House Price Prediction

Andrew Jowe
ajowe@ucdavis.edu

Abstract
We explore predictive modeling of housing prices using Ridge, Lasso, Huber Regression, Random Forest, and XGBoost on the Ames Housing dataset. A consistent preprocessing pipeline ensured comparability across models. Evaluation was based on 10-fold cross-validation and Kaggle leaderboard scores using log RMSE. Ridge and Lasso showed the best generalization, though diagnostic plots revealed assumption violations. Huber Regression underperformed due to its conservative handling of outliers, while tree-based methods overfit despite low training errors. We propose hybrid modeling—combining linear and tree-based approaches—as a promising direction for handling both linear and non-linear data regions.

Contents

1 Introduction 1
1.1 Literature Review 2

2 Exploratory Data Analysis 3
2.1 Missing Value Analysis 3
2.2 Skewness 3
2.3 Correlation 4
2.4 Pairplots 4

3 Methodology 5
3.1 Preprocessing Pipeline 5
3.2 Cross Validation 6
3.3 Model Evaluation 6
3.4 Ridge Regression 6
3.5 Lasso Regression 7
3.6 Huber Regression 7
3.7 Random Forest 7
3.8 XGBoost 7

4 Results 8

5 Limitations 10

6 Conclusion 11

1 Introduction

Accurately estimating housing prices is a long-standing challenge with significant implications for homebuyers, real estate professionals, and policy makers. The Kaggle competition House Prices: Advanced Regression Techniques frames this problem in a structured data science context, offering a rich dataset of residential property attributes from Ames, Iowa. The task is to predict final sale prices based on a variety of features ranging from architectural details to location-based variables.

In this analysis, we aim to answer the following key question: Given a set of engineered features describing a home, which modeling approach most effectively predicts the sale price? We are particularly interested in identifying which regression techniques generalize best to unseen data, and what modeling choices help address common challenges in housing datasets, such as skewness, heteroskedasticity, and the presence of outliers.

This question is important not only for its relevance to predictive modeling but also because housing markets play a central role in the economy. Improved predictive accuracy can aid in more equitable property appraisals, support data-driven urban planning, and inform lending decisions. Moreover, the problem exemplifies many real-world issues in applied machine learning, making it a valuable case study in model evaluation and selection.

1.1 Literature Review

To tackle this problem, we explore a variety of well studied modeling approaches. These include regularized linear models, robust regression techniques, and ensemble tree-based methods.

Regularized Linear Models: Lasso and Ridge regression, which introduces an ℓ_1 and ℓ_2 penalty respectively to the loss function, is known to regularize and mitigate overfitting in regression tasks. For instance, Seng and Khalid (2018) conducted a comparative study using the Ames Housing dataset, evaluating the performance of Ridge and Lasso regression models. They found that Lasso regression outperformed Ridge regression in terms of predictive accuracy, as measured by lower RMSE and higher R^2 values[5].

Robust Regression Techniques: To address the presence of outliers and heavy-tailed distributions in housing data, we considered the use of adaptive Huber regression, a method designed for robust estimation and inference. Unlike ordinary least squares, which is highly sensitive to outliers, Huber regression uses a loss function that is quadratic for small errors and linear for large errors, reducing the influence of outlier observations. Huber Regression is particularly effective when the data contains outliers[4].

Ensemble Tree-Based Methods: Ensemble methods, particularly Random Forest and XGBoost, have gained prominence due to their ability to capture complex nonlinear relationships and interactions among features. Li (2023) applied both Random Forest and XGBoost to the Ames Housing dataset, finding that while XGBoost outperformed Random Forest on lower prediction error, more research is needed to address the $R^2 < 0.9$, which is too low for practical use[1]. If we see similar R^2 values in our analysis, we would like to explore possible reasons to why explained variance is low. Similarly, Sharma et al. (2024) conducted a comprehensive comparison of machine learning models and concluded that while XGBoost provided the most accurate predictions for housing prices[6].

The application of these models in prior studies demonstrates the potential for these to perform well in our study. Their promising results—improved accuracy in regularized regression and ensemble tree-based methods—support our decision to include them in our own comparative analysis of housing price models.

2 Exploratory Data Analysis

Data Preparation, Preprocessing and Visualization

The dataset used in this analysis was sourced from a Kaggle Competition, which contains detailed information on residential home sales in Ames, Iowa[3]. It includes 1,460 observations and 79 variables, spanning a wide range of housing attributes such as structural characteristics, location features, and sale conditions. This dataset serves as the foundation for understanding relationships and patterns that may inform predictive modeling of house prices.

In this section, we conduct an exploratory data analysis (EDA) to understand the structure, patterns, and anomalies within the dataset. All code can be found in our GitHub repository[2].

2.1 Missing Value Analysis

We began our exploratory data analysis (EDA) by identifying columns with missing values. Categorical features with missing values often used NA to indicate the absence of a feature, such as no garage or no masonry veneer. We addressed this by replacing all NaN‘values in categorical columns with the string ”None”.

For numerical columns with missing values — GarageYrBlt, MasVnrArea, and LotFrontage — we visualized their distributions. These plots helped us decide on tailored strategies for each:

- **GarageYrBlt**: All missing values corresponded with GarageType = None. Imputing a year would not be appropriate as the garage was never built. Instead, we binned the values into broader age categories and assign a distinct None label for missing entries.
- **MasVnrArea**: All missing values corresponded with MasVnrType = None. Therefore, we set these missing areas to 0, indicating no masonry veneer.
- **LotFrontage**: As this feature depends on other predictors, we chose to impute it using a Random Forest regressor while excluding the target variable to prevent data leakage.
- **Other Numerical Values**: In the test dataset, there are a couple of other NA numerical values where the corresponding type value is None, representing the absence of a particular feature. We imputed these to 0.

2.2 Skewness

Next, we examined the skewness of all numerical features and the target variable (Sales Price). We found that many features along with the target variable exhibited strong right-skew, defined as skewness greater than 1. We plotted boxplots for these features to visually confirm their distributions, as seen in Figure 1.

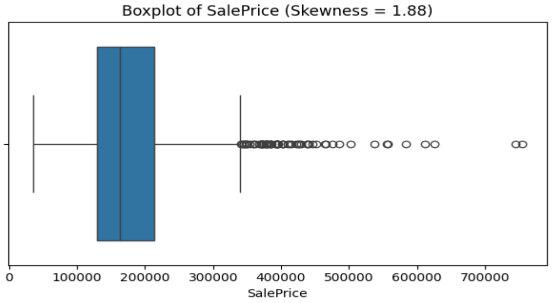

Figure 1: Boxplot before transformation.

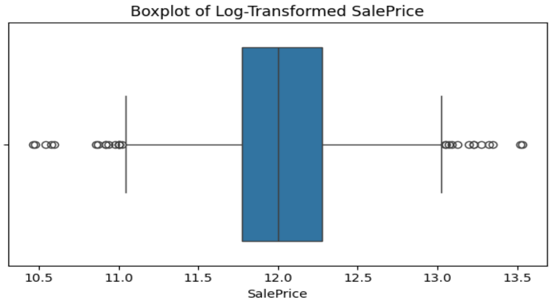

Figure 2: Boxplot after transformation.

To correct the skewness and normalize these features, we applied a `log1p` transformation. Post-transformation boxplots have a distribution that appeared more symmetric and bell-shaped, as seen in Figure 2. This suggests more stable behavior for predictive modeling, especially in linear regression models.

We also saw some zero-inflated features in Figure 3, which can hurt linear models. The dominance of zeros can overshadow the signal from the minority non-zero cases, leading to underestimated or over-penalized coefficients. While no transformation can remove these outliers, we still applied a `log1p` transformation to reduce the impact. We also considered removing observations that are outliers, but this is not an appropriate strategy as the Kaggle competition as the test dataset also includes outliers.

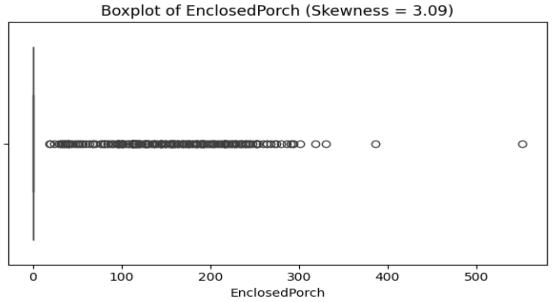

Figure 3: Boxplot of 0 inflated feature.

2.3 Correlation

To understand relationships between predictors and the target, we computed a Pearson correlation matrix and selected the five most correlated numerical features with SalePrice. As illustrated in Figure 4, there are strong intercorrelations among top features.

To inspect potential multicollinearity, we calculated Variance Inflation Factors (VIFs). This allowed us to identify redundant predictors that may require regularization or dimensionality reduction. Upon inspection of the computed values, no predictor reached a VIF of greater than 5, which means that multicollinearity will likely not pose a significant issue for our models and does not necessitate immediate corrective action such as feature elimination or dimensionality reduction.

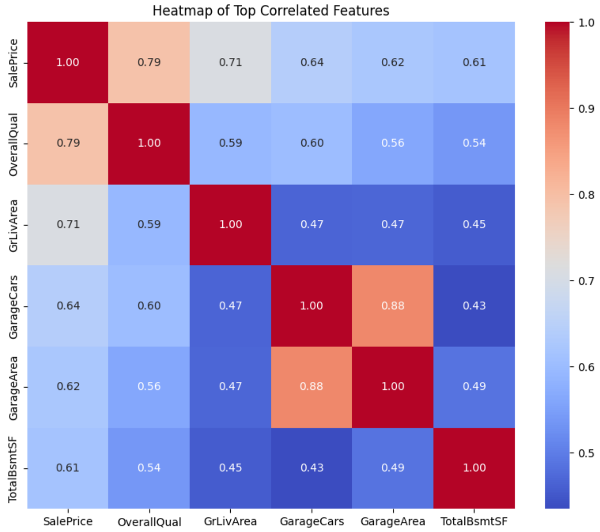

Figure 4: Heatmap between top correlated features and SalePrice.

2.4 Pairplots

Now we will go over some interesting pair plots that could affect the modeling decisions. In

Figure 5, we can see heteroscedasticity between overall quality and sale price. Variance in SalePrice increases as the predictor increases, indicating fanning patterns that might challenge linear models and favor more robust modeling approaches. Applying `log1p` transformation to the sale price helped reduce heteroscedasticity, as seen in Figure 6.

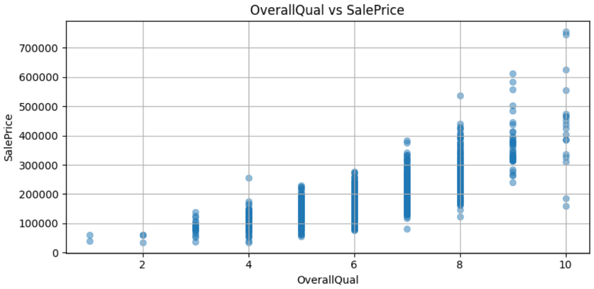

Figure 5: Heteroscedasticity.

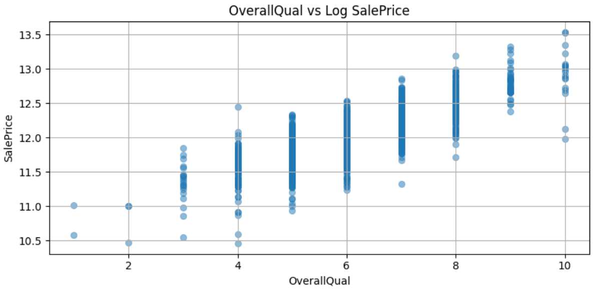

Figure 6: Variance improvement after transformation.

3 Methodology

Model Fitting and Cross-Validation

Lead in text and write-up goes here. Explain and outline our process for fitting models and what we plan to do in the subsequent sections. This section is divided into two columns.

