

# AI-based Project

Assignment 2 – COS30049

Group 81 – Fantastic Realtors

**Names and Student IDS:**

Anthony Ngo – 104548672

Chayan Kapoor – 104202680

Dave Nguyen – 104697710

Table of Contents this page

[AI-based Project 1](#_Toc190204682)

[Introduction 3](#_Toc1032220284)

[Data Collection and Data Processing 3](#_Toc1157099976)

[Visualization and Feature Analysis 6](#_Toc1080000746)

[Machine Learning Model Selection for Property Price Prediction 9](#_Toc1180783156)

[Initial Exploration: Classification and Clustering 9](#_Toc360394792)

[Transition to Regression Models 9](#_Toc1556729960)

[Feature Importance and Residual Analysis 10](#_Toc967309366)

[Justification for Final Model Selection 10](#_Toc839013901)

[Technical implementation 11](#_Toc1718849572)

[Implementation Evaluation 12](#_Toc438277182)

[References: 16](#_Toc334537997)

### **Introduction**

The AI-based project for housing price prediction in Melbourne aims to develop a robust machine learning model that provides accurate insights for potential homebuyers, investors, and anyone interested in understanding the real estate market. Leveraging real-world data from the Melbourne housing market, the project focuses on identifying key features that influence property prices and using advanced machine learning techniques to offer reliable price predictions.

### **Data Collection and Data Processing**

We use visualization methods like correlated heatmaps and boxplots to investigate how various home features impact property values. Our data was collected utilizing a CSV record of Melbourne real estate sales, we used the Kaggle website to find the Melbourne real estate sales dataset (Biju, 2023). This analysis's elements comprise the number of rooms, property category, proximity to the city center, number of bathroom facilities, and parking places. During our work progress, we stumbled into many problems mainly due to low-quality data which made us take more time than we anticipated to fix and find more suitable datasets for generating diagrams like Correlation Matrix for Numerical Features.

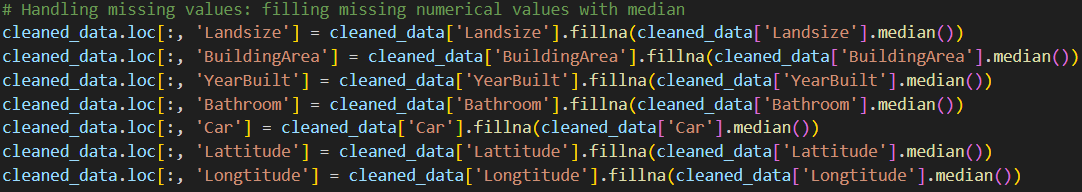
We don't have separate cleaned dataset files as we have cleaned it within the Python code itself and for that, and to preprocess the collected data, we used the following steps:

* Handling Missing Values: Several columns contained missing data, and several solutions were utilized to deal with them.

Numerical Features: Missing values in Landsize, BuildingArea, YearBuilt, Bathroom, Car, Latitude, and Longitude were replaced with the median values from those columns.

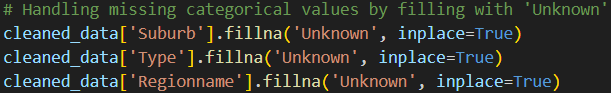
The first line of the ‘Handling missing value’ section searches for missing values in the Landsize column of the cleaned\_data Data Frame. For rows when the Landsize value is absent, the values are substituted with the Landsize column's median value. The median is used to reduce the influence of outliers since it is a reliable value for distributions that are distorted. The loc technique is used to choose the Landsize column, and the colon preceding the column name (:) guarantees that it extends to all rows. The. fillna() function is used to fill in missing data. The input within fillna() indicates that NaN values shall be replaced with the median of the Landsize column.

The second line fills in the missing numbers in the BuildingArea column. Any values that are not present in BuildingArea will be substituted with the column's median value to ensure that there are no additional NaN in this function. The third line resolves the values not present in the YearBuilt column (which represents the year the asset was built), replacing NaN values with the median of all YearBuilt values. The lines after that follow the same principles.

