Módulo 2. Uso de framework o biblioteca de aprendizaje máquina para la implementación de una solución. (Portafolio Implementación)

María del Carmen Vargas Villarreal
A00828570

Machine Learning: Decision Trees

Se utilizarán 2 datasets, el primer modelo será con Iris y el segundo con Wine

Importamos librerias necesarias

In [1]:
```
from sklearn import tree
from sklearn import preprocessing
from IPython.display import Image
import pydotplus
import matplotlib.pyplot as plt

from datetime import datetime

from sklearn.model_selection import train_test_split
from sklearn import metrics
from sklearn.metrics import confusion_matrix

import seaborn as sns
import numpy as np
import pandas as pd
```

Leemos base de datos: Iris.csv

In [2]:
```
df_iris = pd.read_csv('/content/drive/MyDrive/Colab Notebooks/Iris.csv')
df_iris
```

Out[2]:

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	1	5.1	3.5	1.4	0.2	Iris-setosa
1	2	4.9	3.0	1.4	0.2	Iris-setosa
2	3	4.7	3.2	1.3	0.2	Iris-setosa
3	4	4.6	3.1	1.5	0.2	Iris-setosa
4	5	5.0	3.6	1.4	0.2	Iris-setosa
...	...	...	...	...	...	...
145	146	6.7	3.0	5.2	2.3	Iris-virginica
146	147	6.3	2.5	5.0	1.9	Iris-virginica
147	148	6.5	3.0	5.2	2.0	Iris-virginica
148	149	6.2	3.4	5.4	2.3	Iris-virginica
149	150	5.9	3.0	5.1	1.8	Iris-virginica

150 rows × 6 columns

Separamos las variables y etiquetas

In [3]:
```
# Etiqueta
Y_iris = df_iris['Species']

# Variables
X_iris = df_iris.values[:, 1:5] # 5-1
```

Dividimos en training y test set

In [4]:
```
X_train_iris, X_test_iris, y_train_iris, y_test_iris = train_test_split(X_i
```

Implementación del método DecisionTreeClassifier

In [5]:
```
clf_tree_iris = tree.DecisionTreeClassifier(ccp_alpha=0.0, class_weight=Non
                        max_depth=None, max_features=None, max_leaf_nodes=No
                        min_impurity_decrease=0.0,
                        min_samples_leaf=1, min_samples_split=2,
                        min_weight_fraction_leaf=0.0,
                        random_state=None, splitter='best')

# Train Decision Tree Classifer
clf_tree_iris = clf_tree_iris.fit(X_train_iris, y_train_iris)
```

Predict the response for test dataset

In [6]:
```
test_pred = clf_tree_iris.predict(X_test_iris)
```

In [7]:
```
test_pred
```

Out[7]:
```
array(['Iris-versicolor', 'Iris-setosa', 'Iris-virginica',
       'Iris-versicolor', 'Iris-versicolor', 'Iris-setosa',
       'Iris-versicolor', 'Iris-virginica', 'Iris-versicolor',
       'Iris-versicolor', 'Iris-virginica', 'Iris-setosa', 'Iris-setosa',
       'Iris-setosa', 'Iris-setosa', 'Iris-versicolor', 'Iris-virginica',
       'Iris-versicolor', 'Iris-versicolor', 'Iris-virginica',
       'Iris-setosa', 'Iris-virginica', 'Iris-setosa', 'Iris-virginica',
       'Iris-virginica', 'Iris-virginica', 'Iris-virginica',
       'Iris-virginica', 'Iris-setosa', 'Iris-setosa', 'Iris-setosa',
       'Iris-setosa', 'Iris-versicolor', 'Iris-setosa', 'Iris-setosa',
       'Iris-virginica', 'Iris-versicolor', 'Iris-setosa', 'Iris-setosa',
       'Iris-setosa', 'Iris-virginica', 'Iris-versicolor',
       'Iris-versicolor', 'Iris-setosa', 'Iris-setosa'], dtype=object)
```

