Samet Öztürk – 2021702126

2021.01.19

IE 582 Final – Report

1. Introduction

Objective of the final is to construct a model that predicts whether an applicant of a credit will pay the loan or not. To work on that model, train data is provided.

1.1. Descriptive

Provided data is structured. The data includes 1715 applications which have 59 masked variables as well as the loan amount, customer age and the flag of target as “default”. Other than two variables, all columns are decimal or integer. I assumed masked numerical variables are not aimed to be categorical and they will be considered as numerical. Variables have different values from different ranges; hence the data might need scaling for some algorithms. Applied loan amounts are between 771 and 7994 units where 75% of the applications are below 3763 units. Data has no Null values to fill or handle.

2. Approach

My approach can be described under sections pre-process, dimensionality and scaling, class imbalance, cross validation, metric calculation, and model tuning and selection.

2.1. Pre-process

I casted character columns as factor to use in algorithms. These columns are “Var39”, “Var_53”, and “target”. “target” here is the column equivalent to “default” having different labels. I thought that using customer age as a numeric feature may not be logical since id did not sound right to say, “Credibility increases/decreases as people get older.”. Hence, I created 3 bins of ages as “old”, “young” and “mid” as factors to use age information in my study. Since there were no Null values, I did not follow a strategy to handle missing values.

2.2. Dimensionality & Scaling

Data had 61 variables for 1715 observations. To avoid negative effects of high dimensional data, I wanted to decrease the number of columns using PCA. Applying PCA on loan amount and other numerical variables, I see that the first 38 components can cover 99% of the variation of numeric features. So, I decided to use them in my models.

As I have mentioned, values in numerical columns are from various ranges that may cause some models to have bad performance. I used “preProcess” from caret package to scale and center the data. As a result, new numerical variables observed to be within similar limits.

Finally, I concatenated “target”, “loan_amount”, “age_bin” and categorical variables with numerical variables to get the final data that I will use on models.

2.3. Class Imbalance

1542 observations are of Class 0 where Class 1 has 172 observations. Clearly, the classes are imbalanced. To cope with that problem, I used over-sampling thanks to caret package, while training, setting scaling variable to “up”. I avoided using sampled data while evaluating results.

2.4. Cross Validation

I used 2 level cross validation. I used 10-fold cross validation to divide test and train, and calculated money loss metric for each fold. Then, I took the average of 10 different loss metric of one model (each model here is one algorithm and a set of parameters). While using train section of the folds, et each iteration, I used 2-fold cross validation on “caret.train“ function, as well.

2.5. Metric Calculation

We are asked to minimize total money loss due to misclassification. To measure the performance of each model, I created a custom function. The function calculates total money loss as (Total Loan Amount of Misclassification of 1’s(actual) as 0’s(predicted)) + (Loan Amount of Misclassification 0’s(actual) as 1’s(predicted)) * 0.15. Afterwards, to have a value independent of row counts, I divided the total loss with row count and obtained the “money_loss_per_row” values. I calculated the value for each fold of each model and took the average of it over models to get the performance of each model. I selected the models having minimum values of the metric.

2.6. Models

I have used the data obtained after PCA and scaling to train models. I applied 10-fold cross validation for calculating the metric for each model and used 2-fold cross validation while training each fold with caret package. To handle class imbalance, I used over-sampling. I tested CART, Lasso Regression, GBM, Random Forest and SVM. I did not use k-nn since it might be hard to use with high number of variables and it is hard to find a good distance metric. After a few tries, I found relatable parameter grids and calculated the metric for each model. Selection over the parameters and model selection is detailed in “Results” section.

