

You have **2** free member-only stories left this month. [Sign up for Medium and get an extra one](#)

Visualizing Decision Trees with Python (Scikit-learn, Graphviz, Matplotlib)

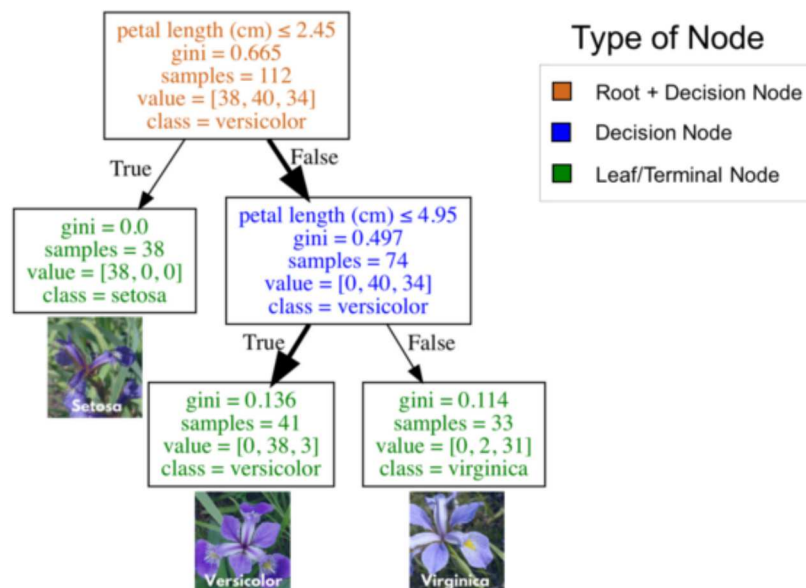
Learn about how to visualize decision trees using matplotlib and Graphviz



Michael Galarnyk · Apr 2, 2020 · 9 min read ★

What class (species) is a flower with the following feature?

petal length (cm): 4.5



Species counts are: setosa=0, versicolor=38, virginica=3
Prediction is **versicolor** as it is the majority class

Decision trees are a popular supervised learning method for a variety of reasons. Benefits of decision trees include that they can be used for both regression and classification, they don't require feature scaling, and they are relatively easy to interpret as you can visualize decision trees. This is not only a powerful way to understand your model, but also to communicate how your model works. Consequently, it would help to know how to make a visualization based on your model.

This tutorial covers:

- How to Fit a Decision Tree Model using Scikit-Learn
- How to Visualize Decision Trees using Matplotlib
- How to Visualize Decision Trees using Graphviz (what is Graphviz, how to install it on Mac and Windows, and how to use it to visualize decision trees)
- How to Visualize Individual Decision Trees from Bagged Trees or Random Forests

As always, the code used in this tutorial is available on my [GitHub](#). With that, let's get started!

How to Fit a Decision Tree Model using Scikit-Learn

In order to visualize decision trees, we need first need to fit a decision tree model using scikit-learn. If this section is not clear, I encourage you to read my [Understanding Decision Trees for Classification \(Python\) tutorial](#) as I go into a lot of detail on how decision trees work and how to use them.

Import Libraries

The following import statements are what we will use for this section of the

tutorial.

```
import matplotlib.pyplot as plt
from sklearn.datasets import load_iris
from sklearn.datasets import load_breast_cancer
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
import pandas as pd
import numpy as np
from sklearn import tree
```

Load the Dataset

The Iris dataset is one of datasets scikit-learn comes with that do not require the downloading of any file from some external website. The code below loads the iris dataset.

```
import pandas as pd
from sklearn.datasets import load_iris
data = load_iris()
df = pd.DataFrame(data.data, columns=data.feature_names)
df['target'] = data.target
```

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	target
0	5.1	3.5	1.4	0.2	0
1	4.9	3.0	1.4	0.2	0
2	4.7	3.2	1.3	0.2	0
3	4.6	3.1	1.5	0.2	0
4	5.0	3.6	1.4	0.2	0

Original Pandas df (features + target)

Splitting Data into Training and Test Sets

The code below puts 75% of the data into a training set and 25% of the data into a test set.

```
X_train, X_test, Y_train, Y_test =
train_test_split(df[data.feature_names], df['target'],
random_state=0)
```

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	target	
0	5.1	3.5	1.4	0.2	0	X_train
1	4.9	3	1.4	0.2	0	X_test
2	4.7	3.2	1.3	0.2	0	Y_train
3	4.6	3.1	1.5	0.2	0	Y_test
4	5	3.6	1.4	0.2	0	
5	5.4	3.9	1.7	0.4	0	
6	4.6	3.4	1.4	0.3	0	
7	5	3.4	1.5	0.2	0	
8	4.4	2.9	1.4	0.2	0	
9	4.9	3.1	1.5	0.1	0	

The colors in the image indicate which variable (X_train, X_test, Y_train, Y_test) the data from the dataframe df went to for a particular train test split. Image by [Michael Galarnyk](#).

