Part 2:

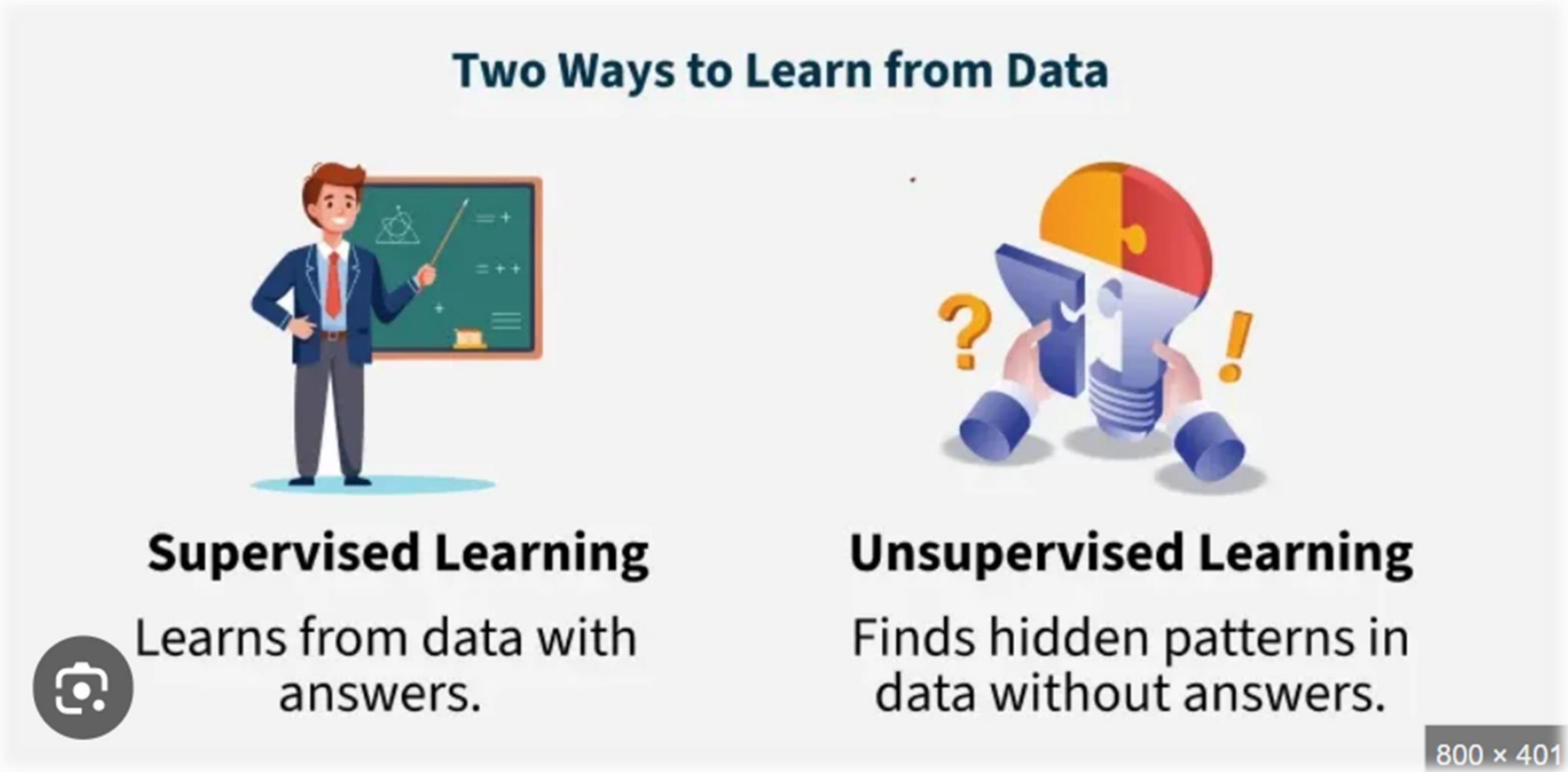
Supervised vs Unsupervised Learning

Supervised Learning

- We have **input data** X and **output labels** y.
- Learn a function f such that $y \approx f(X)$.
- Examples: predicting house prices, classifying emails as spam / not spam.

