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

December 5, 2018

#

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.

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

0.0.1 About 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.

0.0.2 Downloading Data

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

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

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

0.1 Practice

What is the size of data?

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

0.2 Pre-processing

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 [ ]: 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.

```
In [ ]: 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.

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

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

0.4 Practice

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

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

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

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

0.5 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 [ ]: 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

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

0.6 Prediction

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

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

0.7 Evaluation

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

```
In [ ]: 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.

0.8 Practice

Can you calculate the accuracy score without sklearn ?

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

0.9 Visualization

Lets visualize the tree

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

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

```
In [ ]: 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_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')
```

0.10 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

0.10.1 Thanks for completing this lesson!

Notebook created by: Saeed Aghabozorgi