Verificación de predicciones

Los siguientes valores de entrada de las predicciones fueron obtenidas fijándome en las rows del data frame:

In [8]:
```
#Predicción 1
print(clf_tree_iris.predict([[4.6, 3.1, 1.5, 0.3]]))
```

```
['Iris-setosa']
```

In [9]:
```
#Predicción 2
print(clf_tree_iris.predict([[6.7, 2.5, 5.5, 1.4]]))
```

```
['Iris-virginica']
```

In [10]:
```
#Predicción 3
print(clf_tree_iris.predict([[7.7, 4.0, 1.1, 2.1]]))
```

```
['Iris-setosa']
```

Para obtener las primeras 5 predicciones del set de prueba se convierten a lista el array donde se guardan las predicciones finales y el array donde se contienen los datos que se van a predecir:

In [16]:
```
test_pred_list = test_pred.tolist()
y_test_iris_list = y_test_iris.tolist()
```

In [24]:
```
X_test_iris[0]
```

Out[24]:
```
array([6.1, 2.8, 4.7, 1.2], dtype=object)
```

In [32]:
```
print(' SepalLengthCm | SepalWidthCm |  PetalLengthCm | PetalWidthCm | ACTU
for i in range(5):
    print(i, X_test_iris[i], y_test_iris_list[i], test_pred_list[i])
```

```
 SepalLengthCm | SepalWidthCm |  PetalLengthCm | PetalWidthCm | ACTUAL |
PREDICTED
0 [6.1 2.8 4.7 1.2] Iris-versicolor Iris-versicolor
1 [5.7 3.8 1.7 0.3] Iris-setosa Iris-setosa
2 [7.7 2.6 6.9 2.3] Iris-virginica Iris-virginica
3 [6.0 2.9 4.5 1.5] Iris-versicolor Iris-versicolor
4 [6.8 2.8 4.8 1.4] Iris-versicolor Iris-versicolor
```

