

EDA for Real Estate Pricing

Unveiling the Dynamics of House Valuation in a Dynamic Market

This presentation will cover the EDA process for Real Estate Pricing using **Python** libraries, including **Pandas**, **NumPy**, **Matplotlib**, and **Seaborn**.

In this project, I used **Pandas** for loading, cleaning, and organizing data. I handled missing values by dropping columns with extensive gaps, like **Alley** & **MasVnrType**, & filling gaps in critical columns, such as **Electrical**, with the **mode**. **NumPy** supported numerical computations, while **Matplotlib** & **Seaborn** helped visualize key patterns and correlations in the data.

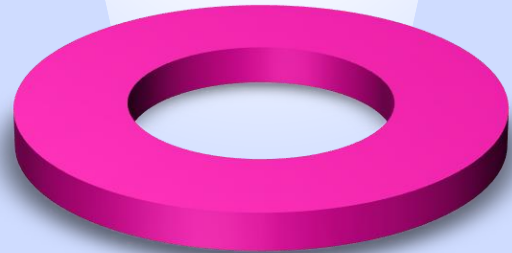
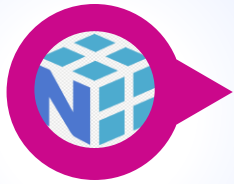
Through **univariate** and **multivariate** analysis, I explored price distributions and identified significant factors like **location**, **lot area**, & **overall quality** that influence prices. **Feature engineering** steps, such as calculating **Total Living Area** & **Price per Square Foot**, enhanced the dataset, revealing deeper insights into real estate trends and laying the groundwork for predictive modelling.



by Debasis Baidya

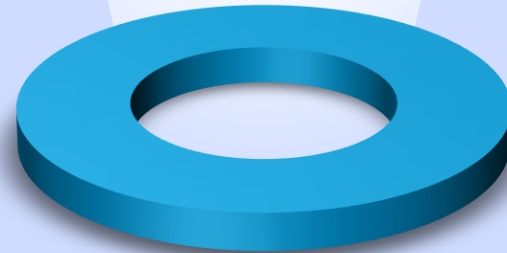
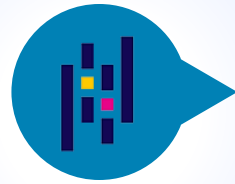


Importing of the necessary Libraries & Modules



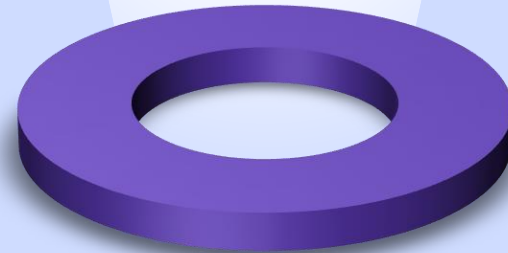
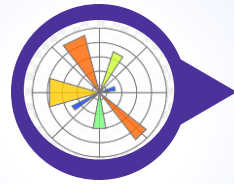
Numpy

It helped with mathematical operations, allowing me to compute statistics and handle missing values efficiently.



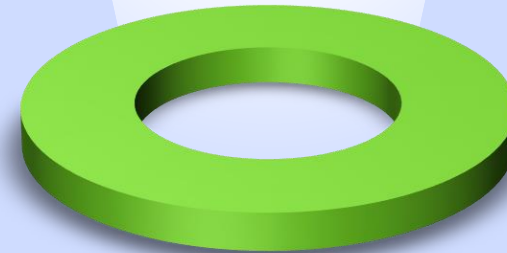
Pandas

To load & organize the datasets, making it easy to explore the data structure and perform operations like merging and filtering.



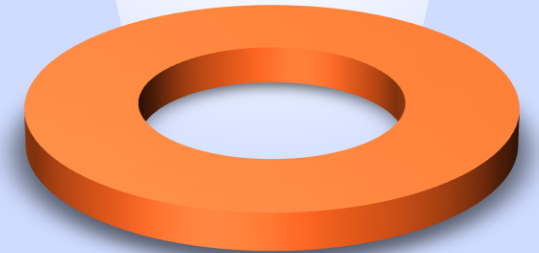
Matplotlib

Used it to create visualizations, such as bar charts and histograms, to better understand the data distributions and trends.



