Loan Delinquency Milestone Report 2

Jorge Londono

Algorithm Selection

This project is a classification problem that does not implement a large amount of features. Therefore, because the data set is quite large, we will focus on testing out 2 ensemble classification models: Random Forest, and Gradient Boost. The advantages of a Random Forest classifier, is that by having each outcome of the decision trees be independent of each other, it helps balance out the chance of overfitting. On the other hand, Gradient Boosting strives to find the most important outputs from the split points and add them together. Both of these algorithms take quite a lot of computational power, therefore it's best to keep that in mind when implementing them.

Steps taken

To begin the machine learning process, the data frame being used has to have a small number of modifications done. For starters, to make computation easier, there needs to be a column added that will take the data from the 'result' column (either 'delinquent' or 'fully paid'), and transform it into a binary output (1 or 0 respectively) called 'result_binary'. This will be the target variable. Next, only the columns for the variables that will be used for the model will be selected from the data frame. Finally, using the pandas function 'get_dummies', the categorical variables will be encoded to be used in our model.

The features that will be used for the model are:

- Application Type
- Term
- DTI (deb to income)
- Fico Range Low
- Home Ownership
- Annual Income
- Loan Amount

Now that the all the modifications for the data frame are complete, the features and the target variable have to be separated in order to test out the models. To do this we take the columns for all the features and turn them into an array using the built-in Python method 'values()'. We also do the same thing to the target variable because the functions we are going to use require them to be in this format. We then split up our data into a training set, and a testing set using sklearn's train_test_split function to see how well the models will perform.

Next, we will begin testing out the models and comparing the outcomes. The first model that will be created will be a simple Random Forest classifier model with the parameters set to default. The reason why nothing is modified in this first test, is because this model's results will be used as a baseline to compare the second test to. The second test will be again a simple Random Forest classifier, but this time we will implement a PCA transformation. PCA helps to deal with high dimensionality problems. It would be a good idea to check if our model would improve with PCA. The next test will use whichever of these models perform best, and then implement a GridSearchCV to tune the hyper parameter of 'n_estimators'. The result of test 3 will give us the best results for the Random Forest Classifier. The final test will be a Gradient Boost classifier with GridSearchCV to optimize the learning_rate parameter, which will give us the best results for this type of algorithm. Finally, we will compare the results using the AUC score and choose the model that scores best.

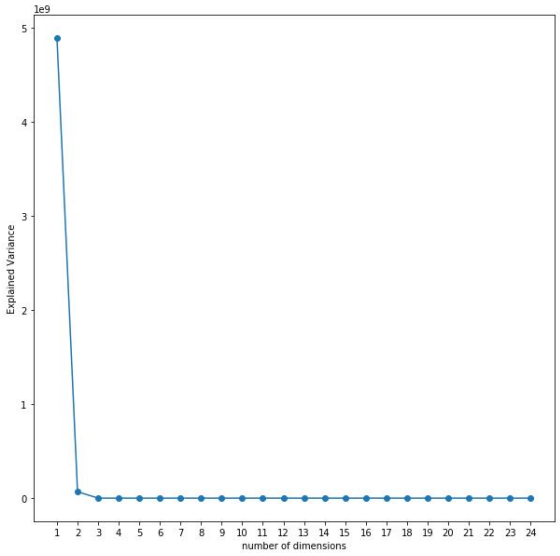
Results

Below are the results of all 4 tests:

Model	AUC Score
Test 1 Algorithm: Random Forest Classifier Parameters: Default Transformation: None	0.638

Test 2 Algorithm: Random Forest Classifier Parameters: Default Transformation: PCA	0.557
Test 3 Algorithm: Random Forest Classifier Parameters: n_estimators: 100 Transformation: None	0.640
Test 4 Algorithm: Gradient Boost Classifier Parameters: learning_rate: 0.5 Transformation: none	0.688

From these results we can look at some key points. To begin with, it seems like the PCA transformation was not helpful at all, as the AUC score was dramatically worse. The n_components variable was set to 2 as per the elbow plot below showed to be the optimal value, yet the results were still better pre-transformation.

