IRON HACK

SUPERVISED LEARNING CLASSIFICATION

WHAT WILL WE COVER?

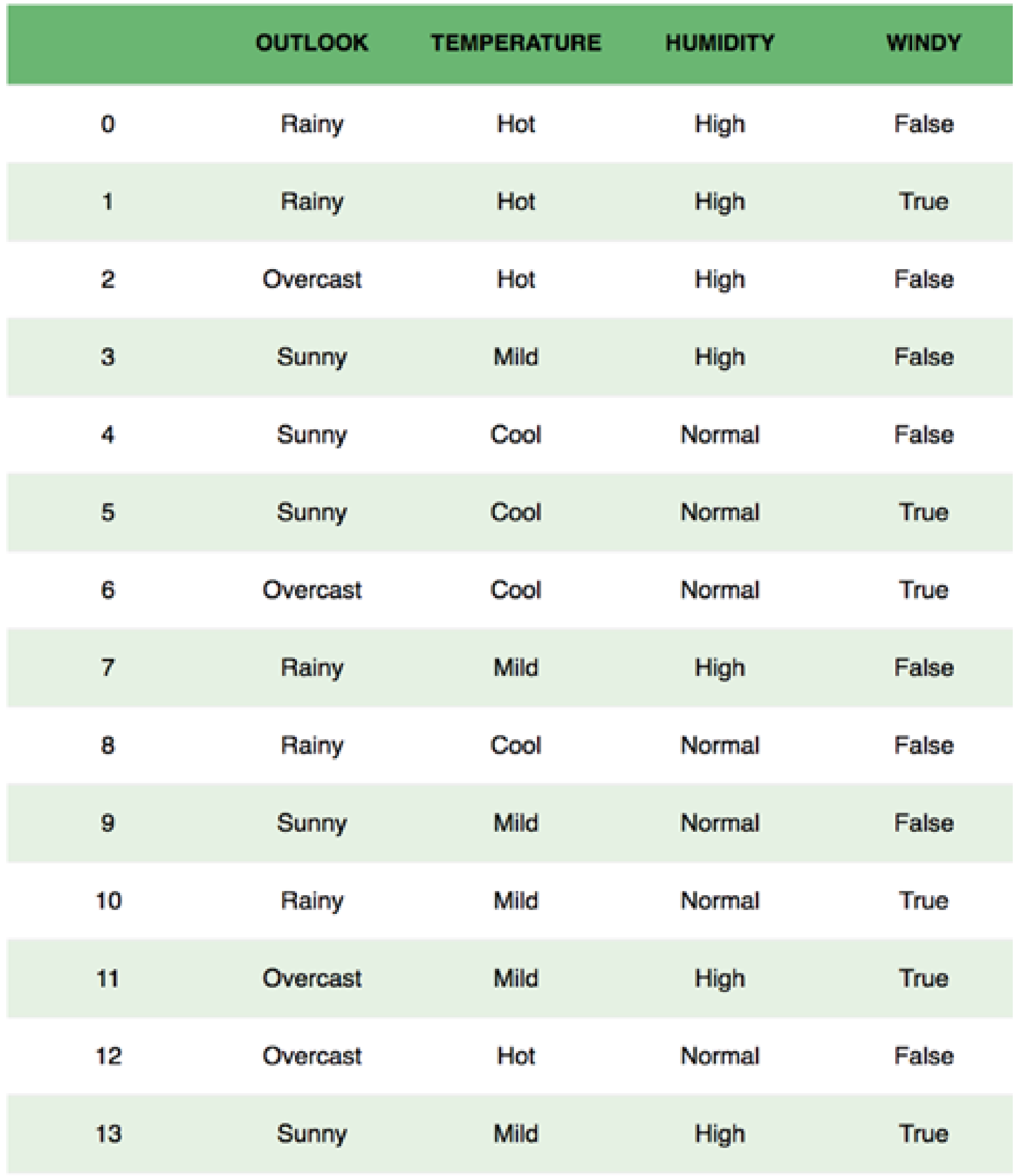
- Recap of what is supervised learning
- Core Concepts required
- Classification Algorithms
- Regression Algorithms

MACHINE LEARNING: SUPERVISED VS UNSUPERVISED

Whenever we have a defined output to predict it is a supervised ML problem

If we don't then we have a unsupervised learning problem

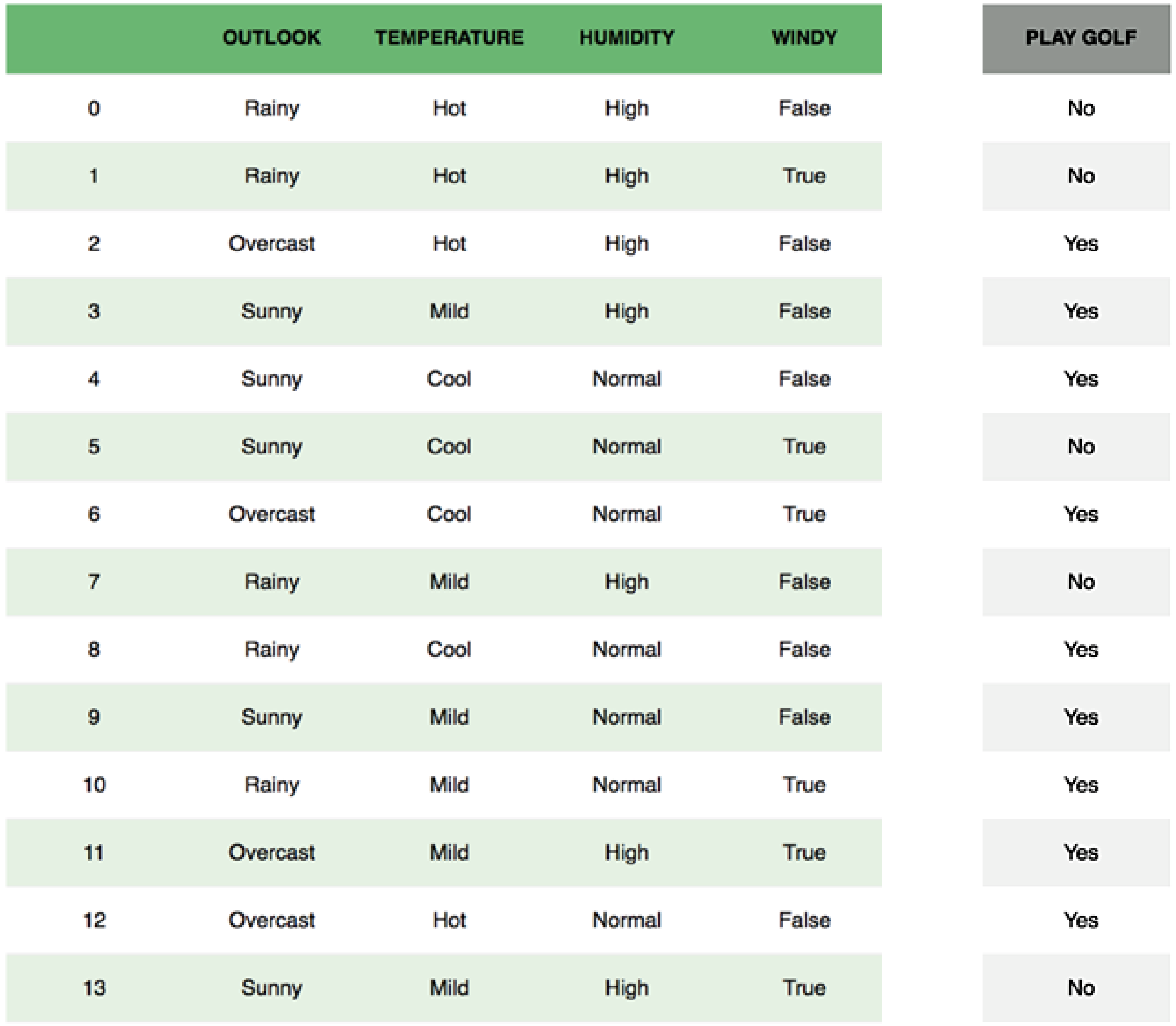
SUPERVISED: SHOULD I PLAY GOLF?



	OUTLOOK	TEMPERATURE	HUMIDITY	WINDY
0	Rainy	Hot	High	False
1	Rainy	Hot	High	True
2	Overcast	Hot	High	False
3	Sunny	Mild	High	False
4	Sunny	Cool	Normal	False
5	Sunny	Cool	Normal	True
6	Overcast	Cool	Normal	True
7	Rainy	Mild	High	False
8	Rainy	Cool	Normal	False
9	Sunny	Mild	Normal	False
10	Rainy	Mild	Normal	True
11	Overcast	Mild	High	True
12	Overcast	Hot	Normal	False
13	Sunny	Mild	High	True

The objective is to predict if based on the weather conditions of a particular day, we should go play Golf?

SUPERVISED: SHOULD I PLAY GOLF?



	OUTLOOK	TEMPERATURE	HUMIDITY	WINDY	PLAY GOLF
0	Rainy	Hot	High	False	No
1	Rainy	Hot	High	True	No
2	Overcast	Hot	High	False	Yes
3	Sunny	Mild	High	False	Yes
4	Sunny	Cool	Normal	False	Yes
5	Sunny	Cool	Normal	True	No
6	Overcast	Cool	Normal	True	Yes
7	Rainy	Mild	High	False	No
8	Rainy	Cool	Normal	False	Yes
9	Sunny	Mild	Normal	False	Yes
10	Rainy	Mild	Normal	True	Yes
11	Overcast	Mild	High	True	Yes
12	Overcast	Hot	Normal	False	Yes
13	Sunny	Mild	High	True	No

But how does my algorithm learn?

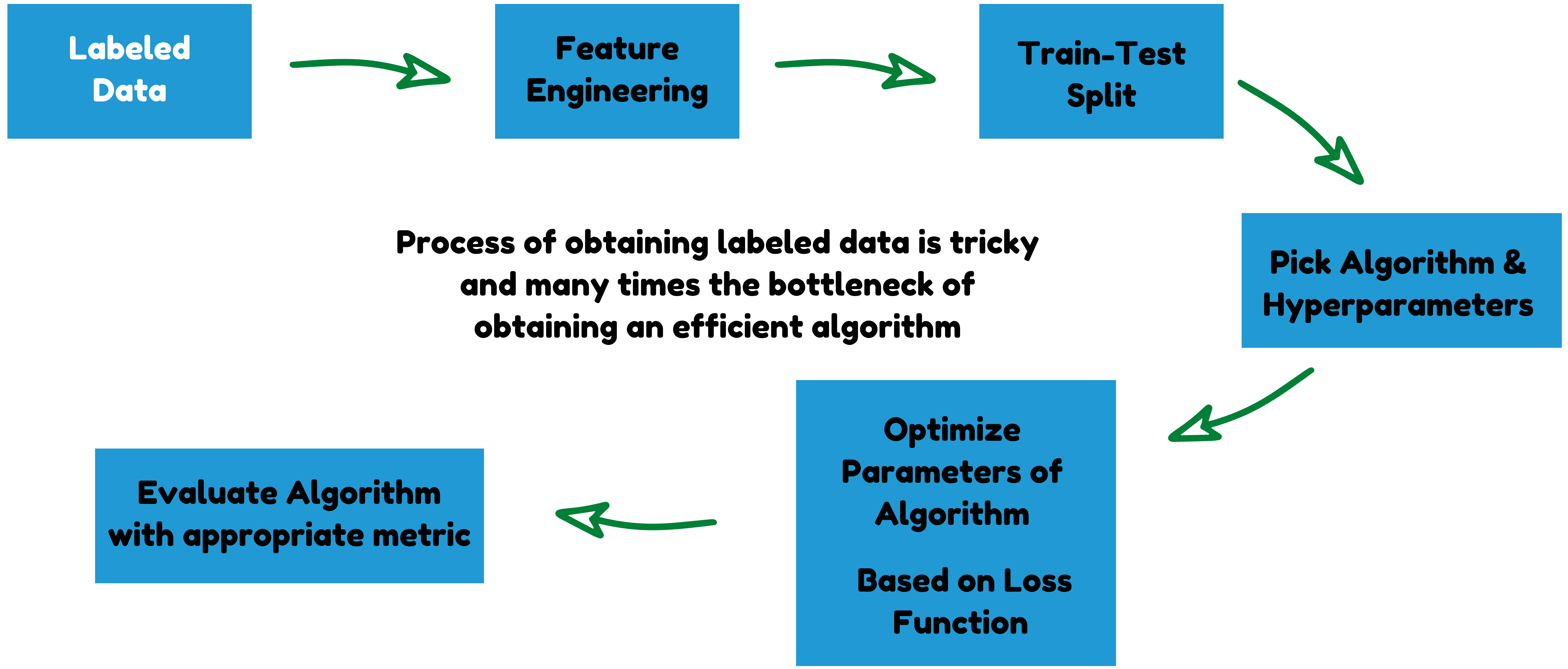
Based on past datapoints!!!

