Exercice lab: Decision Trees

https://www.coursera.org/learn/machine-learning-with-python/ungradedLti/SzC4i/lab-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 respond 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.

Please notice that the practice labs (except the last week assignment) are optional and are provided for you to practice and understand the topic. Therefore, you do not need to submit those, as they are not graded, and won't be updated as complete. Just run the codes to see the results, and feel free to change it.

Ce cours utilise l'outil d'un tiers, Lab: Decision Trees, pour améliorer votre expérience d'apprentissage. L'outil référence des informations de base comme votre nom, votre adresse e-mail et votre ID Coursera.

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

Import the Following Libraries:

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

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

Code
```
import numpy as np
import pandas as pd
from sklearn.tree import DecisionTreeClassifier
```

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 feature sets of this dataset are Age, Sex, Blood Pressure, and Cholesterol of patients, and the target is the drug that each patient responded to.

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 prescribe it to a new patient.

Downloading the Data

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

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

Code
```
!wget -O drug200.csv https://s3-api.us-geo.objectstorage.softlayer.net/cf-courses-data/CognitiveClass/ML0101ENv3/labs/drug200.csv
```

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:

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

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

Practice

What is the size of data?

```
[ ]: # write your code here
```

CODE

```
print(my_data.shape)
```

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.

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

Code

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

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.

```
[ ]: 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]
```

Code

```
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]
```

Now we can fill the target variable.

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

Code
y = my_data["Drug"]
y[0:5]

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**.

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

Code
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.

```
[ ]: X_trainset, X_testset, y_trainset, y_testset = train_test_split(X, y, test_size=0.3, random_sta
```

Code
X_trainset, X_testset, y_trainset, y_testset = train_test_split(X, y, test_size=0.3, random_state=3)

Practice

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

```
[ ]: # your code
```

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

```
[ ]: # your code
```

CODE 1
print(X_trainset, y_trainset.shape)

CODE 2
print(X_testset, y_testset.shape)

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.

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

Code
drugTree = DecisionTreeClassifier(criterion="entropy", max_depth = 4)
drugTree # it shows the default parameters

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

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

Code
drugTree.fit(X_trainset,y_trainset)

Prediction

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

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

Code
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.

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

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

Evaluation

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

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

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

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.

Practice

Can you calculate the accuracy score without sklearn ?

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

Visualization ¶

Lets visualize the tree

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

Code
Notice: You might need to uncomment and install the pydotplus and graphviz libraries if you have not installed these before
!conda install -c conda-forge pydotplus -y
!conda install -c conda-forge python-graphviz -y

```
Solving environment: done


==> WARNING: A newer version of conda exists. <==
  current version: 4.5.11
  latest version: 4.8.3

Please update conda by running

    $ conda update -n base -c defaults conda



## Package Plan ##

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

  added / updated specs:
    - pydotplus


The following packages will be downloaded:

    package                    |            build
    ---------------------------|-----------------
    python_abi-3.6             |          1_cp36m           4 KB  conda-forge
    openssl-1.1.1e             |       h516909a_0         2.1 MB  conda-forge
    pydotplus-2.0.2            |     pyhd1c1de3_3          23 KB  conda-forge
    certifi-2019.11.28         |   py36h9f0ad1d_1         149 KB  conda-forge
    ------------------------------------------------------------
                                           Total:         2.3 MB

The following NEW packages will be INSTALLED:

    pydotplus:  2.0.2-pyhd1c1de3_3 conda-forge
    python_abi: 3.6-1_cp36m        conda-forge

The following packages will be UPDATED:

    certifi:    2019.11.28-py36_0  conda-forge --> 2019.11.28-py36h9f0ad1d_1 conda-forge
    openssl:    1.1.1d-h516909a_0  conda-forge --> 1.1.1e-h516909a_0         conda-forge
```

```
Downloading and Extracting Packages
python_abi-3.6       | 4 KB      | ##################################### | 100%
openssl-1.1.1e       | 2.1 MB    | ##################################### | 100%
pydotplus-2.0.2      | 23 KB     | ##################################### | 100%
certifi-2019.11.28   | 149 KB    | ##################################### | 100%
Preparing transaction: done
Verifying transaction: done
Executing transaction: done
Solving environment: done


==> WARNING: A newer version of conda exists. <==
  current version: 4.5.11
  latest version: 4.8.3

Please update conda by running

    $ conda update -n base -c defaults conda

## Package Plan ##

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

  added / updated specs:
    - python-graphviz


The following packages will be downloaded:

    package                    |            build
    ---------------------------|-----------------
    python-graphviz-0.13.2     |             py_0          18 KB  conda-forge

The following NEW packages will be INSTALLED:

    python-graphviz: 0.13.2-py_0 conda-forge

Downloading and Extracting Packages
python-graphviz-0.13 | 18 KB     | ##################################### | 100%
Preparing transaction: done
Verifying transaction: done
Executing transaction: done
```

