

7. Lasso confusion chart using testing data:



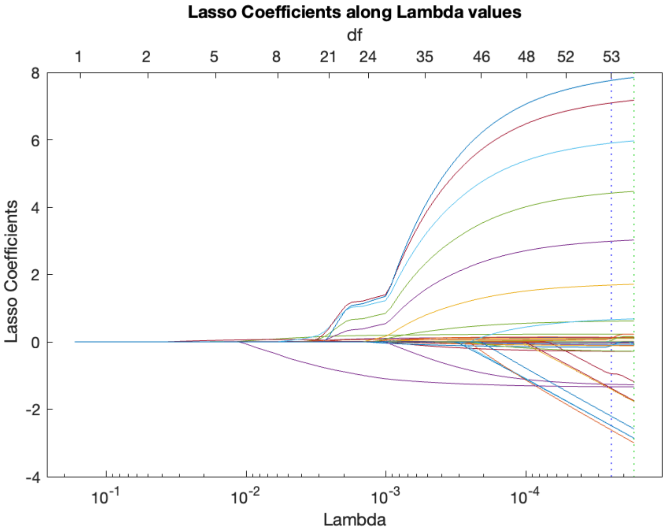
The model true positive (TP) rate: $\frac{181084}{181984+102} = 99.94\%$, showing that it’s effective at detecting positive instances.

The false discovery rate (FDR): $\frac{7828}{7828+15320} = 33.82\%$, suggesting that 33.82% of all model's positive prediction are incorrect. To put in another way, the model has a high rate of falsely classifying instances as positive.



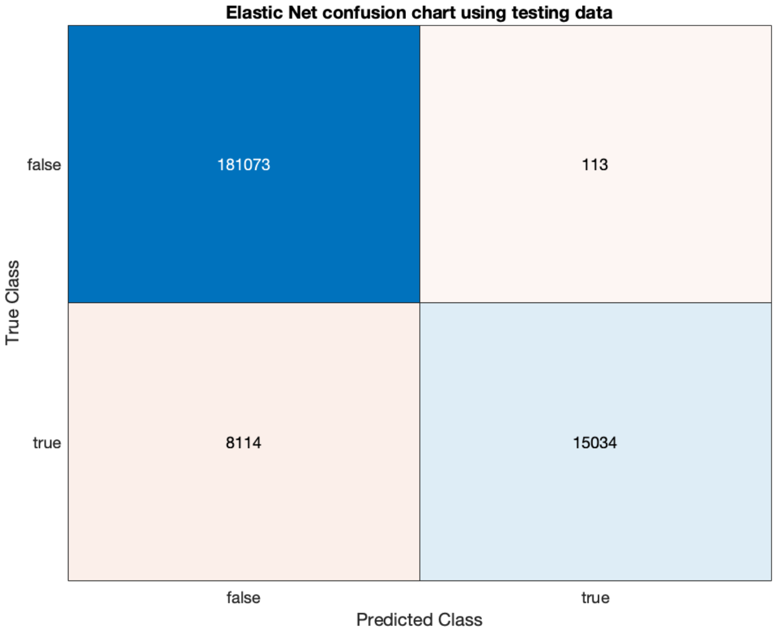
The model true positive (TP) rate: $\frac{181131}{181131+94} = 99.94\%$, showing that it's effective at detecting positive instances.

The false discovery rate (FDR): $\frac{7625}{7625+15530} = 32.93\%$, This indicates that roughly 33% of the model's positive predictions are false positives. This may indicate that the model is producing too many accurate predictions.



The graph above desribes the Lasso Coefficient along Lamda values. This helps to identify the most significant feature in the model. Starting with lamda λ = 0 and all coefficient is equal to zero. Then increasing λ, and we can see the Lasso coefficient change to non-zero values. When λ increases, the coefficient of the red line (recoveries) and blue line (collection recoveries fee) climbs in absolute value while the pink line decreases. This suggests that the 2 variables recoveries and collection recoveries fee are most important feature in the model.

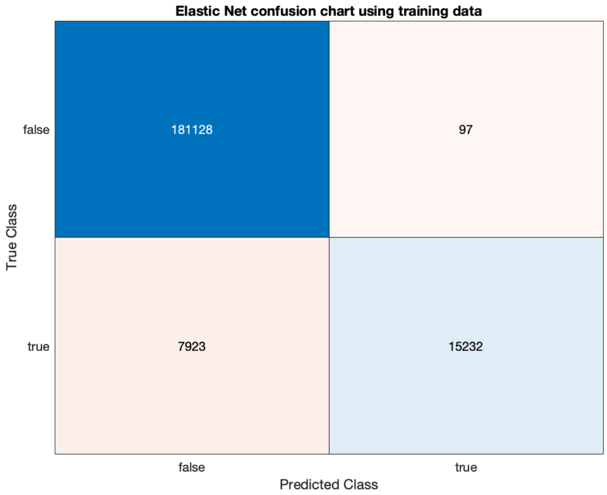
8. Elastic net:



The model true positive (TP) rate: $\frac{181073}{181073+113} = 99.94\%$, It signifies that the model successfully identify nearly all of the positive events in the testing data set.

The model true negative (TN) rate: $\frac{181073}{181073+113} = 64.95\%$ of the time, the model correctly recognise negative cases in testing sample

The false discovery rate (FDR): $\frac{8114}{8114+15034} = 35.05\%$, This indicates that 35.05% of the positive predictions provided by the model are actually false positives in testing sample.

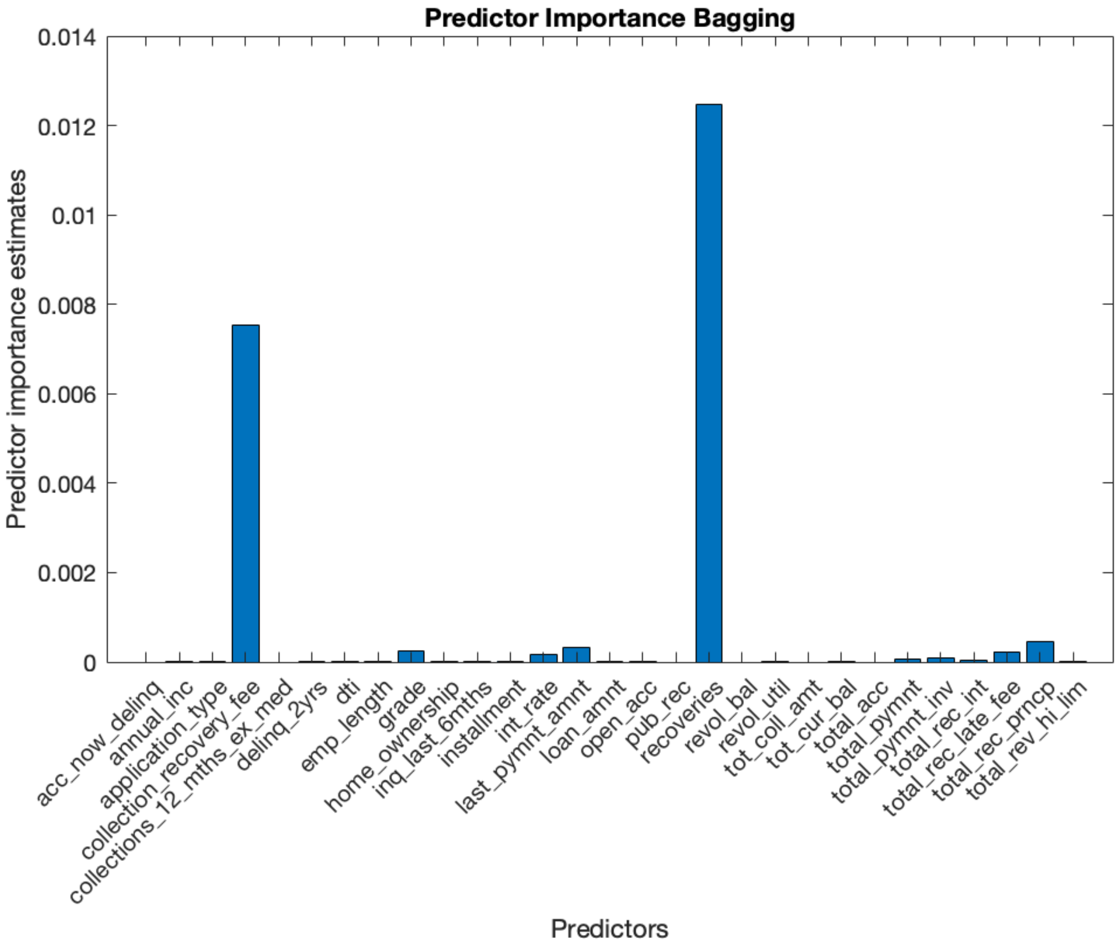


The model true positive (TP) rate: $\frac{181128}{181128+97} = 99.95\%$, It signifies that the model successfully classified 99.5% of the real positive cases as positive in training sample