Scikit-learn 4-Step Modeling Pattern

```
# Step 1: Import the model you want to use
# This was already imported earlier in the notebook so commenting out
# from sklearn.tree import DecisionTreeClassifier

# Step 2: Make an instance of the Model
clf = DecisionTreeClassifier(max_depth = 2,
                             random_state = 0)

# Step 3: Train the model on the data
clf.fit(X_train, Y_train)

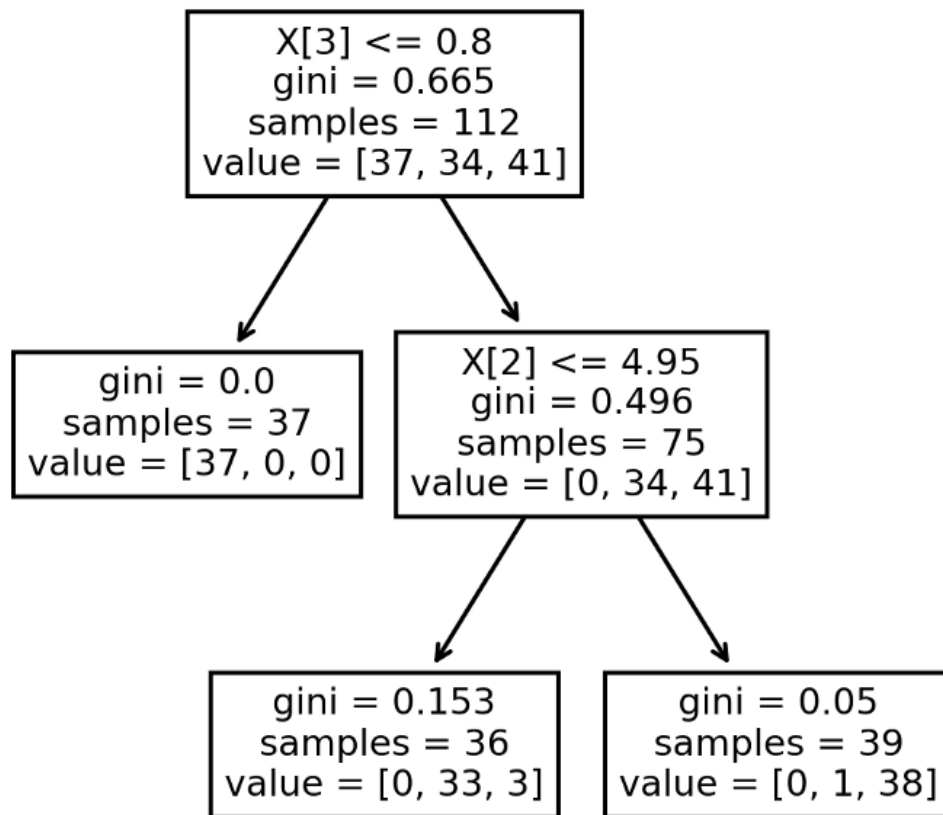
# Step 4: Predict labels of unseen (test) data
# Not doing this step in the tutorial
# clf.predict(X_test)
```

How to Visualize Decision Trees using Matplotlib

As of scikit-learn version 21.0 (roughly May 2019), Decision Trees can now be plotted with matplotlib using scikit-learn's `tree.plot_tree` without relying on the `dot` library which is a hard-to-install dependency which we will cover later on in the blog post.

The code below plots a decision tree using scikit-learn.

```
tree.plot_tree(clf);
```



This is not the most interpretable tree yet.

