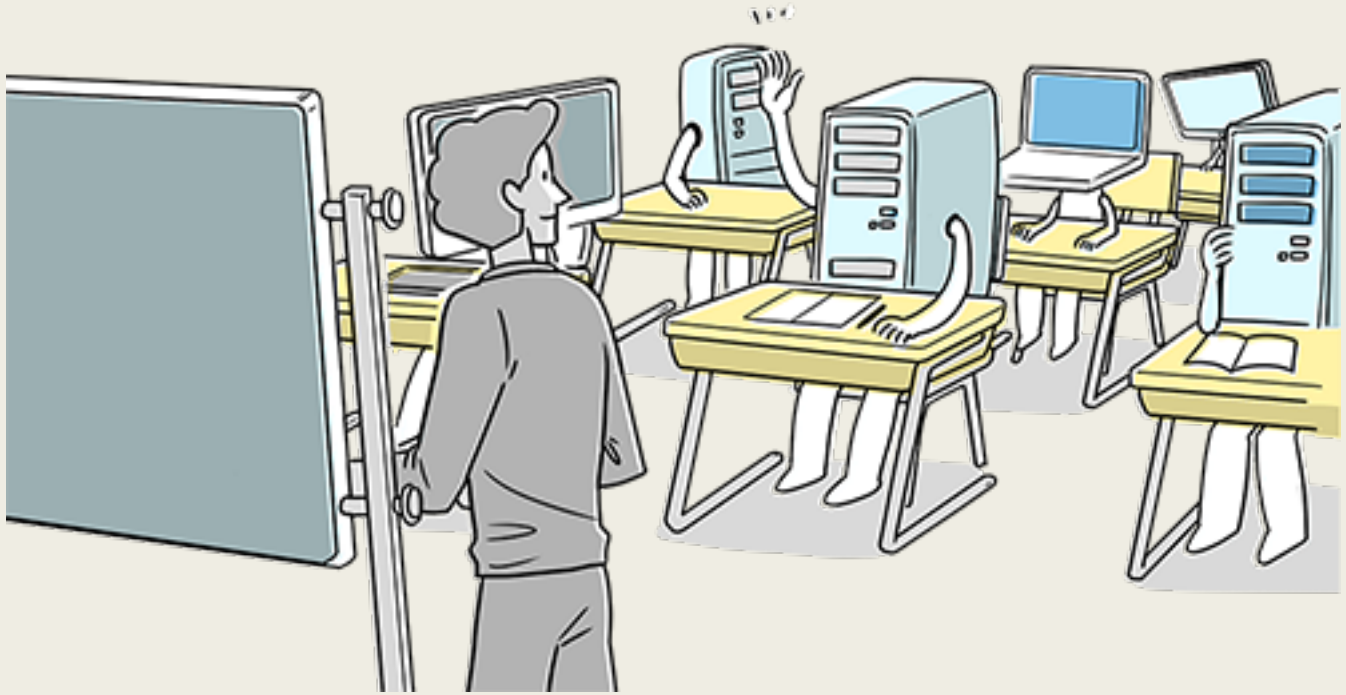
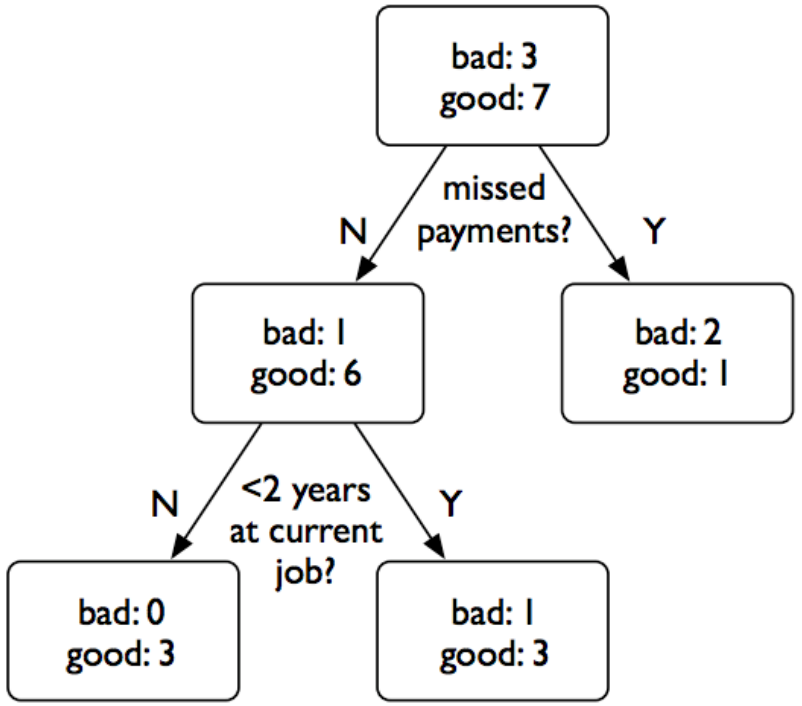


DECISION TREES

Decision trees: Classifying from a set of attributes

Predicting credit risk

<2 years at current job?	missed payments?	defaulted?
N	N	N
Y	N	Y
N	N	N
N	N	N
N	Y	Y
Y	N	N
N	Y	N
N	Y	Y
Y	N	N
Y	N	N



- Each level splits the data according to different attributes
- **Goal**: achieve perfect classification with minimal number of decisions
  - not always possible due to noise or inconsistencies in the data

Decision trees with continuous values

Predicting credit risk

years at current job	# missed payments	defaulted?
7	0	N
0.75	0	Y
3	0	N
9	0	N
4	2	Y
0.25	0	N
5	1	N
8	4	Y
1.0	0	N
1.75	0	N

- Turn continuous values into binary values
- Decision Boundary is **non-linear** !!

Decision Trees

- Any Boolean function can be represented by a decision tree.

