Melbourne Housing Prices

Jose Ramirez

Agenda

- Introduction to the Data Set
- Analysis Models Used
  - Clustering
  - Regression
  - Classification
- Summary of Findings

Introduction to the Data Set

Exploratory Data Analysis

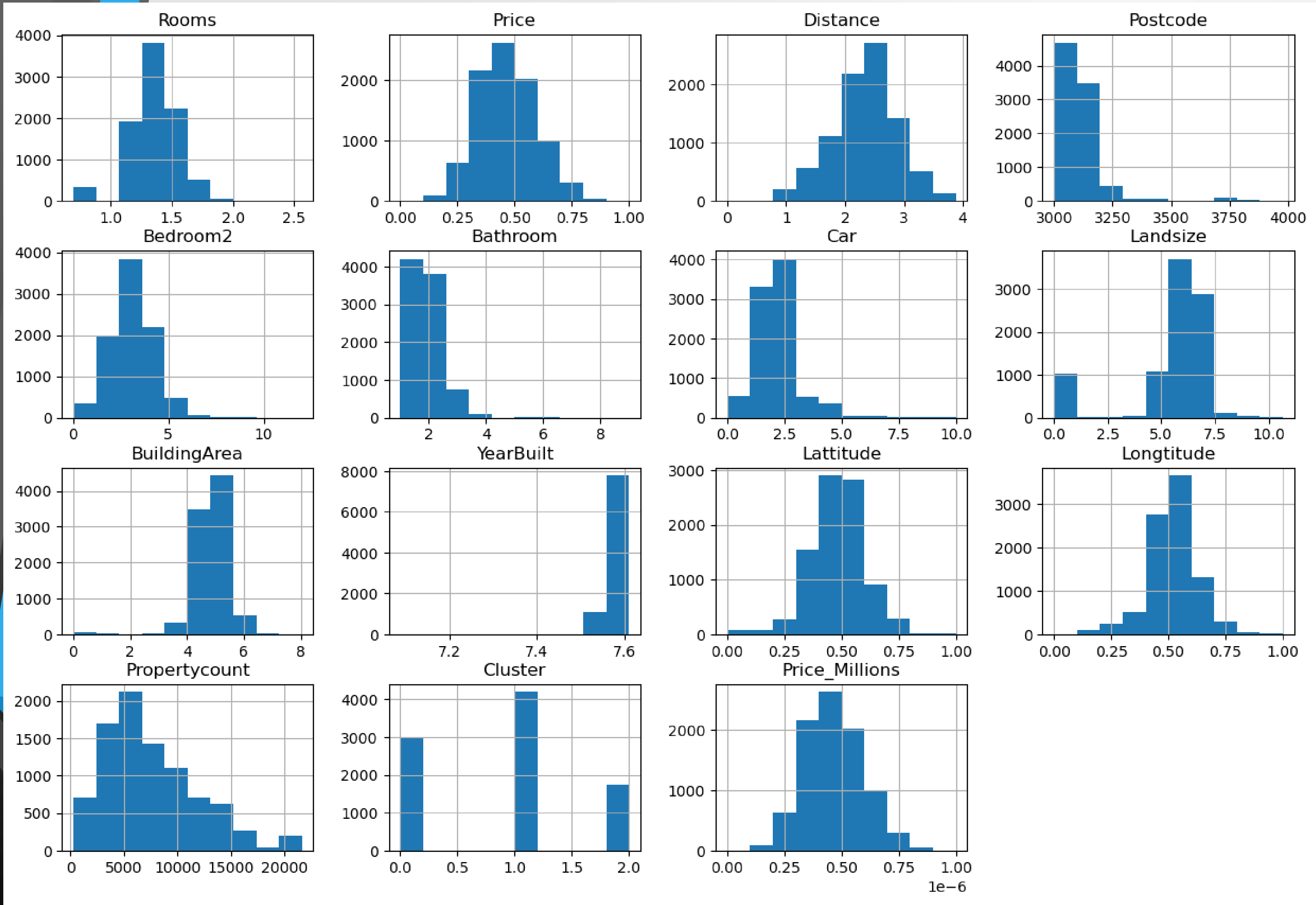
List of data types

```
<class 'pandas.core.frame.DataFrame'>
Index: 34857 entries, Abbotsford to Yarraville
Data columns (total 20 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   Address        34857 non-null  object
 1   Rooms          34857 non-null  int64
 2   Type           34857 non-null  object
 3   Price          27247 non-null  float64
 4   Method         34857 non-null  object
 5   SellerG        34857 non-null  object
 6   Date           34857 non-null  object
 7   Distance       34856 non-null  float64
 8   Postcode       34856 non-null  float64
 9   Bedroom2       26640 non-null  float64
 10  Bathroom       26631 non-null  float64
 11  Car            26129 non-null  float64
 12  Landsize       23047 non-null  float64
 13  BuildingArea   13742 non-null  float64
 14  YearBuilt      15551 non-null  float64
 15  CouncilArea    34854 non-null  object
 16  Lattitude      26881 non-null  float64
 17  Longtitude     26881 non-null  float64
 18  Regionname     34854 non-null  object
 19  Propertycount  34854 non-null  float64
dtypes: float64(12), int64(1), object(7)
memory usage: 5.6+ MB
```

Sum of columns with null values

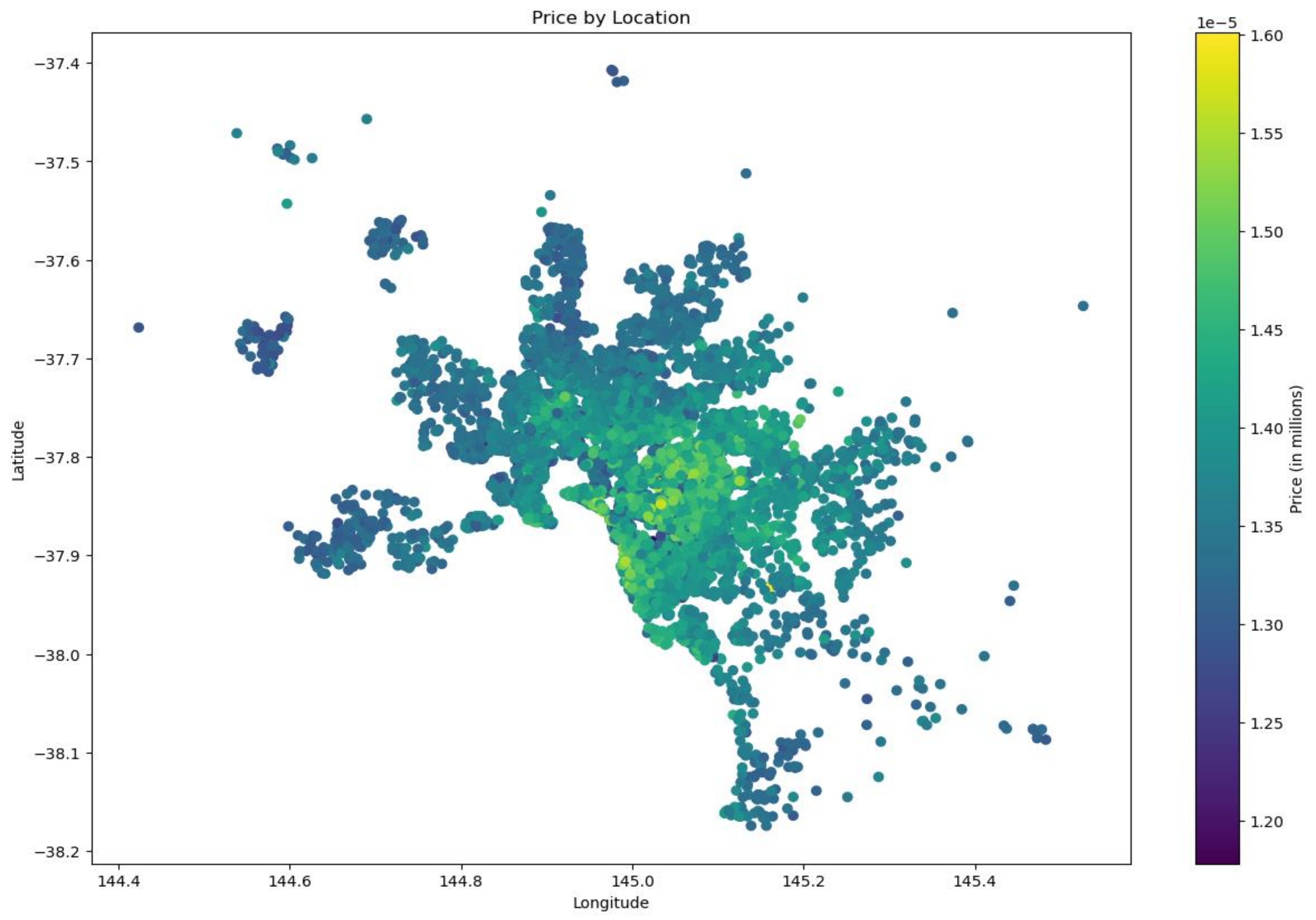
```
BuildingArea     21115
YearBuilt        19306
Landsize         11810
Car               8728
Bathroom          8226
Bedroom2          8217
Longtitude        7976
Lattitude         7976
Price             7610
Regionname           3
CouncilArea          3
Propertycount        3
Postcode             1
Distance             1
Rooms                0
Date                 0
SellerG              0
Method               0
Type                 0
Address              0
dtype: int64
```

