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. The training data was loaded from `train.csv`, where the first column (PID) 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. Sale price with extreme values were removed by identifying rows with price larger or smaller than 3 standard deviations from the mean.

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.

1.2.3 Imbalanced and Irrelevant Variables

As suggested by the instructor, the following variables were removed from the dataset: `Street, Utilities, Condition_2, Roof_Matl, Heating, Pool_QC, Misc_Val, Low_Qual_Fin_SF, Pool_Area, Longitude, Latitude`. These variables were consided imbalanced categorical variables with most samples belonging to a single category or don't offer interpretable information.

1.3 Feature Transformation

1.3.1 Numerical Features

Numerical features were identified by checking the data type of each column. They were further processed with winsorization, skewness transformation and outlier removal.

Winsorization For certain numerical variables, values exceeding the 95th percentile were capped at the 95th percentile. These variables include `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`.

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 0.5 were identified as significantly skewed. These skewed features underwent a logarithmic transformation (`log1p`) to reduce skewness and approximate a normal distribution. Features with such transformations were identified as `skewed_feats` and will perform the same logarithmic transformation on testing set.

Outlier Removal For all numerical features, 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 and the target variable. This process was iteratively applied to all numerical features to enhance the robustness of the models.

1.3.2 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. **Ridge Regression (`RidgeCV`):**
   - Utilized 5-fold 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.
   - Used negative root mean squared error as the scoring metric.
2. **XGBoost Regressor (`XGBRegressor`):**
   - Configured with predefined hyperparameters, including `max_depth`, `learning_rate`, `n_estimators`, and regularization terms (`reg_alpha`, `reg_lambda`).

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`.
- The optimization aimed to minimize RMSE using 5-fold cross-validation over 50 trials. The best parameters identified from this search is applied to retrain the XGBoost model.

1.5 Execution Workflow

For each of the 10 predefined folds:

1. **Preprocessing training data:** ONLY train datasets for the respective fold were loaded. Then applied the cleaning, transformation, and encoding steps as detailed above.
2. **Model Training:** Each regression model was trained on the processed training data.
3. **Preprocessing test data:** Test datasets for the respective fold were loaded. Then applied the same preprocessing steps as training data.
4. **Prediction:** Predictions on the test dataset were generated and saved into `mysubmission1.csv` and `mysubmission2.csv` for Ridge and XGBoost models, respectively.

2 Performance Metrics

Models were evaluated with 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. Both the Ridge regression model and XGBoost model arhived the desired RMSE: 0.125 for the initial 5 training/test splits and 0.135 for the subsequent 5 training/test splits.

RMSE / Runtime	Ridge	XGBoost
fold1	0.116114 / 3.20s	0.111148 / 6.39s
fold2	0.117437 / 2.99s	0.114520 / 6.17s
fold3	0.115723 / 2.99s	0.110675 / 6.60s
fold4	0.115164 / 2.86s	0.115495 / 6.47s
fold5	0.108005 / 2.93s	0.104979 / 6.20s
fold6	0.128512 / 3.00s	0.123666 / 6.16s
fold7	0.130906 / 3.00s	0.129087 / 6.98s
fold8	0.129505 / 3.13s	0.126017 / 6.41s
fold9	0.130229 / 2.72s	0.130740 / 6.41s
fold10	0.122325 / 3.00s	0.122325 / 6.21s

Table 1: RMSE of each model across 10 folds

2.1 Hardware Used

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