Algorithm vs Model

Algorithm

- Definition: mathematical technique or equation (that is, a framework)

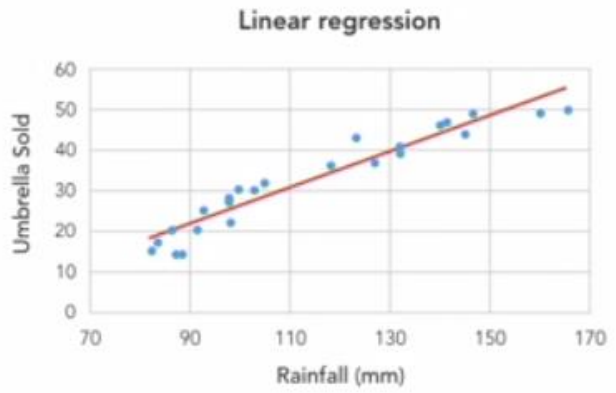
$$y = mx + b$$

- Type: linear regression
- Parameters: m & b

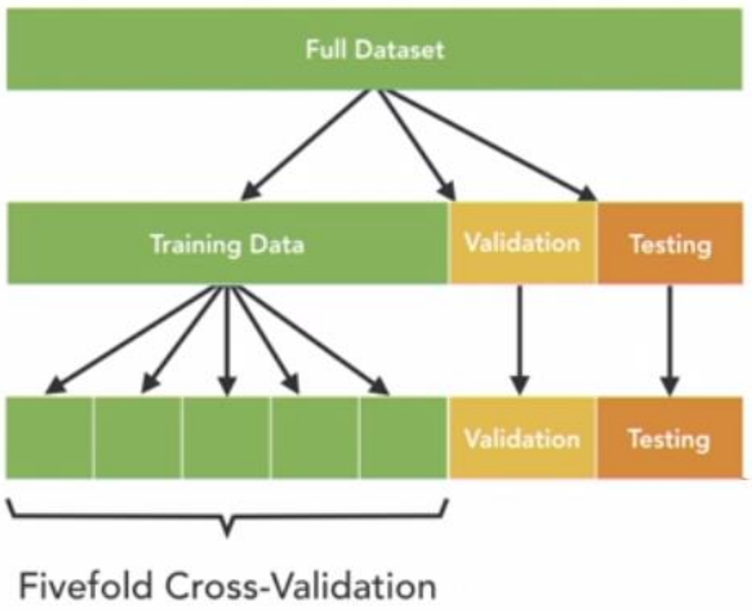
Model

- Definition: equation that is formed by using data to find the parameters in the equation of an algorithm

$$y = 0.45x - 19$$

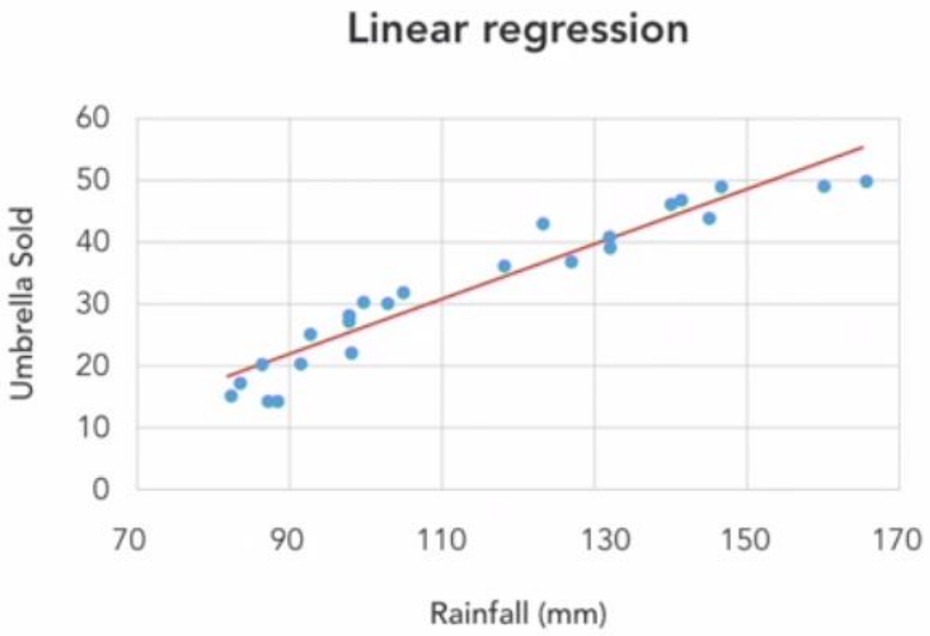


Process



1. Explore and clean the data.
2. Split data into train/validation/test.
3. Fit an initial model and evaluate.
4. Tune hyperparameters.
5. Evaluate on validation set.
6. Select and evaluate the final model on test set.

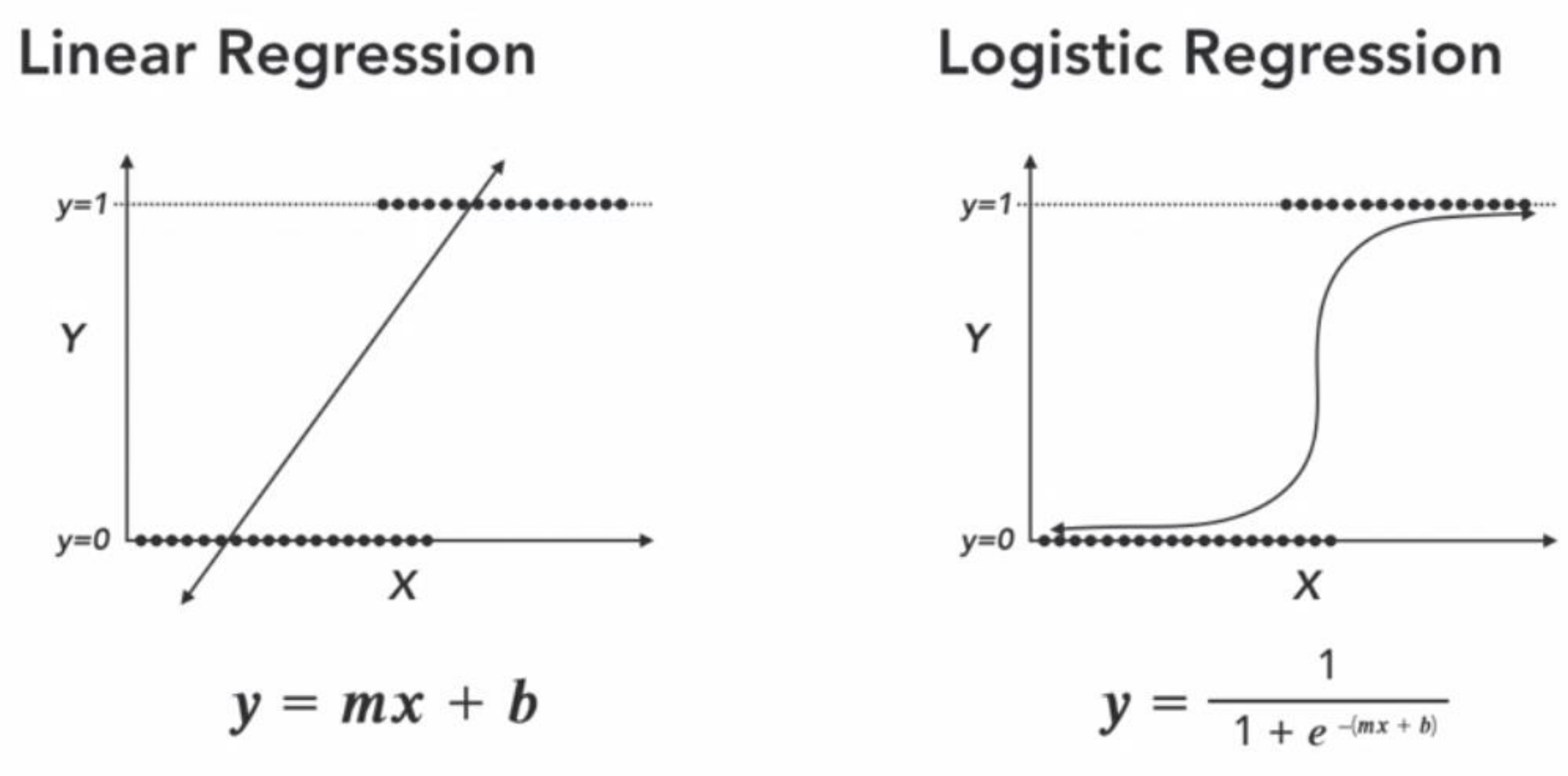
$$y = mx + b$$



Regression is a statistical process for estimating the relationships among variables, often to make a prediction about some outcome.

Logistic regression is a form of regression where the target variable is binary.

Building a Model with Binary Target



Logistic Regression

When to Use It?

- Binary target variable
- Transparency is important or interested in significance of predictors
- Fairly well-behaved data
- Need a quick initial benchmark

When Not to Use It?

- Continuous target variable
- Massive data (rows or columns)
- Unwieldy data
- Performance is the only thing that matters

The **C hyperparameter** is a regularization parameter in logistic regression that controls how closely the model fits to the training data.

Regularization is a technique used to reduce overfitting by discouraging overly complex models in some way.

