Project 1 – Prediction of House Sale Price

PSL – Fall 2021

Team

Name	Netid	Contribution
Bhuvaneswari Venkatraman	bv10	Worked on XGBoost model, data cleaning and processing along with report
Swapnil Naik	ssnaik2	Worked on Lasso\ridge model and report

Introduction

Ames housing dataset is a popular one with around 2930 rows(i.e., houses) and 83 columns (i.e., several variables\features describing each house). There are several type of data available across the dataset where the first column is "PID" and the last column is "Sale price" which is the response variable. On drawing a histogram of sale prices from the data set, we could see some outliers where a house price is > 750k and other one is <12.5k. Mean and median sale price is $180,796 and $160,000 respectively.

Goal

The goal is to predict the final price of a home (in log scale) using AMES housing dataset. The two prediction models chosen are Lasso/Ridge Regression and XGBoost to predict the sale price.

Design Overview

In general, both the models are approached with the similar design workflow as below

Data Preprocessing

- Encode categorical variables to numeric.
- Impute missing values with median value.
- Winsorize all possible numbers to 95th percentile thereby the substitute values are the most extreme retained values.

```
"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"
```

- To avoid feature column count mismatch between train and test data set for lasso, we specially created to function to get difference between each and include with mean values in both the train and test data set.
- Predictors with imbalance data are removed. Sample predictors are in picture below.

```
Street', 'Utilities', 'Condition_2', 'Roof_Matl', 'Heating', 'Pool_QC', 'Misc_Feature',
'Low_Qual_Fin_SF', 'Pool_Area', 'Longitude','Latitude'
```

- To deal with the missing data in the "Garage_Yr_Blt" variable, we replaced them with zero. This replacement makes sense because these records indicated that they don't have a garage built based on a close examination of the data.
- We used cook distance to remove outliers in training data.

Model

Model 1 : Linear – glmnet

We performed a 10 fold cross validation to choose the optimal lambda hyperparameter. Linear model was created with R glmnet package with the below training parameters and lambda value from Lasso to train Ridge model in order to remove colinearity of remaining variables.

- **alpha –** elastic net mixing parameter can range between 0 (for L2 regularization - ridge) and 1 (for L1 regularization lasso – default). We chose 0.7 after several trials to tune for a better performance.
- **lambda –** regularization parameter supplied as a sequence of values which are identified by performing a 10 fold cross validation
- **nlambda –** fits for 20 different lambda values from the given lambda sequence of values defined.
- **family –** gaussian – default parameter coz we would want the glmnet model to standardize y to have unit variance before computing its lambda sequence and then unstandardize the resulting coefficients

Model 2 : XGBoost

We tried out extensive list of combinations to find out an optimal and high performing hyperparameters for eta, maxdepth and subsample. Ran model selection by running for more than 7000 times to choose lowest error and corresponding hyperparameter values.

- **max_depth –** maximum depth of the tree, is set as 4. It is used to control over-fitting as higher depth will allow model to learn relations very specific to a particular sample.
- **eta -** step size shrinkage used in update to prevent overfitting. It is set to 0.0474239 as identified as optimal from multiple runs
- **subsample -** denotes the fraction of observations to be randomly sampled for each tree. It is set to 0.541795 as identified as optimal from multiple runs

Results

Using our testing driver code, which creates 10 train/test split and evalute RMSE using each output of submitted mymain.R script, we getting following RMSE result.

Model	rmse1	rmse2	rmse3	rmse4	rmse5	rmse6	rmse7	rmse8	rmse9	rmse10
XGBoost	0.1118	0.1220	0.1228	0.1138	0.1248	0.1258	0.1249	0.1238	0.1269	0.1293
GLMNet	0.1226	0.1214	0.1235	0.1226	0.1180	0.1295	0.1256	0.1234	0.1222	0.1264

Based the above result, it seems both models achieved the performance target of RMSE below 0.125 for first 5 data sets, and RMSE below 0.135 for last 5 data sets.

Running Time

Using a 2017 model of Macbook Pro (which has a 2.9 GHz quad-core Intel i7 CPU with 16 GB of RAM), the average running time for each train/test split is about 44 seconds. The training of the xgboost model (Model 1) took almost 90% of execution time. The time takes for all other processing steps, including data preprocessing, training of the glmnet model, and making predictions take only a few seconds.

Interesting Findings

- We were not able to satisfy bench mark RMSE requirement for regression model by using only Lasso or Ridge regularization alone. We ended up need to have extra train data preprocessing to remove outliners based on cook's distance. We also need to use Lasso to drop insignificant variables, and cross- validation to get lambda min first. We then use Ridge to regularize the remaining variables which have colinearity among them.

- The hyperparameter tuning for XGBoost runs too long and to tune Lambda, MaX_Depth and Sub- Sample, took 16 hours to find the optimal value by running 7000 models in my laptop.

Conclusion

In this project, we built two models for predicting the Housing prices in Ames data. Following the professor's guidance, we created the first model using the R xgboost package and made the second model using the R glmnet package with an alpha parameter of 0.7 (Elastic-Net). Running ten train/test splits using both models, we achieve the target test RMSE of below 0.125 for the first 5 data set and below 0.135 for the last 5 datasets. The running time on a Macbook pro with 2.9 GHz 4-Core I7 CPU and 16GB is about 44 seconds per data set. The 90% of execution time is spent on xgboost training.