Decision Trees

Estimated time needed: **15** minutes

Objectives

After completing this lab you will be able to:

- Develop a classification model using Decision Tree Algorithm

In this lab exercise, 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.

Table of contents

Import the Following Libraries:

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

if you uisng you own version comment out

In [1]:
```
import piplite
await piplite.install(['pandas'])
await piplite.install(['matplotlib'])
await piplite.install(['numpy'])
await piplite.install(['scikit-learn'])
```

In [7]:
```
import numpy as np
import pandas as pd
from sklearn.tree import DecisionTreeClassifier
import sklearn.tree as tree
```

In [3]:
```
from pyodide.http import pyfetch

async def download(url, filename):
    response = await pyfetch(url)
    if response.status == 200:
        with open(filename, "wb") as f:
            f.write(await response.bytes())
```

About the dataset

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.

Downloading the Data

To download the data, we will use !wget to download it from IBM Object Storage.

In [4]:
```
path= 'https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBMDeveloperS
await download(path,"drug200.csv")
path="drug200.csv"
```

Now, read the data using pandas dataframe:

In [9]:
```
my_data = pd.read_csv("drug200.csv", delimiter=",")
my_data[0:5]
```

Out[9]:

	Age	Sex	BP	Cholesterol	Na_to_K	Drug
0	23	F	HIGH	HIGH	25.355	drugY

	Age	Sex	BP	Cholesterol	Na_to_K	Drug
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

Practice

What is the size of data?

In [11]:
```
my_data.shape
```

Out[11]:
```
(200, 6)
```

▶ Click here for the solution

Pre-processing

Using **my_data** as the Drug.csv data read by pandas, declare the following variables:

- **X** as the **Feature Matrix** (data of my_data)
- **y** as the **response vector** (target)

Remove the column containing the target name since it doesn't contain numeric values.

In [12]:
```
X = my_data[['Age', 'Sex', 'BP', 'Cholesterol', 'Na_to_K']].values
X[0:5]
```

Out[12]:
```
array([[23, 'F', 'HIGH', 'HIGH', 25.355],
       [47, 'M', 'LOW', 'HIGH', 13.093],
       [47, 'M', 'LOW', 'HIGH', 10.114],
       [28, 'F', 'NORMAL', 'HIGH', 7.798],
       [61, 'F', 'LOW', 'HIGH', 18.043]], dtype=object)
```

As you may figure out, 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 **LabelEncoder** to convert the categorical variable into numerical variables.

In [13]:
```
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[13]:
```
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)
```

Now we can fill the target variable.

In [14]:
```
y = my_data["Drug"]
y[0:5]
```

Out[14]:
```
0    drugY
1    drugC
2    drugC
3    drugX
4    drugY
Name: Drug, 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**.

In [15]:
```
from sklearn.model_selection import train_test_split
```

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_trainset, X_testset, y_trainset, y_testset = train_test_split(X, y, test_size=0.3, ra
```

Practice

Print the shape of X_trainset and y_trainset. Ensure that the dimensions match.

In [21]:
```
print(X_trainset.shape)
print(y_trainset.shape)

print('Shape of X training set {}'.format(X_trainset.shape),'&',' Size of Y training se
(140, 5)
(140,)
Shape of X training set (140, 5) &  Size of Y training set (140,)
```

▶ Click here for the solution

Print the shape of X_testset and y_testset. Ensure that the dimensions match.

In [23]:
```
print(X_testset.shape)
print(y_testset.shape)

print('Shape of X test set {}'.format(X_testset.shape),'&','Size of y test set {}'.form
```

```
(60, 5)
(60,)
Shape of X test set (60, 5) & Size of y test set (60,)
```

▶ Click here for the solution

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 [25]:
```
drugTree = DecisionTreeClassifier(criterion="entropy", max_depth = 4)
drugTree # it shows the default parameters
```

Out[25]:
```
▾                DecisionTreeClassifier
DecisionTreeClassifier(criterion='entropy', max_depth=4)
```

Next, we will fit the data with the training feature matrix **X_trainset** and training response vector **y_trainset**

In [26]:
```
drugTree.fit(X_trainset,y_trainset)
```

Out[26]:
```
▾                DecisionTreeClassifier
DecisionTreeClassifier(criterion='entropy', max_depth=4)
```

Prediction

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

In [27]:
```
predTree = drugTree.predict(X_testset)
```

You can print out **predTree** and **y_testset** if you want to visually compare the predictions to the actual values.

In [28]:

