Sale Price Prediction of the Residential Houses in Ames, Iowa

Pratiti Baral, Hana Le

CSC 370: Data Mining

DePauw University
Greencastle, IN 46135, U.S.A.

pratitibaral_2024@depauw.edu
quynhLe_2023@depauw.edu

Abstract

*T*his paper describes various approaches to predicting the sale price of residential houses in Ames, Iowa using machine learning techniques. Real estate companies use machine learning to improve the efficiency and accuracy of many real estate processes and provide better estimates of the sale price of houses in the area in a number of ways. The main objective of this project is to assess different preprocessing strategies, sophisticated data mining algorithms and models, and hyperparameterization approaches to produce the most accurate estimation of the sale price of the houses.
Keywords: attributes, algorithms, models, accuracy

1 Data Description

*I*n this study, we use the Kaggle dataset that contains a range of explanatory variables describing almost every aspect of residential homes and the selling prices of residential houses in Ames, Iowa. Dean De Cock compiled the Ames Housing dataset for data science education. In total, there are 79 attributes for each house, divided into 41 categorical and 38 numerical values. There are 1460 examples in the training set and 1459 examples in the testing set. We use both categorical and numerical data and convert some categorical values to ordinal values to explain and predict the final sale price of unidentified houses. In this project, we go beyond the information provided to analyze housing domains comprehensively. We believe doing so will be beneficial to have a greater understanding of the dataset. Furthermore, after carefully studying the dataset, we proceed to data cleaning to identify the target variables and other irrelevant variables and address any potential issues.

2 Preprocessing

*W*e identify several issues in the dataset, such as missing values, inconsistencies, and outliers. To address the issues, we employ a variety of data cleaning approaches, including handling missing values, converting non-numeric values to numeric values, normalizing the numeric attributes, using get_dummies, and creating new attributes that encompass several other variables in the dataset.

*I*n order to build a neat predictor, we decide to synthesize and categorize certain areas and characteristics of the house, making certain attributes fall under certain categories. The five categories we come up with are: location, square footage, age traits, quality, and rooms. The bundle for each category is described below.

- location = ['Neighborhood', 'Condition1', 'Condition2']
- squareFoot = ['TotalArea', 'BsmtFinSF1', 'GrLivArea', 'LowQualFinSF','TotalBsmtSF', 'LotArea', '1stFlrSF', '2ndFlrSF']
- ageTraits = ['YearOld', 'YearBuilt']
- quality = ['ExterQual', 'ExterCond', 'OverallQual', 'OverallCond' 'FireplaceQu','GarageQual', 'KitchenQual', 'HeatingQC']
- rooms = ['TotRmsAbvGrd', 'NumberOfBathrooms', 'KitchenAbvGr', 'BedroomAbvGr']

2.1 Handling Missing Values

*I*n both the training and testing sets, we begin by handling the missing numeric and categorical (string) values. At first, we attempt to fill in the missing numbers using the mean and the mode, but this approach decreases the accuracy. As a result, we examine the dataset more closely and learn that for the majority of the attributes that have N/A values, the attributes are not present in the house. For numeric attributes, we therefore fill any missing value with 0 to increase accuracy. As for string attributes, if an attribute has fewer than ten missing values, we fill them in with mode; if there are ten or more missing e, we fill them in with "None" to turn the attribute into a string rather than a NA. By filling in these gaps, the data becomes more comprehensive and has no gaps, allowing us to utilize the data more effectively.

2.2 Attribute Engineering

*W*e create new, more informative attributes by combining existing attributes in our dataset, which is a useful technique in attribute engineering. Creating a new attribute that is the sum of multiple other attributes helps capture more complex relationships between the data as well as give a more thorough and comprehensive picture of the data. Most importantly, it reduces the dataset's dimensionality and increases the readability and accuracy of the model.

*T*hus, to have a better sense of how big the houses are, we create a "TotalArea" attribute to encompass the size of each house. The 'TotalArea' attribute is a combination of superficial attributes of the house, including 'GarageArea', 'TotalBsmtSF', '1stFlrSF', '2ndFlrSF', and 'GrLivArea.' We perform this on both training and testing sets, just like with the previous attributes. Furthermore, since bathrooms in each house are divided into the number of full bathrooms and the number of half bathrooms, we think that using these qualities as a single unified property will be more accurate in future computations and predictions. Therefore, to account for all bathrooms in each residence, we create a new attribute named "NumberOfBathrooms." Each complete bathroom would be equal to 1, while a half bathroom would be 0.5, so each row in 'FullBath' and 'BsmtFullBath' attributes are multiplied by 1, and each row in 'HalfBath' and 'BsmtHalfBath' attributes are multiplied by 0.5. Then we add these together to get the total number of bathrooms in the house. We do this on both the training and testing sets.

