LAB # 05

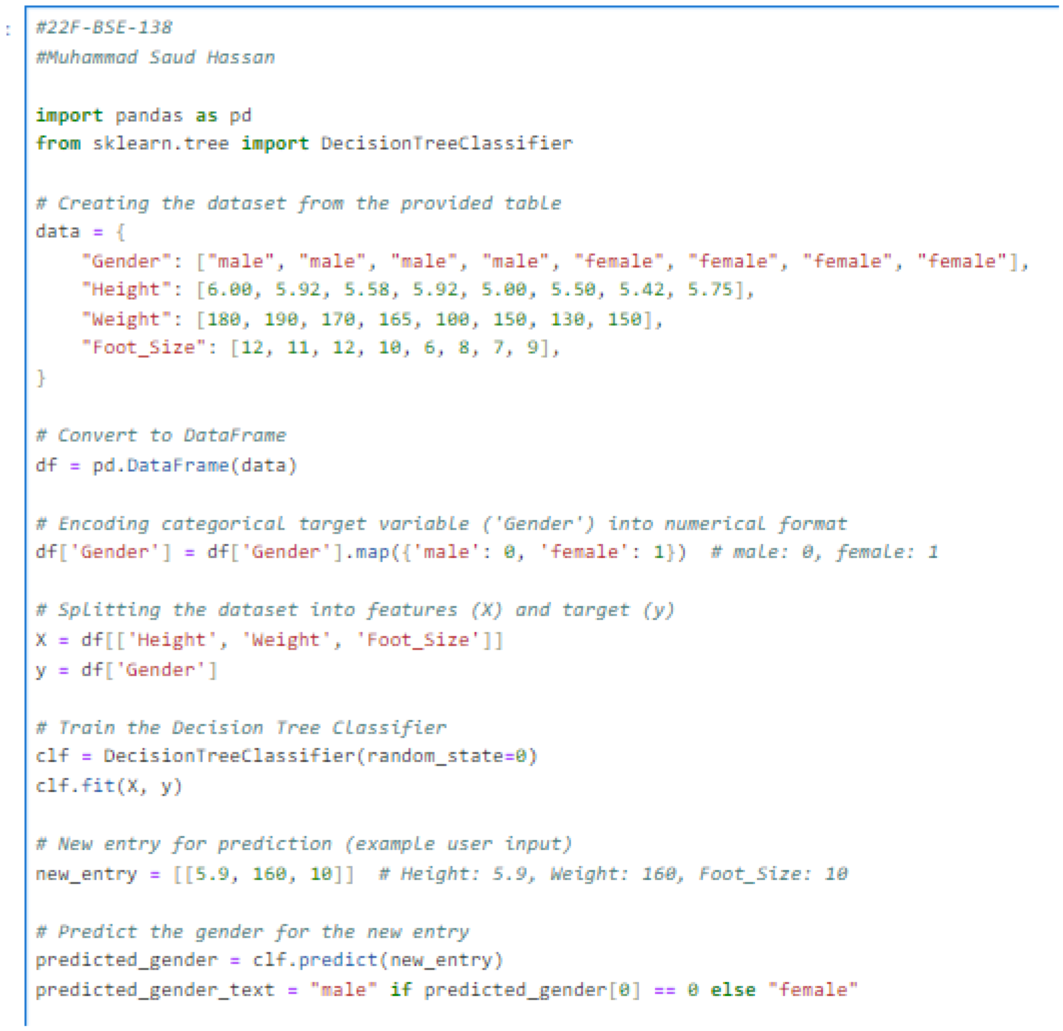
SUPERVISED LEARNING (DECISION TREE)

OBJECTIVE

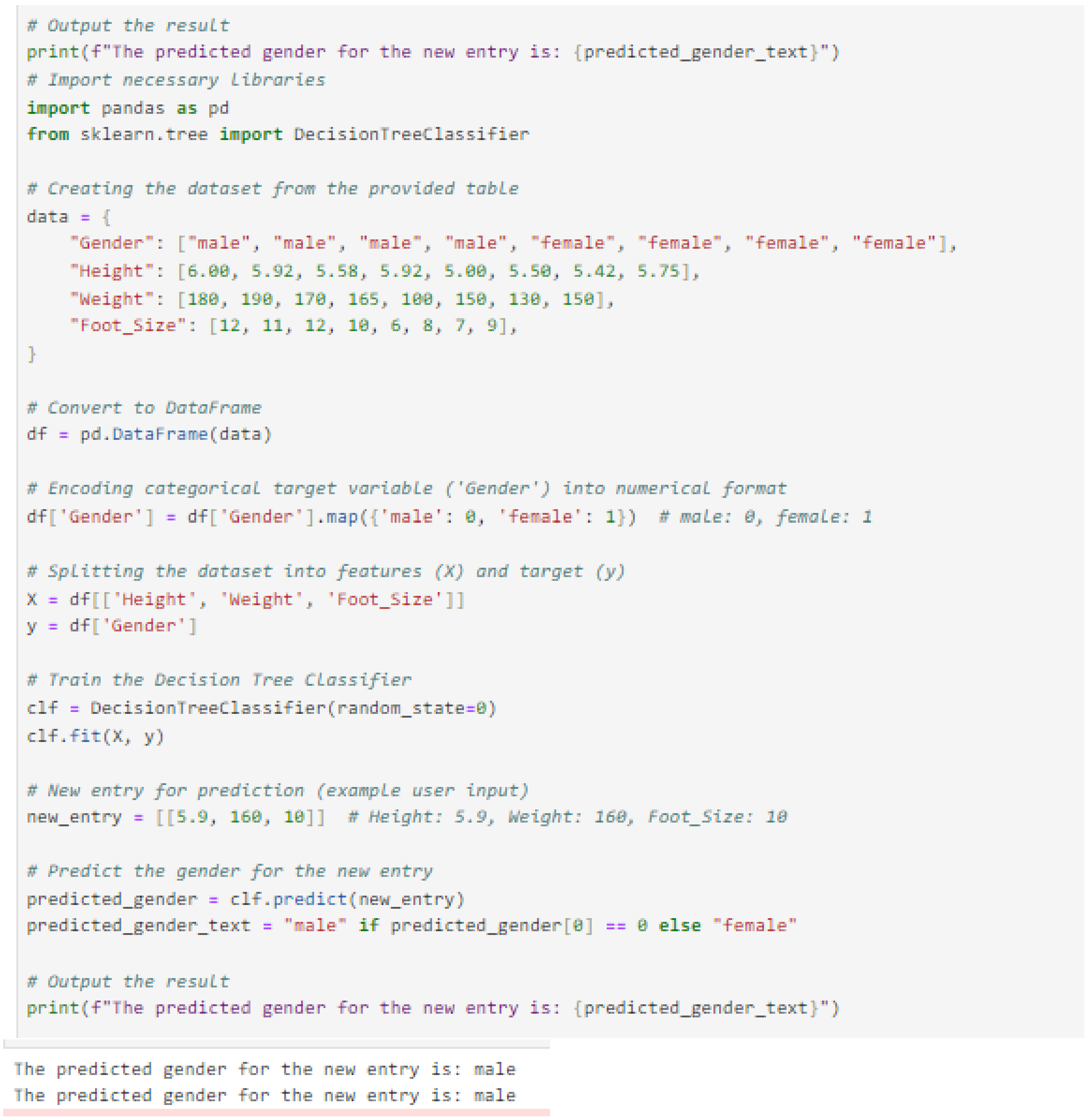
Implementing supervised learning, DTS algorithm for training, testing and classification.