This is the "Supervised" component

These are called the labels

Supervised Learning is performed on labelled data

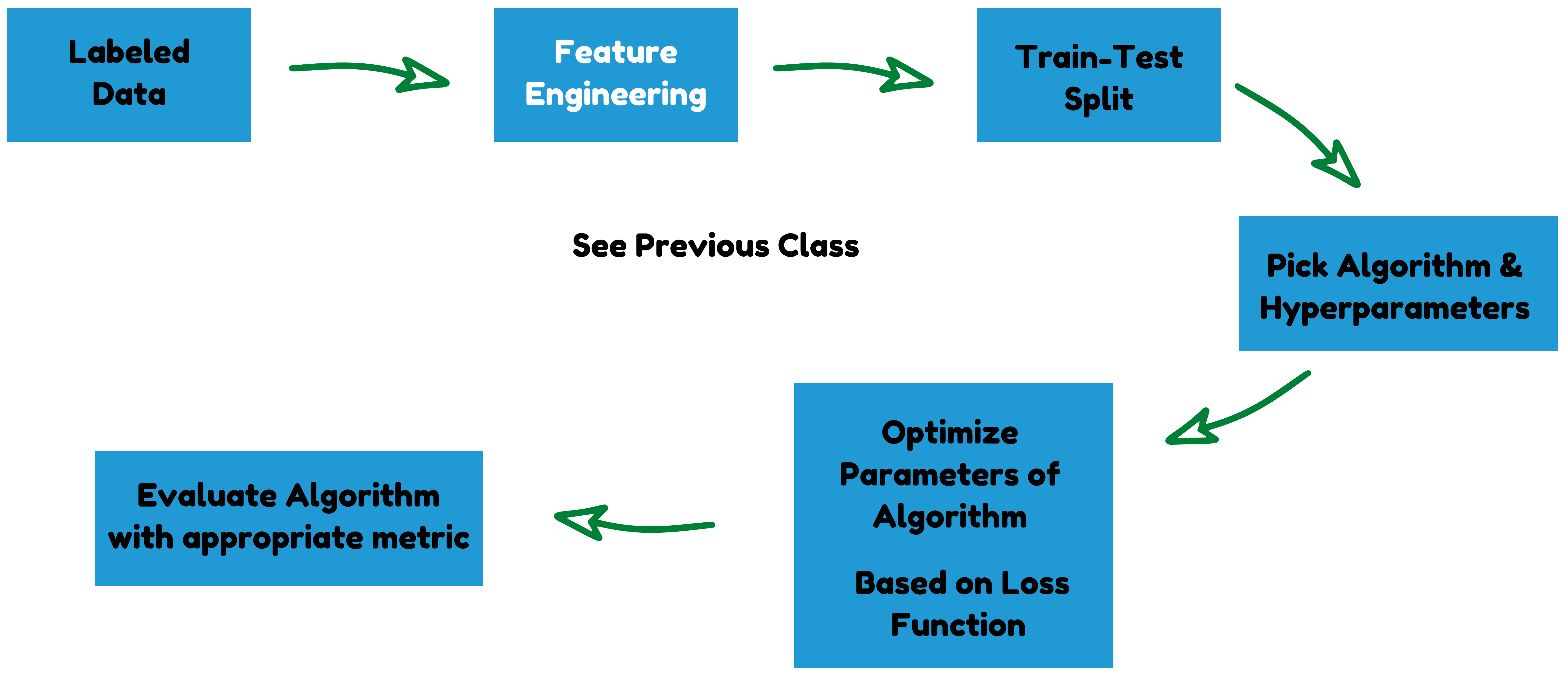
THE MACHINE LEARNING PROCESS



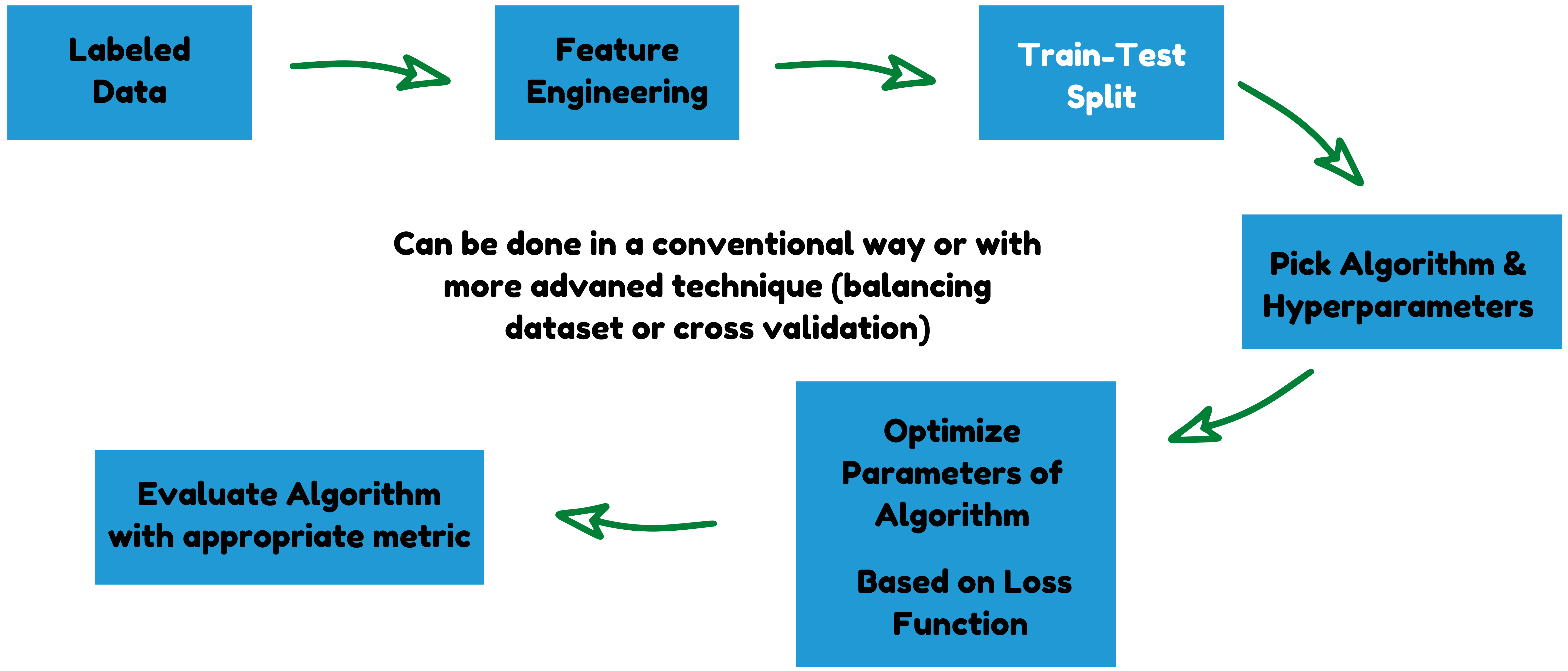
Labeled Data
Feature Engineering
Train-Test Split
Pick Algorithm & Hyperparameters
Optimize Parameters of Algorithm
Based on Loss Function
Evaluate Algorithm with appropriate metric

Process of obtaining labeled data is tricky and many times the bottleneck of obtaining an efficient algorithm

THE MACHINE LEARNING PROCESS


Labeled Data
Feature Engineering
Train-Test Split
Pick Algorithm & Hyperparameters
Optimize Parameters of Algorithm
Based on Loss Function
Evaluate Algorithm with appropriate metric
See Previous Class

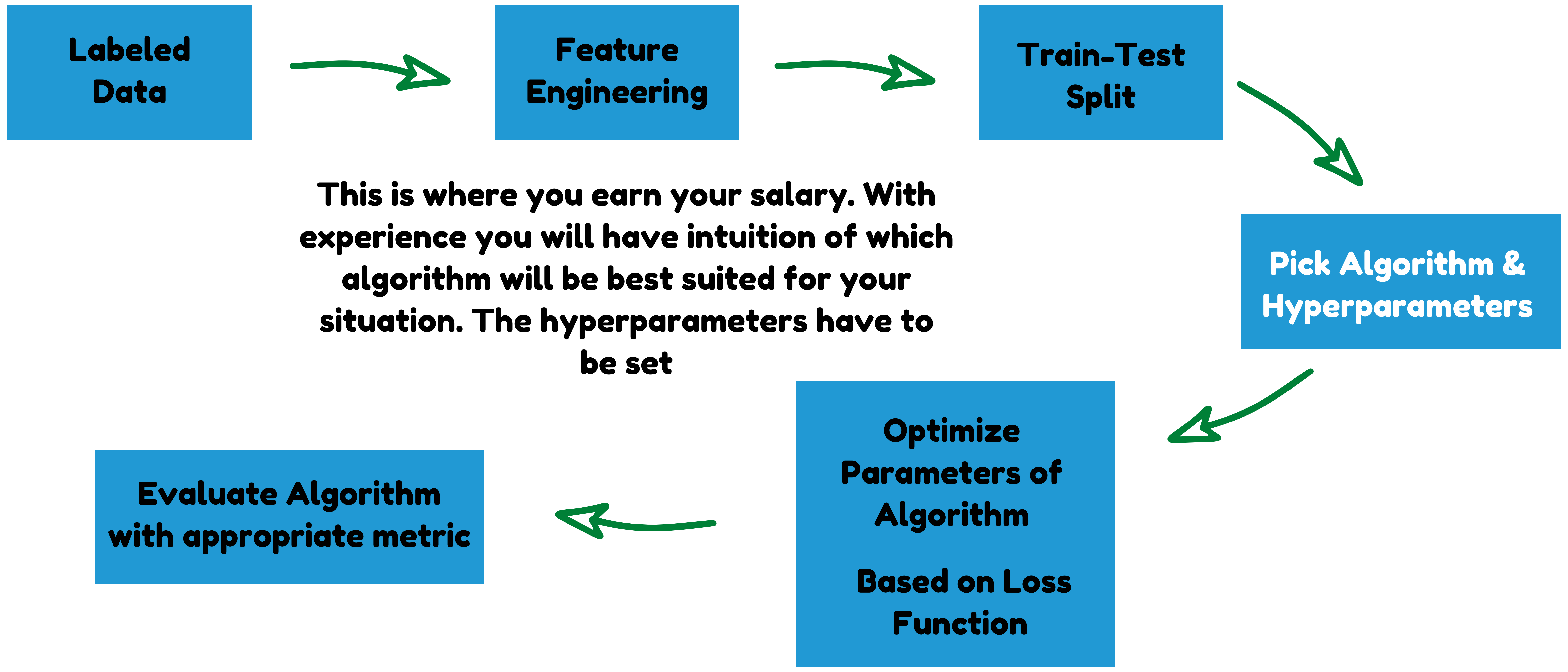
THE MACHINE LEARNING PROCESS


Labeled Data
Feature Engineering
Train-Test Split
Pick Algorithm & Hyperparameters
Optimize Parameters of Algorithm

Based on Loss Function
Evaluate Algorithm with appropriate metric

Can be done in a conventional way or with more advaned technique (balancing dataset or cross validation)

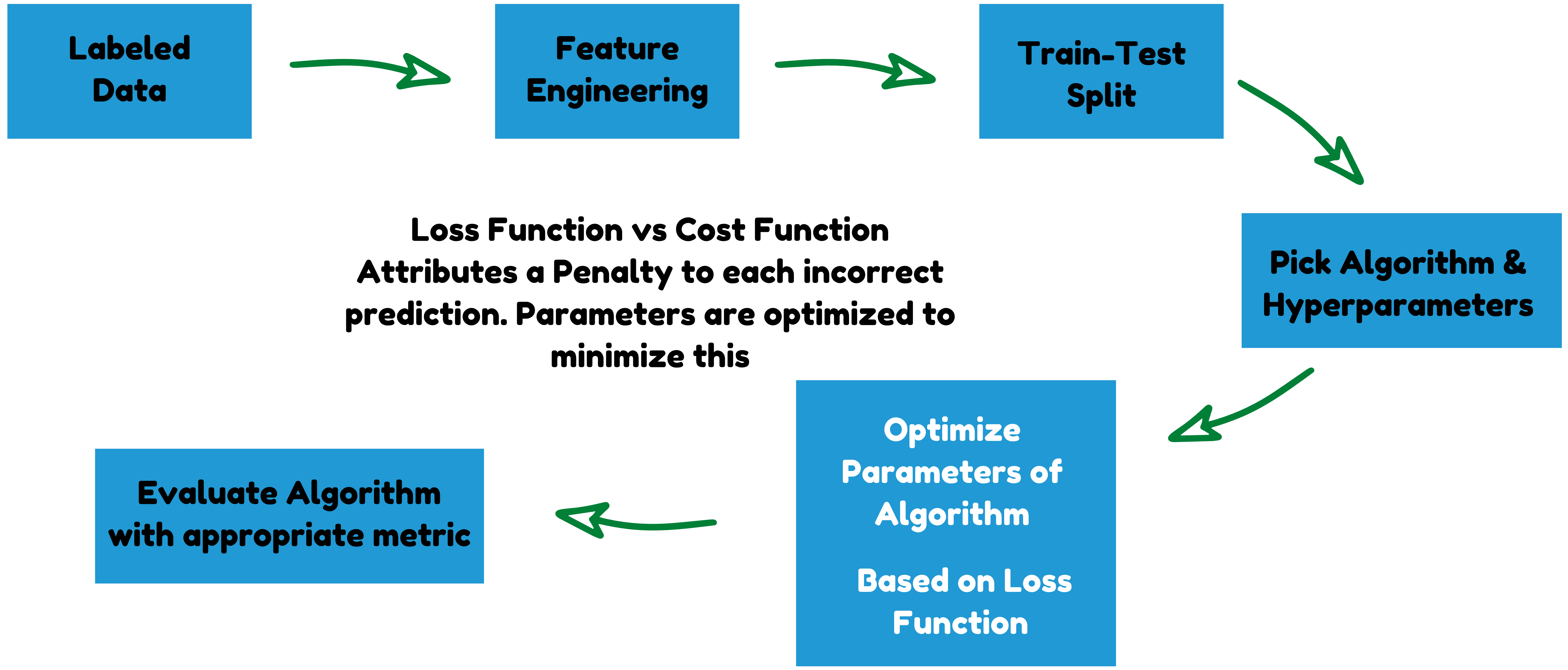
THE MACHINE LEARNING PROCESS


Labeled Data
Feature Engineering
Train-Test Split
Pick Algorithm & Hyperparameters
Optimize Parameters of Algorithm

Based on Loss Function
Evaluate Algorithm with appropriate metric

This is where you earn your salary. With experience you will have intuition of which algorithm will be best suited for your situation. The hyperparameters have to be set

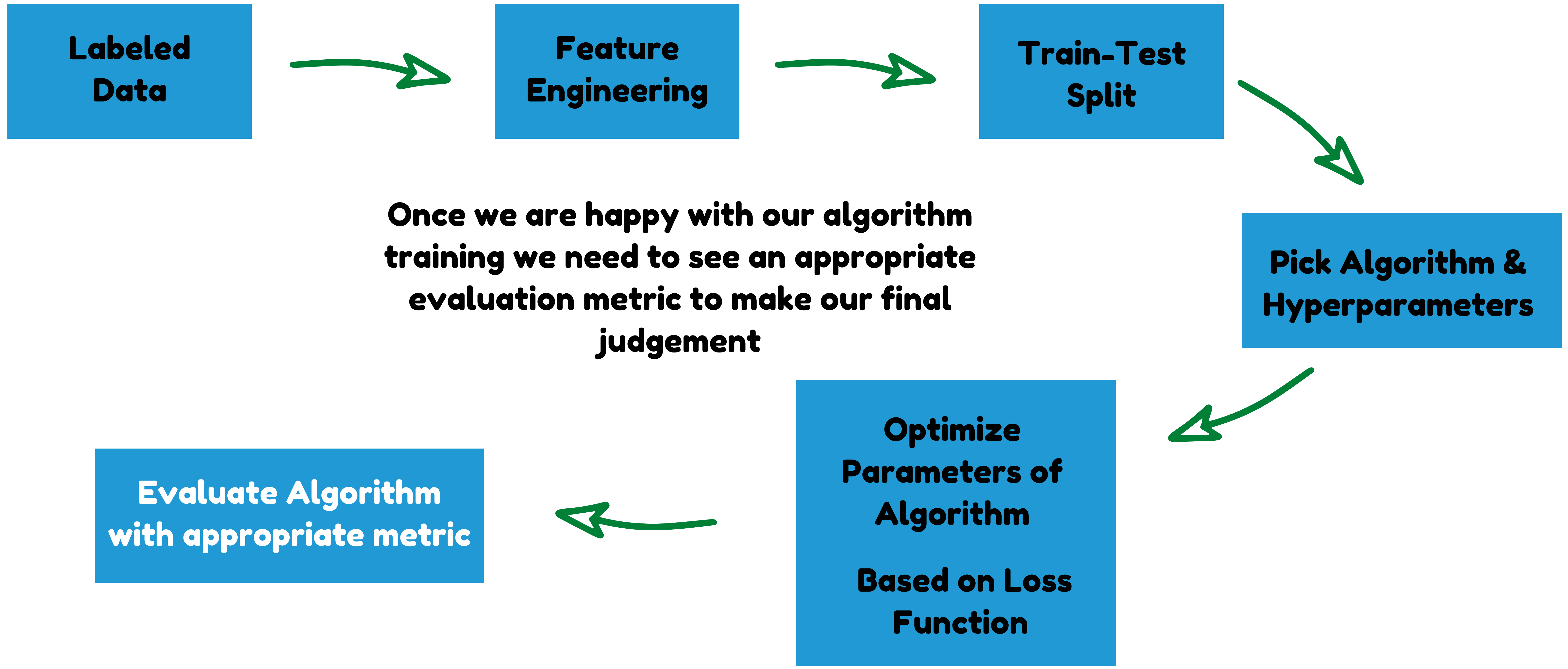
THE MACHINE LEARNING PROCESS


Labeled Data
Feature Engineering
Train-Test Split
Pick Algorithm & Hyperparameters
Optimize Parameters of Algorithm