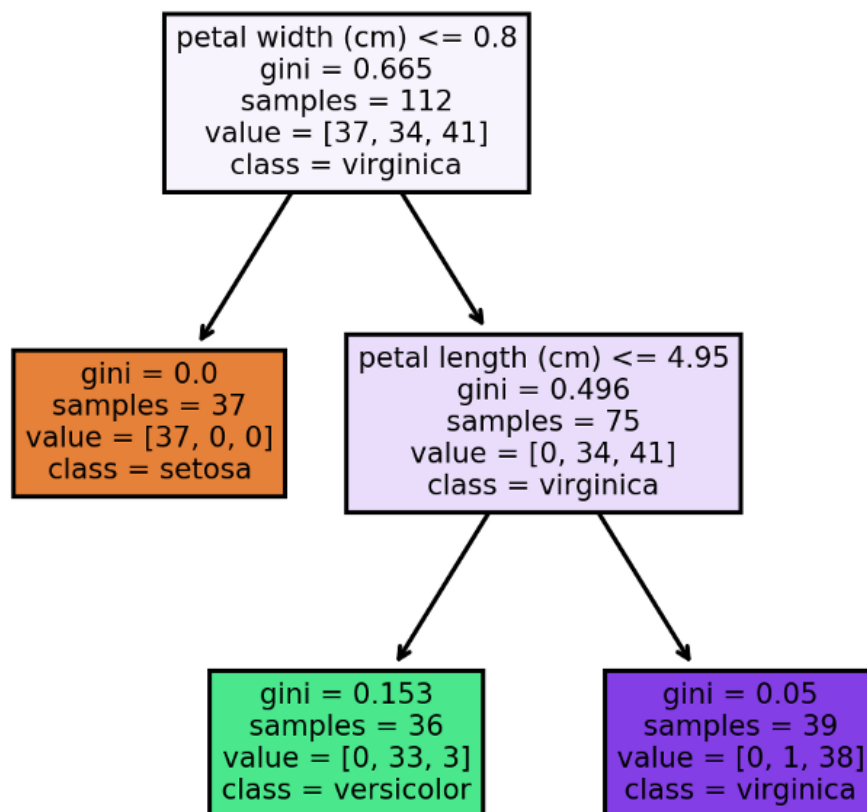
In addition to adding the code to allow you to save your image, the code below tries to make the decision tree more interpretable by adding in feature and class names (as well as setting `filled = True`).

```
fn=['sepal length (cm)','sepal width (cm)','petal length (cm)','petal  
width (cm)']  
cn=['setosa', 'versicolor', 'virginica']
```

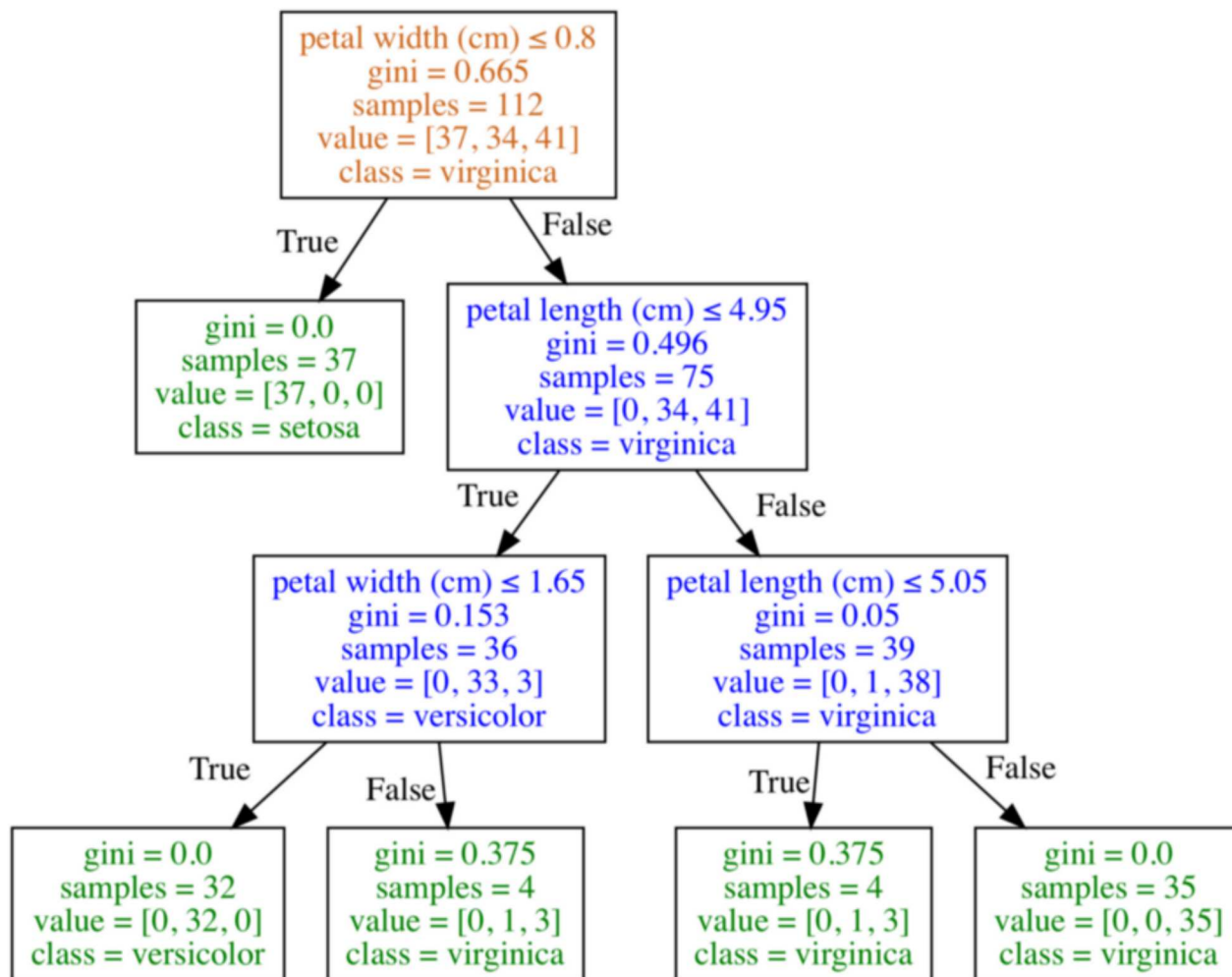
```
fig, axes = plt.subplots(nrows = 1,ncols = 1,figsize = (4,4),  
dpi=300)
```

```
tree.plot_tree(clf,  
               feature_names = fn,  
               class_names=cn,  
               filled = True);
```

```
fig.savefig('imagename.png')
```

