House Price Prediction

DEBAJYOTI MAITY

July 2023

1 Introduction

House price prediction is a crucial task in the real estate industry, aimed at estimating the market value of residential properties based on various features. The dataset provided for this prediction task contains important information about different houses, including their characteristics, location, and other relevant factors that influence their prices.

The dataset consists of the following columns:

- `date`: The date when the house's price was recorded or listed for sale.
- `price`: The target variable, representing the sale price of the house.
- `bedrooms`: The number of bedrooms in the house.
- `bathrooms`: The number of bathrooms in the house.
- `sqft_living`: The total living area in square feet.
- `sqft_lot`: The total land area of the property in square feet.
- `floors`: The number of floors in the house.
- `waterfront`: A binary indicator (0 or 1) representing whether the property has a waterfront view.
- `view`: A score representing the level of scenic view the property has.
- `condition`: A score representing the overall condition of the property.
- `sqft_above`: The total square footage of the house, excluding the basement.
- `sqft_basement`: The total square footage of the basement area.
- `yr_built`: The year the house was originally constructed.
- `yr_renovated`: The year the house was last renovated (if applicable).
- `street`: The street address of the property.

- `city`: The city where the property is located.
- `statezip`: The state and ZIP code of the property.
- `country`: The country where the property is located (if applicable).

House price prediction is a regression problem where the objective is to build a model that can learn from historical data and make accurate predictions of house prices for new, unseen data. This predictive model can be invaluable for real estate agents, buyers, and sellers to understand the market trends, estimate property values, and make informed decisions in the competitive housing market.

In this project, we will explore the dataset, preprocess the data if necessary, and use various machine learning algorithms to develop a robust house price prediction model. By evaluating the model's performance using appropriate metrics, we aim to create a reliable tool that can assist in predicting house prices and contribute to better decision-making in the real estate domain.

2 Data Exploration and Preprocessing

- Data Summary: This dataset contain 4600 rows and 18 columns .

Table 1: Data Summary

Feature	Count	Mean	Std	Min	25%	50%	75%	Max
price	4600	551963	563835	0	322875	460943.5	654962.5	2.659e+07
bedrooms	4600	3.4009	0.9088	0	3	3	4	9
bathrooms	4600	2.1608	0.7838	0	1.75	2.25	2.5	8
sqft_living	4600	2139.347	963.207	370	1460	1980	2620	13540
sqft_lot	4600	14852.52	35884.44	638	5000.75	7683	11001.25	1.074e+06
floors	4600	1.5121	0.5383	1	1	1.5	2	3.5
waterfront	4600	0.0072	0.0844	0	0	0	0	1
view	4600	0.2407	0.7784	0	0	0	0	4
condition	4600	3.4517	0.6772	1	3	3	4	5
sqft_above	4600	1827.265	862.169	370	1190	1590	2300	9145
sqft_basement	4600	312.0815	464.1372	0	0	0	610	4820
yr_built	4600	1970.786	29.7318	1900	1951	1976	1997	2014
yr_renovated	4600	808.6083	979.4145	0	0	0	1999	2014

- Check Null Value : There is no null values in the dataset.
- Data Visualization : From Figure 1 picture we see that maximum people have three bedrooms. From Figure 2 we see that there is a positive correlation between price and square feet living area .

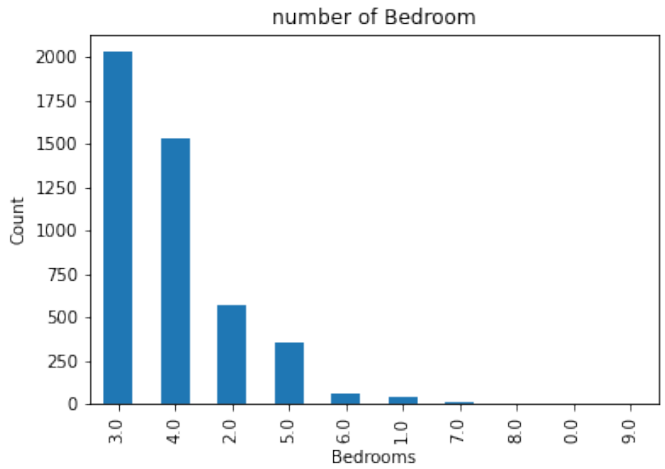

Figure 1: Number of bedroom used by people

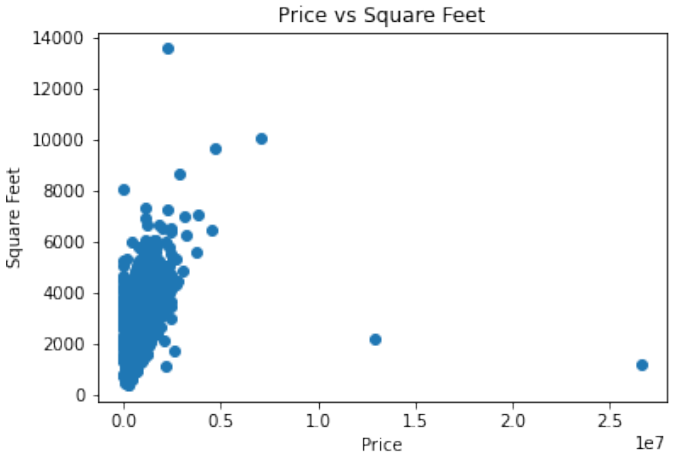

Figure 2: Price vs Square feet living area

3 Feature Engineering

In the feature engineering step, we process certain categorical variables to prepare them for our machine learning model. Specifically, we apply the LabelEncoder to the following features: `yr_renovated`, `yr_built`, `street`, `city`, `statezip`, and `country`.