Lab Tasks

1. Implement the Decision tree algorithm on the data given in the table and predict the new entry entered by the user.



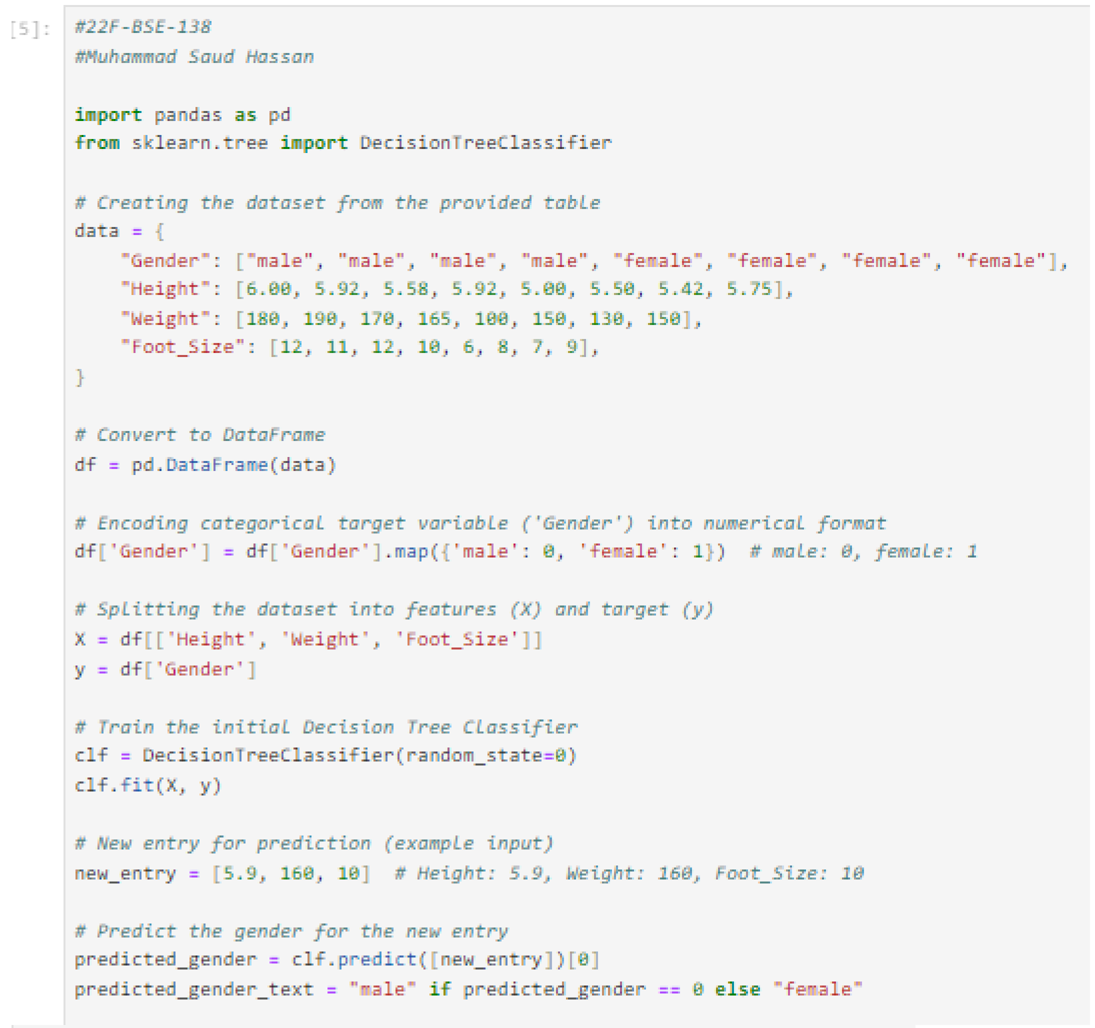
```
#22F-BSE-138
#Muhammad Saud Hassan

import pandas as pd
from sklearn.tree import DecisionTreeClassifier

# Creating the dataset from the provided table
data = {
    "Gender": ["male", "male", "male", "male", "female", "female", "female", "female"],
    "Height": [6.00, 5.92, 5.58, 5.92, 5.00, 5.50, 5.42, 5.75],
    "Weight": [180, 190, 170, 165, 100, 150, 130, 150],
    "Foot_Size": [12, 11, 12, 10, 6, 8, 7, 9],
}

# Convert to DataFrame
df = pd.DataFrame(data)

# Encoding categorical target variable ('Gender') into numerical format
df['Gender'] = df['Gender'].map({'male': 0, 'female': 1})  # male: 0, female: 1

# Splitting the dataset into features (X) and target (y)
X = df[['Height', 'Weight', 'Foot_Size']]
y = df['Gender']

# Train the Decision Tree Classifier
clf = DecisionTreeClassifier(random_state=0)
clf.fit(X, y)

# New entry for prediction (example user input)
new_entry = [[5.9, 160, 10]]  # Height: 5.9, Weight: 160, Foot_Size: 10

# Predict the gender for the new entry
predicted_gender = clf.predict(new_entry)
predicted_gender_text = "male" if predicted_gender[0] == 0 else "female"
```



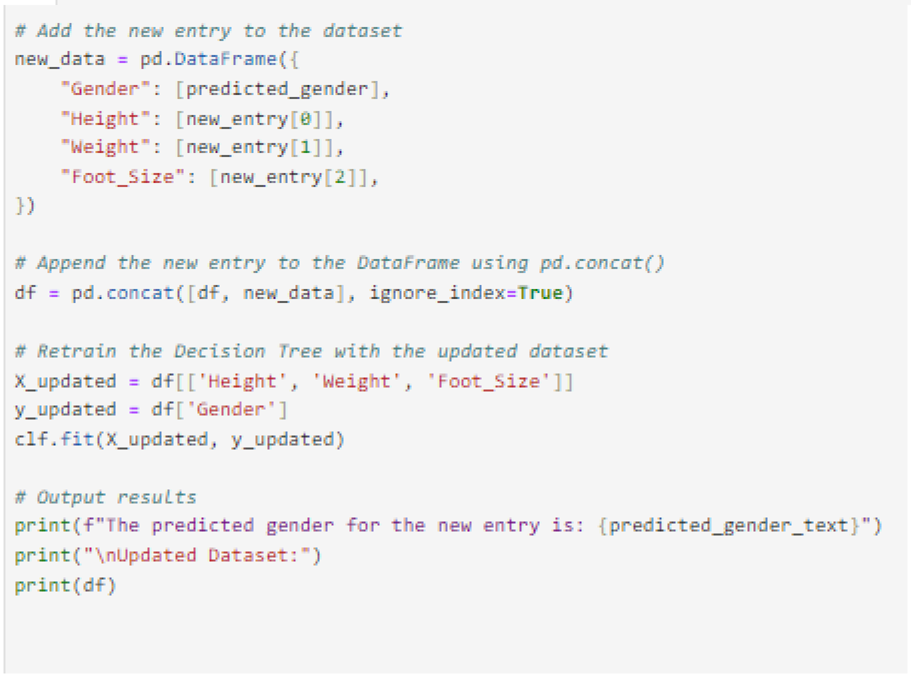
```python
# Output the result
print(f"The predicted gender for the new entry is: {predicted_gender_text}")
# Import necessary libraries
import pandas as pd
from sklearn.tree import DecisionTreeClassifier

# Creating the dataset from the provided table
data = {
    "Gender": ["male", "male", "male", "male", "female", "female", "female", "female"],
    "Height": [6.00, 5.92, 5.58, 5.92, 5.00, 5.50, 5.42, 5.75],
    "Weight": [180, 190, 170, 165, 100, 150, 130, 150],
    "Foot_Size": [12, 11, 12, 10, 6, 8, 7, 9],
}

# Convert to DataFrame
df = pd.DataFrame(data)

# Encoding categorical target variable ('Gender') into numerical format
df['Gender'] = df['Gender'].map({'male': 0, 'female': 1})  # male: 0, female: 1

# Splitting the dataset into features (X) and target (y)
X = df[['Height', 'Weight', 'Foot_Size']]
y = df['Gender']

# Train the Decision Tree Classifier
clf = DecisionTreeClassifier(random_state=0)
clf.fit(X, y)

# New entry for prediction (example user input)
new_entry = [[5.9, 160, 10]]  # Height: 5.9, Weight: 160, Foot_Size: 10

# Predict the gender for the new entry
predicted_gender = clf.predict(new_entry)
predicted_gender_text = "male" if predicted_gender[0] == 0 else "female"

# Output the result
print(f"The predicted gender for the new entry is: {predicted_gender_text}")
```

```
The predicted gender for the new entry is: male
The predicted gender for the new entry is: male
```

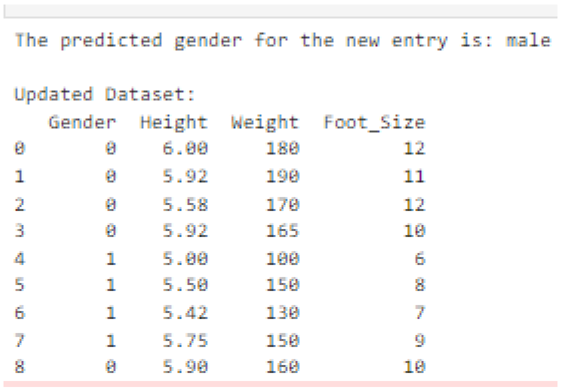
2. Implement Decision Tree using table. 1 in such a way that the new entry becomes the part of the given dataset.



```
[5]: #22F-BSE-138
     #Muhammad Saud Hassan

     import pandas as pd
     from sklearn.tree import DecisionTreeClassifier

     # Creating the dataset from the provided table
     data = {
         "Gender": ["male", "male", "male", "male", "female", "female", "female", "female"],
         "Height": [6.00, 5.92, 5.58, 5.92, 5.00, 5.50, 5.42, 5.75],
         "Weight": [180, 190, 170, 165, 100, 150, 130, 150],
         "Foot_Size": [12, 11, 12, 10, 6, 8, 7, 9],
     }

     # Convert to DataFrame
     df = pd.DataFrame(data)

     # Encoding categorical target variable ('Gender') into numerical format
     df['Gender'] = df['Gender'].map({'male': 0, 'female': 1})  # male: 0, female: 1

     # Splitting the dataset into features (X) and target (y)
     X = df[['Height', 'Weight', 'Foot_Size']]
     y = df['Gender']

     # Train the initial Decision Tree Classifier
     clf = DecisionTreeClassifier(random_state=0)
     clf.fit(X, y)

     # New entry for prediction (example input)
     new_entry = [5.9, 160, 10]  # Height: 5.9, Weight: 160, Foot_Size: 10

     # Predict the gender for the new entry
     predicted_gender = clf.predict([new_entry])[0]
     predicted_gender_text = "male" if predicted_gender == 0 else "female"
```

