Drug-Prediction-Decision-Tree

February 13, 2019

1 Using Decision Tree to Recommend Medications-Classification

1.1 About this notebook

The aim of the project is recommending the right medications to different patient who suffer from the same illness. There are five medications in total, Drug A, Drug B, Drug c, Drug x and y. I will use decision tree, identify useful classifiers to build the model.

Decision Tree is very useful in classification problems. In this case, we will use entropy as the criteria of selecting classifiers. To be more specific, selecting independent variables that will classify observations with higher homogeneity. For example, if all female patients have been prescibed drug A while all male patients got drug B, the variable 'Sex' is a good classifier. Smaller entropy value indicates a better classfier.

It is a part of IBM Data Scientist Certification program. I made necessary modifications and more explorations.

1.2 Variable Description

- **Age**: A patient's age. Numeric.
- **Sex**: A patient's gender. F (female) or M (male). Categorical.
- **BP**: Blood pressure level. Low, Normal, or High. Categorical.
- **Cholesterol**: Cholesterol level. High or Normal. Categorical.
- **Na_to_K**: The ratio of Blood sodium concerntration to potassium concentration. Numeric.
- **Drug**: The name of drug that has been prescribed to that patient. Drug A, Drug B, Drug c, Drug x and Drug y. Categorical.

1.3 Contents

1.4 Preparation

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

In [2]: #read data
        drug=pd.read_csv(r'E:\Data Science\Course 8 Machine Learning\drug200.csv')
        drug.head()
```

Out[2]:

	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

1.5 Understand Data

```
In [9]: #shape of the data
        drug.shape

Out[9]: (200, 6)

In [11]: #types of columns
         drug.dtypes

Out[11]: Age              int64
         Sex             object
         BP              object
         Cholesterol     object
         Na_to_K        float64
         Drug            object
         dtype: object

In [16]: #unique values of drugs
         drug['Drug'].value_counts()

Out[16]: drugY    91
         drugX    54
         drugA    23
         drugB    16
         drugC    16
         Name: Drug, dtype: int64

In [18]: #unique values of BP
         drug['BP'].value_counts()
```

```
Out[18]: HIGH      77
         LOW       64
         NORMAL    59
         Name: BP, dtype: int64

In [19]: #unique values of Cholesterol
         drug['Cholesterol'].value_counts()

Out[19]: HIGH      103
         NORMAL     97
         Name: Cholesterol, dtype: int64

In [17]: #get summary statistics
         drug.describe(include='all')
```

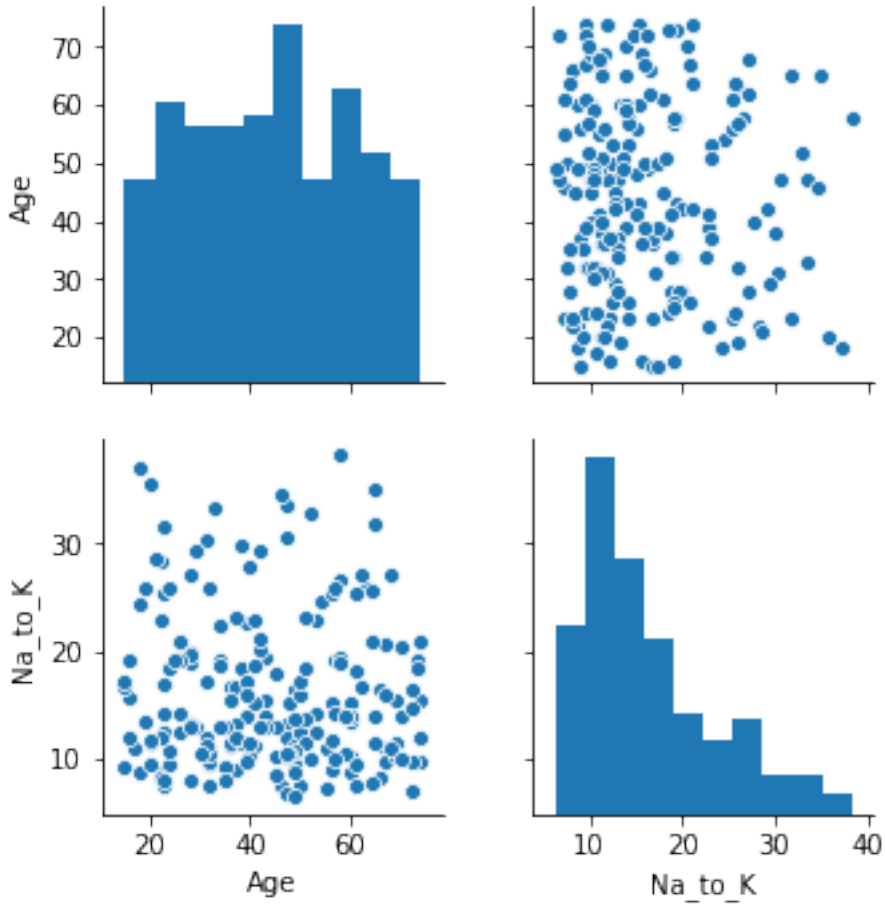
Out[17]:

	Age	Sex	BP	Cholesterol	Na_to_K	Drug
count	200.000000	200	200	200	200.000000	200
unique	NaN	2	3	2	NaN	5
top	NaN	M	HIGH	HIGH	NaN	drugY
freq	NaN	104	77	103	NaN	91
mean	44.315000	NaN	NaN	NaN	16.084485	NaN
std	16.544315	NaN	NaN	NaN	7.223956	NaN
min	15.000000	NaN	NaN	NaN	6.269000	NaN
25%	31.000000	NaN	NaN	NaN	10.445500	NaN
50%	45.000000	NaN	NaN	NaN	13.936500	NaN
75%	58.000000	NaN	NaN	NaN	19.380000	NaN
max	74.000000	NaN	NaN	NaN	38.247000	NaN

