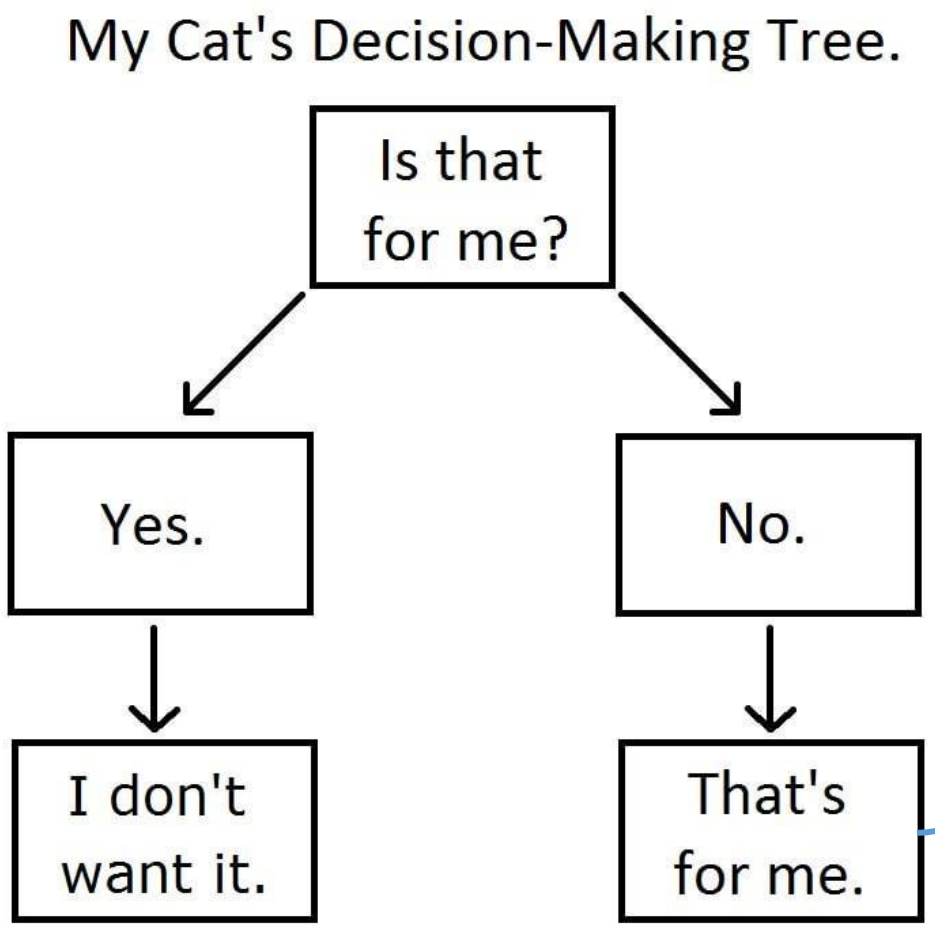
Decision Tree Classifier



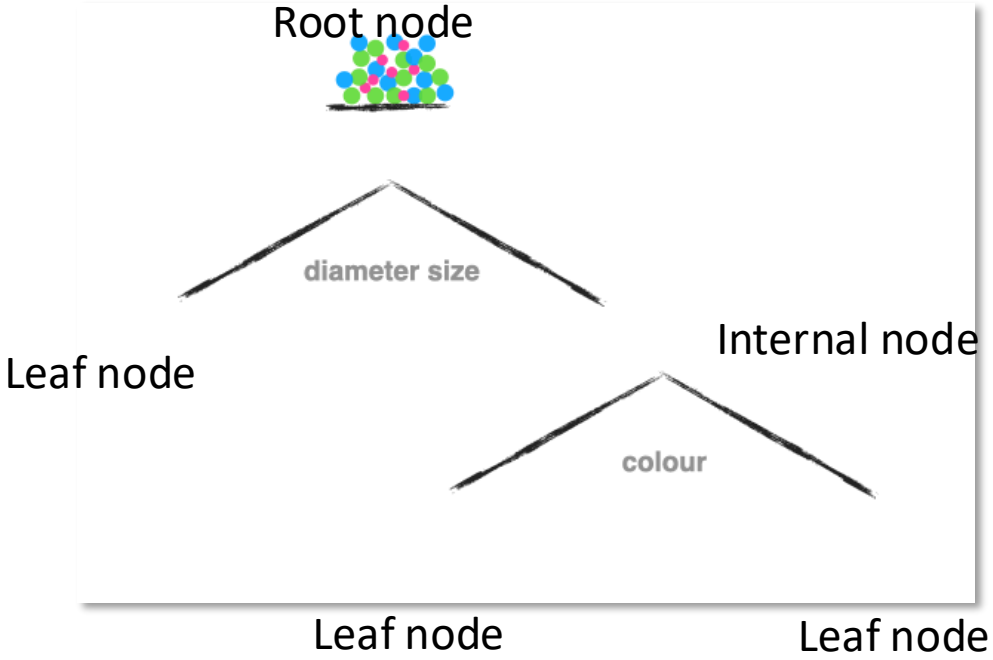
Decision Tree

Definition: Decision tree classifier is a supervised learning classification algorithm by **recursively partitioning** the feature space into subsets based on decision rules.

Definition: The **root node** N_r is the **initial decision point** in a tree, where the dataset D is split into two or more subsets based on a specific feature f^*.

Definition: An **internal node** N_i at depth d in the tree is a decision point where the dataset D_i, passed from the parent node, is split based on feature f_i^*.

