Project I: Predict the Housing Prices in Ames

tianni2@illinois.edu

October 15, 2021

1.Dataset Introduction and Analysis Goal

The dataset we use contains 2930 rows and 83 columns. It is from the Ames Assessor's Office used to compute predicted house prices. Within the 83 columns, housing price would be the response in my analysis. I would build two prediction models(one linear regression model with Elasticnet penalty and one XGBoost tree model). And the performance target would be RMSEs less than

0.125 for the first 5 training/test splits and

0.135 for the remaining 5 training/test splits.

2.Pre-Processing Procedure

2.1 Missing values

There are 159 missing values which all from the "Garage_Yr_Blt". By looking through several correlated variables, I found the 159 houses miss "Garage_Yr_Blt" because no garages have ever been built for these houses. In this case, I would just impute these missing values with 0.

2.2 Remove unnecessary variables

To prevent overfitting, remove some extremely unbalanced categorical variables. And remove "Longitude" and "Latitude" because they are not interpretable.

```
Removed variables:
"Street","Utilities","Condition_2","Roof_Matl","Heating","Pool_QC",
"Misc_Feature","Low_Qual_Fin_SF","Pool_Area","Longitude","Latitude"
```

2.3 Winsorization

Apply winsorization on some numerical variable with upper 95% quantile of that variable based on the train data. Because some area-related variables should be capped to better approximated the selling price by a linear relationship.

```
Winsorized variables:
"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",
```

2.4 Transformed X matrix to a numerical matrix

To fit XGBoost and linear regression model with elasticnet penalty, I need to transform the design matrix to a numerical matrix. For tree model, we need to generate K binary categorical variables when K>2. The transformed matrix could also be used in linear regression model since the redundant features would not be selected by glmnet function.

To make the fitted model using train data works well in test data, I also need to prune the test data matrix to make its columns names and order the same as train data matrix.

2.5 Take the log scale of response into calculation

To prevent infinities and nans popping up in the metric calculations, I would fitted the model to predicted the logarithm of the price. And transform the price back using exponential when output the predicted Sales_Price.

3. Building Model and Tuning Parameter

3.1 Linear regression model with elasticnet penalty 0.5

Instead of using Ridge regression or Lasso regression, I choose linear regression with elasticnec penalty because the alpha could be adjusted through 0 to 1. I could tune alpha to fit a model with better prediction. I create a sequence (0.1,...,1) of alpha and calculate the RMSE with different alpha in a loop and pick the alpha which gives the lowest averaged RMSE(over the ten training/test data split).
Below is the table which gives the averaged RMSE of different alpha.

alpha	Averages RMSE
0.1	0.1229917
0.2	0.1229840
0.3	0.1228717
0.4	0.1227662
0.5	0.1228138
0.6	0.1231402
0.7	0.1230443
0.8	0.1230642
0.9	0.1232108
1	0.1241447

Table 1 Averaged RMSE for different alpha values

Note: The lambda value is always the lambda.min which given by the cv.glmnet function in R.

3.2 XGBoost tree model

There are several parameters in the xgboost function needed to be tuned.

Eta: control the learning rate, It is used to prevent overfitting. Lower value for eta implies larger value for n rounds. Low eta value means model more robust to overfitting but slower to compute. We selected 0.05 by compare the calculated RMSE values.

Max_depth: maximum depth of a tree. Default: 6.

Subsample: subsample ratio of the training instance. Setting it to 0.5 means that xgboost randomly collected half of the data instances to grow trees and this will

prevent overfitting.

Nrounds: max number of boosting iterations. We would use 5000 since we have a relatively small eta.

Below is a table (Table 2) which gives the averaged RMSE of different eta while other 3 parameters are constant.

eta	Averaged RMSE
0.01	0.1192660
0.02	0.1200814
0.03	0.1199727
0.04	0.1218123
0.05	0.1225904
0.06	0.1228900
0.07	0.1247862
0.08	0.1245572
0.09	0.1259697
0.1	0.1252217

Table 2 Averaged RMSEs for different eta values

Split	RMSE(Elastic penalty)	RMSE(XGBoost)
1	0.1227220	0.1116967
2	0.1179511	0.1160690
3	0.1213594	0.1106345
4	0.1202252	0.1143801
5	0.1117356	0.1076268
6	0.1333518	0.1274230
7	0.1256737	0.1306816
8	0.1206078	0.1259632
9	0.1306877	0.1268138
10	0.1233477	0.1213708

Table 3 RMSE values for each split

4. Performance

Using the two models I describe above, I calculated the RMSEs of the ten split training/test data. The summary table is above, Table 3.
We can see from the table that all results satisfy the prediction requirements.

5. Running Time

With 2019 MacBook Pro, 2.4 GHz Intel Core i5 CPU with 16GB memory. The averaged running time for each split is around 2min and 35 seconds. XGBoost takes up about 2min and 30 seconds while the elastic regression only cost 5 seconds.

6. Conclusion

For both linear regression model with elasticnet penalty and XGBoost tree model, not only one parameter combination could satisfy the prediction requirement based on the ten training/test data. To get the lowest RMSE, I choose alpha=0.1 for elasticnet penalty and eta=0.04 for XGBoost. Moreover, since the lower eta value, I need to set a quite high nrounds value, which make the running time for XGBoost model pretty long compared to the linear regression model.

7. Acknowledgment

Professor Liang's Campuswire posting for the Project 1(Fall 2021)
R document about the xgboost function in xgboost package