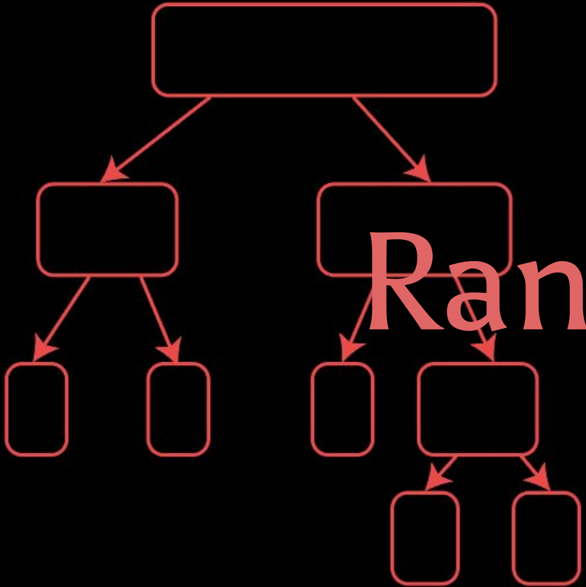
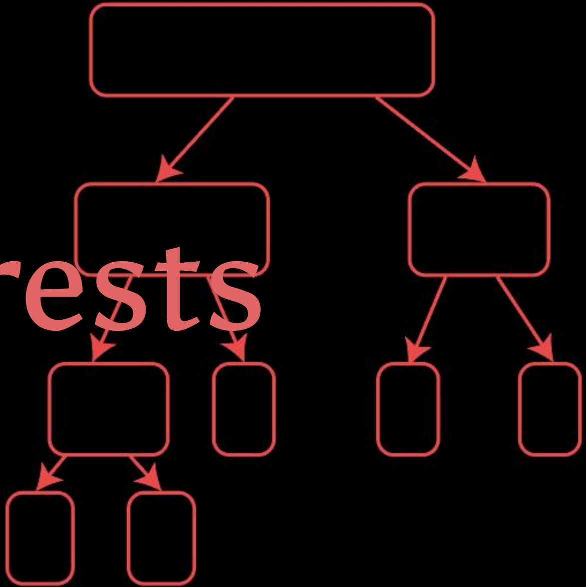
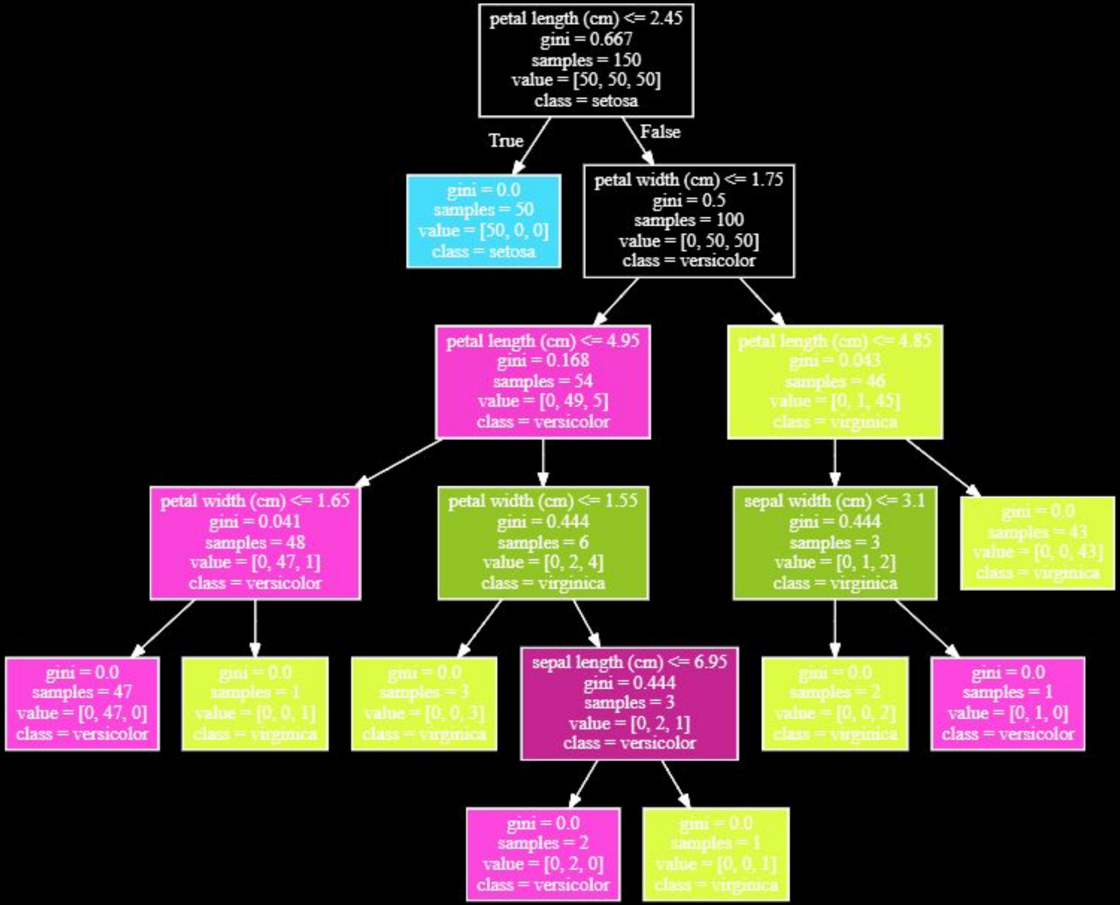


Random Forests



Decision Trees

A model which works to predict whether items in the dataset belong to different classes or regression values by iteratively splitting up the dataset (in the case of Scikit-Learn) based on quantitative decisions on various features.



petal length (cm) <= 2.45
gini = 0.667
samples = 150
value = [50, 50, 50]
class = setosa

True

False

gini = 0.0
samples = 50
value = [50, 0, 0]
class = setosa

petal width (cm) <= 1.75
gini = 0.5
samples = 100
value = [0, 50, 50]
class = versicolor

petal length (cm) <= 4.95
gini = 0.168
samples = 54
value = [0, 49, 5]
class = versicolor

petal length (cm) <= 4.85
gini = 0.043
samples = 46
value = [0, 1, 45]
class = virginica

petal width (cm) <= 1.65
gini = 0.041
samples = 48
value = [0, 47, 1]
class = versicolor

petal width (cm) <= 1.55
gini = 0.444
samples = 6
value = [0, 2, 4]
class = virginica

sepal width (cm) <= 3.1
gini = 0.444
samples = 3
value = [0, 1, 2]
class = virginica

gini = 0.0
samples = 43
value = [0, 0, 43]
class = virginica

gini = 0.0
samples = 47
value = [0, 47, 0]
class = versicolor

gini = 0.0
samples = 1
value = [0, 0, 1]
class = virginica

gini = 0.0
samples = 3
value = [0, 0, 3]
class = virginica

sepal length (cm) <= 6.95
gini = 0.444
samples = 3
value = [0, 2, 1]
class = versicolor

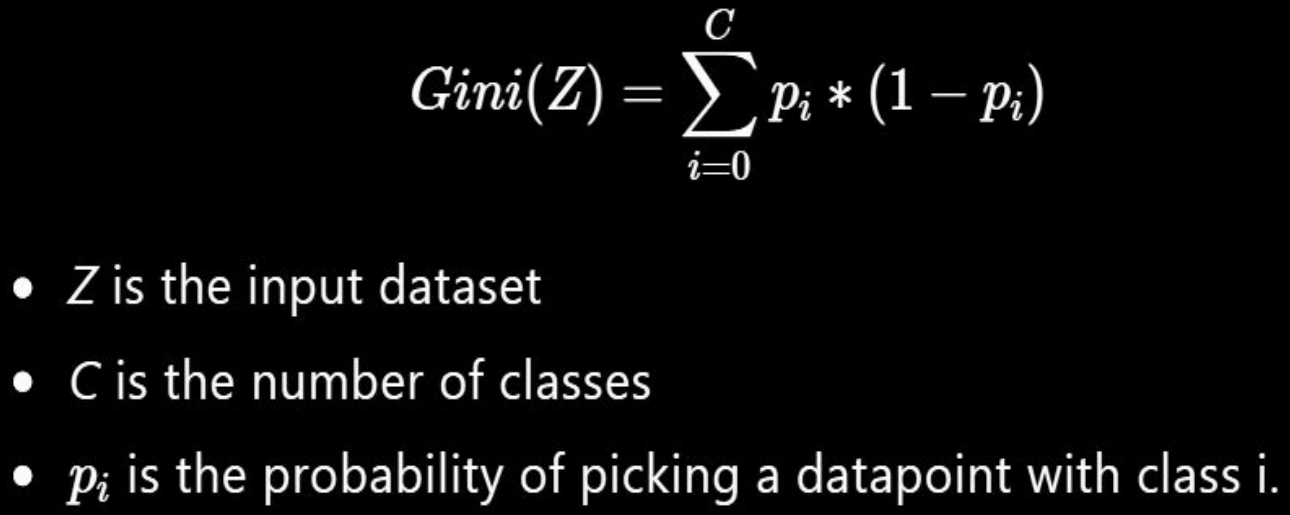
gini = 0.0
samples = 2
value = [0, 0, 2]
class = virginica

gini = 0.0
samples = 1
value = [0, 1, 0]
class = versicolor

gini = 0.0
samples = 2
value = [0, 2, 0]
class = versicolor

gini = 0.0
samples = 1
value = [0, 0, 1]
class = virginica

Gini Impurity



$$Gini(Z) = \sum_{i=0}^{C} p_i * (1 - p_i)$$

- Z is the input dataset
- C is the number of classes
- p_i is the probability of picking a datapoint with class i.

Gini Impurity

Can be thought of in a few ways:

- The quality of a split if the dataset were to be split among those classes.
- Measures the probability we would correctly classify a datapoint, given the current class distribution.

Entropy

$$Entropy(Z) = -(\sum_{i=0}^{C} p_i * log_2(p_i))$$

- Z is the input dataset
- C is the number of classes
- p_i is the probability of picking a datapoint with class i.

Entropy

Can be thought of in a few ways:

- A measure of homogeneity within an existing dataset split among a set of classes C.
  - Equal to 0 when completely homogeneous
  - Equal to 1 when completely non homogeneous

Decision Tree Positives

- Easily understood and interpreted because acts as a white box -- can openly see how the model functions to produce the output.
- Cost of using a decision tree is logarithmic to the number of datapoints
  - CS61B Students: think of the runtime of a binary tree
- Can handle both numeric and categorical data with ease.

Decision Tree Negatives

- Can easily create over-complex trees which do not generalise to new data well.
- Can be highly unstable in functioning when small changes are made to the training datapoints.
- Since Decision Trees rely on greedy heuristics, they cannot be guaranteed to return a globally optimal decision tree.

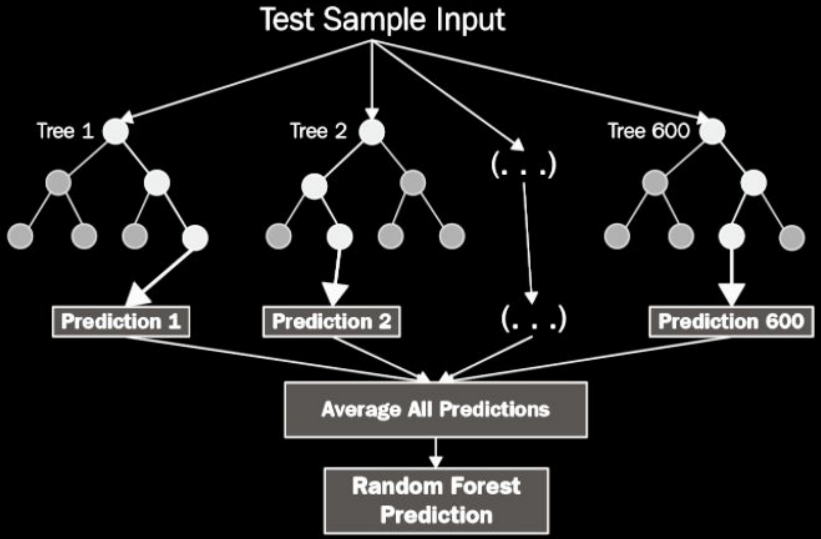
Random Forests

Random Forests

Contain an *ensemble* of *n* decision trees and *m* bootstrap samples, which stop splitting after reaching a specified depth *d.*

Each Decision Tree runs its course and generates a prediction.