Definition: A **leaf node** L at depth d_L is a terminal node in the tree where the dataset subset D_L is no longer split, and the final output O_L is a predicted class label.



When to use decision tree?

1. Handles both categorical and numerical data.
2. Suitable for small datasets.
3. When interpretability is important.

https://ml2gifs.substack.com/p/decision-trees

Finding the Best Split

Algorithm: ID3, C4.5, C5.0 and CART
"scikit-learn uses an optimized version of the CART algorithm"
CART (Classification and Regression Trees)
For CART algorithm, **each split is binary.**

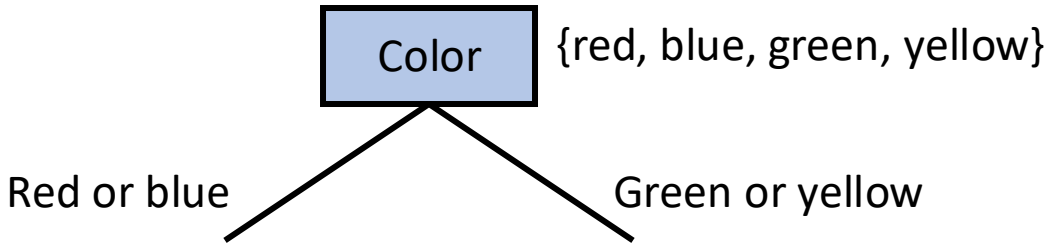
Definition: A categorical feature (or nominal feature) is a variable that represents a set of discrete categories or labels.
Definition: A numerical feature (or continuous feature) is a variable that represents quantitative data.

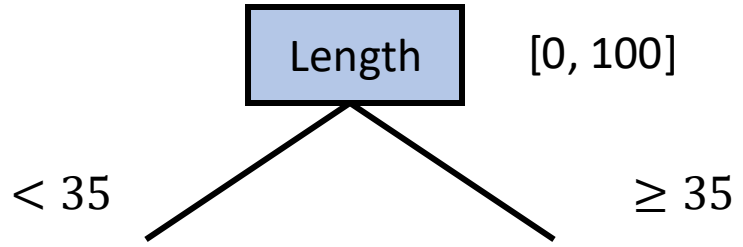
Can we find the smallest possible decision tree that accurately classifies the training set?
No! This is an NP-hard problem.
[Hyafil & Rivest, *Information Processing Letters*, 1976]

Instead, we'll use an information-theoretic heuristic to greedily choose splits.

To find the best split at each node:

1. Select a feature.
2. Select a threshold or subset.
3. Maximize the "purity" of the resulting subsets. The subsets contain **predominantly** one class.





Splitting Criterion: Information Gain

Information theory background

- Consider a problem in which you are using a code to communicate information to a receiver.
- Example: as cars go past, you are communicating the manufacturer of each car.
- Support there are only four types of cars, we could use the following code

Type	Code
Honda	11
Toyota	10
Mitsubishi	01
Subaru	00

- Expected number of bits we have to communicate: 2 bits/car.

We can improve the results if the car types are not equiprobable.

- *Huffman coding: a sequence of code can be unambiguously decoded. (Prefix-free coding)*
- Optimal code uses $-\log_2 P(x)$ bits for event with probability $P(x)$.

Type	Probability	# bits	Code
Honda	0.5	1	1
Toyota	0.25	2	01
Mitsubishi	0.125	3	001
Subaru	0.125	3	000

$$-\sum_{x \in \text{values}(X)} P(x) \log_2 P(x)$$

- Expected number of bits we have to communicate: 1.75 bits/car.

Splitting Criterion: Information Gain

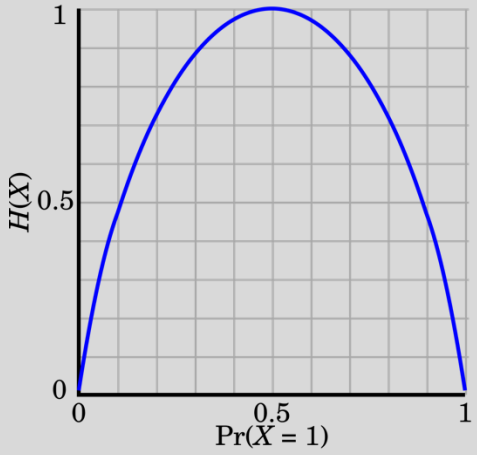
Definition: Entropy is a measure of the uncertainty or information content associated with a **random variable**.

It quantifies the expected **number of bits** needed to encode the outcomes of a random variable, given its probability distribution.

$$H(X) = -\sum_{x \in X} P(x) \log_2 P(x)$$

$$0 \log_2 0 = 0$$

- Entropy is maximized when all outcomes are **equiprobable**.



Definition: Information gain measures the reduction in entropy (uncertainty) after the dataset is split on a feature.