3.1 Preprocessing Pipeline

We constructed a comprehensive preprocessing pipeline to clean and transform the data. To ensure fairness, this pipeline is consistent across all models. The pipeline includes the following steps:

- **Column Dropping:** The `Id` column, which serves as a unique identifier, was removed since it contains no predictive information.
- **Categorical Missing Value Imputation:** Missing values in categorical columns were filled with the string `"None"` to represent the absence of a category.
- **Numeric Missing Value Imputation:** For numerical features with missing values (excluding `LotFrontage` and `GarageYrBuilt`), we applied zero imputation as a neutral placeholder to preserve dataset structure without introducing artificial bias.
- **Garage Year Binning:** The `GarageYrBlt` column was transformed into bins based on year ranges to reduce sparsity and enhance interpretability. Also, this will allow us to categorize NA values as None.
- **Skewness Correction:** Numerical columns with high skewness (greater than 1.0), includ-

ing the target variable, were log-transformed using the `log1p` function to improve distributional symmetry.

- **Encoding and Scaling:** To prepare the features for model training, categorical variables were transformed via one-hot encoding to capture distinct levels, while numerical features were standardized using `StandardScaler` to ensure they contribute comparably to the model.
- **Noisy Column Removal:** To reduce the influence of extreme values, we applied an outlier removal step based on z-scores. Numerical columns with a high proportion of extreme values (z-score $>$ 4.0) were excluded from the dataset if more than 5% of values were deemed outliers. This step helps prevent skewed model behavior and improved generalization in our testing in our testing.
- **LotFrontage Imputation:** To address missing values in the `LotFrontage` column, we trained a `RandomForestRegressor` with the default hyperparameters using the remaining transformed features to predict and impute these values. We excluded Sales Price (the target variable) to prevent data leakage. As a sanity check, we also ran cross validation on this and compared it to the training accuracy. We verified the Log RMSE to be 0.0190 and $R^2 = 0.9629$ in 10 fold cross validation with shuffled splits on the training data. Training Log RMSE is 0.0064 and $R^2 = 0.9949$, which is within range of the cross validation results. These results suggests that this imputation strategy is robust to overfitting and excellent at generalization.

3.2 Cross Validation

To evaluate model performance and tune hyperparameters, we aplied 10-fold cross-validation (CV) with shuffled splits on the training data. This technique partitions the data into ten folds, iteratively training on nine while validating on the tenth, ensuring each observation is used for validation exactly once. Grid search was used in conjunction with cross-validation to identify the best hyperparameter configurations.

3.3 Model Evaluation

To ensure consistency across models, we used the CV log root mean squared error (RMSE) and R^2 as our primary evaluation metrics. To check for overfitting, we also compared the CV log RMSE with the training log RMSE. We will then test real world performance on an unseen test dataset, submit the predictions to Kaggle, and evaluate the Kaggle score—the Root-Mean-Squared-Error (RMSE) between the logarithm of the predicted value and the logarithm of the observed sales price.

3.4 Ridge Regression

We will first attempt Ridge Regression because it is more robust to outliers than normal linear regression. Ridge Regression is a linear model that includes an ℓ_2 penalty on the coefficients to prevent overfitting by shrinking large weights.

During hyperparameter tuning, we found the best regularization parameter $\alpha = 10.0$ across the values $\{0.1, 1.0, 10.0\}$. After training on the full dataset and evaluating our primary metrics, we evaluated residual patterns and constructed Q-Q plots to assess model assumptions. For linear regression models, these assumptions are linearity, homoskedasticity, independence of errors, and normality of residuals.

3.5 Lasso Regression

We also implemented Lasso Regression, a linear model that includes an ℓ_1 penalty on the coefficients. This penalty encourages sparsity, effectively performing feature selection by shrinking some coefficients exactly to zero. Lasso is especially useful when we expect many irrelevant features and want to improve model interpretability.

During hyperparameter tuning, we searched over a range of regularization strengths $\alpha \in \{0.0001, 0.001, 0.01, 0.1\}$. The best performing value found was $\alpha = 0.001$. After training the model on the full dataset using this parameter, we evaluated the primary metrics and analyzed residual plots to assess model assumptions and fit quality.

