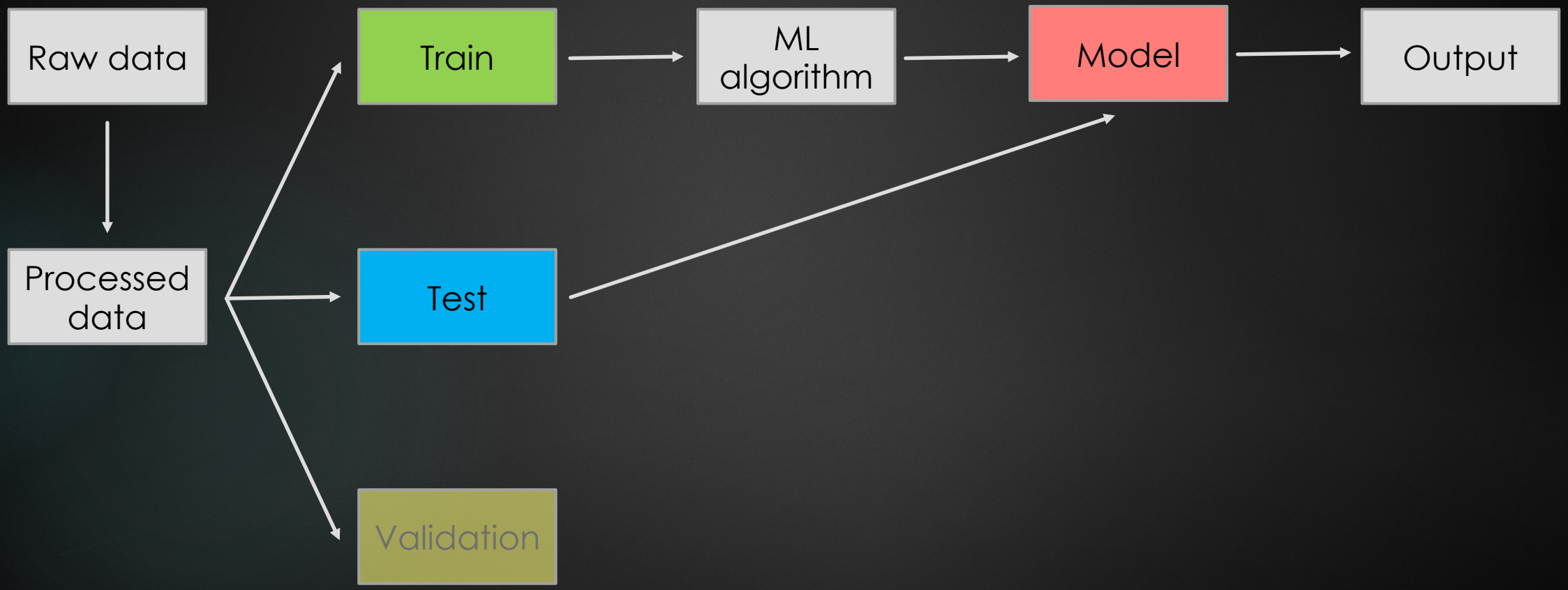
Universitat de Girona

Machine learning algorithms

GUILLEM GUIGÓ I COROMINAS

Managing the data

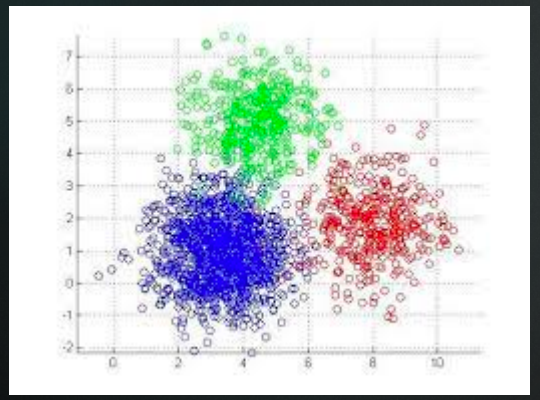