For a split of feature f that divides the dataset D into two subsets D_L and D_R, the **information gain after the split** is:

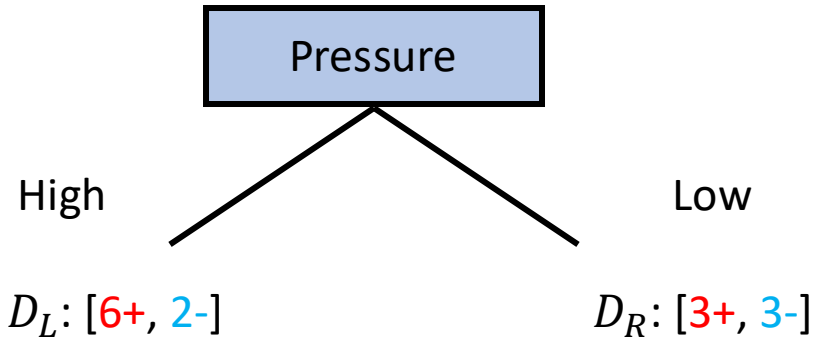
$$\mathrm{IG}_{\mathrm{split}}(D) = H(D) - \left(\frac{|D_{\mathrm{L}}|}{|D|} H(D_{\mathrm{L}}) + \frac{|D_{\mathrm{R}}|}{|D|} H(D_{\mathrm{R}}) \right)$$

The best split is the one that **maximize** $\mathrm{IG}_{\mathrm{split}}$.

Information Gain Example

D: [9+, 5-]

+: Reaction; -: No reaction



$$H(X) = -\sum_{x \in X} P(x) \log_2 P(x)$$

$$\mathrm{IG}_{\mathrm{split}}(D) = H(D) - \left(\frac{|D_{\mathrm{L}}|}{|D|} H(D_{\mathrm{L}}) + \frac{|D_{\mathrm{R}}|}{|D|} H(D_{\mathrm{R}}) \right)$$

$$H(D) = -\frac{9}{14} \log_2 \left(\frac{9}{14} \right) - \frac{5}{14} \log_2 \left(\frac{5}{14} \right) = 0.940$$

$$H(D_{\mathrm{L}}) = -\frac{6}{8} \log_2 \left(\frac{6}{8} \right) - \frac{2}{8} \log_2 \left(\frac{2}{8} \right) = 0.811$$

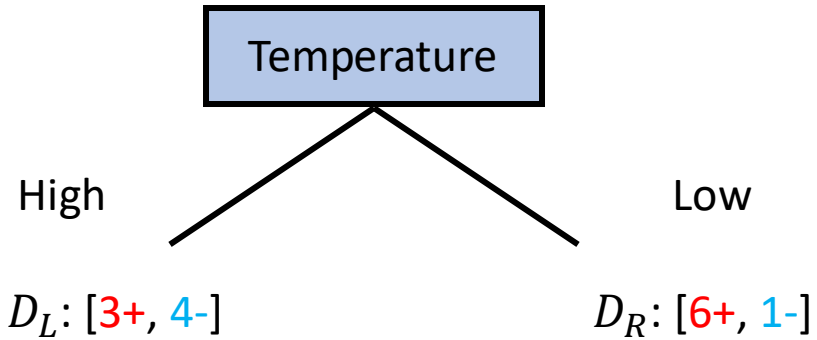
$$H(D_{\mathrm{R}}) = -\frac{3}{6} \log_2 \left(\frac{3}{6} \right) - \frac{3}{6} \log_2 \left(\frac{3}{6} \right) = 1.0$$

$$\mathrm{IG}_{\mathrm{split}}(D) = 0.940 - \left(\frac{8}{14} \cdot 0.811 + \frac{6}{14} \cdot 1.0 \right) = 0.048$$

Information Gain Example

D: [9+, 5-]

+: Reaction; -: No reaction



$$H(X) = -\sum_{x \in X} P(x) \log_2 P(x)$$

$$\mathrm{IG}_{\mathrm{split}}(D) = H(D) - \left(\frac{|D_{\mathrm{L}}|}{|D|} H(D_{\mathrm{L}}) + \frac{|D_{\mathrm{R}}|}{|D|} H(D_{\mathrm{R}}) \right)$$

$$H(D) = -\frac{9}{14} \log_2 \left(\frac{9}{14} \right) - \frac{5}{14} \log_2 \left(\frac{5}{14} \right) = 0.940$$

$$H(D_{\mathrm{L}}) = -\frac{3}{7} \log_2 \left(\frac{3}{7} \right) - \frac{4}{7} \log_2 \left(\frac{4}{7} \right) = 0.985$$

$$H(D_{\mathrm{R}}) = -\frac{6}{7} \log_2 \left(\frac{6}{7} \right) - \frac{1}{7} \log_2 \left(\frac{1}{7} \right) = 0.592$$

$$\mathrm{IG}_{\mathrm{split}}(D) = 0.940 - \left(\frac{7}{14} \cdot 0.985 + \frac{7}{14} \cdot 0.592 \right) = 0.151$$

Is it better to split on pressure or temperature?

Splitting Criterion: Gini Impurity

Definition: Gini impurity measures how often a randomly chosen element from the set would be incorrectly classified if it were randomly labeled according to the distribution of labels in the subset.

For a dataset D with k classes, the **Gini impurity** is

$$\text{Gini}(D) = 1 - \sum_{i=1}^{k} p_i^2$$

- p_i is the proportion of data points in D that belong to class i.
- The Gini impurity for a **perfectly pure node (containing only one class)** is 0, and for a **completely impure node (equal proportion of all classes)**, it is close to 1.