Exploratory Data Analysis



Using the **histogram**, to see the distribution of our dataset

Exploratory Data Analysis

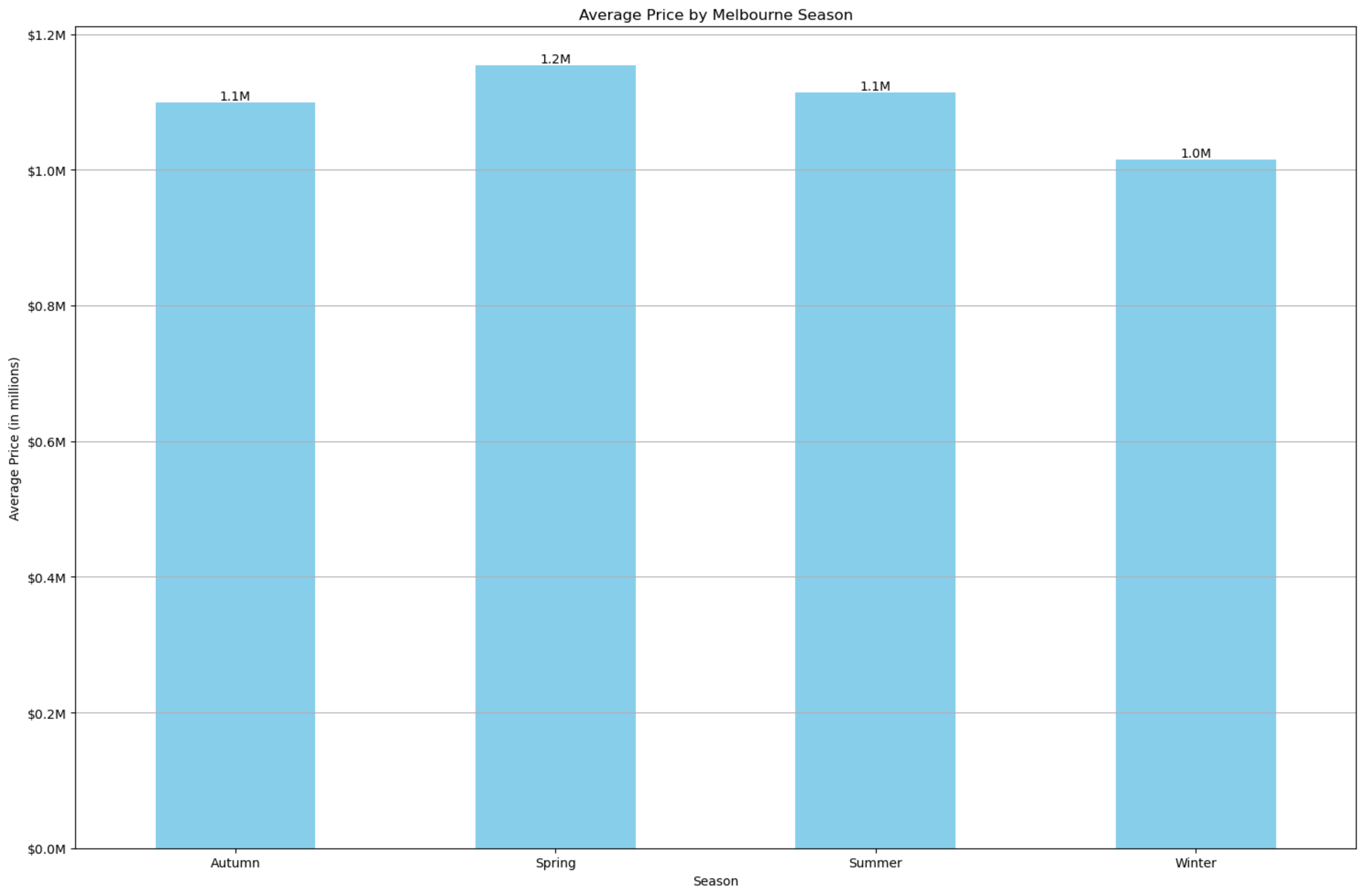


Using the **<u>scatterplot</u>**, we can have a visualization of how housing prices are distributed across Melbourne, Australia.

Exploratory Data Analysis

Seasonal Analysis

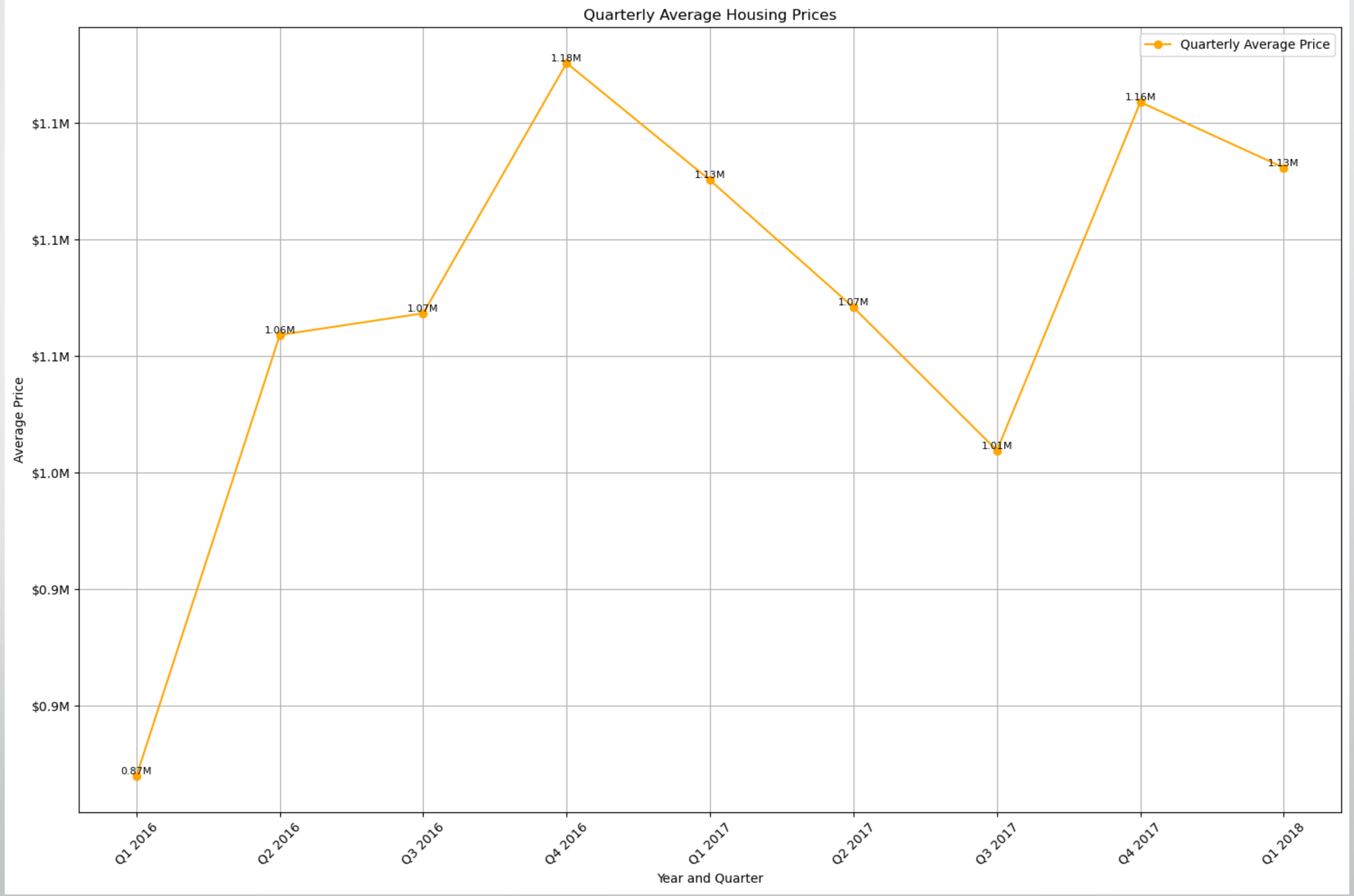
Based on this graph, the highest housing process are recorded during **Spring**, however, there is **no significant difference compared to the other seasons**.



Exploratory Data Analysis

Seasonal Analysis

This graph shows the price trend from 2016 to 2018. This clearly shows the **<u>peak of housing prices during the Q4 of 2016</u>**.



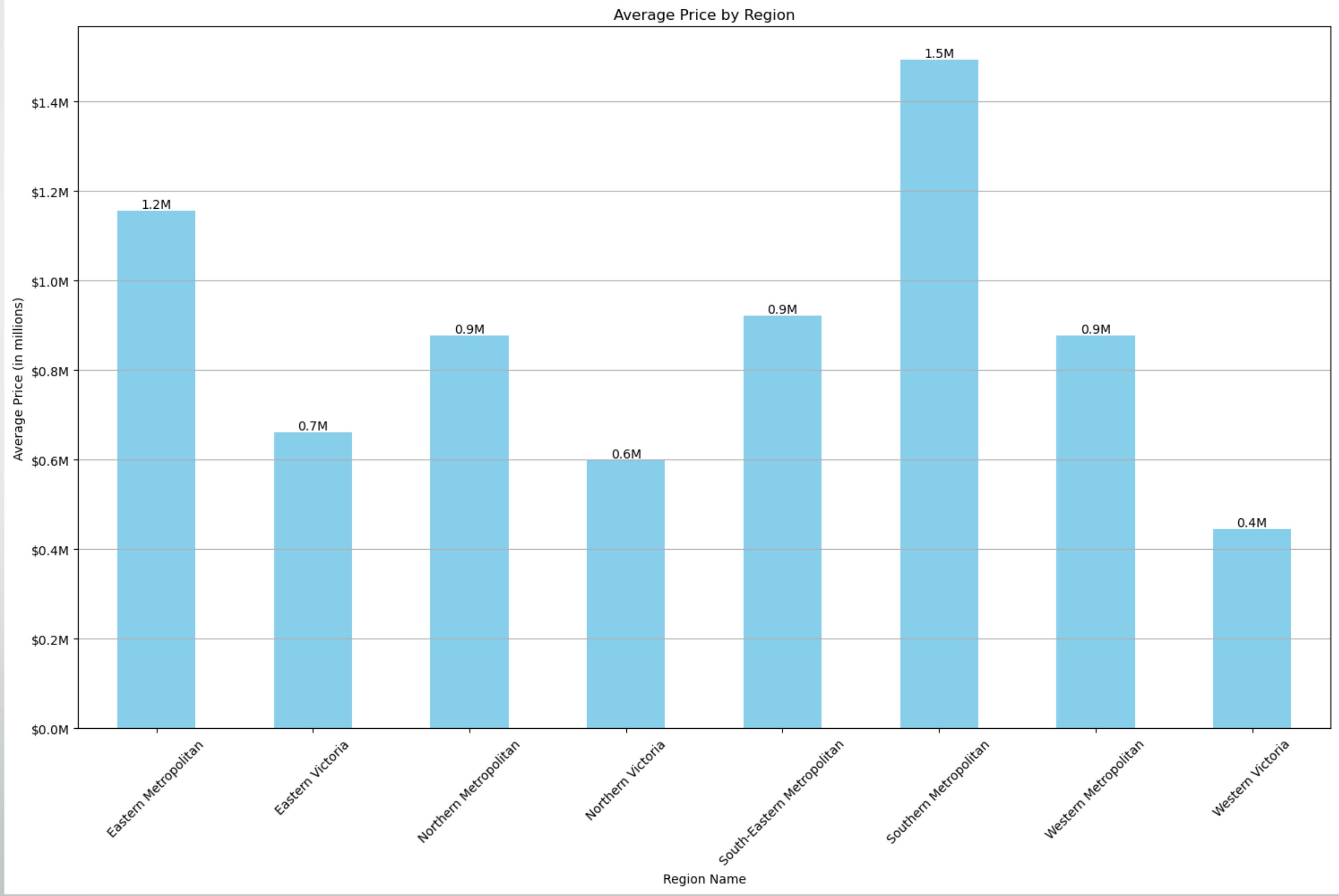
Introduction: Exploratory Data Analysis

Prices by Region

The **<u>highest housing prices is in Southern Metropolitan</u>**.

Based on our research, this region covers most of the wealthiest areas in Melbourne.

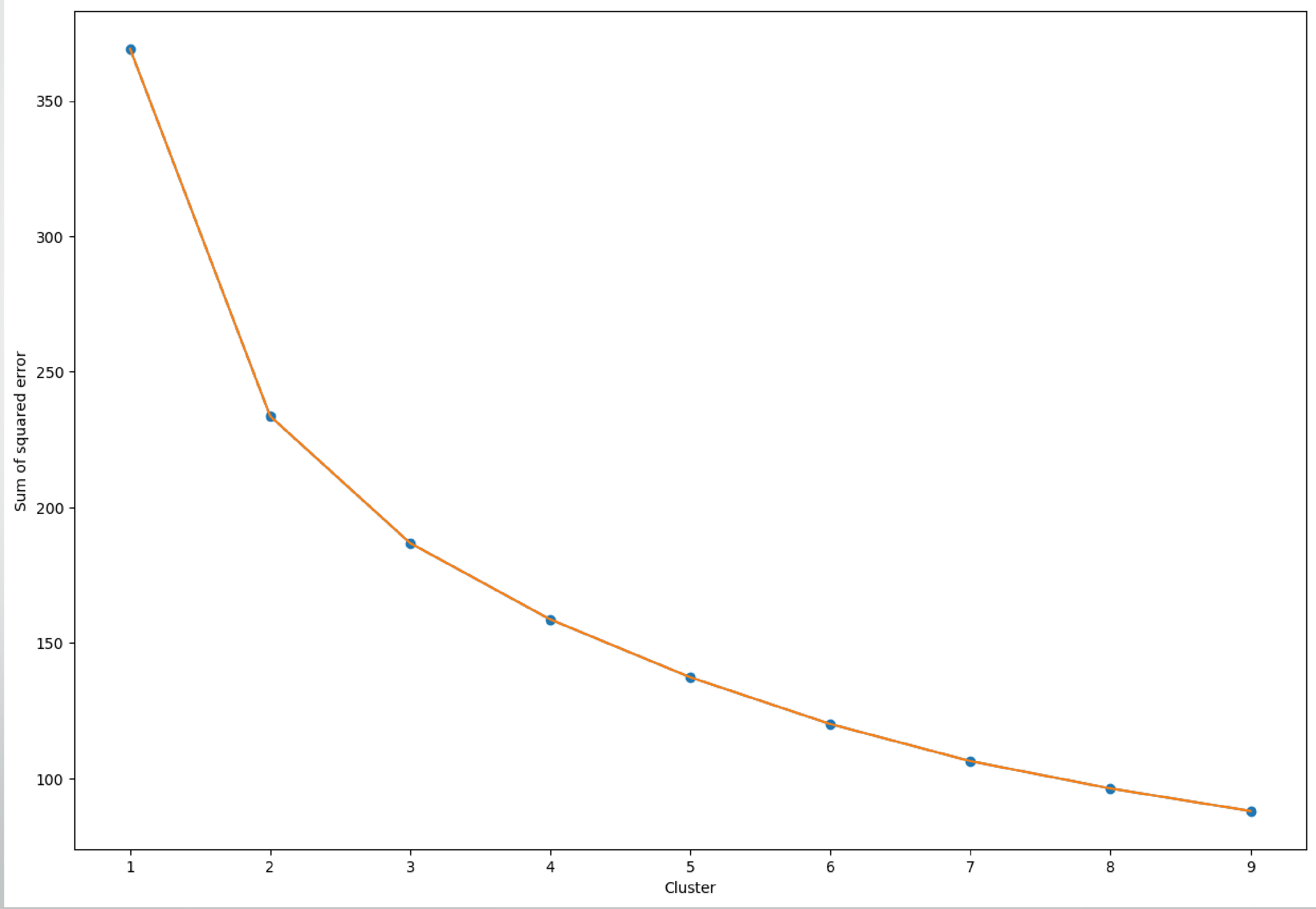
[Click here for Reference]



