Housing Price Prediction Data

Background:

This project is centered around a dataset that focuses on housing prices. The data has been sourced from a CSV file and includes several key features that are instrumental in determining housing prices.

Objective:

The main objective of this project is to develop a predictive model that can accurately estimate housing prices based on the features mentioned. This model aims to be a valuable tool for stakeholders in the real estate market, including buyers, sellers, and investors, by providing reliable price estimations.

Master Data Source (1):

The primary dataset for this project is obtained from Kaggle, specifically the "Housing Price Prediction Data" provided by Muhammad Bin Imran . This dataset offers a rich collection of housing-related features and serves as the foundation for our predictive modeling efforts.

Compare accuracy with other developers:

To evaluate the effectiveness of our predictive model, we compare its performance against other models developed by Kaggle users.

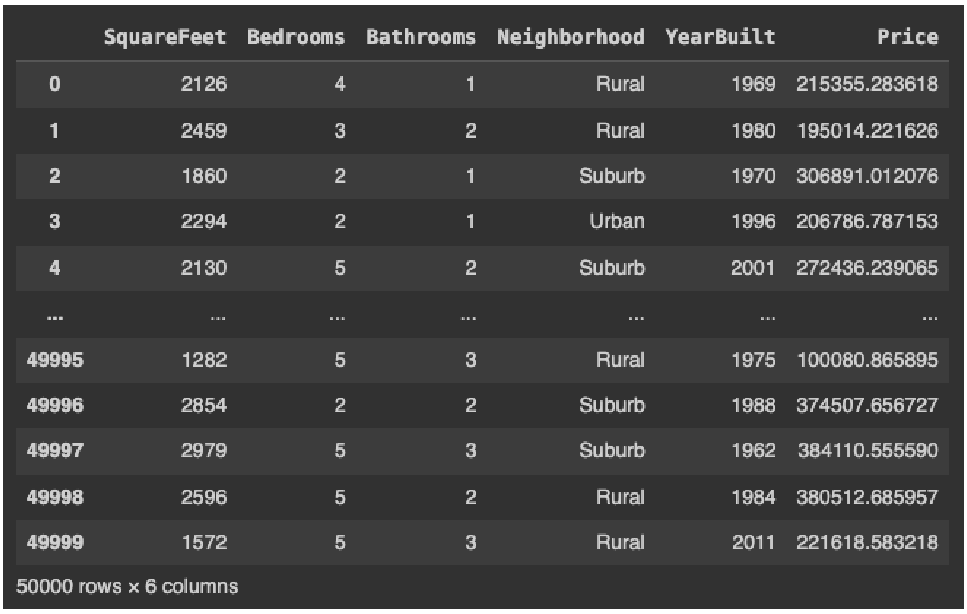
Model by Guan Lin Tao (2):

This model uses ML and Optuna for EDA in housing price prediction. It provides insights into optimization and accuracy in a similar context.

Model by K1RSN7 (3):

Another approach to housing price prediction, offering a different perspective on model construction and accuracy.

Understanding Dataset:



	SquareFeet	Bedrooms	Bathrooms	Neighborhood	YearBuilt	Price
0	2126	4	1	Rural	1969	215355.283618
1	2459	3	2	Rural	1980	195014.221626
2	1860	2	1	Suburb	1970	306891.012076
3	2294	2	1	Urban	1996	206786.787153
4	2130	5	2	Suburb	2001	272436.239065
...	...	...	...	...	...	...
49995	1282	5	3	Rural	1975	100080.865895
49996	2854	2	2	Suburb	1988	374507.656727
49997	2979	5	3	Suburb	1962	384110.555590
49998	2596	5	2	Rural	1984	380512.685957
49999	1572	5	3	Rural	2011	221618.583218

50000 rows × 6 columns

1) Master file CSV

The housing price dataset is composed of 50,000 entries, each representing individual properties with various attributes. The dataset includes the following columns:

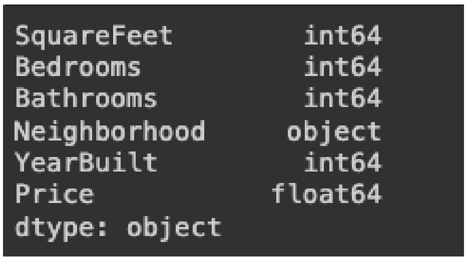
- **SquareFeet:** Indicating the total area of the house in square feet.
- **Bedrooms:** Showing the number of bedrooms in the house.
- **Bathrooms:** Representing the number of bathrooms in the house.
- **Neighborhood: S**pecifying the area where the house is located, with values like 'Rural', 'Suburb', and 'Urban'.
- **YearBuilt:** Denoting the year the house was built.
- **Price:** The buying price of the house, which is the target variable for our predictive model.