- General cases:
  - *Input: binary, multi-valued, or continuous*
  - *output: binary, multi-valued or continuous (regression trees)*

- Best when a small number of attributes provide a lot of information

- Very easy to interpret

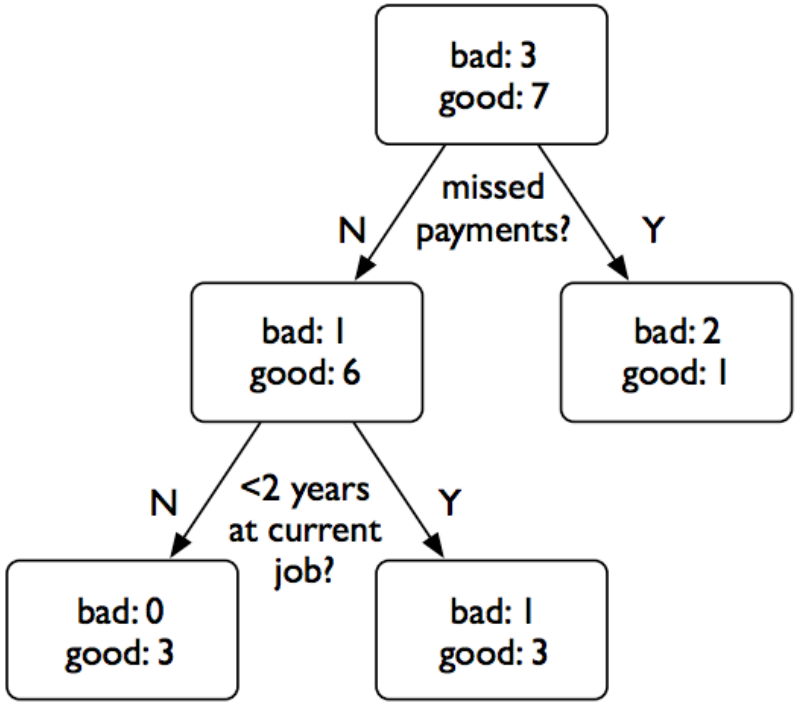
- Work well for non-linear, complex relationship between the features and the response

Decision trees: Learning

- How do we which attribute or value to split on?
- When should we stop splitting?
- What do we do when we can't achieve perfect classification?
- What if tree is too large? Can we approximate with a smaller tree?

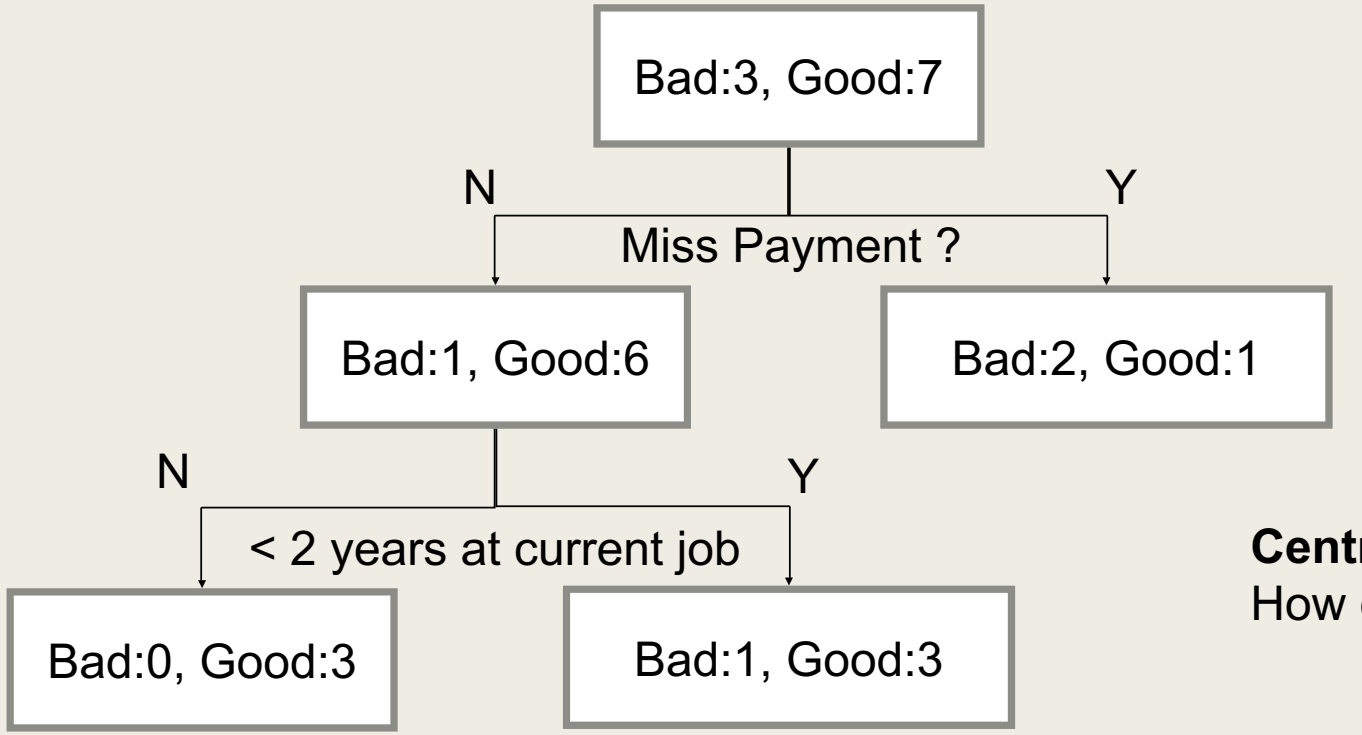
Predicting credit risk

<2 years at current job?	missed payments?	defaulted?
N	N	N
Y	N	Y
N	N	N
N	N	N
N	Y	Y
Y	N	N
N	Y	N
N	Y	Y
Y	N	N
Y	N	N



Decision trees: Learning

1. Starting with whole training data
2. Select attribute or value along feature that gives “best” split
3. Create child nodes based on split
4. Repeat on each child using child data until a stopping criterion is reached



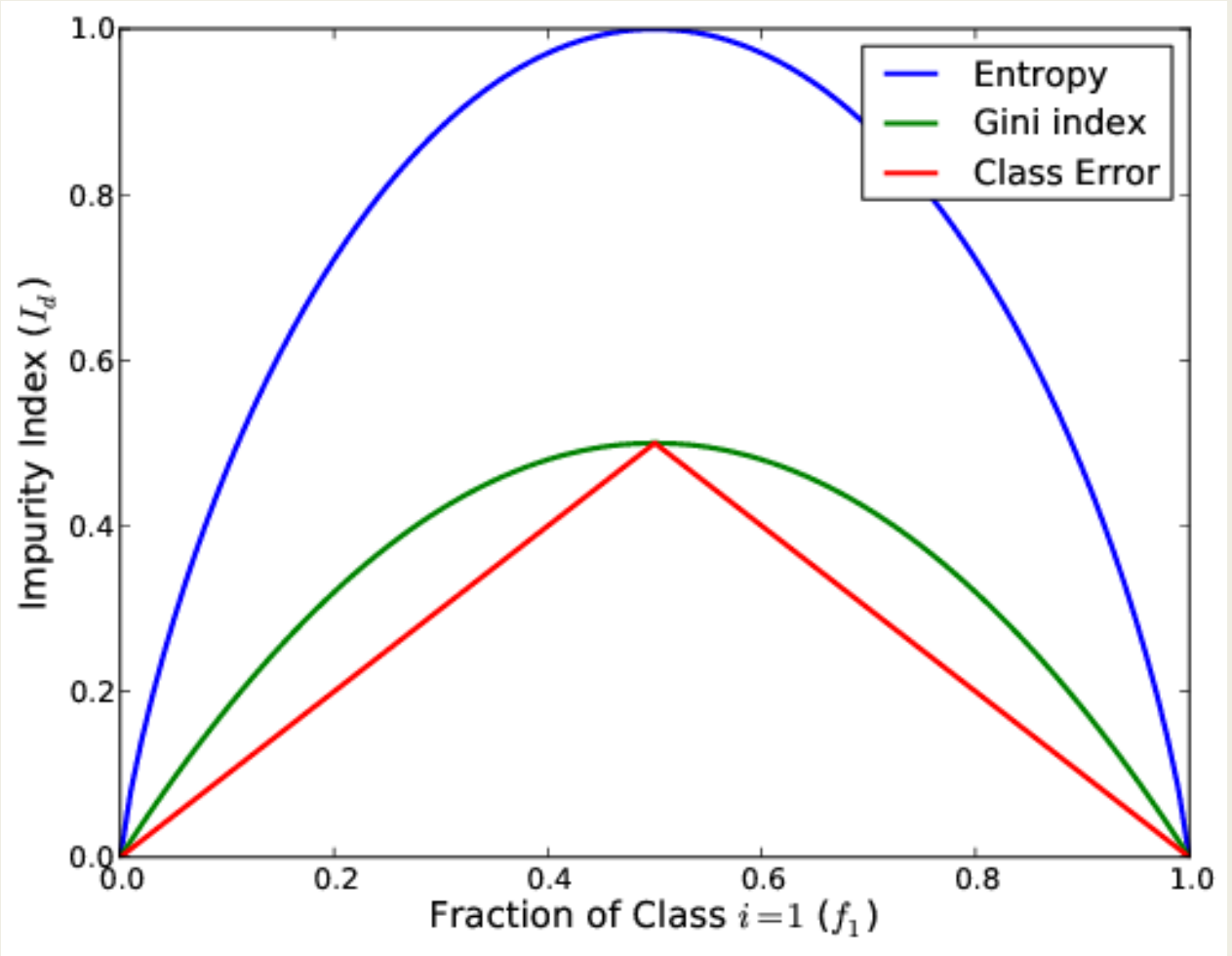
Central problem:
How do we choose the “best” attribute?

Splitting rules

- At each split, we must decide which variable to use for splitting
- This decision is made with reference to a measure of **node impurity**, or how much variation there is upon choosing a split
- How much “information” does an attribute give us about the class?
  - *attributes that perfectly partition should given maximal information*
  - *unrelated attributes should give no information*

Splitting rules

- Classification Trees
  - Gini Impurity
  - Entropy
  - Misclassification Error



- Regression Trees
  - Variance of target value within the node

Splitting Binary Attributes (using Gini)

Example :

	Parent
C0	6
C1	6
Gini = 0.5	

Gini :
$1 - (6/12)^2 - (6/12)^2$
= 0.5

Suppose there are two ways (A and B) to split the data into smaller subset.