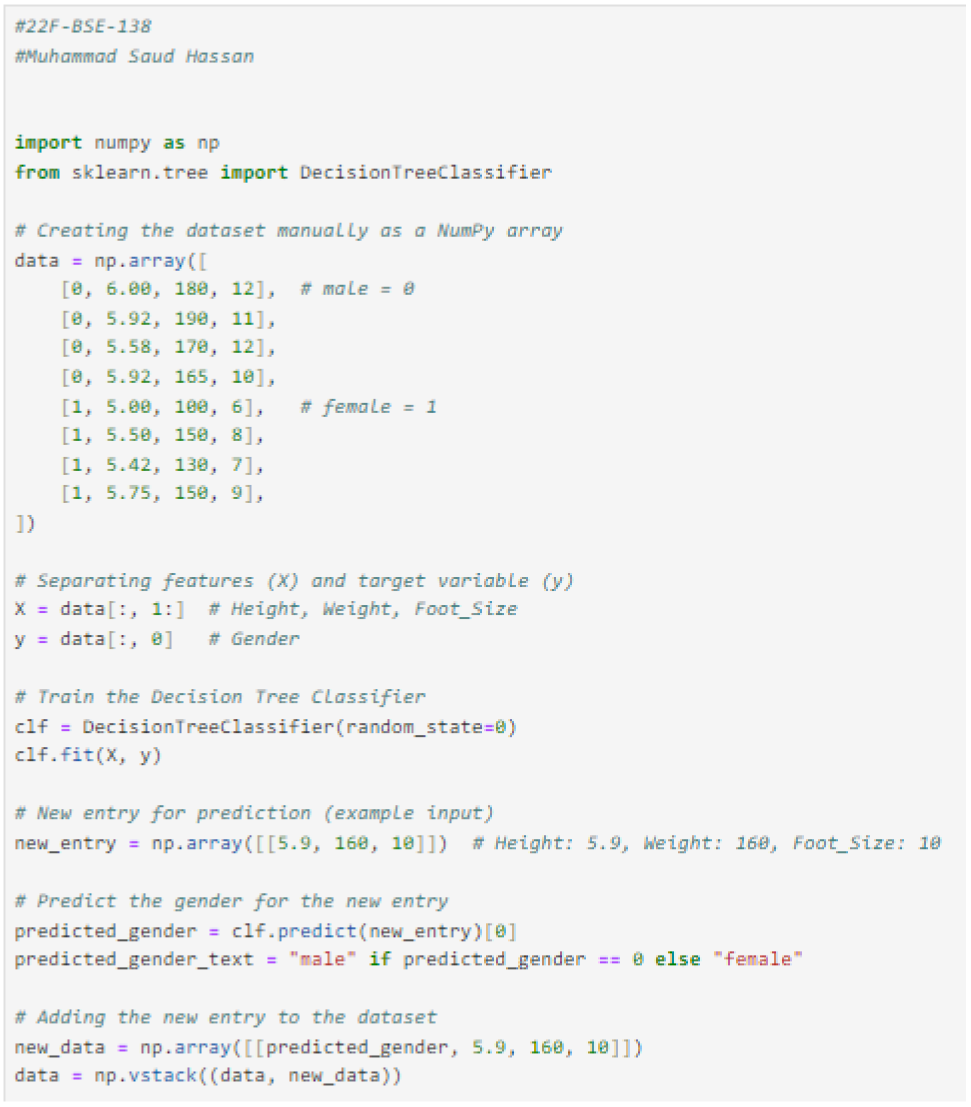


```
# Add the new entry to the dataset
new_data = pd.DataFrame({
    "Gender": [predicted_gender],
    "Height": [new_entry[0]],
    "Weight": [new_entry[1]],
    "Foot_Size": [new_entry[2]],
})

# Append the new entry to the DataFrame using pd.concat()
df = pd.concat([df, new_data], ignore_index=True)

# Retrain the Decision Tree with the updated dataset
X_updated = df[['Height', 'Weight', 'Foot_Size']]
y_updated = df['Gender']
clf.fit(X_updated, y_updated)

# Output results
print(f"The predicted gender for the new entry is: {predicted_gender_text}")
print("\nUpdated Dataset:")
print(df)
```

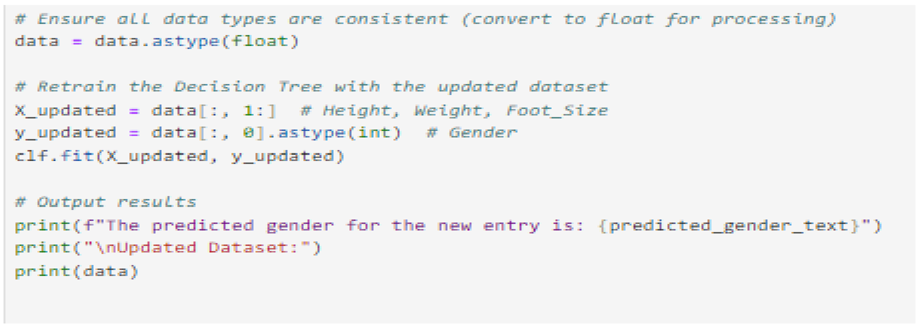


```
The predicted gender for the new entry is: male

Updated Dataset:
   Gender  Height  Weight  Foot_Size
0       0    6.00     180         12
1       0    5.92     190         11
2       0    5.58     170         12
3       0    5.92     165         10
4       1    5.00     100          6
5       1    5.50     150          8
6       1    5.42     130          7
7       1    5.75     150          9
8       0    5.90     160         10
```

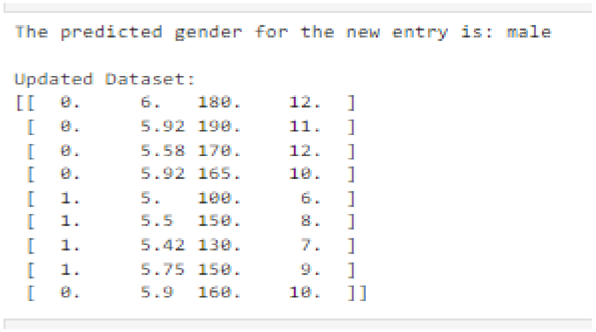
3. Implement Decision Tree using table. 1 without the use of Pandas library. You can use numpy.



```python
#22F-BSE-138
#Muhammad Saud Hassan


import numpy as np
from sklearn.tree import DecisionTreeClassifier

# Creating the dataset manually as a NumPy array
data = np.array([
    [0, 6.00, 180, 12],  # male = 0
    [0, 5.92, 190, 11],
    [0, 5.58, 170, 12],
    [0, 5.92, 165, 10],
    [1, 5.00, 100, 6],   # female = 1
    [1, 5.50, 150, 8],
    [1, 5.42, 130, 7],
    [1, 5.75, 150, 9],
])

# Separating features (X) and target variable (y)
X = data[:, 1:]  # Height, Weight, Foot_Size
y = data[:, 0]   # Gender

# Train the Decision Tree Classifier
clf = DecisionTreeClassifier(random_state=0)
clf.fit(X, y)

# New entry for prediction (example input)
new_entry = np.array([[5.9, 160, 10]])  # Height: 5.9, Weight: 160, Foot_Size: 10

# Predict the gender for the new entry
predicted_gender = clf.predict(new_entry)[0]
predicted_gender_text = "male" if predicted_gender == 0 else "female"

# Adding the new entry to the dataset
new_data = np.array([[predicted_gender, 5.9, 160, 10]])
data = np.vstack((data, new_data))
```



