Applied Machine Learning

Decision Trees

Kevyn Collins-Thompson

Associate Professor of Information & Computer Science
University of Michigan

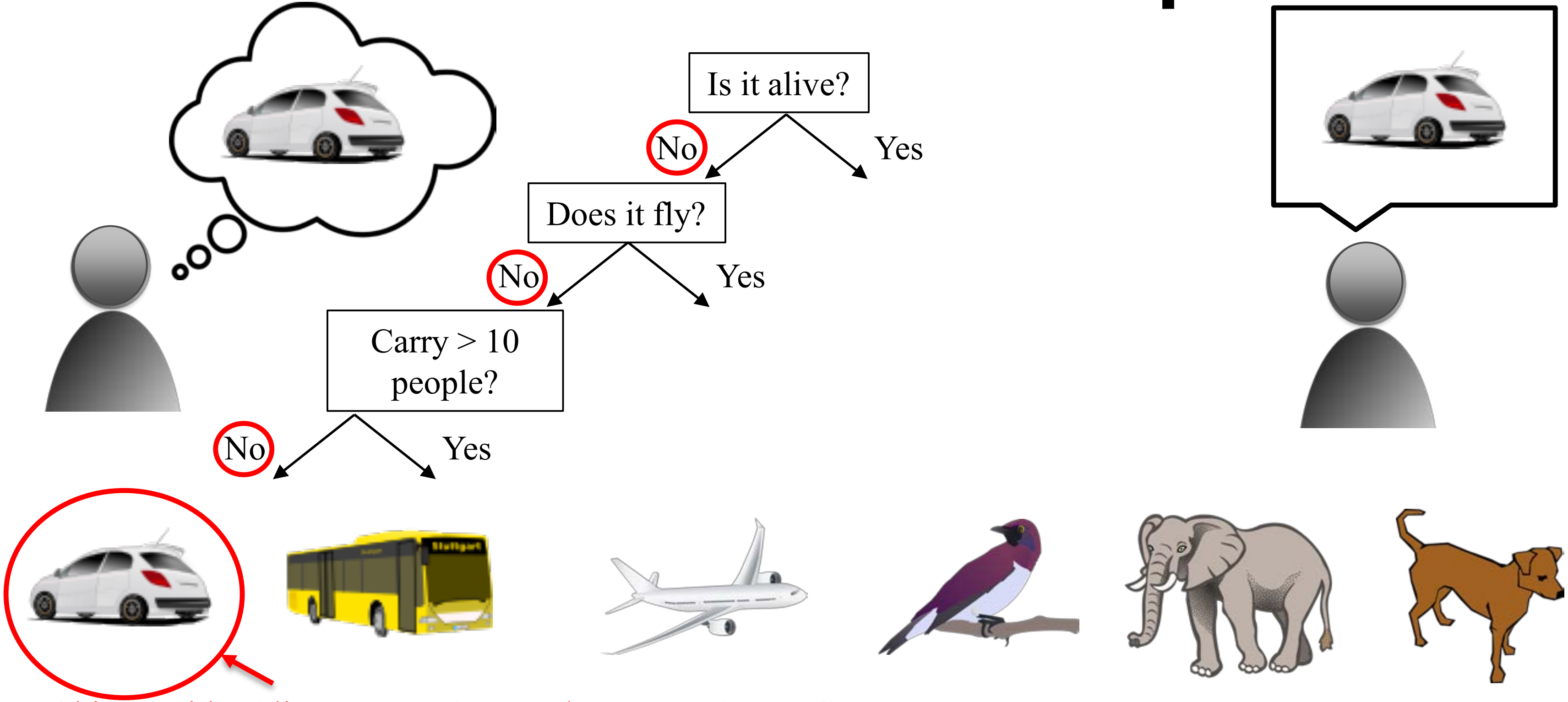
Decision Tree Example

Is it alive?

