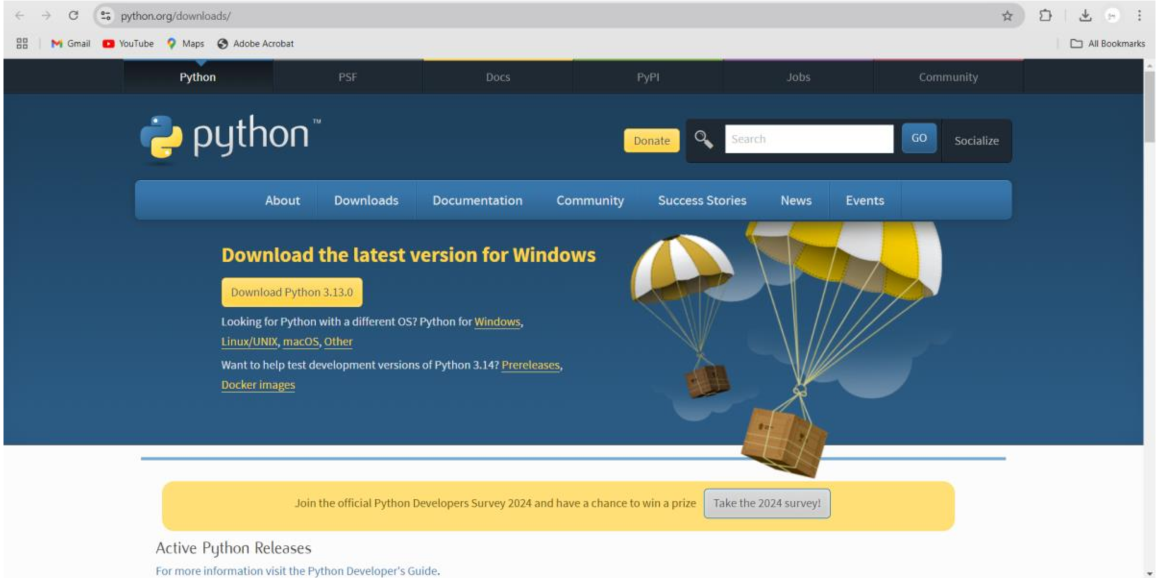
Machine Learning Lab - Practical Experiment using Scikit-learn and Pandas

1. Environment Setup

- Install Python if you haven't already (you can download it from Python's official site).



Download the latest version for Windows

Looking for Python with a different OS? Python for Windows, Linux/UNIX, macOS, Other

Want to help test development versions of Python 3.14? Prereleases, Docker images

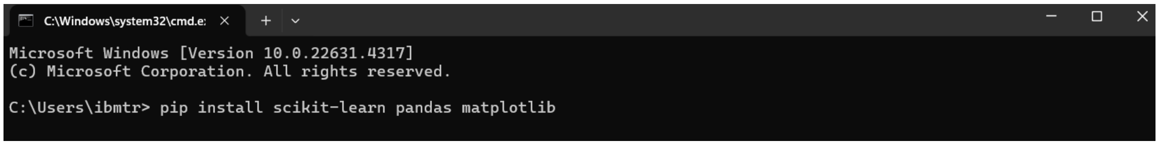
Join the official Python Developers Survey 2024 and have a chance to win a prize

Active Python Releases

For more information visit the Python Developer's Guide.

- Install Required Libraries: Run the following commands in your terminal or command prompt to install the necessary Python packages:

pip install scikit-learn pandas matplotlib

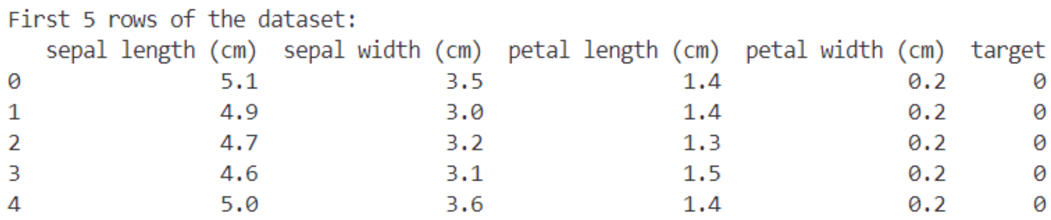


```
Microsoft Windows [Version 10.0.22631.4317]
(c) Microsoft Corporation. All rights reserved.

C:\Users\ibmtr> pip install scikit-learn pandas matplotlib
```

2. Data Preparation

- Loading the Dataset: You can use any dataset for this experiment. For demonstration, I will use a sample dataset such as the Iris dataset, which is built into Scikit-learn.

