Exploratory Data Analysis (EDA)

Objective: Understand data using statistics and visualizations.

🔍 What Does It Tell Us?

1. Data Distribution

- **Histograms** show how values are spread.
    - e.g., In the Iris dataset, sepal_length might be normally distributed while petal_length could be skewed.
    - ✅ Helps decide if scaling or transformation is needed.

2. Outliers and Anomalies

- **Boxplots** expose outliers.
    - e.g., sepal_width might have a few unusually high or low values.
    - ✅ You may choose to remove or treat these outliers.

3. Feature Relationships

- **Pairplots** or **Scatter plots** show how features relate to each other.
    - e.g., petal_length and petal_width have a strong positive correlation.
    - ✅ Useful for feature engineering or reducing dimensionality.

4. Correlation Between Variables

- **Heatmaps** of correlation matrices tell:
    - Which features are **strongly related** (positive or negative).
    - e.g., If two features have a correlation of ~1, they're redundant.
    - ✅ Helps in **feature selection** (avoid multicollinearity)

5. Balance of Categories

- **Countplots** show distribution of classes.
    - e.g., species column shows equal counts for all classes.
    - ✅ If classes are **imbalanced**, special techniques (e.g., SMOTE, weighted loss) may be needed.

6. Summary Statistics

- **Mean, Median, Std, Mode** describe each column's nature.
    - e.g., A large difference between **mean and median** suggests **skewness**.
    - ✅ Informs whether normalization or log transformation is needed.

7. Data Quality Issues

- Missing values? Duplicates?
    - ✅ You'll know what needs to be **cleaned** or **imputed**.

🧠 **Why This Matters in ML?**

Insight	Impact on ML
Outliers	Can ruin model performance (e.g., regression)
Correlation	Helps in feature selection/reduction
Skewed distributions	Require transformation or different model choices
Class imbalance	Requires balancing techniques
Redundant features	Can be removed to avoid overfitting

Tools: Pandas, Matplotlib, Seaborn, Plotly Hints/Mini Guide:

📦 1. Pandas – Data Handling & Manipulation

Purpose: Load, clean, transform, and analyze data (especially tabular data like CSV/Excel).

🔧 Common Commands:

- import pandas as pd

- **# Load dataset**
- df = pd.read_csv('data.csv')
-
- **# View data**
- df.head() **# First 5 rows**
- df.info() **# Structure of DataFrame**
- df.describe() **# Statistics summary**
- df.columns **# Column names**
- df.shape **# (rows, columns)**

🔁 Data Operations:

- df['column'] **# Access column**
- df[['col1', 'col2']] **# Multiple columns**
- df.loc[0] **# Access row by label**
- df.iloc[0] **# Access row by index**
- df.drop('col', axis=1, inplace=True) **# Drop column**
- df.isnull().sum() **# Missing values**
- df.fillna(0) **# Replace nulls with 0**
- df.duplicated().sum() **# Check duplicates**

📊 2. Matplotlib – Base Visualization Library

Purpose: Customizable static plots (basic but powerful)

🔧 Common Commands:

- import matplotlib.pyplot as plt
- **# Line plot**
- plt.plot([1, 2, 3], [4, 5, 6])
- plt.title('Line Plot')
- plt.xlabel('X')
- plt.ylabel('Y')
- plt.show()
- **# Histogram**
- plt.hist(df['col'])
- plt.show()
- **# Bar chart**
- plt.bar(x=['A', 'B'], height=[10, 20])
- plt.show()
- **# Scatter plot**
- plt.scatter(df['x'], df['y'])
- plt.show()

🎨 3. Seaborn – Statistical Visualization (Built on Matplotlib)

Purpose: Beautiful visualizations with simple syntax, great for exploring patterns.

