DIPLOMATURA EN MACHINE LEARNING CON PYTHON

Árboles de Desición

Árboles de Decisión

- Instalación y uso del paquete
- Ejemplo en Python
- Principales parámetros de ajuste y control
- Problema concreto

Un poco de contenido teórico:

¿Qué es un árbol de decisión?

Los algoritmos de aprendizaje basados en árboles se consideran uno de los mejores y más utilizados métodos de aprendizaje supervisado. Los métodos basados en árboles potencian los modelos predictivos con alta precisión, estabilidad y facilidad de interpretación.
Los arboles de decisión son representaciones gráficas de posibles soluciones a una decisión basada en ciertas condiciones, es uno de los algoritmos de aprendizaje supervisado más utilizados en machine Learning y pueden realizar tareas de clasificación (*) o regresión (**) (acrónimo del inglés CART). La comprensión de su funcionamiento suele ser simple y a la vez muy potente.

(*)¿Qué es una tarea de Clasificación?

Los problemas de tipo de clasificación generalmente son aquellos en los que intentamos predecir los valores de una variable dependiente categórica (clase, pertenencia a grupos, etc.) a partir de una o más variables predictoras continuas y / o .
Por ejemplo, podemos estar interesados en predecir quién se graduará o no de la universidad, o quién renovará o no una suscripción. Estos serían ejemplos de problemas simples de clasificación binaria, donde la variable dependiente categórica solo puede asumir dos valores distintos y mutuamente excluyentes.
En otros casos, podríamos estar interesados en predecir cuál de los múltiples productos de consumo alternativos diferentes (por ejemplo, marcas de automóviles) decide comprar una persona, o qué tipo de falla ocurre con diferentes tipos de motores. En esos casos, existen múltiples categorías o clases para la variable dependiente categórica.

(**)¿Qué es una tarea de Regresión?

Los problemas de tipo regresión son generalmente aquellos en los que intentamos predecir los valores de una variable continua a partir de una o más variables predictoras categóricas.
Por ejemplo, podemos querer predecir los precios de venta de casas unifamiliares (una variable dependiente continua) a partir de varios otros predictores continuos (p. Ej., Pies cuadrados) así como predictores categóricos (por ejemplo, estilo de hogar, como rancho, dos pisos, etc., código postal o código de área telefónica donde se encuentra la propiedad, etc., tenga en cuenta que esta última variable sería de naturaleza categórica, aunque contendría datos numéricos valores o códigos).

> Si utilizamos la regresión múltiple simple, o algún modelo lineal general (*GLM*) para predecir los precios de venta de viviendas unifamiliares, determinaríamos una ecuación lineal para estas variables que puede usarse para calcular los precios de venta pronosticados

Utilizamos mentalmente estructuras de árbol de decisión constantemente en nuestra vida diaria sin darnos cuenta:

Por Ejemplo:
¿Llueve? => lleva paraguas. ¿Soleado? => lleva anteojos de sol.
¿Estoy cansado? => toma café. (Decisiones del tipo **IF THIS THEN THAT**)
Los árboles de decisión tienen un primer nodo llamado raíz (root) y luego se descomponen el resto de atributos de entrada en dos ramas (podrían ser más, pero no nos meteremos en eso ahora) planteando una condición que puede ser cierta o falsa.
Se bifurca cada nodo en 2 y vuelven a subdividirse hasta llegar a las hojas que son los nodos finales y que equivalen a respuestas a la solución: Si/No, Comprar/Vender, o lo que sea que estemos clasificando.
Otro ejemplo pueden ser los juegos de adivinanzas:

1. ¿Animal ó vegetal? -Animal
2. ¿Tiene cuatro patas? -Si
3. ¿Hace guau? -Si
4. -> Es un perro!

Terminología importante de un árbol de decisión

1. **Nodo raíz (nodo de decisión superior):** Representa a toda la población o muestra y esto se divide en dos o más conjuntos homogéneos.

