Lab – 01: Introduction to Machine Learning and Python Environment

1. Introduction

Machine Learning (ML) is a branch of Artificial Intelligence that allows computers to learn patterns from data and make predictions without being explicitly programmed.
In this lab, we focus on:

- Setting up Python and Jupyter Notebook environment.
- Learning basic Python operations useful for ML.
- Loading, exploring, and visualizing datasets.
- Understanding the train-test split for ML models.

2. Experiments and Observations

2.1 Setting Up Python and Jupyter Notebook

Procedure:

1. Installed Python, Jupyter Notebook, and required libraries: `numpy`, `pandas`, `matplotlib`, `seaborn`, `scikit-learn`.
2. Verified installation by importing libraries in Jupyter Notebook.

Code:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split

print("All libraries are successfully installed!")
```

Observation:

```
All libraries are successfully installed!
```

2.2 Basic Python Operations for ML

Procedure:

Practiced Python concepts such as:

- Data types: int, float, string
- Lists, dictionaries, loops, functions
- Simple calculations using NumPy

Code :

```
# Data types
x = 10
y = 3.5
name = "Toufike_Ur"

# List operations
numbers = [1, 2, 3, 4]
numbers.append(5)

# Dictionary
student = {"name": "Toufike_Ur", "age": 24}
# Loop
for i in range(3):
    print(i)

# Function
def add(a, b):
    return a + b
print(add(5, 7))

# NumPy operations
arr = np.array([1, 2, 3, 4])
print("Array + 10:", arr + 10)
print("Mean of array:", np.mean(arr))
```

Observation:

```
0
1
2
12
Array + 10: [11 12 13 14]
Mean of array: 2.5
```

2.3 Loading and Exploring Datasets

Procedure:

- Loaded the Iris dataset using `pandas`.
- Displayed first few records and basic statistics.

Code:

```
from sklearn.datasets import load_iris

iris = load_iris()
df = pd.DataFrame(iris.data, columns=iris.feature_names)
df['target'] = iris.target

# Display first 5 records
print(df.head())

# Basic statistics
print(df.describe())
```

Observation :

```
   sepal length (cm)  sepal width (cm)  petal length (cm)  petal width (cm)  \
0                5.1               3.5                1.4               0.2
1                4.9               3.0                1.4               0.2
2                4.7               3.2                1.3               0.2
3                4.6               3.1                1.5               0.2
4                5.0               3.6                1.4               0.2

   target
0       0
1       0
2       0
3       0
4       0
       sepal length (cm)  sepal width (cm)  petal length (cm)  \
count         150.000000        150.000000         150.000000
mean            5.843333          3.057333           3.758000
std             0.828066          0.435866           1.765298
min             4.300000          2.000000           1.000000
25%             5.100000          2.800000           1.600000
50%             5.800000          3.000000           4.350000
75%             6.400000          3.300000           5.100000
max             7.900000          4.400000           6.900000

       petal width (cm)      target
count        150.000000  150.000000
mean           1.199333    1.000000
std            0.762238    0.819232
min            0.100000    0.000000
25%            0.300000    0.000000
50%            1.300000    1.000000
75%            1.800000    2.000000
```

2.4 Data Visualization

Procedure:

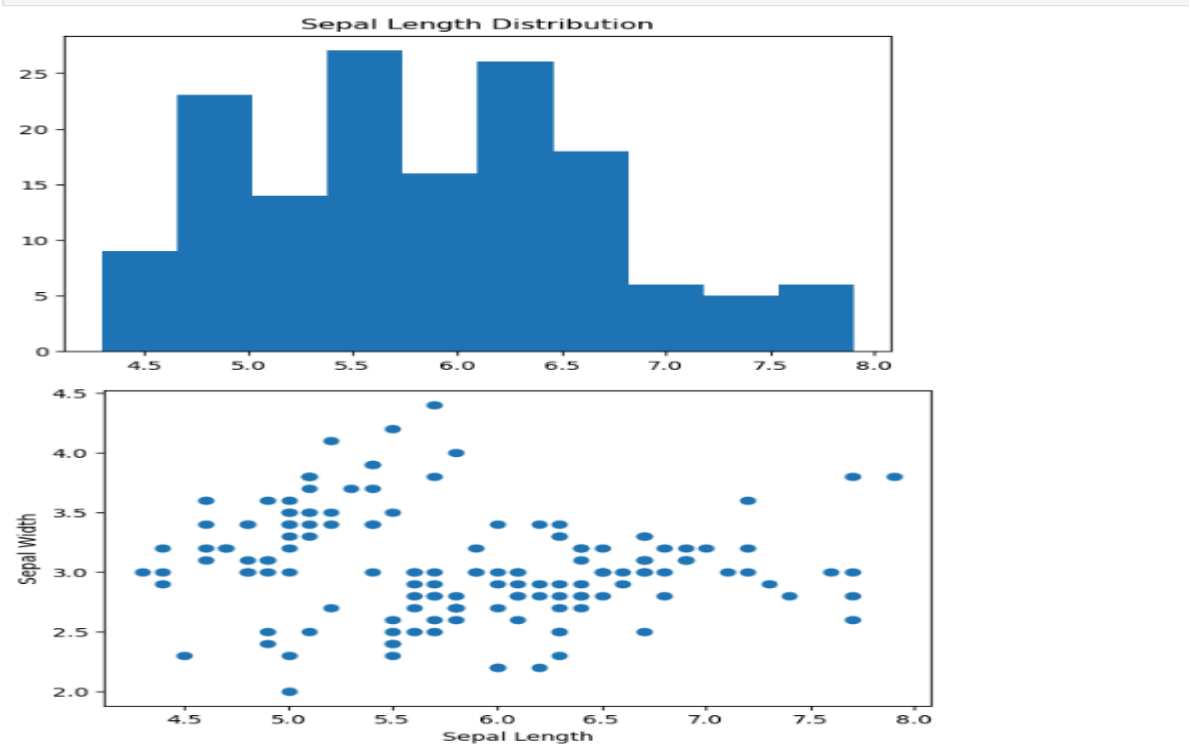
Visualized data distributions, scatter plots, and correlations using `matplotlib` and `seaborn`.

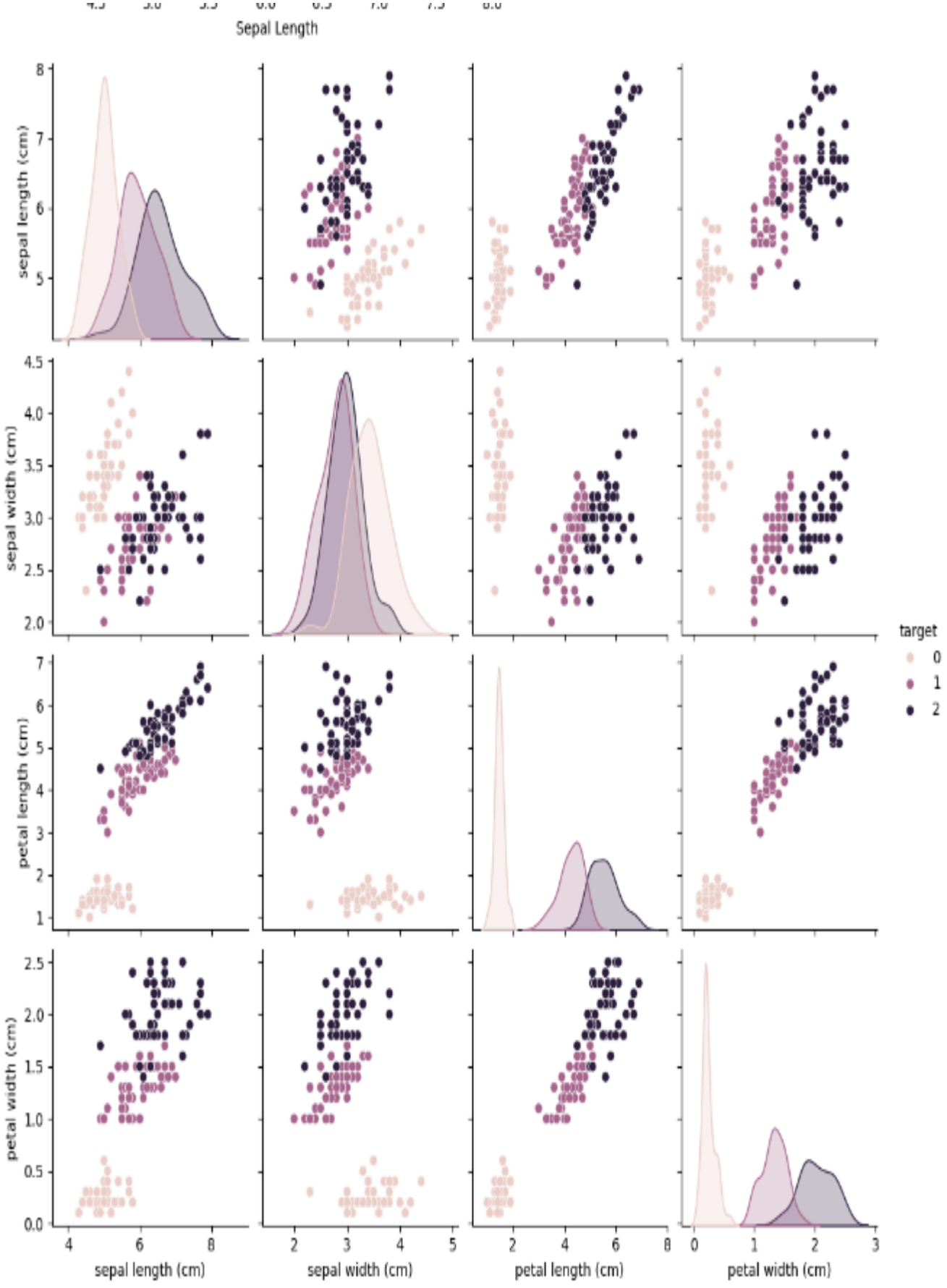
Code:

```
# Histogram
plt.hist(df['sepal length (cm)'])
plt.title("Sepal Length Distribution")
plt.show()

# Scatter plot
plt.scatter(df['sepal length (cm)'], df['sepal width (cm)'])
plt.xlabel("Sepal Length")
plt.ylabel("Sepal Width")
plt.show()
# Seaborn pairplot
sns.pairplot(df, hue='target')
plt.show()
```

Observation :




Sepal Length
sepal length (cm)
sepal width (cm)
petal length (cm)
petal width (cm)
target
0
1
2

2.5 Understanding Train-Test Split

Procedure:

Split the dataset into training (70%) and testing (30%) sets using `train_test_split`.

Code:

```
X = df.drop('target', axis=1)
y = df['target']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

print("Training set shape:", X_train.shape)
print("Testing set shape:", X_test.shape)
```

Observation :

```
Training set shape: (105, 4)
Testing set shape: (45, 4)
```

3. Conclusion

- Python and Jupyter Notebook were successfully installed and verified.
- Basic Python operations and NumPy calculations are essential for ML preprocessing.
- The Iris dataset was successfully loaded, explored, and visualized.
- Train-test split prepares data for ML model training and testing.