- import seaborn as sns
- **# Histogram (with KDE)**
- sns.histplot(df['col'], kde=True)
- **# Boxplot**
- sns.boxplot(data=df, x='category', y='value')
- **# Pairplot (multi scatter + KDE)**
- sns.pairplot(df, hue='class')
- # Heatmap
- corr = df.corr()
- sns.heatmap(corr, annot=True)
- **# Countplot (bar for categorical)**
- sns.countplot(data=df, x='category')

📈 4. Plotly – Interactive Charts (Web-ready)

Purpose: Interactive dashboards and plots (great for notebooks or web apps)

- import plotly.express as px
- **# Interactive scatter plot**
- fig = px.scatter(df, x='x', y='y', color='class')
- fig.show()
- **# Interactive bar plot**
- fig = px.bar(df, x='category', y='value')
- fig.show()
- **# Line chart**
- fig = px.line(df, x='date', y='value')
- fig.show()
- **# Pie chart**
- fig = px.pie(df, names='category', values='value')
- fig.show()

💡 When to Use What?

Library	Best For	Pros	Cons
Pandas	Data handling & analysis	Powerful tabular operations	Not for plotting
Matplotlib	Custom plots	Flexible, full control	Verbose syntax
Seaborn	Quick, beautiful stats plots	Easy syntax, great defaults	Less interactive
Plotly	Interactive dashboards/web plots	Interactivity, easy sharing	Slightly heavier

---

1.Generate summary statistics (mean, median, std, etc.).

Let's walk through **how to generate summary statistics** such as **mean**, **median**, **standard deviation (std)**, etc. using Python with **Pandas** — as required in your EDA Task 2.

✅ **Step 1: Import Libraries**

import pandas as pd

✅ **Step 2: Load the Dataset**

Assuming you're using the **Titanic dataset**:

- df = pd.read_csv('titanic.csv')

✅ **Step 3: Generate Summary Statistics**

🔹 **A. Full Summary (Numeric Columns Only)**

- df.describe()

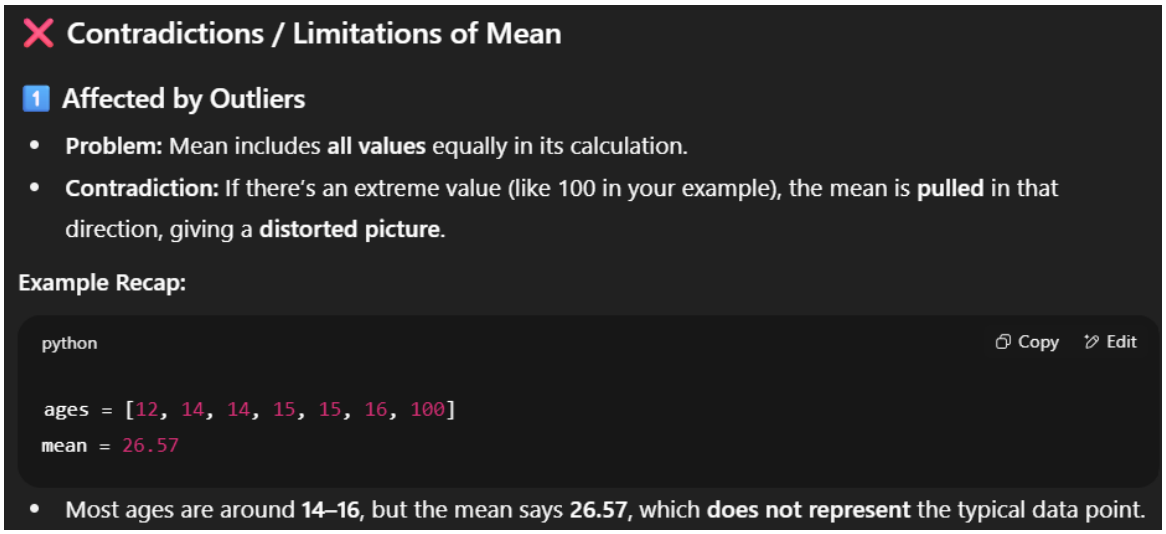
This command gives:

- **count** → number of non-null values
- **mean**
- **std** → standard deviation
- **min**
- **25%, 50% (median), 75%**
- **max**

🔹 B. Mean of All Columns

➢ df.mean(numeric_only=True)

Returns the average value for each numeric column.