Impact of C on a Model

$$C = \frac{1}{\lambda}$$

$\lambda \rightarrow 0 = C \rightarrow \infty$ ⟶ Low regularization, High complexity, More likely to overfit

Low regularization

High complexity

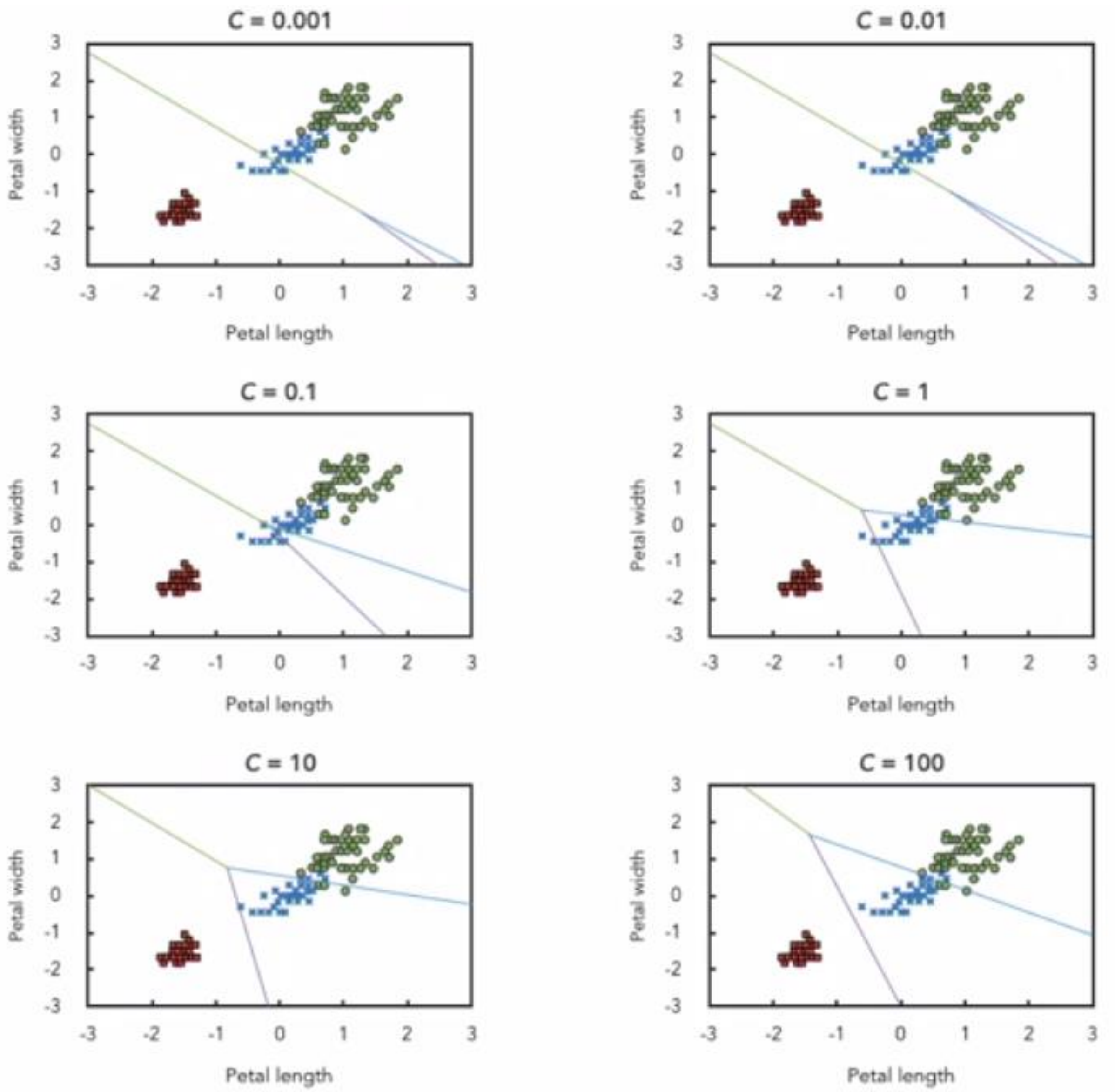
More likely to overfit

$\lambda \rightarrow \infty = C \rightarrow 0$ ⟶

High regularization

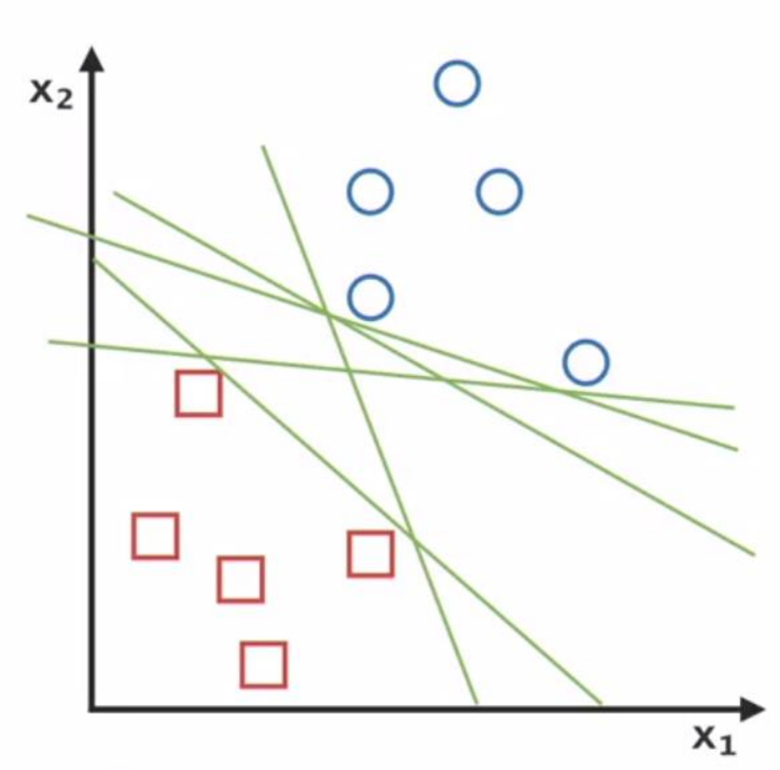
Low complexity

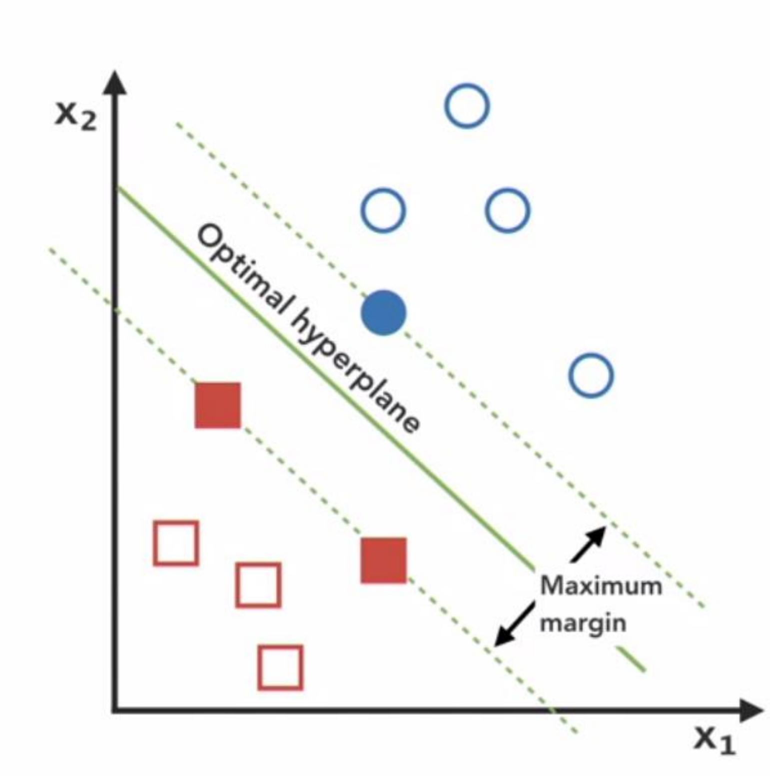
More likely to underfit


C = 0.001
C = 0.01
C = 0.1
C = 1
C = 10
C = 100
Petal width
Petal length
3
2
1
0
-1
-2
-3

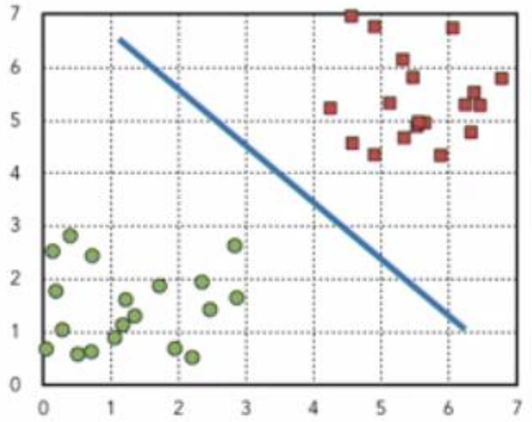
Support Vector Machines

A **support vector machine** is a classifier that finds an optimal hyperplane that maximizes the margin between two classes.

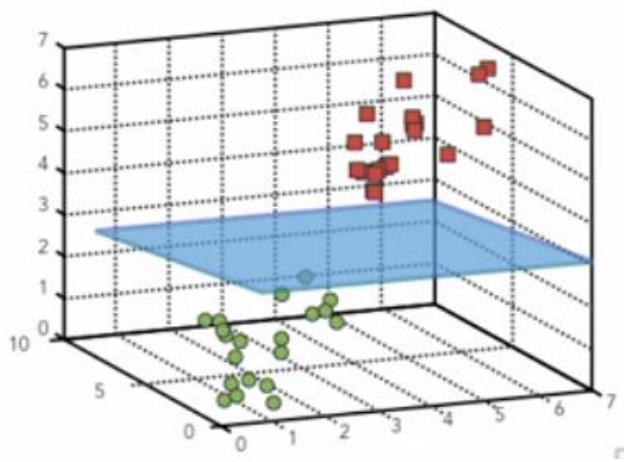




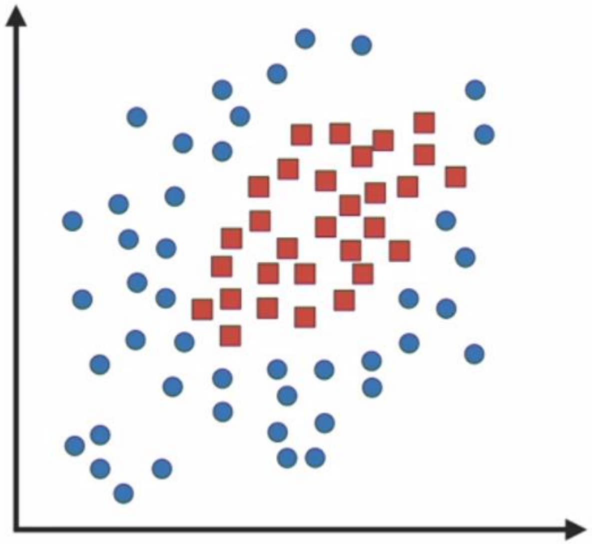
A hyperplane in R^2 is a line

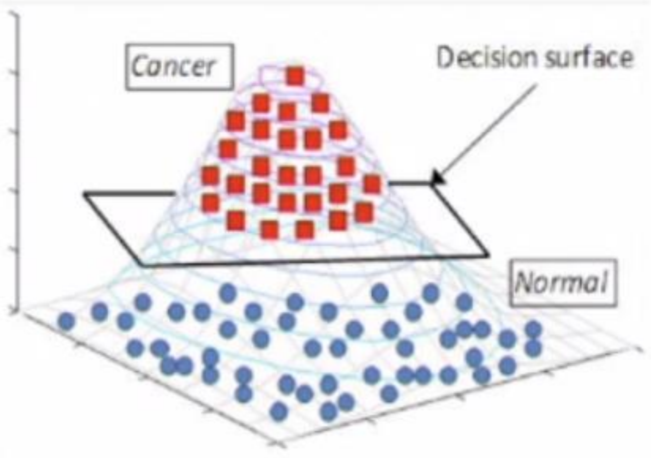


A hyperplane in R^3 is a plane



The **kernel trick** (or kernel method) transforms data that is not linearly separable in n-dimensional space to a higher dimension where it is linearly separable.