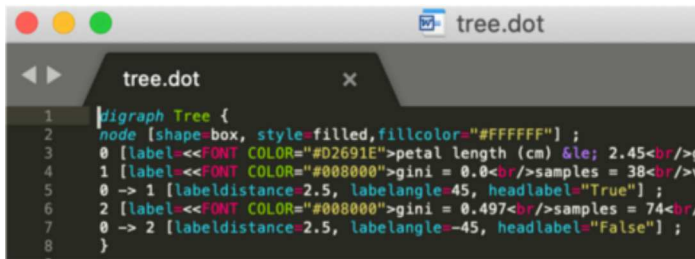


How to Visualize Decision Trees using Graphviz

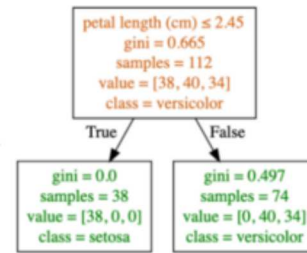


Decision Tree produced through Graphviz. Note that I edited the file to have text colors correspond to whether they are leaf/terminal nodes or decision nodes using a text editor.

Graphviz is open source graph visualization software. Graph visualization is a way of representing structural information as diagrams of abstract graphs and networks. In data science, one use of Graphviz is to visualize decision trees. I should note that the reason why I am going over Graphviz after covering Matplotlib is that getting this to work can be difficult. The first part of this process involves creating a dot file. A dot file is a Graphviz representation of a decision tree. The problem is that using Graphviz to convert the dot file into an image file (png, jpg, etc) can be difficult. There are a couple ways to do this including: installing `python-graphviz` through Anaconda, installing Graphviz through Homebrew (Mac), installing Graphviz executables from the official site (Windows), and using an online converter on the contents of your dot file to convert it into an image.



```
1 graph TD
2   node[petal length (cm) ≤ 2.45  
gini = 0.665  
samples = 112  
value = [38, 40, 34]  
class = versicolor]
3   node -- True --> node1[gini = 0.0  
samples = 38  
value = [38, 0, 0]  
class = setosa]
4   node -- False --> node2[gini = 0.497  
samples = 74  
value = [0, 40, 34]  
class = versicolor]
```



Dot File
(Graphviz representation of
decision tree)

Image File
(png, jpg, etc)

Creating the dot file is usually not a problem. Converting the dot file to a png file can be difficult.

Export your model to a dot file

The code below code will work on any operating system as python generates the dot file and exports it as a file named `tree.dot`.

```
tree.export_graphviz(clf,  
                     out_file="tree.dot",  
                     feature_names = fn,  
                     class_names=cn,  
                     filled = True)
```

Installing and Using Graphviz

Converting the dot file into an image file (png, jpg, etc) typically requires the installation of Graphviz which depends on your operating system and a host of other things. The goal of this section is to help people try and solve the common issue of getting the following error. `dot: command not found`.

```
!dot -Tpng -Gdpi=300 tree.dot -o tree.png
```

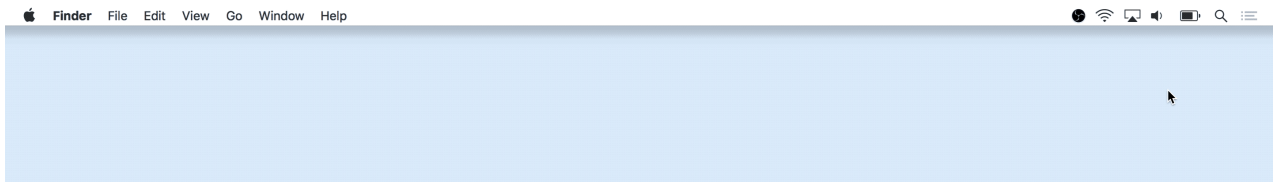
```
/bin/sh: dot: command not found
```

```
dot: command not found
```

How to Install and Use on Mac through Anaconda

To be able to install Graphviz on your Mac through this method, you first need to have Anaconda installed (If you don't have Anaconda installed, you can learn how to install it [here](#)).