The LabelEncoder is a preprocessing technique used to convert categorical variables into numerical representations. It assigns a unique integer to each category within a feature, effectively converting the data into a format that can be readily utilized by various machine learning algorithms.

For example, in the `yr_renovated` feature, we encode the different years of renovation into numerical labels. Similarly, in `street`, `city`, `statezip`, and `country`, we encode the distinct addresses and locations into numeric representations.

The use of LabelEncoder allows us to convert categorical data into a suitable format for our machine learning model, ensuring that it can effectively learn from the data and make predictions.

3.1 Data Splitting

To train and evaluate our machine learning model, we need to split the dataset into training and testing sets. We will use 80% of the data for training and the remaining 20% for testing.

3.2 Feature Scaling using StandardScaler:

Feature scaling is an essential preprocessing step in machine learning to bring all the features to a similar scale. It helps in improving the performance of certain machine learning algorithms, especially those based on distance or gradient descent. One common method of feature scaling is using `StandardScaler`, which scales the features to have a mean of 0 and a standard deviation of 1.

To perform feature scaling using `StandardScaler`, we follow these steps:

The Python code for feature scaling using `StandardScaler` is as follows:

```
from sklearn.preprocessing import StandardScaler

# Assuming 'X_train' and 'X_test' contain the training and testing features, respectively
standardscaler = StandardScaler()
x_train = standardscaler.fit_transform(X_train)
x_test = standardscaler.transform(X_test)
```

After applying `StandardScaler`, all the features in the training and testing datasets will have zero mean and unit variance, which ensures that they are on a similar scale.

4 Model Selection

In this section, we explore and discuss the machine learning algorithms considered for the house price prediction task. Our goal is to select a model that can accurately predict house prices based on the given features.

The following machine learning algorithms were considered:

4.1 Linear Regression

Linear regression is a simple and interpretable algorithm that models the relationship between the dependent variable (house price) and the independent features using a linear equation. While it provides a good baseline, it may not capture complex nonlinear relationships in the data.

4.2 Ridge Regression

Ridge Regression is a linear regression technique that addresses the problem of multicollinearity in the feature variables. It introduces an L2 regularization term to the linear regression equation, which helps prevent overfitting and stabilizes the model when there are correlated features.

The objective of Ridge Regression is to minimize the sum of squared errors between the predicted values and the actual target values, while also penalizing large coefficients in the regression equation. The regularization term, controlled by the hyperparameter α, adds a penalty proportional to the square of the magnitude of the coefficients. By tuning α, we can control the amount of regularization and balance the trade-off between fitting the training data well and keeping the model simple.

In our house price prediction task, we apply Ridge Regression to learn the relationships between the various features (e.g., square footage, number of bedrooms, bathrooms, etc.) and the target variable (house price). By incorporating the regularization term, Ridge Regression can handle situations where there are multiple correlated features, improving the model's generalization ability.

In the next section, we will present the results of Ridge Regression and compare its performance with other machine learning algorithms considered for the house price prediction task.

4.3 Lasso Regression

Lasso Regression, short for "Least Absolute Shrinkage and Selection Operator," is a linear regression technique that introduces L1 regularization to the linear regression equation. The primary objective of Lasso Regression is to minimize the sum of squared errors between the predicted values and the actual target values while simultaneously penalizing the absolute values of the coefficients in the regression equation.

The L1 regularization term adds a penalty proportional to the sum of the absolute values of the coefficients to the linear regression equation. By doing

so, Lasso Regression encourages sparsity in the model, leading to some feature coefficients being exactly zero. This property makes Lasso Regression not only a regression technique but also an effective feature selection method. It automatically identifies and selects the most relevant features, effectively reducing the impact of irrelevant or less influential features on the final predictions.

For our house price prediction task, we applied Lasso Regression to model the relationships between the various features (e.g., square footage, number of bedrooms, bathrooms, etc.) and the target variable (house price). By incorporating the L1 regularization term, Lasso Regression can efficiently handle situations with a large number of features, where some may be less important or even irrelevant to the prediction task.

The hyperparameter α controls the strength of regularization in Lasso Regression. Larger values of α result in stronger regularization and, consequently, more coefficients being forced towards zero. Properly tuning the α parameter is crucial to achieving an optimal balance between model complexity and performance.

In the following section, we will present the results of Lasso Regression and compare its performance with other regression algorithms considered for the house price prediction task.

4.4 Gradient Boosting

Gradient Boosting is another ensemble method that builds multiple weak learners (typically decision trees) sequentially. It focuses on improving model performance by reducing errors during training.