For a split of feature f that divides the dataset D into two subsets D_L and D_R, the **weighted Gini impurity after the split** is:

$$\text{Gini}_{\text{split}}(D) = \frac{|D_{\text{L}}|}{|D|}\text{Gini}(D_{\text{L}}) + \frac{|D_{\text{R}}|}{|D|}\text{Gini}(D_{\text{R}})$$

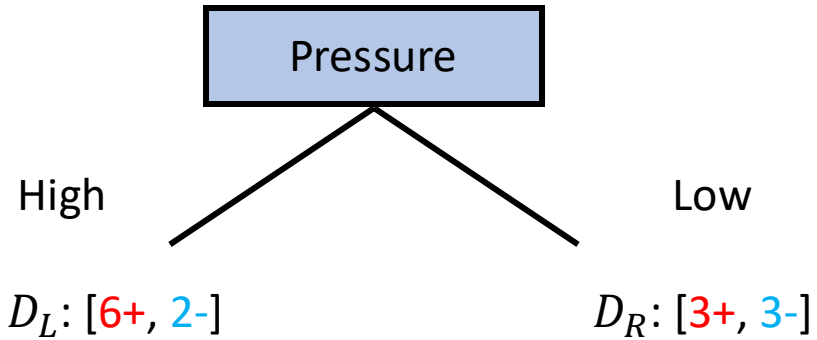
The best split is the one that **minimizes** $\text{Gini}_{\text{split}}$.

https://ml2gifs.substack.com/p/decision-trees

Gini Impurity Example

D: [9+, 5-]

+: Reaction; -: No reaction



$$\text{Gini}(D) = 1 - \sum_{i=1}^{k} p_i^2$$

$$\text{Gini}_{\text{split}}(D) = \frac{|D_{\text{L}}|}{|D|}\text{Gini}(D_{\text{L}}) + \frac{|D_{\text{R}}|}{|D|}\text{Gini}(D_{\text{R}})$$

$$\text{Gini}(D) = 1 - \left(\frac{9}{14}\right)^2 - \left(\frac{5}{14}\right)^2 = 0.459$$

$$\text{Gini}(D_{\text{L}}) = 1 - \left(\frac{6}{8}\right)^2 - \left(\frac{2}{8}\right)^2 = 0.375$$

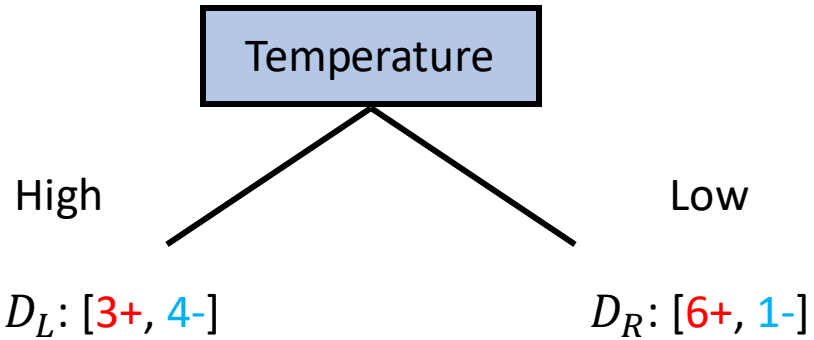
$$\text{Gini}(D_{\text{R}}) = 1 - \left(\frac{3}{6}\right)^2 - \left(\frac{3}{6}\right)^2 = 0.5$$

$$\text{Gini}_{\text{split}}(D) = \frac{8}{14} \cdot 0.375 + \frac{6}{14} \cdot 0.5 = 0.429$$

Gini Impurity Example

D: [9+, 5-]

+: Reaction; -: No reaction



$$\text{Gini}(D) = 1 - \sum_{i=1}^{k} p_i^2$$

$$\text{Gini}_{\text{split}}(D) = \frac{|D_{\text{L}}|}{|D|}\text{Gini}(D_{\text{L}}) + \frac{|D_{\text{R}}|}{|D|}\text{Gini}(D_{\text{R}})$$

$$\text{Gini}(D) = 1 - \left(\frac{9}{14}\right)^2 - \left(\frac{5}{14}\right)^2 = 0.459$$

$$\text{Gini}(D_{\text{L}}) = 1 - \left(\frac{3}{7}\right)^2 - \left(\frac{4}{7}\right)^2 = 0.490$$

$$\text{Gini}(D_{\text{R}}) = 1 - \left(\frac{6}{7}\right)^2 - \left(\frac{1}{7}\right)^2 = 0.245$$

$$\text{Gini}_{\text{split}}(D) = \frac{7}{14} \cdot 0.490 + \frac{7}{14} \cdot 0.245 = 0.368$$

Is it better to split on pressure or temperature?

