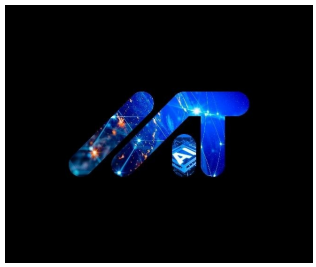


Decision Tree

In this tutorial, you will learn a popular machine learning algorithm, Decision Trees. You will use this classification algorithm to build a model from the historical data of patients, and their response to different medications. Then you will use the trained decision tree to predict the class of an unknown patient, or to find a proper drug for a new patient.

Importing required packages

In [25]:
```
import matplotlib.pyplot as plt
import pandas as pd
from sklearn.tree import DecisionTreeClassifier
import sklearn.tree as tree
from sklearn import preprocessing
from sklearn.model_selection import train_test_split
from sklearn import metrics
import numpy as np
from sklearn.model_selection import train_test_split
%matplotlib inline
```

Let's download and import the data on China's GDP using *pandas* `read_csv()` method.

Download Dataset

Understanding the Data

`drugdata.csv`:

Imagine that you are a medical researcher compiling data for a study. You have collected data about a set of patients, all of whom suffered from the same illness. During their course of treatment, each patient responded to one of 5 medications, Drug A, Drug B, Drug c, Drug x and y.

Part of your job is to build a model to find out which drug might be appropriate for a future patient with the same illness. The features of this dataset are Age, Sex, Blood Pressure, and the Cholesterol of the patients, and the target is the drug that each patient responded to.

It is a sample of multiclass classifier, and you can use the training part of the dataset to build a decision tree, and then use it to predict the class of an unknown patient, or to prescribe a drug to a new patient.

Reading the data

In [2]:
```
df = pd.read_csv("drugdata.csv")

# take a look at the dataset
df.head()
```

Out[2]:

	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

In [3]:
```
df.shape
```

Out[3]:
```
(200, 6)
```

Data Preprocessing

In [10]:
```
df.columns
```

Out[10]:
```
Index(['Age', 'Sex', 'BP', 'Cholesterol', 'Na_to_K', 'Drug'], dtype='object')
```

In [14]:
```
df.dtypes
```

Out[14]:
```
Age              int64
Sex             object
BP              object
Cholesterol     object
Na_to_K        float64
Drug            object
dtype: object
```

In [11]:
```
X=df[['Age', 'Sex', 'BP', 'Cholesterol', 'Na_to_K']].values
Y=df.Drug
```

As you can see, some features in this dataset are categorical, such as **Sex** or **BP**. Unfortunately, Sklearn Decision Trees does not handle categorical variables. We can still convert these features to numerical values using **pandas.get_dummies()** to convert the categorical variable into dummy/indicator variables.

In [12]:
```
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[12]:
```
array([[23, 0, 0, 0, 25.355],
       [47, 1, 1, 0, 13.093],
       [47, 1, 1, 0, 10.114],
       [28, 0, 2, 0, 7.798],
       [61, 0, 1, 0, 18.043]], dtype=object)
```

Setting up the Decision Tree

We will be using **train/test split** on our **decision tree**. Let's import **train_test_split** from **sklearn.cross_validation**.

Now **train_test_split** will return 4 different parameters. We will name them:
X_trainset, X_testset, y_trainset, y_testset

The **train_test_split** will need the parameters:
X, y, test_size=0.3, and random_state=3.

The **X** and **Y** are the arrays required before the split, the **test_size** represents the ratio of the testing dataset, and the **random_state** ensures that we obtain the same splits.

In [16]:
```
X_train, X_test, y_train, y_test  = train_test_split(X, Y, test_size=0.3, random_state=3)
```

In [17]:
```
X_train.shape
```

Out[17]:
```
(140, 5)
```

In [18]:
```
X_test.shape
```

Out[18]:
```
(60, 5)
```

In [19]:
```
y_train.shape
```

Out[19]:
```
(140,)
```

In [20]:
```
y_test.shape
```

Out[20]:
```
(60,)
```

Modeling

We will first create an instance of the **DecisionTreeClassifier** called **drugTree**.
Inside of the classifier, specify *criterion="entropy"* so we can see the information gain of each node.

In [21]:
```
drugTree = DecisionTreeClassifier(criterion="entropy", max_depth = 4)
drugTree # it shows the default parameters
```

Out[21]:
```
DecisionTreeClassifier(criterion='entropy', max_depth=4)
```

Next, we will fit the data with the training feature matrix **X_train** and training response vector **y_train**

In [22]:
```
drugTree.fit(X_train,y_train)
```

Out[22]:
```
DecisionTreeClassifier(criterion='entropy', max_depth=4)
```

Prediction

Let's make some **predictions** on the testing dataset and store it into a variable called **y_pred**.

In [28]:
```
y_pred = drugTree.predict(X_test)
```

Evaluation

In [29]:
```
print("DecisionTrees's Accuracy: ", metrics.accuracy_score(y_test, y_pred))
```

```
DecisionTrees's Accuracy:  0.9833333333333333
```

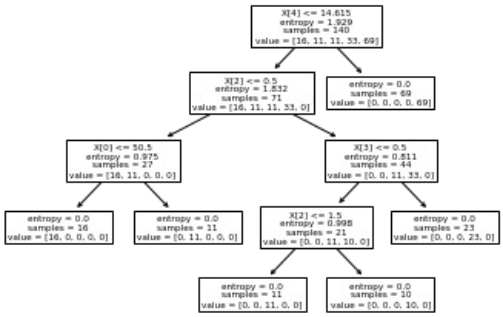
Accuracy classification score computes subset accuracy: the set of labels predicted for a sample must exactly match the corresponding set of labels in y_true.

In multilabel classification, the function returns the subset accuracy. If the entire set of predicted labels for a sample strictly matches with the true set of labels, then the subset accuracy is 1.0; otherwise it is 0.0.

Visualization

Let's visualize the tree

In [31]:
```
tree.plot_tree(drugTree)
plt.show()
```



Thank you

Author

Moazzam Ali