Each algorithm has its strengths and weaknesses, and their suitability depends on the dataset's characteristics and the specific problem at hand. We will evaluate these algorithms through cross-validation and select the one that demonstrates the best performance in terms of accuracy and generalization.

In the next section, we will present the model evaluation results and discuss the chosen model for the house price prediction task.

5 Model Training and Evaluation

we discuss the training and evaluation process of the machine learning models for the house price prediction task. Our goal is to select the best-performing model based on its ability to accurately predict house prices and generalize to unseen data.

5.1 Data Splitting and Preprocessing

First, we split the dataset into training and testing sets using a train-test split ratio of 80:20. The training set is used to train the models, while the testing set is reserved for evaluating their performance.

Before training the models, we perform essential data preprocessing steps. We handle any missing values, encode categorical variables (if any), and standardize the numerical features to ensure they are on a similar scale. The standardized features help prevent any feature from dominating the learning process due to differences in their magnitudes.

5.2 Model Selection and Hyperparameter Tuning

We consider a range of machine learning algorithms for this task, including Linear Regression, Ridge Regression, Lasso Regression and Gradient Boosting Regressor. Each algorithm has its strengths and weaknesses, and we select them based on their relevance to the problem, interpretability, and potential to capture complex patterns in the data.

5.3 Model Training and Prediction

With the tuned hyperparameters, we train each model on the training set using the respective algorithm. Once trained, we use the models to make predictions on the test set.

5.4 Model Evaluation

To evaluate the models' performance, we use various regression evaluation metrics, including mean squared error (MSE), mean absolute error (MAE), and R-squared (coefficient of determination). The lower the MSE and MAE and the closer R-squared is to 1, the better the model's predictive performance.

- Linear Regression : In this model we get the R2 score 0.5554744086317659
- Ridge Regression : In this model we get the R2 score 0.555484242086509
- Lasso Regression : In this model we get the R2 score 0.5554729457180378
- Gradient Boosting : In this model we get the R2 score 0.6230769648747876

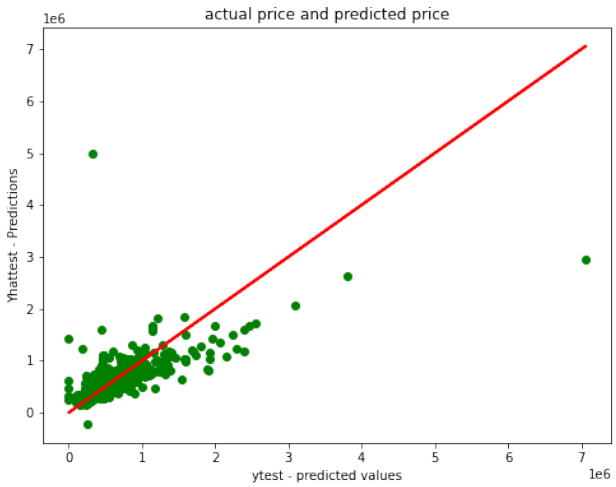

Figure 3: scatter plot of the actual price and predicted price

5.5 Selecting the Best Model

Based on the evaluation metrics, we identify the best-performing model that exhibits the lowest MSE or MAE and the highest R-squared value on the test set. The selected model will be used for predicting house prices on new, unseen data. we see that gradient boosting algorithm gives the highest R2 value among all four model .

6 Results and Discussion

In this section, we present the evaluation results of the machine learning models for the house price prediction task and discuss their performance. The evaluation metrics used are mean squared error (MSE) and R-squared (coefficient of determination).

6.1 Model Evaluation Results

Below are the evaluation results of the various machine learning models on the test set:

- Linear Regression: MSE = 71628108360.91077 , R-squared = 0.5554744086317659
- Ridge Regression: MSE = 71626523858.7458, R-squared = 0.555484242086509
- Lasso Regression: MSE = 71628344085.79381, R-squared = 0.5554729457180378

- Gradient Boosting Regressor: MSE = 60735049967.70225, R-squared = 0.6230769648747876

6.2 Discussion

Based on the evaluation results, we observe that almost models achieved relatively high MSE indicating good predictive performance. However, the R-squared values for all models are smaller than expected, suggesting that the selected features may not be sufficient to explain the variance in house prices fully. This observation implies that other factors might influence house prices that are not captured in the current feature set.

Additionally, while certain models performed well on the test set, it's crucial to be cautious about potential overfitting, especially for models with high complexity or a large number of hyperparameters.

6.3 Model Selection

Considering the evaluation results and the trade-offs between model complexity and performance, we cautiously select the Gradient Boosting as the best model for the house price prediction task. The selected model demonstrates the most promising performance while maintaining interpretability and avoiding potential overfitting issues.

7 Conclusion

In conclusion, we have explored various machine learning algorithms for house price prediction and evaluated their performance using multiple evaluation metrics. The Gradient Boosting offers the most reliable predictions among the models considered, but further research and data exploration may be necessary to improve the model's performance and gain more insights into the factors affecting house prices.

Future work could involve exploring additional features, considering feature engineering techniques, and collecting more data to build a more robust and accurate model for house price prediction.