REP2_Team7_MiniProject

Team 7 Lex Tan Pengqin (U2322095L) Tan Yu Xiu (U2322532B) Lim Jun Shawn (U2320477A)

Kaggle - Melbourne Housing Dataset

Dataset Link: https://www.kaggle.com/datasets/dansbecker/melbourne-housing-snapshot/data
The dataset has 13580 columns and 21 rows, and contains data on the price, features of the estate,

Problem Statement

This project will use regression techniques to predict the trend of housing prices in different estates in Melbourne. Additionally, we will analyze the significance of chosen variables in determining the price, to enhance the accuracy of the predictive model. Below are our guiding questions:

1. Which features have the highest impact on price based on feature importance?
2. How has the price trend varied across different regions over time, and can we identify emerging high-value areas?
3. How does property age and size influence prediction accuracy in our models?

Response Variables

Importantly, we have chosen our indicators of real estate popularity (Response Variables) as Price.

```
import numpy as np
import pandas as pd
import seaborn as sb
import matplotlib.pyplot as plt
from sklearn.preprocessing import StandardScaler, OrdinalEncoder,
OneHotEncoder, PolynomialFeatures
from sklearn.compose import ColumnTransformer
from sklearn.impute import SimpleImputer
from sklearn.pipeline import Pipeline
from sklearn.tree import DecisionTreeRegressor
from sklearn.cluster import KMeans
from yellowbrick.model_selection import FeatureImportances
from sklearn.impute import SimpleImputer
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.linear_model import LinearRegression
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error, mean_absolute_error,
r2_score
from sklearn.feature_selection import RFE
```

Flow

1. Exploratory Data Analysis
2. Preparation of Variables
3. Core Analysis and Respective Improvements
4. Conclusion

Importing dataset

```
housingdata = pd.read_csv('melb_data.csv')
# print(educationdata.head())
# print(educationdata.shape)
print(housingdata.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 13580 entries, 0 to 13579
Data columns (total 21 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   Suburb         13580 non-null  object
 1   Address        13580 non-null  object
 2   Rooms          13580 non-null  int64
 3   Type           13580 non-null  object
 4   Price          13580 non-null  float64
 5   Method         13580 non-null  object
 6   SellerG        13580 non-null  object
 7   Date           13580 non-null  object
 8   Distance       13580 non-null  float64
 9   Postcode       13580 non-null  float64
 10  Bedroom2       13580 non-null  float64
 11  Bathroom       13580 non-null  float64
 12  Car            13518 non-null  float64
 13  Landsize       13580 non-null  float64
 14  BuildingArea   7130 non-null   float64
 15  YearBuilt      8205 non-null   float64
 16  CouncilArea    12211 non-null  object
 17  Lattitude      13580 non-null  float64
 18  Longtitude     13580 non-null  float64
 19  Regionname     13580 non-null  object
 20  Propertycount  13580 non-null  float64
dtypes: float64(12), int64(1), object(8)
memory usage: 2.2+ MB
None
```

Some of our observations:

- There is a good balance between the number of numerical (Dtype = int64/float64) and categorical (Dtype = object) features, allowing for deeper analysis.

```
print(housingdata.describe())
```

```
              Rooms         Price      Distance      Postcode
Bedroom2  \
count  13580.000000  1.358000e+04  13580.000000  13580.000000
13580.000000
mean       2.937997  1.075684e+06     10.137776   3105.301915
2.914728
std        0.955748  6.393107e+05      5.868725     90.676964
0.965921
min        1.000000  8.500000e+04      0.000000   3000.000000
0.000000
25%        2.000000  6.500000e+05      6.100000   3044.000000
2.000000
50%        3.000000  9.030000e+05      9.200000   3084.000000
3.000000
75%        3.000000  1.330000e+06     13.000000   3148.000000
3.000000
max       10.000000  9.000000e+06     48.100000   3977.000000
20.000000

           Bathroom           Car       Landsize  BuildingArea
YearBuilt  \
count  13580.000000  13518.000000   13580.000000   7130.000000
8205.000000
mean       1.534242      1.610075     558.416127    151.967650
1964.684217
std        0.691712      0.962634    3990.669241    541.014538
37.273762
min        0.000000      0.000000       0.000000      0.000000
1196.000000
25%        1.000000      1.000000     177.000000     93.000000
1940.000000
50%        1.000000      2.000000     440.000000    126.000000
1970.000000
75%        2.000000      2.000000     651.000000    174.000000
1999.000000
max        8.000000     10.000000  433014.000000  44515.000000
2018.000000

          Lattitude    Longtitude  Propertycount
count  13580.000000  13580.000000   13580.000000
mean     -37.809203    144.995216    7454.417378
std        0.079260      0.103916    4378.581772
min      -38.182550    144.431810     249.000000
25%      -37.856822    144.929600    4380.000000
50%      -37.802355    145.000100    6555.000000
75%      -37.756400    145.058305   10331.000000
max      -37.408530    145.526350   21650.000000
```

(1) Exploratory Data Analysis (EDA)

1. Categorizing Variables
2. Data Processing
3. Analysis of Respose Variable Price
4. Analysis of Numerical Predictor Variables
5. Analysis of Categorical Predictor Variables

1. Categorizing Variables

```
# Identify numerical and categorical columns using select_dtypes
numerical =
housingdata.select_dtypes(include=[np.number]).columns.tolist()
categorical = housingdata.select_dtypes(include=['object',
'category']).columns.tolist()

# Convert object type columns to category
housingdata[categorical] = housingdata[categorical].astype('category')

print("Numeric features:")
print(numerical)
print()
print("Categorical features:")
print(categorical)
```

```
Numeric features:
['Rooms', 'Price', 'Distance', 'Postcode', 'Bedroom2', 'Bathroom',
'Car', 'Landsize', 'BuildingArea', 'YearBuilt', 'Lattitude',
'Longtitude', 'Propertycount']

Categorical features:
['Suburb', 'Address', 'Type', 'Method', 'SellerG', 'Date',
'CouncilArea', 'Regionname']
```

2. Data Processing

```
#### 2a. Checking for, and handling, NULL values

print(housingdata.isnull().sum())
```

```
Suburb              0
Address             0
Rooms               0
Type                0
Price               0
Method              0
SellerG             0
Date                0
Distance            0
Postcode            0
```

```
Bedroom2            0
Bathroom            0
Car                62
Landsize            0
BuildingArea     6450
YearBuilt        5375
CouncilArea      1369
Lattitude           0
Longtitude          0
Regionname          0
Propertycount       0
dtype: int64
```

Before deciding whether to drop or impute columns with missing values, it's useful to assess their relative importance in predicting our response variable Price.

By training a decision tree regressor and visualizing feature importance, we can prioritize features that contribute significantly to predicting Price, and confidently remove those with low predictive value and high nullity.

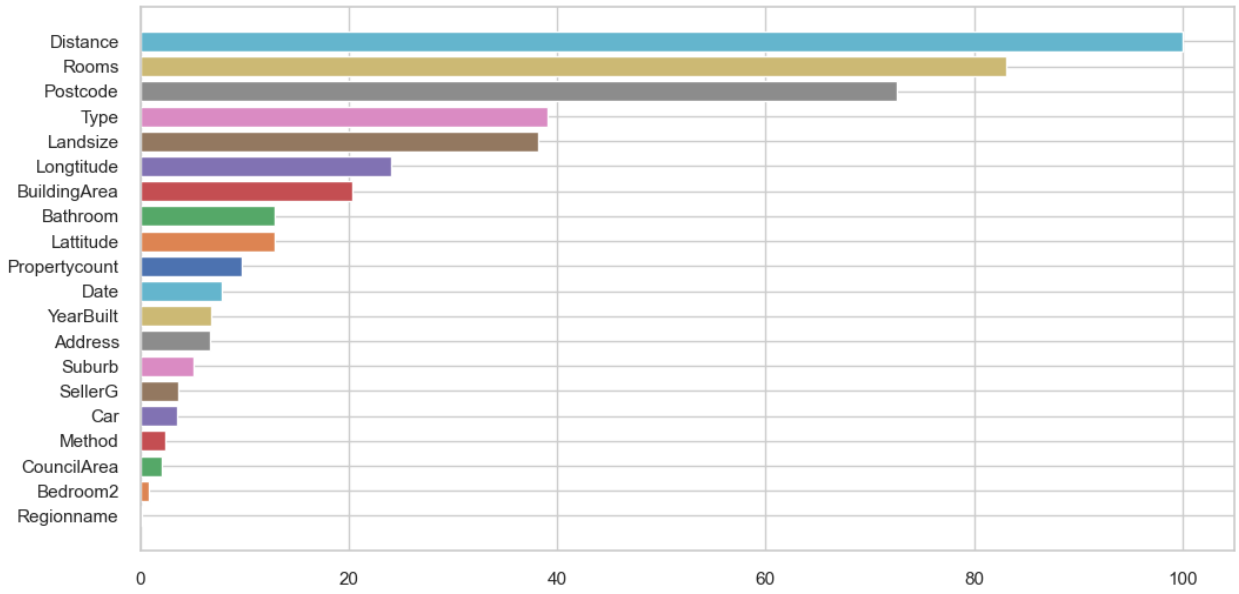
This approach helps us simplify the model without sacrificing accuracy, and ensures we only invest effort in imputing values that matter.

NOTE: This will be a much dumbed-down approach of visualizing feature importance as compared to the Random Forest model we will use later, and is only essential for imputing NULL values.

Decision Tree Regressor

```
encoder = OrdinalEncoder()
housingdata[categorical] =
encoder.fit_transform(housingdata[categorical].astype(str))

X1 = housingdata.drop(columns=["Price"])  # Features
y1= housingdata["Price"]  # Target


model1 = DecisionTreeRegressor(random_state=42)
viz1 = FeatureImportances(model1)
viz1.fit(X1, y1)
```

```
FeatureImportances(ax=<Axes: >,
                   estimator=DecisionTreeRegressor(random_state=42))
```



Imputing

https://medium.com/@ajayverma23/data-imputation-a-comprehensive-guide-to-handling-missing-values-b5c7d11c3488

Handling 'Car' NULL values

Since the number of NULL entries in Car is only about (62/13580) * 100 = 0.46% of the total number of rows, and its relative importance in predicting Price is low, we opt to drop these rows. This approach reduces preprocessing complexity without sacrificing valuable information for our analysis.

```
# drop rows accordingly
housingdata = housingdata.dropna(subset=['Car'])
```

Handling 'BuildingArea' NULL values

Since the number of NULL entries in BuildingArea is about (6450/13580) * 100 = 47.5% of the total number of rows, which is significant, and its relative importance in predicting Price is moderate, we will impute the missing values using the median, grouped by Suburb, Rooms, and Type. This is because building area typically scales with number of rooms and depends on housing type, and suburbs offer spatial consistency.

```
# Impute BuildingArea based on median within Suburb + Distance + Rooms
housingdata['BuildingArea'] = housingdata.groupby(['Suburb', 'Rooms',
'Type'])['BuildingArea'].transform(
    lambda x: x.fillna(x.median())
)

c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
```

```
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
```

```
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
```

```
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
```

```
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
```

```
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
```

```
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
```

```
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
```

```
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
```

```
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
```

```
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
```

```
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
```

```
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
```

```
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
```

```
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
```

```
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
```

```
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
```

```
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
```

```
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
```

```
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
```

```
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
```

Handling 'YearBuilt' NULL values

Since the number of NULL entries in YearBuilt is about (5375/13580) * 100 = 39.6% of the total number of rows, which is moderate, and its relative importance in predicting Price is also moderate, we will impute the missing values using the median, grouped by Suburb and Type. This leverages contextual property characteristics to estimate construction year more accurately.

```python
# Impute YearBuilt based on Suburb + Type
housingdata['YearBuilt'] = housingdata.groupby(['Suburb', 'Type'])
['YearBuilt'].transform(
    lambda x: x.fillna(x.median())
)
```

```
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
```

```
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
```

```
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
```

```
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
```

```
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
c:\GitHub\DSAI_Mini_Project\.env\lib\site-packages\numpy\lib\
_nanfunctions_impl.py:1215: RuntimeWarning: Mean of empty slice
  return np.nanmean(a, axis, out=out, keepdims=keepdims)
```

Handling 'CouncilArea' NULL values

Since the number of NULL entries in CouncilArea is about (1369/13580) * 100 = 10.1% of the total number of rows, which is moderate. Given that CouncilArea has moderate feature importance in predicting Price, we will impute the missing values using the mode, grouped by Suburb, as council areas typically align with geographic boundaries.

```
# Impute CouncilArea (categorical) based on Suburb
housingdata['CouncilArea'] = housingdata.groupby('Suburb')
['CouncilArea'].transform(
    lambda x: x.fillna(x.mode()[0] if not x.mode().empty else
'Unknown')
)
```

Checking the null values again

```
print(housingdata.isnull().sum())
```

```
Suburb             0
Address            0
Rooms              0
Type               0
Price              0
Method             0
SellerG            0
Date               0
Distance           0
Postcode           0
Bedroom2           0
Bathroom           0
Car                0
Landsize           0
BuildingArea     440
YearBuilt         80
CouncilArea        0
Lattitude          0
Longtitude         0
Regionname         0
Propertycount      0
dtype: int64
```

Given that we still have a small proportion of NULL values left even after our imputing, we will choose to drop these rows.

```
housingdata = housingdata.dropna(subset=['BuildingArea'])
housingdata = housingdata.dropna(subset=['YearBuilt'])

print(housingdata.isnull().sum())
```

```
Suburb           0
Address          0
Rooms            0
Type             0
Price            0
Method           0
SellerG          0
Date             0
Distance         0
Postcode         0
Bedroom2         0
Bathroom         0
Car              0
Landsize         0
BuildingArea     0
YearBuilt        0
CouncilArea      0
Lattitude        0
Longtitude       0
Regionname       0
Propertycount    0
dtype: int64
```

After imputing, all the categorical rows have been converted to numerical rows, hence we have to convert them back.

```
housingdata[categorical] = housingdata[categorical].astype('object')

# Check the updated dtypes
print(housingdata.dtypes)
```

```
Suburb            object
Address           object
Rooms              int64
Type              object
Price            float64
Method            object
SellerG           object
Date              object
Distance         float64
Postcode         float64
Bedroom2         float64
Bathroom         float64
Car              float64
Landsize         float64
```

```
BuildingArea     float64
YearBuilt        float64
CouncilArea       object
Lattitude        float64
Longtitude       float64
Regionname        object
Propertycount    float64
dtype: object
```

2b. Checking for, and removing, duplicate values

```
print(housingdata.duplicated().sum())
```

```
0
```

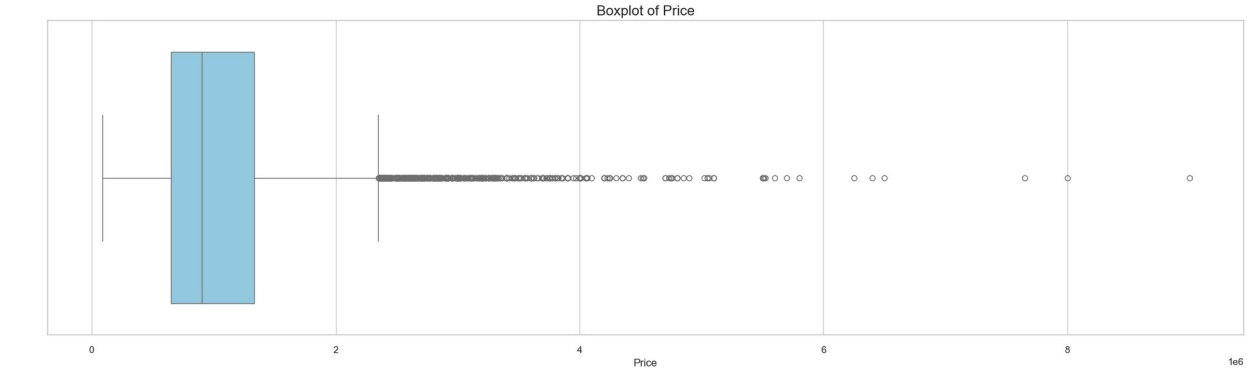
Since there are no duplicate values, we will continue with our analysis.

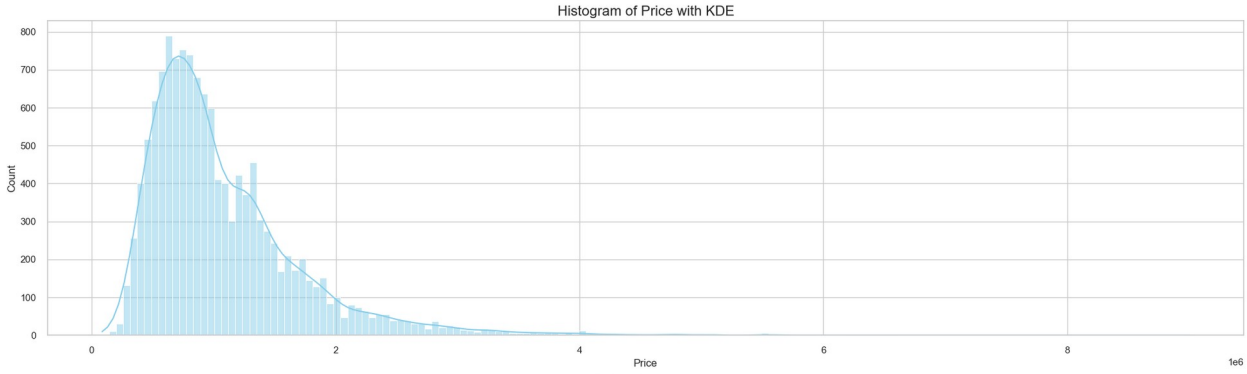
3. Analysis of Response Variable Price

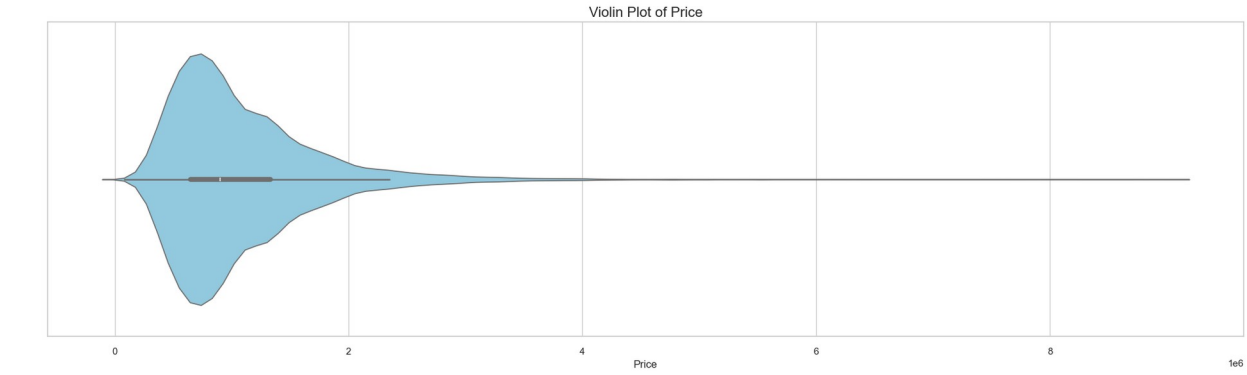
```
price = housingdata["Price"]
price.describe()
```

```
count    1.306600e+04
mean     1.077267e+06
std      6.404485e+05
min      8.500000e+04
25%      6.500000e+05
50%      9.035000e+05
75%      1.330000e+06
max      9.000000e+06
Name: Price, dtype: float64
```

```
f, axes = plt.subplots(3,1, figsize=(20,18))

# Boxplot for "Price"
sb.boxplot(data=housingdata['Price'], orient='h', ax=axes[0],
color='skyblue')
axes[0].set_title('Boxplot of Price', fontsize=16)

# Histogram for "Price" with KDE
sb.histplot(data=housingdata['Price'], ax=axes[1], kde=True,
color='skyblue')
axes[1].set_title('Histogram of Price with KDE', fontsize=16)

# Violin plot for "Price"
sb.violinplot(data=housingdata['Price'], orient='h', ax=axes[2],
color='skyblue')
axes[2].set_title('Violin Plot of Price', fontsize=16)

plt.tight_layout()  # Ensures proper spacing between plots
plt.show()
```







4. Univariate Analysis of Numerical Predictor Variables

4.1 Boxplot for numerical features

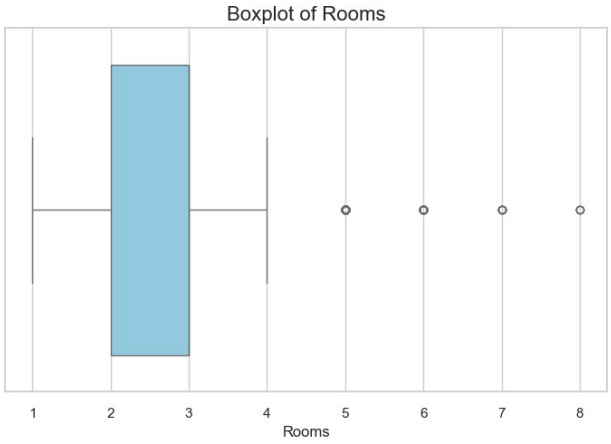
```
numerical_without_price =
housingdata.select_dtypes(include=[np.number]).drop(columns=["Price"])
numerical_without_price.describe()
```

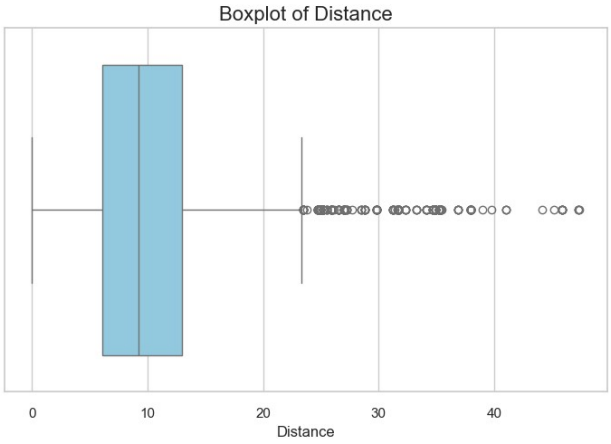
```
              Rooms      Distance      Postcode      Bedroom2
Bathroom  \
count  13066.000000  13066.000000  13066.000000  13066.000000
13066.000000
mean       2.933415     10.048117   3103.903260      2.909230
1.531073
std        0.932472      5.730516     87.174441      0.934042
0.689314
min        1.000000      0.000000   3000.000000      0.000000
```

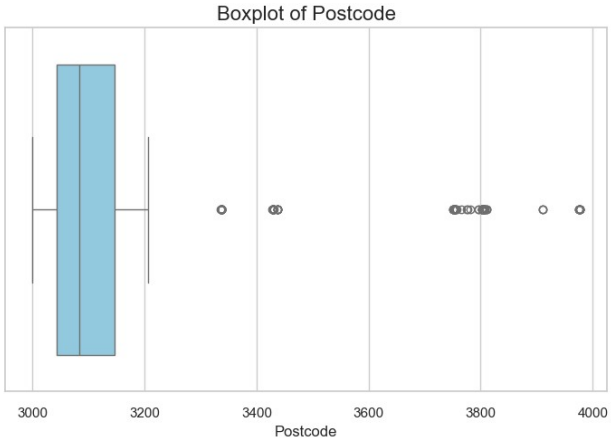
```
0.000000
25%         2.000000      6.100000   3044.000000      2.000000
1.000000
50%         3.000000      9.200000   3084.000000      3.000000
1.000000
75%         3.000000     13.000000   3147.000000      3.000000
2.000000
max         8.000000     47.400000   3977.000000      9.000000
8.000000

                Car       Landsize  BuildingArea     YearBuilt
Lattitude  \
count  13066.000000   13066.000000  13066.000000  13066.000000
13066.000000
mean        1.609597     546.817848    139.664890   1962.001913    -
37.808792
std         0.963456    3978.724641    103.700871     34.130073
0.078424
min         0.000000       0.000000      0.000000   1196.000000    -
38.182550
25%         1.000000     177.250000     94.000000   1940.000000    -
37.856490
50%         2.000000     444.000000    124.500000   1965.000000    -
37.802075
75%         2.000000     650.000000    164.000000   1986.000000    -
37.756293
max        10.000000  433014.000000   6791.000000   2018.000000    -
37.408530

         Longtitude  Propertycount
count  13066.000000   13066.000000
mean     144.994105    7543.149089
std        0.102292    4384.360761
min      144.542370     389.000000
25%      144.928700    4442.000000
50%      144.999700    6567.000000
75%      145.056880   10331.000000
max      145.526350   21650.000000
```

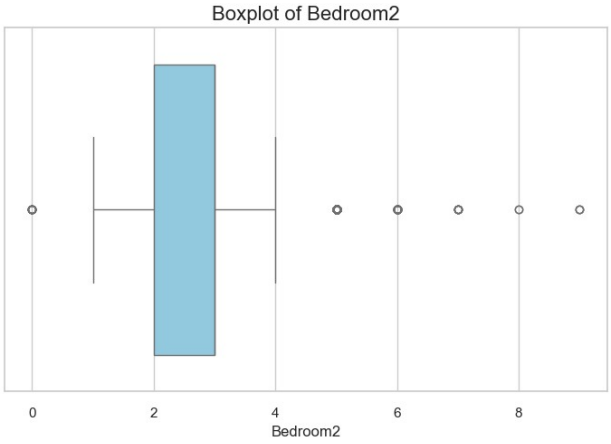
```
# Boxplot for all numerical features individually in a horizontal
layout
num_features = len(numerical_without_price.columns)
plt.figure(figsize=(14, num_features // 2 * 5))  # Adjust the figure
size dynamically

for i, feature in enumerate(numerical_without_price.columns):
    plt.subplot((num_features + 1) // 2, 2, i + 1)  # Ensure 2
boxplots per row
    sb.boxplot(data=housingdata[feature], orient='h', color='skyblue')
    plt.title(f'Boxplot of {feature}', fontsize=16)
```

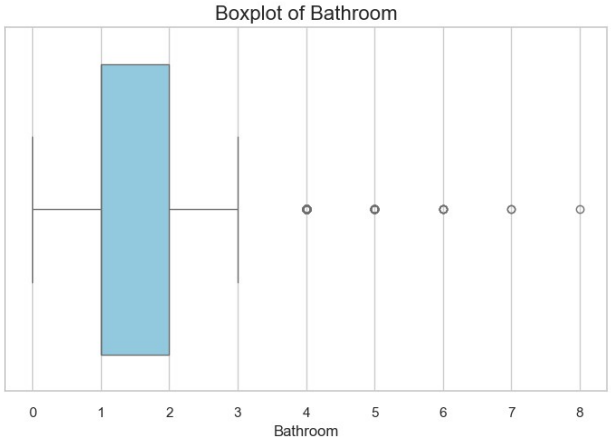
```
plt.tight_layout()  # Ensures proper spacing between plots
plt.show()
```

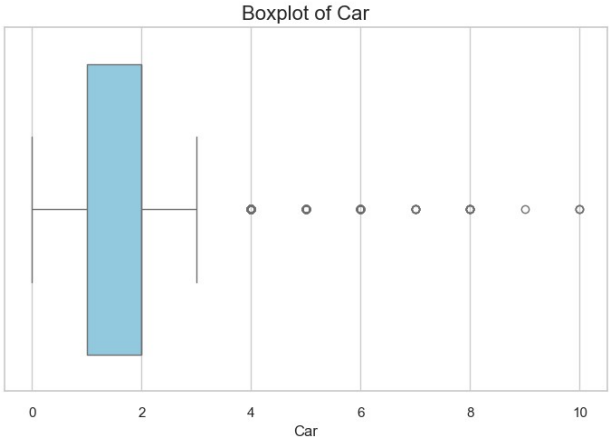

Boxplot of Rooms
1
2
3
4
5
6
7
8
Rooms

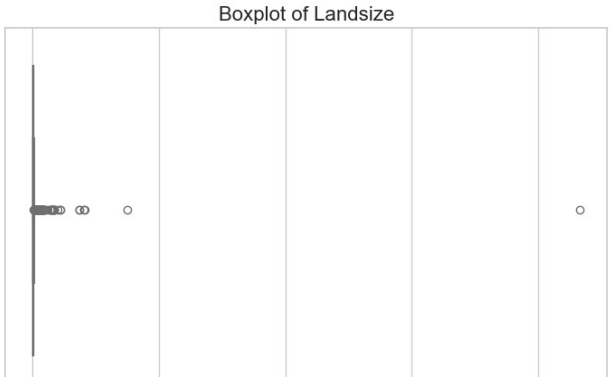

Boxplot of Distance
0
10
20
30
40
Distance

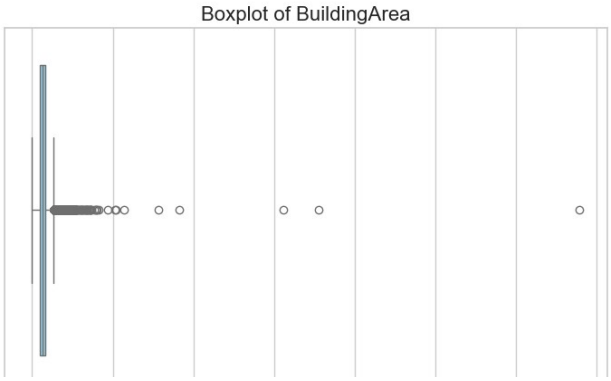

Boxplot of Postcode
3000
3200
3400
3600
3800
4000
Postcode


Boxplot of Bedroom2
0
2
4
6
8
Bedroom2


Boxplot of Bathroom
0
1
2
3
4
5
6
7
8
Bathroom

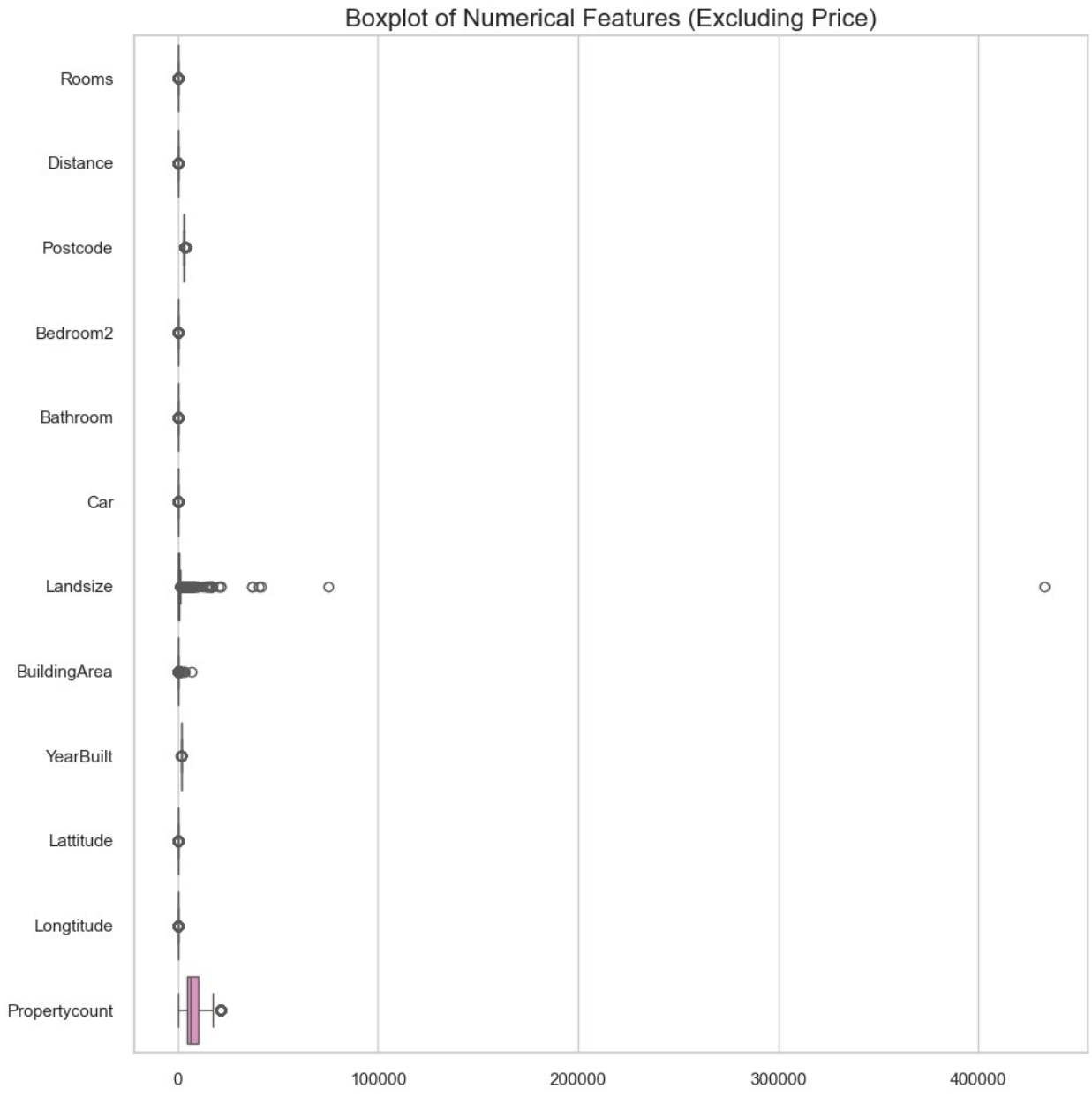

Boxplot of Car
0
2
4
6
8
10
Car


Boxplot of Landsize


Boxplot of BuildingArea

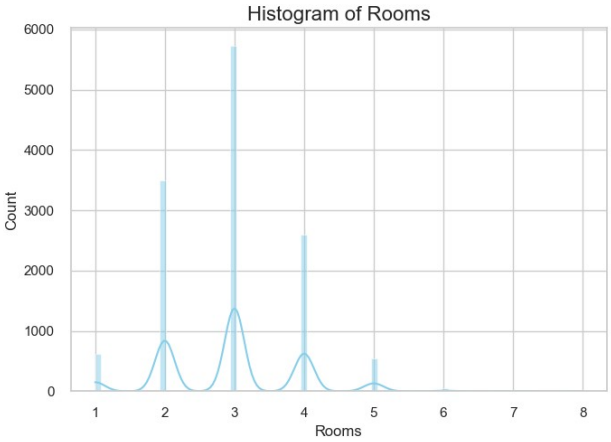
For Comparison with all variables at the same time at the expense of visibility

