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 select 2 best performed models (boosting and lasso) to make the training and output the predictions.

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 later removed due to low importance, I still leave this step to generalize the processing pipeline.
- Handle missing categorical level in test dataset:
  - For categorical level, the value is replaced with the `most frequent` categorical level of the same training predictor.
  - For ordered categorical level, 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`.
- Build new predictors to help training/prediction:
  - `TotBathrooms`: 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: 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 predictors.
  - `Garage_Yr_Blt`, `Garage_Area`, `Garage_Cond`, `Total_Bsmt_SF`, `TotRms_AbvGrd`, `BsmtFin_SF_1`, `First_Flr_SF`, `Second_Flr_SF`, `Bedroom_AbvGr`, `Full_Bath`, `Half_Bath`, `Bsmt_Full_Bath`, `Bsmt_Half_Bath`, `Open_Porch_SF`, `Enclosed_Porch`, `Three_season_porch`, `Screen_Porch`, `Street`, `Utilities`, `Land_Slope`, `Condition_2`, `Roof_Matl`, `Heating`, `Pool_QC`, `Misc_Feature`, `Low_Qual_Fin_SF`, `Pool_Area`, `Misc_Val`, `Longitude`, `Latitude`

Note: Winsorization 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:

```
##       RandomForest     Lasso    Xgboost  My_Lasso
##  [1,]    0.1231475 0.1275106 0.11067872 0.1280429
##  [2,]    0.1223392 0.1175602 0.11858552 0.1182908
##  [3,]    0.1229587 0.1174816 0.11799695 0.1177018
##  [4,]    0.1296665 0.1089831 0.11248625 0.1103584
##  [5,]    0.1112760 0.1058984 0.09649456 0.1078236
##  [6,]    0.1254944 0.1090638 0.11280293 0.1115426
##  [7,]    0.1155418 0.1012035 0.10346998 0.1025929
##  [8,]    0.1197746 0.1128877 0.11311719 0.1158970
##  [9,]    0.1289824 0.1098386 0.11571491 0.1120221
## [10,]    0.1262295 0.1081741 0.11597699 0.1077454
```

```
## Overall Mean: 0.1225411 0.1118602 0.1117324 0.1132018
```

```
## Mean of Worst Three: 0.1282928 0.1208508 0.1175198 0.1213452
```

Computation time: 752.27 seconds

Note: Xgboost on my Mac (same R and xgboost versions) perform badly when `colsample_bytree` and `subsample` are not defaults (1). I believe such discrepancy is due to the default Xgboost on Mac is not optimized. Having said that, if I leave these 2 parameters to the defaults, I can still get decent test results on all 10 test dataset (mean:0.1144079, mean over worst three:0.1206602).

According to the testing results, I choose `Boosting` (xgboost) and `Lasso` models to make the prediction. The parameters for building the models are:

- Xgboost: max_depth = 6, eta = 0.03, nrounds = 500, colsample_bytree = 0.6, subsample = 0.75
- Lasso: use cv.glmnet to choose best lambda and use lambda.min to make prediction.

The computation time is: 11.79 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: 3.24 seconds