The ensemble of predictions are average to create the Random Forest Prediction



Bagging (Bootstrap AGGregatING)

The process of randomly choosing subsets from a dataset to train different decision trees on.

Random Forests Algorithm

1. At the current node, select p features randomly from the available features D.

Random Forests Algorithm

1. At the current node, select p features randomly from the available features D.

2. Using the specified splitting metric, compute the ideal split point for tree i of n. Split this node into daughter nodes, which have various subsets of features from their parent node.

Random Forests Algorithm

1. At the current node, select p features randomly from the available features D.

2. Using the specified splitting metric, compute the ideal split point for tree i of n. Split this node into daughter nodes, which have various subsets of features from their parent node.

3. Repeat the above steps until a the specified tree depth d has been reached.

Random Forests Algorithm

1. At the current node, select *p* features randomly from the available features *D*.

2. Using the specified splitting metric, compute the ideal split point for tree *i* of *n*. Split this node into daughter nodes, which have various subsets of features from their parent node.

3. Repeat the above steps until a the specified tree depth *d* has been reached.

4. For each tree *i* of *k* in the forest, repeat steps 1-3

Random Forests Algorithm

1. At the current node, select p features randomly from the available features D.

2. Using the specified splitting metric, compute the ideal split point for tree i of n. Split this node into daughter nodes, which have various subsets of features from their parent node.

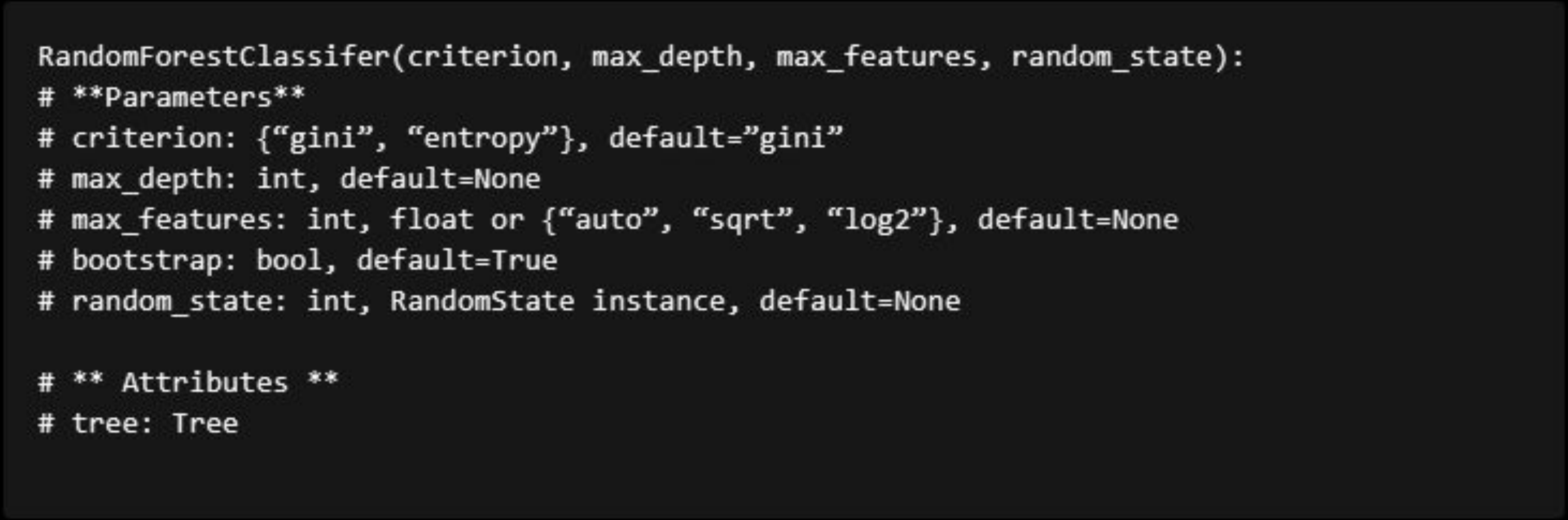
3. Repeat the above steps until a the specified tree depth d has been reached.

