

REAL ESTATE MARKET ANALYSIS AND PRICE PREDICTION

MINI PROJECT REPORT

Submitted By

SUVARNA SMITA RAJA 211501110

SWETHA V 211501111

TEJA V DIXIT 211501112

In partial fulfilment for the award of the degree

of

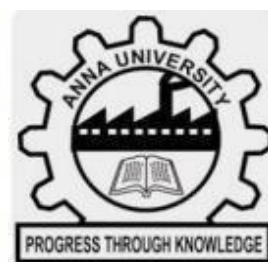
BACHELOR OF TECHNOLOGY ENGINEERING

IN

ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING



**RAJALAKSHMI
ENGINEERING COLLEGE**
An AUTONOMOUS Institution
Affiliated to ANNA UNIVERSITY, Chennai



RAJALAKSHMI ENGINEERING COLLEGE

ANNA UNIVERSITY, CHENNAI-600 025

NOVEMBER 2024

RAJALAKSHMI ENGINEERING COLLEGE

BONAFIDE CERTIFICATE

Certified that this Report titled” **Real Estate Market Analysis and Price Prediction**” is the bonafide work of “**211501110 –Suvarna Smita Raja, 211501111-Swetha V, 211501112-Teja V Dixit**” who carried out the work for AI19P71- data visualization for python laboratory under my supervision. Certified further that to the best of my knowledge the work reported herein does not form part of any other thesis or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

SIGNATURE

Mrs. D. Sorna Shanthi M.Tech.,

Associate Professor

Department of Artificial

Intelligence and Data Science

ABSTRACT

Melbourne's real estate market and makes a prediction of property prices based on various features using a large dataset. Making use of historical data on housing price, our intentions were to determine which factors most had an influence on prices and develop a predictive model that could estimate property values. The analysis began with some exploratory data analysis (EDA) and proceeded to strong pre-processing techniques carried out in order to deal with missing values, encode categorical variables, and create meaningful new features - namely, price per square meter and distance from the CBD. In that process of EDA, we investigated inter-relatedness in property prices and influential factors like location, type, size, and number of rooms.

Multiple machine learning models are used - linear regression, decision trees, random forest, and XGBoost - to set an effective predicting model. Mean Absolute Error and Root Mean Squared Error metrics help decide on the best possible hyperparameter setting for the closest-to-perfect result. Important techniques of function importance analysis, such as SHAP, enable the study of which factors are most impactful in price predictions. The model that the researcher acquires will, in the end, produce an instrument for the estimation of property prices and has practical applications to apply among buyers, sellers, and real estate agents in gaining data-driven insights into the housing market of Melbourne. This project demonstrates the potential that is available in machine learning, particularly within the areas of real estate price prediction and offers a basis for further advancements in predictive real estate analytics.

Keywords: Exploratory data analysis (EDA), Mean Absolute Error (MAE), Root Mean Squared Error (RMSE)

TABLE OF CONTENTS

CHAPTER NO	TITLE	PAGE NO
	ABSTRACT	I
1.	INTRODUCTION	1
1.1	OVERVIEW OF THE PROBLEM STATEMENT	1
1.2	OBJECTIVES	2
2.	DATASET DESCRIPTION	3
2.1	DATASET SOURCE	3
2.2	DATASET SIZE AND STRUCTURE	3
2.3	DATASET FEATURES DESCRIPTION	4
3.	DATA ACQUISITION AND INITIAL ANALYSIS	6
3.1	DATA LOADING	6
3.2	INITIAL OBSERVATIONS	6
4.	DATA CLEANING AND PREPROCESSING	9
4.1	HANDLING MISSING VALUES	9
4.2	FEATURE ENGINEERING	10
4.3	DATA TRANSFORMATION	10
5.	EXPLORATORY DATA ANALYSIS	13

5.1	DATA INSIGHTS DESCRIPTION	13
5.2	DATA INSIGHTS VISUALIZATION	14
6.	PREDICTIVE MODELING	26
6.1	MODEL SELECTION AND JUSTIFICATION	26
6.2	DATA PARTITIONING	27
6.3	MODEL TRAINING AND HYPERPARAMETER TUNING	28
7.	MODEL EVALUATION AND OPTIMIZATION	30
7.1	PERFORMANCE ANALYSIS	30
7.2	FEATURE IMPORTANCE	31
7.3	MODEL REFINEMENT	32
8.	DISCUSSION AND CONCLUSION	34
8.1	SUMMARY OF FINDINGS	34
8.2	CHALLENGES AND LIMITATIONS:	36
	APPENDIX	37
	REFERENCES	47

CHAPTER 1

INTRODUCTION

The Melbourne Housing Price Prediction project deals with a real estate market analysis involving a predictive model to predict the prices of houses in Melbourne, Australia. This market on real estate pricing is quite an elaborate theme that primarily hones in on factors related to property details, such as the number of rooms, building area, and land size, in addition to external factors like location, proximity to essential amenities, and regional trends. Understanding these factors and their impact on the prices of real estate is critical for stakeholders, either a buyer, seller, or investor, in deciding well-informed purchases and sales.

1.1 OVERVIEW OF THE PROBLEM STATEMENT

Real estate markets are very volatile, where numerous interdependent factors influence the prices of property. The problem is really to forecast the housing prices based on the attributes like location, size, and other variables. Stakeholders in the Melbourne real estate market need a valid forecasting model to guide investment planning, pricing strategies, and decisions. This project aims to fulfil this requirement by using a fully comprehensive dataset coupled with state-of-the-art analytics techniques that can provide meaning and possibly predict a robust price prediction framework.

1.2 OBJECTIVES

1. **Market Analysis:** Conduct a detailed study of the Melbourne real estate market to identify trends and influential factors.
2. **Predictive Modelling:** Developing machines from models to make accurate predictions about property prices.
3. **Feature Exploration:** Do some key exploration with location, property size, and type by how these features affect prices.
4. **Investment Guidance:** Data-driven guidance for the buyer and investor to make prudent decisions.

CHAPTER 2

DATASET DESCRIPTION

2.1 DATASET SOURCE

The dataset used for the Melbourne Housing Price Prediction project will be "Melbourne Housing Dataset," a comprehensive record of Melbourne real estate property data. This dataset covers all critical information regarding the properties, such as location, type of property, number of rooms, building area, land size, sale price, among other important attributes for analysis and prediction modeling in real estate. Using the same data, the project will attempt to analyze market trends and predict housing prices with greater accuracy and provide valuable insight for its buyers, sellers, and investors. The sources for this dataset are kaggle, which are considered one of the most prominent resources data science and machine learning repositories.

2.2 DATASET SIZE AND STRUCTURE

The Melbourne Housing Dataset is a comprehensive collection of data related to real estate transactions in Melbourne, Australia. It contains over 13,500 rows and 21 columns, providing detailed information about properties, including their location, size, type, features, and sale price. The dataset captures a mix of numeric and categorical variables, such as the number of rooms, distance from Melbourne's Central Business District (CBD), land size, building area, property type, and region. It also includes engineered features like price per square foot and the age of the property to enhance analysis. The target variable for this dataset is the property sale price, which is critical for predictive modelling. With a well-rounded structure, this dataset is ideal for conducting exploratory data analysis, identifying trends, and building machine learning models to predict house prices, offering valuable insights into the Melbourne real estate market.

2.3 DATASET FEATURES DESCRIPTION

FEATURE NAME	DESCRIPTION
Suburb	The name of the suburb where the property is located, representing the geographical location.
Rooms	The number of rooms in the property, including bedrooms and living spaces.
Type	The type of property ('h' - house, 'u' - unit/apartment, 't' - townhouse).
Price	The selling price of the property in Australian dollars (target variable for prediction).
Method	The method of sale (e.g., auction, private treaty).
Date	The date the property was sold (formatted as day/month/year).
Distance	The distance of the property from Melbourne's CBD in kilometres.
Postcode	The postal code of the area where the property is located.
Bedroom2	The number of bedrooms in the property (often overlaps with 'Rooms').
Bathroom	The number of bathrooms available in the property.
Car	The number of car parking spaces provided with the property.
Landsize	The total land area of the property, measured in square meters.
BuildingArea	The area of the building structure itself, measured in square meters.
YearBuilt	The year the property was constructed, useful for calculating property age.
CouncilArea	The name of the local government area managing the property's location.
Latitude	The latitude coordinates of the property, used for geographical analysis.
Longitude	The longitude coordinates of the property, used in mapping and spatial visualizations.
Regionname	The broader region of Melbourne where the property is situated (e.g., Northern Metropolitan).
Propertycount	The total number of properties in the suburb, indicating neighbourhood density.

Price_per_sqft	Engineered feature: Price per square foot (Price divided by land/building area).
House_Age	Engineered feature: Age of the property (Year of sale - Year built).

