Predicting House Prices with Linear Regression and Gradient Boosting: A Comparative Study

Seungbae Son(43483449)
Master of Data Science
University of California, Irvine
seungbas@uci.edu

Yuan Liu(14883589)
Master of Data Science
University of California, Irvine
yuanl69@uci.edu

Ju Young Cha (61163075)
Master of Data Science
University of California, Irvine
juyc1@uci.edu

Abstract*—Predicting house prices accurately is a challenging task due to the complex and dynamic nature of the housing market. In recent years, machine learning techniques have been widely used for house price prediction. In this paper, we conduct a comparative study of Ridge, Lasso linear regression, and Gradient Boosting for house price prediction on a Kaggle competition dataset. We first perform exploratory data analysis and feature engineering to prepare the data. Then, we train and evaluate models using Ridge, Lasso linear regression, and Gradient Boosting with hyperparameter tuning. Our experimental results show that Gradient Boosting outperforms Ridge and Lasso linear regression in terms of predictive accuracy, with a root mean squared error (RMSE) of 0.0129. However, Lasso linear regression and Ridge provide interpretable feature importance scores, which can be useful for feature selection and model interpretation. Our study highlights the importance of choosing appropriate machine learning techniques for house price prediction, depending on the specific requirements of the application.

***Keywords*—House Price, Linear Regression, Ridge, Lasso, Gradient Boosting**

I. INTRODUCTION

The problem is to predict the sale price of houses in Ames, Iowa based on various features such as the size of the house, number of bedrooms and bathrooms, location, etc. The dataset used for this problem is the Ames Housing Dataset, which was compiled by Dean De Cock for use in data science education. It contains 80 explanatory variables and 1 response variable (the sale price) for 1,460 houses in Ames, Iowa between 2006 and 2010. The dataset is commonly used as a benchmark for regression modeling and feature engineering techniques so we are going to develop a model that accurately predicts the sale price of houses in Ames, Iowa based on the given set of features.

II. DATA EXPLORATION AND VISUALIZATION

The Ames Housing dataset consists of 80 explanatory variables describing almost every aspect of residential homes in Ames. Among 80 variables, there are 44 categorical variables and 35 numerical variables.

The target variable is the sale price of each house. Sales price has 1,460 values with mean $180,921.2 and std $79,442 and this indicates price has a relatively wide range in the dataset. As you can see at figure1, the distribution is skewed to the right, where the tail on the right side of the curve is longer than the tail on the left side, and the mean is larger than the mode. This situation is also called positive skewness. Having a skewed target will affect the overall performance of our machine learning model, so one way to alleviate the situation will be to use log transformation on the skewed target, in our case, the Selling Price to reduce the skewness of the distribution.

TABLE 1. Five-number summary of Sales Price (Y)

Min	25%	50%	75%	Max
34900	129975	163000	214000	755000

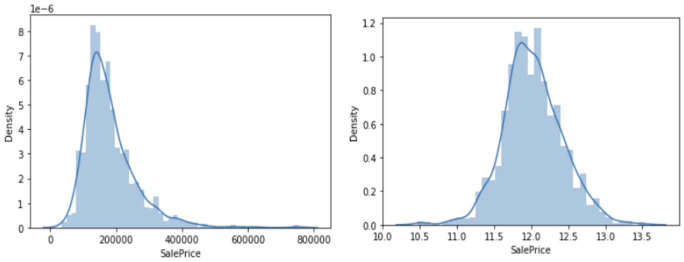

Figure1. Distribution of Sales Price (Y)

The heatmap shows the explanatory variables whose correlation between SalesPrice is over 0.5. The highest is OverallQual which stands for Overall material and finish quality. Other variables are YearBuilt, YearRemodAdd, TotalBsmtSF, and etc,This refers to the condition of the house may highly related to the price.

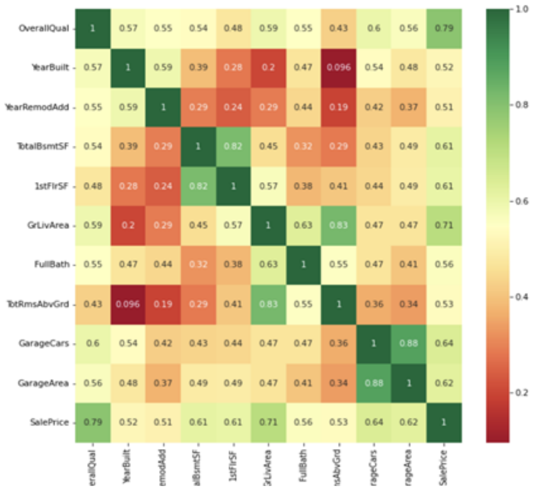

Figure2. Heatmap (correlation between salesprice(Y) and X)

Looking into discrete variables, these tend to be qualifications (Qual) or grading scales (Cond), or refer to the number of rooms, or units (FullBath, GarageCars), or indicate the area of the room (KitchenAbvGr). For most discrete numerical variables, we see an increase in the sale price, with the quality, or overall condition, or number of rooms, or surface such as shown in Figure 3. However, for some variables, we don't see this tendency. Most likely that variable is not a good predictor of sale price.

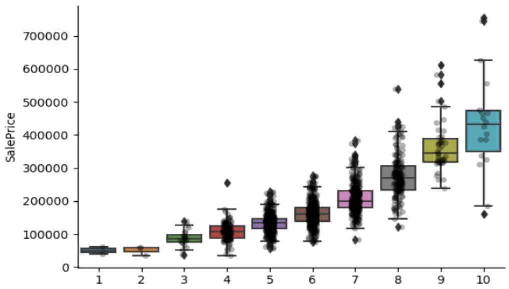

Figure 3. SalesPrice tendency by OverallQual

Some of the important features in the dataset include:

- OverallQual: overall material and finish quality of the house
- YearBuilt: original construction date
- TotalBsmtSF: total square feet of basement area
- GarageCars: size of garage in terms of car capacity
- Neighborhood: physical locations within Ames city limits

There are also many other features related to different aspects of the house such as the number of bedrooms and bathrooms, the type of heating and cooling system, and various other architectural features. We've also compared the shape of the train and test dataset and are able to see the similarities which may not lead to overfitting to the train data.

III. DATA PREPROCESSING

A. Outlier and Missing value

Missing data can be problematic for machine learning algorithms as they require complete data to train models effectively. Thus, we change the missing value to something else in order to fill in the gaps of missing information in our dataset. We have 19 missing variables, from those with categorical variables(16) and numerical variables(3). For example, for the 'PoolQC' feature, missing values (99.52%) are replaced with the string "None" to indicate that the property does not have a pool. For numerical features like 'LotFrontage', missing values are replaced with the median value of the feature in the corresponding neighborhood, which can help to preserve the distribution of the data. For categorical features like 'Electrical', missing values are replaced with the mode of the feature, while for features like 'GarageType' and 'GarageFinish', missing values are replaced with the string "None" to indicate that the property does not have a garage. Overall, filling in missing values with appropriate values can help to improve the quality of our data and make it more suitable for machine learning algorithms to use.

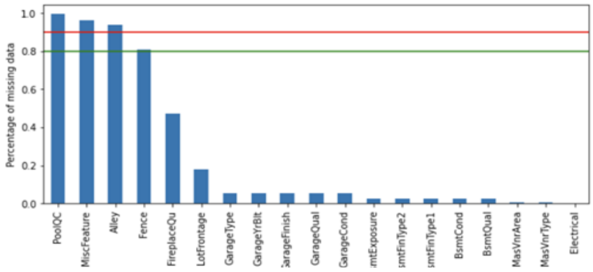

Figure 5. Percentage of Missing values

B. Transformation

The transformation process which converts certain numerical features into categorical features, and then apply label encoding on all the categorical features were conducted. By converting these variables into categorical variables, we can potentially improve the accuracy and interpretability of our model.