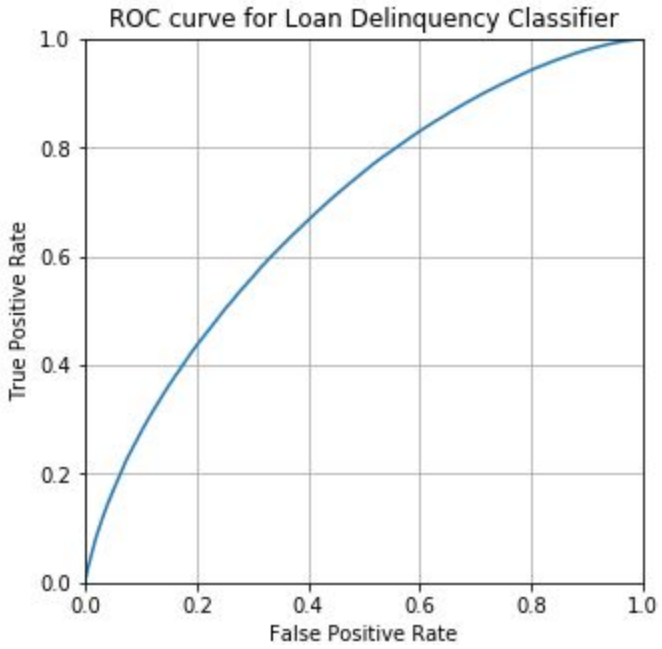


Additionally, for the Random Forest model, the default value for n_estimators (100) was ultimately the best value as per our cross validation test. On the other hand, when it came to cross validating the Gradient Boost model, the best parameter for learning_rate was 0.5 Finally, when compared to the best Random Forest model, the Gradient Boost model had better results and therefore will be the model that we will select. The confusion matrix that shows the

results is as follows:

```
[[426489   3670]
 [103566   4570]]
```

Below is a graph of the ROC curve, which shows the performance of this model:



Additionally, here are some more metrics such as the accuracy, the F1 score, the precision, and the recall:

	precision	recall	f1-score	support
0.0	0.80	0.99	0.89	430159
1.0	0.55	0.04	0.08	108136
accuracy			0.80	538295
macro avg	0.68	0.52	0.48	538295
weighted avg	0.75	0.80	0.73	538295

And finally, here is a table of the feature importances of the model:

	Feature	importance
5	application_type_Joint App	0.381939
3	fico_range_low	0.235659
2	dti	0.123480
0	annual_inc	0.072181
1	loan_amnt	0.053636

Conclusion

For this project, we took the various features of the loans data set and found the best combination of ensemble classification algorithm, parameters to create a model that can best classify if a certain loan will end up being delinquent. Using a Gradient Boost classifier with the the numerous features chosen, as well as the learning rate set to 0.5, we were able to get the best AUC score.

While the results show a good AUC score and accuracy score, this model can be improved in several ways. For starters, ensemble models are very computationally intensive. Because of this, there was a limit to the amount of features that I could include as well as the size of the training set I could use. There are two ways to remedy this problem. The simplest solution is by having more memory to create the more memory intensive models. Another solution would be to use a classification algorithm that is not as complex as the ensemble models. For example, instead of Random Forests or Gradient Boost, we can use logistic regression instead.

Recommendations

With this model, there are a few recommendations that can be made to lenders in order for them to identify delinquent loans before they happen. The first one is that join applications are a very big predictor of delinquent loans. It would be beneficial if they scrutinized these types of loans applications even more. Additionally, Fico scores are very important and credit scores and histories can highlight the possible risk of loan delinquency. Lastly, debt to income ratio is a very good predictor of possible delinquency. Therefore, the ability of a borrower to pay off their loan could be contingent on the burden of their already existing debt.

Further Study

With the findings in this project, we can further try to expand on it or ask other questions. Something that I found interesting while working on this project, is that my model is actually working within a system that is already actively trying to reduce delinquent loans. This data contains loans that have already been approved by the lender. However, this data does not include all the loans that were denied. Therefore, my model is building on the selection that is already occurring. By using application data, I can ask other types of questions. Such as, who is more likely to be denied. Then I could compare with the features from my model to gain new insights.

Another area of study that can stem from this project is looking at small business loans instead of personal loans. We can not only build the same type of model to see how it compares as this project, but we can take a look at what features are important for the success of small businesses, and what are predictors for business that may run into financial problems.