STAT 542 Project 1: House Price Prediction

Wenbo Fu (679744457), Bingyan Liu (668046518)

1 Data and Objective

The data is a set of house price data. The goal is to reduce the Root-Mean-Squared-Error (RMSE) on logarithmic house prices. The RMSEs should be less than 0.125 for the initial 5 folders and less than 0.135 for the last 5 folders.

Details can be found in `https://liangfgithub.github.io/Proj/F23_Proj1.pdf`.

2 Method

Our algorithm contains two parts: data preprocessing and model fit.

The data preprocessing part takes the training data and testing data as inputs, which contains the following steps:

- On the training data set, fill the missing values in the Garage_Yr_Blt column with value 0.
- Remove imbalanced features, where imbalanced features are determined by more than 95% of the sample feature values are the same.
- Winsorize numerical features, by setting all values higher than 0.95 quantile of all sample feature values to the 0.95 quantile.
- Standardize the numerical features with the StandardScaler class.
- One hot encode categorical features with the OneHotEncoder class.
- On the testing data set, follow similar steps: fill the Garage_Yr_Blt column with value 0; remove imbalanced features found from the training data; winsorize numerical features with the 0.95 quantile values computed from the training data; standarize the numerical features with the fitted StandardScaler from the training data; one hot encode the categorical features with the fitted OneHotEncoder from the training data. Output the processed traning data and testing data.

The model fit part takes the processed traning data and the testing data as inputs, which contains the following steps:

- In the linear model part, we choose elastic net model with l_1 ratio 0.5, the best λ is searched with the grid search and 5 fold cross validation, the candidate values are from the set $\{10^{-4+0.5(i-1)}\}$ with i integers from 1 to 15.
- In the tree model part, we choose XGBoost regressor with 5000 trees, 0.05 learning rate, max depth 6, subsampling rate 0.5.
- Use the processed training data to fit the two models and use the fitted model to fit the processing test data to give two predictions. Save the two predictions in mysubmission1.txt and mysubmission2.txt.

3 Result and Discussion

The linear model RMSEs on the 10 folds:

$$0.1204, 0.1167, 0.1169, 0.119, 0.1100, 0.1344, 0.1346, 0.1317, 0.1342, 0.1257 \tag{1}$$

The XGBoost model RMSEs on the 10 folds:

$$0.1173, 0.1202, 0.1121, 0.1187, 0.1082, 0.1263, 0.1337, 0.1258, 0.1342, 0.1172 \tag{2}$$

The preprocessing time on the 10 folds:

$$0.14, 0.12, 0.12, 0.12, 0.15, 0.23, 0.12, 0.12, 0.13, 0.13 \tag{3}$$

The linear model fitting time:

$$5.55, 1.67, 1.73, 1.58, 1.6, 2.92, 2.14, 1.6, 1.73, 1.61 \tag{4}$$

The XGBoost model fitting time:

$$9.64, 9.13, 8.68, 8.93, 9.21, 9.75, 9.26, 11.2, 10.05, 9.74 \tag{5}$$

All the time all in the unit of seconds. The RMSEs satisfy the criterions. The XGBoost model result is slightly better (average RMSE 0.1214) than the linear model restul (average RMSE 0.1244). The code is excuted on a Macbook Air, 8GB memory, Apple M1 chip.

We did not work too much on hyperparameter tuning. We initially used methods described in the instructor's posts: `https://campuswire.com/c/G06C55090/feed/212`, `https://campuswire.com/c/G06C55090/feed/213`, but the RMSE from fold 6 is slightly greater than 0.135. Thus we improved the preprocessing process with automatically choosing features to transform, unlike pre-specifying features as in the posts, which give similar RMSEs but the criterions are satified for all 10 folds.