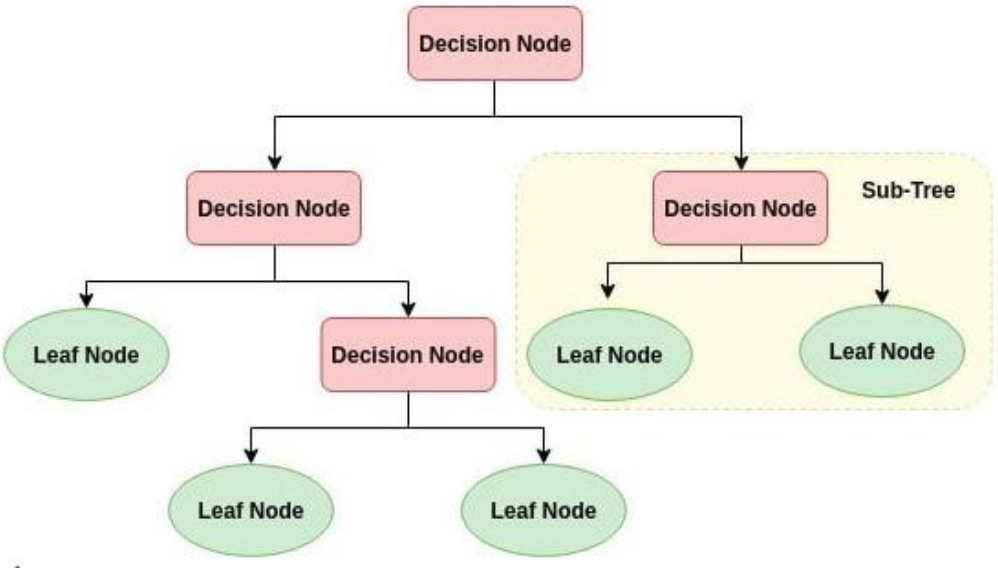
2. **División:** Es un proceso de división de un nodo en dos o más subnodos. [19659024] Nodo de decisión: Cuando un subnodo se divide en subnodos adicionales, se llama nodo de decisión.

3. **Nodo de hoja / terminal:** Los nodos sin hijos (sin división adicional) se llaman Hoja o nodo terminal.

4. **Poda:** Cuando reducimos el tamaño de los árboles de decisión eliminando nodos (opuesto a la división), el proceso se llama poda.

5. **Rama / Subárbol:** Una subsección del árbol de decisión se denomina rama o subárbol.

6. **Nodo padre e hijo:** Un nodo, que se divide en subnodos se denomina nodo principal de subnodos, mientras que los subnodos son hijos de un nodo principal.



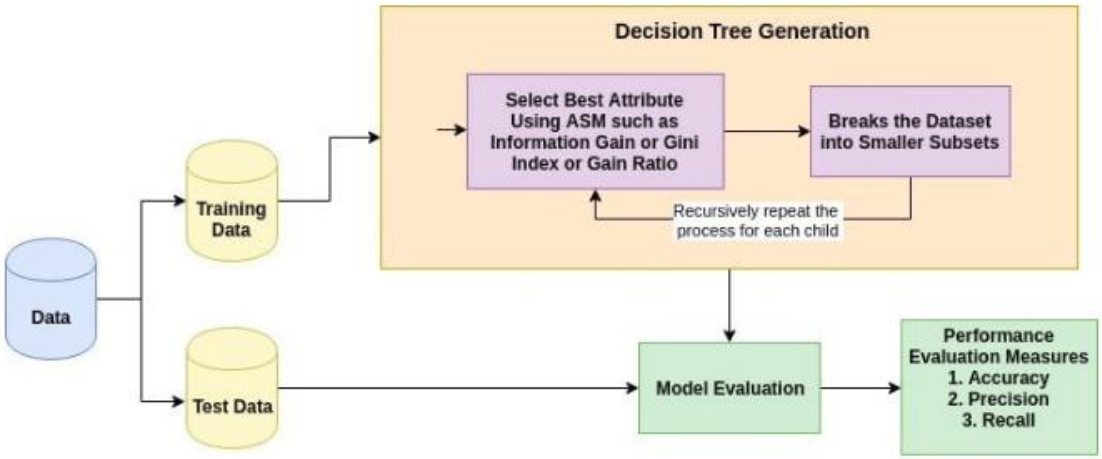
¿Cómo funciona el algoritmo del árbol de decisión en Machine Learning?

La idea básica detrás de cualquier algoritmo de árbol de decisión es el siguiente:

1. Seleccione el mejor atributo utilizando Medidas de selección de atributos (ASM) para dividir los registros.

2. Haga que ese atributo sea un nodo de decisión y divida el conjunto de datos en subconjuntos más pequeños. recursivamente para cada niño hasta que una de las condiciones coincida:

    - Todas las tuplas pertenecen al mismo valor de atributo.
    - No quedan más atributos.
    - No hay más instancias.



¿Qué necesidad hay de usar el Algoritmo de Árbol?

Supongamos que tenemos atributos como Género con valores «hombre ó mujer» y edad en rangos: «menor de 18 ó mayor de 18» para tomar una decisión. Podríamos crear un árbol en el que dividamos primero por género y luego subdividir por edad. Ó **podría ser al revés**: primero por edad y luego por género. El algoritmo es quien *analizando los datos y las salidas -por eso es supervisado!*– decidirá la mejor forma de hacer las divisiones (*split*) entre nodos. Tendrá en cuenta de qué manera lograr una predicción (clasificación ó regresión) con mayor probabilidad de acierto. Parece sencillo, no? Pensemos que si tenemos 10 atributos de entrada cada uno con 2 o más valores posibles, **las combinaciones para decidir el mejor árbol serían cientos ó miles**… Esto ya no es un trabajo para hacer artesanalmente. Y ahí es donde este algoritmo cobra importancia, pues él nos devolverá el árbol óptimo para la **toma de decisión más acertada desde un punto de vista probabilístico**.

