# IAP Gesture Recognition Workshop

## Session 1:
## Gesture Recognition & Machine Learning Fundamentals

Nicholas Gillian

Responsive Environments, MIT Media Lab

Tuesday 8th January, 2013

# My Research

# My Research

- Gesture Recognition for Musician Computer Interaction

# My Research

- Gesture Recognition for Musician Computer Interaction
- Rapid Learning

# My Research

- Gesture Recognition for Musician Computer Interaction
- Rapid Learning
- Free-air Gestures & Fine-grain Control

# My Research

- Gesture Recognition for Musician Computer Interaction
- Rapid Learning
- Free-air Gestures & Fine-grain Control
- Creating tools and software that enable a more diverse group of individuals to integrate gesture-recognition into their own interfaces, art installations, and musical instruments

# My Research

- Gesture Recognition for Musician Computer Interaction
- Rapid Learning
- Free-air Gestures & Fine-grain Control
- Creating tools and software that enable a more diverse group of individuals to integrate gesture-recognition into their own interfaces, art installations, and musical instruments

EyesWeb

GRT

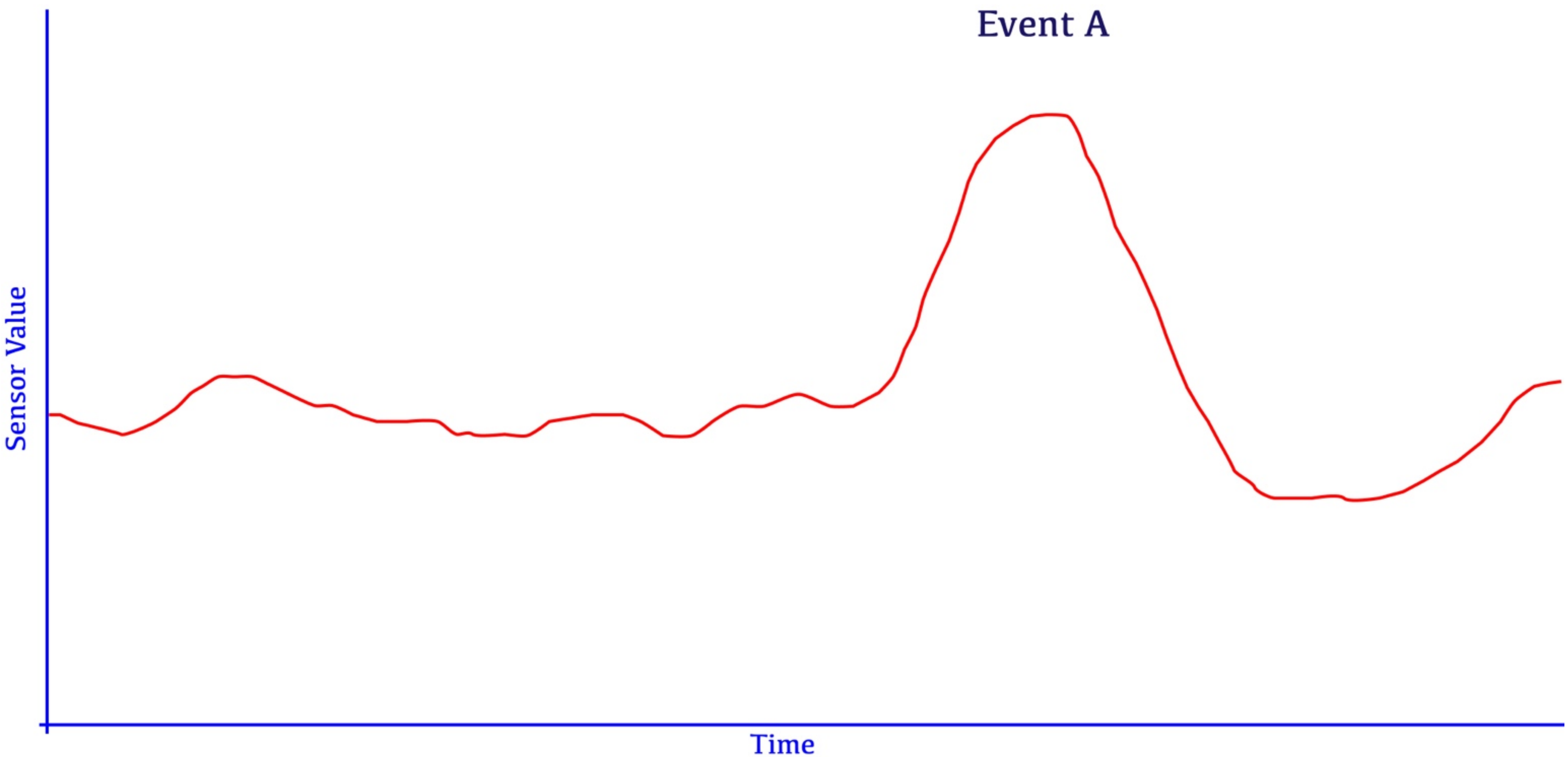
Gesture Recognition Toolkit

# Schedule

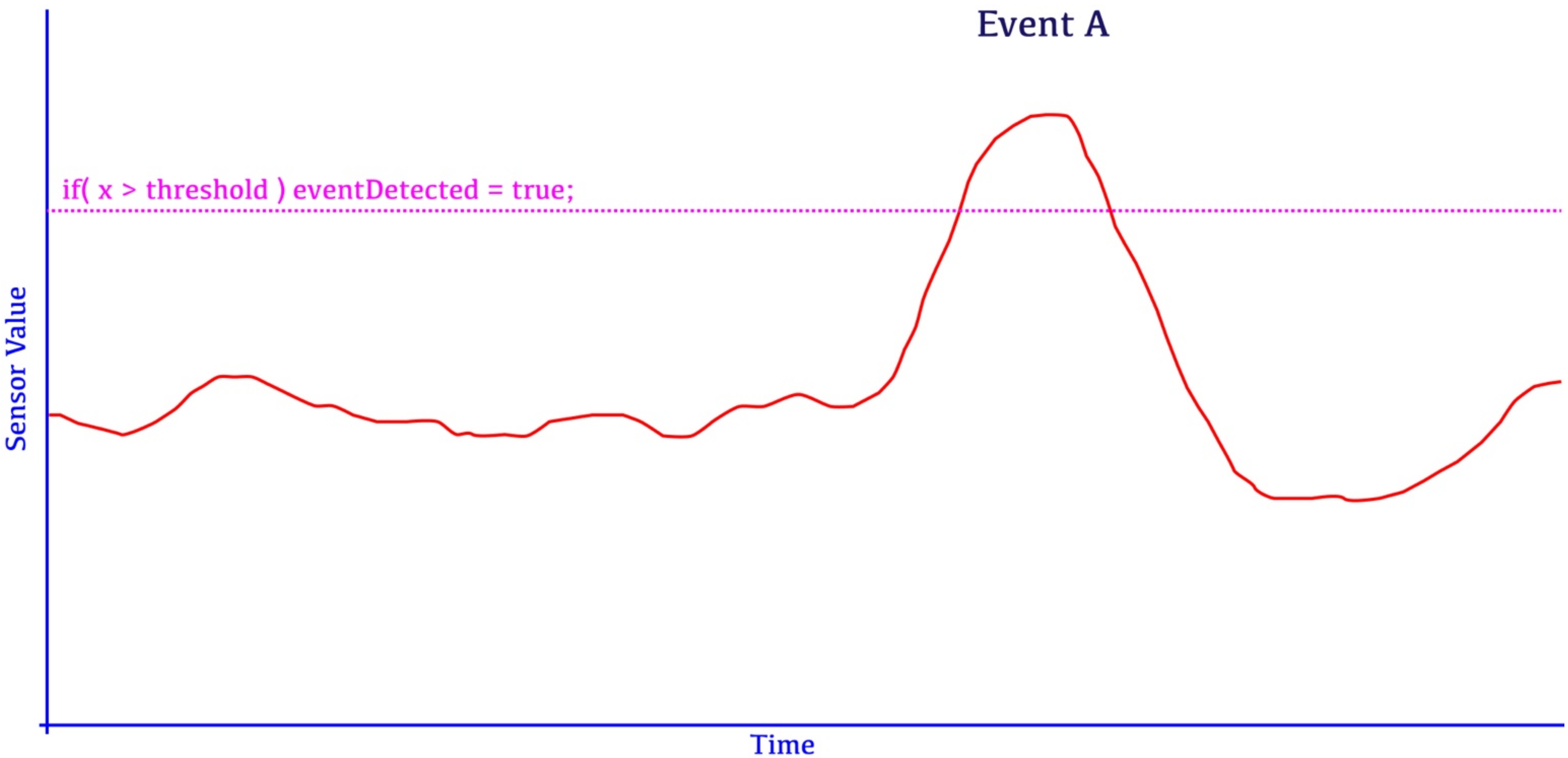
- Machine Learning 101
- Hello World
- Gesture Recognition
- Installation & Setup
- Introduction to the Gesture Recognition Toolkit
- Lunch
- Hands-on Coding Sessions