2.3 One Hot Encoding

*O*n several attributes, such as "Neighborhood", "Street", and "RoofMatl," that have categorical values, we use one-hot encoding to transform these values into a numerical format. One hot encoding method involves transforming categorical values into a numerical format that can be given to Machine Learning algorithms to help them perform better predictions. Categorical values cannot be used directly with most machine learning algorithms because they are based on mathematical operations and require all input and output variables to be numeric. One hot encoding involves creating a separate binary attribute for each value in the categorical attribute. For instance, if the original attribute "Street" has two values ("grvl" and "pave"), we create two new binary attributes: "is_grvl" and "is_pave." Depending on whether the original value was "grvl" or "pave" each of these new variables would have a value of either 0 or 1. This enables categorical data to be numerically represented and used by machine learning algorithms.

2.4 Normalization

*T*o make the data more uniform and potentially generate better prediction accuracy, we normalize some of the numerical attributes, including ‘squareFoot’, ’MSSubClass’, ’LotFrontage’, ’MasVnrArea’, ’MoSold’, ’OpenPorchSF’, ’EnclosedPorch’, ’WoodDeckSF’, and ’BsmtUnfSF.’ Normalization scales attribute values so that they fall in a smaller range, such as -1.0 to 1.0 or 0.0 to 1.0, which improves the performance and convergence of the algorithms. The impact of outliers and other extreme data values can also be lessened with the use of normalization, which helps strengthen the model’s resilience and avoid overfitting.

2.5 Changing Non-Numeric Values to Numeric Values

*W*e also change nominal values to ascending ordinal values based on ranking. We do this by creating the ordinalConversion helper function, where we convert each row with values such as “Po’, ‘Fa’, ‘TA’, ‘Gd’ and ‘Ex’ to 0,1, 2, 3, and 4, respectively and fill ‘NA’ and missing values with the mode. Then we call the ordinalConversion helper function on attributes, such as ’BsmtCond’, ’BsmtExposure’, ’ExterQual’, ’ExterCond’, ’BsmtQual’, ’HeatingQC’, ’KitchenQual’, ’FireplaceQu’, ’GarageQual’, ’PoolQC,’ which have ranked categorical values to convert them to ordinal values based on ranking. Comparatively, ordinal data has a natural order, but nominal data is categorized without a natural order or rank. Additionally, numerical or quantitative data will always be a number that can be measured and used with regression algorithms.

*T*o determine the age of the houses, we construct a ”YearOld” attribute and subtract the ”YearBuilt” attribute in the dataset from the current year (2022). Additionally, we create a ”yrRemodToBool” function to convert the ”YearRemodAdd” attribute into a binary value indicating whether or not the house has been remodeled. By comparing the ”YearRemodAdd” attribute to the ”YearBuilt” attribute, the function can determine whether the house has been renovated or not. If the values of the two attributes are the same, it means the house has not been remodeled, so the value is set to 0. However, if the values are different, it means the house has been remodeled, so the value is set to 1. Then we rename the “YearRemodAdd” attribute to “IsItRemodeled,” which has two values, 0 (signifying a NO) and 1 (signifying a YES), for clarity.

*I*n preprocessing the ”RoofStyle” attribute, we researched and considered the cost of different roof styles and used the cheapest one (Gable) as a reference point. We assigned a value of 1 to Gable and multiplied the values of other roof styles based on their relative cost compared to Gable.