A brief examination of the first few rows gives us a glimpse into the variety of houses in the dataset, ranging in size, number of bedrooms and bathrooms, year of construction, and neighborhood category, along with their corresponding prices. For example, the first entry is a rural house with 2,126 square feet, 4 bedrooms, 1 bathroom, built in 1969, and priced at approximately $215,355.

This dataset provides a comprehensive foundation for building a predictive model to estimate housing prices, with a good mix of categorical and numerical features that can potentially influence housing valuation.

Data Types Verification Prior to Model Training:

Before proceeding with the training of our predictive model, a crucial step is to verify the data types of the features within our housing price dataset. Ensuring data types are accurate is fundamental for several reasons:



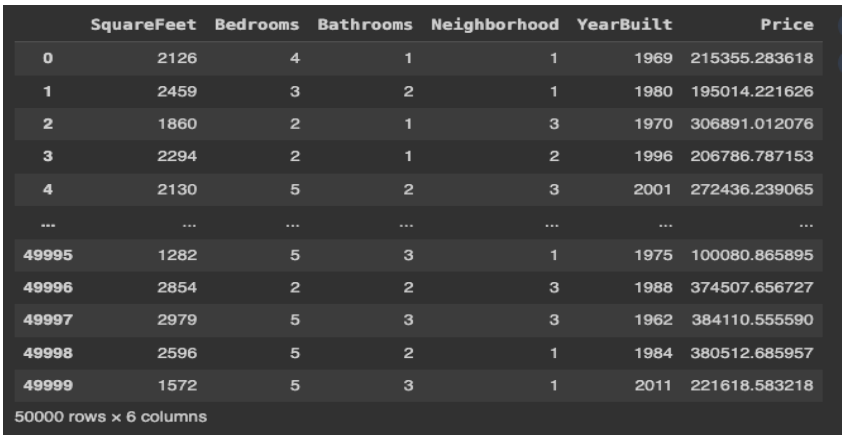
```
SquareFeet        int64
Bedrooms          int64
Bathrooms         int64
Neighborhood     object
YearBuilt         int64
Price           float64
dtype: object
```

2) Data types

Confirming these data types is essential for model compatibility and computational efficiency. For instance, numerical features like **SquareFeet**, **Bedrooms**, **Bathrooms**, and **YearBuilt** can be directly fed into most machine learning models, while the **Neighborhood** feature may require encoding to numerical values. The target variable **Price** must be a float to accommodate the wide range of real estate values and the precision needed for a regression model.

Data Type Transformation for 'Neighborhood' Feature:

To facilitate the use of machine learning algorithms that require numerical input, we have transformed the 'Neighborhood' feature from categorical to numerical values. This process, often referred to as encoding, is critical for preparing non-numeric data for model training. Here's how we've mapped the original categories to integers:



	SquareFeet	Bedrooms	Bathrooms	Neighborhood	YearBuilt	Price
0	2126	4	1	1	1969	215355.283618
1	2459	3	2	1	1980	195014.221626
2	1860	2	1	3	1970	306891.012076
3	2294	2	1	2	1996	206786.787153
4	2130	5	2	3	2001	272436.239065
...	...	...	...	...	...	...
49995	1282	5	3	1	1975	100080.865895
49996	2854	2	2	3	1988	374507.656727
49997	2979	5	3	3	1962	384110.555590
49998	2596	5	2	1	1984	380512.685957
49999	1572	5	3	1	2011	221618.583218

50000 rows × 6 columns

3) Data types of replacement Neighborhood 'Rural': 1, 'Urban': 2, 'Suburb': 3

'Rural': Assigned a value of 1 to represent rural neighborhoods.

'Urban': Assigned a value of 2 to signify urban neighborhoods.

