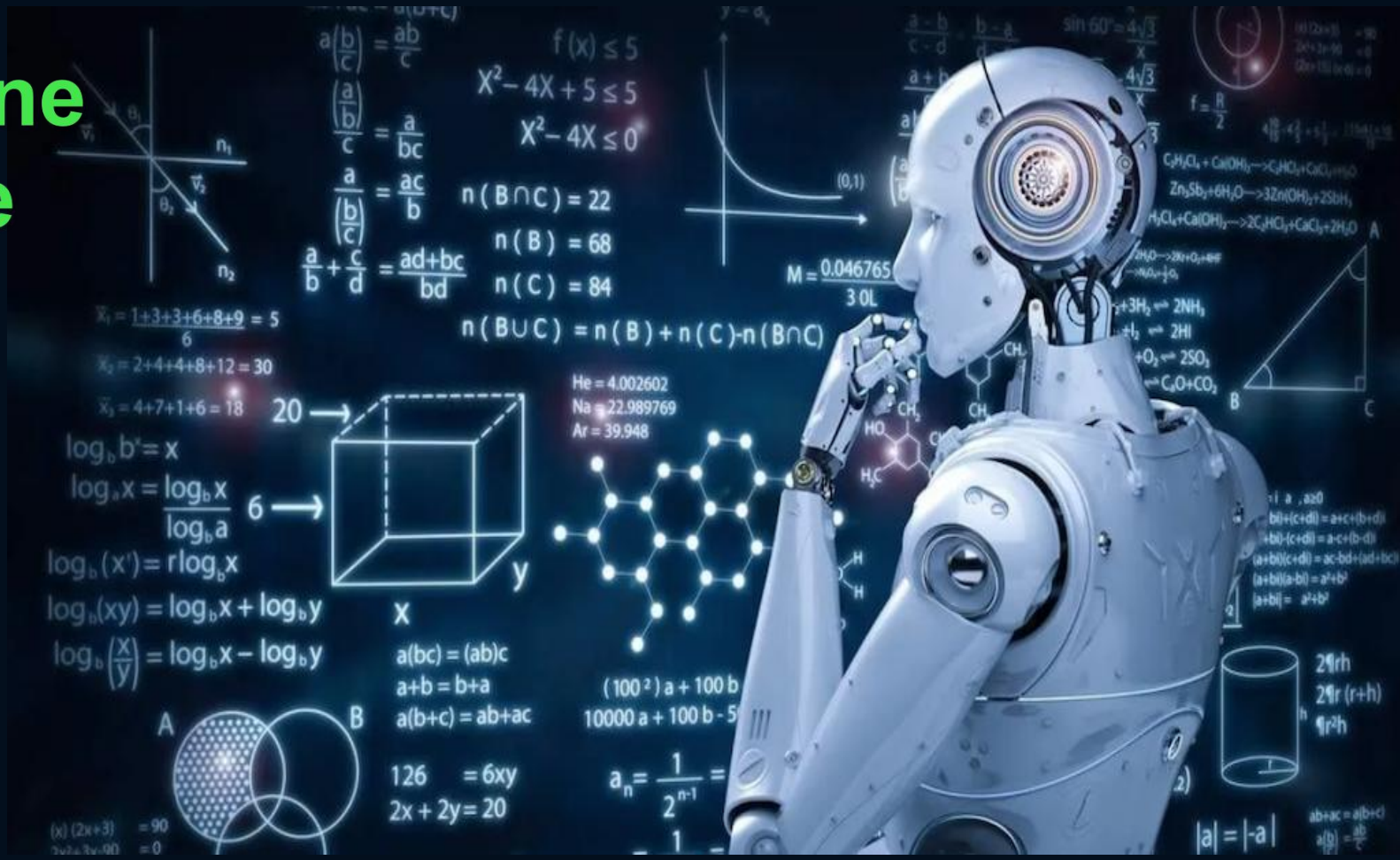
Nottingham Business School

Principles of Machine Learning in Finance

6. Advanced Suprvised Learning | Tree-Based Learnng



Learning Outcomes

- Tree-Based Learning and Decision Trees
- Ensemble Learning and Random Forest
- Hyperparamater Tuning
- Bagging and Boosting
- Tuning and Validation of Decision Trees with Python
- **Coding Activity 6-1**: Supervised ML. Decision Trees || [Decision Tree with Python (European Bank Data Modelling)]
- **Coding Activity 6-2**: Supervised ML. Decision Trees || [Random Forest Model with Python (European Bank Data Modelling)]

Tree-Based Learning

- **Tree-based learning** is a type of supervised machine learning that performs classification and regression tasks
- **Decision tree** is a flow-chart-like supervised classification model, and a representation of various solutions that are available to solve a given problem, based on the possible outcomes of related choices

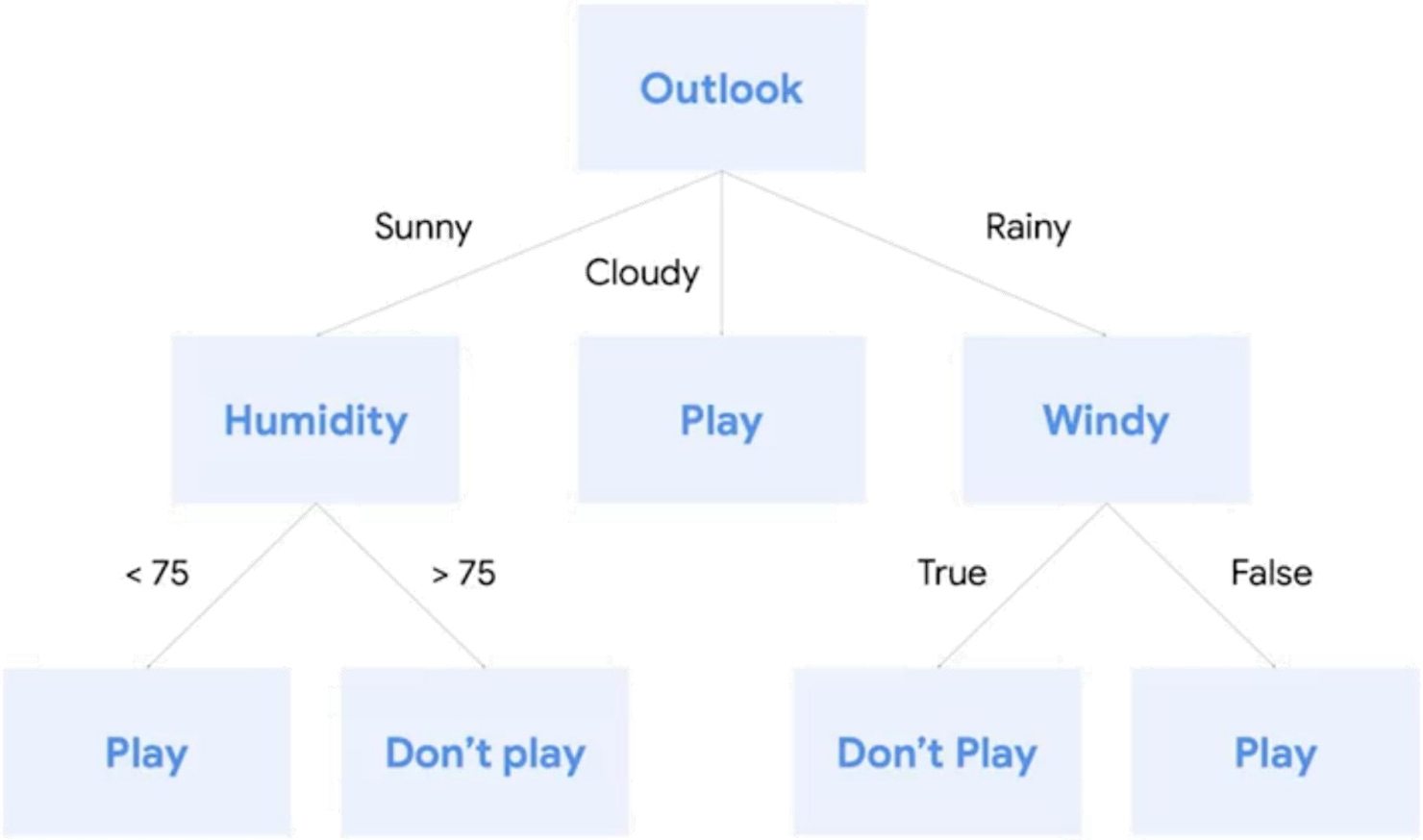
Decision Trees: Advantages and Disadvatages

Decision trees enable data professionals to make predictions about future events based on the information that is currently available:

Advantages	Disadvantages
+ Require no assumptions regarding distribution of data; + Handle collinearity very easily; + Often do not require data preprocessing	- Can be particularly susceptible to overfitting

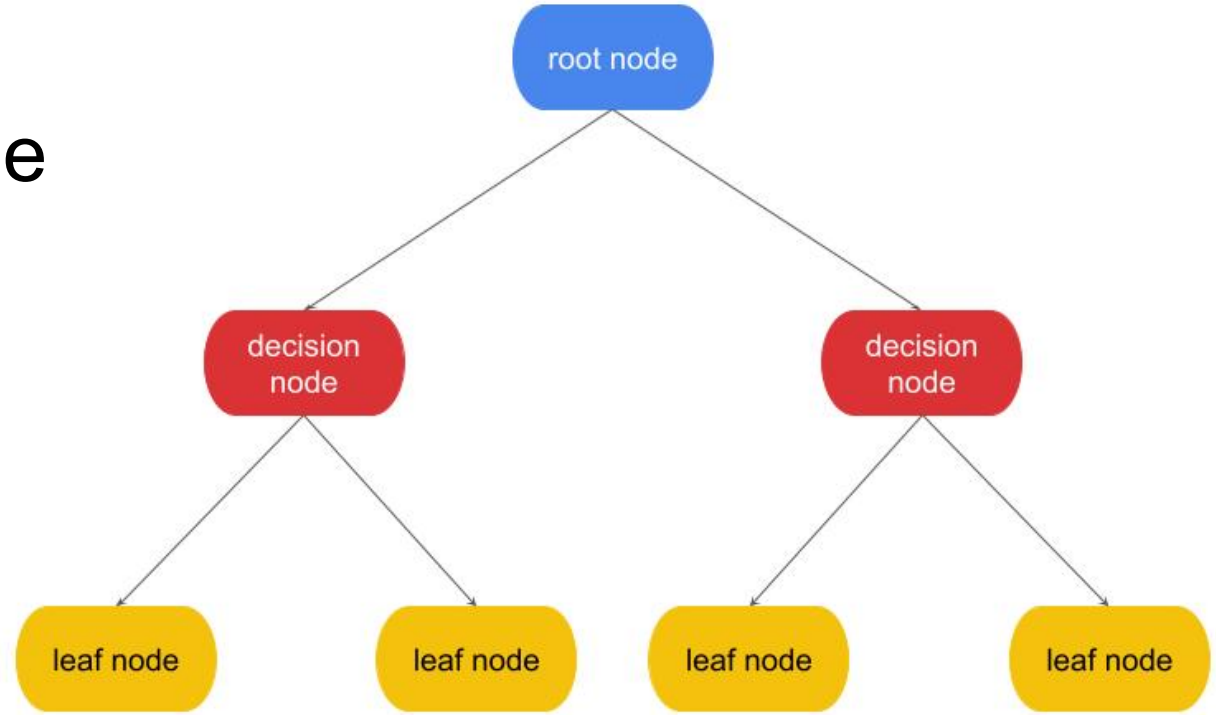
Example 1. Decision Tree

Shall I play football?

