Title: Drug Response Prediction

Objective: In this mini project, I aim to delve into the Decision Trees machine learning algorithm. By constructing a model from past patient data and associated medication responses, my objective is to harness the power of Decision Trees. This will enable me to predict the classification of an unknown patient or suggest the most suitable drug for a new patient. Through hands-on exploration, my goal is to gain a deeper understanding and practical experience in applying this classification algorithm to healthcare contexts.

Table of contents

In [1]:
```
# Surpress warnings:
def warn(*args, **kwargs):
    pass
import warnings
warnings.warn = warn
```

In [2]:
```
import sys
import numpy as np
import pandas as pd
from sklearn.tree import DecisionTreeClassifier
import sklearn.tree as tree
```

About the dataset

Premise: As a medical researcher undertaking a comprehensive study, I've gathered crucial data from a group of patients, all grappling with a common illness. Throughout the course of their treatment, each individual has exhibited a unique response to one of five medications: Drug A, Drug B, Drug C, Drug X, and Drug Y. This dataset holds the key to unraveling patterns and insights that can guide personalized and effective treatment strategies for patients in similar circumstances.

Part of my 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 can be used for the training part of the dataset to build a decision tree, and then to predict the class of an unknown patient, or to prescribe a drug to a new patient.

Downloading the Data

Using pandas library to read the data directly into a dataframe from IBM Object Storage.

In [3]:
```
my_data = pd.read_csv('https://cf-courses-data.s3.us.cloud-object-storage.appdom
my_data.head()
```

Out[3]:

	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 [36]:
```
#Size of data

my_data.shape
```

Out[36]:
```
(200, 6)
```

▶ Click here for the solution

Pre-processing

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

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

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

In [6]:
```
X = my_data[['Age', 'Sex', 'BP', 'Cholesterol', 'Na_to_K']].values
X[0:5]
```

Out[6]:
```
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)
```

In [37]:
```
#Converting categorical features to dummy/indicator using the **LabelEncoder() m
```

In [7]:
```
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[7]:
```
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)
```

In [38]:
```
y = my_data["Drug"] #filling the target variable
y[0:5]
```

Out[38]:
```
0    drugY
1    drugC
2    drugC
3    drugX
4    drugY
Name: Drug, dtype: object
```

Setting up the Decision Tree

Using **train/test split** on the **decision tree**. Let's import **train_test_split** from **sklearn.cross_validation**.

In [39]:
```
from sklearn.model_selection import train_test_split
```

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 [13]:
```
X_trainset, X_testset, y_trainset, y_testset = train_test_split(X, y, test_size=
```

In [40]:
```
print('Shape of X training set{}'.format(X_trainset.shape), '&', 'Size of Y Trai
```

```
Shape of X training set(140, 5) & Size of Y Training set(140,)
```

In [41]:
```
print('Shape of X training set {}'.format(X_testset.shape),'&',' Size of Y train
```

```
Shape of X training set (60, 5) &  Size of Y training set (60,)
```

Modeling

Creating an instance of the **DecisionTreeClassifier** called **drugTree**.
Inside of the classifier, *criterion="entropy"* to can see the information gain of each node.

In [20]:
```
drugTree = DecisionTreeClassifier(criterion="entropy", max_depth = 4)
drugTree # it shows the default parameters
```

Out[20]:
```
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')
```

Fitting the data with the training feature matrix **X_trainset** and training response vector **y_trainset**

In [21]:
```
drugTree.fit(X_trainset,y_trainset)
```

Out[21]:
```
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')
```

Predictions

In [25]:
```
predTree = drugTree.predict(X_testset)
```

In [26]:
```
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

Checking the accuracy of our model.

In [27]:
```
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

In [28]:
```
!conda install -c conda-forge pydotplus -y
!conda install -c conda-forge python-graphviz -y
```

```
Retrieving notices: ...working... done
Collecting package metadata (current_repodata.json): | WARNING conda.models.versi
on:get_matcher(546): Using .* with relational operator is superfluous and depreca
ted and will be removed in a future version of conda. Your spec was 1.7.1.*, but
conda is ignoring the .* and treating it as 1.7.1
done
Solving environment: done


==> WARNING: A newer version of conda exists. <==
  current version: 23.3.1
  latest version: 23.11.0

Please update conda by running

    $ conda update -n base -c conda-forge conda

Or to minimize the number of packages updated during conda update use

     conda install conda=23.11.0



## Package Plan ##

  environment location: /home/jupyterlab/conda/envs/python

  added / updated specs:
    - pydotplus


The following packages will be downloaded:

    package                    |            build
    ---------------------------|-----------------
    ca-certificates-2023.11.17 |       hbcca054_0         151 KB  conda-forge
    certifi-2023.11.17         |     pyhd8ed1ab_0         155 KB  conda-forge
    openssl-1.1.1w             |       hd590300_0         1.9 MB  conda-forge
    pydotplus-2.0.2            |     pyh243d235_4          24 KB  conda-forge
    ------------------------------------------------------------
                                           Total:         2.2 MB

The following NEW packages will be INSTALLED:

  pydotplus          conda-forge/noarch::pydotplus-2.0.2-pyh243d235_4

The following packages will be UPDATED:

  ca-certificates                     2023.5.7-hbcca054_0 --> 2023.11.17-hbcca0
54_0
  certifi                           2023.5.7-pyhd8ed1ab_0 --> 2023.11.17-pyhd8e
d1ab_0
  openssl                              1.1.1t-h0b41bf4_0 --> 1.1.1w-hd590300_0