Seaborn

For generating more advanced visualizations, helping me check for skewness & visualize correlations between different variables.



Math Module

Used `math.ceil` to calculate the number of rows needed for subplots, ensuring all features fit in the boxplots by rounding up the division of total features by columns.

Loading Data & Checking Info

```
1 # Step 2: Loading the Data
2 file_path = r"C:\Users\DEB\Downloads\housing_data.csv"
3 housing_data = pd.read_csv(file_path)
4
5 # Display the first few rows
6 housing_data.head()
```

Unnamed: 0	MSSubClass	MSZoning	LotFrontage	LotArea	Street	Alley	
0	0	SC60	RL	65	8450	Pave	NaN
1	1	SC20	RL	80	9600	Pave	NaN
2	2	SC60	RL	68	11250	Pave	NaN
3	3	SC70	RL	60	9550	Pave	NaN
4	4	SC60	RL	84	14260	Pave	NaN

5 rows × 81 columns

```
1 # DataFrame info
2 print("\nDataFrame Info:")
3 housing_data.info()
```

1

Loading Data

Reading the CSV file using pandas.

2

Displaying the Data

Displaying the first few rows using head() method.

3

Checking Data Information

Using info() to view Column Types, data counts, Data Types.

2. Cleaning the Data:

```
1 # DataFrame for missing values
2 missing_values_df = pd.DataFrame({
3     'Column': housing_data.columns,
4     'Missing Values': housing_data.isnull().sum()
5 })
6
7 # Filter to show only columns with missing values
8 missing_values_df = missing_values_df[missing_values_df['Missing Values'] > 0]
9
10 print("\nMissing Values:")
11 print(missing_values_df)
12
13 # Duplicates
14 print(f"\nDuplicates: {housing_data.duplicated().sum()}")
```

Missing Values:

	Column	Missing Values
Alley	Alley	1369
MasVnrType	MasVnrType	872
Electrical	Electrical	1
GarageYrBlt	GarageYrBlt	81

Duplicates: 0

```
1 import warnings
2 warnings.filterwarnings("ignore")
3
4 # Dropping 'Unnamed: 0', 'Alley' and 'MasVnrType' columns
5 housing_data.drop(columns=['Unnamed: 0', 'Alley', 'MasVnrType'], inplace=True)
6
7 # Filling missing values in 'Electrical' with the mode
8 housing_data['Electrical'].fillna(housing_data['Electrical'].mode()[0], inplace=True)
9
10 # Checking for missing values and column names with missing values
11 missing_columns = housing_data.columns[housing_data.isnull().any()]
12
13 # Columns with missing values
14 print("\nColumns with Missing Values:")
15 print(missing_columns)
```

Columns with Missing Values:
Index(['GarageYrBlt'], dtype='object')

Data Cleaning & Preprocessing

Insights on Missing Values Handling:

1. Identified Columns with Missing Values.
2. Actions Taken:
 - Dropped 'Unnamed: 0', 'Alley', and 'MasVnrType' columns due to high missing values.
 - Filled the missing value in 'Electrical' with the most common entry (mode).
3. Current Status:
 - Rechecked for missing values & kept 'GarageYrBlt' as it is needed.

1

Dropping Columns

Removed unnecessary columns like 'Unnamed: 0', 'Alley', & 'MasVnrType'.

