



(<https://www.bigdatauniversity.com>)

## Decision Trees

En este ejercicio, aprenderás un algoritmo muy popular de machine learning llamado Árboles de Decisión. Utilizarás un algoritmo de clasificación para construir un modelo basado en datos históricos de pacientes y sus respectivos medicamentos. Luego, utilizarás el árbol de decisión recién entrenado para predecir la clase de paciente desconocido o para encontrar la droga adecuada para el mismo.

Import the Following Libraries:

- **numpy** (as **np**)
- **pandas**
- **DecisionTreeClassifier** from **sklearn.tree**

```
In [1]: import numpy as np
import pandas as pd
from sklearn.tree import DecisionTreeClassifier
```

### Acerca del set de datos

Imagina que eres un investigador médico recolectando datos para un estudio. Has colectado datos de un grupo de pacientes, todos sufrieron la misma enfermedad. Durante su tratamiento, cada paciente respondió a una de 5 medicaciones, Droga A, Droga B, Droga c, Droga x e y.

Parte de tu trabajo es construir un modelo para encontrar la droga apropiada para un próximo paciente con la misma enfermedad. El conjunto de características son Edad, Sexo, Presión Sanguínea y Colesterol. El objetivo es la droga ante la cual cada paciente respondió.

Este es un ejemplo de un clasificador binario donde puedes utilizar un set de entrenamiento del set de datos para construir un árbol de decisión para predecir la clase de pacientes desconocidos o para prescribirle a un nuevo paciente.

### Descargando los Datos

Para descargar los datos, utilizaremos `!wget` desde IBM Object Storage.

```
In [2]: !wget -O drug200.csv https://s3-api.us-gio.objectstorage.softlayer.net/cf-co
        -courses-data/CognitiveClass/ML0101ENV3/labs/drug200.csv

--2020-05-27 08:31:38-- https://s3-api.us-gio.objectstorage.softlayer.net/cf-
-courses-data/CognitiveClass/ML0101ENV3/labs/drug200.csv
Resolving s3-api.us-gio.objectstorage.softlayer.net (s3-api.us-gio.objectstor
age.softlayer.net)... 67.228.254.196
Connecting to s3-api.us-gio.objectstorage.softlayer.net (s3-api.us-gio.object
storage.softlayer.net)|67.228.254.196|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 6027 (5.9K) [text/csv]
Saving to: 'drug200.csv'

drug200.csv          100%[=====>]   5.89K  --.-KB/s   in 0s

2020-05-27 08:31:38 (12.9 MB/s) - 'drug200.csv' saved [6027/6027]
```

¿Sabías? Cuando se trata de Machine Learning, seguro trabajarás con grandes datasets (juego de datos). Entonces, ¿dónde podrás guardar esos datos? IBM ofrece una oportunidad única para las empresas, con 10 Tb de IBM Cloud Object Storage: [Regístrate ahora gratuitamente \(http://cocl.us/ML0101EN-IBM-Offer-CC\)](http://cocl.us/ML0101EN-IBM-Offer-CC)

ahora, lee los datos utilizando el marco de datos de panda:

```
In [13]: my_data = pd.read_csv("drug200.csv", delimiter=",")
        #my_data[0:5]
        my_data.head()
```

```
Out[13]:
```

	Age	Sex	BP	Cholesterol	Na_to_K	Drug
0	23	F	HIGH	HIGH	25.355	drugY
1	47	M	LOW	HIGH	13.093	drugC
2	47	M	LOW	HIGH	10.114	drugC
3	28	F	NORMAL	HIGH	7.798	drugX
4	61	F	LOW	HIGH	18.043	drugY

## Práctica

¿Cuál es el tamaño de los datos?

```
In [9]: # escribe tu código aquí
        my_data.shape
```

```
Out[9]: (200, 6)
```

## Pre-procesamiento

Utilizando **my\_data** como los datos de panda el archivo Drug.csv, declara las siguientes variables:

- **X** as the **Feature Matrix** (datos de my\_data)
- **y** como el **vector de respuesta (target)**

Elimina la columna que contiene el target ya que no posee valores numéricos.

```
In [103]: X = my_data[['Age', 'Sex', 'BP', 'Cholesterol', 'Na_to_K']].values
# Return a Numpy representation of the DataFrame
# DataFrame with mixed type columns(e.g., str/object, int64, float32)
# results in an ndarray of the broadest type that accommodates these mixed t
ypes (e.g., object).
my_data.dtypes
X[0:5]
```

```
Out[103]: array([[23, 'F', 'HIGH', 'HIGH', 25.355],
 [47, 'M', 'LOW', 'HIGH', 13.093],
 [47, 'M', 'LOW', 'HIGH', 10.113999999999999],
 [28, 'F', 'NORMAL', 'HIGH', 7.797999999999999],
 [61, 'F', 'LOW', 'HIGH', 18.043]], dtype=object)
```

Como te puedes imaginar, algunas características son de categoría, tales como **Sex** o **BP**. Desafortunadamente, los árboles de Decisión Sklearn no manejan variables categóricas. Pero las podemos convertir en valores numéricos.

