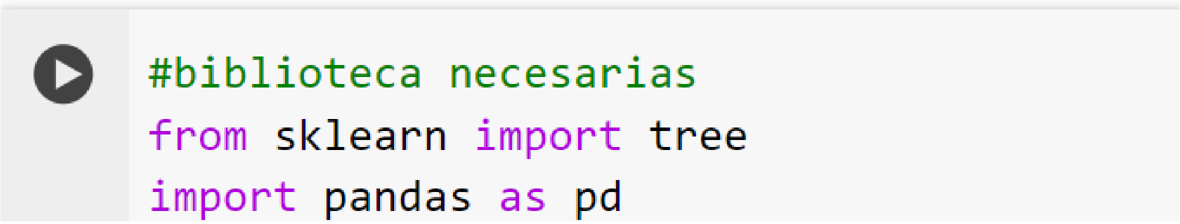
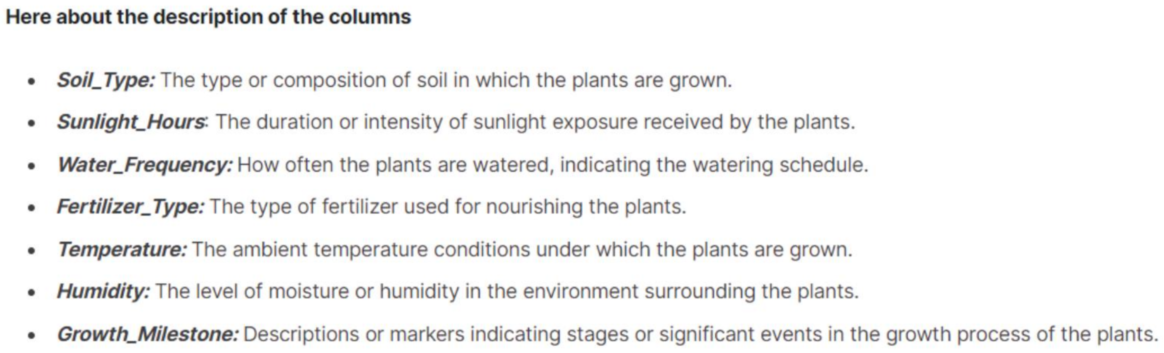
Curva ROC con árboles de Decisión

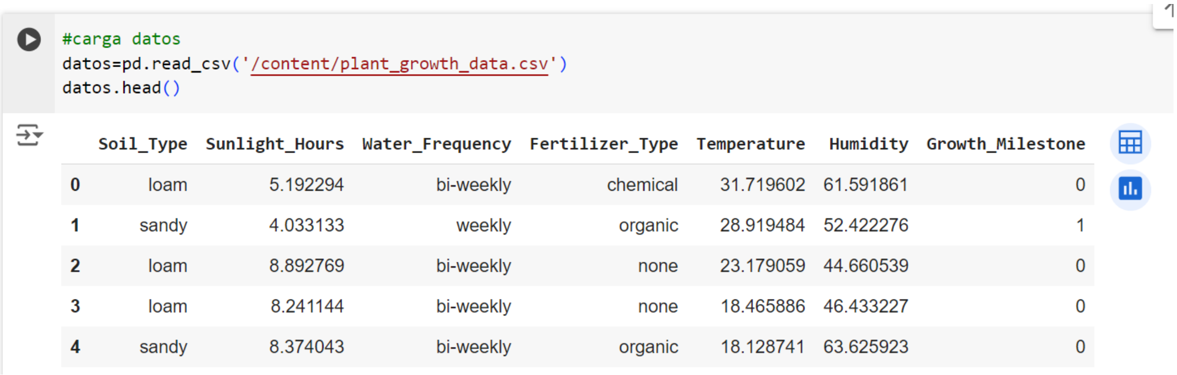


```
#biblioteca necesarias
from sklearn import tree
import pandas as pd
```



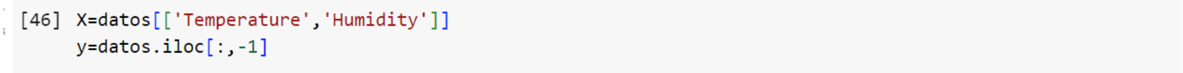
Here about the description of the columns

- ***Soil_Type:*** The type or composition of soil in which the plants are grown.
- ***Sunlight_Hours***: The duration or intensity of sunlight exposure received by the plants.
- ***Water_Frequency:*** How often the plants are watered, indicating the watering schedule.
- ***Fertilizer_Type:*** The type of fertilizer used for nourishing the plants.
- ***Temperature:*** The ambient temperature conditions under which the plants are grown.
- ***Humidity:*** The level of moisture or humidity in the environment surrounding the plants.
- ***Growth_Milestone:*** Descriptions or markers indicating stages or significant events in the growth process of the plants.

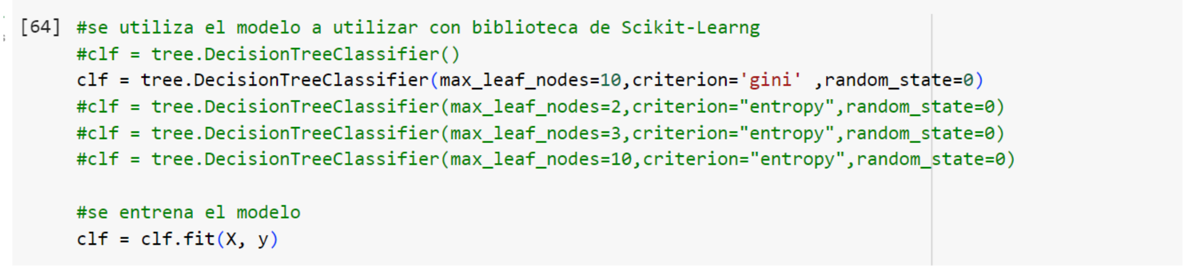


```
#carga datos
datos=pd.read_csv('/content/plant_growth_data.csv')
datos.head()
```

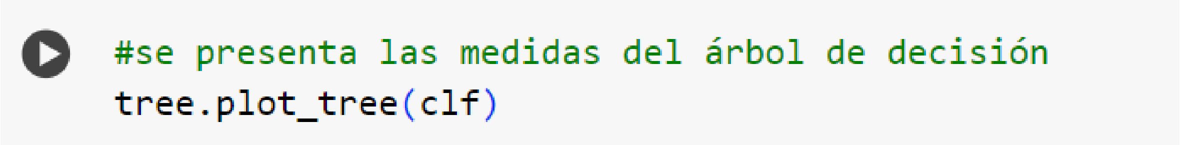
	Soil_Type	Sunlight_Hours	Water_Frequency	Fertilizer_Type	Temperature	Humidity	Growth_Milestone
0	loam	5.192294	bi-weekly	chemical	31.719602	61.591861	0
1	sandy	4.033133	weekly	organic	28.919484	52.422276	1
2	loam	8.892769	bi-weekly	none	23.179059	44.660539	0
3	loam	8.241144	bi-weekly	none	18.465886	46.433227	0
4	sandy	8.374043	bi-weekly	organic	18.128741	63.625923	0



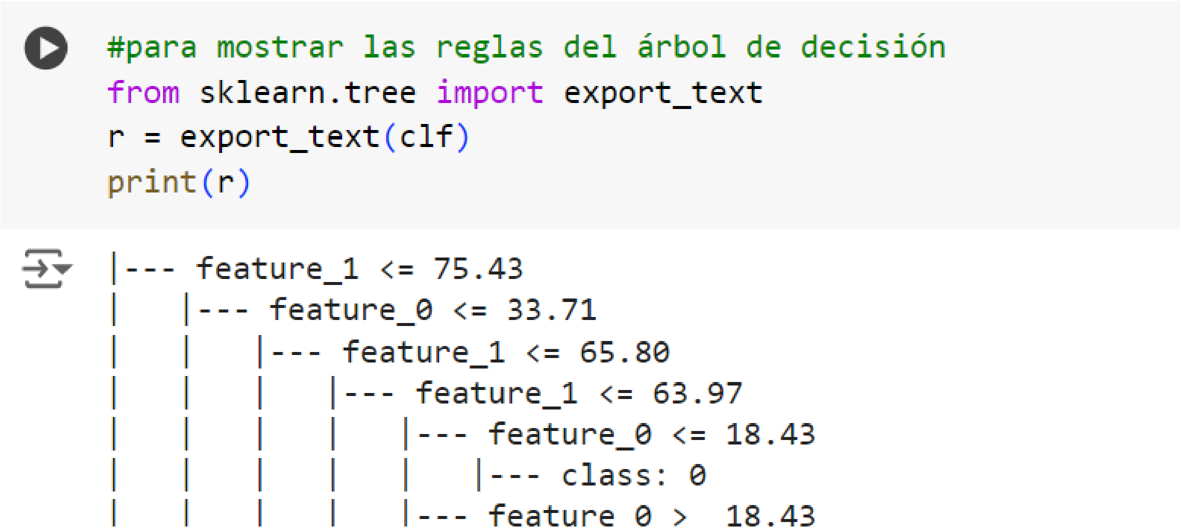
```
[46] X=datos[['Temperature','Humidity']]
     y=datos.iloc[:,-1]
```



```
[64] #se utiliza el modelo a utilizar con biblioteca de Scikit-Learng
     #clf = tree.DecisionTreeClassifier()
     clf = tree.DecisionTreeClassifier(max_leaf_nodes=10,criterion='gini' ,random_state=0)
     #clf = tree.DecisionTreeClassifier(max_leaf_nodes=2,criterion="entropy",random_state=0)
     #clf = tree.DecisionTreeClassifier(max_leaf_nodes=3,criterion="entropy",random_state=0)
     #clf = tree.DecisionTreeClassifier(max_leaf_nodes=10,criterion="entropy",random_state=0)

     #se entrena el modelo
     clf = clf.fit(X, y)
```

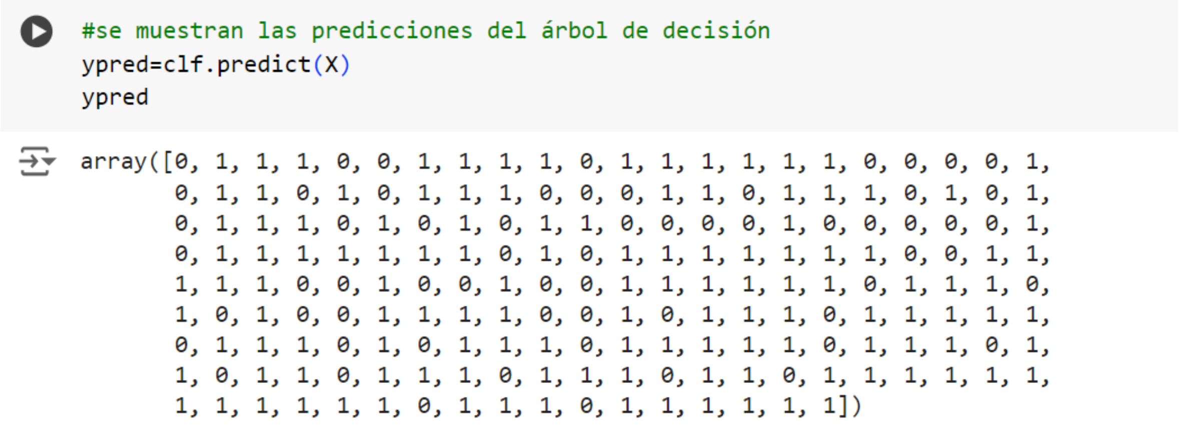


```
#se presenta las medidas del árbol de decisión
tree.plot_tree(clf)
```



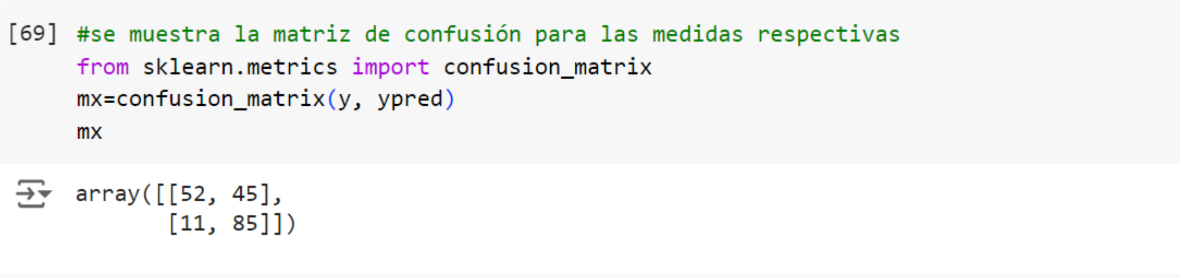
```
#para mostrar las reglas del árbol de decisión
from sklearn.tree import export_text
r = export_text(clf)
print(r)
```

```
|--- feature_1 <= 75.43
|   |--- feature_0 <= 33.71
|   |   |--- feature_1 <= 65.80
|   |   |   |--- feature_1 <= 63.97
|   |   |   |   |--- feature_0 <= 18.43
|   |   |   |   |   |--- class: 0
|   |   |   |   |--- feature 0 >  18.43
```