Downloading and Extracting Packages
ca-certificates-2023 | 151 KB    |                                       |   0%
openssl-1.1.1w       | 1.9 MB    |                                       |   0%
```

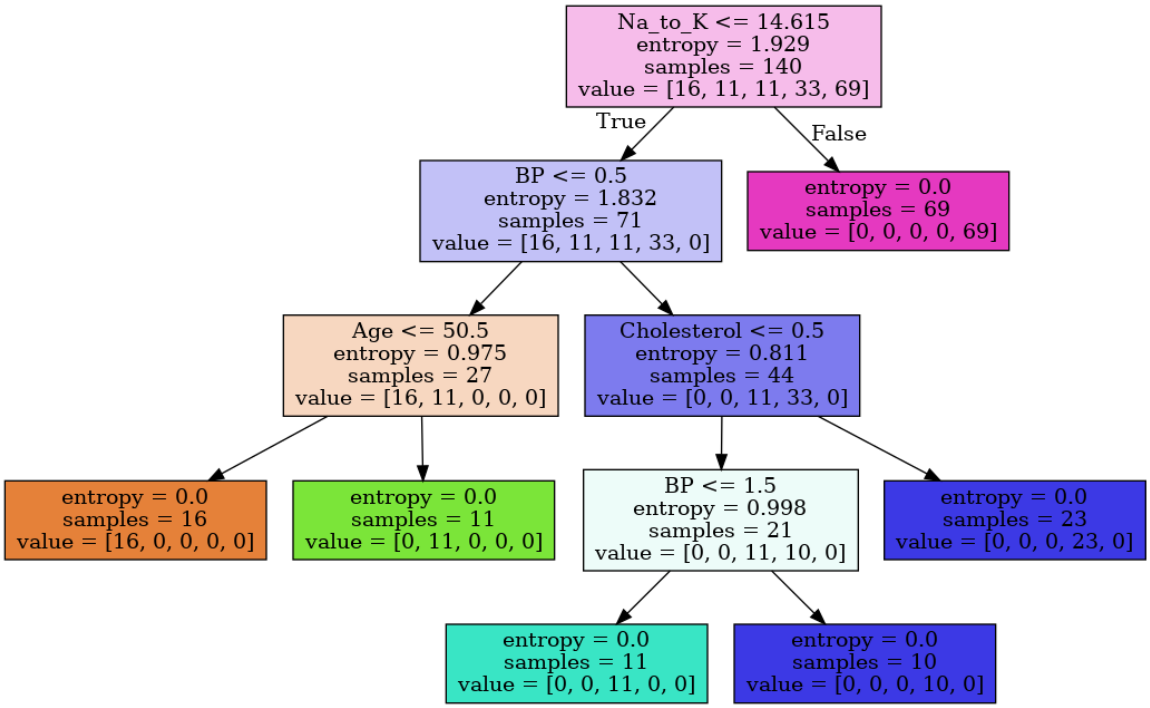
```
pydotplus-2.0.2      | 24 KB     |                                       |   0%


certifi-2023.11.17   | 155 KB    |                                       |   0%
ca-certificates-2023 | 151 KB    | ###9                                  |  11%

pydotplus-2.0.2      | 24 KB     | ########################6             |  67%


ca-certificates-2023 | 151 KB    | ##################################### | 100%
openssl-1.1.1w       | 1.9 MB    | ##############################3       |  82%

pydotplus-2.0.2      | 24 KB     | ##################################### | 100%


certifi-2023.11.17   | 155 KB    | ##################################### | 100%


certifi-2023.11.17   | 155 KB    | ##################################### | 100%




Preparing transaction: done
Verifying transaction: done
Executing transaction: done
Collecting package metadata (current_repodata.json): / WARNING conda.models.versi
on:get_matcher(546): Using .* with relational operator is superfluous and depreca
ted and will be removed in a future version of conda. Your spec was 1.7.1.*, but
conda is ignoring the .* and treating it as 1.7.1
done
Solving environment: done


==> WARNING: A newer version of conda exists. <==
  current version: 23.3.1
  latest version: 23.11.0

Please update conda by running

    $ conda update -n base -c conda-forge conda

Or to minimize the number of packages updated during conda update use

     conda install conda=23.11.0



## Package Plan ##

  environment location: /home/jupyterlab/conda/envs/python

  added / updated specs:
    - python-graphviz


The following packages will be downloaded:
```

```
  package                    |            build
  ---------------------------|-----------------
  python-graphviz-0.12       |             py_0          18 KB  conda-forge
  ------------------------------------------------------------
                                       Total:          18 KB

The following NEW packages will be INSTALLED:

  python-graphviz    conda-forge/noarch::python-graphviz-0.12-py_0


Downloading and Extracting Packages

Preparing transaction: done
Verifying transaction: done
Executing transaction: done
```

In [33]:
```
from sklearn.tree import export_graphviz
export_graphviz(drugTree, out_file='tree.dot', filled=True, feature_names=['Age'
!dot -Tpng tree.dot -o tree.png
```

In [35]:
```
from IPython.display import Image


Image(filename='tree.png')
```

Out[35]:



In [ ]: