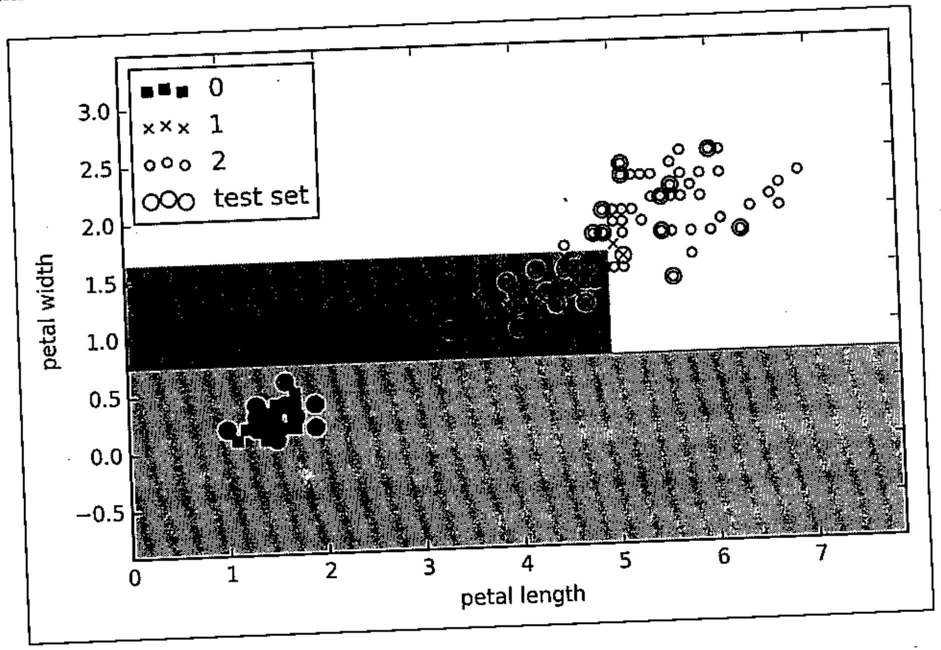
Building a decision tree

Decision trees can build complex decision boundaries by dividing the feature space into rectangles. However, we have to be careful since the deeper the decision tree, the more complex the decision boundary becomes, which can easily result in overfitting. Using scikit-learn, we will now train a decision tree with a maximum depth of 3 using entropy as a criterion for impurity. Although feature scaling may be desired for visualization purposes, note that feature scaling is not a requirement for decision tree algorithms. The code is as follows:

```
>>> from sklearn.tree import DecisionTreeClassifier
>>> tree = DecisionTreeClassifier(criterion='entropy',
...                               max_depth=3, random_state=0)
>>> tree.fit(X_train, y_train)
>>> X_combined = np.vstack((X_train, X_test))
>>> y_combined = np.hstack((y_train, y_test))
>>> plot_decision_regions(X_combined, y_combined,
...                classifier=tree, test_idx=range(105,150))
>>>plt.xlabel('petal length [cm]')
>>>plt.ylabel('petal width [cm]')
>>> plt.legend(loc='upper left')
>>> plt.show()
```

After executing the preceding code example, we get the typical axis-parallel decision boundaries of the decision tree:



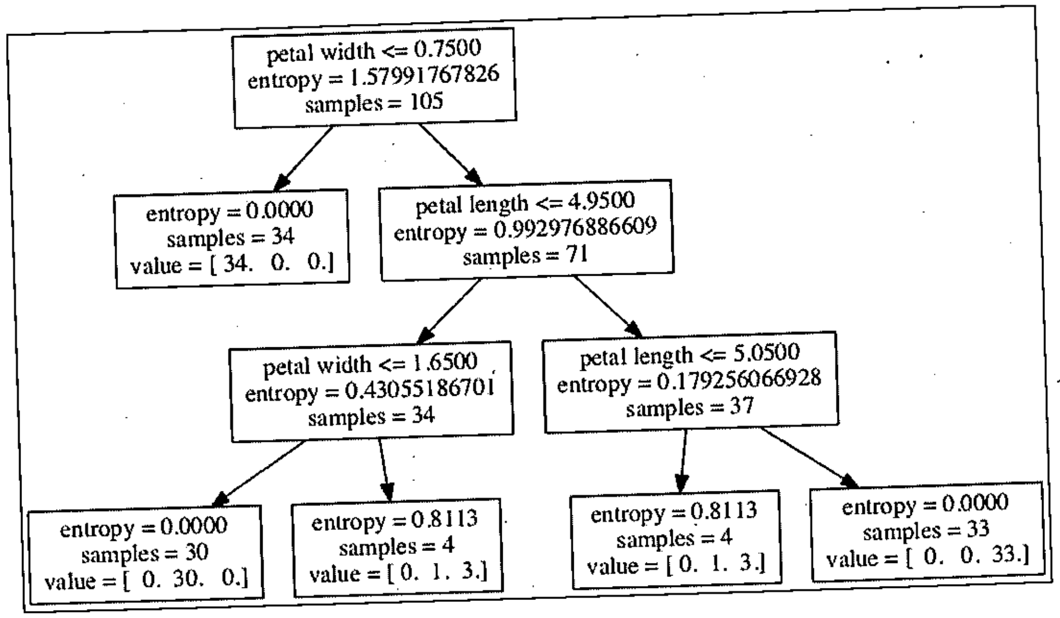
A nice feature in scikit-learn is that it allows us to export the decision tree as a `.dot` file after training, which we can visualize using the GraphViz program. This program is freely available at `http://www.graphviz.org` and supported by Linux, Windows, and Mac OS X.

First, we create the `.dot` file via scikit-learn using the `export_graphviz` function from the `tree` submodule, as follows:

```
>>> from sklearn.tree import export_graphviz
>>> export_graphviz(tree,
...                 out_file='tree.dot',
...                 feature_names=['petal length', 'petal width'])
```

After we have installed GraphViz on our computer, we can convert the `tree.dot` file into a PNG file by executing the following command from the command line in the location where we saved the `tree.dot` file:

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



Looking at the decision tree figure that we created via GraphViz, we can now nicely trace back the splits that the decision tree determined from our training dataset. We started with 105 samples at the root and split it into two child nodes with 34 and 71 samples each using the **petal width** cut-off ≤ 0.75 cm. After the first split, we can see that the left child node is already pure and only contains samples from the Iris-Setosa class (entropy = 0). The further splits on the right are then used to separate the samples from the Iris-Versicolor and Iris-Virginica classes.