Unsupervised Learning

- Only **input data** X (no labels).
- Discover structure or patterns in the data.
- Examples: customer segmentation, clustering images, PCA.


Two Ways to Learn from Data
Supervised Learning
Learns from data with answers.
Unsupervised Learning
Finds hidden patterns in data without answers.
800 × 401

Machine Learning has two main learning styles :

just like students learn in two different ways

Supervised Learning

Definition

•Computer learns using data **with answers**

•Inputs + Outputs available

Simple idea:

Learning with a teacher

Unsupervised Learning

Definition

•Computer learns using data **without answers**

•Only inputs available

Simple idea:

Learning without a teacher

🔵 Supervised Learning Example

Studying Past Exam Papers

You have:

- Questions
- Answers

You learn to solve new questions.

Real-Life Uses:

- Predict house prices
- Detect spam emails
- Predict student marks

🟢 Unsupervised Learning Example

Friend Groups in Classroom

No one tells the groups.
But you observe and find:

- Gamers
- Studious students
- Sports lovers

Real-Life Uses:

- Customer segmentation
- Grouping photos
- Market analysis

Examples of Machine Learning Algorithms

Supervised Learning

- Linear Regression
- Logistic Regression
- *k*-Nearest Neighbors (k-NN)
- Decision Trees
- Random Forests
- Support Vector Machines (SVM)
- Neural Networks

Unsupervised Learning

- *k*-Means Clustering
- Hierarchical Clustering
- DBSCAN
- Principal Component Analysis (PCA)
- Independent Component Analysis (ICA)
- Autoencoders

Python for Machine Learning

Python is widely used in ML because:

- Clean and readable syntax.
- Rich scientific libraries (NumPy, Pandas, Matplotlib).
- Strong ML ecosystem (scikit-learn, TensorFlow, PyTorch).
- Large community and good documentation.

Typical ML workflow in Python:

- Load and clean data.
- Explore and visualize.
- Build and evaluate models.
- Deploy and use models.

Important Python Libraries for ML

Data Handling

- **NumPy** – arrays, matrices, math.
- **Pandas** – tables, CSV, Excel.
- **SciPy** – scientific routines.

ML + Visualization

- **Matplotlib** – plotting graphs.
- **scikit-learn** – classical ML algorithms.
- **Seaborn** – statistical plots.

Eg: open csv file without using Libraries

```
def read_csv_without_libraries(filename):
    data = []
    with open(filename, 'r') as file:
        for line in file:
         values = line.strip().split(',')

         data.append(values)
    return data
csv_data = read_csv_without_libraries('example.csv')
```

Eg: open csv file using Pandas Library

```
import pandas as pd


df = pd.read_csv('example.csv')
```

Basic Python Syntax for ML

Variables

`a = 10, name = "USJ"`

Lists and Arrays

- Python list: `x = [1, 2, 3]`
- NumPy array: `x = np.array([1, 2, 3])`

Function

```
def square(x):
    return x * x
```

Importing Packages

`import numpy as np   import pandas as pd`

Code:

basic python

https://github.com/LakshithaSenavirathna/MachineLearningTutorial/blob/main/ML_tutorial_1.ipynb

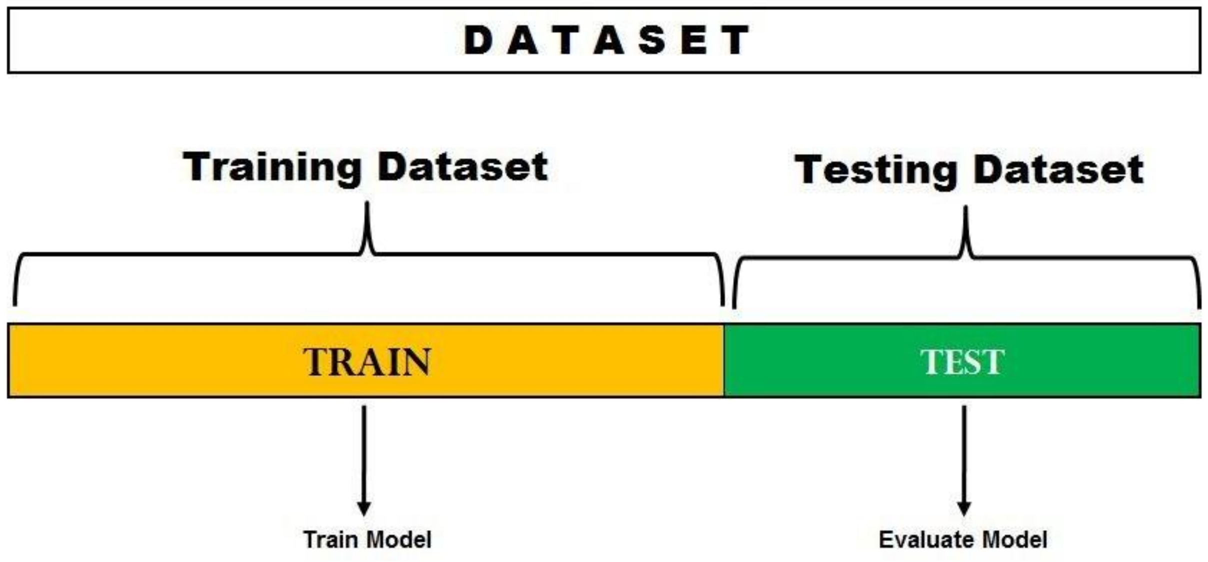
Datasets: Features, Labels, Train/Test Split

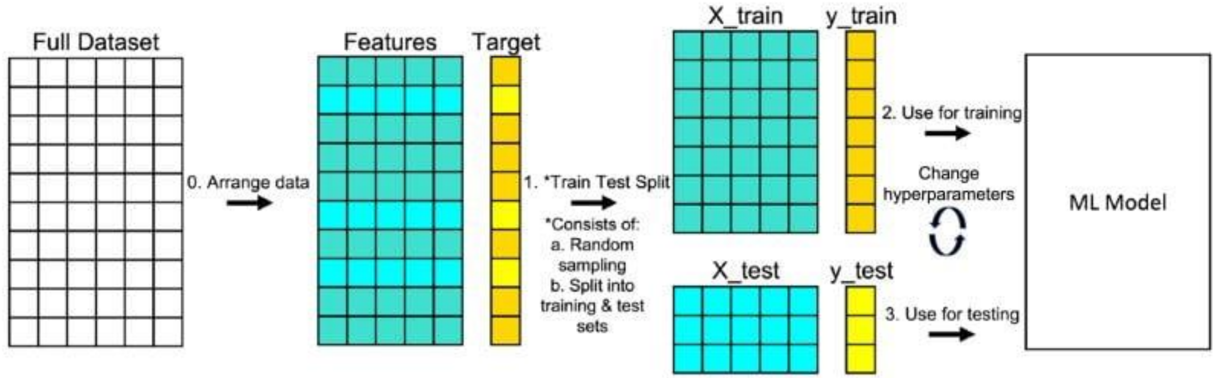
Features and Labels

- **Features** (inputs): variables used to predict.
- **Label** (target): what we want to predict.
- In code: features X, labels y.

