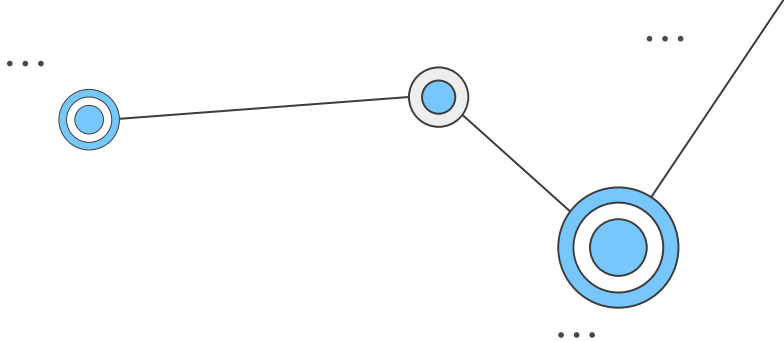


Decision Trees

UNIVERSIDAD CATÓLICA DE CÓRDOBA
JESUITAS

Dr. Francisco Arduh
2023

Entrenando un árbol de decisiones

Creamos y entrenamos el modelo:

```
from sklearn.datasets import load_iris
from sklearn.tree import DecisionTreeClassifier

iris = load_iris()
X = iris.data[:, 2:] # petal length and width
y = iris.target

tree_clf = DecisionTreeClassifier(max_depth=2)
tree_clf.fit(X, y)
```

Exportamos el DT a un gráfico formato .dot

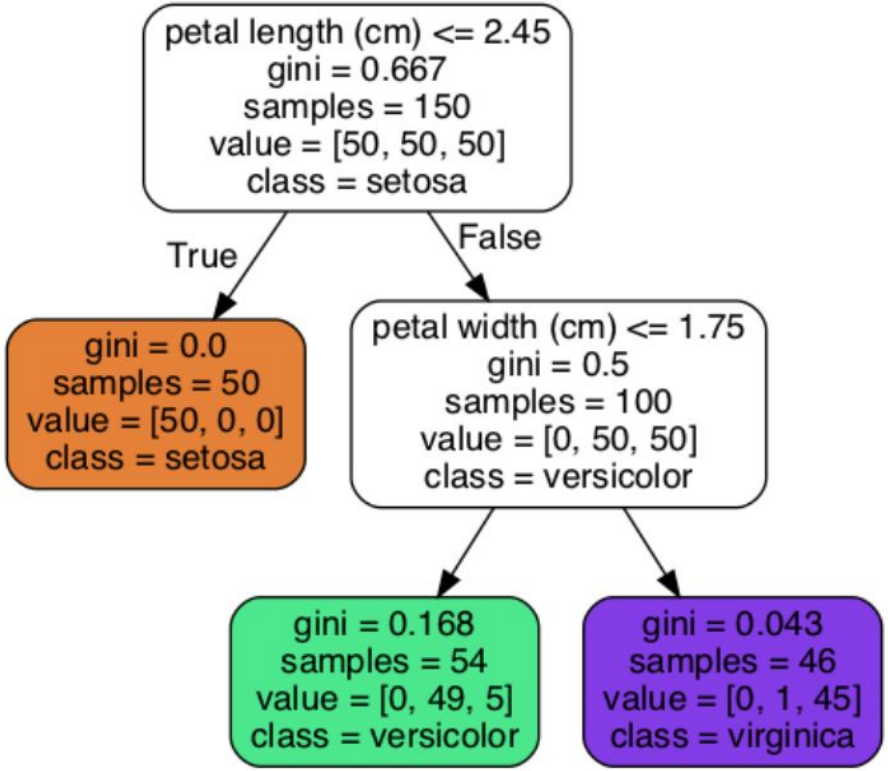
```
from sklearn.tree import export_graphviz

export_graphviz(
        tree_clf,
        out_file=image_path("iris_tree.dot"),
        feature_names=iris.feature_names[2:],
        class_names=iris.target_names,
        rounded=True,
        filled=True
    )
```

Transformamos de .dot a .png con la librería Graphviz:

```
$ dot -Tpng iris_tree.dot -o iris_tree.png
```

¿Cómo funciona?

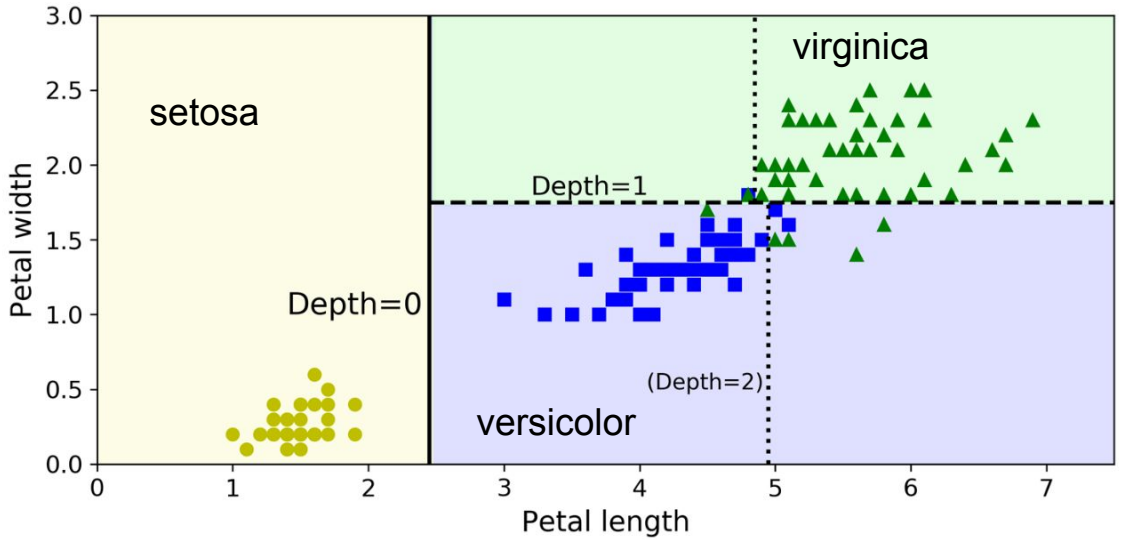


Los datos no necesitan ser escalados o centrados antes de utilizarlos.

Impureza Gini:

$$G_i = 1 - \sum_{k=1}^{n} {p_{i,k}}^2$$

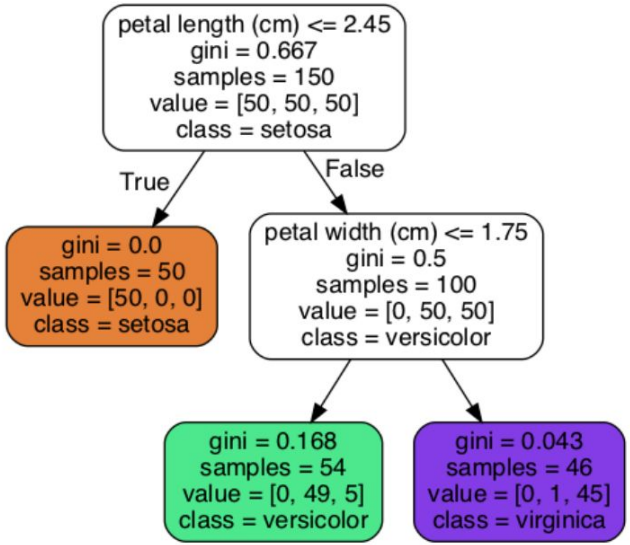
¿Cómo funciona?



A los árboles de decisiones se los llama modelos de “caja blanca” por su fácil interpretabilidad. En contraste, en los modelos “caja negra”, es más difícil explicar el por qué de las predicciones qué hacen.

Estimando probabilidades

```
>>> tree_clf.predict_proba([[5, 1.5]])
array([[0.        , 0.90740741, 0.09259259]])
>>> tree_clf.predict([[5, 1.5]])
array([1])
```



Algoritmo CART

- CART: Classification and Regression Tree.
- Es el algoritmo que utiliza scikit-learn para entrenar a los DT.
- Se elige el par (k, t_k), donde k es la característica y t_k es el corte sobre la misma.
- Se los elige minimizando la función de costo:

$$J(k, t_k) = \frac{m_{\text{left}}}{m} G_{\text{left}} + \frac{m_{\text{right}}}{m} G_{\text{right}}$$

$$\text{donde} \begin{cases} G_{\text{left/right}} & \text{mide la impureza del subset izquierdo/derecho} \\ m_{\text{left/right}} & \text{es el número de instancias en el subset izquierdo/derecho} \end{cases}$$

- Una vez elegido el corte óptimo, se repite el proceso hasta alcanzar la máxima profundidad deseada del algoritmo.

Complejidad computacional

- Resolver el problema en forma óptima O(exp(m))
- Con el algoritmo CART entrenar el modelo tiene un complejidad O(n $\times$ m $\log_2(m)$).
- Realizar una predicción tiene un complejidad computacional O($\log_2(m)$).

Gini o Entropía

Impureza Gini:

$$G_i = 1 - \sum_{k=1}^{n} {p_{i,k}}^2$$

Entropía:

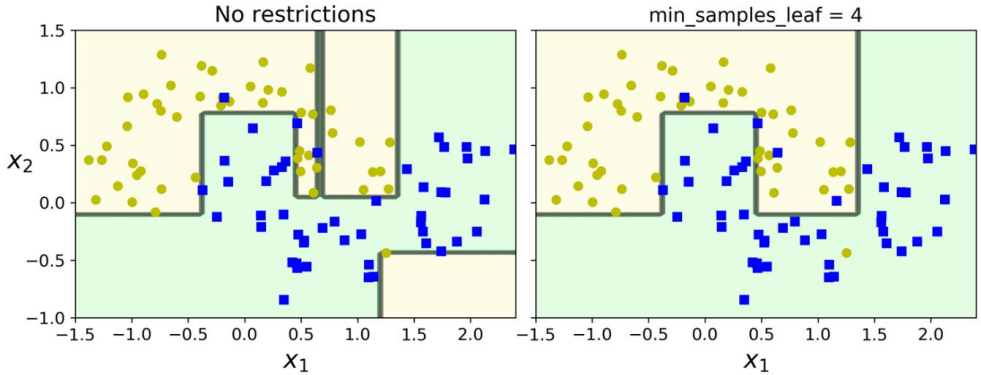
$$H_i = - \sum_{\substack{k=1 \\ p_{i,k} \neq 0}}^{n} p_{i,k} \log_2 \left(p_{i,k}\right)$$

- Gini es un poco más rápido de computar.
- Entropía nos da árboles más balanceados.

Hiperparámetros para regularización

- Es un modelo no paramétrico, es decir no realiza suposiciones a priori de los datos.
- Para restringir los grados de libertad del DT scikit-learn provee algunas variables:

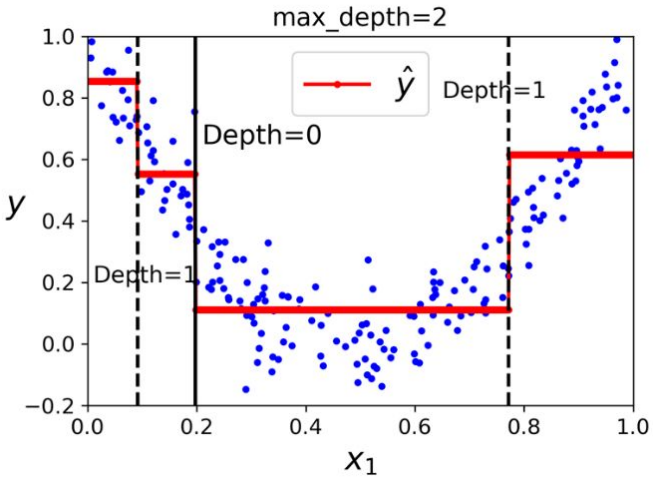
class `sklearn.tree.`**`DecisionTreeClassifier`**(**, criterion='gini', splitter='best', max_depth=None, min_samples_split=2, min_samples_leaf=1, min_weight_fraction_leaf=0.0, max_features=None, random_state=None, max_leaf_nodes=None, min_impurity_decrease=0.0, class_weight=None, ccp_alpha=0.0*) [source]

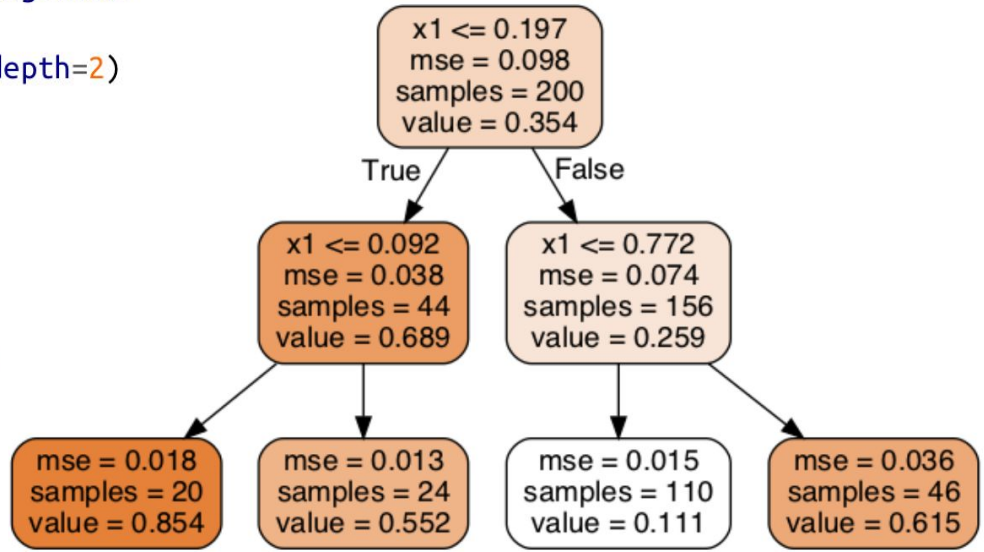


Regresión con DT

```
from sklearn.tree import DecisionTreeRegressor

tree_reg = DecisionTreeRegressor(max_depth=2)
tree_reg.fit(X, y)
```

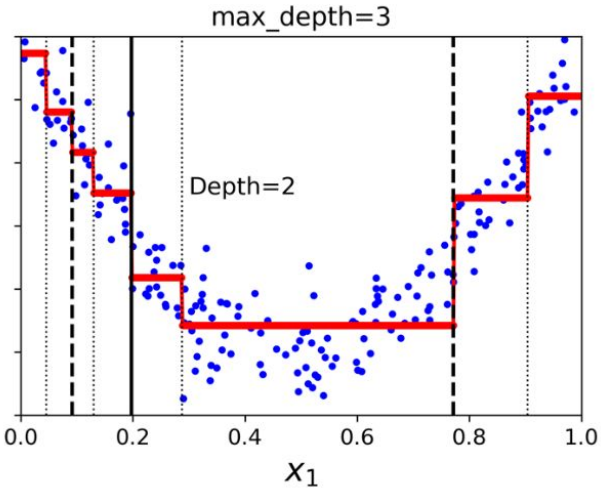


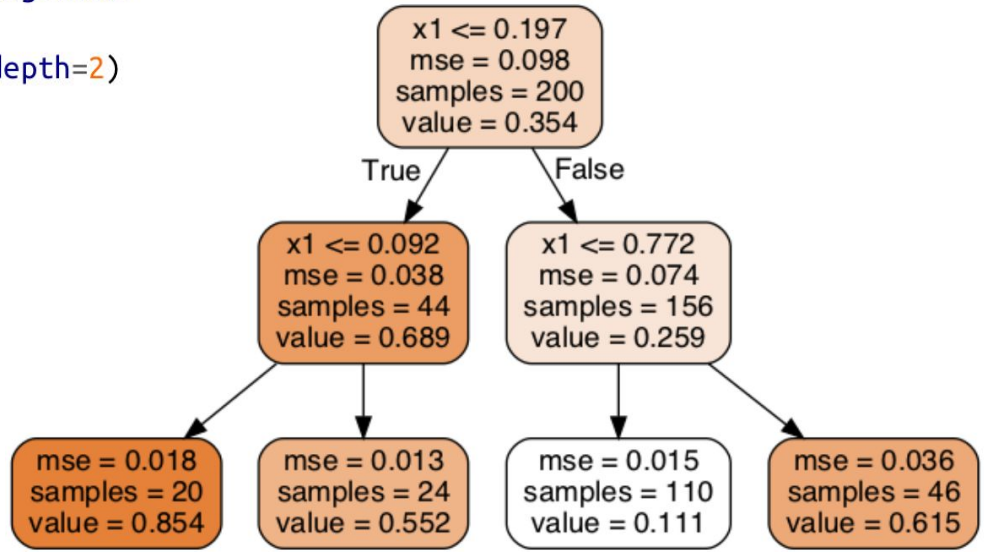


Regresión con DT

```
from sklearn.tree import DecisionTreeRegressor

tree_reg = DecisionTreeRegressor(max_depth=2)
tree_reg.fit(X, y)
```