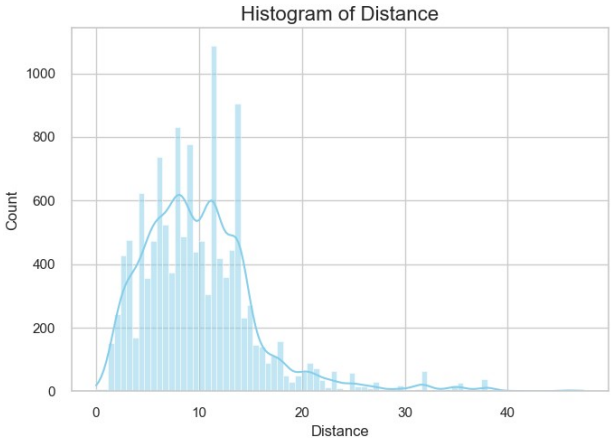
```
plt.figure(figsize=(10, 10))
sb.boxplot(data=numerical_without_price, orient='h', palette="Set2")
plt.title('Boxplot of Numerical Features (Excluding Price)',
fontsize=16)
plt.tight_layout()  # Ensures proper spacing
plt.show()
```

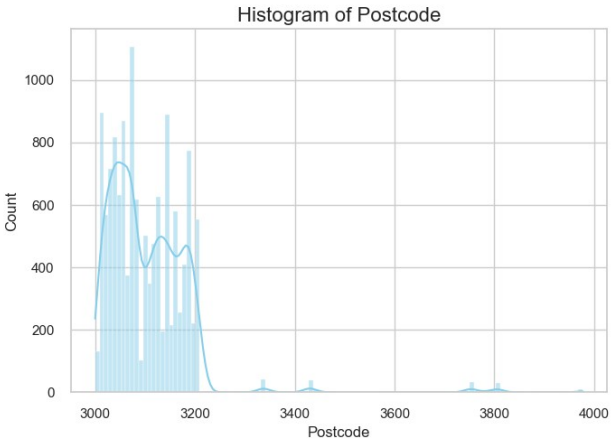


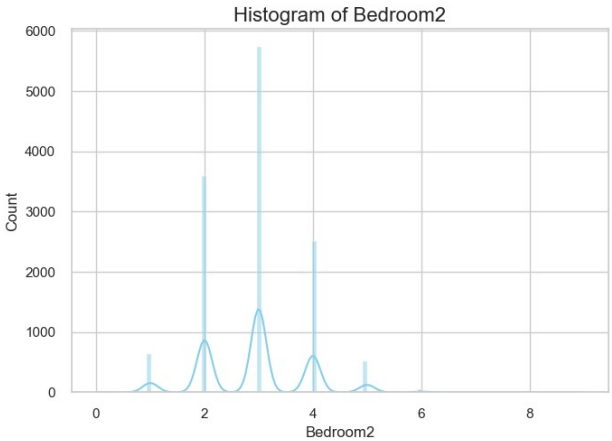
4.2 Histogram for numerical features

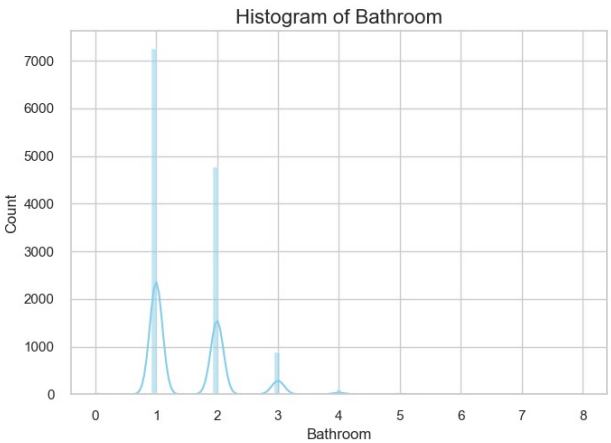
```
# Histogram for all numerical features individually in a horizontal
layout
num_features = len(numerical_without_price.columns)
plt.figure(figsize=(14, num_features // 2 * 5))  # Adjust the figure
size dynamically

for i, feature in enumerate(numerical_without_price.columns):
    plt.subplot((num_features + 1) // 2, 2, i + 1)  # Ensure 2
histograms per row
    sb.histplot(data=housingdata[feature], kde=True, color='skyblue',
ax=plt.gca())
    plt.title(f'Histogram of {feature}', fontsize=16)

plt.tight_layout()  # Ensures proper spacing between plots
plt.show()
```

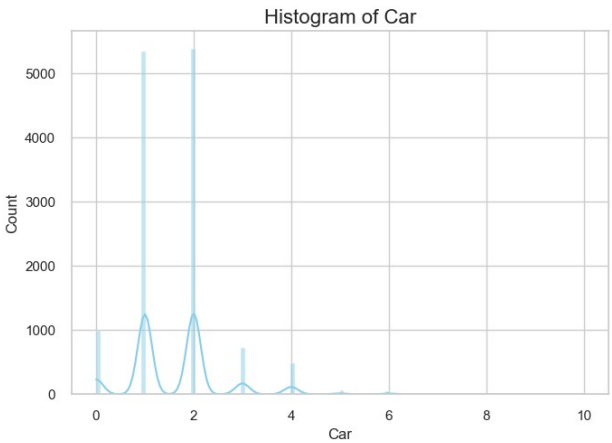

Histogram of Rooms
Count
Rooms


Histogram of Distance
Count
Distance

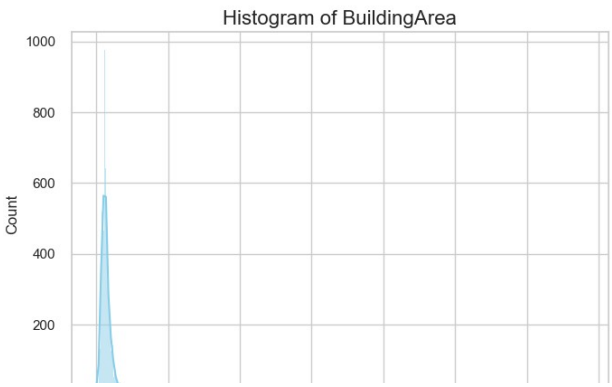

Histogram of Postcode
Count
Postcode


Histogram of Bedroom2
Count
Bedroom2


Histogram of Bathroom
Count
Bathroom

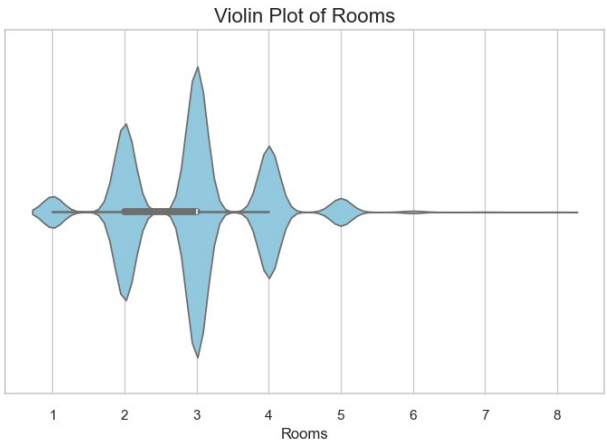

Histogram of Car
Count
Car

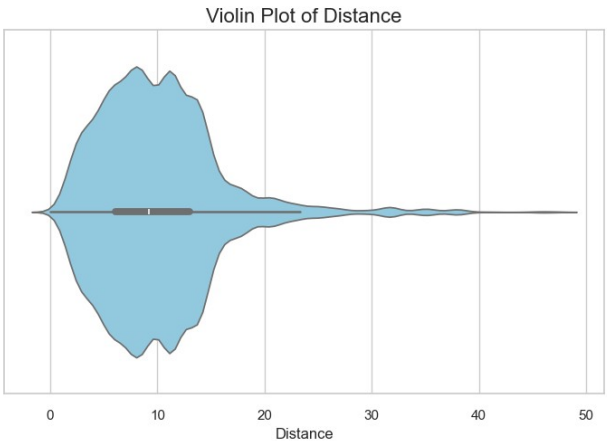

Histogram of Landsize
Count

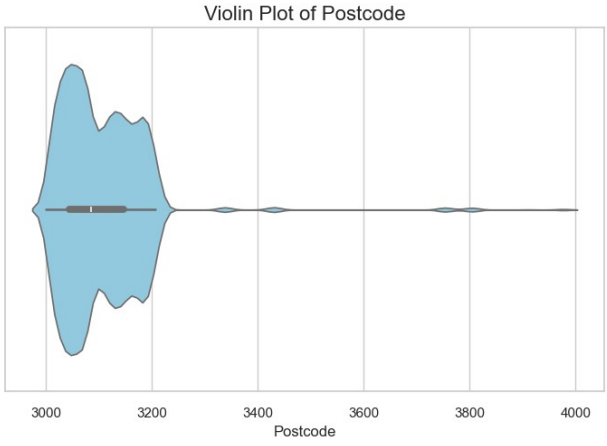

Histogram of BuildingArea
Count

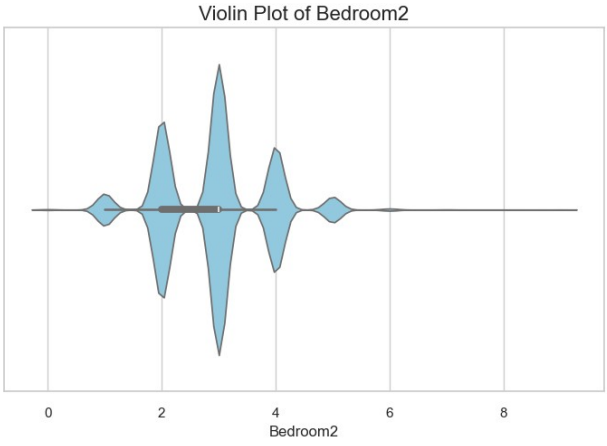
4.3 Violin plot for numerical features

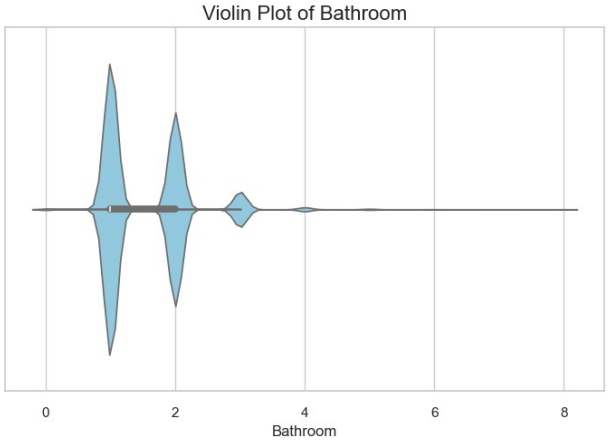
```
# Violin plot for all numerical features individually in a horizontal
layout
num_features = len(numerical_without_price.columns)
plt.figure(figsize=(14, num_features // 2 * 5))  # Adjust the figure
size dynamically

for i, feature in enumerate(numerical_without_price.columns):
    plt.subplot((num_features + 1) // 2, 2, i + 1)  # Ensure 2 violin
plots per row
    sb.violinplot(data=housingdata[feature], orient='h', hue=None,
color='skyblue', ax=plt.gca())  # Set hue=None
    plt.title(f'Violin Plot of {feature}', fontsize=16)

plt.tight_layout()  # Ensures proper spacing between plots
plt.show()
```

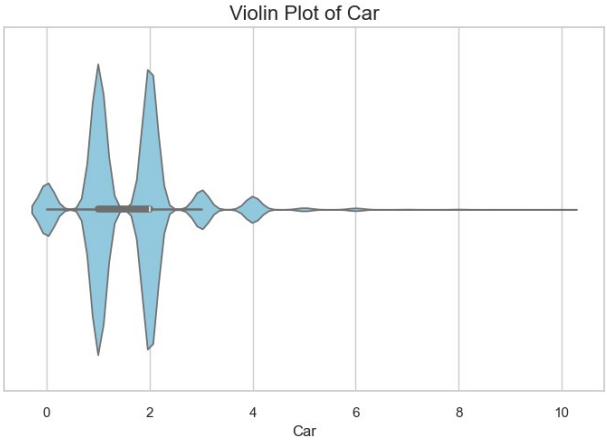

Violin Plot of Rooms
1
2
3
4
5
6
7
8
Rooms

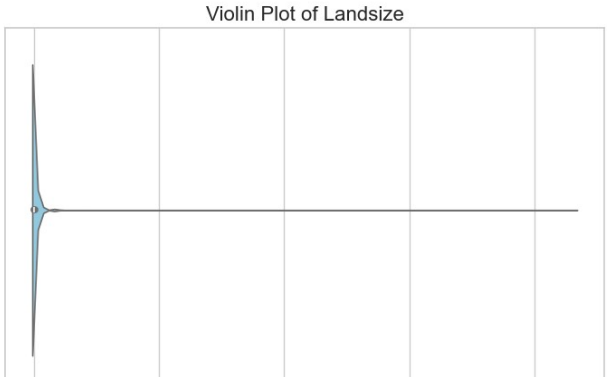

Violin Plot of Distance
0
10
20
30
40
50
Distance

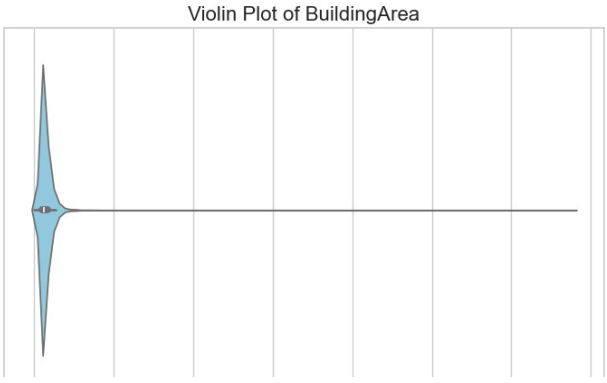

Violin Plot of Postcode
3000
3200
3400
3600
3800
4000
Postcode


Violin Plot of Bedroom2
0
2
4
6
8
Bedroom2


Violin Plot of Bathroom
0
2
4
6
8
Bathroom

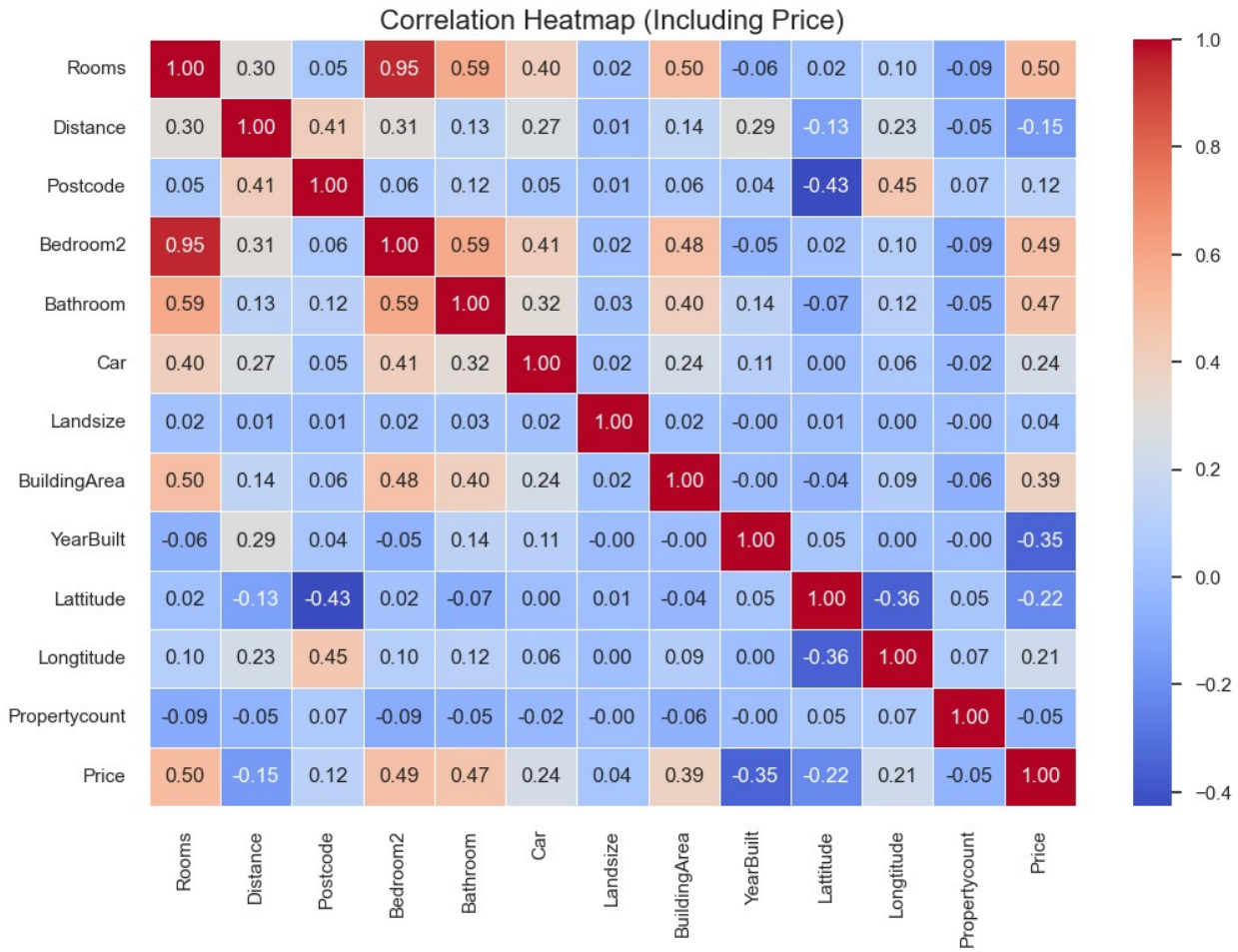

Violin Plot of Car
0
2
4
6
8
10
Car


Violin Plot of Landsize


Violin Plot of BuildingArea

4.4 Correlation Matrix of numerical features

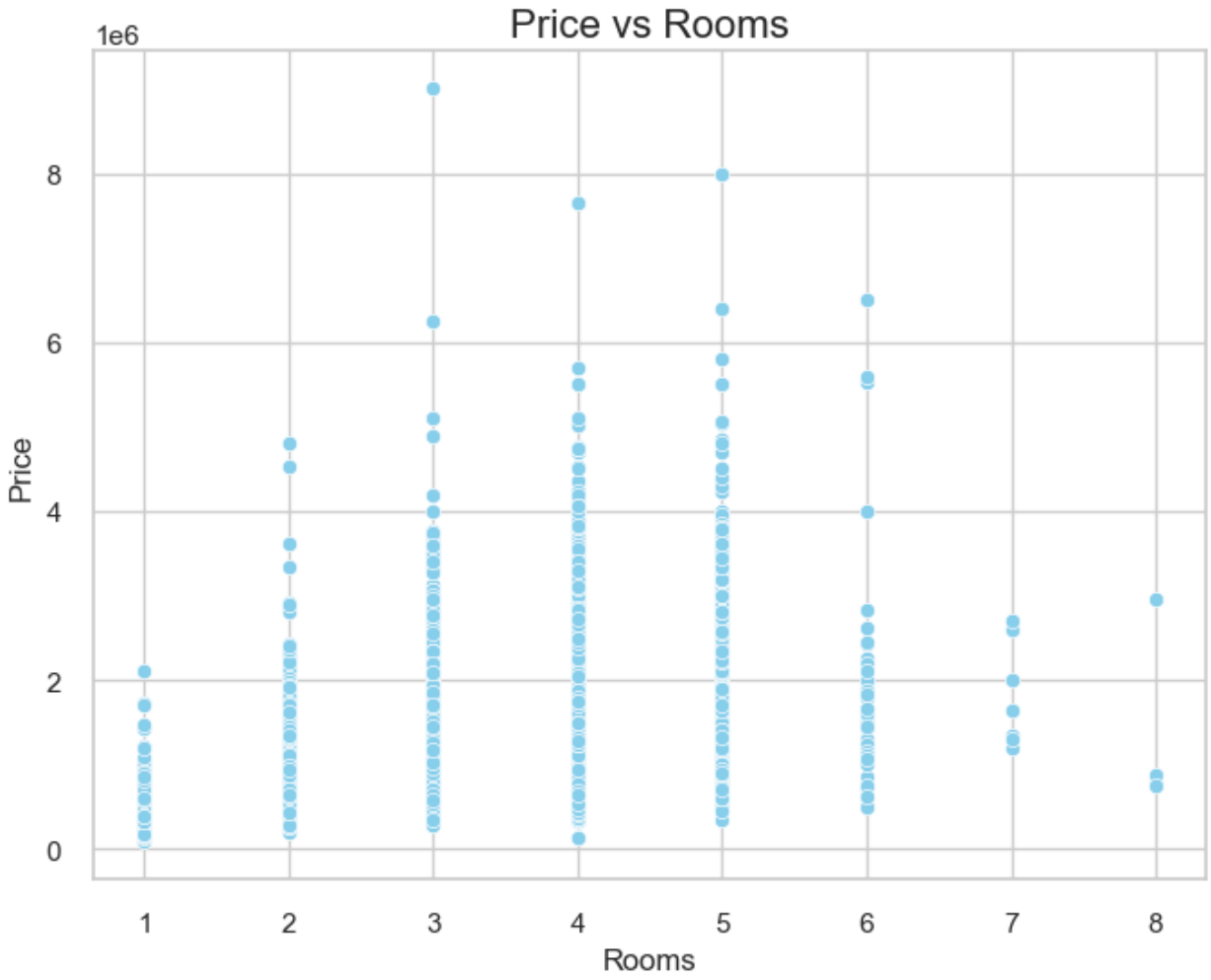
```
plt.figure(figsize=(12, 8))
numeric_features = list(numerical_without_price.columns) + ['Price']
correlation_matrix = housingdata[numeric_features].corr()

sb.heatmap(
    correlation_matrix,
    annot=True,
    cmap="coolwarm",
    fmt=".2f",
    linewidths=0.5
)
plt.title("Correlation Heatmap (Including Price)", fontsize=16)
plt.show()
```



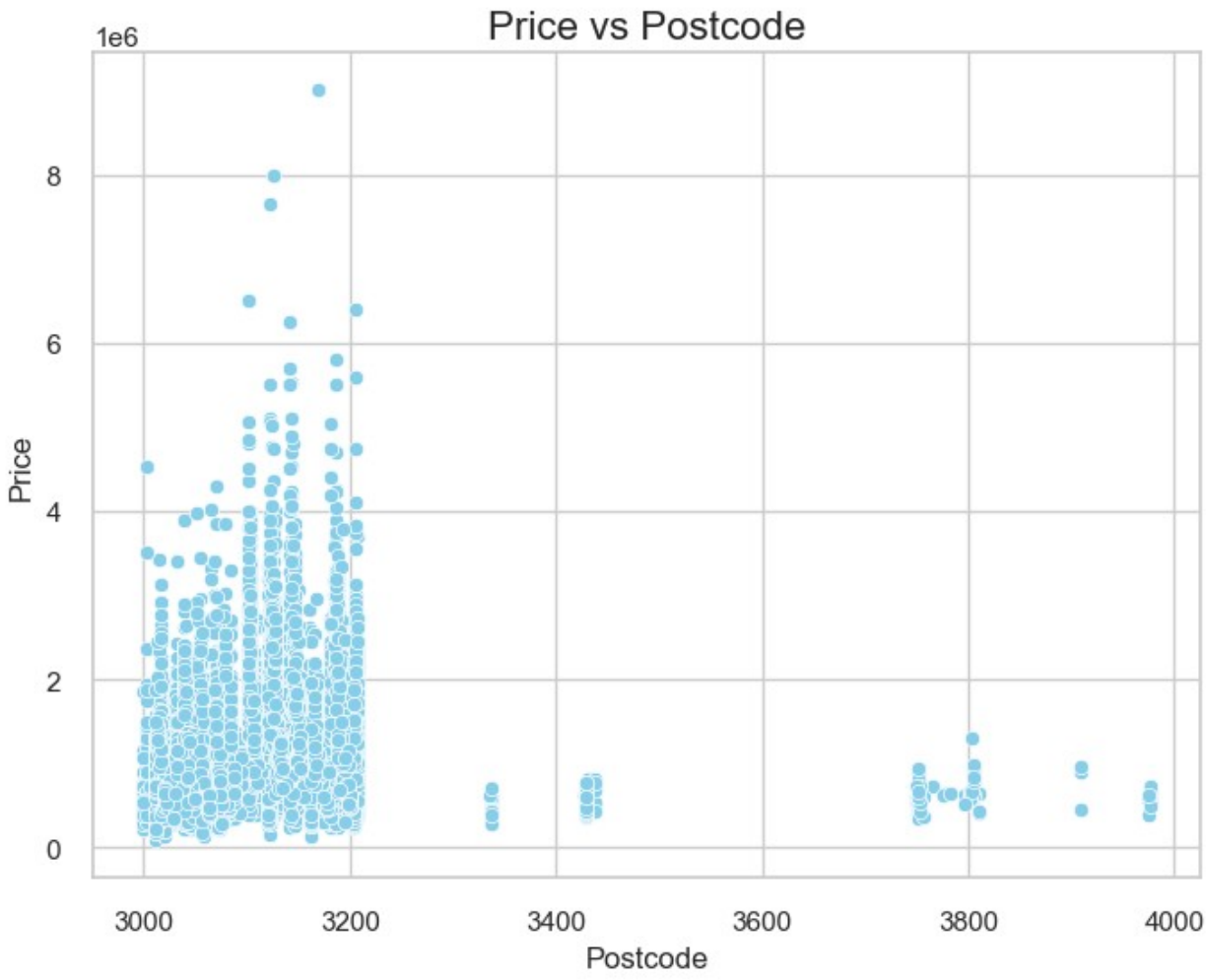
4.5 Scatter Plot of numerical features against Price

