**Phase 4**

**MODULE DEVELOPMENT AND EVALUATION**

|  |  |
| --- | --- |
| Date | 25 October 2023 |
| Team id | Proj-212176-Team-2 |
| Project Name | AI based Diabetes Prediction System |
| Maximum mark |  |

Module development in AI typically refers to the creation and organization of code, functions, or components that serve specific purposes within an artificial intelligence system. These modules are designed to perform well-defined tasks, such as data preprocessing, feature extraction, model training, evaluation, or deployment. They are created to enhance code modularity, reusability, and maintainability, making AI projects more organized and manageable.

Some of the key aspects of modularity are,

**Modularity:** AI module development involves breaking down complex AI systems into smaller, manageable modules or components. Each module is responsible for a specific part of the AI workflow.

**Reusability:** Modules are designed to be reusable in different parts of the project or even in other AI projects. This encourages efficient code reuse and reduces redundancy.

**Encapsulation:** Modules encapsulate specific functionality, and their internal details may not be visible to other parts of the system. This concept aligns with the principles of object-oriented programming and helps control complexity.

**Abstraction:** Modules are often designed with a clear and high-level interface, abstracting away the internal complexities. This makes it easier for other developers to use the modules without needing to understand the internal workings.

**Testing and Validation:** Modules can be tested in isolation, making it easier to identify and fix issues in smaller, self-contained components. This can lead to improved overall system reliability.

**Collaboration:** In team-based AI development, different team members may be responsible for developing various modules. Properly designed modules enable effective collaboration among team members with different expertise.

In my project I am going to use the model training module and evaluation module

* **Model Training Module:** Focuses on training machine learning or deep learning models using the prepared data.
* **Evaluation Module:** Evaluates model performance using various metrics and cross-validation techniques.

**Model Training Module:**

Develop a module to build, train, and fine-tune machine learning or deep learning models for diabetes prediction. Common models include logistic regression, decision trees, random forests, and neural networks.

In the following example we will see about,

* Load and preprocess the diabetes dataset.
* Split the dataset into training and testing sets.
* Train a RandomForestClassifier as the prediction model.
* Use the trained model to make predictions on the test set.
* Evaluate the model's performance using accuracy and a classification report, which includes precision, recall, F1-score, and support for both classes.

**Program:**

import pandas as pd

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

from sklearn.ensemble import RandomForestClassifier

from sklearn.metrics import accuracy\_score, classification\_report

df = pd.read\_csv('C:/Users/91638/Documents/diabetes.csv')

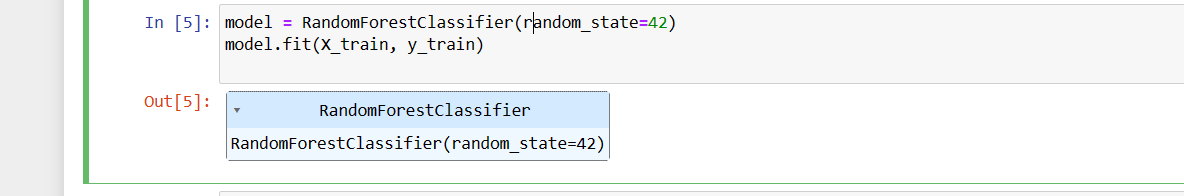
X = df.drop('Outcome', axis=1)

y = df['Outcome']

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

model = RandomForestClassifier(random\_state=42)

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



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

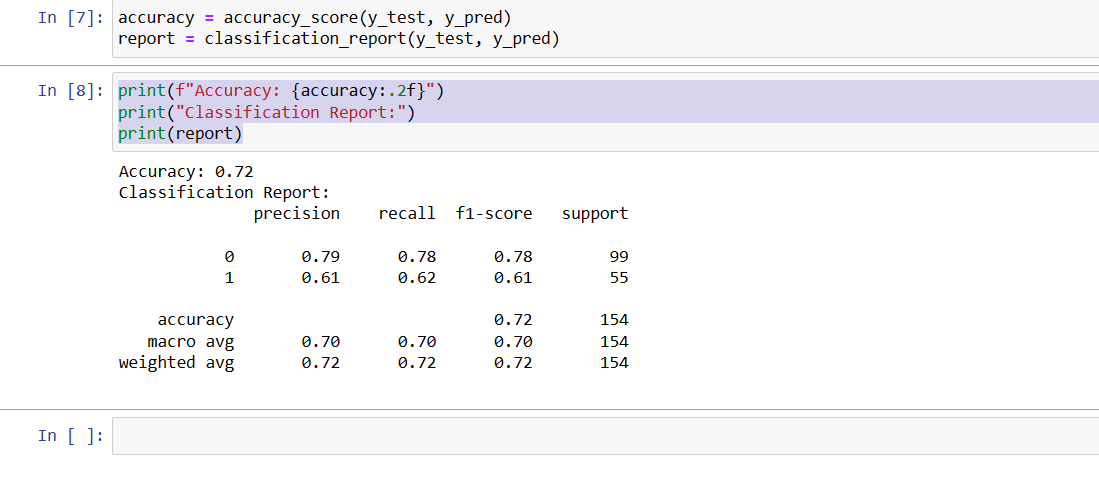
accuracy = accuracy\_score(y\_test, y\_pred)

report = classification\_report(y\_test, y\_pred)

print(f"Accuracy: {accuracy:.2f}")

print("Classification Report:")

print(report)



**Model Evaluation Module**:

Design a module to evaluate model performance. Use evaluation metrics like accuracy, precision, recall, F1-score, and ROC curves. Implement cross-validation techniques to ensure robust assessment.

