CS598 - Project 1

Xiaoming Ji

Computer System

Hardware

- Dell Precision Tower 5810
- CPU: Intel Xeon E5-1607 @ 3.10GHz
- Memory: 32GB

Software

- OS: Windows 10 Professional 64bit
- R: 3.5.1
- R Packages:
  - randomForest_4.6-14
  - glmnet_2.0-16
  - xgboost_0.71.2
  - psych_1.8.4

PART 1

In this part, I pre-processed the data and built 2 models (boosting and lasso) to make the training and evaluations.

Preprocessing and Feature Engineering

Several approaches are taken to pre-process the data.

- Missing value: 'Garage_Yr_Blt' has some missing values, 'Year_Built' is used to fill the value. Note: 'Garage_Yr_Blt' is removed due to low importance, I still leave this step for generailize the processing pipeline.
- Handle categorical value mismatch between train and test dataset:
  - For categorical value only exists in test predictor, the value is replaced with the `most frequent` categorical value of the same training predictor.
  - For ordered value only exists in test predictor, the value is replaced with the `closest` value of the same training predictor.
- Fix the skewness of numeric predictors: take the log for all numeric predictors with an absolute skew greater than 0.8.
- Take log for response variable: Sale_Price.
- Add more predictor to help prediction:
  - Total bath: combine all full and half bath rooms.
  - Age: how old the house was when sold.
  - IsNew: whether this is a new house when sold.
  - Remodeled: This is seen as some sort of penalty parameter that indicates that if the Age is based on a remodeling date, it is probably worth less than houses that were built from scratch in that same year.

  - TotalSqFeet: combine space in living area and basement.
  - TotalPorchSF: combine space of all porches.
- Remove predictors: remove some highly correlated and dominate categorical variables predictors.

Note: Winsorizing is not used because per my testing, it doesn't improve the accuracy.

Models

For evaluation purpose, I build 4 models,

- RandomForest
- Boosting (Xgboost)
- Lasso
- MyLasso (self-implemented lasso)

Evaluation

I tested all 10 test dataset against these models. The RMSEs are:

```
print(rmse)
```

```
##           Lasso
##  [1,] 0.1275106
##  [2,] 0.1288868
##  [3,] 0.1376953
##  [4,] 0.1226088
##  [5,] 0.1058202
##  [6,] 0.1227174
##  [7,] 0.1124693
##  [8,] 0.1126837
##  [9,] 0.1189350
## [10,] 0.1078492
```

```
cat("\n Avg:", colMeans(rmse), "\n" )
```

```
##
##  Avg: 0.1197176
```

Computing time: 17.647 seconds

PART 2

In this part, I use my own lasso implementation to predict the test set. In order to find the best λ, A cross validation function: `cv.mylasso` is implemented. Here I used the pre-found λ=30 to shorten the computation time.

Run against the test set, the results are:

- Test Accuracy: 0.1157969
- Computation Time: 9.471 seconds