Based on Loss Function
Evaluate Algorithm with appropriate metric

Loss Function vs Cost Function
Attributes a Penalty to each incorrect prediction. Parameters are optimized to minimize this

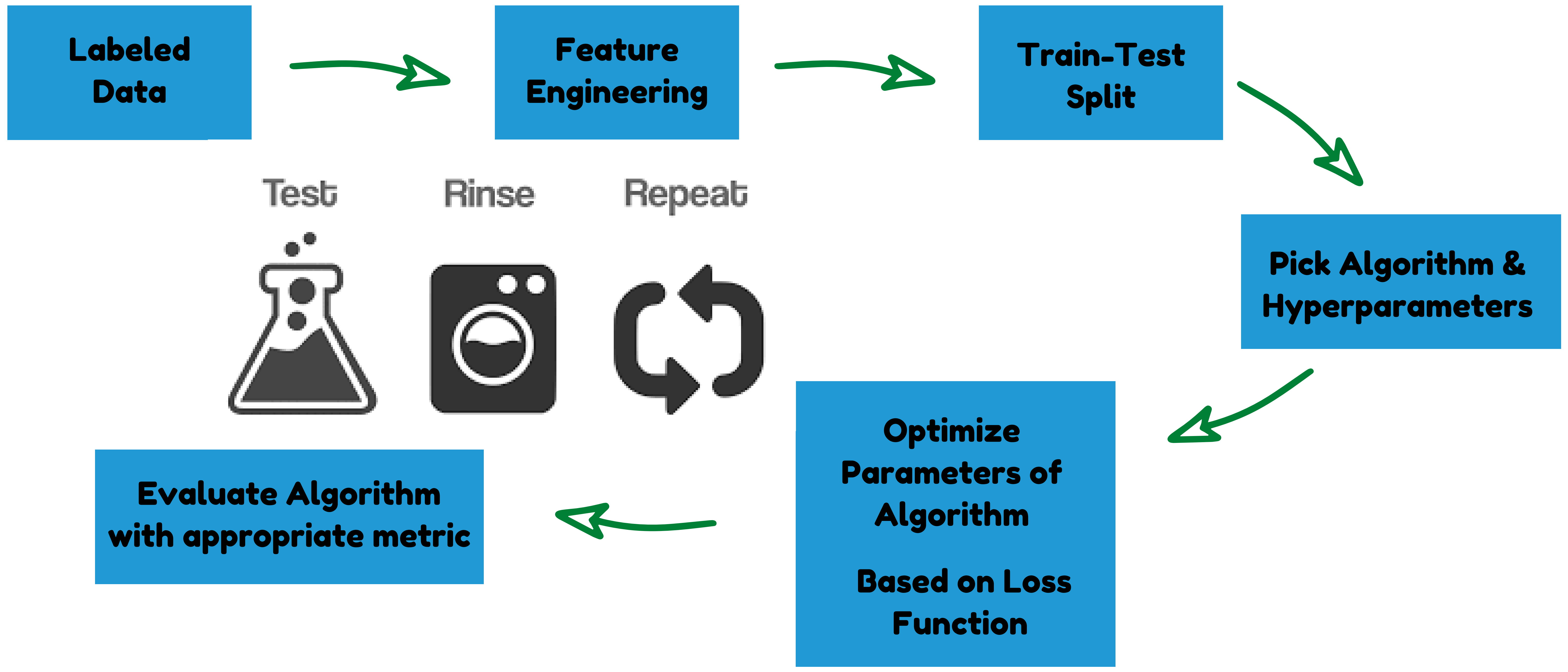
THE MACHINE LEARNING PROCESS


Labeled Data
Feature Engineering
Train-Test Split
Pick Algorithm & Hyperparameters
Optimize Parameters of Algorithm

Based on Loss Function
Evaluate Algorithm with appropriate metric

Once we are happy with our algorithm training we need to see an appropriate evaluation metric to make our final judgement

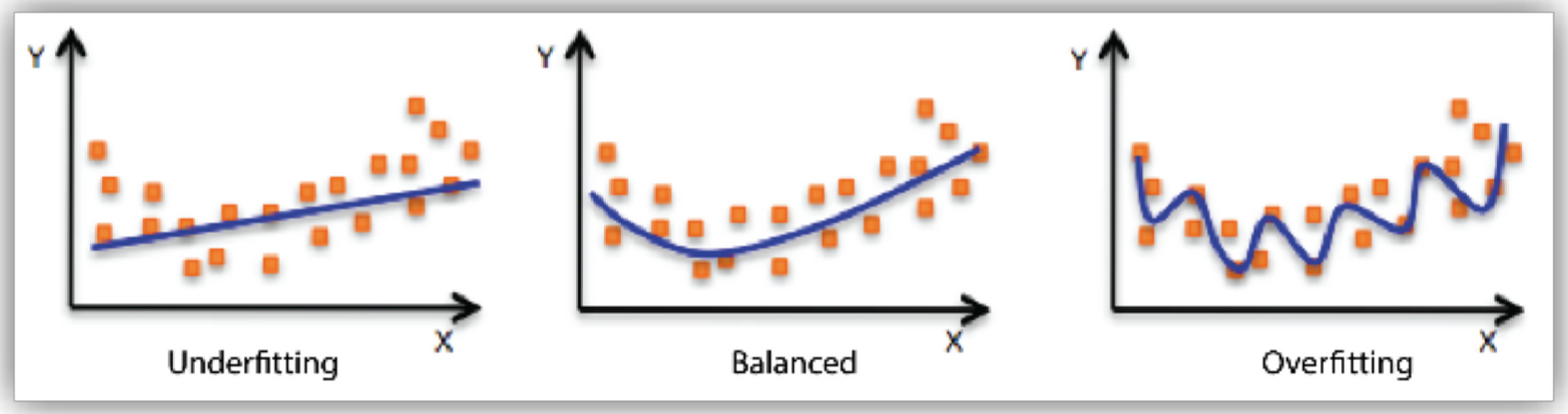
THE MACHINE LEARNING PROCESS


Labeled Data
Feature Engineering
Train-Test Split
Pick Algorithm & Hyperparameters
Optimize Parameters of Algorithm
Based on Loss Function
Evaluate Algorithm with appropriate metric
Test
Rinse
Repeat

CLASSIFICATION ALGORITHMS

- **KNN** **(Regression Friendly) (Multi Class)**
- **Logistic Regression**
- **Decision Trees** **(Regression Friendly) (Multi Class)**
- **Naive-Bayes Classifier** **(Multi Class)**
- **Support Vector Machine (SVM)**

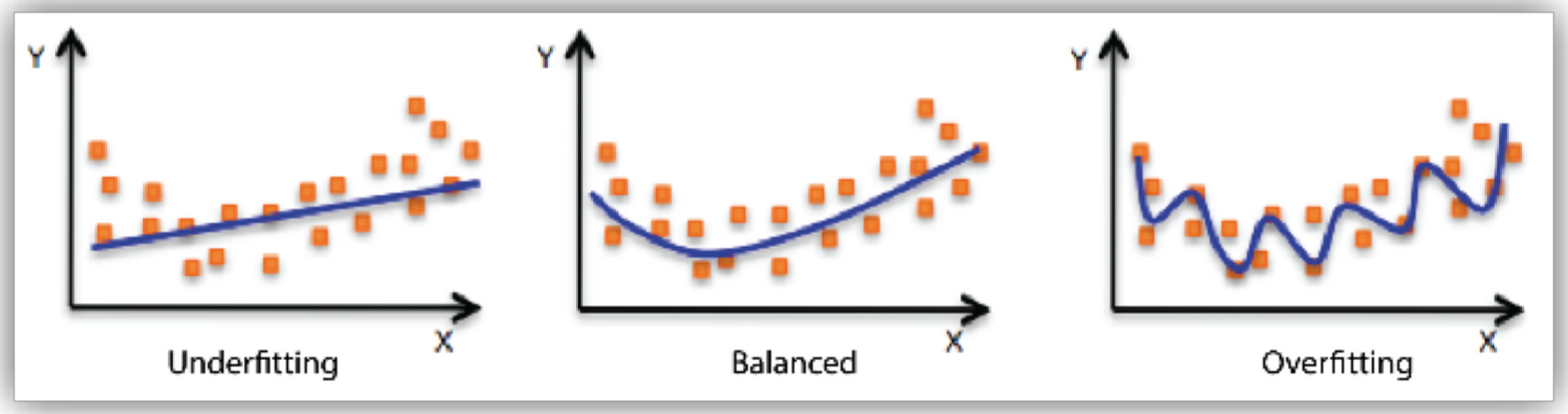
CLASSIFICATION ALGORITHMS



- **This is the biggest danger when creating machine learning algorithms**

Underfitting: model is too simplistic
Overfitting: model adapts too much to the training data and does not generalize properly

CLASSIFICATION ALGORITHMS



- **Model Generelization is how well will the trained model perform on new unseen data?**

- **This is why we perform the train-test split!!!**

- **This way we give the algorithm some data to train, and then a smaller sample to test how does the model behave on unseen data**

K NEAREST NEIGHBOURS

K- Nearest Neighbours (KNN)

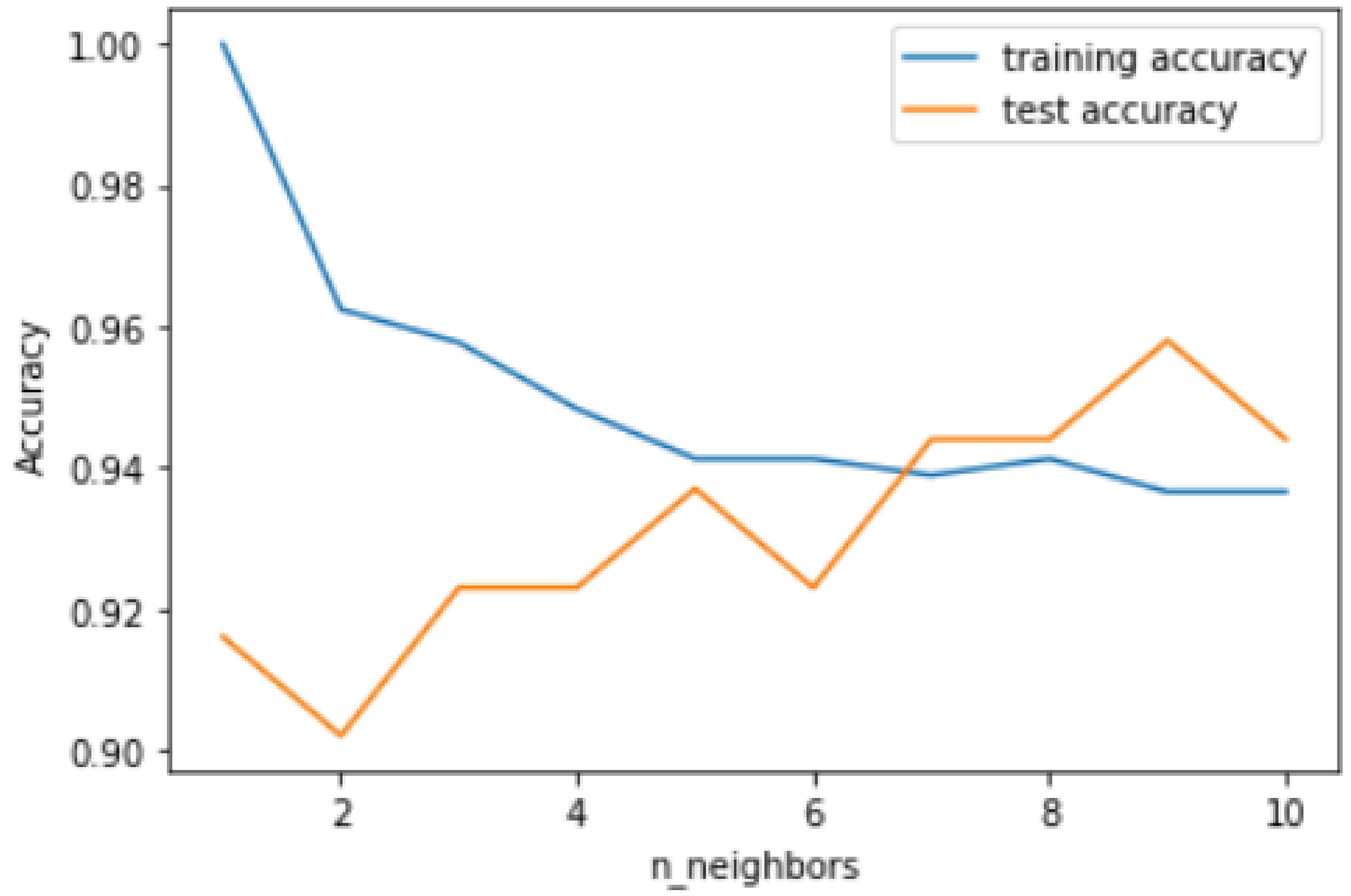
The Copy-Cat of the Neighbours

Hyperparameters:

- **K - # of neighbours**
- **Method of weight of neighbours**

K NEAREST NEIGHBOURS

Example of Overfitting and training vs test tradeoff



KNN - SUMMARY

➕ No assumptions about data – useful, for example, for nonlinear data
Simple algorithm – to explain and understand/interpret

