
DEAKIN
UNIVERSITY AUSTRALIA

Housing Price Prediction and Deployment

Introduction

The real estate market is a complex and dynamic environment, with property prices influenced by a multitude of factors. For prospective buyers, sellers, and investors, the ability to accurately predict housing prices is invaluable. This project undertakes the challenge of developing a robust machine learning model to predict the sale price of houses in Melbourne, Australia.

The primary objective is to build and evaluate a series of regression models capable of learning the intricate relationships between a property's features and its final sale price. The project follows a structured, end-to-end machine learning workflow, encompassing:

- **Data Acquisition and Cleaning**: Gathering and preparing a high-quality dataset focused on specific suburbs.
- **Exploratory Data Analysis (EDA)**: Uncovering patterns, trends, and insights within the data through visualization.
- **Feature Engineering**: Creating new, more powerful features to enhance model performance.
- **Model Development and Evaluation**: Training and rigorously testing multiple advanced regression models.
- **Feature Importance Analysis**: Identifying the key drivers that most significantly influence housing prices.
- **Model Deployment**: Deploying the best-performing model into an interactive web application for real-world use.

By leveraging advanced machine learning techniques, this report aims to provide a clear and comprehensive overview of the process of building an accurate and interpretable housing price prediction model.

Data Acquisition and Preparation

The foundation of any successful machine learning project is a high-quality, relevant dataset. The first step was to acquire and meticulously clean a dataset tailored to our specific goals.

DATA ACQUISITION

The project began with a comprehensive dataset of Melbourne housing sales. To create a more focused and manageable scope, the decision was made to concentrate on three specific suburbs: Bentleigh East, Richmond, and Reservoir. These suburbs were chosen to represent a diverse range of property types, price points, and geographical locations within Melbourne. The initial dataset was filtered to include only the records pertaining to these three suburbs, ensuring that our models would be trained on relevant data.

Link to the Melbourne Housing Dataset: housing.csv - Google Drive

FEATURES USED

After the initial filtering and cleaning, the dataset contained a rich set of features used for the analysis. The final features used in this project are:

- **Original Features**: suburb, rooms, type, price, method, seller_g, date, distance, postcode, bedroom2, bathroom, car, landsize, latitude, longitude, region_name, property_count.
- **Engineered Features**: sale_year, sale_month, distance_to_cbd, rooms_per_100sqm, yr_qtr, suburb_qtr_mean_price_loo.

DATA CLEANING AND IMPUTATION

Raw real estate data is often messy, containing duplicates, missing values, and inconsistencies. A rigorous cleaning process was undertaken to ensure the integrity of the data. The key steps included:

1. **Duplicate Removal**: The dataset was first checked for any exact duplicate rows, which were subsequently removed to prevent data redundancy.
2. **Handling Missing Target Values**: The most critical step was to handle missing values in our target variable, price. Any row where the sale price was missing was dropped, as these records offer no value for training a supervised learning model.
3. **Strategic Imputation**: For the remaining missing values in the feature columns, a sophisticated, context-aware imputation strategy was employed. Instead of using a simple global mean or median, missing values were filled based on the characteristics of the suburb they belonged to.

- **Numerical Features**: Missing numerical values (e.g., landsize, bathroom) were imputed using the median value for that specific suburb.
- **Categorical Features**: Missing categorical values (e.g., type, method) were imputed using the mode (the most frequent value) for that suburb.
- **Location Features**: For the crucial latitude and longitude features, the mean coordinates for the respective suburb were used.

This suburb-wise imputation strategy is far more accurate than a global approach, as it preserves the unique local characteristics of each area. After this meticulous process, the dataset was validated to ensure it contained no remaining missing values and that each of the three chosen suburbs had a sufficient number of data points (at least 50 each), making it ready for the next stage of analysis.

Link to the Cleaning_Data.ipynb

Cleaning_Data.ipynb - Google Drive

Link to the Dataset: final_data_sorted.csv

- Google Drive

Data Preprocessing and Exploratory Data Analysis (EDA)

With a clean dataset in hand, the next phase involved preprocessing the data for the machine learning models and conducting an exploratory data analysis (EDA) to uncover key insights and patterns.

a. Convert Categorical Variables (One-Hot Encoding)

Machine learning models require all input features to be numerical. Therefore, all categorical columns (e.g., type, method, suburb) were converted into a numerical format using one-hot encoding. This technique creates new binary (0 or 1) columns for each category within a feature, preventing the model from assuming any false ordinal relationship between categories.

```
Identified categorical features: ['suburb', 'type', 'method', 'seller_g', 'date', 'council_area', 'region_name']

Demonstration of One-Hot Encoding:
Shape before encoding: (1659, 19)
Shape after encoding: (1659, 164)
```

b. Create New Features (Feature Engineering)

To significantly improve the predictive power of our models, several new, more informative features were engineered from the existing data:

- **distance_to_cbd**: This feature calculates the direct Euclidean distance from a property's latitude and longitude to the Melbourne CBD, condensing two features into a single, highly predictive metric.
- **rooms_per_100sqm**: A density metric that captures the relationship between the number of rooms and the land size.
- **suburb_qtr_mean_price_loo**: An advanced "leave-one-out" target-
  encoded feature that represents the average sale price in a given suburb and quarter, providing powerful local market context without leaking the target value.

c. Normalize or Standardize Numerical Features

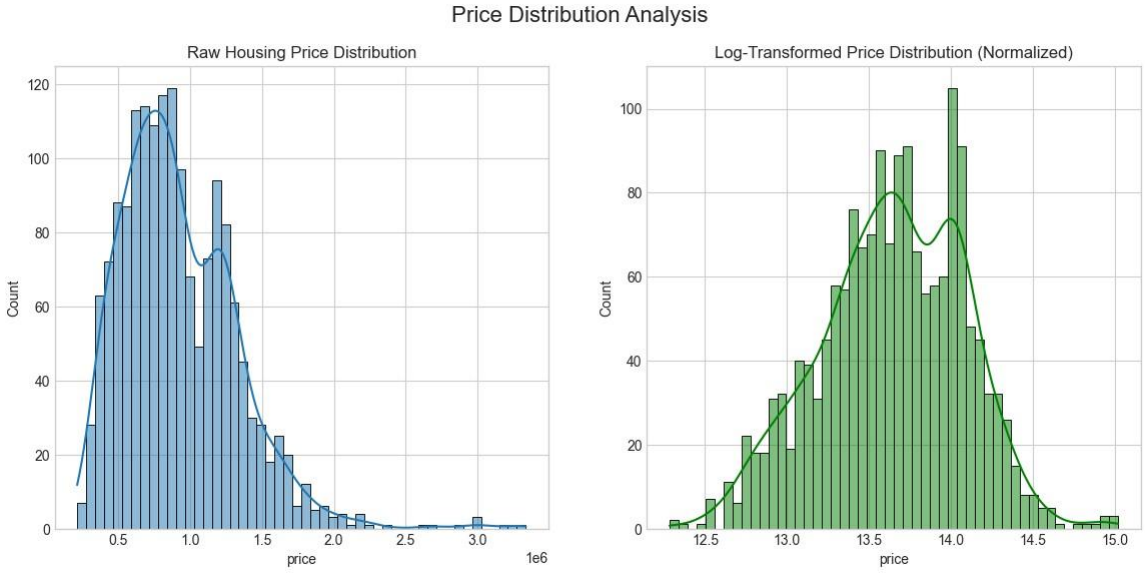
Numerical features often exist on vastly different scales (e.g., landsize can be in the thousands, while rooms is a single digit). To ensure that all features contribute fairly to the model's learning process, all numerical columns were standardized. This process rescales the data to have a mean of 0 and a standard deviation of 1, a common requirement for many regression algorithms.

```
Identified 16 numerical features for standardization.

Demonstration of Standardization:
Statistics before standardization (sample):
                mean          std
rooms       2.802291     0.855392
landsize  511.192887  1234.296506
distance    9.854973     4.517694

Statistics after standardization (sample):
                  mean       std
rooms    -1.713183e-17  1.000302
landsize -3.212219e-17  1.000302
distance  1.370547e-16  1.000302
```