In [61]:
```
#X_test_iris
```

In [34]:
```
#y_test_iris
```

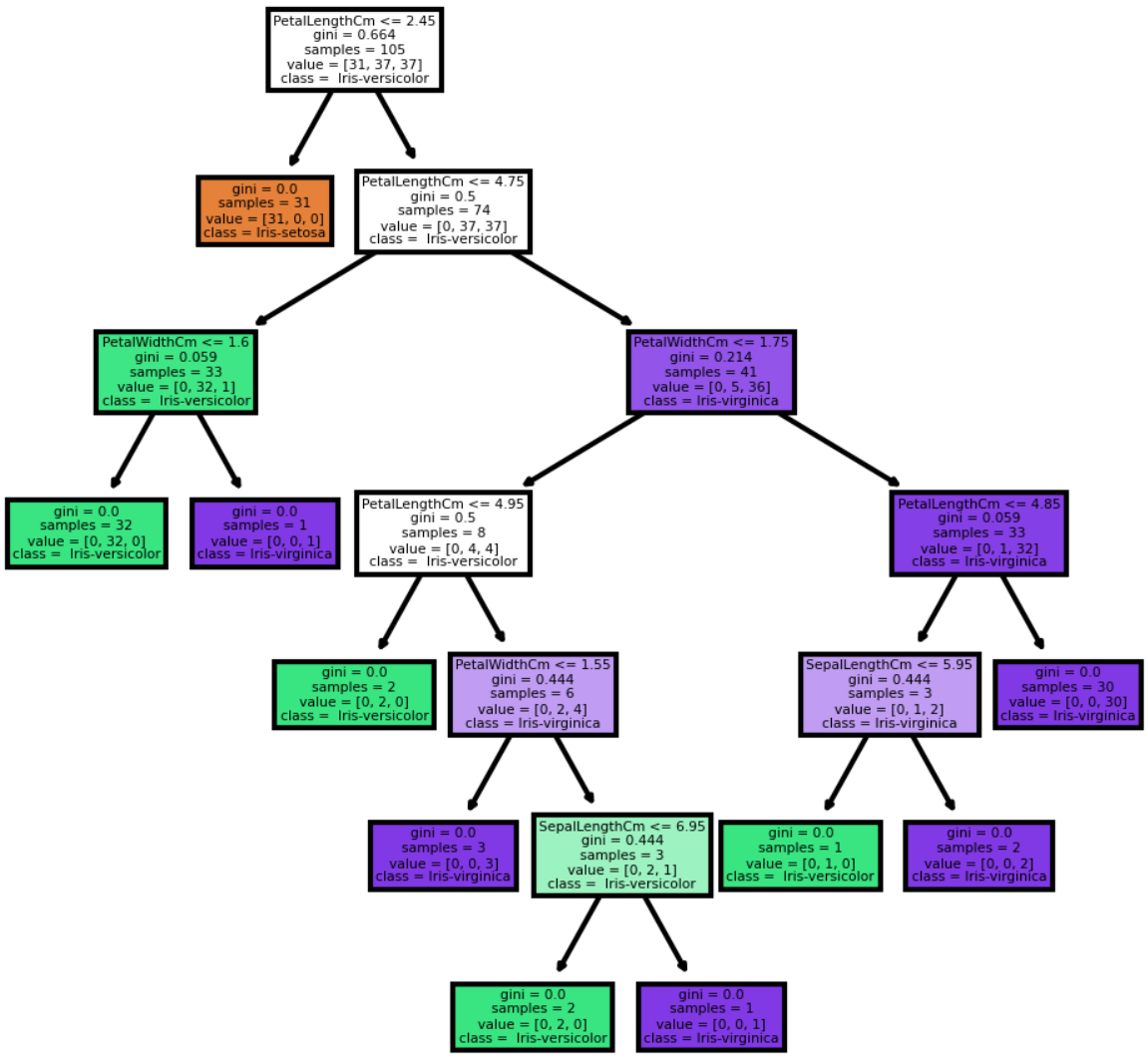
In [33]:
```
#test_pred
```

In [15]:
```
feature_names = ['SepalLengthCm',   'SepalWidthCm', 'PetalLengthCm',     'Pe
target_names = ['Iris-setosa', ' Iris-versicolor', 'Iris-virginica']

#Tamaño de los árboles
fig, axes = plt.subplots(nrows = 1,ncols = 1,figsize = (4,4), dpi=300)

# Crear los datos
tree.plot_tree(clf_tree_iris, filled=True,
                           feature_names=feature_names,
                           class_names=target_names)

fig.savefig('imagename.png')
```



Matriz de confusión

In [37]:
```
from sklearn.metrics import confusion_matrix
confusion_matrix(y_test_iris, test_pred)
```

Out[37]:
```
array([[19,  0,  0],
       [ 0, 13,  0],
       [ 0,  0, 13]])
```

In [38]:
```
from sklearn.metrics import accuracy_score
accuracy_score(y_test_iris, test_pred)
```

Out[38]:
```
1.0
```

Segunda implementación (debido a que sale un Accuracy perfecto en el dataset Iris)

Abrimos data set Wine

In [39]:
```
import pandas as pd # importar libreria

columns = ["Classification", "Alcohol","Malic acid","Ash","Alcalinity of as
    ,"Proanthocyanins", "Color intensity", "Hue", "OD280/OD315 of diluted w

df1 = pd.read_csv('/content/drive/MyDrive/Colab Notebooks/Copia de wine.dat
df1 = df1.drop(['index'], axis=1) # abrir el archivo de datos con los nombr
df1.head()
```

Out[39]:

	Classification	Alcohol	Malic acid	Ash	Alcalinity of ash	Magnesium	Total phenols	Flavanoids	Nonflavanoid phenols	P
0	1	14.23	1.71	2.43	15.6	127	2.80	3.06	0.28	
1	1	13.20	1.78	2.14	11.2	100	2.65	2.76	0.26	
2	1	13.16	2.36	2.67	18.6	101	2.80	3.24	0.30	
3	1	14.37	1.95	2.50	16.8	113	3.85	3.49	0.24	
4	1	13.24	2.59	2.87	21.0	118	2.80	2.69	0.39	

Verificamos columna de clasificación

In [40]:
```
df1['Classification']
```

Out[40]:
```
0      1
1      1
2      1
3      1
4      1
      ..
173    3
174    3
175    3
176    3
177    3
Name: Classification, Length: 178, dtype: int64
```

Obtenemos valores únicos de la columna de clasificación

In [41]:
```
df1.Classification.unique()
```

Out[41]:
```
array([1, 2, 3])
```

Para una interpretación más sencilla se reemplazaron las clasificaciones numéricas a categóricas

In [42]:
```
df1.Classification.replace([1,2,3], ['Cultivar 1', 'Cultivar 2', 'Cultivar
df1.Classification
```

Out[42]:
```
0      Cultivar 1
1      Cultivar 1
2      Cultivar 1
3      Cultivar 1
4      Cultivar 1
          ...
173    Cultivar 3
174    Cultivar 3
175    Cultivar 3
176    Cultivar 3
177    Cultivar 3
Name: Classification, Length: 178, dtype: object
```

In [43]:
```
df1.Classification.unique()
```

Out[43]:
```
array(['Cultivar 1', 'Cultivar 2', 'Cultivar 3'], dtype=object)
```

Separamos las variables de la variable target

In [66]:
```
df_x = df1.drop(["Classification"], axis=1).values
df_y = df1["Classification"]


# Print df_x first 5
df_x
```

Out[66]:
```
array([[1.423e+01, 1.710e+00, 2.430e+00, ..., 1.040e+00, 3.920e+00,
        1.065e+03],
       [1.320e+01, 1.780e+00, 2.140e+00, ..., 1.050e+00, 3.400e+00,
        1.050e+03],
       [1.316e+01, 2.360e+00, 2.670e+00, ..., 1.030e+00, 3.170e+00,
        1.185e+03],
       ...,
       [1.327e+01, 4.280e+00, 2.260e+00, ..., 5.900e-01, 1.560e+00,
        8.350e+02],
       [1.317e+01, 2.590e+00, 2.370e+00, ..., 6.000e-01, 1.620e+00,
        8.400e+02],
       [1.413e+01, 4.100e+00, 2.740e+00, ..., 6.100e-01, 1.600e+00,
        5.600e+02]])
```

Separamos entre training y test set

In [67]:
```
from sklearn.model_selection import train_test_split


X_train, X_test, y_train, y_test = train_test_split(df_x, df_y, test_size=0
```

Construimos el clasificador

In [68]:
```
classifier = tree.DecisionTreeClassifier(ccp_alpha=0.0, class_weight=None,
                         max_depth=None, max_features=None, max_leaf_nodes=No
                         min_impurity_decrease=0.0,
                         min_samples_leaf=1, min_samples_split=2,
                         min_weight_fraction_leaf=0.0,
                         random_state=None, splitter='best')

# Train Decision Tree Classifer
classifier = classifier.fit(X_train, y_train)
```

Generar predicción

In [69]:
```
test_pred_decision_tree = classifier.predict(X_test)
```

Obtenemos columna

In [70]:
```
df1.columns
```

Out[70]:
```
Index(['Classification', 'Alcohol', 'Malic acid', 'Ash', 'Alcalinity of a
sh',
       'Magnesium', 'Total phenols', 'Flavanoids', 'Nonflavanoid phenol
s',
       'Proanthocyanins', 'Color intensity', 'Hue',
       'OD280/OD315 of diluted wines', 'Proline'],
      dtype='object')
```

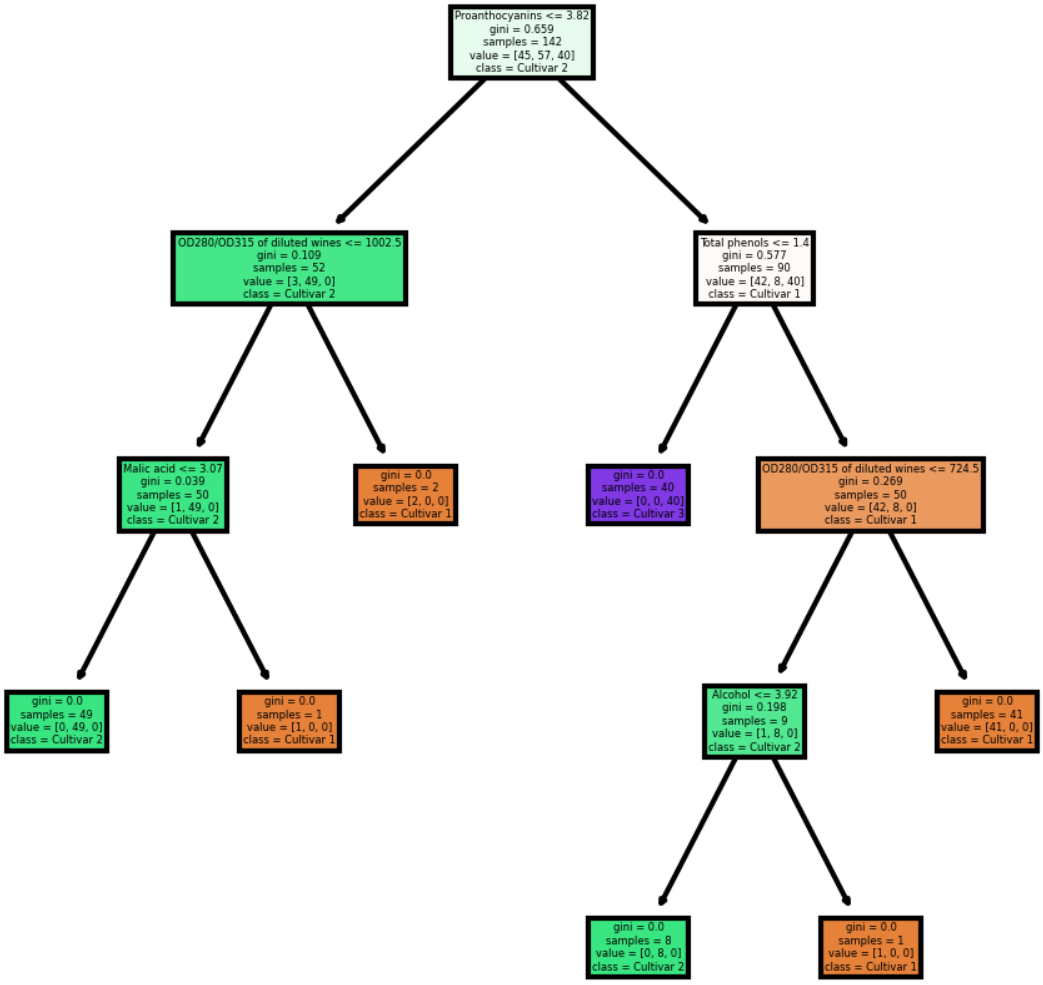
Graficamos el árbol de decisión

In [71]:
```
feature_names = ['Classification', 'Alcohol', 'Malic acid', 'Ash', 'Alcalin
       'Magnesium', 'Total phenols', 'Flavanoids', 'Nonflavanoid phenols',
       'Proanthocyanins', 'Color intensity', 'Hue',
       'OD280/OD315 of diluted wines', 'Proline']
target_names = ['Cultivar 1', 'Cultivar 2', 'Cultivar 3']

#Tamaño de los árboles
fig, axes = plt.subplots(nrows = 1,ncols = 1,figsize = (4,4), dpi=300)

# Crear los datos
tree.plot_tree(classifier, filled=True,
                           feature_names=feature_names,
                           class_names=target_names)

fig.savefig('imagename.png')
```