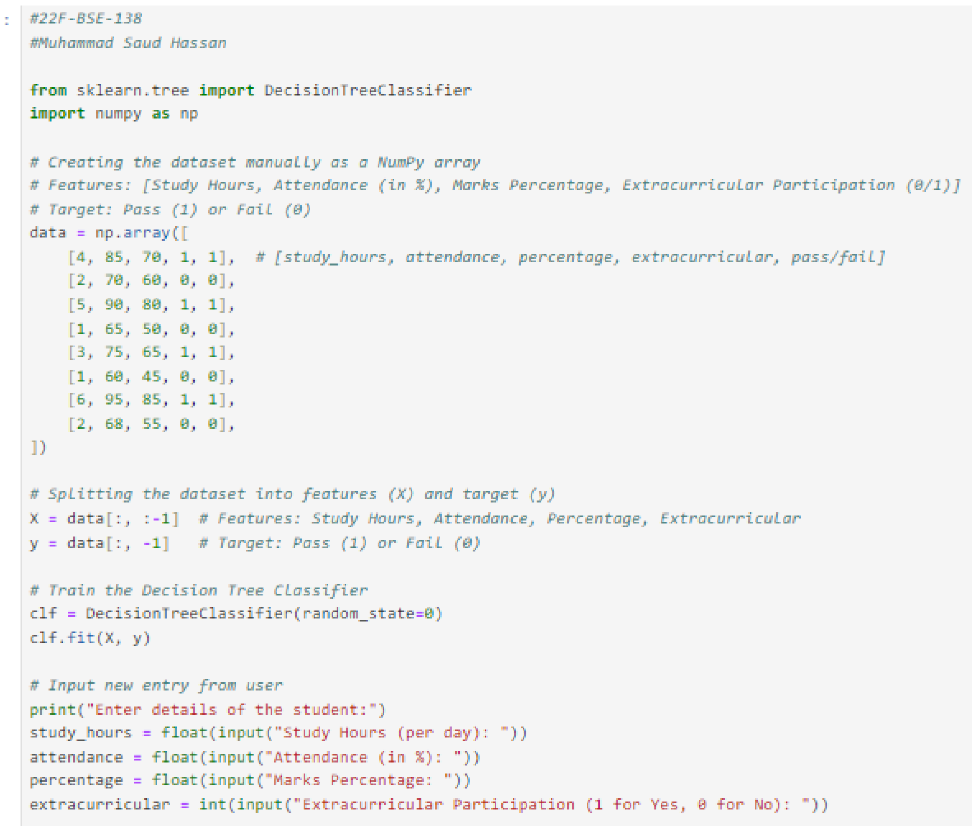
```
# Ensure all data types are consistent (convert to float for processing)
data = data.astype(float)

# Retrain the Decision Tree with the updated dataset
X_updated = data[:, 1:]  # Height, Weight, Foot_Size
y_updated = data[:, 0].astype(int)  # Gender
clf.fit(X_updated, y_updated)

# Output results
print(f"The predicted gender for the new entry is: {predicted_gender_text}")
print("\nUpdated Dataset:")
print(data)
```



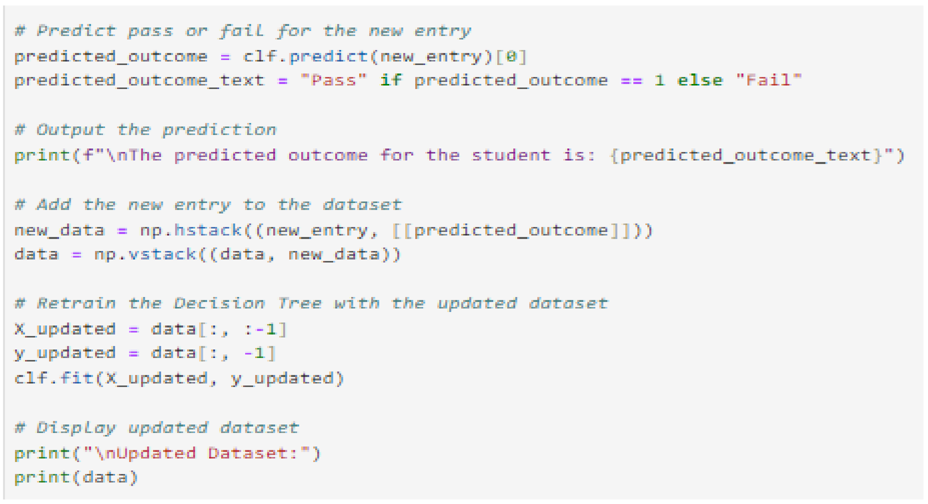
```
The predicted gender for the new entry is: male

Updated Dataset:
[[  0.      6.    180.     12.   ]
 [  0.      5.92  190.     11.   ]
 [  0.      5.58  170.     12.   ]
 [  0.      5.92  165.     10.   ]
 [  1.      5.    100.      6.   ]
 [  1.      5.5   150.      8.   ]
 [  1.      5.42  130.      7.   ]
 [  1.      5.75  150.      9.   ]
 [  0.      5.9   160.     10.   ]]
```

