Decision Tree Classifier Building in Scikit-learn

(https://www.datacamp.com/community/tutorials/decision-tree-classification-python)

Importing Required Libraries

Let's first load the required libraries.

```
# Load libraries

import pandas as pd

from sklearn.tree import DecisionTreeClassifier # Import Decision Tree
Classifier

from sklearn.model_selection import train_test_split # Import
train_test_split function

from sklearn import metrics #Import scikit-learn metrics module for accuracy
calculation
```

Loading Data

Let's first load the required Pima Indian Diabetes dataset using pandas' read CSV function. You can download the data from https://www.kaggle.com/uciml/pima-indians-diabetes-database

```
col_names = ['pregnant', 'glucose', 'bp', 'skin', 'insulin', 'bmi',
'pedigree', 'age', 'label']

# load dataset

pima = pd.read_csv("pima-indians-diabetes.csv", header=None, names=col_names)

pima.head()
```

	pregnant	glucose	bp	skin	insulin	bmi	pedigree	age	label
0	6	148	72	35	0	33.6	0.627	50	1
1	1	85	66	29	0	26.6	0.351	31	0
2	8	183	64	0	0	23.3	0.672	32	1
3	1	89	66	23	94	28.1	0.167	21	0
4	0	137	40	35	168	43.1	2.288	33	1

Feature Selection

Here, you need to divide given columns into two types of variables dependent(or target variable) and independent variable(or feature variables).

```
#split dataset in features and target variable

feature_cols = ['pregnant', 'insulin', 'bmi',

'age','glucose','bp','pedigree']

X = pima[feature_cols] # Features

y = pima.label # Target variable
```

Splitting Data

To understand model performance, dividing the dataset into a training set and a test set is a good strategy.

Let's split the dataset by using function train_test_split(). You need to pass 3 parameters features, target, and test_set size.

```
# Split dataset into training set and test set
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,
random_state=1) # 70% training and 30% test
```

Building Decision Tree Model

Let's create a Decision Tree Model using Scikit-learn.

```
# Create Decision Tree classifer object
clf = DecisionTreeClassifier()

# Train Decision Tree Classifer
clf = clf.fit(X_train,y_train)

#Predict the response for test dataset
y_pred = clf.predict(X_test)
```

Evaluating Model

Let's estimate, how accurately the classifier or model can predict the type of cultivars.

Accuracy can be computed by comparing actual test set values and predicted values.

```
# Model Accuracy, how often is the classifier correct?
print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
Accuracy: 0.6753246753246753
```

Well, you got a classification rate of 67.53%, considered as good accuracy. You can improve this accuracy by tuning the parameters in the Decision Tree Algorithm.

Visualizing Decision Trees

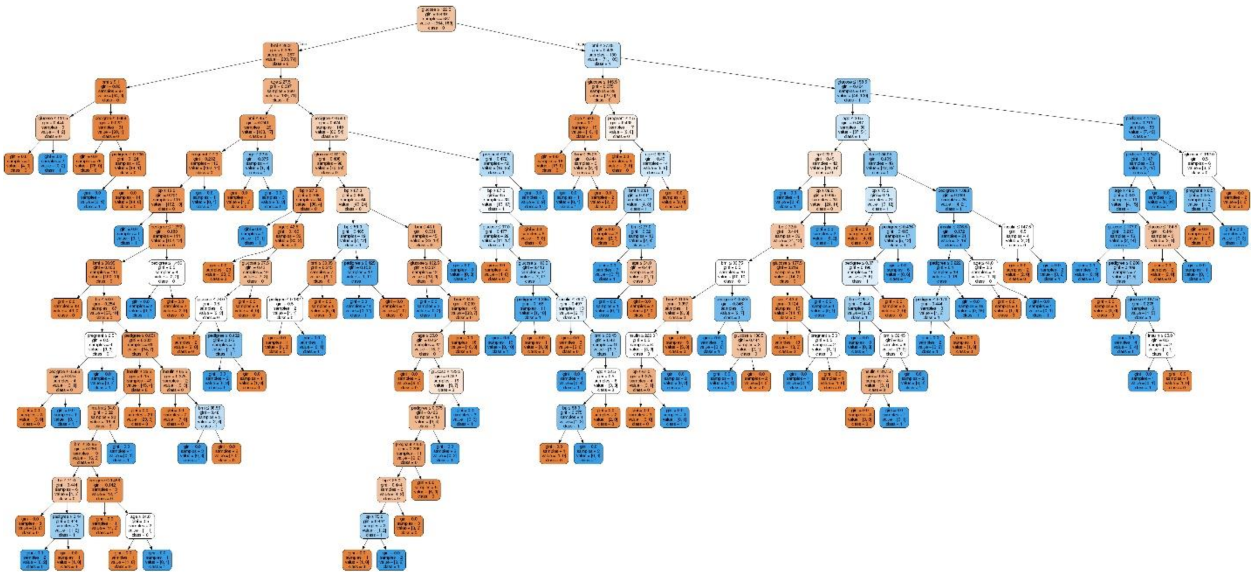
You can use Scikit-learn's *export_graphviz* function for display the tree within a Jupyter notebook. For plotting tree, you also need to install graphviz and pydotplus.

