# Creating and Visualizing Decision Trees with Python

Russell Follow
Aug 5, 2017 · 2 min read

Decision trees are the building blocks of some of the most powerful supervised learning methods that are used today. A decision tree is basically a binary tree flowchart where each node splits a group of observations according to some feature variable. The goal of a decision tree is to split your data into groups such that every element in one group belongs to the same category. Decision trees can also be used to approximate a continuous target variable. In that case, the tree will make splits such that each group has the lowest mean squared error.

One of the great properties of decision trees is that they are very easily interpreted. You do not need to be familiar at all with machine learning techniques to understand what a decision tree is doing. Decision tree graphs are very easily interpreted, plus they look cool! I will show you how to generate a decision tree and create a graph of it in a Jupyter Notebook (formerly known as IPython Notebooks). In this example, I will be using the classic iris dataset. Use the following code to load it.

```
import sklearn.datasets as datasets
import pandas as pd
iris=datasets.load_iris()
df=pd.DataFrame(iris.data, columns=iris.feature_names)
y=iris.target
```

Sklearn will generate a decision tree for your dataset using an optimized version of the CART algorithm when you run the following code.

```
from sklearn.tree import DecisionTreeClassifier
dtree=DecisionTreeClassifier()
dtree.fit(df,y)
```

You can also import DecisionTreeRegressor from sklearn.tree if you want to use a decision tree to predict a numerical target variable. Try switching one of the columns of df with our y variable from above and fitting a regression tree on it.

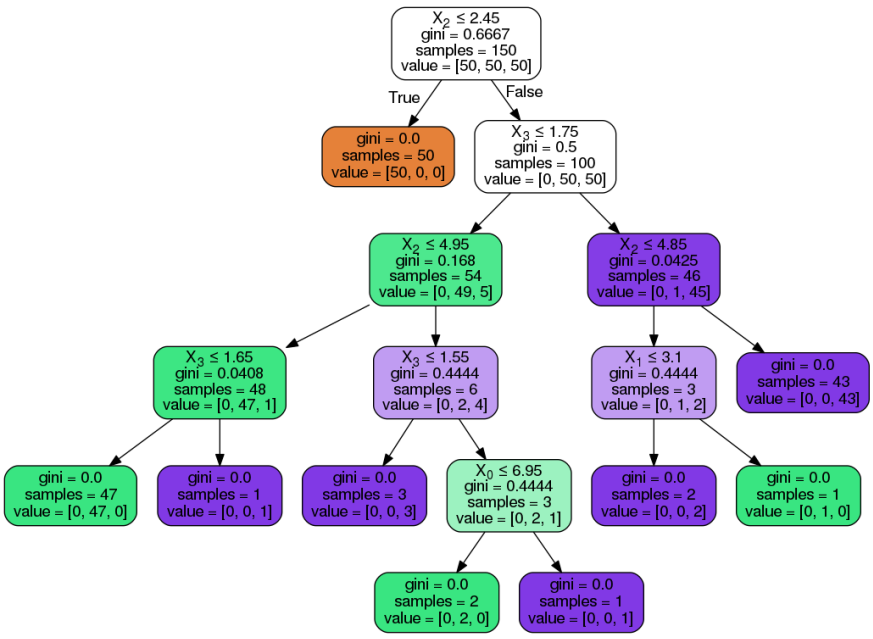
Now that we have a decision tree, we can use the pydotplus package to create a visualization for it.

```
from sklearn.externals.six import StringIO
from IPython.display import Image
from sklearn.tree import export_graphviz
import pydotplus
```

```
dot_data = StringIO()
```

```
export_graphviz(dtree, out_file=dot_data,
                filled=True, rounded=True,
                special_characters=True)
```

```
graph = pydotplus.graph_from_dot_data(dot_data.getvalue())
Image(graph.create_png())
```




The ‘value’ row in each node tells us how many of the observations that were sorted into that node fall into each of our three categories. We can

see that our feature X2, which is the petal length, was able to completely distinguish one species of flower (Iris-Setosa) from the rest.

The biggest drawback to decision trees is that the split it makes at each node will be optimized for the dataset it is fit to. This splitting process will rarely generalize well to other data. However, we can generate huge numbers of these decision trees, tuned in slightly different ways, and combine their predictions to create some of our best models today.