- Gable: \$1 to \$2 per square foot → Average = 1.5 → Base = 1
- Flat: \$2.5 to \$9 per square foot → Average = 5.75 → Base Avg = 3.83
- Gambrel: \$8 to \$15 per square foot → Average = 11.5 → Base Avg = 7.6
- Hip: \$8 to \$12 per square foot of roofing → Average = 10 → Base Avg = 6.66
- Mansard: \$8 to \$20 per square foot → Average = 14 → Base Avg = 9.33
- Shed: \$3.75 per square foot → Base AVG = 2.5

*A*fter converting categorical values of ”RoofStyle” attribute to ordinal values, and tuning the algorithms, the accuracy increases significantly. We replicate the process on ”houseStyle” attribute and obtain an increased accuracy. The ”RoofMaterial” property, however, exhibits a modest drop in accuracy when the same technique is used. This may be because most houses have the same type of roof material, so preprocessing the attribute in the same manner as we do for roof style has no positive effect. As a result, we opted to ignore the attribute due to its irrelevance.

3 Algorithms

*T*o predict the price of the houses, different models, including Linear Regression, Linear Ridge, Gradient Boosting, and Lasso Regression, are examined to determine which model would produce the optimum result.

3.1 Linear Regression

*L*inear Regression is an algorithm that provides the relationship between two variables, the independent variable and the dependent variable, to predict the outcome of an event. The independent variable remains unchanged despite the change of other variables. Meanwhile, as the independent variable fluctuates, the dependent variable also changes. The regression model predicts the value of that dependent variable, which is also known as the outcome being analyzed.

3.2 Linear Ridge Regression

*L*inear Ridge Regression is the linear regression extension in which the loss function is adjusted to minimize the model's complexity. Ridge works on hyperparameter optimization under the presumption that the data set was fairly small. It includes a penalty term or learning, called alpha, which indicates how big of a step we want to take. This term helps to create a simpler model. The result obtained from using Linear Ridge, however, is just slightly better than the result obtained from Linear Regression.

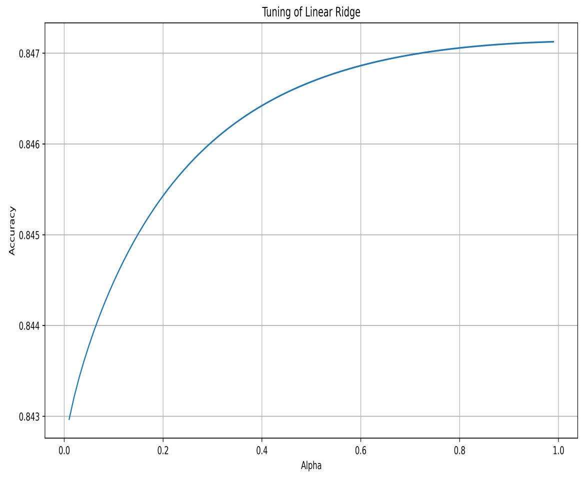

Figure 1: Relative Accuracies Corresponding to Alpha

3.3 Lasso regression

*L*asso Regression, or Least Absolute Shrinkage and Selection Operator, is another modification of linear regression. By limiting the sum of the absolute values of the model coefficients, the loss function in Lasso Regression is altered to reduce the model's complexity. Lasso regression penalizes less important features of your dataset by reducing their coefficients to zero and, thereby, deleting them.

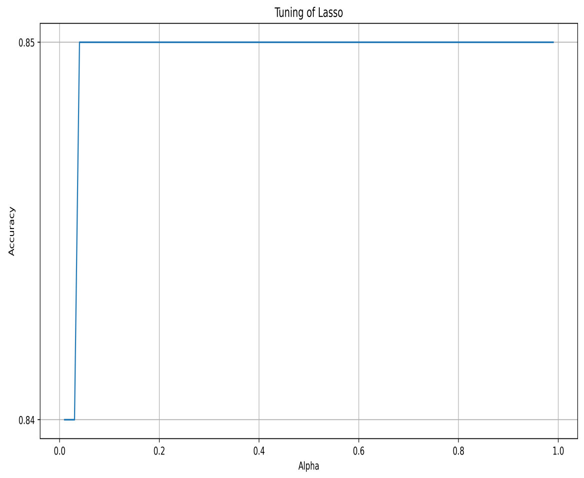

Figure 2: Relative Accuracies Corresponding to Alpha

3.4 Gradient Boosting

*G*radient Boosting is a machine learning technique that gives a prediction model in the form of an ensemble of weaker prediction models, typically known as decision trees. Because it generates the result based on the combination of other models, it can converge better than Linear Regression, Ridge and Lasso Regression. Applying the models to our dataset also shows that Gradient Boosting gives significantly better results than the two others. However, this powerful ability may lead to overfitting. Therefore, the learning rate was critical in developing a model that does not overfit the data.