`pip install graphviz`

`pip install pydotplus`

export_graphviz function converts decision tree classifier into dot file and pydotplus convert this dot file to png or displayable form on Jupyter.

```
from sklearn.tree import export_graphviz

from sklearn.externals.six import StringIO

from IPython.display import Image

import pydotplus


dot_data = StringIO()

export_graphviz(clf, out_file=dot_data,

                filled=True, rounded=True,

                special_characters=True,feature_names =
feature_cols,class_names=['0','1'])

graph = pydotplus.graph_from_dot_data(dot_data.getvalue())

graph.write_png('diabetes.png')

Image(graph.create_png())
```



In the decision tree chart, each internal node has a decision rule that splits the data. Gini referred as Gini ratio, which measures the impurity of the node. You can say a node is pure when all of its records belong to the same class, such nodes known as the leaf node.

Here, the resultant tree is unpruned. This unpruned tree is unexplainable and not easy to understand. In the next section, let's optimize it by pruning.

Optimizing Decision Tree Performance

- **criterion : optional (default=”gini”) or Choose attribute selection measure**: This parameter allows us to use the different-different attribute selection measure. Supported criteria are “gini” for the Gini index and “entropy” for the information gain.
- **splitter : string, optional (default=”best”) or Split Strategy**: This parameter allows us to choose the split strategy. Supported strategies are “best” to choose the best split and “random” to choose the best random split.
- **max_depth : int or None, optional (default=None) or Maximum Depth of a Tree**: The maximum depth of the tree. If None, then nodes

are expanded until all the leaves contain less than min_samples_split samples. The higher value of maximum depth causes overfitting, and a lower value causes underfitting (Source).

In Scikit-learn, optimization of decision tree classifier performed by only pre-pruning. Maximum depth of the tree can be used as a control variable for pre-pruning. In the following the example, you can plot a decision tree on the same data with max_depth=3. Other than pre-pruning parameters, You can also try other attribute selection measure such as entropy.

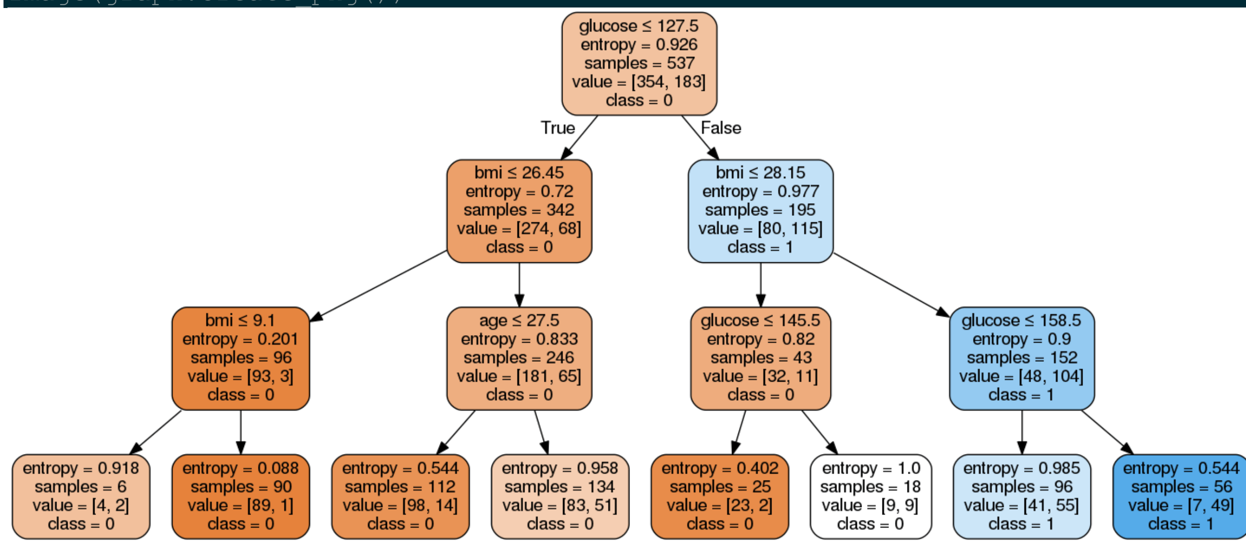
```
# Create Decision Tree classifer object

clf = DecisionTreeClassifier(criterion="entropy", max_depth=3)


# Train Decision Tree Classifer

clf = clf.fit(X_train,y_train)


#Predict the response for test dataset

y_pred = clf.predict(X_test)


# Model Accuracy, how often is the classifier correct?

print("Accuracy:",metrics.accuracy_score(y_test, y_pred))

Accuracy: 0.7705627705627706
```

Well, the classification rate increased to 77.05%, which is better accuracy than the previous model.

Visualizing Decision Trees

```
from sklearn.externals.six import StringIO

from IPython.display import Image
```

```
from sklearn.tree import export_graphviz
import pydotplus
dot_data = StringIO()
export_graphviz(clf, out_file=dot_data,
                filled=True, rounded=True,
                special_characters=True, feature_names =
feature_cols,class_names=['0','1'])
graph = pydotplus.graph_from_dot_data(dot_data.getvalue())
graph.write_png('diabetes.png')
Image(graph.create_png())
```



This pruned model is less complex, explainable, and easy to understand than the previous decision tree model plot.