A

	N1
C0	4
C1	3

Gini Index: 0.4898

	N2
C0	2
C1	3

Gini Index: 0.480

B

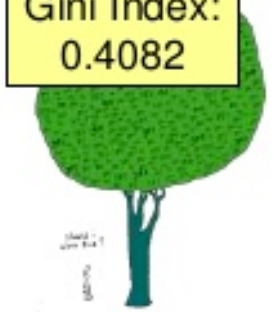
	N1
C0	1
C1	4

Gini Index: 0.320

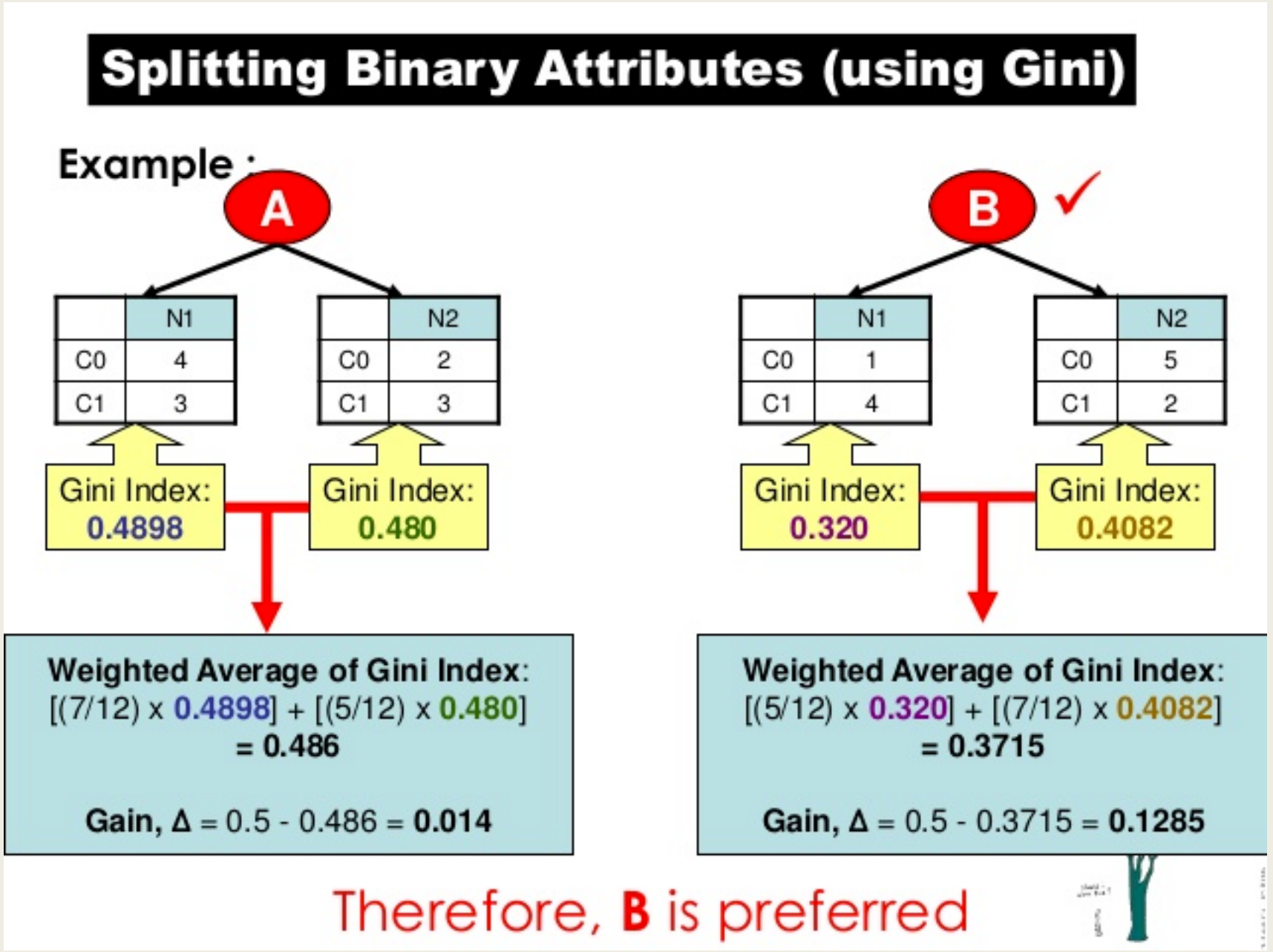
	N2
C0	5
C1	2

Gini Index: 0.4082

Which one is a better split??
Compute the **weighted average of the Gini index** of both attribute

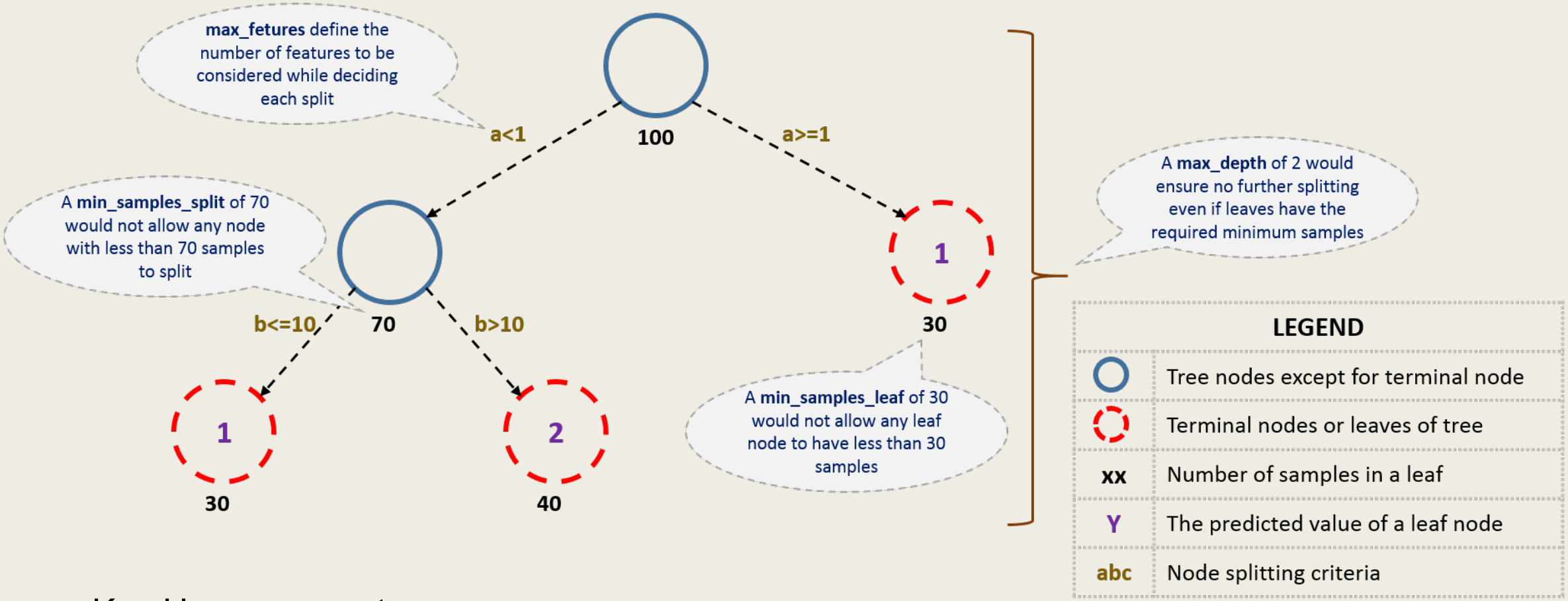


Source: **https://www.slideshare.net/knottisme/classification-using-decision-tree-53984611**

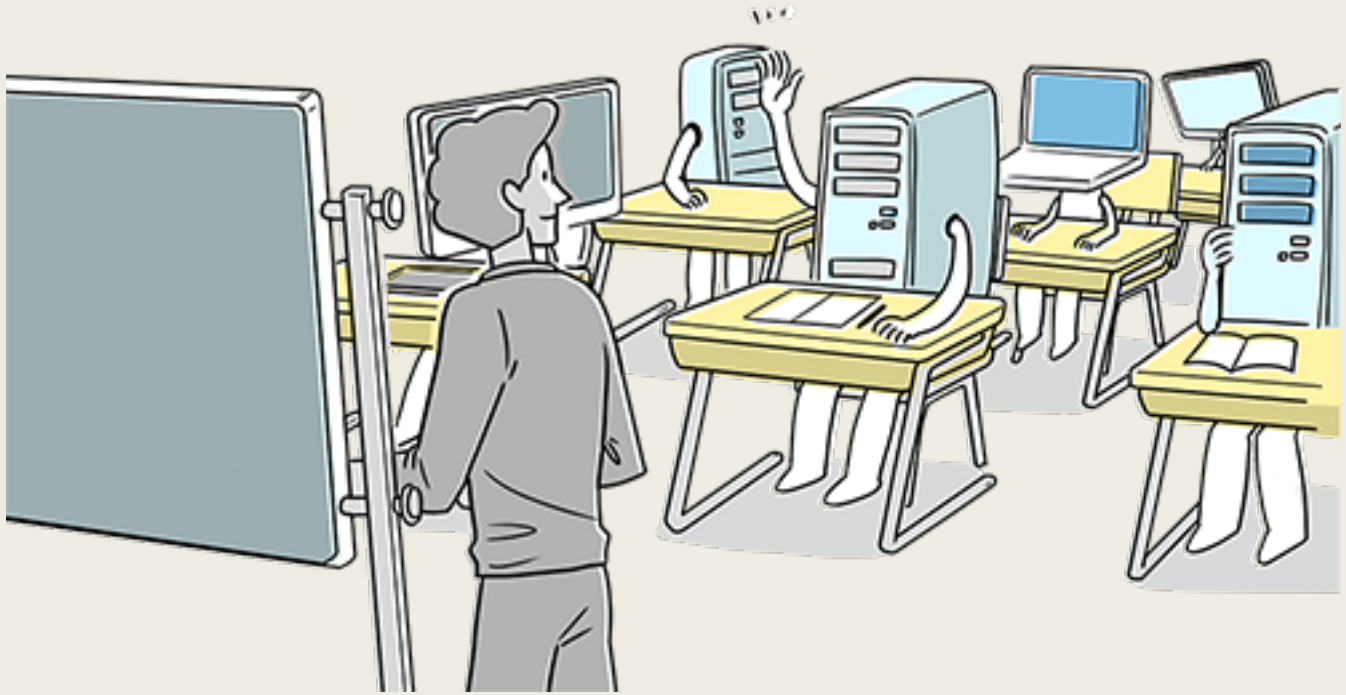


Source: https://www.slideshare.net/knottisme/classification-using-decision-tree-53984611