```
In [16]: import seaborn as sns

In [17]: sns.pairplot(drug)

Out[17]: <seaborn.axisgrid.PairGrid at 0x209cdd661d0>
```



1.6 Pre-processing

```
In [3]: #extract independent variables
        X = drug[['Age', 'Sex', 'BP', 'Cholesterol', 'Na_to_K']].values

In [4]: #convert Pandas Dataframe to Numpy array and recode categorical variables into dummies
        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[4]: array([[23, 0, 0, 0, 25.355],
               [47, 1, 1, 0, 13.093],
               [47, 1, 1, 0, 10.114],
               [28, 0, 2, 0, 7.797999999999999],
               [61, 0, 1, 0, 18.043]], dtype=object)
```

```
In [5]: #define the dependent variable
        y = drug["Drug"]
```

1.7 Setting up the decision tree

```
In [6]: #import packages
        from sklearn.model_selection import train_test_split
```

```
In [7]: #we set the percentage of test set 30% and make sure they have same dimension
        X_trainset, X_testset, y_trainset, y_testset = train_test_split(X, y, test_size=0.3, ra
        print('shape of training data:', X_trainset.shape)
        print('shape of y training data', y_trainset.shape)
        print('shape of training data:', X_testset.shape)
        print('shape of y training data', y_testset.shape)
```

```
shape of training data: (140, 5)
shape of y training data (140,)
shape of training data: (60, 5)
shape of y training data (60,)
```

1.8 Modeling

```
In [8]: #we use entropy to evaluate classfiers
        drugTree = DecisionTreeClassifier(criterion="entropy", max_depth = 4)
        drugTree # it shows the default parameters
```

```
Out[8]: 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')
```

```
In [9]: #train the model with train data
        drugTree.fit(X_trainset,y_trainset)
```

```
Out[9]: 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')
```

1.9 Prediction

```
In [10]: #make predictions on testing dataset and store the predictions into a variable called
         predTree = drugTree.predict(X_testset)

In [11]: #print the prediction results
         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
```

1.10 Evaluation

```
In [12]: from sklearn import metrics
         import matplotlib.pyplot as plt
         print("DecisionTrees's Accuracy: ", metrics.accuracy_score(y_testset, predTree))
```

```
DecisionTrees's Accuracy:  0.9833333333333333
```

The accuracy is 0.983 which is very good (close to 1).

The **Accuracy classification score** computes subset 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. (from sklearn documentation)

1.11 Visualization

