

# Machine Learning for Robotics

**Shafayetul Islam**

Robotics and AI Instructor

## Contact Me



Email: [shafatsib@gmail.com](mailto:shafatsib@gmail.com)



Phone: +880178-0404749



Dhaka, Bangladesh

[Google Scholar](#) [Github](#) [Linkedin](#) [Youtube](#)



# **Class 03-**

## **- ML Pipeline**

# | Machine Learning (ML) Pipeline

The **Machine Learning (ML) Pipeline** is the **step-by-step process** used to build, train, evaluate, and deploy an ML model in a **structured and repeatable way**.

Problem → Data → Features → Model → Evaluation → Deployment

# | Machine Learning (ML) Pipeline

## 1. Problem

**Define what you want to predict and why**

Includes:

- Problem definition
- Task type
  - Regression (house price)
  - Classification (spam / not spam)
  - Estimation (robot position)
- Success criteria (metrics)

Key questions:

- What is the input?
- What is the output?
- How will success be measured?

# Machine Learning (ML) Pipeline

## 2. Data

Collect, understand, and clean raw data

Data Collection	Data Understanding & EDA	Data Cleaning
<ul style="list-style-type: none"><li>• CSV / Excel</li><li>• Sensors</li><li>• Databases</li><li>• APIs</li><li>• Logs</li></ul>	<ul style="list-style-type: none"><li>• <code>df.head()</code></li><li>• <code>df.info()</code></li><li>• <code>df.describe()</code></li><li>• Missing values</li><li>• Outliers</li><li>• Class imbalance</li></ul>	<ul style="list-style-type: none"><li>• Handle missing values</li><li>• Remove duplicates</li><li>• Fix incorrect entries</li><li>• Drop irrelevant columns</li></ul>

Goal: turn **raw data** → **clean data**

# Machine Learning (ML) Pipeline

## 3. Features

Convert clean data into meaningful numeric inputs

<b>Feature Engineering</b> <ul style="list-style-type: none"><li>● Create new features</li><li>● Encode categorical variables</li><li>● Scale / normalize values</li></ul>	<b>Train / Test Split</b> <ul style="list-style-type: none"><li>● Separate features (<b>X</b>) and labels (<b>y</b>)</li><li>● Split data to ensure fair evaluation</li></ul>
--	---

Goal: produce **model-ready features**

# Machine Learning (ML) Pipeline

## 4. Model

**Select and train the learning algorithm**

- |  |  |
|--|--|
| <ul style="list-style-type: none"><li>● Model selection<ul style="list-style-type: none"><li>○ Linear / Logistic Regression</li><li>○ Decision Tree</li><li>○ Random Forest</li><li>○ Neural Network</li></ul></li></ul> | <ul style="list-style-type: none"><li>● Model training</li></ul> |
|--|--|

Goal: learn patterns from features

# | Machine Learning (ML) Pipeline

## 5.1 Evaluation

Measure how good the model really is

- |  |  |
|--|--|
| <ul style="list-style-type: none"><li>● Performance metrics<ul style="list-style-type: none"><li>○ Accuracy</li><li>○ Precision / Recall</li><li>○ F1-score</li><li>○ MSE / RMSE</li></ul></li></ul> | <ul style="list-style-type: none"><li>● Model tuning<ul style="list-style-type: none"><li>○ Hyperparameters</li><li>○ Feature selection</li><li>○ Cross-validation</li></ul></li></ul> |
|--|--|