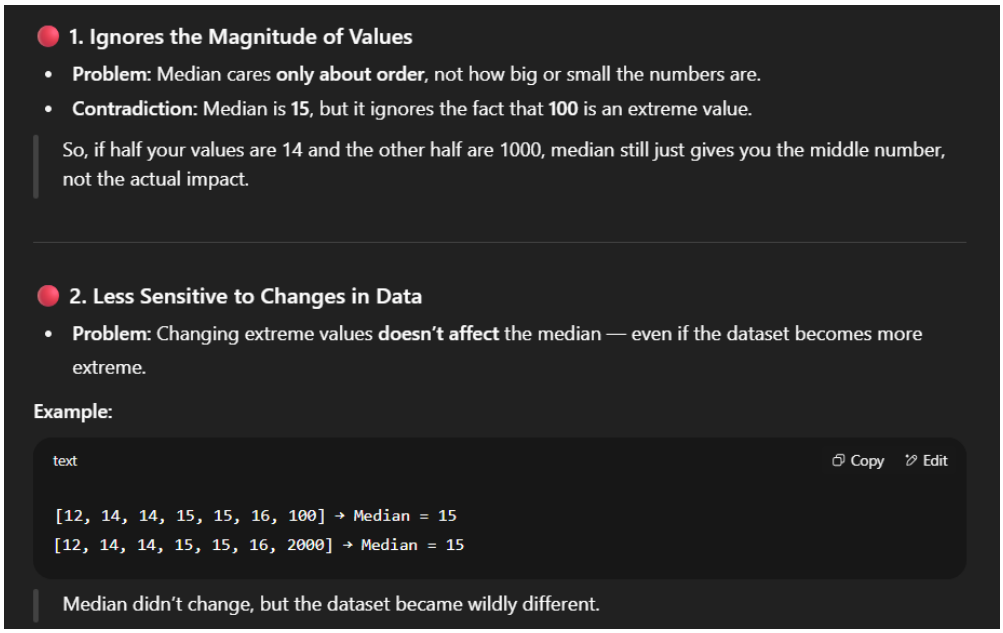


Scenario	Use Instead
Data has outliers	Use **Median**
Data is highly skewed	Use **Median/IQR**
Want central tendency + spread	Use **Boxplot**
Categorical or ordinal data	Use **Mode** or **Counts**

🔹 C. Median of All Columns

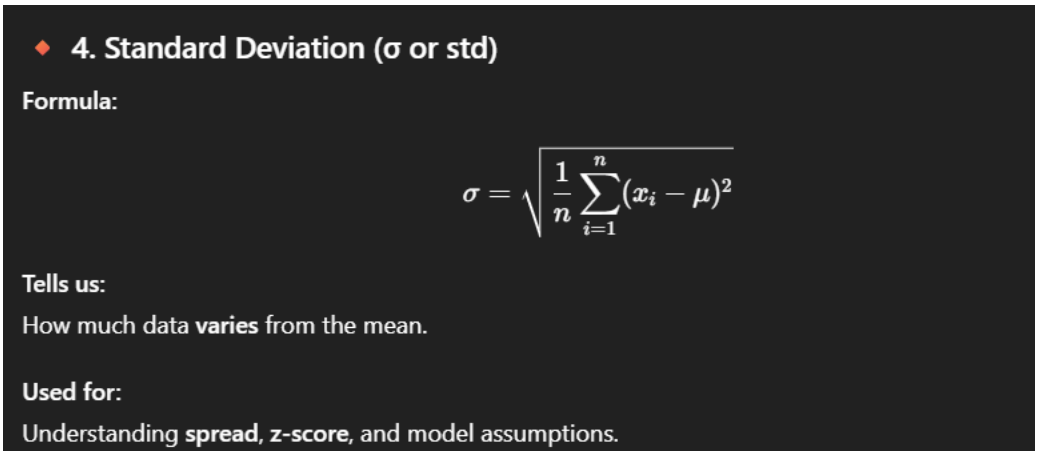
➢ df.median(numeric_only=True)

