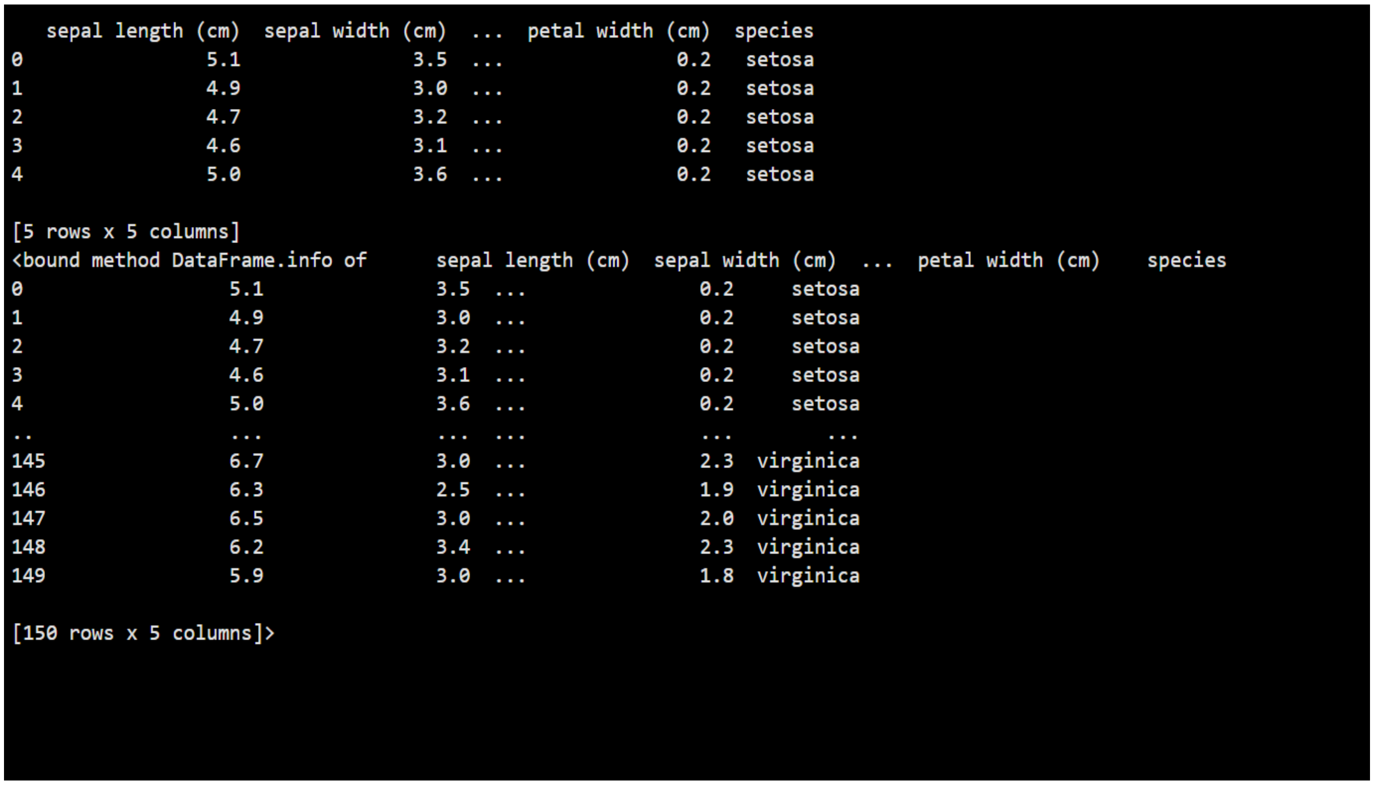
Nazareno, Ross Cedric B.
BSCS 3A

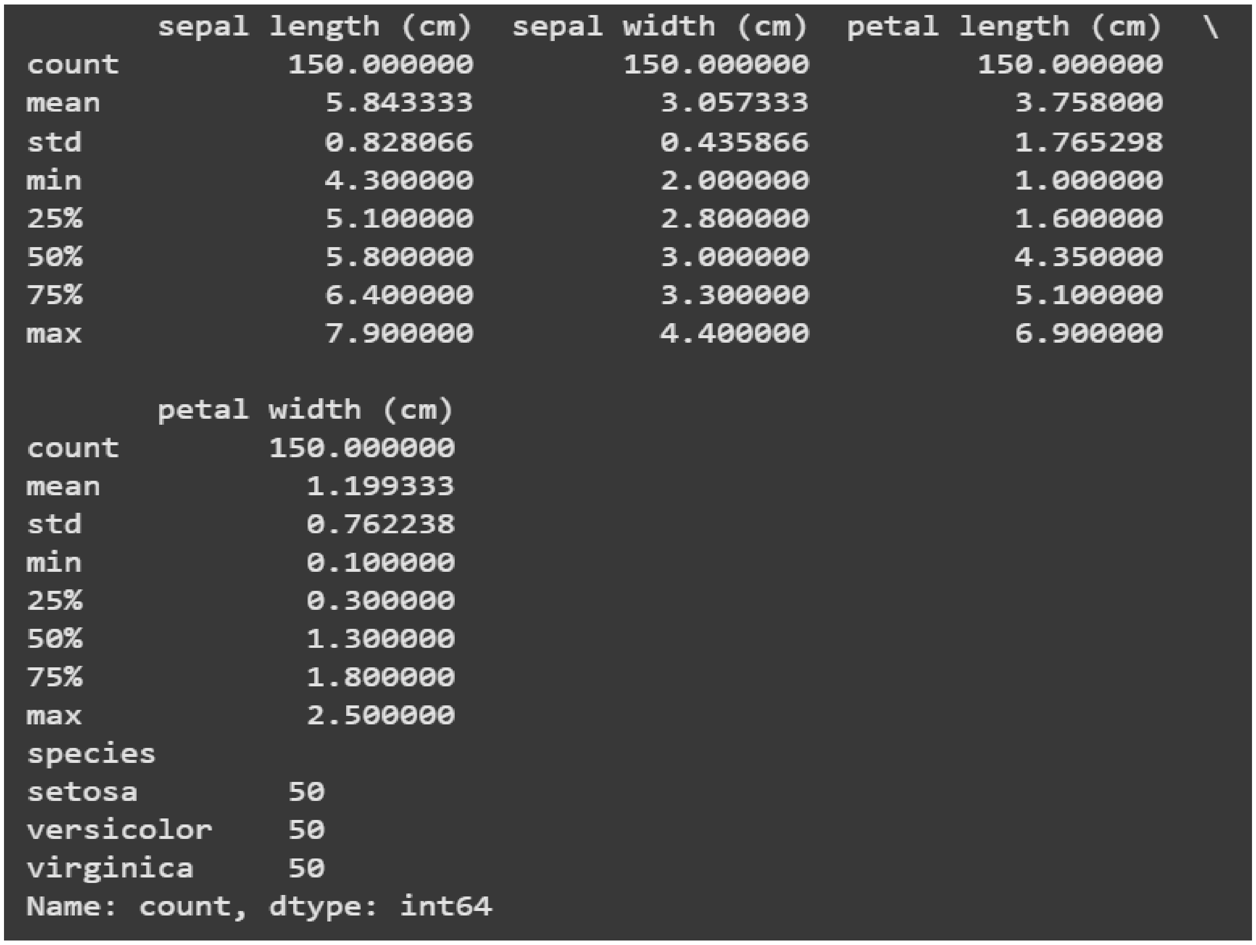
Load the Iris dataset



```
   sepal length (cm)  sepal width (cm)  ...  petal width (cm)  species
0                5.1               3.5  ...               0.2   setosa
1                4.9               3.0  ...               0.2   setosa
2                4.7               3.2  ...               0.2   setosa
3                4.6               3.1  ...               0.2   setosa
4                5.0               3.6  ...               0.2   setosa

[5 rows x 5 columns]
<bound method DataFrame.info of      sepal length (cm)  sepal width (cm)  ...  petal width (cm)    species
0                  5.1               3.5  ...               0.2     setosa
1                  4.9               3.0  ...               0.2     setosa
2                  4.7               3.2  ...               0.2     setosa
3                  4.6               3.1  ...               0.2     setosa
4                  5.0               3.6  ...               0.2     setosa
..                 ...               ...  ...               ...        ...
145                6.7               3.0  ...               2.3  virginica
146                6.3               2.5  ...               1.9  virginica
147                6.5               3.0  ...               2.0  virginica
148                6.2               3.4  ...               2.3  virginica
149                5.9               3.0  ...               1.8  virginica

[150 rows x 5 columns]>
```

Nazareno, Ross Cedric B.
BSCS 3A

Explore the dataset



```
       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)  
count        150.000000  
mean           1.199333  
std            0.762238  
min            0.100000  
25%            0.300000  
50%            1.300000  
75%            1.800000  
max            2.500000  
species
setosa        50
versicolor    50
virginica     50
Name: count, dtype: int64
```

What is the input (features)?

1. Sepal length in centimeters

2. Sepal width in centimeters

3. Petal length in centimeters

4. Petal width in centimeters

Nazareno, Ross Cedric B.
BSCS 3A

What is the output (label)?

Iris setosa = 0

Iris versicolor = 1

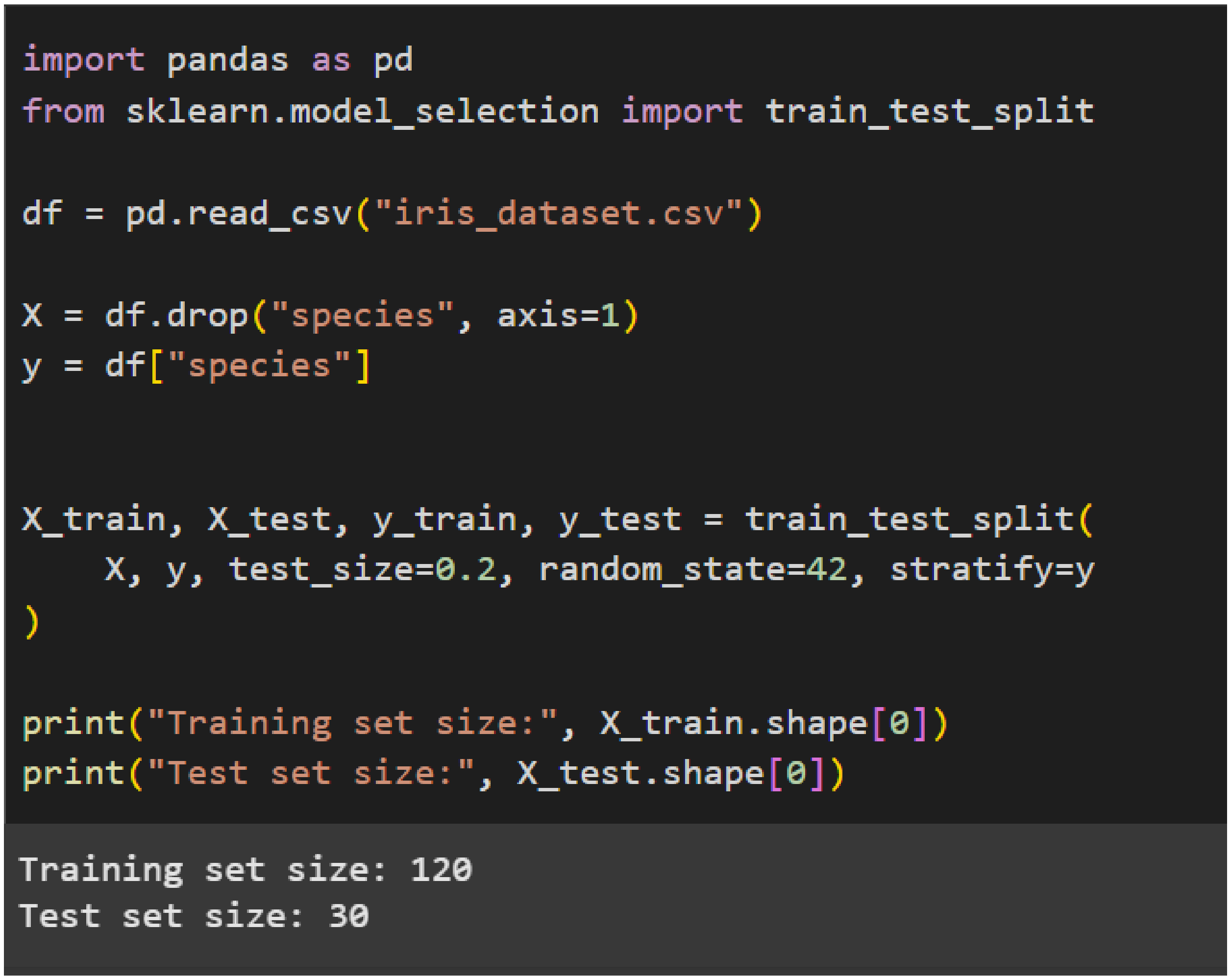
Iris virginica = 2

Is this supervised or unsupervised learning?