Hyper-parameters

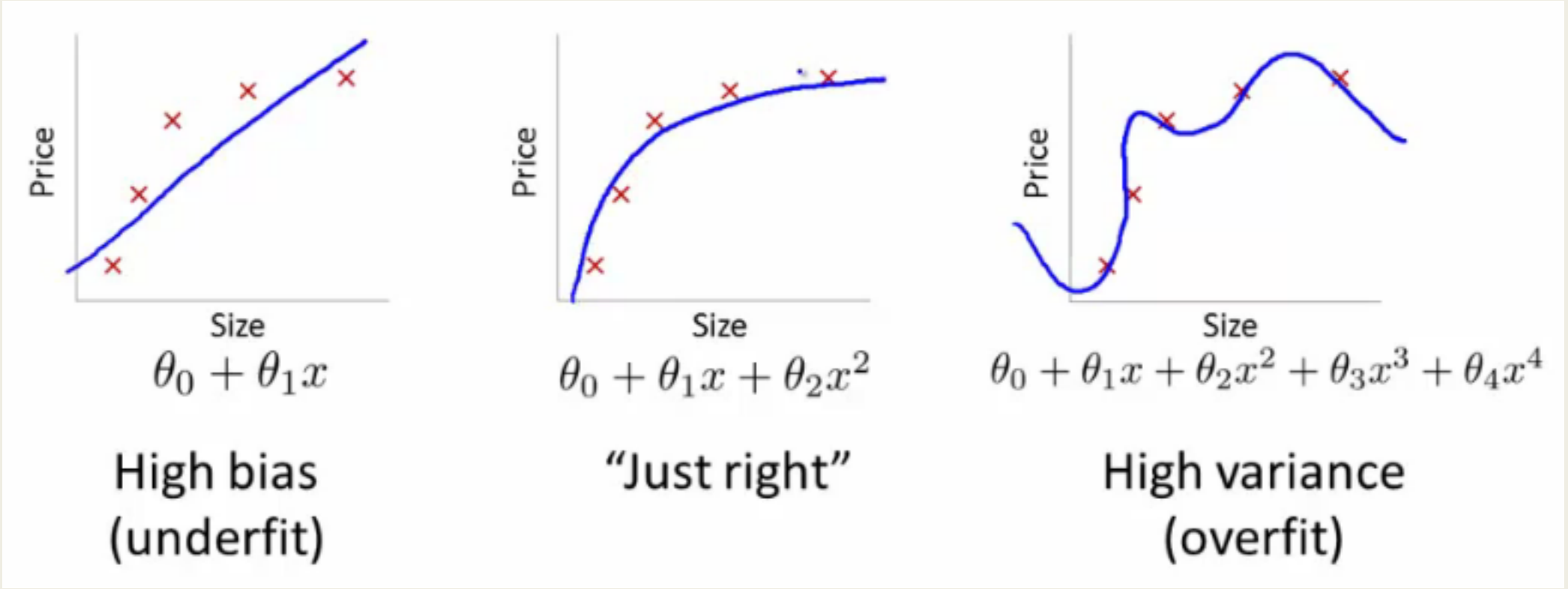


- Key Hyperparameters
  1. Max depth of the trees
  2. Min samples leaf



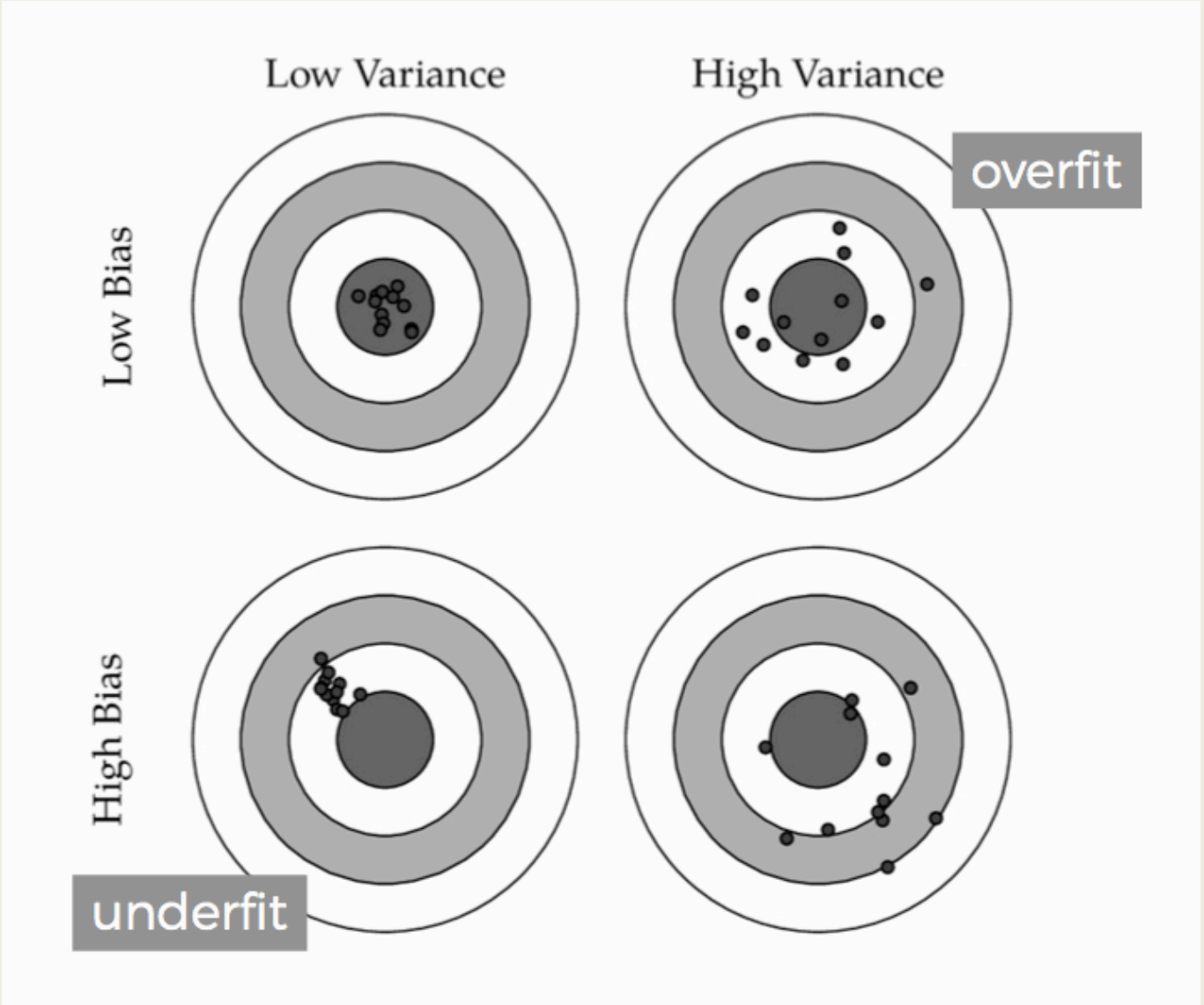
MODEL SELECTION AND HYPER-PARAMETER TUNING