The model true negative (TN) rate: $\frac{15232}{15232+7923} = 65.78\%$, it means that 65.78% of the negative cases in the testing data were accurately identified as negative by the model.

The false discovery rate (FDR): $\frac{7923}{7923+15232} = 34.22\%$, approximately 1/3 of the positive predictions made by the model are predicted to be false positive.

9. Predictor important bagging:



The bar chart illustrates the significant bagging predictors. As can be seen from the bar chart, collection recovery fee, recoveries and total principal receive to date are the most important predictors. Besides, grade, interest rate and late fee received to date are also highly correlated with loan status.

10. Coding script:

```
% ****************************
%  IF2211 coursework
%  author: Chuanping Sun
%  Bayes Business School
%  City University of London
% ****************************

clear all; clc; warning off;


%% readtable / treat missing values / specify import types
opts = detectImportOptions('trainData.csv');
opts.MissingRule = 'omitrow'; % missing values can cause problematic
estimation results
                              % some functions are robust with missing
data,
                              % while some are not

categoricalVariableNames = {'grade', 'emp_length', 'home_ownership',  ...
       'collections_12_mths_ex_med', 'application_type',
'acc_now_delinq'};
opts = setvartype(opts, categoricalVariableNames, 'categorical' );

opts = setvaropts(opts, 'emp_length', 'TreatAsMissing', {'n/a'});

% remove variables that have too many missing values
miss_var = {'mths_since_last_delinq'}; % this variable has 110000+ missing
data
selected = setdiff(opts.VariableNames(3:end), miss_var);
opts.SelectedVariableNames = selected;

% readtable with options
trainTbl = readtable('trainData.csv', opts);

%convert loan_status to logical values
key = 'Charged Off';
trainTbl.loan_status = strcmpi(trainTbl.loan_status, key);

summary(trainTbl)

% check variable levels to determine which variables are categorical
func = @(x) numel(unique(x));
varLevel = varfun(func,trainTbl);

% ------------------------------------------------------------------------
-
% similarly for testdata

testTbl = readtable('testData.csv', opts);

%convert loan_status to logical values
testTbl.loan_status = strcmpi(testTbl.loan_status, key);

summary(testTbl)
% ------------------------------------------------------------------------
-
% group predictor variables
```

```
predictorNames = setdiff(trainTbl.Properties.VariableNames,
'loan_status');
numVariableNames = setdiff(predictorNames, categoricalVariableNames); %
numerical predictors

%% convert categorical variables to dummy variables
% because some functions such as corr(), ridge() and lassoglm() cannot
deal with categorical predictors

% create dummy variable for 'grade'
grade_dum = dummyvar(trainTbl.grade);
grade_dum2 = dummyvar(testTbl.grade);
types = categories(trainTbl.grade);
dum_names = join([repmat({'grade'}, length(types), 1),types], '_' );
grade_trainTbl = array2table(grade_dum, 'VariableNames', dum_names);
grade_testTbl = array2table(grade_dum2, 'VariableNames', dum_names);

% create dummy variables for 'emp_length'
emp_length_dum = dummyvar(trainTbl.emp_length);
emp_length_dum2 = dummyvar(testTbl.emp_length);
types = categories(trainTbl.emp_length);
dum_names = join ([repmat({'emp_length'}, length(types), 1),types], '_' );
emp_length_trainTbl = array2table(emp_length_dum, 'VariableNames',
dum_names);
emp_length_testTbl = array2table(emp_length_dum2, 'VariableNames',
dum_names);

% create dummy variable for 'home_ownership'
home_ownership_dum = dummyvar(trainTbl.home_ownership);
home_ownership_dum2 = dummyvar(testTbl.home_ownership);
types = categories(trainTbl.home_ownership);
dum_names = join([repmat({'home_ownership'}, length(types), 1),types], '_'
);
home_ownership_trainTbl = array2table(home_ownership_dum, 'VariableNames',
dum_names);
home_ownership_testTbl = array2table(home_ownership_dum2, 'VariableNames',
dum_names);

% create dummy variable for 'collections_12_mths_ex_med'
collections_12_mths_ex_med_dum =
dummyvar(trainTbl.collections_12_mths_ex_med);
collections_12_mths_ex_med_dum2 =