Open a terminal. You can do this by clicking on the Spotlight magnifying glass at the top right of the screen, type terminal and then click on the Terminal icon.



Type the command below to install Graphviz.

```
conda install python-graphviz
```

After that, you should be able to use the `dot` command below to convert the dot file into a png file.

```
dot -Tpng tree.dot -o tree.png
```

How to Install and Use on Mac through Homebrew

If you don't have Anaconda or just want another way of installing Graphviz on your Mac, you can use [Homebrew](#). I previously wrote an article on how to install Homebrew and use it to convert a dot file into an image file [here](#) (see the Homebrew to Help Visualize Decision Trees section of the tutorial).

How to Install and Use on Windows through Anaconda

This is the method I prefer on Windows. To be able to install Graphviz on your Windows through this method, you first need to have Anaconda installed (If you don't have Anaconda installed, you can learn how to install it [here](#)).

Open a terminal/command prompt and enter the command below to install Graphviz.

```
conda install python-graphviz
```

After that, you should be able to use the `dot` command below to convert the dot file into a png file.

```
dot -Tpng tree.dot -o tree.png
```

```
Select Administrator: Anaconda Prompt

(base) C:\Users\Administrator\Desktop>dot
'dot' is not recognized as an internal or external command,
operable program or batch file.

(base) C:\Users\Administrator\Desktop>conda install python-graphviz
Solving environment: done

## Package Plan ##

  environment location: C:\ProgramData\Anaconda3

added / updated specs:
- python-graphviz

The following packages will be downloaded:

  package | build | size
  -----|-----|-----
  graphviz-2.38 | hfd603c8_2 | 37.7 MB
  python-graphviz-0.8.4 | py37_1 | 28 KB
  -----|-----|-----
  Total: | | 37.8 MB

The following NEW packages will be INSTALLED:

  graphviz: 2.38-hfd603c8_2
  python-graphviz: 0.8.4-py37_1

Proceed ([y]/n)? y

Downloading and Extracting Packages
graphviz-2.38 | 37.7 MB | ##### | 100%
python-graphviz-0.8.4 | 28 KB | ##### | 100%
Preparing transaction: done
Verifying transaction: done
Executing transaction: done

(base) C:\Users\Administrator\Desktop>dot -Tpng tree.dot -o tree.png

(base) C:\Users\Administrator\Desktop>_
```

Windows installing of Graphviz through conda. This should fix the 'dot' is not recognized as an internal or external command, operable program or batch file issue.

How to Install and Use on Windows through Graphviz Executable

If you don't have Anaconda or just want another way of installing Graphviz on your Windows, you can use the following link to [download and install it](#).



Graphviz - Graph Visualization Software

Windows Packages

Note: These Visual Studio packages do not alter the PATH variable or access the registry at all. If you wish to use the command-line interface to Graphviz or are using some other program that calls a Graphviz program, you will need to set the PATH variable yourself.

If you aren't familiar with altering the PATH variable and want to use dot on the command line, I encourage other approaches. [There are many Stackoverflow questions based on this particular issue.](#)

How to Use an Online Converter to Visualize your Decision Trees

If all else fails or you simply don't want to install anything, you can use [an online converter](#).