4. For each tree i of k in the forest, repeat steps 1-3

5. Aggregate or vote on the outputs calculated by each of the trees.

Scikit-Learn for Decision Trees and Random Forests

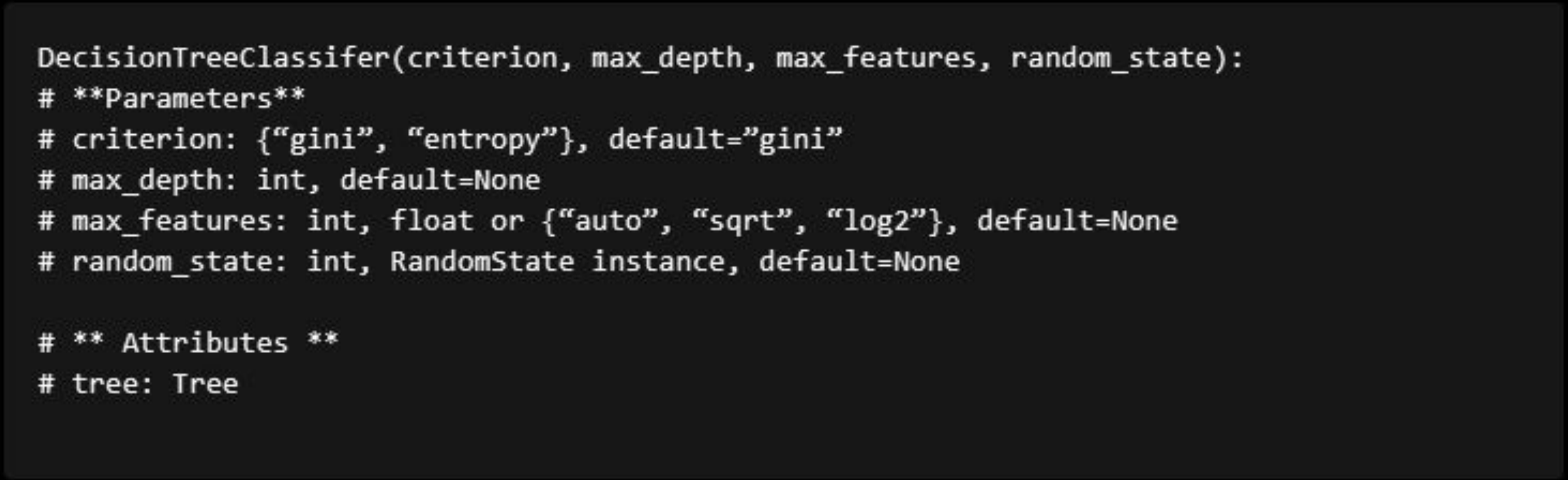
Random Forest Classifier



```
RandomForestClassifer(criterion, max_depth, max_features, random_state):
# **Parameters**
# criterion: {“gini”, “entropy”}, default=”gini”
# max_depth: int, default=None
# max_features: int, float or {“auto”, “sqrt”, “log2”}, default=None
# bootstrap: bool, default=True
# random_state: int, RandomState instance, default=None

# ** Attributes **
# tree: Tree
```

Decision Tree Classifier



```
DecisionTreeClassifer(criterion, max_depth, max_features, random_state):
# **Parameters**
# criterion: {“gini”, “entropy”}, default=”gini”
# max_depth: int, default=None
# max_features: int, float or {“auto”, “sqrt”, “log2”}, default=None
# random_state: int, RandomState instance, default=None

# ** Attributes **
# tree: Tree
```