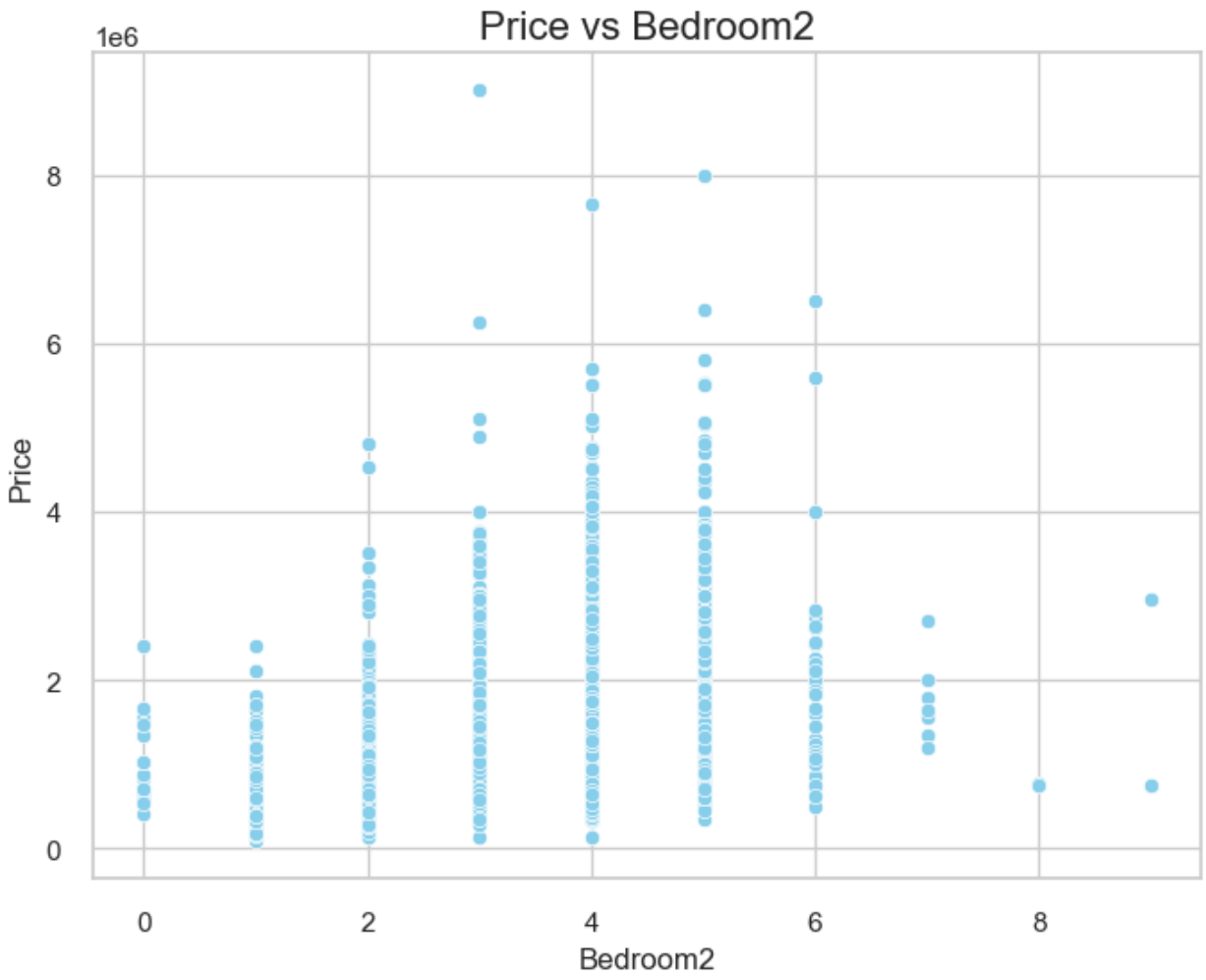
```
# Scatter Plots for Numerical Features excluding 'Price' against Price
for col in numerical_without_price.columns:
    plt.figure(figsize=(8, 6))
```

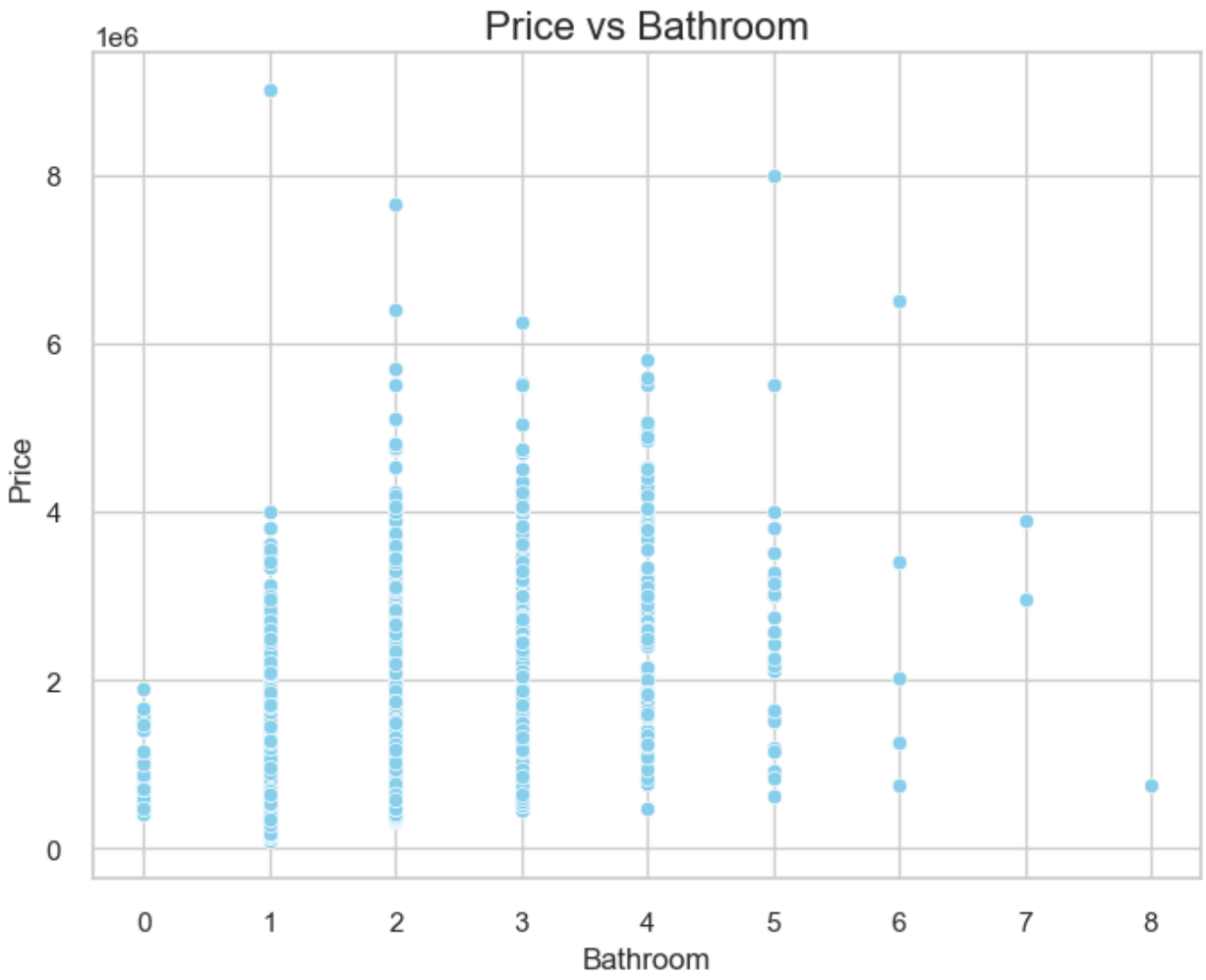
```
    sb.scatterplot(data=housingdata, x=col, y="Price",
color='skyblue')
    plt.title(f"Price vs {col}", fontsize=16)
    plt.show()
```

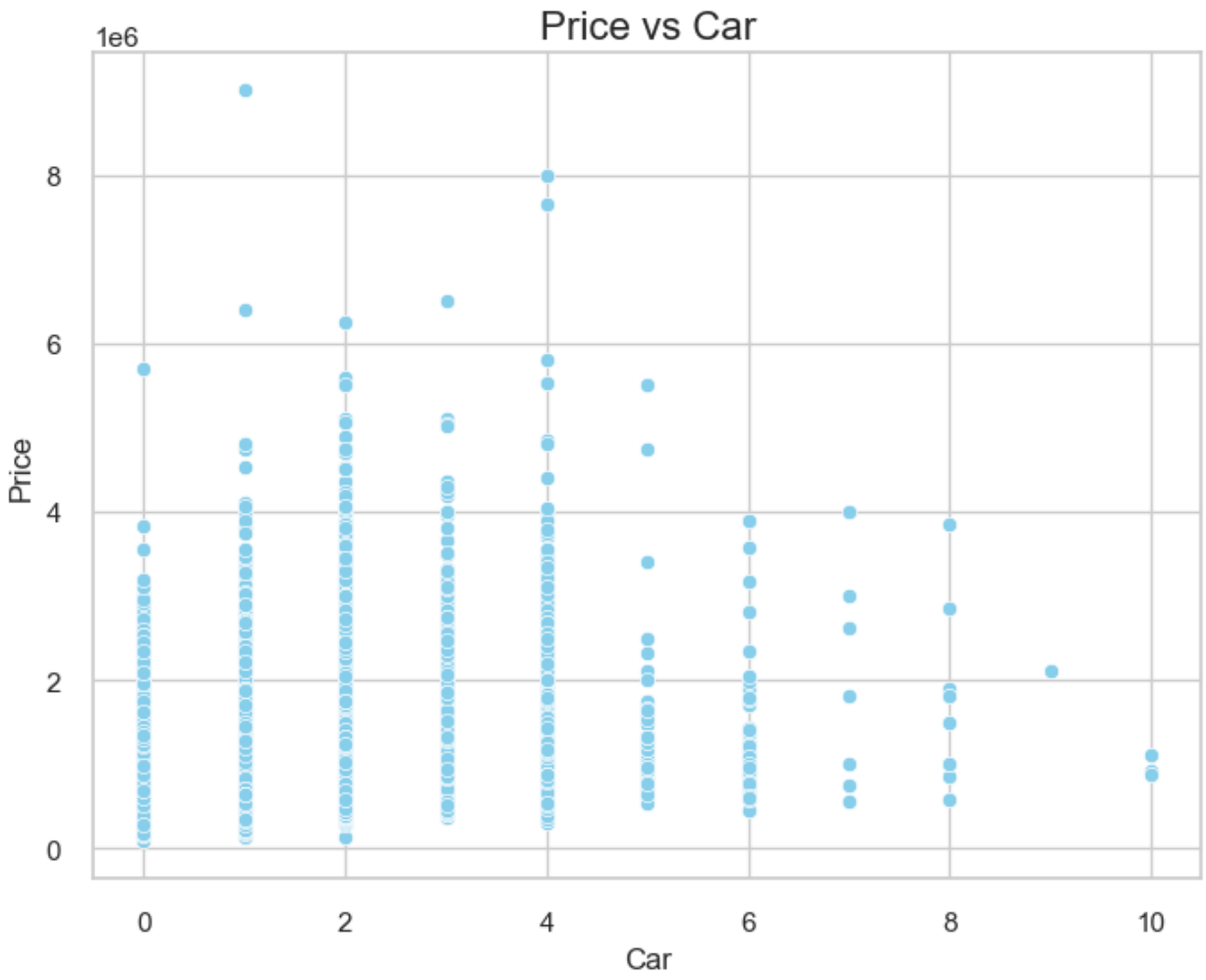


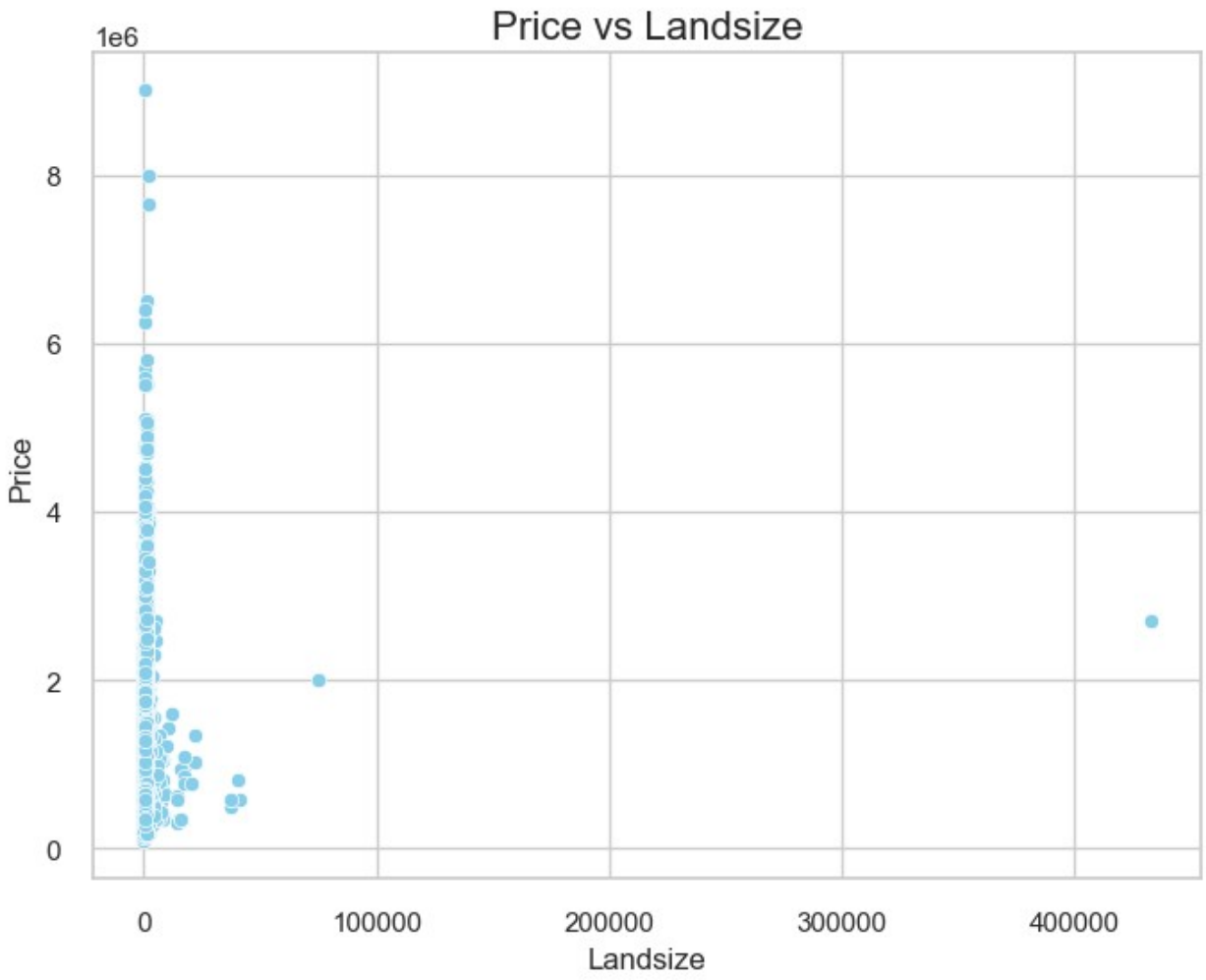

Price vs Distance
1e6
8
6
4
2
0
Price
0
10
20
30
40
Distance


Price vs Postcode
1e6
8
6
4
2
0
Price
3000
3200
3400
3600
3800
4000
Postcode

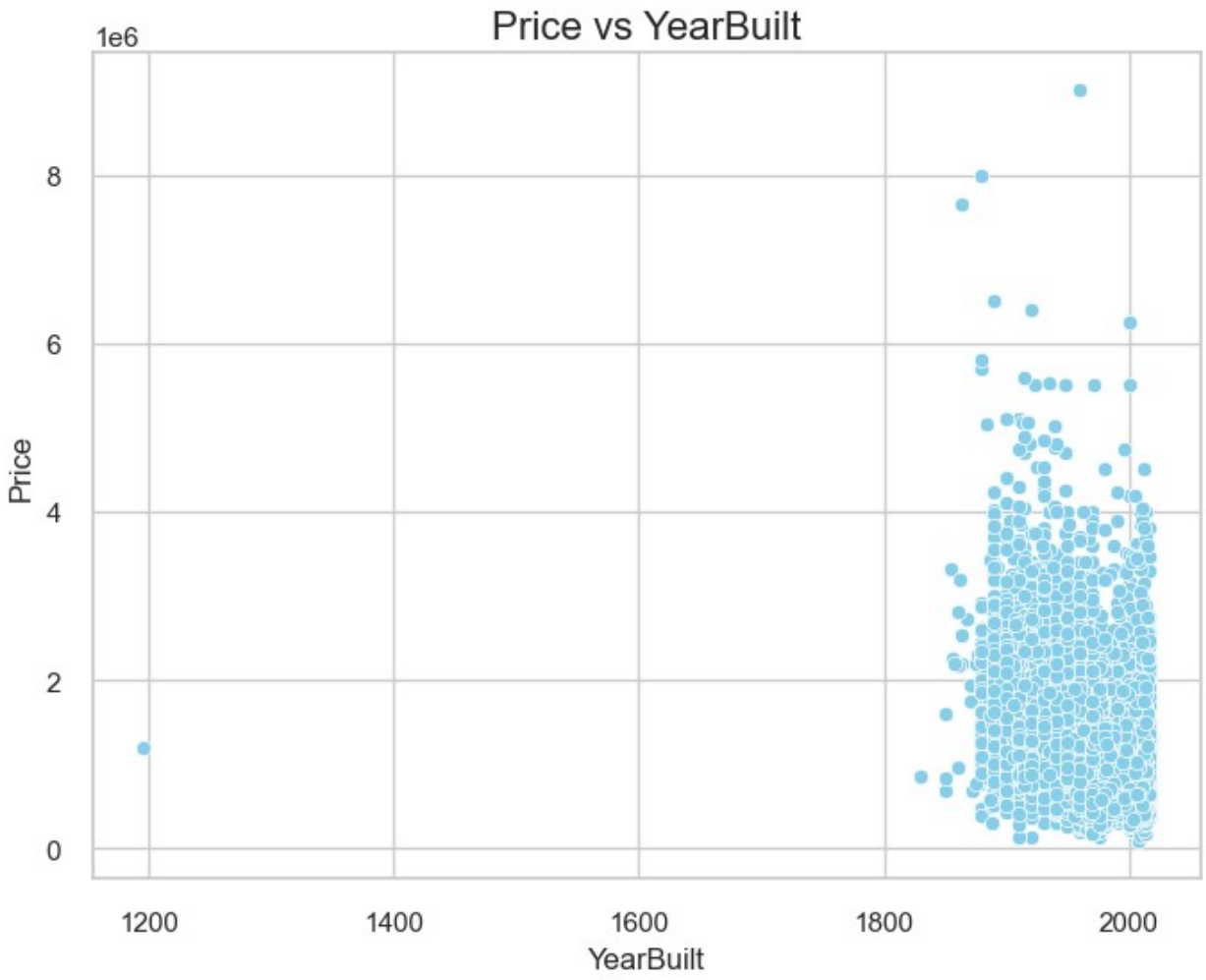

Price vs Bedroom2
1e6
8
6
4
2
0
Price
0
2
4
6
8
Bedroom2

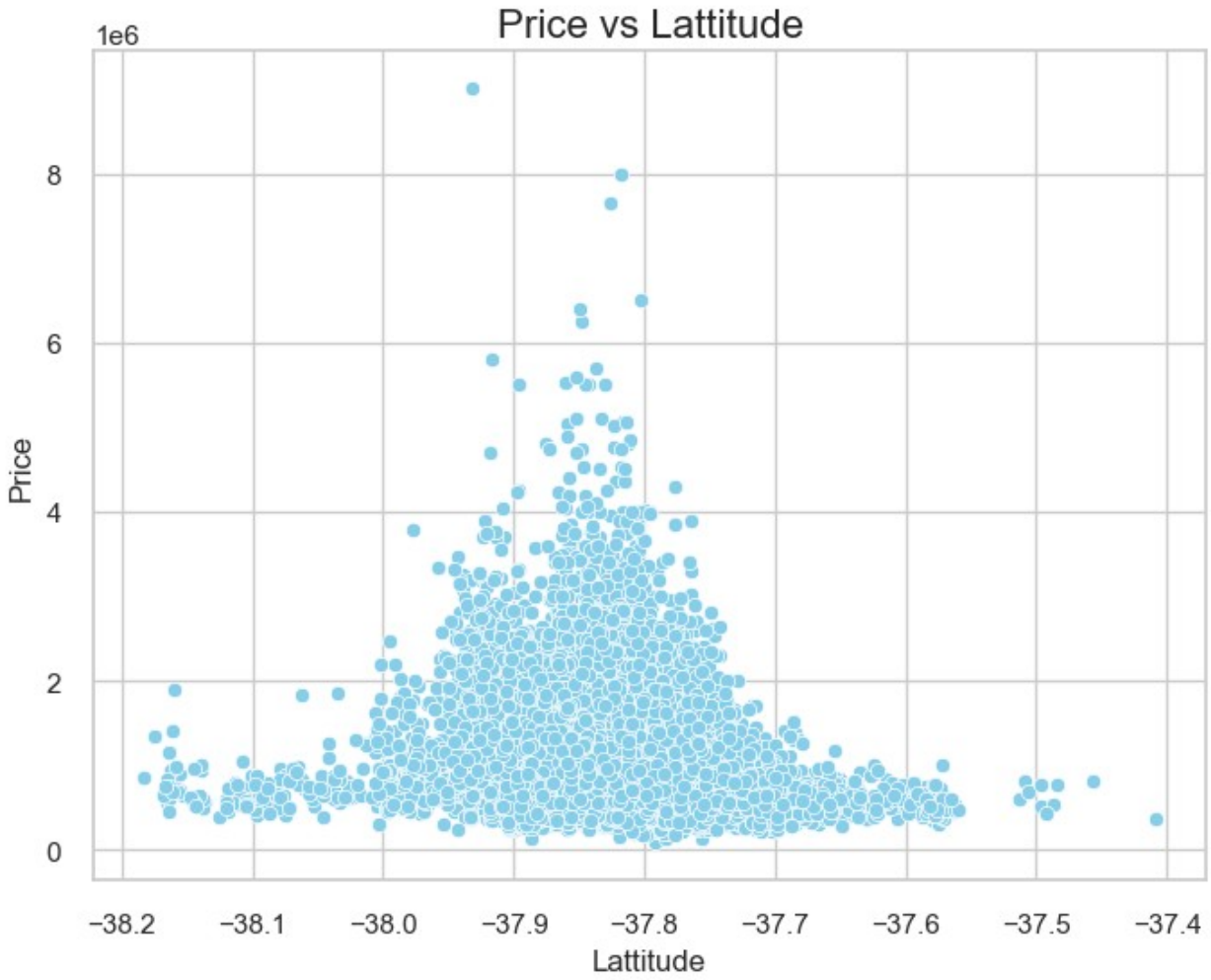

Price vs Bathroom
1e6
8
6
4
2
0
Price
0
1
2
3
4
5
6
7
8
Bathroom

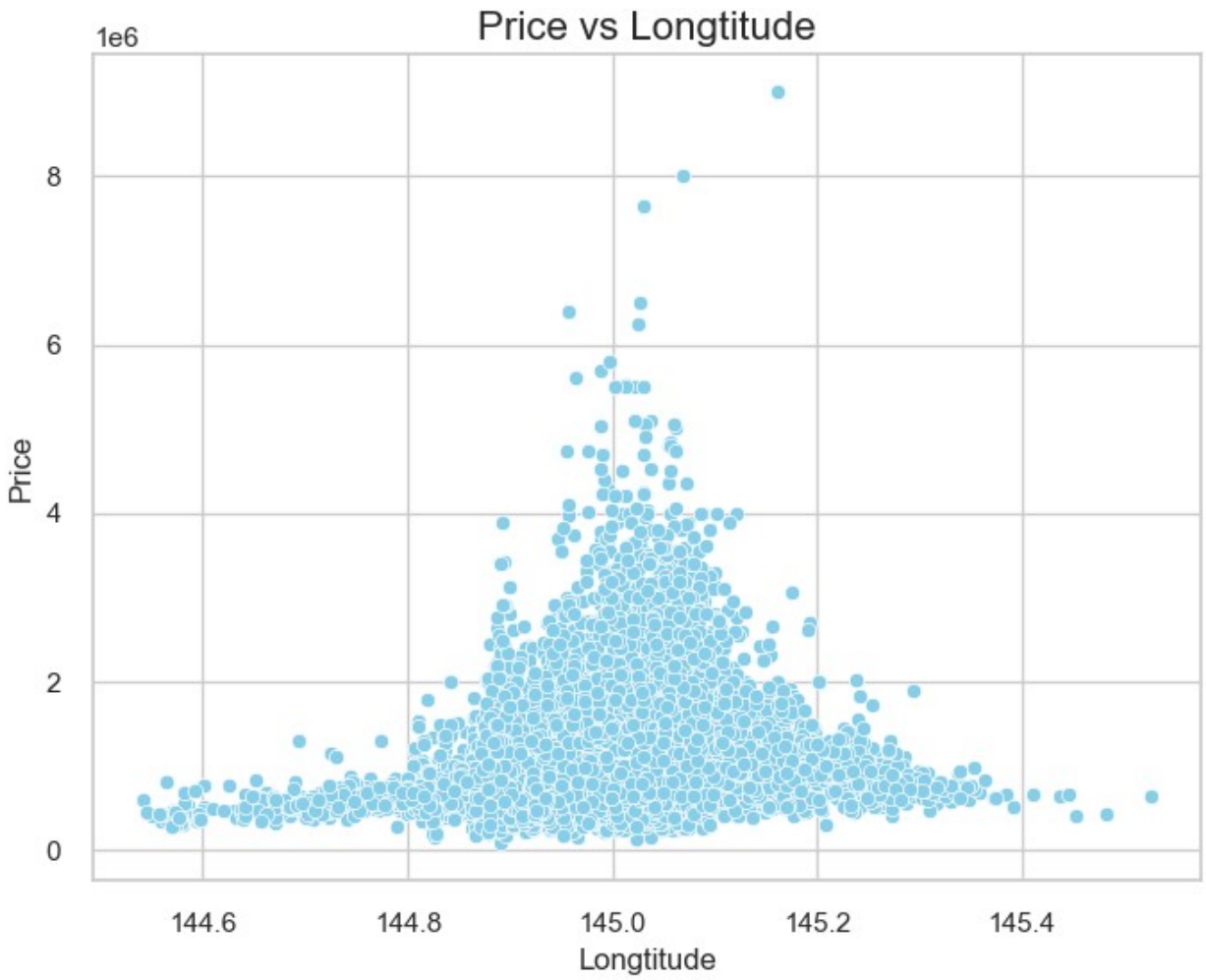

Price vs Car
1e6
8
6
4
2
0
Price
0
2
4
6
8
10
Car


Price vs Landsize
1e6
8
6
4
2
0
Price
0
100000
200000
300000
400000
Landsize


Price vs BuildingArea
1e6
8
6
4
2
0
Price
0
1000
2000
3000
4000
5000
6000
7000
BuildingArea


Price vs YearBuilt
1e6
8
6
4
2
0
Price
1200
1400
1600
1800
2000
YearBuilt


Price vs Lattitude
1e6
8
6
4
2
0
Price
−38.2
−38.1
−38.0
−37.9
−37.8
−37.7
−37.6
−37.5
−37.4
Lattitude


Price vs Longtitude
1e6
8
6
4
2
0
Price
144.6
144.8
145.0
145.2
145.4
Longtitude

5. Analysis of Categorical Variables

5.1 Distribution of each variable with a count plot 5.2 Box Plot of categorical variable and price

To recap, the categorical variables of the dataset are ['Suburb', 'Address', 'Type', 'Method', 'SellerG', 'Date', 'CouncilArea', 'Regionname'].