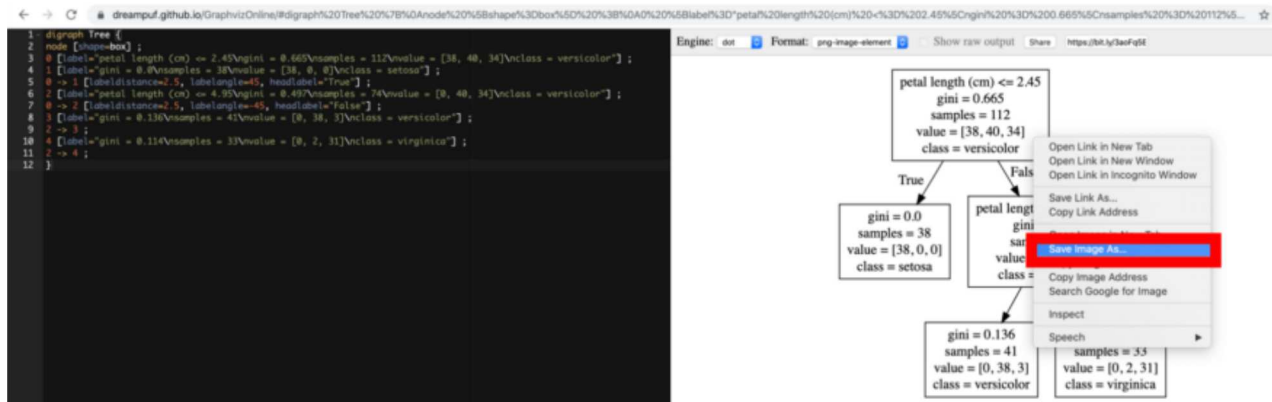
In the image below, I opened the file with Sublime Text (though there are many different programs that can open/read a dot file) and copied the content of the file.

The screenshot shows a Sublime Text editor window titled 'iris_depth2_decisionTreeExample.dot'. The editor contains a Graphviz dot file. A context menu is open over the file content, with the 'Copy' option highlighted. The file content is as follows:

```
1 digraph TD {
2   node [shape=box];
3   0 [label="petal length (cm) <= 2.45 \ngini = 0.665 \nsamples = 112 \nvalue = [38, 40, 34] \nclass = versicolor"];
4   1 [label="gini = 0.0 \nsamples = 38 \nvalue = [38, 0, 0] \nclass = versicolor"];
5   0 -> 1 [labeldistance=2.5, labelangle=45, headlabel="True"];
6   2 [label="petal length (cm) <= 4.95 \ngini = 0.497 \nsamples = 74 \nvalue = [0, 38, 36] \nclass = versicolor"];
7   0 -> 2 [labeldistance=2.5, labelangle=-45, headlabel="False"];
8   3 [label="gini = 0.136 \nsamples = 41 \nvalue = [0, 38, 3] \nclass = versicolor"];
9   2 -> 3;
10  4 [label="gini = 0.114 \nsamples = 33 \nvalue = [0, 2, 31] \nclass = versicolor"];
11  2 -> 4;
12 }
```

Copying the contents of a dot file

In the image below, I pasted the content from the dot file onto the left side of the online converter. You can then choose what format you want and then save the image on the right side of the screen.



Save visualization to computer

Keep in mind that there are other online converters that can help accomplish the same task.

How to Visualize Individual Decision Trees from Bagged Trees or Random Forests

Bagged Trees using Scikit-Learn (Python)



A weakness of decision trees is that they don't tend to have the best predictive accuracy. This is partially because of high variance, meaning that different splits in the training data can lead to very different trees. The [video](#) above covers Bagged Trees which is an ensemble model. This means using multiple learning algorithms to obtain a better predictive performance than could be obtained from any of the constituent learning algorithms alone. In this case, many trees protect each other from their individual errors. The interesting thing is that the thumbnail from the video above could be a diagram for either Bagged Trees or Random Forests (another ensemble model). The differences between how Bagged Trees and Random Forests models work is a subject for another blog, but what is important to note is that for both models we grow N trees where N is the number of decision trees a user specifies. Consequently after you fit a model, it would be nice to look at the individual decision trees that make up your model.

Fit a Random Forest Model using Scikit-Learn

In order to visualize individual decision trees, we need first need to fit a Bagged Trees or Random Forest model using scikit-learn (the code below fits a Random Forest model).

```
# Load the Breast Cancer (Diagnostic) Dataset
data = load_breast_cancer()
df = pd.DataFrame(data.data, columns=data.feature_names)
df['target'] = data.target

# Arrange Data into Features Matrix and Target Vector
X = df.loc[:, df.columns != 'target']
y = df.loc[:, 'target'].values

# Split the data into training and testing sets
X_train, X_test, Y_train, Y_test = train_test_split(X, y,
                                                    random_state=0)

# Random Forests in `scikit-learn` (with N = 100)
rf = RandomForestClassifier(n_estimators=100,
                           random_state=0)
```

```
rf.fit(X_train, Y_train)
```

