## Lesson Content for Day 3: Data Preprocessing and Visualization

---

### Topic 1: Data Preprocessing

1. **What is Data Preprocessing?**
    - A critical step in the data analysis pipeline that prepares raw data for analysis or model training.
    - Ensures the data is clean, consistent, and suitable for further analysis.
2. **Steps in Data Preprocessing**:
    - **Handling Missing Values**:
        - Replace missing values with the mean, median, or mode.
        - Drop rows or columns with excessive missing values.
        - Use advanced techniques like predictive imputation.
    - **Removing Outliers**:
        - Use statistical methods such as Z-score or the IQR (Interquartile Range) method.
    - **Normalization**:
        - Scales data to a fixed range, typically [0, 1].
    - **Standardization**:
        - Transforms data to have a mean of 0 and a standard deviation of 1.

---

### Topic 2: Data Visualization

1. **Importance of Visualization**:
    - Helps identify trends, patterns, and anomalies in the data.
    - Aids in understanding relationships between variables.
2. **Common Visualization Techniques**:
    - **Histogram**: Displays the distribution of a single variable.
    - **Scatter Plot**: Shows relationships between two continuous variables.
    - **Box Plot**: Highlights the spread of data and outliers.
3. **Popular Visualization Libraries**:
    - **Matplotlib**:
        - A flexible library for creating static, animated, and interactive visualizations.
    - **Seaborn**:
        - Built on Matplotlib, provides a high-level interface for drawing attractive and informative statistical graphics.

---

Python Libraries categorized by usage:

Box Plot

### Box Plot: Explanation & Use Cases

A **Box Plot** (also known as a **Box-and-Whisker Plot**) is a graphical representation of the distribution of a dataset. It helps visualize:

- **Central tendency** (median)
- **Spread** (interquartile range)
- **Outliers**
- **Skewness**

---

# 1. Components of a Box Plot

A box plot consists of five key components:

1. **Minimum (Q0)**: The lowest value excluding outliers.
2. **First Quartile (Q1)**: The 25th percentile (25% of data is below this value).
3. **Median (Q2)**: The 50th percentile (middle value of the dataset).
4. **Third Quartile (Q3)**: The 75th percentile (75% of data is below this value).
5. **Maximum (Q4)**: The highest value excluding outliers.

👉 **Interquartile Range (IQR) = Q3 - Q1**
👉 **Whiskers** extend to 1.5 × IQR beyond Q1 and Q3.
👉 **Outliers** are values that fall outside this range.