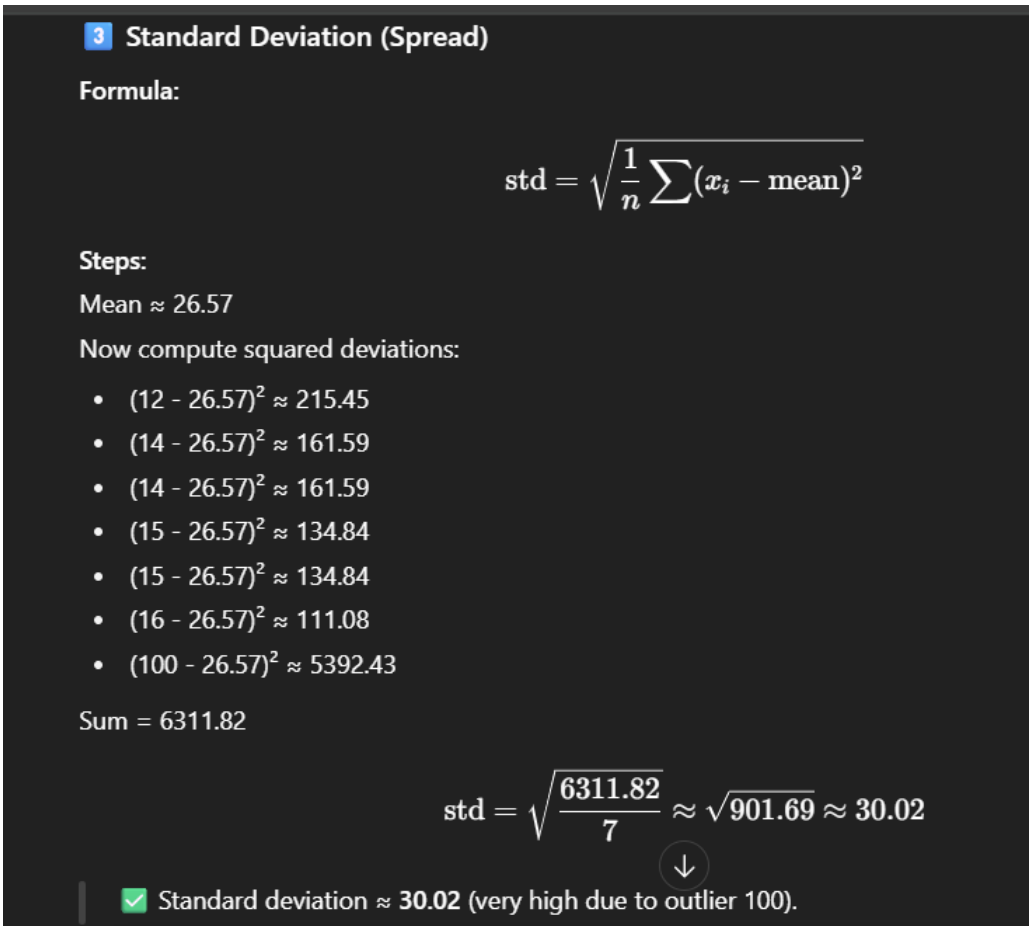
Returns the median (50th percentile) for each numeric column.



- **D. Standard Deviation (std)**
  - df.std(numeric_only=True)

Shows how spread out the values are from the mean.





- **E. Min, Max, Count, Variance**
  - df.min(numeric_only=True) **# Minimum values**
  - df.max(numeric_only=True) **# Maximum values**
  - df.count() **# Non-null counts**
  - df.var(numeric_only=True) **# Variance**

🔹 F. Skewness & Kurtosis

- **Skewness** → Asymmetry
    - **Tells us:**
      Whether the data is **symmetric**, **right-skewed**, or **left-skewed**.
- **Kurtosis** → Peakedness
    - **Tells us:**
      The **peakedness** of the distribution.
        - High kurtosis = sharp peak (leptokurtic)
        - Low kurtosis = flat (platykurtic)
        - Normal kurtosis = 3 (mesokurtic)
    - **Used for:**
      Understanding **tail behavior**.
- df.skew(numeric_only=True)
- df.kurt(numeric_only=True)

4️⃣ Skewness (Asymmetry)

Skewness can be estimated by comparing **mean vs median**:

- If `Mean > Median` → **Right-skewed** (like our example)
- If `Mean < Median` → **Left-skewed**
- If `Mean ≈ Median` → **Symmetrical**

> ✅ Since **Mean (26.57)** > **Median (15)** → Right-skewed due to outlier 100.

