✅ Data Wrangling & Preprocessing

Data wrangling (or preprocessing) means **cleaning and preparing raw data** so that it becomes suitable for analysis or machine learning.
Real-world data is often incomplete, inconsistent, or not structured well — this step **fixes all those issues.**

🔹 1. Handling Missing Values

What are missing values?

Missing values are **empty cells** or **null values** in a dataset where data is not available (e.g., a person's age is blank).

Why it's important?

Machine learning models can't handle missing values — you must fix them first.

Techniques to Handle Them:

- **Remove rows or columns**:
  - If only a few values are missing, remove them.
  - df.dropna() removes rows with missing values.
- **Fill with a constant value**:
  - Replace with 0, "Unknown", or any fixed number.
  - df.fillna(0) or df['city'].fillna("Unknown")
- **Imputation (Statistical filling)**:
  - **Mean/Median/Mode Imputation**:

    df['age'].fillna(df['age'].mean(), inplace=True)
  - **Forward Fill / Backward Fill** (used in time series):

    df.fillna(method='ffill') # forward

    df.fillna(method='bfill') # backward

Example:

If someone's age is missing, fill it with the **average age** of the dataset.

🔹 2. Data Transformation

Data transformation changes the format, structure, or values of data to **make it more meaningful** or to **improve model performance**.

Common Types of Transformations:

✅ **Encoding Categorical Data:**

- Converting words (like "Male", "Female") into numbers.
- Techniques:
  - **Label Encoding**: Male = 0, Female = 1
  - **One-Hot Encoding**: Creates separate columns for each category

✅ **Log Transformation:**

- Used to **reduce skewness** in highly spread data.
- Example: Use np.log(income) if income has large variations.

✅ **Feature Extraction:**

- Get new values from existing ones.
- Example: From a date column, extract "year", "month", "day".

✅ **Binning (or Discretization):**

- **Convert continuous data into categories or intervals**.
- Example: Convert age (a number) into groups:

Python

```
bins = [0, 18, 35, 60, 100]

labels = ['Teen', 'Young', 'Adult', 'Senior']

df['age_group'] = pd.cut(df['age'], bins=bins, labels=labels)
```

- Useful when you want to simplify numerical data.

🔹 3. Type Conversion (Data Type Casting)

Real datasets often contain **wrong or inconsistent data types**, so we must convert them.

Examples:

- **String to Integer**:

python

```
df['salary'] = df['salary'].astype(int)
```

- **String to DateTime**:

python

df['join_date'] = pd.to_datetime(df['join_date'])

- **Float to Integer**, or vice versa:

python

df['price'] = df['price'].astype(float)

Why it matters?

- Models only work with **numeric or datetime types**.
- Wrong types can lead to **errors** in calculations and training.

🔹 4. Feature Engineering

Creating new **meaningful features** from raw data that help machine learning models perform better.

Examples:

- **From Date column**:
  - Extract Year, Month, Day, Weekday.
- **BMI** from weight and height:

python

df['BMI'] = df['weight'] / (df['height'] ** 2)

- **Full Name** from First and Last Name:

python

df['full_name'] = df['first_name'] + " " + df['last_name']

- **Text features**:
  - Count number of words, characters, hashtags, etc.

Good feature engineering can **improve model accuracy** even more than using a complex algorithm.

🔹 5. Scaling

Scaling means bringing all features to a **similar numeric range** so that **no one feature dominates**.

Why is it needed?

Some models (like KNN, SVM, Gradient Descent-based models) are sensitive to large numbers.

Techniques:

✅ **Min-Max Scaling:**

- Converts values to a range between **0 and 1**.
- Formula:

x'=x−xminxmax−xminx' = \frac{x - x_{min}}{x_{max} - x_{min}}x'=xmax−xminx−xmin

- Code:

python

CopyEdit

```
from sklearn.preprocessing import MinMaxScaler

scaler = MinMaxScaler()

X_scaled = scaler.fit_transform(X)
```

✅ **Standardization (Z-score Scaling):**

- Centers data with **mean = 0** and **standard deviation = 1**.
- Formula:

x'=x−μσx' = \frac{x - \mu}{\sigma}x'=σx−μ

- Code:

python

CopyEdit

```
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()

X_scaled = scaler.fit_transform(X)
```

🔹 6. Normalization

Normalization is the process of **scaling individual rows (not columns)** so that the **magnitude of each row vector becomes 1**.

When used?

