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

```python
# Step 2: Loading the Data
file_path = r"C:\Users\DEB\Downloads\housing_data.csv"
housing_data = pd.read_csv(file_path)

# Display the first few rows
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

```python
# DataFrame info
print("\nDataFrame Info:")
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:

```
# DataFrame for missing values
missing_values_df = pd.DataFrame({
    'Column': housing_data.columns,
    'Missing Values': housing_data.isnull().sum()
})

# Filter to show only columns with missing values
missing_values_df = missing_values_df[missing_values_df['Missing Values'] > 0]

print("\nMissing Values:")
print(missing_values_df)

# Duplicates
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
```

```
import warnings
warnings.filterwarnings("ignore")

# Dropping 'Unnamed: 0', 'Alley' and 'MasVnrType' columns
housing_data.drop(columns=['Unnamed: 0', 'Alley', 'MasVnrType'], inplace=True)

# Filling missing values in 'Electrical' with the mode
housing_data['Electrical'].fillna(housing_data['Electrical'].mode()[0], inplace=True)
```

```
# Checking for missing values and column names with missing values
missing_columns = housing_data.columns[housing_data.isnull().any()]

# Columns with missing values
print("\nColumns with Missing Values:")
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

Distribution of Sale Price
01

Visualized the distribution of sale prices using a histogram with a kernel density estimate (KDE) curve.

Overall Quality
02

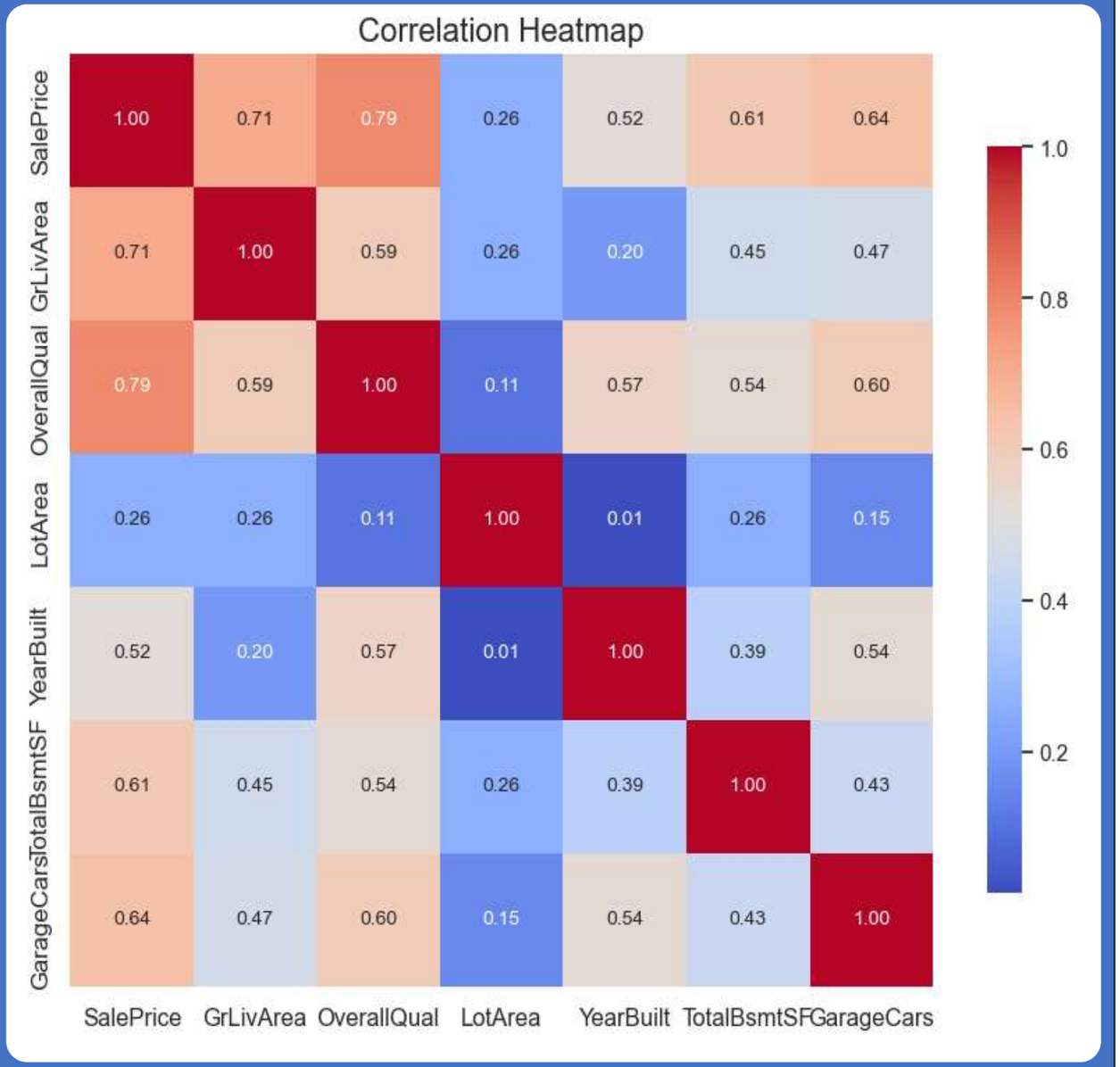
Count plot showing distribution of overall quality ratings.

Distribution of Lot Area
03

Visualized the distribution of lot areas using a histogram with a KDE curve.

Distribution of Year Built
04

Visualized the distribution of years built using a histogram with a KDE curve.



Multivariate Analysis

