Introduction

Our dataset contains 47,116 rows and 29 columns. It includes comprehensive information related to the California real estate listings, including details about properties, their locations, pricing, and other relevant attributes. The data appears to be sourced from a platform from Redfin. Below is a description of some important columns in this dataset:

- **PRICE**: The sale price of the property, indicating its monetary value.
- **PROPERTY TYPE**: The type of property being listed, such as "Single Family," "Condo," or other property types.
- **ZIP OR POSTAL CODE**: The postal code or ZIP code associated with the property's location
- **HOA/MONTH**: Monthly homeowners association (HOA) fees associated with the property, if applicable.
- **SQUARE FEET**: The total square footage of the property, indicating its size.

This dataset is valuable for real estate market analysis, property valuation, and understanding market trends. This data is helpful for researchers, real estate professionals, and analysts to gain insights into property sales and market dynamics.

Data Preparation

We applied two steps while doing data preparation. The first step is data cleaning, and the second step is defining features and targets. We performed data cleaning by first checking whether there existed any duplicate rows. Later, we used the IQR method to find out the outliers in the dataset. These outliers, if left unaddressed, can distort the statistical properties of the dataset and adversely impact the performance of any modeling. We then tried to drop the outliers using the same method. During the data manipulation, we decided to further narrow down 'SALE TYPE' because we realized that over ninety-five percent of sales are from 'MLS Listing'. By streamlining our dataset in this manner serves the purpose of enhancing our future data training performance.

In the second phase of data cleaning, we wanted to apply correlation on the whole dataset to define features. "SQUARE FEET", "BEDS" and "HOA/MONTH" have high correlation numbers above average. However, we also found out that there are a lot of null numbers in "HOA/MONTH." Keeping this column for the training model may cause overfitting. So we decided to drop it. As a result, we defined the dataset's features which are 'BEDS', 'BATHS', ‘CITY’, ‘SQUARE FEET’, ‘YEAR BUILD’, ‘DAYS ON MARKET’, ‘PROPERTY TYPE’, ‘LOT SIZE’. After we define the features data, we want to check if there are NaN values in there; if so we will drop them. Last, we set the dataset's target as ‘PRICE’ because our goal is to study the correlations between different features with California housing prices. Since we dropped null values in the features, in order to make the price column’s length the same as the features, we used the features indexes as a reference to narrow down.

Data Visualization

In order to help us better understand the whole dataset, we decided to do some visualization for this whole dataset.

- Count of House by Price

  This is a histogram graph about counting houses by price. We can see the range of the house price is very large; however, most of the houses are between 0 - 1 Million.

- Count of Houses by Square Feet

  This is a histogram graph about counting houses by square feet. Most of the houses are between 1000 - 2000 square feet. And there’s still a large number of houses over 2000 square feet.

- Price and Square Feet

  This is a box plot that represents the relationship between square feet and price. The larger the house is, the higher the price is.

- Price and Beds

  This is a box plot that represents the relationship between bedrooms and price. House price increases when there are more bedrooms. However, there’s a large number of houses with 2-3 bedrooms that are very expensive.

- Price and Year Built

  This is a box plot that represents the relationship between the year built and price. It’s very easy to find out that all the houses are very expensive no matter which year the houses were built.

- Price and Baths

This is a box plot that represents the relationship between bathrooms and price. House price increases when a house contains more bathrooms. However, there's a large number of houses with 3 bathrooms that are very expensive.

- Price and HOA/Month

  This is a box plot that represents the relationship between HOA/Month and price. It's not hard to find out that HOA is higher when the house price increases. And there's still a large amount of houses that have very little HOA/MONTH.

- Price and Lot Size

  This is a scatter plot that represents the relationship between lot size and house price. From the graph we can see that most of the houses are within 50000, however, there are houses that are very pricy within that lot size.

<u>Machine Learning Models</u>

Linear regression: We use the linear regression model as our baseline model. Firstly, we used get_dummies to convert categorical variables into dummy variables. Secondly, we apply the linear regression model to our data. The R Squared Score is about 0.74, which is better than our expectation.

Logistic regression: We first split feature columns into two types: the first type contains all numerical values, and the other type contains categorical columns such as 'PROPERTY TYPE'. After we created ColumnTransformer, we utilized SimpleImputer and as well as OneHotEncoder for the categorical variables so that we can convert them to numerical values. However, the accuracy of the logistical regression model is relatively low. The R Squared Score is only approximately 0.446. It shows this model is not a good fit for our dataset. We suspect that the reason causing a low accuracy might be that the categorical features are more complex. Therefore, we explore further using other machine learning models, such as gradient boosting.

Gradient Boosting regression: After experimenting with logistic regression, we decided to try gradient boosting regression. It represents the ensemble algorithm that builds decision trees sequentially. We hope it can help us capture the patterns in the dataset. However, it performs worse than our baseline model. The R2 score only reached 0.72. Therefore, we turned to another variant of linear regression: the lasso regression model.

Lasso regression: After running the basic lasso model with the most common parameters, the R2 score performs the same as our basic linear regression model. We suspect that tuning the parameters can help with improving the model performance. Therefore, we use the grid search method and hope that we can find better results by changing the lasso alpha and max iterations. Unfortunately, the performance did not significantly improve.

Neural network: We keep exploring what are some other types of models that we learned in the course that can be applied in our analysis, and we decided to experiment with neural network since it can capture complex relationships between data. We tried many different architectures, and eventually we reached a 0.799 R2 score. What surprised us is that adding more complicated layers to the architecture doesn't always improve the R2 score. The most complicated architecture we used only reached a 0.798 R2 score, which did not outperform a simpler architecture.

Conclusion

One of the key takeaways from this project is that we learned the importance of distinguishing between numerical features and categorical features. At first, we included zip codes in our selected features because we believe that zip codes can provide a more detailed idea about how location affects housing prices. However, later, we found that our model takes the zipcode column as a numerical feature. We had two options either transform the zipcode column to a categorical feature or use the city column instead. After our experiment, using the city column makes more improvement of the model performance. The second takeaway we learned is that we need to find a balance between correlation and null values. From our finding in the heatmap, column 'HOA/MONTH' has a higher correlation value to the price column compared to other columns. However, when we run code to check for null values, this column has 34777 null values, which means 73.8% of this column is empty. We decided not to put this column into our selected features since the sample size will be much smaller after dropping null values. The last takeaway is that even though tuning the parameters can be very time-consuming, it is vital in improving the model's performance. No parameters or model can suit all datasets.