Comparamos el valor actual con el predicho por el modelo

In [72]:
```
test_pred_decision_tree_list = test_pred_decision_tree.tolist()
```

In [73]:
```
y_test_list = y_test.tolist()
```

In [80]:
```
import numpy as np
np.set_printoptions(suppress=True) # Para que los valores no salgan como no

#X_test
```

In [82]:
```
df1.columns
```

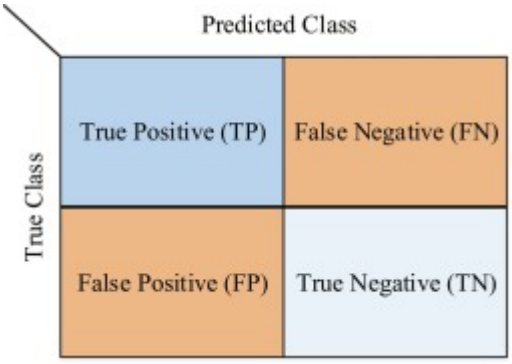
Out[82]:
```
Index(['Classification', 'Alcohol', 'Malic acid', 'Ash', 'Alcalinity of a
sh',
       'Magnesium', 'Total phenols', 'Flavanoids', 'Nonflavanoid phenol
s',
       'Proanthocyanins', 'Color intensity', 'Hue',
       'OD280/OD315 of diluted wines', 'Proline'],
      dtype='object')
```

In [95]:
```
# Se imprimen primero los títulos de cada variables
# Después los valores de cada variable
# Y para cada conjunto de variables de entrada se imprime el valor esperado
# 'Real' hace referencia a valor esperado
# 'Pred' hace referencia al valor predicho

print('Alcohol', 'Malic acid', 'Ash', 'Alcalinity of ash','Magnesium', 'Tot
       'OD280/OD315 of diluted wines', 'Proline')
for i in range(5):
  print(i, X_test[i], 'Real: ',y_test_list[i],' | Pred: ', test_pred_decisi
```

```
Alcohol Malic acid Ash Alcalinity of ash Magnesium Total phenols Flavanoi
ds Nonflavanoid phenols Proanthocyanins Color intensity Hue OD280/OD315 o
f diluted wines Proline
0 [ 13.64   3.1    2.56  15.2  116.     2.7    3.03   0.17   1.66   5.1
   0.96   3.36 845.  ] Real:  Cultivar 1  | Pred:  Cultivar 1
1 [  14.21    4.04    2.44   18.9   111.      2.85    2.65    0.3     1.2
5
    5.24    0.87    3.33 1080.  ] Real:  Cultivar 1  | Pred:  Cultivar 1
2 [ 12.93   2.81   2.7   21.    96.     1.54   0.5    0.53   0.75   4.6
   0.77   2.31 600.  ] Real:  Cultivar 3  | Pred:  Cultivar 3
3 [  13.73    1.5     2.7    22.5   101.      3.      3.25    0.29    2.3
8
    5.7     1.19    2.71 1285.  ] Real:  Cultivar 1  | Pred:  Cultivar 1
4 [ 12.37   1.17   1.92  19.6   78.     2.11   2.     0.27   1.04   4.68
   1.12   3.48 510.  ] Real:  Cultivar 2  | Pred:  Cultivar 2
```

Obtenemos matriz de confusión



In [53]:
```
cf_matrix = confusion_matrix(y_test, test_pred_decision_tree)
cf_matrix
```

Out[53]:
```
array([[13,  1,  0],
       [ 0, 14,  0],
       [ 1,  0,  7]])
```

Métrica Accuracy

In [90]:
```
from sklearn.metrics import accuracy_score
accuracy_score(y_test, test_pred_decision_tree)
```

Out[90]:
```
0.9444444444444444
```