## **Lesson Documentation: Supervised Machine Learning with Python**

### **1. Lesson Overview**

Supervised Machine Learning is a fundamental approach in artificial intelligence and data science that involves **training models using labeled data** — where each training example is paired with an output label. The goal is for the model to **learn a mapping** between inputs and outputs and make accurate predictions on new, unseen data.

This lesson introduces the **concepts, workflow, and basic implementation** of supervised learning using **Python** and its core machine learning library, **scikit-learn**.

### **2. Learning Objectives**

By the end of this lesson, students should be able to:

1. Define supervised learning and distinguish it from unsupervised learning.
2. Explain the structure of labeled data (features and target variables).
3. Identify common supervised learning algorithms (Regression and Classification).
4. Implement a simple supervised learning model in Python.
5. Evaluate model performance using standard metrics.

### **3. Concept Explanation**

#### **What is Supervised Learning?**

Supervised learning is a method in which a model learns from a dataset that contains **input-output pairs**.  
 The **input** is a set of features (independent variables), and the **output** is the target (dependent variable).

The model’s job is to approximate a function ( f ) such that:

[Y = f(X)]

Where:

* ( X ) → Input data (features)
* ( Y ) → Output data (label)

#### **Types of Supervised Learning:**

1. **Regression** – Predicts continuous outcomes.  
   * Example: Predicting house prices, temperature, or income.
2. **Classification** – Predicts categorical outcomes.  
   * Example: Spam detection, disease diagnosis, or sentiment analysis.

### **4. The Supervised Learning Workflow**

1. **Define the Problem:** Identify what you want to predict.
2. **Collect and Prepare Data:** Organize labeled data with features (X) and target (Y).
3. **Split Data:** Divide into training and testing sets (e.g., 80/20).
4. **Select a Model:** Choose an appropriate algorithm (e.g., Linear Regression, Decision Tree).
5. **Train the Model:** Feed training data into the algorithm to learn relationships.
6. **Evaluate the Model:** Test the model on unseen data using metrics like accuracy or R².
7. **Deploy and Monitor:** Use the model for predictions and track performance over time.

### **5. Example in Python**

#### **Dataset**

Let’s assume we have data predicting **student exam scores (Y)** based on **hours studied (X)**.

| **Hours Studied (X)** | **Exam Score (Y)** |
| --- | --- |
| 1.5 | 35 |
| 3.0 | 50 |
| 4.5 | 65 |
| 5.0 | 70 |
| 6.5 | 80 |
| 8.0 | 88 |

#### **Step-by-Step Implementation**

# Step 1: Import Libraries

import pandas as pd

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

from sklearn.linear\_model import LinearRegression

from sklearn.metrics import r2\_score, mean\_squared\_error

# Step 2: Create the Dataset

data = {

'Hours\_Studied': [1.5, 3.0, 4.5, 5.0, 6.5, 8.0],

'Exam\_Score': [35, 50, 65, 70, 80, 88]

}

df = pd.DataFrame(data)

# Step 3: Define Variables

X = df[['Hours\_Studied']]

y = df['Exam\_Score']

# Step 4: Split Data

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

# Step 5: Train Model

model = LinearRegression()

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

# Step 6: Make Predictions

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

# Step 7: Evaluate Model

r2 = r2\_score(y\_test, y\_pred)

mse = mean\_squared\_error(y\_test, y\_pred)

print("R² Score:", r2)

print("Mean Squared Error:", mse)

print("Model Coefficient (m):", model.coef\_[0])

print("Intercept (b):", model.intercept\_)

### **6. Interpreting the Results**

* **Regression Equation:** [  
   Y = mX + b  
   ]  
   For example, if ( m = 7.5 ) and ( b = 25 ), then:  
   [  
   Exam\ Score = 7.5 \times Hours\ Studied + 25  
   ]
* **Meaning:** For each additional hour studied, the exam score increases by approximately 7.5 points.
* **Evaluation Metrics:**
  + **R² (Coefficient of Determination):** Measures how well the model fits the data.
  + **MSE (Mean Squared Error):** Indicates how far predictions deviate from actual values.

### **7. Visualization**

import matplotlib.pyplot as plt

plt.scatter(X, y, color='blue', label='Actual Data')

plt.plot(X, model.predict(X), color='red', label='Regression Line')

plt.xlabel('Hours Studied')

plt.ylabel('Exam Score')

plt.title('Linear Regression Example')

plt.legend()

plt.show()

The scatter plot shows actual data points (blue dots) and the model’s prediction (red line).

### **8. In-Class Exercise**

Using Python:

* Create a dataset for **Income (Y)** vs **Years of Experience (X)**.
* Train a regression model.
* Plot the line of best fit.
* Interpret the slope and intercept in your own words.

### **9. Assessment Section**

1. Define supervised learning in your own words.
2. What is the main difference between regression and classification?
3. How is R² interpreted in a regression model?
4. Write the Python command used to train a Linear Regression model.
5. Explain the role of train\_test\_split() in model evaluation.

### **10. Real-World Applications**

* **Finance:** Predicting stock prices or credit risk.
* **Healthcare:** Diagnosing diseases based on symptoms.
* **Engineering:** Estimating energy consumption from sensor data.
* **Marketing:** Predicting customer purchase behavior.

### **11. Summary and Further Reading**

**Key Takeaways:**

* Supervised learning maps input features to known outputs.
* Regression handles continuous targets; classification handles categorical ones.
* The learning process involves training, testing, and evaluating a predictive model.

**Recommended Readings:**

* *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow* – Aurélien Géron
* *Python Machine Learning* – Sebastian Raschka
* Scikit-learn Documentation:<https://scikit-learn.org/stable/>