3.6 Huber Regression

As we saw outliers and model assumption violations in Ridge Regression model, we attempted using Huber Regression which is less sensitive to outliers. Huber Regression is a robust regression technique that uses a combination of squared and absolute error losses.

When tuning the hyperparameters, we found the best parameters $\alpha = 1 \times 10^{-5}$ and $\epsilon = 1.75$ over the values {1e-5, 1e-4, 1e-3} and {1.2, 1.35, 1.5, 1.75} respectively. After identifying the best configuration, we evaluated the primary metrics, analyzed residuals, and created Q-Q plots to assess normality.

3.7 Random Forest

Since the same problems occurred with Huber Regression—sensitivity to remaining outliers and heteroskedasticity—we transitioned to ensemble-based methods that are known to be more robust in such settings. Random Forest, in particular, is a non-parametric ensemble method that builds multiple decision trees using bootstrapped samples and aggregates their predictions. This approach reduces variance, improves generalization, and naturally handles nonlinearities and complex interactions among features. Moreover, because each tree is trained on a random subset of features, Random Forest is less prone to overfitting than individual decision trees, making it a suitable choice for capturing the complex relationships present in housing price data.

We found the best hyperparameters to be `max_depth = None`, `max_features = 0.5`, and `n_estimators = 100` over the search space of {50, 100, 200}, {10, 15, 20, None}, and {0.5, ”sqrt”} respectively. After selecting the best hyperparameters, the model was retrained on the full training set, evaluated on the primary metrics, and checked feature importance to understand the dominant drivers of sale price.

3.8 XGBoost

We selected XGBoost as another model to compare because of its effectiveness in structured tabular data and its ability to capture nonlinear relationships through gradient boosting. Unlike Random Forests, which average predictions from many de-correlated trees, XGBoost builds trees sequentially, where each new tree corrects the errors made by the previous ones. This boosting framework allows XGBoost to achieve strong predictive performance while incorporating regularization to reduce overfitting. Given its success in many machine learning competitions and benchmarks, we believed it was well-suited for modeling the complex patterns in housing price data.

The best model used 200 estimators, a maximum depth of 3, a learning rate of 0.1, a subsample of 0.8, a column sample by tree of 0.8, and a minimum child weight of 1. These were selected over the search space: `n_estimators` = {50, 100, 200}, `max_depth` = {3, 6, 10}, `learning_rate` = {0.01, 0.05, 0.1}, `subsample` = {0.6, 0.8, 1.0}, `colsample_bytree` = {0.6, 0.8, 1.0}, and `min_child_weight` = {1, 3, 5}. With these hyperparameters, we retrained the model and conducted a final evaluation. We also verified feature importance to identify the most influential features driving housing prices.

4 Results

Comparison of Various Methods

Table 1: Comparison of Model Performance

Model	Train LOG RMSE	CV LOG RMSE	CV R^2	Kaggle Score
Ridge Regression	0.10	0.13	0.8248	0.12324
Lasso Regression	0.11	0.13	0.8315	0.12456
Huber Regression	0.10	0.14	0.5841	0.14244
Random Forest	0.05	0.14	0.8658	0.14042
XGBoost	0.07	0.14	0.8318	0.12986

In this section, we present the results of our predictive modeling. We compare the performance of several regression models including Ridge Regression, Huber Regression, Random Forest, and XGBoost.

Model Performance Summary

Table 1 summarizes the key performance metrics across all models. The model that best generalized was selected based on the lowest cross-validated LOG RMSE, the smallest gap between Train LOG RMSE and CV LOG RMSE, and the lowest Kaggle Score combined. Ridge Regression had the lowest Kaggle Score, followed by Lasso Regression. Both of these models have a small gap between the CV LOG RMSE and Train LOG RMSE, indicating strong generalization and minimal overfitting. In contrast, Random Forest achieved the lowest training error (Train LOG RMSE = 0.05), but its much higher CV LOG RMSE (0.14) reveals a substantial gap, highlighting significant overfitting. XGBoost exhibited a similar pattern: while its training error was low (0.07), the increase to a CV LOG RMSE of 0.14 demonstrates a pronounced overfitting effect. These results are surprising as both of these tree models are generally robust against overfitting. This overfitting effect on our tree based models can explain both our low R^2 values, suggesting less explained variability, and higher Kaggle Scores, confirming worse generalization.