Data Analysis Model

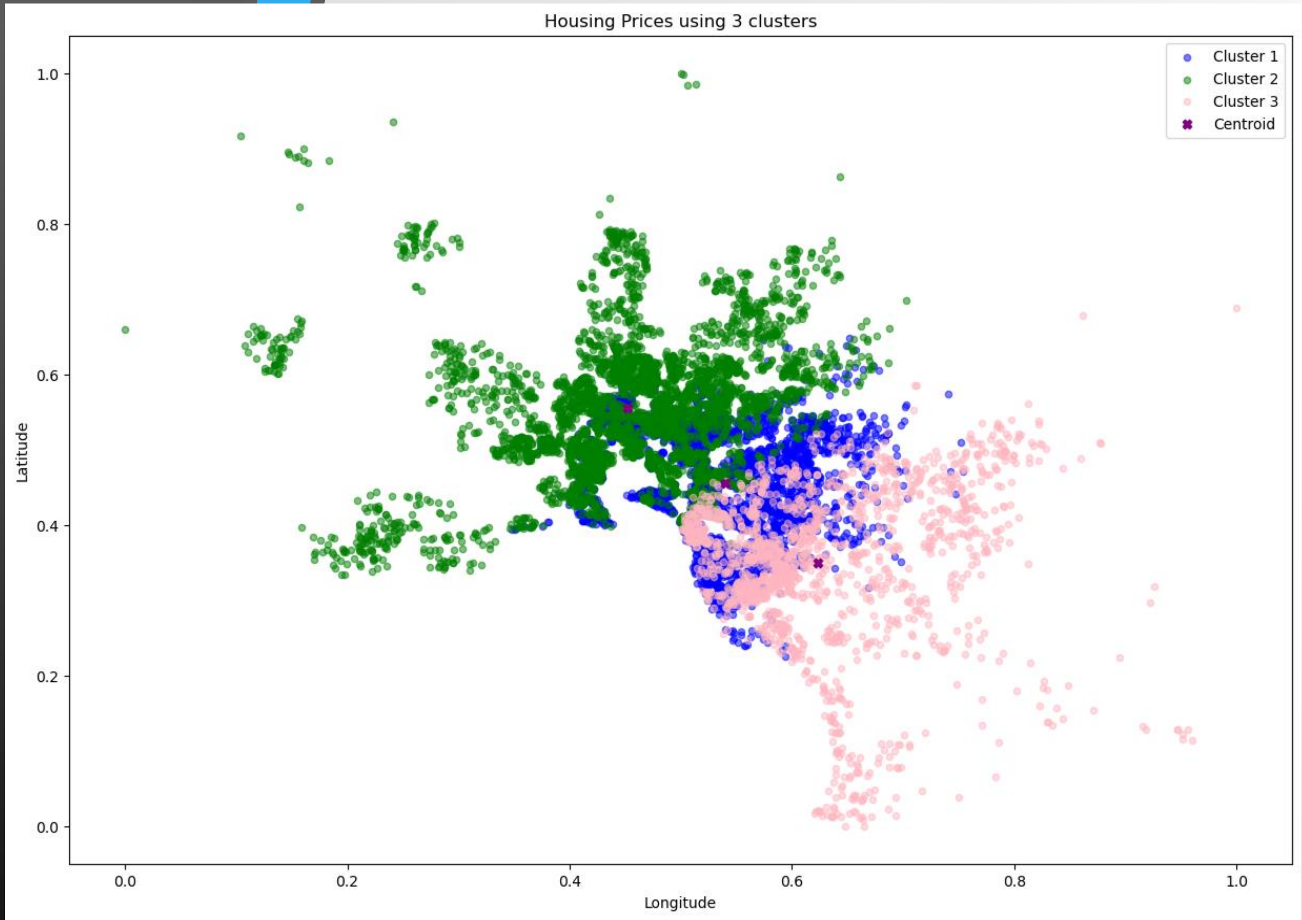
Descriptive Model: Clustering

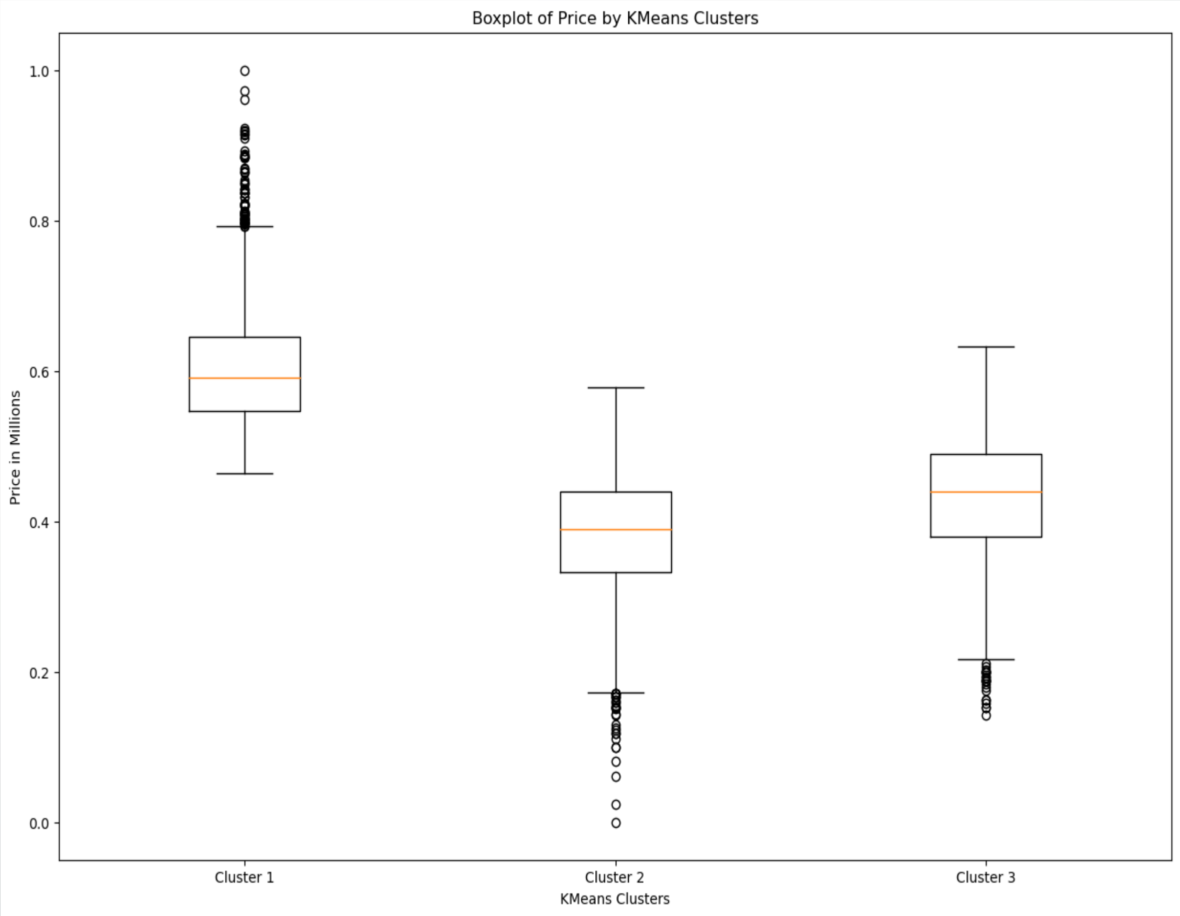
In this model, we did a **K-means clustering** using three features from our dataset which are Longitude, Latitude and Price.