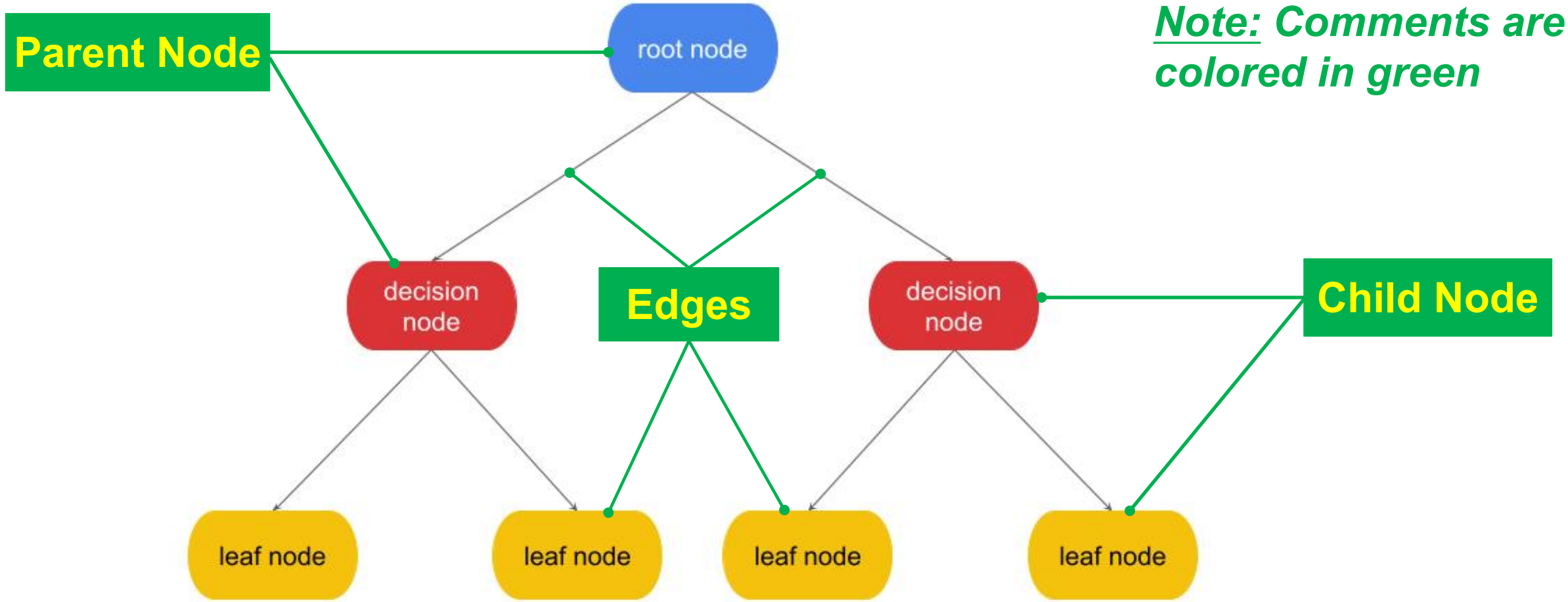


Decision Tree: Types of nodes

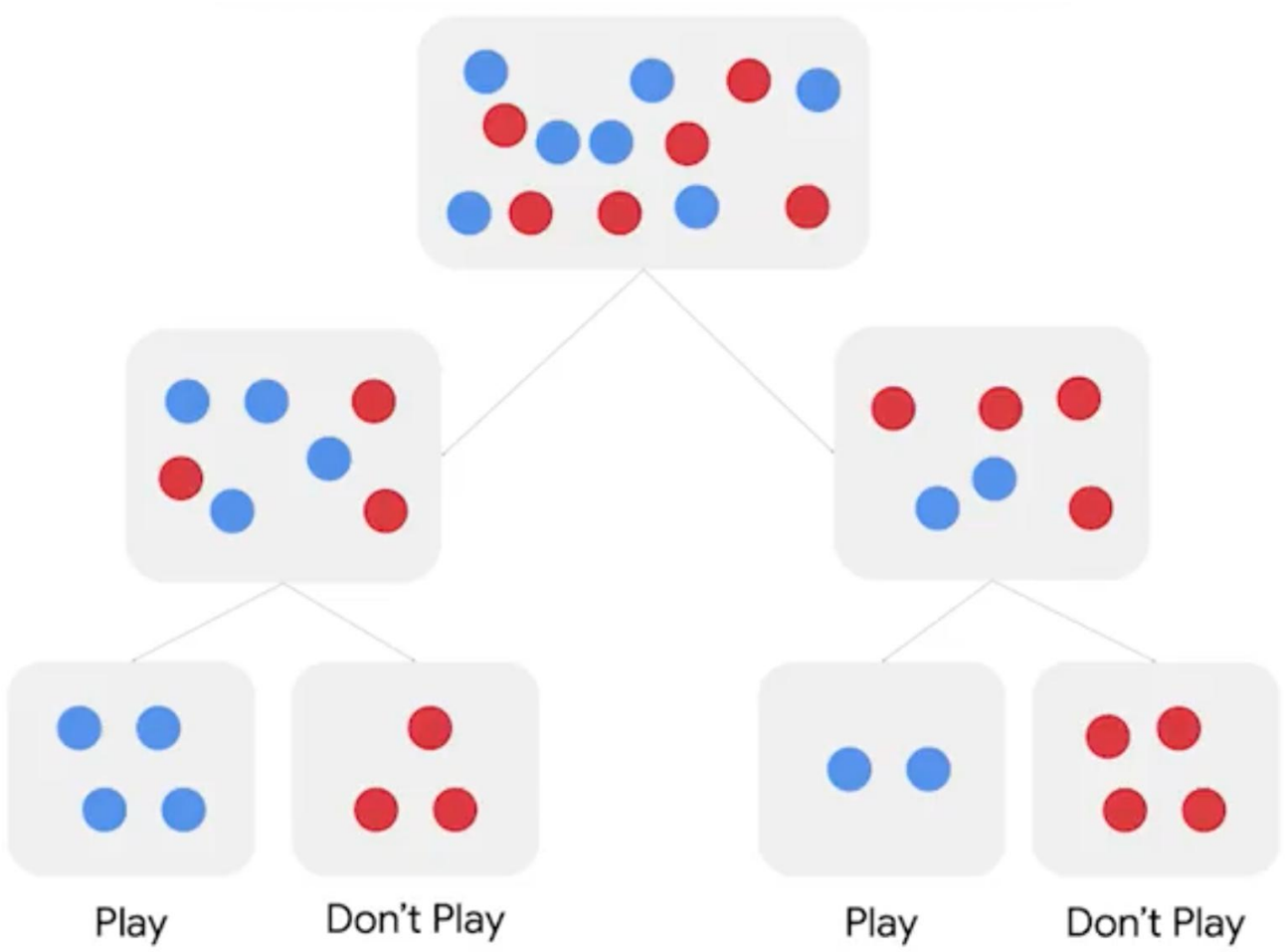
- **Root node** is the first node of the tree, where the first decision is made;
- **Decision nodes** are nodes of the tree where decisions are made
- **Leaf node** is the node where a final prediction is made
- **Child node** is a node that is pointed to from another node
- **Parent node** is the node that is pointing to a child node



Decision Tree: Types of nodes (2)



Example 2. Decision Tree



Decision Trees: Hyperparameter Tuning

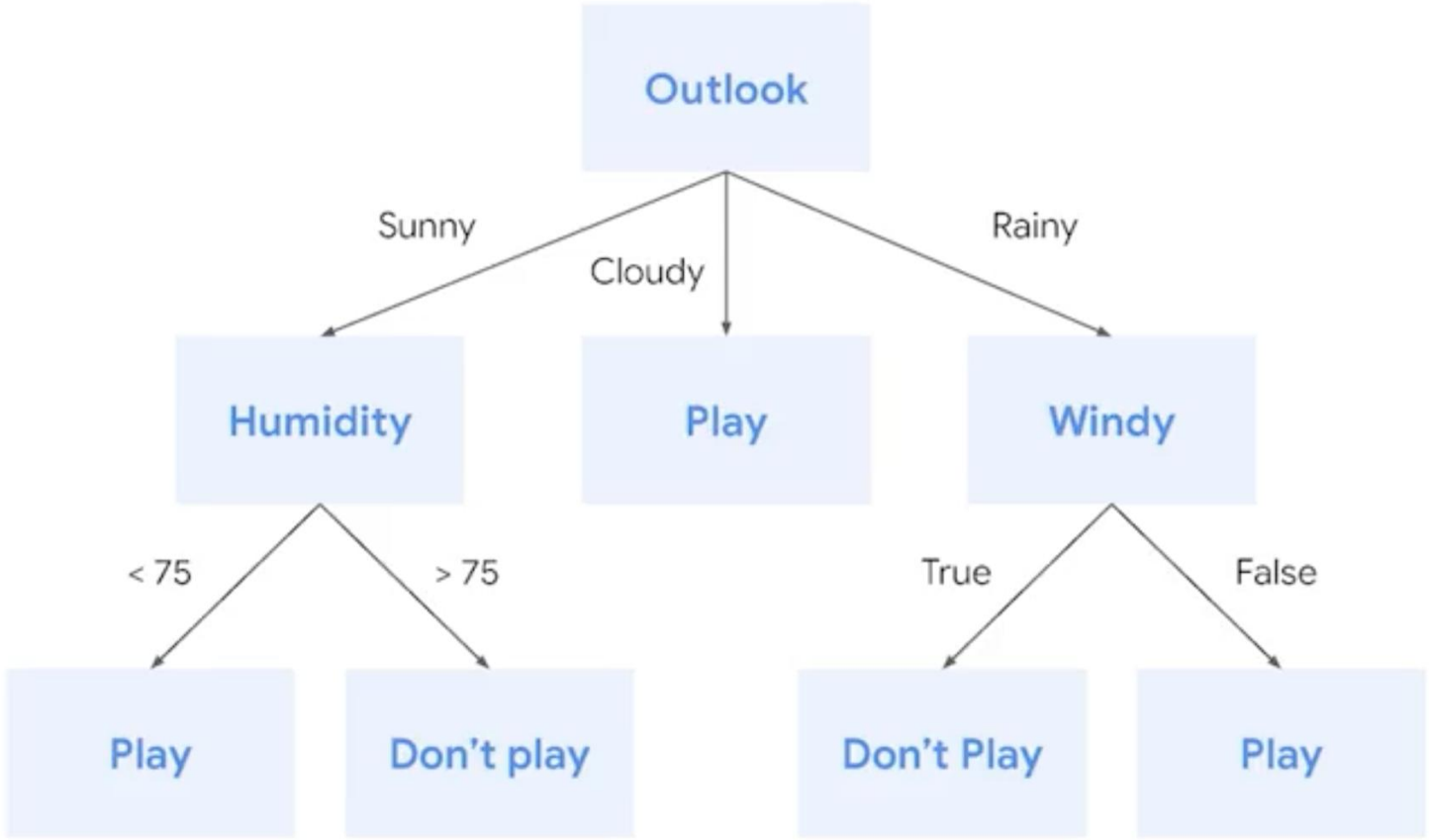
Hyperparameter tuning is the process of adjusting the parameters to find the best values that will result in the most optimal model.