Goal: ensure **generalization**, not memorization

# | Machine Learning (ML) Pipeline

## 6. Deployment

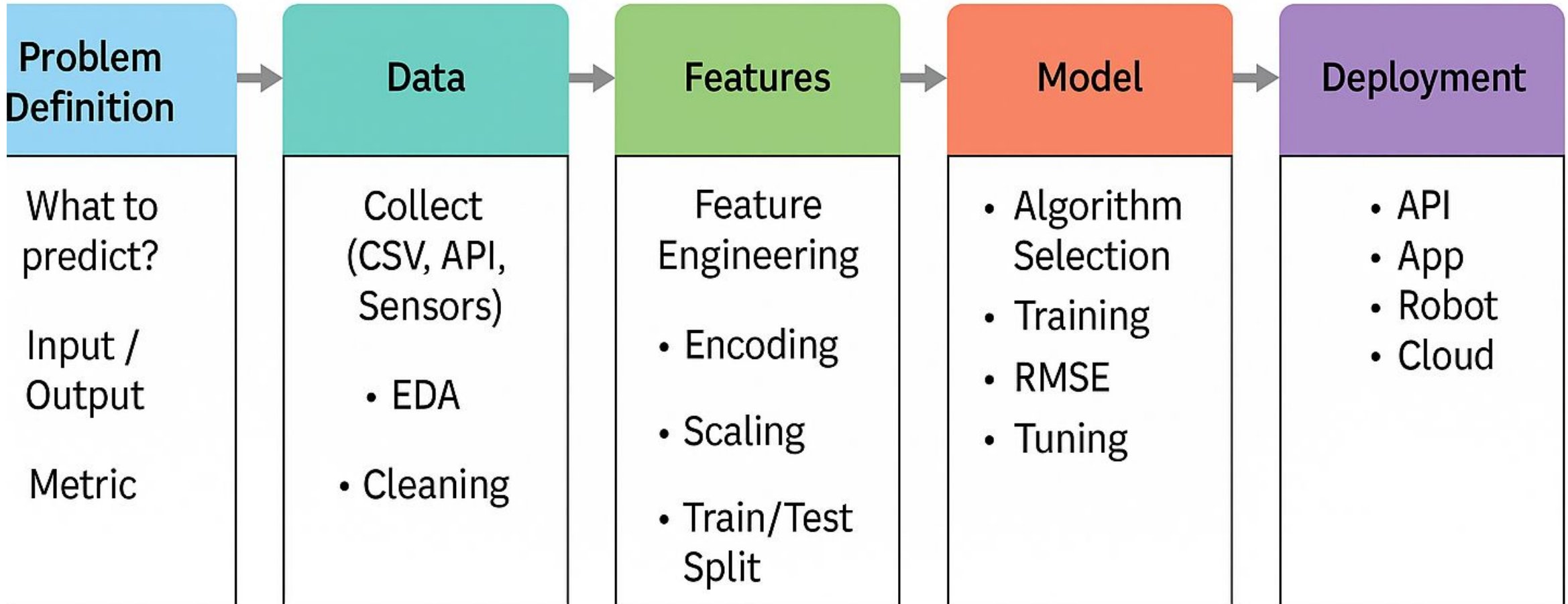
**Use the trained model in real applications**

Includes:

- Inference on new data
- Integration with:
  - Web API
  - Mobile app
  - Robot system
  - Cloud service

**Goal: deliver real-world value**

# Machine Learning (ML) Pipeline





# **Class 03-** **- Pandas**

# | Pandas

Pandas are used in Machine Learning because they provide powerful tools for data loading, cleaning, preprocessing, exploration, and feature engineering, which form the foundation of any successful ML pipeline.

Example of few Operations:

- Data loading
- Structured data representation
- Data cleaning
- Data preprocessing & feature engineering
- Easy data exploration

# | 0. Importing Pandas

```
import pandas as pd
```

# | Series in pandas

A Series in pandas is a one-dimensional labeled array used to store a single column of data (or a single row).