No | Yes

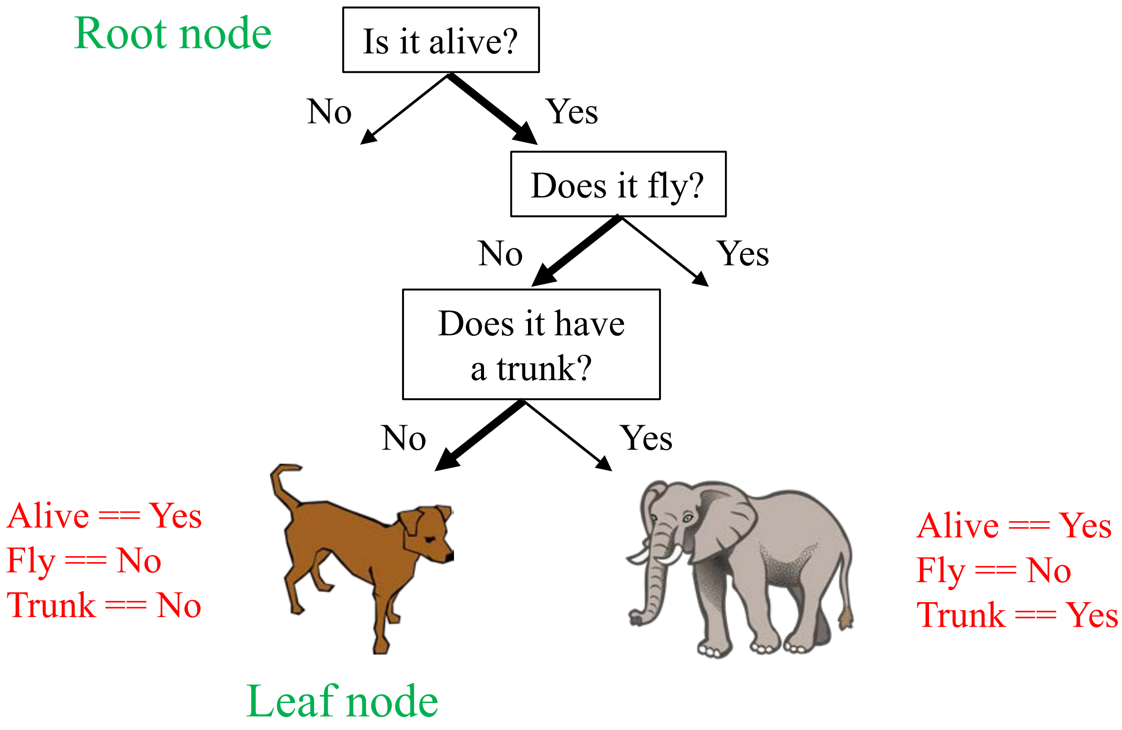
Objects with Alive == No

Decision Tree Example



Objects with: Alive == No AND Fly == No AND Carry > 10 == No

Decision Tree Example



The Iris Dataset

Iris setosa *Iris versicolor* *Iris virginica*

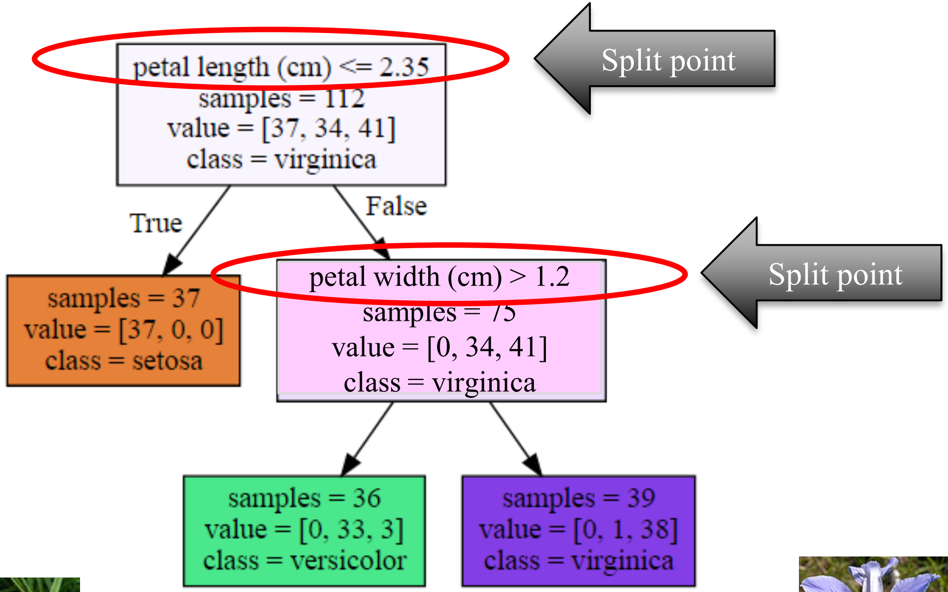
150 flowers
3 species
50 examples/species

Photo credit: Radomił Binek via https://en.wikipedia.org/wiki/Iris_flower_data_set

