Project 1, PSL Fall 2024

Olivia Dalglish (od4) and Arindam Saha (saha2)

Contribution:
Olivia: Primarily worked on the linear model
Arindam: Primarily worked on the tree model
We brainstormed various ideas regarding both models to achieve the benchmark.

The goal of this project is to predict the sale price from homes in Ames, Iowa. The dataset contains 81 variables on the homes in the dataset, which was split into 10 separate folds. The folds were effectively treated as individual datasets, and preprocessing steps were applied individually to the folds. This report outlines the key steps taken in preprocessing the dataset and implementing predictive models.

I. Technical Details

a) Data Preprocessing: This section describes the data preprocessing steps applied prior to fitting the model. The same processing was applied to both the training and test datasets. For the training data, we excluded features from the dataset that offered no benefit to the model, as a means of simplifying the model and reducing noise. Naturally, we excluded "PID" (a unique identifier for each property) and "Sale_Price" (the target variable) from the training data, along with "Street", "Utilities", "Condition_2", "Roof_Matl", "Heating", "Pool_QC", "Misc_Feature", "Low_Qual_Fin_SF", "Pool_Area", "Longitude", and "Latitude." These features either provided little variance or were deemed irrelevant based on exploratory analysis. Categorical features, i.e., features that are non-numeric, were encoded using one-hot encoding (with a minimum frequency of 0.01 to minimize the addition of columns that are effectively noise), while numerical features were standardized using StandardScaler, which ensures each feature has a mean of zero and unit variance. This standardization is critical for linear models, which are sensitive to feature scaling. Missing values in the "Garage_Yr_Blt" feature were imputed with zeros, assuming the absence of a garage. Additionally, columns such as "Lot_Frontage", "Lot_Area", "Mas_Vnr_Area", "BsmtFin_SF_2", "Bsmt_Unf_SF", "Total_Bsmt_SF", "Second_Flr_SF", "First_Flr_SF", "Gr_Liv_Area", "Garage_Area", "Wood_Deck_SF", "Open_Porch_SF", "Enclosed_Porch", "Three_season_porch", "Screen_Porch", and "Misc_Val" were winsorized to cap extreme outliers at the 95th percentile. For capped columns, values in the test dataset were limited to the maximum values observed in the training data.
For the test data, we mirrored the preprocessing steps performed on the training data to ensure consistency. Additionally, a log transformation was applied to the target variable ("Sale_Price") to account for its skewed distribution, a common practice in regression tasks involving price predictions.

b) Modeling: We fit two models, a linear model and a tree model. For the linear model, we observed the best results with sklearn's ElasticNetCV (default settings), which combines

the strengths of Lasso and Ridge regression. This hybrid approach is ideal because it performs both feature selection (via Lasso's L1 regularization) and reduces multicollinearity (via Ridge's L2 regularization). ElasticNetCV automatically tunes the hyperparameters through cross-validation, selecting the best alpha (penalty term) to balance model complexity and accuracy. For the tree model, we used the XGBRegressor model, which leverages boosting to improve predictive performance. We configured the model with 5000 estimators (trees) and set the maximum tree depth to 6. Additionally, a learning rate of 0.05 was applied to control the contribution of each tree, preventing overfitting while improving generalization. We also set subsample to 0.5 to prevent ovefitting.

II. Performance Metrics

The following table reports the root-mean squared error for each of the 10 folders for both the linear and tree models.

	Linear RMSE	**Tree RMSE**	**Execution Time**
Fold 1	0.1196	0.1149	12.37
Fold 2	0.1155	0.1181	9.91
Fold 3	0.1128	0.1137	9.76
Fold 4	0.1174	0.1178	12.2
Fold 5	0.1094	0.1057	10.68
Fold 6	0.1284	0.1213	12.47
Fold 7	0.1265	0.129	11.79
Fold 8	0.1236	0.1191	11.15
Fold 9	0.1329	0.1263	10.98
Fold 10	0.1186	0.1128	11.98
Average	0.12047	0.11787	11.329

The above statistics were from running mymain.py on a MacBook Pro, 2.3 GHz Quad-Core Intel Core i7, with 16 GB memory. The run time includes the time to preprocess, train and evaluate both models.

III. Conclusion

In conclusion, the linear model and tree model performed similarly across folds, where RMSE was smaller for linear on 4 folds, and smaller on the tree model for the other 6 folds. The execution time seemed to be disproportionally longer for folds 1,4,6, which could be attributed to the amount of variance in those folds. For future work, the 10 folds could be combined so that cross validation could be performed on a larger combined set of data, as opposed to being treated as 10 distinct datasets.