## Creating a Series

```
import pandas as pd
```

```
s = pd.Series([22, 25, 30, 35, 40])  
print(s)
```

index	age
0	22
1	25
2	30
3	35
4	40

**Series**

# | Series in pandas

## Creating a Series with custom index

```
import pandas as pd

s = pd.Series([22, 25, 30, 35, 40],
              index=["a", "b", "c", "d", "e"])
print(s)
```

index	age
a	22
b	25
c	30
d	35
e	40

**Series**

# | Dataframe in Pandas

A DataFrame (df) is a two-dimensional labeled data structure in pandas, similar to a table, used to store and manipulate structured data in Machine Learning.

age	salary	experience	label
22	30000	0	0
25	40000	2	0
30	60000	5	1
35	80000	8	1
40	90000	10	1

**df**

# | Dataframe in Pandas

```
import pandas as pd
```

```
data = [  
    {"age": 22, "salary": 30000, "experience": 0, "label": 0},  
    {"age": 25, "salary": 40000, "experience": 2, "label": 0},  
]
```

```
df = pd.DataFrame(data)  
print(df)
```

age	salary	experience	label
22	30000	0	0
25	40000	2	0
30	60000	5	1
35	80000	8	1
40	90000	10	1

# | Dataframe Indexing

## 1) Column indexing

**Single column → Series**

```
df["salary"]
```

**Multiple columns → DataFrame**

```
df[["age", "salary"]]
```

age	salary	experience	label
22	30000	0	0
25	40000	2	0
30	60000	5	1
35	80000	8	1
40	90000	10	1

# | Dataframe Indexing

## 2) Row indexing by position: **iloc**

Use integer positions (0-based).

```
df.iloc[0]           # first row
df.iloc[0:2]         # rows 0 and 1
df.iloc[0, 1]        # row 0, column 1 (salary here)
df.iloc[:, 1]        # all rows, column 1
```

age	salary	experience	label
22	30000	0	0
25	40000	2	0
30	60000	5	1
35	80000	8	1
40	90000	10	1

# | Dataframe Indexing

## 3) Row indexing by label: **loc**

```
df.loc[0]          # row with index label 0
df.loc[0:2]        # includes both ends (0,1,2) if index is sorted
df.loc[:, "salary"] # all rows, salary column
df.loc[df["salary"] > 35000, ["age", "salary"]] # filter +select columns
```

## Key difference:

- **iloc** slicing end is **excluded** ( $0:2 \rightarrow 0,1$ )
- **loc** slicing end is often **included** ( $0:2 \rightarrow 0,1,2$ )

age	salary	experience	label
22	30000	0	0
25	40000	2	0
30	60000	5	1
35	80000	8	1
40	90000	10	1

# | Dataframe Indexing

## 4) Conditional indexing (filtering)

```
df[df["salary"] > 50000]  
df[(df["salary"] > 30000) & (df["age"] >= 25)]
```

## 5) Quick ML pattern: select X and y

```
X = df[["age", "salary"]]  
y = df["label"]
```

## 6) Setting values (editing)

```
df.loc[df["salary"] < 35000, "label"] = 0
```

age	salary	experience	label
22	30000	0	0
25	40000	2	0
30	60000	5	1
35	80000	8	1
40	90000	10	1

# Dataframe Indexing

- `df["col"]` → one column
- `df[["c1", "c2"]]` → multiple columns
- `df.iloc[row, col]` → by position
- `df.loc[row_label, col_name]` → by label/name
- `df[condition]` → filtering rows

age	salary	experience	label
22	30000	0	0
25	40000	2	0
30	60000	5	1
35	80000	8	1
40	90000	10	1

# | 1. Load dataset

```
df = pd.read_csv("data.csv")
```

```
url = 'https://example.com/data.csv'  
df_url = pd.read_csv(url)
```

```
df_excel = pd.read_excel('path/to/your/file.xlsx',  
sheet_name='Sheet1')
```

```
df_json = pd.read_json('path/to/your/file.json')
```

```
df_html_list =  
pd.read_html('en.wikipedia.org_(programming_language)')
```

## | 2. View and understand the data

```
df.head()          # first 5 rows
df.tail()          # last 5 rows
df.shape           # (rows, columns)
df.columns         # feature names
df.info()          # data types + missing values
```

### 3. Basic statistics (EDA – Exploratory Data Analysis)

```
df.describe()
```

Gives:

- mean
- std
- min / max
- quartiles

## 4. Select columns (features & label)

```
X = df[["age", "salary", "experience"]]  
y = df["label"]
```

age	salary	experience	label
22	30000	0	0
25	40000	2	0
30	60000	5	1
35	80000	8	1
40	90000	10	1

**df**

age	salary	experience	label
22	30000	0	0
25	40000	2	0
30	60000	5	1
35	80000	8	1
40	90000	10	1

**X**

**Y**

# 5. Select rows

1. `df["salary"]`
2. `df[df["salary"] > 50000]`
3. `df[df["experience"] >= 5]`

index	salary
0	30000
1	40000
2	60000
3	80000
4	90000

`df["salary"]`

age	salary	experience	label
30	60000	5	1
35	80000	8	1
40	90000	10	1

`df[df["salary"] > 50000]`

age	salary	experience	label
30	60000	5	1
35	80000	8	1
40	90000	10	1

`df[df["experience"] >= 5]`

## 6. Create new features

```
df["salary_per_year"] = df["monthly_salary"] * 12
```

age	monthly_salary	experience	label
22	2500	0	0
25	3000	2	0
30	5000	5	1
35	6500	8	1
40	7500	10	1

Original Dataset (df)

age	monthly_salary	experience	salary_per_year	label
22	2500	0	30000	0
25	3000	2	36000	0
30	5000	5	60000	1
35	6500	8	78000	1
40	7500	10	90000	1

After: New Feature Added (salary\_per\_year)

# | 7. Encode categorical data

```
df["gender"] = df["gender"].map({"M": 0, "F": 1})
```

age	salary	experience	gender	label	age	salary	experience	gender	label
22	30000	0	M	0	22	30000	0	0	0
25	40000	2	F	0	25	40000	2	1	0
30	60000	5	M	1	30	60000	5	0	1
35	80000	8	F	1	35	80000	8	1	1
40	90000	10	M	1	40	90000	10	0	1

Original Dataset (df)

After: Encoded gender Column

## 8. Drop unnecessary columns

```
df.drop(columns=["id", "name"], inplace=True)
```

id	name	age	salary	experience	label
101	arif	22	30000	0	0
102	rina	25	40000	2	0
103	sumon	30	60000	5	1

Original Dataset (df)

age	salary	experience	label
22	30000	0	0
25	40000	2	0
30	60000	5	1

After: id and name removed

## 9. Sort and group data (basic EDA)

```
df.sort_values(by="salary", ascending=False)  
df.groupby("label").mean()
```

age	salary	experience	label
22	30000	0	0
25	40000	2	0
30	60000	5	1

Original Dataset (df)

age	salary	experience	label
30	60000	5	1
25	40000	2	0
22	30000	0	0

After: Sorting by salary

label	age	salary	experience
0	23.5	35000	1.0
1	35.0	76666.7	7.7

After: Group By

## 10. Convert pandas → NumPy (for ML models)

```
X_np = X.values
```

```
y_np = y.values
```

## | 11. Save processed dataset

```
df.to_csv("clean_data.csv", index=False)
```



# **Class 03-**

## **- Data Visualization**

# | Data Visualization

**Data visualization** is the process of **representing data in graphical form** (charts, plots, graphs) so that humans can **easily understand patterns, trends, and insights** that are hard to see in raw numbers or tables.