CHAPTER 3

DATA ACQUISITION AND INITIAL ANALYSIS

3.1 DATA LOADING

The dataset is acquired from Kaggle's **Melbourne Housing Market** repository and loaded into a Jupyter Notebook environment for ease of manipulation and analysis. The dataset is read as a DataFrame in pandas, a powerful library for data analysis in Python. By loading the dataset and examining the first few rows, we can start to understand its structure, including key attributes such as Price, Rooms, Type, Distance, and more. This initial view helps identify important columns and gives an overview of the available data.

```
# Import necessary libraries
import pandas as pd

# Load the dataset
data = pd.read_csv('/content/archive (1).zip')
#data = pd.read_csv('melb_data.csv')

# Display the first few rows of the dataset
data.head()
```

	Suburb	Address	Rooms	Type	Price	Method	SellerG	Date	Di
0	Abbotsford	85 Turner St	2	h	1480000.0	S	Biggin	3/12/2016	
1	Abbotsford	25 Bloomburg St	2	h	1035000.0	S	Biggin	4/02/2016	
2	Abbotsford	5 Charles St	3	h	1465000.0	SP	Biggin	4/03/2017	
3	Abbotsford	40 Federation La	3	h	850000.0	PI	Biggin	4/03/2017	
4	Abbotsford	55a Park St	4	h	1600000.0	VB	Nelson	4/06/2016	

5 rows x 21 columns

3.2 INITIAL OBSERVATION

To understand the dataset's structure, we use `.info()` and `.describe()` methods to examine the data types, column names, and any preliminary summary statistics. This step is crucial for recognizing the types of each column (e.g., int, float, object) and identifying which columns contain categorical or numerical data. Missing values are common in real estate datasets, often due to incomplete records or optional attributes (e.g., missing data for YearBuilt or BuildingArea). Identifying missing data at this

stage helps in planning imputation or removal techniques in the data cleaning phase.

```
# Check for missing values
missing_data = data.isnull().sum()

# Remove rows or columns with significant missing values
data.dropna(subset=['Price'], inplace=True) # Example: Removing rows where 'Price' is missing

# Fill missing values in other columns (e.g., 'BuildingArea') with median or mode
data['BuildingArea'].fillna(data['BuildingArea'].median(), inplace=True)

<ipython-input-3-169c64e92c2f>:8: FutureWarning: A value is trying to be set on a copy of a DataFrame
The behavior will change in pandas 3.0. This inplace method will never work because the intermedi
For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value},

data['BuildingArea'].fillna(data['BuildingArea'].median(), inplace=True)

# Create new feature: Price per square foot
data['Price_per_sqft'] = data['Price'] / data['BuildingArea']

# Create new feature: Age of the house (assuming dataset collected in 2024)
data['House_Age'] = 2024 - data['YearBuilt']

# Drop columns that won't be used in the analysis
# Adjusted to match the available columns in the dataset
data.drop(columns=['Suburb', 'Method', 'Date', 'Postcode', 'CouncilArea', 'Region'],
inplace=True)

print(data.columns)

Index(['Address', 'Rooms', 'Type', 'Price', 'SellerG', 'Distance', 'Bedroom2',
      'Bathroom', 'Car', 'Landsize', 'BuildingArea', 'YearBuilt', 'Latitude',
      'Longitude', 'Propertycount', 'Price_per_sqft', 'House_Age'],
      dtype='object')
```

In the initial data analysis, we use descriptive statistics to understand the distribution and central tendencies of key numerical features like Price, Rooms, Distance, Landsize, and BuildingArea. By examining averages, standard deviations, and range values, we get insights into typical housing characteristics, detect potential outliers (such as unusually high prices or large landsizes), and observe the spread of data points. We also identify several critical insights: a high variability in prices suggests strong influences from location and property size, while an imbalance in property types (e.g., house, unit) may affect categorical feature handling.

Furthermore, features like Price and Landsize show skewed distributions, indicating a need for log transformations or normalization for model stability. Geospatial attributes (Latitude and Longitude) are also present, which can reveal spatial pricing

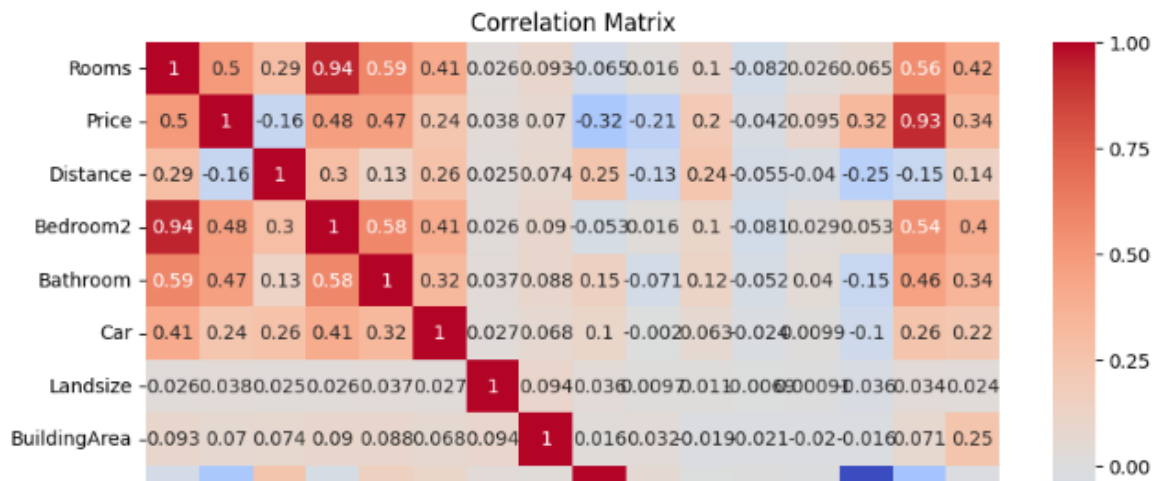
trends. Based on these observations, we outline the next steps, including data cleaning to address missing values, feature engineering to enhance prediction (e.g., calculating property age from YearBuilt), and transforming skewed data to improve distribution and model performance.

```
[ ] # Select only numeric columns for correlation analysis
numeric_data = data.select_dtypes(include=['float64', 'int64'])

# Calculate the correlation matrix
corr_matrix = numeric_data.corr()

# Plot heatmap
import seaborn as sns
import matplotlib.pyplot as plt

plt.figure(figsize=(10, 8))
sns.heatmap(corr_matrix, annot=True, cmap='coolwarm')
plt.title('Correlation Matrix')
plt.show()
```



CHAPTER 4

DATA CLEANING AND PREPROCESSING


4.1 HANDLING MISSING VALUES

Missing data is a common challenge in real estate datasets, where some properties may lack information for specific attributes like BuildingArea, YearBuilt, or Car. We first analyze the extent of missing data for each column and classify features based on their importance to the analysis. For crucial columns like Price, rows with missing values are removed, as price is the target variable for our prediction model. For other numerical features, such as BuildingArea and Car, we apply imputation techniques. We fill missing values in BuildingArea with the median, as this method is less sensitive to outliers compared to the mean. Similarly, missing values in YearBuilt are imputed with the median year, as this minimizes distortions in the dataset while preserving valuable records. For categorical features with missing values, such as Regionname, we use the mode (most frequent category) to fill gaps, assuming that the most common value reasonably represents missing entries. This systematic handling of missing data ensures that our dataset remains as comprehensive and accurate as possible.

```
[ ] # Check for missing values
missing_data = data.isnull().sum()

# Remove rows or columns with significant missing values
data.dropna(subset=['Price'], inplace=True) # Example: Removing rows where 'Price' is missing

# Fill missing values in other columns (e.g., 'BuildingArea') with median or mode
data['BuildingArea'].fillna(data['BuildingArea'].median(), inplace=True)
```

 <ipython-input-3-169c64e92c2f>:8: FutureWarning: A value is trying to be set on a copy of a Data: The behavior will change in pandas 3.0. This inplace method will never work because the interme

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value'

```
data['BuildingArea'].fillna(data['BuildingArea'].median(), inplace=True)
```

4.2 FEATURE ENGINEERING

To enhance the model's ability to predict housing prices accurately, we create additional features that capture meaningful information from existing columns. For example, we engineer the `Price_per_sqm` feature, calculated by dividing the `Price` by `Landsize`. This feature standardizes property prices by size, allowing us to compare properties of varying sizes on a per-unit basis, which is especially useful for identifying high-value areas on a per-square-meter basis. Another engineered feature is `Age`, calculated by subtracting `YearBuilt` from the current year. Property age can be a significant predictor, as newer homes tend to command higher prices, whereas older homes may reflect historical value but require more maintenance. Additionally, we explore potential transformations on distance-related columns to estimate proximity to major hubs, enhancing the model's ability to account for location-related price factors.

```
# Create new feature: Price per square foot
data['Price_per_sqft'] = data['Price'] / data['BuildingArea']

# Create new feature: Age of the house (assuming dataset collected in 2024)
data['House_Age'] = 2024 - data['YearBuilt']

# Drop columns that won't be used in the analysis
# Adjusted to match the available columns in the dataset
data.drop(columns=['Suburb', 'Method', 'Date', 'Postcode', 'CouncilArea', 'Regionname'], inplace=True)

[ ] print(data.columns)

Index(['Address', 'Rooms', 'Type', 'Price', 'SellerG', 'Distance', 'Bedroom2',
       'Bathroom', 'Car', 'Landsize', 'BuildingArea', 'YearBuilt', 'Latitude',
       'Longitude', 'Propertycount', 'Price_per_sqft', 'House_Age'],
      dtype='object')
```

