ML0101EN-Clas-Decision-Trees-drug-py-v1

March 6, 2019

Decision Trees

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

Table of contents

```
<ol>
    <li><a href="#about_dataset">About the dataset</a></li>
    <li><a href="#downloading_data">Downloading the Data</a></li>
    <li><a href="#pre-processing">Pre-processing</a></li>
    <li><a href="#setting_up_tree">Setting up the Decision Tree</a></li>
    <li><a href="#modeling">Modeling</a></li>
    <li><a href="#prediction">Prediction</a></li>
    <li><a href="#evaluation">Evaluation</a></li>
    <li><a href="#visualization">Visualization</a></li>
</ol>
```

Import the Following Libraries:

```
<li> <b>numpy (as np)</b> </li>
<li> <b>pandas</b> </li>
<li> <b>DecisionTreeClassifier</b> from <b>sklearn.tree</b> </li>
```

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

```
<h2>About the dataset</h2>
Imagine that you are a medical researcher compiling data for a study. You have collected data ab
<br>
<br>
Part of your job is to build a model to find out which drug might be appropriate for a future pa
<br>
<br>
It is a sample of binary 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 a unknown patient, or to presc
```

```
<h2>Downloading the Data</h2>
To download the data, we will use !wget to download it from IBM Object Storage.

In [2]: !wget -O drug200.csv https://s3-api.us-geo.objectstorage.softlayer.net/cf-courses-data/C

--2019-03-06 13:42:55--  https://s3-api.us-geo.objectstorage.softlayer.net/cf-courses-data/Cogni
Resolving s3-api.us-geo.objectstorage.softlayer.net (s3-api.us-geo.objectstorage.softlayer.net).
Connecting to s3-api.us-geo.objectstorage.softlayer.net (s3-api.us-geo.objectstorage.softlayer.n
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

2019-03-06 13:42:56 (92.7 MB/s) - drug200.csv saved [6027/6027]
```

Did you know? When it comes to Machine Learning, you will likely be working with large datasets. As a business, where can you host your data? IBM is offering a unique opportunity for businesses, with 10 Tb of IBM Cloud Object Storage: Sign up now for free

now, read data using pandas dataframe:

```
In [3]: my_data = pd.read_csv("drug200.csv", delimiter=",")
        my_data[0:5]

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

<h3>Practice</h3>
What is the size of data?

In [4]: # write your code here

<h2>Pre-processing</h2>
```

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

```
<li> <b> X </b> as the <b> Feature Matrix </b> (data of my_data) </li>
<li> <b> y </b> as the <b> response vector (target) </b> </li>
```

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

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

```
Out[5]: 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)
```

As you may figure out, some features in this dataset are categorical such as **Sex** or **BP**. Unfortunately, Sklearn Decision Trees do not handle categorical variables. But still we can convert these features to numerical values. **pandas.get_dummies()** Convert categorical variable into dummy/indicator variables.

```
In [6]: 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[6]: 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)
```

Now we can fill the target variable.

```
In [7]: y = my_data["Drug"]
        y[0:5]

Out[7]: 0    drugY
        1    drugC
        2    drugC
        3    drugX
        4    drugY
        Name: Drug, dtype: object
```

```
<h2>Setting up the Decision Tree</h2>
We will be using <b>train/test split</b> on our <b>decision tree</b>. Let's import <b>train_test
```

```
In [8]: 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 [9]: X_trainset, X_testset, y_trainset, y_testset = train_test_split(X, y, test_size=0.3, ran
```

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

```
In [10]: # your code
```

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

```
In [11]: # your code
```

```
<h2>Modeling</h2>
We will first create an instance of the <b>DecisionTreeClassifier</b> called <b>drugTree</b>.<br
Inside of the classifier, specify <i> criterion="entropy" </i> so we can see the information gai
```

```
In [12]: drugTree = DecisionTreeClassifier(criterion="entropy", max_depth = 4)
         drugTree # it shows the default parameters
```

```
Out[12]: 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')
```

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

```
In [13]: drugTree.fit(X_trainset,y_trainset)
```

```
Out[13]: 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')
```

```
<h2>Prediction</h2>
Let's make some <b>predictions</b> on the testing dataset and store it into a variable called <b
```

```
In [14]: predTree = drugTree.predict(X_testset)
```

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

```
In [15]: 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
```

```
<h2>Evaluation</h2>
Next, let's import <b>metrics</b> from sklearn and check the accuracy of our model.
```

```
In [16]: 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 match with the true set of labels, then the subset accuracy is 1.0; otherwise it is 0.0.

0.1 Practice

Can you calculate the accuracy score without sklearn ?