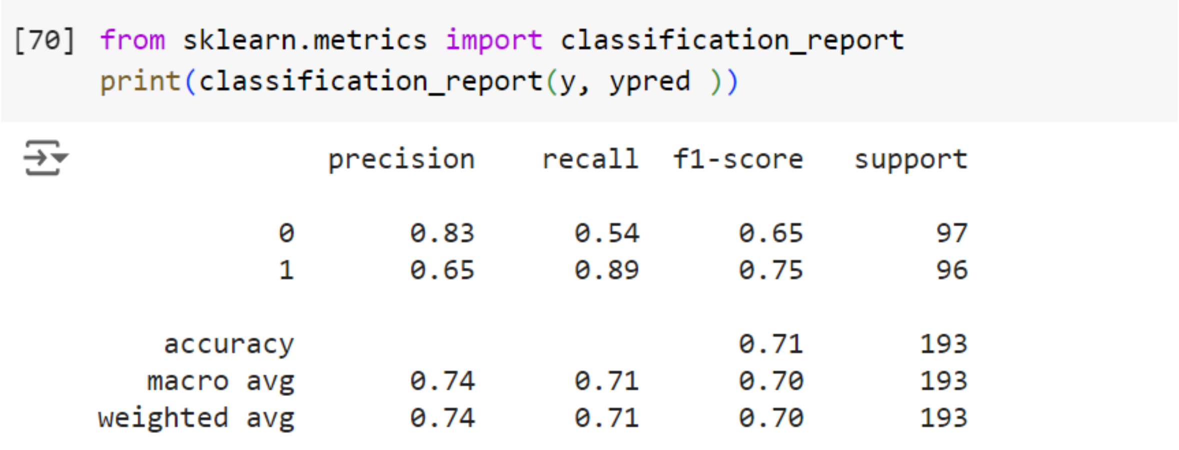
```
#se muestran las predicciones del árbol de decisión
ypred=clf.predict(X)
ypred
```

```
array([0, 1, 1, 1, 0, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 1,
       0, 1, 1, 0, 1, 0, 1, 1, 1, 0, 0, 0, 1, 1, 0, 1, 1, 1, 0, 1, 0, 1,
       0, 1, 1, 1, 0, 1, 0, 1, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1,
       0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1,
       1, 1, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0,
       1, 0, 1, 0, 0, 1, 1, 1, 1, 0, 0, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1,
       0, 1, 1, 1, 0, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1,
       1, 0, 1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1])
```



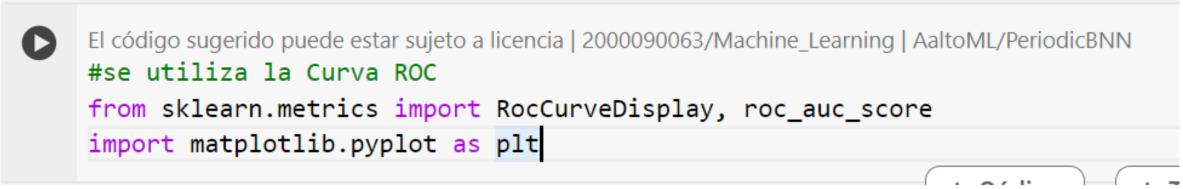
```
[69] #se muestra la matriz de confusión para las medidas respectivas
     from sklearn.metrics import confusion_matrix
     mx=confusion_matrix(y, ypred)
     mx
```

```
array([[52, 45],
       [11, 85]])
```