2

Filling Missing Values

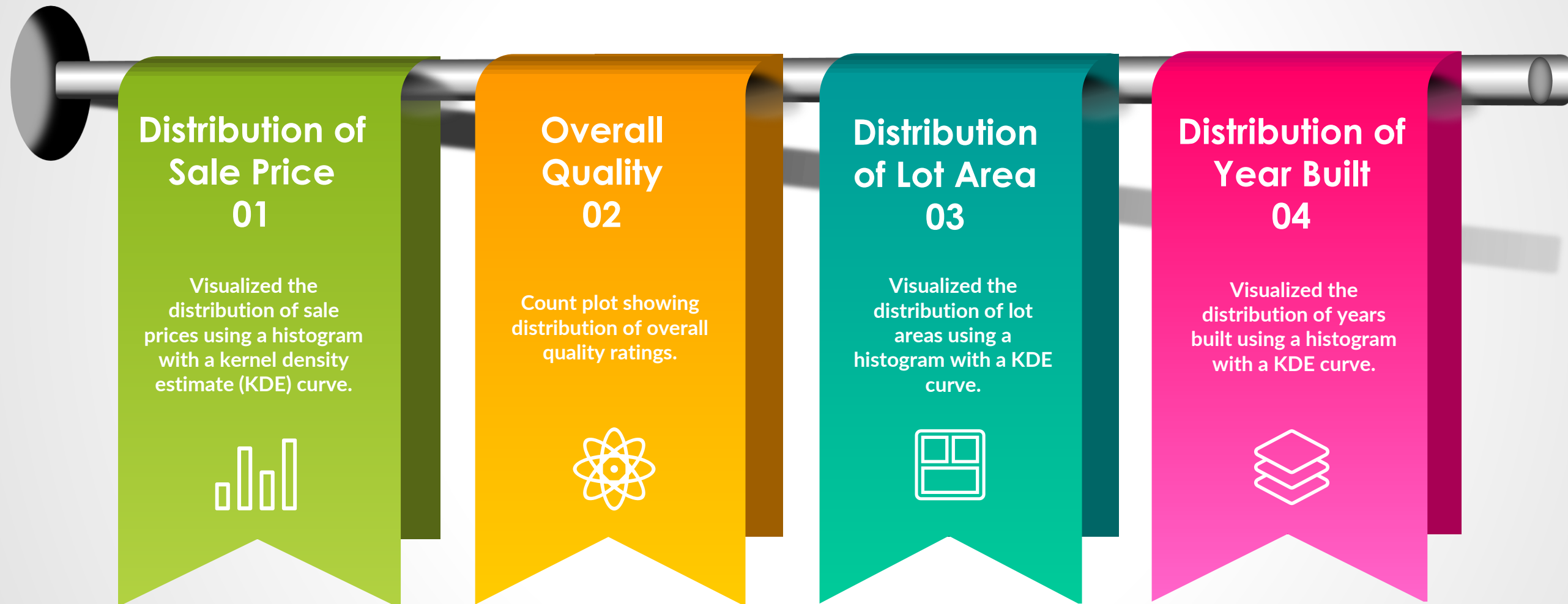
Used **mode** to fill missing values in 'Electrical' column.

