Exploratory Data Analysis (EDA)

---

1. Collection and Data Overview

Understanding the Dataset

- **Dataset Structure**:
    - **Rows** → Represent individual observations.
    - **Columns** → Represent features (variables).
- **Data Types**:
    - **Numerical (Continuous/Discrete)** → Age, Salary, Temperature.
    - **Categorical (Nominal/Ordinal)** → Gender, Rating, Size (S/M/L).
    - **DateTime** → Timestamps, Events over time.

📌 **Implications:**

- Numerical variables allow mathematical operations and statistical modeling.
- Categorical variables require encoding before use in ML models.
- Date-time variables need conversion into meaningful components (year, month, day, etc.).

Checking Dataset Properties in Python

```
import pandas as pd

df = pd.read_csv("data.csv")  # Load dataset
print(df.head())  # View first 5 rows
print(df.shape)  # (rows, columns)
print(df.info())  # Data types and non-null counts
print(df.describe())  # Summary statistics for numerical columns
```

---

2. Handling Dates (Introduction)

- **Date-time columns** often need transformation for analysis.
- Convert a date column to `datetime` type:

```
df['date'] = pd.to_datetime(df['date'])
```

- Extract important components:

```
df['year'] = df['date'].dt.year
df['month'] = df['date'].dt.month
df['day'] = df['date'].dt.day
```

📌 **Why it's important?**

- Helps in **trend analysis**, seasonality detection, and time-based grouping.

---

3. Missing Values Analysis

- **Check missing values**:

```
print(df.isnull().sum())  # Count missing values in each column
print(df.isnull().mean() * 100)  # Percentage of missing values
```

📌 **Why does missing data occur?**

- **Human error**, **data corruption**, or **improper data collection**.

---

4. Imputation (Introduction & Simple Techniques)

Handling missing values depends on **data type and missing percentage**.

- **Drop missing values** (if they are very few):

```
df.dropna(inplace=True)
```

- **Impute with mean/median/mode**:

```
df['age'].fillna(df['age'].mean(), inplace=True)  # For numerical data
df['category'].fillna(df['category'].mode()[0], inplace=True)  # For cate
```

```
gorical data
```

- **Forward/Backward fill (for time series data)**:

```
df.fillna(method='ffill', inplace=True)  # Forward fill
df.fillna(method='bfill', inplace=True)  # Backward fill
```

📌 **Why is imputation important?**

- Missing values **affect statistical calculations and ML models**, so they must be handled carefully.

---

5. Descriptive Statistics and Interpretation

- Used to **summarize and understand the distribution** of data.
- Common functions:

```
print(df.describe())  # Summary statistics (mean, std, min, max, etc.)
print(df.median())  # Median of numerical columns
print(df.var())  # Variance
print(df.skew())  # Skewness (symmetry of data)
print(df.kurt())  # Kurtosis (sharpness of peak)
```

📌 **Key Insights from Statistics:**

- **Mean ≠ Median** → Indicates skewness.
- **High variance** → Data is spread out.
- **High kurtosis** → Data has many extreme values.

---

6. Data Visualization

(a) Univariate Scatter Plots (for outlier detection & distribution)

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

plt.scatter(df.index, df["age"])
```

```
plt.title("Univariate Scatter Plot")
plt.show()
```

📌 **Use case**: Detects **outliers or clusters** in data.

(b) Frequency Plots

Histograms (show distribution of a numerical variable)

```
df["age"].hist(bins=20)
plt.title("Histogram of Age")
plt.xlabel("Age")
plt.ylabel("Frequency")
plt.show()
```

📌 **Use case**: Identifies **data distribution shape and spread**.

Distribution Plots – KDE (Kernel Density Estimation)

```
sns.kdeplot(df["age"], shade=True)
plt.title("KDE Plot of Age")
plt.show()
```

📌 **Use case**: Smoothed version of histogram, useful for seeing **probability distributions**.

(c) Box Plots (Detecting Outliers)

```
sns.boxplot(x=df["age"])
plt.title("Box Plot of Age")
plt.show()
```

📌 **Interpretation**:

- **Median (middle line)** – Central tendency.
- **Box (IQR: Q1 to Q3)** – Spread of data.
- **Whiskers** – Range within 1.5x IQR.
- **Outliers (dots outside whiskers)** – Extreme values.

7. Outliers (Introduction & Importance of Stats to Interpret Plots)

Outliers are values that **significantly differ** from the rest of the data.

📌 **Why are outliers important?**

- They **skew statistical analysis** and **affect ML model performance**.

Detecting Outliers

- **Using Boxplot**
- **Using Z-score**:

```
from scipy import stats
df_outliers = df[(np.abs(stats.zscore(df["age"])) > 3)]
```

- **Using IQR Method**:

```
Q1 = df["age"].quantile(0.25)
Q3 = df["age"].quantile(0.75)
IQR = Q3 - Q1
df_outliers = df[(df["age"] < (Q1 - 1.5 * IQR)) | (df["age"] > (Q3 + 1.5 * IQR))]
```

8. Relevance of Statistics in Interpreting Plots

✅ **Histograms & KDE Plots** → Identify normal/skewed distribution.

✅ **Box Plots** → Identify outliers & data spread.

✅ **Scatter Plots** → Detect unusual clusters or patterns.

✅ **Descriptive Statistics** → Help validate interpretations from visualizations.

Final Summary

Topic	Key Takeaway
Data Overview	Rows, columns, data types, and their implications
Handling Dates	Convert to datetime, extract year/month/day

Topic	Key Takeaway
Missing Values	Detect and handle using imputation
Descriptive Statistics	Mean, median, variance, skewness, kurtosis
Univariate Plots	Histograms, KDE plots, box plots
Outliers	Use box plots, Z-score, and IQR to detect anomalies
Stats & Plots	Visual tools + statistical measures = better data insights