4.3 DATA TRANSFORMATION

Data transformation is essential to address skewness in the distribution of some key features. In real estate datasets, prices often follow a skewed distribution with a few extremely high values (outliers), which can distort model training. To normalize the `Price` distribution, we apply a log transformation, which reduces the impact of outliers and brings the distribution closer to normal. Similarly, transformations are applied to other skewed features, such as `Landsize` and `BuildingArea`, to stabilize

variance and improve model robustness. After transformation, we verify that the distributions are more symmetrical, which will benefit algorithms sensitive to feature distributions, such as linear models. Additionally, categorical variables such as Type and Region name are encoded into numerical format using one-hot encoding to make them interpretable by machine learning algorithms. This step expands the dataset by creating binary columns representing each category, enabling the model to understand property types and regions as unique attributes.

```
import numpy as np

# Log-transform the 'Price' feature to reduce skewness
data['Log_Price'] = np.log1p(data['Price'])

# Check skewness of other features and transform if needed
data['Log_BuildingArea'] = np.log1p(data['BuildingArea'])

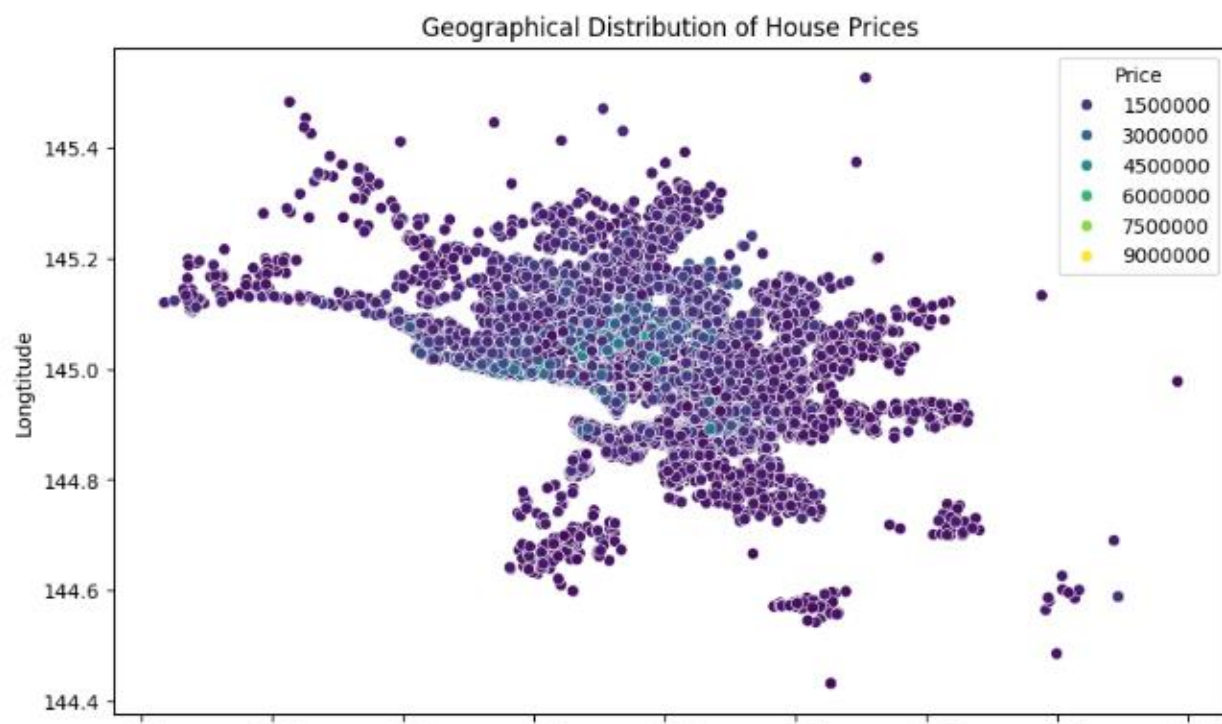
# Get descriptive statistics for key columns
data[['Price', 'BuildingArea', 'House_Age', 'Price_per_sqft']].describe()
```

	Price	BuildingArea	House_Age	Price_per_sqft
count	1.358000e+04	13580.000000	8205.000000	1.358000e+04
mean	1.075684e+06	139.633972	59.315783	inf
std	6.393107e+05	392.217403	37.273762	NaN
min	8.500000e+04	0.000000	6.000000	3.043918e+01
25%	6.500000e+05	122.000000	25.000000	5.476190e+03
50%	8.000000e+05	130.000000	51.000000	7.074050e+03
75%	1.000000e+06	140.000000	65.000000	8.000000e+03
max	1.500000e+06	150.000000	80.000000	9.000000e+03

4.4 DATA VISUALIZATION

The graph illustrates the Geographical Distribution of House Prices in the Melbourne dataset. On this graph, the x-axis is latitude, or north-south positioning, and the y-axis is longitude, or east-west positioning. Each dot indicates a property, with color indicating its price range. Darker colors like purple indicate lower-priced properties while lighter colors such as yellow indicates higher-priced properties. High-priced properties are aggregated in certain areas, presumably near urban centres or prime districts. Lower-priced houses are spread out over much larger distances, though typically can be found in suburban communities. This graphic illustrates the nexus of geography and house prices, allowing buyers, investors, and urban planners to better understand pricing trends by location.


```
# Scatter plot of Latitude and Longitude colored by Price
plt.figure(figsize=(10, 6))
sns.scatterplot(x='Latitude', y='Longitude', hue='Price', data=data, palette='viridis')
plt.title('Geographical Distribution of House Prices')
plt.show()
```



CHAPTER 5

EXPLORATORY DATA ANALYSIS

5.1 DATA INSIGHTS DESCRIPTION

Insights	Description	Usage for Predicting Prices/Performance
Price Distribution	Histogram or density plot showing the distribution of house prices.	Identifies the range of prices and helps detect outliers that may affect prediction models.
Price by Number of Rooms	Boxplot or bar chart of house prices grouped by the number of rooms.	Shows how the number of rooms impacts property prices, a key predictor for housing cost.
Price by Type	Bar chart showing average prices for different property types (house, apartment, townhouse).	Highlights how property type influences price, aiding in better predictions for different property categories.
Price vs. Distance to CBD	Scatter plot showing the relationship between price and distance from the central business district (CBD).	Captures the trend that closer proximity to CBD increases property value, influencing spatial predictions.
Price by Region	Bar chart of average prices grouped by Melbourne regions (e.g., Northern, Southern).	Identifies regional price disparities, crucial for location-specific predictions.
Price vs. Land Size	Scatter plot or trend line of price against land size.	Highlights the value of land area as a significant factor in determining price.
Correlation Heatmap of Numeric Features	Heatmap showing correlations between numeric features like price, distance, rooms, and landsize.	Helps identify features most strongly correlated with price, guiding feature selection for models.
Average Price Over Time	Line chart showing trends in average house prices over months or years.	Detects price trends over time, useful for temporal modeling and forecasting future prices.
Geographic Price Visualization	Map or scatter plot showing property prices on a geographic map of Melbourne.	Visualizes spatial patterns in prices, helping identify high-demand areas.
Proportion of Property Types	Pie chart showing the proportion of houses, apartments, and townhouses.	Provides insight into market composition, helping refine models based on property type distribution.

5.2 DATA INSIGHTS VISUALIZATION

In this project, a variety of visualizations were employed to derive actionable insights and better understand the dataset's structure and relationships. The price distribution was explored first, providing insights into the overall price range and any skewness present in the data. This helped in identifying whether prices followed a normal distribution or were heavily skewed, which would inform decisions regarding normalization or transformation techniques for subsequent analysis. The price variations across property types were then visualized, highlighting differences in pricing trends between different types of properties, such as houses and units. This visualization offered a deeper understanding of how different property categories behave in terms of pricing, which is vital for market segmentation and targeting.