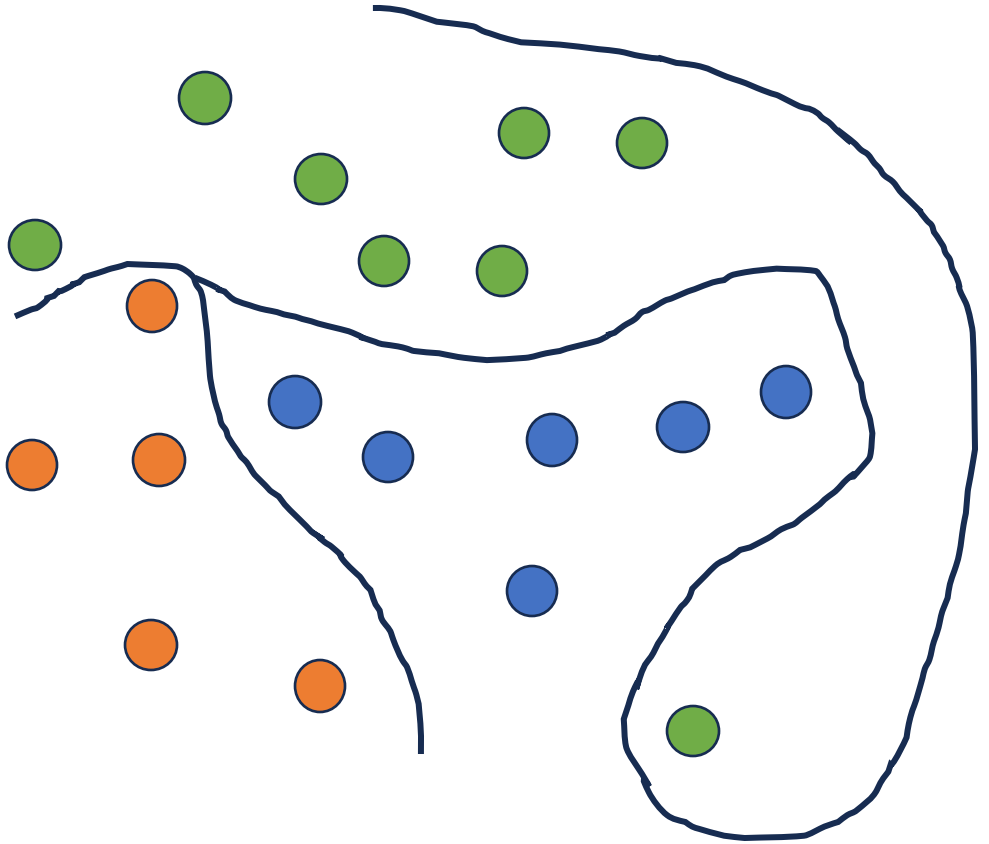
Stopping Criteria

Common stopping criteria:

- Maximum depth
- Maximum number of leaf nodes
- Minimum samples per leaf
- Minimum samples per split
- Minimum information gain / impurity reduction

Why do we need stopping criteria?

Stopping criteria in decision trees are essential to prevent the tree from becoming overly complex, which can lead to **overfitting**.



How many features are used to split at each node?

Is the training of a decision tree done in a "single pass" through the data, or does it involve "multiple passes" like gradient descent?

Decision Tree Hyperparameters

sklearn.tree.DecisionTreeClassifier

class `sklearn.tree.`**`DecisionTreeClassifier`**(**, criterion='gini', splitter='best', max_depth=None, min_samples_split=2, min_samples_leaf=1, min_weight_fraction_leaf=0.0, max_features=None, random_state=None, max_leaf_nodes=None, min_impurity_decrease=0.0, class_weight=None, ccp_alpha=0.0*) [source]

max_depth : *int, default=None*
The maximum depth of the tree. If None, then nodes are expanded until all leaves are pure or until all leaves contain less than min_samples_split samples.

min_samples_split : *int or float, default=2*
The minimum number of samples required to split an internal node:

- If int, then consider `min_samples_split` as the minimum number.
- If float, then `min_samples_split` is a fraction and `ceil(min_samples_split * n_samples)` are the minimum number of samples for each split.

High max_depth: Leads to a larger, more complex tree, increasing the risk of overfitting.
Low max_depth: Results in a simpler model, which may lead to underfitting.

High min_samples_split: Produces a simpler, shallower tree with a lower risk of overfitting, but may be too restrictive.
Low min_samples_split: Allows more frequent splits, potentially leading to overfitting by creating a more complex tree.

Decision Tree Hyperparameters

sklearn.tree.DecisionTreeClassifier

class `sklearn.tree.`**`DecisionTreeClassifier`**(**, criterion='gini', splitter='best', max_depth=None, min_samples_split=2, min_samples_leaf=1, min_weight_fraction_leaf=0.0, max_features=None, random_state=None, max_leaf_nodes=None, min_impurity_decrease=0.0, class_weight=None, ccp_alpha=0.0*) [source]

max_features : *int, float or {"auto", "sqrt", "log2"}, default=None*

The number of features to consider when looking for the best split:

- If int, then consider `max_features` features at each split.
- If float, then `max_features` is a fraction and `max(1, int(max_features * n_features_in_))` features are considered at each split.
- If "sqrt", then `max_features=sqrt(n_features)`.
- If "log2", then `max_features=log2(n_features)`.
- If None, then `max_features=n_features`.

Note: the search for a split does not stop until at least one valid partition of the node samples is found, even if it requires to effectively inspect more than `max_features` features.

random_state : *int, RandomState instance or None, default=None*

Controls the randomness of the estimator. The features are always randomly permuted at each split, even if `splitter` is set to `"best"`. When `max_features < n_features`, the algorithm will select `max_features` at random at each split before finding the best split among them. But the best found split may vary across different runs, even if `max_features=n_features`. That is the case, if the improvement of the criterion is identical for several splits and one split has to be selected at random. To obtain a deterministic behaviour during fitting, `random_state` has to be fixed to an integer. See Glossary for details.

High max_features: More features are considered at each split. May increase the risk of overfitting.

Low max_features: Reduces the chance of overfitting.

Fix a random state for reproducibility.