Decision Tree Splits

samples at this leaf have:
petal length <= 2.35

setosa



versicolor

samples at this leaf have:
petal length > 2.35
AND petal width <= 1.2

samples at this leaf have:
petal length > 2.35
AND petal width > 1.2

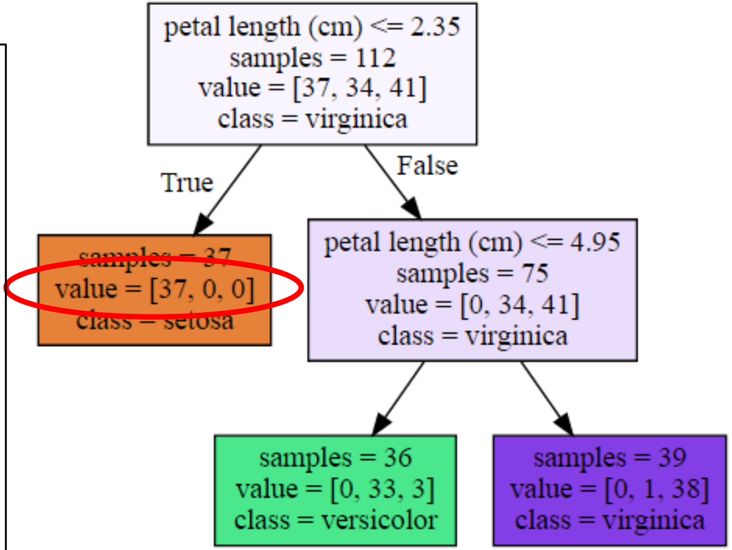
virginica

Informativeness of Splits

The *value* list gives the number of samples of each class that end up at this leaf node during training.

The iris dataset has 3 classes, so there are three counts.

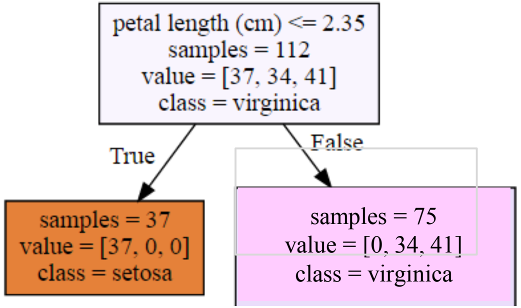
This leaf has 37 setosa samples, zero versicolor, and zero virginica samples.



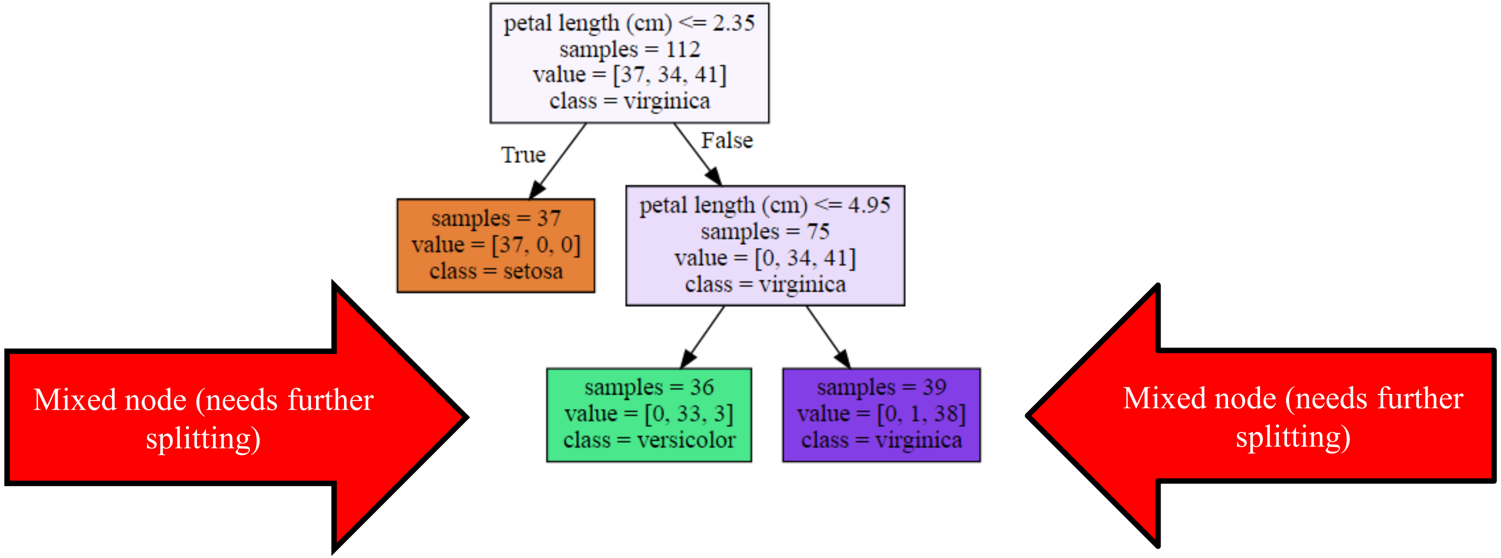
This leaf has 0 setosa, 33 versicolor, and 3 virginica samples.