Visualizing your Estimators

You can now view all the individual trees from the fitted model. In this section, I will visualize all the decision trees using matplotlib.

```
rf.estimators_
```

```
rf.estimators_
[DecisionTreeClassifier(ccp_alpha=0.0, class_weight=None, criterion='gini',
                        max_depth=None, max_features='auto', 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='deprecated',
                        random_state=209652396, splitter='best'),
 DecisionTreeClassifier(ccp_alpha=0.0, class_weight=None, criterion='gini',
                        max_depth=None, max_features='auto', 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='deprecated',
                        random_state=398764591, splitter='best'),
 DecisionTreeClassifier(ccp_alpha=0.0, class_weight=None, criterion='gini',
                        max_depth=None, max_features='auto', 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='deprecated',
                        random_state=924231285, splitter='best'),
 DecisionTreeClassifier(ccp_alpha=0.0, class_weight=None, criterion='gini',
                        max_depth=None, max_features='auto', 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='deprecated',
                        random_state=1000000000, splitter='best'),
 ...,
 DecisionTreeClassifier(ccp_alpha=0.0, class_weight=None, criterion='gini',
                        max_depth=None, max_features='auto', 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='deprecated',
                        random_state=1000000000, splitter='best'))

# We have 100 estimators
print(len(rf.estimators_))

100

rf.estimators_[0]
DecisionTreeClassifier(ccp_alpha=0.0, class_weight=None, criterion='gini',
                        max_depth=None, max_features='auto', 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='deprecated',
                        random_state=209652396, splitter='best')
```

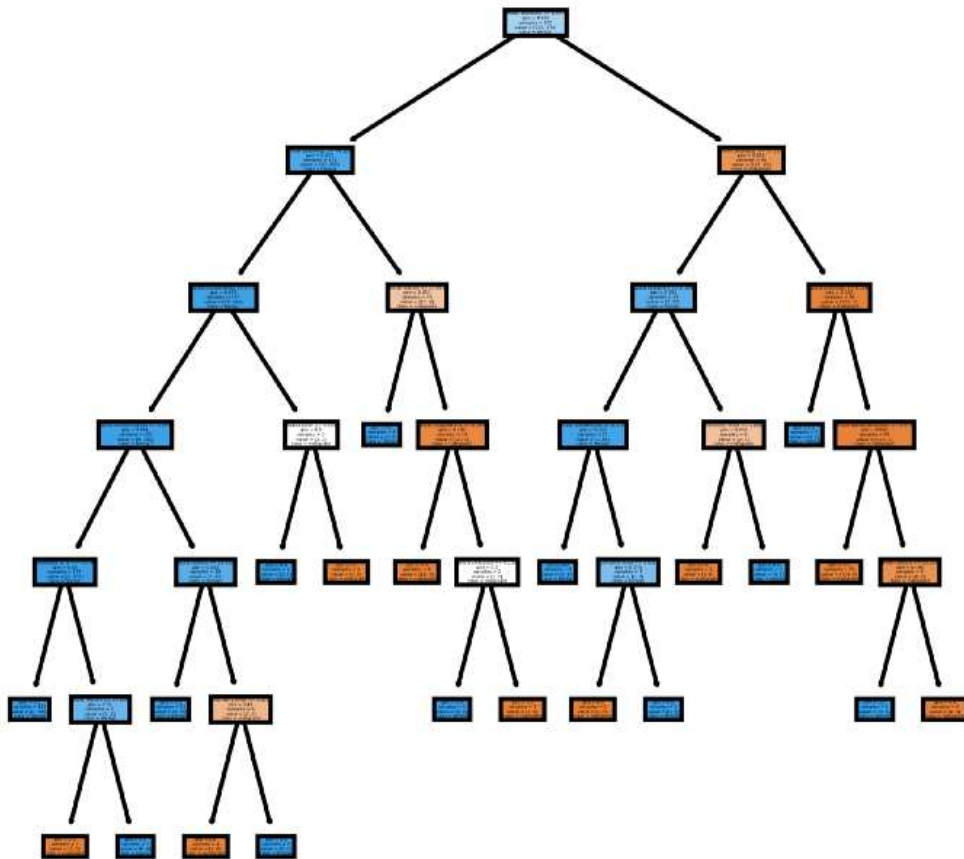
In this example, notice how we have 100 estimators.

You can now visualize individual trees. The code below visualizes the first decision tree.


```

fn=data.feature_names
cn=data.target_names
fig, axes = plt.subplots(nrows = 1,ncols = 1,figsize = (4,4),
dpi=800)
tree.plot_tree(rf.estimators_[0],
               feature_names = fn,
               class_names=cn,
               filled = True);
fig.savefig('rf_individualtree.png')

```



Note that individual trees in Random Forest and Bagged trees are grow deep

You can try to use matplotlib subplots to visualize as many of the trees as

you like. The code below visualizes the first 5 decision trees. I personally don't prefer this method as it is even harder to read.