Decision Tree Notebook Example

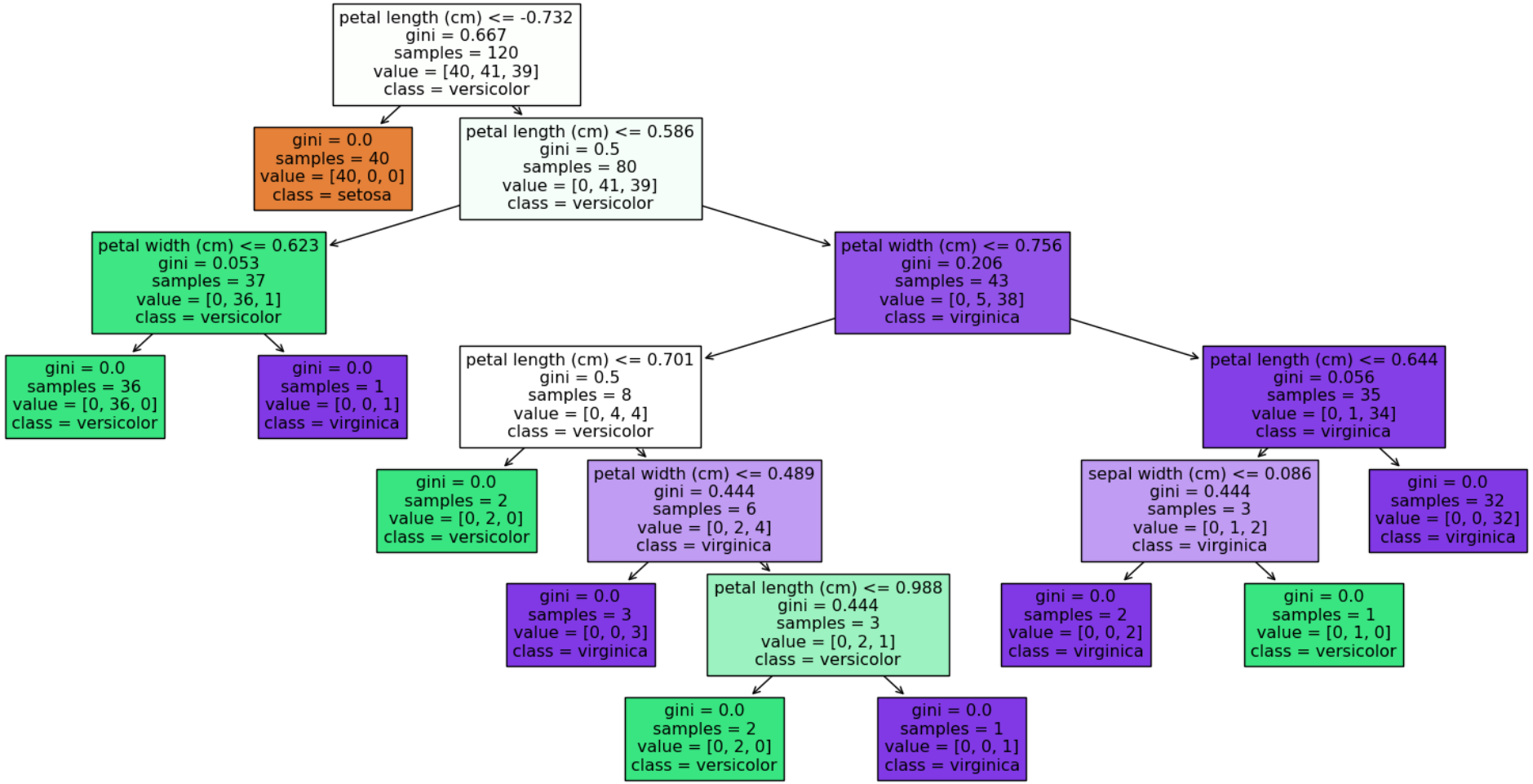
3. Simple Decision Tree

```
iris = datasets.load_iris()
X = iris.data
y = iris.target

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

```
[98] # train the decision tree classifier
     tree_classifier = DecisionTreeClassifier(criterion="gini", random_state=42)
     tree_classifier.fit(X_train, y_train)
     y_pred = tree_classifier.predict(X_test)

     acc = accuracy_score(y_test, y_pred)
     print(f"Accuracy: {acc*100:0.2f}%")
```

```
Accuracy: 100.00%
```

Decision Tree Visualization

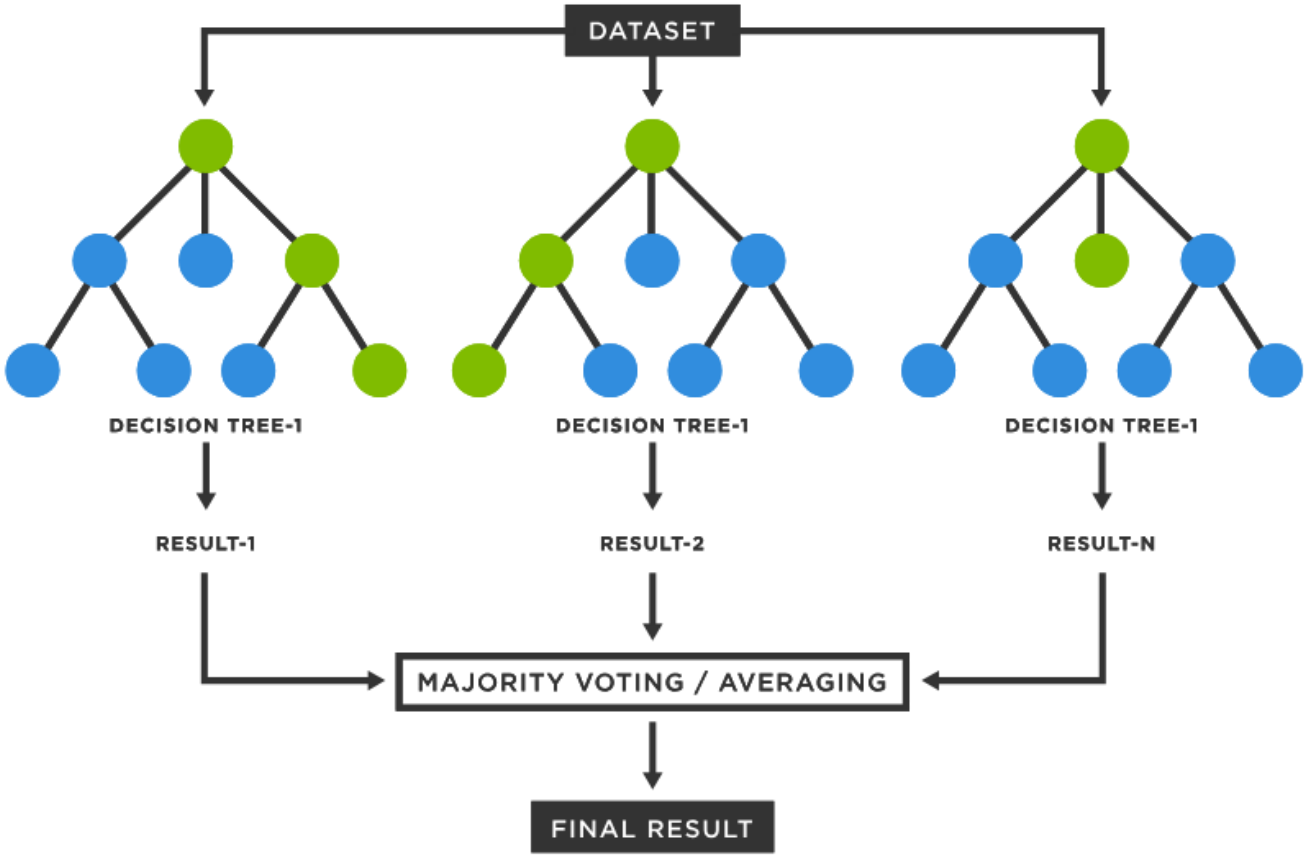


Random Forest Classifier

Can weak models be combined to create a more accurate predictor?

Definition: Random forest classifier is a supervised learning classification algorithm that belongs to the family of **ensemble methods.**

It operates by constructing **multiple decision trees** during training and outputs the **mode** of the classes.



When to use random forest?

1. Handles both categorical and numerical data.
2. When interpretability is important.
3. Mitigate overfitting.

https://medium.com/@denizgunay/random-forest-af5bde5d7e1e

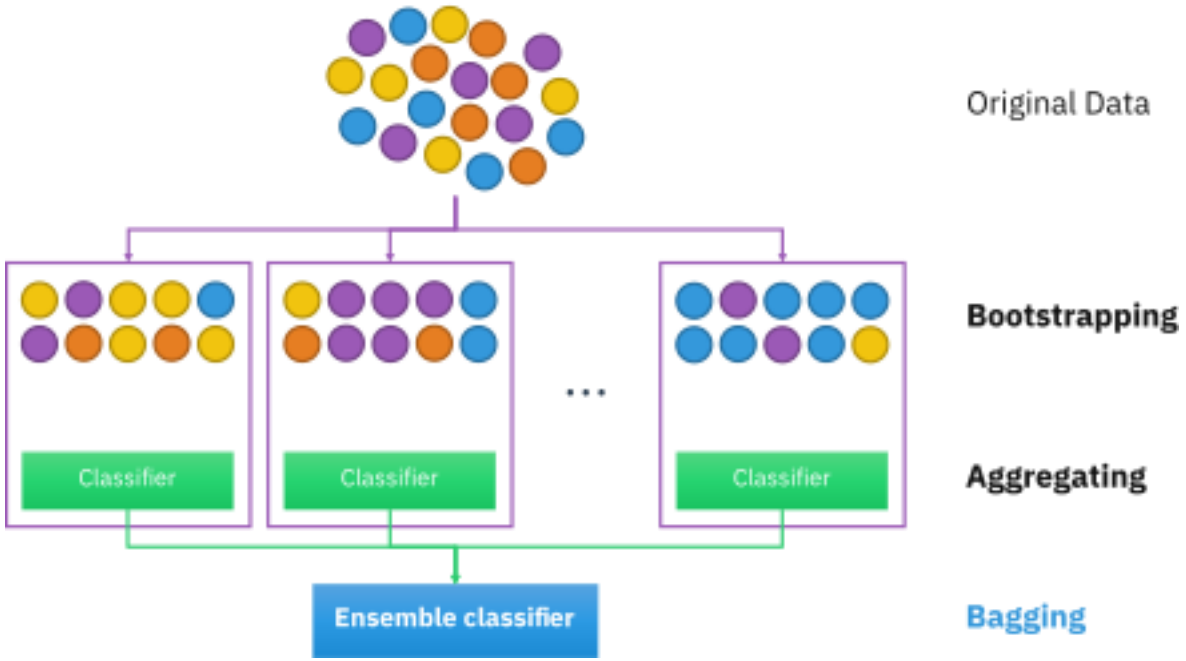
Bootstrapping

Definition: Bootstrapping is a resampling method where a dataset is sampled **with replacement** to create multiple subsets (bootstrap samples).

In bootstrapping, the sample size is the same as the original dataset. The same data point can appear multiple times.

About 63% of the data is included in each sample, while the remaining 37% is known as Out-of-Bag (OOB) data.

$$\lim_{n\to\infty}\left(1-\frac{1}{n}\right)^{n}=1/e\approx 0.37$$



https://en.wikipedia.org/wiki/Bootstrap_aggregating

Random Forest Hyperparameters

sklearn.ensemble.RandomForestClassifier

class `sklearn.ensemble.`**`RandomForestClassifier`**(*n_estimators=100, *, criterion='gini', max_depth=None, min_samples_split=2, min_samples_leaf=1, min_weight_fraction_leaf=0.0, max_features='sqrt', max_leaf_nodes=None, min_impurity_decrease=0.0, bootstrap=True, oob_score=False, n_jobs=None, random_state=None, verbose=0, warm_start=False, class_weight=None, ccp_alpha=0.0, max_samples=None*) [source]

Parameters:

n_estimators : *int, default=100*

The number of trees in the forest.

Changed in version 0.22: The default value of `n_estimators` changed from 10 to 100 in 0.22.

criterion : *{"gini", "entropy", "log_loss"}, default="gini"*

The function to measure the quality of a split. Supported criteria are "gini" for the Gini impurity and "log_loss" and "entropy" both for the Shannon information gain, see Mathematical formulation. Note: This parameter is tree-specific.

High n_estimators: May lead to diminishing returns beyond a certain point, potentially increasing overfitting. Results in higher computational cost and longer training times.

Low n_estimators: Increases the risk of underfitting and higher model variance but offers faster computation.

Random Forest Notebook Example

1. Simple Random Forest

```
[107] iris = datasets.load_iris()
      X = iris.data
      y = iris.target

      X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

      scaler = StandardScaler()
      X_train = scaler.fit_transform(X_train)
      X_test = scaler.transform(X_test)
```

```
[108] # train the random forest classifier
      forest_clf = RandomForestClassifier(n_estimators=100, criterion="gini", random_state=42)
      forest_clf.fit(X_train, y_train)
      y_pred = forest_clf.predict(X_test)

      acc = accuracy_score(y_test, y_pred)
      print(f"Accuracy: {acc*100:0.2f}%")
```

```
Accuracy: 100.00%
```

Random Forest Notebook Example

2. Feature Importance

```
[110] importances = forest_clf.feature_importances_

      for feature, importance in zip(iris.feature_names, importances):
          print(f"{feature}: {importance:0.4f}")
```

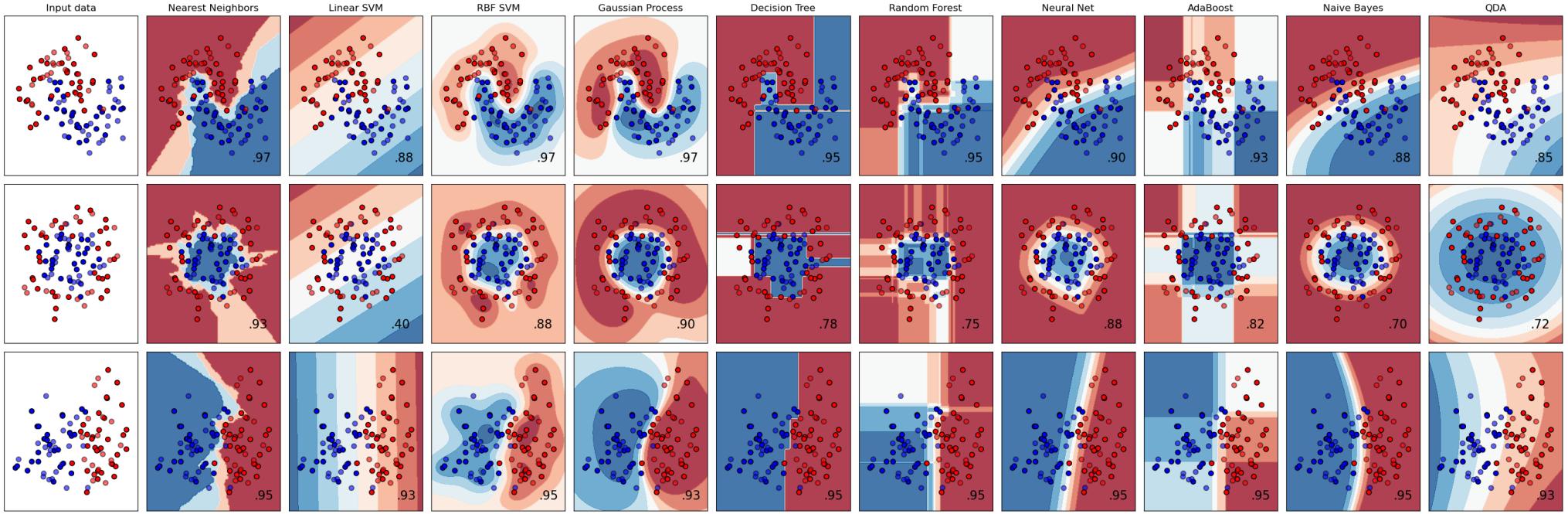
```
sepal length (cm): 0.1081
sepal width (cm): 0.0304
petal length (cm): 0.4400
petal width (cm): 0.4215
```

Mean decrease in impurity

$$\text{Importance}(f) = \frac{1}{T} \sum_{t=1}^{T} \sum_{n \in N_t} I(f,n) \Delta i_n$$

- T is the total number of trees in the forest.
- N_t is the set of nodes in tree t.
- $I(f,n)$ is an indicator function that equals 1 if feature f was used to split node n, and 0 otherwise.
- Δi_n is the reduction in impurity (Gini impurity or entropy) caused by the split at node n.

Other Classification Methods



K-Nearest neighbors: non-parametric, instance-based algorithm that classifies a data point by voting from its "k" nearest neighbors in the feature space.

Gradient boosting machine: an ensemble method that builds multiple weak learners (decision trees), each correcting the errors of the previous ones. (XGBoost)

Always start with a simple model (Occam's Razor) before moving to complex options like neural networks.