dummyvar(testTbl.collections_12_mths_ex_med);
types = categories(trainTbl.collections_12_mths_ex_med);
dum_names = join([repmat({'collections_12_mths_ex_med'}, length(types),
1),types], '_' );
collections_12_mths_ex_med_trainTbl =
array2table(collections_12_mths_ex_med_dum, 'VariableNames', dum_names);
collections_12_mths_ex_med_testTbl =
array2table(collections_12_mths_ex_med_dum2, 'VariableNames', dum_names);

% create dummy variable for 'application_type'
application_type_dum = dummyvar(trainTbl.application_type);
application_type_dum2 = dummyvar(testTbl.application_type);
types = categories(trainTbl.application_type);
dum_names = join([repmat({'application_type'}, length(types), 1),types],
'_' );
application_type_trainTbl = array2table(application_type_dum,
'VariableNames', dum_names);
```

```
application_type_testTbl = array2table(application_type_dum2,
'VariableNames', dum_names);

% create dummy variable for 'acc_now_delinq'
acc_now_delinq_dum = dummyvar(trainTbl.acc_now_delinq);
acc_now_delinq_dum2 = dummyvar(testTbl.acc_now_delinq);
types = categories(trainTbl.acc_now_delinq);
dum_names = join([repmat({'acc_now_delinq'}, length(types), 1),types], '_'
);
acc_now_delinq_trainTbl = array2table(acc_now_delinq_dum, 'VariableNames',
dum_names);
acc_now_delinq_testTbl = array2table(acc_now_delinq_dum2, 'VariableNames',
dum_names);


% create predictor table. Note that first column of dummy variables are
% used as reference level

Predictor_trainTbl = [trainTbl(:,numVariableNames),
grade_trainTbl(:,2:end), ...
               emp_length_trainTbl(:,2:end),
home_ownership_trainTbl(:,2:end), ...
               collections_12_mths_ex_med_trainTbl(:,2:end), ...
               application_type_trainTbl(:,2:end),
acc_now_delinq_trainTbl(:,2:end) ];

Predictor_testTbl = [testTbl(:,numVariableNames), grade_testTbl(:,2:end),
...
               emp_length_testTbl(:,2:end),
home_ownership_testTbl(:,2:end), ...
               collections_12_mths_ex_med_testTbl(:,2:end), ...
               application_type_testTbl(:,2:end),
acc_now_delinq_testTbl(:,2:end) ];

new_trainTbl = [Predictor_trainTbl, trainTbl(:,{'loan_status'})];
new_testTbl = [Predictor_testTbl, testTbl(:,{'loan_status'})];


%% preliminary analysis / covariance analysis
% In this block, using the new_trainTbl to analyse the correlation between
% predicors and the response variable 'loan_status'. Use a heatmap to
% display the correlation structure. And find out the top 10 and bottom 10
% correlated predictors, call them top10 and bottom10. Include the graph
in
% your reporting.
%
##########################################################################
% -----------------------------------------------------------------------
-

%move loan_status to the first column
new_trainTbl = movevars(new_trainTbl,'loan_status','Before',1);
%compute correlation coefficient
corr_coeff = corr(new_trainTbl{:, :}, new_trainTbl{:,'loan_status'});
vnames = new_trainTbl.Properties.VariableNames(:,:);
%draw correlation coefficient colormap
figure
imagesc(corr_coeff)
colorbar;
```

```
set(gca,'XTick',1:length(vnames), 'XTickLabel',
vnames,'XTickLabelRotation',45,  ...
         'YTick', 1:length(vnames), 'YTickLabel', vnames,
'TickLabelInterpreter', 'none');
title('correlation-coefficent-colormap')

%sort correlation as descendence
[sortedCorr, idx] = sort(corr_coeff, 'descend');
%find out top 10 and bottom 10 correlated predictors
top10 = new_trainTbl.Properties.VariableNames(idx(1:10));
bottom10 = new_trainTbl.Properties.VariableNames(idx(end-9:end-1));

% -----------------------------------------------------------------------
-
%
########################################################################

%% GLS
% estimate a logistical model using the trainTbl (i.e., contains
categorical variables)
% hint: use the function fitglm().
%
########################################################################
% -----------------------------------------------------------------------
-

%Determine which variables are catagories

varLevel

%since we have been given the categorical variables we input that GLS

%Logistical model



mdl1 = fitglm(trainTbl, 'ResponseVar', 'loan_status', 'CategoricalVar',
categoricalVariableNames, 'Distribution', 'Binomial');