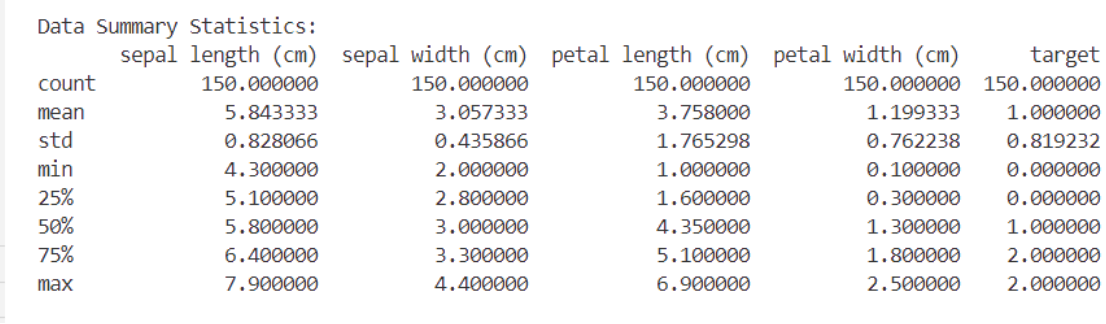
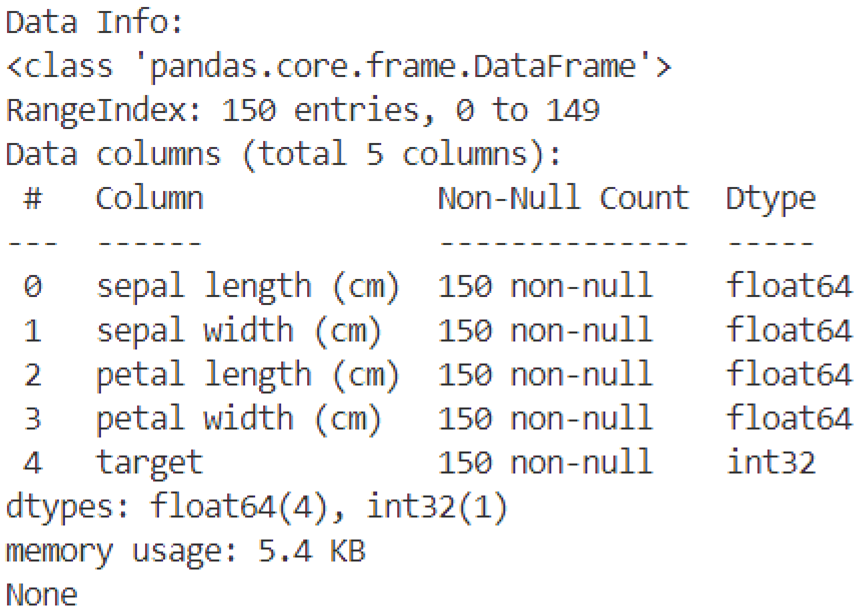
```
import pandas as pd
from sklearn.datasets import load_iris

# Load the iris dataset
iris = load_iris()

# Convert to a DataFrame for easier manipulation
data = pd.DataFrame(iris.data, columns=iris.feature_names)
data['target'] = iris.target

# Display the first few rows of the data
print(data.head())
```



```
First 5 rows of the dataset:
   sepal length (cm)  sepal width (cm)  petal length (cm)  petal width (cm)  target
0                5.1               3.5                1.4               0.2       0
1                4.9               3.0                1.4               0.2       0
2                4.7               3.2                1.3               0.2       0
3                4.6               3.1                1.5               0.2       0
4                5.0               3.6                1.4               0.2       0
```

- Basic Data Exploration: Before training the model, it’s important to explore the data.

```
print(data.describe())  # Summary statistics
print(data.info())      # Data types and null values
```

```



~~~
Data Summary Statistics:
       sepal length (cm)  sepal width (cm)  petal length (cm)  petal width (cm)      target
count         150.000000        150.000000         150.000000        150.000000  150.000000
mean            5.843333          3.057333           3.758000          1.199333    1.000000
std             0.828066          0.435866           1.765298          0.762238    0.819232
min             4.300000          2.000000           1.000000          0.100000    0.000000
25%             5.100000          2.800000           1.600000          0.300000    0.000000
50%             5.800000          3.000000           4.350000          1.300000    1.000000
75%             6.400000          3.300000           5.100000          1.800000    2.000000
max             7.900000          4.400000           6.900000          2.500000    2.000000
~~~