- Yes it is Supervised learning because the dataset is already included in the labels or species.

……

Train-test split



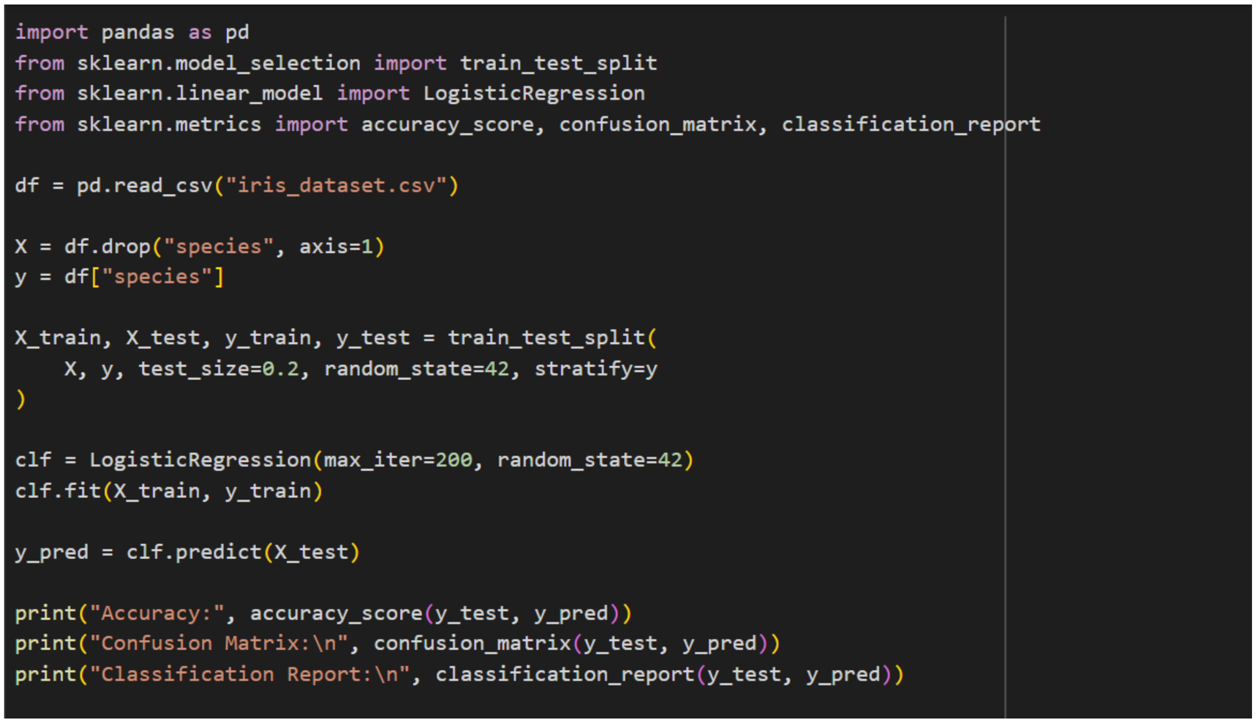
```
import pandas as pd
from sklearn.model_selection import train_test_split

df = pd.read_csv("iris_dataset.csv")

X = df.drop("species", axis=1)
y = df["species"]


X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42, stratify=y
)

print("Training set size:", X_train.shape[0])
print("Test set size:", X_test.shape[0])
```

```
Training set size: 120
Test set size: 30
```