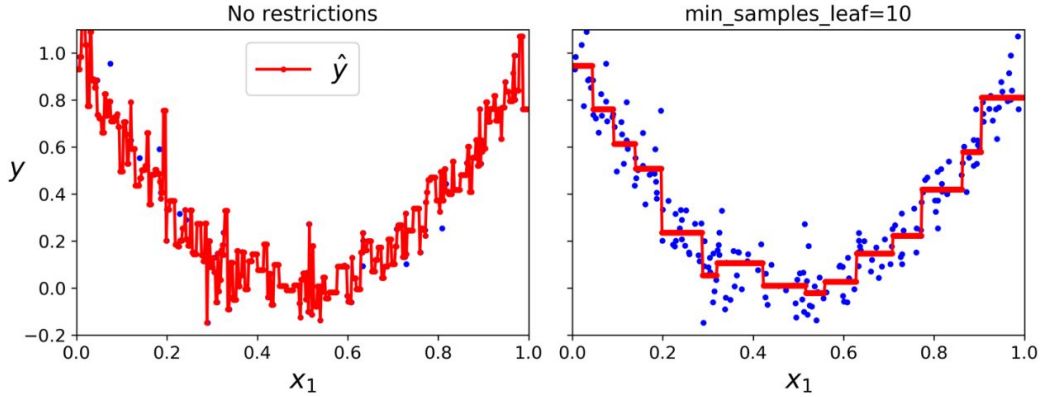




Regresión con DT

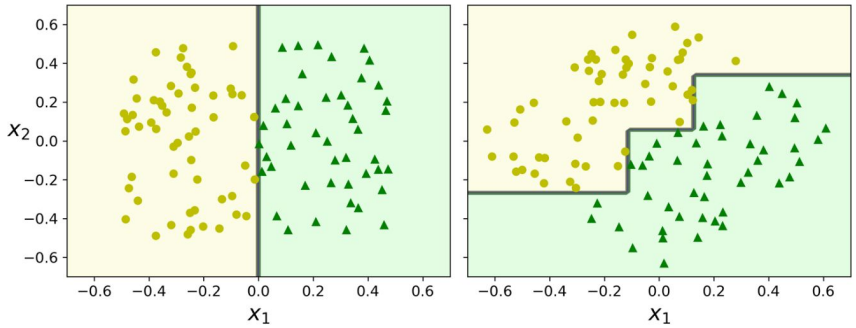
$$J(k, t_k) = \frac{m_{\text{left}}}{m}\text{MSE}_{\text{left}} + \frac{m_{\text{right}}}{m}\text{MSE}_{\text{right}} \quad \text{where} \begin{cases} \text{MSE}_{\text{node}} = \sum_{i \in \text{node}} \left(\hat{y}_{\text{node}} - y^{(i)}\right)^2 \\ \hat{y}_{\text{node}} = \frac{1}{m_{\text{node}}} \sum_{i \in \text{node}} y^{(i)} \end{cases}$$

Regularización:

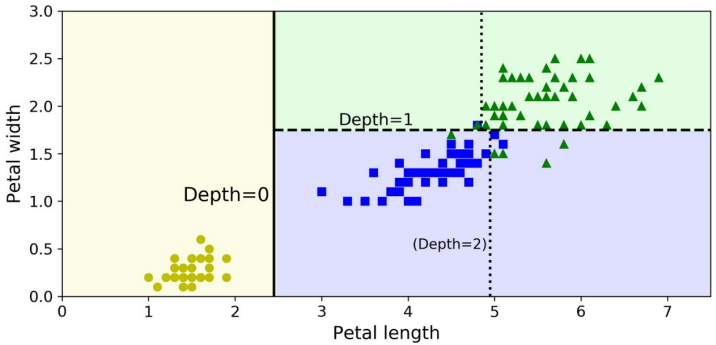


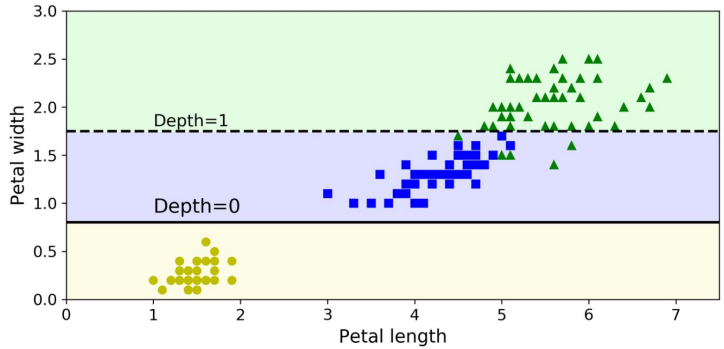
Inestabilidad

Es sensible a la rotación:



Es sensible a pequeños cambios en la muestra:





El algoritmo es estocástico, por lo que se puede obtener distintos modelos con el mismo training set