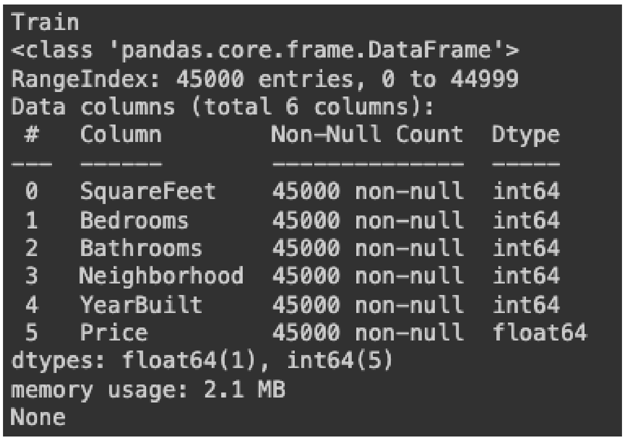
'Suburb': Assigned a value of 3 to denote suburban neighborhoods.

Exploratory Data Analysis:

As part of our exploratory data analysis (EDA), we have split the dataset into two distinct sets: training and testing. This is a critical step in preparing the data for the development and validation of our predictive model.

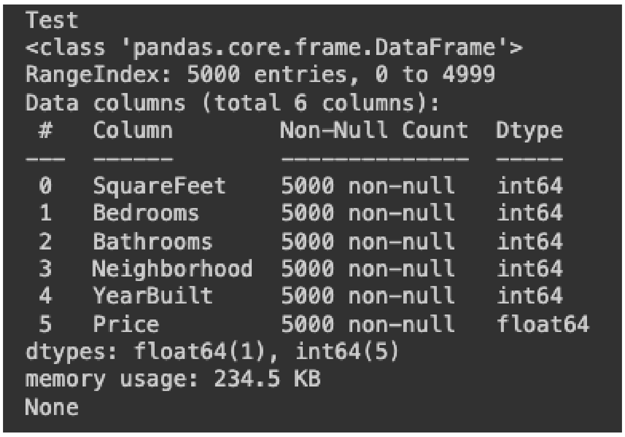
Data Splitting:

- test_size: 0.1
- Random_state: 42



```
Train
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 45000 entries, 0 to 44999
Data columns (total 6 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   SquareFeet    45000 non-null  int64
 1   Bedrooms      45000 non-null  int64
 2   Bathrooms     45000 non-null  int64
 3   Neighborhood  45000 non-null  int64
 4   YearBuilt     45000 non-null  int64
 5   Price         45000 non-null  float64
dtypes: float64(1), int64(5)
memory usage: 2.1 MB
None
```

4.1) EDA: Train

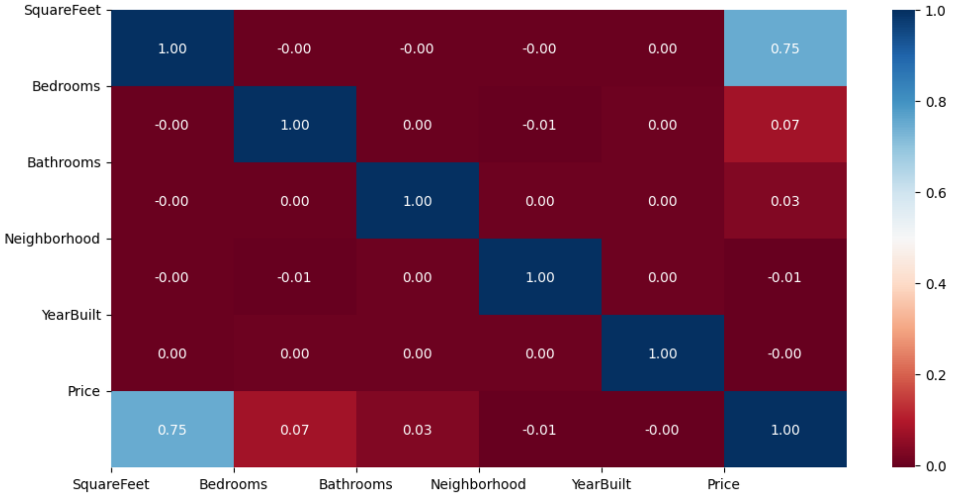


```
Test
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5000 entries, 0 to 4999
Data columns (total 6 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   SquareFeet    5000 non-null   int64
 1   Bedrooms      5000 non-null   int64
 2   Bathrooms     5000 non-null   int64
 3   Neighborhood  5000 non-null   int64
 4   YearBuilt     5000 non-null   int64
 5   Price         5000 non-null   float64
dtypes: float64(1), int64(5)
memory usage: 234.5 KB
None
```