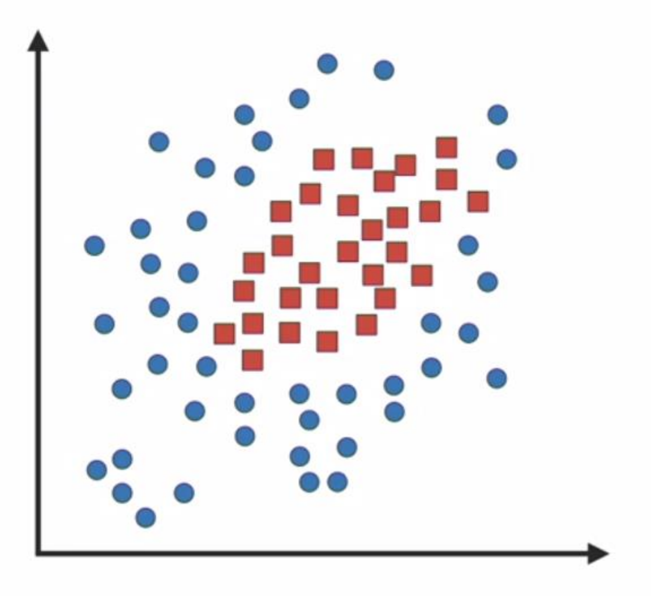
When to Use It?

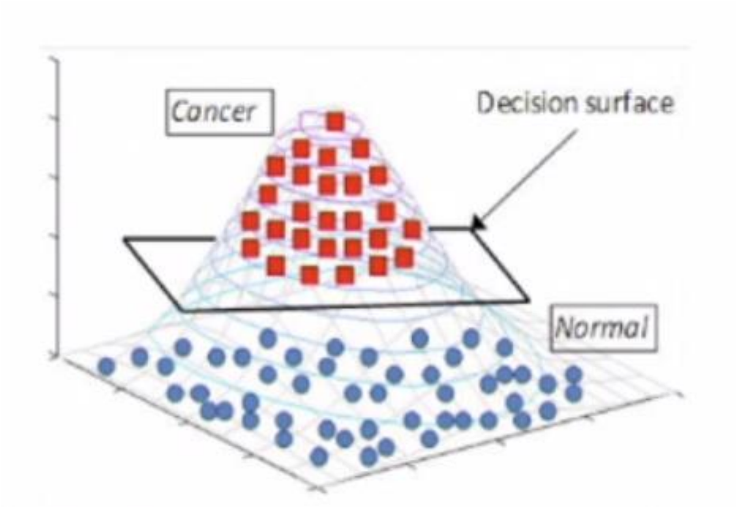
- Binary target variable
- Feature-to-row ratio is very high
- Very complex relationships
- Lots of outliers

When Not to Use It?

- Feature-to-row ratio is very low
- Transparency is important or interested in significance of predictors
- Looking for a quick benchmark model

The **kernel trick** (or kernel method) transforms data that is not linearly separable in n-dimensional space to a higher dimension where it is linearly separable.

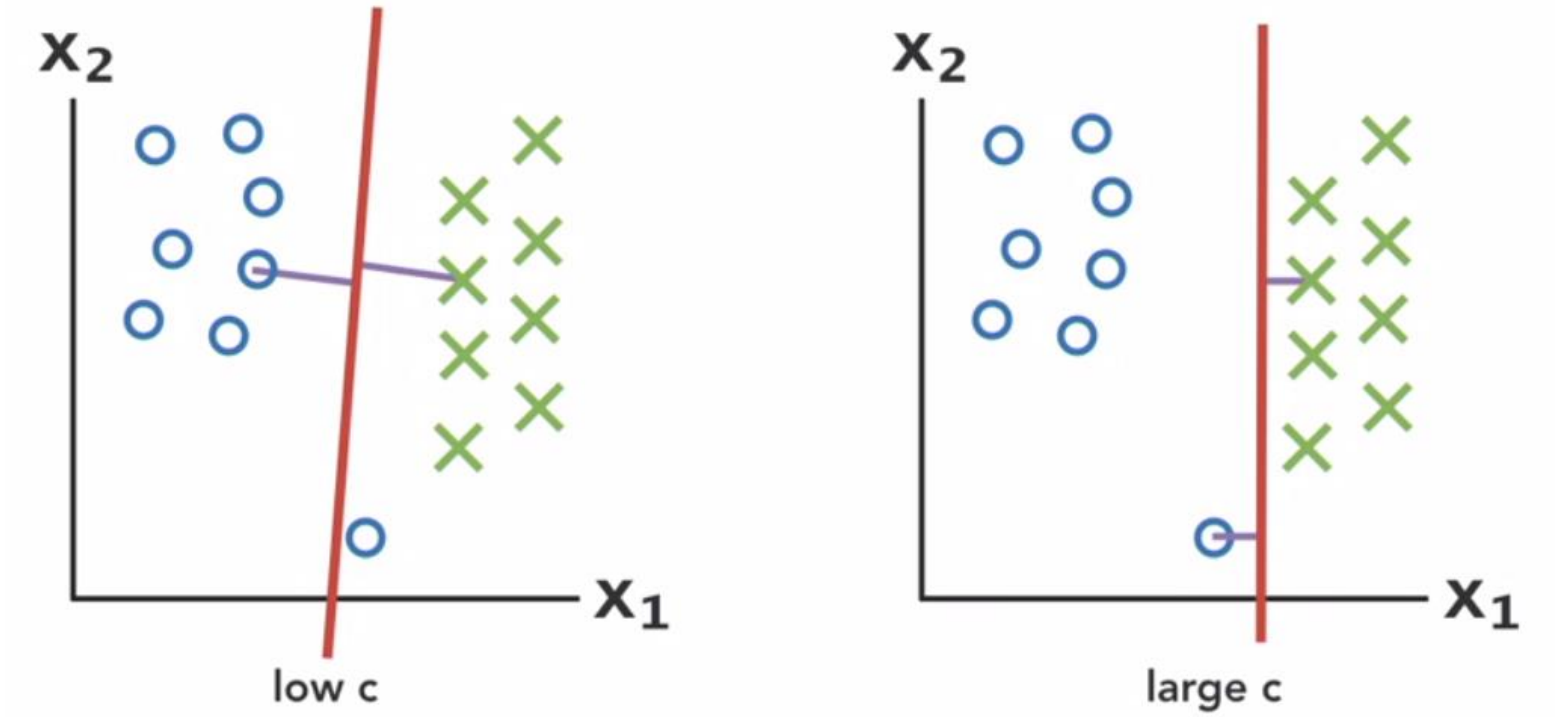




The **C hyperparameter** is a penalty term that determines how closely the model fits to the training set.

Impact of C on a Model

$C \to \infty$	$C \to 0$
↓	↓
Large penalty for misclassification in training	Small penalty for misclassification in training
↓	↓
Small margin	Large margin

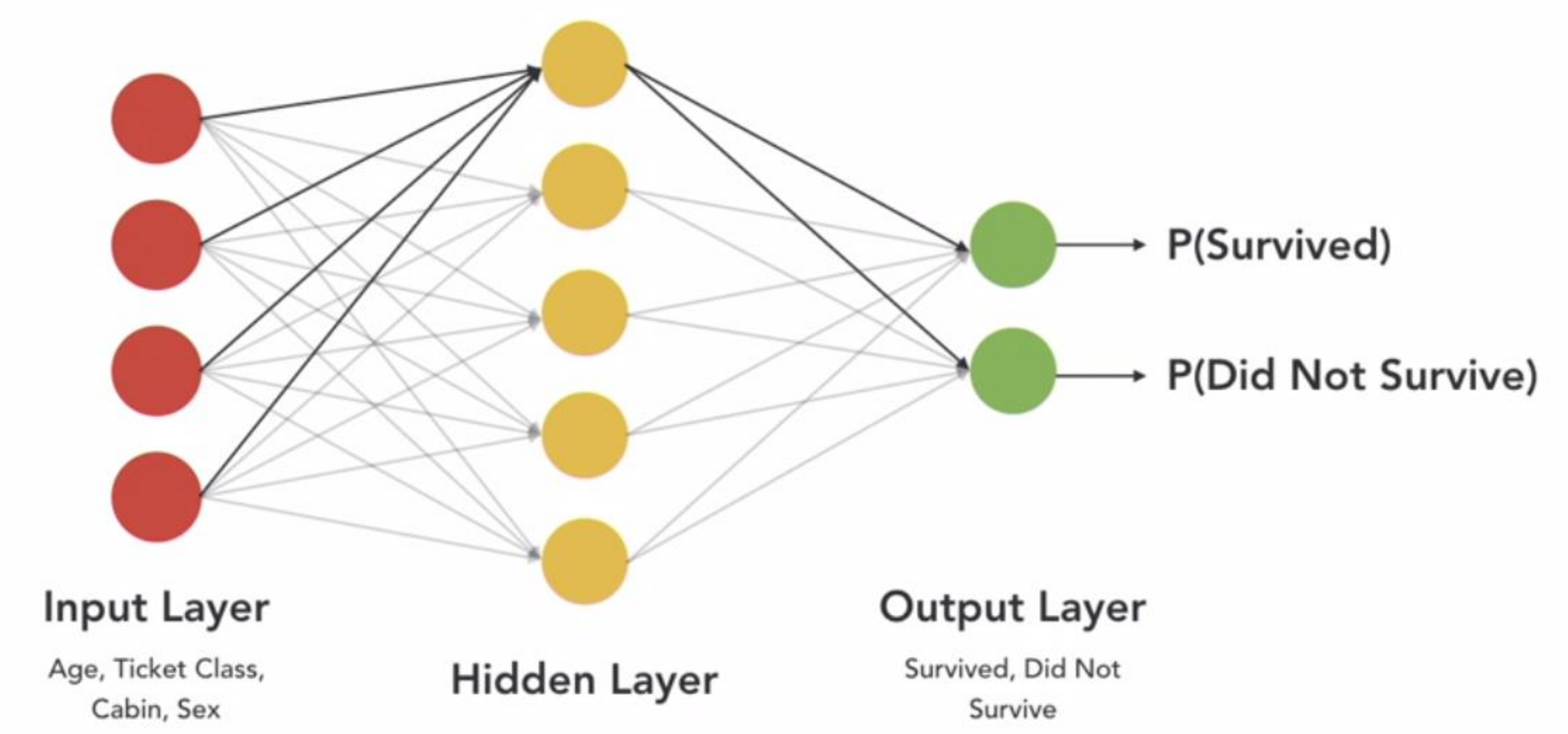


x2
x1
low c
x2
x1
large c

Multi-Layer Perceptrons

A **multilayer perceptron** is a classic feed-forward artificial neural network, the core component of deep learning.

