**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 a unknown patient, or to find a proper drug for a new patient.

**Table of contents**

1. [About the dataset](https://jupyterlab-4-labs-prod-jupyterlab-us-east-0.labs.cognitiveclass.ai/user/akulasamson/files/labs/coursera/ML0101EN/https%3A/%23about_dataset?_xsrf=2%7C34313e40%7C6238ca29ce5b2e428c5db70dff082614%7C1630087457)
2. [Downloading the Data](https://jupyterlab-4-labs-prod-jupyterlab-us-east-0.labs.cognitiveclass.ai/user/akulasamson/files/labs/coursera/ML0101EN/https%3A/%23downloading_data?_xsrf=2%7C34313e40%7C6238ca29ce5b2e428c5db70dff082614%7C1630087457)
3. [Pre-processing](https://jupyterlab-4-labs-prod-jupyterlab-us-east-0.labs.cognitiveclass.ai/user/akulasamson/files/labs/coursera/ML0101EN/https%3A/%23pre-processing?_xsrf=2%7C34313e40%7C6238ca29ce5b2e428c5db70dff082614%7C1630087457)
4. [Setting up the Decision Tree](https://jupyterlab-4-labs-prod-jupyterlab-us-east-0.labs.cognitiveclass.ai/user/akulasamson/files/labs/coursera/ML0101EN/https%3A/%23setting_up_tree?_xsrf=2%7C34313e40%7C6238ca29ce5b2e428c5db70dff082614%7C1630087457)
5. [Modeling](https://jupyterlab-4-labs-prod-jupyterlab-us-east-0.labs.cognitiveclass.ai/user/akulasamson/files/labs/coursera/ML0101EN/https%3A/%23modeling?_xsrf=2%7C34313e40%7C6238ca29ce5b2e428c5db70dff082614%7C1630087457)
6. [Prediction](https://jupyterlab-4-labs-prod-jupyterlab-us-east-0.labs.cognitiveclass.ai/user/akulasamson/files/labs/coursera/ML0101EN/https%3A/%23prediction?_xsrf=2%7C34313e40%7C6238ca29ce5b2e428c5db70dff082614%7C1630087457)
7. [Evaluation](https://jupyterlab-4-labs-prod-jupyterlab-us-east-0.labs.cognitiveclass.ai/user/akulasamson/files/labs/coursera/ML0101EN/https%3A/%23evaluation?_xsrf=2%7C34313e40%7C6238ca29ce5b2e428c5db70dff082614%7C1630087457)
8. [Visualization](https://jupyterlab-4-labs-prod-jupyterlab-us-east-0.labs.cognitiveclass.ai/user/akulasamson/files/labs/coursera/ML0101EN/https%3A/%23visualization?_xsrf=2%7C34313e40%7C6238ca29ce5b2e428c5db70dff082614%7C1630087457)

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

**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 a 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.

[ ]:



**!**wget **-**O drug200.csv https:**//**cf**-**courses**-**data.s3.us.cloud**-**object**-**storage.appdomain.cloud**/**IBMDeveloperSkillsNetwork**-**ML0101EN**-**SkillsNetwork**/**labs**/**Module**%**203**/**data**/**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](http://cocl.us/ML0101EN-IBM-Offer-CC)

Now, read the data using pandas dataframe:

[ ]:



my\_data **=** pd.read\_csv("drug200.csv", delimiter**=**",")

my\_data[0:5]

**Practice**

What is the size of data?

[ ]:



*# write your code here*

​

​

​

Click here for the solution

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]

​

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 **pandas.get\_dummies()** to convert the 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]

​

Now we can fill the target variable.

[ ]:



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

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\_state**=**3)

**Practice**

Print the shape of X\_trainset and y\_trainset. Ensure that the dimensions match.

**Did you know? IBM Watson Studio lets you build and deploy an AI solution, using the best of open source and IBM software and giving your team a single environment to work in.**[**Learn more here.**](https://cocl.us/ibm_watson_studio_infobox)

[ ]:



*# your code*

​

​

Click here for the solution

print('Shape of X training set {}'.format(X\_trainset.shape),'&',' Size of Y training set {}'.format(y\_trainset.shape))

Print the shape of X\_testset and y\_testset. Ensure that the dimensions match.

[ ]:



*# your code*

​

​

Click here for the solution

print('Shape of X training set {}'.format(X\_testset.shape),'&',' Size of Y training set {}'.format(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*

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)

**Prediction**

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

[ ]:



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.

[ ]:



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))

**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.

**Visualization**

Let's visualize the tree

[ ]:



*# 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*

[ ]:



**from** io **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]

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?**