---

# 2. Interpretation of Box Plot

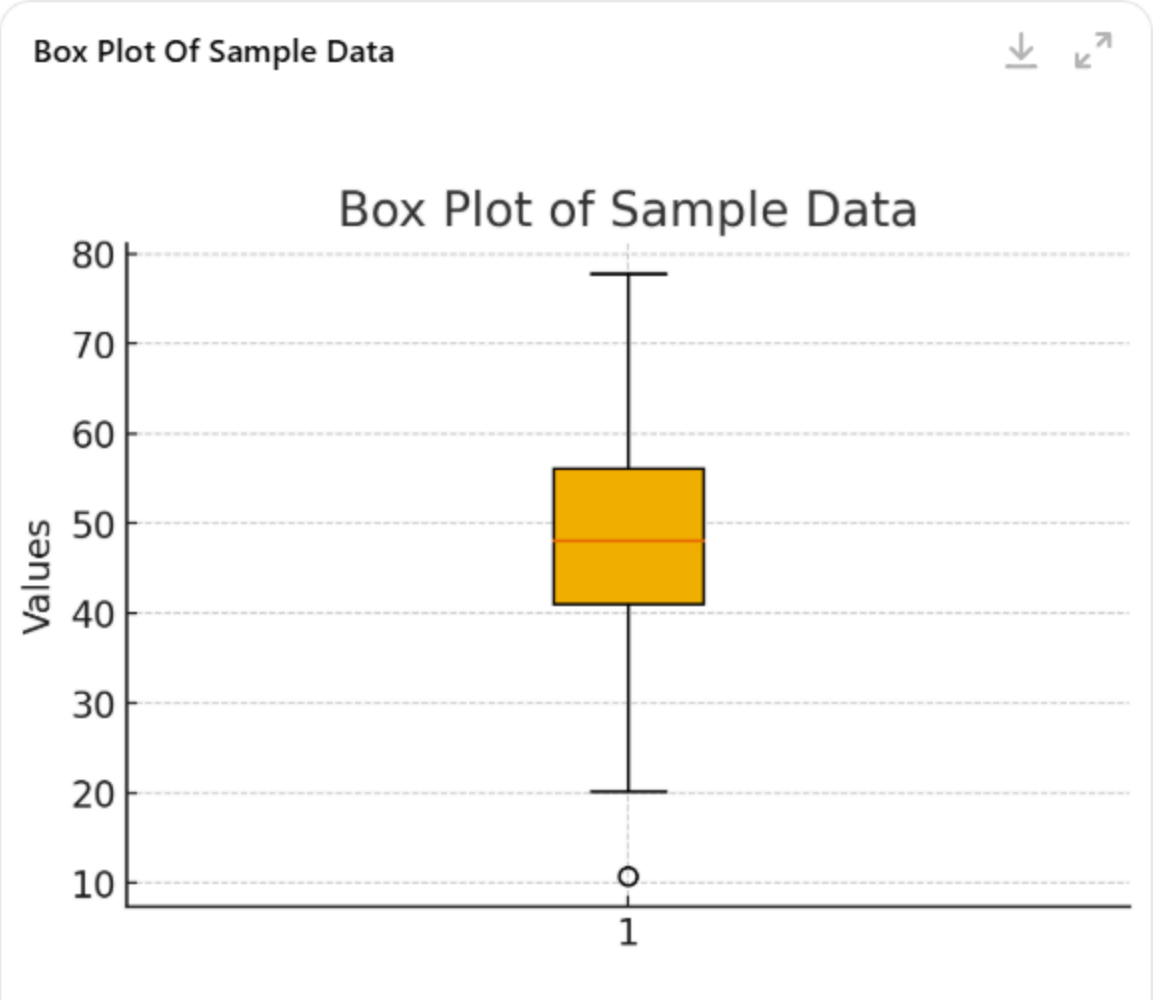
- **If the median is centered:** Data is **symmetrical**.
- **If the median is closer to Q1:** Data is **right-skewed (positive skew)**.
- **If the median is closer to Q3:** Data is **left-skewed (negative skew)**.
- **If whiskers are unequal:** Data has **variability**.
- **If there are many outliers:** Data has **extreme values**.

---

# 3. Example: Box Plot in Python

I'll generate a sample dataset and display a **Box Plot** using Python. Let me generate the plot for you.

Here is the **Box Plot** for a sample dataset.




### Interpretation:

- The **box** represents the **interquartile range (IQR)** (middle 50% of data).
- The **horizontal line inside the box** represents the **median** (50th percentile).
- The **whiskers** extend up to 1.5 × IQR.
- Any **points outside the whiskers** are **outliers**.

Here's a categorization of Python libraries based on their usage:

---

### 1. Data Science & Machine Learning

- **Data Manipulation & Analysis**: `pandas`, `numpy`, `polars`, `dask`
- **Machine Learning**: `scikit-learn`, `xgboost`, `lightgbm`, `catboost`

- **Deep Learning**: `tensorflow`, `keras`, `pytorch`, `jax`
- **Natural Language Processing (NLP)**: `nltk`, `spacy`, `transformers`
- **Computer Vision**: `opencv`, `PIL (Pillow)`, `imageio`, `torchvision`
- **Data Visualization**: `matplotlib`, `seaborn`, `plotly`, `bokeh`

