**Supervised Learning**

**Introduction**

Supervised Learning is a way of teaching computers using examples. It’s like guiding a student step-by-step until they can solve problems on their own. In this module, we’ll explore how supervised learning works, different types of it, and even build a simple machine learning model.

**What is Supervised Learning?**

Supervised learning means training a machine with labeled examples, just like how we learn from practice. For instance, imagine teaching a robot to recognize fruits. You show it pictures of apples and oranges and tell it which is which. After enough examples, the robot starts identifying them on its own.

There are two main types of supervised learning:

1. **Classification:** When the goal is to sort data into categories (e.g., "cat" or "dog").
2. **Regression:** When the goal is to predict a number (e.g., house prices or temperature).

**Real-Life Examples of Supervised Learning**

* **Classification:** Sorting emails into "spam" or "not spam."
* **Regression:** Predicting house prices based on their features like size or location.

**Popular Algorithms in Supervised Learning**

Here are some commonly used algorithms:

**For Classification:**

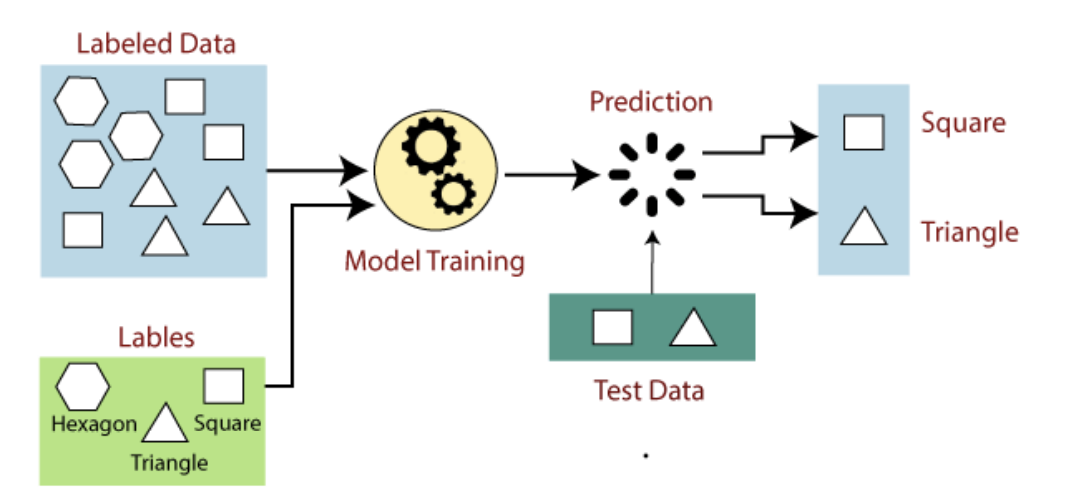
* Decision Trees
* K-Nearest Neighbors (KNN)
* Support Vector Machines (SVM)
* Logistic Regression
* Random Forest Classifier
* Naive Bayes

**For Regression:**

* Linear Regression
* Polynomial Regression
* Ridge Regression
* Lasso Regression
* Support Vector Regression (SVR)
* Random Forest Regressor

**How Supervised Learning Works**

1. **Labeled Data:**  
   The process begins with labeled data. This data includes input features (e.g., shapes like hexagons, triangles, and squares) and corresponding labels (e.g., "Hexagon," "Triangle," "Square"). The labels are the correct answers that the model will learn to predict.
2. **Model Training:**  
   The labeled data is fed into a machine learning algorithm, which trains a model. During training, the model learns patterns and relationships between the input features and their corresponding labels.
3. **Test Data:**  
   Once the model is trained, it is tested using new data (test data) that it hasn't seen before. This test data contains input features but no labels during the testing phase.
4. **Prediction:**  
   The trained model makes predictions on the test data, identifying which category (e.g., "Square" or "Triangle") each input belongs to.
5. **Evaluation:**  
   The predictions are compared with the actual labels of the test data to evaluate the model's accuracy and performance.



**Interactive Activity (Using the Diagram)**

1. **Activity Name:** Shape Classifier
2. **Objective:** Build a simple supervised learning model to classify shapes.
3. **Materials Needed:** Computer/laptop with Python installed.

**Instructions:**

* Create a dataset of shapes (hexagon, triangle, square) with features like the number of sides, angles, and edge lengths.
* Use the scikit-learn library to:
  + Train a Decision Tree Classifier on labeled data.
  + Test the model using unseen data (test data).
  + Evaluate its accuracy.

**Code Example:**  
This code builds a classifier based on shape features (e.g., number of sides, angles).

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

from sklearn.tree import DecisionTreeClassifier

from sklearn.metrics import accuracy\_score

# Example data: [number of sides, angles]

X = [[6, 120], [3, 60], [4, 90], [6, 120], [3, 60], [4, 90]]

y = ['Hexagon', 'Triangle', 'Square', 'Hexagon', 'Triangle', 'Square'] # Labels

# Split data into training and testing sets

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.5, random\_state=42)

# Train the model

model = DecisionTreeClassifier()

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

# Test the model

predictions = model.predict(X\_test)

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

**Advantages:**

* Produces accurate results when enough labeled data is available.
* Simple to understand and implement.

**Disadvantages:**

* Requires a large amount of labeled data.
* Can overfit if the model is too complex.

**Activity 1: Fruit Classifier (Classification)**

**Goal:** Build a simple program that can identify fruits as apples or oranges.

**What You Need:**

* A computer with Python installed.
* A small dataset of fruit examples with labels (e.g., "apple," "orange").

**Steps:**

1. Load your dataset.
2. Split it into training and testing parts.
3. Train a Decision Tree Classifier.
4. Test it and calculate how well it performs.

**Code Example:**

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

from sklearn.tree import DecisionTreeClassifier

from sklearn.metrics import accuracy\_score

# Example data

X = [[1], [2], [3], [4]] # Features (e.g., weight)

y = ['apple', 'apple', 'orange', 'orange'] # Labels

# Split data

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.5, random\_state=42)

# Train model

model = DecisionTreeClassifier()

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

# Test model

predictions = model.predict(X\_test)

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

**Activity 2: House Price Predictor (Regression)**

**Goal:** Predict house prices based on features like area and number of rooms.

**What You Need:**

* A computer with Python installed.
* A small dataset with features (e.g., area, number of rooms) and house prices.

**Steps:**

1. Load your dataset.
2. Split it into training and testing parts.
3. Train a Linear Regression model.
4. Test it and measure its accuracy.

**Code Example:**

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

from sklearn.linear\_model import LinearRegression

from sklearn.metrics import mean\_squared\_error

# Example data

X = [[1000, 2], [1500, 3], [2000, 4], [2500, 5]] # Features (area, rooms)

y = [300000, 450000, 600000, 750000] # Prices

# Split data

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.5, random\_state=42)

# Train model

model = LinearRegression()

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

# Test model

predictions = model.predict(X\_test)

print("Mean Squared Error:", mean\_squared\_error(y\_test, predictions))

**Reflection Questions**

1. What are the two main types of supervised learning? Provide an example of each.
2. Why is it important to split data into training and testing sets?
3. How does accuracy help in evaluating a model?

**Conclusion**

Supervised learning enables machines to learn from labeled data and make predictions on new data. Algorithms like Decision Trees and Linear Regression are tools we can use to solve real-world problems. By practicing the activities, you’ll understand how to train and test machine learning models, making you confident in applying supervised learning concepts.