Razvan Radoi, AI Master 2021
1st year, Spring Semester

SSL ASSIGNMENT REPORT

Submitted in partial fulfillment of the 1st Milestone of the SSL Course Assignment

Chosen topic

House prices prediction[1].

Approaches and Alternatives

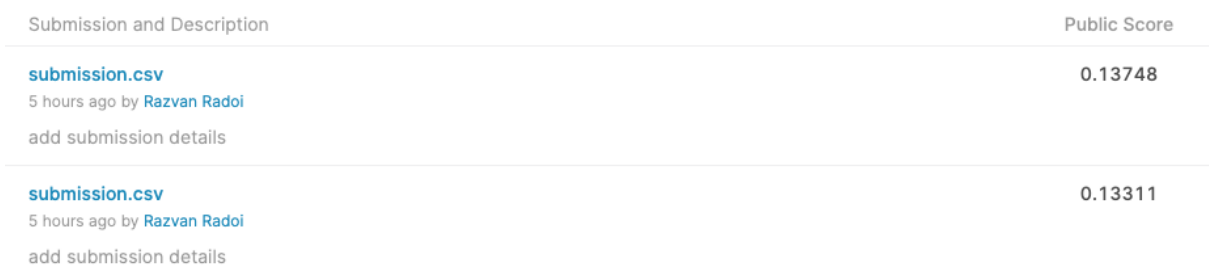
Data Transforms

The dataset provided for this challenge consists of 79 columns that describe house features of residential proprieties in Ames, Iowa. The columns are not "complete" – there are missing values, and some features are numerical, while others are categorical. The data is provided in 2 .csv files, one for training and one for testing. For the training data, labels are also provided – the house sale price.
In my approach, I initially concatenate the train and test data, dropping the ids and the sales price from the data frame. I do the concatenation in order to make sure that transforms like scaling and inputting values are done with respect to the whole dataset. I do the column dropping because Ids are not relevant for the data science experiment and because storing the Sales Price separately made it easier to work with the data – furthermore, applying various transforms to the labels is known to not help in regression experiments.
To account for categorical columns, I first identify the columns that need to be replaced and then for those, I use a Label Encoder [2] to transform labels to numerical ones. An alternative I've tried was one hot encoding [3]. However, experimentally I've observed that performance was better with the Label Encoder – the already large dataset (79 columns) was expanded to hundreds of columns using the one hot encoding technique – this was most likely the cause of the marginally poorer performance. Another briefly considered alternative was the pandas *get_dummies* method. Again, experimentally, results were poorer than when using the Label Encoder.
After applying the label encoding, I use a Simple Imputer [4] to input missing values. I use a "median" value approach – this is to avoid issues with the vastly more popular "average" imputing that would arise from previously using a Label Encoder.
The final transform I apply to the concatenated dataset before splitting it back intro "train" and "test" is to use a Standard Scaler [5] to standardize features by removing the mean and scaling to unit variance.

Fitting the Data

To fit the data, I used an array of regressors: XGBoost Regressor, XGBoost RF Regressor, classical Random Forest Regressor, Gradient Boosting Regressor, Bayesian Ridge, Huber Regression, AdaBoost Regressor, Extra Trees Regressor and Lasso Lars.
Using the dataset transformed as described above, the best performance over a 5-fold of the train dataset as measured by the R^2 score was obtained using the Gradient Boosting Regressor and the XGBoost Regressor - Gradient Boosting score: 0.8925 std: (0.0209); XGBoost Regressor score: 0.8741 std: (0.0285). For the XGBoost approach, I also tried the optimizations outlined in [6], however the performance gain was marginal, and these optimizations were not used in the final XGBoost submission.

Both techniques fared well enough to place the two submissions in the top 35% of the competition leaderboard, as can be seen in the Figure 1, below, or in the public leaderboard.



Submission and Description	Public Score
submission.csv 5 hours ago by Razvan Radoi add submission details	0.13748
submission.csv 5 hours ago by Razvan Radoi add submission details	0.13311

Figure 1. Submissions Performance.

Referenced Techniques and Resources

[1] – https://www.kaggle.com/c/house-prices-advanced-regression-techniques
[2] – https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.LabelEncoder.html
[3] – https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.OneHotEncoder.html
[4] – https://scikit-learn.org/stable/modules/generated/sklearn.impute.SimpleImputer.html
[5] – https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html
[6] – https://www.kaggle.com/serigne/stacked-regressions-top-4-on-leaderboard

Regression Models Sources

[1] – https://scikit-learn.org/stable/modules/ensemble.html
[2] – https://scikit-learn.org/stable/modules/linear_model.html
[3] - https://xgboost.readthedocs.io/en/latest