Train / Test Split

- Split the data into:
  - **Training set** – used to fit the model.
  - **Test set** – used to evaluate performance.
- Helps estimate generalization to unseen data.





scikit-learn Interface Pattern

Most models in `scikit-learn` follow this pattern:

1. **Create** a model `model = LinearRegression()`
2. **Fit** on training data `model.fit(X_train, y_train)`
3. **Predict** on new data `y_pred = model.predict(X_test)`

Same interface for many algorithms: trees, SVM, k-NN, etc.

```python
import pandas as pd

# Load the CSV file
data = pd. read_csv (" simple_regression_data .csv ")

# Look at the first few rows
print ( data . head ())

# Summary statistics
print ( data . describe ())
```

Code:

load csv

https://github.com/LakshithaSenavirathna/MachineLearningTutorial/blob/main/ML_tutorial_2.ipynb

Simple Linear Regression with One Variable

Model

Assume a linear relationship between x and y:

$$y = \beta_0 + \beta_1 x + \varepsilon,$$

where:

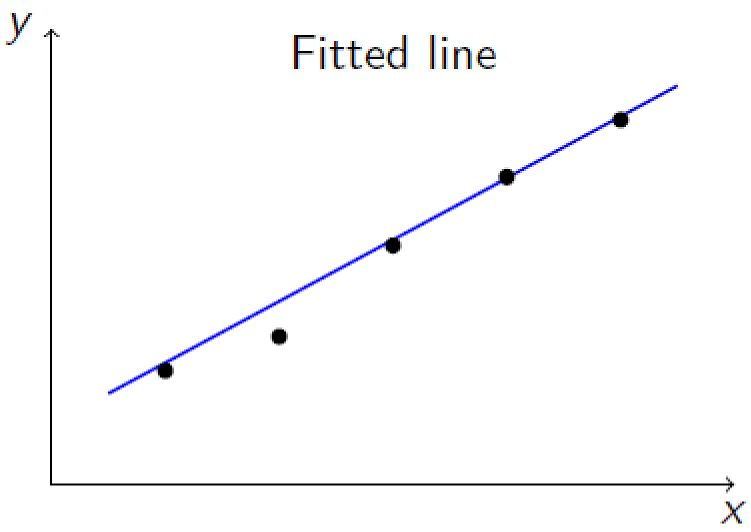
- β_0 – intercept,
- β_1 – slope,
- ε – error term.

Idea

- Observe data points (x_i, y_i).
- Choose β_0, β_1 so the line is "close" to the points.
- Minimize the sum of squared errors:

$$SSE = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$$

Visualizing Simple Linear Regression



Requirements for the Data Set

- Numerical variables: one predictor x and one response y.
- Enough points (e.g. $n \geq 20$) to see the relationship.
- Approximate linearity in the scatter plot of y vs x.
- No extreme outliers dominating the fit.
- Reasonable spread in x; not all x_i almost equal.

Exploring and Visualizing the Data

Statistical Summaries

- Mean, median, standard deviation of x and y.
- Minimum, maximum, quartiles.
- Correlation between x and y.

Graphics

- Scatter plot of y vs x.
- Histogram of residuals after fitting.
- Residual vs fitted plot to check assumptions.

Code

https://github.com/LakshithaSenavirathna/MachineLearningTutorial/blob/main/ML_tutorial_3.ipynb

scikit-learn Pipeline (Conceptual)

A typical `scikit-learn` pipeline:

- **Step 1: Preprocessing**
  - Handle missing values.
  - Scale / normalize features.
  - Encode categorical variables.
- **Step 2: Model**
  - LinearRegression, RandomForestRegressor, etc.
- **Step 3: Prediction**
  - Use the trained pipeline to predict on new data.

In code (idea): `Pipeline([("scaler", StandardScaler()), ("model", LinearRegression())])`

Overall Python ML Workflow

1. Collect data.
2. Clean and preprocess data.
3. Explore and visualize data.
4. Choose and train ML model(s).
5. Evaluate on test data.
6. Deploy the model and monitor performance.

Each step can be demonstrated with short Python examples in a Jupyter notebook.

Thank You ☺