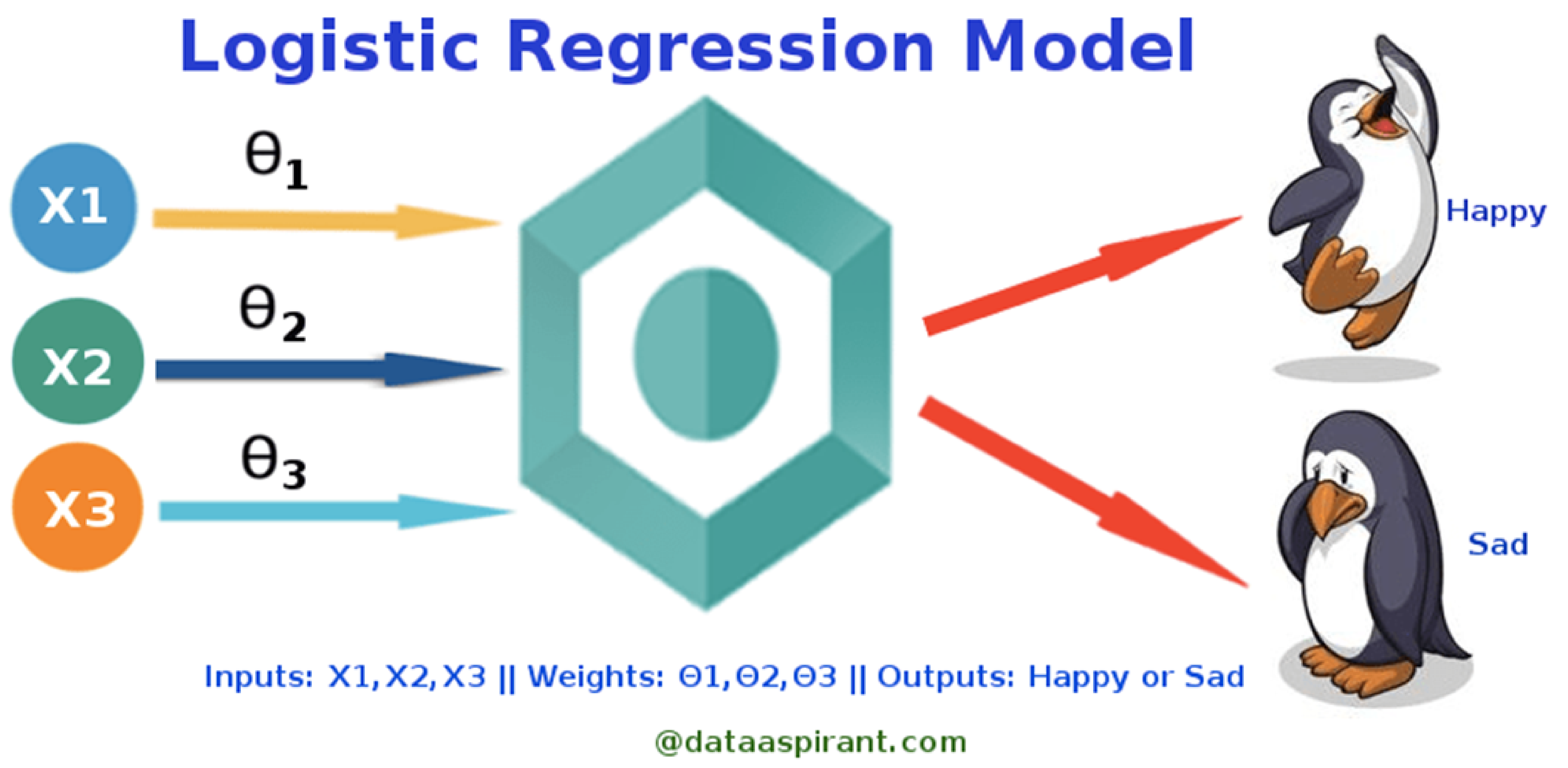
⊖ Requires all data to be stored in memory to compute
Can under perform with many variables
Very sensitive to scale

LOGISTIC REGRESSION

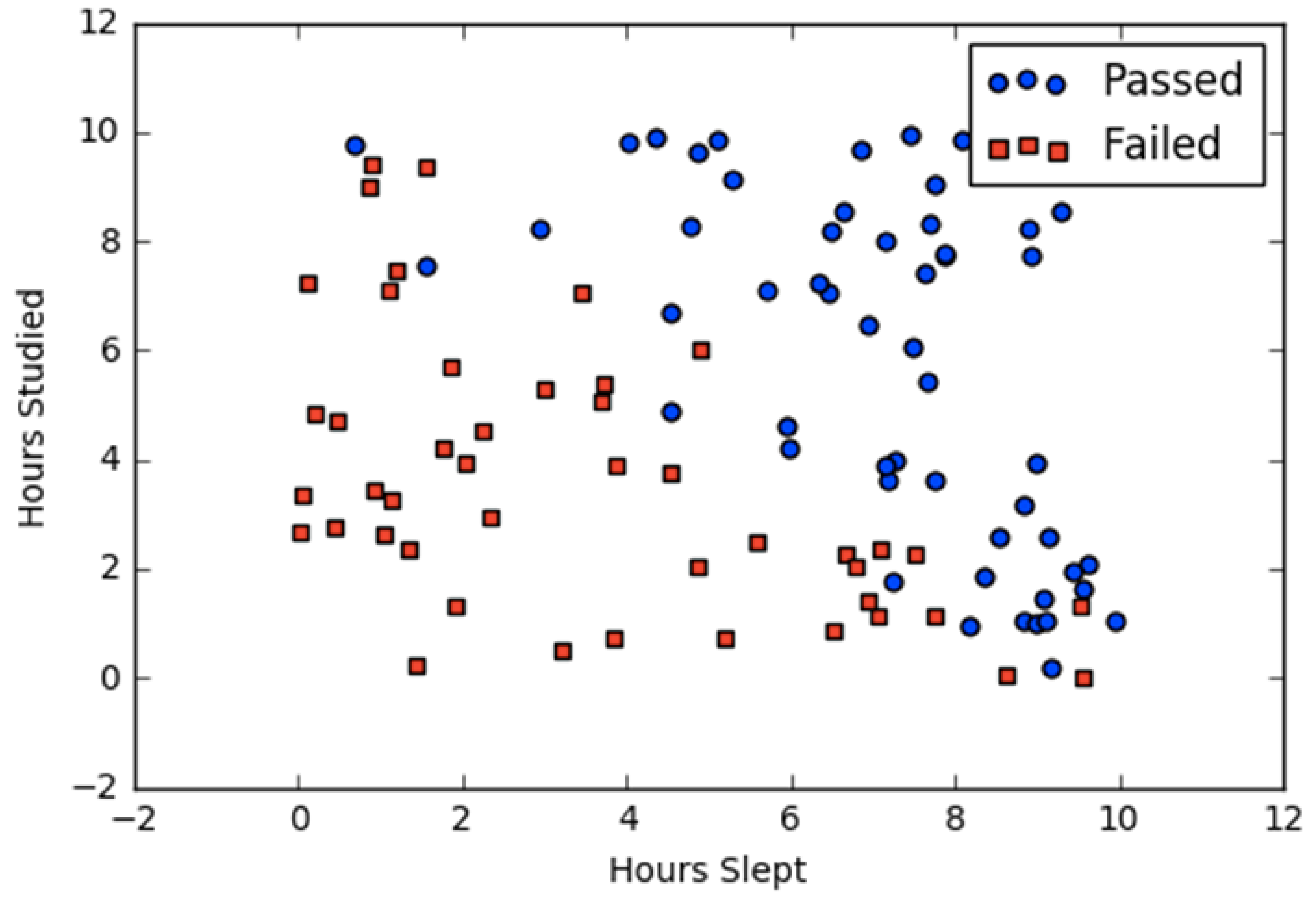
Binary Classification Algorithm

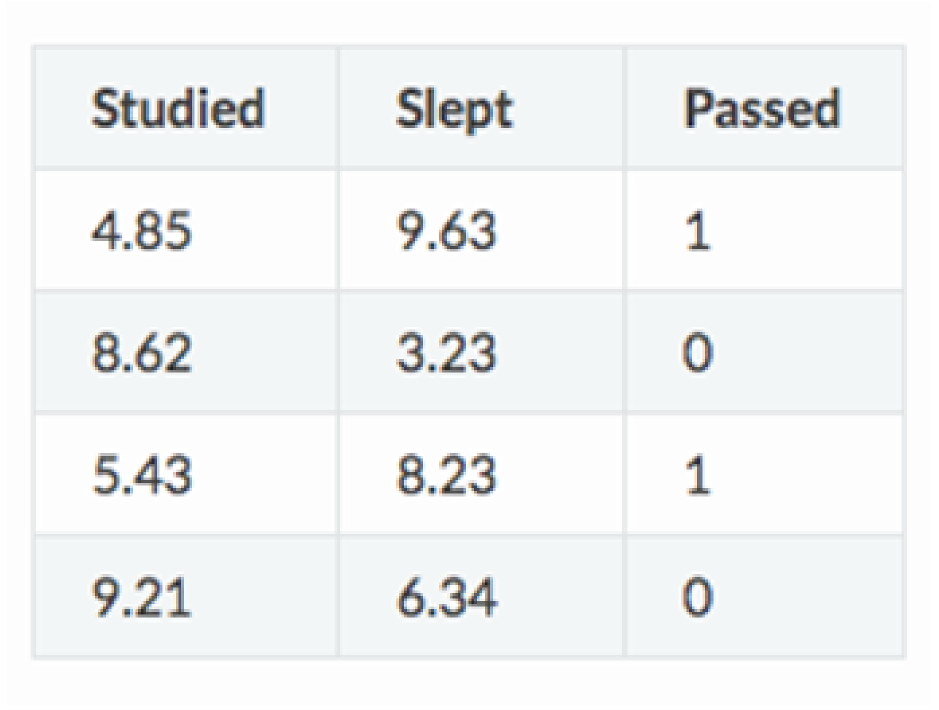
Takes in numerical inputs, takes a weighted sum of them and converts the result of that to a function that is "almost always" 0 or 1

Hyperparameter: Decision Boundary


Logistic Regression Model
X1
X2
X3
θ_1
θ_2
θ_3
Happy
Sad
Inputs: X1,X2,X3 || Weights: Θ1,Θ2,Θ3 || Outputs: Happy or Sad
@dataaspirant.com

LOGISTIC REGRESSION



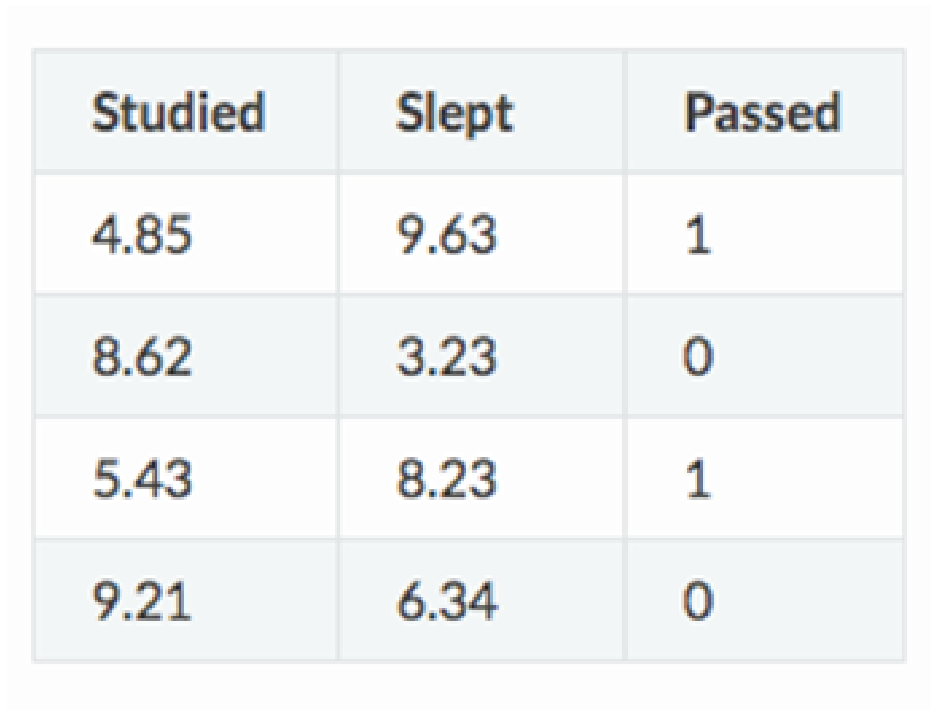


Studied	Slept	Passed
4.85	9.63	1
8.62	3.23	0
5.43	8.23	1
9.21	6.34	0

Steps:

- **Write usual linear combinations of input**
- **Pass result through sigmoid function**
- **Optimize parameters to minimize error**
- **Compare result to decision boundary**
- **Make Prediction**

LOGISTIC REGRESSION - LINEAR REGRESSION STEP



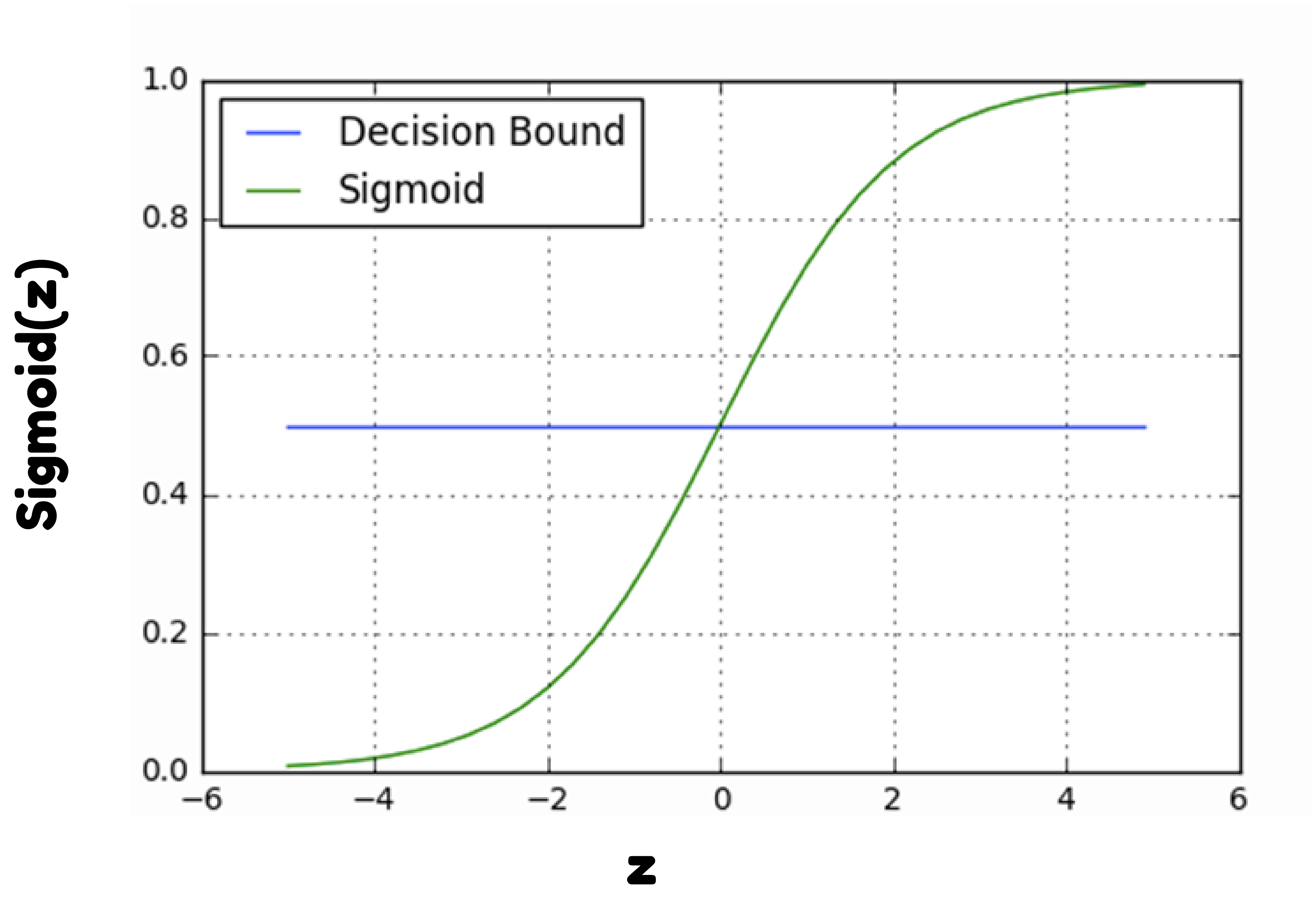
Studied	Slept	Passed
4.85	9.63	1
8.62	3.23	0
5.43	8.23	1
9.21	6.34	0

$$z = W_0 + W_1 Studied + W_2 Slept$$

We do not optimize yet the parameters of this linear regression and we don't yet compare z to anything

