**AI Lab – Integrated Lab on Scikit-learn, Preprocessing, and AI Pipelines**

**Objective:**

To introduce basic machine learning using **Scikit-learn**, along with **data preprocessing** and **building AI pipelines** in Python.

1. **Introduction to Scikit-learn (Machine Learning Basics)**

**Machine Learning (ML)** is the core part of Artificial Intelligence (AI). It allows computers to learn from data and make decisions.

**Supervised Learning** is a type of ML where we teach the model using labeled data (input + output).

**Scikit-learn** is a Python library used to build ML models easily.

**Common Terms:**

* **Dataset**: Collection of data (features + target)
* **Model**: Mathematical formula to make predictions
* **fit()**: Train the model
* **predict()**: Make prediction
* **accuracy\_score**: Measure how correct the model is

**List of Libraries which is used,**

**Scikit-learn (sklearn) – Core AI & ML Library**

| **Module** | **Function** | **Purpose** |
| --- | --- | --- |
| sklearn.datasets | load\_iris(), load\_digits(), load\_breast\_cancer() | Loads free, ready-to-use datasets for practice |
| sklearn.model\_selection | train\_test\_split() | Splits your data into training and testing sets |
| sklearn.tree | DecisionTreeClassifier() | A model that uses decision trees for classification |
| sklearn.neighbors | KNeighborsClassifier() | A model that classifies using the nearest neighbors |
| sklearn.linear\_model | LogisticRegression() | A simple model for binary/multi-class classification |
| sklearn.preprocessing | LabelEncoder, StandardScaler | Prepares the data: encodes text, scales numbers |
| sklearn.pipeline | Pipeline() | Combines multiple steps (scaling + modeling) into one clean unit |
| sklearn.metrics | accuracy\_score, classification\_report | Checks how well your model is performing |

**Code Example:**

from sklearn.datasets import load\_iris

from sklearn.model\_selection import train\_test\_split

from sklearn.tree import DecisionTreeClassifier

from sklearn.metrics import accuracy\_score

# Load dataset

data = load\_iris()

# Split data

X\_train, X\_test, y\_train, y\_test = train\_test\_split(data.data, data.target, test\_size=0.2)

# Create and train model

model = DecisionTreeClassifier()

model.fit(X\_train, y\_train)

# Predict and evaluate

predictions = model.predict(X\_test)

print("Accuracy:", accuracy\_score(y\_test, predictions))

**Lab Task:**

* Replace DecisionTreeClassifier with KNeighborsClassifier and compare results.

1. **Data Preprocessing and Feature Engineering**

Before training a model, we must **clean and prepare data**. This step is called **Data Preprocessing**.

**Common Techniques:**

* **Missing Values**: Fill empty data using mean, median, or mode.
* **Label Encoding**: Convert text data into numbers.
* **Feature Scaling**: Normalize values for better model performance.

**Code Example:**

import pandas as pd

from sklearn.preprocessing import LabelEncoder, StandardScaler

# Sample dataset

df = pd.DataFrame({

'Age': [25, 30, None, 35],

'Gender': ['Male', 'Female', 'Female', 'Male']

})

# Handle missing value

df['Age'].fillna(df['Age'].mean(), inplace=True)

# Encode Gender

df['Gender'] = LabelEncoder().fit\_transform(df['Gender'])

# Scale Age

scaler = StandardScaler()

df['Age'] = scaler.fit\_transform(df[['Age']])

print(df)

**Lab Task:**

* Create a similar dataset with 2 numeric columns and 1 text column.
* Apply missing value handling, label encoding, and scaling.

1. **Building an AI Pipeline**

A **Pipeline** is a step-by-step process where we combine data preprocessing and model training into one unit.

**Why Pipelines?**

* Cleaner code
* Avoids mistakes
* Good for real-world applications

**Key Terms:**

* **Pipeline**: Combines multiple steps
* **StandardScaler**: Normalizes data
* **LogisticRegression**: A classification algorithm
* **classification\_report**: Shows model performance

**Code Example:**

from sklearn.pipeline import Pipeline

from sklearn.preprocessing import StandardScaler

from sklearn.linear\_model import LogisticRegression

from sklearn.datasets import load\_breast\_cancer

from sklearn.model\_selection import train\_test\_split

from sklearn.metrics import classification\_report

# Load data

data = load\_breast\_cancer()

X\_train, X\_test, y\_train, y\_test = train\_test\_split(data.data, data.target, test\_size=0.2)

# Create pipeline

pipeline = Pipeline([

('scaler', StandardScaler()),

('classifier', LogisticRegression())

])

# Train and evaluate

pipeline.fit(X\_train, y\_train)

y\_pred = pipeline.predict(X\_test)

print(classification\_report(y\_test, y\_pred))

**Lab Task:**

* Use load\_digits() dataset.
* Create a pipeline with StandardScaler and KNeighborsClassifier.

| **Topic** | **What You Learned** |
| --- | --- |
| Scikit-learn | Build and test ML models |
| Preprocessing | Clean and transform raw data |
| Pipeline | Combine steps for full AI workflow |