Hyperparameters are parameters that can be set before the model is trained:

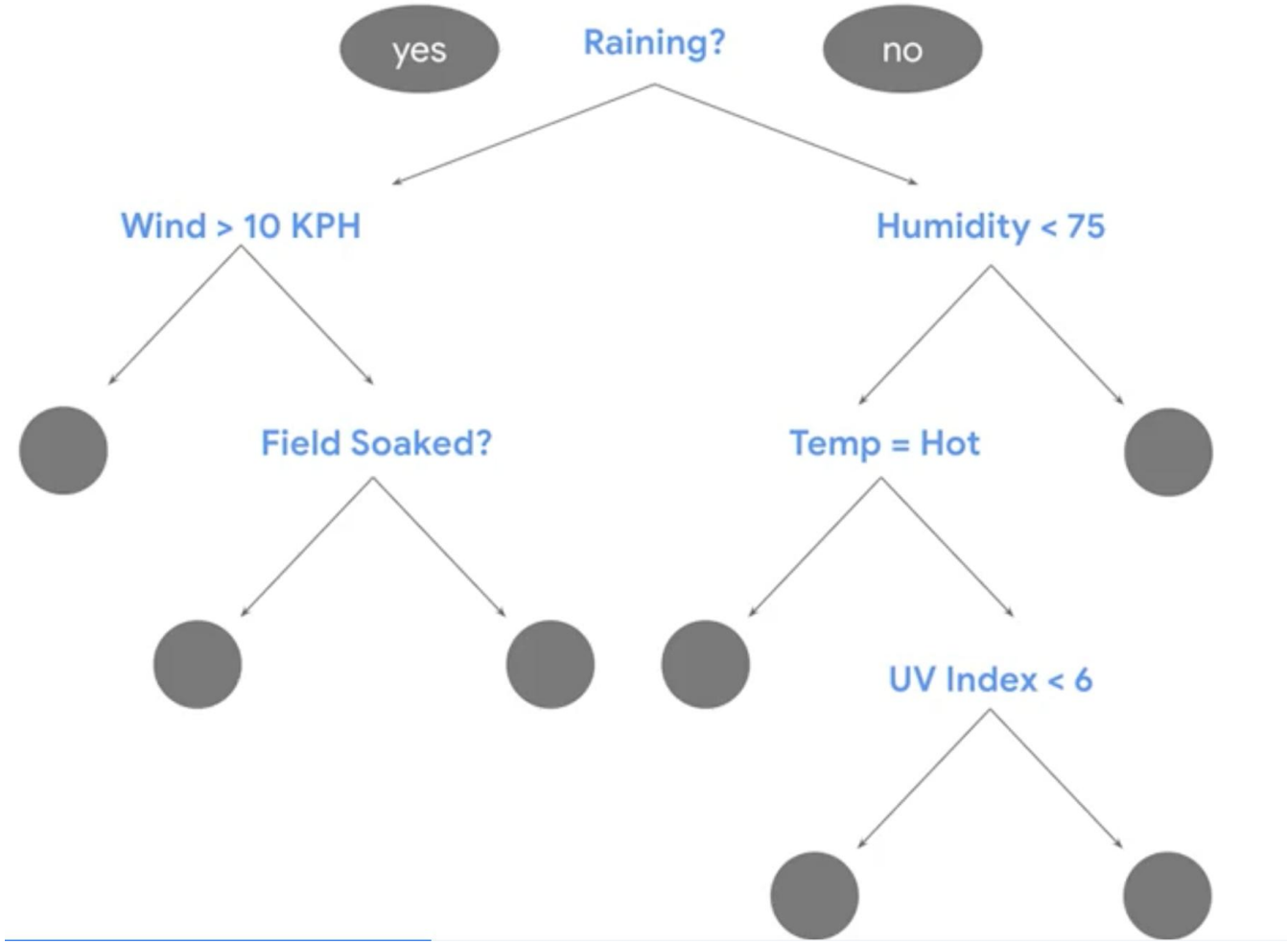
- **Max depth** defines how "long" a decision tree can get;
- **Min samples leaf** defines the minimum number of samples for a leaf node.

Decision Trees: Hyperparameter Tuning (2)

Shall I play football?



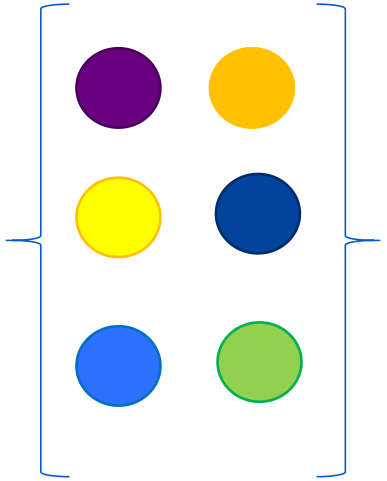
Decision Trees: Hyperparameter Tuning (3)



		max_depth		
		6	8	None
min_samples_leaf	1	(1, 6)	(1, 8)	(1, None)
	3	(3, 6)	(3, 8)	(3, None)
	5	(5, 6)	(5, 8)	(5, None)

Decision Trees: GridSearch

GridSearch is a tool to confirm that a model achieves its intended purpose by systematically checking every combination of hyperparameters to identify which set produces the best results, based on the selected metric



GridSearch

Model 1

Model 2

Model 3

Model Validation and Cross-Validation

Model Validation is the set of processes and activities intended to verify that models are performing as expected

Cross-validation is a process that uses different portions of the data to test and train a model on different iterations

Example 3. Model Validation

Label	features1	features2	features3
0	-0.79764903	0.79764903	1.35714286
1	-1.00208768	1.00208768	1.71428571
0	1.18733509	1.18733509	1.92857143
0	-1.2741652	1.2741652	2.07142857
0	-0.97044552	0.97044552	1.57142857
2	-1.83622829	1.83622829	2.64285714
0	-1.08588351	1.08588351	1.57142957
1	-0.63944909	0.63944909	0.92857143
0	2.3118544	2.3118544	3.35714286
1	1.15057529	1.15057529	1.64285714
0	1.50225338	1.50225338	2.14285714
1	-0.82858622	0.82858622	1.14285714
1	-0.81674324	0.81674324	1.14285714
0	-1.20542441	1.20542441	1.71428571
2	-1.53276956	1.53276956	2.07142857

Train

Label	features1	features2	features3
0	-0.79764903	0.79764903	1.35714286
1	-1.00208768	1.00208768	1.71428571
0	1.18733509	1.18733509	1.92857143
0	-1.2741652	1.2741652	2.07142857
0	-0.97044552	0.97044552	1.57142857
2	-1.83622829	1.83622829	2.64285714
0	-1.08588351	1.08588351	1.57142957
1	-0.63944909	0.63944909	0.92857143
0	2.3118544	2.3118544	3.35714286
1	1.15057529	1.15057529	1.64285714

Test

Label	features1	features2	features3
0	1.50225338	1.50225338	2.14285714
1	-0.82858622	0.82858622	1.14285714
1	-0.81674324	0.81674324	1.14285714
0	-1.20542441	1.20542441	1.71428571
2	-1.53276956	1.53276956	2.07142857

Example 3. Model Validation (2)

Label	features1	features2	features3
0	-0.79764903	0.79764903	1.35714286
1	-1.00208768	1.00208768	1.71428571
0	1.18733509	1.18733509	1.92857143
0	-1.2741652	1.2741652	2.07142857
0	-0.97044552	0.97044552	1.57142857
2	-1.83622829	1.83622829	2.64285714
0	-1.08588351	1.08588351	1.57142957
1	-0.63944909	0.63944909	0.92857143
0	2.3118544	2.3118544	3.35714286
1	1.15057529	1.15057529	1.64285714
0	1.50225338	1.50225338	2.14285714
1	-0.82858622	0.82858622	1.14285714
1	-0.81674324	0.81674324	1.14285714
0	-1.20542441	1.20542441	1.71428571
2	-1.53276956	1.53276956	2.07142857

Train

Label	features1	features2	features3
0	-0.79764903	0.79764903	1.35714286
1	-1.00208768	1.00208768	1.71428571
0	1.18733509	1.18733509	1.92857143
0	-1.2741652	1.2741652	2.07142857
0	-0.97044552	0.97044552	1.57142857
2	-1.83622829	1.83622829	2.64285714
0	-1.08588351	1.08588351	1.57142957
1	-0.63944909	0.63944909	0.92857143
0	2.3118544	2.3118544	3.35714286

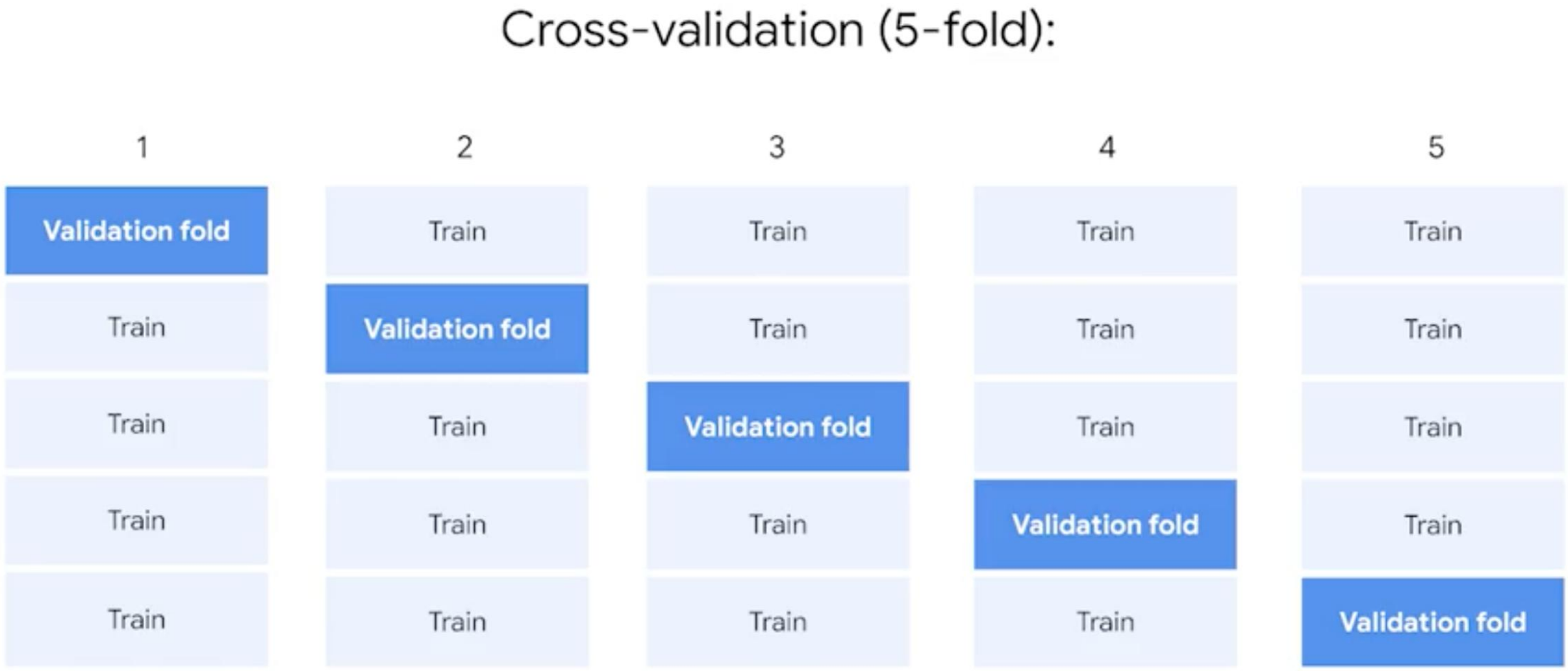
Validate

1	1.15057529	1.15057529	1.64285714
0	1.50225338	1.50225338	2.14285714
1	-0.81674324	0.81674324	1.14285714

Test

1	-0.82858622	0.82858622	1.14285714
0	-1.20542441	1.20542441	1.71428571
2	-1.53276956	1.53276956	2.07142857

Example 4. Cross-Validation



Coding Activity 6-1. Supervised ML. Decision Tree

Lab 6-1. Supervised ML. Decision Tree ||

[Decision Tree with Python (European Bank Data Modelling)]

Steps to follow:

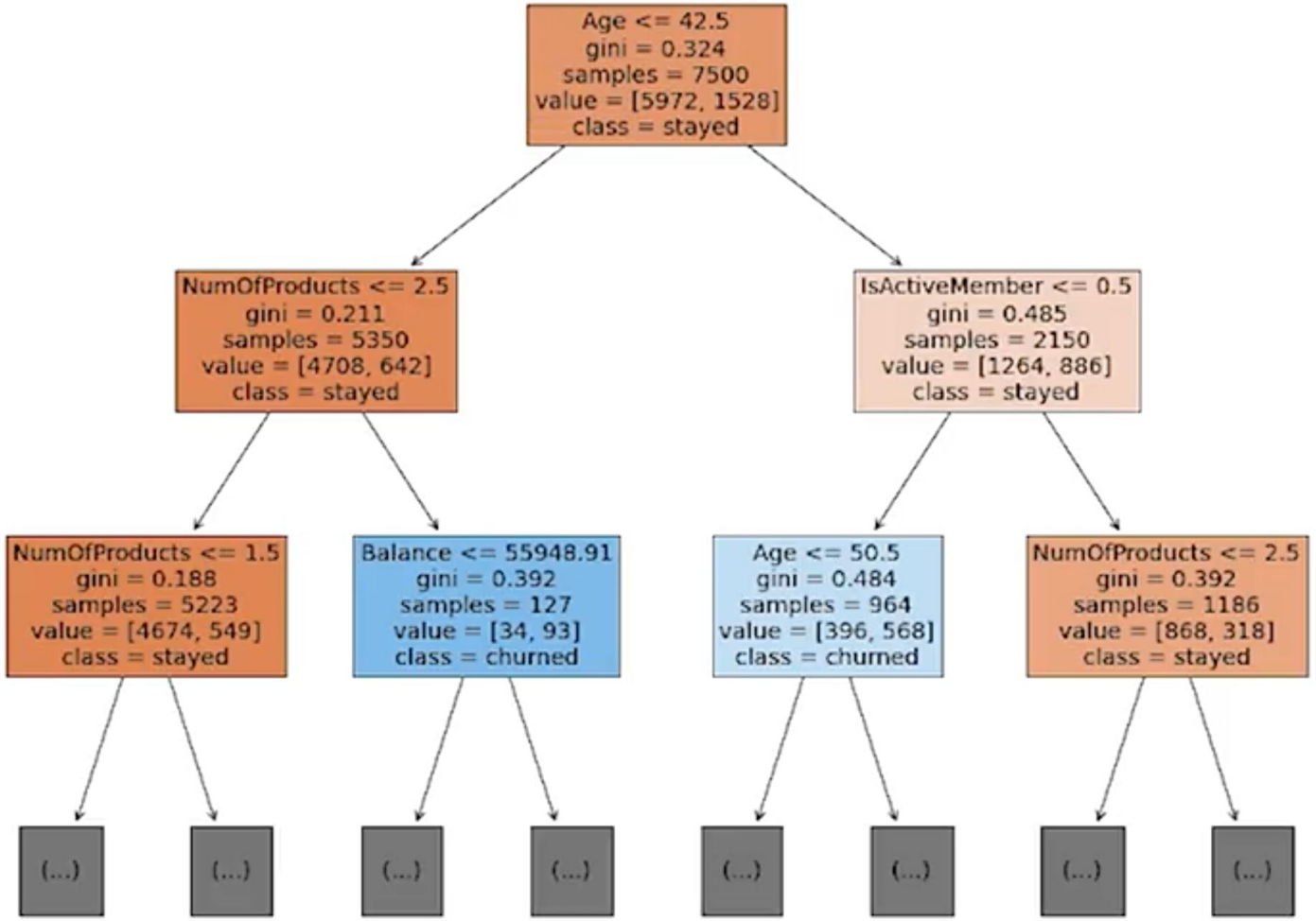
1. Upload the following files from the module learning room:
   - Jupiter notebook
   - "Lab_6-1_Decision_Tree_with_Python.ipynb"
   - Dataset csv-file "Bank_data_modelling.csv"
2. Follow along in the Jupiter notebook

Coding Activity 6-1. Evaluation Metrics

Assumption: A metric that balances precision and recall is best. The metric that helps a modeler achieve this balance is F_1 score:

$$F_1 = 2 \bullet \frac{Precision \bullet Recall}{Precision + Recall}$$

Coding Activity 6-1. Decision Tree



Purity of the node: Metrics

Metrics to determine the **purity of the node**:

- Gini impurity
- Entropy
- Information gain
- Log loss

Gini Impurity

$$GI = 1 - \sum_{i=1}^{N} P(i)^2 ; \; GI \in [0, 0.5]$$

where: i = class; P(i) = the probability of samples belonging to class i in a given node

Gini impurity (GI) = 0: No impurity - the node is a leaf and all of its samples are of a single class.

Gini impurity (GI) = 0.5: The classes are all equally represented in that node.

Ensemble Learning

Ensemble learning or 'ensembling' is a technique that enables a modeler to use multiple decision trees simultaneously in order to produce very powerful models: aggregating their outputs to make a prediction.

Base learner is an each ndividual model that comprises an ensemble.

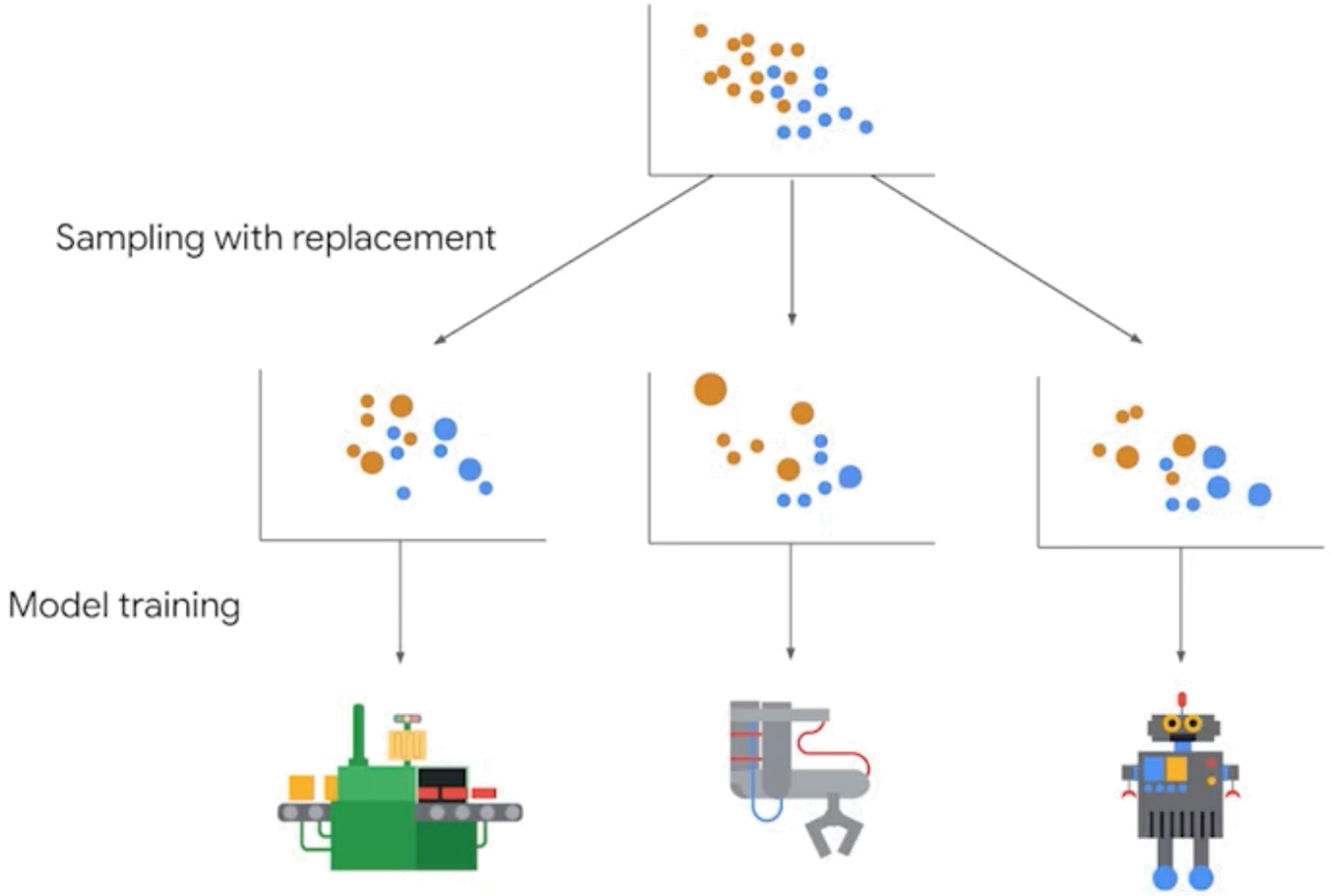
Bootstrat Aggregating (Bagging)