LOGISTIC REGRESSION - SIGMOID FUNCTION

This is where things become Nonlinear



$$z = W_0 + W_1 Studied + W_2 Slept$$

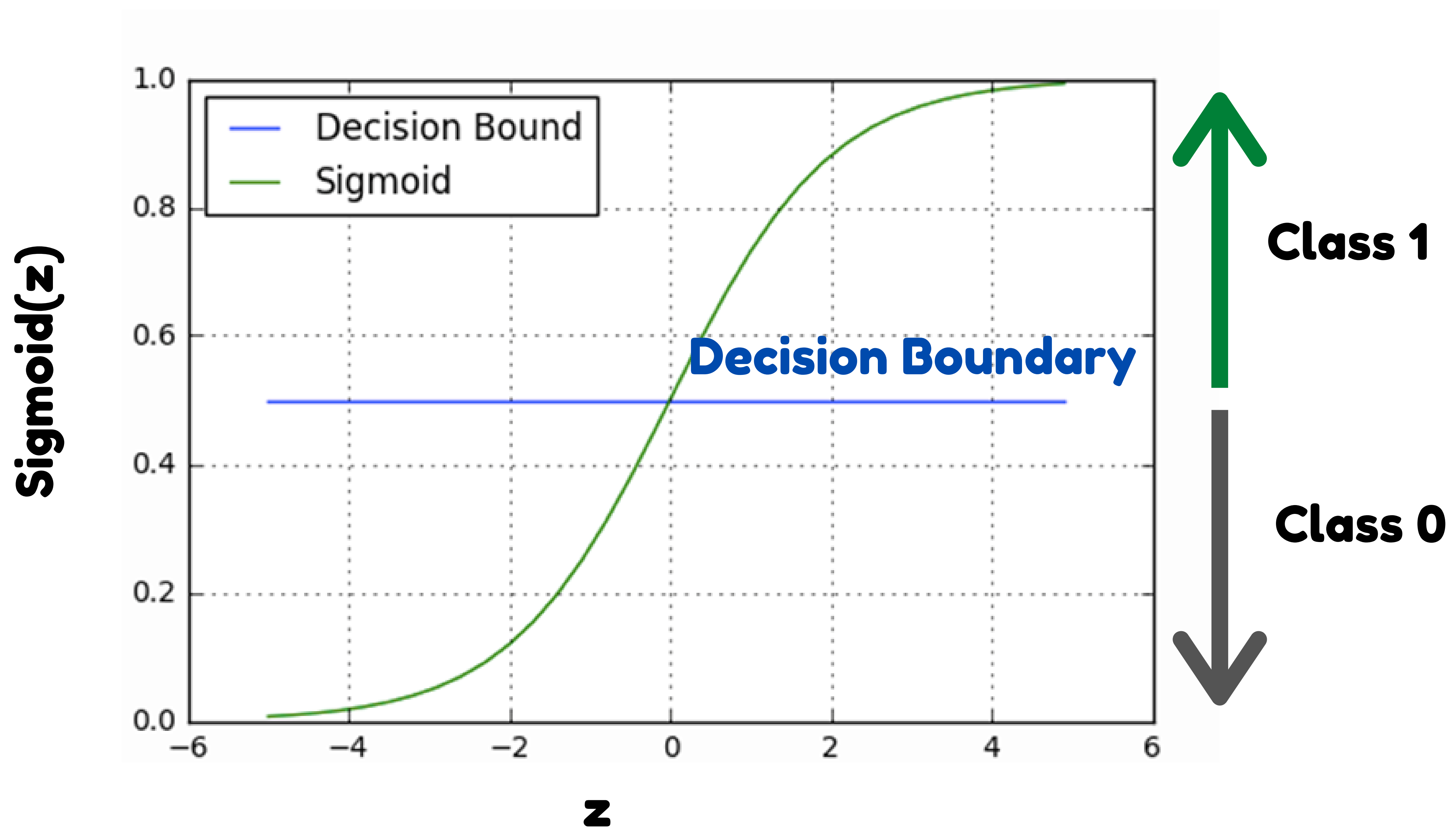
$$f(z) = \frac{1}{1+e^{-z}}$$

$$h(\vec{x}) = \frac{1}{1+e^{-w_0+w_1x_1+w_2x_2}}$$

This function goes from 0 to 1 and typically does so relatively "sharply"
We <u>now</u> optimize W1 and W2 to "fit" the observations as well as possible

LOGISTIC REGRESSION - SIGMOID FUNCTION

This is where things become Nonlinear



$$z = W_0 + W_1 Studied + W_2 Slept$$

For a set of the regression parameters, predictions are made and then compared with labels

Cost Function is used to evaluate "Penalty"

$$J(\theta) = \frac{1}{m}\sum_{i=1}^{m} \mathrm{Cost}(h_\theta(x^{(i)}), y^{(i)})$$

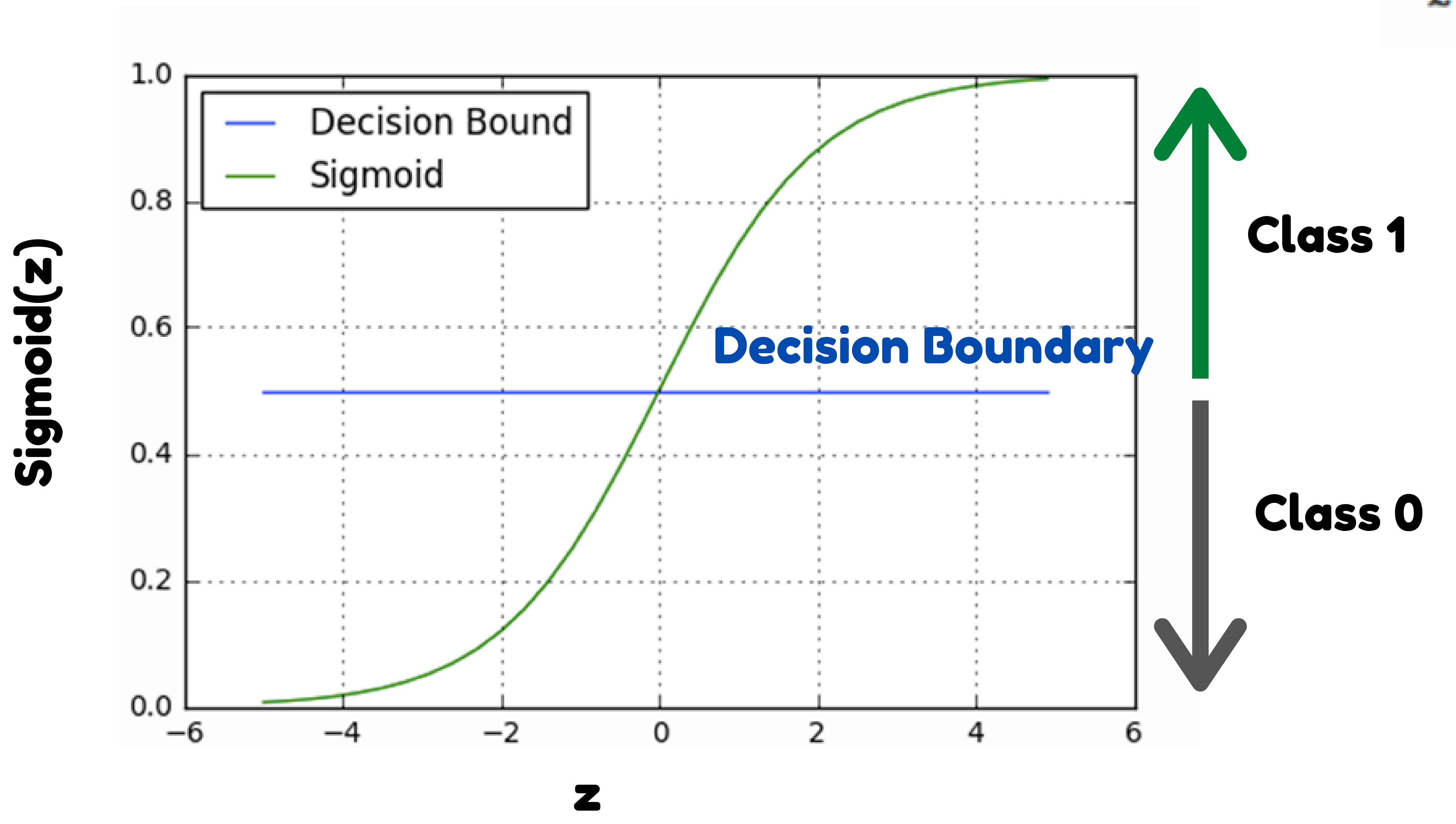
$$\mathrm{Cost}(h_\theta(x), y) = -\log(h_\theta(x)) \quad \text{if y} = 1$$

$$\mathrm{Cost}(h_\theta(x), y) = -\log(1 - h_\theta(x)) \quad \text{if y} = 0$$

LOGISTIC REGRESSION - SIGMOID FUNCTION

This is where things become Nonlinear

$$z = W_0 + W_1 Studied + W_2 Slept$$



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

$$P(class = 1) = \frac{1}{1 + e^{-z}}$$

LOGISTIC REGRESSION - SUMMARY

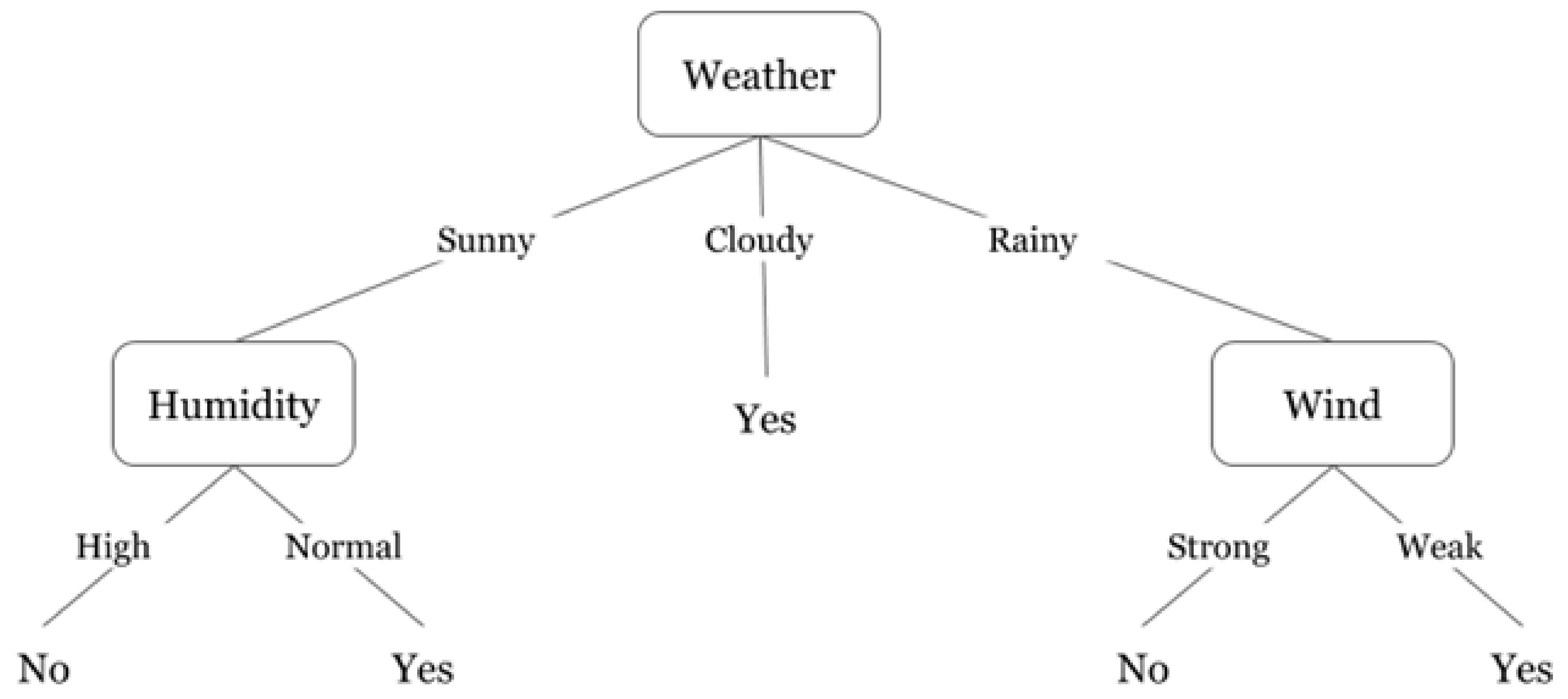
➕

Easy to implement
Trains quickly
Can be stacked to generate some really powerful models

⊖

Does not work as well for multi-class
Or at all for non-linearly separatable data

DECISION TREES



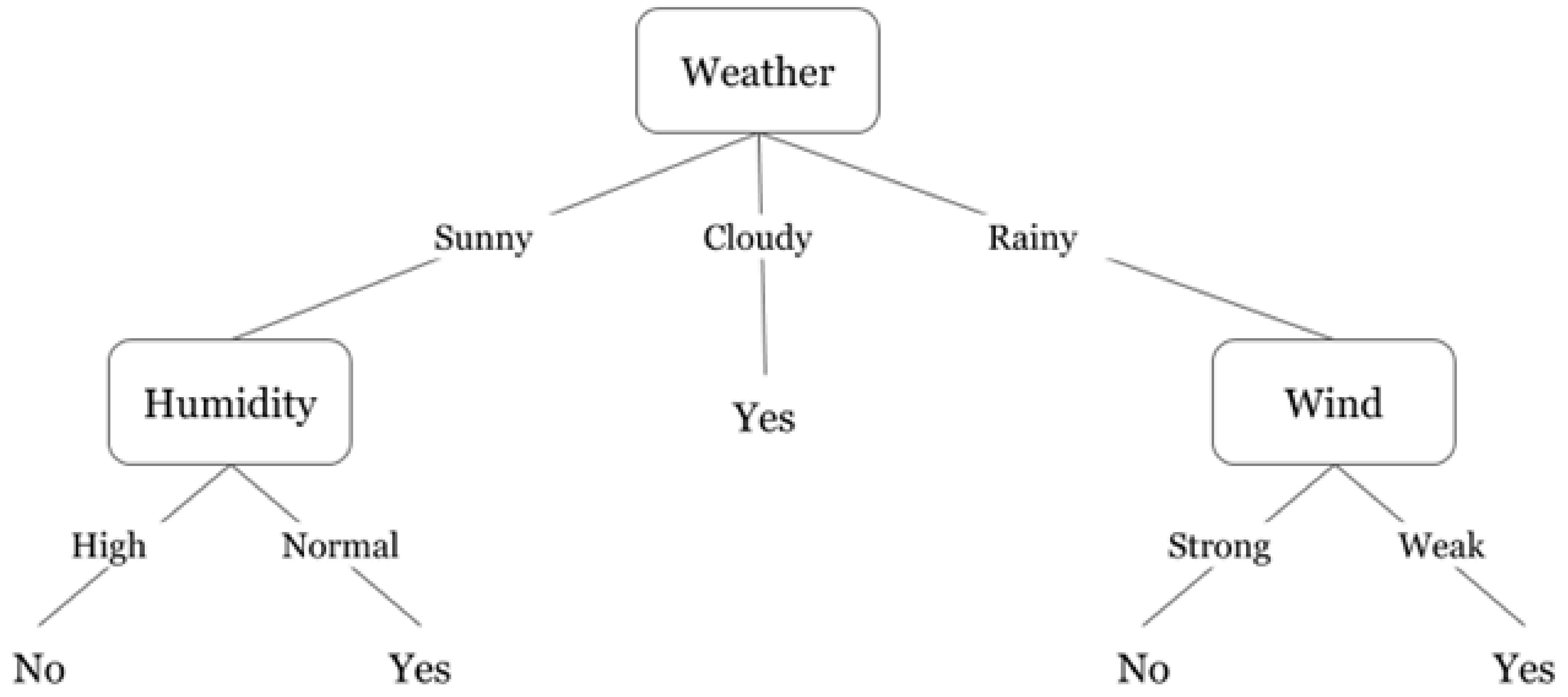
- Each level of the tree asks a "question" about a specific feature and divides the paths into several nodes