Model Evaluation: Overfitting

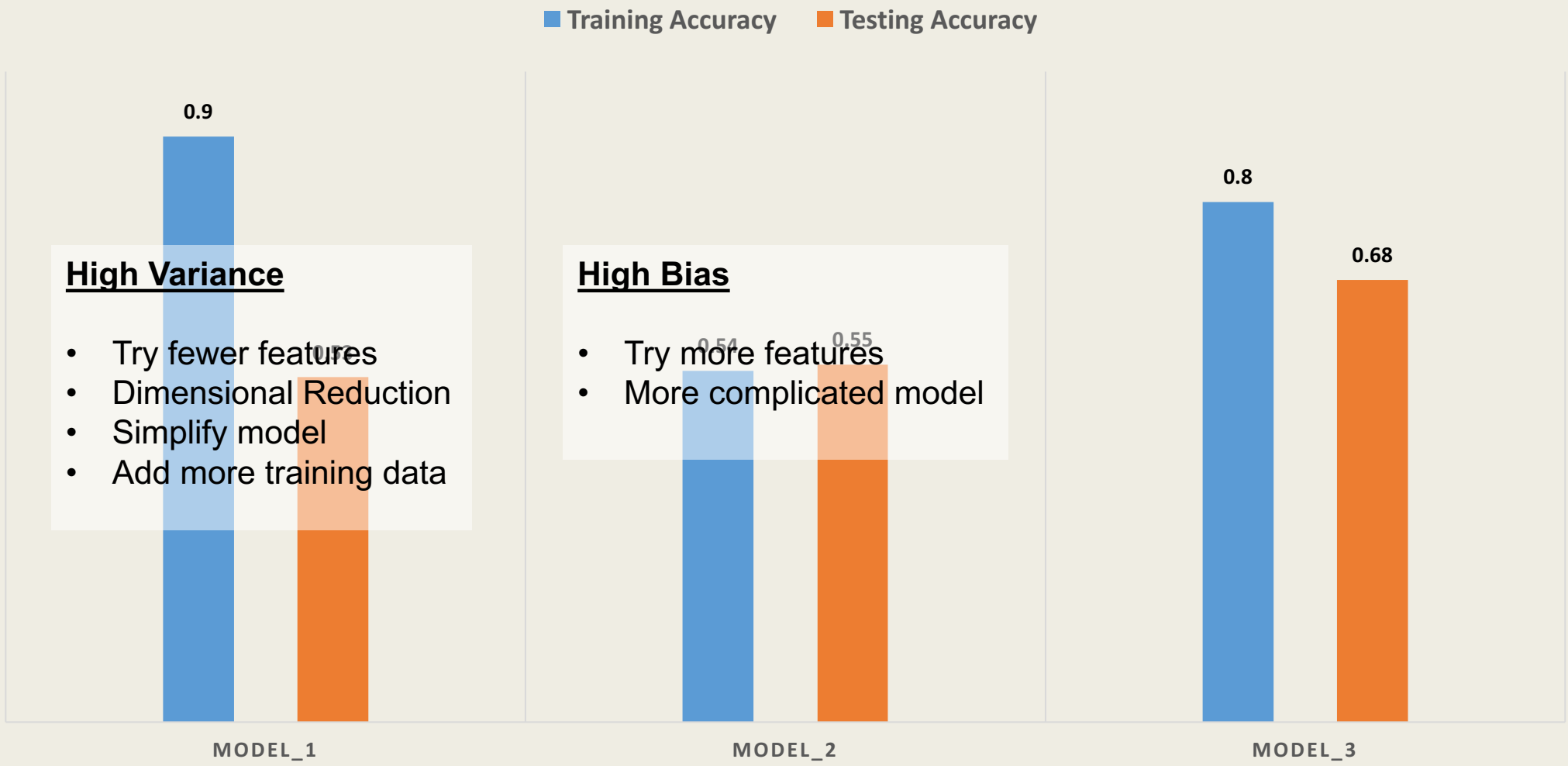


Bias-Variance Tradeoff

Model Evaluation: Bias Variance Trade-Off



Bias Variance Trade-Off



High Variance

- Try fewer features
- Dimensional Reduction
- Simplify model
- Add more training data