3

Checking Remaining Missing Values

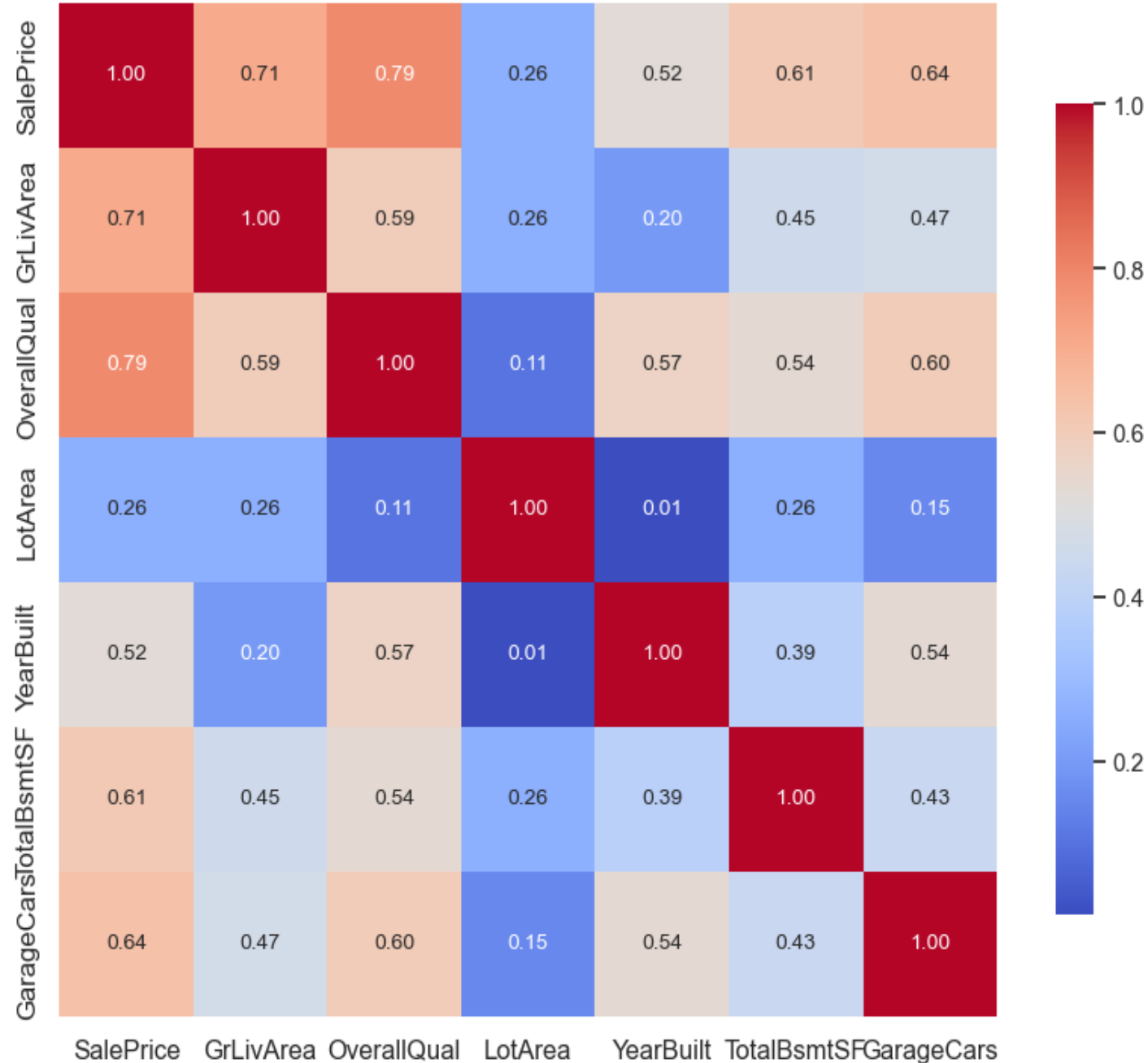
Identify columns still containing null values.

Univariate Analysis



Multivariate Analysis

Correlation Heatmap



```
1 import pandas as pd
2 import matplotlib.pyplot as plt
3 import seaborn as sns
4
5 # Relevant columns for analysis
6 columns_of_interest = ['SalePrice', 'GrLivArea', 'OverallQual', 'LotArea', 'YearBuilt', 'TotalBsmtSF', 'GarageCars']
7 filtered_data = housing_data[columns_of_interest]
8
9 # Correlation matrix
10 correlation_matrix = filtered_data.corr()
11
12 # Heatmap for correlation matrix
13 plt.figure(figsize=(10, 8))
14 sns.heatmap(correlation_matrix, annot=True, fmt=".2f", cmap='coolwarm',
15             annot_kws={"size": 10}, cbar_kws={"shrink": .8})
16 plt.title('Correlation Heatmap', fontsize=16)
17 plt.xticks(fontsize=12)
18 plt.yticks(fontsize=12)
19 plt.show()
```

```
1 # Pairplot for selected variables
2 sns.pairplot(filtered_data, diag_kind='kde')
3 plt.suptitle('Pairplot of Key Variables Impacting House Prices', y=1.02, fontsize=16)
4 plt.show()
```

Feature Engineering

1

Total Living Area

Sum of first floor, second floor, and basement square footage.

2

Total Bathrooms

Combined count of full and half bathrooms, including basement bathrooms.

3

Property Age

Calculated as the difference between the current year (2024) and the year built.

4

High Quality Indicator

Binary indicator (0 or 1) based on whether the overall quality rating exceeds 7.

5

Price per Square Foot

Calculated by dividing the sale price by the living area square footage.

6

Year Built Age

Difference between the current year (2024) and the year built, indicating the age of the property.

Outlier Detection and Handling

1

Identify Outliers

Use Interquartile Range (IQR) method to detect outliers.

