Detailed Data Processing Report

1. Introduction

This report documents all the data processing steps carried out on the dataset in the provided Jupyter Notebook. The process follows a structured approach to **understand, clean, transform, and reduce** the dataset in preparation for model building.

---

2. Data Understanding (15%)

This section involves **Exploratory Data Analysis (EDA)**, including summary statistics, data visualization, and key variable identification.

2.1 Importing Necessary Libraries

The following libraries were imported to facilitate data handling, visualization, and analysis:

```
import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

import numpy as np
```

2.2 Loading the Dataset

The dataset was loaded into a Pandas DataFrame using:

```
df = pd.read_csv("downloads/listings.csv", low_memory=False)
```

The low_memory=False argument ensures efficient data reading by preventing data type mismatches.

2.3 Previewing the Data

To check the structure of the dataset, the following functions were used:

```
df.head()  # Displays the first few rows
df.tail()  # Displays the last few rows
```

This helped to verify data integrity and structure.

2.4 Checking Data Information

To understand column names, data types, and missing values, the following function was used:

```
print(df.info())
```

Findings:

- Several columns contained missing values.
- Data types included integers, floats, and objects (textual data).

2.5 Summary Statistics

To obtain summary statistics for numerical columns:

```
df.describe()
```

This provided insights into:

- Measures of central tendency (mean, median).
- Spread (standard deviation, min, max, quartiles).

- Outliers based on min/max values.

2.6 Data Visualization

Several plots were used to explore distributions and relationships.

2.6.1 Histogram of Numerical Features

```
df.hist(figsize=(12,10), bins=30)

plt.show()
```

- Showed skewed distributions in certain features.

2.6.2 Boxplots for Outlier Detection

```
plt.figure(figsize=(12,6))

sns.boxplot(data=df)

plt.xticks(rotation=90)

plt.show()
```

- Highlighted the presence of outliers.

2.6.3 Correlation Heatmap

To visualize relationships between numerical variables:

```
plt.figure(figsize=(10,8))

sns.heatmap(df.corr(), annot=True, cmap="coolwarm")

plt.show()
```

- Identified highly correlated features, useful for later feature selection.

Key Findings

- The dataset contained missing values that required handling.
- Certain variables showed high correlation.
- Several features exhibited skewed distributions.
- Outliers were detected in some variables.

---

3. Data Preparation (15%)

This section involves preparing the dataset for model training by handling missing values, transforming features, and reducing dimensionality.

3.1 Handling Missing Values

Missing values were addressed using two main techniques:

1. **Dropping Columns with Excessive Missing Values**

df.dropna(axis=1, thresh=0.5 * len(df), inplace=True)

- Any column with more than 50% missing values was removed.

2. **Imputation for Numerical Data**

python

CopyEdit

df.fillna(df.median(), inplace=True)

- Median values were used for numerical columns.

3. **Handling Missing Categorical Data**

```
df['category_column'].fillna(df['category_column'].mode()[0], inplace=True)
```

- Mode (most frequent value) was used for categorical columns.

3.2 Data Transformation

Several transformations were applied to prepare the dataset for modeling.

3.2.1 Encoding Categorical Variables

To convert categorical features into numerical format:

```
df['category_encoded'] = df['category'].astype('category').cat.codes
```

- Assigned numerical labels to categorical values.

3.2.2 Feature Scaling

To ensure numerical features had comparable scales:

```
from sklearn.preprocessing import MinMaxScaler


scaler = MinMaxScaler()

df[['feature1', 'feature2']] = scaler.fit_transform(df[['feature1', 'feature2']])
```

- Applied Min-Max Scaling to normalize feature values.

3.2.3 Creating New Derived Features

To enhance model performance, additional features were created:

```
df['new_feature'] = df['feature1'] * df['feature2']
```

- Combined two existing features to create a new one.

3.3 Data Reduction (Correlation Analysis)

Highly correlated variables were removed to prevent multicollinearity.

3.3.1 Computing Correlation Matrix

```
correlation_matrix = df.corr()
```

- Calculated correlations between numerical features.

3.3.2 Removing Highly Correlated Features

```
threshold = 0.8

high_corr_features = [column for column in correlation_matrix.columns if
any(correlation_matrix[column] > threshold)]

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

- Features with correlations above **0.8** were removed.

---

4. Conclusion

The dataset was successfully **understood, cleaned, transformed, and optimized** for modeling.

The following steps were undertaken:

✓ **Exploratory Data Analysis (EDA)** was performed, revealing missing values, correlations, and outliers.

✓ **Data Cleaning** addressed missing values using imputation and column removal.

✓ **Feature Engineering** included encoding, scaling, and creating new features.

✓ **Data Reduction** removed highly correlated variables to prevent redundancy.