‘Bagging’ =

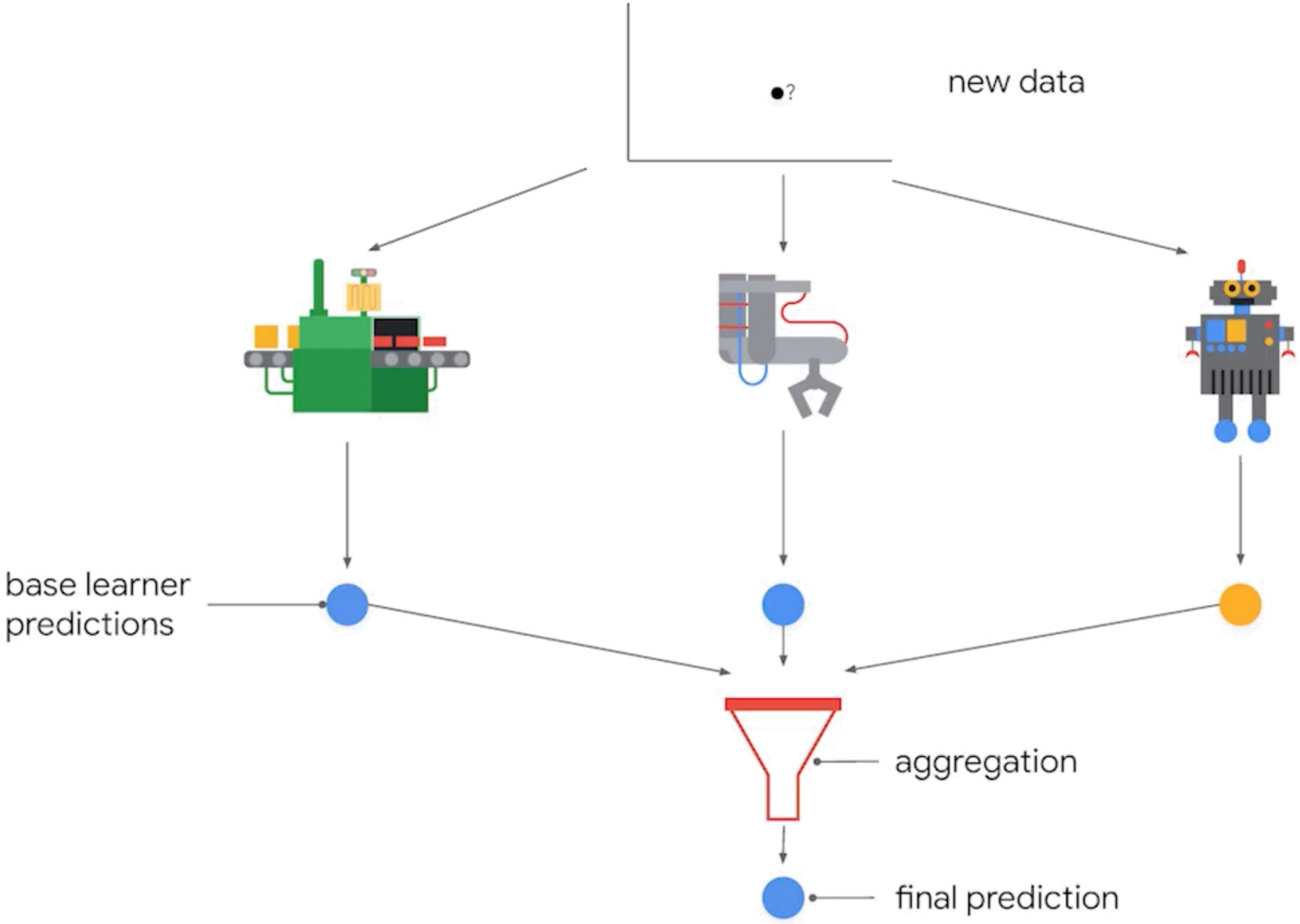
‘Bootstrap’ **+ ‘Aggregat****ing’**

Bootstrapping:

Sampling with replacement

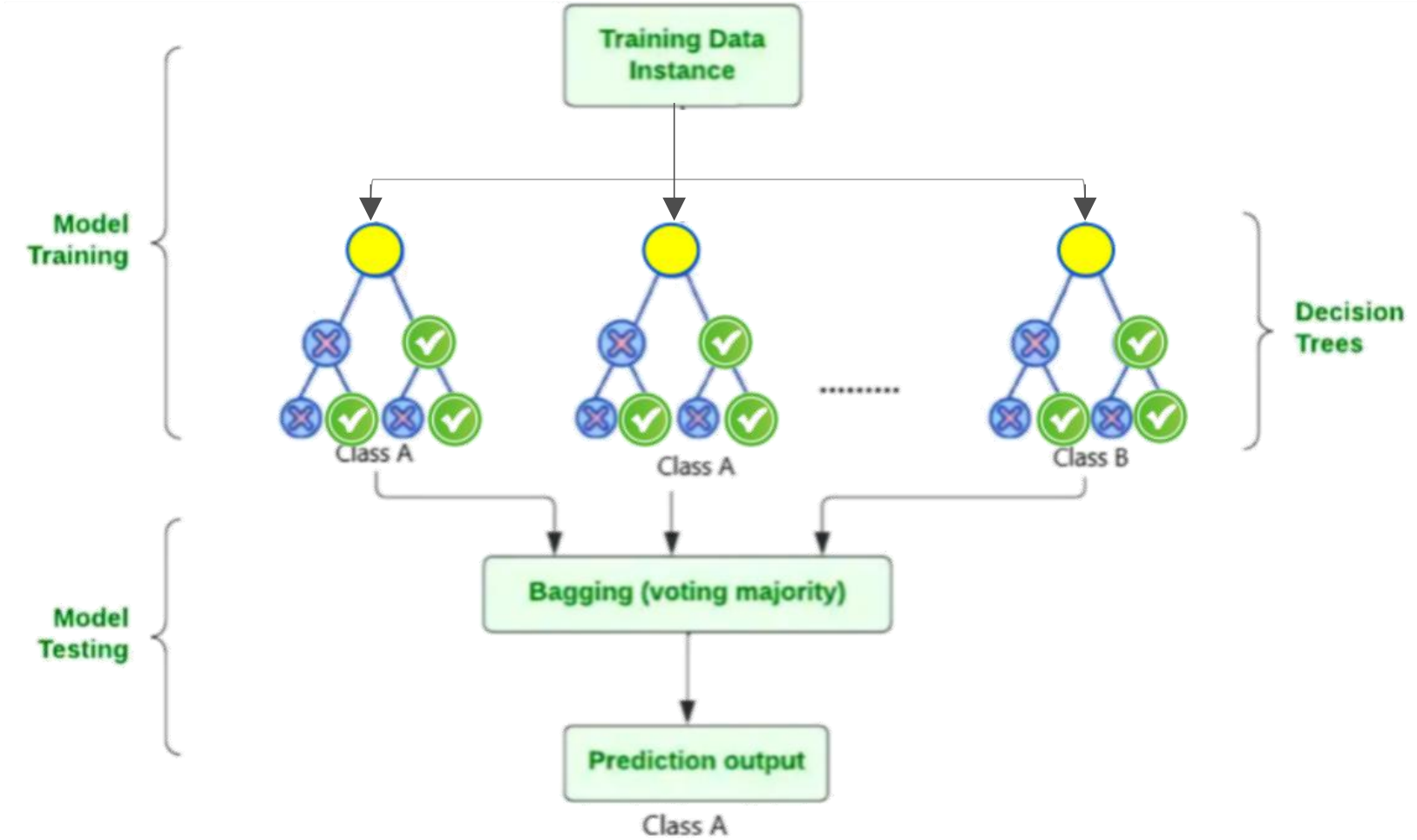


Bootstrat Aggregating (Bagging) (2)



Random Forest

Random Forest is an ensemble of decision trees trained on bootstrapped data with randomly selected features



Random Forest: Tunning

Coninue splittling until:

- leaf size is pure
- reach min leaf size or max depth
- performance metrics achieved

Max depth: defines how “long” a decision tree can get;

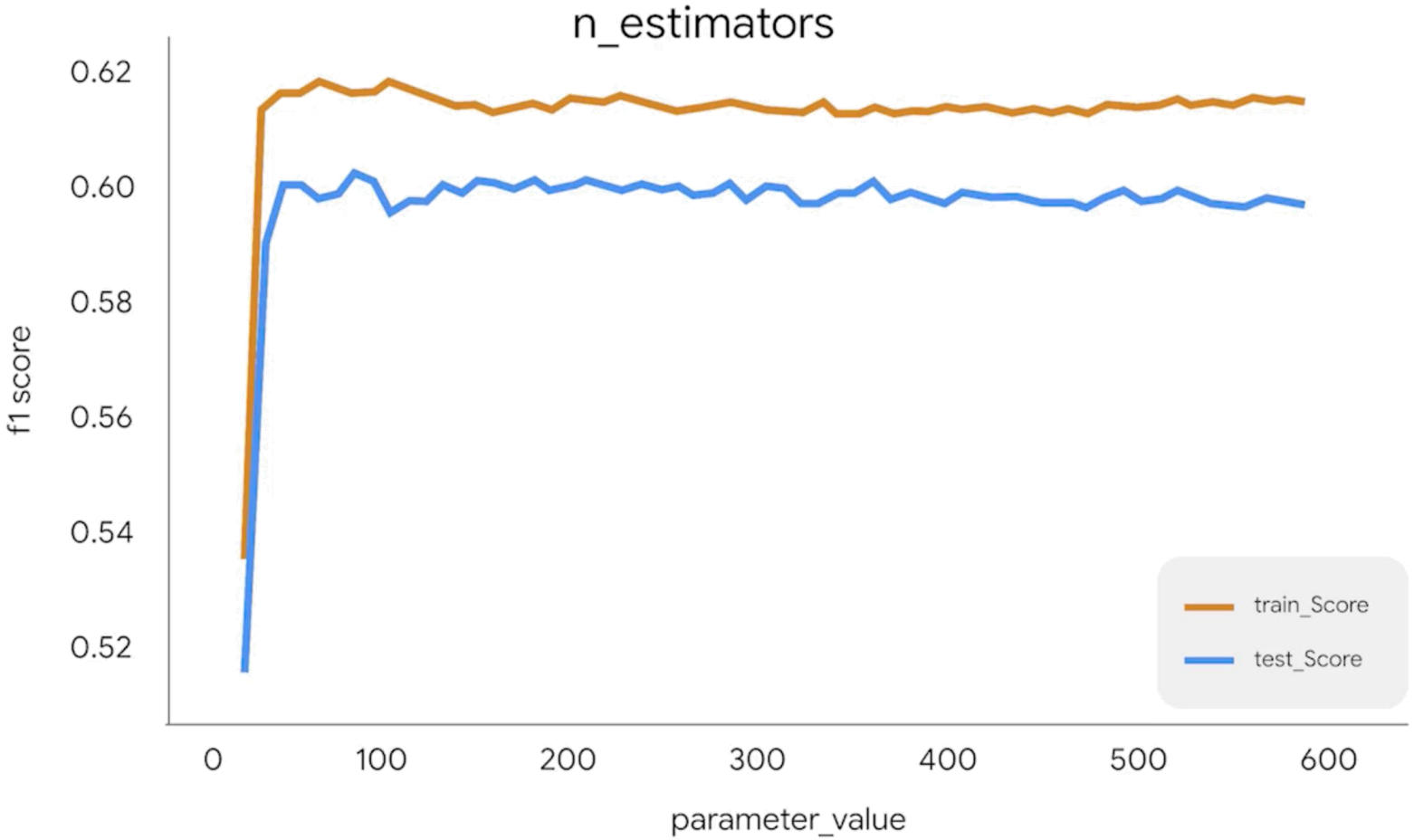
Min samples leaf: defines the minimum number of samples for a leaf node.