2

Cap Outliers

Set upper and lower bounds based on IQR.

3

Replace Extreme Outliers

Use median for features with many remaining outliers.

```
1 # Identifying outliers using IQR
2 def identify_outliers_iqr(df, column):
3     Q1 = df[column].quantile(0.25)
4     Q3 = df[column].quantile(0.75)
5     IQR = Q3 - Q1
6     lower_bound = Q1 - 1.5 * IQR
7     upper_bound = Q3 + 1.5 * IQR
8     return df[(df[column] < lower_bound) | (df[column] > upper_bound)]
9
10 # Count of outliers per feature
11 outlier_counts = {feature: len(identify_outliers_iqr(housing_data, feature)) for feature in numerical_features}
12
13 print("Count of outliers per feature (IQR):")
14 for feature, count in outlier_counts.items():
15     print(f"{feature}: {count}")
```

7. Handling Outliers in the Dataset:

```
1 import pandas as pd
2 import numpy as np
3
4 # Function to count outliers based on IQR
5 def count_outliers_iqr(df, column):
6     Q1 = df[column].quantile(0.25)
7     Q3 = df[column].quantile(0.75)
8     IQR = Q3 - Q1
9     lower_bound = Q1 - 1.5 * IQR
10    upper_bound = Q3 + 1.5 * IQR
11
12    # Count outliers
13    outliers_count = df[(df[column] < lower_bound) | (df[column] > upper_bound)].shape[0]
14
15    return outliers_count
16
17 # Capping outliers
18 def cap_outliers(df, column):
19     Q1 = df[column].quantile(0.25)
20     Q3 = df[column].quantile(0.75)
21     IQR = Q3 - Q1
22     lower_bound = Q1 - 1.5 * IQR
23     upper_bound = Q3 + 1.5 * IQR
24
25    # Cap outliers
26    df[column] = np.clip(df[column], lower_bound, upper_bound)
27    return df
28
29 # Function to replace outliers with median
30 def replace_outliers_with_median(df, column):
31     Q1 = df[column].quantile(0.25)
32     Q3 = df[column].quantile(0.75)
33     IQR = Q3 - Q1
34     lower_bound = Q1 - 1.5 * IQR
35     upper_bound = Q3 + 1.5 * IQR
36     median_value = df[column].median()
37
38    # Replace outliers with median
39    df[column] = np.where((df[column] < lower_bound) | (df[column] > upper_bound), median_value, df[column])
40    return df
```

```
# Handle outliers for all features
for feature in features_to_check:
    cleaned_data = cap_outliers(cleaned_data, feature) # Cap outliers
    outlier_count = count_outliers_iqr(cleaned_data, feature)

    if outlier_count > 5: # If still many outliers, replace with median
        cleaned_data = replace_outliers_with_median(cleaned_data, feature)
```


Feature Engineering and Size Impact

01

Bedrooms vs. Price

Boxplot showing how the number of bedrooms affects sale price.

02

Full Bathrooms vs. Price

Boxplot illustrating the impact of the number of full bathrooms on sale price.

03

Total Living Area vs. Price

Scatter plot depicting the relationship between total living area and sale price.

04

Total Bathrooms vs. Price

Boxplot showing how total bathrooms influence sale price.

05

Property Age vs. Price

Scatter plot indicating how property age affects sale price.

06

Price per Square Foot vs. Price

Scatter plot demonstrating the correlation between price per square foot and sale price.

07

High Quality Indicator vs. Price

Boxplot illustrating the effect of overall quality on sale price.

Time Trends in Housing Prices

1

Grouped by Year

Aggregated sale prices for each year sold.