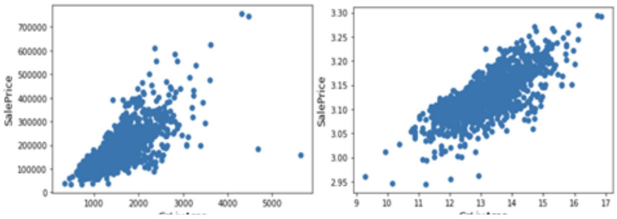

Figure 6. Log Transformation

C. Aggregation

We need to work on if there are any features that can be added up and make a better interpretation. The total square footage feature is important because it gives a better indication of the overall size of a house. Rather than considering the square footage of individual floors or the basement separately, this feature takes into account the combined living space of all floors. This can be a more accurate representation of a home's size and value.

D. Skewness

Additionally, handling skewed features is important because many machine learning algorithms, such as linear regression, assume that the data is normally distributed. However, in real-world datasets, some features can have skewed distributions, which can negatively impact the performance of the model. Therefore, applying techniques such as the box-cox transformation can help to normalize the data and improve the accuracy of the model. Overall, handling skewness of a distribution helps to improve the performance of machine learning models by reducing the impact of skewed data.

IV. LINEAR REGRESSION - BASE MODEL

A. Model Explanation

Simple Linear regression, lasso, and ridge regression models were employed as the base models. These models differ in how they handle overfitting and feature selection in predictive analysis.

- **Linear regression**: It is a basic statistical method for modeling the relationship between a dependent variable and one or more independent variables.
- **Lasso regression** : Lasso is a regularization technique that adds a penalty term to the linear regression objective function, which helps in feature selection by shrinking the coefficients of less important features to zero
- **Ridge regression**: Ridge is also a regularization technique that adds a penalty term to the linear regression objective function, but instead of shrinking the coefficients to zero, it shrinks them towards zero

Lasso and Ridge regression improve predictive power by addressing overfitting and feature selection. Lasso is great for feature selection, while Ridge helps with reducing multicollinearity and stabilizing coefficients.

B. Performance validation

Cross-validation is utilized for performance validation by employing various values of K in the model. This allows for the comparison of the performance of regression models.

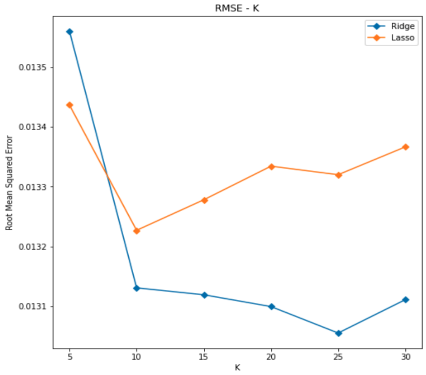

Figure 7. The RMSE against K

The graph above illustrates the outcome of cross-validation with respect to different K values. In the case of lasso regression, the lowest RMSE was achieved at K = 10. In contrast, for Ridge regression, as the K value increases, the model's performance improves until K = 25, beyond which the RMSE starts to increase.

C. Adaption to under- and over-fitting.

In an effort to mitigate the issue of under- and over-fitting in the base models, hyperparameter tuning was implemented. Specifically, GridSearchCV, a hyperparameter tuning technique, available in Scikit-learn, was used to explore the model's parameter space.

D. Hyperparameters in Lasso and Ridge regression

- **alpha**: Constant that multiplies the regularization term, controlling the strength of the penalty applied to the absolute values of the model's coefficients.

- **max_iter**: The maximum number of iterations should perform before converging to a solution.