```
In [21]: !pip install pydotplus
```

```
Requirement already satisfied: pydotplus in d:\anaconda3\lib\site-packages (2.0.2)
Requirement already satisfied: pyparsing>=2.0.1 in d:\anaconda3\lib\site-packages (from pydotpl
```

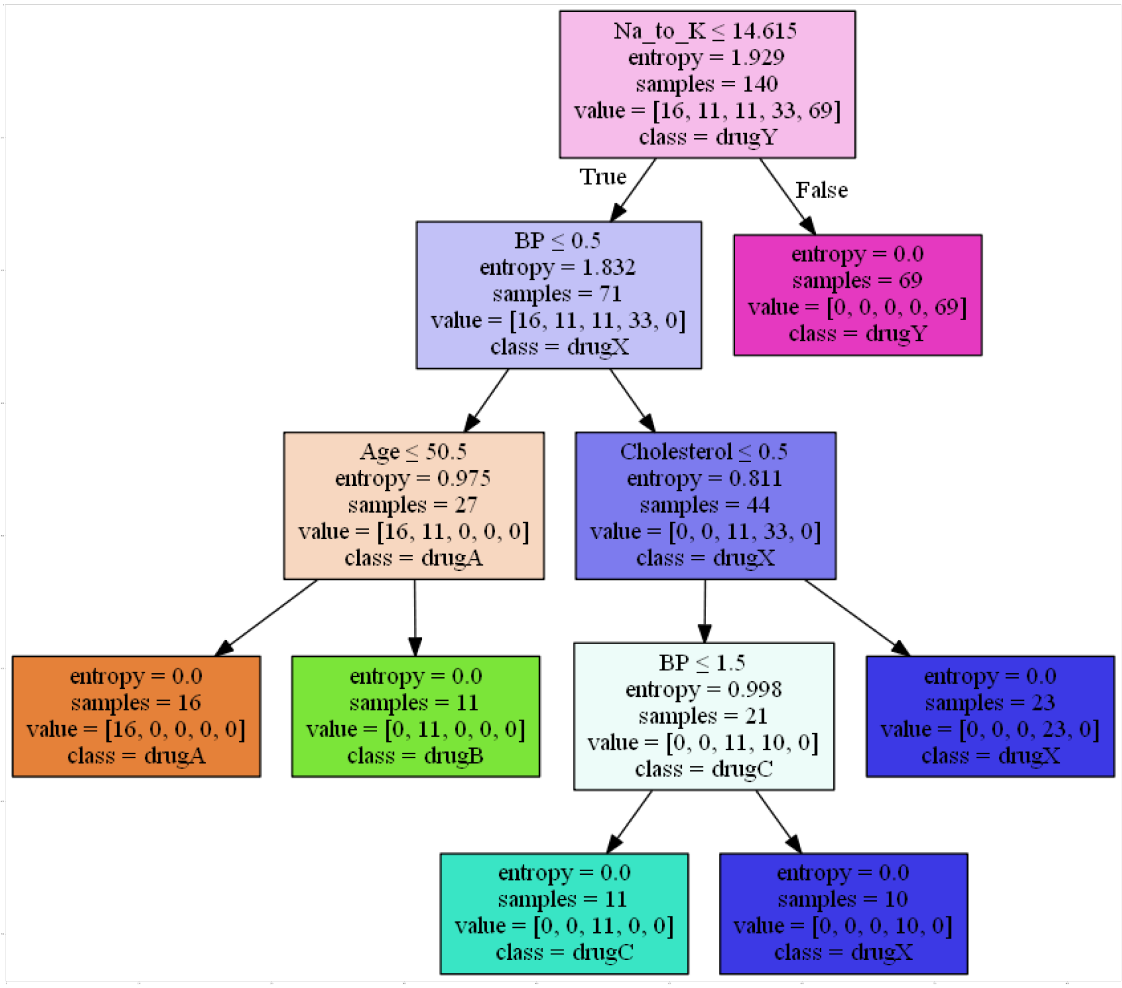
```
In [18]: !pip install graphviz
```

```
Collecting graphviz
  Downloading https://files.pythonhosted.org/packages/1f/e2/ef2581b5b86625657afd32030f90cf2717
Installing collected packages: graphviz
Successfully installed graphviz-0.10.1
```

```
In [14]: #import packages
         from sklearn.externals.six import StringIO
         import pydotplus
         import matplotlib.image as mpimg
         from IPython.display import Image
         from sklearn import tree
         from sklearn.tree import export_graphviz
         import graphviz
         %matplotlib inline
```

```
In [15]: dot_data = StringIO()
         filename = "drugtree.png"
         featureNames = drug.columns[0:5]
         targetNames = drug["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())
         #Image(graph.create_png())
         graph.write_png(filename)
         img = mpimg.imread(filename)
         plt.figure(figsize=(100, 200))
         plt.imshow(img,interpolation='nearest')
```

```
Out[15]: <matplotlib.image.AxesImage at 0x209cdd89550>
```


Na_to_K ≤ 14.615
entropy = 1.929
samples = 140
value = [16, 11, 11, 33, 69]
class = drugY
True
False
BP ≤ 0.5
entropy = 1.832
samples = 71
value = [16, 11, 11, 33, 0]
class = drugX
entropy = 0.0
samples = 69
value = [0, 0, 0, 0, 69]
class = drugY
Age ≤ 50.5
entropy = 0.975
samples = 27
value = [16, 11, 0, 0, 0]
class = drugA
Cholesterol ≤ 0.5
entropy = 0.811
samples = 44
value = [0, 0, 11, 33, 0]
class = drugX
entropy = 0.0
samples = 16
value = [16, 0, 0, 0, 0]
class = drugA
entropy = 0.0
samples = 11
value = [0, 11, 0, 0, 0]
class = drugB
BP ≤ 1.5
entropy = 0.998
samples = 21
value = [0, 0, 11, 10, 0]
class = drugC
entropy = 0.0
samples = 23
value = [0, 0, 0, 23, 0]
class = drugX
entropy = 0.0
samples = 11
value = [0, 0, 11, 0, 0]
class = drugC
entropy = 0.0
samples = 10
value = [0, 0, 0, 10, 0]
class = drugX