*Fig.1: Filling missing numerical values with median with included cleaned data.*

This part will guarantee that any missing values in these columns are replaced with acceptable and reliable statistical values (medians), providing the data with completeness and readiness for additional examination and training models.

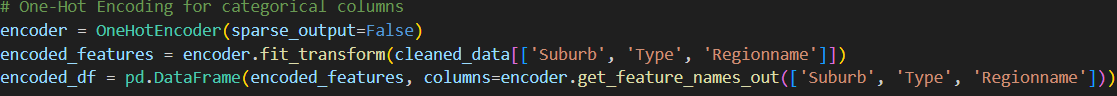
Categorical Features: Missing values in Suburb, Type, and Regionname have been replaced with 'Unknown'.



*Fig.2: Handling missing categories values by filling with ‘Unknown’*

Categorical columns (Suburb, Type, and Regionname) were converted to numerical form using one-hot encoding, allowing them to be employed in machine learning models.

OneHotEncoder is an algorithm in the sklearn.preprocessing module that converts categorical input into one-hot encoded format. sparse\_output=False specifies that the encoded output should be an abundant array instead of a sparse matrix (which conserves memory for big, sparse datasets but is more difficult to comprehend directly). The result is a DataFrame (encoded\_df) with every individual classification from the initial columns assigned a value of 1 or 0 for every single row.

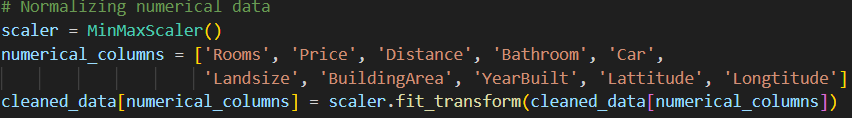
*Fig.3: One-hot encoding method.*

* Normalisation of Numeric Features:

Numerical variables such as Rooms, Distance, Bathroom, Car, Landsize, BuildingArea, YearBuilt, Lattitude, and Longitude were normalized using Min-Max scaling to convert their values into a 0-1 range. MinMaxScaler is a transformer in the sklearn.preprocessing package. It scales each characteristic to a predetermined range, usually [0, 1]. By converting all values to an identical range (0 to 1), the technique assures that higher values do not outweigh smaller ones.

cleaned\_data[numerical\_columns] picks an element of the dataset (cleaned\_data) that only includes the numerical columns established before. scaler.fit\_transform(): This strategy accomplishes two things:

* Fit: The fit() function determines the lowest and maximum values for each numerical column in the dataset. The scaler analyses the data to get the lowest and greatest values for each attribute.
* The transform() function then executes the min-max scaling conversion to each value in the specified columns. Each value is adjusted using the previously given algorithm, guaranteeing the values are between 0 and 1.

*Fig.4: Normalizing numerical data.*

* Log-transformation of the target variables (price):

A Log-transformation was used to smooth out the variation in home prices. This procedure helps to mitigate the effects of excessive price fluctuations and enhances the model's efficiency.



*Fig.5: Enhances the model's efficiency through Log-transform.*

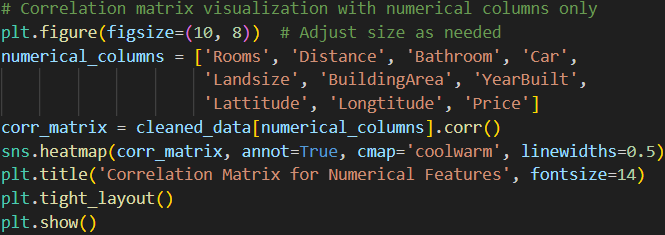
### **Visualization and Feature Analysis**

* Correlation Matrix:

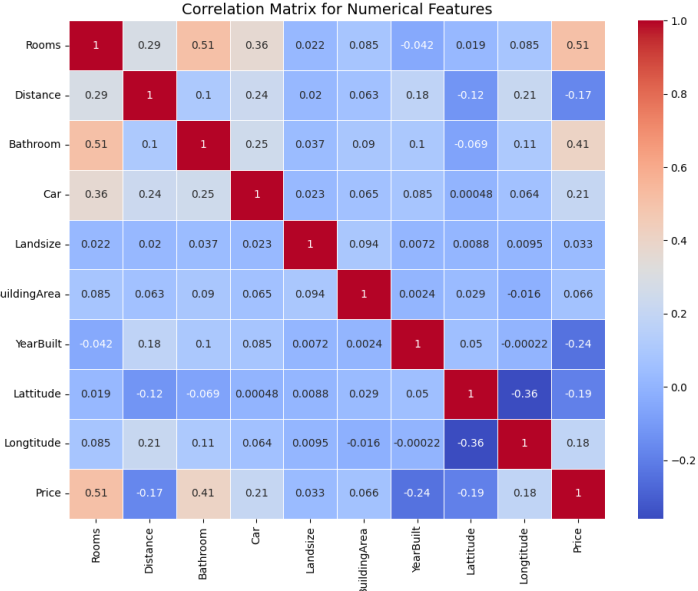
A heatmap was created to see the correlations between numerical variables and the goal variable (price). The correlation matrix revealed some important insights:

* Rooms had a moderately favorable connection with Price.
* Distance exhibited a somewhat negative association, showing that houses further outside the city possess lower prices.
* Landsize and BuildingArea had a weak positive association with Price, however they still provided valuable data.

‘plt.figure(figsize=(10), 8)’ produces a new figure within Matplotlib with the provided dimensions of 10 units broad by 8 units height. The corr\_matrix method computes the pairwise correlation factors among the numerical columns specified in numerical\_columns. The correlation coefficient measures the extent to which two variables are linearly connected. 'corr\_matrix' is the heatmap's input data, derived from the previously produced correlation matrix. 'annot=True' includes numerical annotations to every cell in the heatmap, displaying the precise correlation value in the center of each colored square. 'cmap='coolwarm' indicates the color mapping used for the heatmap. 'linewidths=0.5' puts a 0.5-width margin between the cells to help distinguish between them.



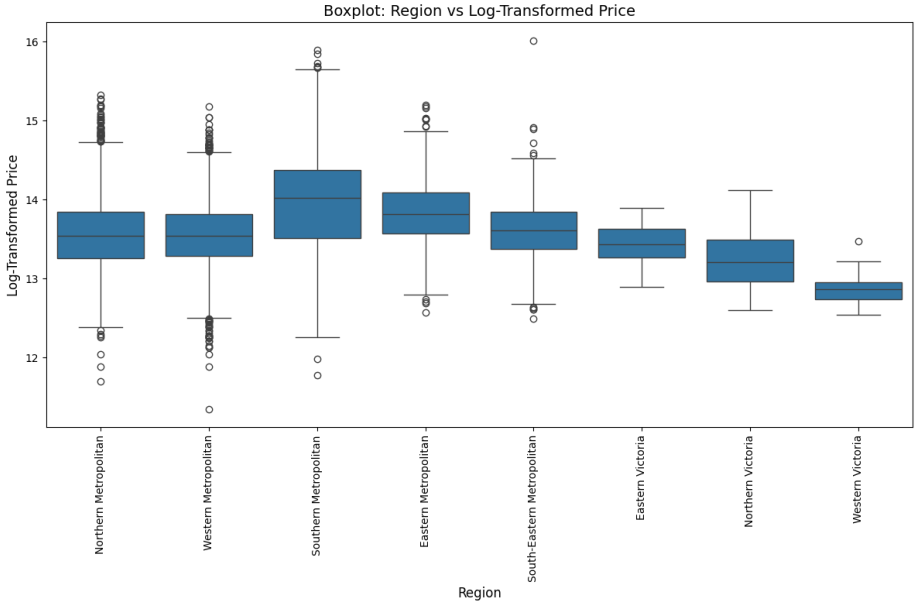
*Fig.6: Codes to generate heatmap.*



*Fig.7: Correlation Matrix for Numerical Features diagram.*

* Boxplot of Region compared to Price:

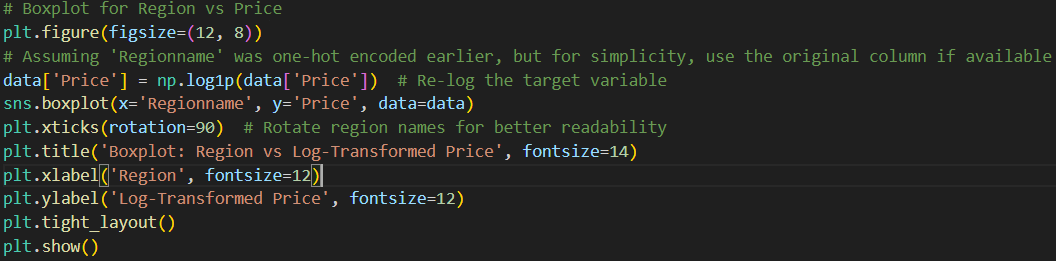
We visualized pricing variations throughout Melbourne's various areas. Applying a log-transformed pricing scale, this boxplot demonstrated large price discrepancies among areas, with more wealthy areas, such as Southern Metropolitan, frequently experiencing higher property values.

*Fig.8: Boxplot graph (Region vs Log-Transformed Price).*

'plt.figure(figsize=(12, 8))' generates a new figure for the graph with the supplied dimensions of 12 units wide and 8 units high. Data['Price'] = np.log1p(data['Price']) The np.log1p() method applies a log conversion on the Price column.

The ‘log1p()’ function calculates the ideal logarithm of 1 + Price, and this is a frequent strategy for normalizing skewed data, particularly for prices with exceptions or a strong right skew. The log transformation makes the price dispersion more controllable and easier to model using algorithms. 'sns.boxplot(x='Regionname', y='Price', data=data)' generates a boxplot with Seaborn.

Because there are so many areas, displaying the region titles horizontally may cause them to overlap. 'plt.xticks(rotation=90)' adjusts the x-axis names by 90 degrees, showing them vertically for better reading.

*Fig.9: Codes to generate Boxplot graph.*

### **Machine Learning Model Selection for Property Price Prediction**

In this project, we focused on selecting machine learning models to predict property prices in Melbourne. Since property prices are continuous, we sought models capable of accurately predicting these values. Initially, we explored classification and clustering approaches but found them ineffective, leading us to focus on regression models.

#### **Initial Exploration: Classification and Clustering**

We first tested classification models, including the Random Forest Classifier and Gradient Boosting Classifier, by converting property prices into categories (e.g., low, medium, high). However, both models performed poorly, yielding low precision and F1 scores:

* Random Forest Classifier achieved a precision score of 0.5834 and an F1 score of 0.4662.
* Gradient Boosting Classifier had a precision score of 0.5458 and an F1 score of 0.4907.

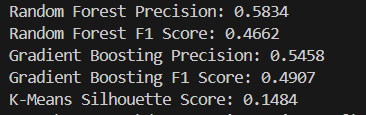


Fig.10: Classifier results.

These low scores indicated that the classification models struggled to capture the nuances in property prices. We also tried K-Means Clustering, but it failed to reveal meaningful groupings, as evidenced by a low silhouette score of 0.1484. Based on these results, we abandoned both classification and clustering methods.

### **Transition to Regression Models**

Recognizing that property prices are best handled as continuous values, we turned to regression models, specifically Random Forest Regressor and Gradient Boosting Regressor. These models are well-suited for tasks involving complex relationships between features and continuous target variables.[1][2]

1. Random Forest Regressor: This model uses an ensemble of decision trees to predict values. By averaging the predictions from multiple trees, it reduces overfitting and improves accuracy. Our testing showed that the Random Forest Regressor performed exceptionally well, with a mean squared error (MSE) of 0.0007 and an R² score of 0.8196, meaning it explained about 81.96% of the variance in property prices. These results made it our top choice for the task.[2]
2. Gradient Boosting Regressor: Gradient Boosting builds models iteratively, with each subsequent model correcting errors made by the previous one. It produced an MSE of 0.0008 and an R² score of 0.7707. Although slightly less accurate than Random Forest, it still performed well and demonstrated its effectiveness for this problem.

### **Feature Importance and Residual Analysis**

One of the key advantages of using tree-based models like Random Forest and Gradient Boosting is the ability to interpret feature importance. In the Random Forest Regressor, the most influential factors driving property prices were the number of rooms, distance to the city, and the property’s regional location. These findings aligned with domain knowledge, as these are well-known drivers of property value.

We also performed residual analysis to assess model accuracy across different price ranges. Residual plots showed that both models produced consistent predictions with minimal bias, confirming their robustness.

### **Justification for Final Model Selection**

Ultimately, we selected Random Forest Regressor and Gradient Boosting Regressor as our final models due to their superior performance in predicting continuous property prices. These models offered high accuracy, with low mean squared error and high R² scores and provided valuable insights into the key features influencing property prices. The ability to analyze feature importance and evaluate residuals further confirmed the reliability of these models.[2]

While regression models were the primary focus, we also incorporated K-Means Clustering into our final model selection.[3] Although clustering does not predict continuous values, it was valuable for identifying hidden structures and patterns in the data. By segmenting properties into distinct clusters based on their characteristics (e.g., number of rooms, location, property size), K-Means provided insights into potential groupings within the dataset. Although the silhouette score of 0.2159 indicated moderate clustering performance, K-Means offered a broader understanding of how properties might group together based on shared attributes, making it a useful supplementary tool for interpreting property price dynamics.

In conclusion, Random Forest Regressor emerged as the most reliable model, with Gradient Boosting Regressor providing additional value. Both models effectively captured the complexity of property prices and delivered robust predictions, while K-Means Clustering offered complementary insights into the property market. This combination of models allowed us to balance accurate price prediction with an understanding of broader property trends, making them the ideal choices for this task.

### **Technical implementation**

Using Python, the project was implemented with these five key libraries:

* pandas for data manipulation
* scikit-learn for machine learning models and preprocessing
* matplotlib and seaborn are used for visualizations
* numpy is used for scientific computing

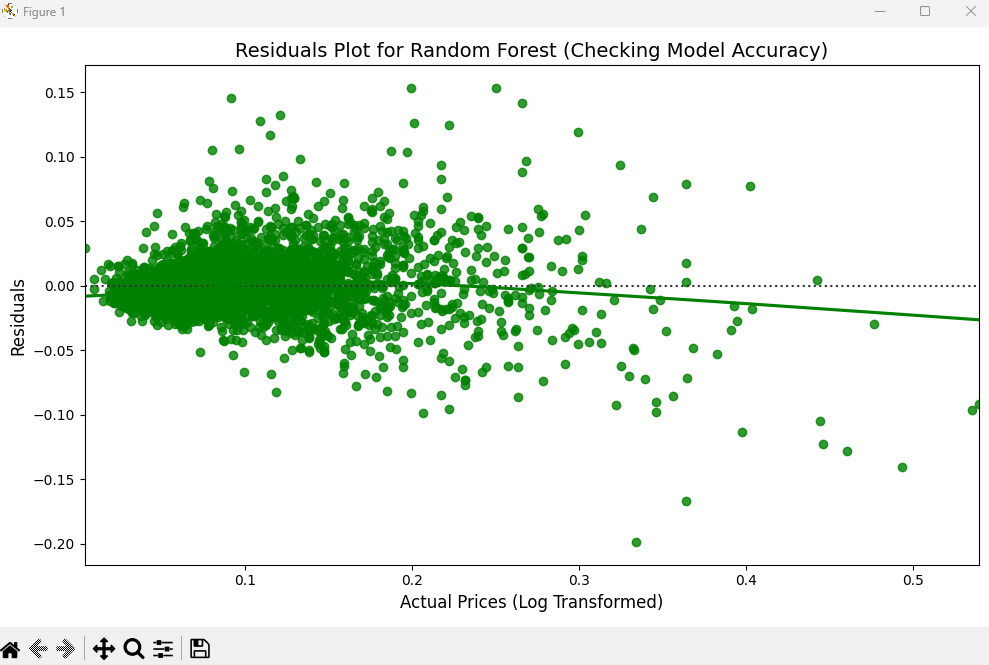
The dataset, “Property Sales of Melbourne City.csv”, was cleaned and used for predicting housing prices using regression models. Key features, such as Suburb, Type, and Regionname were one-hot encoded.

