Project 1: Predict the Housing Prices in Ames

xxx xxx (netid), xxx xxx (netid)

CS 598 Practical Statistical Learning (Fall 2020)

1. Team members and contributions

We, XXX (netid) and XXX (netid) formed a two-member team for this project. We each contributed about 50% of time and effort. We discussed the project requirements, conducted research, and iteratively refined the implementation. XXX mainly focused on first method using eXtreme Gradient Boosting(XGBoost), and XXX mainly on the second method using Glmnet. We reviewed each other work and finalized coding and report together.

2. Technical details

2.1 Goal

This project aims to build two prediction models using the Ames Housing Data, …… … achieve the specified performance target of 0.125 for the first five training/test splits and 0.135 for the remaining five training/test splits. The target values are measured in RMSE on the log scale of the sale price. We are also required to process train and test data separately.

2.2 Overview of steps

To predict the sale price, we utilized R glmnet and xgboost packages. We closely followed the guidance provided by the professor's Piazza postings. To ensure our solution meets the performance target, we created testing driver codes to validate all ten split prediction results by sourcing submitted mymain.R and calculating RMSE on a log scale. We processed training and test data in separate steps. For each train and test data set, we conducted the following steps:

1. …
2. ,,,
3. … 4. 5.
4.

10.

2.3 Common data preprocessing (applies to both Models)

We created a reusable R function to preprocess the training data that can be applied for both models. We first removed the "PID" and the "Sale_price" columns as they don't make sense as training inputs. We did, however, need to save the "Sale_price" as the training label. 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.

Next, to set up the training input, we identify all categorical variables and create a binary dummy variable for each level of categorical variables. ...

The final prepared training input also included all remaining non-categorial variables.

We end up with **304** feature columns for the XGBoost training model due to this processing. This preprocessing function also saves and returns the created dummy columns information for later processing of test data.

2.4 Data pre-processing specific to linear regression model

Before applying the above common data preprocessing to the linear regression model, we introduced two additional preprocessing steps - ...

Based on the professor's Piazza posting, we applied ....

2.5 Training of Model 1 (XGBoost)

We built the first model with the R Extreme Gradient Boosting package (xgboost) with the following training parameters. This set of parameters is the same one that appeared in the professor's Piazza posting, and we found they can meet the performance objective.

- `max_depth = 6`, which specifies the maximum depth of a tree, 6 is the default value.
- `eta = 0.05`, which specifies the step size shrinkage used in the update to prevents overfitting. The default value is 0.3. We used a lower value of 0.05 to make the model more robust to overfitting.
- `nrounds = 5000`. This parameter specifies the maximum number of iterations. We also tried lower 1000, 2000, 3000, 4000, and found 5000 yields better results. However, a large `nrounds` causes more computation time.
- `subsample = 0.5`, which is the subsample ratio of the training instance. The value 0.5 means that xgboost randomly collected half of the data instances to grow trees. This partial ratio helps to prevent overfitting and also reduces the computation because of fewer data to process.

Since the final performance objective(RMSE) is measured in a log scale, we used the log-transformed sale price as the training predictor value.

2.6 Training of Model 2 (Linear regression with Elasticnet penalty)

...

2.7 testing result

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	0.113	0.118	0.112	0.114	0.109	0.126	0.132	0.126	0.130	0.120
Elastic-net	0.123	0.117	0.122	0.120	0.112	0.133	0.126	0.121	0.130	0.124

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.

2.8 Running time

Using a 2015 model of Macbook Pro (which has a 2.8 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.

2.9 Interesting findings

While working on this project, we discovered the following interesting findings:

- Xgboost took much longer time to train. In our case, Xgboost took 95% of total processing time.
- R xgboost package can utilizes multi-core parelell processing on Macos when openmp library is installed on the OS before installing xgboost R package. Otherwise, xgboost training is limited to a single cpu core, which can be serveral time slower for typical computer with multi-core CPU.
- We think Xgboost can be a good model for doing supervised learning with properly tuned parameters.

3 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 0.5 (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 2.8 GHz 4-Core I7 CPU and 16GB is about 44 seconds per data set. The 90% of execution time is spent on xgboost training.

4 Acknowledgment

- Kaggle competition (https://www.kaggle.com/c/house-prices-advanced-regression-techniques)
- Professor Liang's Piazza posting for the Project 1 (Fall 2020)