Before we begin analysis of the categorical variables, let's undo the encoding performed earlier for the Decision Tree Regression so we can obtain the original string values. Let's take a look at the description of the categorical variables as well.

```
housingdata[categorical] =
encoder.inverse_transform(housingdata[categorical])
housingdata[categorical].describe().T
```

```
             count unique                   top  freq
Suburb       13066    300             Reservoir   359
Address      13066  12874          28 Blair St     3
Type         13066      3                   h  9142
Method       13066      5                   S  8701
```

```
SellerG      13066    263                  Nelson  1532
Date         13066     58              27/05/2017   445
CouncilArea  13066     33                     nan  1252
Regionname   13066      8  Southern Metropolitan  4547
```

Additionally, let us perform frequency counts for each categorical variable.

```
print("\n=== Frequency Counts for Categorical Variables ===\n")
for col in categorical:
    print(f"--- {col} ---")
    print(housingdata[col].value_counts())
    print("\n")
```

```
=== Frequency Counts for Categorical Variables ===

--- Suburb ---
Suburb
Reservoir          359
Richmond           259
Bentleigh East     249
Preston            237
Brunswick          221
                  ...
Whittlesea           1
Wallan               1
Campbellfield        1
Plumpton             1
Kooyong              1
Name: count, Length: 300, dtype: int64


--- Address ---
Address
28 Blair St        3
14 Arthur St       3
13 Robinson St     3
2 Bruce St         3
5 Margaret St      3
                  ..
20 Airds Rd        1
16 Benambra Dr     1
50 Caroline Dr     1
7 Dove Ct          1
129 Charles St     1
Name: count, Length: 12874, dtype: int64


--- Type ---
Type
```

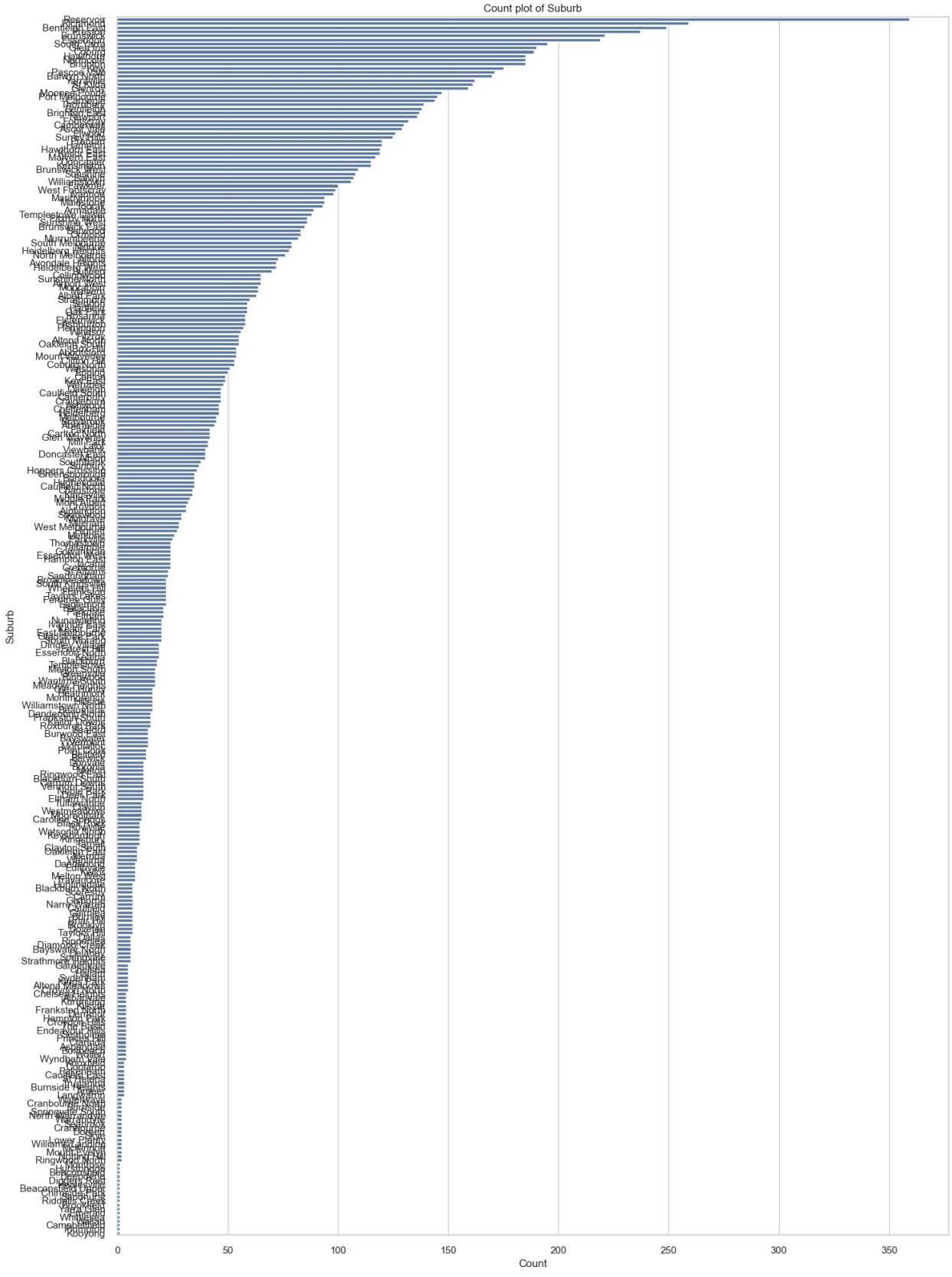
```
h    9142
u    2903
t    1021
Name: count, dtype: int64


--- Method ---
Method
S     8701
SP    1634
PI    1509
VB    1134
SA      88
Name: count, dtype: int64


--- SellerG ---
SellerG
Nelson           1532
Jellis           1277
hockingstuart    1131
Barry             972
Ray               669
                 ...
Area                1
SN                  1
PRD                 1
Rexhepi             1
Point               1
Name: count, Length: 263, dtype: int64


--- Date ---
Date
27/05/2017    445
3/06/2017     377
12/08/2017    360
27/11/2016    359
17/06/2017    343
4/03/2017     329
25/02/2017    326
29/07/2017    317
10/12/2016    313
24/06/2017    312
22/07/2017    307
8/07/2017     297
15/10/2016    295
18/03/2017    288
12/11/2016    287
3/12/2016     282
```

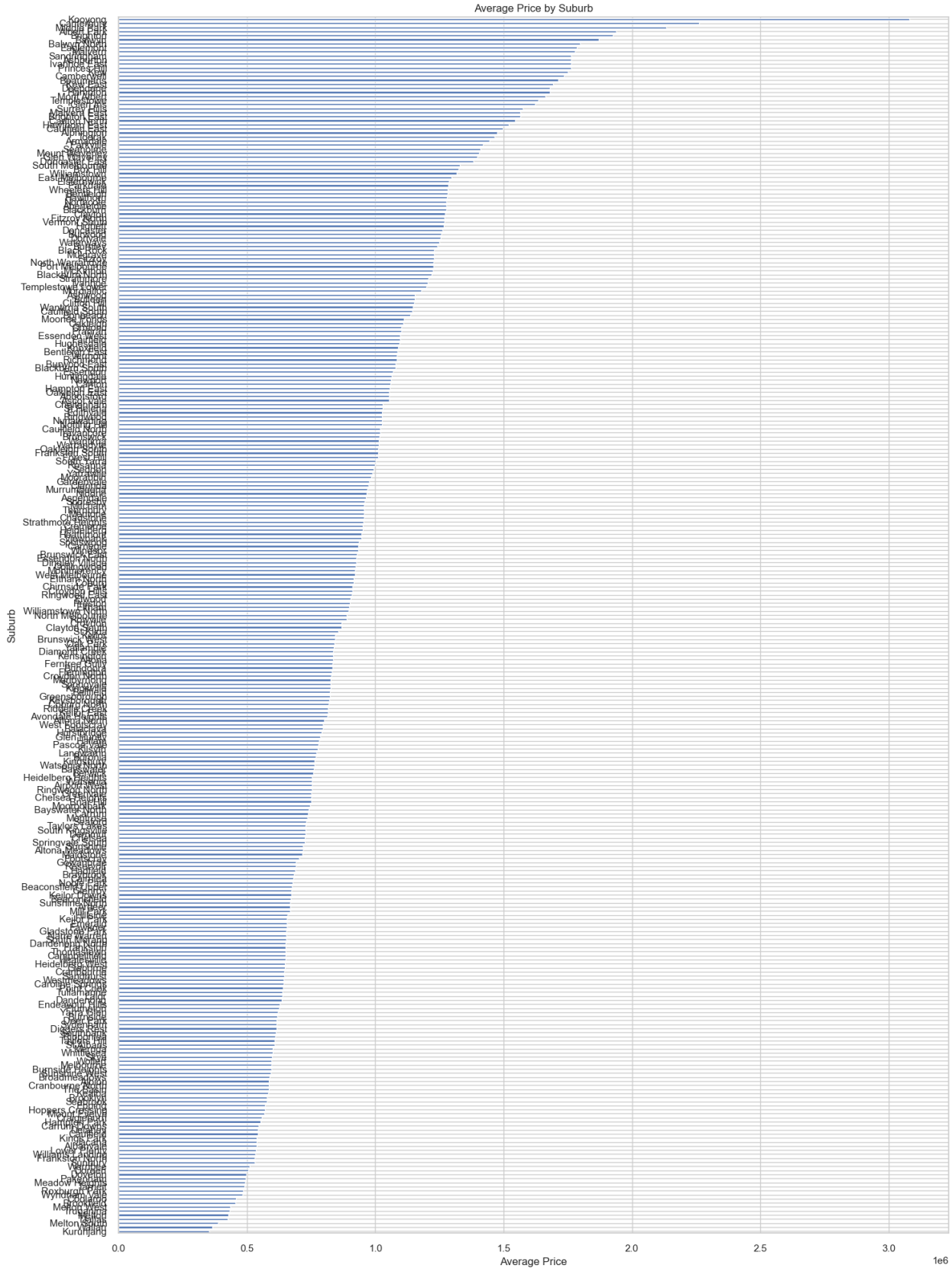
```
8/04/2017     276
15/07/2017    274
19/11/2016    263
20/05/2017    255
18/06/2016    255
1/07/2017     254
28/05/2016    254
13/05/2017    247
8/10/2016     244
7/05/2016     242
17/09/2016    240
16/09/2017    233
24/09/2016    232
28/08/2016    232
26/08/2017    229
10/09/2016    229
23/09/2017    224
16/04/2016    214
29/04/2017    210
4/06/2016     209
27/06/2016    206
3/09/2016     204
22/05/2016    200
3/09/2017     194
7/11/2016     193
22/08/2016    191
19/08/2017    189
14/05/2016    187
9/09/2017     181
30/07/2016    163
6/08/2016     152
6/05/2017     136
16/07/2016    136
13/08/2016    131
26/07/2016    131
22/04/2017    130
23/04/2016     99
11/02/2017     82
12/06/2016     45
11/03/2017     35
4/02/2016      26
28/01/2016      2
Name: count, dtype: int64


--- CouncilArea ---
CouncilArea
nan                1252
Moreland           1148
```

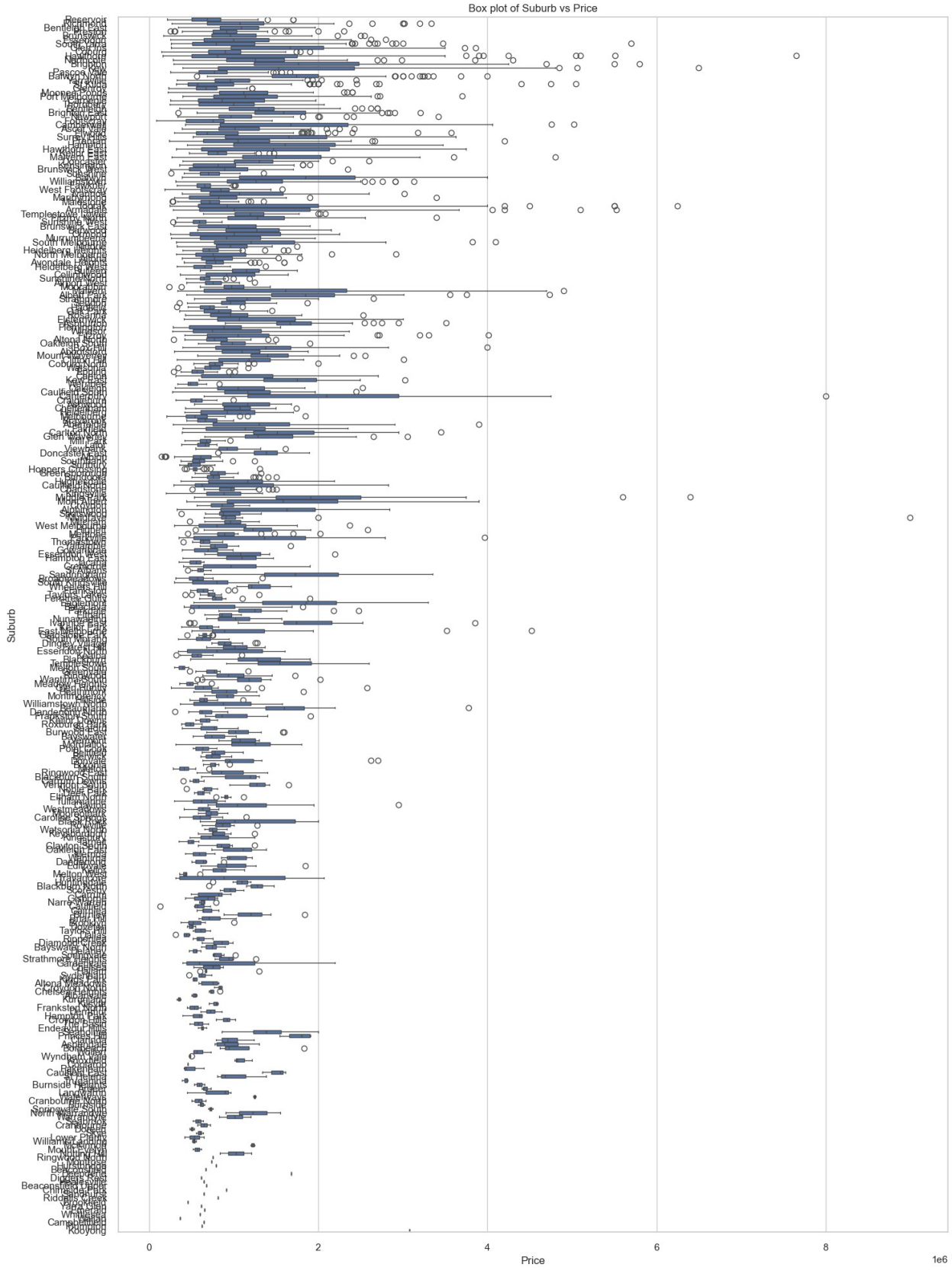
```
Boroondara            1135
Moonee Valley          971
Darebin                926
Glen Eira              826
Stonnington            699
Maribyrnong            681
Yarra                  633
Port Phillip           610
Banyule                567
Bayside                470
Melbourne              446
Hobsons Bay            423
Brimbank               411
Monash                 320
Manningham             301
Whitehorse             284
Kingston               191
Hume                   154
Whittlesea             153
Wyndham                 83
Knox                    71
Maroondah               65
Melton                  61
Frankston               48
Greater Dandenong       43
Casey                   36
Nillumbik               32
Yarra Ranges            13
Macedon Ranges           6
Cardinia                 6
Unavailable              1
Name: count, dtype: int64


--- Regionname ---
Regionname
Southern Metropolitan         4547
Northern Metropolitan         3779
Western Metropolitan          2866
Eastern Metropolitan          1366
South-Eastern Metropolitan     404
Eastern Victoria                41
Northern Victoria               33
Western Victoria                30
Name: count, dtype: int64
```

Analysis of 'Suburb' Variable

```
# Count plot for Suburb
plt.figure(figsize=(15, 20))
sb.countplot(data=housingdata, y='Suburb',
order=housingdata['Suburb'].value_counts().index)
plt.title('Count plot of Suburb')
plt.xlabel('Count')
plt.ylabel('Suburb')
plt.tight_layout()
plt.show()

# Average price bar plot for Suburb
# We'll compute the average price per suburb and plot it
avg_price = housingdata.groupby('Suburb')
['Price'].mean().sort_values(ascending=True)

plt.figure(figsize=(15, 20))
avg_price.plot(kind='barh')
plt.title('Average Price by Suburb')
plt.xlabel('Average Price')
plt.ylabel('Suburb')
plt.tight_layout()
plt.show()

# Box plot of Price distribution by Suburb
plt.figure(figsize=(15, 20))
sb.boxplot(data=housingdata, y='Suburb', x='Price',
order=housingdata['Suburb'].value_counts().index)
plt.title('Box plot of Suburb vs Price')
plt.xlabel('Price')
plt.ylabel('Suburb')
plt.tight_layout()
plt.show()
```


Count plot of Suburb
Suburb
Count
0
50
100
150
200
250
300
350


Average Price by Suburb
Suburb
Average Price
0.0
0.5
1.0
1.5
2.0
2.5
3.0
1e6


Box plot of Suburb vs Price
Suburb
Price
0
2
4
6
8
1e6

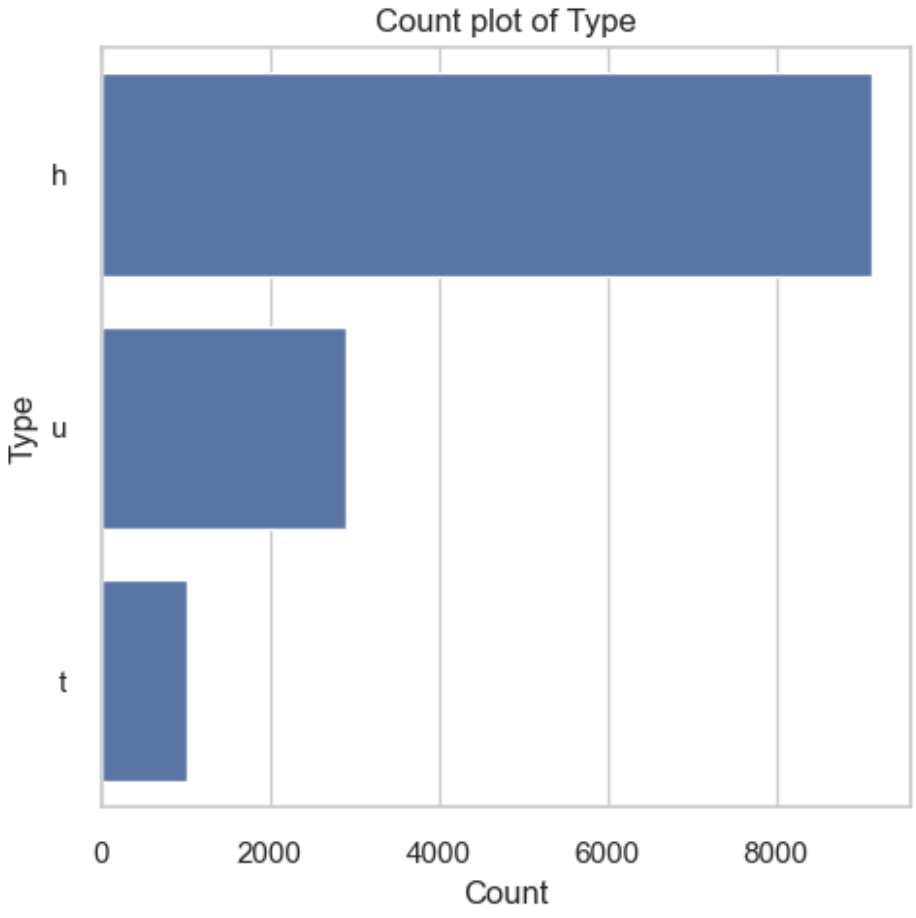
Analysis of 'Address' Variable

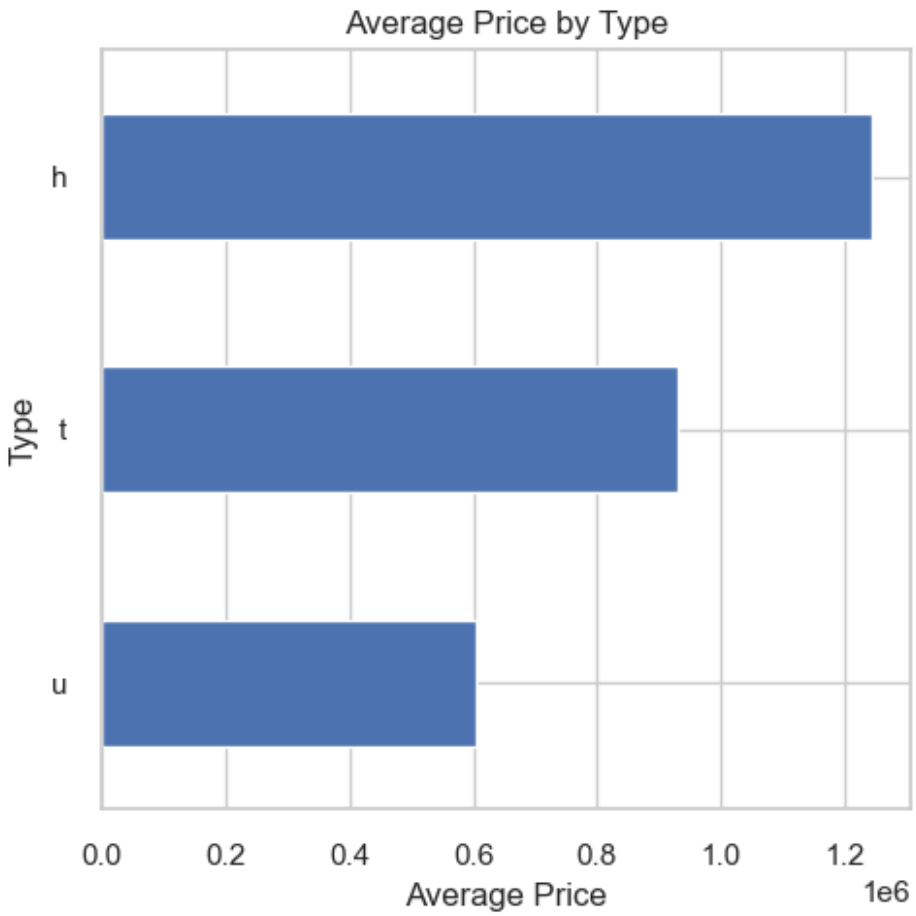
```
# # Count plot for Address
# plt.figure(figsize=(15, 20))
# sb.countplot(data=housingdata, y='Address',
order=housingdata['Address'].value_counts().index)
# plt.title('Count plot of Address')
# plt.xlabel('Count')
# plt.ylabel('Address')
# plt.tight_layout()
# plt.show()

# # Average price bar plot for Address
# # We'll compute the average price per Address and plot it
# avg_price = housingdata.groupby('Address')
['Price'].mean().sort_values(ascending=True)

# plt.figure(figsize=(15, 20))
# avg_price.plot(kind='barh')
# plt.title('Average Price by Address')
# plt.xlabel('Average Price')
# plt.ylabel('Address')
# plt.tight_layout()
# plt.show()

# # Box plot of Price distribution by Address
# plt.figure(figsize=(15, 20))
# sb.boxplot(data=housingdata, y='Address', x='Price',
order=housingdata['Address'].value_counts().index)
# plt.title('Box plot of Address vs Price')
# plt.xlabel('Price')
# plt.ylabel('Address')
# plt.tight_layout()
# plt.show()
```

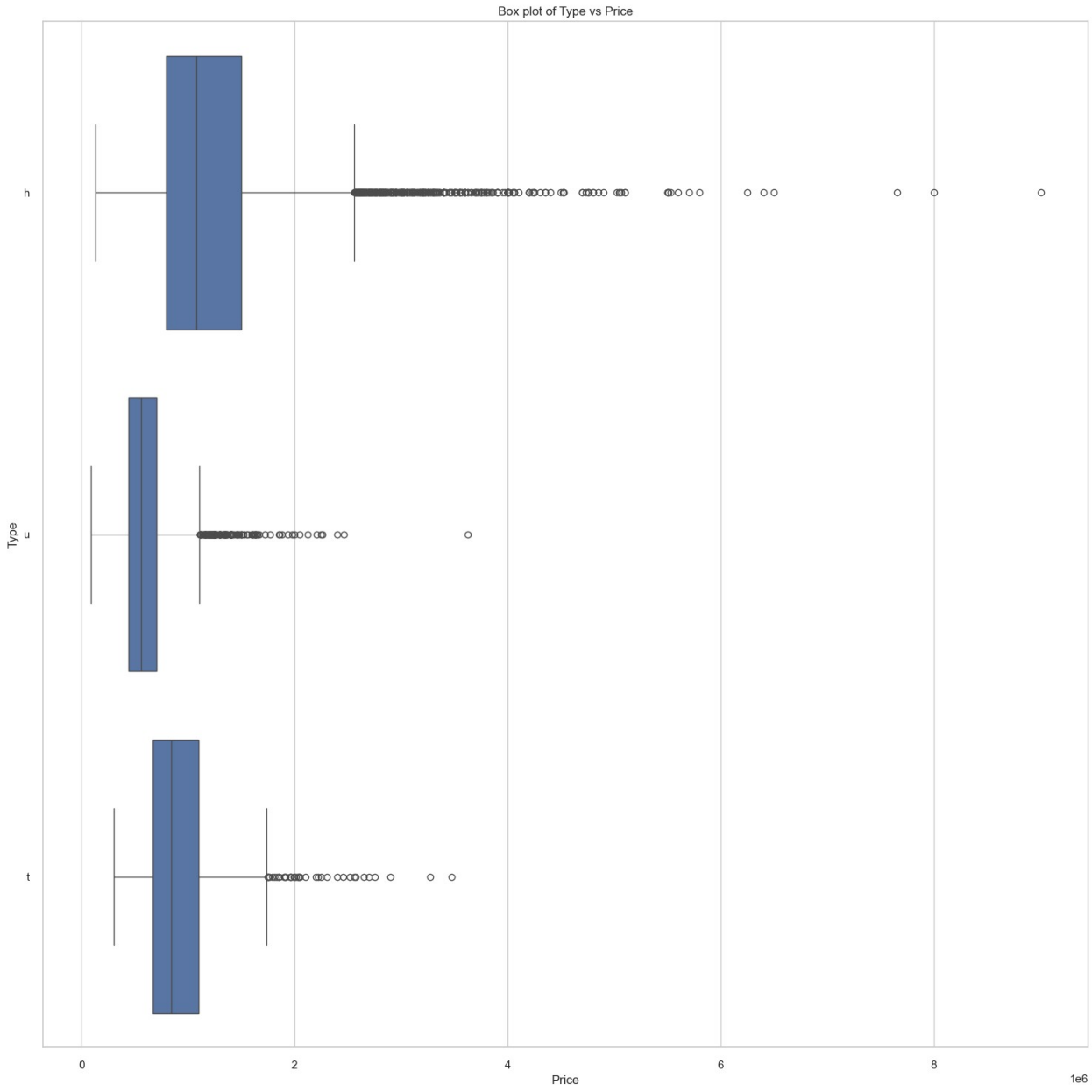
Analysis of 'Type' Variable

```
# Count plot for Type
plt.figure(figsize=(5,5))
sb.countplot(data=housingdata, y='Type',
order=housingdata['Type'].value_counts().index)
plt.title('Count plot of Type')
plt.xlabel('Count')
plt.ylabel('Type')
plt.tight_layout()
plt.show()

# Average price bar plot for Type
# We'll compute the average price per Type and plot it
avg_price = housingdata.groupby('Type')
['Price'].mean().sort_values(ascending=True)
```

```
plt.figure(figsize=(5,5))
avg_price.plot(kind='barh')
plt.title('Average Price by Type')
plt.xlabel('Average Price')
plt.ylabel('Type')
plt.tight_layout()
plt.show()

# Box plot of Price distribution by Type
plt.figure(figsize=(15,15))
sb.boxplot(data=housingdata, y='Type', x='Price',
order=housingdata['Type'].value_counts().index)
plt.title('Box plot of Type vs Price')
plt.xlabel('Price')
plt.ylabel('Type')
plt.tight_layout()
plt.show()
```



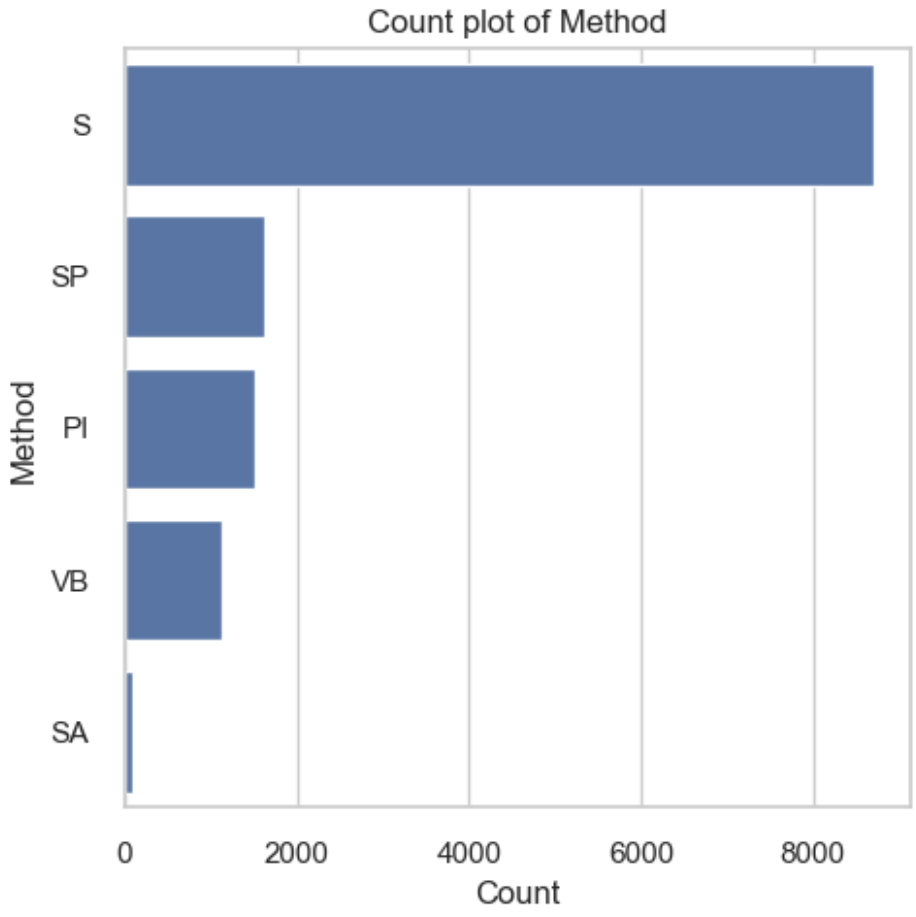

Average Price by Type
Type
h
t
u
0.0
0.2
0.4
0.6
0.8
1.0
1.2
1e6
Average Price

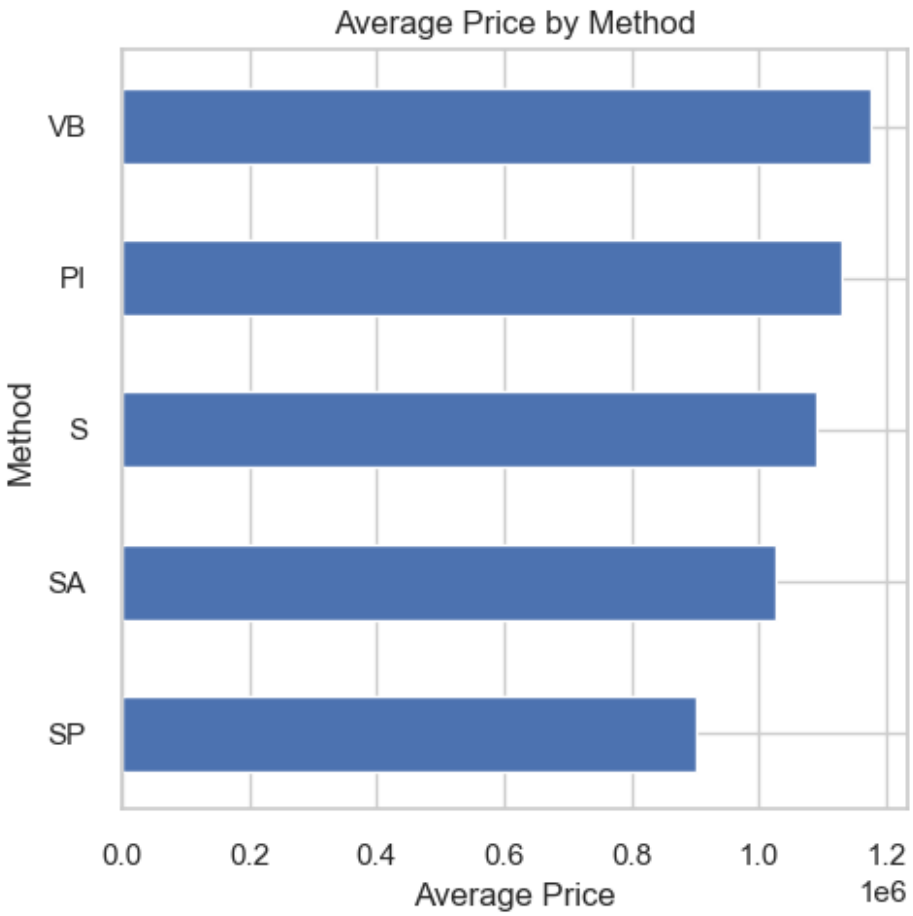


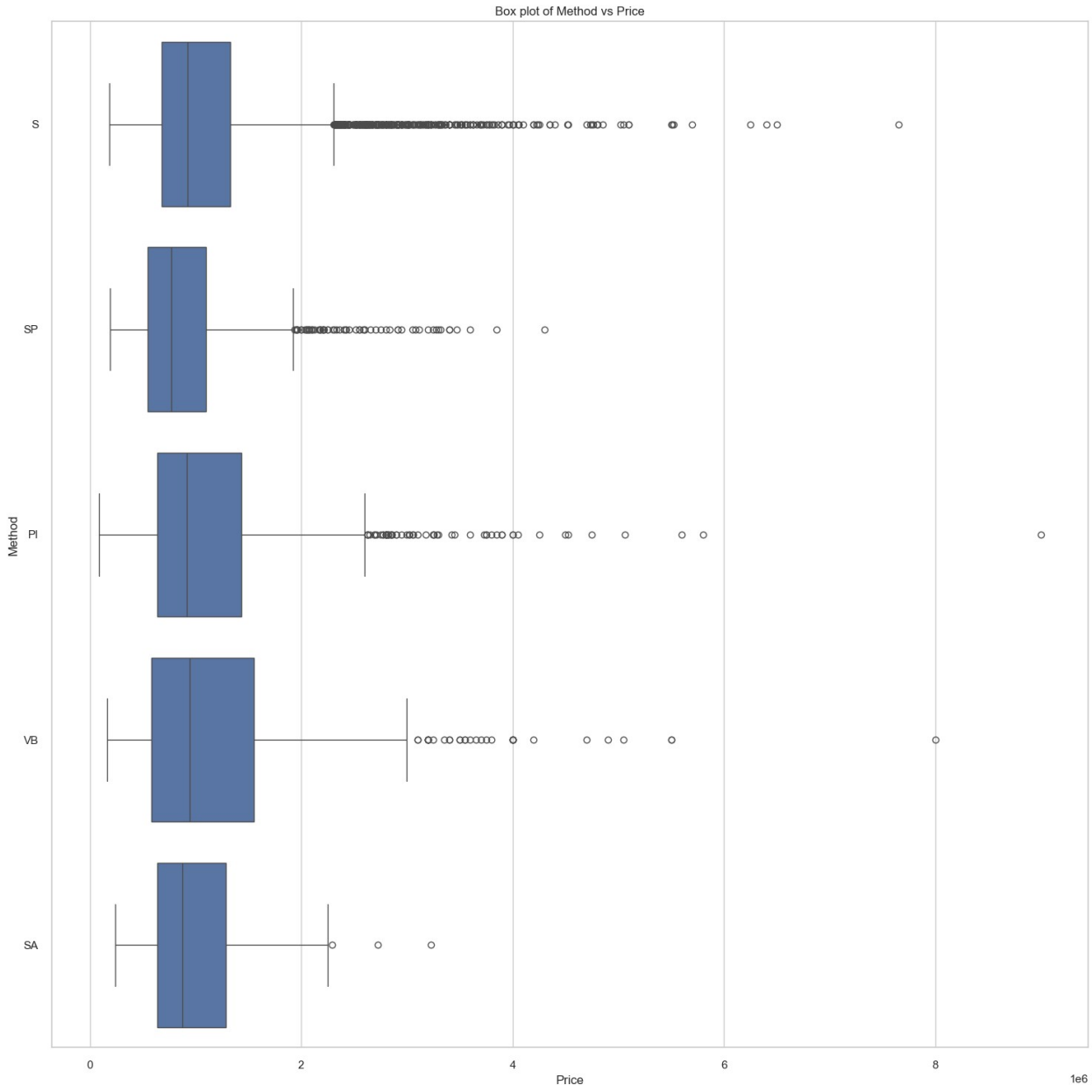
Analysis of 'Method' Variable

```
# Count plot for Method
plt.figure(figsize=(5,5))
sb.countplot(data=housingdata, y='Method',
order=housingdata['Method'].value_counts().index)
plt.title('Count plot of Method')
plt.xlabel('Count')
plt.ylabel('Method')
plt.tight_layout()
plt.show()

# Average price bar plot for Method
```

```
# We'll compute the average price per Method and plot it
avg_price = housingdata.groupby('Method')
['Price'].mean().sort_values(ascending=True)

plt.figure(figsize=(5,5))
avg_price.plot(kind='barh')
plt.title('Average Price by Method')
plt.xlabel('Average Price')
plt.ylabel('Method')
plt.tight_layout()
plt.show()

# Box plot of Price distribution by Method
plt.figure(figsize=(15,15))
sb.boxplot(data=housingdata, y='Method', x='Price',
order=housingdata['Method'].value_counts().index)
plt.title('Box plot of Method vs Price')
plt.xlabel('Price')
plt.ylabel('Method')
plt.tight_layout()
plt.show()
```



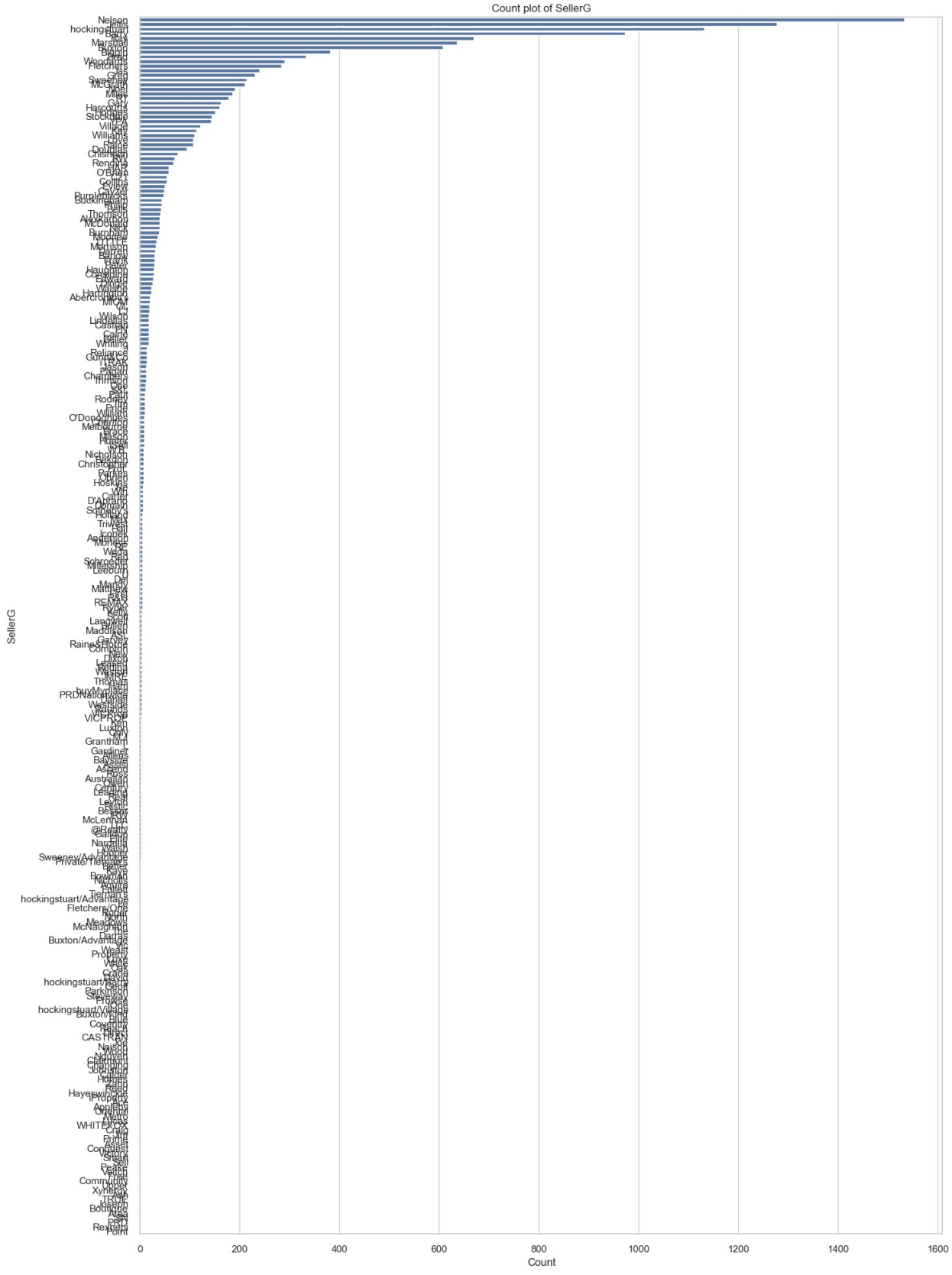

Average Price by Method
VB
PI
S
SA
SP
Method
0.0
0.2
0.4
0.6
0.8
1.0
1.2
1e6
Average Price

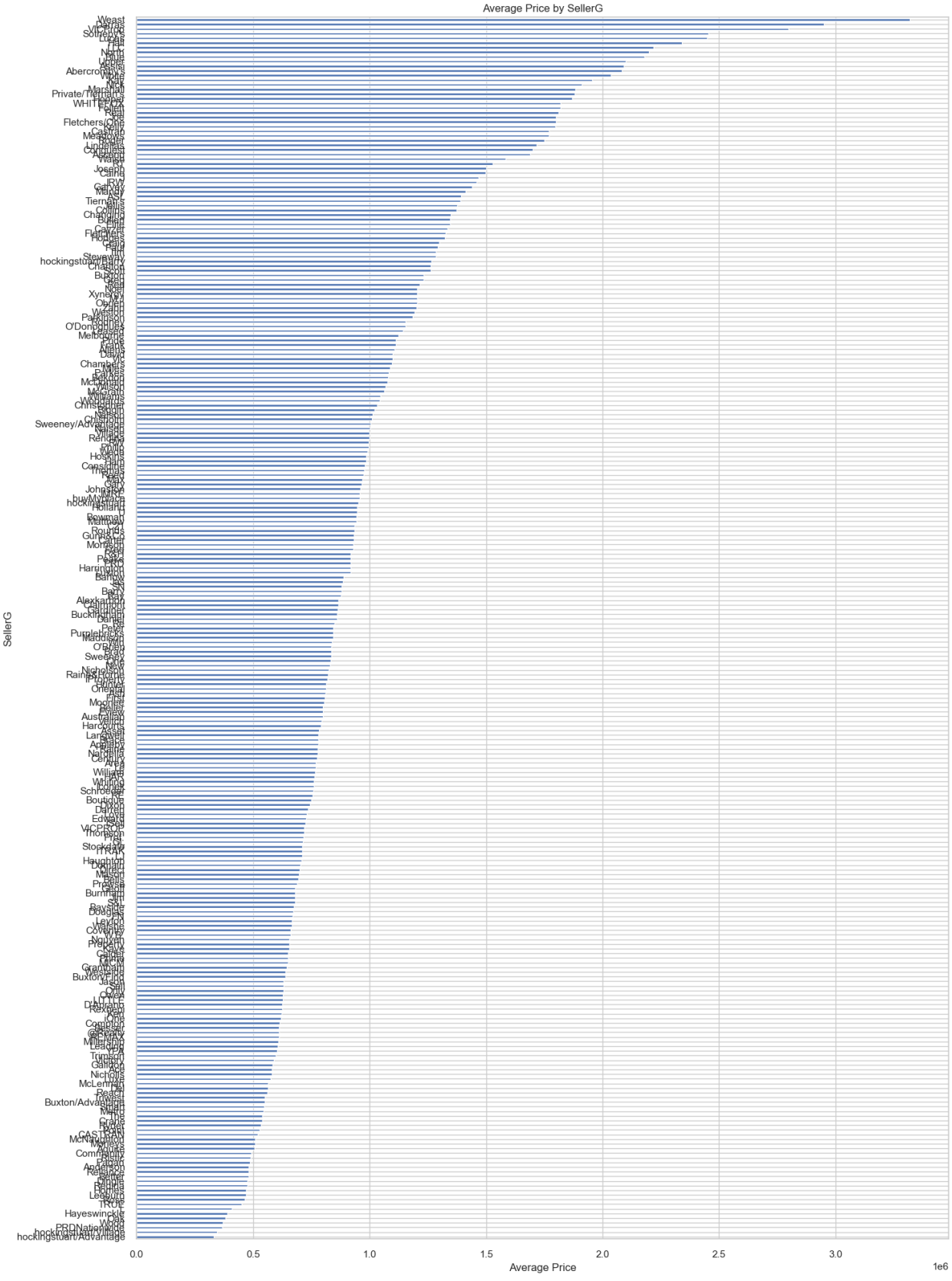


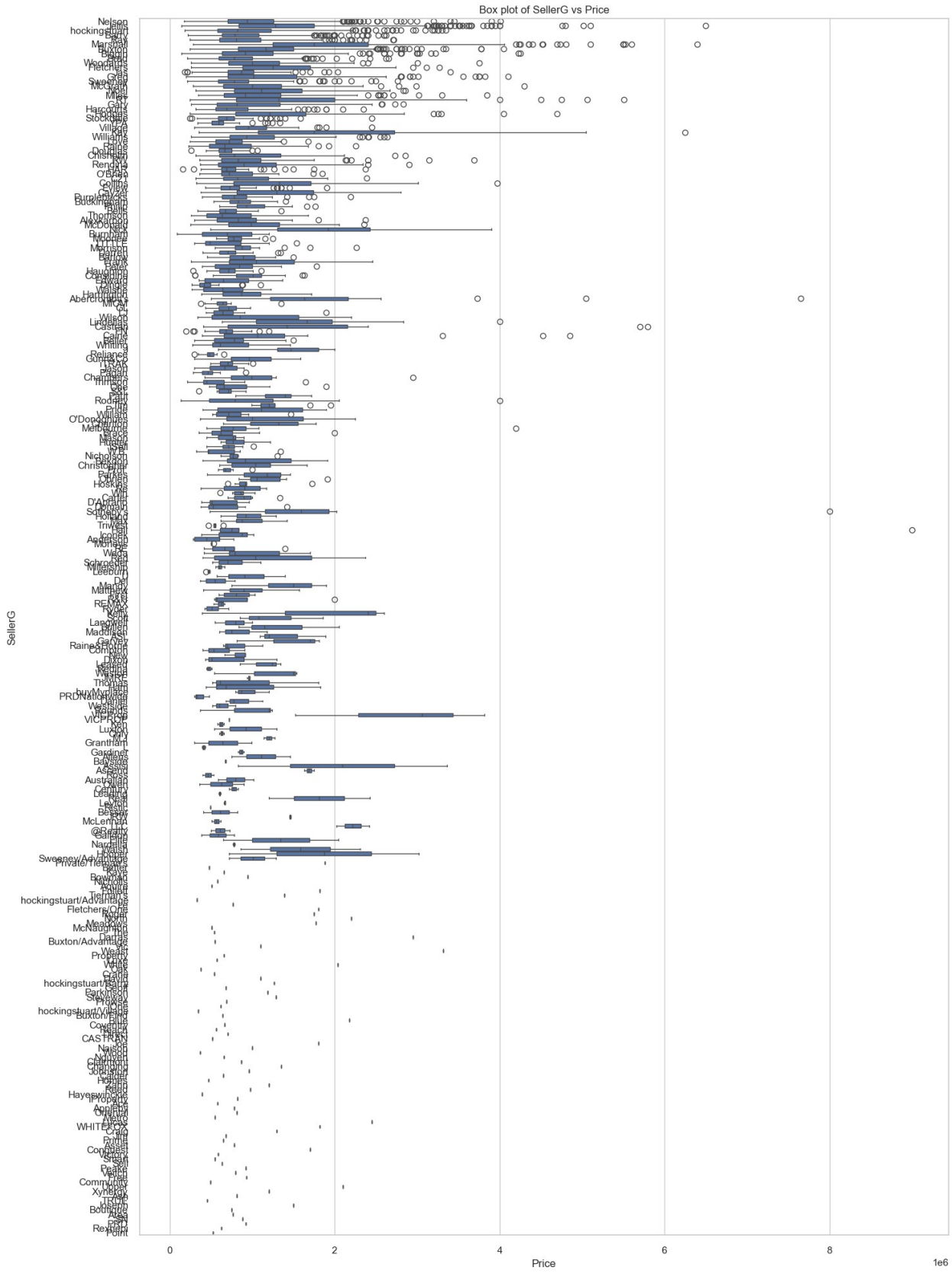
Analysis of 'SellerG' Variable

```
# Count plot for SellerG
plt.figure(figsize=(15,20))
sb.countplot(data=housingdata, y='SellerG',
order=housingdata['SellerG'].value_counts().index)
plt.title('Count plot of SellerG')
plt.xlabel('Count')
plt.ylabel('SellerG')
plt.tight_layout()
plt.show()

# Average price bar plot for SellerG
```

```
# We'll compute the average price per SellerG and plot it
avg_price = housingdata.groupby('SellerG')
['Price'].mean().sort_values(ascending=True)

plt.figure(figsize=(15,20))
avg_price.plot(kind='barh')
plt.title('Average Price by SellerG')
plt.xlabel('Average Price')
plt.ylabel('SellerG')
plt.tight_layout()
plt.show()

# Box plot of Price distribution by SellerG
plt.figure(figsize=(15,20))
sb.boxplot(data=housingdata, y='SellerG', x='Price',
order=housingdata['SellerG'].value_counts().index)
plt.title('Box plot of SellerG vs Price')
plt.xlabel('Price')
plt.ylabel('SellerG')
plt.tight_layout()
plt.show()
```

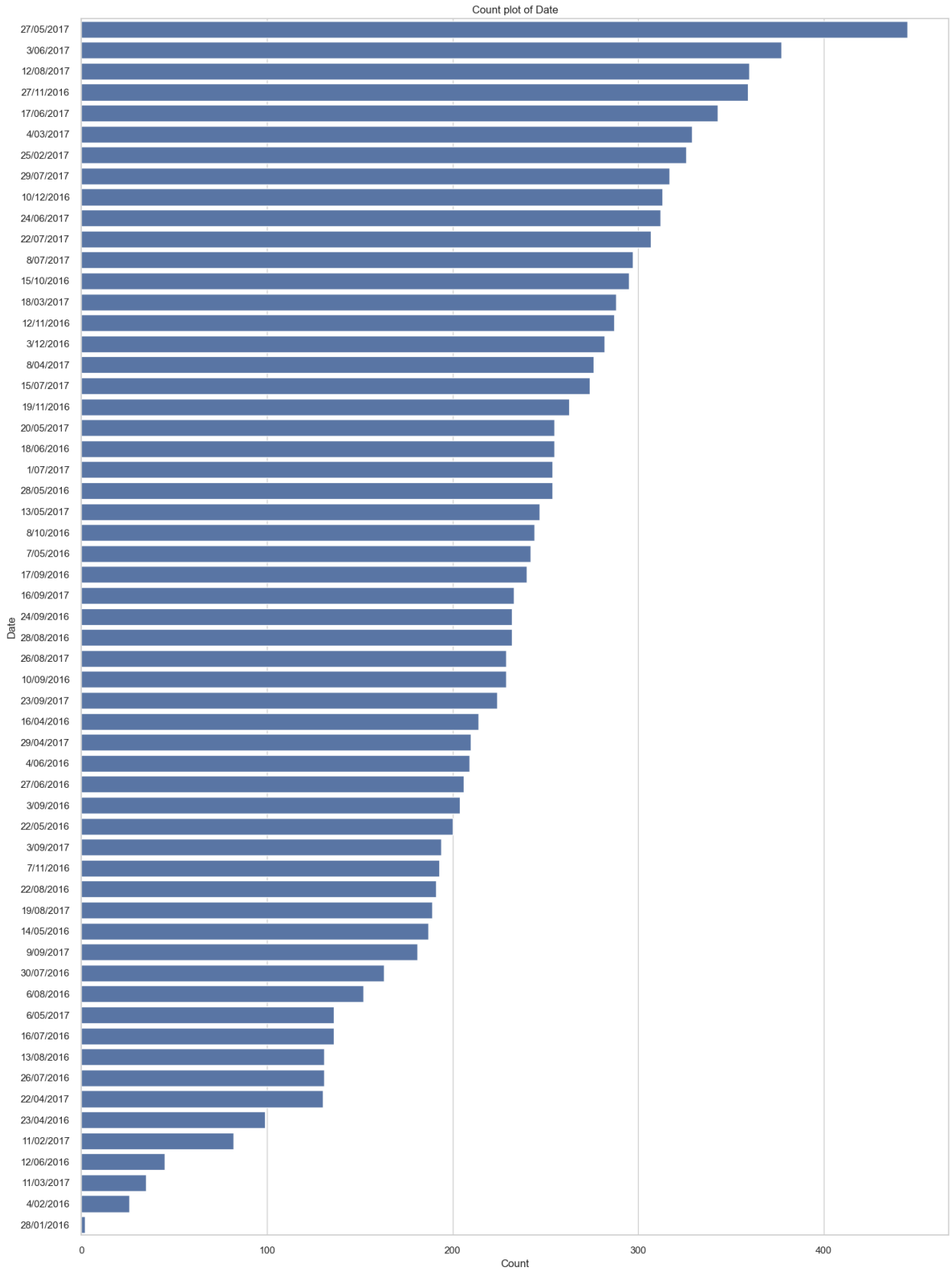

Count plot of SellerG
SellerG
Count
0
200
400
600
800
1000
1200
1400
1600

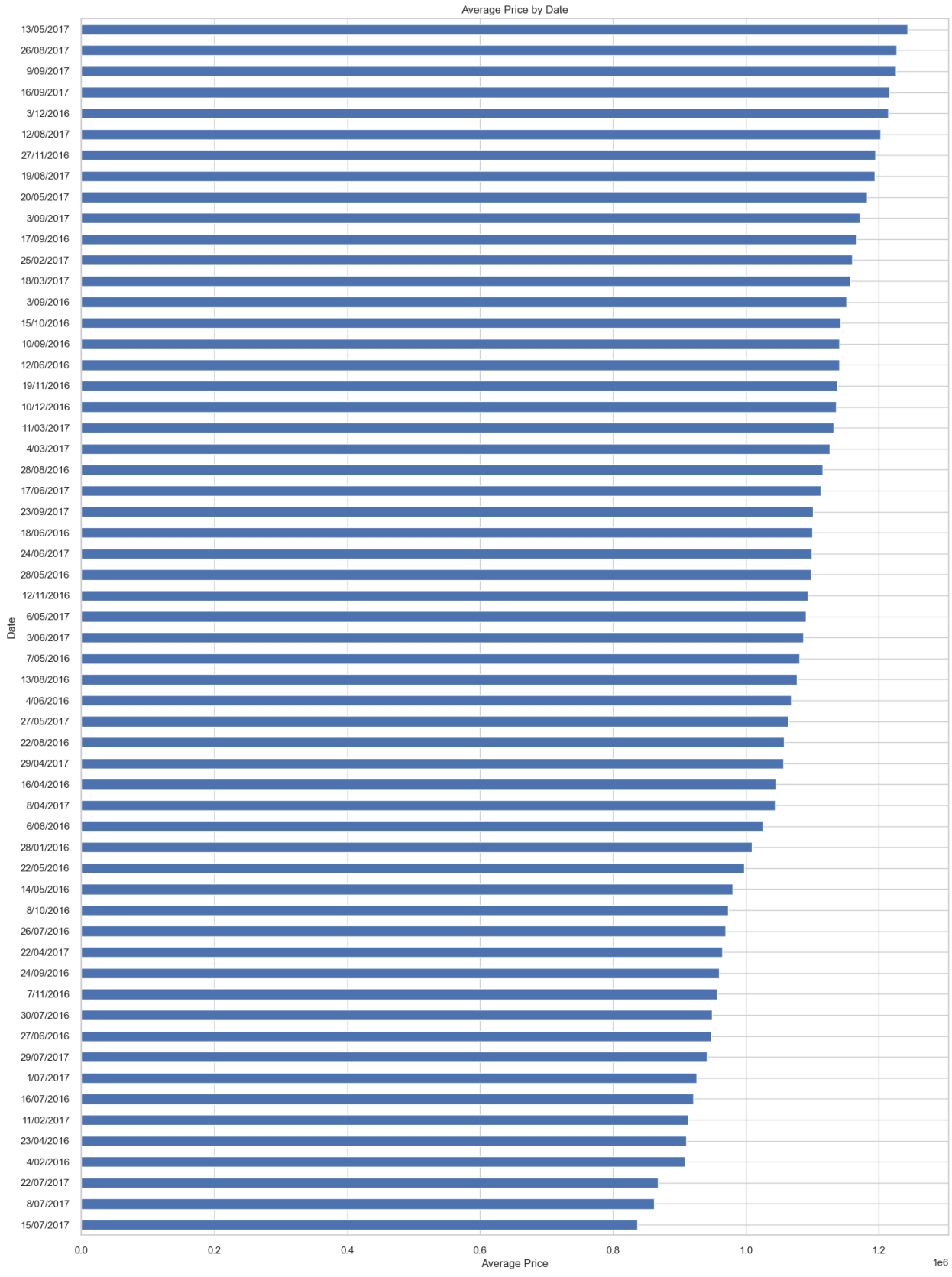

Average Price by SellerG
SellerG
Average Price
0.0
0.5
1.0
1.5
2.0
2.5
3.0
1e6

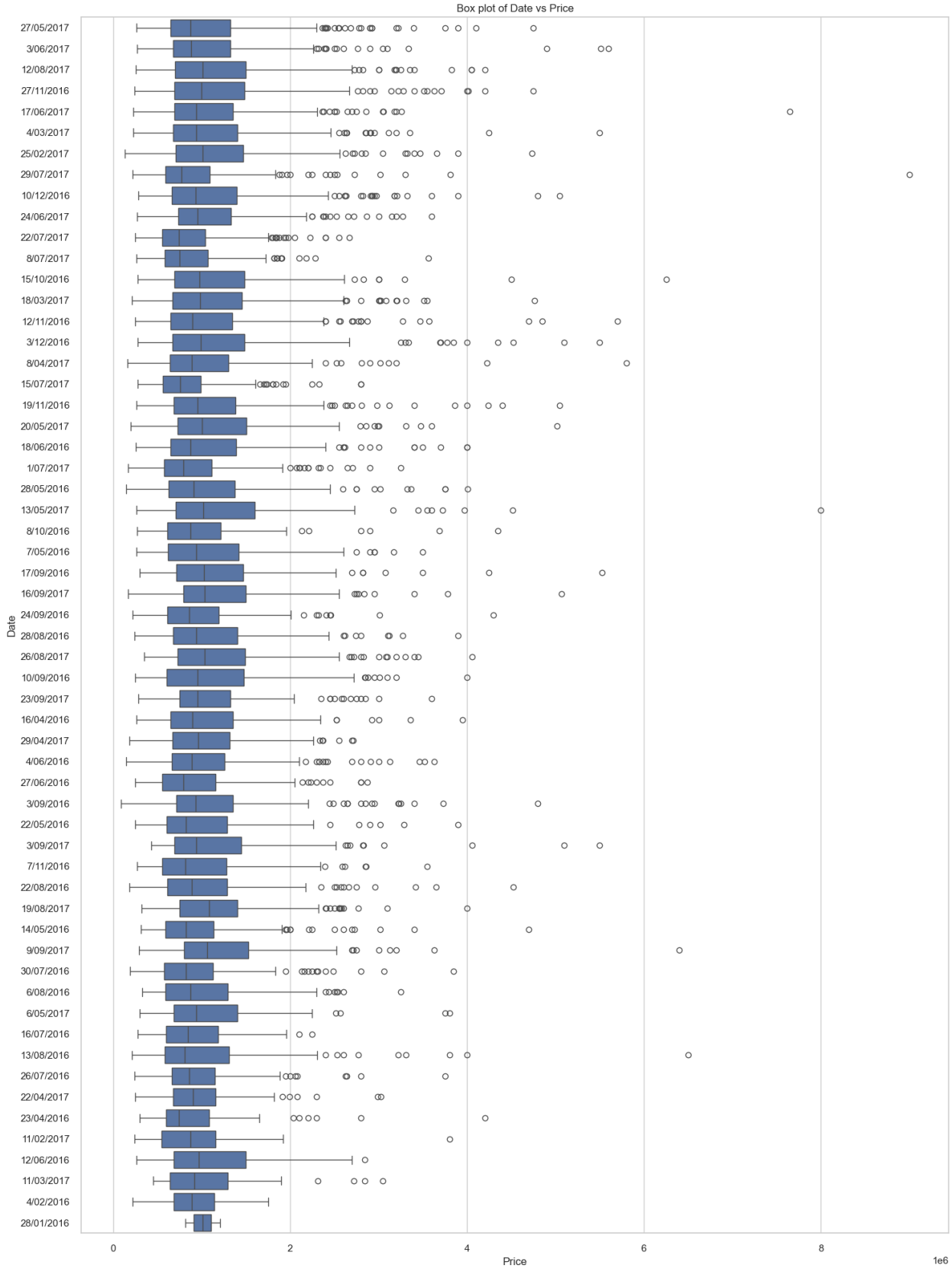

Box plot of SellerG vs Price
SellerG
Price
0
2
4
6
8
1e6

Analysis of 'Date' Variable

```
# Count plot for Date
plt.figure(figsize=(15, 20))
sb.countplot(data=housingdata, y='Date',
order=housingdata['Date'].value_counts().index)
plt.title('Count plot of Date')
plt.xlabel('Count')
plt.ylabel('Date')
plt.tight_layout()
plt.show()

# Average price bar plot for Date
# We'll compute the average price per Date and plot it
avg_price = housingdata.groupby('Date')
['Price'].mean().sort_values(ascending=True)

plt.figure(figsize=(15, 20))
avg_price.plot(kind='barh')
plt.title('Average Price by Date')
plt.xlabel('Average Price')
plt.ylabel('Date')
plt.tight_layout()
plt.show()

# Box plot of Price distribution by Date
plt.figure(figsize=(15,20))
sb.boxplot(data=housingdata, y='Date', x='Price',
order=housingdata['Date'].value_counts().index)
plt.title('Box plot of Date vs Price')
plt.xlabel('Price')
plt.ylabel('Date')
plt.tight_layout()
plt.show()
```

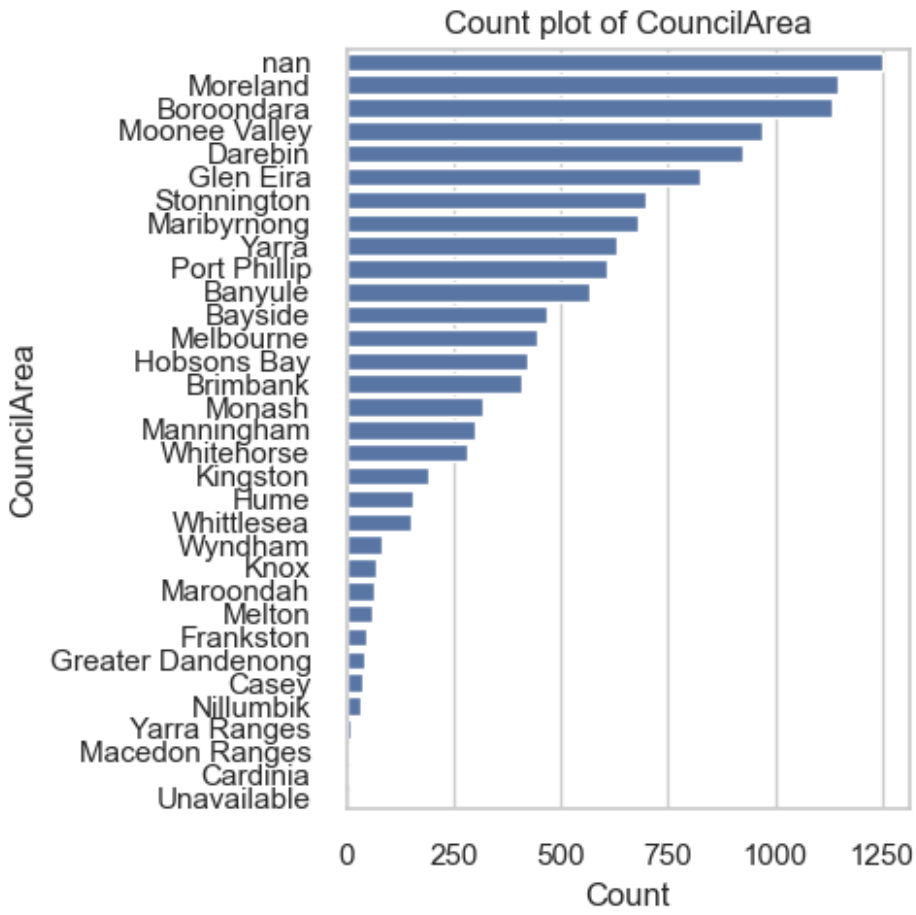

Count plot of Date
27/05/2017
3/06/2017
12/08/2017
27/11/2016
17/06/2017
4/03/2017
25/02/2017
29/07/2017
10/12/2016
24/06/2017
22/07/2017
8/07/2017
15/10/2016
18/03/2017
12/11/2016
3/12/2016
8/04/2017
15/07/2017
19/11/2016
20/05/2017
18/06/2016
1/07/2017
28/05/2016
13/05/2017
8/10/2016
7/05/2016
17/09/2016
16/09/2017
24/09/2016
28/08/2016
26/08/2017
10/09/2016
23/09/2017
16/04/2016
29/04/2017
4/06/2016
27/06/2016
3/09/2016
22/05/2016
3/09/2017
7/11/2016
22/08/2016
19/08/2017
14/05/2016
9/09/2017
30/07/2016
6/08/2016
6/05/2017
16/07/2016
13/08/2016
26/07/2016
22/04/2017
23/04/2016
11/02/2017
12/06/2016
11/03/2017
4/02/2016
28/01/2016
Date
0
100
200
300
400
Count

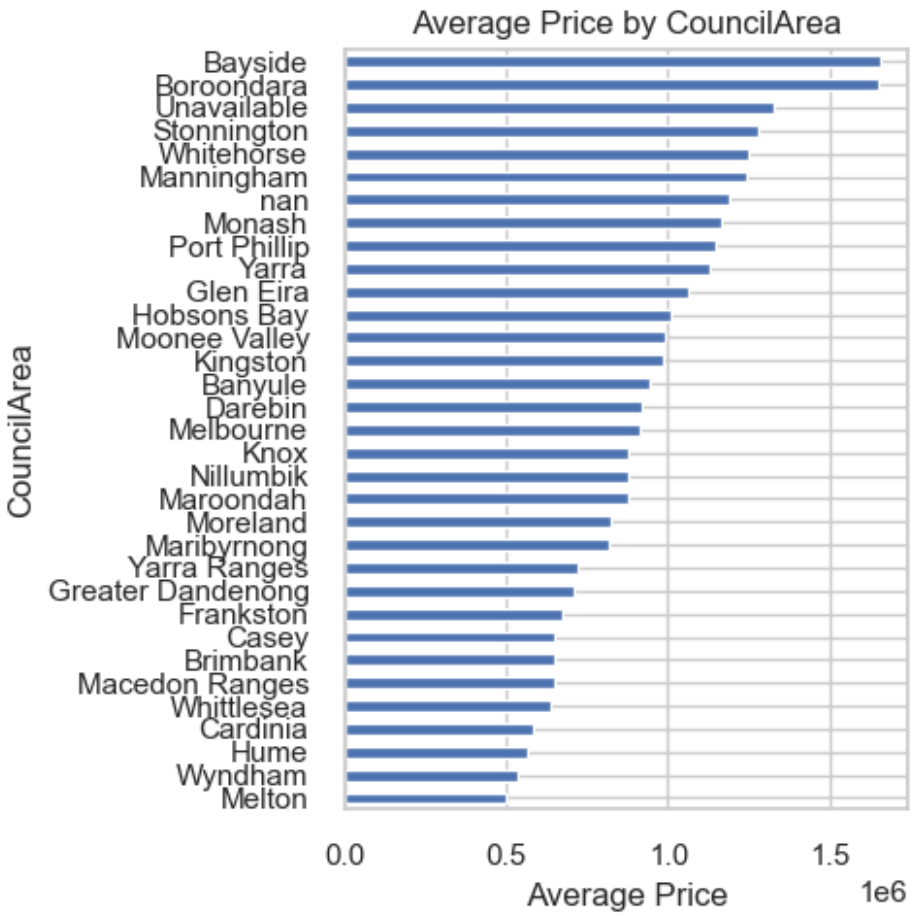

Average Price by Date
13/05/2017
26/08/2017
9/09/2017
16/09/2017
3/12/2016
12/08/2017
27/11/2016
19/08/2017
20/05/2017
3/09/2017
17/09/2016
25/02/2017
18/03/2017
3/09/2016
15/10/2016
10/09/2016
12/06/2016
19/11/2016
10/12/2016
11/03/2017
4/03/2017
28/08/2016
17/06/2017
23/09/2017
18/06/2016
24/06/2017
28/05/2016
12/11/2016
6/05/2017
3/06/2017
7/05/2016
13/08/2016
4/06/2016
27/05/2017
22/08/2016
29/04/2017
16/04/2016
8/04/2017
6/08/2016
28/01/2016
22/05/2016
14/05/2016
8/10/2016
26/07/2016
22/04/2017
24/09/2016
7/11/2016
30/07/2016
27/06/2016
29/07/2017
1/07/2017
16/07/2016
11/02/2017
23/04/2016
4/02/2016
22/07/2017
8/07/2017
15/07/2017
Date
0.0
0.2
0.4
0.6
0.8
1.0
1.2
1e6
Average Price

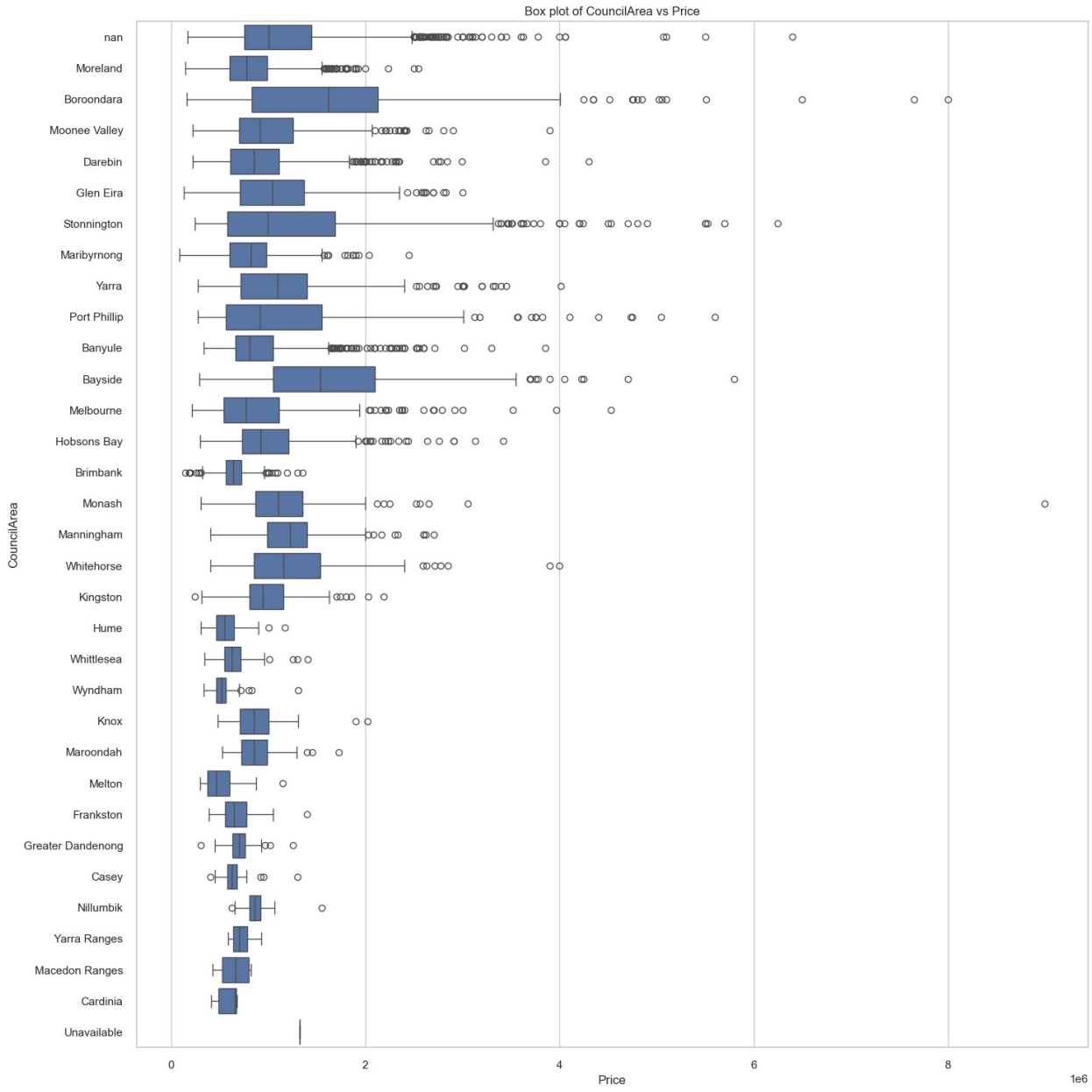

Box plot of Date vs Price
27/05/2017
3/06/2017
12/08/2017
27/11/2016
17/06/2017
4/03/2017
25/02/2017
29/07/2017
10/12/2016
24/06/2017
22/07/2017
8/07/2017
15/10/2016
18/03/2017
12/11/2016
3/12/2016
8/04/2017
15/07/2017
19/11/2016
20/05/2017
18/06/2016
1/07/2017
28/05/2016
13/05/2017
8/10/2016
7/05/2016
17/09/2016
16/09/2017
24/09/2016
28/08/2016
26/08/2017
10/09/2016
23/09/2017
16/04/2016
29/04/2017
4/06/2016
27/06/2016
3/09/2016
22/05/2016
3/09/2017
7/11/2016
22/08/2016
19/08/2017
14/05/2016
9/09/2017
30/07/2016
6/08/2016
6/05/2017
16/07/2016
13/08/2016
26/07/2016
22/04/2017
23/04/2016
11/02/2017
12/06/2016
11/03/2017
4/02/2016
28/01/2016
Date
0
2
4
6
8
Price
1e6

Analysis of 'CouncilArea' Variable

```
# Count plot for CouncilArea
plt.figure(figsize=(5,5))
sb.countplot(data=housingdata, y='CouncilArea',
order=housingdata['CouncilArea'].value_counts().index)
plt.title('Count plot of CouncilArea')
plt.xlabel('Count')
plt.ylabel('CouncilArea')
plt.tight_layout()
plt.show()

# Average price bar plot for CouncilArea
# We'll compute the average price per CouncilArea and plot it
avg_price = housingdata.groupby('CouncilArea')
['Price'].mean().sort_values(ascending=True)

plt.figure(figsize=(5,5))
avg_price.plot(kind='barh')
plt.title('Average Price by CouncilArea')
plt.xlabel('Average Price')
plt.ylabel('CouncilArea')
plt.tight_layout()
plt.show()

# Box plot of Price distribution by CouncilArea
plt.figure(figsize=(15,15))
sb.boxplot(data=housingdata, y='CouncilArea', x='Price',
order=housingdata['CouncilArea'].value_counts().index)
plt.title('Box plot of CouncilArea vs Price')
plt.xlabel('Price')
plt.ylabel('CouncilArea')
plt.tight_layout()
plt.show()
```


Count plot of CouncilArea
CouncilArea
nan
Moreland
Boroondara
Moonee Valley
Darebin
Glen Eira
Stonnington
Maribyrnong
Yarra
Port Phillip
Banyule
Bayside
Melbourne
Hobsons Bay
Brimbank
Monash
Manningham
Whitehorse
Kingston
Hume
Whittlesea
Wyndham
Knox
Maroondah
Melton
Frankston
Greater Dandenong
Casey
Nillumbik
Yarra Ranges
Macedon Ranges
Cardinia
Unavailable
0
250
500
750
1000
1250
Count

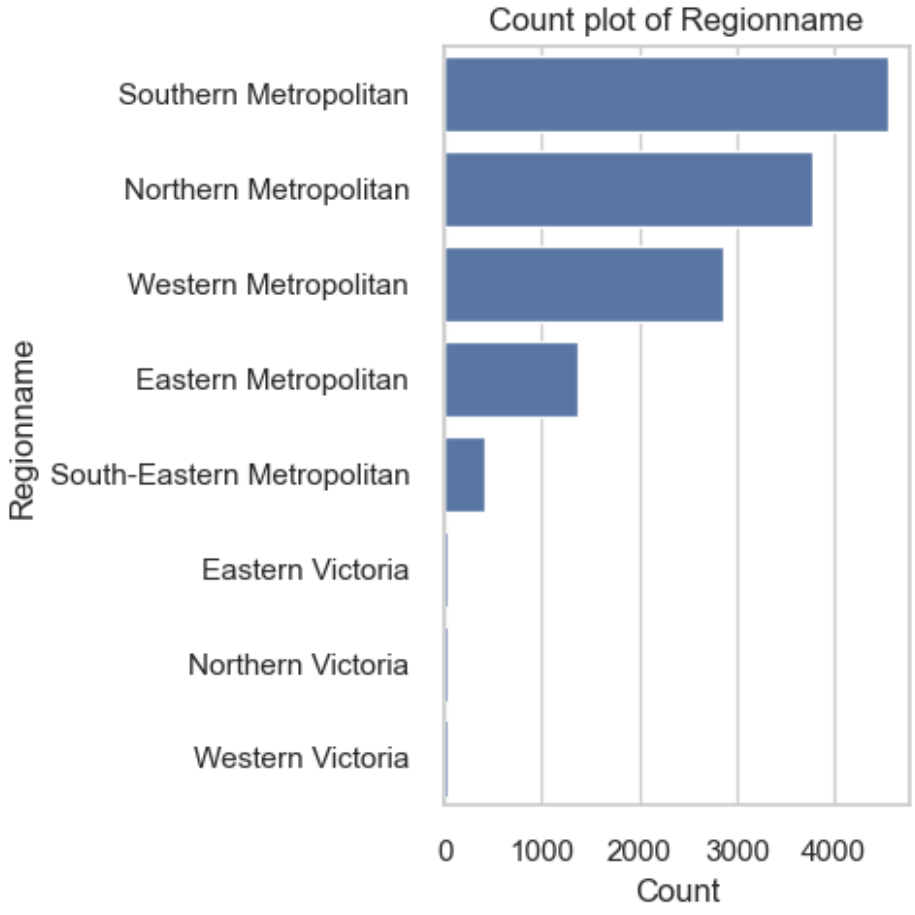

Average Price by CouncilArea
CouncilArea
Bayside
Boroondara
Unavailable
Stonnington
Whitehorse
Manningham
nan
Monash
Port Phillip
Yarra
Glen Eira
Hobsons Bay
Moonee Valley
Kingston
Banyule
Darebin
Melbourne
Knox
Nillumbik
Maroondah
Moreland
Maribyrnong
Yarra Ranges
Greater Dandenong
Frankston
Casey
Brimbank
Macedon Ranges
Whittlesea
Cardinia
Hume
Wyndham
Melton
0.0
0.5
1.0
1.5
Average Price
1e6



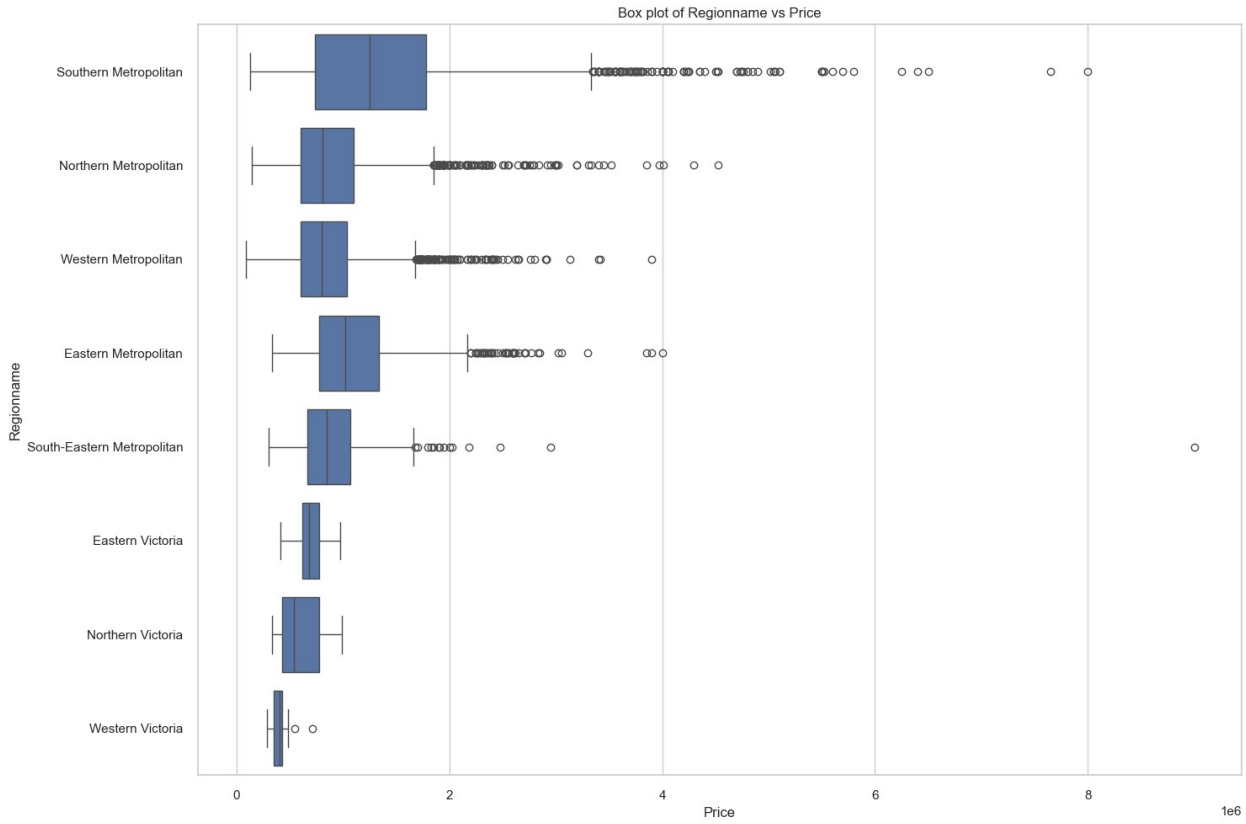
Analysis of 'Regionname' Variable

```
# Count plot for Regionname
plt.figure(figsize=(5,5))
sb.countplot(data=housingdata, y='Regionname',
order=housingdata['Regionname'].value_counts().index)
plt.title('Count plot of Regionname')
plt.xlabel('Count')
plt.ylabel('Regionname')
plt.tight_layout()
plt.show()

# Average price bar plot for Regionname
```

```
# We'll compute the average price per Regionname and plot it
avg_price = housingdata.groupby('Regionname')
['Price'].mean().sort_values(ascending=True)

plt.figure(figsize=(5,5))
avg_price.plot(kind='barh')
plt.title('Average Price by Regionname')
plt.xlabel('Average Price')
plt.ylabel('Regionname')
plt.tight_layout()
plt.show()

# Box plot of Price distribution by Regionname
plt.figure(figsize=(15,10))
sb.boxplot(data=housingdata, y='Regionname', x='Price',
order=housingdata['Regionname'].value_counts().index)
plt.title('Box plot of Regionname vs Price')
plt.xlabel('Price')
plt.ylabel('Regionname')
plt.tight_layout()
plt.show()
```




Average Price by Regionname
Southern Metropolitan
Eastern Metropolitan
South-Eastern Metropolitan
Northern Metropolitan
Western Metropolitan
Eastern Victoria
Northern Victoria
Western Victoria
Regionname
0.0
0.5
1.0
Average Price
1e6



(2) Preparation of Response & Predictor Variables

1. Dimensionality/Cardinality Reduction for Categorical Variables
2. Feature Engineering

1. Dimensionality/Cardinality Reduction for Categorical Variables

Some of the categorical variables in this dataset contain a large number of unique categories (high cardinality), contributing to a lot of unnecessary noise in our analysis and reduced interpretability.

Hence we will perform dimensionality (cardinality) reduction on selected categorical features (Suburb, SellerG, and Address) by grouping less common categories into an 'Other' category. The threshold we will choose for this is a 1% (Will probs change) proportion; categories within a variable that make up less than 1% of the number of rows in the dataset will be all classified into an 'Other' category.

Define a function for grouping the less common categories:

```
def group_rare_categories(df, column, threshold=0.01):
    category_counts = df[column].value_counts(normalize=True)

    # Identify categories with frequency less than the threshold
    rare_categories = category_counts[category_counts <
threshold].index
```

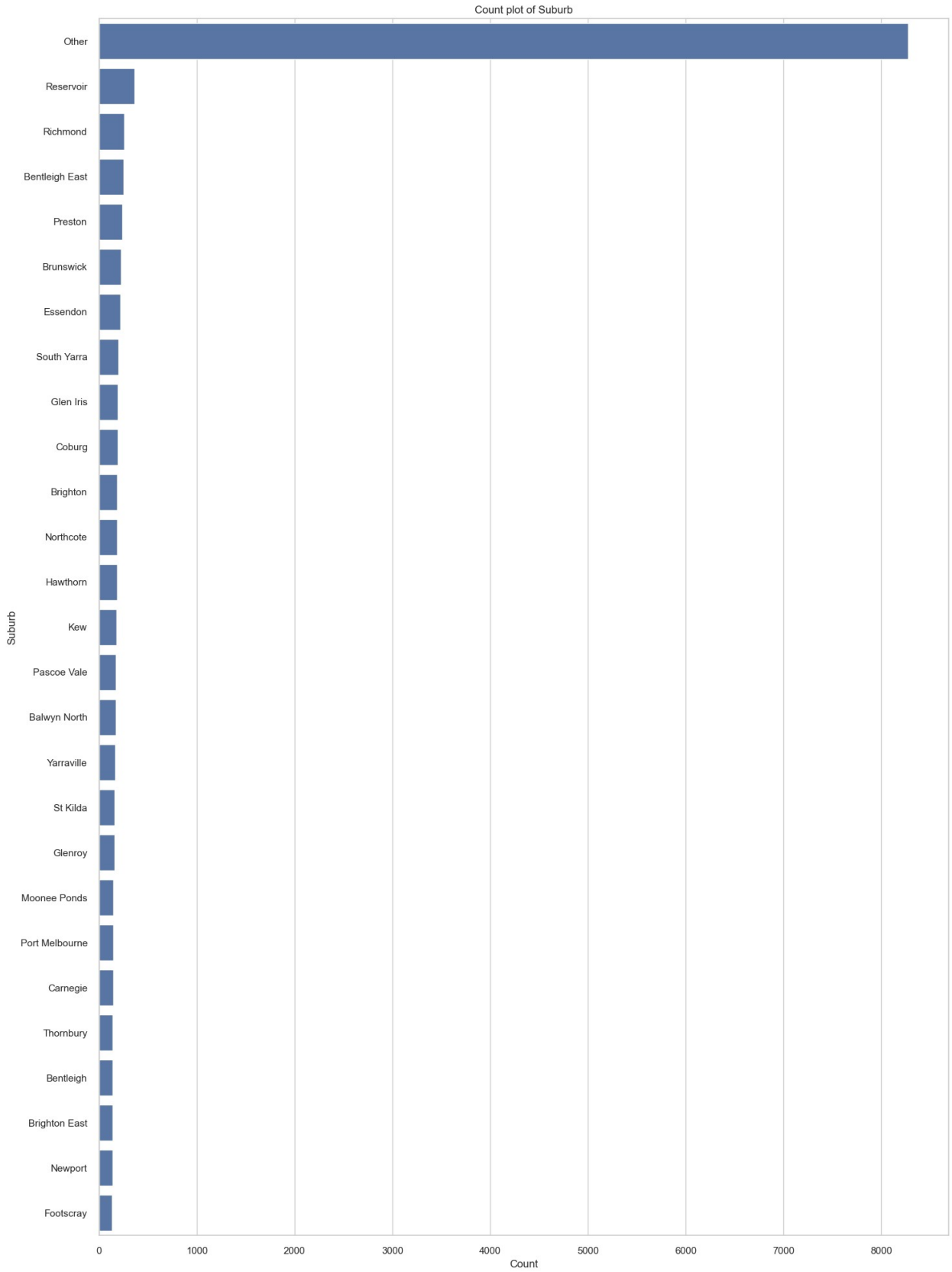
```
    # Replace rare categories with "Other"
    df[column] = df[column].apply(lambda x: 'Other' if x in
rare_categories else x)

    return df
```

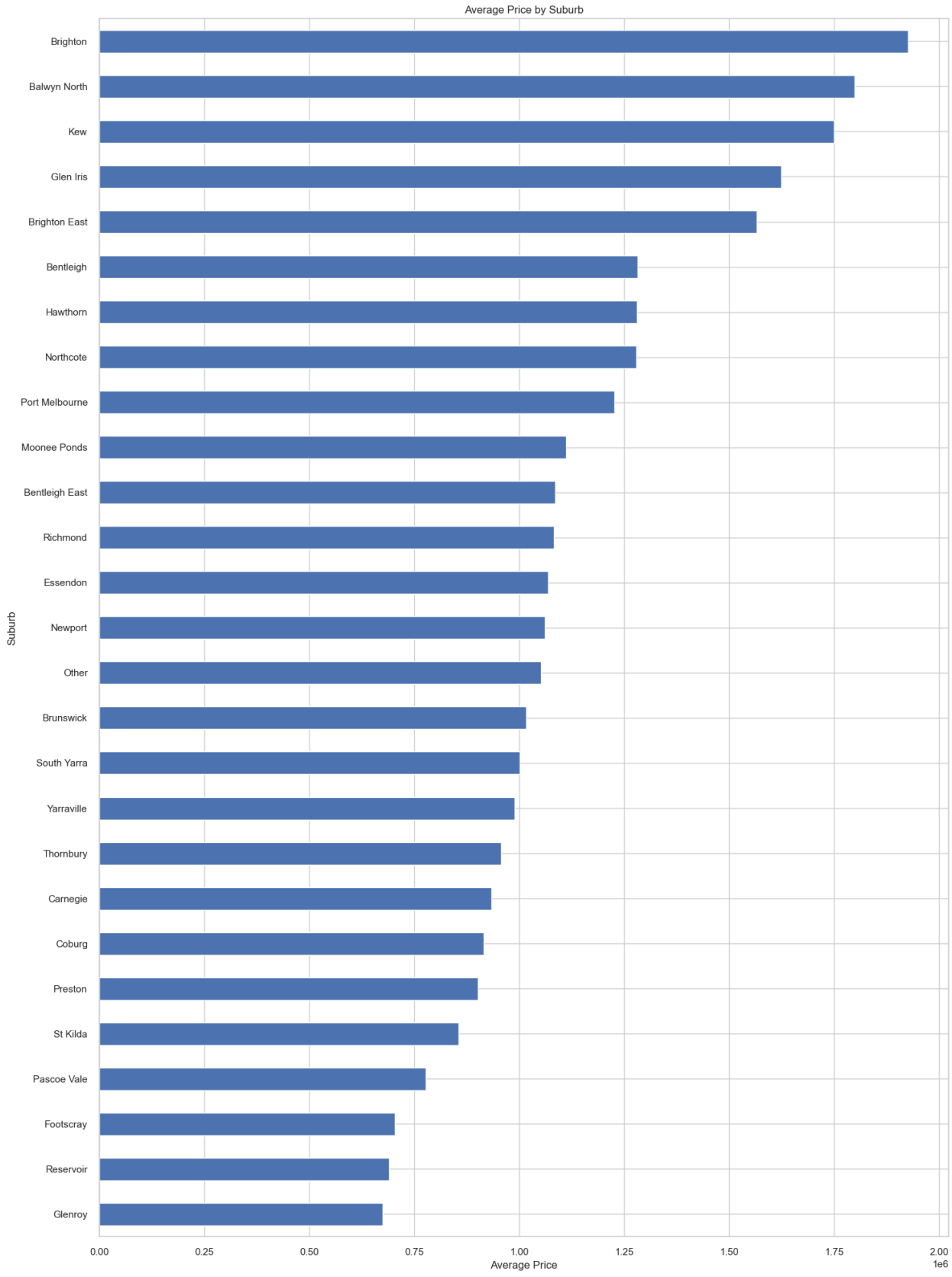
Dimensionality Reduction of Suburb

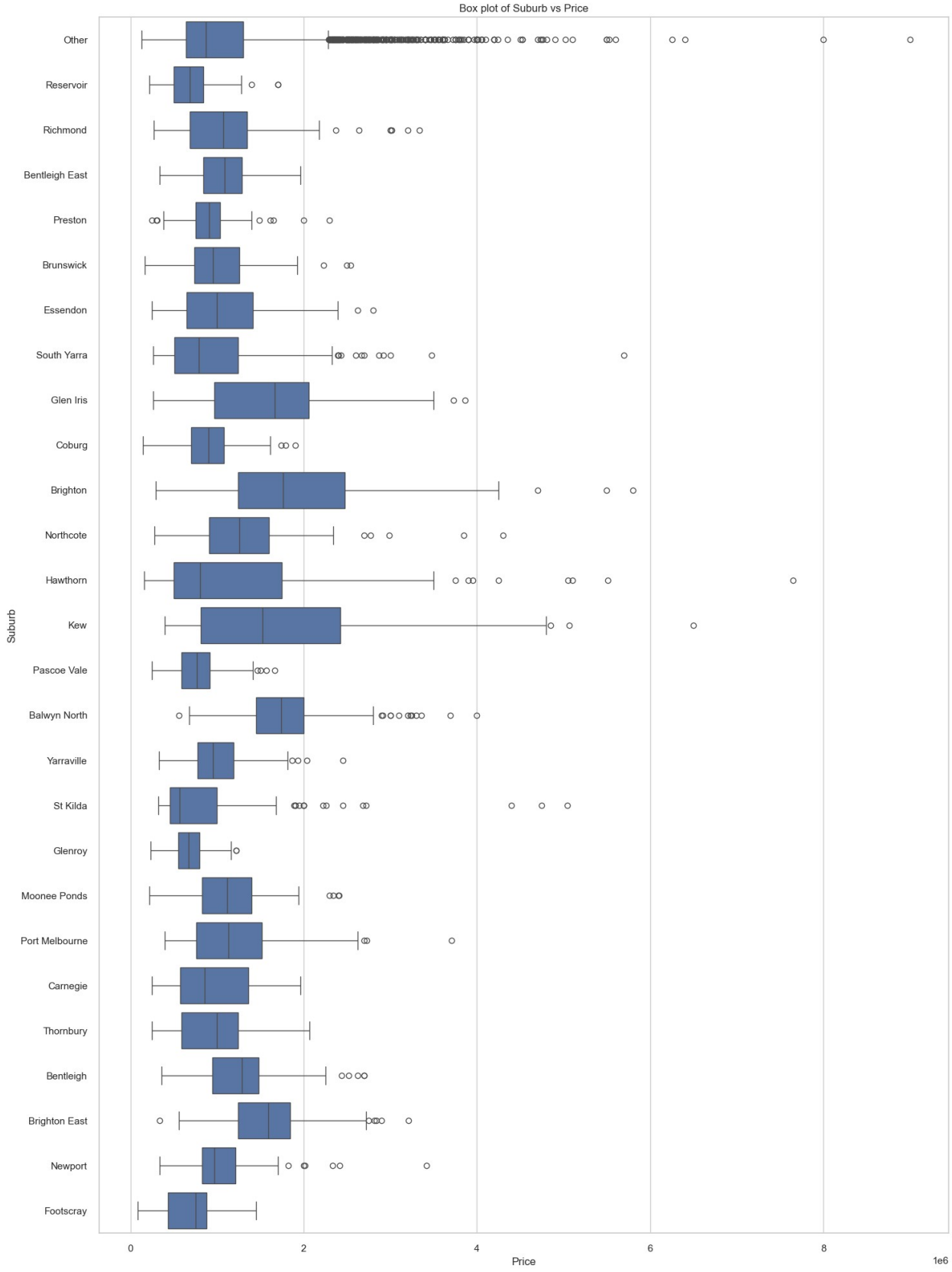
```
housingdata = group_rare_categories(housingdata, 'Suburb',
threshold=0.01)
```