- number of levels is known as the tree depth

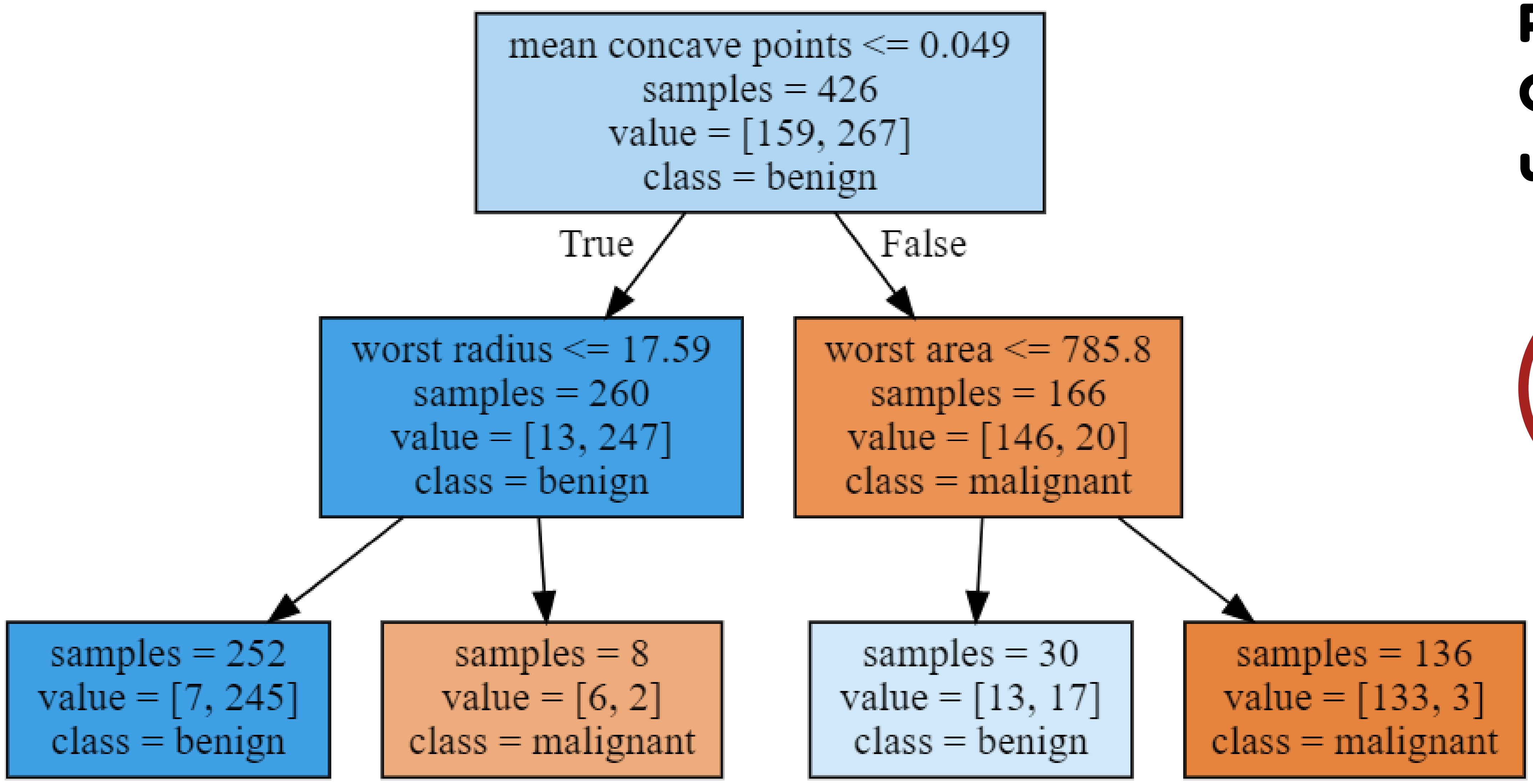
- The question you ask each level is the one that gives you the most accurate answer if you were to stop then

DECISION TREES

	OUTLOOK	TEMPERATURE	HUMIDITY	WINDY	PLAY GOLF
0	Rainy	Hot	High	False	No
1	Rainy	Hot	High	True	No
2	Overcast	Hot	High	False	Yes
3	Sunny	Mild	High	False	Yes
4	Sunny	Cool	Normal	False	Yes
5	Sunny	Cool	Normal	True	No
6	Overcast	Cool	Normal	True	Yes
7	Rainy	Mild	High	False	No
8	Rainy	Cool	Normal	False	Yes
9	Sunny	Mild	Normal	False	Yes
10	Rainy	Mild	Normal	True	Yes
11	Overcast	Mild	High	True	Yes
12	Overcast	Hot	Normal	False	Yes
13	Sunny	Mild	High	True	No



DECISION TREES - SUMMARY



+

Very robust to different types of features, including NaNs, categorical, numerical
Handles non-linearity
Robust to outliers
Can be stacked into arguably the most useful models in modern ML

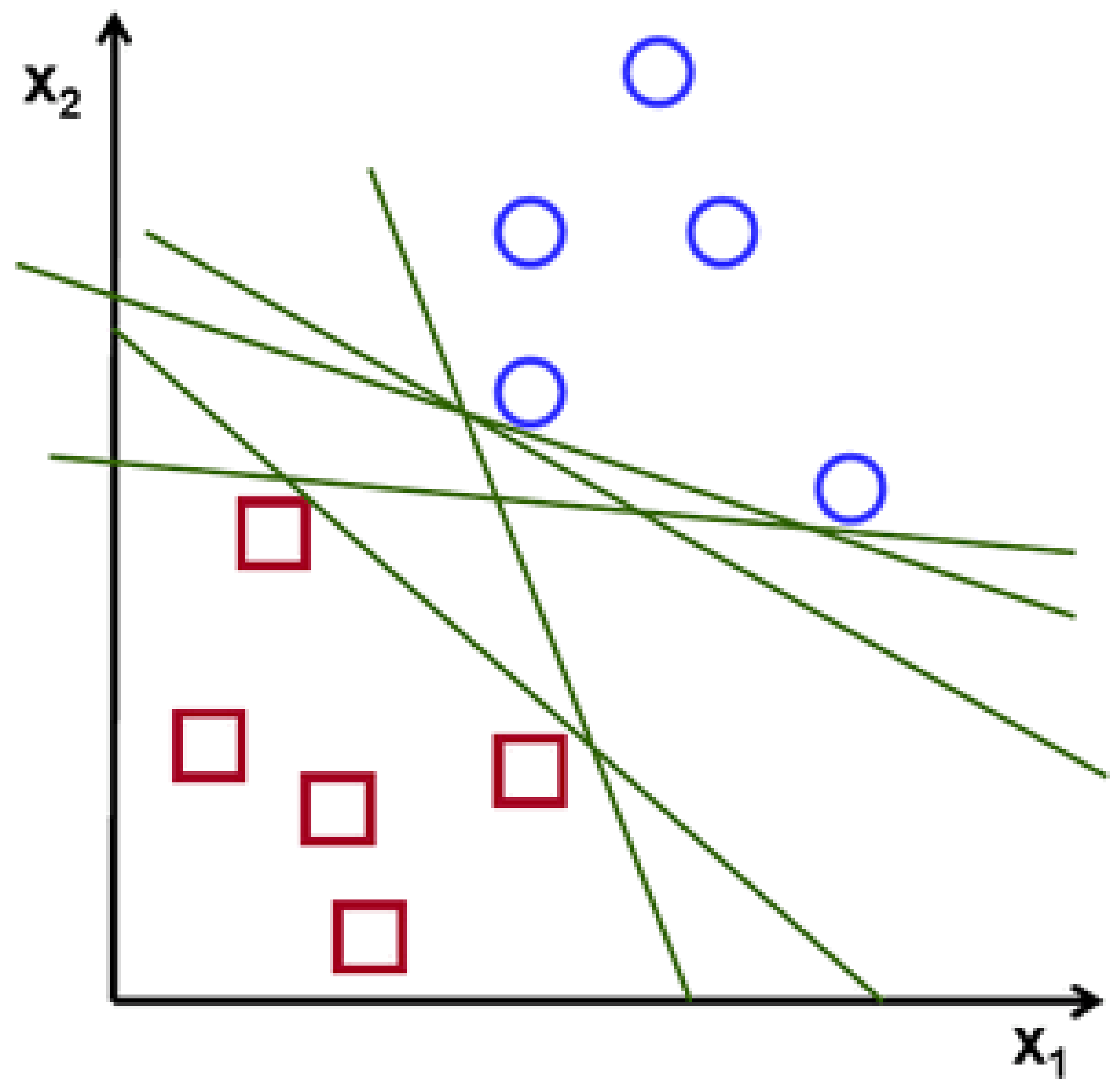
−

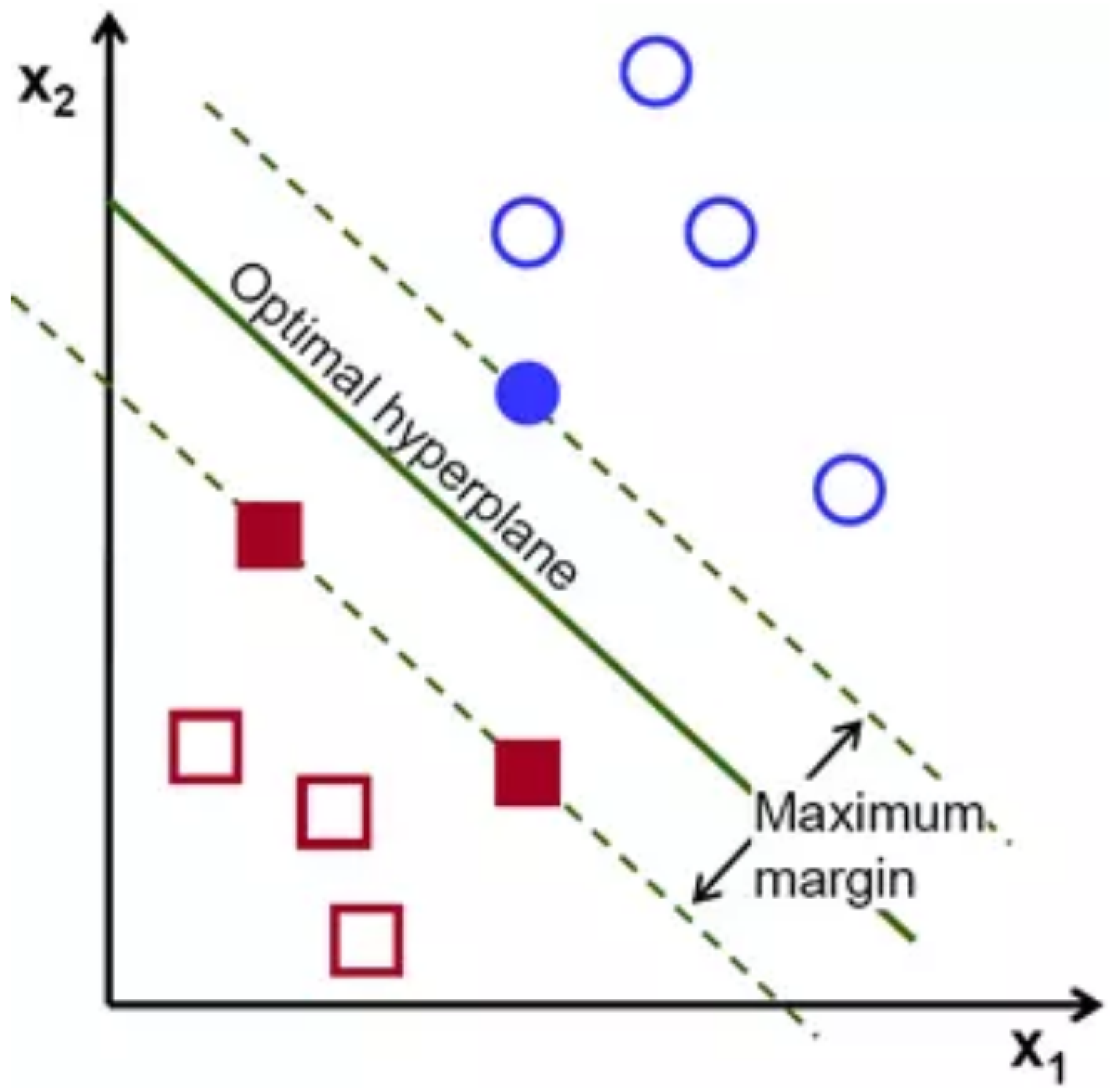
Very quickly overfit if hyperparameters are not controlled

Generalization Problems: what happens if feature outside possible ranges appears?