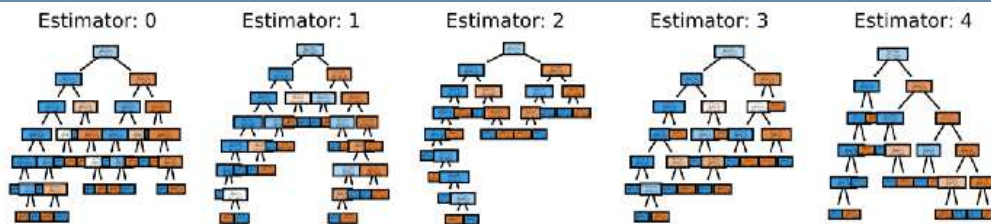
```
# This may not be the best way to view each estimator as it is small

fn=data.feature_names
cn=data.target_names
fig, axes = plt.subplots(nrows = 1,ncols = 5,figsize = (10,2),
dpi=3000)

for index in range(0, 5):
    tree.plot_tree(rf.estimators_[index],
                    feature_names = fn,
                    class_names=cn,
                    filled = True,
                    ax = axes[index]);

    axes[index].set_title('Estimator: ' + str(index), fontsize = 11)

fig.savefig('rf_5trees.png')
```



miss. [Take a look.](#)

Your email

✉ Get this newsletter

By signing up, you will create a Medium account if you don't already have one. Review our [Privacy Policy](#) for more information about our privacy practices.

you just want to see each of the 100 estimators for the random forest model fit in this tutorial without running the code, you can look at the video below.

Python Matplotlib Decision Tree Scikit Learn Graphviz

100 Estimators for a Random Forest



Concluding Remarks

This tutorial covered how to visualize decision trees using Graphviz and Matplotlib. That the way to visualize decision trees using Matplotlib is a newer method so it might change or be improved upon in the future. Graphviz is currently more flexible as you can always modify your dot files to make them more visually appealing like I did using [dot language](#) or even just alter the orientation of your decision tree. One thing we did not cover was how to use [dtreeviz](#) which is another library that can visualize decision trees. There is an excellent post on it [here](#).

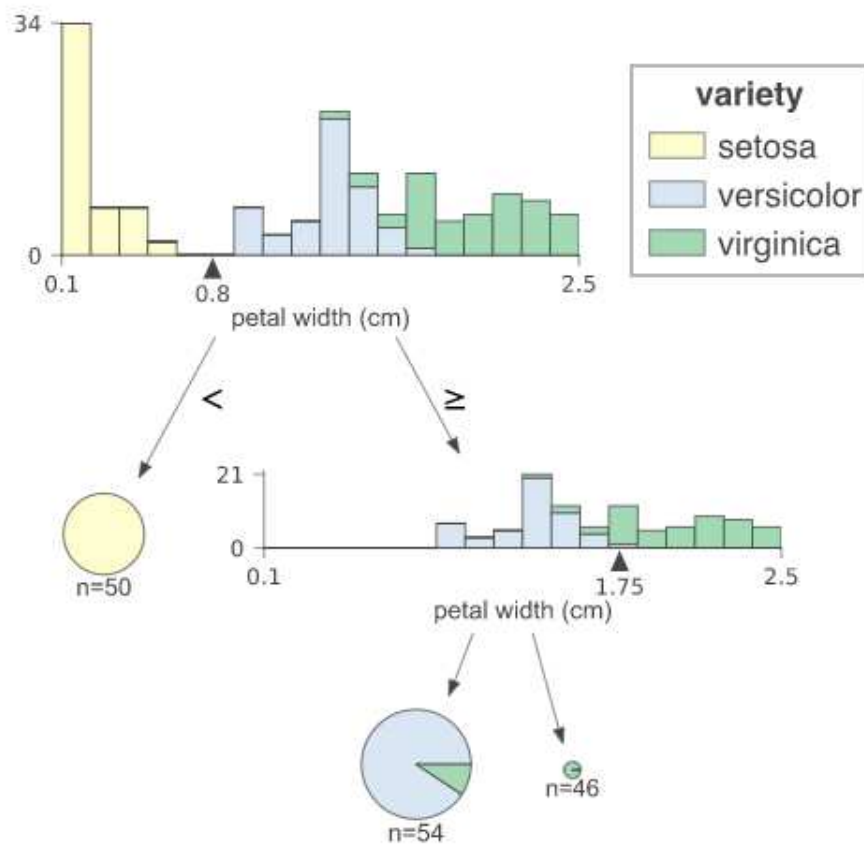


Image from produced by [dtreeviz library](#).

My next machine learning tutorial goes over [How to Speed up Scikit-Learn Model Training](#). If you have any questions or thoughts on the tutorial, feel free to reach out in the comments below or through [Twitter](#).