Project 1: Predict the Housing Prices in Ames

sh53, 655723273, SICHENG HE

1 Introduction

Ames Housing Data has 2930 rows and 83 columns. There are 81 explanatory variables describing different aspects of residential homes. Our goal is to predict the final price of a home with these explanatory variables.

2 Data processing

The following procedures are step by step.

2.1 Handling missing values

Follow the guideline on missing value by Prof. Liang, the 159 houses miss 'Garage_Yr_Blt' because no garages have ever been built for these houses. So missing values are replaced with 0.

2.2 Remove some features

I decide to remove some imbalanced categorical features, 'Street', 'Utilities', 'Condition_2', 'Roof_Matl', 'Heating', 'Pool_QC', 'Misc_Feature', 'Low_Qual_Fin_SF', 'Pool_Area'. I also remove 'Longitude','Latitude', 'Bsmt_Full_Bath' and 'Bsmt_Half_Bath' because I think they have little relation with the price.
Also 'PID' is removed because apparentlu it's not a useful feature. The 'PID' for test data is stored since we need to include it when giving the final prediction.

2.3 Winsorization

I apply winsorization on some area related features, {"Lot_Frontage", "Lot_Area", "Mas_Vnr_Area", "BsmtFin_SF_2", "Bsmt_Unf_SF", "Total_Bsmt_SF", "Second_Flr_SF", 'First_Flr_SF', "Gr_Liv_Area", "Garage_Area", "Wood_Deck_SF", "Open_Porch_SF", 'Enclosed_Porch', 'Three_season_porch', 'Screen_Porch', "Misc_Val"}
I compute the 95% quantile of these features based on the train data. Then replace all values in the train and the test that are bigger than 95% quantile.

2.4 Standardization

I standardize all numerical features using 'sklearn.preprocessing.StandardScaler()' to make the Ridge regression converge more stable. I use the same scaler when processing test data.

2.5 one-hot encoding

For categorical features with K levels, I generate K binary variables using one-hot encoding on the train data. For the test data, if there is a new level, the resulting one-hot encoded columns for this feature will be all zeros. Since I set "handle_unknown='ignore'" in 'sklearn.preprocessing.OneHotEncoder()'

2.6 Feature interaction

I choose some important features to do feature interaction because I think these interactions may contribute to the accuracy of the model.
For example, when interacting 'Year_Built' and 'Bsmt_Unf_SF', a new column named 'Year_Built_x_Bsmt_Unf_SF' will be generated. And it is the product of previous two columns.
Other feature interaction: {'Year_Built x 'Lot_Area', 'Year_Built x 'Garage_Area', 'Mo_Sold x 'Garage_Area', 'Mo_Sold x 'Year_Built', 'Mo_Sold x 'Bsmt_Unf_SF'}

3 Model fitting and prediction

The price is transformed to log scale when fitting two models.

3.1 Ridge regression

I choose the parameter α by 10-fold cross validation
using 'sklearn.linear_model.RidgeCV(alphas=ridge_alphas, normalize=False, cv=10)'. The range is set as 'np.linspace(4, 15, 100)' and each optimal α falls in it for each train-test split. Then fit a Ridge regression model using the optimal α on train data.

3.2 XGBoost

I train a xgboost model using 'xgboost.XGBRegressor()' with the following training parameters, which gives a relatively good result.

- $n\_estimators = 500$. Number of gradient boosted trees.
- $max\_depth = 5$. Maximum tree depth for base learners.
- $learning\_rate = 0.05$. Boosting learning rate. Step size shrinkage used in update to prevents overfitting.
- $subsample = 0.5$. Subsample ratio of the training instance. Setting it to 0.5 means that XGBoost would randomly sample half of the training data prior to growing trees.
- $random\_state = 2021$. Random number seed.

3.3 Running time and the accuracy

Runing time and the accuracy for 10 train-test split are computed. The prediction takes little time so I negelect it.
Note that time for fitting ridge regression includes the cross validation procedure.
My computer system: RedmiBook Pro 14, 1.80GHz, 16GB memory, Windows10.

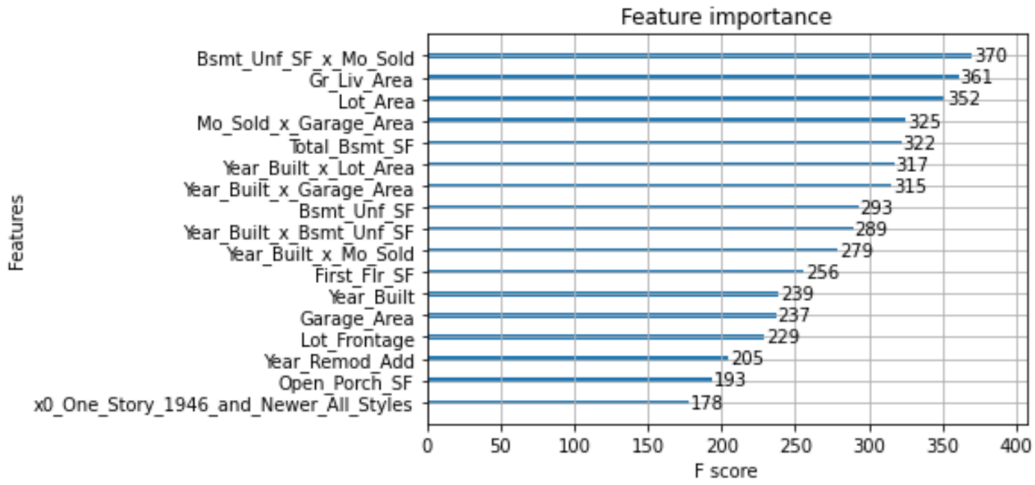
Splits	ridge_train(s)	ridge_RMSE	xgb_train(s)	xgb_RMSE
1	19.6	0.1226	2.636	0.1193
2	18.832	0.1158	2.336	0.1166
3	18.994	0.1135	2.509	0.1118
4	18.744	0.1144	2.749	0.1235
5	19.885	0.1088	2.447	0.1131
6	19.245	0.1335	2.584	0.1295
7	18.63	0.1307	2.278	0.1342
8	19.162	0.126	2.546	0.1264
9	18.177	0.1292	2.385	0.1275
10	18.547	0.1226	2.304	0.1216
Total time	189.816		24.773	

Table 1: Training time and RMSE for 10 splits

4 Discussion

- Feature engineering is important.
  Feature removing, winsorization and including categorical features by one-hot encoding all improve the test accuracy in my models.

- Feature interaction can help, but requiring additional information.
  I choose some important features to do feature interaction and some of the interacted features are important in the XGBoost model.



You can see that several interacted features have high importance from the importance plot of the XGBoost model(top 17). They contribute to the test accuracy. However it is difficult to decide how to interact different features.