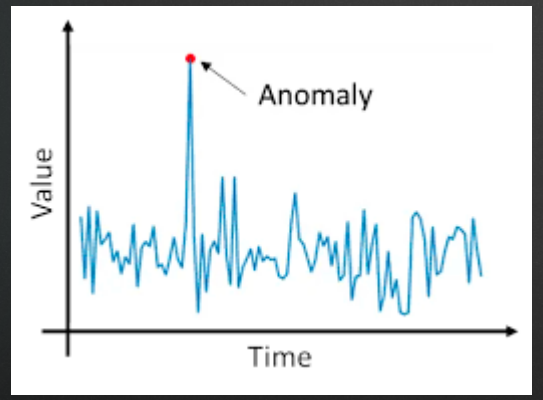
Supervised vs unsupervised learning

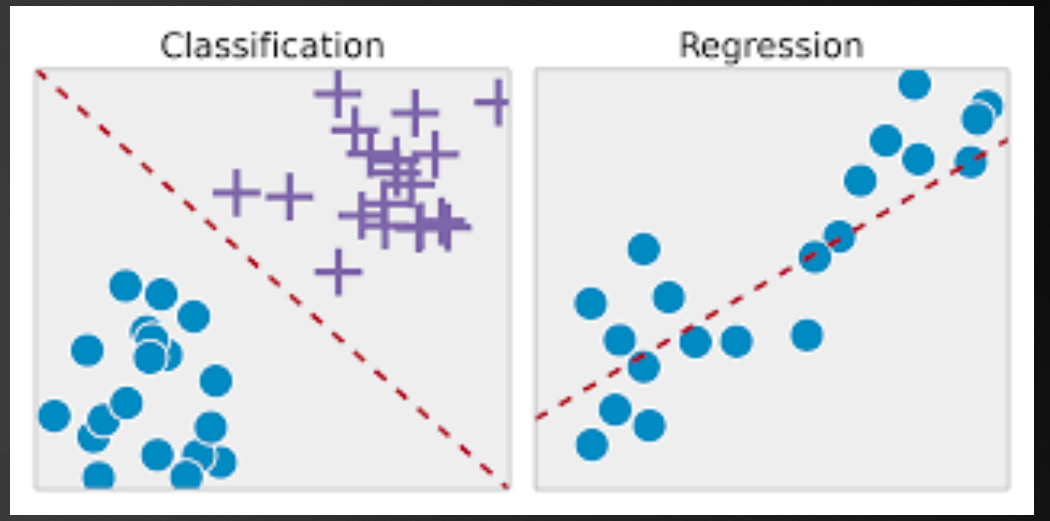
- Supervised: We use labels for our target -> Classification / regression
- Unsupervised: No labels ->