# | Data Visualization

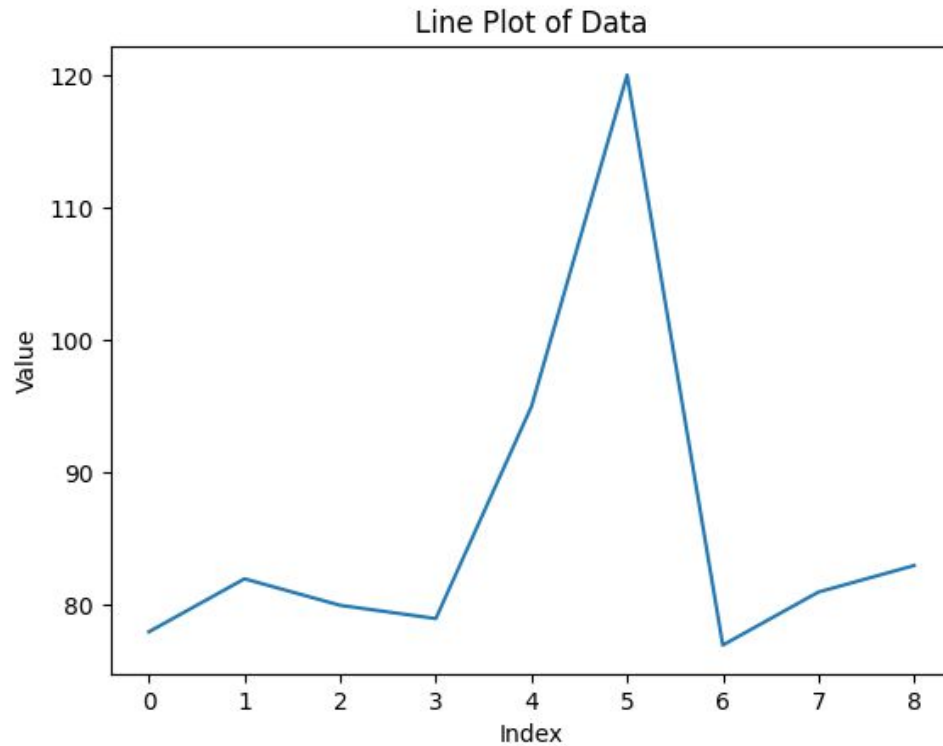
Index	Value
0	78
1	82
2	80
3	79
4	95
5	120
6	77
7	81
8	83