A critical aspect of this project was handling missing values in fields like Landsize, BuildingArea, and YearBuilt, they were filled by applying “Unknown” for categorical fields while filling in numerical fields with median values. For regression analysis, the Random Forest Regressor and Gradient Boosting Regressor were specifically chosen due to their capability to handle complex datasets. Furthermore, K-Means clustering was added to predict patterns in housing types which allows for an unsupervised learning approach to help identify any relationships between property characteristics.

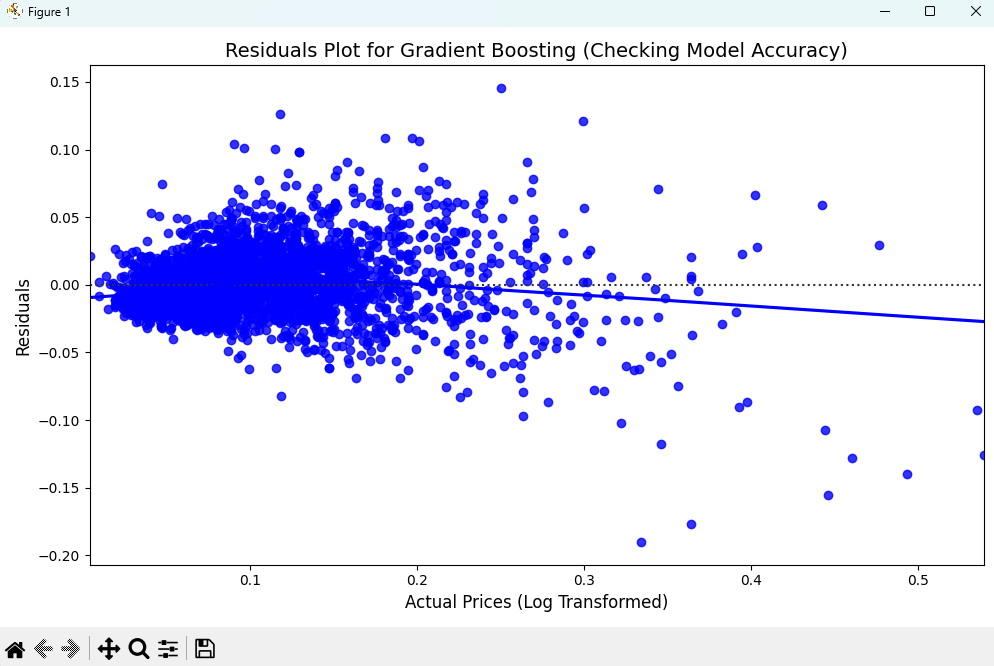
The project involved significant data management, especially around missing data and encoding features. Notably many visualizations were created to identify outliers which help in the model selection process.

### **Implementation Evaluation**

The Project Initially involved 2 classifier models and one clustering model but was changed to 2 regressors, Random Forest and Gradient Boosting, which were used to predict housing prices. Each model was evaluated using the R2 score and mean squared error (MSE). Out of the two the Random Forest model achieved better results as it had a good R2 score and a lower MSE when compared to the Gradient Boosting model which signifies better predictive performance.