This leaf has 0 setosa, 1 versicolor, and 38 virginica samples.

Pure node (all one class: perfect classification)

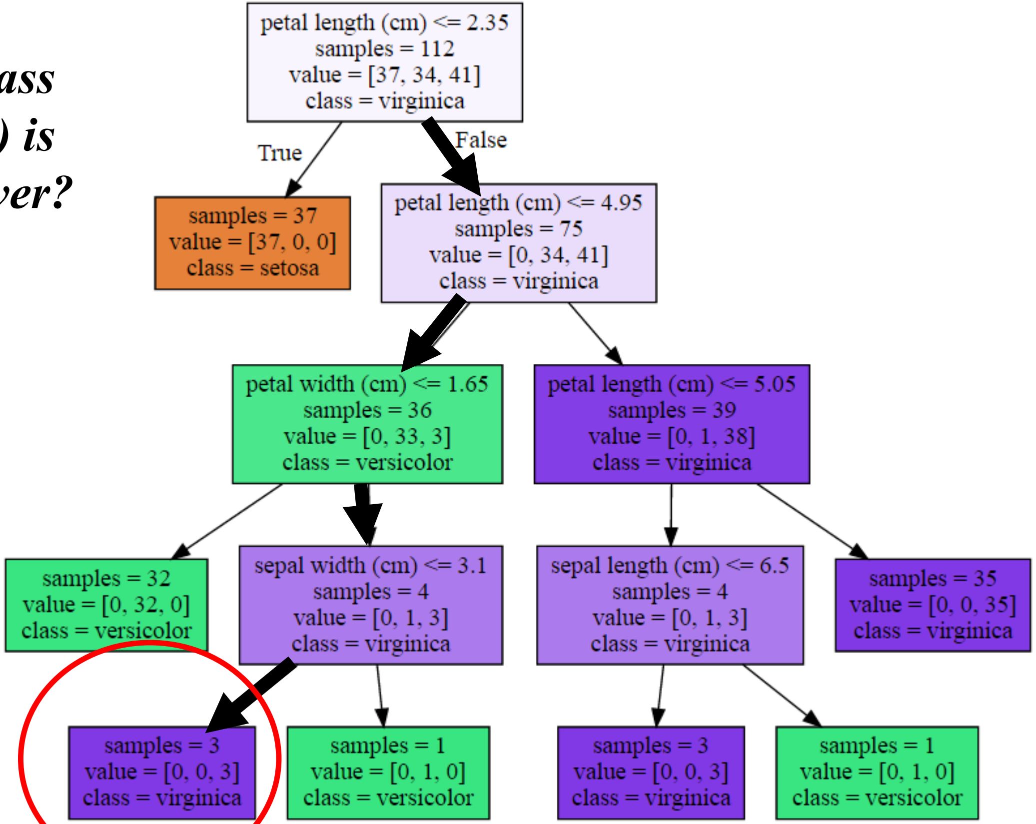


Mixed node (mixture of classes, still needs further splitting)


petal length (cm) <= 2.35
samples = 112
value = [37, 34, 41]
class = virginica
True
False
samples = 37
value = [37, 0, 0]
class = setosa
petal length (cm) <= 4.95
samples = 75
value = [0, 34, 41]
class = virginica
samples = 36
value = [0, 33, 3]
class = versicolor
samples = 39
value = [0, 1, 38]
class = virginica
Mixed node (needs further splitting)
Mixed node (needs further splitting)

What class (species) is this flower?