Dimensionality reduction, clustering,

Anomaly detection



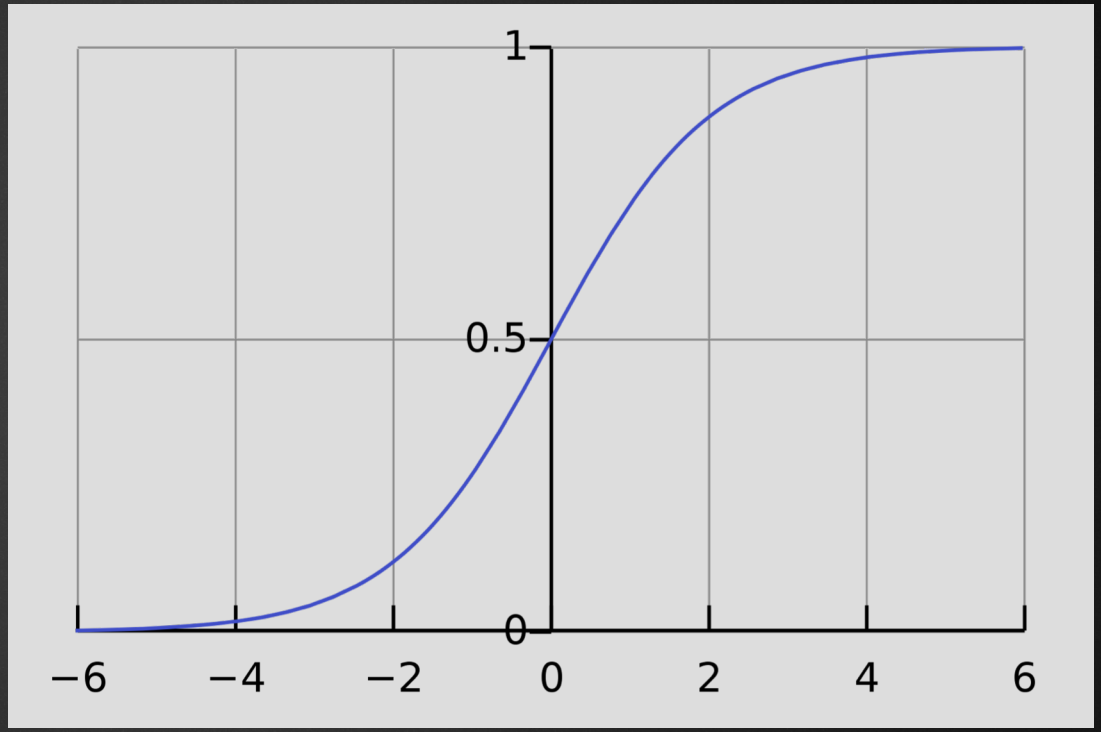




Classification example: Logistic regression

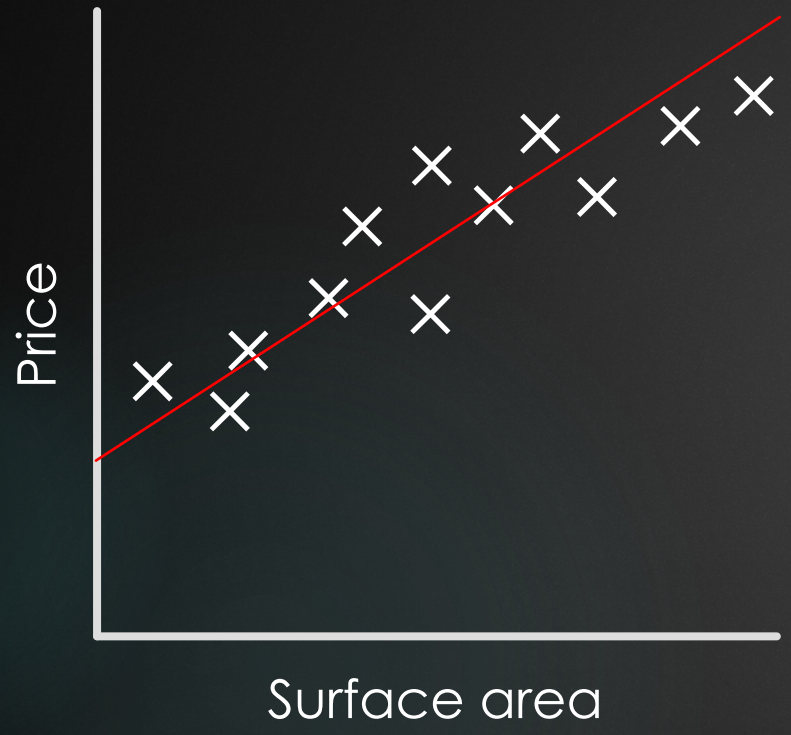
- Logistic function:

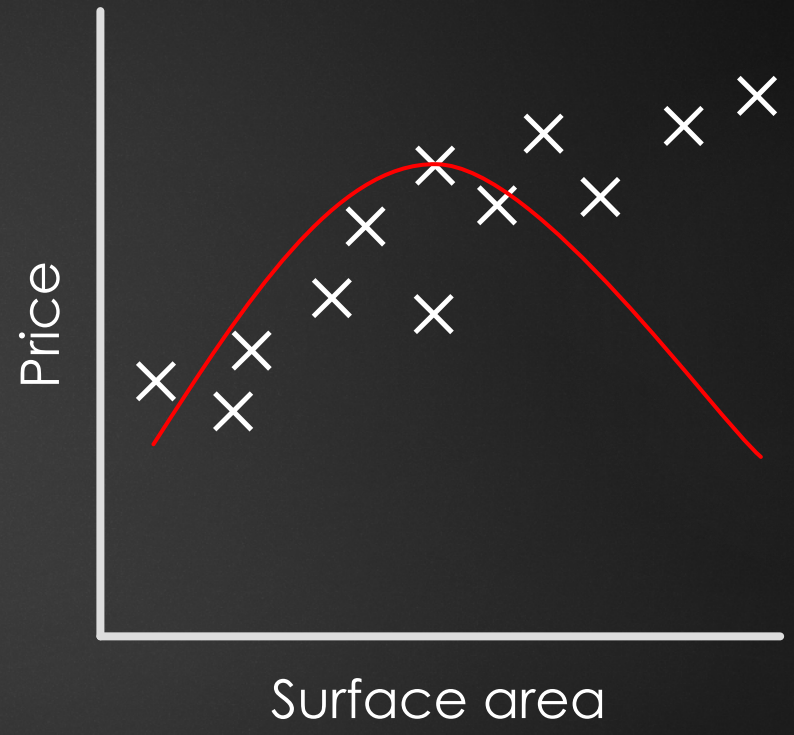
$$f(x) = \frac{1}{1 + e^{-x}}$$



- Output is a probability
- Used for classification

Bias-variance tradeoff

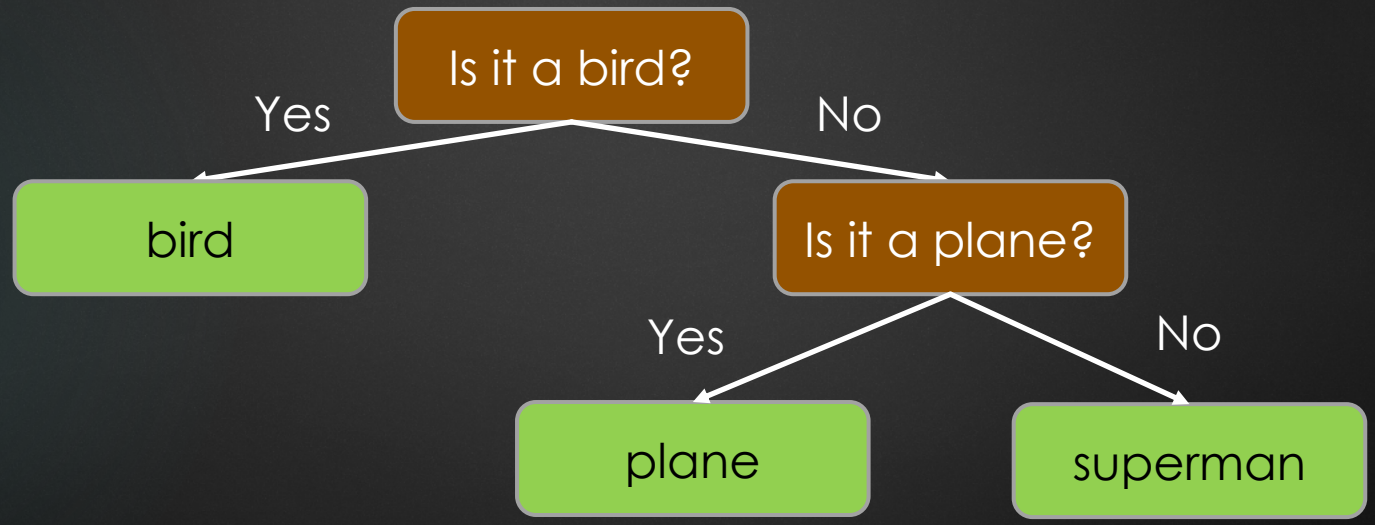




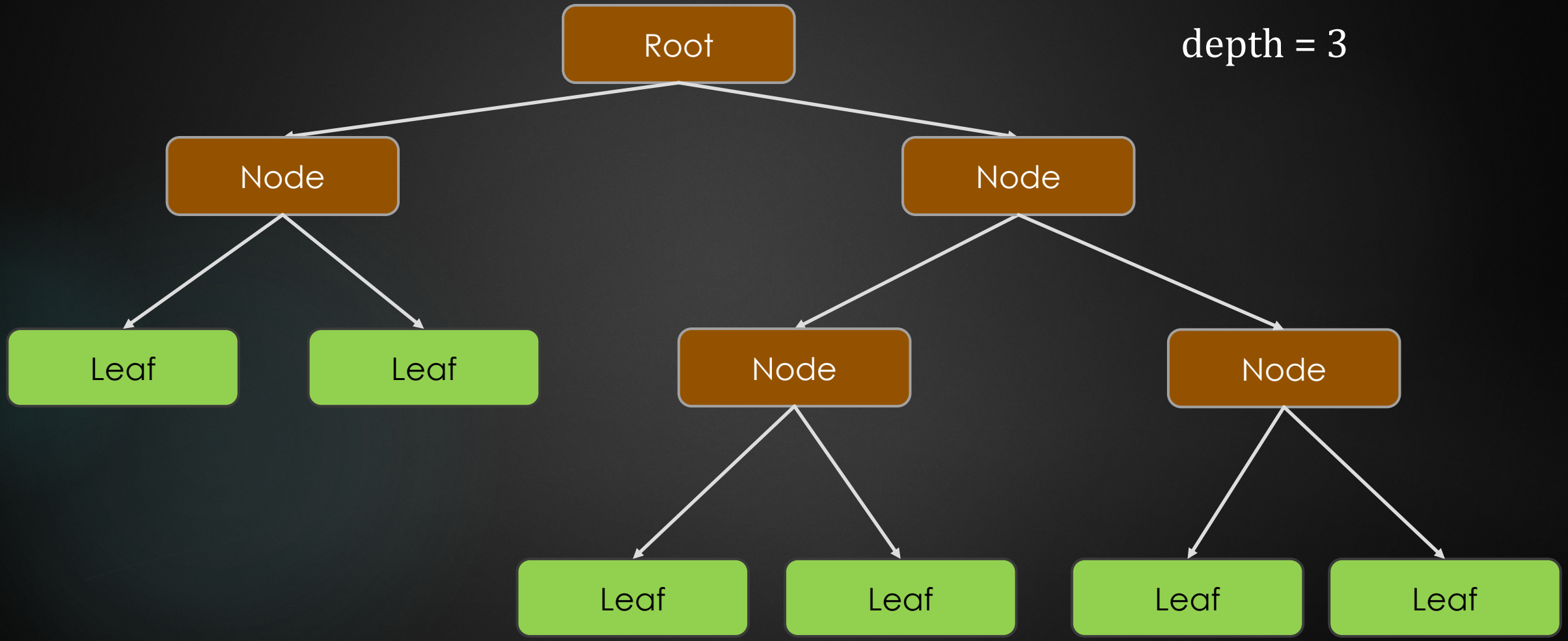
In statistics, the **bias of an estimator** (or **bias function**) is the difference between this estimator's expected value and the true value of the parameter being estimated. An estimator or decision rule with zero bias is called ***unbiased***. In statistics, "bias" is an *objective* property of an estimator