**pandas.get\_dummies()** Convertir variable categórica en indicadores de variables.

```
In [104]: from sklearn import preprocessing

le_sex = preprocessing.LabelEncoder()
le_sex.fit(['F', 'M'])
X[:,1] = le_sex.transform(X[:,1])

le_BP = preprocessing.LabelEncoder()
le_BP.fit(['LOW', 'NORMAL', 'HIGH'])
X[:,2] = le_BP.transform(X[:,2])

le_Chol = preprocessing.LabelEncoder()
le_Chol.fit(['NORMAL', 'HIGH'])
X[:,3] = le_Chol.transform(X[:,3])

X[0:5]
```

```
Out[104]: array([[23, 0, 0, 0, 25.355],
 [47, 1, 1, 0, 13.093],
 [47, 1, 1, 0, 10.113999999999999],
 [28, 0, 2, 0, 7.797999999999999],
 [61, 0, 1, 0, 18.043]], dtype=object)
```

Ahora, podemos completar la variable objetivo (target).

```
In [100]: y = my_data["Drug"]
y[0:5]
```

```
Out[100]: 0    drugY
1    drugC
2    drugC
3    drugX
4    drugY
Name: Drug, dtype: object
```

## Configurando el Arbol de Decisión

Estaremos utilizando **entrenar/probar separar** en nuestro **árbol de decisión**. Importemos **train\_test\_split** de **sklearn.cross\_validation**.

```
In [101]: from sklearn.model_selection import train_test_split
```

Ahora **train\_test\_split** devolverá 4 parámetros diferentes. Los nombraremos:  
X\_trainset, X\_testset, y\_trainset, y\_testset

El **train\_test\_split** necesitará los parámetros:  
X, y, test\_size=0.3, and random\_state=3.

La **X** e **y** son los arreglos necesarios antes de la operación dividir/separar, **test\_size** representa el grado del dataset de pruebas, y el **random\_state** asegura que obtendremos las mismas divisiones.

```
In [169]: X_trainset, X_testset, y_trainset, y_testset = train_test_split(X, y, test_s
           : ize=0.3, random_state=3)
           print('X_trainset: ',X_trainset[0:5])
           print('y_trainset: ',y_trainset[0:5])
```

```
X_trainset: [[26 0 0 1 19.160999999999998]
             [41 0 2 1 22.905]
             [28 0 2 0 19.675]
             [19 0 0 0 13.312999999999999]
             [50 1 2 1 15.79]]
y_trainset: 77      drugY
73      drugY
71      drugY
78      drugA
42      drugY
Name: Drug, dtype: object
```

## Práctica

Imprimir la forma X\_trainset e y\_trainset. Asegurarse que las dimensiones coincidan

```
In [51]: # your code
           print('Dimension of X_trainset:',X_trainset.shape)
           print('Dimension of y_trainset:',y_trainset.shape)

           if X_trainset.shape[0] == y_trainset.shape[0]:
               print('OK, Dimensions match')
           else:
               print("NOK, Dimension don't match")
```

```
Dimension of X_trainset: (140, 5)
Dimension of y_trainset: (140,)
OK, Dimensions match
```

Imprimir la forma de X\_testset e y\_testset. Asegurarse que las dimensiones coincidan

```
In [52]: # tu código
print('Dimension of X_testset:',X_testset.shape)
print('Dimension of y_testset:',y_testset.shape)

if X_testset.shape[0] == y_testset.shape[0]:
    print('OK, Dimensions match')
else:
    print("NOK, Dimension don't match")

Dimension of X_testset: (60, 5)
Dimension of y_testset: (60,)
OK, Dimensions match
```

## Modelando

Primero crearemos una instancia del **DecisionTreeClassifier** llamada **drugTree**.

