Project 1

Wenqing Zhu (UIN: 653942417) and Siyuan Qian (UIN:673877907)

I. Contributions

Joint contribution: Discussing the data pre-processing, including predictor selection, object predictor conversion and etc.. Part I and Part II implementation discussion.

Siyuan Qian: Coding implementation of Part I. Wenqing Zhu: Coding implementation of Part II.

II. Hardwares and Softwares

- Apple Macbook Pro
- CPU: Apple M1 pro
- Memory: 16 GB
- Operation System: macOS Ventura 13.4.1(c)
- Language: Python
- Packages: pandas, numpy, copy, sklearn, xgboost,time

III. Pre-processing

- Missing values in 'Garage_Yr_Blt' are replaced by 0.
- Continuous variables are only standardized in Part I.
- We delete some variables which are highly correlated or imbalanced: 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
- We added some variables according to existing data:
  - Total_bathroom = Full_Bath + Half_Bath
  - Age = Year_Remod_Add - Year_Built
  - Total_Area = Total_Bsmt_SF + Gr_Liv_Area
- Winsorization: After standardization, we replace small(large) continuous variables beyond 2σ as minimum(maximum) values @ 2σ

IV. Part I

Models: Lasso, Ridge, Lasso-assisted Ridge(where predictors with zero coefficient in Lasso are eliminated in Ridge)

In my implementation of Part I, I first choose Lasso Regression and tried several ways of data pre-processing and it shows good results on the first five data sets but RMSE in especially data6&7 are higher than requirement. Then, I use Lasso to eliminate the predictors whose coefficients are zero and then put the rest data into Ridge model but it did not show enough improvement. Finally, I directly choose Ridge regression and use sklearn.linear_model.RidgeCV for cross validation and get good results. RMSE: 0.123 Computation Time: 1.7 ns

	Ridge	Lasso	Lasso-assisted Ridge
[1]	0.12271965	0.12180703	0.12699033
[2]	0.11864446	0.11801923	0.11863911
[3]	0.11447305	0.11353173	0.12032083
[4]	0.11627447	0.1199546	0.12293461
[5]	0.10983837	0.11005877	0.11324096
[6]	0.13386634	0.13694421	0.13601891
[7]	0.13346003	0.13642044	0.13660096
[8]	0.1276175	0.12820735	0.12910334
[9]	0.13175045	0.13408022	0.13197842
[10]	0.12354879	0.12586535	0.12635091
Mean	0.12321931	0.12448893	0.12621784

TABLE I: RMSE of various methods

V. PART II

I experimented with two tree-based models: the Random Forest model and the XGBoost model. All predictors were retained in both attempts.

Before utilizing the Random Forest function, some data preprocessing was performed. Initially, missing values in the ”Garage_Yr_Blt” variable was addressed by replacing them with zero. Subsequently, the X matrix was converted into a fully numerical matrix by applying the OneHotEncoder function to each categorical variable. It is also worth noting that, as recommended in Prof. Liang’s Campuswire post, we created K binary dummy variables for categorical variables with K levels to mitigate the missing-intercept issue in tree-based models.

I explored various sets of parameters for the Random Forest model; however, the results did not match the benchmark performance. The initial column in Table 2 presents the outcomes of the Random Forest model with 500 trees.

Subsequently, I implemented the XGBoost model, following the same data preprocessing steps as the Random Forest. I utilized the following parameters: n_estimators = 500, eta = 0.05, subsample = 0.5, and the model’s performance, as assessed by RMSE, is detailed in Table 2.

Running time is 9.86 s.

	Random Forest	XGBoost
[1]	0.13851242	0.11378088
[2]	0.14101569	0.11755641
[3]	0.13282424	0.11589603
[4]	0.13853494	0.11843562
[5]	0.12796759	0.10784543
[6]	0.14911654	0.12866615
[7]	0.15281254	0.13257012
[8]	0.14439276	0.12798373
[9]	0.14867602	0.13178519
[10]	0.12354879	0.11686705
Mean	0.14130731	0.12113866

TABLE II: RMSE of tree-based methods