APAN5200_009
Predictive Analysis Competition (PAC) Project Report
yy3407
Yan Ping Yu

When I got the data, the first thing I did was clean the data by looking at the structure and dropping the "description" columns to reduce the complexity of the analysis. Next, I dealt with the missing values through various methods based on each variable's characteristics. For example, I calculated the percentage of NA for every variable, dropping the high missing percentage column. Then, I transformed some categorical variables into dummy variables so that we could use Mice Ranger of Random Forest method to fill NA in a short period but reasonably. However, some categorical variables weren't filled successfully by Mice Ranger, so I decided to fill NA with mode if the variable's NA percentage is less than 10% and fill NA with probability if the variable's NA percentage ranges from 10% to 40%. Through this practical consideration, my goal was to minimize the data biases when handling the missing values.

In the second part, I did the feature extraction and feature selection so that we can reduce data dimensionality as well as overfitting, deal with multicollinearity, and thus improve the model performance. So in the beginning, I used Bivariate Filter approach to observe the numeric variables' correlation, removing small correlation coefficients and high p-value variables. For categorical variables, I examined the relationship between independent variables and the dependent variable, price, through Linear Regression model. I also used nearZeroVar function to check whether there are low-variance variables and utilized lump as well as other categories methods to handle rare values, thus simplifying the model. What's more, I conducted Multivariate Filter to identify and remove the redundant variables if VIF is more than 10, minimizing the biases within the data. In fact, I also tried to do Subset Selection, but the implementation failed due to the large data size. Even though I dropped some high collinearity predictors to run, but it still failed. Therefore, from this experience, I learned that if I had more time, I should drop more features through correlation metrics and by asking some experts with domain knowledge. Or maybe I could extract sample data from the original data and make sure my sample data has the same distribution as the original data to reduce data size for running the Subset Selection.

In addition to preprocessing the trained data, we should also apply the same preprocessing steps for test data, scoringData, to make features consistent and not

affect the model prediction. Besides, since scoringData existed new values of some variables that trained data didn't have, I removed those variables in both datasets so that the model can be run with better performance.

After completing data preprocessing, I started building the models by using Linear Regression. In the beginning, I tried to put all predictors from the result of the preprocessing to run the model, but it took many hours to complete. Thus, I decided to just pick the variables whose correlation coefficients are higher than 0.4, and then 0.3. Before learning cross validation from the course, I tried to split the original dataset into train and test samples to examine the model performance. According to the examination, the RMSE for test samples of the latter was lower than the previous. After submission, the RMSEs of these two models are 8882.62408 and 8806.57471 respectively, suggesting that Linear Regression model with correlation coefficients higher than 0.3 has better performance. Next, based on the previous results, I used Generalized Additive Models (GAM) to model the non-linear relationships, lowering the RMSE score to 8408.81454.

However, I believe putting more independent variables would lead to higher prediction accuracy. Thus, I chose to build a less time-consuming regression tree with all predictors. Additionally, I also established two regression tree models with features that were selected by Lasso Regression and Principal Component Analysis (PCA). To be more specific, I used these two approaches for numeric variables to shrink the coefficients of less important features to zero and reduce the dimension complexity, helping screen significant features and avoiding overfitting. The reason for not applying all predictors to Lasso Regression and Principal Component Analysis (PCA) was that the pre-requisite of doing so was to transform all categorical variables into dummy variables, which would take time. And since most machine learning models can handle categorical variables, there was no big motivation for me to do this higher time-cost transformation. Nevertheless, after submitting these three models, I got three higher RMSEs than the ones from Linear Regression models. I thought it was because first, although Lasso Regression and Principal Component Analysis (PCA) can avoid overfitting, these two selections would lead to loss of data information, and oversimplifying the relationships among predictors. Second, the regression tree model was easily overfitting, leading to high variance in prediction and thus high RMSE. Therefore, I made the improvement by tuning the all predictors regression tree with cross-validation, and the RMSE was 3809.52927, which showed using cross-validation to find the best hyperparameters worked.

The next model I used was tuned random forest. However, because I set many hyperparameters to run the cross validation, I didn't get the outcome even though I spent over 12 hours running it. Consequently, I shifted my focus to alternative models, such as xgb and xgboost. Notably, the xgboost model demonstrated exceptional performance at a low RMSE of 1702.67443. Encouraged by this success, I tried more potentially effective models, such as deep learning keras model, light gbm, stacking models, etc. The stacking model includes xgboost, gbm, and keras. To refine these models, I employed a combination of train-test-split and cross-validation techniques, manually experimenting with various hyperparameters and assessing their impact based on RMSEs of test samples. Nonetheless, after the submission, these complex models failed to surpass the predictive accuracy of the first xgboost model. I thought the reasons might be those models were subject to overfitting, a common pitfall of more complex algorithms when applied to training data. Reflecting on this experience, I recognize the significance of rigorous cross-validation and the utilization of systematic approaches like expand.grid for hyperparameter tuning. While manual hyperparameter selection might seem expedient, it lacks the comprehensive scope and precision of automated grid searches. In conclusion, I believe a more strategic approach would be to narrow the range of hyperparameters and concentrate on fine-tuning the xgboost model with a blend of cross-validation and systematic hyperparameter optimization. This method could potentially strike a balance between model complexity and the risk of overfitting, thereby enhancing predictive performance.