London Housing Dataset Analysis - Complete Documentation

Overview

This Jupyter Notebook analyzes the London Housing Dataset, which contains comprehensive housing market data from 1995 to 2019. The dataset includes:

- **Monthly average house prices** across different areas
- **Yearly number of houses sold**
- **Monthly number of crimes committed**

This analysis is designed for beginners learning data analysis with Python and Pandas, demonstrating fundamental data manipulation and exploration techniques.

Dataset Information

- **Source**: Kaggle
- **Format**: CSV file
- **Time Period**: 1995-2019
- **Geographic Coverage**: London areas
- **File Name**: `Housing_Dataset.csv`

Code Analysis and Documentation

1. Initial Setup and Data Loading

```python
import pandas as pd
```

Purpose: Import the Pandas library for data manipulation and analysis.

```python
data = pd.read_csv(r"Housing_Dataset.csv")
```

Purpose: Load the CSV file into a Pandas DataFrame.

- Uses raw string (`r""`) to handle file paths correctly
- Creates a DataFrame object named `data`

```python
data
```

Purpose: Display the entire dataset to get an overview. **Expected Output**: A tabular view showing all columns and rows with basic DataFrame information.

2. Data Quality Assessment

2.1 Count Non-Null Values

```python
data.count()
```

Purpose: Count non-null (non-missing) values in each column. **Expected Output**: Series showing count of valid entries per column.

Example Output:

```
date              1000
area              1000
average_price      995
code               998
houses_sold        990
no_of_crimes       988
dtype: int64
```

2.2 Identify Missing Values

```python
data.isnull().sum()
```

Purpose: Count null/missing values in each column. **Expected Output**: Series showing count of missing values per column.

Example Output:

```
date              0
area              0
average_price     5
code              2
houses_sold      10
no_of_crimes     12
dtype: int64
```

2.3 Visualize Missing Values

python

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

sns.heatmap(data.isnull())
plt.show()
```

Purpose: Create a heatmap visualization of missing values.

- **White areas**: Missing values
- **Dark areas**: Present values **Expected Output**: A heatmap showing the pattern of missing data across the dataset.

3. Data Type Conversion

3.1 Convert Date Column

python

```python
data.dtypes  # Check current data types
```

Purpose: Display current data types of all columns.

python

```python
data.date = pd.to_datetime(data.date)
```

Purpose: Convert the 'date' column from string to datetime format.

- Enables date-based operations and extractions
- Improves performance for temporal analysis

Verification:

```python
data.dtypes
```

Expected Output: Shows 'date' column as `datetime64[ns]` type.

4. Feature Engineering

4.1 Extract Year Information

```python
data['year'] = data.date.dt.year
```

Purpose: Create a new column containing only the year from the date.

- Uses `.dt.year` accessor to extract year component
- Useful for yearly aggregations and analysis

Expected Output: New 'year' column with integer values (1995-2019).

4.2 Insert Month Column

```python
data.insert(1, 'month', data.date.dt.month)
```

Purpose: Insert a new 'month' column at position 1 (second column).

- `insert(position, column_name, values)`
- Extracts month number (1-12) from date

Expected Output: DataFrame with 'month' as the second column.

4.3 Remove Columns

```python
data.drop(['month', 'year'], axis=1, inplace=True)
```

Purpose: Remove the 'month' and 'year' columns from the DataFrame.

- `axis=1`: Drop columns (not rows)
- `inplace=True`: Modify the original DataFrame

5. Data Analysis Questions

Question D: Records with Zero Crimes

python

```
data[data.no_of_crimes == 0]
```

Purpose: Filter and display all records where the number of crimes is zero. **Expected Output**: Subset of DataFrame containing only rows with zero crimes.

To count these records:

python

```
len(data[data.no_of_crimes == 0])
```

Expected Output: Integer representing the count of zero-crime records.

Question E: Price Analysis for England

python

```
# First, recreate the year column
data['year'] = data.date.dt.year

# Filter for England only
df1 = data[data.area == 'england']

# Calculate yearly statistics
df1.groupby('year').average_price.mean()  # Average price per year
df1.groupby('year').average_price.max()   # Maximum price per year
df1.groupby('year').average_price.min()   # Minimum price per year
```

Purpose: Analyze price trends in England by year. **Expected Output**: Series showing price statistics grouped by year.

Example Output:

```
year
1995    85000.50
1996    87250.25
1997    89500.75
...
2019   245000.00
Name: average_price, dtype: float64
```

Question F: Crime Statistics by Area

python

```python
# Maximum crimes per area
data.groupby('area').no_of_crimes.max()

# Minimum crimes per area (sorted)
data.groupby('area').no_of_crimes.min().sort_values(ascending=False)
```

Purpose: Find the range of crime numbers across different areas. **Expected Output**: Series showing crime statistics per area.

Example Output:

```
area
westminster    1250
camden         1100
tower_hamlets   980
...
richmond        45
Name: no_of_crimes, dtype: int64
```

Question G: Count Records by Price Threshold

python

```python
data[data.average_price < 100000].area.value_counts()
```

Purpose: Count records per area where average price is below £100,000. **Expected Output**: Series showing count of affordable housing records per area.

Example Output:

```
area
barking_and_dagenham    156
newham                  134
croydon                 128
...
kensington_and_chelsea    2
Name: area, dtype: int64
```

Key Learning Points

1. Data Exploration Techniques

- `.count()`: Count non-null values

- `.isnull().sum()`: Count missing values
- `.dtypes`: Check data types
- `.head()`: Preview first few rows

2. Data Visualization

- **Seaborn heatmaps**: Visualize missing data patterns
- **Matplotlib**: Display plots

3. Data Manipulation

- `pd.to_datetime()`: Convert strings to datetime
- `.dt` **accessor**: Extract date components
- `.insert()`: Add columns at specific positions
- `.drop()`: Remove columns or rows

4. Data Analysis

- **Boolean indexing**: Filter data based on conditions
- `.groupby()`: Group data for aggregation
- `.value_counts()`: Count unique values
- `.sort_values()`: Sort data

5. Aggregation Functions

- `.max()`: Maximum values
- `.min()`: Minimum values
- `.mean()`: Average values
- `.sum()`: Sum values

Best Practices Demonstrated

1. **Data Quality Assessment**: Always check for missing values and data types
2. **Feature Engineering**: Create derived columns for better analysis
3. **Filtering**: Use boolean indexing for targeted analysis
4. **Grouping**: Leverage groupby operations for categorical analysis
5. **Visualization**: Use plots to understand data patterns

Common Pitfalls to Avoid

1. **Not checking data types**: Always verify datetime conversions

2. **Ignoring missing values**: Address null values before analysis
3. **Forgetting inplace parameter**: Use `inplace=True` for permanent changes
4. **Incorrect filtering syntax**: Use proper boolean indexing syntax

Extensions and Further Analysis

This notebook provides a foundation for more advanced analysis:

- **Time series analysis**: Trend analysis over time
- **Correlation analysis**: Relationship between price and crime
- **Geographic analysis**: Spatial patterns in housing data
- **Predictive modeling**: Price prediction models
- **Statistical testing**: Hypothesis testing on area differences

Conclusion

This notebook demonstrates fundamental data analysis techniques using a real-world dataset. The step-by-step approach makes it ideal for beginners while covering essential pandas operations that form the foundation of data science workflows.