- In **deep learning**, text processing, or when using **distance-based algorithms**.

Code Example:

python

CopyEdit

from sklearn.preprocessing import Normalizer

normalizer = Normalizer()

X_normalized = normalizer.fit_transform(X)

Example:

If a row is [3, 4], after normalization it becomes:

[35,45]\left[\frac{3}{5}, \frac{4}{5}\right][53,54]

Where 5 is the length (square root of $3^2 + 4^2 = 5$)

📌 Final Summary Table

Topic	Description	Techniques/Examples
Handling Missing Values	Fix null data	dropna(), fillna(), mean, forward-fill
Data Transformation	Modify data shape or scale	Encoding, log transform, binning
Binning	Convert numbers into categories	pd.cut() with bins and labels
Type Conversion	Change datatype to correct one	astype(), pd.to_datetime()
Feature Engineering	Create new features from existing ones	BMI, full name, year from date
Scaling	Make features lie in similar range	Min-Max, Standardization
Normalization	Adjust rows to unit length	Normalizer() from sklearn

✅ 2. Exploratory Data Analysis (EDA)

EDA is the process of **examining and understanding the data before building a model**.
It helps you answer questions like:

- What's the data about?
- Are there any patterns or trends?
- Are there missing values or outliers?
- Are variables related to each other?

It often involves **graphs, summary statistics, and comparisons**.

🔹 1. Descriptive Statistics

Descriptive statistics are numbers that **summarize the basic features** of your data.

Common Measures:

Statistic	Meaning	Example Command (Pandas)
Mean	Average value	df['age'].mean()
Median	Middle value	df['age'].median()
Mode	Most frequent value	df['gender'].mode()
Min / Max	Smallest / Largest value	df['salary'].min()
Standard Deviation (std)	Spread from the mean	df['age'].std()
Count	Total number of entries	df['name'].count()

Quick Summary:

python

CopyEdit

```
df.describe()
```

This gives you **mean, std, min, max, 25%, 50%, 75% values** for numeric columns.

🔹 2. Data Distributions & Outliers

📌 Data Distribution:

This refers to **how data values are spread** across a variable.
Visual tools like **histograms** and **boxplots** help you understand the shape of the distribution.

Types of Distributions:

- **Normal Distribution** (bell curve)
- **Skewed Right** (positively skewed)
- **Skewed Left** (negatively skewed)
- **Uniform Distribution**

python

CopyEdit

```
import seaborn as sns
```

sns.histplot(df['salary'])

📌 **Outliers:**

Outliers are values that are **much higher or lower** than the rest of the data.
They can affect the mean and model accuracy.

Detection Methods:

- **Boxplot**:

python

CopyEdit

sns.boxplot(df['age'])

  - Anything beyond the "whiskers" is an outlier.

- **Z-score** or **IQR (Interquartile Range)** methods.

Handling Outliers:

- Remove them (if they are errors).
- Cap them (set a maximum/minimum).
- Log transform to reduce their effect.

🔹 **3. Correlations**

Correlation shows **how two numeric variables move together**.

- If one increases and the other increases: **Positive Correlation**
- If one increases and the other decreases: **Negative Correlation**
- No pattern: **Zero/No Correlation**

Correlation Coefficient (Pearson's r):

- Ranges from **-1 to +1**
  - +1: Strong positive
  - 0: No correlation
  - -1: Strong negative

python

CopyEdit

df.corr() # Gives correlation matrix

sns.heatmap(df.corr(), annot=True)

Used to check which features are related, e.g., **"height" and "weight"** might have strong positive correlation.

🔹 4. Univariate, Bivariate & Multivariate Analysis

These describe how many **variables** you're analyzing at a time.

✅ Univariate Analysis (One variable)

- Focus: **Distribution** of a single variable
- Used for: Finding outliers, shape, summary
- Graphs:
  - Histogram
  - Boxplot
  - Pie chart (for categorical)

python

CopyEdit

```
sns.histplot(df['age'])
```

✅ Bivariate Analysis (Two variables)

- Focus: Relationship between two variables
- Use when checking if **one variable affects another**
- Graphs:
  - Scatter plot (numeric vs numeric)
  - Bar plot (categorical vs numeric)
  - Correlation heatmap

python

CopyEdit

```
sns.scatterplot(x='age', y='salary', data=df)
```

✅ Multivariate Analysis (3+ variables)