Dentro del clasificador, especificaremos *criterion="entropy"* para que podamos ver la nueva información de cada nodo.

```
In [53]: drugTree = DecisionTreeClassifier(criterion="entropy", max_depth = 4)
drugTree # muestra los parámetros por omisión

Out[53]: DecisionTreeClassifier(class_weight=None, criterion='entropy', max_depth=4,
max_features=None, max_leaf_nodes=None,
min_impurity_decrease=0.0, min_impurity_split=None,
min_samples_leaf=1, min_samples_split=2,
min_weight_fraction_leaf=0.0, presort=False, random_state=None,
splitter='best')
```

Luego, adaptaremos los datos con la matriz de entrenamiento **X\_trainset** y el vector de respuesta **y\_trainset**

```
In [54]: drugTree.fit(X_trainset,y_trainset)

Out[54]: DecisionTreeClassifier(class_weight=None, criterion='entropy', max_depth=4,
max_features=None, max_leaf_nodes=None,
min_impurity_decrease=0.0, min_impurity_split=None,
min_samples_leaf=1, min_samples_split=2,
min_weight_fraction_leaf=0.0, presort=False, random_state=None,
splitter='best')
```

## Predicción

Ahora hagamos algunas **predicciones** en el dataset de pruebas y guardémoslas en una variable llamada **predTree**.

```
In [55]: predTree = drugTree.predict(X_testset)
```

Puedes imprimir **predTree** y **y\_testset** si quieres comparar visualmente la predicción con los valores actuales.

```
In [ ]:
```

## Evaluación

Luego, importemos **metrics** de sklearn y revisemos la precisión de nuestro modelo.

```
In [57]: from sklearn import metrics
import matplotlib.pyplot as plt
print("Precisión de los Arboles de Decisión: ", metrics.accuracy_score(y_testset, predTree))
```

Precisión de los Arboles de Decisión: 0.9833333333333333

**Accuracy classification score** calcula la precisión del subconjunto: las etiquetas predichas para una muestra deben coincidir con las correspondientes etiquetas en `y_true`.

En la clasificación multietiqueta, la función devuelve un subconjunto de precisión. Si el conjunto de etiquetas predichas para una muestra coincide totalmente con el conjunto de etiquetas, entonces la precisión del subconjunto es 1.0; de no ser así, es 0.0.

## Práctica

¿Puedes calcular la precisión sin sklearn ?

```
In [141]: # tu código aquí
          from sklearn import preprocessing

          le_predTree_drug = preprocessing.LabelEncoder()
          le_predTree_drug.fit(['drugA', 'drugB', 'drugC', 'drugX', 'drugY'])

          #Predict_matrix[:,1] = le_y_testset_drug.transform((Predict_matrix[:,1]))
```

```
40      drugY
51      drugX
139     drugX
197     drugX
170     drugX
82      drugC
183     drugY
46      drugA
70      drugB
100     drugA
179     drugY
83      drugA
25      drugY
190     drugY
159     drugX
173     drugY
95      drugX
3       drugX
41      drugB
58      drugX
14      drugX
143     drugY
12      drugY
6       drugY
182     drugX
161     drugB
128     drugY
122     drugY
101     drugA
86      drugX
64      drugB
47      drugC
158     drugC
34      drugX
38      drugX
196     drugC
4       drugY
72      drugX
67      drugX
145     drugX
156     drugA
115     drugY
155     drugC
15      drugY
61      drugA
175     drugY
120     drugY
130     drugY
23      drugY
153     drugX
31      drugB
103     drugX
89      drugY
132     drugX
109     drugY
126     drugY
17      drugA
30      drugX
178     drugY
162     drugX
Name: Drug, dtype: object
```



```

-----
KeyError                                Traceback (most recent call last)
<ipython-input-141-4f40c9bc7268> in <module>
     10 print(M[0])
     11
--> 12 y_testset[0]

~/conda/envs/python/lib/python3.6/site-packages/pandas/core/series.py in __getitem__(self, key)
     869         key = com.apply_if_callable(key, self)
     870         try:
--> 871             result = self.index.get_value(self, key)
     872
     873             if not is_scalar(result):

~/conda/envs/python/lib/python3.6/site-packages/pandas/core/indexes/base.py in get_value(self, series, key)
     4402         k = self._convert_scalar_indexer(k, kind="getitem")
     4403         try:
-> 4404             return self._engine.get_value(s, k, tz=getattr(series.dtype, "tz", None))
     4405         except KeyError as e1:
     4406             if len(self) > 0 and (self.holds_integer() or self.is_bool):
lean():

pandas/_libs/index.pyx in pandas._libs.index.IndexEngine.get_value()

pandas/_libs/index.pyx in pandas._libs.index.IndexEngine.get_value()

pandas/_libs/index.pyx in pandas._libs.index.IndexEngine.get_loc()

pandas/_libs/hashtable_class_helper.pxi in pandas._libs.hashtable.Int64HashTable.get_item()

pandas/_libs/hashtable_class_helper.pxi in pandas._libs.hashtable.Int64HashTable.get_item()

KeyError: 0