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

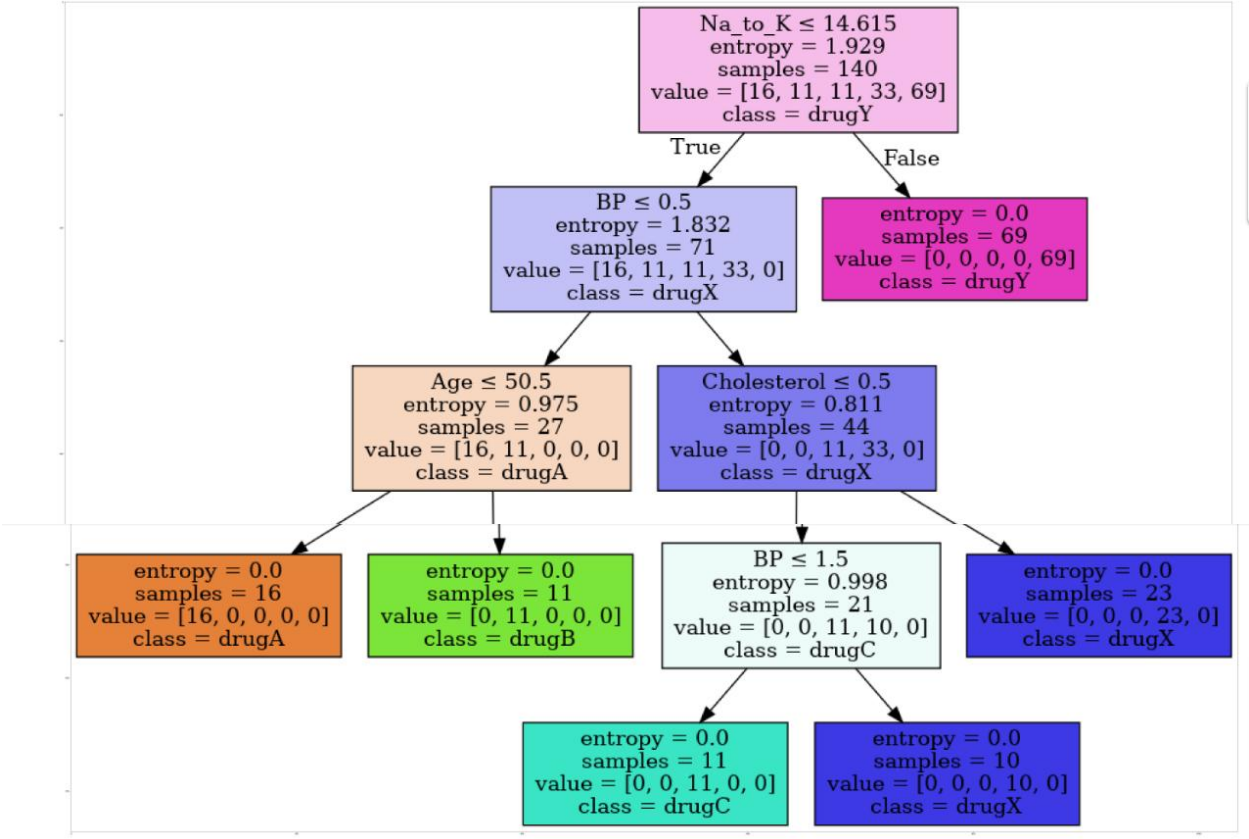
Code

```
from sklearn.externals.six import StringIO
import pydotplus
import matplotlib.image as mpimg
from sklearn import tree
%matplotlib inline
```

```
[ ]: 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= n
     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')
```

Code

```
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')
```



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