~~~
Data Info:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 5 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   sepal length (cm)  150 non-null    float64
 1   sepal width (cm)   150 non-null    float64
 2   petal length (cm)  150 non-null    float64
 3   petal width (cm)   150 non-null    float64
 4   target             150 non-null    int32  
dtypes: float64(4), int32(1)
memory usage: 5.4 KB
None
~~~

- Data Cleaning (if needed): In case you're working with a custom dataset that has missing values or other issues, clean it:

# Example: Fill missing values or drop rows with missing data
data = data.dropna()  # or you can use data.fillna()
```

3. Splitting the Dataset

- Split the dataset into training and testing sets:

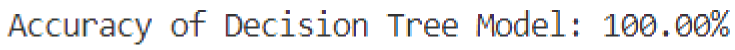
```
from sklearn.model_selection import train_test_split

# Separate features (X) and target labels (y)
X = data.drop('target', axis=1)
y = data['target']

# Split into train and test sets (80% train, 20% test)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

4. Model Training and Evaluation

- Train a machine learning model using Scikit-learn. Let's use the Decision Tree Classifier as an example.

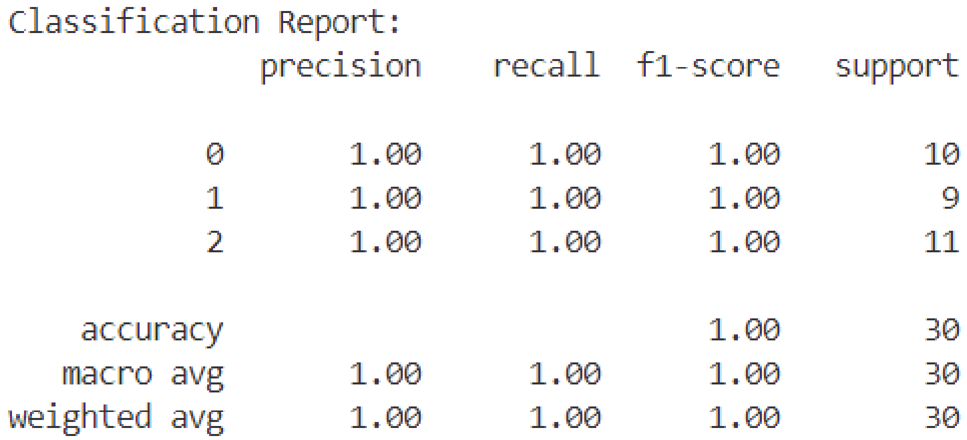
```
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score, classification_report

# Initialize the model
model = DecisionTreeClassifier()

# Train the model
model.fit(X_train, y_train)

# Make predictions
y_pred = model.predict(X_test)
```

```
# Evaluate the model
accuracy = accuracy_score(y_test, y_pred)
print(f'Accuracy: {accuracy * 100:.2f}%')
```



```
Accuracy of Decision Tree Model: 100.00%
```

```
# Detailed classification report
print(classification_report(y_test, y_pred))
```



```
Classification Report:
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        10
           1       1.00      1.00      1.00         9
           2       1.00      1.00      1.00        11

    accuracy                           1.00        30
   macro avg       1.00      1.00      1.00        30
weighted avg       1.00      1.00      1.00        30
```

5. Visualizing the Results

- You can visualize the decision tree or any other results using Matplotlib and other visualization libraries.