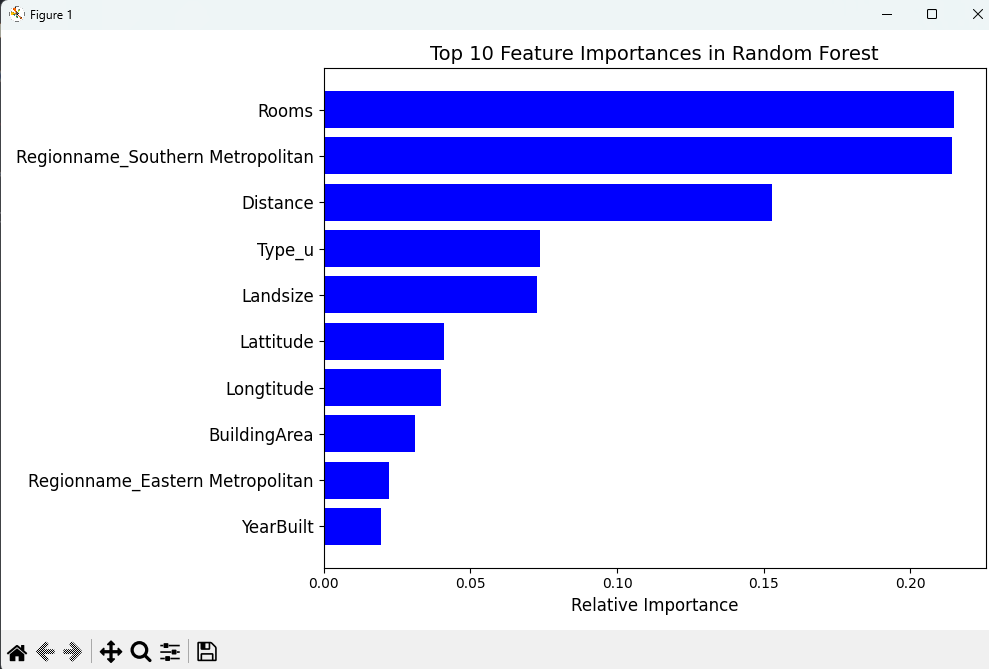


*Fig.11: Residual Plot for Random Forest.*



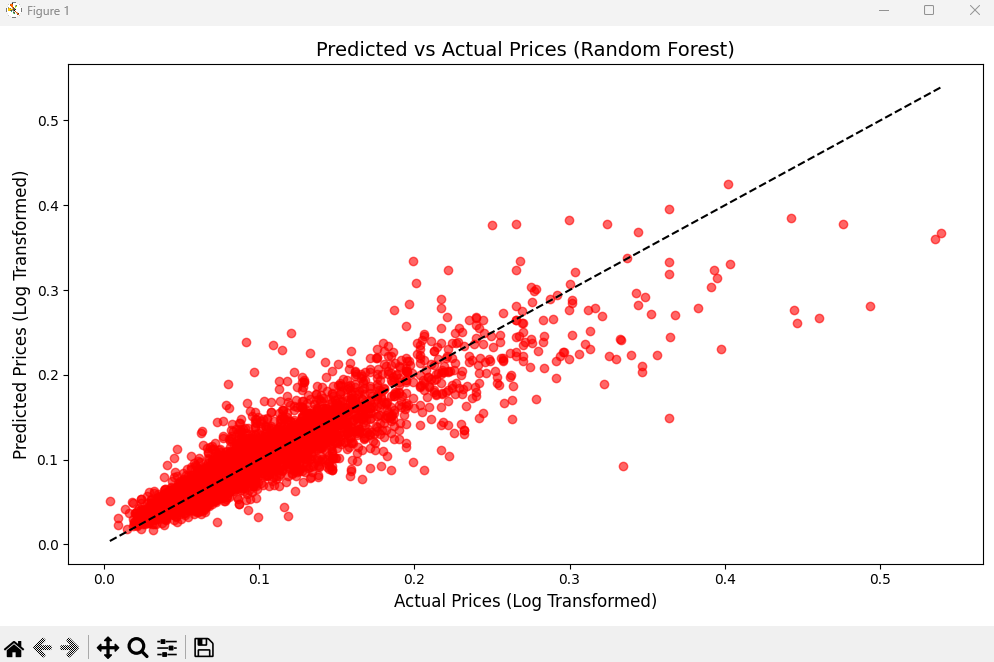
*Fig.12: Residual Plot for Gradient Boosting.*

There were many variables that have significant influence in predicting housing prices, but our feature importance analysis reveals that Distance, BuildingArea, and Landsize were the most influential. Below is the feature important plot that the Random Forest model illustrates to be the most significant.