Dataset

# | Data Visualization

Index	Value
0	78
1	82
2	80
3	79
4	95
5	120
6	77
7	81
8	83

Dataset

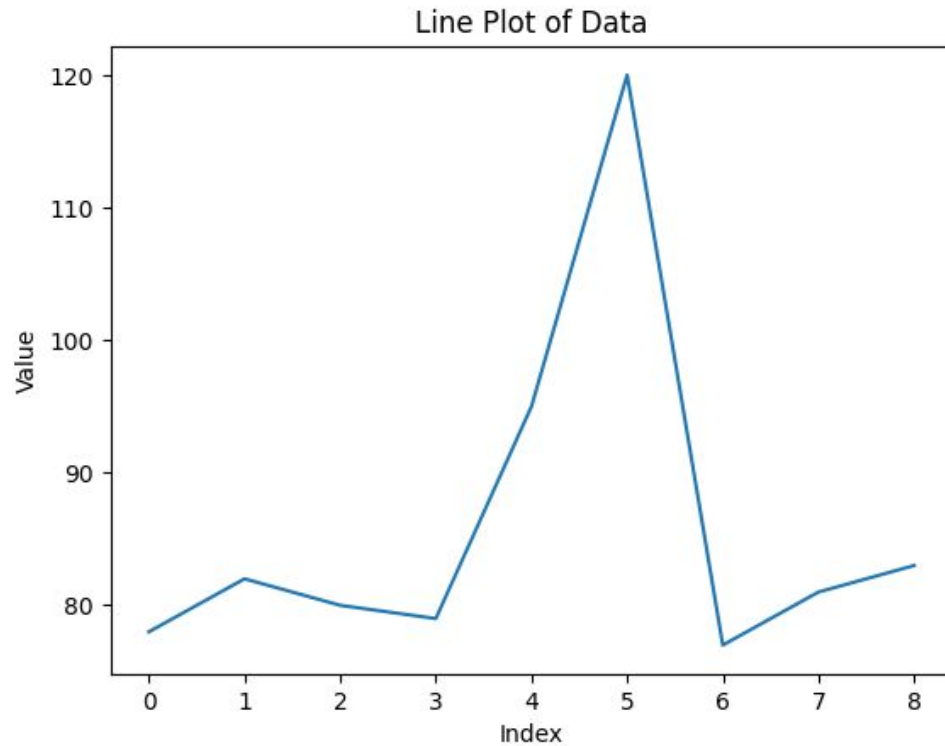


Line Plot

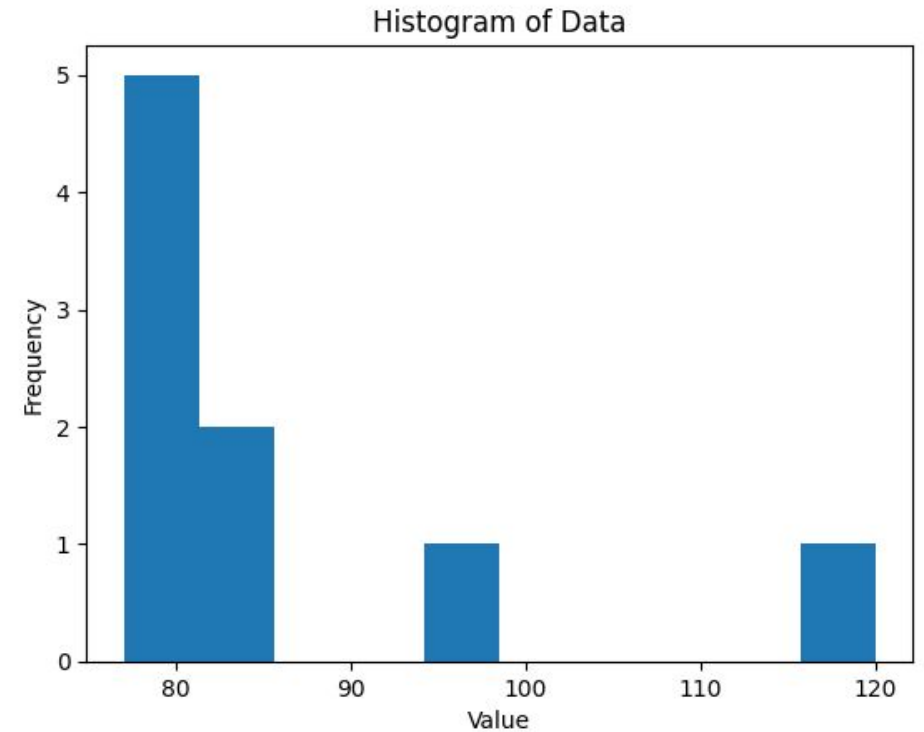
# | Data Visualization

Index	Value
0	78
1	82
2	80
3	79
4	95
5	120
6	77
7	81
8	83

Dataset



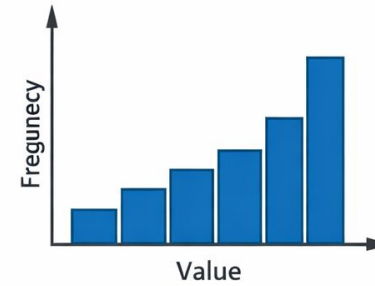
Line Plot



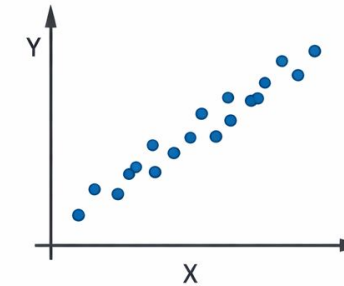
Histogram

# Plots (Diagrams)

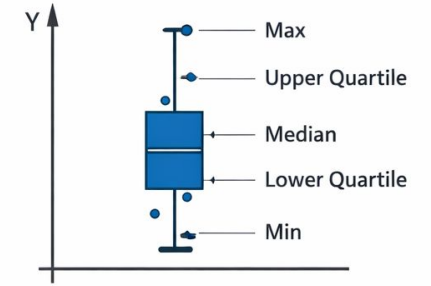
Plot Type	Purpose
Histogram	Distribution of data
Scatter plot	Relationship between variables
Box plot	Outliers and spread
Heatmap	Correlation
Line plot	Trend over time
Bar chart	Category comparison



