Julia Qiu

STAT 303-3: Project Report – Regression

EDA + Cleaning:

After performing EDA and creating visualizations, I noticed two main issues:

(1) **There were many irrelevant variables**. When assessing the numeric predictors, I considered the magnitude of the correlation coefficients as well as the predictors' feature importances in my initial linear + nonlinear models. When assessing categorical predictors, I used groupby aggregation to compute and compare the mean and median values across categories. After developing several models, I also noticed how a few of the variables (e.g. loan_amnt, out_prncp_inv) reappeared in various model summaries, whether as hinge partitions in MARS or as the few coefficients that weren't shrunken to 0 in Lasso regression.

(2) **Several of the categorical variables had extremely high cardinality**. Emp_title is one example of this, as it had over 900 levels. I tried multiple methods of dimension reduction, including trying to categorize levels by keywords (e.g. 'vp', 'sales', 'manager') and then performing groupby aggregation, but this ultimately proved useless, so I ignored the variable in my analysis. This ended up being the case with many categorical variables. I split the date-related variables by month and year and created 'other' categories with variables like addr_state and sub_grade. This regrouping was useful when there were too few observations in some of the original categories or when the distribution of money_made_inv was significantly different across the new categories. Still, after building several different models with the transformed predictors, I realized that including fewer categorical predictors consistently improved prediction (going back to point (1), there seemed to be a lot of unimportant predictors). Therefore, I only ended up considering grade, term, initial_list_status, and application_type in my models.

Final Datasets:

Following data cleaning, I used test_train_split to create the datasets used to train my model. 70% of observations were included in my training data and 30% were included in my testing data.

Model Building Process:

The ensemble model that I ended up submitting to Kaggle was a stacked ensemble model (combined with MARS) featuring a MARS, XGBoost, and Random Forest model. My second model was a weighted average of these three models. The tuning process for each was as follows:

(1) MARS (max_terms = 500; max_degree = 3)
I focused on tuning max_degree, as the model's accuracy didn't vary substantially with different values of max_terms. I selected max_degree based on what minimized 5-fold cross-validation error. MARS ended up performing the best on my data (RMSE = 590.96). I suspect this is because there were many high leverage predictors, and MARS

was able to partition the observations appropriately instead of imposing a global constraint.

(2) **XGBoost (n_estimators = 1000, learning_rate = 0.1, reg_lambda = 0.001, max_depth = 4)**
I used 5-fold cross-validation to tune the XGBoost Model. The values I considered were: [3,4,5] for max_depth, [0.01, 0.05, 0.1, 0.2] for learning_rate, [0, 0.01, 0.001] for reg_lambda, and [150, 175, 250, 500, 1000] for n_estimators. The resulting model had RMSE of 632.77. I also considered other combinations of optimal parameter values based on the cv_results dataframe output, including those with a lower learning rate. However, those combinations performed worse on the test data and also compensated for the lower learning rate with added complexity in other areas. For example, by increasing max_depth or reducing reg_lambda.

(3) **Random Forest (n_estimators = 450, max_features = 28, max_depth = 13)**
I tuned n_estimators, max_features, and max_depth using OOB R-squared. Max_depth was not as helpful in improving prediction accuracy, as expected, but I still tuned it in the hopes that it would help prevent overfitting. The random forest model ended up performing the worst (RMSE = 857.95) out of all the base models. However, adding it to my final ensemble model still increased its prediction accuracy.

Other non-linear models I considered adding to my final ensemble were:

- **AdaBoost (base_estimator = DecisionTreeRegressor(max_depth = 10), n_estimators = 500, learning_rate = 0.01)**
  I tuned this model using 5-fold cross-validation. I considered many potential ranges of hyperparameter values, but RMSE was consistently above 850. I suspect this is because there are many outliers present in the dataset. Since AdaBoost is sensitive to outliers, it makes sense that the model's performance would be poor. Adding the AdaBoost model to my final ensemble model also reduced its prediction accuracy.

Final Ensemble Model #1:

After trying multiple methods of ensembling, including voting, stacking with linear regression, stacking with lasso, and stacking with random forest, I found that MARS with degree 1 performed the best (RMSE = 497.29) with individual predictions from the MARS, XGBoost, and Random Forest models. This performed the best on the Kaggle leaderboard (RMSE 604.68).

Final Ensemble Model #2:

My second best model was a weighted average of the individual MARS, random forest, and XGBoost models' predictions. This weighted average performed better than the other stacking / voting ensemble methods I considered (excluding the MARS stacked model). I weighted models according to their individual prediction accuracies, with MARS (the most accurate) being weighted the highest at .5, XGBoost at .4, and random forest at .1. This resulted in RMSE = 547.96, which was higher than any individual model's RMSE, but the performance on the Kaggle leaderboard was not amazing (RMSE 626.69518).