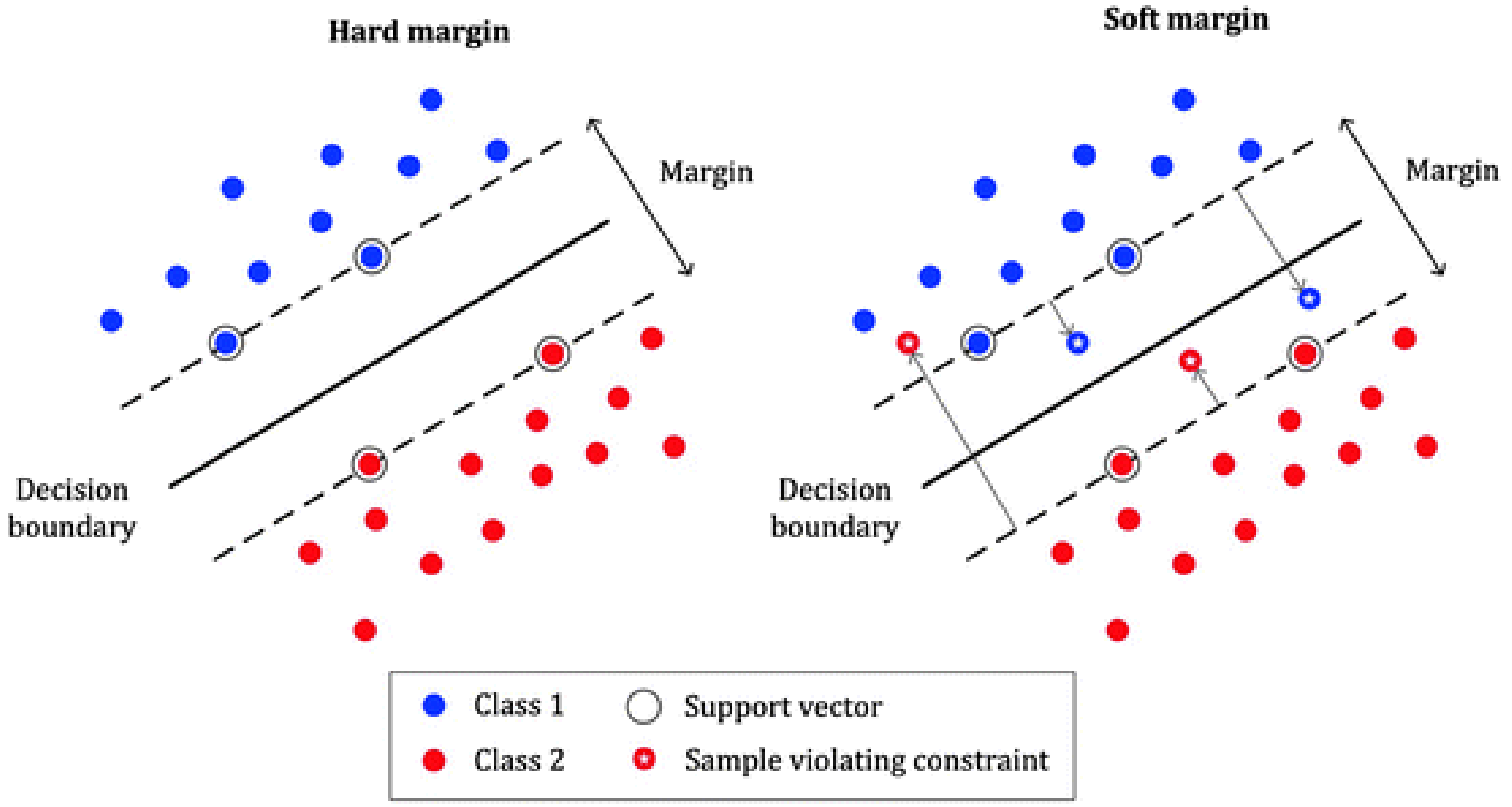
Online Learning problem: quickly change when re-trained

SUPPORT VECTOR MACHINE (SVM)





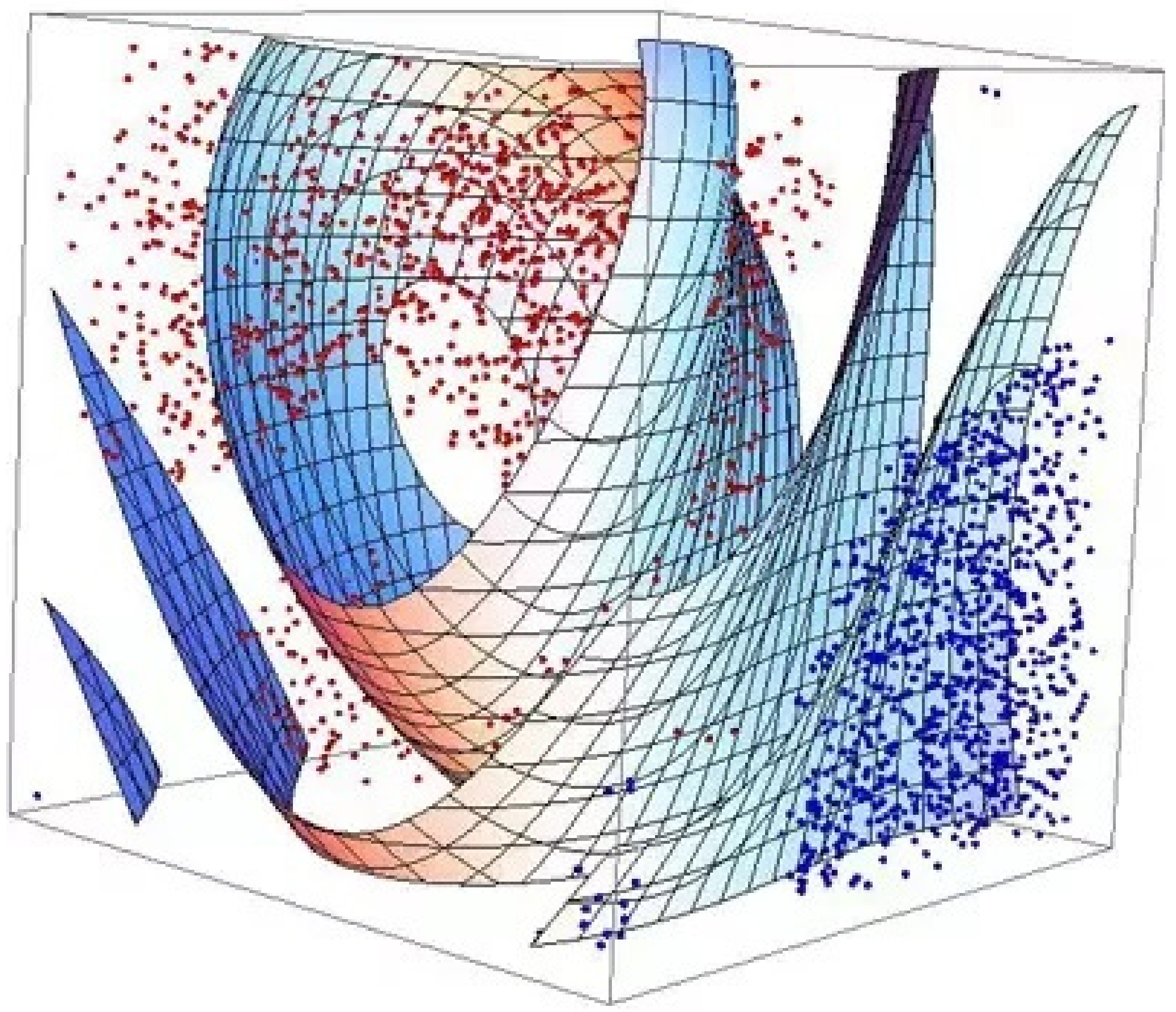
SUPPORT VECTOR MACHINE (SVM)



Our objective is to find a linear boundary that has the maximum margin, i.e the maximum distance between data points of both classes. Maximizing the margin distance provides some reinforcement so that future data points can be classified with more confidence.

We can force the SVM to maximize the distance to any point in any of the sets or add a loss function to allow but penalize some violations of the boundary

SUPPORT VECTOR MACHINE (SVM)



What if there is no such boundary?

We can usually transform the underlying datapoints in such a way that a linear boundary exists. Then we revert that transformation and our boundary stops being linear, but is still effective for classification.

This is called a Kernel trick and usually requires very good understanding of the data

SVM - SUMMARY

+

Can Model NonLinear Boundaries (if nonlinear version is used)

Unlikely to overfit

⊖

Memory intensive: is more successful on small dataset

CROSS VALIDATION

Just to make the process more confusing!

Currently we've learnt that the train-test split is performed in a random way.

But are we leaving information on the table? How do we know a super representative point is being placed in the test dataset thus not allowing our algorithm to "learn"?

This is where Cross Validation comes in

Cros Validation is the process of performing several different test-train splits

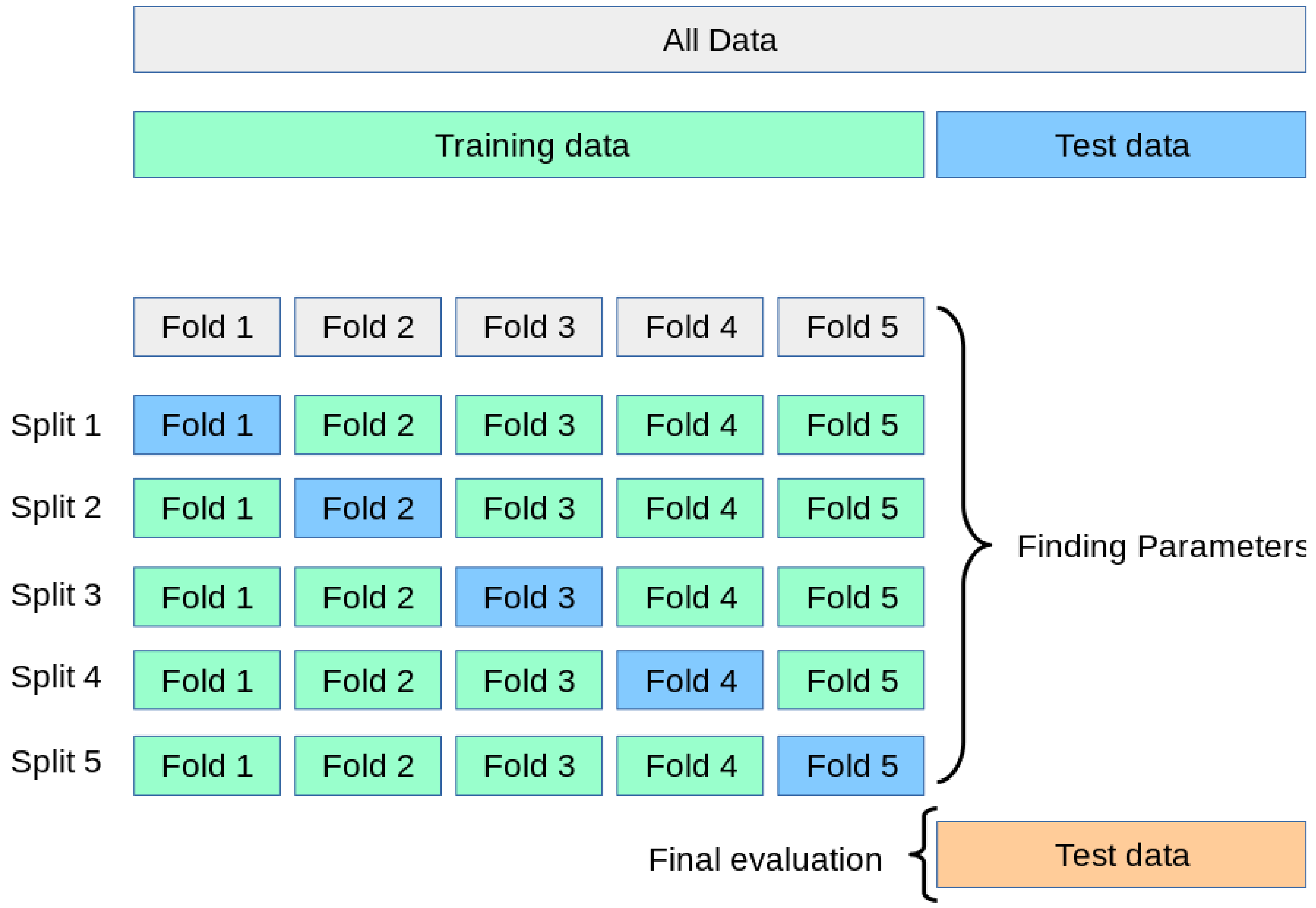
CROSS VALIDATION

Currently we've learnt that the train-test split is performed in a random way.

But are we leaving information on the table? How do we know a super representative point is being placed in the test dataset thus not allowing our algorithm to "learn"?

