**AI Lab Topic: "Model Evaluation and Cross-Validation in Machine Learning"**

**Lab Objectives**

* Understand why model evaluation is important
* Explain overfitting and underfitting
* Use **cross-validation** to evaluate ML models more accurately
* Compare different models to choose the best one

**Beginner-Friendly Theory**

**1. What is Model Evaluation?**

When we build an AI model, we need to check:

“How good is this model?”

Model evaluation helps us test our model’s **accuracy**, **performance**, and **generalization**.

We don’t want a model that:

* Only works well on training data (memorizing)
* Fails to work on new, unseen data

**2. What is Train-Test Split?**

We divide the dataset into:

* **Training Set**: For training the model (usually 70–80%)
* **Testing Set**: For testing the model (remaining 20–30%)

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

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

But this has a **problem**:

One test split may not show the real performance of a model!

**3. Overfitting and Underfitting**

| **Concept** | **Description** | **Example** |
| --- | --- | --- |
| **Overfitting** | Model memorizes training data, fails on test data | 100% accuracy on train, 60% on test |
| **Underfitting** | Model is too simple, performs poorly on both | 60% train and 50% test accuracy |

**Visualization:**

We’ll use graphs in class to show how a model can overfit or underfit a curve.

**4. What is Cross-Validation?**

Instead of splitting data once, we split it **multiple times**. This is called **K-Fold Cross-Validation**.

**K-Fold Cross Validation:**

* Split the dataset into **K parts** (e.g. K=5)
* Train & test the model **K times**
* Each time, use a different fold for testing
* **Final result = average of all K scores**

from sklearn.model\_selection import cross\_val\_score

scores = cross\_val\_score(model, X, y, cv=5)

print("CV Score:", scores)

print("Average Score:", scores.mean())

**5. Comparing Models**

We can apply cross-validation to multiple models:

from sklearn.datasets import load\_iris

from sklearn.model\_selection import cross\_val\_score

from sklearn.tree import DecisionTreeClassifier

from sklearn.linear\_model import LogisticRegression

from sklearn.neighbors import KNeighborsClassifier

# Load dataset

data = load\_iris()

X, y = data.data, data.target

# Create models

models = {

'Decision Tree': DecisionTreeClassifier(),

'Logistic Regression': LogisticRegression(max\_iter=200),

'KNN': KNeighborsClassifier()

}

# Evaluate each model

for name, model in models.items():

scores = cross\_val\_score(model, X, y, cv=5)

print(f"{name} → CV Accuracy: {scores.mean():.2f}")

**Optional: Plot Results Using Matplotlib**

import matplotlib.pyplot as plt

names = list(models.keys())

accuracies = [cross\_val\_score(m, X, y, cv=5).mean() for m in models.values()]

plt.bar(names, accuracies, color='skyblue')

plt.title("Model Accuracy Comparison")

plt.ylabel("Cross-Validation Accuracy")

plt.ylim(0.8, 1.0)

plt.show()

**Student Lab Tasks:**

**Task 1:**

Use cross-validation to evaluate the performance of KNeighborsClassifier on the Iris dataset.

**Task 2:**

Use load\_digits() dataset and compare the performance of:

* Decision Tree
* Logistic Regression
* K-Nearest Neighbors (KNN)

**Task 3:**

Create a bar chart of accuracy results using matplotlib.