```
In [17]: # your code here
```

```
<h2>Visualization</h2>
Lets visualize the tree
```

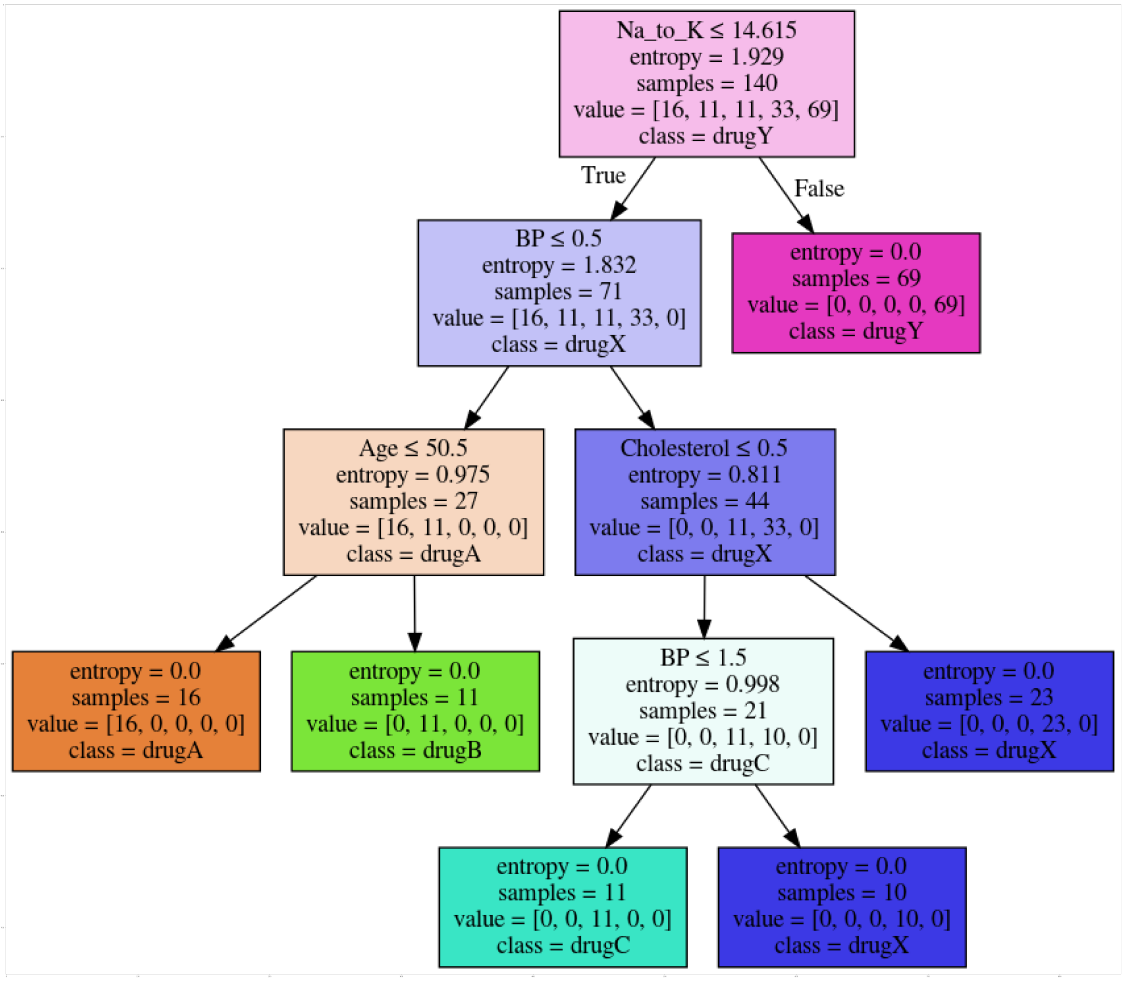
```
In [18]: # Notice: You might need to uncomment and install the pydotplus and graphviz libraries
         # !conda install -c conda-forge pydotplus -y
         # !conda install -c conda-forge python-graphviz -y
```

```
In [19]: from sklearn.externals.six import StringIO
         import pydotplus
         import matplotlib.image as mpimg
         from sklearn import tree
         %matplotlib inline
```

```
In [20]: 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_
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[20]: <matplotlib.image.AxesImage at 0x7f40aa247278>



Want to learn more?

IBM SPSS Modeler is a comprehensive analytics platform that has many machine learning algorithms. It has been designed to bring predictive intelligence to decisions made by individuals, by groups, by systems – by your enterprise as a whole. A free trial is available through this course, available here: SPSS Modeler

Also, you can use Watson Studio to run these notebooks faster with bigger datasets. Watson Studio is IBM's leading cloud solution for data scientists, built by data scientists. With Jupyter

notebooks, RStudio, Apache Spark and popular libraries pre-packaged in the cloud, Watson Studio enables data scientists to collaborate on their projects without having to install anything. Join the fast-growing community of Watson Studio users today with a free account at Watson Studio

Thanks for completing this lesson!

Author: Saeed Aghabozorgi

Saeed Aghabozorgi, PhD is a Data Scientist in IBM with a track record of developing enterprise level applications that substantially increases clients' ability to turn data into actionable knowledge. He is a researcher in data mining field and expert in developing advanced analytic methods like machine learning and statistical modelling on large datasets.