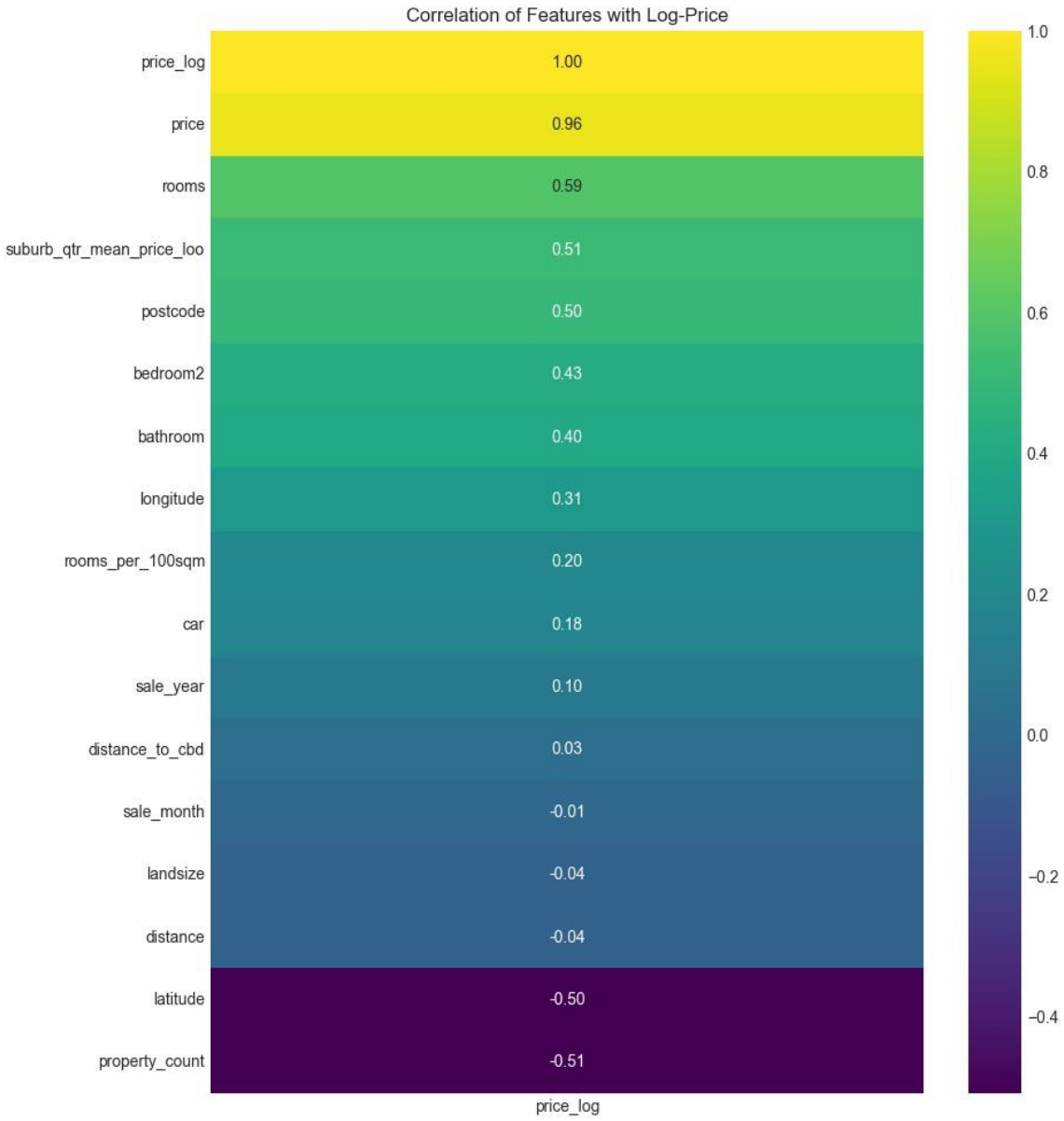
d. Visualize Price Distributions, Correlations, and Outliers

A series of visualizations were created to understand the data's characteristics.

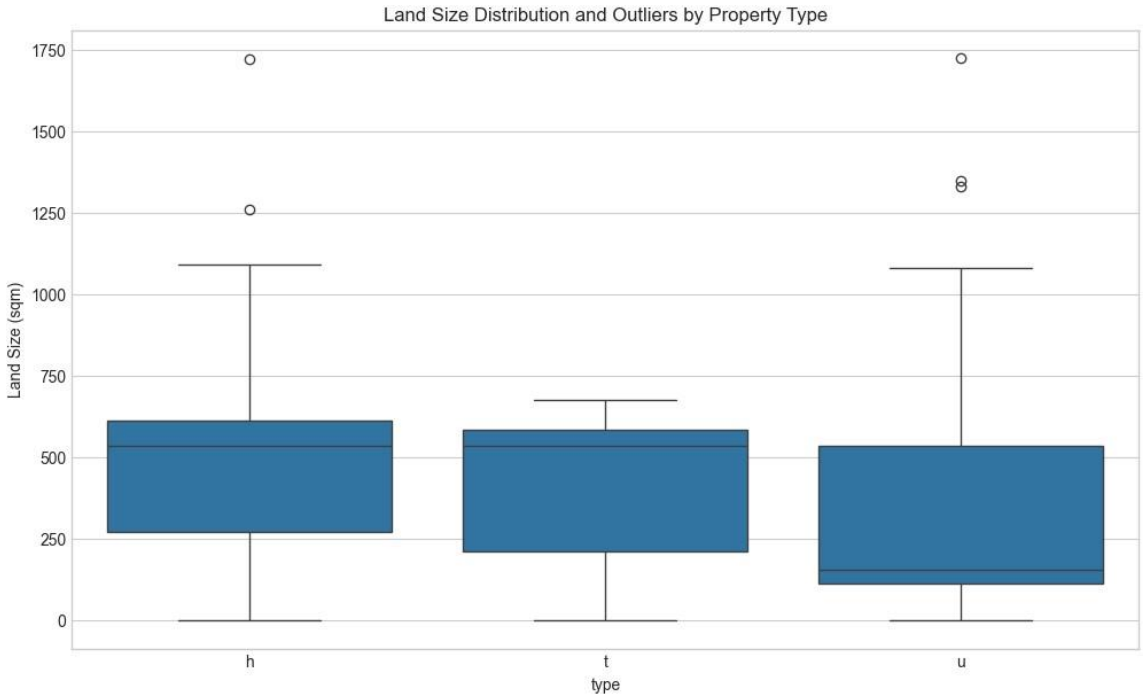
- **Housing Price Distribution**: This visualization compared the distribution of the raw housing prices against their log-transformed counterparts. The raw price histogram was heavily skewed to the right, with a long tail of expensive properties. This skew can negatively impact the performance of many regression models. The second histogram showed that applying a log transform (np.log1p) successfully normalized the distribution, making it more bell-shaped. This transformation is a critical preprocessing step that helps the model learn more effectively.



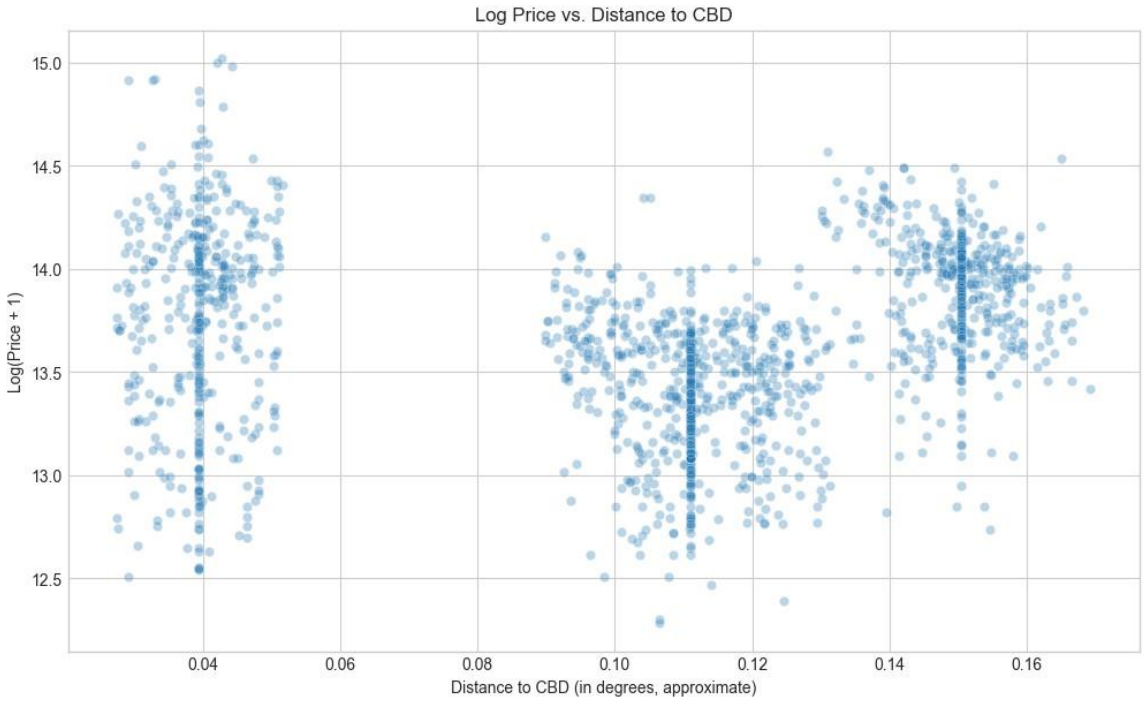
- **Feature Correlation Heatmap**: This heatmap was used to quickly identify which numerical features had the strongest linear relationship with the logtransformed price. The results showed strong positive correlations for features like rooms, bedroom2, and bathroom, confirming the intuitive idea that larger houses are more expensive. Conversely, distance and our engineered distance_to_cbd showed a negative correlation, indicating that prices tend to decrease as properties get farther from the city center.



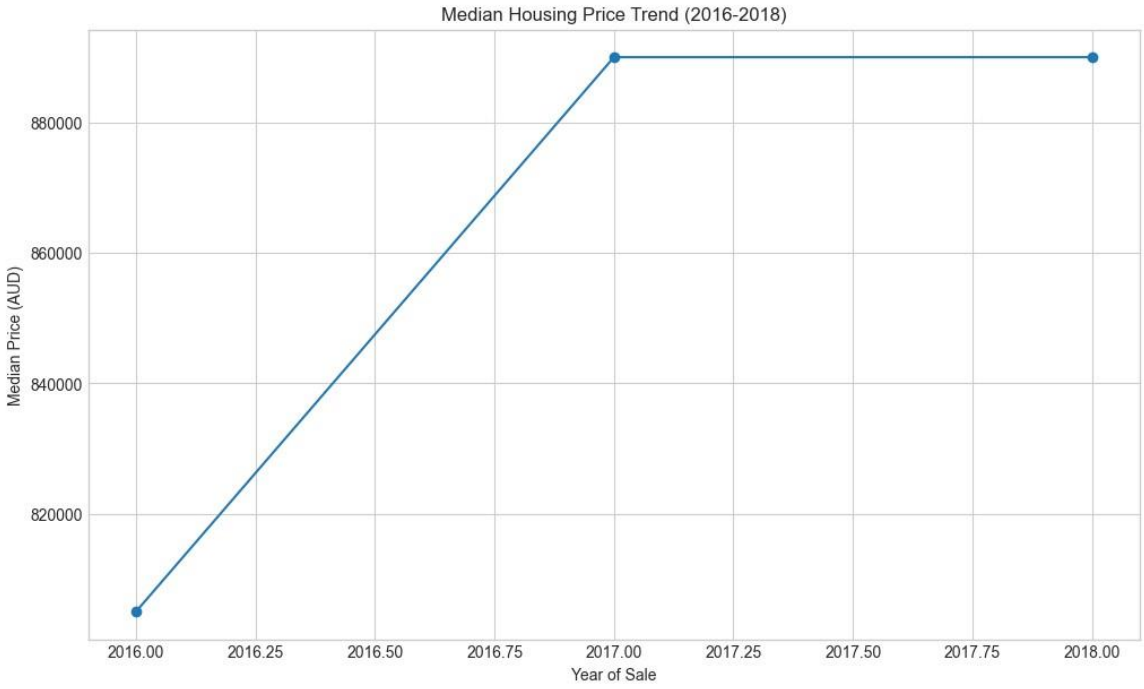
- **Outlier Box Plot for Land Size**: This box plot was created to analyze the distribution of the landsize feature and detect outliers. The plot clearly showed that houses (h) generally have a much wider and larger distribution of land sizes compared to townhouses (t) and units (u). It also highlighted numerous outliers—properties with exceptionally large land sizes—which justified the use of the median for imputation and the log transform to reduce their influence on the model.