Home Tasks

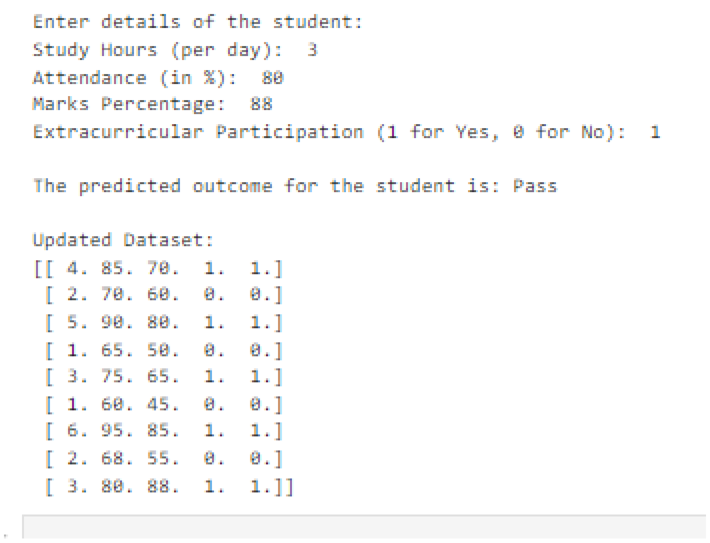
- A student will pass or fail based on their study hours, attendance, percentage and participation in extra circular activities. Implement the Decision tree algorithm on the data given and predict the new entry entered by the user.



```
#22F-BSE-138
#Muhammad Saud Hassan

from sklearn.tree import DecisionTreeClassifier
import numpy as np

# Creating the dataset manually as a NumPy array
# Features: [Study Hours, Attendance (in %), Marks Percentage, Extracurricular Participation (0/1)]
# Target: Pass (1) or Fail (0)
data = np.array([
    [4, 85, 70, 1, 1],  # [study_hours, attendance, percentage, extracurricular, pass/fail]
    [2, 70, 60, 0, 0],
    [5, 90, 80, 1, 1],
    [1, 65, 50, 0, 0],
    [3, 75, 65, 1, 1],
    [1, 60, 45, 0, 0],
    [6, 95, 85, 1, 1],
    [2, 68, 55, 0, 0],
])

# Splitting the dataset into features (X) and target (y)
X = data[:, :-1]  # Features: Study Hours, Attendance, Percentage, Extracurricular
y = data[:, -1]   # Target: Pass (1) or Fail (0)

# Train the Decision Tree Classifier
clf = DecisionTreeClassifier(random_state=0)
clf.fit(X, y)

# Input new entry from user
print("Enter details of the student:")
study_hours = float(input("Study Hours (per day): "))
attendance = float(input("Attendance (in %): "))
percentage = float(input("Marks Percentage: "))
extracurricular = int(input("Extracurricular Participation (1 for Yes, 0 for No): "))
```

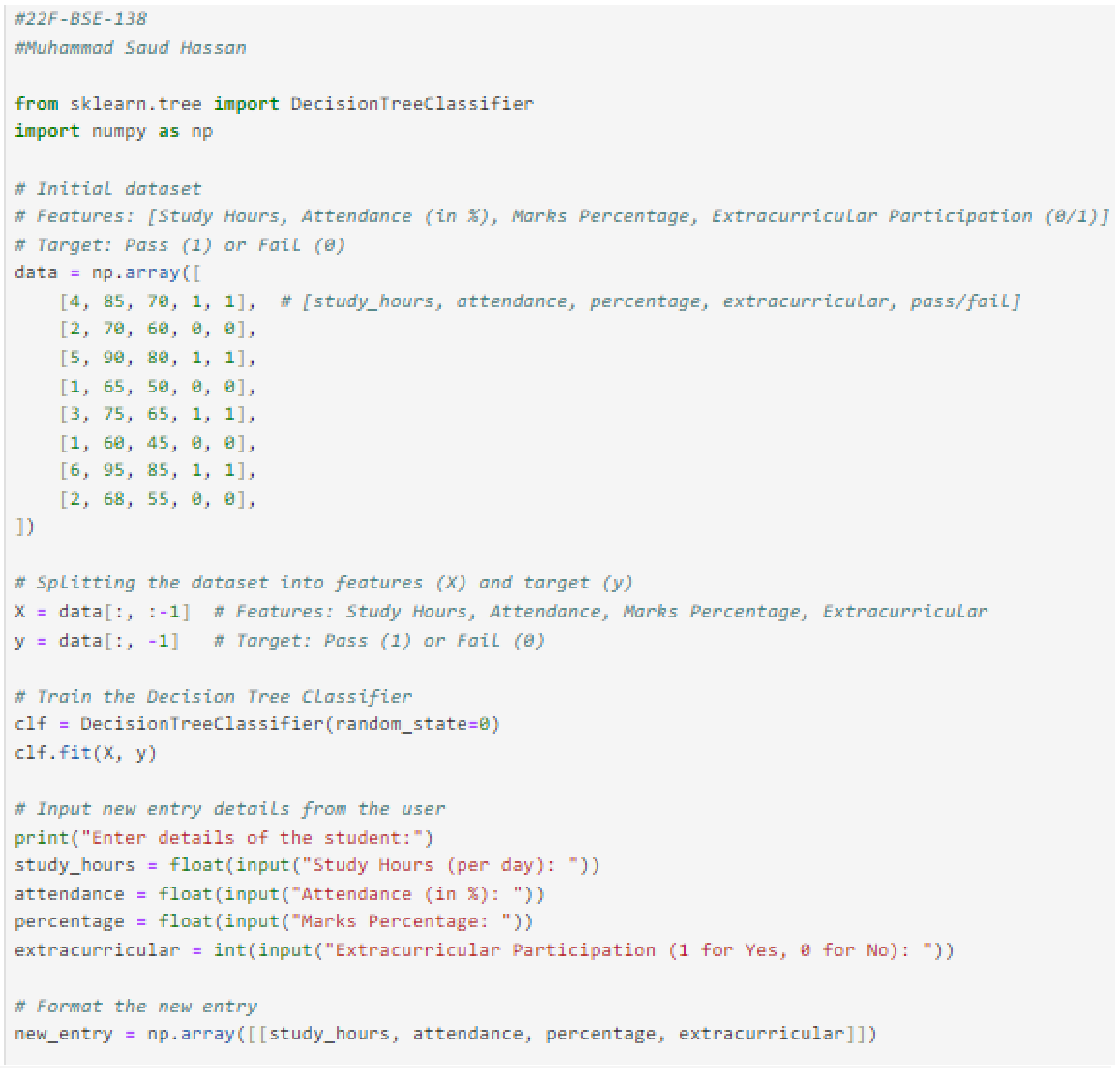


```python
# Predict pass or fail for the new entry
predicted_outcome = clf.predict(new_entry)[0]
predicted_outcome_text = "Pass" if predicted_outcome == 1 else "Fail"

# Output the prediction
print(f"\nThe predicted outcome for the student is: {predicted_outcome_text}")

# Add the new entry to the dataset
new_data = np.hstack((new_entry, [[predicted_outcome]]))
data = np.vstack((data, new_data))

# Retrain the Decision Tree with the updated dataset
X_updated = data[:, :-1]
y_updated = data[:, -1]
clf.fit(X_updated, y_updated)

# Display updated dataset
print("\nUpdated Dataset:")
print(data)
```

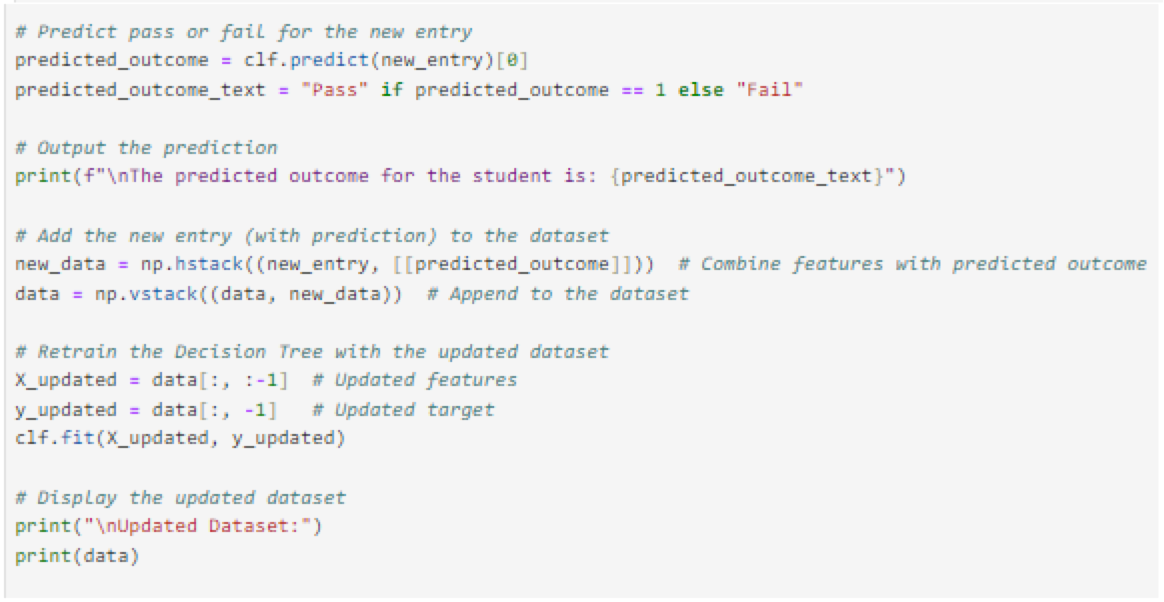


```
Enter details of the student:
Study Hours (per day):  3
Attendance (in %):  80
Marks Percentage:  88
Extracurricular Participation (1 for Yes, 0 for No):  1

The predicted outcome for the student is: Pass

Updated Dataset:
[[ 4. 85. 70.  1.  1.]
 [ 2. 70. 60.  0.  0.]
 [ 5. 90. 80.  1.  1.]
 [ 1. 65. 50.  0.  0.]
 [ 3. 75. 65.  1.  1.]
 [ 1. 60. 45.  0.  0.]
 [ 6. 95. 85.  1.  1.]
 [ 2. 68. 55.  0.  0.]
 [ 3. 80. 88.  1.  1.]]
```

- A student will pass or fail based on their study hours, attendance, percentage and participation in extra circular activities. Implement Decision Tree using given data in such a way that the new entry becomes the part of the given dataset.



```
#22F-BSE-138
#Muhammad Saud Hassan

from sklearn.tree import DecisionTreeClassifier
import numpy as np

# Initial dataset
# Features: [Study Hours, Attendance (in %), Marks Percentage, Extracurricular Participation (0/1)]
# Target: Pass (1) or Fail (0)
data = np.array([
    [4, 85, 70, 1, 1],  # [study_hours, attendance, percentage, extracurricular, pass/fail]
    [2, 70, 60, 0, 0],
    [5, 90, 80, 1, 1],
    [1, 65, 50, 0, 0],
    [3, 75, 65, 1, 1],
    [1, 60, 45, 0, 0],
    [6, 95, 85, 1, 1],
    [2, 68, 55, 0, 0],
])

# Splitting the dataset into features (X) and target (y)
X = data[:, :-1]  # Features: Study Hours, Attendance, Marks Percentage, Extracurricular
y = data[:, -1]   # Target: Pass (1) or Fail (0)

# Train the Decision Tree Classifier
clf = DecisionTreeClassifier(random_state=0)
clf.fit(X, y)

# Input new entry details from the user
print("Enter details of the student:")
study_hours = float(input("Study Hours (per day): "))
attendance = float(input("Attendance (in %): "))
percentage = float(input("Marks Percentage: "))
extracurricular = int(input("Extracurricular Participation (1 for Yes, 0 for No): "))

# Format the new entry
new_entry = np.array([[study_hours, attendance, percentage, extracurricular]])
```