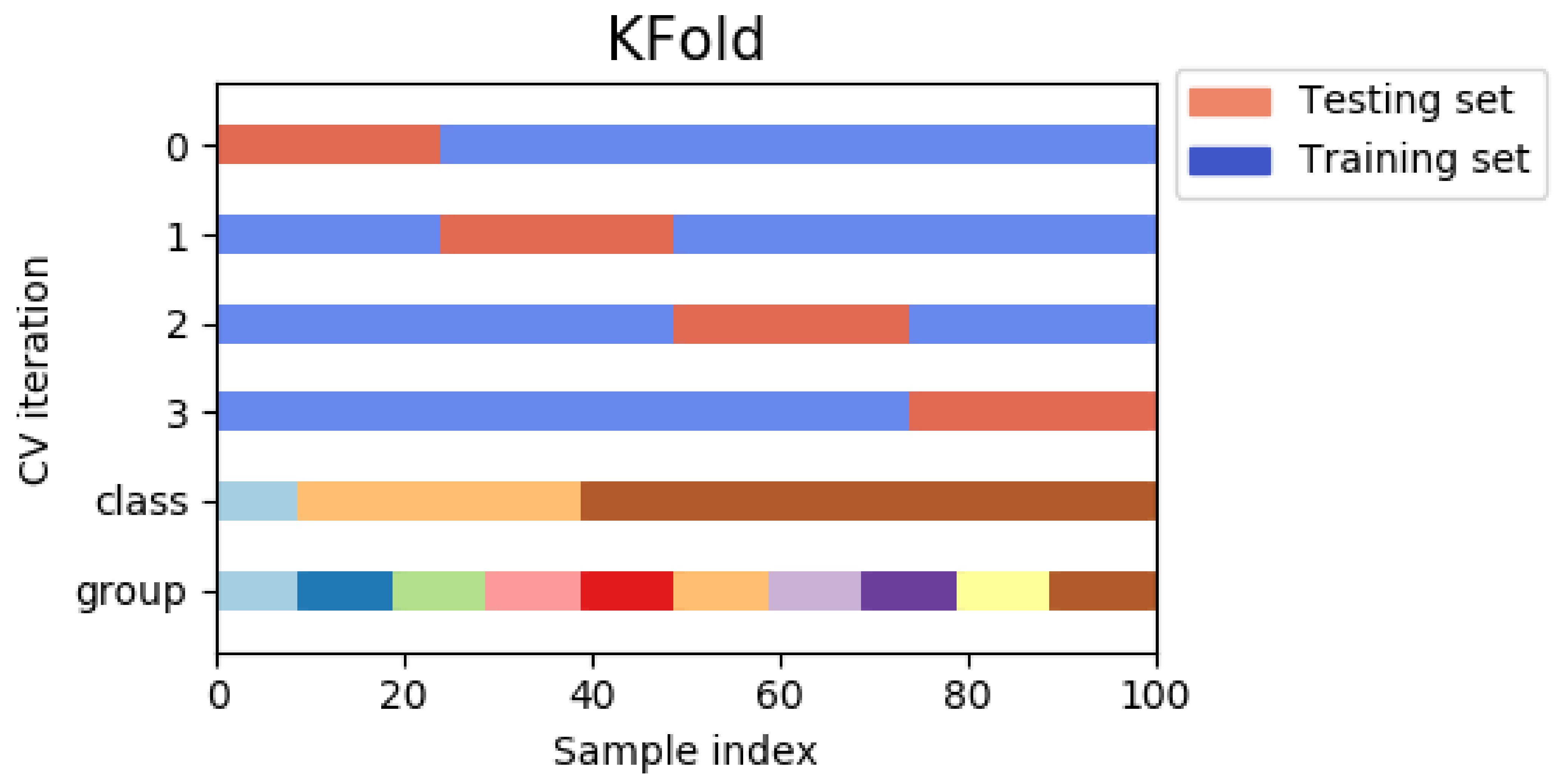
This is where Cross Validation comes in

Cros Validation is the process of performing several different test-train splits



CROSS VALIDATION

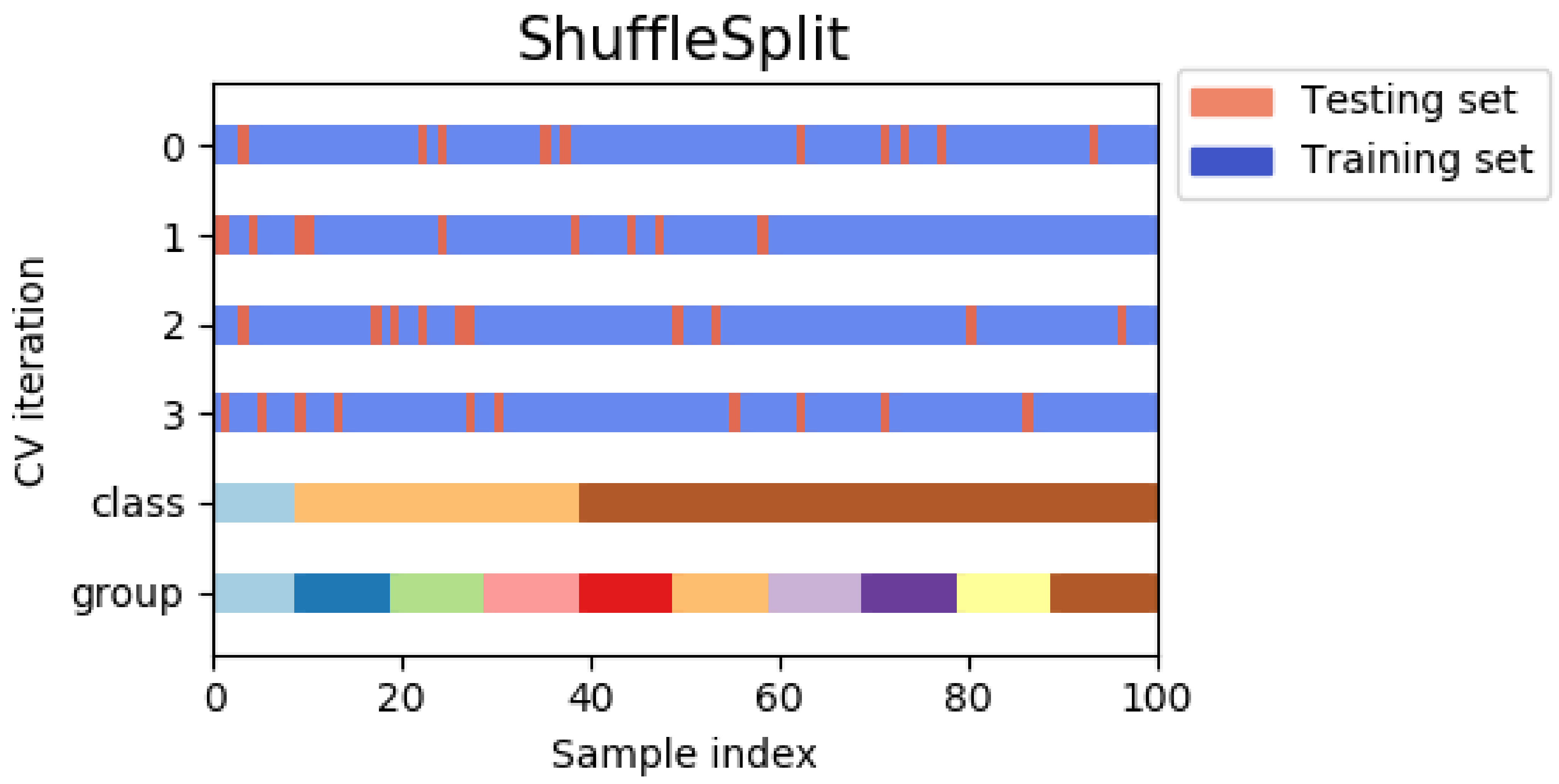
K-Fold Cross Validation



This is the simplest Cross Validation technique whereby data is split into k times into different Test-Train splits always preserving the length of each

CROSS VALIDATION

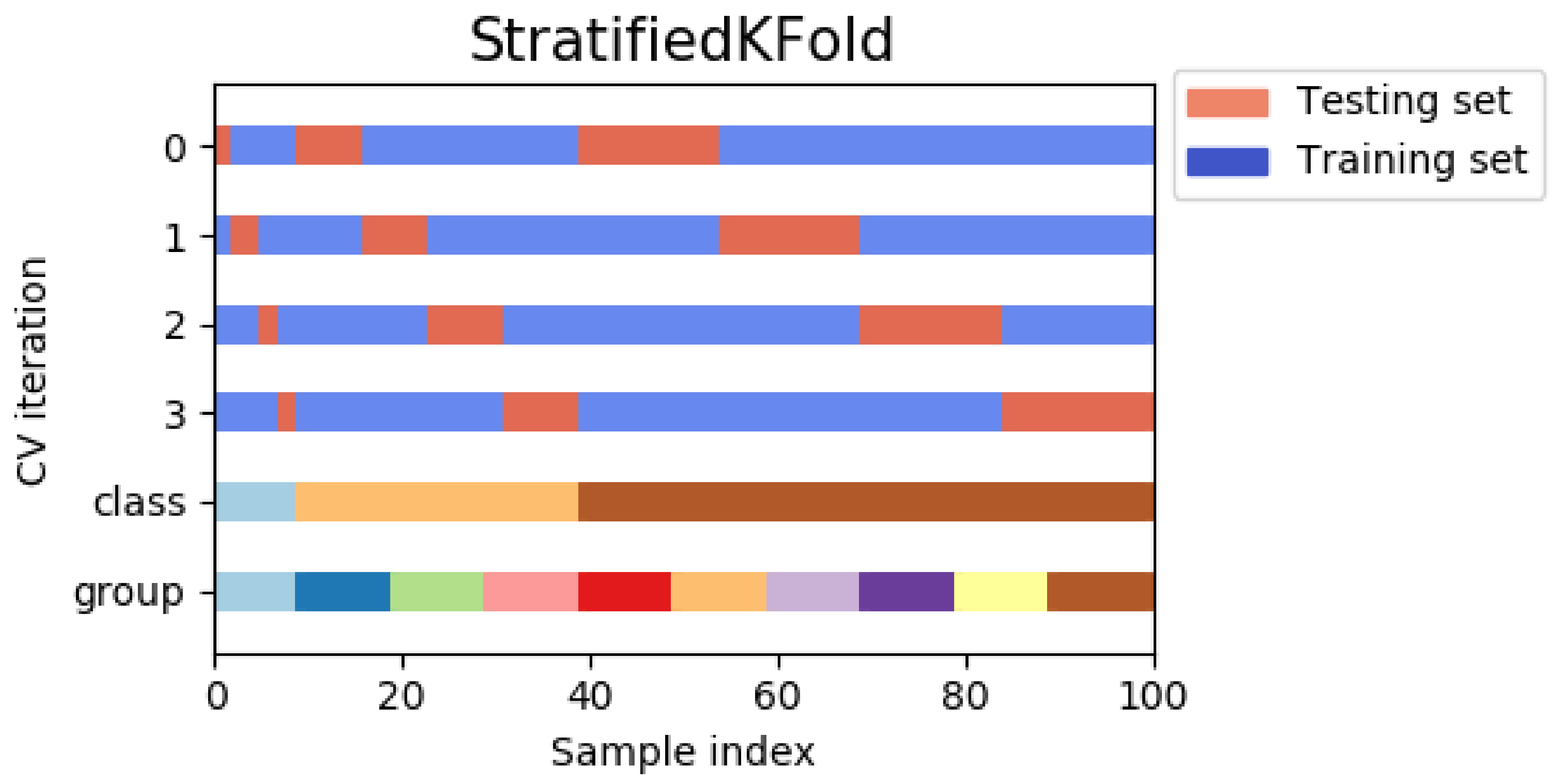
Shuffle Split Cross Validation



Performing a shuffled split further ensures more randomness is created when performing the split

CROSS VALIDATION

Stratified KFold

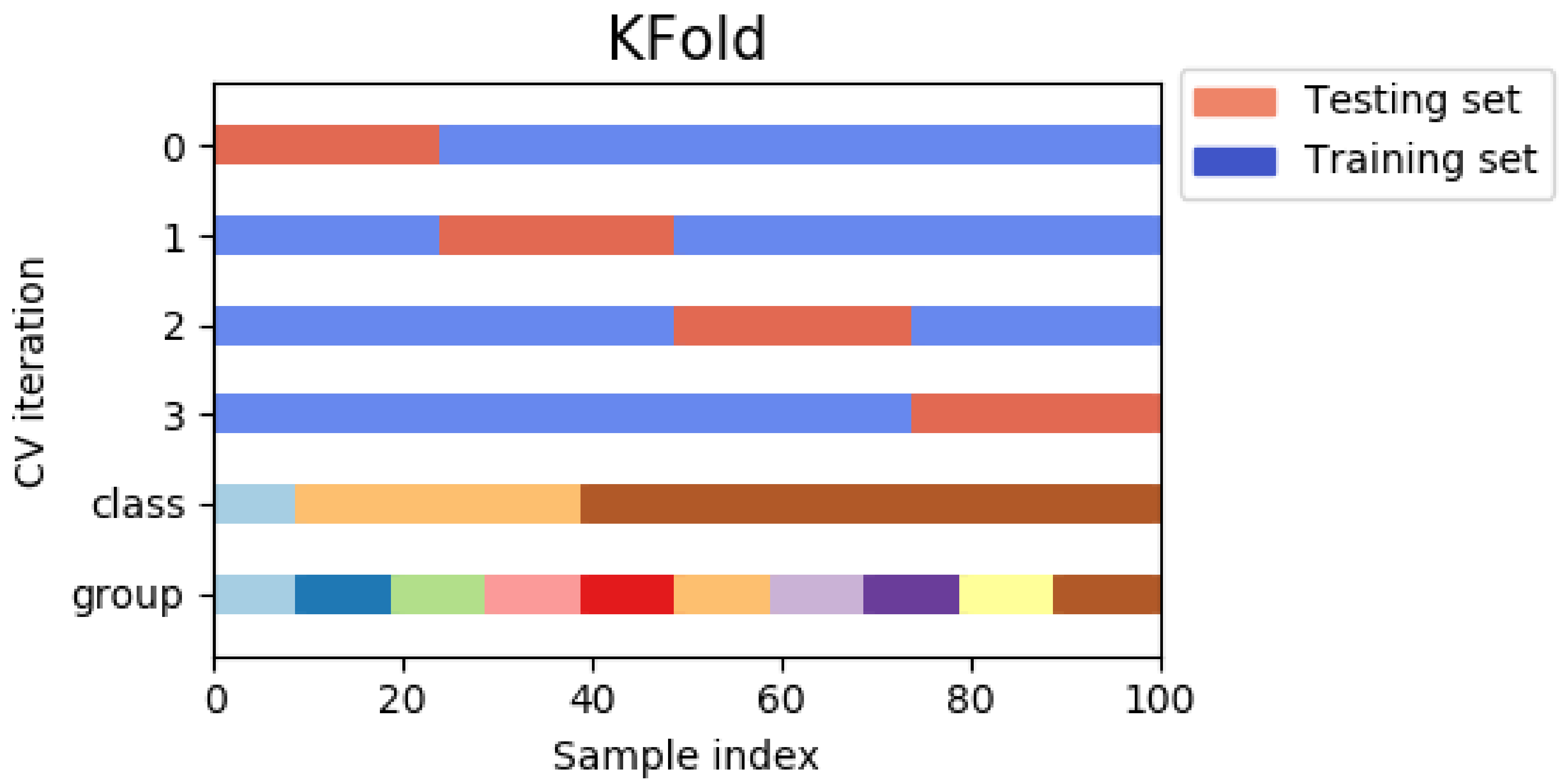


Whenever you have a severe class imbalance you may want to perform stratified sampling: you create your splits always preserving the same ratio of each class types (1/10)

CROSS VALIDATION

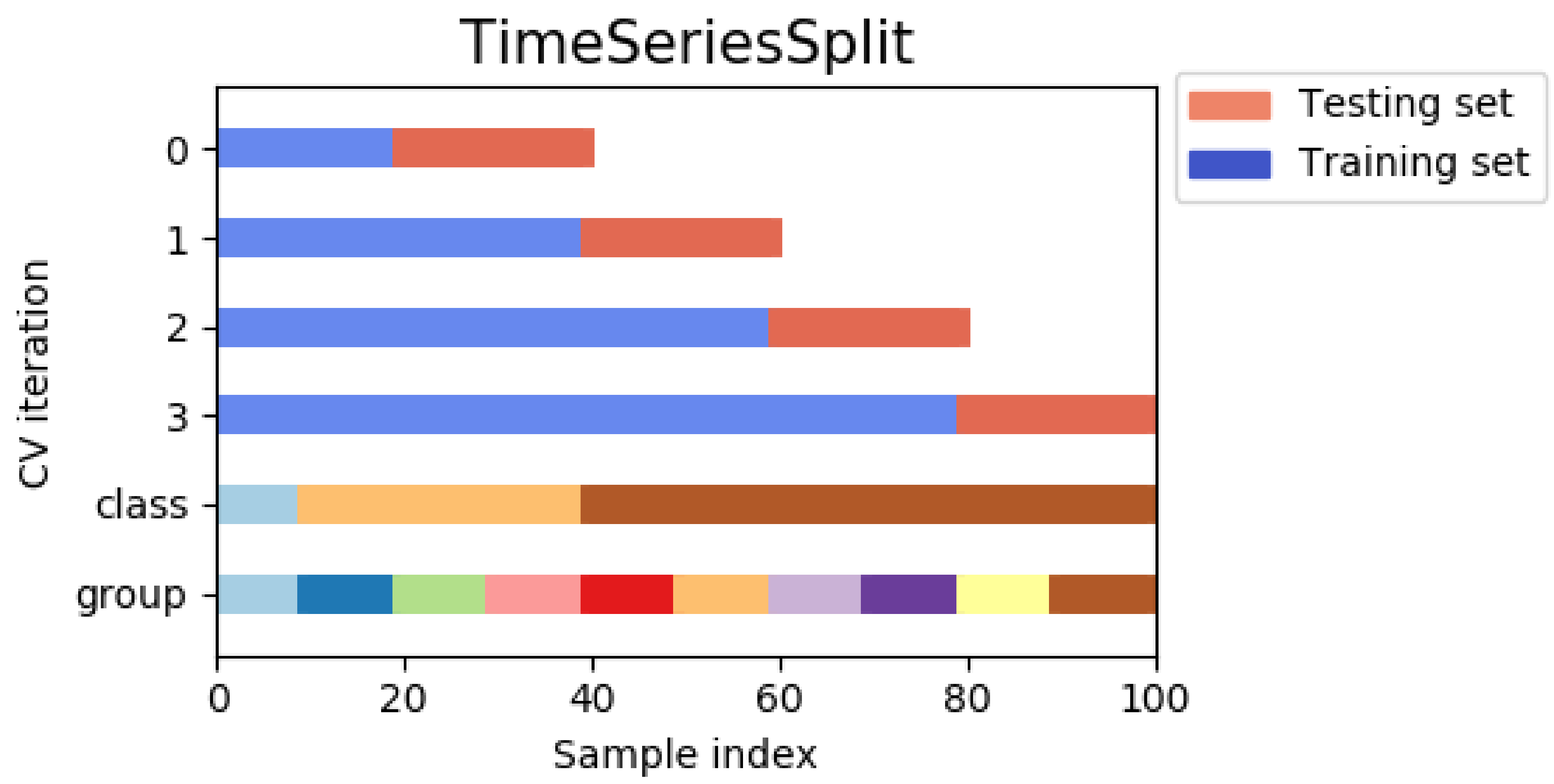
How about the train-testing process for time series data?

Consider the problem if we used K-Fold



CROSS VALIDATION

How about the train-testing process for time series data?



Super common mistake that I see in using Time Series Data is to use data from the future to build a model to predict the past!!