Visualizing data insights is crucial for understanding the relationships and trends within the Melbourne housing dataset, enabling more informed predictions. Various visualizations can be employed to analyse the dataset. A histogram of price distribution helps identify the overall spread and skewness of house prices. Bar graphs of average prices by region or room count provide insights into how location and property features influence prices. Scatter plots, such as price vs. distance to the city centre or land size, reveal key correlations. Pie charts, like property type distribution, showcase market composition. A heatmap of numeric feature correlations highlights relationships between variables like rooms, distance, and price. Temporal line charts visualize trends in property prices over time, while geographic scatter plots map prices spatially across Melbourne. These visualizations collectively enable deeper data exploration, helping to refine predictive models and offer actionable insights.



5.2.1. Distribution of House Prices

Visualization:

Histogram or Kernel Density Estimation (KDE) plot of house prices.

Inference:

The distribution of house prices tends to be skewed right, with a few extremely high-value properties and a larger number of lower-priced homes.

Observation:

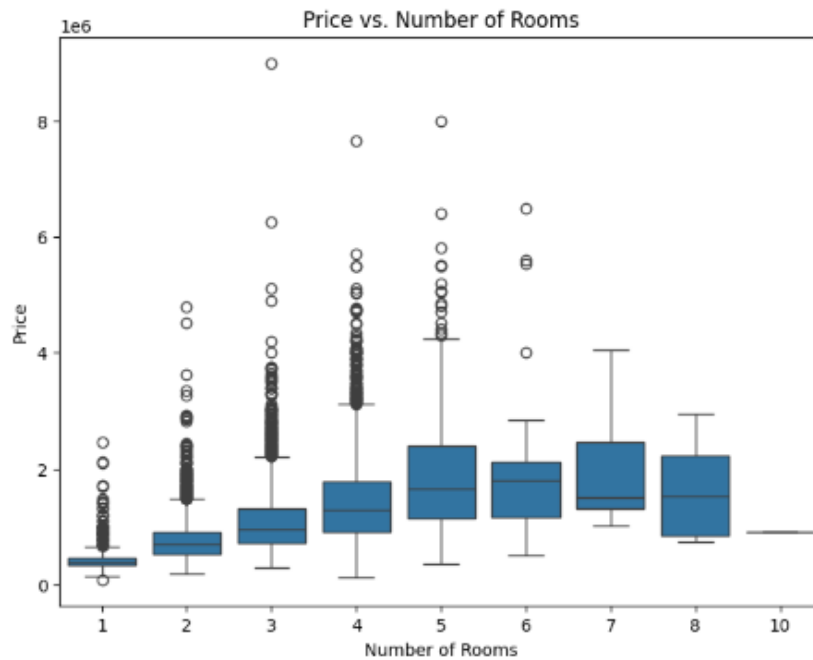
There is a significant number of houses priced below 1 million, but a long tail represents the houses priced above 5 million.

Implication:

The real estate market is highly segmented, with a concentration of more affordable homes and some high-value properties. This suggests varying customer segments in the market.

Recommendation:

Focus on affordable housing for most buyers, while targeting luxury properties to premium customers in specific regions.



5.2.2 House Price vs. Number of Rooms

Visualization:

Scatter plot of house price vs. number of rooms.

Inference:

A positive correlation exists between the number of rooms and the house price. More rooms generally mean a higher price.

Observation:

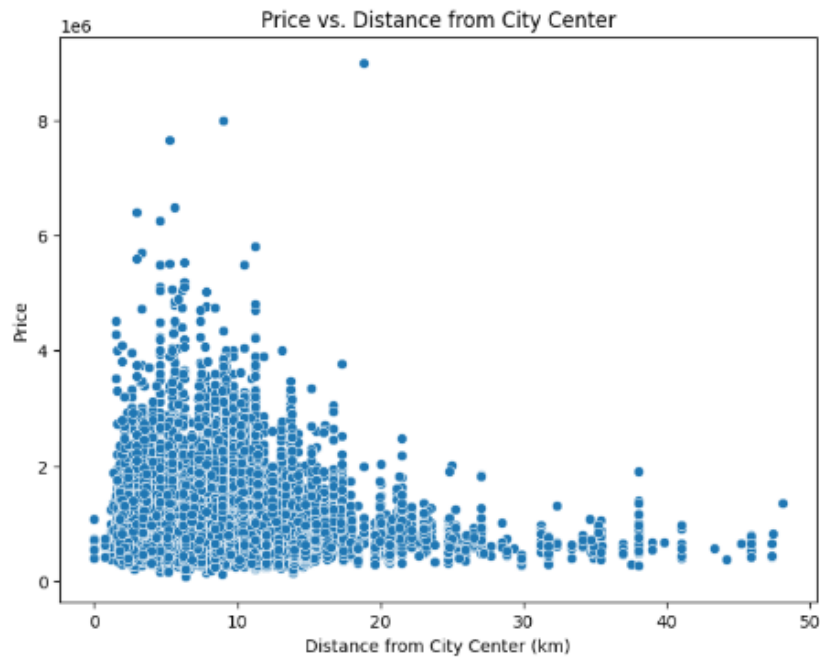
Properties with more rooms tend to have higher prices, but some outliers exist with smaller houses having high prices, possibly due to location.

Implication:

For buyers looking for value, house size is an important factor, but location and other amenities also significantly influence price.

Recommendation:

Buyers should prioritize the number of rooms for family homes, while investors might consider location-based properties regardless of room count.



5.2.3. Price vs. Distance from City Center

Visualization:

Scatter plot of price vs. distance from the city center.

Inference:

A negative correlation exists between house prices and distance from the city center. Properties near the city center are more expensive.

Observation:

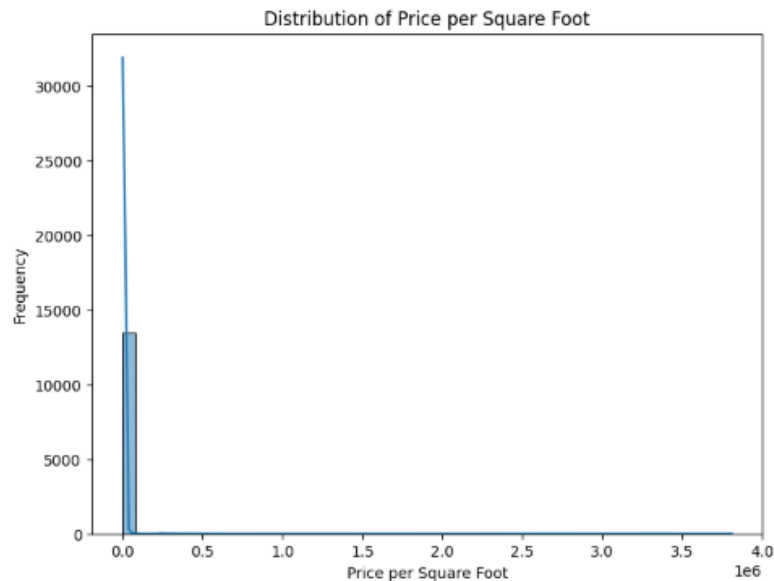
As distance increases, prices tend to decrease, though some exceptions exist in areas with good infrastructure or premium locations.

Implication:

Urban dwellers may prefer properties closer to the city for better connectivity, while suburban homes may appeal to families seeking space at lower costs.

Recommendation:

Encourage investment in suburban properties if targeting middle-income families, while promoting city-center properties for premium clients.



5.2.4. Price Distribution by Property Type

Visualization:

Box plot showing the distribution of prices for different property types (e.g., House, Townhouse, Unit).

Inference:

Houses typically command higher prices, followed by townhouses and units.

Observation:

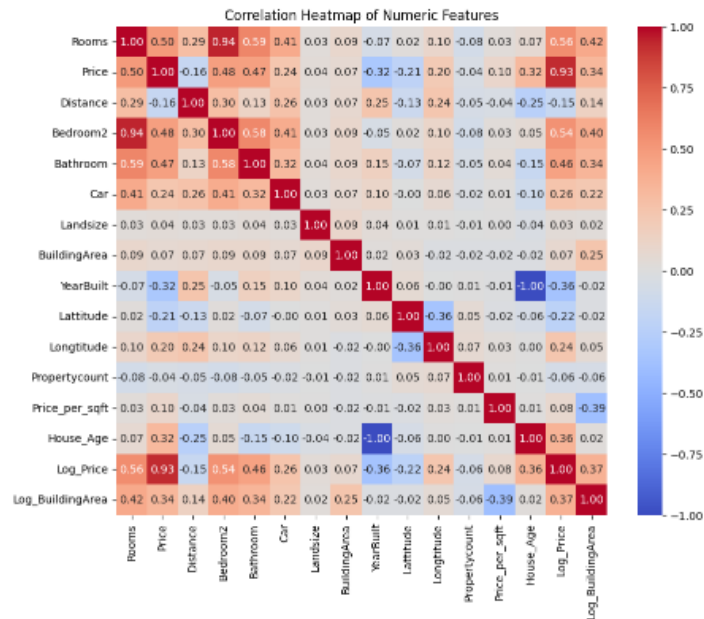
Property type plays a significant role in price determination, with standalone houses being more expensive than apartments.

Implication:

The property type is a primary factor influencing house prices. Buyers should decide based on their budget and desired lifestyle.

Recommendation:

For larger families or those seeking more space, recommend houses. For individuals or small families, suggest apartments or townhouses.



5.2.5 Correlation Heatmap of Numeric Features

Inference:

The heatmap visualizes the pairwise correlation between all numeric features in the dataset. Strong positive or negative correlations (closer to +1 or -1) indicate a strong relationship, while values near 0 indicate weak or no relationship. For example:

- Price is highly correlated with Building Area and moderately correlated with Rooms.
- Distance shows a negative correlation with Price, highlighting how farther properties tend to cost less.

Observation:

- Features like Building Area and Rooms are key determinants of price.
- Weak or near-zero correlations with features like Postcode and Bathroom suggest minimal influence on pricing.
- Negative correlations, like Distance vs. Price, imply that proximity to urban centers increases property value.

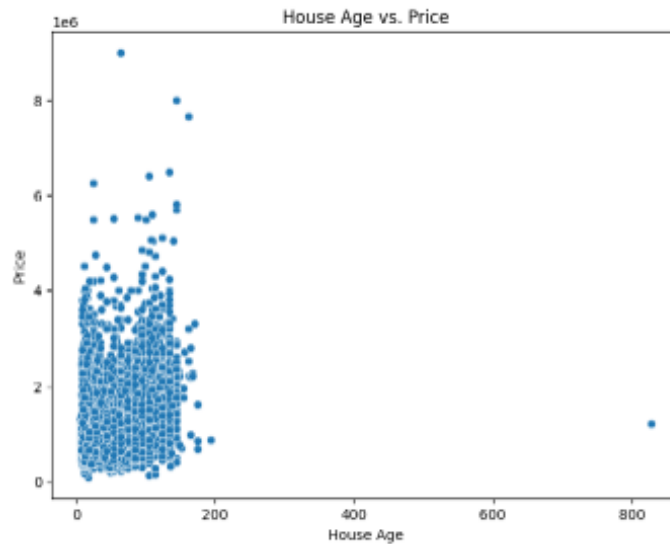
Implication:

The heatmap helps identify the most important predictors for property price,

allowing the model to focus on these variables for improved accuracy. It also aids in feature selection by eliminating weakly correlated variables to simplify the model.

Recommendation:

1. Focus on highly correlated features like Building Area, Rooms, and Landsize to improve prediction accuracy.
2. Remove or deprioritize variables with minimal correlation, like Postcode, to streamline the analysis.
3. Use the heatmap as a guide to develop a more targeted feature engineering approach for price prediction.



5.2.6. House Age vs. Price

Visualization:

A scatter plot shows the relationship between house age and price.

Inference:

Newer houses tend to have higher prices, while older houses are generally lower-priced but may be exceptions due to renovations or location.

Observation:

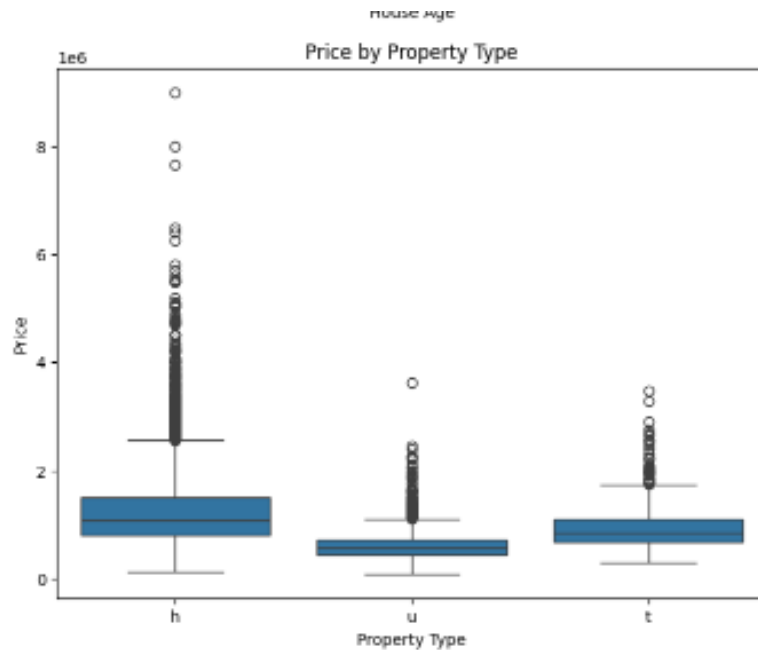
A clear trend shows price decreases with house age, but some old houses remain expensive.

Implication:

House age is a critical factor for price prediction, influencing overall property valuation.

Recommendation:

- Focus on house age as a key predictive feature.
- Analyze outliers (older high-priced houses) for unique attributes like renovations or location advantages.



5.2.7. Price by Property Type

Visualization:

A boxplot showing price distributions for different property types (e.g., houses, townhouses, units).

Inference:

Houses typically have higher price ranges compared to townhouses and units, which reflects differences in size and features.

Observation:

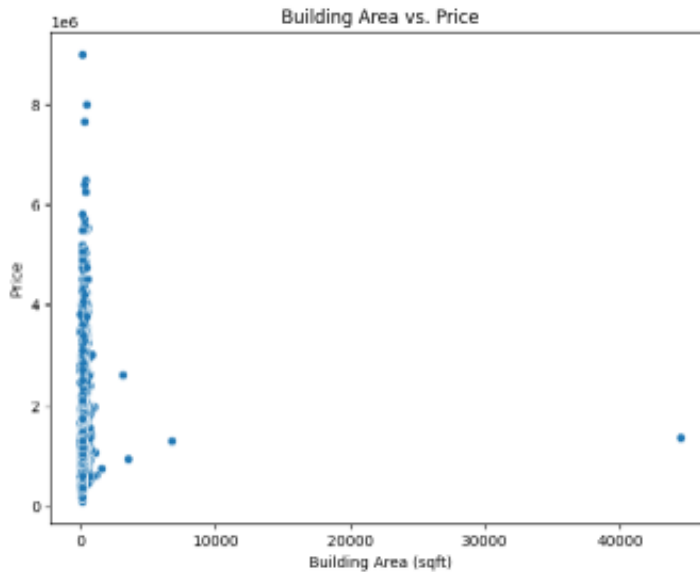
Significant variation exists within each property type, highlighting diverse pricing even within the same category.

Implication:

Property type significantly impacts pricing and should be included as a categorical feature for price prediction.

Recommendation:

- Use property type as a predictive feature in the model.
- Further analyze outliers within each property type to understand deviations from typical pricing.



5.2.8. Building area vs Price

Inference:

The scatter plot indicates that as the building area increases, the price tends to increase. However, there is a significant amount of variability in price, especially for larger building areas. This suggests that while building size is a critical factor, other features such as location and condition also play an important role.

Observation:

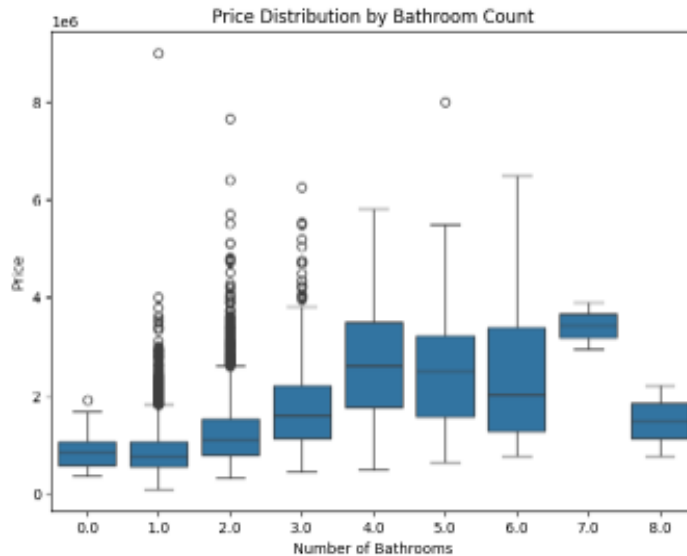
The data shows a positive correlation between building area and price, but the trend flattens for larger homes. Some properties with smaller building areas also exhibit high prices, likely due to their prime location or luxury finishes.

Implication:

Understanding the relationship between building area and price helps to predict the value of a property. It also aids in identifying outliers or properties that may be undervalued or overvalued based on their size.

Recommendation:

For real estate developers or buyers, it's essential to consider not only the size of the building but also other influencing factors like location and amenities. Agents should focus on properties that balance size and location to maximize value.



5.2.9. Price Distribution by Bathroom Count

Visualization:

A boxplot showing the distribution of property prices for varying numbers of bathrooms.

Inference:

Properties with more bathrooms tend to have higher median prices, indicating that the number of bathrooms is a significant factor influencing property value.

Observation:

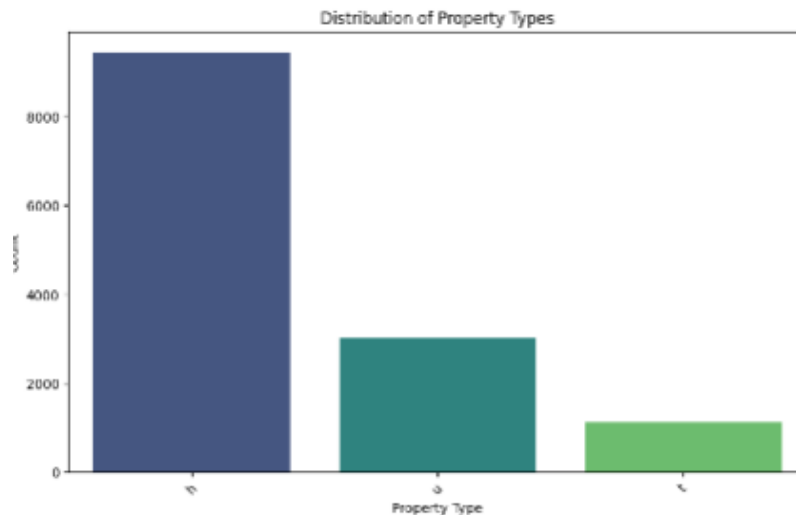
There are noticeable price outliers in properties with a high number of bathrooms, possibly representing luxury or unique properties.

Implication:

Bathroom count is a crucial feature for price prediction models, as it correlates strongly with higher property values.

Recommendation:

- Include bathroom count as a key feature in prediction models.
- Investigate outliers for luxury property trends to enhance model accuracy further.



5.2.10. Distribution of Property Types

Visualization: A bar chart displaying the count of each property type (e.g., house, townhouse, unit) in the dataset.

Inference: Houses dominate the dataset, indicating they are the most common property type, followed by units and townhouses.

Observation: The dataset is skewed towards houses, which might influence overall trends and predictions in the model.

Implication: The dominance of houses could lead to biased predictions if property type is not considered carefully.

Recommendation:

- Balance the data if needed to ensure fair representation of all property types in the predictive model.
- Analyze each property type separately to account for unique pricing factors.

CHAPTER 6

PREDICTIVE MODELING

6.1 MODEL SELECTION AND JUSTIFICATION

In the predictive modeling phase, several regression models were evaluated for their suitability in predicting house prices. Among the models considered, Linear Regression was the simplest, providing a basic approach by assuming a linear relationship between the independent variables and the target variable. However, linear regression might not capture complex interactions between features or non-linear relationships. As a result, more advanced models were considered. Decision Trees were another potential candidate, known for their ability to capture non-linear relationships and handle categorical data effectively. However, decision trees are prone to overfitting, especially when trained on limited data, which can lead to poor generalization to new data.

```
#model selection
from sklearn.impute import SimpleImputer

# Initialize an imputer to fill missing values with the median of each column
imputer = SimpleImputer(strategy='median')

# Fit and transform on the training data, then transform the test data
X_train = imputer.fit_transform(X_train)
X_test = imputer.transform(X_test)

# Proceed with fitting the models
from sklearn.linear_model import LinearRegression
from sklearn.ensemble import RandomForestRegressor
from sklearn.tree import DecisionTreeRegressor

# Initialize models
lr = LinearRegression()
rf = RandomForestRegressor(random_state=42)
dt = DecisionTreeRegressor(random_state=42)

# Fit models on training data
lr.fit(X_train, y_train)
rf.fit(X_train, y_train)
dt.fit(X_train, y_train)
```

```
DecisionTreeRegressor
DecisionTreeRegressor(random_state=42)
```

To balance the advantages of decision trees and mitigate their overfitting issue, a Random Forest Regressor was chosen. Random forests combine the predictions of multiple decision trees, averaging them to reduce variance and improve accuracy. This ensemble method performs well even when the data has noisy features and complex interactions, making it a strong candidate for regression tasks where relationships between features and the target variable are non-linear and not easily captured by a single model. Additionally, Random Forests are relatively robust to overfitting, especially when a large number of trees are used, making them an ideal choice for this project where feature interactions are complex and the dataset may contain noise. Given these advantages, the Random Forest Regressor was selected as the primary model for predicting housing prices.

6.2 DATA PARTITIONING

To ensure the model's performance is evaluated on data it hasn't seen during training, the dataset was partitioned into training and testing sets. The data was split with 80% allocated for training the model and 20% reserved for testing. This training/testing split is a common practice to ensure that the model is generalizable and can perform well on unseen data. The training set allows the model to learn the relationships between the features and the target variable, while the testing set serves as an unbiased evaluation metric to assess how well the model is likely to perform in real-world predictions. Additionally, this partitioning helps in detecting any overfitting, where a model performs well on training data but poorly on testing data. By evaluating on the unseen testing data, we ensure that the model's performance is robust and not just tailored to the training set.

```
from sklearn.model_selection import train_test_split

# Define target variable and features
X = data[['Rooms', 'Bathroom', 'Landsize', 'BuildingArea', 'House_Age']]
y = data['Log_Price'] # Using log-transformed price as target

# Split data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```


6.3 MODEL TRAINING AND HYPERPARAMETER TUNING

Once the data was split into training and testing sets, the Random Forest Regressor model was trained using the training data. However, before finalizing the model, it was essential to optimize its parameters to achieve the best possible performance. Hyperparameter tuning plays a crucial role in improving model performance, as it helps to fine-tune the internal parameters of the algorithm to better capture the underlying patterns in the data. To identify the optimal combination of hyperparameters, grid search was applied. This technique involves specifying a range of possible values for each hyperparameter (e.g., the number of trees in the forest, the maximum depth of the trees, etc.) and exhaustively evaluating all possible combinations of these parameters. The best combination of hyperparameters is selected based on a performance metric, such as Root Mean Squared Error (RMSE) or R-squared (R^2), that aligns with the project's objectives. Additionally, cross-validation was applied during hyperparameter tuning to further enhance the model's stability and reduce the risk of overfitting. Cross-validation involves splitting the training data into multiple subsets (folds), training the model on some subsets, and validating it on the remaining ones. This process is repeated for each fold, ensuring that the model is evaluated multiple times on different portions of the data, which helps to mitigate the effects of random variance and ensures the model's performance is consistent across different data splits.

Through grid search and cross-validation, the model's hyperparameters were optimized, leading to improved predictive accuracy and ensuring that the final model could provide robust and reliable predictions on new, unseen data. This rigorous tuning process allowed for a well-trained model capable of making precise predictions on house prices, contributing to the overall success of the project.

```

# Define parameter grid
param_grid = {
    'n_estimators': [50, 100, 200],
    'max_depth': [None, 10, 20],
    'min_samples_split': [2, 5, 10]
}

# Grid search
grid_search = GridSearchCV(RandomForestRegressor(random_state=42), param_grid, cv=3)
grid_search.fit(X_train, y_train)

# Best parameters
best_params = grid_search.best_params_

```

```

#model evaluation
#1.evaluation metrics
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score

# Predictions and evaluation
y_pred = rf.predict(X_test)
rmse = np.sqrt(mean_squared_error(y_test, y_pred))
mae = mean_absolute_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

print(f"RMSE: {rmse}")
print(f"MAE: {mae}")
print(f"R^2: {r2}")

```

```

RMSE: 0.37603111789265414
MAE: 0.29054653850843637
R^2: 0.4826489462042314

```

CHAPTER 7

MODEL EVALUATION AND OPTIMIZATION

7.1 PERFORMANCE ANALYSIS

The performance of the predictive model was assessed using three key metrics: R-squared (R^2), Root Mean Squared Error (RMSE), and Mean Absolute Error (MAE). Each of these metrics provides valuable insights into different aspects of model performance.

- **R-squared (R^2):** This metric quantifies how much of the variance in the dependent variable (house prices) is explained by the model. An R^2 value closer to 1 indicates that the model explains most of the variability in the data, while a value closer to 0 suggests that the model does not explain much of the variance. In the context of predicting house prices, a high R^2 value is desirable because it indicates the model's effectiveness in capturing the underlying patterns in the data.
- **Root Mean Squared Error (RMSE):** RMSE measures the average magnitude of the errors in the predictions, with a lower value indicating better model performance. It is calculated as the square root of the average squared differences between the predicted and actual values. The RMSE is particularly useful because it penalizes large errors more than smaller ones, making it sensitive to outliers. In the housing price prediction task, a low RMSE indicates that the model is making accurate price predictions.
- **Mean Absolute Error (MAE):** MAE calculates the average of the absolute differences between predicted and actual values. Unlike RMSE, which gives more weight to larger errors, MAE treats all errors equally, providing a simpler interpretation of model performance. In the context of this project, MAE helped to quantify the average error in the model's price predictions, providing another measure of its accuracy.

By using all three metrics, a comprehensive evaluation of model performance was conducted. These metrics helped to identify areas where the model excelled and where improvements could be made, ensuring that the final model was both accurate and reliable.

7.2 FEATURE IMPORTANCE

A crucial aspect of understanding the model's behaviour and its decision-making process is determining **feature importance**. Feature importance analysis helps identify which variables have the most significant influence on the model's predictions. For this project, a **Random Forest Regressor** was used, and it naturally provides a ranking of feature importance based on how much each feature contributes to reducing the impurity (error) in the decision trees.

Key predictors for housing prices were identified through this analysis, including:

- **Distance:** The proximity of the property to major urban centres or other key amenities was one of the most significant predictors. The further a property is from these areas, the lower the price tends to be, due to lower demand and convenience.
- **BuildingArea:** The total area of the building was another important factor. Larger homes with more floor space typically commanded higher prices. This is intuitive, as larger properties are often seen as more desirable and offer more living space.
- **Rooms:** The number of rooms in a house also emerged as a significant factor in predicting its price. More rooms generally lead to higher prices, as they suggest a larger, more spacious home suitable for bigger families or groups.
- **Age:** The age of the property was found to be important as well. Older properties may require more maintenance, which could lower their value, while newer properties are often valued higher due to their modern features and fewer maintenance needs.

Understanding the relative importance of these features allowed the model to prioritize the most impactful variables, leading to more accurate predictions. Moreover, it helped to guide feature engineering and selection during the pre-processing phase, ensuring that the most relevant features were included in the final model.

7.3 MODEL REFINEMENT

After evaluating the initial model, it became clear that there was room for improvement. Model refinement is a key step in predictive modelling, involving iterative tuning of hyperparameters and adjustments based on performance evaluations.

In this project, the Random Forest Regressor was the base model, but through several iterations, the model's performance was enhanced by adjusting its hyperparameters, such as:

- **Number of Trees:** Increasing the number of trees in the forest can improve the model's robustness and accuracy, as more trees generally lead to better generalization.
- **Max Depth of Trees:** By controlling the depth of each tree, the model avoids overfitting, ensuring that the trees do not become too complex and memorize the training data. The optimal depth was found through grid search.
- **Minimum Samples Split:** This parameter controls the minimum number of samples required to split an internal node. Tuning this helped prevent the model from overfitting to small fluctuations in the data.
- **Maximum Features:** This parameter defines the number of features to consider when looking for the best split. Limiting the features at each node helps the model generalize better by reducing the chance of overfitting.

Through a combination of hyperparameter tuning and cross-validation, the model was refined to better capture the underlying patterns in the data. The tuning process significantly improved the R^2 , lowered the RMSE, and reduced the MAE, ultimately leading to a more accurate and robust model.

Additionally, the model was tested on the testing data after each refinement iteration to ensure that the improvements were not just overfitting to the training data but were genuinely enhancing the model's ability to generalize to unseen data. The final model, after these iterative refinements, demonstrated improved predictive power and better stability in its predictions.

CHAPTER 8

DISCUSSION AND CONCLUSION

8.1 SUMMARY OF FINDINGS

This project aimed to predict house prices in Melbourne using a variety of property features. After extensive data pre-processing, feature engineering, and exploratory data analysis, several key factors influencing house prices were identified. Notably, property location, size (represented by variables like BuildingArea and Rooms), and age of the property were found to have the most significant impact on the house prices. The dataset revealed that properties closer to the city centre generally had higher prices, consistent with urban economics, where proximity to amenities and infrastructure increases desirability.

The Random Forest Regressor model emerged as the best performing model for this task. It outperformed other models, such as linear regression and decision trees, by handling complex feature interactions and non-linearity in the data. The model's ability to create multiple decision trees and average their predictions helped mitigate overfitting, ensuring that it generalized well on unseen data. The performance metrics, including R^2 , RMSE, and MAE, indicated that the Random Forest model could predict house prices with a high degree of accuracy and low error, making it suitable for practical applications in real estate market predictions.

Furthermore, feature importance analysis demonstrated that distance to key amenities and property size were the most influential features in predicting property prices, followed by factors such as age and number of rooms. These findings underscore the importance of these variables when considering investments or pricing strategies in the housing market.

8.2 CHALLENGES AND LIMITATIONS

Throughout the project, several challenges were encountered:

1. **Handling Missing Values:** Missing data is a common issue in real-world datasets. In this project, certain critical features had missing values, particularly related to property characteristics like building area and property age. To address this, various imputation techniques were explored, including mean and median imputation. While these methods helped retain most of the data, there is always a risk that imputation may introduce bias, especially if the missing data is not missing at random.
2. **Data Skewness in Property Prices:** The distribution of house prices was heavily skewed, with a long right tail indicating that a small number of properties had extremely high prices. This skewness posed a challenge for some models, as many machine learning algorithms perform better when the data is normally distributed. To address this, a **log transformation** was applied to the price data, which helped normalize the distribution and improve model performance. However, despite these efforts, the model's ability to predict extreme house prices (high outliers) was still somewhat limited.
3. **Limited Ability to Incorporate Temporal Changes:** One of the main limitations of this study was that the dataset did not include temporal (time-based) information. Housing prices can fluctuate over time due to various factors, such as economic conditions, government policies, and seasonal demand. This project treated the dataset as a static snapshot of the market, without considering the evolving nature of the real estate market over time. A time series analysis, which could incorporate trends, seasonality, and economic cycles, would likely improve the model's predictive power if such data were available.
4. **Feature Selection and Overfitting:** Despite the efforts to identify important features through feature importance analysis, certain features may still have

contributed noise to the model. Random Forest models, while robust against overfitting, can still suffer from it, especially when too many irrelevant features are included. Fine-tuning the model through cross-validation and removing unnecessary features could help further enhance performance.

5. **Interpretability of the Model:** Random Forests, while powerful, are often considered black-box models due to the difficulty in interpreting how individual features influence the final predictions. This lack of transparency can be a drawback when it comes to explaining predictions to non-technical stakeholders, such as real estate investors or policymakers. Future work could explore models that balance predictive accuracy with interpretability, such as **Gradient Boosting Machines (GBM)** or **Explainable AI (XAI)** techniques, which offer ways to understand how predictions are made.
6. **Geographic and Demographic Variability:** Although this analysis identified significant patterns in Melbourne's housing market, these patterns might not apply universally across other regions. Factors such as regional demographics, local economic conditions, and different housing policies can greatly affect property values. Future work could extend the analysis to other cities or regions, considering local variables to improve generalizability.

APPENDIX

```
# Import necessary libraries

import pandas as pd

# Load the dataset

data = pd.read_csv('/content/archive (1).zip')

#data = pd.read_csv('melb_data.csv')

# Display the first few rows of the dataset

data.head()

data.info()

data.describe()

# Check for missing values

missing_data = data.isnull().sum()


# Remove rows or columns with significant missing values

data.dropna(subset=['Price'], inplace=True) # Example: Removing rows where 'Price'
is missing


# Fill missing values in other columns (e.g., 'BuildingArea') with median or mode

data['BuildingArea'].fillna(data['BuildingArea'].median(), inplace=True)

# Create new feature: Price per square foot

data['Price_per_sqft'] = data['Price'] / data['BuildingArea']


# Create new feature: Age of the house (assuming dataset collected in 2024)

data['House_Age'] = 2024 - data['YearBuilt']
```

```

# Drop columns that won't be used in the analysis

# Adjusted to match the available columns in the dataset

data.drop(columns=['Suburb', 'Method', 'Date', 'Postcode', 'CouncilArea',
'Regionname'], inplace=True)

print(data.columns)

import numpy as np

# Log-transform the 'Price' feature to reduce skewness

data['Log_Price'] = np.log1p(data['Price'])

# Check skewness of other features and transform if needed

data['Log_BuildingArea'] = np.log1p(data['BuildingArea'])

# Get descriptive statistics for key columns

data[['Price', 'BuildingArea', 'House_Age', 'Price_per_sqft']].describe()

# Select only numeric columns for correlation analysis

numeric_data = data.select_dtypes(include=['float64', 'int64'])

# Calculate the correlation matrix

corr_matrix = numeric_data.corr()

# Plot heatmap

import seaborn as sns

import matplotlib.pyplot as plt

```

```

plt.figure(figsize=(10, 8))

sns.heatmap(corr_matrix, annot=True, cmap='coolwarm')

plt.title('Correlation Matrix')

plt.show()

# Scatter plot of Latitude and Longitude colored by Price

plt.figure(figsize=(10, 6))

sns.scatterplot(x='Latitude', y='Longitude', hue='Price', data=data, palette='viridis')

plt.title('Geographical Distribution of House Prices')

plt.show()

from sklearn.model_selection import train_test_split

# Define target variable and features

X = data[['Rooms', 'Bathroom', 'Landsize', 'BuildingArea', 'House_Age']]

y = data['Log_Price'] # Using log-transformed price as target

# Split data into training and testing sets

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)

#model selection

from sklearn.impute import SimpleImputer

# Initialize an imputer to fill missing values with the median of each column

imputer = SimpleImputer(strategy='median')

```

```
# Fit and transform on the training data, then transform the test data
```

```
X_train = imputer.fit_transform(X_train)
```

```
X_test = imputer.transform(X_test)
```

```
# Proceed with fitting the models
```

```
from sklearn.linear_model import LinearRegression
```

```
from sklearn.ensemble import RandomForestRegressor
```

```
from sklearn.tree import DecisionTreeRegressor
```

```
# Initialize models
```

```
lr = LinearRegression()
```

```
rf = RandomForestRegressor(random_state=42)
```

```
dt = DecisionTreeRegressor(random_state=42)
```

```
# Fit models on training data
```

```
lr.fit(X_train, y_train)
```

```
rf.fit(X_train, y_train)
```

```
dt.fit(X_train, y_train)
```

```
#model tuning
```

```
from sklearn.model_selection import GridSearchCV
```

```
# Define parameter grid
```

```
param_grid = {
```

```
    'n_estimators': [50, 100, 200],
```

```

    'max_depth': [None, 10, 20],

    'min_samples_split': [2, 5, 10]

}

# Grid search

grid_search = GridSearchCV(RandomForestRegressor(random_state=42),
param_grid, cv=3)

grid_search.fit(X_train, y_train)

# Best parameters

best_params = grid_search.best_params_

#model evaluation

#1.evaluation metrics

from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score

# Predictions and evaluation

y_pred = rf.predict(X_test)

rmse = np.sqrt(mean_squared_error(y_test, y_pred))

mae = mean_absolute_error(y_test, y_pred)

r2 = r2_score(y_test, y_pred)

print(f"RMSE: {rmse}")

print(f"MAE: {mae}")

print(f"R^2: {r2}")

```

#2.cross validation

```
from sklearn.model_selection import cross_val_score
```

Cross-validation

```
cv_scores = cross_val_score(rf, X, y, cv=5, scoring='neg_mean_squared_error')
```

```
cv_rmse = np.sqrt(-cv_scores)
```

```
print(f'Cross-validated RMSE: {cv_rmse.mean()}')
```

#Residual analysis

Residual plot

```
residuals = y_test - y_pred
```

```
plt.scatter(y_pred, residuals)
```

```
plt.hlines(y=0, xmin=min(y_pred), xmax=max(y_pred), colors='red')
```

```
plt.xlabel('Predicted')
```

```
plt.ylabel('Residuals')
```

```
plt.title('Residual Analysis')
```

```
plt.show()
```

#Insights

#1.Price Distribution Analysis

```
import matplotlib.pyplot as plt
```

```
import seaborn as sns
```

```
plt.figure(figsize=(8, 6))
```

```
sns.histplot(data['Price'], bins=50, kde=True)
```

```
plt.title("Distribution of House Prices")
```

```
plt.xlabel("Price")
plt.ylabel("Frequency")
plt.show()

#2.Price vs. Number of Rooms

plt.figure(figsize=(8, 6))
sns.boxplot(x='Rooms', y='Price', data=data)
plt.title("Price vs. Number of Rooms")
plt.xlabel("Number of Rooms")
plt.ylabel("Price")
plt.show()
```

```
#3.Price vs. Distance from City Center

plt.figure(figsize=(8, 6))
sns.scatterplot(x='Distance', y='Price', data=data)
plt.title("Price vs. Distance from City Center")
plt.xlabel("Distance from City Center (km)")
plt.ylabel("Price")
plt.show()
```

```
#5.Correlation Heatmap of Numeric Features
```

```
import seaborn as sns
```

```
import matplotlib.pyplot as plt
```

```
# Select only numeric columns for the correlation matrix
```

```
numeric_data = data.select_dtypes(include=['float64', 'int64'])
```



```

# Calculate the correlation matrix

corr_matrix = numeric_data.corr()


# Plot heatmap

plt.figure(figsize=(10, 8))

sns.heatmap(corr_matrix, annot=True, cmap="coolwarm", fmt=".2f")

plt.title("Correlation Heatmap of Numeric Features")

plt.show()


#6. House Age vs. Price

plt.figure(figsize=(8, 6))

sns.scatterplot(x='House_Age', y='Price', data=data)

plt.title("House Age vs. Price")

plt.xlabel("House Age")

plt.ylabel("Price")

plt.show()


#7. Price by Property Type

plt.figure(figsize=(8, 6))

sns.boxplot(x='Type', y='Price', data=data)

plt.title("Price by Property Type")

plt.xlabel("Property Type")

plt.ylabel("Price")

plt.show()

```

#8. Building Area vs. Price

```
plt.figure(figsize=(8, 6))  
sns.scatterplot(x='BuildingArea', y='Price', data=data)  
plt.title("Building Area vs. Price")  
plt.xlabel("Building Area (sqft)")  
plt.ylabel("Price")  
plt.show()
```

#9. Price Distribution by Bathroom Count

```
plt.figure(figsize=(8, 6))  
sns.boxplot(x='Bathroom', y='Price', data=data)  
plt.title("Price Distribution by Bathroom Count")  
plt.xlabel("Number of Bathrooms")  
plt.ylabel("Price")  
plt.show()
```

#10. Count of properties by type

```
property_type_counts = data['Type'].value_counts()
```

Plotting the distribution of property types

```
plt.figure(figsize=(10, 6))  
sns.barplot(x=property_type_counts.index, y=property_type_counts.values,  
palette='viridis')  
plt.title("Distribution of Property Types")  
plt.xlabel("Property Type")  
plt.ylabel("Count")
```

```

plt.xticks(rotation=45)

plt.show()

# Step 1: Generate Predictions

y_pred = lr.predict(X_test) # replace `lr` with your trained model if using a different
model

# Step 2: Convert Predictions back from log-transformed (if applicable)

# If you've log-transformed the target during training (using np.log), use np.exp to get
back the original scale

y_pred_original_scale = np.exp(y_pred)

# Step 3: Create a DataFrame with Actual and Predicted Prices

predictions_df = pd.DataFrame({

    'Actual Price': np.exp(y_test), # Use np.exp if y_test was log-transformed

    'Predicted Price': y_pred_original_scale

})

# Step 4: Display the Predicted Prices

print(predictions_df.head()) # Display the first few rows

# Optional: Save predictions to a CSV file if needed

predictions_df.to_csv("predicted_prices.csv", index=False)

# Displaying the first few rows of predicted and actual prices for reference

predictions_df.head(10) # Display the first 10 rows for comparison

```

REFERENCES

- [1] J. Brownlee, *Master Machine Learning Algorithms*, 1st ed. Melbourne, Australia: Machine Learning Mastery, 2016.
- [2] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*, 2nd ed. New York: Springer, 2009.
- [3] D. C. Montgomery, E. A. Peck, and G. G. Vining, *Introduction to Linear Regression Analysis*, 5th ed. Hoboken, NJ: Wiley, 2012.
- [4] M. Abadi et al., "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015. [Online].
- [5] J. Friedman, T. Hastie, and R. Tibshirani, "Regularization paths for generalized linear models via coordinate descent," *Journal of Statistical Software*, vol. 33, no. 1, pp. 1–22, 2010.
- [6] G. James, D. Witten, T. Hastie, and R. Tibshirani, *An Introduction to Statistical Learning: With Applications in R*, 2nd ed. New York: Springer, 2021.
- [7] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*, Cambridge, MA: MIT Press, 2016.
- [8] F. Chollet, *Deep Learning with Python*, 2nd ed. Shelter Island, NY: Manning Publications, 2021.
- [9] R. B. Cattell, "The scree test for the number of factors," *Multivariate Behavioral Research*, vol. 1, no. 2, pp. 245–276, 1966.
- [10] P. Domingos, "A few useful things to know about machine learning," *Communications of the ACM*, vol. 55, no. 10, pp. 78–87, Oct. 2012.
- [11] S. Raschka and V. Mirjalili, *Python Machine Learning: Machine Learning and Deep Learning with Python, scikit-learn, and TensorFlow 2*, 3rd ed. Birmingham, U.K.: Packt Publishing, 2019.
- [12] A. Ng, "Machine learning yearning," 2019. [Online].

- [13] D. Silver et al., "Mastering the game of Go with deep neural networks and tree search," *Nature*, vol. 529, no. 7587, pp. 484–489, 2016.
- [14] R. Tibshirani, "Regression shrinkage and selection via the Lasso," *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, vol. 58, no. 1, pp. 267–288, 1996.
- [15] H. Zou and T. Hastie, "Regularization and variable selection via the elastic net," *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, vol. 67, no. 2, pp. 301–320, 2005.