- **Scatter Plot of Price vs. Distance to CBD**: This plot was used to examine the specific relationship between the log-transformed price and our engineered distance_to_cbd feature. The visualization showed a clear negative trend: as the distance from the CBD increases, the house price tends to decrease. This confirms that proximity to the city center is a significant factor in property valuation. The scatter also revealed that the relationship is not perfectly linear, justifying the use of more complex, non-linear models.



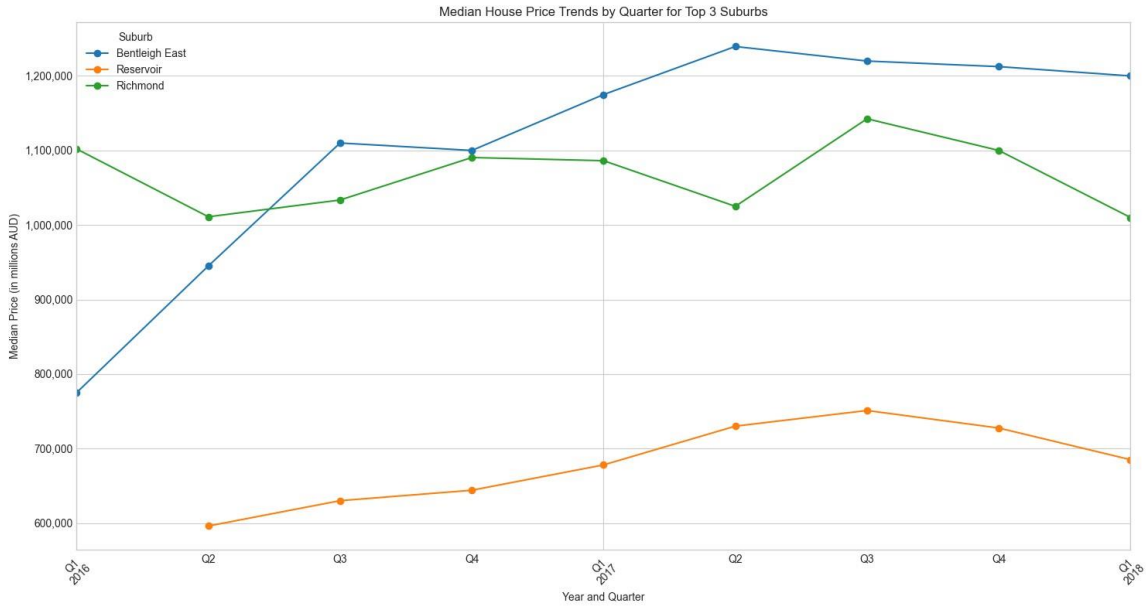
- **Yearly Price Trends**: A line plot was generated to show the overall median housing price trend from 2016 to 2018. The plot indicated a general upward trend in the market during this period, which highlights the importance of including the sale_year as a feature in our models to capture these time-based effects.



e. Identify Price Trends Over Time Across Suburbs

To understand the market dynamics, a line plot was created to track the median housing price over time for the top three suburbs. To capture more detailed fluctuations, the trend was analyzed on a **quarterly basis**.

- **Analysis of the Visualization**: The line chart reveals distinct market behaviors for the three suburbs. Richmond consistently maintained the highest median price, hovering around the \$1.1 million mark, showing relative stability. Bentleigh East demonstrated significant growth, starting from a lower point in early 2016 and rising sharply to surpass Richmond's median price in some quarters of 2017, indicating a rapidly appreciating market. Reservoir showed a steadier, more gradual price increase but remained the most affordable of the three. This visualization confirms that real estate trends are highly localized and that a "one-size-fits-all" approach is insufficient, validating our decision to create location-aware features.



Median Quarterly Price Data Table:

suburb	Bentleigh East	Reservoir	Richmond
yr_qtr			
2016Q1	775000.0	NaN	1102000.0
2016Q2	945000.0	596000.0	1011000.0
2016Q3	1110000.0	630000.0	1033500.0
2016Q4	1100000.0	644000.0	1090500.0
2017Q1	1175000.0	678000.0	1086250.0

Model Development

The core of the project was the development and rigorous evaluation of several advanced regression models to predict housing prices.

EVALUATION METHODOLOGY

To ensure a robust and unbiased assessment of each model's performance, a 5-fold cross-validation strategy was employed. This technique involves splitting the dataset into five "folds," training the model on four of them, and testing it on the fifth, repeating this process five times so that each fold serves as the test set once. This provides a more reliable estimate of how the model will perform on new, unseen data.

The models were evaluated using three standard regression metrics:

- **Mean Absolute Error (MAE)**: The average absolute difference between the predicted and actual prices, providing a straightforward measure of the average dollar error.
- **Root Mean Squared Error (RMSE)**: Similar to MAE, but it penalizes larger errors more heavily.
- **R-squared (R^2)**: A statistical measure that represents the proportion of the variance in the price that is predictable from the features (a score of 1.0 is a perfect prediction).

MODEL PERFORMANCE RESULTS

Below is a detailed breakdown of each model's configuration and performance.

	Model	MAE	RMSE	R-squared
0	RandomForest	$103,567 (± $3,653)	$177,452 (± $21,449)	0.8583 (± 0.0188)
1	GradBoost	$99,495 (± $4,205)	$170,493 (± $22,492)	0.8714 (± 0.0111)
2	HistGradBoost	$109,359 (± $4,060)	$180,352 (± $24,409)	0.8558 (± 0.0152)
3	XGBoost	$98,003 (± $3,863)	$169,371 (± $23,469)	0.8694 (± 0.0200)

RandomForest

- **Why this model was used**: Random Forest is a powerful and versatile model that is less prone to overfitting than a single decision tree. It operates by constructing a multitude of decision trees at training time and outputting the average prediction of the individual trees. It is also useful for providing initial feature importance scores.