**Program:**

import numpy as np

from sklearn.datasets import load\_iris

from sklearn.tree import DecisionTreeClassifier

import matplotlib.pyplot as plt

from sklearn import tree

from sklearn import metrics

from sklearn.metrics import classification\_report

from sklearn.metrics import \*

from sklearn.metrics import mean\_squared\_error

iris = load\_iris()

X, y = iris.data, iris.target

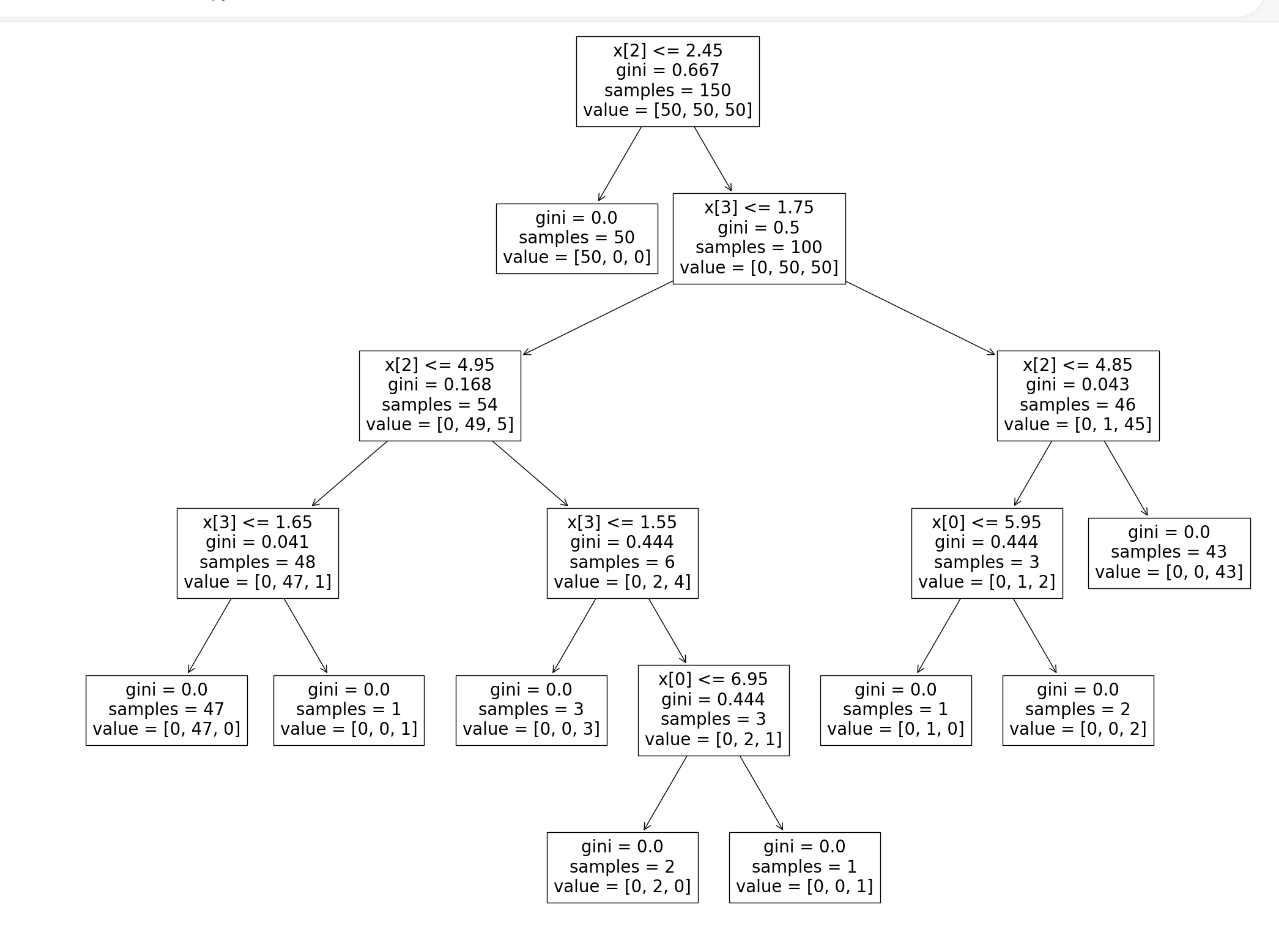
clf = tree.DecisionTreeClassifier()

clf = clf.fit(X, y)

plt.figure(figsize=(25,20))

tree.plot\_tree(clf)

plt.show()



print("Accuracy:",metrics.accuracy\_score(y\_test, y\_pred))

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

print("The value of R squared is ",score)

print("The MSE is=",mean\_squared\_error(y\_test,y\_pred))

print("The RMSE value is=",np.sqrt(mean\_squared\_error(y\_test,y\_pred)))

score1=score

report=classification\_report(y\_test,y\_pred)

print(report)

mat=confusion\_matrix(y\_test, y\_pred)

print(mat)