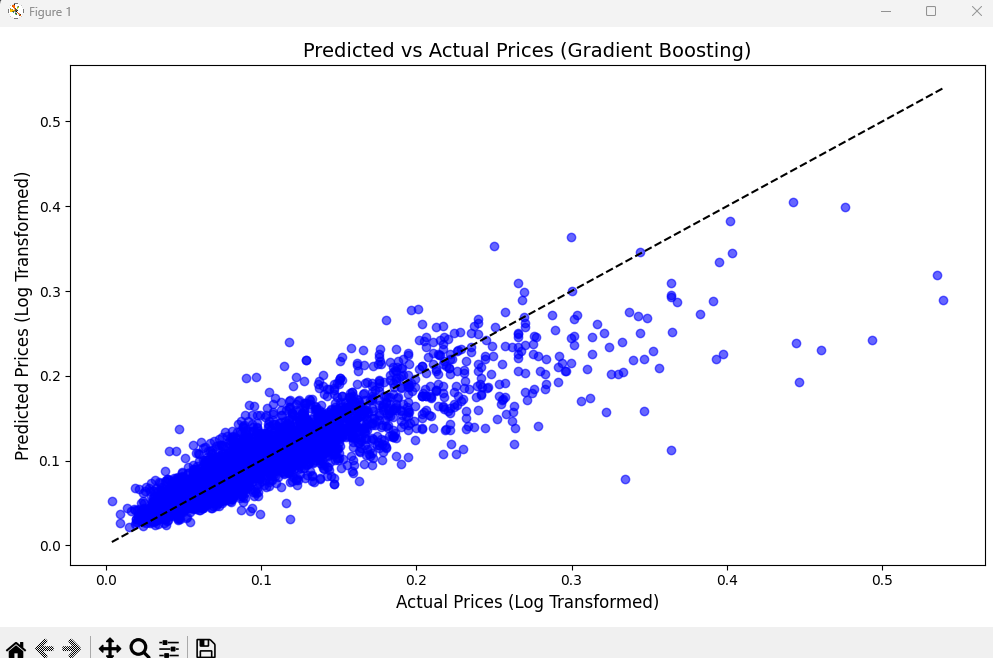


*Fig.13: Top 10 features influencing housing price predictions.*

To better compare the Random Forest and Gradient Boosting models, Residual Plots were created for both to demonstrate the Predicted vs. Actual price scatter plots which further highlight Random Forest providing more accurate predictions.



*Fig.14: Scatter plot for Predicted Vs. Actual Prices using Random Forest.*



*Fig.15: Scatter Plot for Predicted Vs. Actual Prices using Gradient Boosting.*

Conclusion:

In conclusion, this project successfully demonstrated the application of machine learning techniques, particularly Random Forest and Gradient Boosting regressors, to predict housing prices in Melbourne. After an initial exploration of classification and clustering models, the transition to regression models provided significant improvements in predictive accuracy. Random Forest Regressor emerged as the best model, achieving high accuracy and delivering reliable predictions while also offering insights into the most influential features, such as the number of rooms, distance to the city, and property location

### **References:**

1. Dabbura, I. (2018). *K-means clustering: Algorithm, applications, evaluation methods, and drawbacks*. [online] Medium. Available at: <https://towardsdatascience.com/k-means-clustering-algorithm-applications-evaluation-methods-and-drawbacks-aa03e644b48a>. [Accessed 20 Sep. 2024].
2. GeeksforGeeks (2019). *Random Forest Regression in Python*. [online] GeeksforGeeks. Available at: <https://www.geeksforgeeks.org/random-forest-regression-in-python/#:~:text=Random%20Forest%20Regression%20is%20a> [Accessed 22 Sep. 2024].
3. Scikit-learn.org. (2019). *1. Supervised learning — scikit-learn 0.21.3 documentation*. [online] Available at: <https://scikit-learn.org/stable/supervised_learning.html#supervised-learning>. [Accessed 26 Sep. 2024].
4. Biju, A.A. (2023) *Property sales: Melbourne City*, *Kaggle*. Available at: <https://www.kaggle.com/datasets/amalab182/property-salesmelbourne-city> (Accessed: 01 October 2024).
5. Friedman, J.H. (2001) *Greedy function approximation: a gradient boosting machine*. Annals of Statistics, 29(5), pp. 1189-1232. Available at: <https://doi.org/10.1214/aos/1013203451> [Accessed 01 Oct. 2024].
6. Waskom, M.L. (2021) Seaborn: statistical data visualization. Journal of Open-Source Software, 6(60), p. 3021. Available at: <https://seaborn.pydata.org/> [Accessed 01 Oct. 2024].