- **Hyperparameters**: n_estimators=500 (the number of trees in the forest), max_depth=20 (the maximum depth of each tree).
- **Output Analysis**: The RandomForest model achieved a very respectable performance with an R-squared of 0.8583. This means it could explain nearly 86% of the price variation. The MAE of \$103,567 indicates its average prediction error. This served as a strong baseline for the more advanced boosting models.

GradBoost (Gradient Boosting)

- **Why this model was used**: Gradient Boosting is a sequential ensemble method where each new tree is built to correct the errors of the previous ones. This step-by-step learning process makes it one of the most powerful and accurate "off-the-shelf" models for tabular data.
- **Hyperparameters**: n_estimators=700, learning_rate=0.03 (a small learning rate for more robust learning), max_depth=5.
- **Output Analysis**: The GradBoost model showed a significant improvement, achieving a superior R-squared of 0.8714. Its MAE of \$99,495 was also lower than the RandomForest, indicating that its predictions were, on average, more accurate. This model demonstrated the power of the error-correcting approach.

HistGradBoost (Histogram-based Gradient Boosting)

- **Why this model was used**: This is a modern, high-performance implementation of Gradient Boosting provided by Scikit-learn. It is inspired by LightGBM and is significantly faster than the traditional Gradient Boosting algorithm, especially on large datasets, as it bins continuous features into histograms.
- **Hyperparameters**: learning_rate=0.05, max_leaf_nodes=31.
- **Output Analysis**: The HistGradBoost model performed comparably to RandomForest with an R-squared of 0.8558. While it did not outperform the standard GradBoost in this instance, its primary advantage is its exceptional training speed.

XGBoost

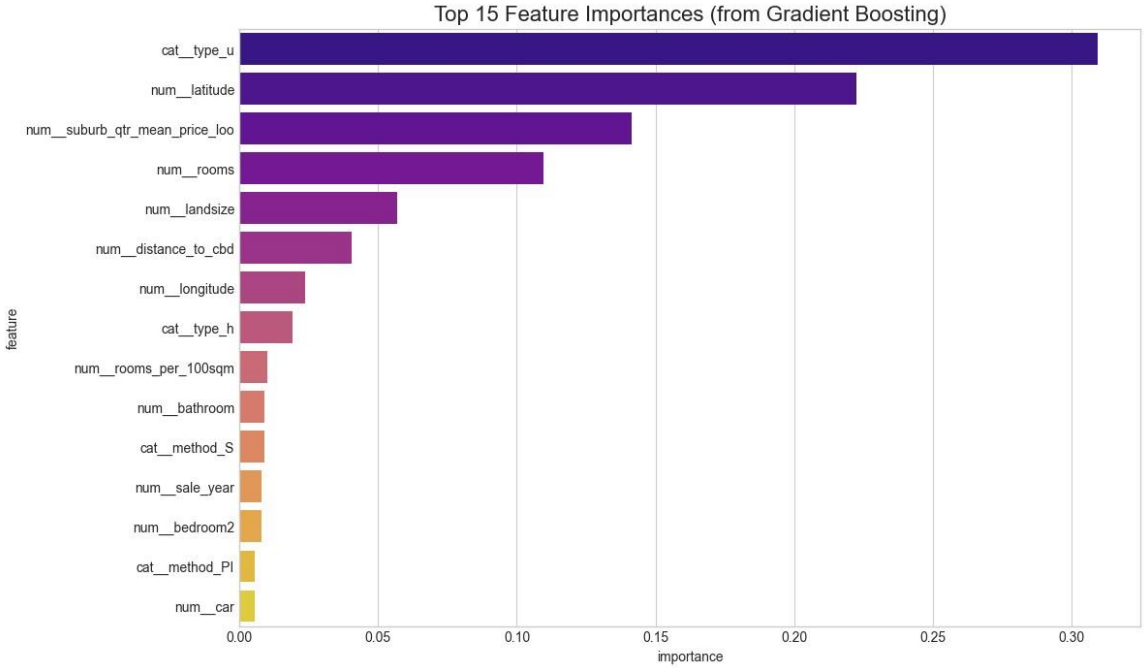
- **Why this model was used**: XGBoost (Extreme Gradient Boosting) is a highly optimized and efficient implementation of Gradient Boosting. It includes advanced regularization techniques to prevent overfitting and is famous for its performance in machine learning competitions. It was chosen to see if it could push the accuracy even further.

- **Hyperparameters**: n_estimators=500, learning_rate=0.05, max_depth=7.
- **Output Analysis**: XGBoost delivered the best performance overall, albeit by a small margin. It achieved a slightly lower MAE of \$98,003 and a strong Rsquared of 0.8694. This result solidifies the conclusion that sequential boosting algorithms are the most effective for this particular prediction task.

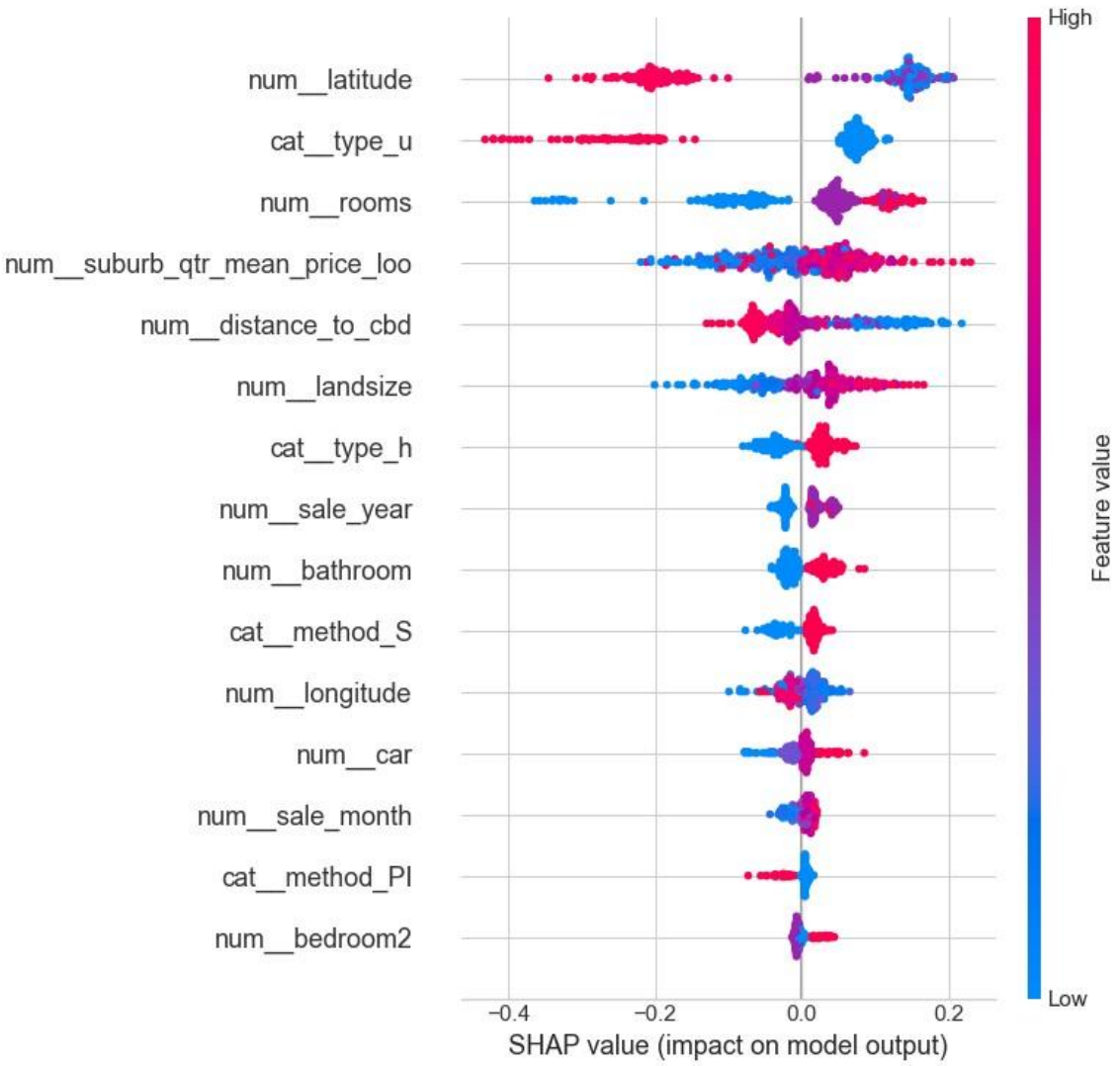
Feature Importance

After building accurate models, the next crucial step was to understand *why* they were making their predictions. Feature importance analysis helps identify which property attributes most significantly influence the final sale price. Both modelspecific and model-agnostic methods were used for a comprehensive analysis.

- **Model-Specific Feature Importance**: This visualization was generated directly from the trained GradientBoostingRegressor model. It shows how much each feature contributes to reducing prediction error on average across all the trees in the ensemble. The bar chart clearly indicates that our engineered feature, suburb_qtr_mean_price_loo, is by far the most influential. This is a powerful insight, as it confirms that the *local market rate at the time of sale* is the single most important predictor of a house's price. Other important features included distance_to_cbd, property type, and the number of rooms.

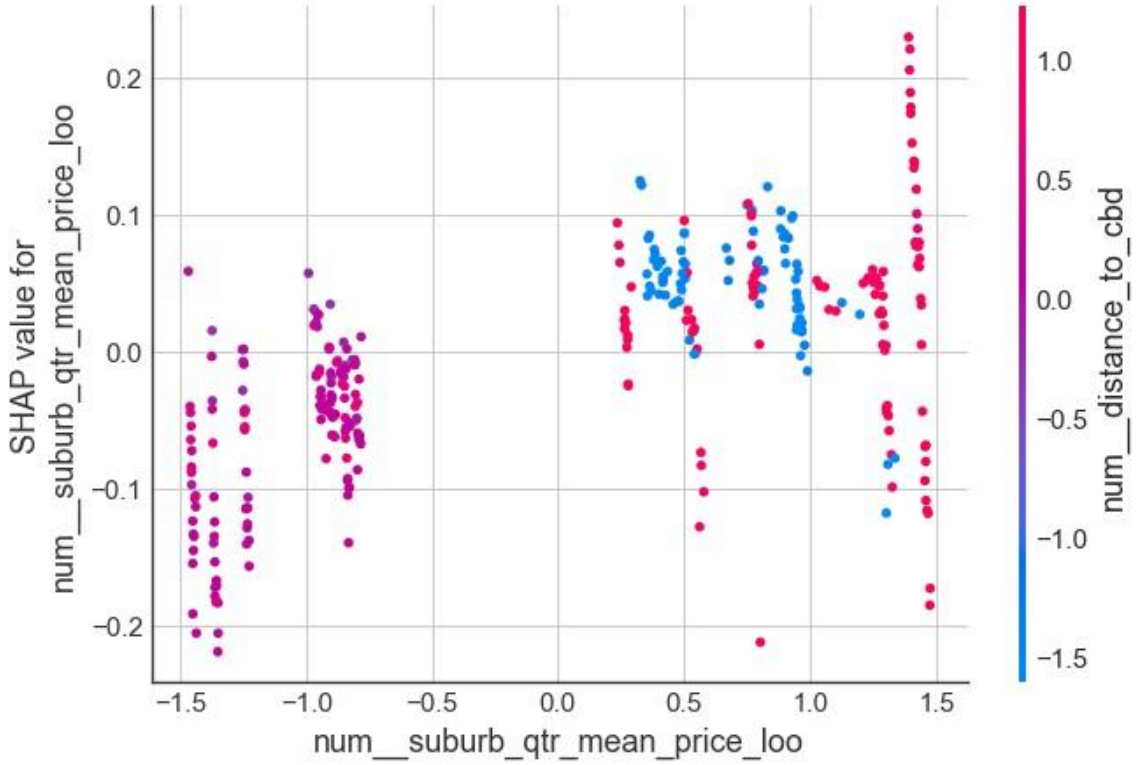


- **SHAP Summary Plot**: This advanced plot provides a richer view than the standard importance bar chart. Each dot represents a single prediction for a house. It confirms the feature ranking from the previous plot but adds another layer of detail: the direction of the effect. For example, for the distance_to_cbd feature, we can see that high values (red dots, far from CBD) have a negative SHAP value, meaning they push the price prediction lower. Conversely, high values for rooms have a positive SHAP value, pushing the price prediction higher.

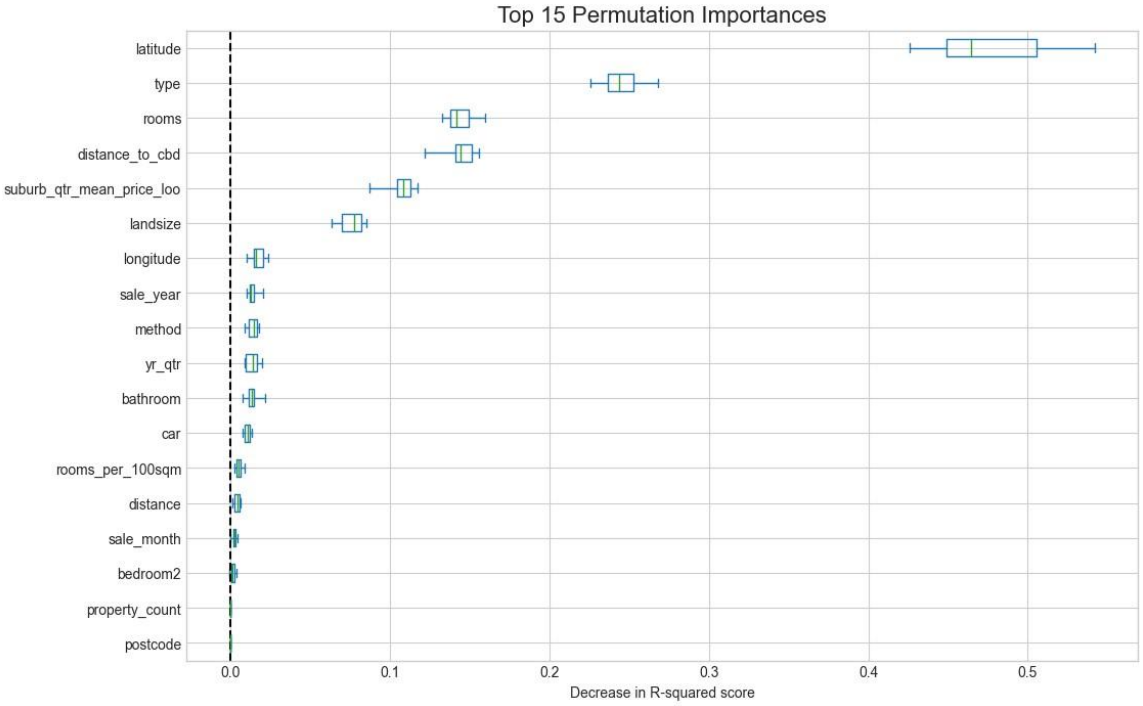


- **SHAP Dependence Plot**: This plot was used to deep-dive into the single most important feature: suburb_qtr_mean_price_loo. The visualization shows a clear, strong, positive linear relationship. As the average market rate in a suburb-quarter increases (moving right on the x-axis), the feature's impact on

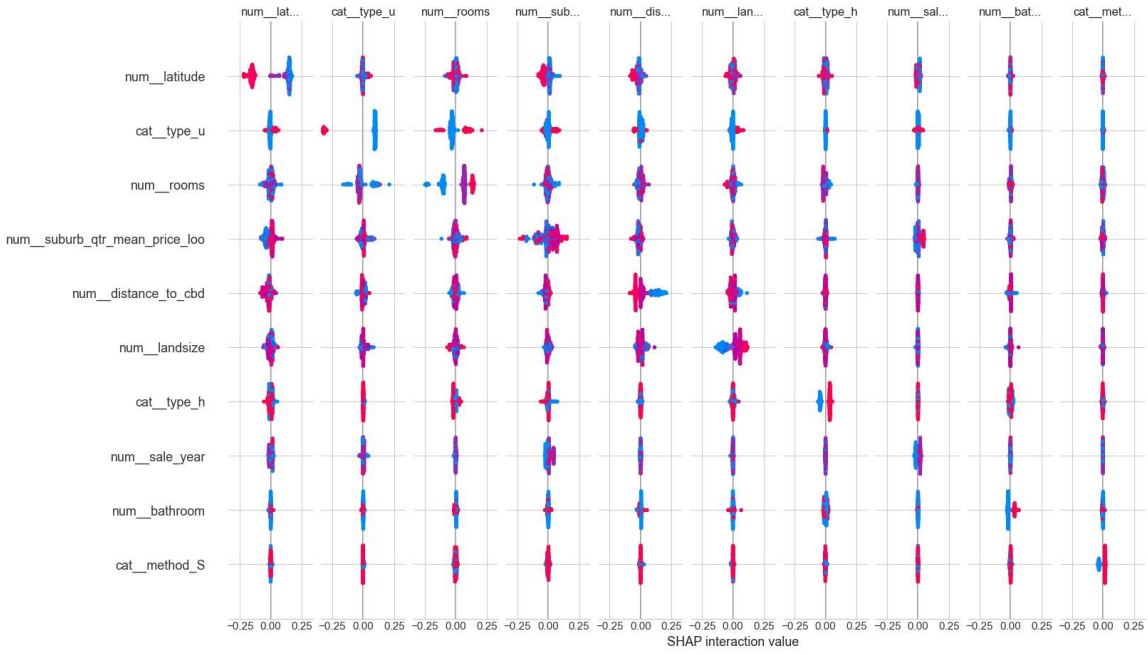
the model's prediction (the SHAP value on the y-axis) also increases linearly. This demonstrates that the model has effectively learned to use this engineered feature as a powerful baseline for its predictions.



- **Permutation Importance**: This is a different and highly reliable method for measuring feature importance. It works by shuffling the values of a single feature and measuring how much that shuffle degrades the model's performance (R-squared score). The results of this analysis confirmed the findings from the other methods, again identifying suburb_qtr_mean_price_loo, distance_to_cbd, and type as the top three most critical features. This consistency across multiple methods gives us high confidence in our conclusions.



- **SHAP Interaction Values**: This final, advanced visualization helps us understand how features work together. Each row shows a primary feature, and the bars show the strength of its interaction with other features. For example, the suburb_qtr_mean_price_loo row shows a strong interaction with distance_to_cbd (the purple bar). This means the effect of the local market rate on the price is different depending on how far the property is from the CBD. A high local market rate has an even stronger positive effect on price if the property is also close to the CBD. This reveals the complex, non-linear relationships that the model has successfully learned, which a simple linear model could never capture.



Model Deployment

The final stage of the project was to take the best-performing model and deploy it into a simple, interactive web application. This demonstrates the practical, realworld applicability of the model, allowing users to get instant price predictions.

TECHNOLOGY AND LOGIC

The web application was built using Gradio, a user-friendly Python library designed for creating machine learning demos. The core logic of the app.py script is as follows:

1. **Model Training**: When the application first starts, it loads the complete dataset, applies the same advanced feature engineering, and trains the final GradientBoostingRegressor model on 100% of the data. This trained model is then held in memory.