Decision trees

- A decision tree is a decision support hierarchical model that uses a tree-like model of decisions and their possible consequences.
- "*If* condition 1 *and* condition 2 *then* outcome"



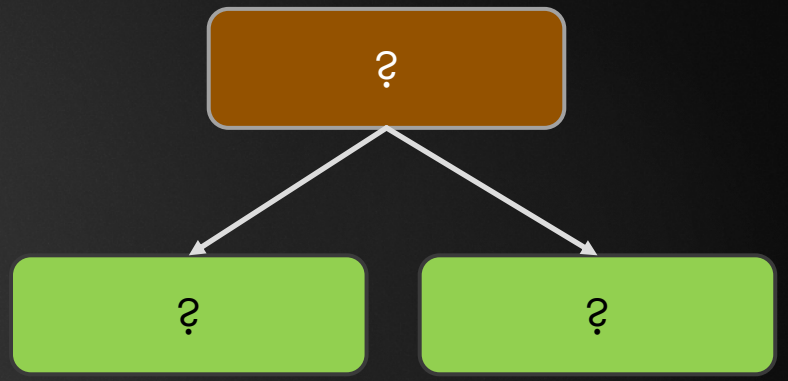
Decision trees



Decision tree learning

Tumor cells

Mean radius	Mean smoothness	Mean concavity	M or B?
14.97	0.08421	0.01947	0
15.7	0.09597	0.06593	1
13.48	0.1016	0.1063	1
14.5	0.1101	0.08842	0
17.46	0.09812	0.1417	1
12.47	0.09965	0.08005	0
11.89	0.1225	0.05929	0
13.61	0.09488	0.08625	1
14.6	0.08682	0.0839	1
11.26	0.0802	0.09274	0



Gini impurity

$$b_i = 1 - p_0^2 - p_1^2$$

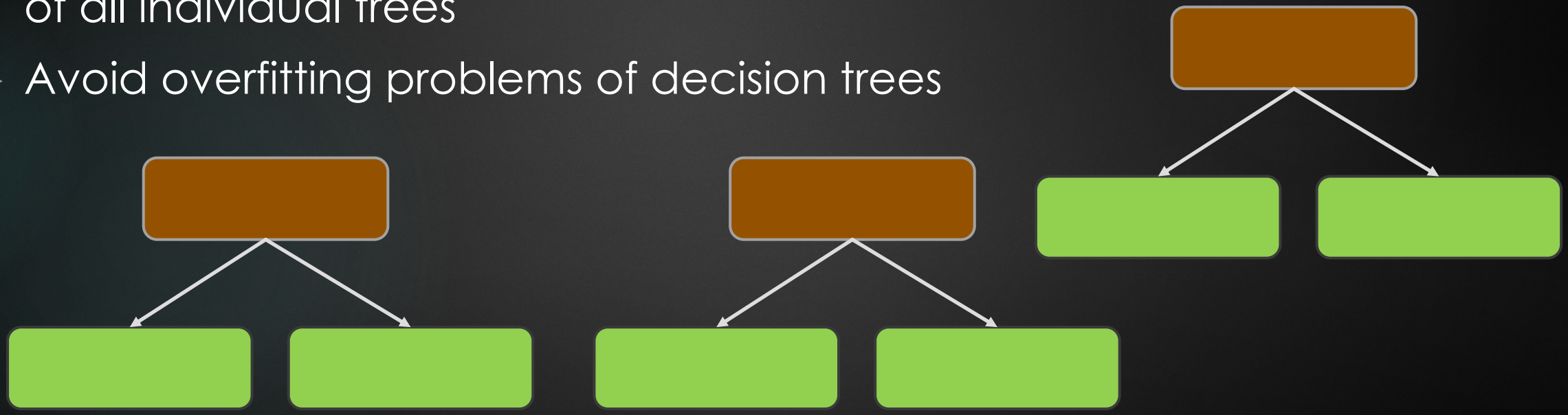
$$Gini(N) = \alpha b_1 + \beta b_2$$

Decision tree learning

- Easy to build, use and interpret
- Require little data preparation
- Perform well on large datasets
- Tend to overfit training data
- Generalization problems
- Limit the number of elements on each leaf
- Prune the trees