Alternatively: A **multilayer perceptron** is a connected series of nodes (in the form of a directed acyclic graph), where each node represents a function or a model.


P(Survived)
P(Did Not Survive)
Input Layer
Age, Ticket Class, Cabin, Sex
Hidden Layer
Output Layer
Survived, Did Not Survive

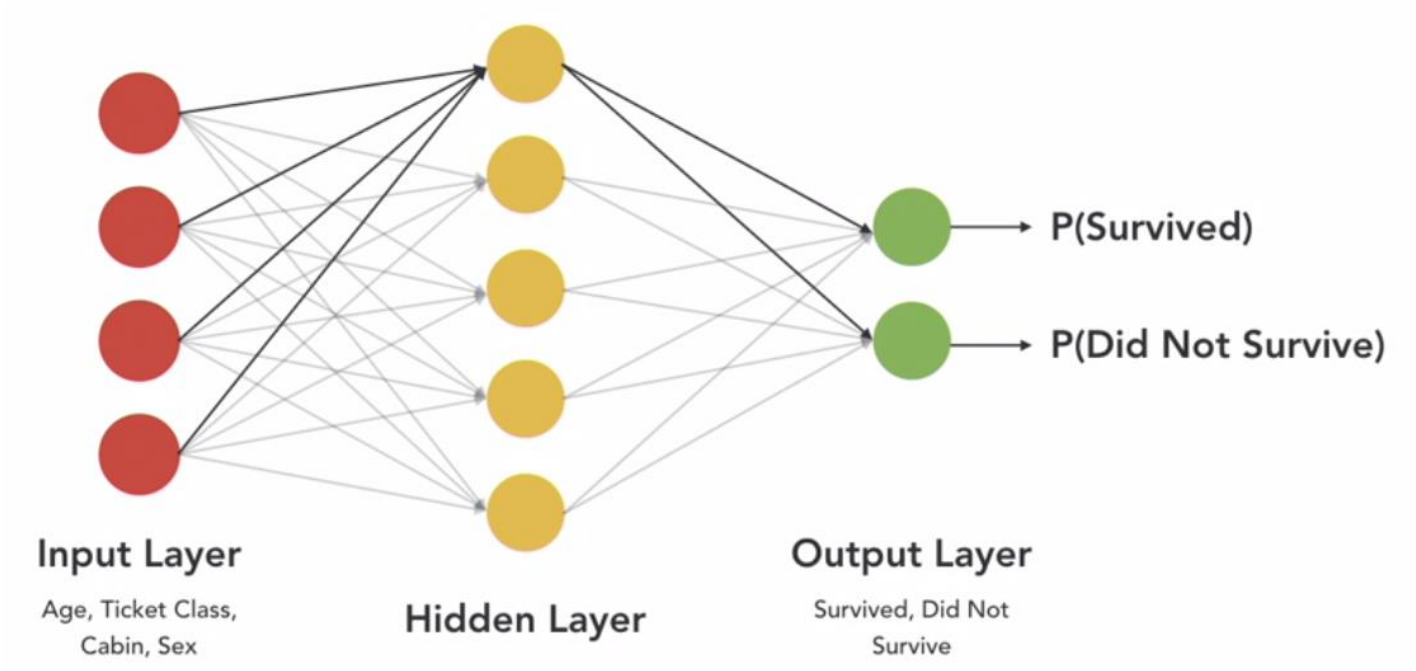
When to Use It?

- Categorical or continuous target variable
- Very complex relationships or performance is the only thing that matters
- When control over the training process is very important

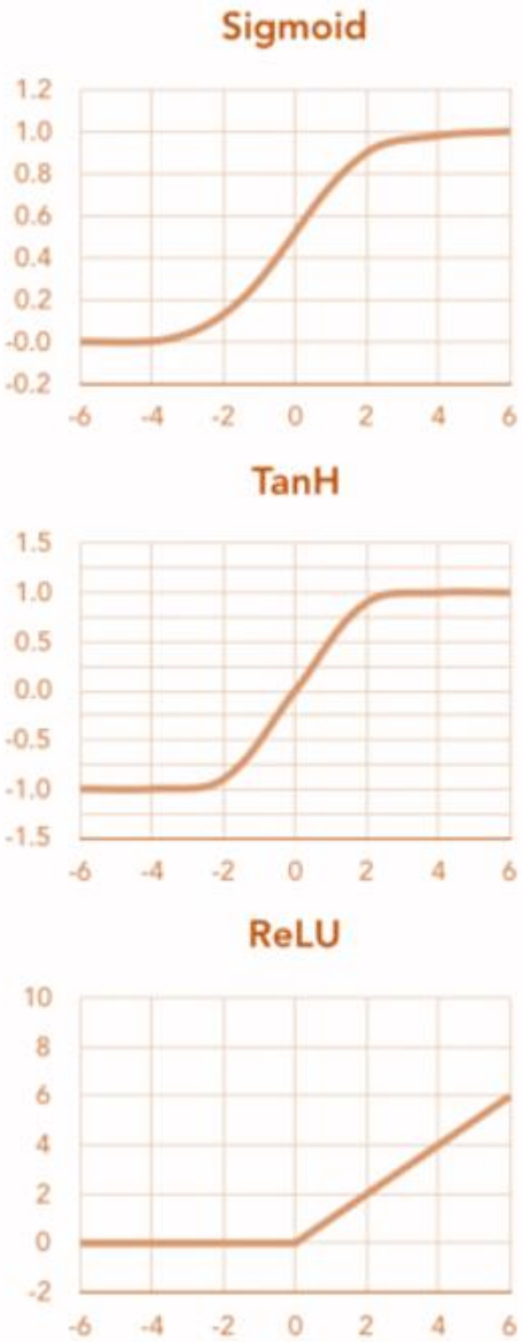
When Not to Use It?

- Image recognition, time series, etc.
- Transparency is important or interested in significance of predictors
- Need a quick benchmark model
- Limited data available

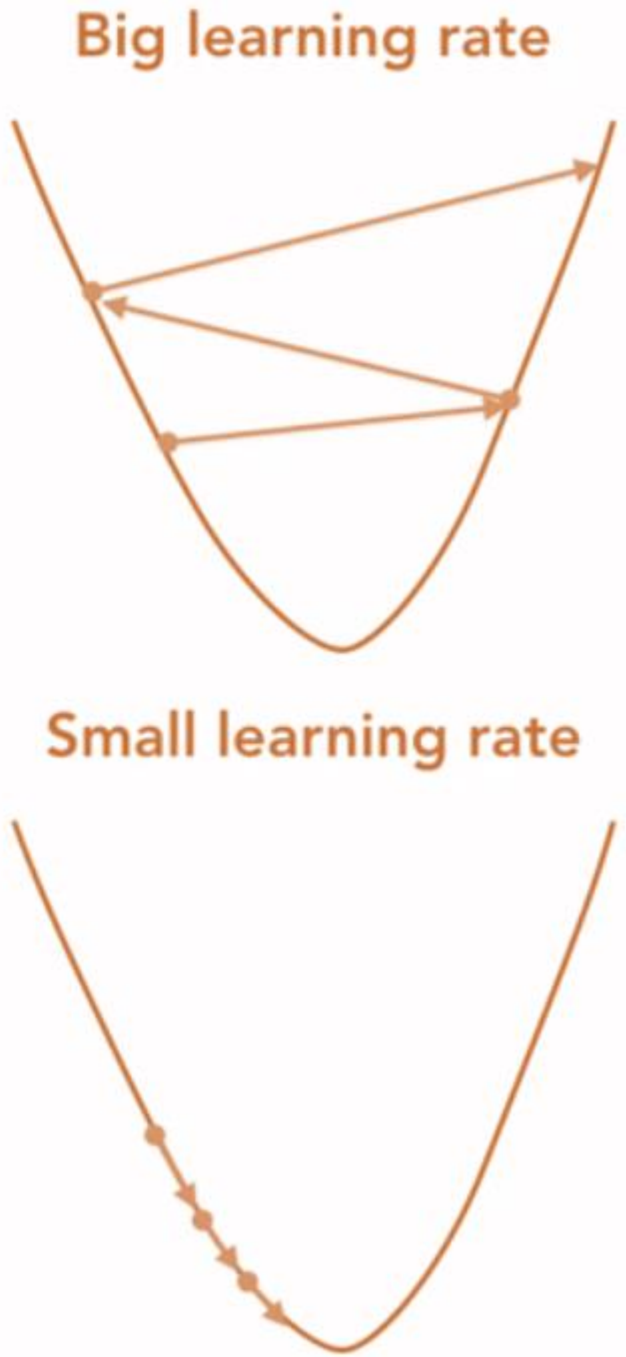
The **hidden layer**–size hyperparameter determines how many hidden layers there will be and how many nodes in each layer.



The **activation function** hyperparameter dictates the type of nonlinearity that is introduced to the model.