Analysis

While Ridge Regression and Lasso Regression demonstrated the best generalization performance among all models, it is important to note that both of these violated key model assumptions. As shown in Figure 7, the residuals display heteroskedasticity, particularly for higher predicted sale prices—suggesting that variance increases with the fitted values. Figure 8 further illustrates this issue, showing several outliers that deviate substantially from a normal distribution. We saw a similar residual plot with Lasso Regression. These patterns likely contributed to the lower R^2 value, indicating that the model did not explain as much variance in the target as might be expected for a good linear fit. Nonetheless, when disregarding these violations and outliers, the data exhibits a good linear fit, making Ridge Regression a reasonable choice under the assumption of approximate linearity. This could be a possible explanation to overfitting with the tree-based models, as discussed earlier, which can introduce unnecessary complexity.

We also examined the feature importance rankings from both tree-based models and found that many of the top predictors overlapped with those identified in the correlation heatmap during EDA. While this alignment supports the idea that the models are leveraging meaningful predictors, it does not fully rule out overfitting—particularly given the train/test performance gaps. A more detailed investigation would be needed to identify whether specific features are being overfit, which is often a subjective task requiring dedicated interpretability analysis beyond the scope of this study.

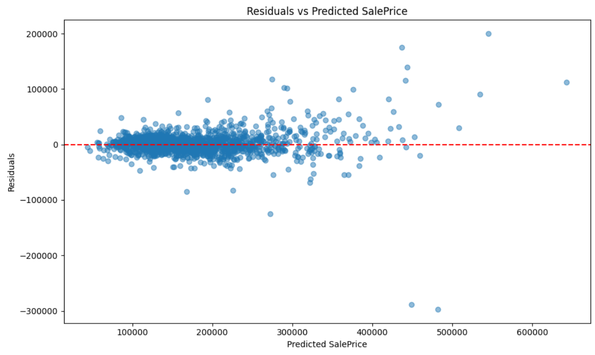

Figure 7: Residual plot of Ridge Regression showing heteroskedasticity and fanning.

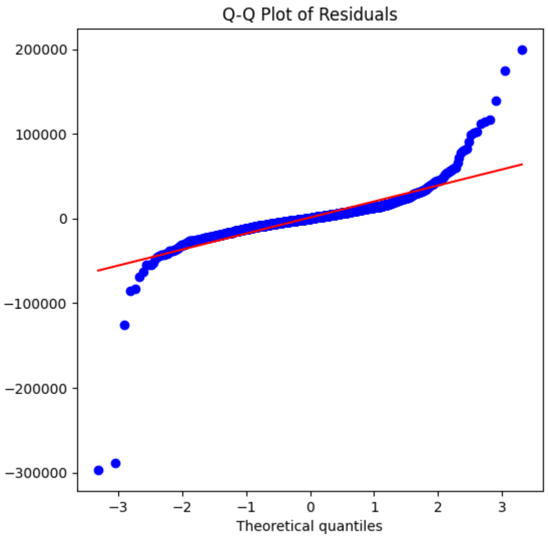

Figure 8: QQ plot shows that a few outliers deviate heavily from the normal distribution.

We tried retraining the Ridge Regression and Lasso Regression models without outliers from the training dataset. Outliers are defined as any points with standardized residuals greater than 3 in absolute value ($|r_i| > 3$). After retraining, we observed an improvement in cross-validation accuracy and more explained variability (much higher R^2) and more explained variability (much higher R^2) due to the more stable variance and reduced influence of extreme values. The residual plot showed no violations of model assumptions. Specifically, the residuals appeared randomly scattered around zero with a consistent spread across predicted values, indicating homoskedasticity and a lack of non-linearity or omitted variable bias. However, this came at the cost of decreased performance on the Kaggle test set. This suggests that while removing outliers can help a model better capture the dominant trends in the data, it may also hinder the model's ability to generalize to real-world data that includes such extreme values. Consequently, this tradeoff must be carefully considered in competitive or production settings where outliers are part of the evaluation set.