High Bias

- Try more features
- More complicated model

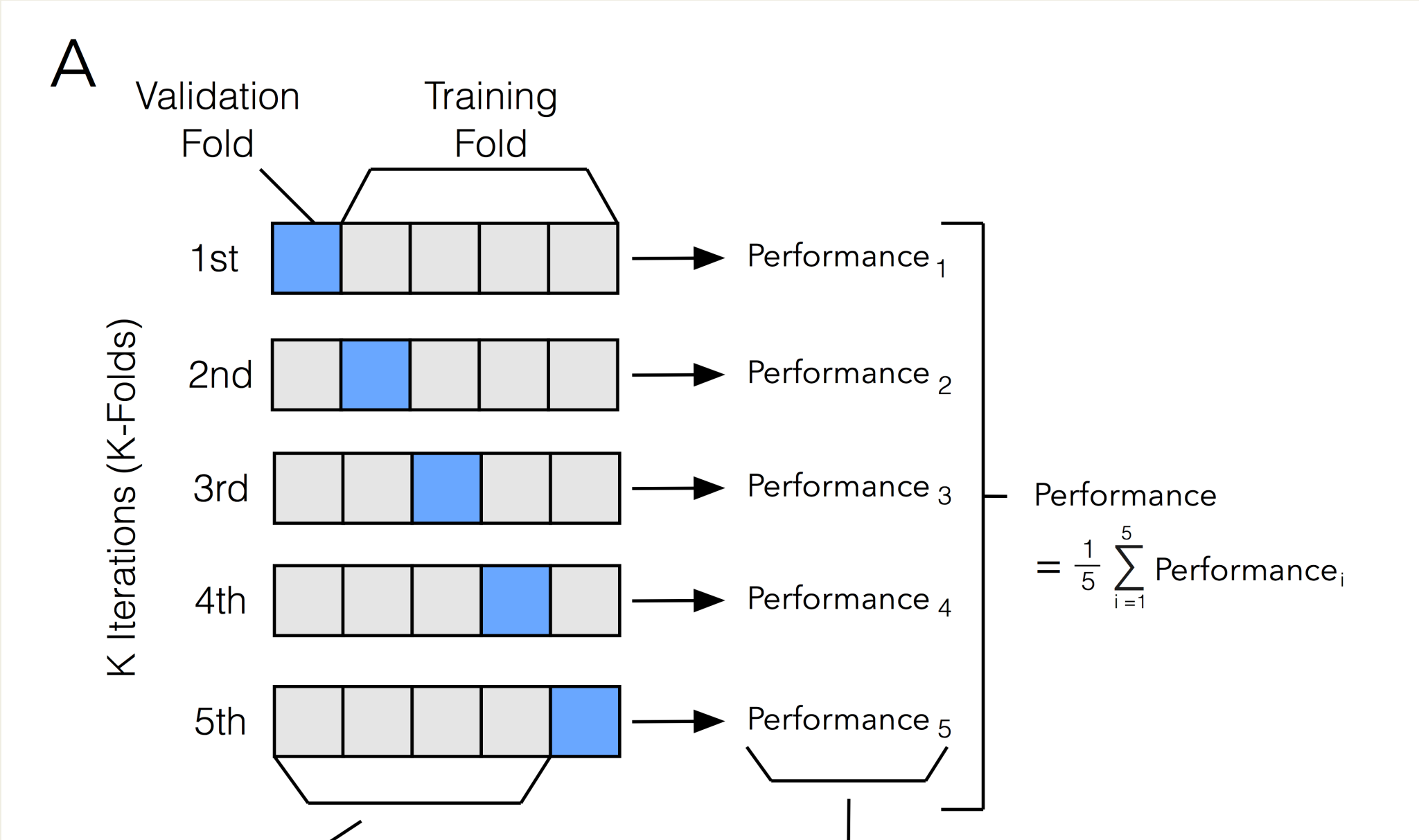
Estimating Test Errors

- We are interested in how well our model predicts the response on a new observation
- However, we don’t always have a large designated test set
- K-fold Cross Validation

K-fold Cross Validation

- Divide the set of observations into k groups, or folds
- At each iteration, one fold is treated as a validation set, and the method is fit on the remaining k − 1 folds. Repeat the process k times

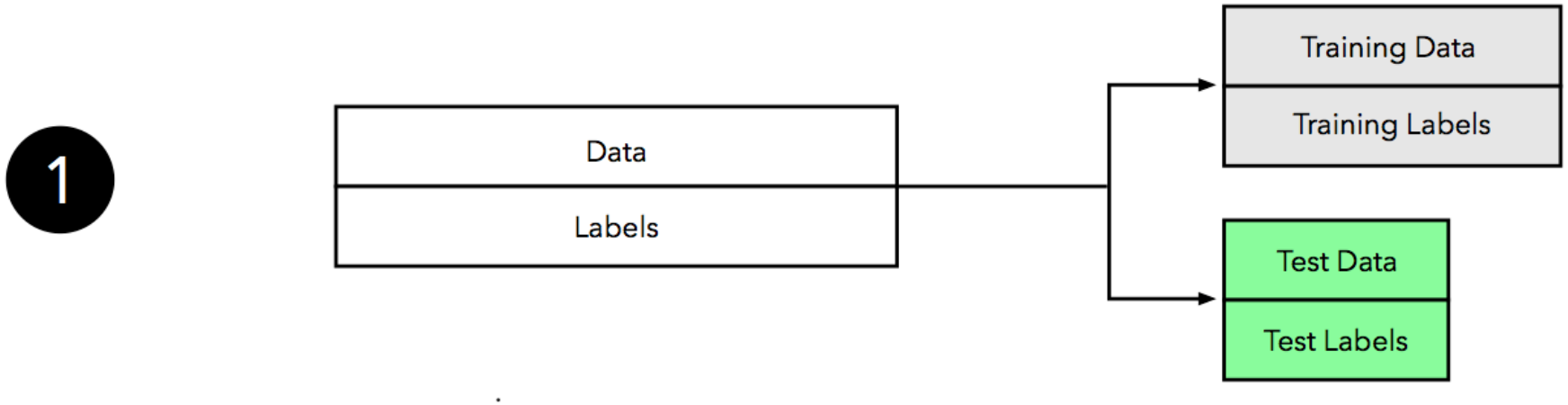
K-fold Cross Validation



Hyper-parameter Tuning

- Decision Trees Hyper-parameters
  - Max depth of the trees
  - Min samples leaf
- Logistic Regression Hyper-parameters
  - Regularization Parameter (lambda)
- The process of finding the best-performing model from a set of models that were produced by different hyper-parameter settings is called *model selection*

