PYTHON FOR COMPUTATIONAL PROBLEM SOLVING

Introduction to Machine Learning in Python using Scikit-Learn

UE25CS151A

Department of Computer Science and Engineering
Prof. Kundhavai K R, CSE Department

What is Machine Learning?

Machine Learning (ML) = **teaching computers to learn from data**.

Instead of writing explicit rules for every situation:

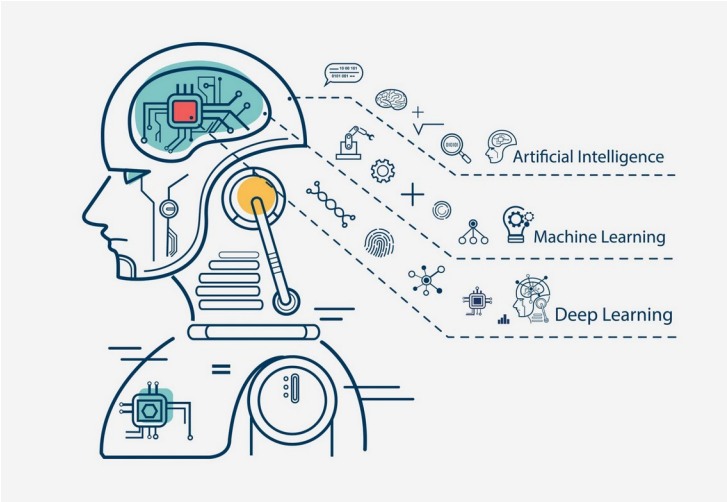
- We give a computer **lots of examples**
  It **learns patterns** automatically
- **Analogy**

Traditional programming:
"If email contains 'free money', mark as spam."

Machine Learning:
Give the computer **10,000 labeled emails** → it learns what spam looks like.

Real World Examples

- **YouTube/Netflix → video recommendations**
  Gmail → spam detection
  Amazon → product suggestions
  Banks → fraud detection
  Healthcare → disease prediction



Types of Machine Learning

Supervised Learning

- Data has **labels**
  Learn mapping: **X → y**
  Examples:
    Spam detection
    Iris flower classification
    Predicting house prices

Types of Machine Learning

Unsupervised Learning

- Data has **no labels**
- Model discovers patterns/groups

  Examples:

  Customer segmentation

  Clustering documents

Types of Machine Learning

Reinforcement Learning

- Learn by **trial & error**
  Reward-based learning
  Examples:
    - Game AI (Chess/Go)
    - Robotics
    - Self-driving cars

Why Python for ML

Why?

- Most popular ML language
  Simple syntax
  Huge ecosystem

Powerful Python Libraries

- NumPy
  Pandas
  Matplotlib
  Scikit-learn (sklearn) — our ML toolkit

Understanding Datasets

A dataset is like an **Excel table** with rows & columns. The dataset described below is iris dataset in built in sklearn module.

Two important parts: Features (X) — “Questions”

Inputs collected from each sample
Iris example features:

- Sepal Length
- Sepal Width
- Petal Length
- Petal Width

Labels (y) — “Answers”

Output you want to predict Iris example labels:

- Setosa
- Versicolor
- Virginica

Installation

Install scikit-learn

```
pip install scikit-learn
```

Example Dataset - IRIS

- 150 samples (flowers)
- 4 features (measurements)
- 1 label (species)
- Built into sklearn
- Most widely used beginner dataset

Loading the Iris Dataset

```
from sklearn.datasets import load_iris

iris = load_iris()

X = iris.data      # features

y = iris.target    # labels

print(X.shape)

print(y.shape)

print(X[:5])

print(y[:5])
```

Output Explanation

X Shape → (150, 4)

- 150 flowers
- 4 features each

y Shape → (150,)

- 150 labels

First 5 feature rows

Example: [5.1 3.5 1.4 0.2]
Means:

- Sepal len = 5.1
  Sepal wid = 3.5
  Petal len = 1.4
  Petal wid = 0.2

First 5 labels → [0 0 0 0 0]

- 0 = Setosa
  1 = Versicolor
  2 = Virginica

The Big Problem

If we train the model on **all 150 flowers**, how do we check if it actually learned?

It might:

✔ Memorize

✘ Fail to generalize

Analogy

Practice questions ≠ Final exam

You won’t know true performance unless you test on new questions.

Solution: Train–Test Split

Split dataset:

Training Set (75%)

- Model learns patterns
  Like practice questions

Testing Set (25%)

- Model is evaluated
  Like final exam

Code : Splitting Data

```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, random_state=42)

print("Training samples:", X_train.shape[0])   # 112

print("Testing samples:", X_test.shape[0])     # 38
```

PES UNIVERSITY

Split Explanation

- **X** → All input features
  y → All labels
  test_size = 0.25 → 25% of data used for testing
  random_state = 42 → Fixes randomness, ensuring the same split every run

What is a Model

A model is the **brain**.

It: Learns the relationship between X and y
Predicts labels for new data

- **We will use: KNN (K-Nearest Neighbors)**

Concept:

- Look at the **k closest data points**
  Majority class wins

Analogy:
"You are who your closest neighbors are."

Steps to build a model

3-Step ML Process

Create the model
Train the model
Test the model

Example Code for KNN

```
from sklearn.neighbors import KNeighborsClassifier

from sklearn.metrics import accuracy_score

knn = KNeighborsClassifier(n_neighbors=3)  # model

knn.fit(X_train, y_train)                  # training

y_pred = knn.predict(X_test)               # testing

accuracy = accuracy_score(y_test, y_pred)  # evaluation
```

Evaluating Perforamance

Output:

Predictions: [...]

Actual: [...]

Accuracy: 96.XX%

Accuracy Score

- Value between 0 and 1
  Multiply by 100 for %
  Example: 0.965 → **96.5% accuracy**

Full ML Workflow Summary

1. **Load Data :**Define X and y
2. **Train–Test Split**

- 75% training
  25% testing

3. **Choose Model :** KNN
4. **Train :** model.fit(X_train, y_train)

5. **Test & Evaluate :**

- model.predict(X_test)
  accuracy_score()

THANK YOU

Department of Computer Science and Engineering

Prof. Kundhavai K R, CSE Department

Ack: Teaching Assistant:
Adithya Jeyaramsankar– PES2UG22CS029