% -----------------------------------------------------------------------
-
%
########################################################################

% forecast using training data, and compute glsMSE_train
%
########################################################################
% -----------------------------------------------------------------------
-


y_train_predict = predict(mdl1, trainTbl);
% compute glsMSE_train (mean square error)
glsMSE_train = mean((trainTbl.loan_status - y_train_predict).^2);
glsMSE_train
% -----------------------------------------------------------------------
-
%
########################################################################
```

```
% forecast using test data, and compute glsMSE_test
%
#########################################################################
% ------------------------------------------------------------------------
-
yhat_test = predict(mdl1, testTbl);

% Set threshold value of 0.5
yhat_test_binary = yhat_test >= 0.5;

% Calculate the mean squared error (MSE) between predicted and actual
values
glsMSE_test = mean((testTbl.loan_status - yhat_test_binary).^2);
glsMSE_test

% ------------------------------------------------------------------------
-
%
#########################################################################



%% stepwise regression models
% Fit a stepwise regression model using only the top10 predictors you have
% found in the covariance analysis above.
swMdl = stepwiseglm(new_trainTbl(:, [top10; {'loan_status'}]),
'constant', 'upper', 'linear', ...
              'Distribution', 'binomial' );

new_trainTbl1 = new_trainTbl(:, [top10, {'loan_status'}]);
swMdl = stepwiseglm(new_trainTbl1, 'constant', 'Upper', 'linear',
'Distribution', 'binomial');

% forecast using training sample, and compute MSE for the training sample
swForecast_train = predict(swMdl, Predictor_trainTbl(:,top10));
swForecast_train = (swForecast_train >=0.5);
swMSE_train = mean((swForecast_train - trainTbl.loan_status).^2);
fprintf('MSE of the step-wise estimator using training sample is: %6.4f
\n', swMSE_train);

%true
Predictor_trainTbl1 = new_trainTbl(:, top10);
swForecast_train = predict(swMdl, Predictor_trainTbl1);
swForecast_train = (swForecast_train >=0.5);
swMSE_train = mean((swForecast_train - trainTbl.loan_status).^2);
fprintf('MSE of the step-wise estimator using training sample is: %6.4f
\n', swMSE_train);

% forecast using testing sample, and compute MSE for the testing sample,
% call it swMSE_test.
%
#########################################################################
% ------------------------------------------------------------------------
-

Predictor_test = new_testTbl(:, top10);
swForecast_test = predict(swMdl, Predictor_test);
swForecast_test = (swForecast_test >=0.5);
```

```
swMSE_test = mean((swForecast_test - testTbl.loan_status).^2);
fprintf('MSE of the step-wise estimator using testing sample is: %6.4f
\n', swMSE_test);


% -----------------------------------------------------------------------
-
%
##########################################################################




%% lasso
% model estimation using training data
% lasso cannot handle categorical predictors
[B_lasso,fitInfo] = lassoglm(Predictor_trainTbl{:,:},
trainTbl.loan_status, 'binomial', 'CV', 3);

B0 = fitInfo.Intercept(fitInfo.IndexMinDeviance);
coef = [B0; B_lasso(:, fitInfo.IndexMinDeviance)];

% predict loan_status using training sample
lassohat_train = glmval(coef, Predictor_trainTbl{:,:}, 'logit'); % specify
'logit' for binomial response
lassoForecast_train = (lassohat_train >=0.5);
lassoMSE_train = mean((lassoForecast_train -trainTbl.loan_status).^2);
fprintf('MSE of the lasso estimator using training sample is: %6.4f \n',
lassoMSE_train);
% output confusion chart for training sample
figure()
c_lasso_train = confusionchart(trainTbl.loan_status, lassoForecast_train);
title('LASSO confusion chart using training data')

% predict loan_status using testing sample. Hint: follow the method we
used
% for training sample
%
##########################################################################
% -----------------------------------------------------------------------
-