2

Calculated Averages

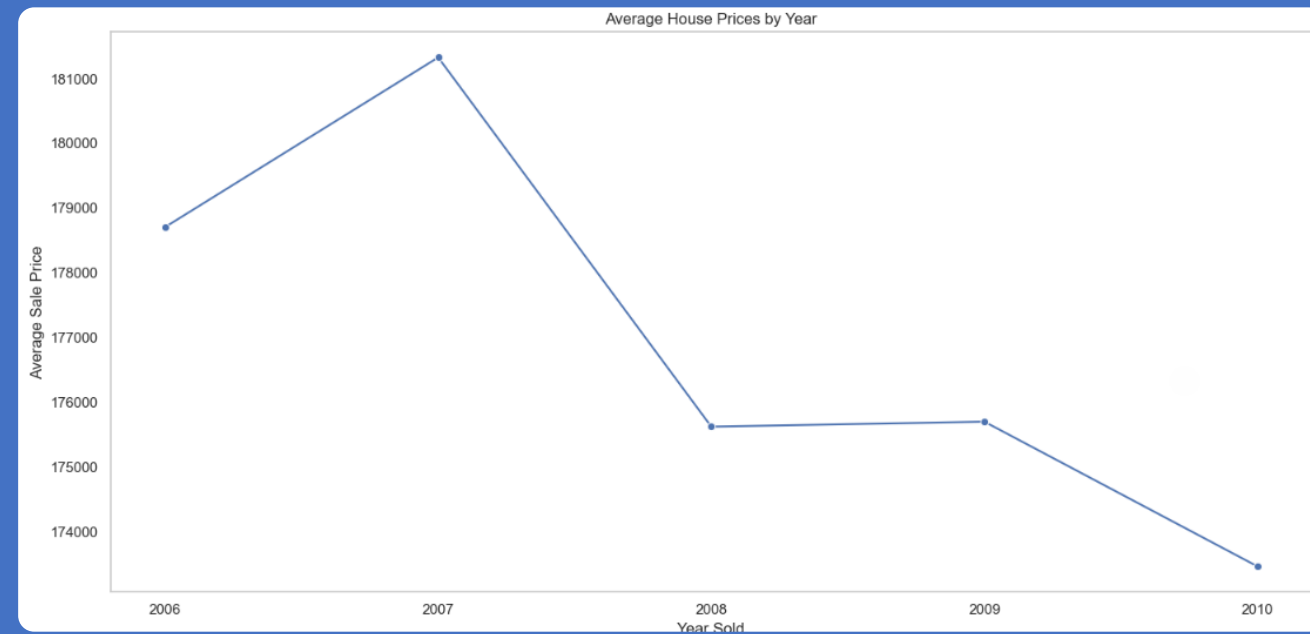
Computed mean sale price per year.

3

Plot Trend

Created line plot showing price trends over time.

```
1 import pandas as pd
2 import matplotlib.pyplot as plt
3 import seaborn as sns
4
5 # Aggregate sale prices by year
6 yearly_prices = cleaned_data.groupby('YrSold')['SalePrice'].mean().reset_index()
7
8 plt.figure(figsize=(14, 7))
9 sns.lineplot(data=yearly_prices, x='YrSold', y='SalePrice', marker='o')
10 plt.title('Average House Prices by Year')
11 plt.xlabel('Year Sold')
12 plt.ylabel('Average Sale Price')
13 plt.xticks(yearly_prices['YrSold'])
14 plt.grid()
15 plt.tight_layout()
16 plt.show()
```



Impact of Amenities on Price



Pool Quality

Analyzed how pool quality affects home prices.



Fireplace Quality

Assessed impact of fireplace quality on home values.



Garage Type

Examine price differences based on garage types.



Basement Quality

Evaluated how basement quality influences prices.



Thank You!

Project Summary

1

- ❑ This project performs an **EDA** on real estate pricing data to identify factors influencing property values. Key steps include data cleaning, missing value handling, and feature engineering, focusing on variables like location, lot size, and quality to prepare the dataset for predictive modelling.

Future Enhancements

3

- ❑ **Advanced Modelling:** Implement regression models to predict prices.
- ❑ **Expanded Feature Engineering:** Include more detailed neighbourhood metrics.

Key Features

2

- ❑ **Data Cleaning:** Removed unnecessary columns and handled missing values (e.g., mode and median imputation).
- ❑ **Feature Engineering:** Added Total Living Area and Price per Square Foot for enhanced insights.
- ❑ **Visual Analysis:** Used plots to reveal price distribution and feature correlations.

Q&A

4

- ❑ I am available to answer any questions you may have. Feel free to reach out to me for any questions or clarifications.