- Focus: Relationship among **multiple variables**
- Helps in identifying **complex interactions**
- Graphs:
  - Pairplot
  - Heatmaps
  - Grouped boxplots
  - 3D plots (for advanced use)

python

CopyEdit

```python
sns.pairplot(df[['age', 'salary', 'experience']])
```

📝 Summary Table

Analysis Type	Variables	Purpose	Tools/Plots
Descriptive Stats	1	Summary of data	describe(), mean(), std()
Distribution	1	Shape and outliers	Histogram, Boxplot
Correlation	2	Relationship between numeric features	corr(), Heatmap
Univariate	1	Distribution of a single variable	Histogram, Pie Chart
Bivariate	2	Relation between two variables	Scatterplot, Barplot
Multivariate	3+	Patterns between multiple variables	Pairplot, Heatmap, Grouped Plots

✅ 3. Introduction to Machine Learning

📌 What is Machine Learning (ML)?

Machine Learning is a branch of Artificial Intelligence (AI) that allows computers to **learn from data** and **make decisions or predictions** without being explicitly programmed.

For example, a spam filter in your email learns from past messages to automatically detect spam in the future.

🔷 **Types of Learning in Machine Learning**

🔹 **1. Supervised Learning**

In supervised learning, we **train the model using labeled data** — which means both the input and the correct output (answer) are provided.

Example:

If you give a dataset of **student study hours (input)** and their **exam scores (output)**, the model learns the relationship.

After training, you can give it a new number of study hours, and it will **predict the expected score**.

Supervised Learning has two main types:

✅ **a. Classification**

- **Used when the output is a category (label).**
- Predicts a **class**, like Yes/No, Spam/Not Spam, Disease/No Disease.

Examples:

- Email → Spam or Not Spam
- Image → Cat or Dog
- Student → Pass or Fail

Algorithms:

- Logistic Regression
- Decision Tree
- Random Forest
- K-Nearest Neighbors (KNN)
- Support Vector Machine (SVM)

✅ **b. Regression**

- **Used when the output is a continuous number**.
- Predicts a **real value**, like price, temperature, salary.

Examples:

- Predicting house prices
- Forecasting sales revenue
- Estimating exam scores based on hours studied

Algorithms:

- Linear Regression
- Polynomial Regression
- Decision Trees (for regression)
- Random Forest Regressor

🔹 2. Unsupervised Learning

In unsupervised learning, we **don't provide any labels/output** — the model just tries to **find patterns or structure** in the input data.

Example:

You give a bunch of customer purchase data, and the model automatically groups similar customers (e.g., by buying behavior).

✅ Clustering (Main type of Unsupervised Learning)

- Clustering is the process of **grouping similar data points together**.
- The model tries to divide the data into **clusters** without knowing the correct answer.

Examples:

- Grouping customers by interests
- Finding different types of users on a website
- Grouping similar news articles

Algorithms:

- K-Means Clustering
- Hierarchical Clustering
- DBSCAN

🔷 Important Concepts

🔹 **Bias vs. Variance**

These two are **sources of error** in machine learning models.

✅ **Bias:**

- Error due to **too simple model**.
- Model cannot capture patterns properly.
- Leads to **underfitting**.

Think of a straight line trying to fit a curve.

✅ **Variance:**

- Error due to **too complex model**.
- Model captures noise from training data and fails on new data.
- Leads to **overfitting**.

Think of a wiggly line that perfectly fits training data but fails on test data.

🟰 **Goal:**

You need to find a **balance** between bias and variance for a good model.

🔹 **Inference vs. Prediction**

These are two **goals** of machine learning models.

✅ **Inference:**

- **Understanding the relationship** between input and output.
- Example: How much does age affect salary?
- Focus is on **interpreting** the model.

✅ **Prediction:**

- **Using the model to predict** outcomes for new data.
- Example: Predict tomorrow's temperature.
- Focus is on **accuracy and generalization**.

Linear regression is good for **inference**.
Random forest is better for **prediction**.

📌 Summary Table

Concept	Description	Example
Supervised Learning	Model learns from input-output pairs	Predict score from study hours
Classification	Output is a category (label)	Spam or Not Spam
Regression	Output is a continuous number	Predict house price
Unsupervised Learning	No labels, find hidden patterns	Group similar customers
Clustering	Group similar data points	Segment users into clusters
Bias	Error from too simple model	Straight line underfitting curve
Variance	Error from too complex model	Overfit curve on noise
Inference	Understand relationships	How salary changes with experience
Prediction	Predict future values	Predict next month's sales