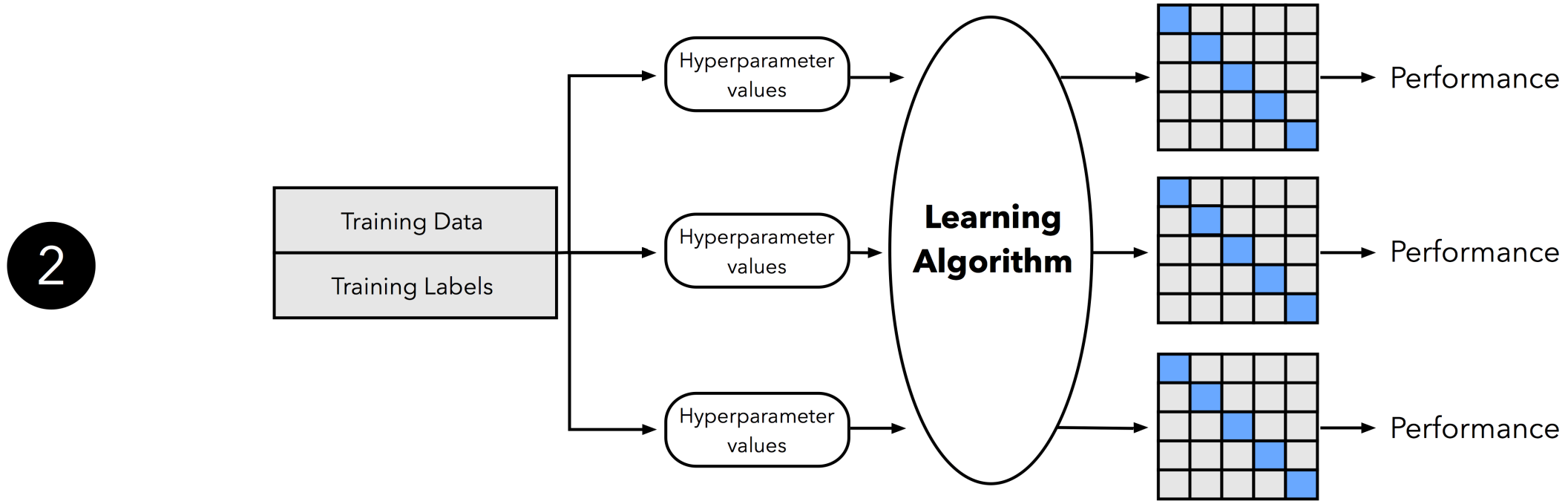
Model Selection



Step1: Train-Test Splitting

- We split our dataset into two parts, a training and an independent test set
- We tuck away the test set for the final model evaluation step at the end

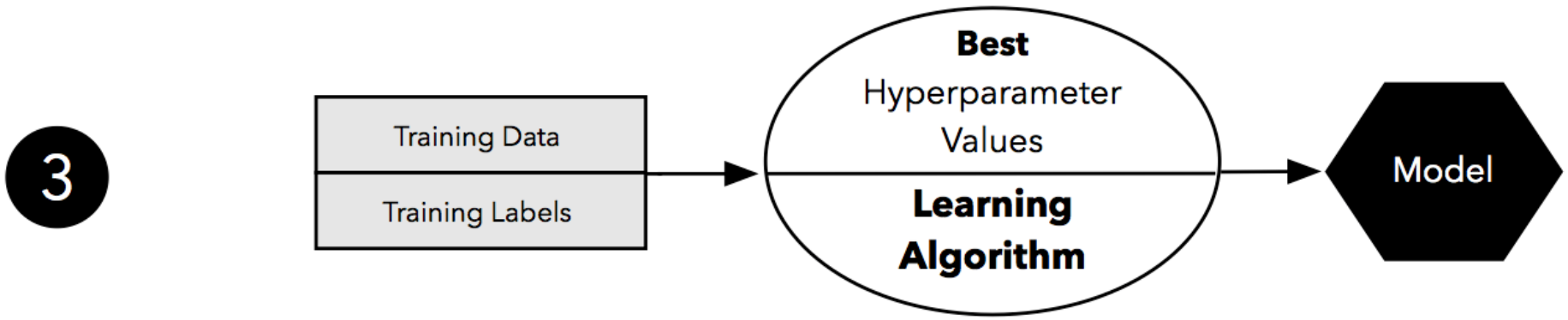
Model Selection



Step2: Hyper-parameters search

- We now experiment with various Hyperparameter settings using K-Fold cross validation on training data

Model Selection

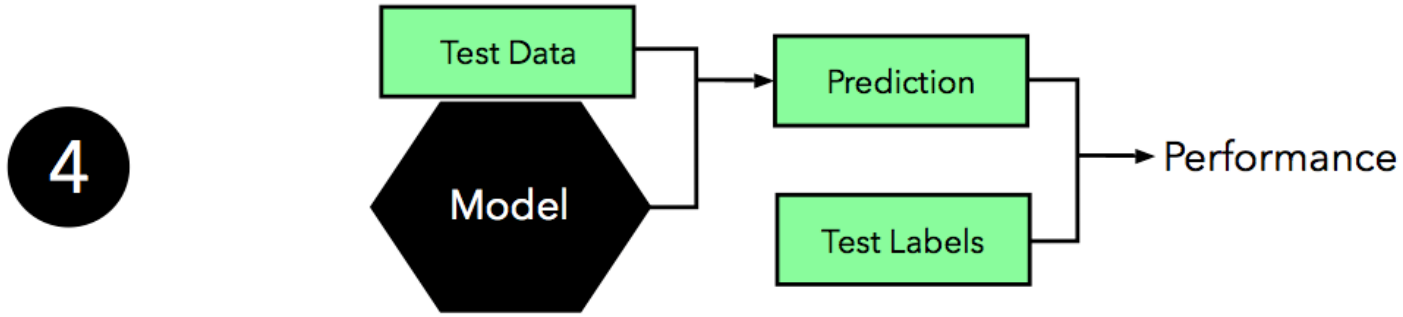


Step3: Choose the best hyper-parameter setting

- Take the hyper-parameter settings that correspond to the best-performing model

- we can then use the complete training set for model fitting.

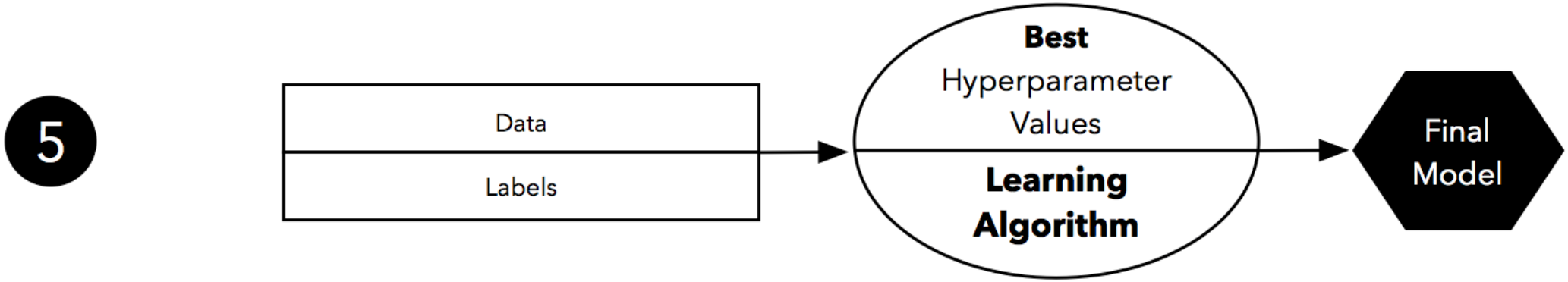
Model Selection



Step4: Evaluate the model on test set

- Now it's time to make use of the independent test set that we withheld
- We use the test set to evaluate the model that we obtained from step 3.

Model Selection



Step5: Fit the model to all data and ready to be deployed

- When we completed the evaluation stage, we can fit a model to all our data, which could be the model for (the so-called) *deployment*.

Model Selection

Credit

https://sebastianraschka.com/blog/2016/model-evaluation-selection-part3.html.