Random Forest: Tunning Parameters

max_features: specifies the number of features that each tree randomly selects during training

Random Forest: Tunning Parameters (2)

n_estimators: specifies the number of trees your model will build in its ensemble



Magic Commands in Python

Magic commands (“magics”) are commands that are built into Python to sumplify common tasks. Magic commands always begin with either ‘%’ or either ‘%%’.

%%time

- a magic command that gives you the runtime of the sell it’s entered in

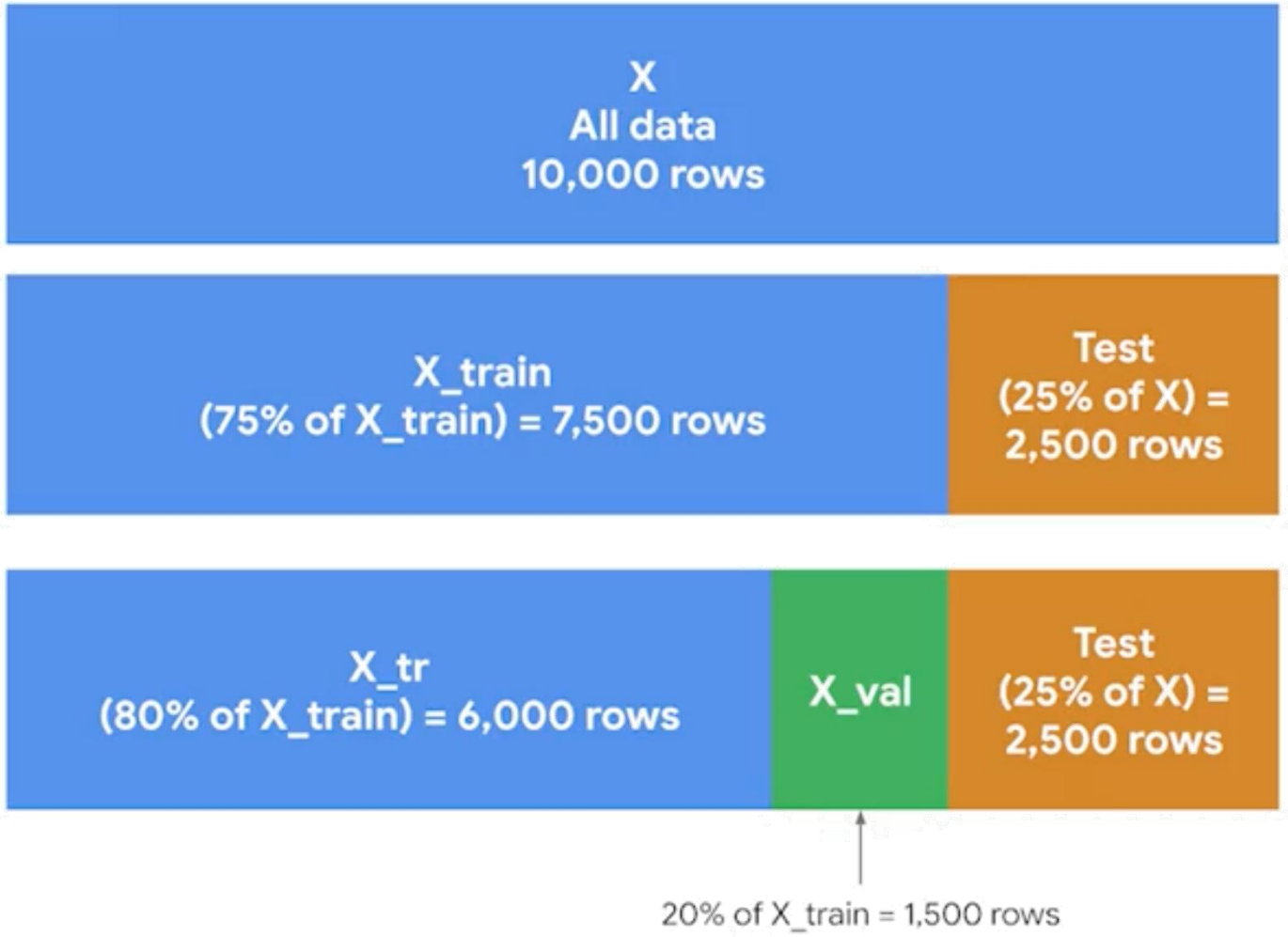
Coding Activity 6-2. Supervised ML. Random Forest

Lab 6-2. Supervised ML. Decision Tree || [Random Forest Model with Python (European Bank Data Modelling)]

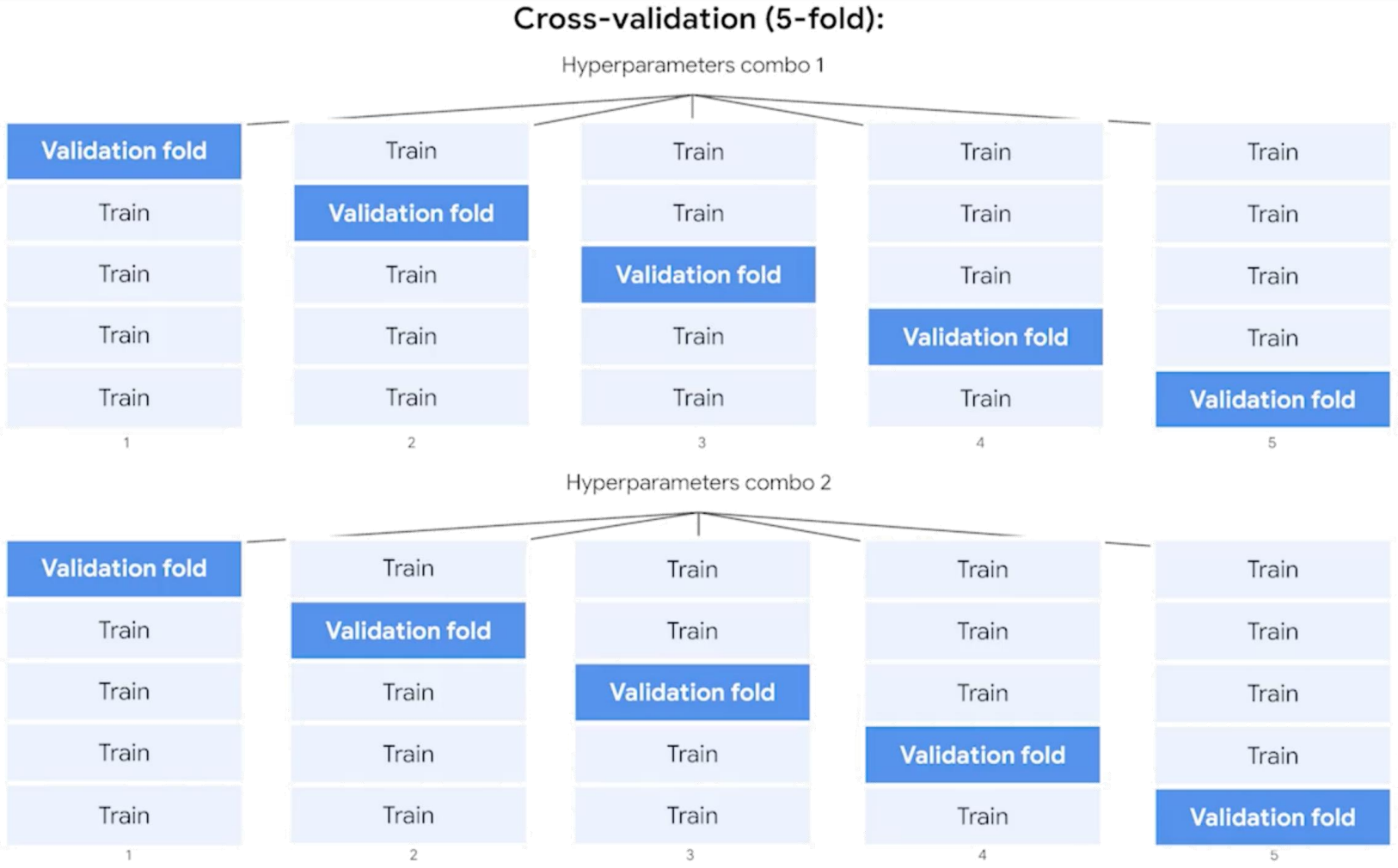
Steps to follow:

1. Upload the following files from the module learning room:
   - Jupiter notebook
     "Lab_6-2_Random_Forest_Model_with_Python.ipynb"
   - Dataset csv-file "Bank_data_modelling.csv"
2. Follow along in the Jupiter notebook

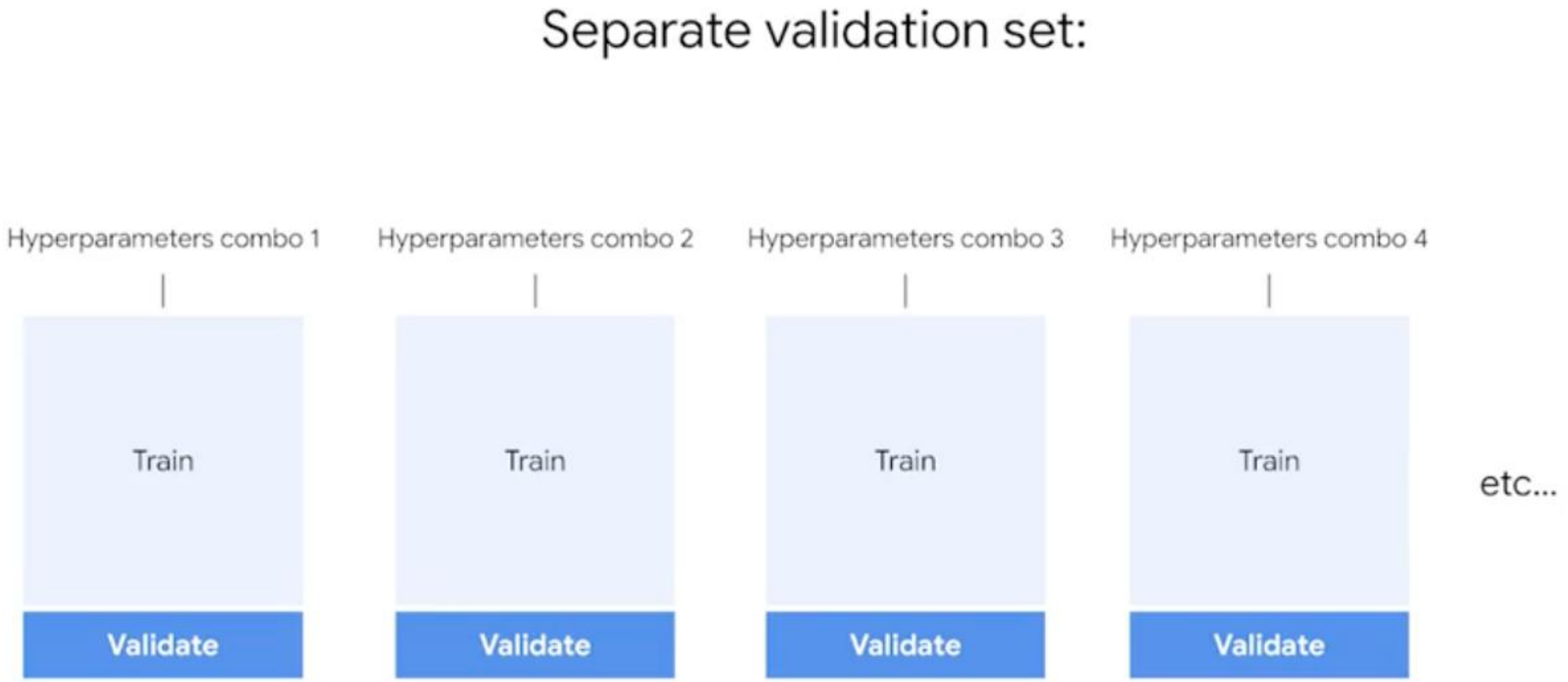
Coding Activity 6-2. Supervised ML. Random Forest



