CS 155 Group Project 1 Report

Joon Young Park, Derek Ing, Daniel Li, Hernan Caceres

February 2022

1 Introduction

1. **Group members**:

   Hernan Caceres, Daniel Li, Derek Ing, Joon Young Park

2. **Kaggle team name**: Candice

3. **Ranking on private leaderboard**: 12^{th}

4. **AUC score on the private leaderboard**: 0.70243

5. **Colab Link**:

   Click here.

6. **Piazza Link**:

   Click here.

7. **Division of Labor**:

   Note that every members has contributed to data parsing and encoding process. Also, all members were present and actively participated in group discussions for brainstorming and improving model performances. Furthermore, every members worked on performing GridSearch for hyper-tuning and improving accuracy of a model.

   (a) Derek Ing: Focused on designing / hyper-tuning Adaboost models.

   (b) Daniel Li: Worked on target encoding States column and designing / hyper-tuning XGBoost models.

   (c) Joon Young Park: Focused on desgining /hyper-tuning logistic regression models. Also worked on trying out various ensemble methods.

   (d) Hernan Caceres: Preprocessing / Focused on designing / hyper-tuning Random Forest models.

2 Overview [15 points]

2.1 Models and techniques attempted

Throughout this project, we have mainly tried 4 different models: Logistic Regression, XGBoost, Adaboost and Random Forest. For each models, we have utilized Scikit learn GridsearchCV module for hyperparemeter optimization. With regards to data processing, we have used various encoding techniques, including one-hot encoding and target encoding, to turn the categorical/ordinal features into numerical features. Here, we have also incorporated another boosting model that hasn't been taught (or addressed) in the previous lectures - XGBoost. Throughout the research, we figured that this model is also an effective boosting model, so we have implemented this model out of ordinary.

2.2 Work timeline

First, after data processing, we used logistic model to see how well the model accurately predicts on the data. Once getting an accuracy of around 0.63, we decided to branch out from logistic regression and try out other aforementioned models. However, for Friday and Saturday, we encountered some data processing problem in which we spent some time improving upon our data processing (which we will elaborate further below). Noticing that the model's accuracy got significantly better after this, we have spent Sunday and Monday running grid search and trying out different parameters for hyper-tuning the model. By the end of Monday night, we made a decent process in improving the accuracy for each models. However, on Tuesday, we have concluded to stick with the XGBoost model (since it had the highest accuracy out of all the models used) so we focused on further improving upon data processing and running more extensive grid searches for Adaboost model in hopes of optimizing the model's parameters and achieving the highest accuracy possible. By the end of Tuesday night, we were able to achieve the highest accuracy out of all the attempts.

3 Approach [20 points]

3.1 Data exploration, processing and manipulation

We have incorporated several data cleaning and feature engineering skills to parse our data. Primarily using Python, we imported libraries such as NumPy, Pandas, Feature Engine and Sklearn. For preprocessing, to start we noticed initially the number of unique IDs is the same size as the number of rows in our dataframe, and for obvious reasons was initially deleted. Upon examining correlation of our numerical features, it was observed that installment had a correlation above our set threshold of 0.9. Examining unique categories for each feature, we checked if any of the categories hold the majority of the values, which we deemed to be also 0.9, for there isn't enough variability in this case to include the feature. As a result, pub_rec_bank and application_type were dropped. Since grade and subgrade are ordinal features, these features were ordinal encoded. Features were one hot encoded since they were nominal features. For missing data cells, we inputted the missing values with -1 for after all features were converted to numerics.

3.2 Details of models and techniques

The first model that we have tried is logistics regression model, since it is an efficient model in accurately predicting categorical variables. We concluded that this would be a good basis to start out and get a general sense on what features to add/remove by determining the relationship between variables and the "loan_status" attribute so that we could improve upon using other models. After checking that the accuracy of logistic models weren't good enough, we decided to try out random forest as it is an effective model for non-linear models. In addition, we concluded that 200,000 (or at least 100,000 training datasets) datasets are sufficient training dataset for random forest model to be used, and we know that this model is generally more accurate than linear model, including logistic model. After trying out this model, we wanted to see if boosting model would work better, as we have learned from the lecture that boosting generally reduces bias using low-variance models. Out of this, we decided to use adaboost and XGBoost since we have also learned that adaboost is more suitable in classification than gradient boost and XGBoost is just a well known boosting that generally works well.