%model estimation using testing data
% lasso cannot handle categorical predictors
[B_lasso2,fitInfo2] = lassoglm(Predictor_testTbl{:,:},
testTbl.loan_status, 'binomial', 'CV', 3);
B0_test = fitInfo2.Intercept(fitInfo2.IndexMinDeviance);
coef_test = [B0_test; B_lasso2(:, fitInfo2.IndexMinDeviance)];
%predict loan_Status using test sample
lassohat_test = glmval(coef_test, Predictor_testTbl{:,:}, 'logit'); %
specify 'logit' for binomial response
lassoForecast_test = (lassohat_test >=0.5);
lassoMSE_test = mean((lassoForecast_test -testTbl.loan_status).^2);
fprintf('MSE of the lasso estimator using testing sample is: %6.4f \n',
lassoMSE_test);


% -----------------------------------------------------------------------
-
```

```
swMSE_test = mean((swForecast_test - testTbl.loan_status).^2);
fprintf('MSE of the step-wise estimator using testing sample is: %6.4f
\n', swMSE_test);


% ---------------------------------------------------------------------
-
%
##########################################################################




%% lasso
% model estimation using training data
% lasso cannot handle categorical predictors
[B_lasso,fitInfo] = lassoglm(Predictor_trainTbl{:,:},
trainTbl.loan_status, 'binomial', 'CV', 3);

B0 = fitInfo.Intercept(fitInfo.IndexMinDeviance);
coef = [B0; B_lasso(:, fitInfo.IndexMinDeviance)];

% predict loan_status using training sample
lassohat_train = glmval(coef, Predictor_trainTbl{:,:}, 'logit'); % specify
'logit' for binomial response
lassoForecast_train = (lassohat_train >=0.5);
lassoMSE_train = mean((lassoForecast_train -trainTbl.loan_status).^2);
fprintf('MSE of the lasso estimator using training sample is: %6.4f \n',
lassoMSE_train);
% output confusion chart for training sample
figure()
c_lasso_train = confusionchart(trainTbl.loan_status, lassoForecast_train);
title('LASSO confusion chart using training data')

% predict loan_status using testing sample. Hint: follow the method we
used
% for training sample
%
##########################################################################
% ---------------------------------------------------------------------
-

%model estimation using testing data
% lasso cannot handle categorical predictors
[B_lasso2,fitInfo2] = lassoglm(Predictor_testTbl{:,:},
testTbl.loan_status, 'binomial', 'CV', 3);
B0_test = fitInfo2.Intercept(fitInfo2.IndexMinDeviance);
coef_test = [B0_test; B_lasso2(:, fitInfo2.IndexMinDeviance)];
%predict loan_Status using test sample
lassohat_test = glmval(coef_test, Predictor_testTbl{:,:}, 'logit'); %
specify 'logit' for binomial response
lassoForecast_test = (lassohat_test >=0.5);
lassoMSE_test = mean((lassoForecast_test -testTbl.loan_status).^2);
fprintf('MSE of the lasso estimator using testing sample is: %6.4f \n',
lassoMSE_test);


% ---------------------------------------------------------------------
-
```

```
%
#########################################################################
% output confusion chart for testing sample
figure()
c_lasso_test = confusionchart(testTbl.loan_status, lassoForecast_test);
title('LASSO confusion chart using testing data')

% plot the lasso coefficient (B_lasso)along lambda values, and save the
figure
% as png file for reporting. You don't need to standarise the dataset.
figure()
%
#########################################################################
% ------------------------------------------------------------------------
-
% plot lasso coefficient using lamda
lassoPlot(B_lasso,fitInfo,'plottype','lambda','xscale','log');
%label x
xlabel('Lambda');
%label y
ylabel(' Lasso Coefficients');
title('Lasso Coefficients along Lambda values');
%save figure
saveas(gcf, 'lasso_coefficients.png');


% ------------------------------------------------------------------------
-
%
#########################################################################


%% Elastic Net
% Do the same analysis as you did with the lasso, but use the elastic
% net model with alpha parameter set to 0.5.
%
#########################################################################
% ------------------------------------------------------------------------
-

% model estimation using training sample
% elastic net cannot handle categorical predictors