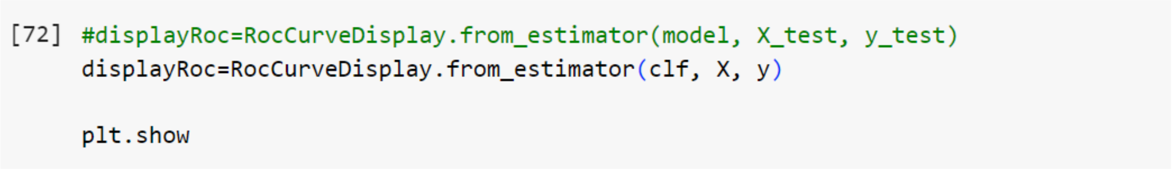
```
[70] from sklearn.metrics import classification_report
     print(classification_report(y, ypred ))
```

```
              precision    recall  f1-score   support

           0       0.83      0.54      0.65        97
           1       0.65      0.89      0.75        96

    accuracy                           0.71       193
   macro avg       0.74      0.71      0.70       193
weighted avg       0.74      0.71      0.70       193
```



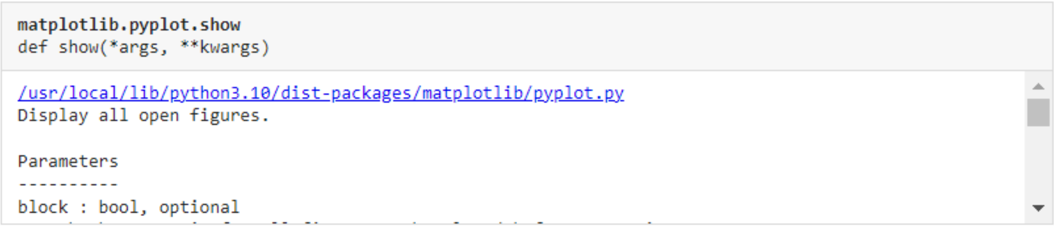
El código sugerido puede estar sujeto a licencia | 2000090063/Machine_Learning | AaltoML/PeriodicBNN

```
#se utiliza la Curva ROC
from sklearn.metrics import RocCurveDisplay, roc_auc_score
import matplotlib.pyplot as plt
```



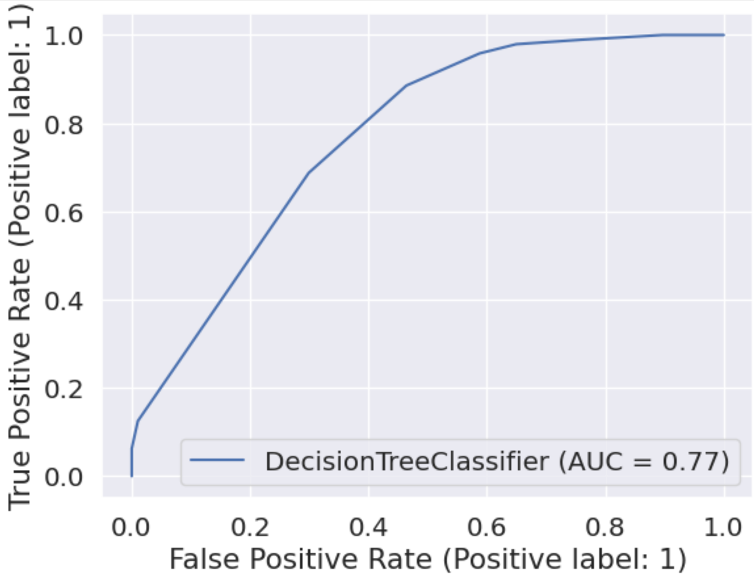
```
[72] #displayRoc=RocCurveDisplay.from_estimator(model, X_test, y_test)
     displayRoc=RocCurveDisplay.from_estimator(clf, X, y)

     plt.show
```

aparece



```
matplotlib.pyplot.show
def show(*args, **kwargs)
```

```
/usr/local/lib/python3.10/dist-packages/matplotlib/pyplot.py
Display all open figures.

Parameters
----------
block : bool, optional
```


True Positive Rate (Positive label: 1)
False Positive Rate (Positive label: 1)
DecisionTreeClassifier (AUC = 0.77)