---

### 2. Big Data & Distributed Computing

- **Big Data Processing**: `pyspark`, `dask`, `modin`
- **Parallel Computing**: `ray`, `joblib`, `multiprocessing`
- **Message Queues**: `kafka-python`, `celery`, `pika (RabbitMQ)`

---

### 3. Web Development

- **Web Frameworks**: `django`, `flask`, `fastapi`, `bottle`
- **API Development**: `fastapi`, `flask-restful`, `connexion`
- **Web Scraping**: `beautifulsoup4`, `scrapy`, `selenium`, `playwright`
- **Authentication & Security**: `authlib`, `pyjwt`, `cryptography`
- **Template Engines**: `jinja2`, `mako`

---

### 4. DevOps, Automation & Cloud

- **Infrastructure as Code (IaC)**: `terraform`, `pulumi`, `boto3 (AWS)`, `google-cloud-sdk`
- **Automation**: `ansible`, `fabric`, `paramiko`
- **CI/CD & Testing**: `pytest`, `unittest`, `tox`, `selenium`
- **Containerization & Orchestration**: `docker-py`, `kubernetes`

---

### 5. Database & Storage

- **SQL Databases**: `sqlalchemy`, `sqlite3`, `pg8000`, `psycopg2`
- **NoSQL Databases**: `pymongo (MongoDB)`, `cassandra-driver`, `redis-py`
- **ORMs**: `sqlalchemy`, `django ORM`, `tortoise-orm`
- **File Handling**: `h5py`, `pyarrow`, `fastparquet`

---

## 6. Networking & APIs

- **HTTP Clients**: `requests`, `httpx`, `urllib3`
- **WebSockets**: `websockets`, `socket.io`
- **GraphQL**: `gql`, `ariadne`, `graphene`
- **Email Handling**: `smtplib`, `email`

---

## 7. Cybersecurity & Cryptography

- **Cryptographic Libraries**: `cryptography`, `pycryptodome`, `hashlib`
- **Penetration Testing**: `scapy`, `requests-toolbelt`, `paramiko`
- **Password Hashing**: `bcrypt`, `argon2`

---

## 8. General Purpose Utilities

- **Command-line Tools**: `click`, `argparse`, `typer`
- **Logging & Debugging**: `logging`, `loguru`, `debugpy`
- **Configuration Management**: `configparser`, `dynaconf`, `toml`
- **Datetime & Timezones**: `datetime`, `pytz`, `arrow`
- **Unit Conversion & Math**: `sympy`, `decimal`, `math`

---

Would you like me to refine any specific category based on your needs?

### Hands-On Activity: Titanic Dataset

**Objective**: Learn to preprocess and visualize data.

- **Steps**:
    1. **Load the Dataset**:
        - Use the Titanic dataset (can be downloaded from Kaggle).
    2. **Clean the Data**:
        - Impute missing values in the `Age` column with the median.
        - Drop rows where the `Cabin` column is missing.
    3. **Normalize and Standardize**:
        - Normalize the `Fare` column.
    4. **Visualize**:
        - Create a histogram for the `Age` column.

- Use a scatter plot to show the relationship between `Fare` and `Age`.

**Sample Code**:

python

Copy code

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

# Load Titanic dataset
df = pd.read_csv('titanic.csv')

# Handle missing values
df['Age'].fillna(df['Age'].median(), inplace=True)
df.drop(columns=['Cabin'], inplace=True)

# Normalize Fare
df['Fare_Normalized'] = (df['Fare'] - df['Fare'].min()) / (df['Fare'].max() - df['Fare'].min())

# Visualization
plt.figure(figsize=(10, 6))
sns.histplot(df['Age'], kde=True, bins=30)
plt.title('Age Distribution')
plt.show()

plt.figure(figsize=(10, 6))
sns.scatterplot(x='Age', y='Fare', data=df)
plt.title('Scatter Plot of Age vs. Fare')
plt.show()
```

---

**Reference Materials**

- **Pandas Documentation**: Pandas User Guide
- **Matplotlib Documentation**: Matplotlib User Guide
- **Seaborn Documentation**: Seaborn User Guide