4 Model Selection [20 points]

4.1 Scoring

For our XGBoost and logistic regression models, we used logistic loss as our learning objective. Minimizing logistic loss is the same as maximizing log likelihood. Thus, we can maximize the likelihood that the model observes the given data and the data points are classified correctly. We used Gini impurity to train each classifier in AdaBoost and Random Forest models. When we trained the AdaBoost and Random Forest models, we used both Gini and Entropy impurity to train the trees. The models trained with Gini impurity gave better validation AUC scores. We scored all of our models using AUC, and the model that had the best AUC score was XGBoost. The ROC curve tells us the model's true positive rates versus false positive rates. The area under the ROC curve tells us how good the model is at distinguishing between positive and negative classes.

4.2 Validation and test

We found that cross validation was very slow for some models, so we used sklearn's train test split to tune our models. With sklearn's train test split, we used a test size of 0.2 of the whole training set. After tuning all the hyperparameters, we tested each model with k-fold cross validation with five folds and averaged the AUC scores for each fold. From the testing, XGBoost had the highest average validation AUC score of 0.695. The model with the second highest average validation AUC score was AdaBoost, with 0.683. The random forest and logistic regression models ended up with average validation scores 0.65. The results of testing told us that XGBoost was the best at distinguishing between loans that were fully paid or charged off.

5 Conclusion [20 points]

5.1 Insights

Among all of the features in the data, we were able to determine the most influential features on determining whether or not the loan was paid off by running a Logistic Regression on the dataset with L1 normalization. After the regression finished running, we returned the final weights and established which ones were positively influential and negatively influential. The most important features in no particular order are: subgrade, term, open_acc, grade, dti, installment, emp_length, loan_amount, annual_income, revol_bal, revol_util. Out of these weights, we saw that loan_amount, grade, subgrade, annual_income were all negative. All the other traits were positive. Since loan is classified as 0, these traits being negative allows the probability of the data being classified as 1 (not paid) to decrease, thus these traits are most likely to influence a decision being made for fully paid. From this project, we learned the importance of processing data. No matter how we set up our model, our original prepossessing data ordinal encoded the states, leading us to run into poor errors. However, after implementing target encoding, we were able to produce much better results. We also originally did not include any attributes in the dataset that consisting of strings (specifically, purpose). However, we decided that information being excluded could potentially be important for classification. Thus, we ended up one-hot encoding multiple attributes from the dataset: home_ownership, verification_status, purpose. Ultimately, we could have spent more time preprocessing data, focusing on extracting more information from the data and less on running GridSearch. We eventually realized that GridSearch is too computationally and time-expensive to actively spend time on. Instead, running our own makeshift GridSearch as well as working on other parameters on the side were much more efficient.

5.2 Challenges

The main obstacle we encountered was lack of data processing. Initially, our strategy was to hot encode most of the features, including both ordinal and nominal features. After trying out different parameters for different models and not showing much improvement in accuracy, we have concluded that there is a problem with how we processed our data. Some things that we didn't account for was one-hot encoding "states" attribute, where we later concluded that this would unnecessarily add 50 variables, leading to over-fitting problems. Next time, we could spend more time in data processing, making sure that we used an appropriate encoding techniques for each states. In addition, without putting much thought, we filled our Nan values with -1, which didn't help improving our accuracy much. Instead, missing data should have been approached dependent on the feature using mode and arbitrary number imputation and "missing" labels.

6 Extra Credit [5 points]

6.1 Why AUC

Since we are dealing with a binary classification problem, it makes sense to use AUC score as the competition metric. The ROC curve plots the sensitivity against specificity at various threshold values. Thus, the area under the curve, or AUC, measures the ability of a classifier to distinguish between classes. An AUC score of 1 would mean the model can distinguish between all the positive and the negative class points without error. AUC is the best measure of the effectiveness of the classifier, because the higher the AUC score, the more likely the model distinguishes between negative and positive points, which is the ultimate goal when performing classification.

6.2 Parallelization

Among the machine learning methods that we have used, we can parallelize the method for optimizing hyperparameters. To explicate, we have incorporated grid search to optimize our hyper-parameter, and one of the parameter for the GridSearchCV method is n_jobs, and increasing n_jobs number could potentially lead to more efficient hyperparameter optimization through parallelization.

6.3 Additional data

Additional data that could be useful to collect is credit score of the loaner. This data could provide useful since credit score is a very good historical indicator of how reliable and responsible a loaner is with their money. More potential data that could be useful is the age of the loaner. There could be potential relationships between the age of the loaner and the monetary security of the loaner. Two additional variables to collect data for are the borrower's number of inquiries by creditors in the last 6 months and the number of times the borrower had been 30+ days past due on a payment in the past year. A higher number of inquiries by creditors for an individual may indicate they exhibit patterns that warrant inquiries more frequently. An increased number of times a borrower is past due also may be an indicator of a risky and/or irresponsible individual, which may indicate less likely to fully pay off a loan.