[B_enet, fitInfo] = lassoglm(Predictor_trainTbl{:,:},
trainTbl.loan_status, 'binomial', 'CV', 3, 'Alpha', 0.5);
B0 = fitInfo.Intercept(fitInfo.IndexMinDeviance);
coef_enet = [B0; B_enet(:, fitInfo.IndexMinDeviance)];

%predict loan_status using train data
enetResponse_train = glmval(coef_enet, Predictor_trainTbl{:,:}, 'logit');
% specify 'logit' for binomial response
enetForecast_train = (enetResponse_train >=0.5);
%compute mean square error using train data
enetMSE_train = mean((enetForecast_train - trainTbl.loan_status).^2);
fprintf('Mean square error of the Elastic Net estimator using training
sample is: %6.4f \n', enetMSE_train);

% output confusion chart for training sample
figure()
```

```
chart_enet_train = confusionchart(trainTbl.loan_status,
enetForecast_train);
title('Elastic Net confusion chart using training data')

%predict loan_status using test data
enetResponse_test = glmval(coef_enet, Predictor_testTbl{:,:}, 'logit'); %
specify 'logit' for binomial response
enetForecast_test = (enetResponse_test >=0.5);
%compute mean square error using test data
enetMSE_test = mean((enetForecast_test - testTbl.loan_status).^2);
fprintf('Mean square error of the Elastic Net estimator using testing
sample is: %6.4f \n', enetMSE_test);

% output confusion chart for testing sample
figure()
chart_enet_test = confusionchart(testTbl.loan_status, enetForecast_test);
title('Elastic Net confusion chart using testing data')

% ---------------------------------------------------------------------------
-
%
########################################################################


%% bagging
% fit an ensemble model using bagging algo (using a tree as the weak
% learner. output the estimated model as 'bagMdl'.)
t = templateTree('MaxNumSplits', 5, 'PredictorSelection','interaction-
curvature','Reproducible', true); % using curvature algo when include
categorical predictors
%
########################################################################
% ---------------------------------------------------------------------------
-

numTree = 100; % number of trees in the ensemble
bagMdl = fitensemble(trainTbl(:,predictorNames), trainTbl.loan_status,
'Bag', numTree, t, 'Type', 'Classification');
%
########################################################################

% forecast using training sample
bagForecast_train = predict(bagMdl,trainTbl(:,predictorNames));
bagMSE_train = mean((bagForecast_train - trainTbl.loan_status).^2);
fprintf('MSE of the bagging ensemble estimator using training sample is:
%6.4f \n', bagMSE_train);

% forecast using testing sample, and save MSE as bagMSE_test
%
########################################################################
% ---------------------------------------------------------------------------
-
bagForecast_test = predict(bagMdl,testTbl(:,predictorNames));
bagMSE_test = mean((bagForecast_test - testTbl.loan_status).^2);

% ---------------------------------------------------------------------------
-
%
########################################################################
```

```
fprintf('MSE of the bagging ensemble estimator using testing sample is:
%6.4f \n', bagMSE_test);

% predictor importance estimation
% note 'PredictorSelection','interaction-curvature' ensures importance is
% not baised towards variables with many levels
imp = bagMdl.predictorImportance;
figure();
%
##########################################################################
% -------------------------------------------------------------------------
-

bar(imp);
title('Predictor Importance Bagging');
ylabel('Predictor importance estimates');
xlabel('Predictors');
h = gca
h.XTick = 1:length(bagMdl.PredictorNames);
h.XTickLabel = bagMdl.PredictorNames;
h.XTickLabelRotation = 45;  %this rotation makes the x variables easier to
read
h.TickLabelInterpreter = 'none';
% -------------------------------------------------------------------------
-
%
##########################################################################

% find top 10 important predictors
bagImportanceTbl = array2table(abs(imp)', 'RowNames',
bagMdl.PredictorNames, 'VariableNames', {'abs_coef'});
bagImportanceTbl = sortrows(bagImportanceTbl, 'abs_coef', 'descend');
bagTop10 = bagImportanceTbl.Properties.RowNames(1:10);