```python
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# Relevant columns for analysis
columns_of_interest = ['SalePrice', 'GrLivArea', 'OverallQual', 'LotArea', 'YearBuilt', 'TotalBsmtSF', 'GarageCars']
filtered_data = housing_data[columns_of_interest]

# Correlation matrix
correlation_matrix = filtered_data.corr()

# Heatmap for correlation matrix
plt.figure(figsize=(10, 8))
sns.heatmap(correlation_matrix, annot=True, fmt=".2f", cmap='coolwarm',
            annot_kws={"size": 10}, cbar_kws={"shrink": .8})
plt.title('Correlation Heatmap', fontsize=16)
plt.xticks(fontsize=12)
plt.yticks(fontsize=12)
plt.show()
```

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

```python
# Identifying outliers using IQR
def identify_outliers_iqr(df, column):
    Q1 = df[column].quantile(0.25)
    Q3 = df[column].quantile(0.75)
    IQR = Q3 - Q1
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR
    return df[(df[column] < lower_bound) | (df[column] > upper_bound)]

# Count of outliers per feature
outlier_counts = {feature: len(identify_outliers_iqr(housing_data, feature)) for feature in numerical_features}

print("Count of outliers per feature (IQR):")
for feature, count in outlier_counts.items():
    print(f"{feature}: {count}")
```

7. Handling Outliers in the Dataset:

```python
import pandas as pd
import numpy as np

# Function to count outliers based on IQR
def count_outliers_iqr(df, column):
    Q1 = df[column].quantile(0.25)
    Q3 = df[column].quantile(0.75)
    IQR = Q3 - Q1
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR

    # Count outliers
    outliers_count = df[(df[column] < lower_bound) | (df[column] > upper_bound)].shape[0]

    return outliers_count

# Capping outliers
def cap_outliers(df, column):
    Q1 = df[column].quantile(0.25)
    Q3 = df[column].quantile(0.75)
    IQR = Q3 - Q1
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR

    # Cap outliers
    df[column] = np.clip(df[column], lower_bound, upper_bound)
    return df

# Function to replace outliers with median
def replace_outliers_with_median(df, column):
    Q1 = df[column].quantile(0.25)
    Q3 = df[column].quantile(0.75)
    IQR = Q3 - Q1
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR
    median_value = df[column].median()

    # Replace outliers with median
    df[column] = np.where((df[column] < lower_bound) | (df[column] > upper_bound), median_value, df[column])
    return df
```

```python
# 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

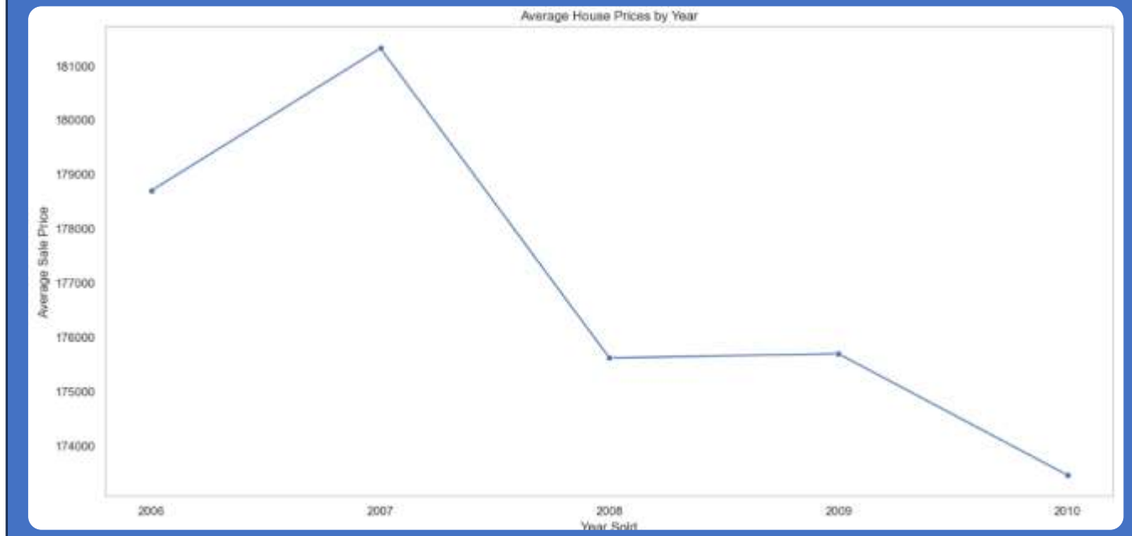
Plot Trend

Created line plot showing price trends over time.

```python
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# Aggregate sale prices by year
yearly_prices = cleaned_data.groupby('YrSold')['SalePrice'].mean().reset_index()

plt.figure(figsize=(14, 7))
sns.lineplot(data=yearly_prices, x='YrSold', y='SalePrice', marker='o')
plt.title('Average House Prices by Year')
plt.xlabel('Year Sold')
plt.ylabel('Average Sale Price')
plt.xticks(yearly_prices['YrSold'])
plt.grid()
plt.tight_layout()
plt.show()
```



Impact of Amenities on Price

Pool Quality

Analyzed how pool quality affects home prices.

Garage Type

Examine price differences based on garage types.

Fireplace Quality

Assessed impact of fireplace quality on home values.

Basement Quality

Evaluated how basement quality influences prices.

Thank You!

1 Project Summary

- 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.

2 Key Features

- **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.

3 Future Enhancements

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

4 Q&A

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