```
print (predTree [0:5])
print (y_testset [0:5])
```

```
['drugY' 'drugX' 'drugX' 'drugX' 'drugX']
40     drugY
51     drugX
139    drugX
197    drugX
170    drugX
Name: Drug, dtype: object
```

Evaluation

Next, let's import **metrics** from sklearn and check the accuracy of our model.

In [29]:

```
from sklearn import metrics
import matplotlib.pyplot as plt
print("DecisionTrees's Accuracy: ", metrics.accuracy_score(y_testset, predTree))
```

```
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

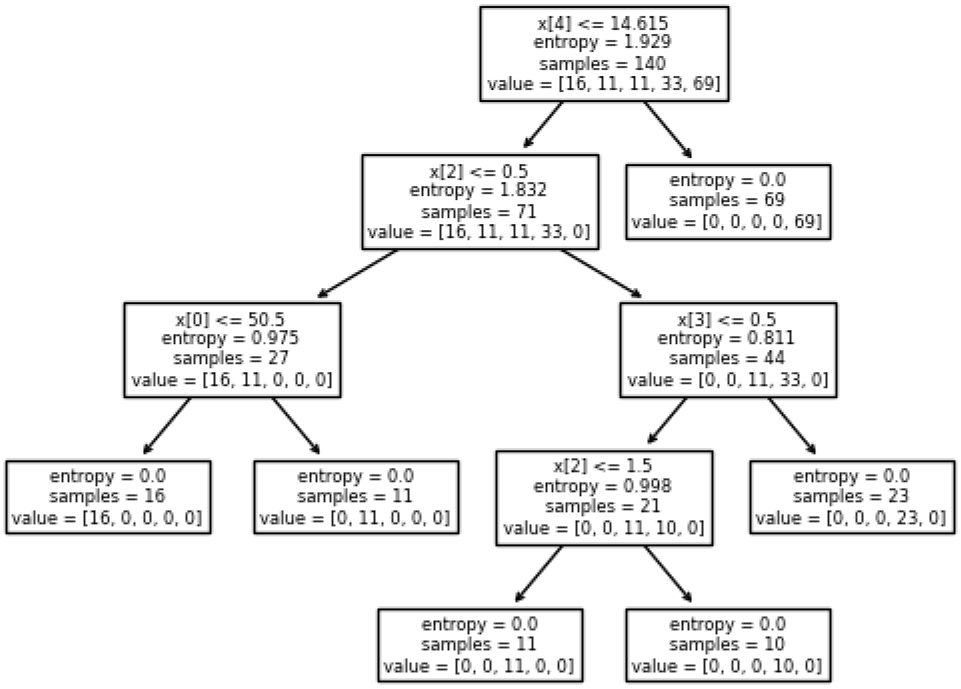
Let's visualize the tree

In [30]:

```
# Notice: You might need to uncomment and install the pydotplus and graphviz libraries
#!conda install -c conda-forge pydotplus -y
#!conda install -c conda-forge python-graphviz -y
```

In [31]:

```
tree.plot_tree(drugTree)
plt.show()
```



Thank you for completing this lab!

Author

Saeed Aghabozorgi

Other Contributors

Joseph Santarcangelo

<!--

Change Log

Date (YYYY-MM-DD)	Version	Changed By	Change Description
2023-04-05	2.3	Anita Verma	Changed pandas.get_dummies() to LabelEncoder
2020-11-20	2.2	Lakshmi	Changed import statement of StringIO
2020-11-03	2.1	Lakshmi	Changed URL of the csv
2020-08-27	2.0	Lavanya	Moved lab to course repo in GitLab

--!>