Let's check what Suburb looks like after dimensionality reduction.

```
# Count plot for Suburb
plt.figure(figsize=(15,20))
sb.countplot(data=housingdata, y='Suburb',
order=housingdata['Suburb'].value_counts().index)
plt.title('Count plot of Suburb')
plt.xlabel('Count')
plt.ylabel('Suburb')
plt.tight_layout()
plt.show()

# Average price bar plot for Suburb
# We'll compute the average price per Suburb and plot it
avg_price = housingdata.groupby('Suburb')
['Price'].mean().sort_values(ascending=True)

plt.figure(figsize=(15,20))
avg_price.plot(kind='barh')
plt.title('Average Price by Suburb')
plt.xlabel('Average Price')
plt.ylabel('Suburb')
plt.tight_layout()
plt.show()

# Box plot of Price distribution by Suburb
plt.figure(figsize=(15,20))
sb.boxplot(data=housingdata, y='Suburb', x='Price',
order=housingdata['Suburb'].value_counts().index)
plt.title('Box plot of Suburb vs Price')
plt.xlabel('Price')
plt.ylabel('Suburb')
plt.tight_layout()
plt.show()
```


Count plot of Suburb
Other
Reservoir
Richmond
Bentleigh East
Preston
Brunswick
Essendon
South Yarra
Glen Iris
Coburg
Brighton
Northcote
Hawthorn
Kew
Pascoe Vale
Balwyn North
Yarraville
St Kilda
Glenroy
Moonee Ponds
Port Melbourne
Carnegie
Thornbury
Bentleigh
Brighton East
Newport
Footscray
Suburb
0
1000
2000
3000
4000
5000
6000
7000
8000
Count


Average Price by Suburb
Brighton
Balwyn North
Kew
Glen Iris
Brighton East
Bentleigh
Hawthorn
Northcote
Port Melbourne
Moonee Ponds
Bentleigh East
Richmond
Essendon
Newport
Other
Brunswick
South Yarra
Yarraville
Thornbury
Carnegie
Coburg
Preston
St Kilda
Pascoe Vale
Footscray
Reservoir
Glenroy
Suburb
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00
1e6
Average Price

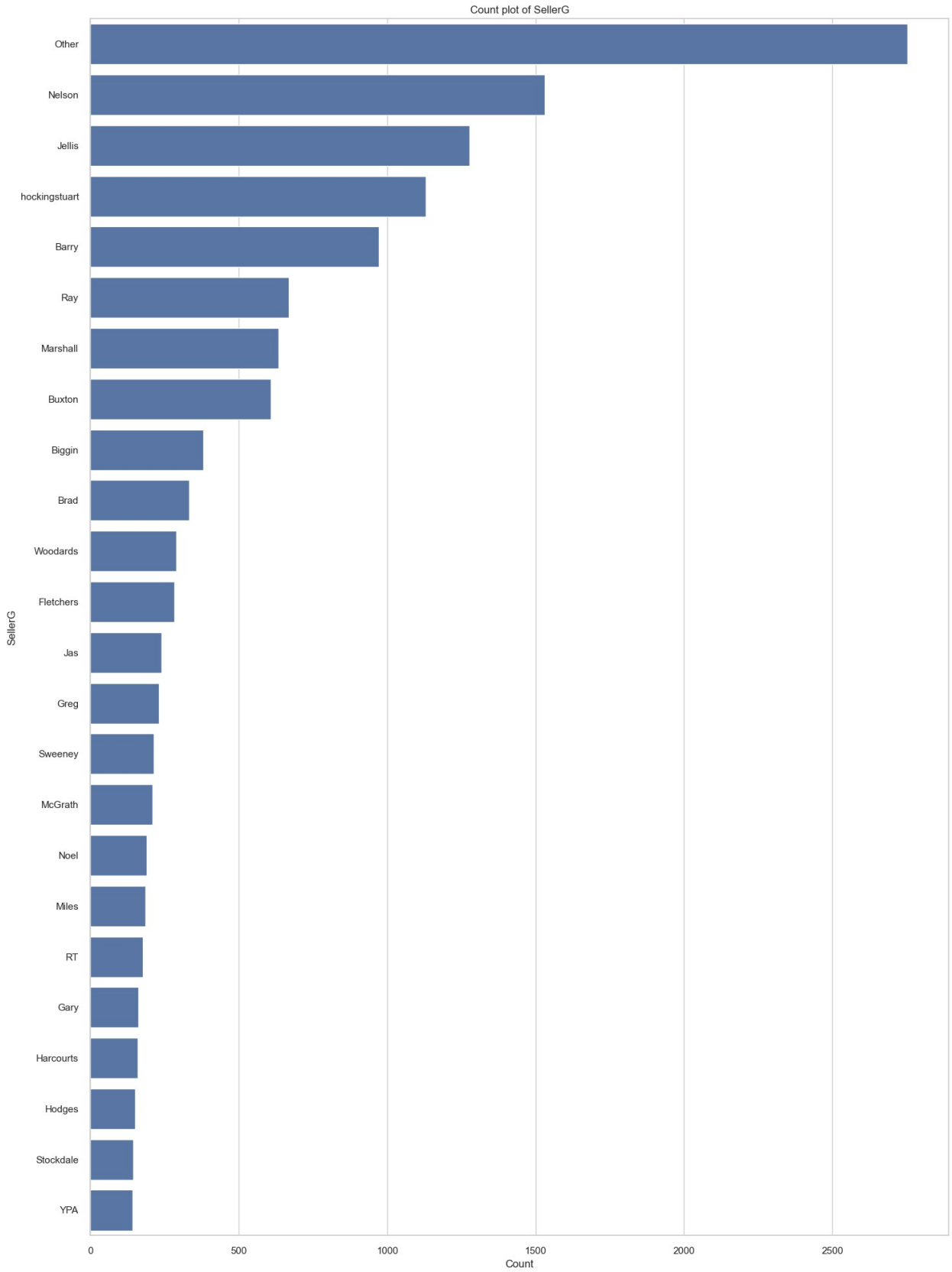

Box plot of Suburb vs Price
Other
Reservoir
Richmond
Bentleigh East
Preston
Brunswick
Essendon
South Yarra
Glen Iris
Coburg
Brighton
Northcote
Hawthorn
Kew
Pascoe Vale
Balwyn North
Yarraville
St Kilda
Glenroy
Moonee Ponds
Port Melbourne
Carnegie
Thornbury
Bentleigh
Brighton East
Newport
Footscray
Suburb
0
2
4
6
8
Price
1e6

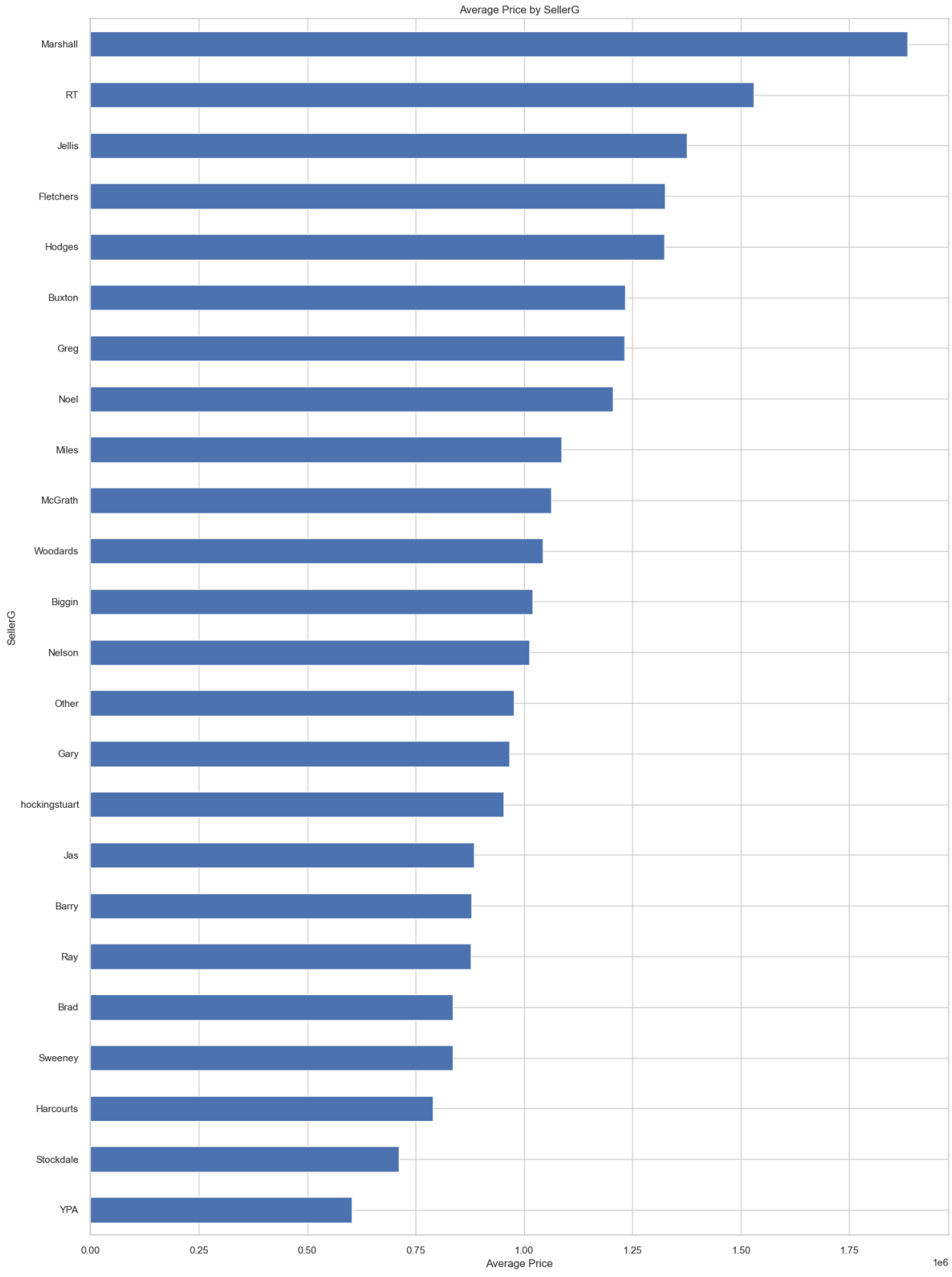
Dimensionality Reduction of SellerG

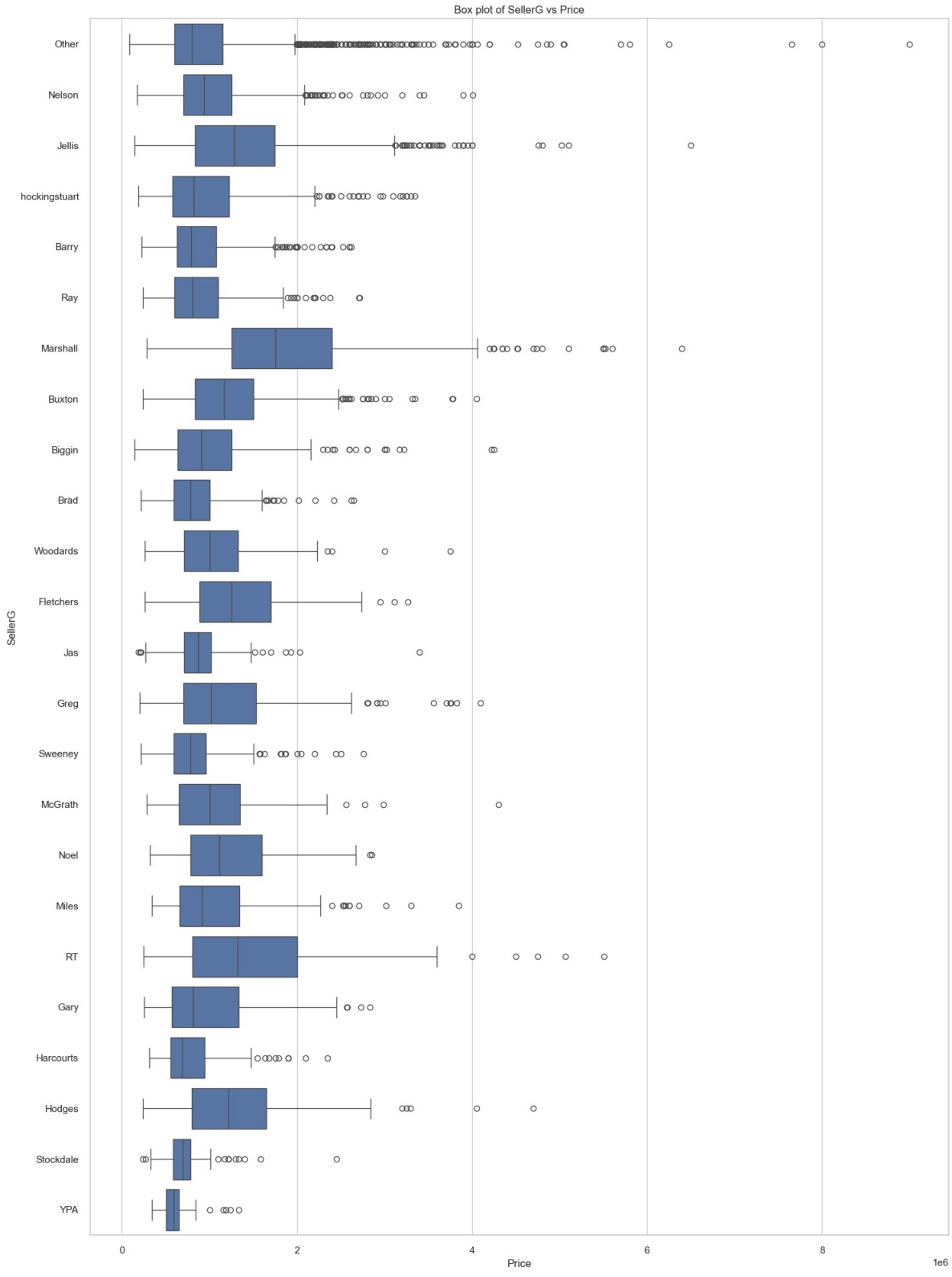
```
housingdata = group_rare_categories(housingdata, 'SellerG',
threshold=0.01)
```

Let's check what SellerG looks like after dimensionality reduction.

```
# Count plot for SellerG
plt.figure(figsize=(15,20))
sb.countplot(data=housingdata, y='SellerG',
order=housingdata['SellerG'].value_counts().index)
plt.title('Count plot of SellerG')
plt.xlabel('Count')
plt.ylabel('SellerG')
plt.tight_layout()
plt.show()

# Average price bar plot for SellerG
# We'll compute the average price per SellerG and plot it
avg_price = housingdata.groupby('SellerG')
['Price'].mean().sort_values(ascending=True)

plt.figure(figsize=(15,20))
avg_price.plot(kind='barh')
plt.title('Average Price by SellerG')
plt.xlabel('Average Price')
plt.ylabel('SellerG')
plt.tight_layout()
plt.show()

# Box plot of Price distribution by SellerG
plt.figure(figsize=(15,20))
sb.boxplot(data=housingdata, y='SellerG', x='Price',
order=housingdata['SellerG'].value_counts().index)
plt.title('Box plot of SellerG vs Price')
plt.xlabel('Price')
plt.ylabel('SellerG')
plt.tight_layout()
plt.show()
```


Count plot of SellerG
Other
Nelson
Jellis
hockingstuart
Barry
Ray
Marshall
Buxton
Biggin
Brad
Woodards
Fletchers
Jas
Greg
Sweeney
McGrath
Noel
Miles
RT
Gary
Harcourts
Hodges
Stockdale
YPA
SellerG
0
500
1000
1500
2000
2500
Count


Average Price by SellerG
Marshall
RT
Jellis
Fletchers
Hodges
Buxton
Greg
Noel
Miles
McGrath
Woodards
Biggin
Nelson
Other
Gary
hockingstuart
Jas
Barry
Ray
Brad
Sweeney
Harcourts
Stockdale
YPA
SellerG
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
Average Price
1e6


Box plot of SellerG vs Price
Other
Nelson
Jellis
hockingstuart
Barry
Ray
Marshall
Buxton
Biggin
Brad
Woodards
Fletchers
Jas
Greg
Sweeney
McGrath
Noel
Miles
RT
Gary
Harcourts
Hodges
Stockdale
YPA
SellerG
0
2
4
6
8
Price
1e6

Dimensionality Reduction of Address - Feature Elimination

We choose to drop 'Address' entirely as it is highly granular and provides little to no relevance in predicting house prices. Removing it contributes to reducing the overall dimensionality of the dataset.

```
housingdata = housingdata.drop(columns=['Address'])
```

2. Feature Engineering

Feature Engineering of 'Date' Variable

Since we are planning to use 'Date' as a time-based feature in trend analysis, we convert it from a string to datetime format and extract additional time-based features such as 'Year', 'Month', and 'YearMonth'. These derived variables will allow us to identify patterns across different regions over time.

```
# Convert 'Date' to datetime format
housingdata['Date'] = pd.to_datetime(housingdata['Date'],
dayfirst=True)

# Create new time-based features
housingdata['Year'] = housingdata['Date'].dt.year
housingdata['Month'] = housingdata['Date'].dt.month
housingdata['YearMonth'] = housingdata['Date'].dt.to_period('M')

print(housingdata[['Year', 'Month', 'YearMonth']].head())
```

```
   Year  Month YearMonth
0  2016     12   2016-12
1  2016      2   2016-02
2  2017      3   2017-03
3  2017      3   2017-03
4  2016      6   2016-06
```

(3) Core Analysis

1. Regression Analysis to predict Feature Importance
2. K-Means Clustering to observe Price Trends across different regions

1. Regression Analysis

```
# Separate target
y = housingdata["Price"]
X = housingdata.drop(columns=["Price"])

# Identify categorical features
categorical_cols = X.select_dtypes(include=["object",
"category"]).columns
numerical_cols = X.select_dtypes(include=["int64", "float64"]).columns
```

```
# Identify datetime columns
datetime_cols = X.select_dtypes(include=["datetime",
"datetime64[ns]"]).columns

# Convert datetime features to numeric (example: just using year for
simplicity)
for col in datetime_cols:
    X[col] = pd.to_datetime(X[col], errors='coerce')  # ensure correct
format
    X[col + "_year"] = X[col].dt.year
    X[col + "_month"] = X[col].dt.month
    X = X.drop(columns=[col])  # drop original datetime
# Convert Period columns to string, then to datetime, then to integer
year/month
for col in X.columns:
    if X[col].dtype.name.startswith("period"):
        X[col] = X[col].astype(str)
        try:
            X[col] = pd.to_datetime(X[col])
            X[col] = X[col].dt.year  # or .dt.month if more relevant
        except Exception as e:
            print(f"Could not convert column {col}: {e}")

# Now re-encode categorical features
encoder = OrdinalEncoder()
X[categorical_cols] =
encoder.fit_transform(X[categorical_cols].astype(str))

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

# LINEAR REGRESSION
lr = LinearRegression()
lr.fit(X_train, y_train)
linear_importance = pd.Series(lr.coef_,
index=X.columns).sort_values(ascending=False)
print("Linear Regression Feature Importance:")
print(linear_importance)

#  POLYNOMIAL (QUADRATIC) REGRESSION
poly = PolynomialFeatures(degree=2, include_bias=False)
X_poly = poly.fit_transform(X)
X_train_poly, X_test_poly, y_train_poly, y_test_poly =
train_test_split(X_poly, y, test_size=0.2, random_state=42)

lr_poly = LinearRegression()
lr_poly.fit(X_train_poly, y_train_poly)
poly_feature_names = poly.get_feature_names_out(X.columns)
```

```python
poly_importance = pd.Series(lr_poly.coef_,
index=poly_feature_names).sort_values(ascending=False)
print("\nPolynomial Regression Feature Importance (Top 20):")
print(poly_importance.head(20))

#  RANDOM FOREST REGRESSOR
rf = RandomForestRegressor(n_estimators=100, random_state=42)
rf.fit(X_train, y_train)
rf_importance = pd.Series(rf.feature_importances_,
index=X.columns).sort_values(ascending=False)
print("\nRandom Forest Feature Importance:")
print(rf_importance)
```

```
Linear Regression Feature Importance:
Longtitude       1.130116e+06
Bathroom         2.274854e+05
Rooms            1.324517e+05
Car              5.365276e+04
Date_year        3.712733e+04
Year             3.712733e+04
YearMonth        3.712733e+04
Regionname       3.246026e+04
Bedroom2         2.773026e+04
Month            7.063070e+03
Date_month       7.063070e+03
Postcode         8.630514e+02
BuildingArea     6.202253e+02
Landsize         2.999353e+00
Propertycount   -1.071838e+00
SellerG         -2.210479e+03
CouncilArea     -3.228786e+03
YearBuilt       -3.505618e+03
Suburb          -6.166301e+03
Method          -7.299810e+03
Distance        -4.678672e+04
Type            -1.653281e+05
Lattitude       -8.739891e+05
dtype: float64

Polynomial Regression Feature Importance (Top 20):
YearBuilt                4.524386e+06
Bedroom2 Lattitude       5.434729e+05
Lattitude^2              5.078956e+05
Lattitude Longtitude     3.451043e+05
Lattitude Regionname     2.599722e+05
Bedroom2 Longtitude      1.184148e+05
Distance                 1.155625e+05
Rooms Bathroom           1.140714e+05
Longtitude YearMonth     8.788114e+04
Longtitude Year          8.788114e+04
```

```
Longtitude Date_year     8.788114e+04
Suburb Longtitude        8.195856e+04
Distance Lattitude       6.266741e+04
Type                     5.219580e+04
Car Longtitude           4.626355e+04
Longtitude Month         3.793352e+04
Longtitude Date_month    3.793352e+04
Type Bathroom            3.709590e+04
Suburb                   3.278605e+04
Bathroom Car             2.408232e+04
dtype: float64

Random Forest Feature Importance:
BuildingArea     0.334750
Distance         0.137943
YearBuilt        0.135078
Postcode         0.090814
Landsize         0.077687
Lattitude        0.045949
Longtitude       0.041570
Bathroom         0.029875
Propertycount    0.016122
Rooms            0.013018
Car              0.010164
Type             0.009797
CouncilArea      0.009219
SellerG          0.008947
Month            0.007579
Method           0.007564
Date_month       0.007330
Bedroom2         0.006944
Suburb           0.004480
Date_year        0.001425
YearMonth        0.001393
Year             0.001353
Regionname       0.001001
dtype: float64
```

Let's check the R^2 values.

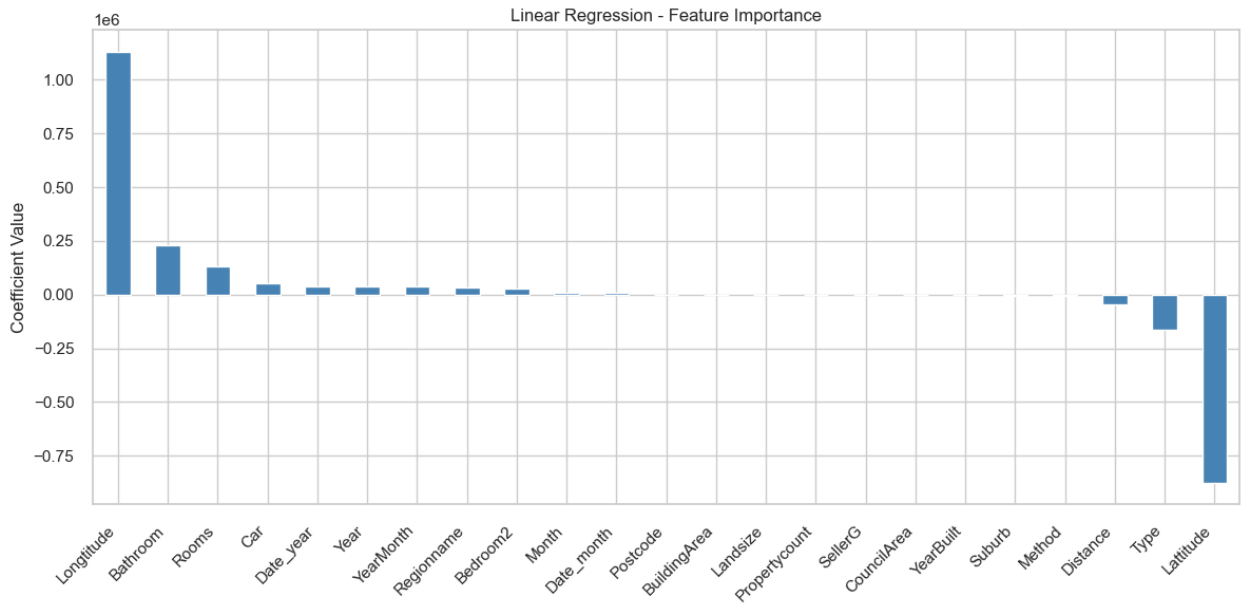
```python
# === LINEAR REGRESSION METRICS ===
y_pred_lr = lr.predict(X_test)
print("\nLinear Regression Performance:")
print(f"R^2 Score: {r2_score(y_test, y_pred_lr):.4f}")
print(f"MAE: {mean_absolute_error(y_test, y_pred_lr):,.2f}")
print(f"MSE: {mean_squared_error(y_test, y_pred_lr):,.2f}")
print(f"RMSE: {np.sqrt(mean_squared_error(y_test, y_pred_lr)):.2f}")

# === POLYNOMIAL REGRESSION METRICS ===
```

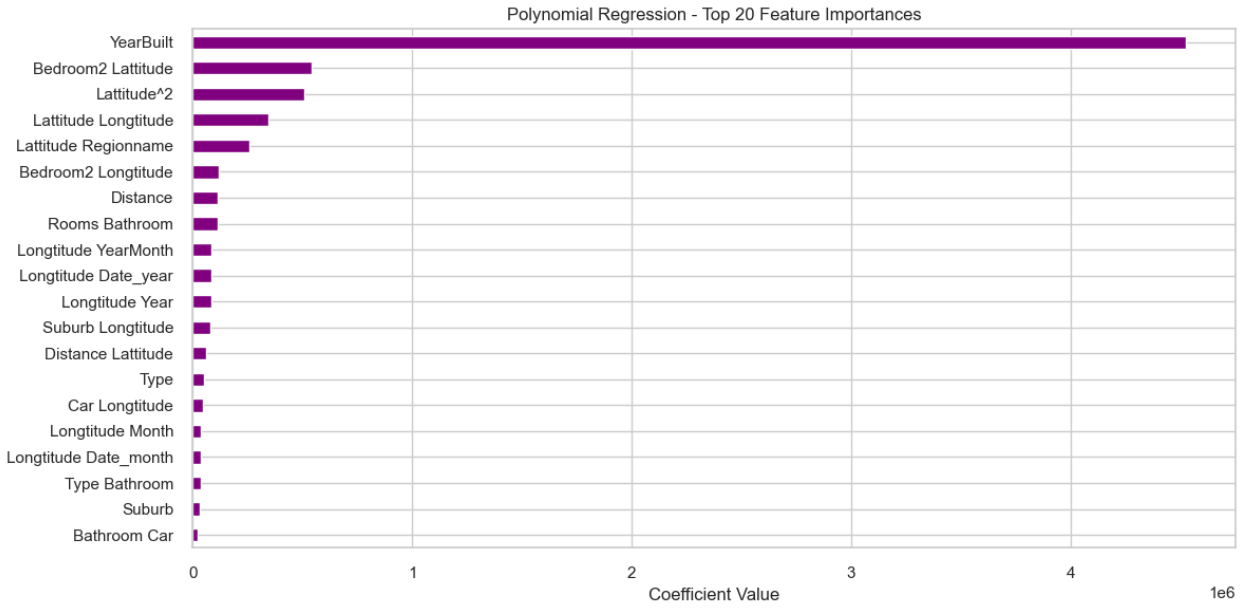
```python
y_pred_poly = lr_poly.predict(X_test_poly)
print("\nPolynomial Regression Performance:")
print(f"R^2 Score: {r2_score(y_test_poly, y_pred_poly):.4f}")
print(f"MAE: {mean_absolute_error(y_test_poly, y_pred_poly):,.2f}")
print(f"MSE: {mean_squared_error(y_test_poly, y_pred_poly):,.2f}")
print(f"RMSE: {np.sqrt(mean_squared_error(y_test_poly,
y_pred_poly)):.2f}")

# === RANDOM FOREST REGRESSION METRICS ===
y_pred_rf = rf.predict(X_test)
print("\nRandom Forest Regression Performance:")
print(f"R^2 Score: {r2_score(y_test, y_pred_rf):.4f}")
print(f"MAE: {mean_absolute_error(y_test, y_pred_rf):,.2f}")
print(f"MSE: {mean_squared_error(y_test, y_pred_rf):,.2f}")
print(f"RMSE: {np.sqrt(mean_squared_error(y_test, y_pred_rf)):.2f}")
```

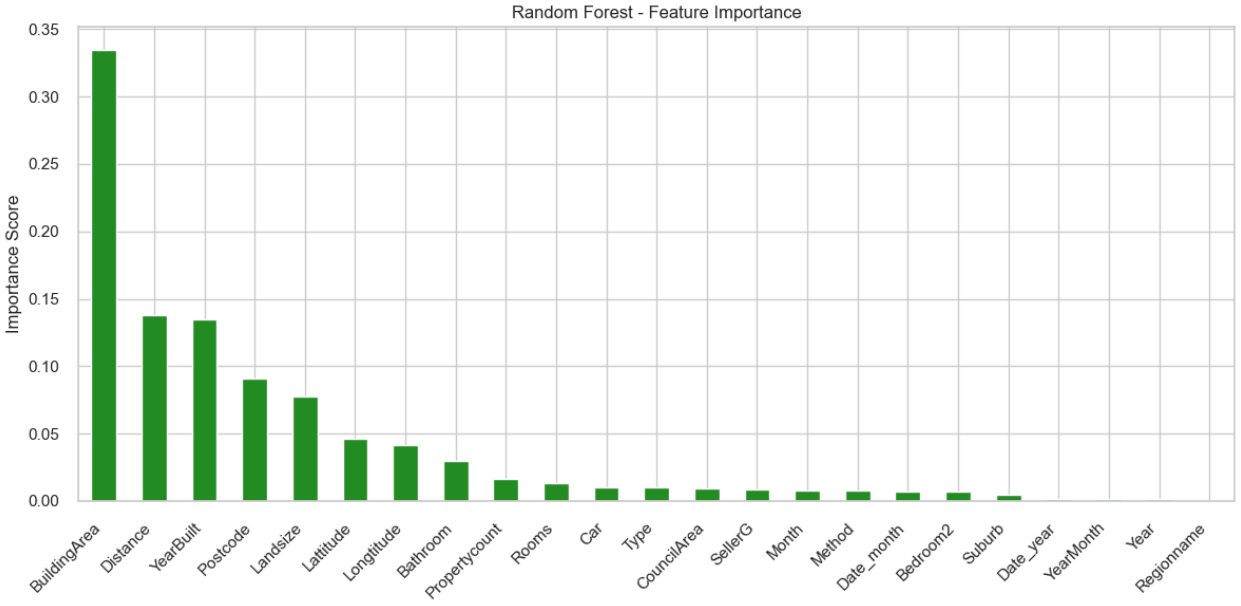
```
Linear Regression Performance:
R^2 Score: 0.6087
MAE: 265,977.57
MSE: 151,539,907,633.81
RMSE: 389281.27

Polynomial Regression Performance:
R^2 Score: 0.1976
MAE: 226,580.08
MSE: 310,747,067,755.52
RMSE: 557446.92

Random Forest Regression Performance:
R^2 Score: 0.7979
MAE: 168,930.00
MSE: 78,290,445,563.36
RMSE: 279804.30
```

```python
# Set style
sb.set(style="whitegrid")
plt.rcParams.update({'figure.figsize': (12, 6)})

#  Plot: Linear Regression Feature Importance
plt.figure()
linear_importance.sort_values(ascending=False).plot(kind='bar',
color='steelblue')
plt.title("Linear Regression - Feature Importance")
plt.ylabel("Coefficient Value")
plt.xticks(rotation=45, ha="right")
plt.tight_layout()
plt.show()

#  Plot: Polynomial Regression Feature Importance (Top 20)
```

```
plt.figure()
poly_importance.head(20).sort_values().plot(kind='barh',
color='purple')
plt.title("Polynomial Regression - Top 20 Feature Importances")
plt.xlabel("Coefficient Value")
plt.tight_layout()
plt.show()

#  Plot: Random Forest Feature Importance
plt.figure()
rf_importance.sort_values(ascending=False).plot(kind='bar',
color='forestgreen')
plt.title("Random Forest - Feature Importance")
plt.ylabel("Importance Score")
plt.xticks(rotation=45, ha="right")
plt.tight_layout()
plt.show()
```







Improvements to Baseline Model

This is a base default reference that we will be using as comparison after we improve the model by applying the following techniques:

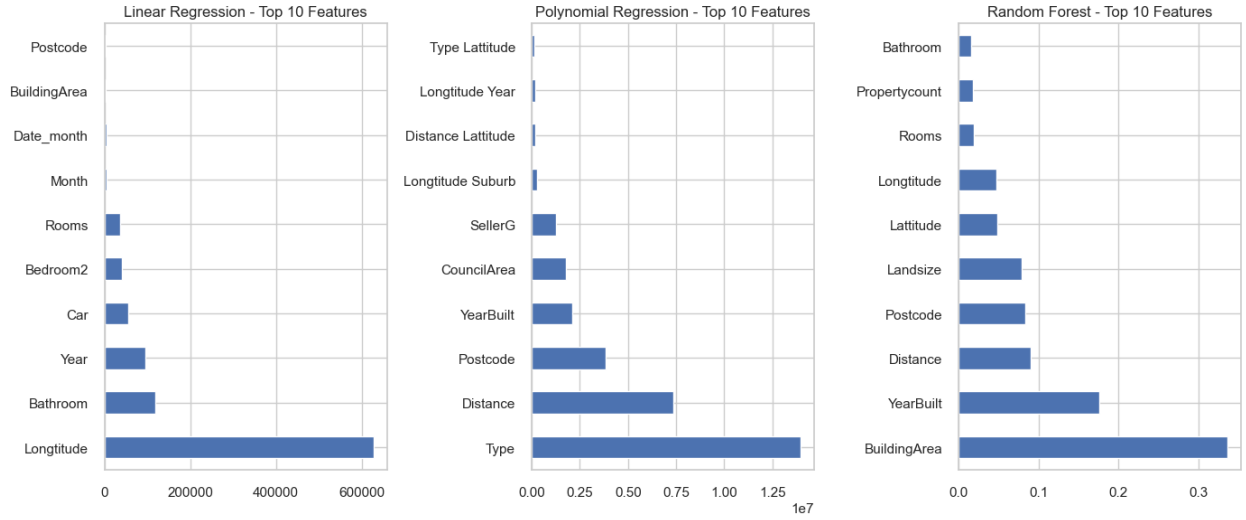
1. Dealing with Outliers
2. Feature Selection
3. Hyperparameter Tuning

```
# Separate target
y = housingdata["Price"]
X = housingdata.drop(columns=["Price"])
```

```
#  Handle datetime and period features
datetime_cols = X.select_dtypes(include=["datetime",
"datetime64[ns]"]).columns
for col in datetime_cols:
    X[col] = pd.to_datetime(X[col], errors='coerce')
    X[col + "_year"] = X[col].dt.year
    X[col + "_month"] = X[col].dt.month
    X.drop(columns=[col], inplace=True)

for col in X.columns:
    if X[col].dtype.name.startswith("period"):
        X[col] = pd.to_datetime(X[col].astype(str), errors="coerce")
        X[col] = X[col].dt.year

#  Encode categorical variables
categorical_cols = X.select_dtypes(include=["object",
"category"]).columns
encoder = OrdinalEncoder()
X[categorical_cols] =
encoder.fit_transform(X[categorical_cols].astype(str))

#  Remove outliers
numerical_cols = X.select_dtypes(include=["int64", "float64"]).columns
z_scores = np.abs((X[numerical_cols] - X[numerical_cols].mean()) /
X[numerical_cols].std())
X = X[(z_scores < 3).all(axis=1)]
y = y[X.index]  # keep y aligned with X

#  Feature Selection
# 1. Correlation-based selection
corr_with_target = X[numerical_cols].corrwith(y).abs()
top_corr_features = corr_with_target[corr_with_target >
0.1].index.tolist()

# 2. Random Forest-based selection
rf_temp = RandomForestRegressor(n_estimators=100, random_state=42)
rf_temp.fit(X, y)
rf_feature_importance = pd.Series(rf_temp.feature_importances_,
index=X.columns).sort_values(ascending=False)
top_rf_features = rf_feature_importance.head(20).index.tolist()

# 3. Combine both
selected_features = list(set(top_corr_features + top_rf_features))
X = X[selected_features]

#  Train/Test Split
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)

#  LINEAR REGRESSION
```

```
lr = LinearRegression()
lr.fit(X_train, y_train)
linear_importance = pd.Series(lr.coef_,
index=X.columns).sort_values(ascending=False)

#  POLYNOMIAL REGRESSION (degree=2)
poly = PolynomialFeatures(degree=2, include_bias=False)
X_poly = poly.fit_transform(X)
X_train_poly, X_test_poly, y_train_poly, y_test_poly =
train_test_split(X_poly, y, test_size=0.2, random_state=42)

lr_poly = LinearRegression()
lr_poly.fit(X_train_poly, y_train_poly)
poly_feature_names = poly.get_feature_names_out(X.columns)
poly_importance = pd.Series(lr_poly.coef_,
index=poly_feature_names).sort_values(ascending=False)

#  RANDOM FOREST REGRESSION with HYPERPARAMETER TUNING
param_grid = {
    'n_estimators': [100, 200],
    'max_depth': [None, 10, 20],
    'min_samples_split': [2, 5],
}

grid_search = GridSearchCV(RandomForestRegressor(random_state=42),
param_grid,
                           cv=5, scoring='r2', n_jobs=-1, verbose=1)
grid_search.fit(X_train, y_train)
best_rf = grid_search.best_estimator_
best_rf.fit(X_train, y_train)

rf_importance = pd.Series(best_rf.feature_importances_,
index=X.columns).sort_values(ascending=False)

#  PLOTTING
plt.figure(figsize=(14, 6))
plt.subplot(1, 3, 1)
linear_importance.head(10).plot(kind='barh')
plt.title("Linear Regression - Top 10 Features")

plt.subplot(1, 3, 2)
poly_importance.head(10).plot(kind='barh')
plt.title("Polynomial Regression - Top 10 Features")

plt.subplot(1, 3, 3)
rf_importance.head(10).plot(kind='barh')
plt.title("Random Forest - Top 10 Features")

plt.tight_layout()
plt.show()
```

```
Fitting 5 folds for each of 12 candidates, totalling 60 fits
```



Let's check the improved R^2 values.

```
# === IMPROVED LINEAR REGRESSION METRICS ===
y_pred_lr_improved = lr.predict(X_test)
print("\nImproved Linear Regression Performance:")
print(f"R^2 Score: {r2_score(y_test, y_pred_lr_improved):.4f}")
print(f"MAE: {mean_absolute_error(y_test, y_pred_lr_improved):,.2f}")
print(f"MSE: {mean_squared_error(y_test, y_pred_lr_improved):,.2f}")
print(f"RMSE: {np.sqrt(mean_squared_error(y_test,
y_pred_lr_improved)):.2f}")

# === IMPROVED POLYNOMIAL REGRESSION METRICS ===
y_pred_poly_improved = lr_poly.predict(X_test_poly)
print("\nImproved Polynomial Regression Performance:")
print(f"R^2 Score: {r2_score(y_test_poly, y_pred_poly_improved):.4f}")
print(f"MAE: {mean_absolute_error(y_test_poly,
y_pred_poly_improved):,.2f}")
print(f"MSE: {mean_squared_error(y_test_poly,
y_pred_poly_improved):,.2f}")
print(f"RMSE: {np.sqrt(mean_squared_error(y_test_poly,
y_pred_poly_improved)):.2f}")

# === TUNED RANDOM FOREST METRICS ===
y_pred_rf_tuned = best_rf.predict(X_test)
print("\nTuned Random Forest Performance:")
print(f"R^2 Score: {r2_score(y_test, y_pred_rf_tuned):.4f}")
print(f"MAE: {mean_absolute_error(y_test, y_pred_rf_tuned):,.2f}")
print(f"MSE: {mean_squared_error(y_test, y_pred_rf_tuned):,.2f}")
print(f"RMSE: {np.sqrt(mean_squared_error(y_test,
y_pred_rf_tuned)):.2f}")
```

```
Improved Linear Regression Performance:
R^2 Score: 0.6575
MAE: 227,415.40
MSE: 113,370,838,444.63
RMSE: 336705.86

Improved Polynomial Regression Performance:
R^2 Score: 0.7450
MAE: 184,056.73
MSE: 84,382,196,837.42
RMSE: 290486.14

Tuned Random Forest Performance:
R^2 Score: 0.8000
MAE: 154,657.33
MSE: 66,179,588,509.94
RMSE: 257253.94
```

2. Price Trend

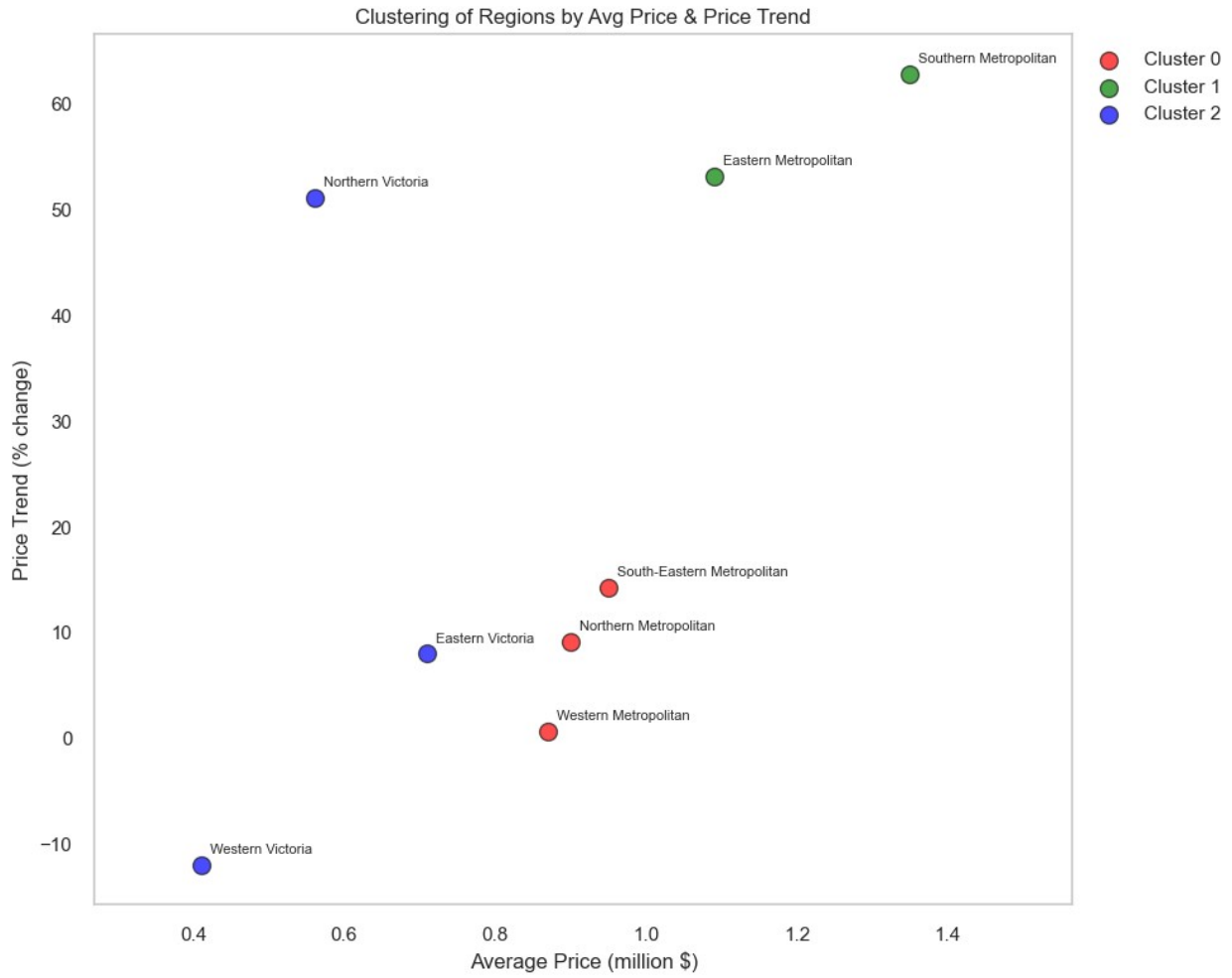
```
# 1. Ensure Date is datetime and sort
housingdata["Date"] = pd.to_datetime(housingdata["Date"],
errors="coerce")
housingdata = housingdata.sort_values("Date")

# 2. Aggregate average price per Region per Date
region_date = (
    housingdata
    .groupby(["Regionname", "Date"])["Price"]
    .mean()
    .reset_index()
)

# 3. Build the trend table with first/last dates, prices, avg price,
pct change
region_trends = []
for region in region_date["Regionname"].unique():
    rd = region_date[region_date["Regionname"] ==
region].sort_values("Date")
    if len(rd) < 2:
        continue
    first_date  = rd.iloc[0]["Date"]
    last_date   = rd.iloc[-1]["Date"]
    first_price = rd.iloc[0]["Price"]
    last_price  = rd.iloc[-1]["Price"]
    pct_change  = ((last_price - first_price) / first_price) * 100 if
first_price != 0 else 0
    avg_price   = rd["Price"].mean()
    region_trends.append({
```

```
        "Regionname": region,
        "FirstDate":  first_date,
        "LastDate":   last_date,
        "AvgPrice":   avg_price,
        "PctChange":  pct_change
    })

trend_df = pd.DataFrame(region_trends)

# 4. Convert avg price to millions and round
trend_df["AvgPrice_M"] = (trend_df["AvgPrice"] / 1e6).round(2)

# 5. K-means clustering on [AvgPrice_M, PctChange]
features = trend_df[["AvgPrice_M", "PctChange"]]
scaler = StandardScaler()
X_scaled = scaler.fit_transform(features)

kmeans = KMeans(n_clusters=3, random_state=42)
trend_df["Cluster"] = kmeans.fit_predict(X_scaled)

# 6. Sort by descending pct change (emerging → lowest)
trend_df = trend_df.sort_values("PctChange",
ascending=False).reset_index(drop=True)


print(

trend_df[["Regionname","FirstDate","LastDate","AvgPrice_M","PctChange"
,"Cluster"]]
    .to_string(
        index=False,
        formatters={
            "FirstDate": lambda d: d.strftime("%Y-%m-%d"),
            "LastDate":  lambda d: d.strftime("%Y-%m-%d"),
            "AvgPrice_M": "{:.2f}".format,
            "PctChange":  "{:.2f}".format,
        }
    )
)

# 8. Static clustering scatterplot
fig, ax = plt.subplots(figsize=(10, 8))
colors = {0: "red", 1: "green", 2: "blue"}

for c in sorted(trend_df["Cluster"].unique()):
    sub = trend_df[trend_df["Cluster"] == c]
    ax.scatter(
        sub["AvgPrice_M"],
        sub["PctChange"],
        c=colors[c],
```

```
        label=f"Cluster {c}",
        s=100,
        alpha=0.7,
        edgecolor="k"
    )
    for _, row in sub.iterrows():
        ax.annotate(
            row["Regionname"],
            xy=(row["AvgPrice_M"], row["PctChange"]),
            xytext=(5, 5),
            textcoords="offset points",
            ha="left",
            va="bottom",
            fontsize=8
        )

# Remove inner gridlines but keep the border
ax.grid(False)
for spine in ax.spines.values():
    spine.set_visible(True)

ax.margins(x=0.15, y=0.05)

xmin, xmax = ax.get_xlim()
ax.set_xlim(xmin, xmax * 1.05)

ax.set_xlabel("Average Price (million $)")
ax.set_ylabel("Price Trend (% change)")
ax.set_title("Clustering of Regions by Avg Price & Price Trend")
ax.legend(loc="upper left", bbox_to_anchor=(1, 1))
plt.tight_layout()
plt.show()
```

```
                Regionname  FirstDate   LastDate AvgPrice_M PctChange
Cluster
     Southern Metropolitan 2016-01-28 2017-09-23       1.35     62.85
1
      Eastern Metropolitan 2016-02-04 2017-09-23       1.09     53.17
1
         Northern Victoria 2017-05-27 2017-09-23       0.56     51.14
2
South-Eastern Metropolitan 2016-02-04 2017-09-23       0.95     14.28
0
     Northern Metropolitan 2016-02-04 2017-09-23       0.90      9.12
0
          Eastern Victoria 2017-05-27 2017-09-23       0.71      8.02
2
      Western Metropolitan 2016-02-04 2017-09-23       0.87      0.69
0
```

```
          Western Victoria 2017-05-27 2017-09-23       0.41    -11.91
2
```



3. Influence of Property Age and Size on Prediction Accuracy

```
housingdata['PropertyAge'] = housingdata['Date'].dt.year -
housingdata['YearBuilt']
housingdata['PropertyAge'].fillna(housingdata['PropertyAge'].median(),
inplace=True)

features = ['Rooms', 'Distance', 'Postcode', 'Bedroom2', 'Bathroom',
'Car',
            'Landsize', 'BuildingArea', 'PropertyAge']
X = housingdata[features].fillna(0)
y = housingdata['Price']

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

# Baseline model
```

```
base_model = RandomForestRegressor(random_state=42)
base_model.fit(X_train, y_train)
base_preds = base_model.predict(X_test)

# Initial evaluation
base_mae = mean_absolute_error(y_test, base_preds)
base_mse = mean_squared_error(y_test, base_preds)
rmse = np.sqrt(mean_squared_error(y_test, base_preds))
base_mae = mean_absolute_error(y_test, base_preds)

print(f"Initial MAE: ${base_mae:,.2f}")
print(f"Initial MSE: {base_mse:.2e}")
print(f"RMSE: ${rmse:.2f}")

housingdata['AbsError'] = np.abs(base_model.predict(X) - y)
age_bins = pd.qcut(housingdata['PropertyAge'], 10)
age_error = housingdata.groupby(age_bins)['AbsError'].mean()
print("\nProperty Age Error Analysis:")
print(age_error)

# Size Metric Analysis
size_metrics = ['Landsize', 'BuildingArea']
for metric in size_metrics:
    q25, q75 = X[metric].quantile([0.25, 0.75])
    lower = X[X[metric] <= q25]
    upper = X[X[metric] >= q75]

    lower_mae = mean_absolute_error(y[lower.index],
base_model.predict(lower))
    upper_mae = mean_absolute_error(y[upper.index],
base_model.predict(upper))

    print(f"\n{metric} Impact:")
    print(f"Smaller properties (<{q25:.0f}m²) MAE: ${lower_mae:,.2f}")
    print(f"Larger properties (>{q75:.0f}m²) MAE: ${upper_mae:,.2f}")
    print(f"Error increase: {(upper_mae -
lower_mae)/lower_mae*100:.1f}%")
```

```
C:\Users\tanyx\AppData\Local\Temp\ipykernel_32140\856777084.py:2:
FutureWarning: A value is trying to be set on a copy of a DataFrame or
Series through chained assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never
work because the intermediate object on which we are setting values
always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try
using 'df.method({col: value}, inplace=True)' or df[col] =
df[col].method(value) instead, to perform the operation inplace on the
original object.
```

```
housingdata['PropertyAge'].fillna(housingdata['PropertyAge'].median(),
inplace=True)

Initial MAE: $176,788.78
Initial MSE: 1.07e+11
RMSE: $326584.15

C:\Users\tanyx\AppData\Local\Temp\ipykernel_32140\856777084.py:28:
FutureWarning: The default of observed=False is deprecated and will be
changed to True in a future version of pandas. Pass observed=False to
retain current behavior or observed=True to adopt the future default
and silence this warning.
  age_error = housingdata.groupby(age_bins)['AbsError'].mean()


Property Age Error Analysis:
PropertyAge
(-2.001, 9.0]      71067.410175
(9.0, 19.0]        79499.810786
(19.0, 37.0]       66585.052251
(37.0, 46.0]       65962.906458
(46.0, 51.0]       64726.198076
(51.0, 57.0]       73008.901682
(57.0, 67.0]       94876.985238
(67.0, 86.0]      131123.198594
(86.0, 106.0]     122819.075716
(106.0, 821.0]    143251.234171
Name: AbsError, dtype: float64

Landsize Impact:
Smaller properties (<177m²) MAE: $66,342.32
Larger properties (>650m²) MAE: $121,294.82
Error increase: 82.8%

BuildingArea Impact:
Smaller properties (<94m²) MAE: $58,343.37
Larger properties (>164m²) MAE: $142,290.94
Error increase: 143.9%
```

Improvements to Baseline Model

This is a base default reference that we will be using as comparison after we improve the model by applying the following techniques:

1. Feature Engineering
2. Hyperparameter tuning

```
housingdata['Rooms_Bathroom'] = housingdata['Rooms'] *
housingdata['Bathroom']
```

```
housingdata['Landsize_BuildingArea'] = housingdata['Landsize'] *
housingdata['BuildingArea']

improved_features = features + ['Rooms_Bathroom',
'Landsize_BuildingArea']
X_improved = housingdata[improved_features].fillna(0)

param_grid = {
    'n_estimators': [200, 300],
    'max_depth': [15, 25, None],
    'min_samples_split': [2, 5]
}
grid_search = GridSearchCV(base_model, param_grid, cv=3,
scoring='neg_mean_absolute_error', n_jobs=-1)
grid_search.fit(X_improved, y)
best_model = grid_search.best_estimator_

# Split with improved features
X_train_imp, X_test_imp, y_train_imp, y_test_imp =
train_test_split(X_improved, y, test_size=0.2, random_state=42)

# Final evaluation
best_preds = best_model.predict(X_test_imp)
improved_mae = mean_absolute_error(y_test_imp, best_preds)
improved_mse = mean_squared_error(y_test_imp, best_preds)

print(f"\nImproved MAE: ${improved_mae:,.2f}")
print(f"Improved MSE: {improved_mse:.2e}")
print(f"MAE Improvement: {(base_mae - improved_mae)/base_mae*100:.1f}
%")

# Property Age Impact Analysis
housingdata['AbsError'] = np.abs(best_model.predict(X_improved) - y)
age_bins = pd.qcut(housingdata['PropertyAge'], 10)
age_error = housingdata.groupby(age_bins)['AbsError'].mean()
print("\nProperty Age Error Analysis:")
print(age_error)

# Size Metric Analysis
size_metrics = ['Landsize', 'BuildingArea']
for metric in size_metrics:
    q25, q75 = X_improved[metric].quantile([0.25, 0.75])
    lower = X_improved[X_improved[metric] <= q25]
    upper = X_improved[X_improved[metric] >= q75]

    lower_mae = mean_absolute_error(y[lower.index],
best_model.predict(lower))
    upper_mae = mean_absolute_error(y[upper.index],
best_model.predict(upper))
```

```
    print(f"\n{metric} Impact:")
    print(f"Smaller properties (<{q25:.0f}m²) MAE: ${lower_mae:,.2f}")
    print(f"Larger properties (>{q75:.0f}m²) MAE: ${upper_mae:,.2f}")
    print(f"Error increase: {(upper_mae -
lower_mae)/lower_mae*100:.1f}%")


Improved MAE: $80,438.03
Improved MSE: 2.34e+10
MAE Improvement: 54.5%

C:\Users\tanyx\AppData\Local\Temp\ipykernel_32140\3027385510.py:16:
FutureWarning: The default of observed=False is deprecated and will be
changed to True in a future version of pandas. Pass observed=False to
retain current behavior or observed=True to adopt the future default
and silence this warning.
  age_error = housingdata.groupby(age_bins)['AbsError'].mean()


Property Age Error Analysis:
PropertyAge
(-2.001, 9.0]      64440.303684
(9.0, 19.0]        69576.406971
(19.0, 37.0]       62911.475313
(37.0, 46.0]       60042.263362
(46.0, 51.0]       60647.010968
(51.0, 57.0]       66224.509914
(57.0, 67.0]       84956.617678
(67.0, 86.0]      118427.006981
(86.0, 106.0]     113433.351692
(106.0, 821.0]    129495.922793
Name: AbsError, dtype: float64

Landsize Impact:
Smaller properties (<177m²) MAE: $61,496.75
Larger properties (>650m²) MAE: $109,233.73
Error increase: 77.6%

BuildingArea Impact:
Smaller properties (<94m²) MAE: $55,946.06
Larger properties (>164m²) MAE: $126,015.07
Error increase: 125.2%
```

(4) Conclusion

1. Which features have the highest impact on price based on feature importance?
2. How has the price trend varied across different regions over time, and can we identify emerging high-value areas?
3. How does property age and size influence prediction accuracy in our models?

4. What can stakeholders take away from our project?
5. Our Takeaways

1. Which features have the highest impact on price based on feature importance?

We found that BuildingArea, and location-based features such as Longitude and Latitude are the most influential predictors of price.
Through Random Forest regression, we gained better accuracy compared to linear models, especially after fine-tuning and feature engineering.
This is because Random Forest models are ensemble-based and capable of capturing complex, non-linear interactions between features — something that linear models inherently struggle with. Additionally, Random Forests are more robust to outliers and multicollinearity, and naturally provide feature importance metrics, making them highly suitable for structured tabular data like real estate.

2. How has the price trend varied across different regions over time, and can we identify emerging high-value areas?

Our K-means clustering approach uncovered patterns in regional price trends, with Southern and Eastern Metropolitan emerging as high-growth regions. Conversely, Western Victoria faced a decline.
While clustering helped uncover spatial price dynamics, we encountered limitations in our dataset's time-series resolution. Most entries lacked consistent temporal granularity, making it difficult to perform dynamic clustering over time. This constrained our ability to track shifting trends month-over-month or year-over-year. With a more complete and time-aligned dataset, we could better capture emerging hotspots and accurately forecast future growth areas.

3. How does property age and size influence prediction accuracy in our models?

We identified model weaknesses in predicting prices for older and larger properties, which traditionally introduce more variability. By engineering better features and tuning the model, we halved our MAE, making our predictions more robust and reliable.
This variability is likely due to older properties having undergone renovations or depreciations not consistently captured in the dataset, and larger properties being more sensitive to neighborhood and design factors. These nuances add noise that models struggle to generalize over. Addressing this required outlier handling, scaling transformations, and refined encoding of categorical variables linked to property type and condition.

4. What can stakeholders take away from our project?

Investors and buyers should pay attention to size and age-related price deviations — these are where overpricing or undervaluation might be hidden.

Urban planners and policymakers can use models like ours to monitor and anticipate growth in specific regions, helping guide infrastructure development or cooling measures.

In the context of Singapore, such models could be applied to areas like the Greater Southern Waterfront or Jurong Lake District, where long-term development plans are in motion. Understanding where undervalued areas exist can also support more equitable housing policies.

5. Our Takeaways

- **Use non-synthetic, real-world datasets**
  Real datasets reflect actual noise, complexity, and true feature interactions, which ultimately help models generalize more effectively to new data.

- **Use datasets with a healthy mix of numerical and categorical features**
  This enables broader experimentation with preprocessing techniques and supports diverse modeling approaches like tree-based algorithms and clustering.

- **Perform prudent Exploratory Data Analysis (EDA)**
  EDA is crucial for uncovering hidden patterns, detecting outliers, managing missing values, and ensuring a clean dataset for modeling.

- **Craft problem statements that align with the dataset's nature and your end-goal**
  Rather than forcing a question onto the dataset, it's often more powerful to derive a question from what the data naturally supports. This leads to more actionable and data-driven outcomes.

- **Select models based on problem type, data size, and interpretability needs**
  For instance:

  - Use tree-based models for structured, mixed-type datasets where interpretability is key.
  - Choose neural networks for high-dimensional data with complex non-linear patterns.