Project1 AMES Housing REPORT

Yuan Gao (netid:yuang7)

10/19/2020

Introduction and Exploration

The goal of this project is to build predictive models based on Ames Housing data to predict the sale price of homes with small RMSE. The Ames Housing Data has 2930 rows and 83 columns. The last column is the response variable, Sale_Price.

The training dataset has 2051 row 83 columns
There are three data types in this training dataset, including 46 categorical, 34 integer and 3 numeric

Data Preprocessing & Feature Engineering

Data Transformation

The distribution of the reponse, Sale_Price, is skewed right, meaning that there are a large number of pricy houses in our dataset. The skewness may violate normailty assumption and result in less accurate prediction. Therefore, we perform log transformation of Sale_Price in the hope of making the distributionmore symmetrical and achieve normality prior to fitting our prediction models.
Observing the histgram plots of before and after transformation, we see the the right-skewed distribution is transformed to a fairly normal distribution.

Missing Values

Find the number of missing value by columns in the training dataset. After performing the calculation, we find that there is only one column, Garage_Yr_Blt has missing values, and the number of its missing value is 107.
By analyzing the relative columns to Garage_Yr_Blt for these rows with missing values. Garage_Type, Garage_Qual, Garage_Cond all have 'No_Garage' as value for these rows. We believe that there is no garage in these houses.So We will simply replace its missing value with 0.
We will replace missing value with 0 for Garage_Yr_Blt in training data as well as in test dataset.

Remove Variables

Variable removed: 'Street', 'Utilities', 'Condition_2', 'Roof_Matl', 'Heating', 'Pool_QC', 'Misc_Feature', 'Low_Qual_Fin_SF', 'Pool_Area', Longitude','Latitude'. They are from analyzing the training data. We will remove them from both training and test data.

Method: I performed barplot on categorical columns, to compare the number of variables between groups for each column.
For columns of non categorical data, I performed histogram plot to observe the distribution of the variable. By observing all the plots, I decided to remove some variables with highly unbalanced data.

- Categorical variables who has heavly unbalance groups:
  'Street' has 11 Grvl and 2040 Pave,
  'Utility' has 2049 AllPub and 2 NoSewr,

'Condition_2' has 4 Artery,11 Feedr,2025 Norm,4 PosA, 4 PosN, 1 RRAn, 2 RRNn,
'Roof_Matl' has 2021 CompShg, 1 Membran, 1 Metal, 19 Tar&Grv, 4 WdShake, 5 WdShngl,
'Heating' has 1 Floor, 2022 GasA, 18 GasW, 6 Grav, 2 OthW, 2Wall 'Pool_QC' has 2 Excellent, 2 Fair, 2 Good, 2044 No_Pool,2Typical
'Misc_Feature' has 2 Gar2, 1975 None, 2 Othr, 71 Shed, 1 TenC

- Numeric/Integer variable has too many 0s:
'Low_Qual_Fin_SF' has 2022 0s
'Pool_Area' has 2044 0s

- For Easier Interpretation
I also dropped 'Longitude' and 'Latitude' because they are hard to interpret. Also we have neighborhood variable, which represent the location of the house already.

Deal with Outliers using Winsorization

Winsorization is to minimize the influence of outliers in the dataset. It is done by substituting the data under or above the threshhold with the threshold data. In this way the outliers is replaced by a less extreme data and therefore make the data less spread out and predict better. It will improve the robustness of the statistical inference.
We will perform winsorization on both training and testdata.For this training dataset, the more extreme values are winsorized to the 95% quantile. When winsorize on testdata, use the 95% quantile of the trainning data.

Numerical Matrix Generation
Before using our data to fit the models, we need to convert the our data into numerical matrix. This is because xgboost can only take numeric matrix.
For each categorical variables wit h K levels, we generate K binary variables when K>2.

Models and Results

Performance Target is to get RMSEs less than
- 0.125 for the first 5 training/test splits and
- 0.135 for the remaining 5 training/test splits.
The model selection is based one RMSE for these 10 training/test splits

Model Selection

A. Linear Regression:Elastic Net Regression

I have tried fitting lasso regression model and elastic net regression model. some of the training/test splits fail to get RMSE less than the threshold in lasso regression model. The elastic net regression model works well to pass all the RMSE threshold.

Lasso Regression RMSE for 10 splits:
0.1271395 0.1216720 0.1244006 0.1347718 0.1204031 0.1409125 0.1368296 0.1364899 0.1383060 0.1313936

Pick tuning parameter lambda: cv.glmnet use cross validatino to find the optimal lambda.1se, which is one standard error from the lambda.min. Like lambda.min, lambda.lse will provide small rmse and easier to interpret than lambda.min.
Fit Model: Fit lasso and elastic net regression model using our training data and lambda.lse. Prediction: Fit testdata to the desired model

B. Tree-based model: Xgboost

- Parameters max_depth: It controls the depth of the tree. Larger the depth, more complicated will be the tree and more chance of high variance. Can be tuned by cross validation. I use 6, which is

the default value for max_depth eta: It controls the learning rate, usually is between 0.01-0.3. The smaller the learning rate, the slower the computation. Usually in overfitting, we will reduce eta and increase nrounds to improve the modeling.
subsamples: Number of samples in a tree. The default is 1. Here I use 0.5 in order to avoid overfitting.
nrounds: Number of iterations/ Number of trees

Tuning nrounds: I have tried 100,300,500,1000,3000,5000,6000 for nrounds. Only nrounds of 100 fails the RMSE threshold. The most of 10 RMSE will reduce significantly as nrounds increase from 100 to 500. From 500 to 6000, RMSE doesn't improve significantly and from 5000 to 6000 most RMSE get worse.I settle with nrounds = 1000 for the RMSE meet my target and increase nrounds won't help much but increase computation time significantly.

Models Picked: Elastic Net Regression & Xgboost

Result Table: RMSE for 10 Splits for these 2 models

Model	split 1	split 2	split 3	split 4	split 5
Elastic	0.1224717	0.1181711	0.1203595	0.1196950	0.1114353
xgboost	0.1173353	0.1200864	0.1135985	0.1147410	0.1122219

Model	split 6	split 7	split 8	split 9	split 10
Elastic	0.1332790	0.1261135	0.1194532	0.1296899	0.1234283
Xgboost	0.1323147	0.1329846	0.1255887	0.1271440	0.1234586

Both models meet the target RMSEs

Run time for elastic net regression & Xgboost

-Runtime for elastic net regession for 10 Cross Validation:Time difference of 21.28602 secs
-Runtime for Xgboost for 10 Cross Validation:Time difference of 25.32977 mins
-computer system: macOs, MacBook Pro, 1.4 GHz Intel Core i5

Lessons learned

- Data preprocessing, feature engineering, and parameter tuning in modeling are all very important in making a predictive model. A small difference in them may have very different result.
- We should preprocess the testdata according to the training data.
  I coded the testdata to numerical matrix without refering to training data variables. As a result, new variables in testdata, which was unseen in training data, resulted in error when making prediction.

Conclusion: Both elastic net regression and xgboost works well in this dataset. Xgboost has more tuning parameter and is more flexible and may require more working in tuning. With good data preprocessing and feature engineering, both linear regression and tree based model can work.