Random forests

- Ensemble learning method for classification and regression
- For classification tasks, the output is the majority voting of trees
- For regression tasks, the output is the mean of the prediction of all individual trees
- Avoid overfitting problems of decision trees



Random forests

- Create a bootstrapped dataset of the same size as the original dataset by selecting random samples from the dataset
- Build *n* decision trees, selecting at each step *m* variables
- Evaluate the random forest with the samples that didn't make it to the bootstrapped dataset
- Repeat with a different *m*

Random forest

- In practice…

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]

Scikit-learn: Machine Learning in Python, Pedregosa *et al.*, JMLR 12, pp. 2825-2830, 2011.

Random forest

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.

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.

Random forest

min_samples_leaf : *int or float, default=1*

The minimum number of samples required to be at a leaf node. A split point at any depth will only be considered if it leaves at least `min_samples_leaf` training samples in each of the left and right branches. This may have the effect of smoothing the model, especially in regression.

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

Changed in version 0.18: Added float values for fractions.

min_weight_fraction_leaf : *float, default=0.0*

The minimum weighted fraction of the sum total of weights (of all the input samples) required to be at a leaf node. Samples have equal weight when sample_weight is not provided.

Random forest

max_features : *{“sqrt”, “log2”, None}, int or float, default=”sqrt”*

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 “auto”, then `max_features=sqrt(n_features)`.
- If “sqrt”, then `max_features=sqrt(n_features)`.
- If “log2”, then `max_features=log2(n_features)`.
- If None, then `max_features=n_features`.

Random forest

max_leaf_nodes : *int, default=None*

Grow trees with `max_leaf_nodes` in best-first fashion. Best nodes are defined as relative reduction in impurity. If None then unlimited number of leaf nodes.

min_impurity_decrease : *float, default=0.0*

A node will be split if this split induces a decrease of the impurity greater than or equal to this value.

The weighted impurity decrease equation is the following:

```
N_t / N * (impurity - N_t_R / N_t * right_impurity
                    - N_t_L / N_t * left_impurity)
```

where `N` is the total number of samples, `N_t` is the number of samples at the current node, `N_t_L` is the number of samples in the left child, and `N_t_R` is the number of samples in the right child.

`N`, `N_t`, `N_t_R` and `N_t_L` all refer to the weighted sum, if `sample_weight` is passed.

New in version 0.19.

Random forest

bootstrap : *bool, default=True*

Whether bootstrap samples are used when building trees. If False, the whole dataset is used to build each tree.

oob_score : *bool, default=False*

Whether to use out-of-bag samples to estimate the generalization score. Only available if bootstrap=True.