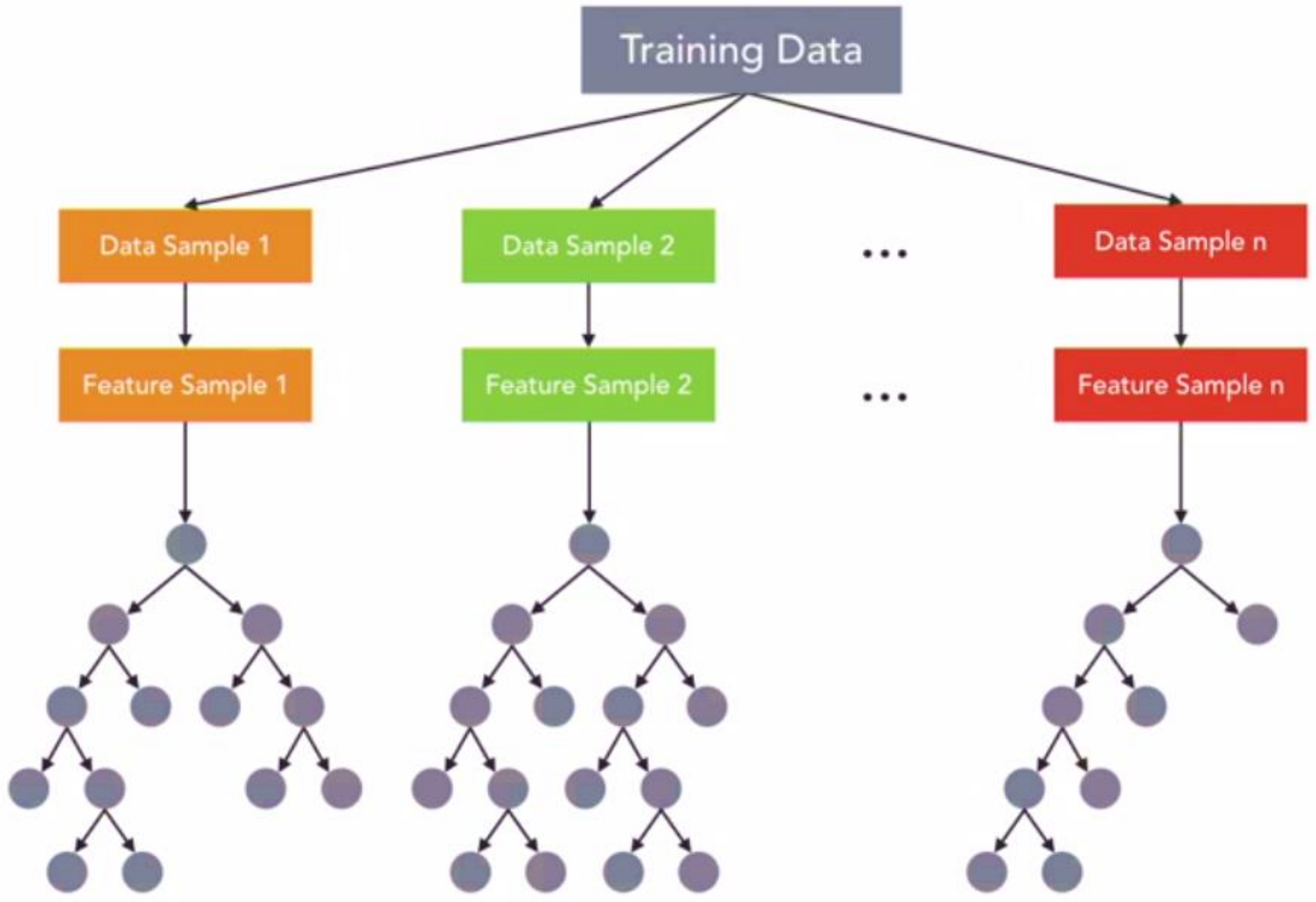
The **learning rate** hyperparameter facilitates both how quickly and whether or not the algorithm will find the optimal solution.



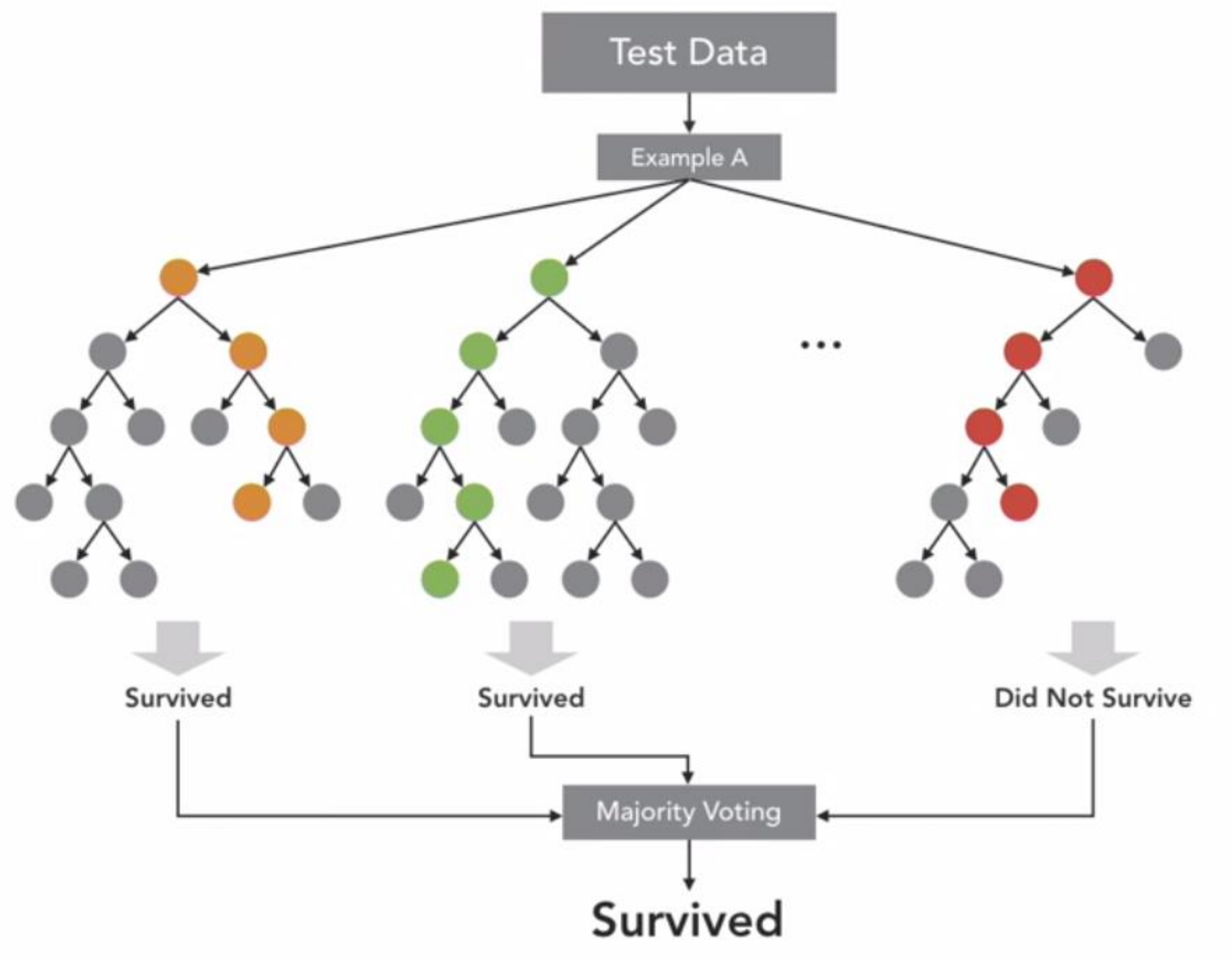
Random Forest

A **random forest** merges a collection of independent decision trees to get a more accurate and stable prediction.

Ensemble methods combine several machine learning models in order to decrease both bias and variance.



Training Data
Data Sample 1
Data Sample 2
...
Data Sample n
Feature Sample 1
Feature Sample 2
...
Feature Sample n


Test Data
Example A
...
Survived
Survived
Did Not Survive
Majority Voting
Survived

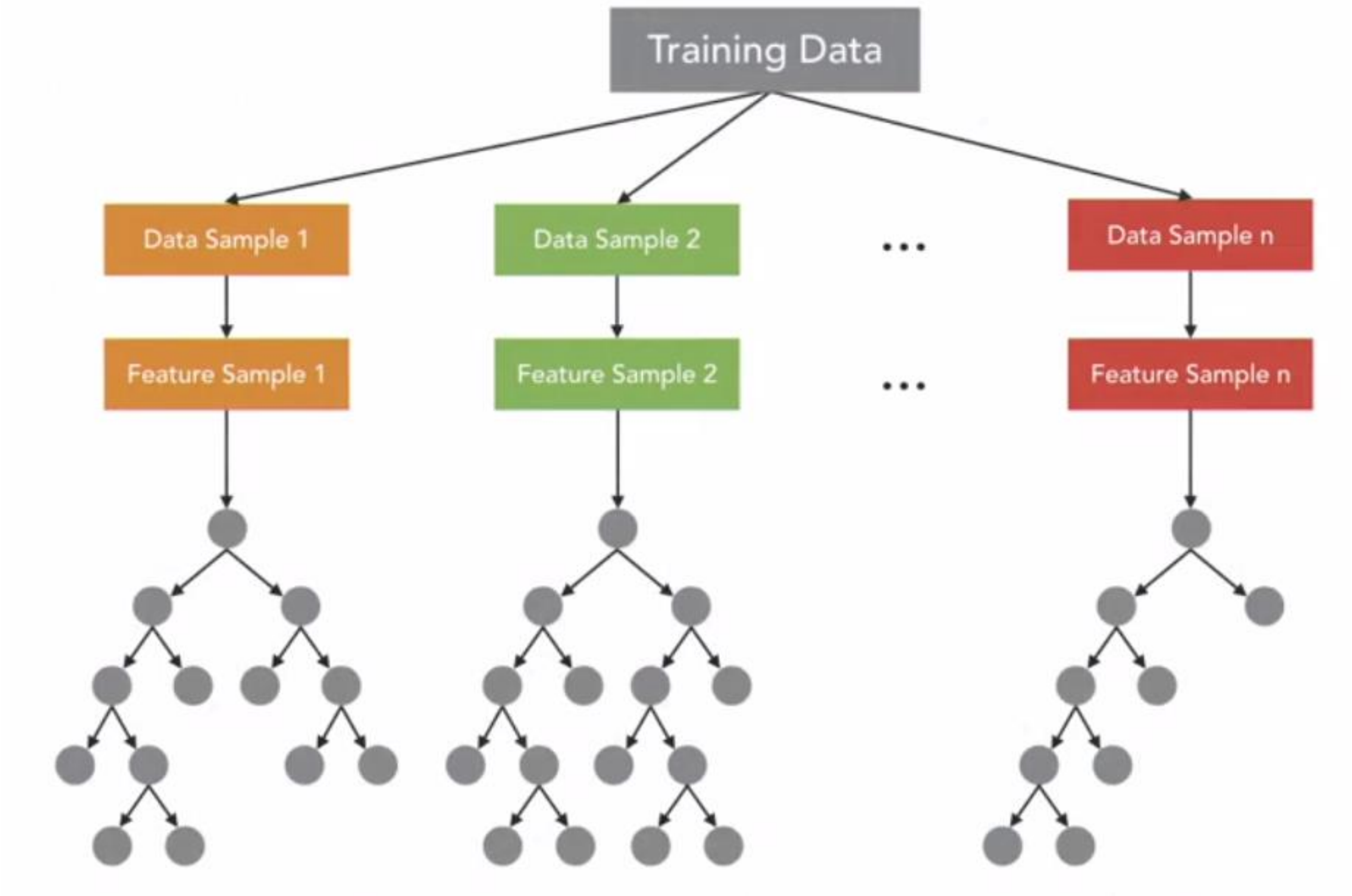
When to Use It?

- Categorical or continuous target variable
- Interested in significance of predictors
- Need a quick benchmark model
- If you have messy data, such as missing values, outliers

When Not to Use It?

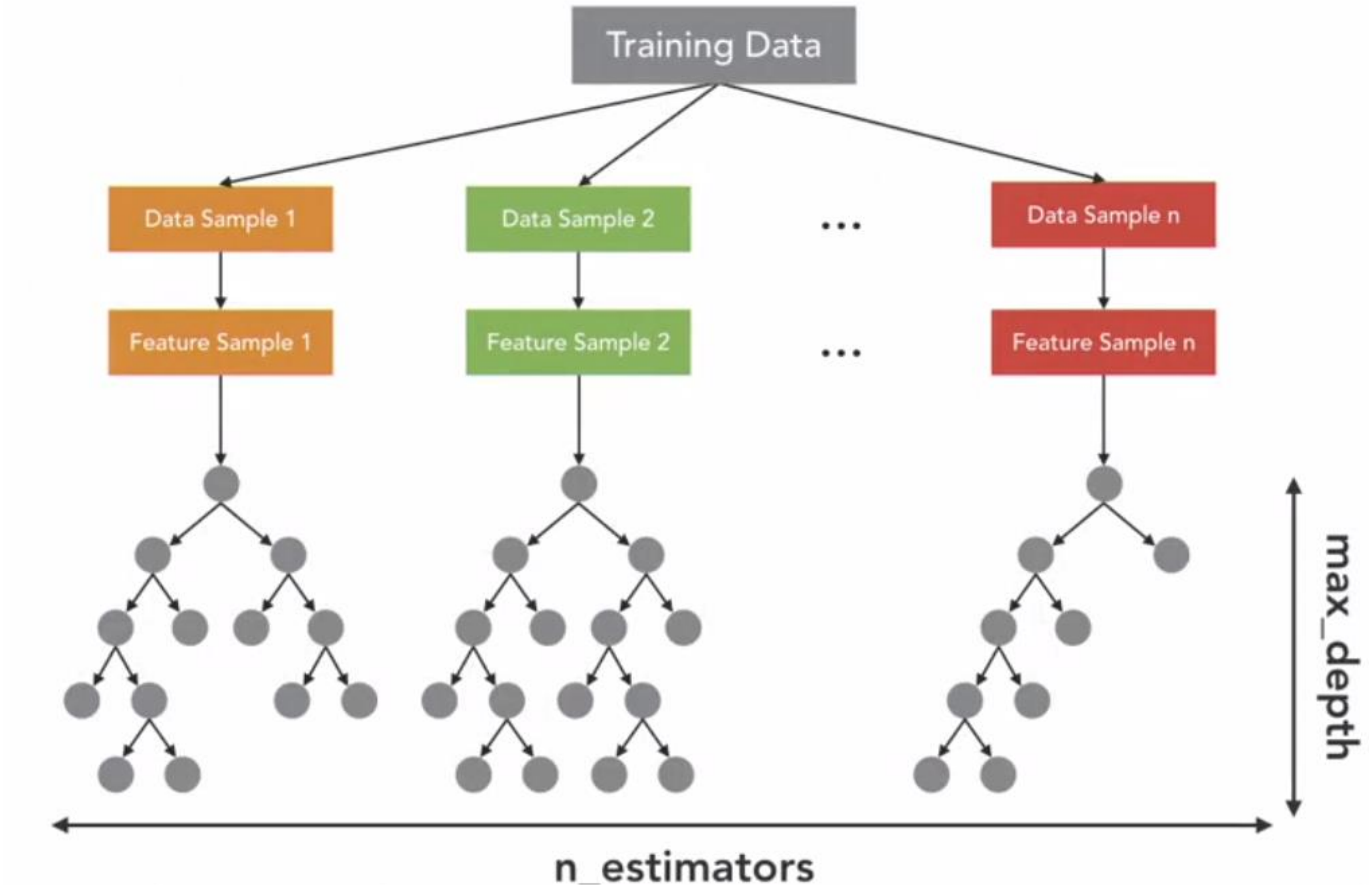
- If you're solving a very complex, novel problem
- Transparency is importnat
- Prediction time is important

Hyperparameters



The **n_estimators** hyperparameter controls how many individual decision trees will be built.

The **max_depth** hyperparameter controls how deep each individual decision tree can go.


Training Data
Data Sample 1
Data Sample 2
...
Data Sample n
Feature Sample 1
Feature Sample 2
...
Feature Sample n
max_depth
n_estimators

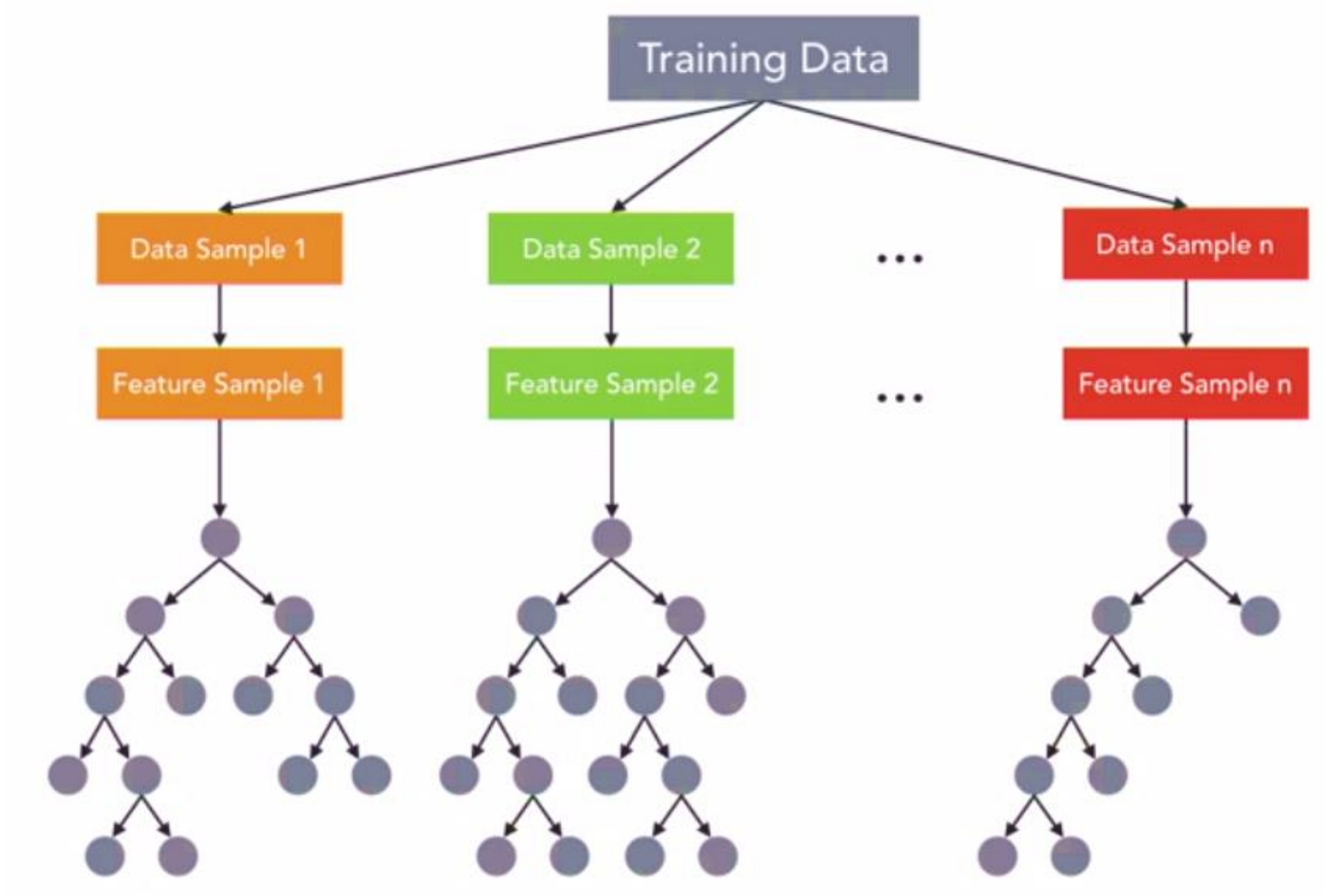
Boosting

Boosting is an ensemble method that aggregates a number of weak models to create one strong model.

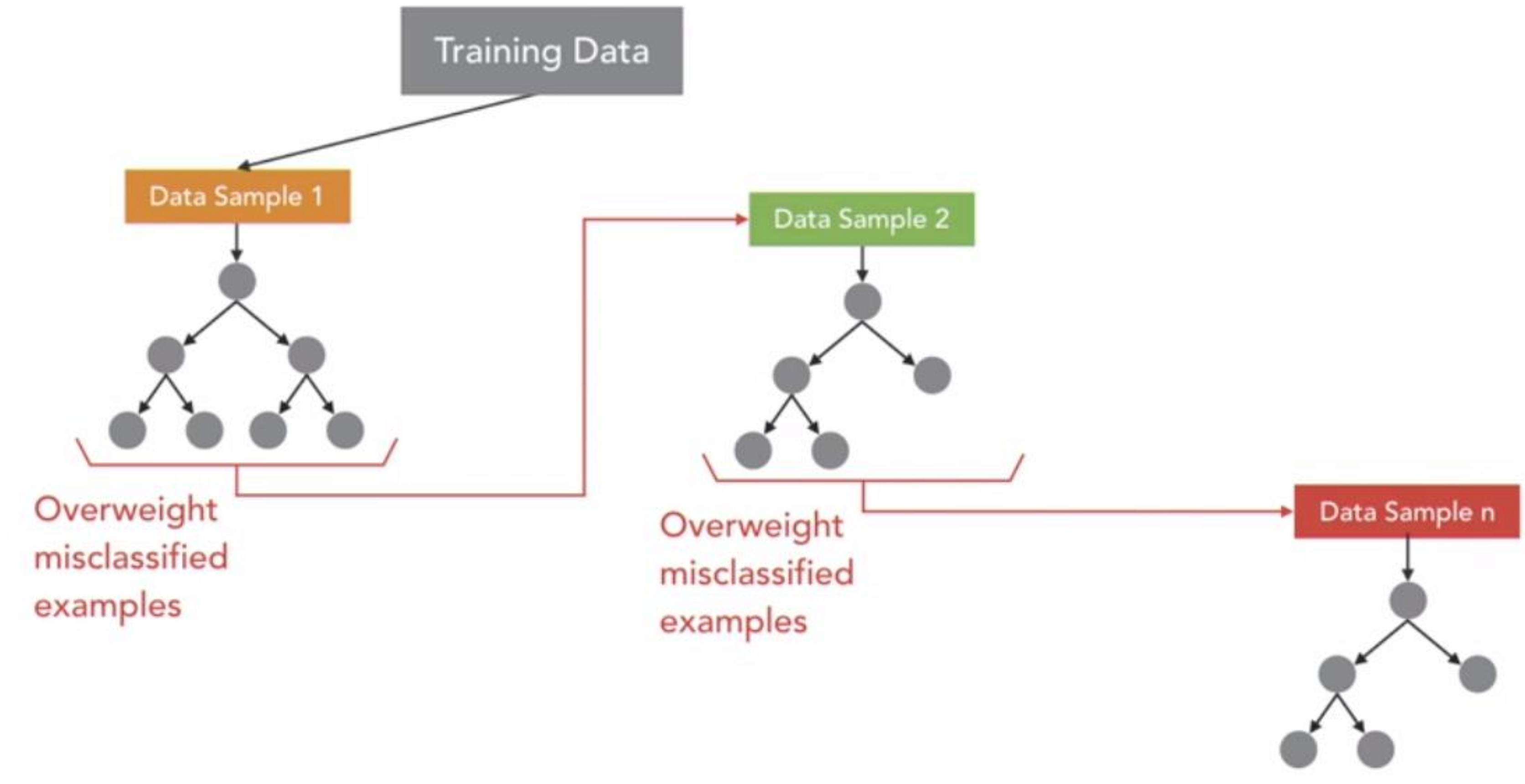
A weak model is one that is only slightly better than random guessing. A strong model is one that is strongly correlated with the true classification.

