**1. What is an error rate?**

Error rate is a simple way to measure how often a model makes mistakes. It's the percentage of wrong predictions out of the total number of predictions.

Code example:

from sklearn.metrics import accuracy\_score

# Suppose y\_true is the true labels and y\_pred is the predicted labels

y\_true = [0, 1, 1, 0, 1, 0]

y\_pred = [0, 1, 0, 0, 1, 1]

# Calculate accuracy

accuracy = accuracy\_score(y\_true, y\_pred)

# Calculate error rate

error\_rate = 1 - accuracy

**2. Where you could use other machine-learning models?**

Different machine learning models can be used to improve your results depending on your task:

**Logistic Regression** for simple binary classification tasks.

**K-Nearest Neighbors (KNN)** for classifying data points based on their closest neighbors.

**Support Vector Machines (SVM)** for separating data with a clear margin.

**Naive Bayes** for text classification, like spam detection.

**Decision Trees** for creating decision rules that are easy to understand.

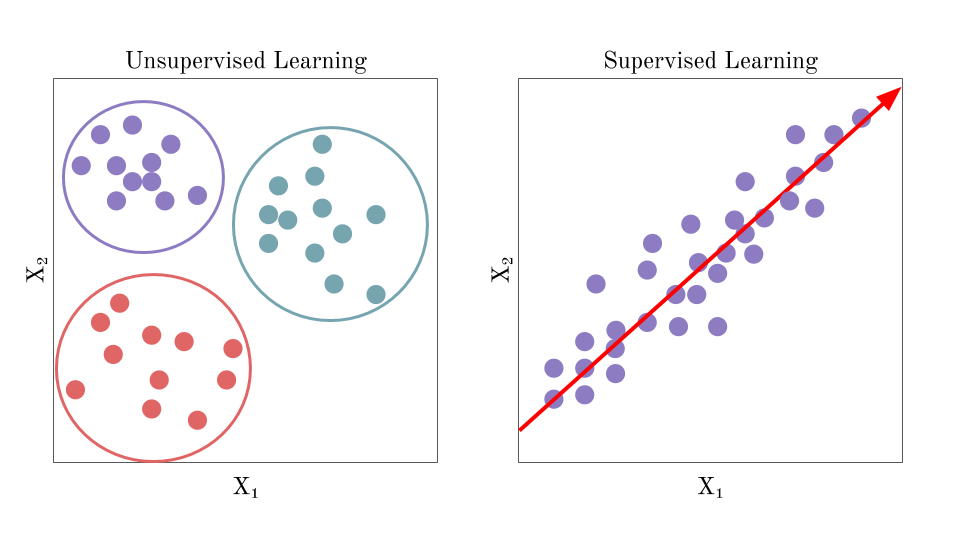
**Neural Networks** for more complex tasks like recognizing images or understanding speech.

By experimenting with these different models, you can find the one that works best for your specific data and problem.

**3. What is the difference between supervised and unsupervised training?**

Supervised learning is used when the goal is to predict a specific outcome based on input data that has associated labels.

Unsupervised learning is used when the goal is to explore the data and find patterns or groupings without predefined labels.



**4. How to import different models from the scikit-learn package?**

* **importing specific models directly by their names:**

from sklearn.linear\_model import LogisticRegression

from sklearn.ensemble import RandomForestClassifier

from sklearn.svm import SVC

from sklearn.neighbors import KNeighborsClassifier

from sklearn.tree import DecisionTreeClassifier

from sklearn.naive\_bayes import GaussianNB

from sklearn.cluster import KMeans

from sklearn.decomposition import PCA

from sklearn.linear\_model import LinearRegression

* **importing all models from a specific module (using the \* notation):**

from sklearn.linear\_model import \*

from sklearn.ensemble import \*

from sklearn.svm import \*

from sklearn.tree import \*

from sklearn.naive\_bayes import \*

* **importing the whole module and then referring to the models by their full path:**

import sklearn.linear\_model as lm

import sklearn.ensemble as ens

import sklearn.svm as svm

# Using the models

logistic\_regression = lm.LogisticRegression()

random\_forest = ens.RandomForestClassifier()

support\_vector\_machine = svm.SVC()

**5. How can you evaluate the performance of a machine learning model in scikit-learn?**

Evaluating the performance of a machine learning model in scikit-learn can be done using various metrics and techniques depending on the type of model:

* Classification: Use metrics like accuracy, precision, recall, F1-score, and ROC-AUC.
* Regression: Use metrics like MAE, MSE, and R².
* Cross-Validation: Ensures that the model generalizes well by evaluating it on multiple subsets of the data.
* Clustering: Use metrics like silhouette score and adjusted Rand index.

**6. What metrics are commonly used for evaluation?**

for **continuous data** (non binary)the following metrics can be used: R-squared score, mean absolute error, mean squared error. These metrics are applicable for regression models.

There are examples for data used during the class:

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

# Calculate R-squared

r2 = r2\_score(final\_df["Actual Sales"], final\_df["Predicted Sales"])

print(f"R-squared: {r2}")

# Calculate Mean Absolute Error

mae = mean\_absolute\_error(final\_df["Actual Sales"], final\_df["Predicted Sales"])

print(f"Mean Absolute Error: {mae}")

# Calculate Mean Square Error

mse = mean\_squared\_error(final\_df["Actual Sales"], final\_df["Predicted Sales"])

print(f"MSE: {mse}")

**7. What is model overfitting, and how can it be prevented?**

Overfitting is when a model performs well on training data but poorly on unseen data due to learning noise.

Prevention techniques include cross-validation, regularization, pruning, early stopping, using ensemble methods, and ensuring adequate data.