disp(bagTop10)



%% boosting
% fit a boosting ensemble model using a tree as the weak learner.
% It is very similar to the procedure above. Save MSE for the training and
% testing sample as boostMSE_train and boostMSE_test, respectively. Find
% the top 10 important predictors.
%
##########################################################################
% -------------------------------------------------------------------------
-

boostMdl = fitensemble(trainTbl, 'loan_status', 'AdaBoostM1', 100,
'Tree');

% Make predictions using the boosting ensemble model for the training and
testing samples
boostForecast_train = predict(boostMdl, trainTbl);
boostForecast_test = predict(boostMdl, testTbl);

% Compute MSE for the training and testing samples
boostMSE_train = mean((boostForecast_train - trainTbl.loan_status).^2);
boostMSE_test = mean((boostForecast_test - testTbl.loan_status).^2);
```

```
fprintf('MSE of the boosting ensemble estimator using training sample is:
%6.4f \n', boostMSE_train);
fprintf('MSE of the boosting ensemble estimator using testing sample is:
%6.4f \n', boostMSE_test);
% Find the top 10 important predictors
importance = boostMdl.predictorImportance;
[~,idx] = sort(importance,'descend');
boosttop10 = predictorNames(idx(1:10));
disp('Top 10 important predictors using boosting ensemble estimator:');
disp(boosttop10);

% ---------------------------------------------------------------------
-
%
#########################################################################



%% random forest
% fit a random forest with 50 trees. Randomly select 1/3 of total
% predictors to build each tree. Name this model rfMdl.
%
#########################################################################
% ---------------------------------------------------------------------
-

rng(1); % for reproducibility

ntrees = 50;
mtry = floor(numel(predictorNames)/3);

rfMdl = TreeBagger(ntrees, trainTbl, 'loan_status', 'Method',
'classification', ...
    'PredictorSelection', 'allsplits', 'OOBPrediction', 'on',
'OOBPredictorImportance', 'on', ...
    'MinLeafSize', 5, 'NumPrint', 10, 'MaxNumSplits', 100,
'NumPredictorsToSample', mtry);

% ---------------------------------------------------------------------
-
%
#########################################################################
% MSE using training sample
rfForecast_train = predict(rfMdl, trainTbl(:,predictorNames)); % output is
a cell array -> convert to numeric vector
rfForecast_train = cellfun(@str2double, rfForecast_train);
rfMSE_train = mean((rfForecast_train - trainTbl.loan_status).^2);
fprintf('MSE of the random forest estimator using training sample is:
%6.4f \n', rfMSE_train);

% MSE using testing sample
rfForecast_test = predict(rfMdl, testTbl(:,predictorNames));
rfForecast_test = cellfun(@str2double, rfForecast_test);
rfMSE_test = mean((rfForecast_test - testTbl.loan_status).^2);
fprintf('MSE of the random forest estimator using testing sample is: %6.4f
\n', rfMSE_test);

% Predictor importance estimation
```

```
%
#########################################################################
% ------------------------------------------------------------------------
-

rng(1); % for reproducibility

ntrees = 50;
mtry = floor(numel(predictorNames)/3);

rfMdl = TreeBagger(ntrees, trainTbl, 'loan_status', 'Method',
'classification', ...
    'PredictorSelection', 'allsplits', 'OOBPrediction', 'on',
'OOBPredictorImportance', 'on', ...
    'MinLeafSize', 5, 'NumPrint', 10, 'MaxNumSplits', 100,
'NumPredictorsToSample', mtry);

% Estimate predictor importance
imp = rfMdl.OOBPermutedPredictorDeltaError;
fprintf('Estimate predictor importance is: %6.4f \n', imp);

% ------------------------------------------------------------------------
-
%
#########################################################################

%% save data
save('assignment.mat', 'trainTbl','testTbl', 'top10',
'bottom10','glsMSE_train', 'glsMSE_test', ...
    'swMSE_train', 'swMSE_test', 'lassoMSE_train', 'lassoMSE_test',
'enetMSE_train', 'enetMSE_test',  ...
    'bagMSE_test', 'bagMSE_train', 'boostMSE_train', 'boostMSE_test',
'rfMSE_train', 'rfMSE_test');
    % change the .mat file name as firstname_surname_ID.mat.
```