Descriptive Model: Clustering

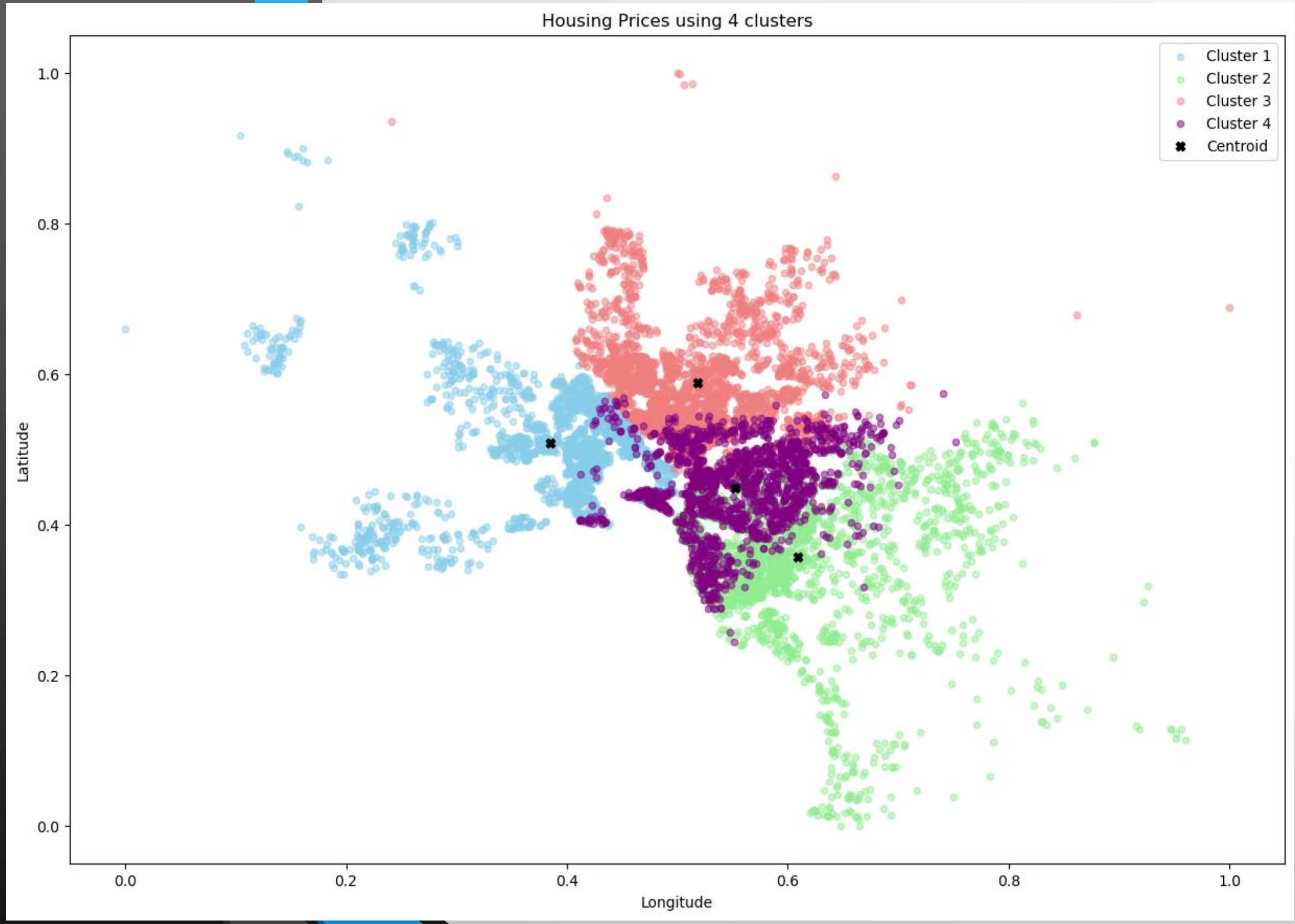
Using 3 Clusters

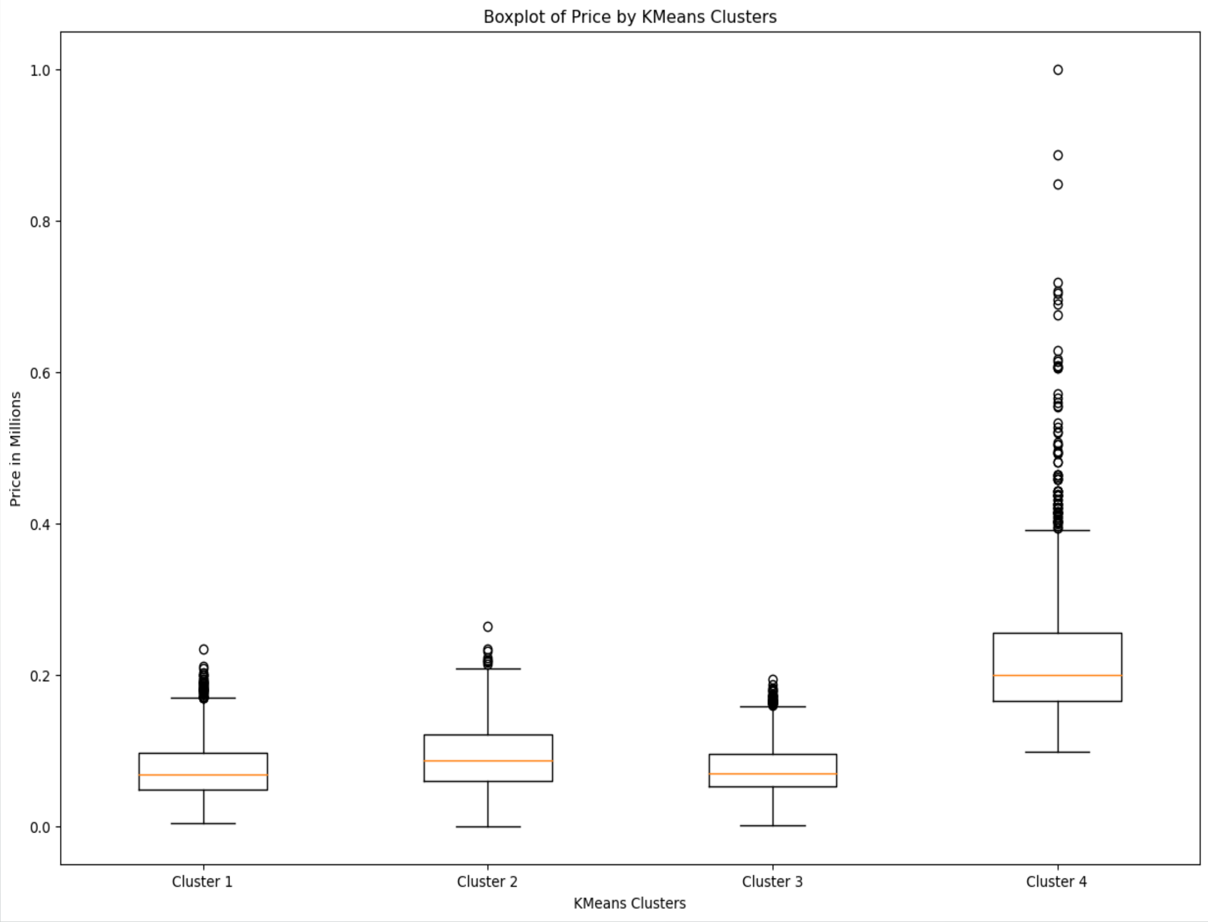




Descriptive Model: Clustering

Using 4 Clusters

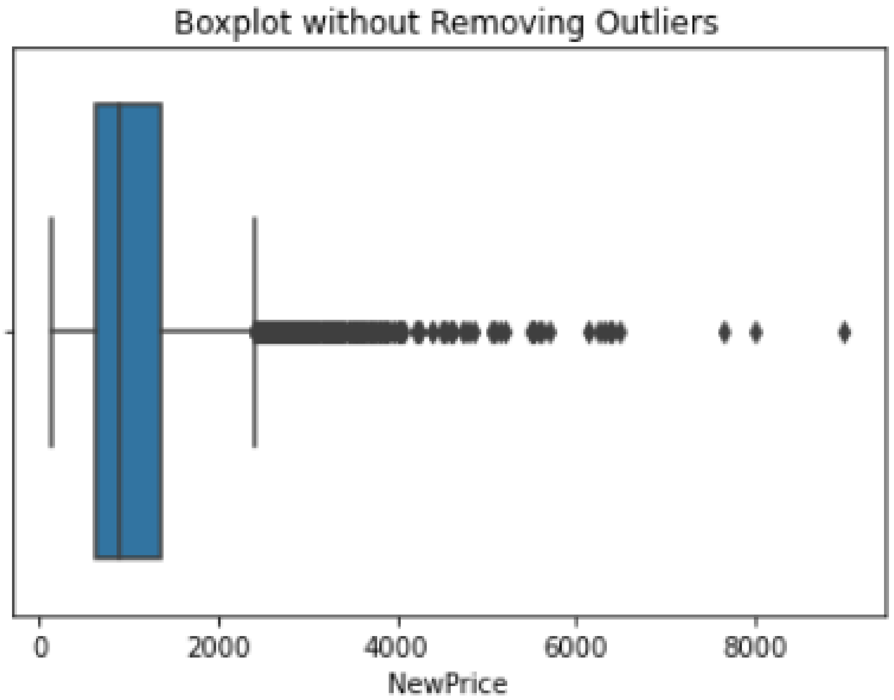




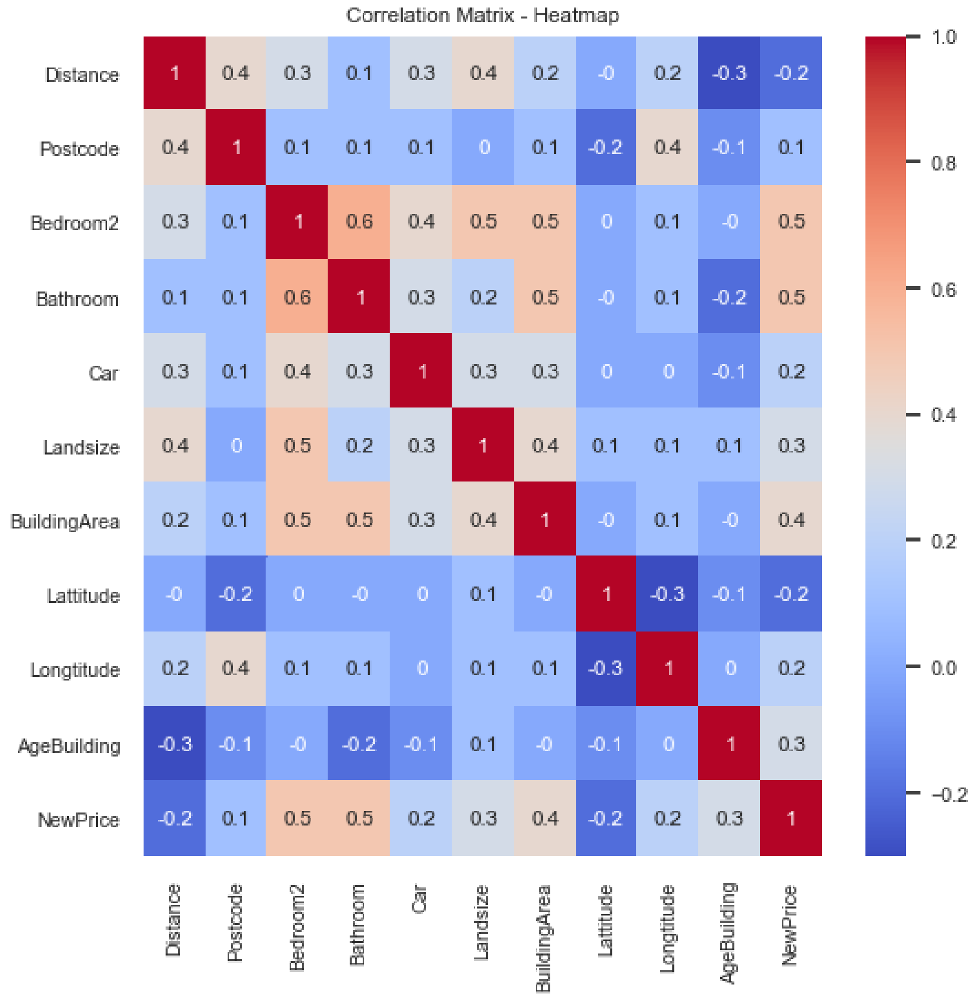
Data Analysis: Regression

Outliers in response Price (Y)

In [18]:
```python
# Create a boxplot without removing outliers
plt.figure(figsize=(6, 4))
sns.boxplot(x=df['NewPrice'])
plt.title('Boxplot without Removing Outliers')
plt.show()
```



Correlation of Features (X)



Predictive Model: Regression

Data Preparation and Training for Regression Models

1. Drop categorical data and NA

```
In [5]: #Drop the unnecessary columns
        df.drop(columns = {'Suburb','Address', 'Rooms', 'Type', 'Method', 'SellerG',
                           'Date', 'CouncilArea', 'Regionname', 'Propertycount'}, axis = 1, inplace = True)
```

```
#drop na values
df.dropna(inplace=True)
```

2. Normalization of the features

```
In [16]: df['Distance'] = np.log(df['Distance'] + 1)
         df['Postcode'] = np.log(df['Postcode'] + 1)
         df['Landsize'] = np.log(df['Landsize'] + 1)
         df['BuildingArea'] = np.log(df['BuildingArea'] + 1)
```

3. Feature Scaling

```
In [24]: features = df.drop("NewPrice",axis=1)
         response = df["NewPrice"]

In [25]: #Feature scaling using MinMaxScaler()
         from sklearn.preprocessing import MinMaxScaler

         min_max_scaler = MinMaxScaler(feature_range=(-1, 1))
         housing_num_min_max_scaled = min_max_scaler.fit_transform(features)
```

4. Training and Test (test_size=0.2, random_state = 42)

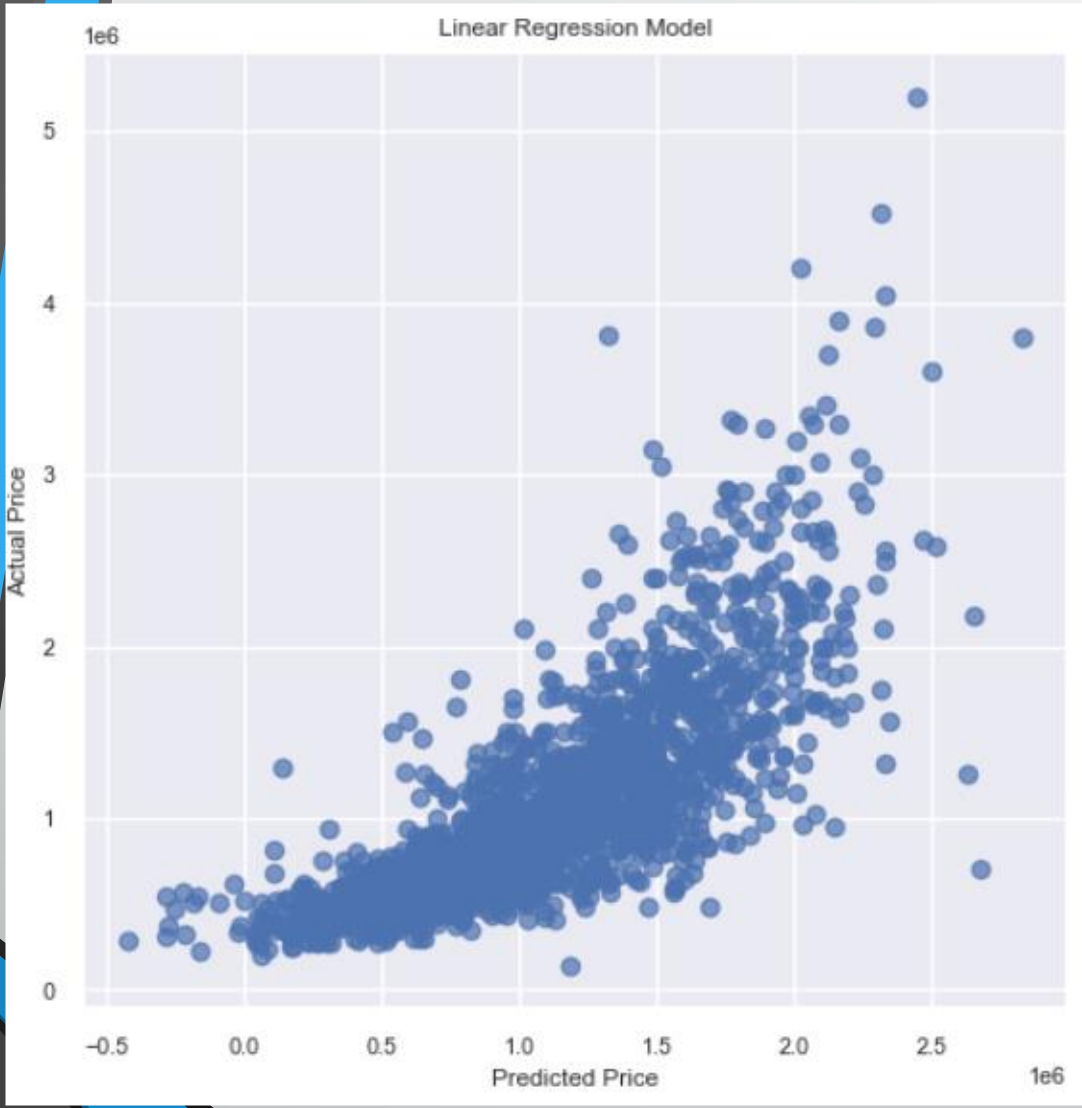
```
In [26]: # Import 'train_test_split'
         from sklearn.model_selection import train_test_split

In [27]: # Split the data into training and testing subsets
         X_train, X_test, Y_train, Y_test = train_test_split(housing_num_min_max_scaled,
                                                              response, test_size=0.2, random_state = 42)

         # Success
         print("Training and testing split was successful.")

         Training and testing split was successful.
```

Predictive Model: Linear Regression




```
r2_train = model_lr.score(X_train,Y_train)
print ("R^2 in Training Set: ", r2_train)

r2_test = model_lr.score(X_test,Y_test)
print ("R^2 in Test Set is: ", r2_test)
```

```
R^2 in Training Set:  0.5613764739459174
R^2 in Test Set is:  0.5997601617350488
```

```
from sklearn.metrics import mean_squared_error

rmse = mean_squared_error(Y_test, predictions_lr)**0.5

print ('RMSE is: \n', rmse)
```

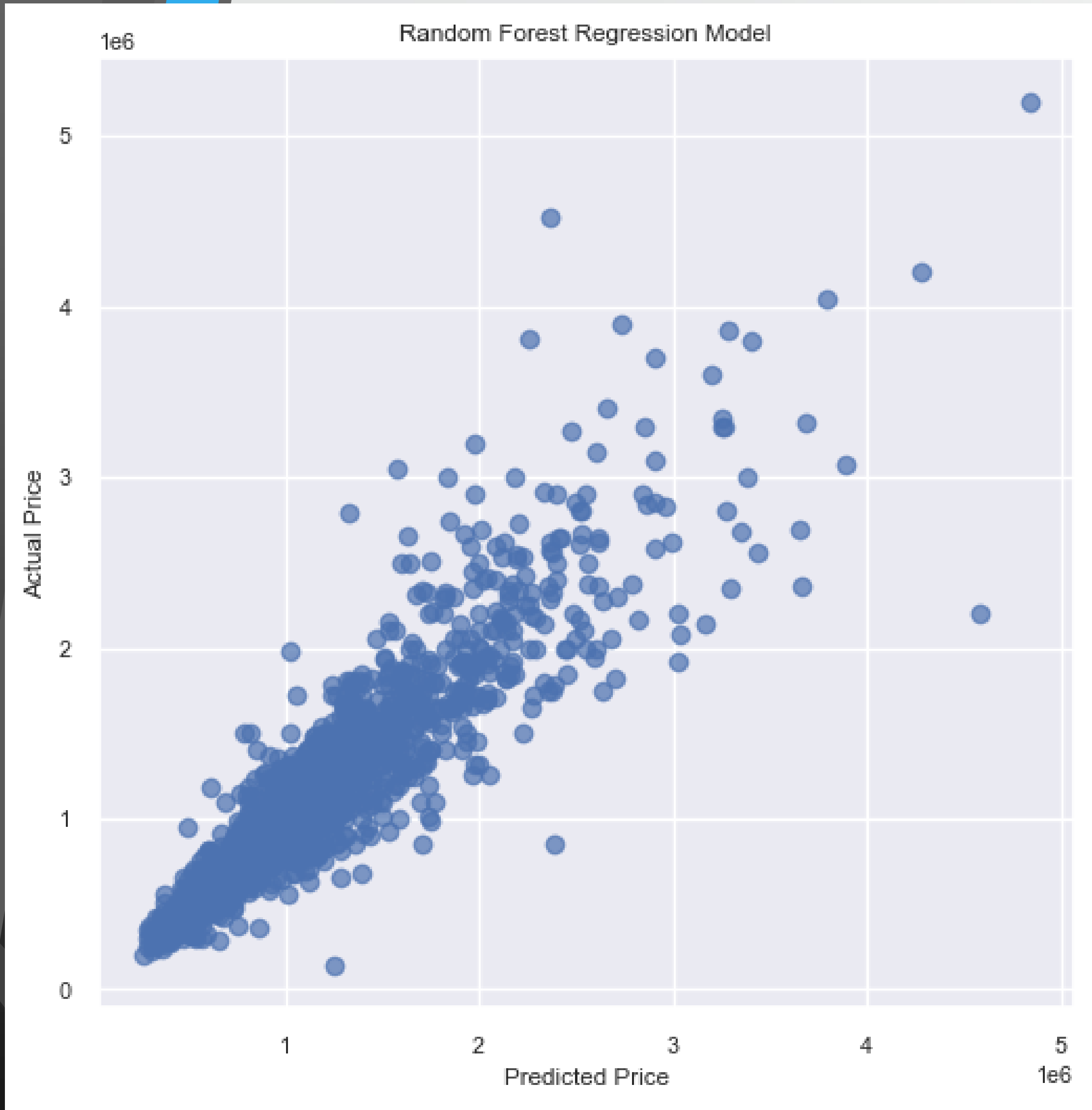
```
RMSE is: 
 392143.72119057097
```

<u>R2 and RMSE</u> were computed to check the performance.