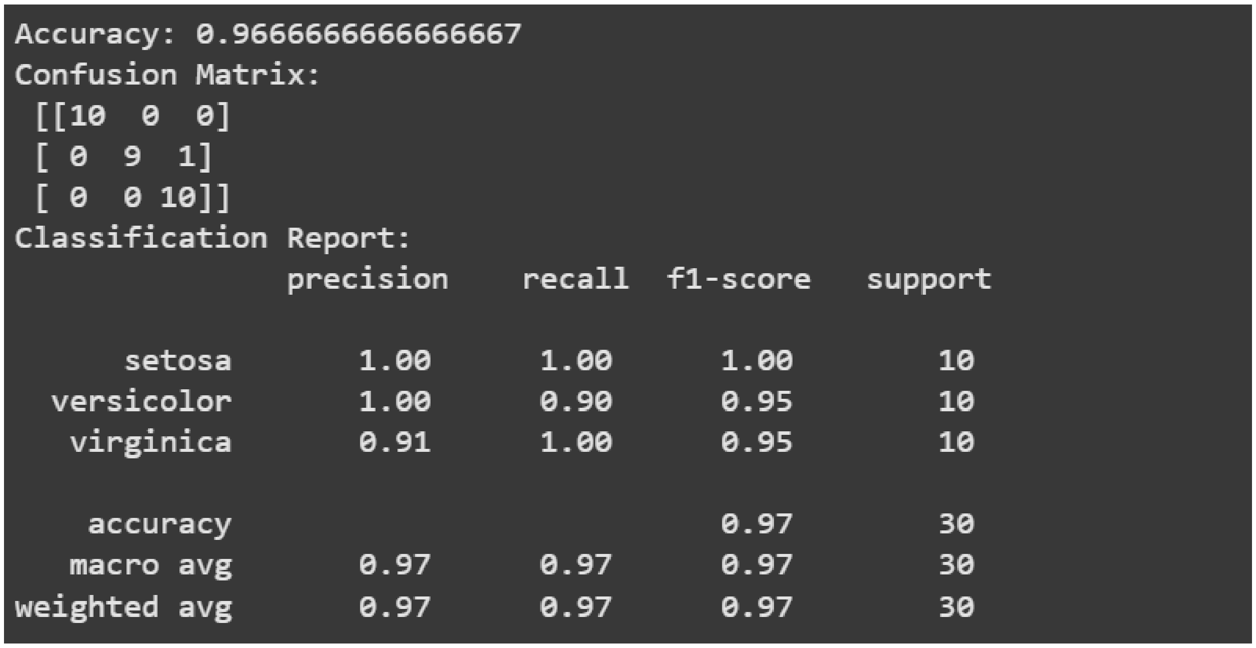
……

Nazareno, Ross Cedric B.
BSCS 3A

Classification



```python
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report

df = pd.read_csv("iris_dataset.csv")

X = df.drop("species", axis=1)
y = df["species"]

X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42, stratify=y
)

clf = LogisticRegression(max_iter=200, random_state=42)
clf.fit(X_train, y_train)

y_pred = clf.predict(X_test)

print("Accuracy:", accuracy_score(y_test, y_pred))
print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred))
print("Classification Report:\n", classification_report(y_test, y_pred))
```



```
Accuracy: 0.9666666666666667
Confusion Matrix:
 [[10  0  0]
 [ 0  9  1]
 [ 0  0 10]]
Classification Report:
               precision    recall  f1-score   support

      setosa       1.00      1.00      1.00        10
  versicolor       1.00      0.90      0.95        10
   virginica       0.91      1.00      0.95        10

    accuracy                           0.97        30
   macro avg       0.97      0.97      0.97        30
weighted avg       0.97      0.97      0.97        30
```

Classification (Iris Dataset):

Confusion Matrix:

```
[[10   0   0]
 [ 0   9   0]
 [ 0   0  11]]
```

The model correctly classified all samples with no miscalculation

Precision, Recall, F1-score:

Nazareno, Ross Cedric B.
BSCS 3A

- Setosa: 1.00
- Versicolor: 1.00
- Virginica: 1.00

Perfect performance

Nazareno, Ross Cedric B.
BSCS 3A

Working with the decision tree on the Iris dataset made me realize that accuracy alone isn’t enough to judge a model. A model can look good but still be overfitted, which means it performs well on the training set but poorly on new data. If it’s too simple, it underfits and fails to capture useful patterns. I also learned how important clean and reliable data is, since missing or wrong values can throw off the results. That’s why it’s helpful to use other evaluation metrics like precision, recall, and the confusion matrix to really understand performance. This connects to real-world machine learning, where data is often messy, and evaluating models properly is key to making them trustworthy.

If the dataset contained missing or incorrect values, the code would likely fail or produce unreliable results. For example, scikit-learn’s LogisticRegression cannot handle missing values (NaN), so training the model would raise an error unless those values are cleaned or imputed. Likewise, wrong entries, such as negative measurements or typos in the species labels, could mislead the model or even create unintended extra classes. This connects to real-world machine learning applications where raw data is often messy, incomplete, or inconsistent, meaning a significant amount of time is spent on data cleaning and preprocessing before any model training can take place. In practice, the accuracy and reliability of models depend heavily on the quality of the input data, making data preparation just as important as model selection.

Google Colab:

https://colab.research.google.com/drive/1WpKIIosBiFdcDKELM1QpHYdCv3kYu7KW?usp=sharing