Curso de especialización en Inteligencia Artificial y Big Data

UD04. DESARROLLO DE APLICACIONES DE IA: ALGORITMOS

ARBOLES DE DECISION (RESUMEN)

Carlos Sáenz Adán

1. Arboles de decisión

Antes de comenzar con la explicación, te recomiendo que eches un ojo al siguiente vídeo. En este video se realiza una explicación muy bien ilustrada de la regresión lineal.

https://www.youtube.com/watch?v=tYPi6qcCQbg

Los arboles de decision son algoritmos de *machine learning* versátiles que pueden realizar tanto tareas de clasificación como de regresión. Son algoritmos muy potentes que permiten ajustar conjuntos de datos complejos.

1.1. Entrenar y visualizar árboles de decisión

Para entender los árboles de decisión, vamos a construir uno para ver cómo hace las predicciones.

```
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, random_state=42)
tree_clf.fit(X, y)
```

Podremos visualizar el árbol de decisión primero utilizando el método export_graphviz() para producir un archivo de definición gráfico llamado iris_tree.dot. Al utilizar graphviz, necesitaremos instalarlo antes, para ello podemos ejecutar

```
!pip install graphviz
```

Posteriormente, el código para generar el fichero será:

```
from graphviz import Source
from sklearn.tree import export_graphviz

export_graphviz(
        tree_clf,
        out_file=os.path.join(IMAGES_PATH, "iris_tree.dot"),
        feature_names=iris.feature_names[2:],
        class_names=iris.target_names,
        rounded=True,
```

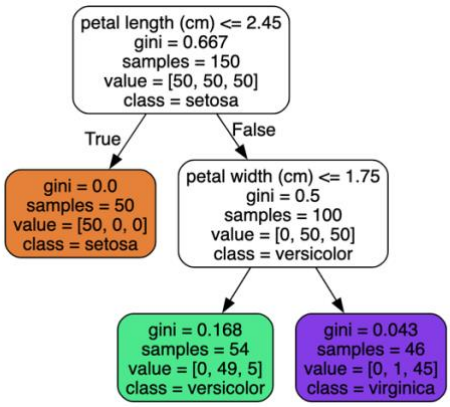
```
    filled=True
)
```

A continuación, utilizaremos la herramienta de línea de comandos dot que se encuentra en el paquete Graphviz para convertir este fichero .dot en otro de formato png o pdf.

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

En nuestro caso, podemos verlo de la siguiente forma, no hace falta convertirlo.

```
Source.from_file(os.path.join("iris_tree.dot"))
```



1.2. Hacer predicciones

Imagina que encuentras una flor iris y quieres clasificarla. Atendiendo al anterior gráfico comenzaremos por la raíz (profundidad 0, arriba): este nodo pregunta si la longitud del pétalo es inferior o igual a 2.45cm. Si lo es, bajamos al nodo hijo de la izquierda (profundidad 1, izquierda). En este caso, se trata de un "nodo terminal" así que no formula preguntas: simplemente mira la clase predicha para ese nodo y en este caso se indica que es *setosa*.

En el caso de encontrar una flor cuya longitud de pétalo es mayor que 2.45cm. Tienes que bajar por el nodo hijo de la derecha, que como no es un nodo termina, pregunta ¿La anchura del pétalo es menor o igual a 1.75? En caso verdadero la flor se clasificará como versicolor y en caso negativo se clasificará como virgínica.

En cada nodo podemos encontrar además de la pregunta (primera línea) y la clase (última línea) una serie de métricas que nos dan información:

- Samples: cuenta a cuántas instancias de entrenamiento se aplica. Por ejemplo, 50 instancias de entrenamiento tienen menos de 2.45cm de pétalo.

- Value: indica a cuántas instancias de entrenamiento de cada clase se aplica el nodo: por ejemplo, el node de abajo a la derecha se aplica a 0 setosas, 1 versicolor, y 45 virgínicas.

- Gini: mide la impureza, un nodo es puro (gini=0) si todas las instancias de entrenamiento a las que

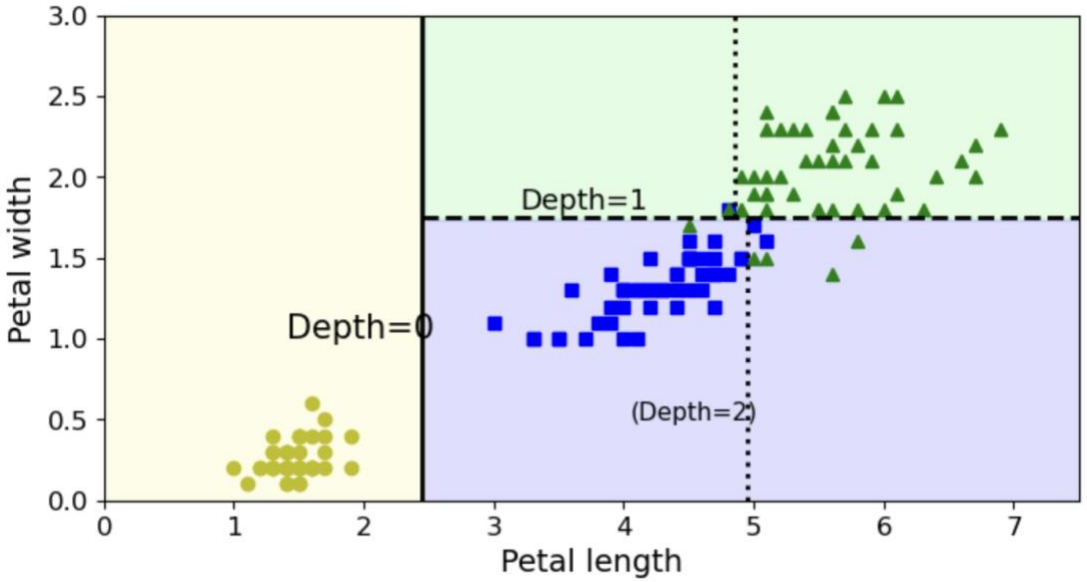
se aplica pertenecen a la misma clase. Por ejemplo, el nodo de profundidad 1 a la izquierda sólo se aplica a las setosas; por lo tanto, su índice de Gini es 0. En el caso del nodo de profundidad 1 a la derecha es 0.5 porque se aplica por igual a versicolor y virgínica.