```

## Visualización

Observemos el árbol

```

In [106]: from sklearn.externals.six import StringIO
import pydotplus
import matplotlib.image as mpimg
from sklearn import tree
%matplotlib inline

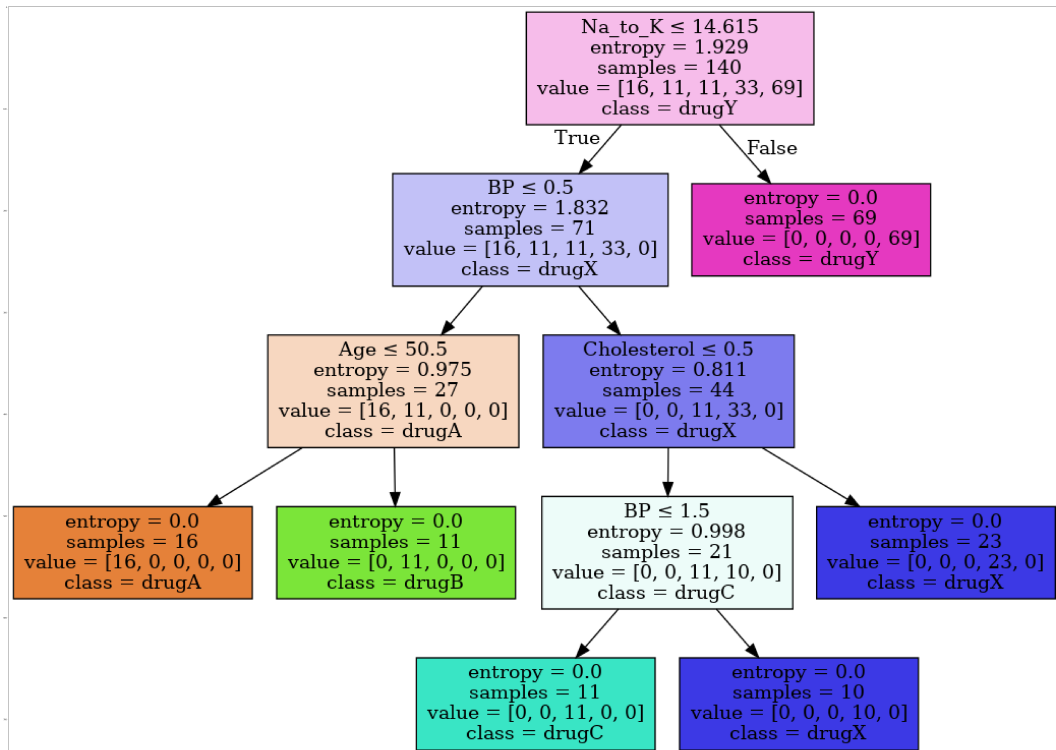
```

```

In [108]: dot_data = StringIO()
filename = "drugtree.png"
featureNames = my_data.columns[0:5]
targetNames = my_data["Drug"].unique().tolist()
out=tree.export_graphviz(drugTree,feature_names=featureNames, out_file=dot_data,
                        class_names= np.unique(y_trainset), filled=True, special_characters=True,rotate=False)
graph = pydotplus.graph_from_dot_data(dot_data.getvalue())
graph.write_png(filename)
img = mpimg.imread(filename)
plt.figure(figsize=(100, 200))
plt.imshow(img,interpolation='nearest')

```

Out[108]: <matplotlib.image.AxesImage at 0x7faca4402da0>



## ¿Deseas aprender más?

IBM SPSS Modeler es una plataforma para analytics que contiene varios algoritmos de machine learning. Fue diseñada para acercar inteligencia predictiva a las decisiones hechas por individuos, grupos, sistemas, toda la empresa. Un free trial está disponible a través de este curso en: [SPSS Modeler \(http://cocl.us/ML0101EN-SPSSModeler\)](http://cocl.us/ML0101EN-SPSSModeler).

Asi mismo, puedes utilizar Watson Studio para ejecutar estos notebooks más rápido y con datasets más grandes. Watson Studio es una solución en la nube lider de IBM's para científicos de datos, construída por científicos de datos. Con Jupyter notebooks, RStudio, Apache Spark y librerías conocidas pre instaladas en la nube, Watson Studio posibilita a los científicos de datos colaborar en sus proyectos sin tener que instalar nada. Sumate a la comunidad de usuarios Watson Studio hoy mismo por medio de una cuenta gratuita en [Watson Studio \(https://cocl.us/ML0101EN\\_DSX\)](https://cocl.us/ML0101EN_DSX)

## ¡Gracias por completar esta lección!

Laboratorio creado por: [Saeed Aghabozorgi \(https://ca.linkedin.com/in/saeedaghabozorgi\)](https://ca.linkedin.com/in/saeedaghabozorgi)

---

Copyright © 2018 [Cognitive Class \(https://cocl.us/DX0108EN\\_CC\)](https://cocl.us/DX0108EN_CC). Este lab y su código fuente fueron registrados bajo los términos de [MIT License \(https://bigdatauniversity.com/mit-license/\)](https://bigdatauniversity.com/mit-license/).