🔸 10. Covariance

Formula:

$$\text{Cov}(X,Y) = \frac{1}{n}\sum_{i=1}^{n}(x_i - \mu_X)(y_i - \mu_Y)$$

Tells us:

Direction of linear relationship between **two variables**.

- Positive → both increase
- Negative → one increases, other decreases

🔸 11. Correlation (Pearson's r)

Formula:

$$r = \frac{\text{Cov}(X, Y)}{\sigma_X \cdot \sigma_Y}$$

Tells us:

Strength & direction of the linear relationship between variables.

- r = 1 → perfect positive
- r = -1 → perfect negative
- r = 0 → no correlation

Used for:

Feature selection, multicollinearity detection.

6️⃣ Interquartile Range (IQR)

Steps:

Sorted data: `[12, 14, 14, 15, 15, 16, 100]`

- Q1 (25th percentile): between 2nd and 3rd = 14
- Q3 (75th percentile): between 5th and 6th = 15.5

$$\text{IQR} = Q3 - Q1 = 15.5 - 14 = 1.5$$

Outlier threshold:

$$\text{Upper} = Q3 + 1.5 \times IQR = 15.5 + 2.25 = 17.75$$

$$\text{Lower} = Q1 - 1.5 \times IQR = 14 - 2.25 = 11.75$$

> ✅ Any value < **11.75** or > **17.75** is an outlier. → So **100** is an outlier.

2.Create histograms and boxplots for numeric features.

📊 Histogram (for Distribution)

Purpose: Shows the frequency distribution of a single numeric feature. Helps detect skewness, modality (unimodal, bimodal), and spread.

python Copy Edit

```python
# Histogram of Age
plt.figure(figsize=(8, 5))
plt.hist(df['Age'].dropna(), bins=30, edgecolor='black')
plt.title('Histogram of Age')
plt.xlabel('Age')
plt.ylabel('Frequency')
plt.grid(True)
plt.show()
```

✅ **Tells us:**

- How age values are distributed (e.g., more people in 20–40 age range)
- Skewness in distribution
- Presence of peaks or gaps

📦 Boxplot (for Outliers & Spread)

Purpose: Shows median, quartiles, and outliers in a numeric feature.

python Copy Edit

```python
# Boxplot of Age
plt.figure(figsize=(8, 4))
sns.boxplot(x=df['Age'])
plt.title('Boxplot of Age')
plt.xlabel('Age')
plt.grid(True)
plt.show()
```

✅ **Tells us:**

- Central tendency (median)
- Spread (Interquartile Range - IQR)
- Outliers (points beyond whiskers)
- Symmetry or skewness in data

3.Use pairplot/correlation matrix for feature relationships.

🔶 1️⃣ Pairplot (from Seaborn)

📌 What it is:

- A grid of scatter plots for **each pair of numeric features**
- Shows **distribution** on the diagonal
- Great for **small to medium-sized datasets**

💡 Why use it:

- Visualize pairwise relationships
- Spot clusters, outliers, and correlations
- Understand class separation (with hue)

✅ Code Example:

python Copy Edit

```python
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.datasets import load_iris
import pandas as pd

# Load sample data
iris = sns.load_dataset("iris")

# Create pairplot
sns.pairplot(iris, hue="species")  # `hue` colors points by class
plt.show()
```

🧠 What it shows:

- Relationship between sepal/petal length & width
- How species are distributed
- If some features are good at classifying data

2️⃣ Correlation Matrix

📌 What it is:

A square table showing **correlation coefficients (r)** between features.

Value of **r ranges from -1 to 1**:

- **+1 = perfect positive correlation**
- **-1 = perfect negative correlation**
- **0 = no linear correlation**

✅ Code Example:

python Copy Edit

```python
# Correlation matrix
correlation_matrix = iris.corr(numeric_only=True)

# Heatmap for visualization
plt.figure(figsize=(8, 6))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', linewidths=0.5)
plt.title("Correlation Matrix")
plt.show()
```

🧠 What it tells you:

- Which features are **strongly correlated**
- Which are **redundant** (e.g., high correlation → you may drop one)
- Relationships useful for **feature selection**

4.Identify patterns, trends, or anomalies in the data.

🎯 Objective:

Understand hidden structures or outliers that could influence modeling decisions.

🔍 1️⃣ Patterns

These are **regularities** or **recurring behaviors** in your dataset.

Pattern Type	Example
Seasonal	Sales go up in December
Class-based	Older passengers survived less (Titanic)
Distribution	Most values cluster around a mean (normal distribution)

🛠 **How to Detect:**

- **Histograms / KDE plots** (for distribution)
- **GroupBy + Mean/Count** (to compare categories)
- **Line plots** (for time series)

➢ titanic.groupby("Pclass")["Survived"].mean()

📈 2️⃣ **Trends**

These show **directional movement** in the data.

✅ **Examples:**

- Increase in salary with experience
- Decrease in disease cases after vaccination

🛠 **How to Detect:**

- **Line plots** (for continuous/time data)
- **Regression plots** (Seaborn's regplot or lmplot)

➢ sns.lmplot(x="Age", y="Fare", data=titanic)

⚠️ 3️⃣ **Anomalies (Outliers)**

Anomalies are **data points that deviate significantly** from the rest of the dataset.

✅ **Examples:**

- A fare of $5000 on Titanic
- A 2-year-old with a salary of ₹1 crore

🛠 **How to Detect:**

- **Boxplots** (for numerical data)
- **Z-score or IQR** methods
- **Scatter plots** (for pairwise anomalies)

➢ sns.boxplot(x=titanic["Fare"])
➢ Or:
➢ from scipy.stats import zscore
➢ titanic["z_fare"] = zscore(titanic["Fare"])
➢ titanic[titanic["z_fare"].abs() > 3]

📌 **Summary:**

Tool	Purpose
Histograms	Detect distribution patterns
Boxplots	Spot outliers
Line/Reg Plots	Observe trends
Correlation	Reveal relationships
GroupBy	Compare patterns across groups

5.Make basic feature-level inferences from visuals.

To **make basic feature-level inferences from visuals**, you're interpreting what plots tell you about individual features (columns) or relationships between them.

🎯 **Objective:**

Use **plots** to draw **meaningful insights** that guide:

- Data cleaning
- Feature engineering
- Model selection

✅ **Common Visuals & What You Can Infer:**

1️⃣ **Histogram / KDE Plot**

Purpose: Understand the distribution of a single feature.

- sns.histplot(titanic['Age'], kde=True)

Inferences You Can Make:

- Is the feature normally distributed?
- Are there many missing or 0 values?
- Are there extreme outliers (skewed tails)?

✅ *E.g., Age is right-skewed — many young passengers, few very old.*

2️⃣ **Boxplot**

Purpose: Spot outliers and compare distributions across categories.

- sns.boxplot(x="Pclass", y="Fare", data=titanic)

Inferences:

- Are there **outliers** in Fare?
- Do 1st class passengers generally pay more?
- Are medians or ranges different between classes?

✅ *E.g., Median Fare is highest for Pclass=1. Many outliers in Fare.*

3️⃣ Scatter Plot / Pairplot

Purpose: Examine relationships between numerical features.

- sns.pairplot(titanic[['Age', 'Fare', 'Pclass']], hue='Pclass')

Inferences:

- Linear/nonlinear relationships?
- Are there visible clusters or class separations?

✅ *E.g., Higher class (Pclass=1) tends to have higher Fare and older Age.*

4️⃣ Countplot (for categorical data)

- sns.countplot(x='Survived', data=titanic)

Inferences:

- Class imbalance (e.g., more people died than survived)
- Dominant categories

✅ *E.g., Majority didn't survive on Titanic.*

5️⃣ Correlation Heatmap

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

Inferences:

- Strong or weak correlations
- Multicollinearity (for feature selection)

✅ *E.g., Fare and Pclass are negatively correlated.*

🧠 **General Inference Strategy:**

Visual	Ask Yourself...
Histogram	Is the data skewed? Are there natural bins or clusters?
Boxplot	Are there any outliers? Do groups differ significantly?
Scatter	Are features related? Any visible patterns or trends?
Countplot	Which categories dominate? Class imbalance?
Heatmap	Which features are correlated? Any redundant variables?