```
import matplotlib.pyplot as plt
from sklearn import tree

# Plot the decision tree
plt.figure(figsize=(12,8))
tree.plot_tree(model, feature_names=iris.feature_names,
class_names=iris.target_names, filled=True)
  plt.show()
```

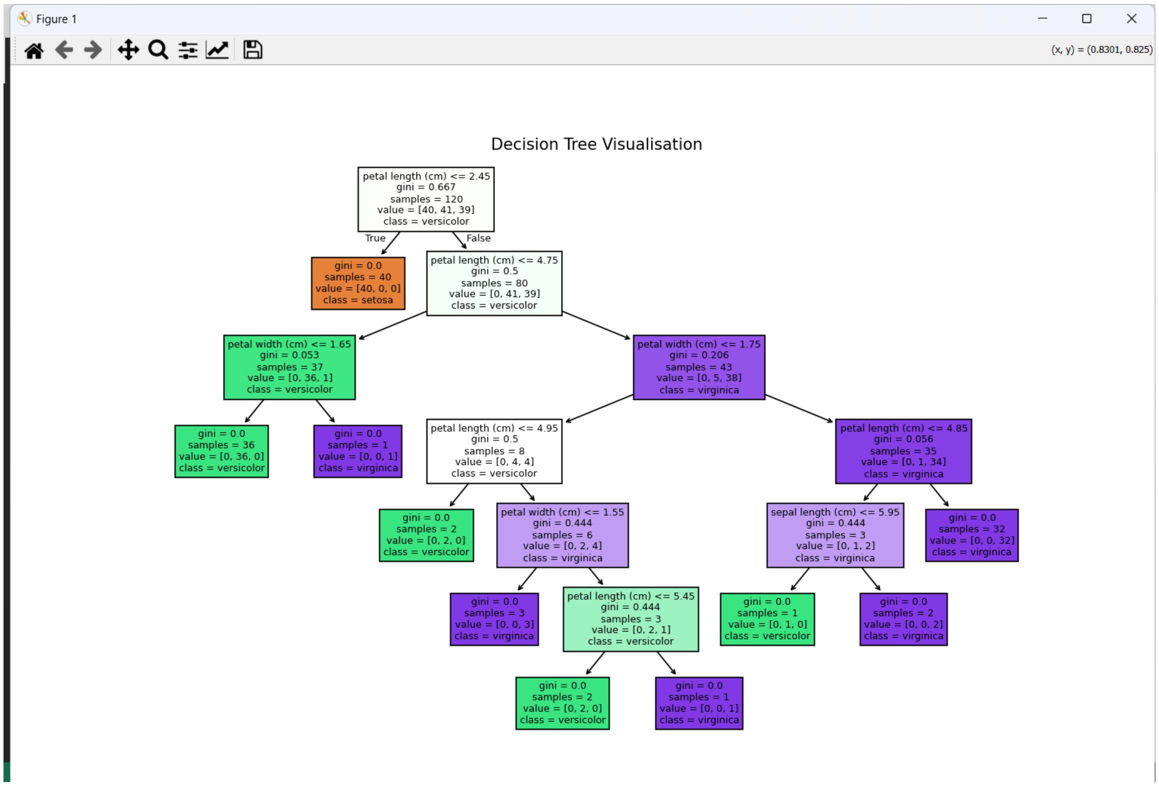


Figure 1
(x, y) = (0.8301, 0.825)
Decision Tree Visualisation
petal length (cm) <= 2.45
gini = 0.667
samples = 120
value = [40, 41, 39]
class = versicolor
True
False
gini = 0.0
samples = 40
value = [40, 0, 0]
class = setosa
petal length (cm) <= 4.75
gini = 0.5
samples = 80
value = [0, 41, 39]
class = versicolor
petal width (cm) <= 1.65
gini = 0.053
samples = 37
value = [0, 36, 1]
class = versicolor
petal width (cm) <= 1.75
gini = 0.206
samples = 43
value = [0, 5, 38]
class = virginica
gini = 0.0
samples = 36
value = [0, 36, 0]
class = versicolor
gini = 0.0
samples = 1
value = [0, 0, 1]
class = virginica
petal length (cm) <= 4.95
gini = 0.5
samples = 8
value = [0, 4, 4]
class = versicolor
petal length (cm) <= 4.85
gini = 0.056
samples = 35
value = [0, 1, 34]
class = virginica
gini = 0.0
samples = 2
value = [0, 2, 0]
class = versicolor
petal width (cm) <= 1.55
gini = 0.444
samples = 6
value = [0, 2, 4]
class = virginica
sepal length (cm) <= 5.95
gini = 0.444
samples = 3
value = [0, 1, 2]
class = virginica
gini = 0.0
samples = 32
value = [0, 0, 32]
class = virginica
gini = 0.0
samples = 3
value = [0, 0, 3]
class = virginica
petal length (cm) <= 5.45
gini = 0.444
samples = 3
value = [0, 2, 1]
class = versicolor
gini = 0.0
samples = 1
value = [0, 1, 0]
class = versicolor
gini = 0.0
samples = 2
value = [0, 0, 2]
class = virginica
gini = 0.0
samples = 2
value = [0, 2, 0]
class = versicolor
gini = 0.0
samples = 1
value = [0, 0, 1]
class = virginica