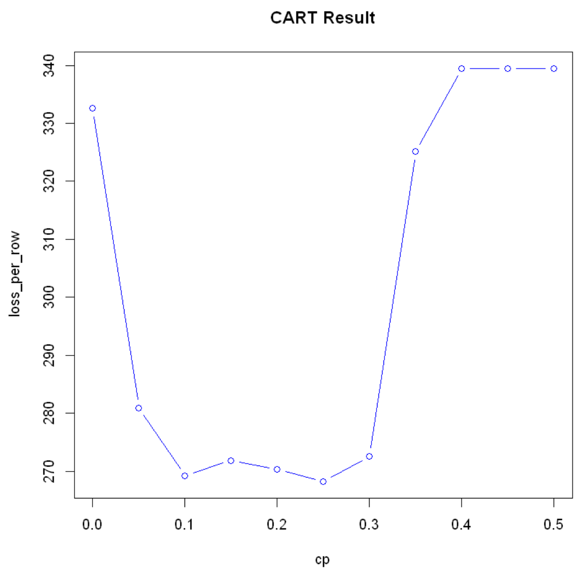
3. Results

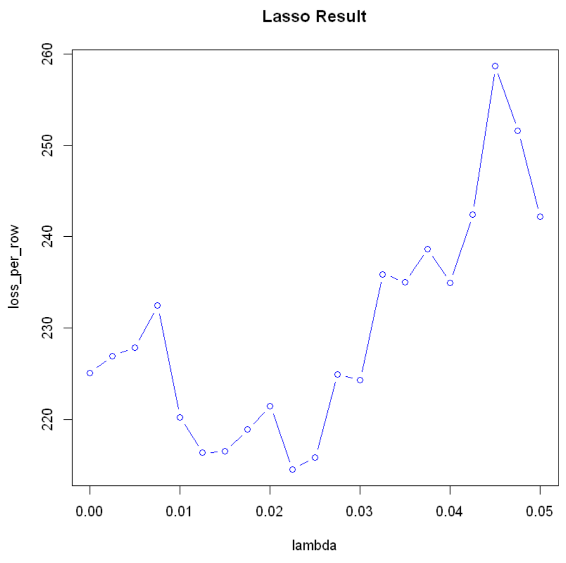
3.1. CART

I tried 11 “cp” parameters from 0 to 0.5 for Classification and Regression Tree models. The best performing model was the one with cp = 0.25, and it has average money loss per row value around 268. AS in the graph below, the models with cp between 0.1 and 0.3 are performing similar and relatively better than other models.

3.2. Lasso Regression

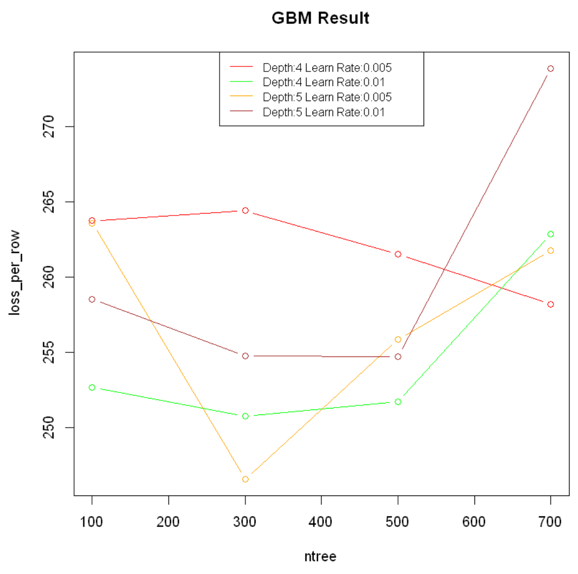
I used 21 different “lambda” values to train different Lasso Regression models. The best performing model was the one with lambda = 0.0225 with average money loss per row value around 214. Other loss values are also provided below. Relatively, Lasso Regression results were better than others.

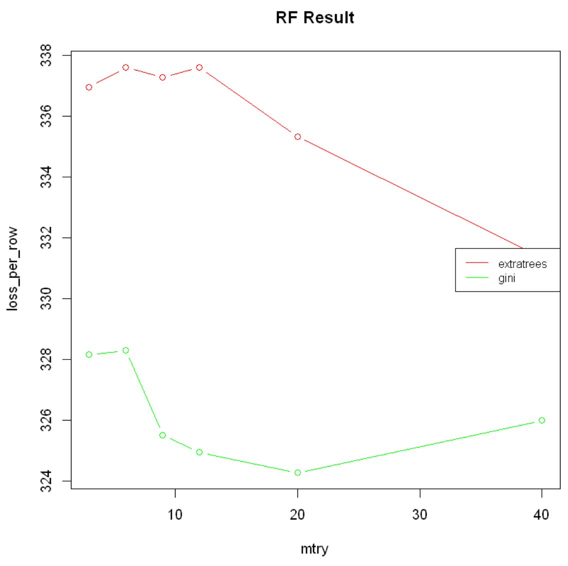




3.3. GBM

I trained GBM models with interaction depths 4 and 5, number of trees with 100, 300, 500, 700 and learning rates with 0.005 and 0.1. Among these 16 models, the best one was with 5 depth, 300 trees and 0.005 learning rate. Its loss value is around 246. Rest of the results are provided in the graph below.