4.2) EDA: Test

Analysis of Feature Correlations:

Our exploratory data analysis includes a correlation heatmap, which provides valuable insights into the relationships between the different features in our dataset. Here's what we can discern from the image:

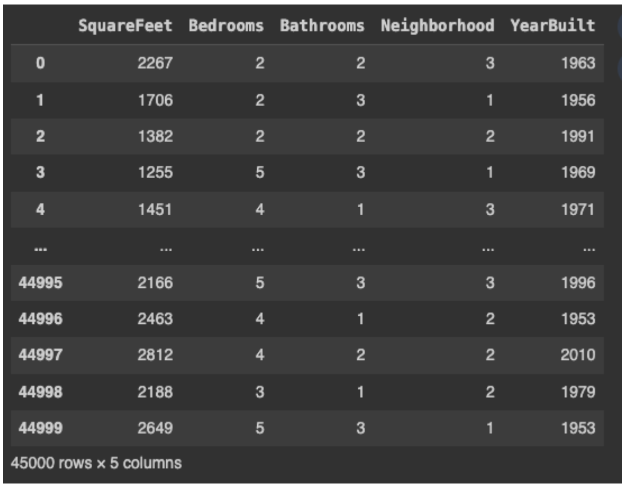


5) Heatmap

- **SquareFeet and Price:** There is a moderate positive correlation (0.75) between the size of the house (in square feet) and its price. This suggests that as the area of the house increases, the price is likely to increase as well, which aligns with market expectations.
- **Bedrooms, Bathrooms, and YearBuilt with Price:** The number of bedrooms and bathrooms shows a weak positive correlation with the house price (0.07 and 0.03, respectively), suggesting these features have a slight influence on the price. The year the house was built also has a very weak positive correlation with price (0.07), indicating that newer houses may fetch slightly higher prices, though the effect is minimal.
- **Neighborhood and Price:** The 'Neighborhood' feature, after encoding, shows negligible correlation with price (-0.01), indicating that the way we have numerically encoded this feature may not capture the true influence of location on house prices.
- **Inter-feature Correlations:** Among the features themselves, most correlations are close to zero, implying no strong linear relationships between them. Notably, 'Bedrooms' and 'Bathrooms' have no correlation with 'YearBuilt', suggesting that the number of bedrooms or bathrooms in a house is independent of the year it was built.

Preparation for machine learning:

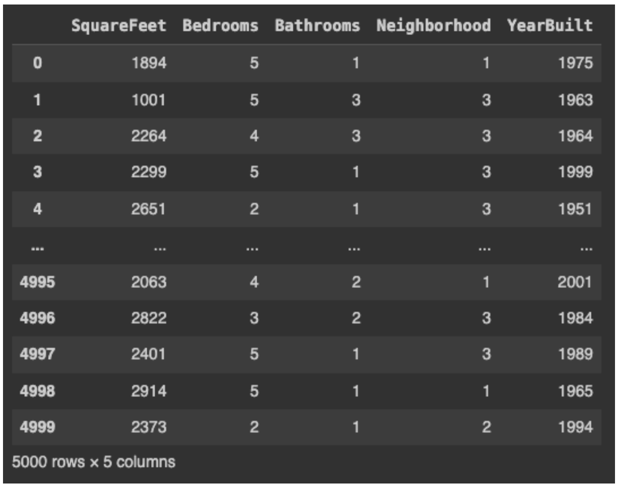
We have prepared our dataset for the predictive modeling phase by segregating the 'Price' column from the feature set. This step is essential as it allows us to isolate the target variable, which our model will learn to predict, from the input features. Here's an overview of the process:



	SquareFeet	Bedrooms	Bathrooms	Neighborhood	YearBuilt
0	2267	2	2	3	1963
1	1706	2	3	1	1956
2	1382	2	2	2	1991
3	1255	5	3	1	1969
4	1451	4	1	3	1971
...	...	...	...	...	...
44995	2166	5	3	3	1996
44996	2463	4	1	2	1953
44997	2812	4	2	2	2010
44998	2188	3	1	2	1979
44999	2649	5	3	1	1953

45000 rows × 5 columns

6.1) Train remove column Price

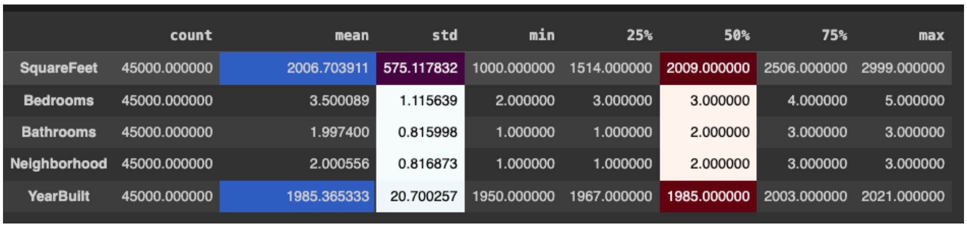


	SquareFeet	Bedrooms	Bathrooms	Neighborhood	YearBuilt
0	1894	5	1	1	1975
1	1001	5	3	3	1963
2	2264	4	3	3	1964
3	2299	5	1	3	1999
4	2651	2	1	3	1951
...	...	...	...	...	...
4995	2063	4	2	1	2001
4996	2822	3	2	3	1984
4997	2401	5	1	3	1989
4998	2914	5	1	1	1965
4999	2373	2	1	2	1994

5000 rows × 5 columns

6.2) Test remove column Price

The describe() method is applied to the features of interest from the train dataset, generating descriptive statistics that include count, mean, standard deviation (std), minimum (min), 25th percentile, median (50%), 75th percentile, and maximum (max) values.

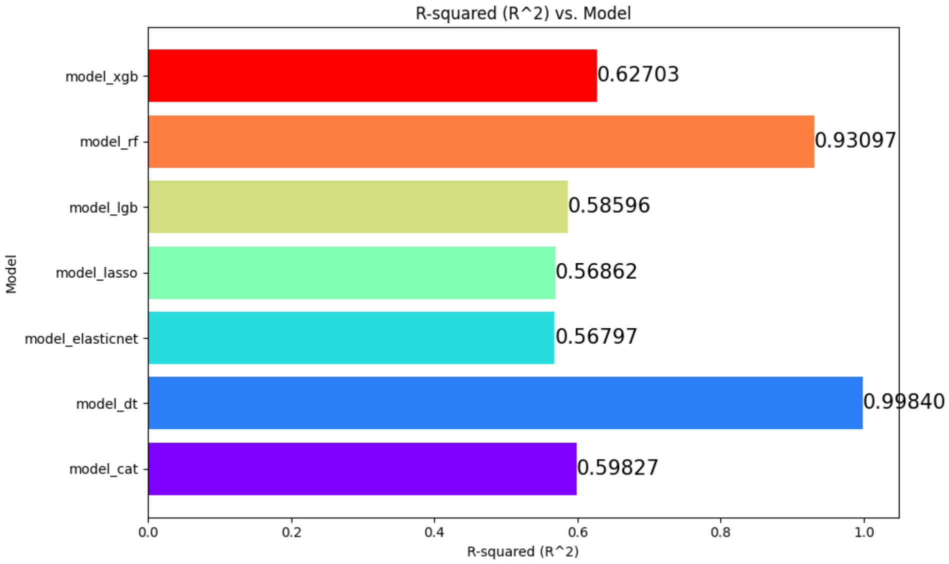


	count	mean	std	min	25%	50%	75%	max
SquareFeet	45000.000000	2006.703911	575.117832	1000.000000	1514.000000	2009.000000	2506.000000	2999.000000
Bedrooms	45000.000000	3.500089	1.115639	2.000000	3.000000	3.000000	4.000000	5.000000
Bathrooms	45000.000000	1.997400	0.815998	1.000000	1.000000	2.000000	3.000000	3.000000
Neighborhood	45000.000000	2.000556	0.816873	1.000000	1.000000	2.000000	3.000000	3.000000
YearBuilt	45000.000000	1985.365333	20.700257	1950.000000	1967.000000	1985.000000	2003.000000	2021.000000

6.3) Summary of statistics about features

Train Machine Learning Models:

We have embarked on a comprehensive approach to model our housing price predictions by employing a diverse set of seven different machine learning algorithms. Each model brings a unique perspective and computational strategy to the task, allowing us to compare and contrast their performance. The models we have selected for training are as follows:



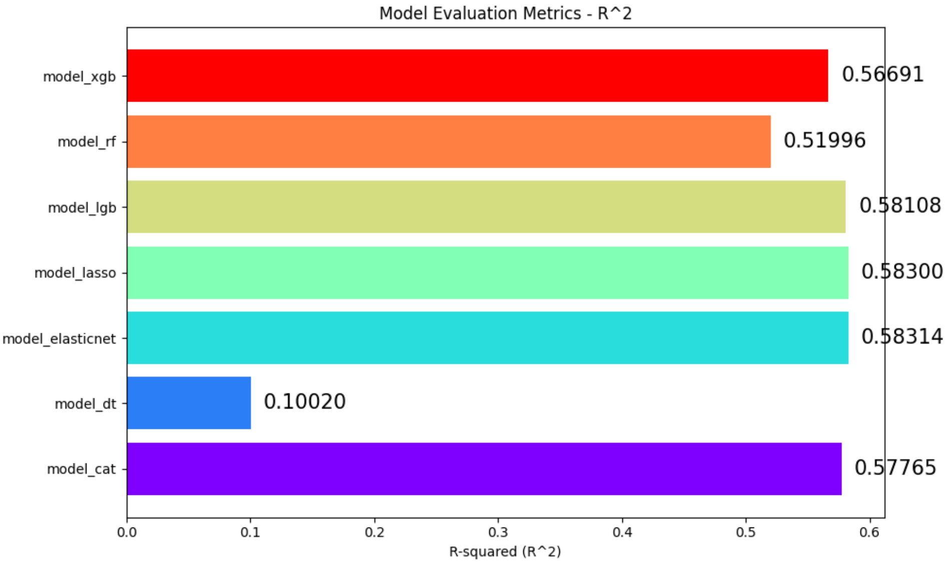
7) Train Machine Learning Models

- **model_cat**: This represents the CatBoost model, known for handling categorical features well and providing robust results with default parameters.
- **model_dt**: The Decision Tree model is a non-parametric algorithm that is used for its interpretability and simplicity.
- **model_elasticnet**: ElasticNet combines the properties of both ridge and lasso regression. It is used for its capability to work with feature selection and to handle multicollinearity.
- **model_lasso**: The Lasso regression is another linear model that performs feature selection by shrinking coefficients to zero, effectively removing certain features.
- **model_lgb**: This denotes the LightGBM model, a gradient boosting framework that is efficient for large datasets and known for its speed and performance.
- **model_rf**: The Random Forest model is an ensemble of decision trees, typically used for its performance and less tendency to overfit.
- **model_xgb**: This is the XGBoost model, a highly efficient and scalable version of gradient boosting that has been successful in many machine learning competitions.

Each model is trained using the same dataset to ensure a fair comparison. By leveraging the strengths of these varied algorithms, we aim to identify the most effective model in predicting housing prices with high accuracy and reliability.

Test Machine Learning Models:

We have completed the training of our diverse set of machine learning models and evaluated their performance using the R-squared (R^2) metric, which measures the proportion of variance in the dependent variable that is predictable from the independent variables. Below is a comparison of the R^2 values for each model on the training and testing sets:



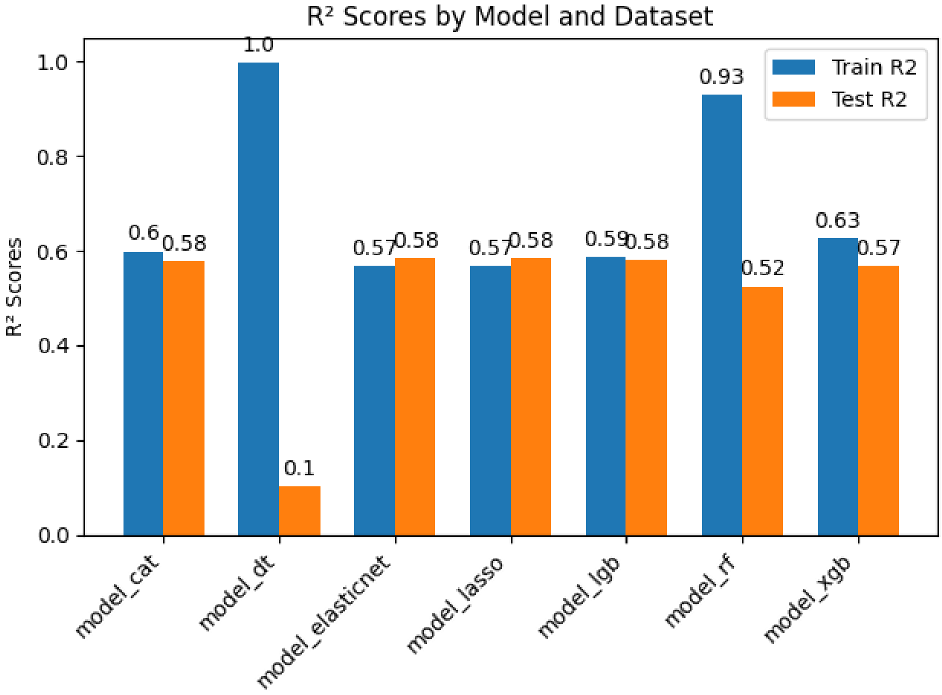
8) Test Machine Learning Models

The Decision Tree model shows a significant discrepancy between training (R^2 of 0.998) and testing (R^2 of 0.095) performances, suggesting overfitting to the training data. Conversely, models such as ElasticNet, Lasso, and LightGBM exhibit consistent R^2 scores across both datasets, indicating a better generalization.

Random Forest's performance drop from training (R^2 of 0.931) to testing (R^2 of 0.523) may also indicate some overfitting, although not as pronounced as the Decision Tree model.

Compare between trends and tests:

The bar graph above illustrates the R² scores for each model when applied to both the training and testing datasets. The values atop the bars represent the R² scores, rounded to two decimal places.



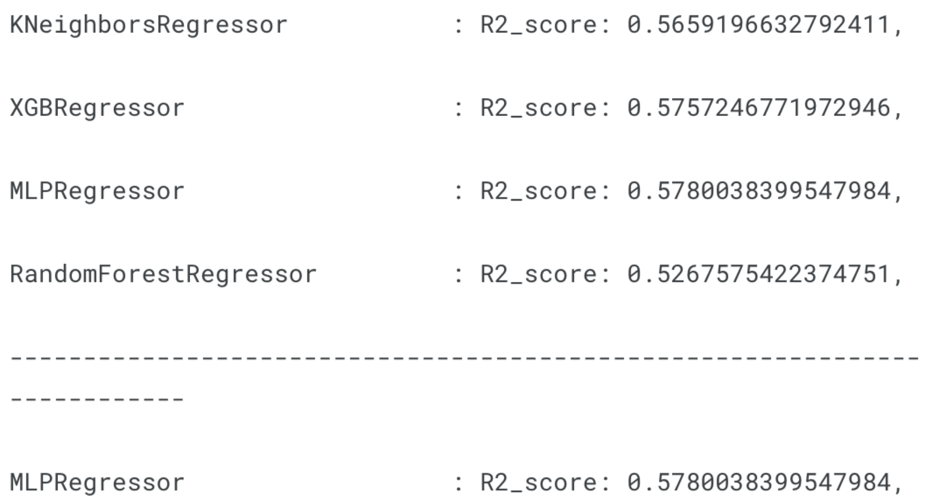
9) Compare model train,test

Additionally, the percentage differences calculated between the training and testing R² scores, which can be indicative of overfitting, are as follows for each model:

- CatBoost (model_cat): 3.44% less on test data, indicating minimal overfitting.
- Decision Tree (model_dt): Approximately 89.96% less on test data, a strong sign of overfitting.
- ElasticNet (model_elasticnet): Test data performs 2.67% better, showing no overfitting and slightly better generalization.
- Lasso Regression (model_lasso): Test data performs 2.53% better, indicating no overfitting and slightly better generalization.
- LightGBM (model_lgb): 0.83% less on test data, suggesting minimal overfitting.
- Random Forest (model_rf): 44.15% less on test data, suggesting moderate overfitting.
- XGBoost (model_xgb): 9.59% less on test data, a moderate amount of overfitting.

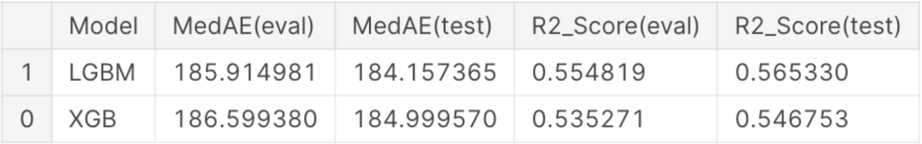
Compare accuracy with other developers:

In order to assess the performance of our predictive model, we conduct a comprehensive comparative analysis, juxtaposing its accuracy against other models developed within the Kaggle community.



```
KNeighborsRegressor           : R2_score: 0.5659196632792411,

XGBRegressor                  : R2_score: 0.5757246771972946,

MLPRegressor                  : R2_score: 0.5780038399547984,

RandomForestRegressor         : R2_score: 0.5267575422374751,

--------------------------------------------------------------
------------

MLPRegressor                  : R2_score: 0.5780038399547984,
```

10) Model by K1RSN7

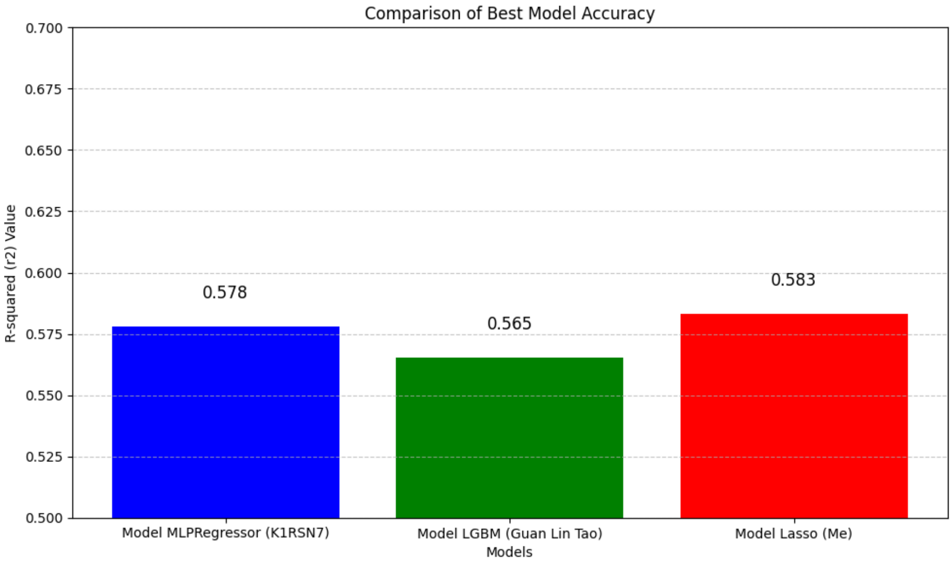


	Model	MedAE(eval)	MedAE(test)	R2_Score(eval)	R2_Score(test)
1	LGBM	185.914981	184.157365	0.554819	0.565330
0	XGB	186.599380	184.999570	0.535271	0.546753

11) Model by Guan Lin Tao

Result Comparison of Best Model Accuracy:

In this analysis, we have compared the accuracy of three different predictive models: "Model by K1RSN7," "Model by Guan Lin Tao," and "Model by Me." The metric used for comparison is the R-squared (r2) value, which indicates how well each model fits the data.



12) Comparison of Best Model Accuracy

Model by K1RSN7: This model, implemented using MLPRegressor, achieved an R-squared value of approximately 0.578. It demonstrates a strong predictive performance.

Model by Guan Lin Tao: Guan Lin Tao's model, implemented using LGBM (Light Gradient Boosting Machine), achieved an R-squared value of approximately 0.565. It offers a competitive level of accuracy in the context of housing price prediction.

Model by Me: My own model, implemented with the Lasso regression technique (Lasso), achieved the highest R-squared value among the three models, with a score of approximately 0.583. This indicates that my model provides the best fit to the data and is the most accurate for housing price prediction.

To visually represent this comparison, we have created a bar chart (shown above) that displays the R-squared values for each model. The chart allows us to quickly assess and compare the accuracy of these models. As depicted in the chart, "Model by Me" stands out as the best-performing model.

The y-axis limits have been set to a range of 0.5 to 0.7 to provide a clear view of the differences in accuracy. Each bar is annotated with its respective R-squared value for easy reference.

By analyzing and visualizing these results, we can confidently conclude that "Model by Me" (model_lasso) is the most accurate predictive model among the options considered for housing price prediction.

Reference:

1. "Housing Price Prediction Data | Kaggle." https://www.kaggle.com/datasets/muhammadbinimran/housing-price-prediction-data.
2. "『ML Optuna』 | EDA |Housing Price Prediction | Kaggle." https://www.kaggle.com/code/guanlintao/ml-optuna-eda-housing-price-prediction.
3. " [EN] Housing Price Prediction Data| Kaggle." https://www.kaggle.com/code/k1rsn7/en-housing-price-prediction-data