Technical Report of CS598 Project 1 - Predict the Housing Prices in Ames

Yu-Chung Lee
{ycl7@illinois.edu}
UID: 665451160
Net ID: ycl7

1 Technical Details

1.1 Data Loading and Initial Preparation

The dataset was organized into training and testing sets, each stored in separate CSV files within designated fold directories. For each fold, the training data was loaded from `train.csv`, where the first column (presumably an identifier) was excluded, and the target variable was extracted from the last column. The target variable underwent a logarithmic transformation using the `log1p` function to stabilize variance and normalize the distribution. Similarly, the testing data was loaded from `test.csv`, and the corresponding true target values were obtained from `test_y.csv`, also applying the logarithmic transformation.

1.2 Data Cleaning

1.2.1 Numerical Features

- **Garage Year Built (`Garage_Yr_Blt`):** Values exceeding the maximum year of 2011 were considered corrupted and set to `NaN`. Missing values in `Garage_Yr_Blt` were subsequently imputed using the `Year_Built` feature to ensure consistency.

1.2.2 Categorical Features

Features with missing values were identified by calculating the sum of nulls in each column. Any categorical feature containing missing values was excluded from the analysis by dropping these columns entirely. This step ensured that the dataset used for modeling did not contain incomplete categorical information.

1.3 Feature Engineering

1.3.1 Numerical Features

A predefined list of numerical features (`linear_numeric_feats`) was selected for analysis. These included variables such as `Lot_Area`, `Year_Built`, `Total_Bsmt_SF`, `Gr_Liv_Area`, and geographic coordinates (`Latitude`, `Longitude`).

Skewness Transformation The skewness of each numerical feature was assessed using the `skew` function from the `scipy.stats` module. Features exhibiting skewness greater than a threshold of 0.75 were identified as significantly skewed. These skewed features underwent a logarithmic transformation (`log1p`) to reduce skewness and approximate a normal distribution.

1.3.2 Outlier Removal

For each feature in `linear_numeric_feats`, outliers were detected using the Z-score method. Data points with an absolute Z-score exceeding a threshold of 5 were considered outliers and subsequently removed from both the feature set (`X_train_processed`) and the target variable (`y_train_processed`). This process was iteratively applied to all relevant numerical features to enhance the robustness of the models.

1.3.3 Categorical Encoding

Categorical variables were transformed into numerical representations using one-hot encoding via the `pd.get_dummies` function. To ensure consistency between training and testing datasets, the testing set was reindexed to match the columns of the training set, filling any missing categories with zeros.

1.4 Model Implementation

Multiple regression models were employed to predict the target variable, each utilizing different algorithms and hyperparameters:

1. **Linear Regression (`LinearRegression`):**

- A basic linear regression model was fitted to the processed training data.
- **Training Error:** Calculated by predicting on the training set and computing the Root Mean Squared Error (RMSE) against the actual values.
- **Test Error:** RMSE calculated on the test set predictions.

2. **Ridge Regression (`RidgeCV`):**
   - Utilized cross-validated Ridge regression with a pipeline that included `RobustScaler` for feature scaling.
   - A range of alpha values was explored using `np.logspace(-1, 3, 100)` to identify the optimal regularization strength.
   - Both training and test RMSE were reported.
3. **Lasso Regression (`LassoCV`):**
   - Employed cross-validated Lasso regression to enforce sparsity in the model coefficients.
   - Alpha values were sampled from a logarithmic space between `1e-5` and `1e-2`.
   - Training and test RMSE were computed based on the optimal alpha.
4. **XGBoost Regressor (`XGBRegressor`):**
   - Configured with predefined hyperparameters, including `max_depth`, `learning_rate`, `n_estimators`, and regularization terms (`reg_alpha`, `reg_lambda`).
   - The model was trained on the processed data, and both training and test RMSE were evaluated.

1.4.1 Hyperparameter Tuning

An Optuna-based hyperparameter optimization framework was set up to fine-tune the `XGBRegressor` parameters.

- The `objective` function defined the search space for parameters such as `max_depth`, `learning_rate`, `n_estimators`, `min_child_weight`, `subsample`, `colsample_bytree`, `reg_alpha`, and `reg_lambda`.
- A Tree-structured Parzen Estimator (TPE) sampler with a fixed random seed ensured reproducibility.
- The optimization aimed to minimize RMSE using 5-fold cross-validation over 50 trials. The best parameters identified from this search were intended to be applied to retrain the XGBoost model, although this step was commented out in the current implementation.

1.5 Execution Workflow

For each of the 10 predefined folds:

1. **Data Loading:** Train and test datasets for the respective fold were loaded.
2. **Data Cleaning and Transformation:** Applied the cleaning, transformation, and encoding steps as detailed above.
3. **Model Training and Evaluation:** Each regression model was trained on the processed training data and evaluated on both training and test sets using RMSE as the performance metric.
4. **Reporting:** RMSE values for each model were printed for both training and test datasets to assess performance consistency and potential overfitting.

2 Performance Metrics

Submission are evaluated on Root-Mean-Squared-Error (RMSE) between the natural logarithm of the predicted price and the natural logarithm of the observed sales price. The following table summarizes the RMSE achieved by each model across all 10 training/test splits. Lower RMSE values indicate better predictive performance.

	RMSE (1-5 folds)	RMSE (6-10 folds)
Ridge	0.11183	0.12515
Lasso	0.11183	0.12515
XGBoost	0.11183	0.12515

Table 1: RMSE of each model across 10 folds

- Hardware used in this project:
  - CPU: AMD Ryzen 5 3600 6-Core Processor with 48GB RAM
  - GPU: GeForce RTX 3090 Ti 24GB