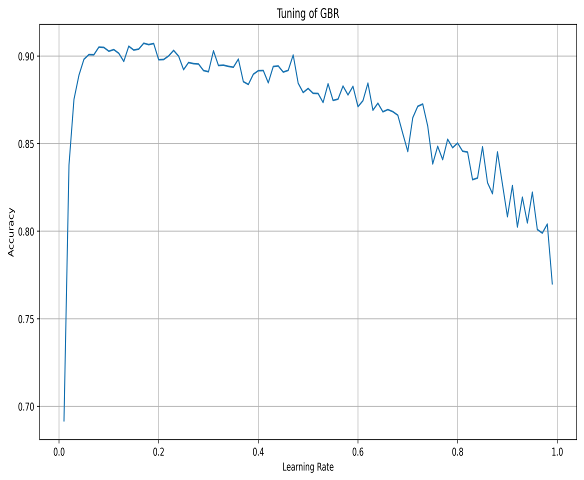

Figure 3: Relative Accuracies Corresponding to Learning Rate

4 Results

*P*redictions become more accurate as we fine-tune the models. After tuning, the performance of all models improves, with slightly better results for Ridge Regression and Gradient Boosting compared to Lasso Regression. In general, Gradient Boosting algorithm outperforms the other two models in the Kaggle challenges both before and after tuning. This demonstrates that Boosting is a powerful strategy because it employs multiple models to improve on prior models' faults, which provides an advantage over regression models. Although it requires more processing effort to run, hyper-tuning is essential to generate the most accurate results possible.

4.1 Before Tuning

Linear CV Average Score: 0.8426484345656672

Ridge CV Average Score: 0.845012593661918

GBR CV Average Score: 0.9059154820569821

Lasso CV Average Score: 0.8453768612542956

4.2 After Tuning

Table 1: CV Score Comparison of Different Models

Model	Tuning Parameter	Accuracy
Linear Rigde	Alpha: 0.99	0.847
Lasso	Alpha: 0.99	0.845
Gradient Boosting	Learning Rate: 0.19	0.909

5 Conclusion

*I*t is critical that we understand the domain of data in the first place for this project. After that, pre-processing is the most important stage in this project. It helps make sense of the data and makes the dataset looks cleaner in general so that it is easier to generate models and calculate accurate predictions. Filling in the gap of missing data seems to create a huge impact because it makes the data itself more complete. That way, when given the following predictors related to the location, size, age traits, quality of the home, and the rooms in the house, the algorithms have enough information to generate better results. Values that were strongly related to pricing were critical in the efficient production of data.

In terms of model, Lasso gives the best result among the Regression models, while Gradient Boosting outperforms all the Regression models. So the highest accuracy is achieved through the use of Gradient Boosting as the prediction model, especially after being tuned.

References

*A*ggarwal, Tavish. "Lasso and Ridge Detailed Explanation." Teach Leader,https://www.techladder.in/article/lasso-and-ridge-regression-detailed-explanation. Accessed 15 Dec. 2022.

Gaurav. "An Introduction to Gradient Boosting Decision Trees." Machine Learning +, 12 Jun. 2021, https://www.machinelearningplus.com/machine-learning/an-introduction-to-gradient-boosting-decision-trees/. Accessed 15 Dec. 2022.

Jain, Shubham. “A comprehensive beginners guide for Linear, Ridge and Lasso Regression in Python and R.” Analytics Vidhya, 22 Jun. 2017, edited 18 Oct. 2020, https://www.analyticsvidhya.com/blog/author/shubham-jain/. Accessed 15 Dec. 2022.

“Linear Regression.” Yale University Courses, http://www.stat.yale.edu/Courses/1997-98/101/linreg.htm. Accessed 15 Dec. 2022.

Moorthi, Aparna. “How Lasso Regression Works in Machine Learning.” Dataaspirant, 26 Nov. 2020, https://dataaspirant.com/lasso-regression/. Accessed 15 Dec. 2022.

Vasudev, Rakshit. “What is one Hot Encoding? Why and WHen You Have to Use it?” 12 August, 2017. https://hackernoon.com/what-is-one-hot-encoding-why-and-when-do-you-have-to-use-it-e3c6186d008f. Accessed 15 Dec. 2022.