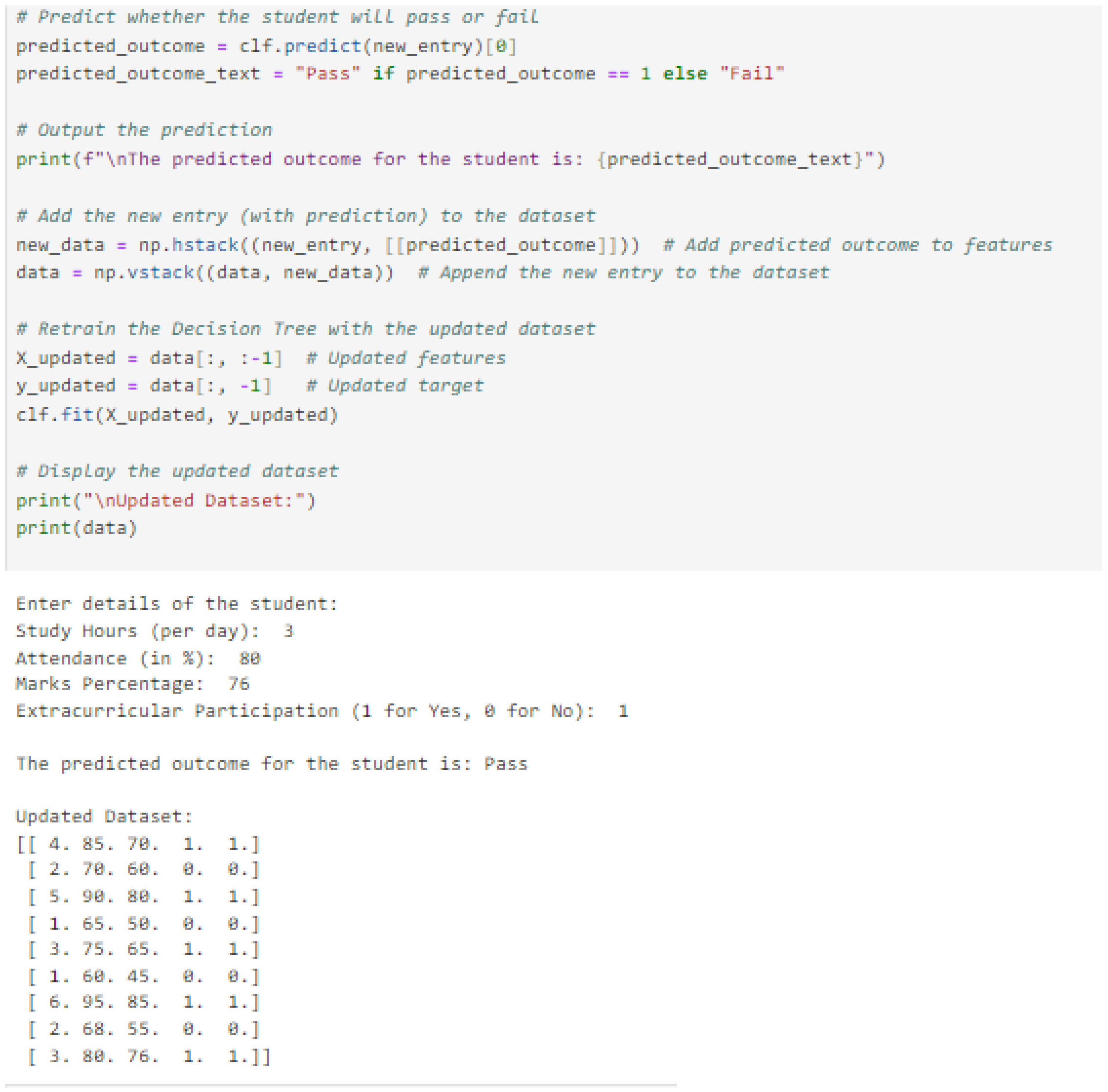


```
# Predict pass or fail for the new entry
predicted_outcome = clf.predict(new_entry)[0]
predicted_outcome_text = "Pass" if predicted_outcome == 1 else "Fail"

# Output the prediction
print(f"\nThe predicted outcome for the student is: {predicted_outcome_text}")

# Add the new entry (with prediction) to the dataset
new_data = np.hstack((new_entry, [[predicted_outcome]]))  # Combine features with predicted outcome
data = np.vstack((data, new_data))  # Append to the dataset

# Retrain the Decision Tree with the updated dataset
X_updated = data[:, :-1]  # Updated features
y_updated = data[:, -1]   # Updated target
clf.fit(X_updated, y_updated)

# Display the updated dataset
print("\nUpdated Dataset:")
print(data)
```

```
Enter details of the student:
Study Hours (per day):  3
Attendance (in %):  80
Marks Percentage:  78
Extracurricular Participation (1 for Yes, 0 for No):  1

The predicted outcome for the student is: Pass

Updated Dataset:
[[ 4. 85. 70.  1.  1.]
 [ 2. 70. 60.  0.  0.]
 [ 5. 90. 80.  1.  1.]
 [ 1. 65. 50.  0.  0.]
 [ 3. 75. 65.  1.  1.]
 [ 1. 60. 45.  0.  0.]
 [ 6. 95. 85.  1.  1.]
 [ 2. 68. 55.  0.  0.]
 [ 3. 80. 78.  1.  1.]]
```

- A student will pass or fail based on their study hours, attendance, percentage and participation in extra circular activities. Implement Decision Tree using given data without the use of Pandas library. You can use numpy.

```python
#22F-BSE-138
#Muhammad Saud Hassan

import numpy as np
from sklearn.tree import DecisionTreeClassifier

# Initial dataset as a NumPy array
# Columns: [Study Hours, Attendance (%), Marks Percentage, Extracurricular Participation (1/0), Pass/Fail (1/0)]
data = np.array([
    [4, 85, 70, 1, 1],
    [2, 70, 60, 0, 0],
    [5, 90, 80, 1, 1],
    [1, 65, 50, 0, 0],
    [3, 75, 65, 1, 1],
    [1, 60, 45, 0, 0],
    [6, 95, 85, 1, 1],
    [2, 68, 55, 0, 0],
])

# Splitting features (X) and target (y)
X = data[:, :-1]  # Features: [Study Hours, Attendance, Marks Percentage, Extracurricular]
y = data[:, -1]   # Target: Pass (1) or Fail (0)

# Train the Decision Tree Classifier
clf = DecisionTreeClassifier(random_state=0)
clf.fit(X, y)

# Input new student details
print("Enter details of the student:")
study_hours = float(input("Study Hours (per day): "))
attendance = float(input("Attendance (in %): "))
percentage = float(input("Marks Percentage: "))
extracurricular = int(input("Extracurricular Participation (1 for Yes, 0 for No): "))

# Create a new entry for prediction
new_entry = np.array([[study_hours, attendance, percentage, extracurricular]])
```



```
# Predict whether the student will pass or fail
predicted_outcome = clf.predict(new_entry)[0]
predicted_outcome_text = "Pass" if predicted_outcome == 1 else "Fail"

# Output the prediction
print(f"\nThe predicted outcome for the student is: {predicted_outcome_text}")

# Add the new entry (with prediction) to the dataset
new_data = np.hstack((new_entry, [[predicted_outcome]]))  # Add predicted outcome to features
data = np.vstack((data, new_data))  # Append the new entry to the dataset

# Retrain the Decision Tree with the updated dataset
X_updated = data[:, :-1]  # Updated features
y_updated = data[:, -1]   # Updated target
clf.fit(X_updated, y_updated)

# Display the updated dataset
print("\nUpdated Dataset:")
print(data)
```

```
Enter details of the student:
Study Hours (per day):  3
Attendance (in %):  80
Marks Percentage:  76
Extracurricular Participation (1 for Yes, 0 for No):  1

The predicted outcome for the student is: Pass

Updated Dataset:
[[ 4. 85. 70.  1.  1.]
 [ 2. 70. 60.  0.  0.]
 [ 5. 90. 80.  1.  1.]
 [ 1. 65. 50.  0.  0.]
 [ 3. 75. 65.  1.  1.]
 [ 1. 60. 45.  0.  0.]
 [ 6. 95. 85.  1.  1.]
 [ 2. 68. 55.  0.  0.]
 [ 3. 80. 76.  1.  1.]]
```