Another interesting note is the surprisingly poor performance of Huber Regression, which was introduced in an attempt to correct the model violations in Ridge Regression caused by outliers. Although Huber Regression is designed to be robust to outliers, it underperformed compared to both Ridge Regression and tree-based methods. One possible explanation is that because it is less influenced by large outliers during training, it might ignore meaningful variation in the target if it mistakes high-leverage points for noise, resulting in a lower R^2 value compared to the other linear regression models. Like the other linear models, Huber Regression also exhibited heteroskedasticity and fanning patterns in it's residuals despite utilizing ℓ_2 regularization.

5 Limitations

One key limitation of this analysis is the dataset's geographic specificity. The data is based solely on housing transactions from Ames, Iowa, which may not generalize to broader housing markets that exhibit different economic, demographic, or architectural characteristics. Consequently, model performance and feature importance may shift when applied to other regions.

Another limitation involves the computational resources available during hyperparameter tuning. Due to constraints in compute power, the grid search space for model selection was restricted. A more exhaustive search could potentially yield better-performing models. Notably, while Ridge Regression showed strong results in our analysis, ensemble methods like XGBoost have performed better on the public Kaggle leaderboard for this competition, suggesting that our final model choice may have been influenced by these computational trade-offs.

Additionally, while our exploratory data analysis helped identify potential outliers, determining the optimal strategy for handling them remains inherently uncertain. There is no universally correct method for detecting or removing outliers, and different approaches—such as using domain knowledge, statistical thresholds, or model-based residuals—can lead to varying results. This introduces subjectivity into the preprocessing pipeline, and it is possible that improved or alternative EDA strategies could enhance model performance or generalizability. The same can be said for determining which predictors the tree models are overfitting on.

6 Conclusion

In conclusion, our ridge linear model has the best predictive power in the Kaggle competition, providing the best estimates of Sales Prices based on various predictors and attributes of a home. However, due to assumption violations, we cannot recommend this model to predict house sale prices in a real world scenario. We also cannot recommend our tree based models as these show signs of overfitting, resulting in biased estimates in the real world.

In ideal circumstances, if we were to remove observations that are outliers and highly predicted sales values from both the training and test dataset, our Ridge Regression would likely be the best fit. Without model violations, the true relationship between predictors and sale price seems linear, which makes Ridge Regression a better fit than any of the tree models, as our tree models would be considered too complex and easily overfit. We tried a couple of strategies to handle these outliers, both of which failed to generalize on the real world test dataset, as it includes extremities.

A promising direction for future studies is to explore hybrid modeling strategies, where the dataset is partitioned based on sale price or predicted outlier status. In such an approach, a linear model like Ridge Regression could be applied to lower sale prices, where the relationship with predictors appears linear and homoskedastic, while a tree-based model could be used for higher-value homes, which tend to exhibit non-linear behavior and greater variance. This idea stems from our EDA observations that higher sales prices skew the overall linear relationship, suggesting that a single model may not be ideal across the full range of values. Although we did not implement or evaluate such a hybrid model in this study, it offers a potentially effective method for improving predictive performance and warrants further investigation.

References

[1] Han Li. House price prediction and analysis based on random forest and xgboost models. *Highlights in Business, Economics and Management*, 21, 2023.

[2] Random Logic. Sta160 project repository. https://github.com/random-logic/STA160, 2025. Accessed: 2025-06-05.

[3] Anna Montoya and DataCanary. House prices - advanced regression techniques. https://kaggle.com/competitions/house-prices-advanced-regression-techniques, 2016. Kaggle.

[4] scikit-learn developers. *sklearn.linear_model.HuberRegressor*. scikit-learn, 2025. Accessed: 2025-06-07.

[5] Jia Xin Seng and Kamil Khalid. Modelling house price using ridge regression and lasso regression. *International Journal of Engineering & Technology*, 7(4.30):498–501, 2018.

[6] Hemlata Sharma, Hitesh Harsora, and Bayode Ogunleye. An optimal house price prediction algorithm: Xgboost. *Analytics*, 3(1):30–45, 2024.