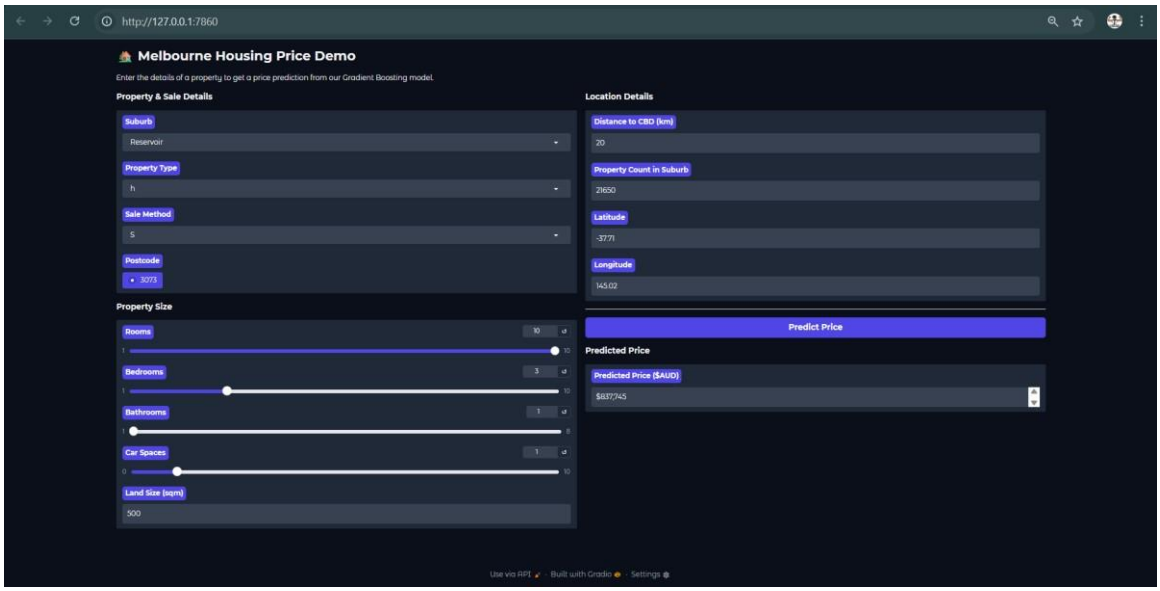
2. **User Interface**: A Gradio Blocks interface is created with a series of dropdowns, sliders, and number inputs, allowing a user to enter the features of a property. A key feature of the interface is the interactive postcode selection, where the available postcode choices automatically update based on the selected suburb.

3. **Prediction Function**: When the user clicks the "Predict" button, a prediction function is triggered. This function takes the user's inputs, organizes them

into a DataFrame, applies the exact same feature engineering and preprocessing steps, and feeds the final, clean data into the trained model pipeline.

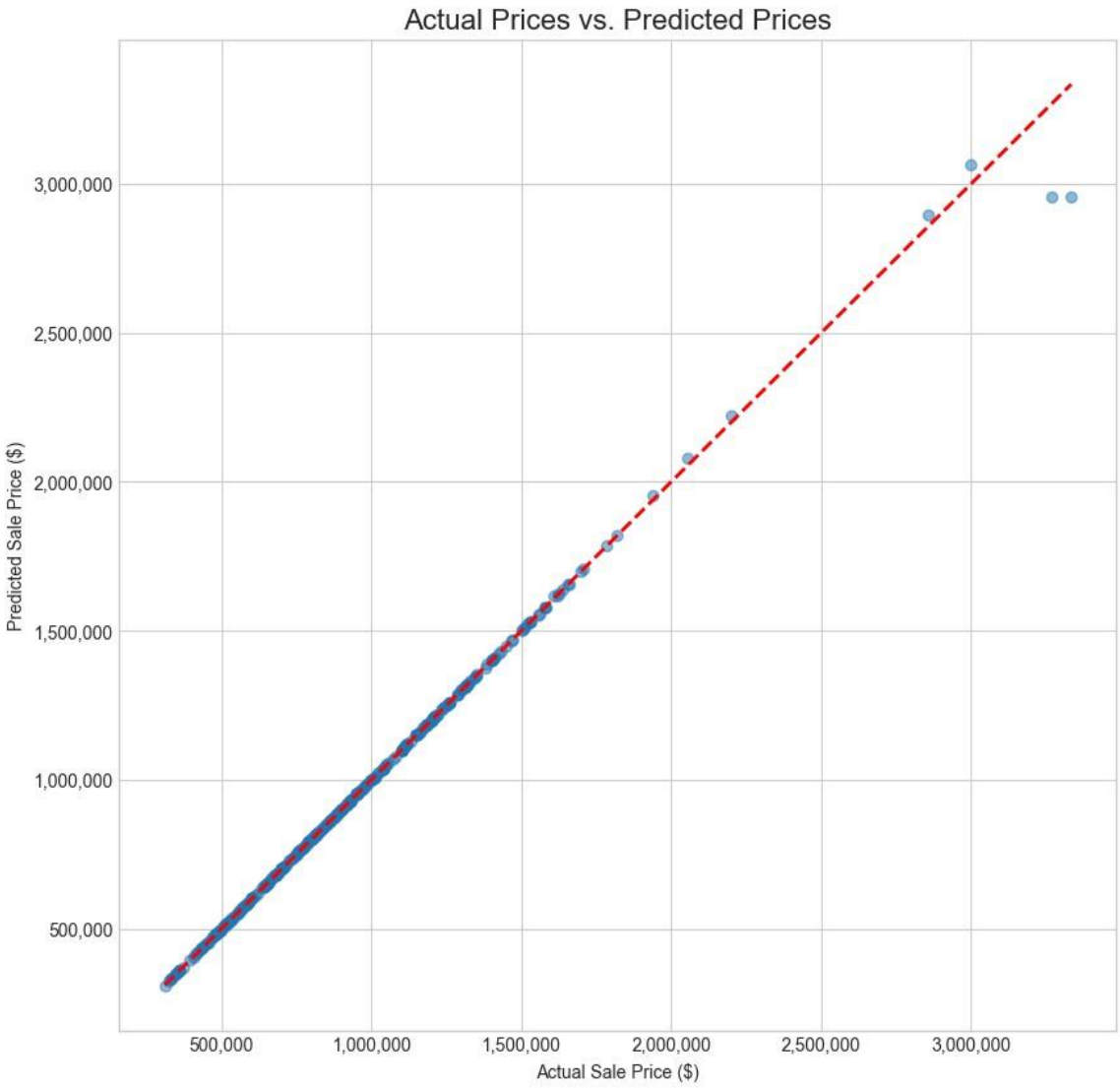
4. **Output**: The model returns a prediction on the log scale, which the function immediately converts back into a real dollar value using np.expm1(). This final price is then formatted and displayed to the user.

This deployment successfully transforms a complex machine learning pipeline into an accessible and practical tool for estimating Melbourne housing prices.



FINAL MODEL VALIDATION

Before deployment, a final visual validation was performed. The best model was trained on a training set and used to make predictions on a held-out test set. The resulting scatter plot of Actual Prices vs. Predicted Prices showed the data points forming a tight, straight line along the diagonal, providing a clear visual confirmation of the model's high accuracy and reliability.



Video Demo

A video demonstration of the final, deployed Gradio web application has been prepared to showcase its functionality and ease of use. The video walks through the process of entering property details into the interface and receiving a real-time price prediction from the trained machine learning model.

Task 8.2D Video