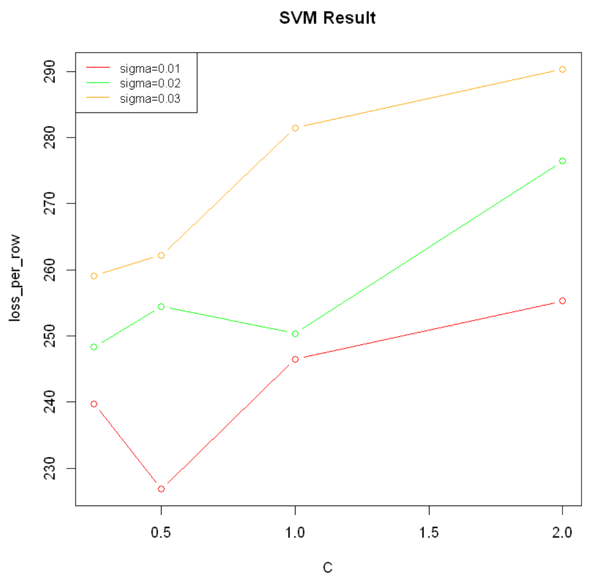


3.4. Random Forest

For random forest, I used split rules “gini” and “extratrees” and randomly selected variable numbers 3, 6, 9, 12, 20 and 40. From these 12, the best one was with “gini” rule and 20 variables. The loss is around 324. For random forest models, I realized that Specificity value is close to 0 where Sensitivity values are close to 1 for all random forest models. It is probably flagging most rows with the same value. Hence, random forest is performing worse, in general. All results are provided in the graph above.

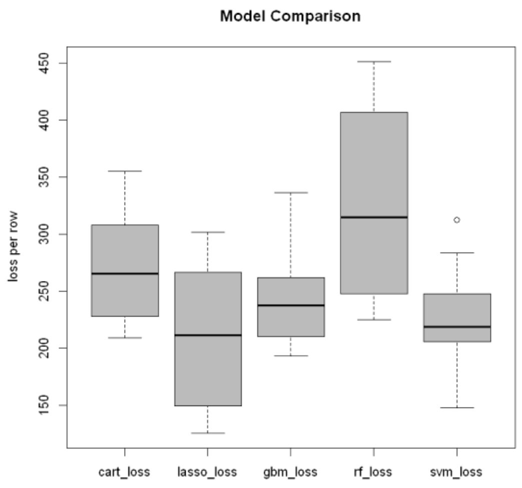
3.5. SVM

I used radial kernel for support vector machines where C values were 0.25, 0.5, 1 and 2 and sigma values were 0.01, 0.02 and 0.03. The best model among 12 had sigma 0.01 and C 0.5 with loss value around 227. All results are provided below.



3.6. Model Comparison & Selection

First, I selected the best parameter set for each algorithm. Then, I compared the best models of algorithms with each other. Combining loss values for 10-folds from best parameter sets, I constructed the boxplot below to compare robustness and money loss metrics of the models.



Here, the least average loss is from lasso regression, but I selected SVM regression with related parameters since, variation of the loss for SVM looks relatively less although SVM model has more average money loss.

4. Conclusions and Future Work

Using PCA to reduce dimensionality, scaling to have comparable variables, oversampling to balance classes, cross validation to avoid train bias parameter tuning and model comparison to select the best model, I found a model classifying applications with average 214-unit cost. One can work on the followings to find a better performing model:

- Dimensionality reduction: PCA may not be the best model to reduce dimensions as it requires linearity assumption. Some other method can be tries or only correlated columns can be selected.
- Scaling: I used the built-in scaling function where there can be a superior method for the specific problem here.
- Parameter Grids: Due to time limitations, I only used certain number of parameters, some more can also be tried to see if there is a better model.
- Sampling: I used oversampling and under sampling which resulted in similar results, yet I did not report. We can also try SMOTE and some other methods to see If they are better.
- Feature engineering: Age binning can be made better, or maybe leaving it as integer can also perform better. In addition, some numerical features in data can also be transformed into more meaningful factor values. To do that, we need to know column meaning, as well.