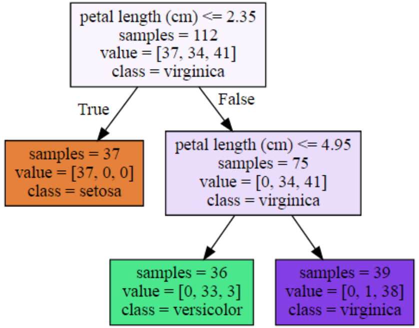
petal length: 3.0
petal width: 2.0
sepal width: 2.0
sepal length: 4.2



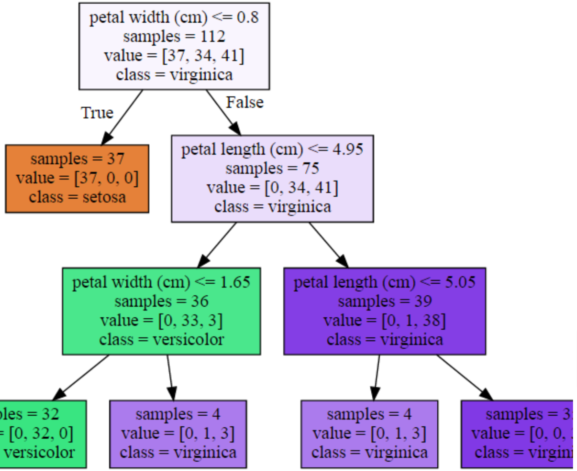
Leaf counts are: setosa = 0, versicolor = 0, virginica = 3
Predicted class is majority class at this leaf: **virginica**

Controlling the Model Complexity of Decision Trees

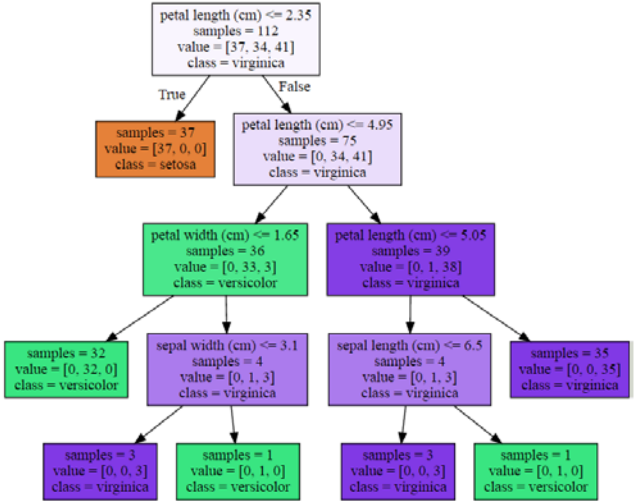
`max_depth = 2`



`max_depth = 3`

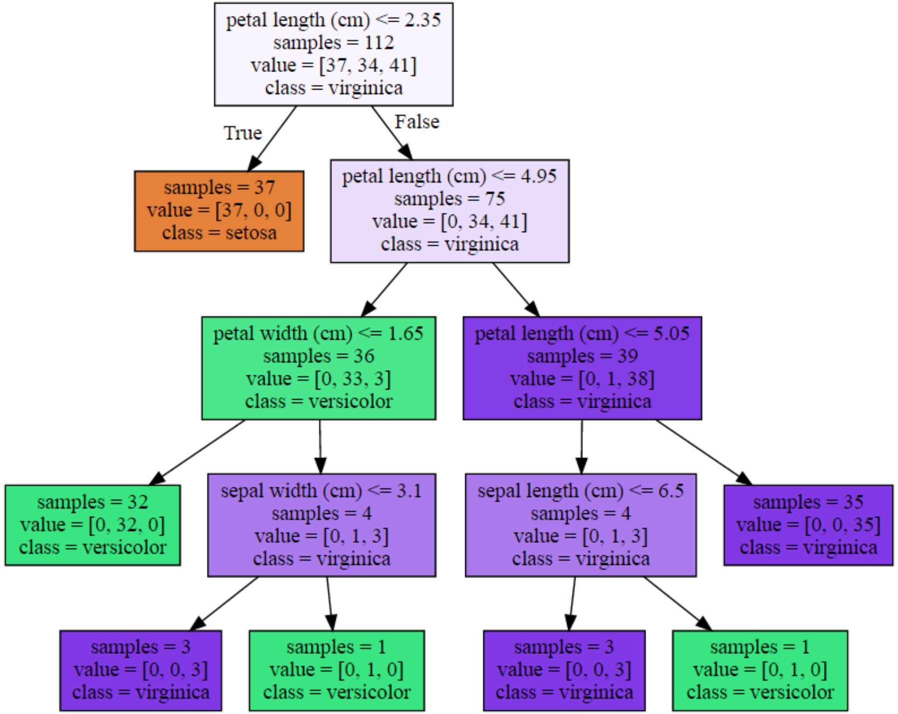


`max_depth = 4`



Other parameters: Max. # of leaf nodes: `max_leaf_nodes` Min. samples to consider splitting: `min_samples_leaf`