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

Por ejemplo, para el nodo de profundidad 2 a la izquierda, se obtendrá de la siguiente forma:

$$gini = 1 - \left({}^{0}/_{54}\right)^2 - \left({}^{49}/_{54}\right)^2 - \left({}^{5}/_{54}\right)^2 \cong 0.168$$

La siguiente imagen muestra los límites de decisión de este árbol.

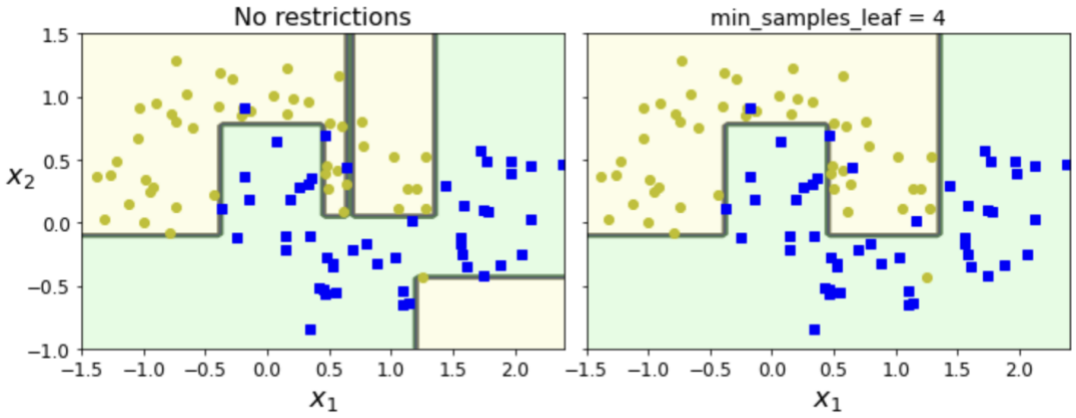


1.3. Hiperparámetros de regularización

Los arboles de decisión asumen muy pocas cosas sobre los datos de entrenamiento. Si se deja sin restricciones, la estructura del árbol se adaptará a los datos de entrenamiento, ajustándolos mucho, seguramente sobreajustándolos. Para evitar el sobreajuste, hay que restringir la libertad de decisión del árbol durante el entrenamiento. Esto lo conseguiremos a través de los hiperparámetros de la clase DecissionTreeClassifier:

- max_depth: restringe la profundidad del árbol.
- min_samples_split: el número mínimo de muestras que debe tener un nodo para poder dividirse.
- Min_samples_leaf: el número mínimo de muestras que debe tener un nodo terminal.
- Max_leaf_nodes: el número máximo de nodos terminales.
- Max_features: número máximo de características que se evalúan en un nodo.

La siguientes figuras muestran dos árboles de decisión entrenados con el mismo conjunto de datos. A la izquierda se ve un modelo con los hiperparámetros por defecto (sin darle valor) y a la derecha un árbol con min_samples_leaf=4. Se ve claramente que el modelo de la izquierda está sobreajustado.



1.4. Modelos de regresión

Los árboles de decisión también pueden servir para realizar tareas de regresión. Vamos a crear un árbol de regresión con la clase DecisionTreeRegressor de Scikit-Learn:

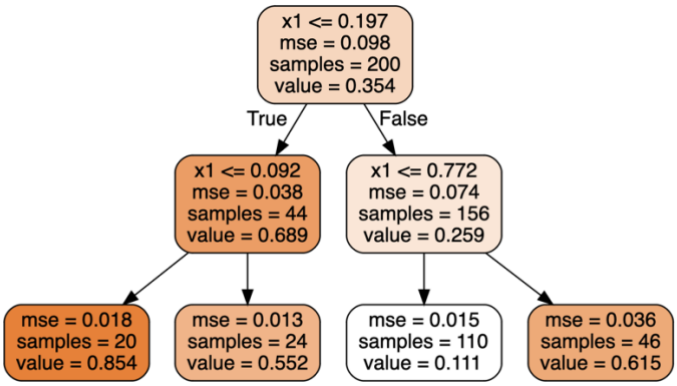
```
angle = np.pi / 180 * 20
rotation_matrix = np.array([[np.cos(angle), -np.sin(angle)], [np.sin(angle), np.cos(angle)]])
Xr = X.dot(rotation_matrix)

tree_clf_r = DecisionTreeClassifier(random_state=42)
tree_clf_r.fit(Xr, y)

plt.figure(figsize=(8, 3))
plot_decision_boundary(tree_clf_r, Xr, y, axes=[0.5, 7.5, -1.0, 1], iris=False)

plt.show()
```

La siguiente figura muestra el árbol resultante, si te das cuenta, ahora en lugar de predecir la clase en cada nodo se predice un valor, y ofrece el error cuadrático medio sobre las instancias.



Gráficamente podemos verlo de la siguiente forma: