

Module 1: Introduction to Machine Learning and Python

1. Introduction to Machine Learning (ML)

What is Machine Learning?

Machine Learning is a subset of Artificial Intelligence (AI) that enables systems to learn from data, improve performance over time, and make decisions without being explicitly programmed.

Importance of Machine Learning

- Automates analytical model building.
- Can uncover hidden patterns in data.
- Scales easily with more data.
- Powers applications like recommendation systems, speech recognition, and fraud detection.

Types of Machine Learning:

Type | Description | Example

----|-----|-----

Supervised Learning | Model learns from labeled data (input-output pairs). | Email spam classification

Unsupervised Learning | Model finds patterns in data without labeled output. | Customer segmentation

Reinforcement Learning | Model learns by trial-and-error to maximize a reward. | Game playing agents like AlphaGo

Applications of Machine Learning

- Healthcare (disease prediction)
- Finance (credit scoring)

- Retail (recommendation engines)
- Self-driving cars
- NLP (chatbots, translation)

2. Python for Data Analysis

Why Python?

- Simple syntax
- Huge ecosystem of ML/data science libraries
- Great for rapid prototyping and analysis

Popular Python Libraries:

Library | Purpose

-----|-----

NumPy | Supports large, multi-dimensional arrays and matrices

Pandas | High-level data manipulation and analysis

Matplotlib | Plotting and data visualization

Seaborn | Statistical data visualization

Scikit-learn | Core library for ML algorithms

TensorFlow / PyTorch | Deep learning frameworks

3. Python Programming Basics

Example Code - Basic Data Manipulation:

```
import pandas as pd
```

```
data = {'Name': ['Alice', 'Bob'], 'Age': [25, 30]}
```

```
df = pd.DataFrame(data)

print(df)

df['Salary'] = [50000, 60000]

print(df.describe())
```

NumPy Array Example:

```
import numpy as np

arr = np.array([1, 2, 3, 4])

print(arr * 2)
```

Data Visualization with Matplotlib:

```
import matplotlib.pyplot as plt

x = [1, 2, 3, 4]

y = [2, 4, 6, 8]

plt.plot(x, y)

plt.xlabel("X-axis")

plt.ylabel("Y-axis")

plt.title("Simple Line Graph")

plt.show()
```

4. Data Preprocessing

What is Data Preprocessing?

Transforming raw data into a clean dataset for a machine learning model. It's a key step for improving model performance.

Key Steps in Data Preprocessing:

a) Data Cleaning

Handling Missing Values:

```
df.fillna(df.mean(), inplace=True)
```

Removing Duplicates:

```
df.drop_duplicates(inplace=True)
```

b) Handling Outliers

Use boxplots to detect outliers.

Use Z-score or IQR method to remove them.

c) Feature Scaling

Min-Max Scaling:

```
from sklearn.preprocessing import MinMaxScaler
```

```
scaler = MinMaxScaler()
```

```
df_scaled = scaler.fit_transform(df)
```

Standardization (Z-score normalization):

```
from sklearn.preprocessing import StandardScaler
```

```
scaler = StandardScaler()
```

```
df_scaled = scaler.fit_transform(df)
```

d) Encoding Categorical Variables

Label Encoding:

```
from sklearn.preprocessing import LabelEncoder
```

```
le = LabelEncoder()
```

```
df['Gender'] = le.fit_transform(df['Gender'])
```

One-Hot Encoding:

```
pd.get_dummies(df, columns=['City'])
```

Summary

Topic | Key Points

-----|-----

ML Introduction | Learn from data, improve performance

ML Types | Supervised, Unsupervised, Reinforcement

Python for ML | Libraries: NumPy, Pandas, Matplotlib, Sklearn

Preprocessing | Cleaning, Handling missing values/outliers, Feature Scaling, Encoding