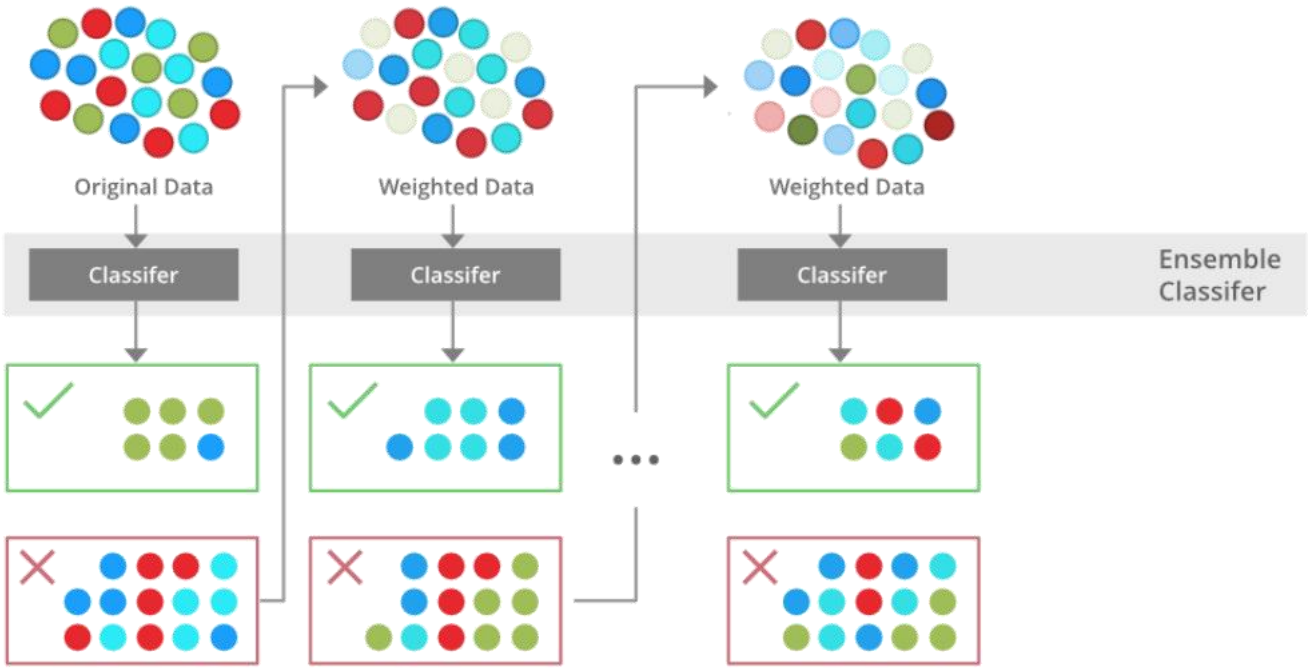
Coding Activity 6-2. Supervised ML. Random Forest



Coding Activity 6-2. Supervised ML. Random Forest



Boosting



Boosting is a technique that builds an ensemble of weak learners sequentially, with each consecutive learner trying to correct the errors of the one that preceded it

Boosting: Comparison to Radom Forest

Similarities to random forest and bagging:

- Ensemble technique
- Aggregates weak learners

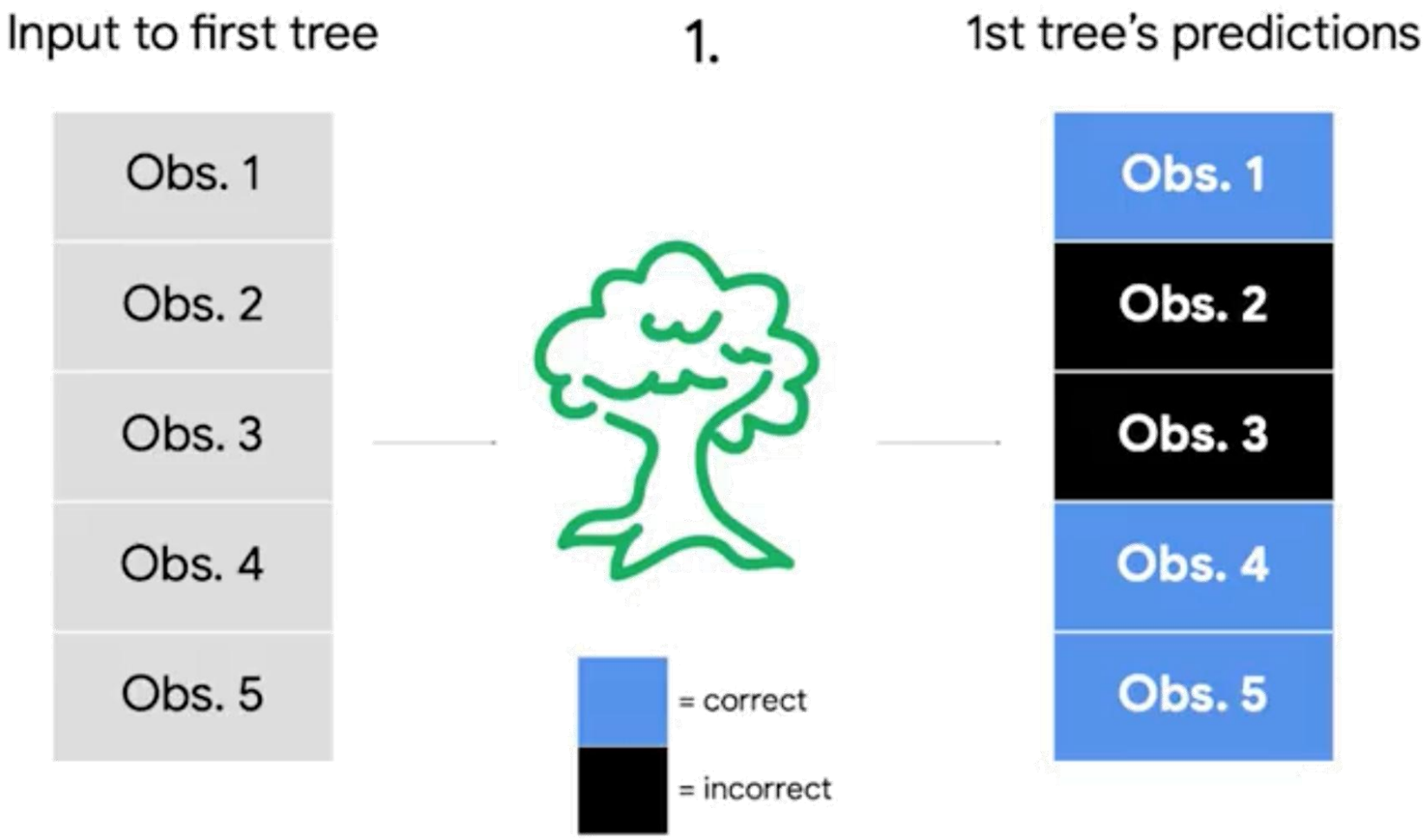
Differences from random forest and bagging:

- Learners are built sequentially, not in parallel
- Not limited to tree-based learners

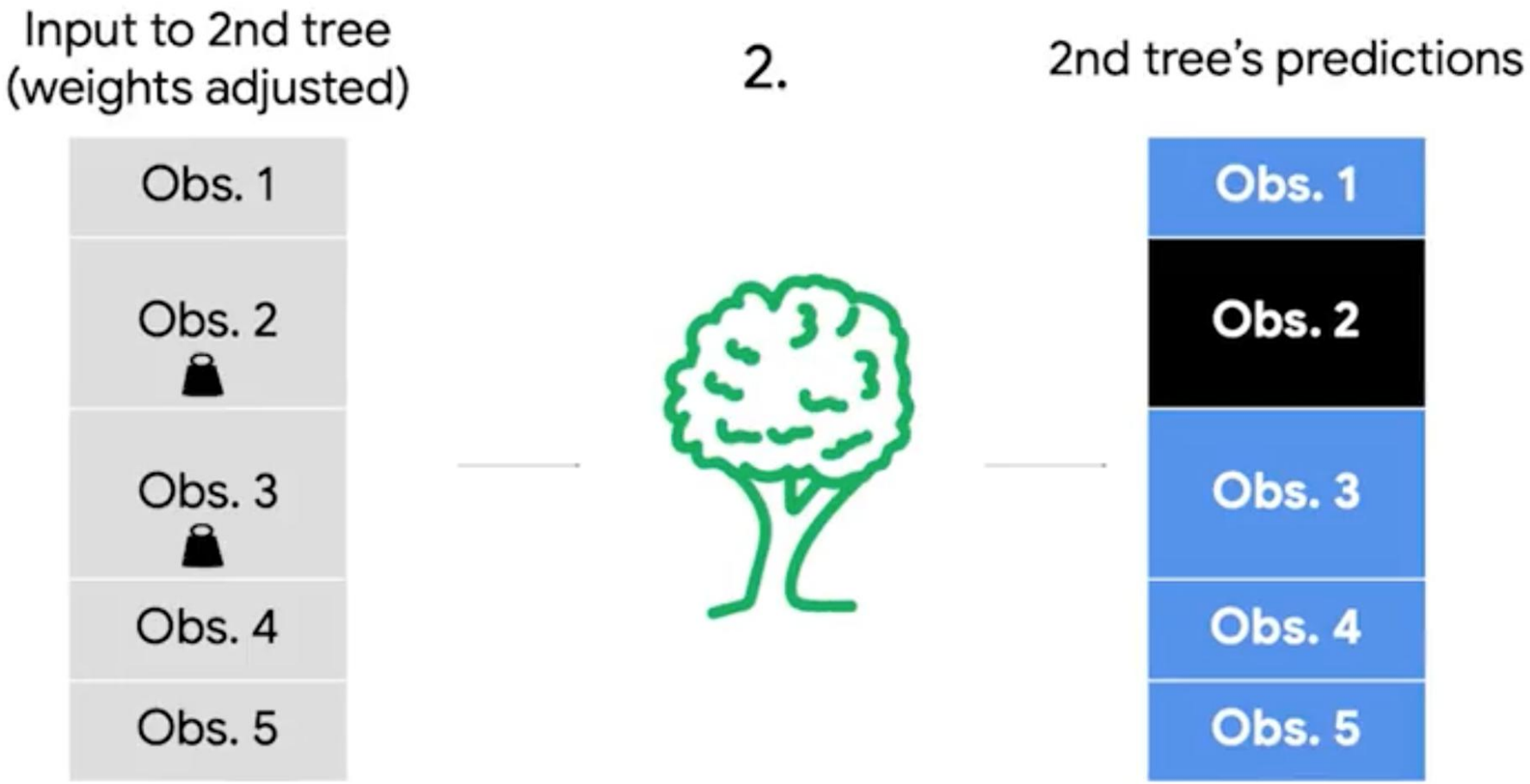
Adaptive Boosting (AdaBoost)

Adaptive boosting (AdaBoost) is a boosting methodology where each consecutive base learner assigns greater weight to the observations incorrectly predicted by the predicting learner

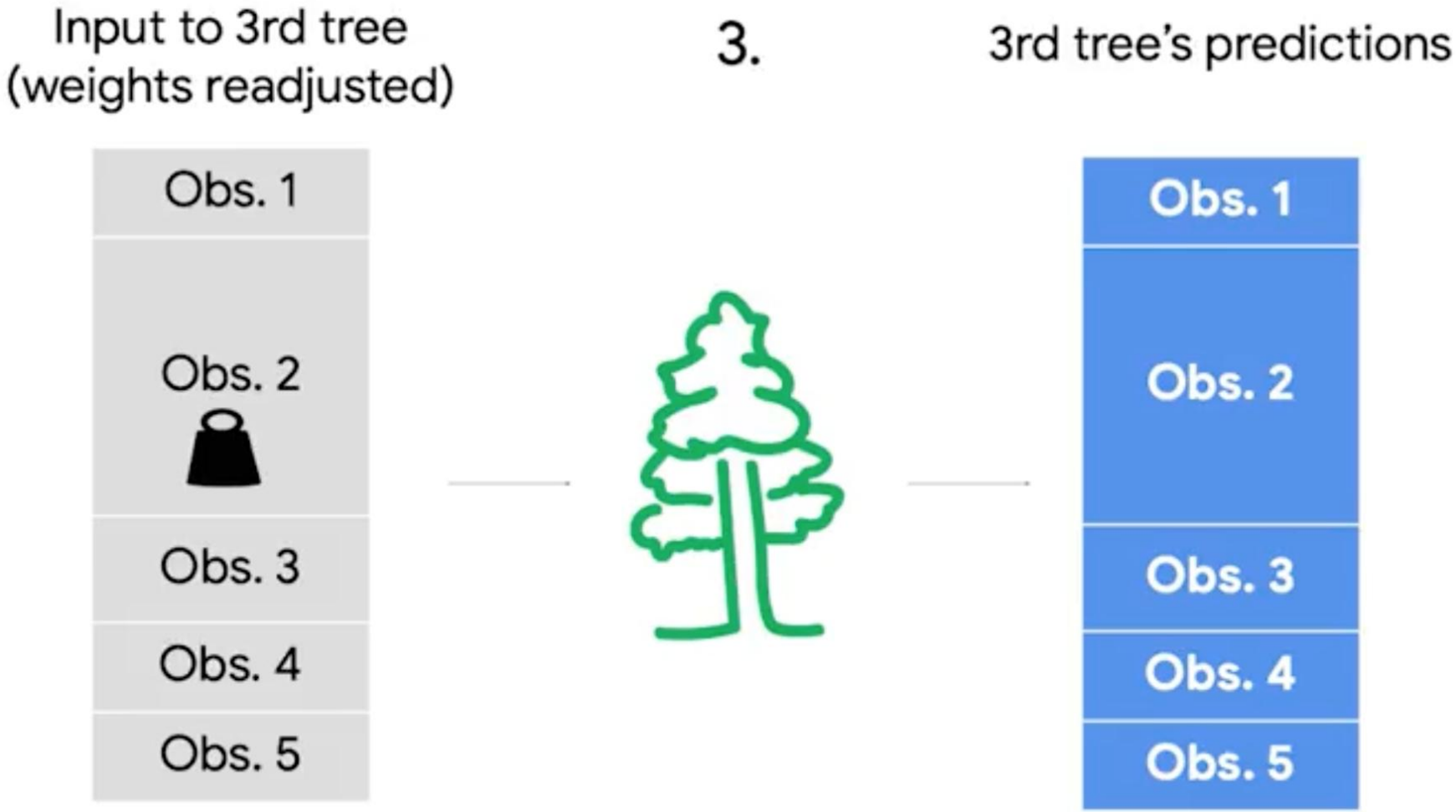
Example 5. Adaptive Boosting (AdaBoost)



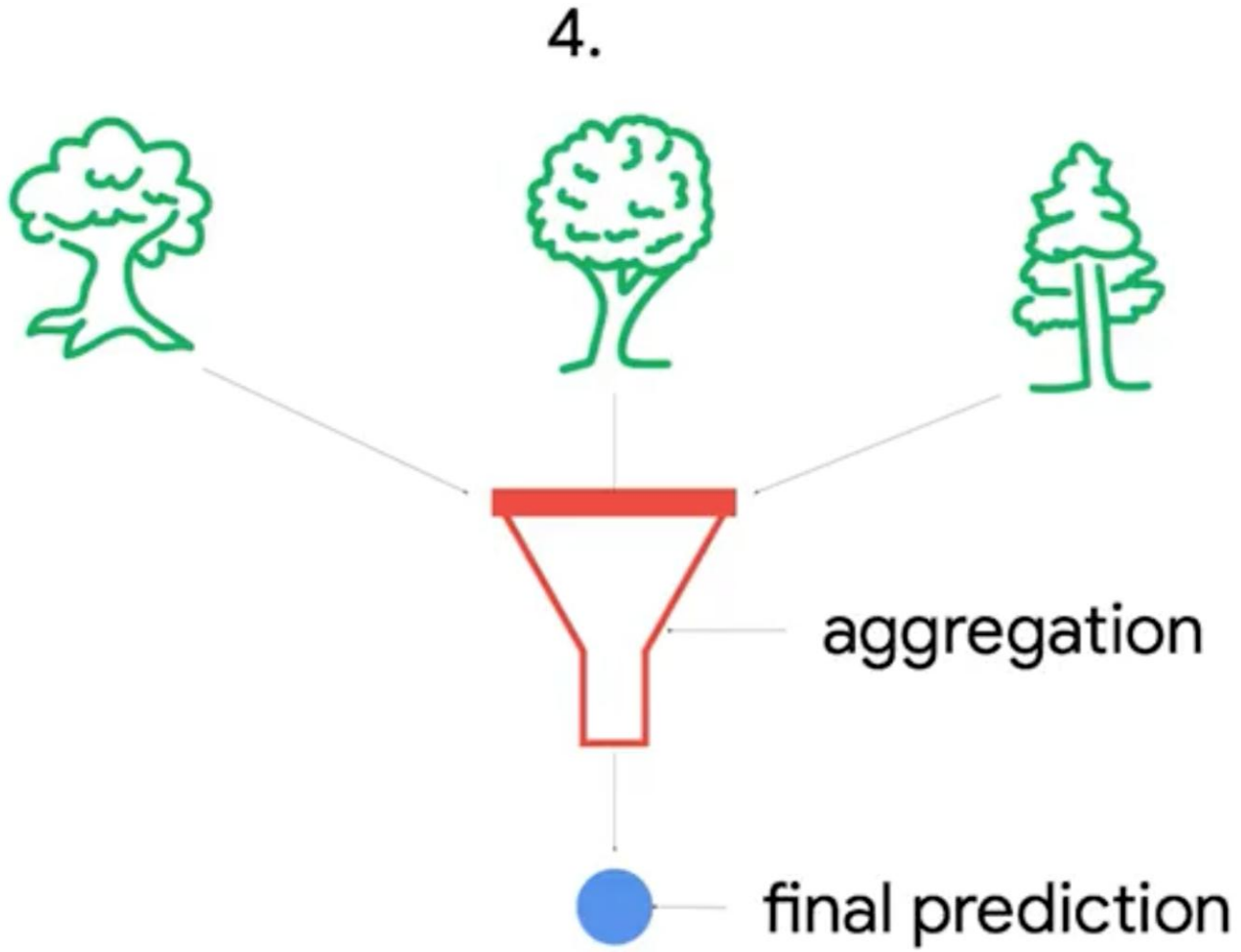
Example 5. Adaptive Boosting (AdaBoost) (2)



Example 5. Adaptive Boosting (AdaBoost) (3)

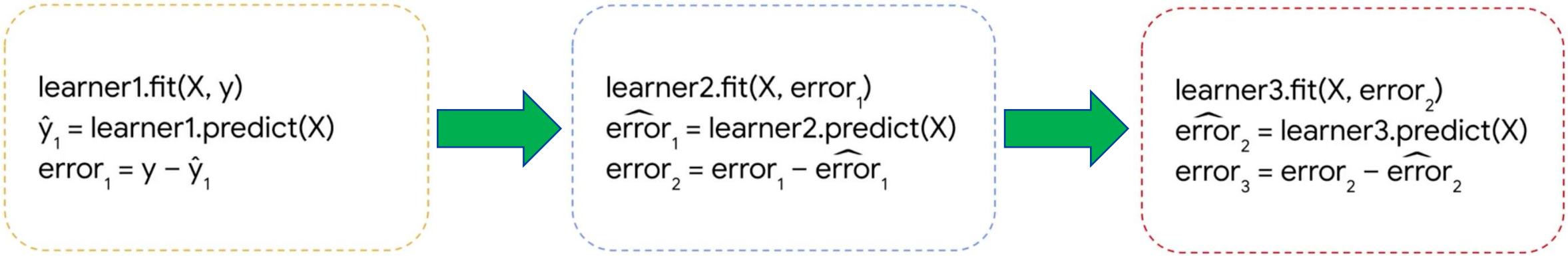


Example 5. Adaptive Boosting (AdaBoost) (4)



Gradient Boosting

Gradient boosting is a boosting methodology where each consecutive base learner in sequence is built to predict the residual errors of the model that preceded it

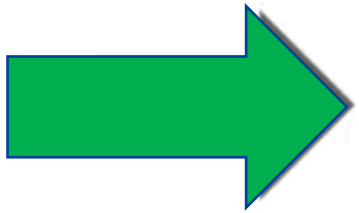


Example 6. Gradient Boosting

learner1.fit(X, y)
$\hat{y}_1$ = learner1.predict(X)
$error_1 = y - \hat{y}_1$

learner2.fit(X, $error_1$)
$\widehat{error}_1$ = learner2.predict(X)
$error_2 = error_1 - \widehat{error}_1$

learner3.fit(X, $error_2$)
$\widehat{error}_2$ = learner3.predict(X)
$error_3 = error_2 - \widehat{error}_2$



Final prediction =

$learner1.predict(X_{new})$

+

$learner2.predict(X_{new})$

+

$learner3.predict(X_{new})$

Gradient Boosting Machines (GBMs)

Gradient boosting machines are model ensembles that use gradient boosting

Advantages	Disadvantages
+ High accuracy	- Tuning many parameters can be time-consuming
+ Generally scalable	- Difficult to interpret
+ Work well with missing data	- Have difficulty with extrapolation
+ Don't require scaling	- Prone to overfitting if too many parameters are tuned
+ Can handle outliers easily	

Black Box Model and Extrapolation

Black Box Model any model whose predictions cannot be precisely explained

Extrapolation is a model's ability to predict new vaues that fall outside of the range of values in the training data

Q&A

Thank you!