Boosting effectively learns from its mistakes with each iteration.

Random Forest

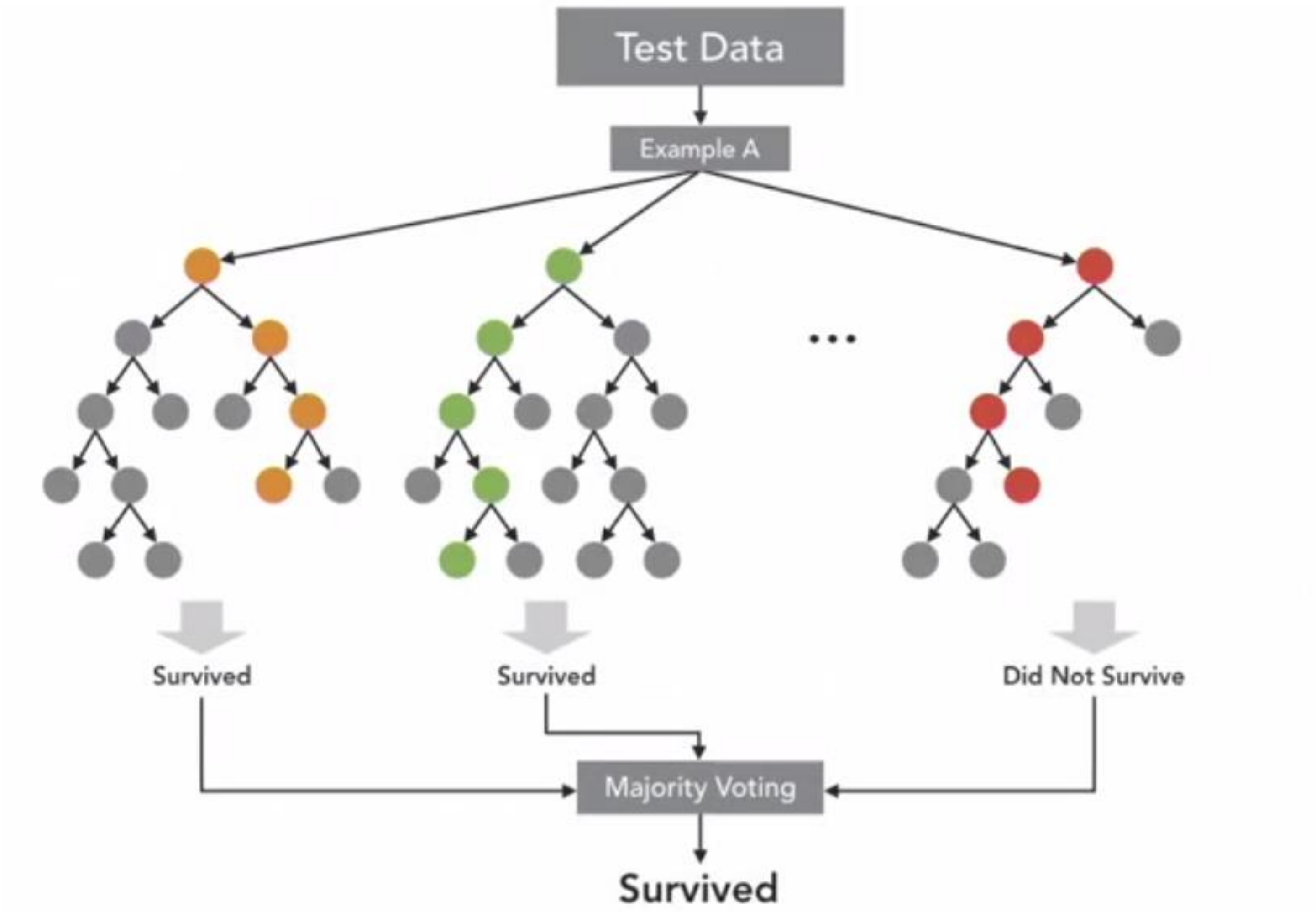


Boosting



Training Data
Data Sample 1
Data Sample 2
Data Sample n
Overweight misclassified examples
Overweight misclassified examples

Random Forest



Boosting


Test Data
Example A
...
Survived
Survived
Did Not Survive
Weighted Voting
Survived

Boosting

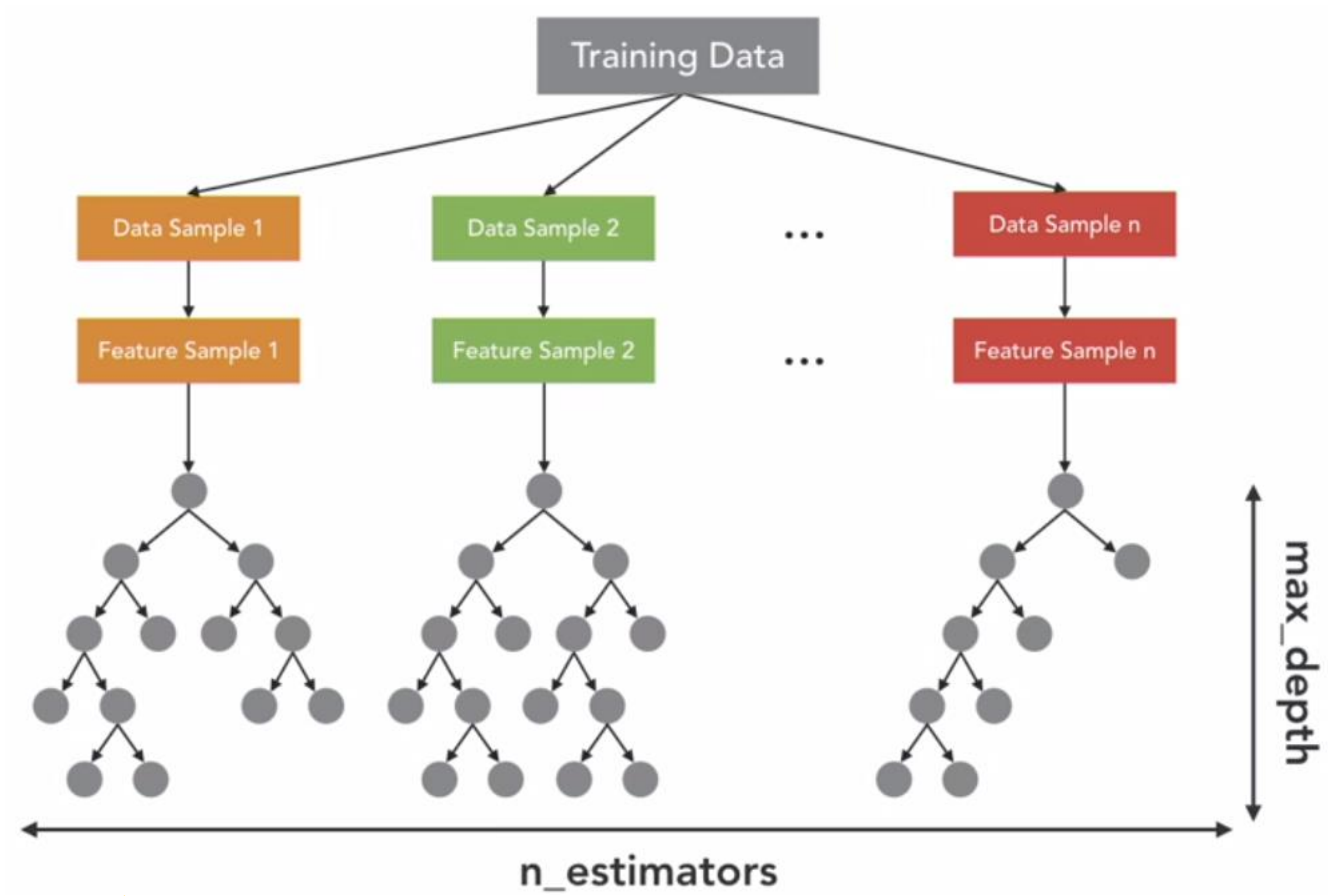
When to Use It?