n_jobs : *int, default=None*

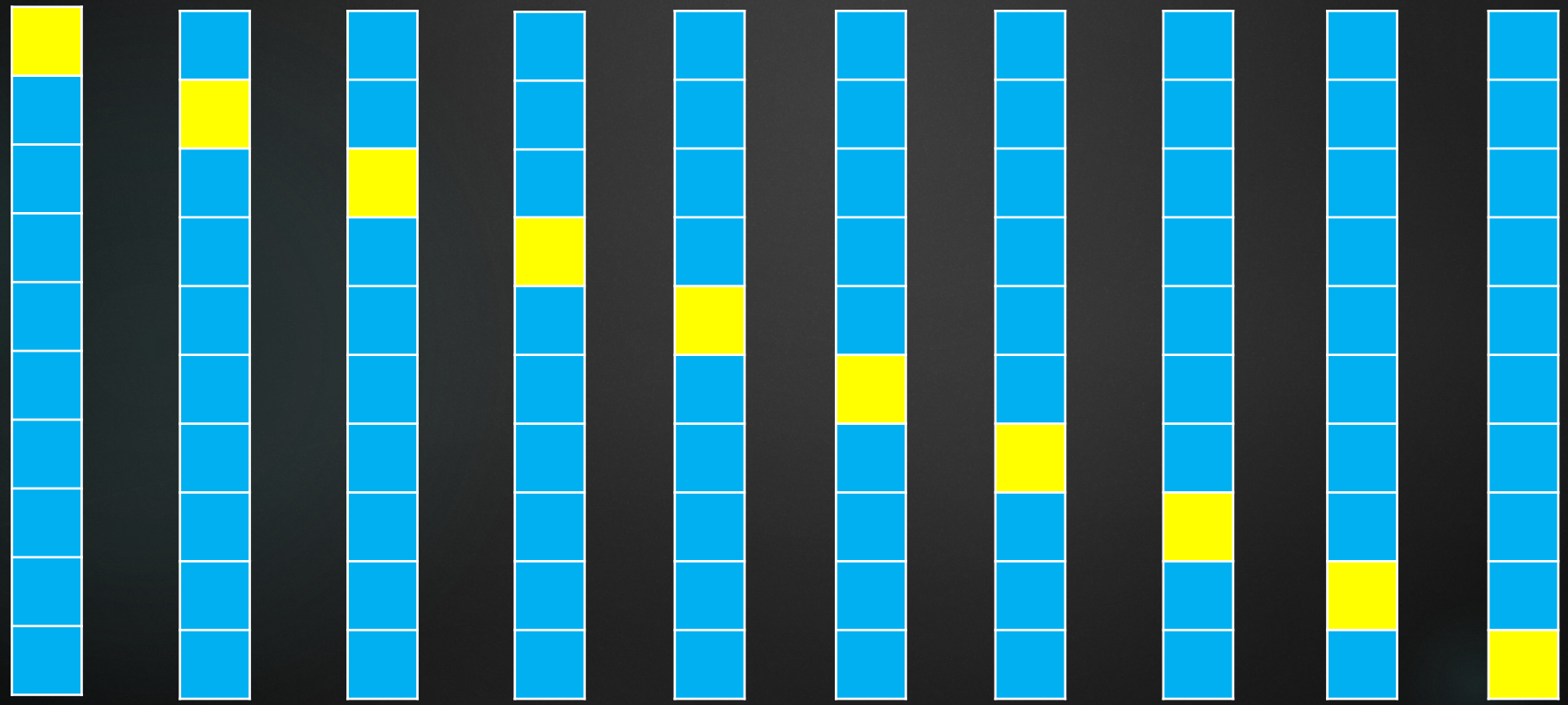
The number of jobs to run in parallel. `fit`, `predict`, `decision_path` and `apply` are all parallelized over the trees. `None` means 1 unless in a `joblib.parallel_backend` context. `-1` means using all processors. See Glossary for more details.

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

Controls both the randomness of the bootstrapping of the samples used when building trees (if `bootstrap=True`) and the sampling of the features to consider when looking for the best split at each node (if `max_features < n_features`). See Glossary for details.

Cross validation

- Optimize hyperparameters: n trees, max features, min samples leaf...
- k-fold cross validation
- k = representative sample, k = 10, leave-one-out



Images sources

- Classification vs regression: https://towardsdatascience.com/supervised-vs-unsupervised-learning-14f68e32ea8d
- Clustering: https://www.kdnuggets.com/2019/09/hierarchical-clustering.html
- Anomaly detection: https://www.analyticsvidhya.com/blog/2021/06/univariate-anomaly-detection-a-walkthrough-in-python/
- Sigmoid function: https://en.wikipedia.org/wiki/Sigmoid_function#/media/File:Logistic-curve.svg