Decision Tree

Introduction

- Author: mdtrung@hcmut.edu.vn, thuanle@hcmut.edu.vn
- Content
  - Load data from CSV, split into trainning set and test set
  - Build a Decision Tree model to predict the value.
  - Evaluating the results

Setting up Lab Exercise

If you haven't already done so, follow the instructions above to start your Jupyter Notebook server. Create a new Python 3 notebook and name it as you see fit e.g., DTreeTutorial.

Enter the following code into your first cell, paying close attention to the comments to understand what each line is doing. This will set up compatability and needed libraries. When done entering it your notebook, hit Shift+Enter to execute.

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
```

Load and prep the data

Since our file is in CSV format, we will use panda's read_csv method to read our CSV data file. Execute the following script to do so:

In [2]:
```
# Importing dataset
dataset = pd.read_csv("bill_authentication.csv")
```

In [3]:
```
# Data analysis: Execute the following command to see the number of rows and column
s in our dataset:

dataset.shape
```

Out[3]:
```
(1372, 5)
```

In [ ]:
```
# The output will show "(1372,5)", which means that our dataset has 1372 records an
d 5 attributes.
```

In [4]:
```
dataset.head()
```

Out[4]:

	Variance	Skewness	Curtosis	Entropy	Class
0	3.62160	8.6661	-2.8073	-0.44699	0
1	4.54590	8.1674	-2.4586	-1.46210	0
2	3.86600	-2.6383	1.9242	0.10645	0
3	3.45660	9.5228	-4.0112	-3.59440	0
4	0.32924	-4.4552	4.5718	-0.98880	0

Preparing the Data

In this section we will divide our data into attributes and labels and will then divide the resultant data into both training and test sets. By doing this we can train our algorithm on one set of data and then test it out on a completely different set of data that the algorithm hasn't seen yet. This provides you with a more accurate view of how your trained algorithm will actually perform.

To divide data into attributes and labels, execute the following code:

In [5]:
```
X = dataset.drop('Class', axis=1)
y = dataset['Class']
```

In [ ]:
```
#Here the X variable contains all the columns from the dataset, except the "Class"
column, which is the label.
#The y variable contains the values from the "Class" column.
#The X variable is our attribute set and y variable contains corresponding labels.
```

Dividing our data into training and test sets.

In [6]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.20)
```

In [ ]:
```
#In the code above, the test_size parameter specifies the ratio of the test set, wh
ich we use to split up 20% of the data in to the test set and 80% for training.
```

Training and Making Predictions

In [7]:
```
from sklearn.tree import DecisionTreeClassifier
classifier = DecisionTreeClassifier()
classifier.fit(X_train, y_train)
```

Out[7]:
```
DecisionTreeClassifier(class_weight=None, criterion='gini', max_depth=None,
            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 [8]:
```
# To make predictions, the predict method of the DecisionTreeClassifier class is us
ed.
# Take a look at the following code for usage:
y_pred = classifier.predict(X_test)
```

Evaluating the Algorithm

For classification tasks some commonly used metrics are confusion matrix, precision, recall, and F1 score.

In [9]:
```
from sklearn.metrics import classification_report, confusion_matrix
print(confusion_matrix(y_test, y_pred))
print(classification_report(y_test, y_pred))
```

```
[[146   3]
 [  0 126]]
             precision    recall  f1-score   support

          0       1.00      0.98      0.99       149
          1       0.98      1.00      0.99       126

avg / total       0.99      0.99      0.99       275
```

Extra references

- Read more about:
  - cross-validation (https://en.wikipedia.org/wiki/Cross-validation_(statistics%)

Exercises

In [ ]:
```
*Give the Titanic data file with records found on Kaggle. Your task is to build a c
lassifier which can predict whether they survived or not?

    - Link: https://www.dropbox.com/request/N62XApeQuC7D3WWOPmDj

    - Deadline: 10:00 am next Tuesday
```