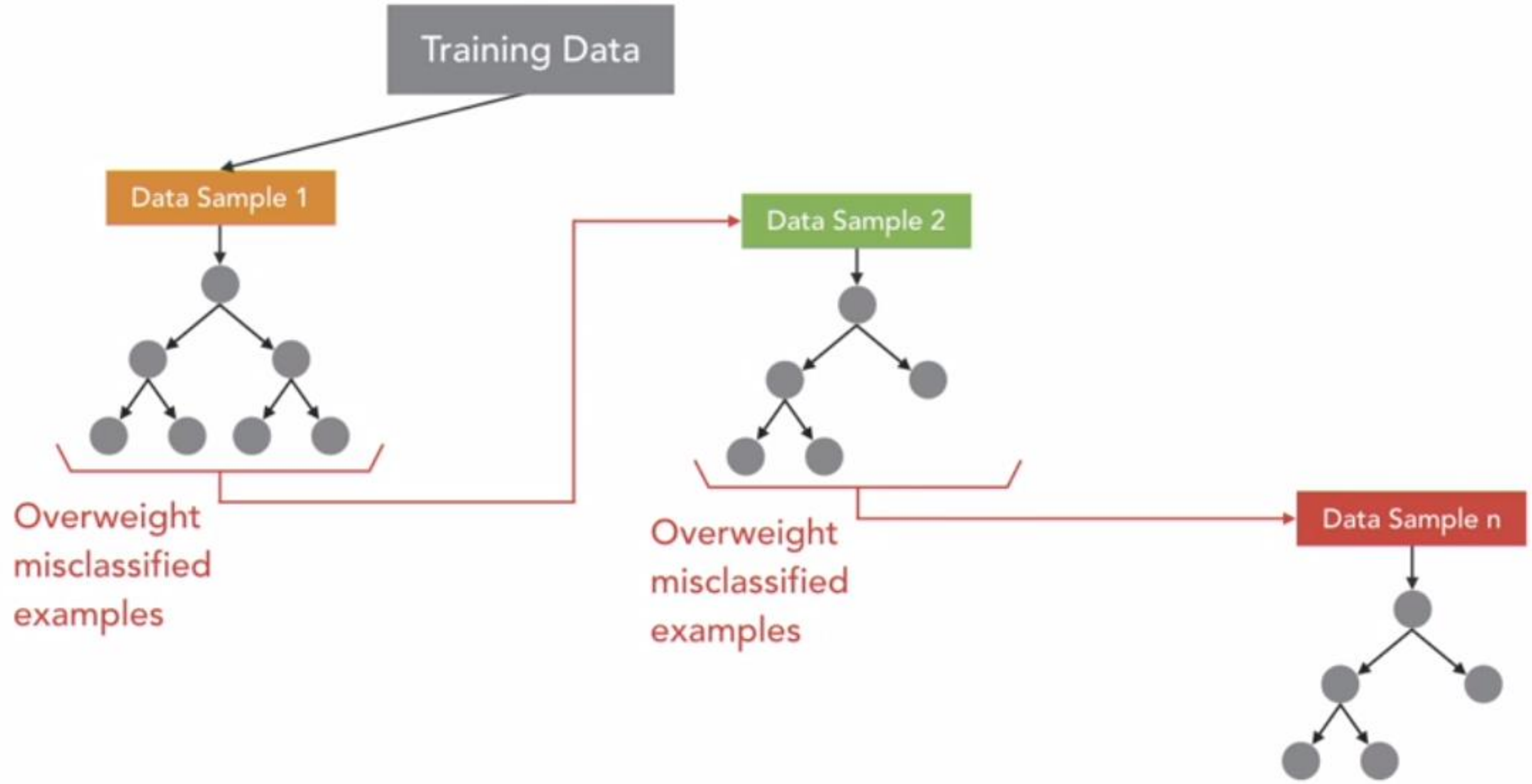
- Categorical or continuous target variable
- Useful on nearly any type of problem
- Interested in significance of predictors
- Prediction time is important

When Not to Use It?

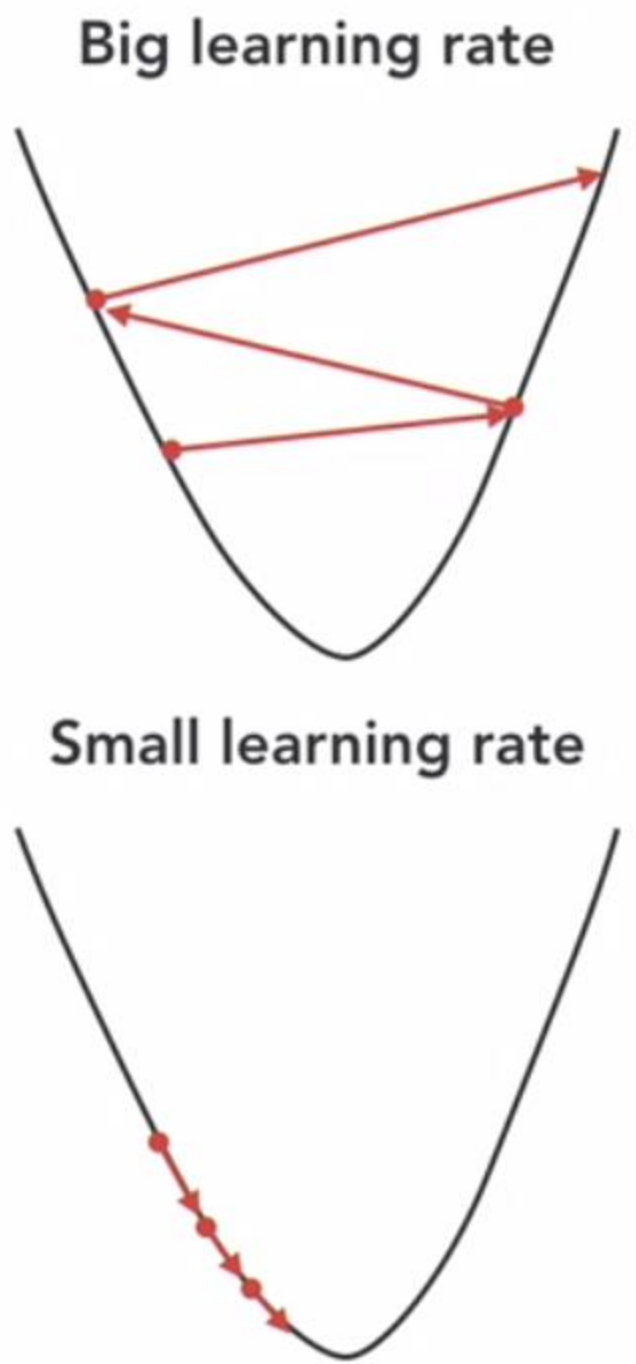
- Transparency is important
- Training time is important or compute power is limited
- Data is really noisy

Hyperparameters




Training Data
Data Sample 1
Overweight misclassified examples
Data Sample 2
Overweight misclassified examples
Data Sample n

The **learning rate** hyperparameter facilitates both how quickly and whether or not the algorithm will find the optimal solution.



Summary

Which algorithm generates the best model for this given problem?

"No free lunch" theorem: No algorithm works best for every problem.

Accuracy

- How do they handle data of different sizes, such as short and fat, long and skinny?
- How will they handle the complexity of feature relationships?
- How will they handle messy data?

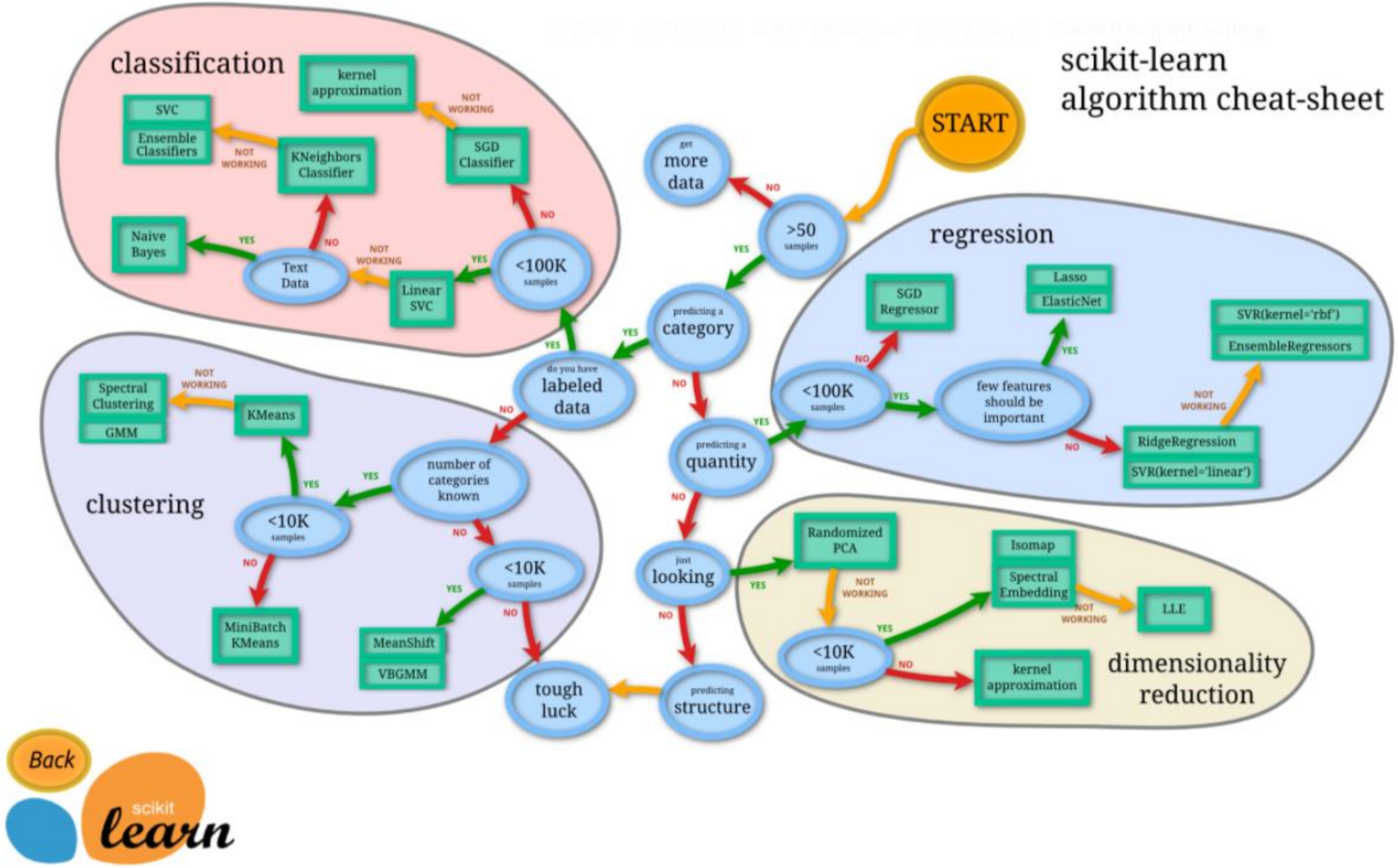
Latency

- How long will it take to train?
- How long will it take to predict?

Comparison

	Problem Type	Train Speed	Predict Speed	Interpretability	Performance	Performance with Limited Data
Logistic Regression	Classification	Fast	Fast	Medium	Lower	Higher
Support Vector Machines	Classification	Slow	Moderate	Low	Medium	Higher
Multilayer Perception	Both	Slow	Moderate	Low	High	Lower
Random Forest	Both	Moderate	Moderate	Low	Medium	Lower
Boosted Trees	Both	Slow	Fast	Low	High	Lower

Congratulations



scikit-learn
algorithm cheat-sheet
START
>50 samples
get more data
predicting a category
do you have labeled data
predicting a quantity
just looking
predicting structure
tough luck
classification
<100K samples
SGD Classifier
kernel approximation
Linear SVC
Text Data
Naive Bayes
KNeighbors Classifier
SVC
Ensemble Classifiers
clustering
number of categories known
<10K samples
KMeans
Spectral Clustering
GMM
MiniBatch KMeans
MeanShift
VBGMM
regression
SGD Regressor
few features should be important
Lasso
ElasticNet
RidgeRegression
SVR(kernel='linear')
SVR(kernel='rbf')
EnsembleRegressors
dimensionality reduction
Randomized PCA
Isomap
Spectral Embedding
LLE
kernel approximation
YES
NO
NOT WORKING
Back
scikit learn