<u>R2 in training</u>: 56.14%

<u>R2 in test</u>: 59.98%

Predictive Model: Random Forest Regression




```
# Initialize the Random Forest Regressor
rf_regressor = RandomForestRegressor(n_estimators=100, random_state=42)
```

```
# Train the model
rf_regressor.fit(X_train, y_train)
```

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

```
# Make predictions on the test set
predictions_rf = rf_regressor.predict(X_test)
```

```
# Evaluate the model
from sklearn.metrics import mean_squared_error, r2_score

r2_train = rf_regressor.score(X_train,Y_train)
print ("R^2 in Training Set: ", r2_train)

r2_test = rf_regressor.score(X_test,Y_test)
print ("R^2 in Test Set is: ", r2_test)
```

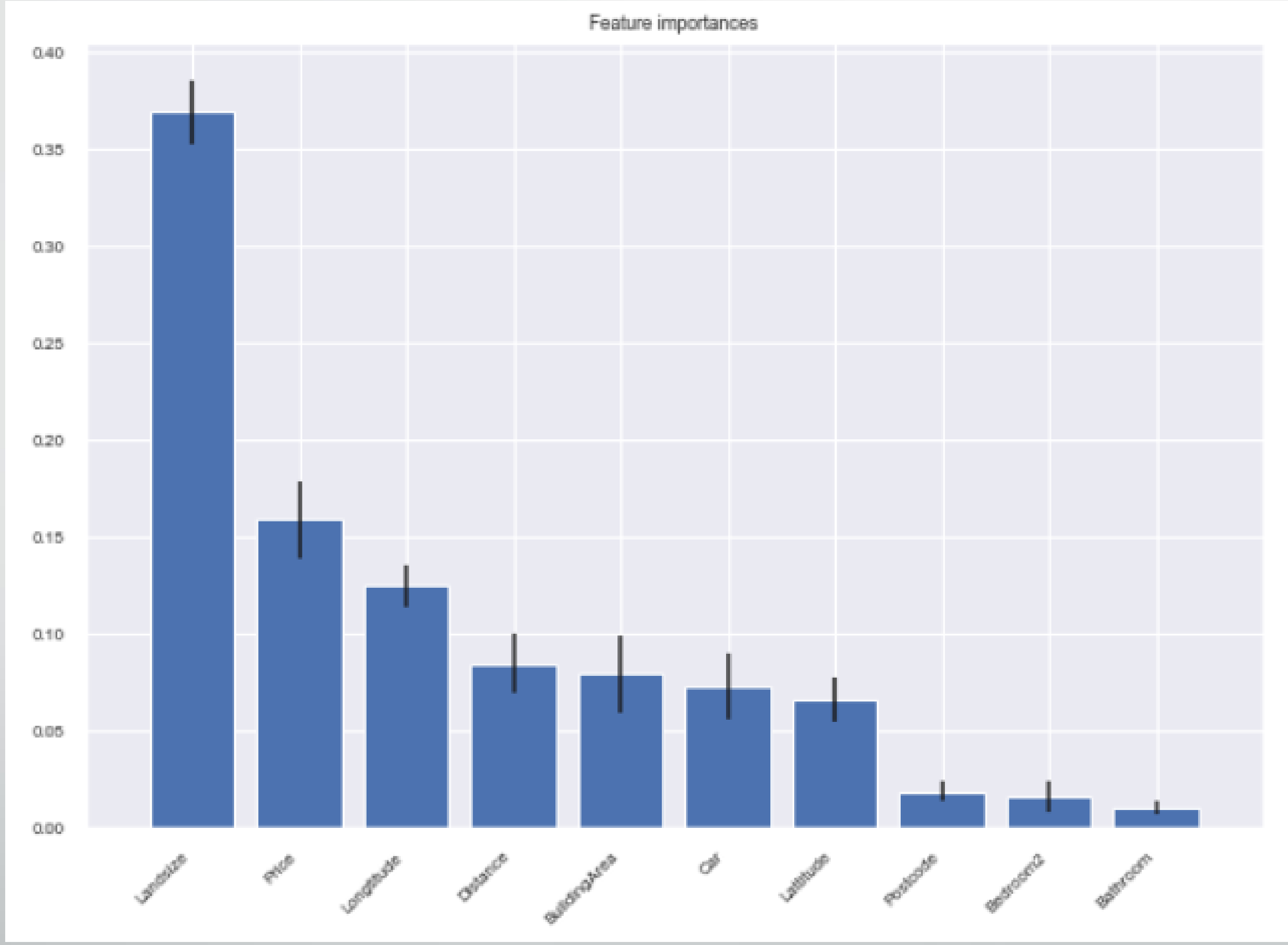
```
R^2 in Training Set:  0.9715980620825022
R^2 in Test Set is:  0.8382991069502059
```

```
mse = mean_squared_error(y_test, predictions_rf)
print(f"Mean Squared Error (MSE): {mse}")
```

```
Mean Squared Error (MSE): 62127322247.15614
```

<u>R2 in training</u>: 97.16%
<u>R2 in test</u>: 83.83%
Evidence of some level of Overfitting.

Feature Importance: Regression



Predictions of New Data: Regression

```
#new data with dropped columns
#Distance, Postcode, Bedroom2, Bathroom, Car, Landsize,
        #BuildingArea, Lattitude, Longtitude, AgeBuilding

new_data = [
    [3.3, 3206, 4, 2, 1, 330, 207, -37.8477, 144.9558, 5],
    [18, 3037, 3, 2, 1, 453, 153, -37.68811, 144.75, 103],
    [5.9, 3032, 1, 1, 1, 0, 58, -37.7723, 144.9094, 51]
]
```

```
#Scaling the new data, using the same MinMax Scaler from the original dataset
new_data_scaled = min_max_scaler.transform(new_data)
```

```
# Predict prices for scaled new data
predicted_prices = rf_regressor.predict(new_data_scaled)
print("Predicted prices for new data:")
for i, price in enumerate(predicted_prices):
    print(f"Predicted Price {i+1}: ${price:,.0f}")
```

```
Predicted prices for new data:
Predicted Price 1: $2,448,955
Predicted Price 2: $1,413,530
Predicted Price 3: $1,298,934
```

Predictive Model: Classification

```
#Drop the unnecessary columns
df.drop(columns = {'Address','CouncilArea', 'Rooms', 'SellerG', 'Date', 'Propertycount', 'Suburb'}, axis = 1, inplace = True)
```

[ ]

```
df.dropna(inplace=True)
```

[ ]

```
# Convert YearBuilt to AgeBuilding
current_year = datetime.now().year
df['AgeBuilding'] = current_year-df['YearBuilt'].astype(int)
```

[ ]

```
df.drop('YearBuilt', axis=1, inplace=True)
```

[ ]

```
print(df.columns)
```

[13] ✓ 0.0s

```
Index(['Suburb', 'Address', 'Rooms', 'Type', 'Price', 'Method', 'SellerG',
       'Date', 'Distance', 'Postcode', 'Bedroom2', 'Bathroom', 'Car',
       'Landsize', 'BuildingArea', 'YearBuilt', 'CouncilArea', 'Lattitude',
       'Longtitude', 'Regionname', 'Propertycount'],
      dtype='object')
```

Data Clean up

- Delete the columns that are not required in the model
- Delete the NaN values
- Convert YearBuilt to AgeBuilding
- Create dummy encoding for the columns Type, Method and RegionName.

Predictive Model: Classification

```
[ ] df = pd.get_dummies(df, columns=['Type'])
```

```
[ ] df = pd.get_dummies(df, columns=['Method'])
```

```
[ ] df = pd.get_dummies(df, columns=['Regionname'])
```

Dummy encoding was used.

```
# Define bins and labels for the price categories
#Bins were defined based on Q1 (25%) and Q3 (75%)
bins = [0,641,1345, float('inf')]  # specify the bin edges of the Prices
labels = ['Low', 'Medium','High']  # specify labels for each Price category
```