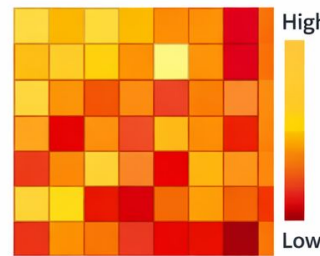
Histogram



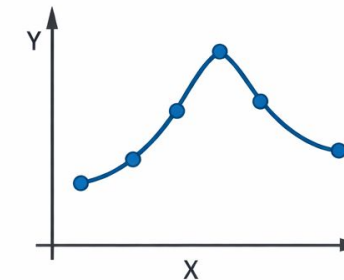
Scatter Plot



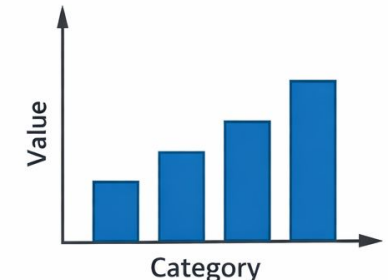
Box Plot



Heatmap



Line Plot



Bar Chart

# | Why data visualization is used

## 1. To understand data (Exploratory Data Analysis)

Before using ML models, you must understand:

- Data distribution
- Missing values
- Outliers
- Relationships between features

Visualization makes these visible.

Examples:

- Histogram → data spread
- Box plot → outliers
- Scatter plot → relationships

## 2. To find patterns and trends

Visualization helps detect:

- Increasing or decreasing trends
- Seasonality
- Clusters
- Class separation

Example:

- Time-series plot → trend over time
- Scatter plot → linear or non-linear relation

## 3. To detect errors and data quality issues

Common issues:

- Wrong units
- Sensor faults
- Data entry mistakes
- Extreme noise

Plots reveal problems **before** model training.

Example:

- Sudden spikes in sensor data
- Negative values where impossible

# | Why data visualization is used

## 4. To choose the right ML model

Visualization guides decisions like:

- Linear vs non-linear model
- Need for scaling
- Feature usefulness
- Class imbalance

Example:

- Clear linear pattern → Linear Regression
- Overlapping classes → need complex classifier

## 5. To evaluate model performance

After training, visualization helps answer:

- Is the model overfitting?
- How accurate are predictions?
- Where does the model fail?

Examples:

- Loss vs epochs
- Confusion matrix
- Prediction vs actual plot
- ROC curve

## 6. To communicate results clearly

Visualization helps explain:

- Findings to teammates
- Results to non-technical people
- Decisions to stakeholders

A plot is more convincing than raw numbers.



# **Class 03-**

## **- Matplotlib**

# | Why Matplotlib is used

**Matplotlib** is the **base plotting library** in Python.

## **What it is good at**

- Full control over plots
- Works with NumPy, Pandas, and Scikit-learn
- Lightweight and flexible
- Used everywhere (research, industry, robotics)

## **Common ML use cases**

- Line plots (loss vs epochs)
- Scatter plots (feature vs target)
- Histograms (data distribution)
- Bar charts (class counts)
- Plotting predictions vs ground truth



# **Class 03- - Seaborn**

# | Why Seaborn is used

Seaborn is built on top of matplotlib and is **ML/data-science friendly**.

## What it is good at

- Cleaner, prettier plots by default
- Statistical visualizations
- Direct support for Pandas DataFrames
- Faster EDA (Exploratory Data Analysis)

## Common ML use cases

- Feature distribution analysis
- Correlation heatmaps
- Class separation visualization
- Comparing multiple features at once



**ANY QUESTIONS ?**

# THANK YOU

## Contact Me



Email: [shafatsib@gmail.com](mailto:shafatsib@gmail.com)



Phone: +880178-0404749



Dhaka, Bangladesh

[Google Scholar](#) [Github](#) [Linkedin](#) [Youtube](#)