**MINI PROJECT 2:**

**Building a Classification Model with scikit-learn**

**Objective**

The primary objective of this mini-project is to build and evaluate classification models using scikit-learn on a real-world dataset. This will involve data preprocessing, model selection, training, evaluation, and potentially hyperparameter tuning.

**Dataset Selection**

For this project, I'll use the Iris dataset, a built-in dataset in scikit-learn. This dataset is a classic example for classification tasks and is well-suited for understanding the basics of machine learning.

**Data Loading and Exploration**

from sklearn.datasets import load\_iris

import pandas as pd

iris = load\_iris()

data = pd.DataFrame(iris.data, columns=iris.feature\_names)

data["target"] = iris.target

print(data.head())

**Data Preprocessing**

Since the Iris dataset is relatively clean, with no missing values or categorical features, minimal preprocessing is needed.

**Data Splitting**

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

X = data.drop("target", axis=1)

y = data["target"]

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

**Model Selection and Training**

Let’s train two classification models: Logistic Regression and Decision Tree.

from sklearn.linear\_model import LogisticRegression

from sklearn.tree import DecisionTreeClassifier

# Logistic Regression

logreg = LogisticRegression()

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

# Decision Tree

dt = DecisionTreeClassifier()

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

**Model Evaluation**

from sklearn.metrics import accuracy\_score, precision\_score, recall\_score, f1\_score, roc\_auc\_score

# Logistic Regression

y\_pred\_logreg = logreg.predict(X\_test)

accuracy\_logreg = accuracy\_score(y\_test, y\_pred\_logreg)

precision\_logreg = precision\_score(y\_test, y\_pred\_logreg, average="weighted")

recall\_logreg = recall\_score(y\_test, y\_pred\_logreg, average="weighted")

f1\_logreg = f1\_score(y\_test, y\_pred\_logreg, average="weighted")

roc\_auc\_logreg = roc\_auc\_score(y\_test, logreg.predict\_proba(X\_test)[:, 1])

# Decision Tree

y\_pred\_dt = dt.predict(X\_test)

accuracy\_dt = accuracy\_score(y\_test, y\_pred\_dt)

precision\_dt = precision\_score(y\_test, y\_pred\_dt, average="weighted")

recall\_dt = recall\_score(y\_test, y\_pred\_dt, average="weighted")

f1\_dt = f1\_score(y\_test, y\_pred\_dt, average="weighted")

roc\_auc\_dt = roc\_auc\_score(y\_test, dt.predict\_proba(X\_test)[:, 1])

print("Logistic Regression:")

print("Accuracy:", accuracy\_logreg)

print("Precision:", precision\_logreg)

print("Recall:", recall\_logreg)

print("F1-score:", f1\_logreg)

print("ROC AUC:", roc\_auc\_logreg)

print("\nDecision Tree:")

print("Accuracy:", accuracy\_dt)

print("Precision:", precision\_dt)

print("Recall:", recall\_dt)

print("F1-score:", f1\_dt)

print("ROC AUC:", roc\_auc\_dt)

**Conclusion**

Based on the evaluation metrics, you can compare the performance of Logistic Regression and Decision Tree. You can also explore other classification algorithms and experiment with different hyperparameters to find the best model for your specific dataset.