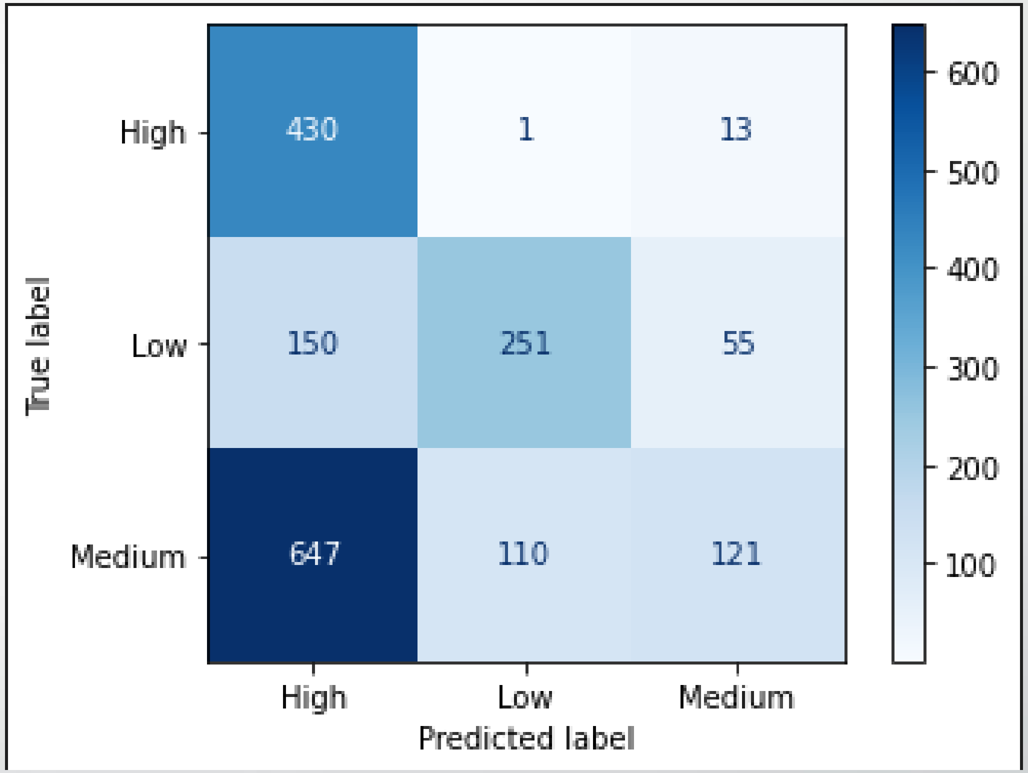
```
# Create a new column 'Price_Category' with the categorical values
df['Price_Category'] = pd.cut(df['NewPrice'], bins=bins, labels=labels, right=False)
```

A Categories Price column (Low, Medium, High) was also created which we will use for analysis on this model.

Predictive Model: Classification

	precision	recall	f1-score	support
High	0.35	0.97	0.51	444
Low	0.69	0.55	0.61	456
Medium	0.64	0.14	0.23	878
accuracy			0.45	1778
macro avg	0.56	0.55	0.45	1778
weighted avg	0.58	0.45	0.40	1778

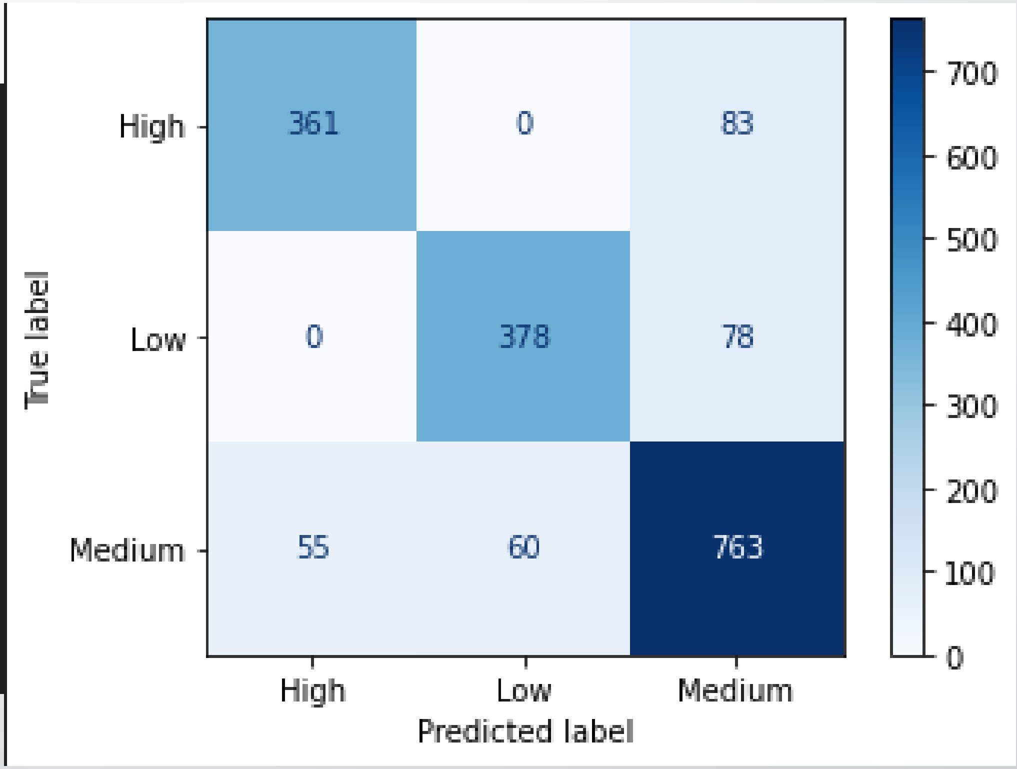
We only got **<u>45% accuracy using Naïve Bayes Model</u>**. Hence, we will explore another model to get a better accuracy score.



Using the **<u>confusion matrix</u>**, we can see that several instances were incorrectly classified into the wrong classes.

Predictive Model: Classification

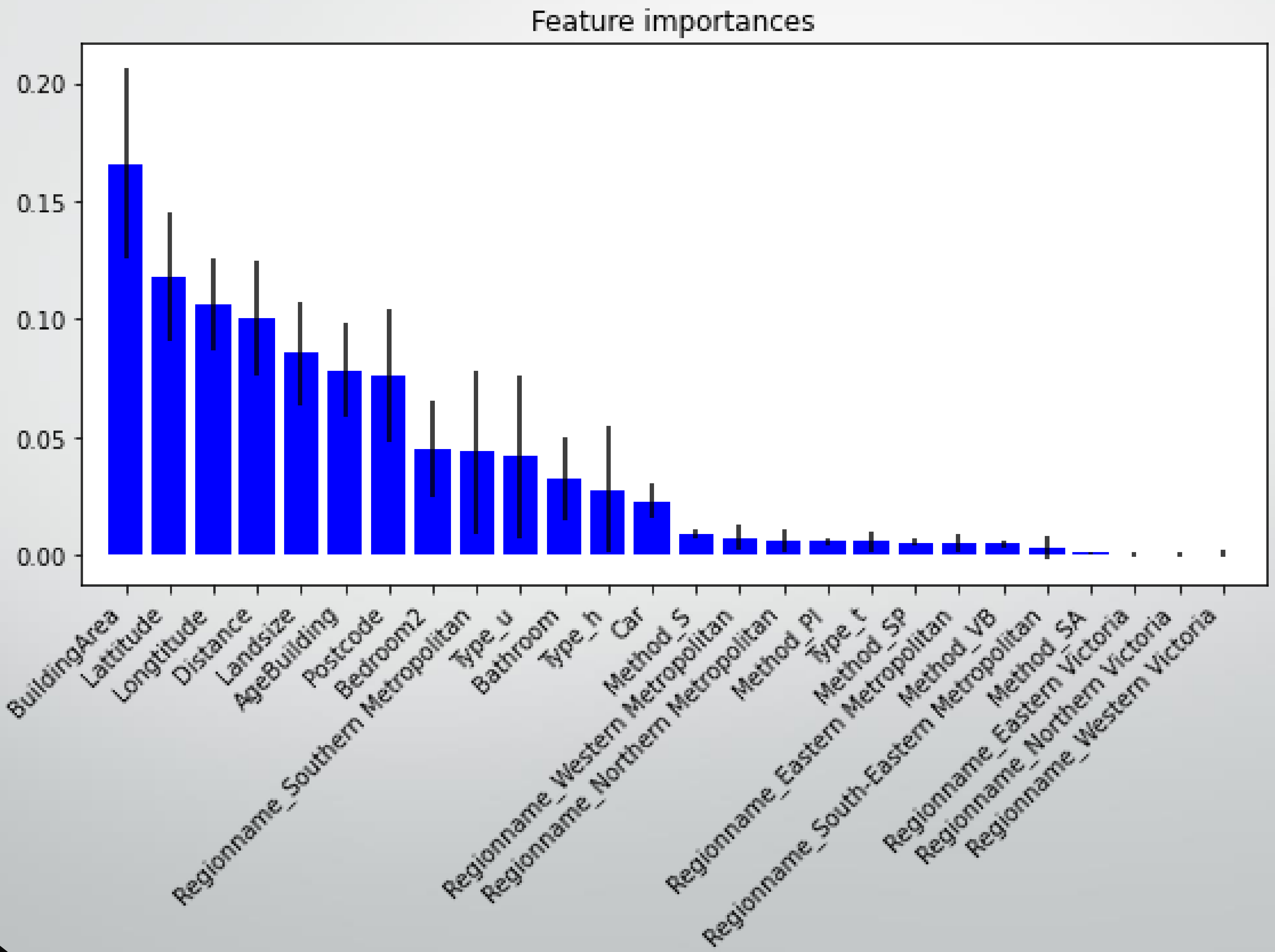
	precision	recall	f1-score	support
High	0.87	0.81	0.84	444
Low	0.86	0.83	0.85	456
Medium	0.83	0.87	0.85	878
accuracy			0.84	1778
macro avg	0.85	0.84	0.84	1778
weighted avg	0.85	0.84	0.84	1778



Using **<u>Random Forest Classification Model</u>**, we achieved an **<u>84% accuracy score</u>**.

We also have fewer misclassified instances.

Predictive Model: Classification



Summary

Summary of Findings

Clustering	**3 clusters**	**4 clusters**
Regression	**Linear** R2 in Training Set: 56.13% R2 in Test Set: 59.97% RMSE:392143	**Random Forest** R2 in Training Set: 97.16% R2 in Test Set: 83.83% MSE: 62127322247
Classification	**Naïve Bayes: 45%**	**Random Forest: 84%**

Thank you!