Visualizing Decision Trees

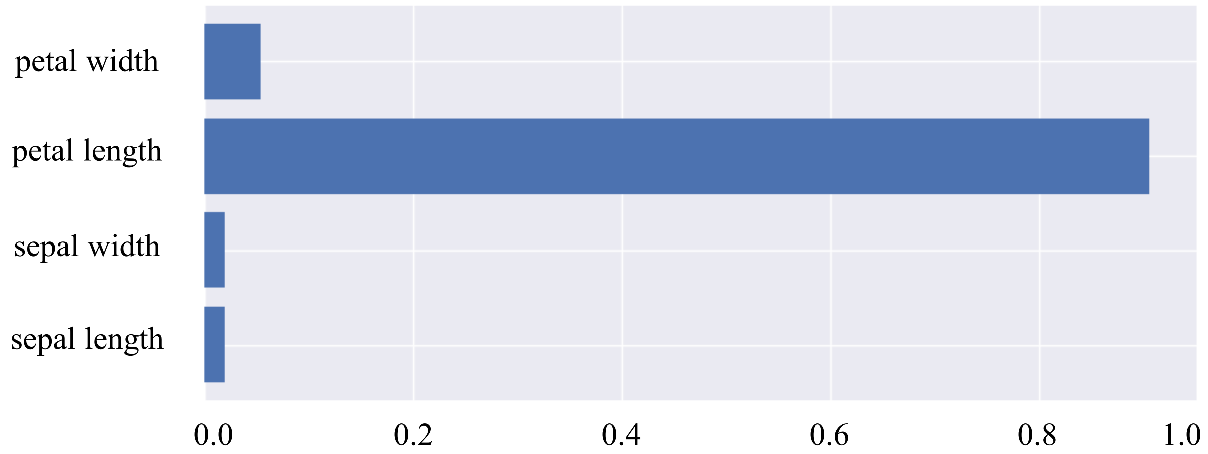


See: `plot_decision_tree()` function in `adspy_shared_utilities.py` code

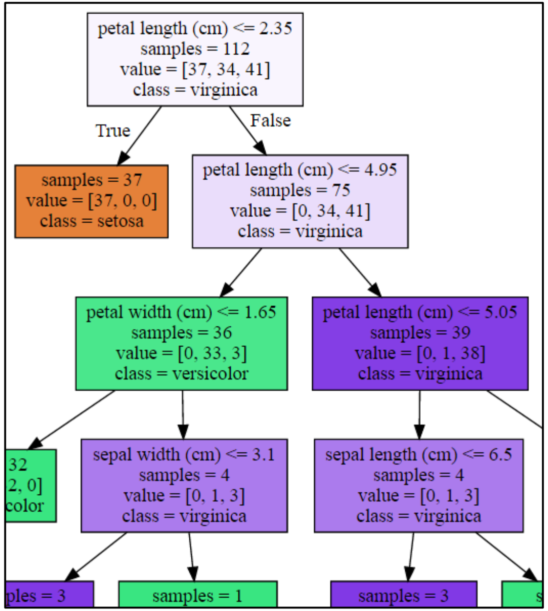
Feature Importance: How important is a feature to overall prediction accuracy?

- A number between 0 and 1 assigned to each feature.
- Feature importance of 0 → the feature was not used in prediction.
- Feature importance of 1 → the feature predicts the target perfectly.
- All feature importances are normalized to sum to 1.

Feature Importance Chart



Feature importance



Decision tree

See: `plot_feature_importances()` function in `adspy_shared_utilities.py` code

Decision Trees: Pros and Cons

Pros:

- Easily visualized and interpreted.
- No feature normalization or scaling typically needed.
- Work well with datasets using a mixture of feature types (continuous, categorical, binary)

Cons:

- Even after tuning, decision trees can often still overfit.
- Usually need an ensemble of trees for better generalization performance.

Decision Trees: DecisionTreeClassifier Key Parameters

- `max_depth`: controls maximum depth (number of split points). Most common way to reduce tree complexity and overfitting.

- `min_samples_leaf`: threshold for the minimum # of data instances a leaf can have to avoid further splitting.

- `max_leaf_nodes`: limits total number of leaves in the tree.

- In practice, adjusting only one of these (e.g. `max_depth`) is enough to reduce overfitting.