Exploratory Data Analysis (EDA) - Part 1

# Data Cleaning

## Overview

Data cleaning is a crucial step in the data analysis process, involving the identification and correction of errors and inconsistencies in the dataset. It ensures the accuracy and reliability of the data.

## Steps in Data Cleaning

1. Identify Missing Values:  
- Detect and mark missing data points.  
- Methods: `.isnull()`, `.isna()` in Pandas.  
  
2. Remove Duplicates:  
- Identify and remove duplicate records.  
- Methods: `.drop\_duplicates()` in Pandas.  
  
3. Handle Outliers:  
- Detect and treat outliers to avoid skewed analysis.  
- Methods: Z-score, IQR method.  
  
4. Standardize Data:  
- Ensure consistency in data format and units.  
- Example: Date formats, currency units.  
  
5. Correct Errors:  
- Identify and correct errors in the dataset.  
- Example: Typographical errors, incorrect entries.

## Example

```python  
import pandas as pd  
  
# Load data  
data = pd.read\_csv("data.csv")  
  
# Identify missing values  
missing\_values = data.isnull().sum()  
  
# Remove duplicates  
data\_cleaned = data.drop\_duplicates()  
  
# Handle outliers using Z-score  
from scipy import stats  
data\_cleaned = data\_cleaned[(np.abs(stats.zscore(data\_cleaned)) < 3).all(axis=1)]  
```

# Imputation Techniques

## Overview

Imputation techniques are used to fill in missing data values to maintain the integrity of the dataset. Choosing the right method depends on the nature of the data and the extent of missingness.

## Types of Imputation

### Mean/Median Imputation

Replace missing values with the mean or median of the column.  
Suitable for numerical data.  
Example:  
```python  
data["column"].fillna(data["column"].mean(), inplace=True)  
```

### Mode Imputation

Replace missing values with the mode (most frequent value) of the column.  
Suitable for categorical data.  
Example:  
```python  
data["column"].fillna(data["column"].mode()[0], inplace=True)  
```

### Forward/Backward Fill

Use the previous or next value to fill missing data.  
Suitable for time series data.  
Example:  
```python  
data["column"].fillna(method="ffill", inplace=True)  
data["column"].fillna(method="bfill", inplace=True)  
```

### K-Nearest Neighbors (KNN) Imputation

Use the values of the nearest neighbors to fill missing data.  
Suitable for both numerical and categorical data.  
Example:  
```python  
from sklearn.impute import KNNImputer  
imputer = KNNImputer(n\_neighbors=5)  
data\_imputed = imputer.fit\_transform(data)  
```

# Data Analysis and Visualization

## Scatter Diagram

### Overview

A scatter diagram (scatter plot) is a graphical representation of the relationship between two numerical variables. Each point on the scatter plot represents an observation.

### Interpretation

Positive Correlation: Points trend upward.  
Negative Correlation: Points trend downward.  
No Correlation: Points are randomly scattered.

### Example

```python  
import matplotlib.pyplot as plt  
  
# Scatter plot  
plt.scatter(data["variable1"], data["variable2"])  
plt.xlabel("Variable 1")  
plt.ylabel("Variable 2")  
plt.title("Scatter Plot of Variable 1 vs Variable 2")  
plt.show()  
```

## Correlation Analysis

### Overview

Correlation analysis measures the strength and direction of the relationship between two numerical variables. The correlation coefficient (r) ranges from -1 to 1.

### Interpretation

r = 1: Perfect positive correlation.  
r = -1: Perfect negative correlation.  
r = 0: No correlation.

### Calculation

```python  
# Calculate correlation  
correlation = data["variable1"].corr(data["variable2"])  
print("Correlation Coefficient:", correlation)  
```

## Transformations

### Overview

Transformations are applied to data to meet the assumptions of statistical techniques or to improve the interpretability of the analysis. Common transformations include log, square root, and Box-Cox.

### Types of Transformations

#### Log Transformation

Reduces right skewness.  
Example:  
```python  
data["log\_variable"] = np.log(data["variable"])  
```

#### Square Root Transformation

Reduces right skewness, less strong than log.  
Example:  
```python  
data["sqrt\_variable"] = np.sqrt(data["variable"])  
```

#### Box-Cox Transformation

Transforms non-normal dependent variables into a normal shape.  
Example:  
```python  
from scipy.stats import boxcox  
data["boxcox\_variable"], \_ = boxcox(data["variable"] + 1) # Adding 1 to avoid log(0)  
```

### Example

```python  
import numpy as np  
from scipy.stats import boxcox  
  
# Log transformation  
data["log\_variable"] = np.log(data["variable"])  
  
# Square root transformation  
data["sqrt\_variable"] = np.sqrt(data["variable"])  
  
# Box-Cox transformation  
data["boxcox\_variable"], \_ = boxcox(data["variable"] + 1) # Adding 1 to avoid log(0)  
```