Principales parámetros de ajuste y control

¿Cómo lograr un árbol óptimo?

Para obtener el árbol óptimo y valorar cada subdivisión entre todos los árboles posibles y conseguir el nodo raíz y los subsiguientes, el algoritmo deberá medir de alguna manera las predicciones logradas y valorarlas para comparar de entre todas y obtener la mejor. Para medir y valorar, utiliza diversas funciones, siendo las más conocidas y usadas los «Índice» y «Ganancia de información» que utiliza la denominada «entropía». La división de nodos continuará hasta que lleguemos a la profundidad máxima posible del árbol ó se limiten los nodos a una cantidad mínima de muestras en cada hoja.
A continuación describiremos muy brevemente cada una de las estrategias nombradas:

Índice Gini:

Se utiliza para atributos con valores continuos (Ejemplo precio de una casa). Esta función de coste mide el «grado de impureza» de los nodos, es decir, cuán desordenados o mezclados quedan los nodos una vez divididos**.** Deberemos minimizar ese GINI index.

Ganancia de información:

Se utiliza para atributos categóricos (cómo en hombre/mujer). Este criterio intenta estimar la información que aporta cada atributo basado en la «teoría de la información». Para medir la aleatoriedad de incertidumbre de un valor aleatorio de una variable «X» se define la Entropía.
Al obtener la medida de entropía de cada atributo, podemos calcular la ganancia de información del árbol. Deberemos maximizar esa ganancia.

https://scikit-learn.org/stable/modules/tree.html

```
from sklearn import tree
from sklearn.tree import export_graphviz
from pydotplus import graph_from_dot_data



#Se crea la instancia del árbol de decisión.

clf = tree.DecisionTreeClassifier()
```

- Problema y Ejemplo en Python
  En el siguiente ejemplo vamos a pasar medidas de zapatos y vamos a intentar predecir si las medidas que le pasamos es de hombre o mujer

```
#Se importa la librería sklearn el módulo tree

from sklearn import tree
from sklearn.tree import export_graphviz
from pydotplus import graph_from_dot_data



#Se crea la instancia del árbol de decisión.

clf = tree.DecisionTreeClassifier()
```

```
#[altura, peso, talla de zapato]

X = [[181, 80, 44], [177, 70, 43], [160, 60, 38], [154, 54, 37], [166, 65, 40],

    [190, 90, 47], [175, 64, 39],

    [177, 70, 40], [159, 55, 37], [171, 75, 42], [181, 85, 43]]

#La salida donde se dice si es hombre o mujer

Y = ['hombre', 'hombre', 'mujer', 'mujer', 'hombre', 'hombre', 'mujer', 'mujer',

    'mujer', 'hombre', 'hombre']

#Se le pasa los datos  X y Y

clf = clf.fit(X, Y)

feature_names = ['X', 'Y']

#Se definen los datos 1 y 2

dato1 = [190, 70, 43]

dato2 = [185, 62, 37]

prediction = clf.predict([dato1])

#Se muestra el resultado de la predicción de dato1

print(prediction)

prediction = clf.predict([dato2])

dot_data = export_graphviz(clf)
graph = graph_from_dot_data(dot_data)
graph.write_png('clf.png')
```

Principales parámetros:

criterion*{"gini", "entropy"}, default="gini"*

Representa el criterio utilizado para medir la pureza de una hoja.

splitter*{"best", "random"}, default="best"*

Es la estrategia utilizada para elegir la división. Best implica la mejor en absoluto y random elige la mejor de un conjunto aleatorio.

max_depth*int, default=None*

Es la máxima cantidad de divisiones que se propone para un árbol. Cuando se alcanza ese valor no se continua dividiendo.

min_samples_split *int or float, default=2*

Es la mínima cantidad de datos que debe tener un nodo para que se considere subdividirlo. Alcanzada una cantidad menor no se sigue subdividiendo.

min_samples_leaf *int or float, default=1*

Es la cantidad mínima de casos necesarios para ser una hoja.

max_leaf_node s*int, default=None*

Máximo número de hojas.

min_impurity_decrease*float, default=0.0*

Un nodo sólo aceptará ser dividido si la división propuesta consigue una disminución de la impureza mayor o igual al valor del parámetro

min_impurity_split *float, default=0*

Un nodo no se dividirá si su impureza es menor al valor del parámetro

ccp_alpha*non-negative float, default=0.0*

Parametro de complejidad utilizado para el Minimal Cost-Complexity Pruning.