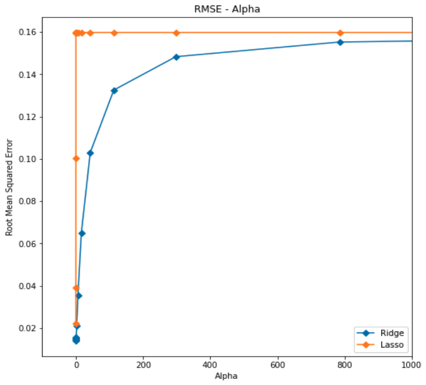

Figure 8. The RMSE against Alpha

The graph illustrates the impact of hyperparameter alpha on the performance of Lasso and Ridge regression models. As alpha increases, the Root mean squared error (RMSE) also increases, indicating a decrease in model performance. Overall, Ridge regression outperforms Lasso regression in cross-validation evaluation when alpha is less than 1000.

For hyperparameter tuning, we used Scikit-learn's GridSearchCV. GridSearchCV is a valuable tool that streamlines the process of testing various hyperparameters for a given estimator and selects the best combination of hyperparameters that yield the highest model performance. The evaluation metric used here was RMSE.

TABLE 2. Base Models RMSE Comparison

Data	Linear	Lasso	Ridge
Train	0.1241	0.1133	0.1140
Test	0.3339	0.1245	0.1230

The above results demonstrate that Ridge and Lasso exhibited similar performance. While Lasso slightly outperformed Ridge on the training set, Ridge performed better on the test set. However, for both the training and test datasets, the simple linear regression model had the poorest performance out of the three models.

V. Gradient Boosting – Advanced Model

A. Model Explanation

The Root mean Gradient boosting involves the creation of a strong predictive model by combining a series of weak models, often decision trees, in a sequential manner. During training, the algorithm seeks to minimize the difference between the actual target values and the predictions of the weak model by fitting the residual errors of the previous model with a new model. This process continues until a satisfactory level of accuracy is achieved. One of the main advantages of gradient boosting is its ability to handle heterogeneous data types and outliers effectively. It can also handle large datasets with high dimensionality and has a natural regularization mechanism, which helps to prevent overfitting. In our dataset, there are over 200 features, so Gradient boosting can be a good choice. However, it may be computationally expensive and require careful tuning of hyperparameters to achieve optimal performance.

B. Performance validation

Parameterize the model. There are few important parameters and their default values which control how the forest behaves during the training process.

- Learning rate = 0.1
- N_estimators = 100
- Max_depth = 3

We firstly explored how these parameters affect the model.

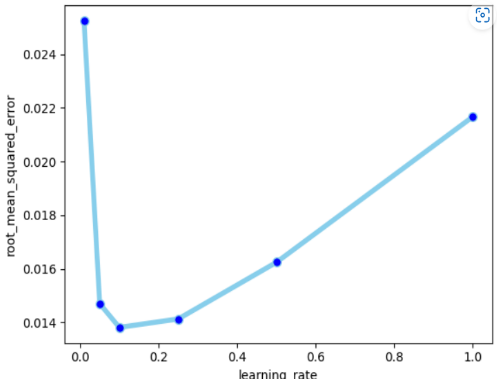

Figure 9. The RMSE against learning rate

Learning rate controls the magnitude each tree contributes to the final results. If it is small then the whole learning process would be slow, and the model would become robust. However, a small learning rate would require more decision trees in the bag, which may cause model overfitting. According to the experience result, we chose the learning rate to be 0.05.

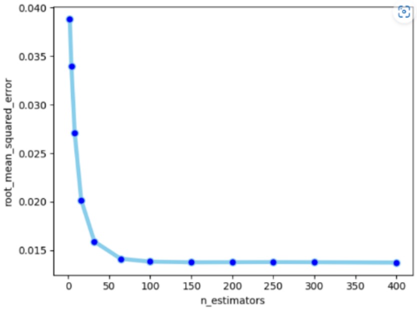

Figure 10. The RMSE against N_estimators

N_estimators control how many sequential trees we have in our bag to train and predict the results. We can clearly see that the RMSE goes down quickly when the number of N_estimators is increased to near 50. And it stays nearly the same when we continuously increase the trees. This result suggests that when the number of trees reaches 100, adding more trees would not help the overall model that much.

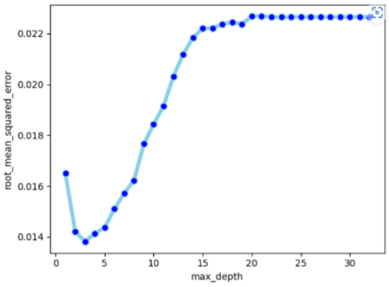

Figure 11. The RMSE against max_depth

The max depth denotes the max depth each individual tree can reach. If it is small, each tree may not have enough prediction power. On the other hand, higher depth trees could also overfit the model, which leads to worse RMSE

After fitting the model to the data, we calculate the feature importances. And based on that, we notice there are few features such as TotalSF which contribute to a large part of prediction power. So this leads us to try to reduce the features we used, which can be done by setting a max features number the model can use. We enforce that the model can pick only 20 features.

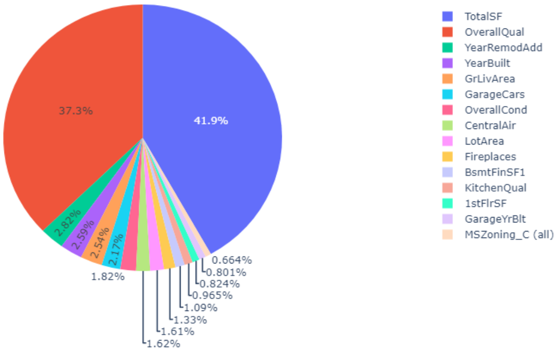

Figure 12. Feature Importances

After all, our parameters become as follows.

- Learning_rate = 0.05
- N_estimators = 400
- Max_depth = 3
- Max_features = 20

TABLE 3. Parameter Setting

	Default	Selected
RMSE	0.0138	0.0129
R_square	0.95	0.96
Mean_fit_time	0.48	0.25

RMSE and Mean_fit_time are the result of k-fold cross validation with k = 5. The result tells us the selected parameters help us reduce the model complexity and improve the accuracy of the model at the same time. And the R_square is approximately 0.96, which indicates that the model is robust which can explain most variance of the Sale prices.

VI. CONCLUSION

The significant difference between Ridge regression and Gradient Boosting models in terms of their performance metrics is reflected in their root mean squared error (RMSE) values. The Gradient Boosting model outperforms the Ridge regression model with an RMSE of 0.0129, compared to 0.1230. This is due to several factors. Gradient Boosting excels in modeling non-linear relationships and can handle outliers more effectively than Ridge regression by employing a robust loss function. Moreover, Gradient Boosting can automatically select important features, making it ideal for high-dimensional data. Lastly, Gradient Boosting is an ensemble learning technique that combines multiple models to produce more accurate predictions and improves the generalization ability of the model. Overall, we can conclude that Gradient Boosting is a powerful and popular choice for regression problems.

REFERENCES

[1] *G. (2023a, March 14). Sales Price Prediction - Complete Workflow. Kaggle.https://www.kaggle.com/code/gustavofelici/sales-price-prediction-complete-workflow#*

[2] *Data-Processing House Prices - Advanced Regression Techniques | Kaggle. (n.d.). https://www.kaggle.com/competitions/house-prices-advanced-regression-techniques/data*

[3] *V. (2023b, March 20). Comprehensive Exploratory Data Analysis. Kaggle. https://www.kaggle.com/code/validmodel/comprehensive-exploratory-data-analysis/notebook#%F0%9F%A7%AA-Target*

[4] *https://www.kaggle.com/code/gustavofelici/sales-price-prediction-complete-workflow#Data-Processing*

[5] *Tune Hyperparameters with GridSearchCV . https://www.analyticsvidhya.com/blog/2021/06/tune-hyperparameters-with-gridsearchcv/*