# Basic Pattern Recognition Problem

# Basic Pattern Recognition Problem

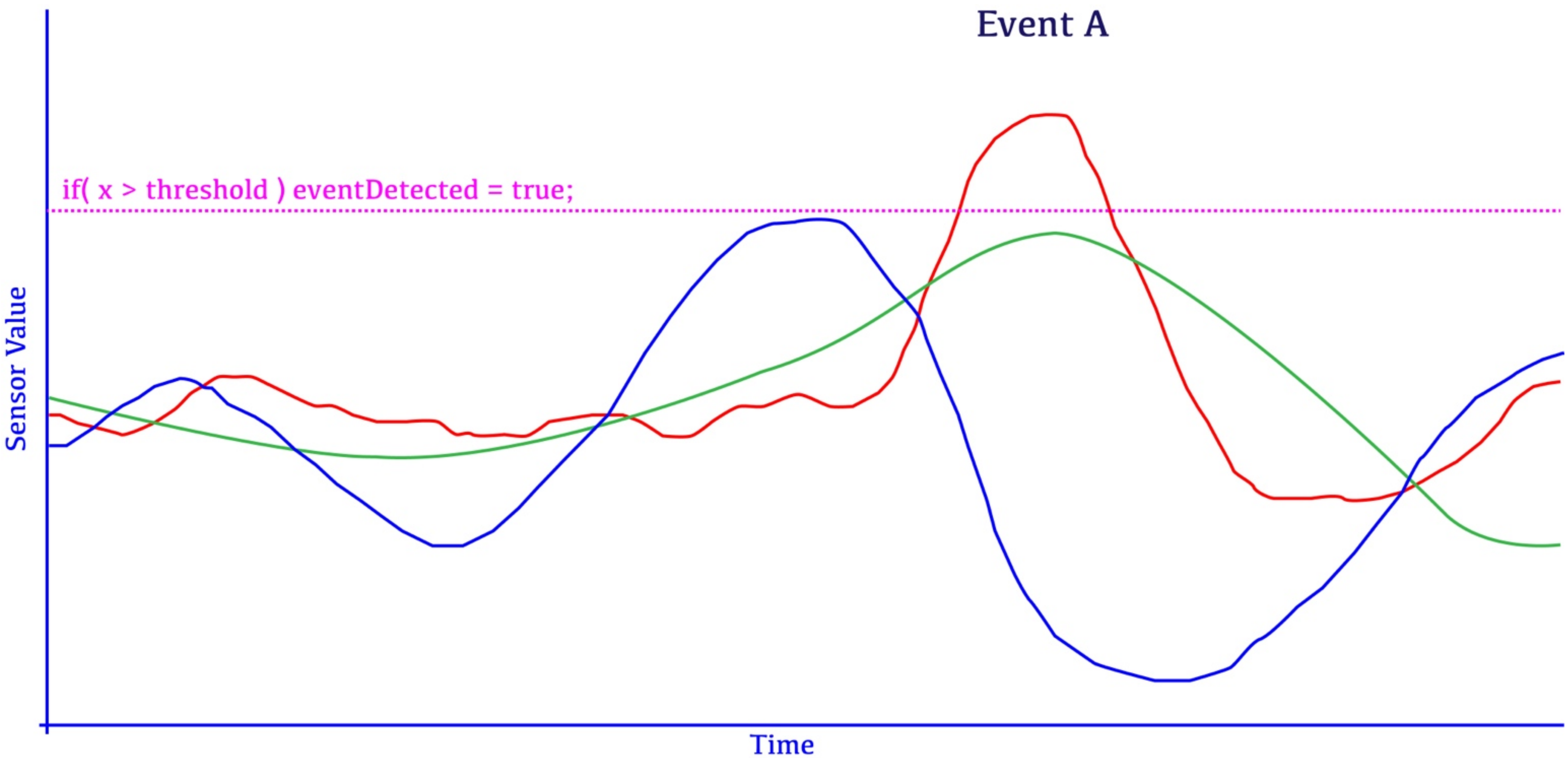


# Basic Pattern Recognition Problem



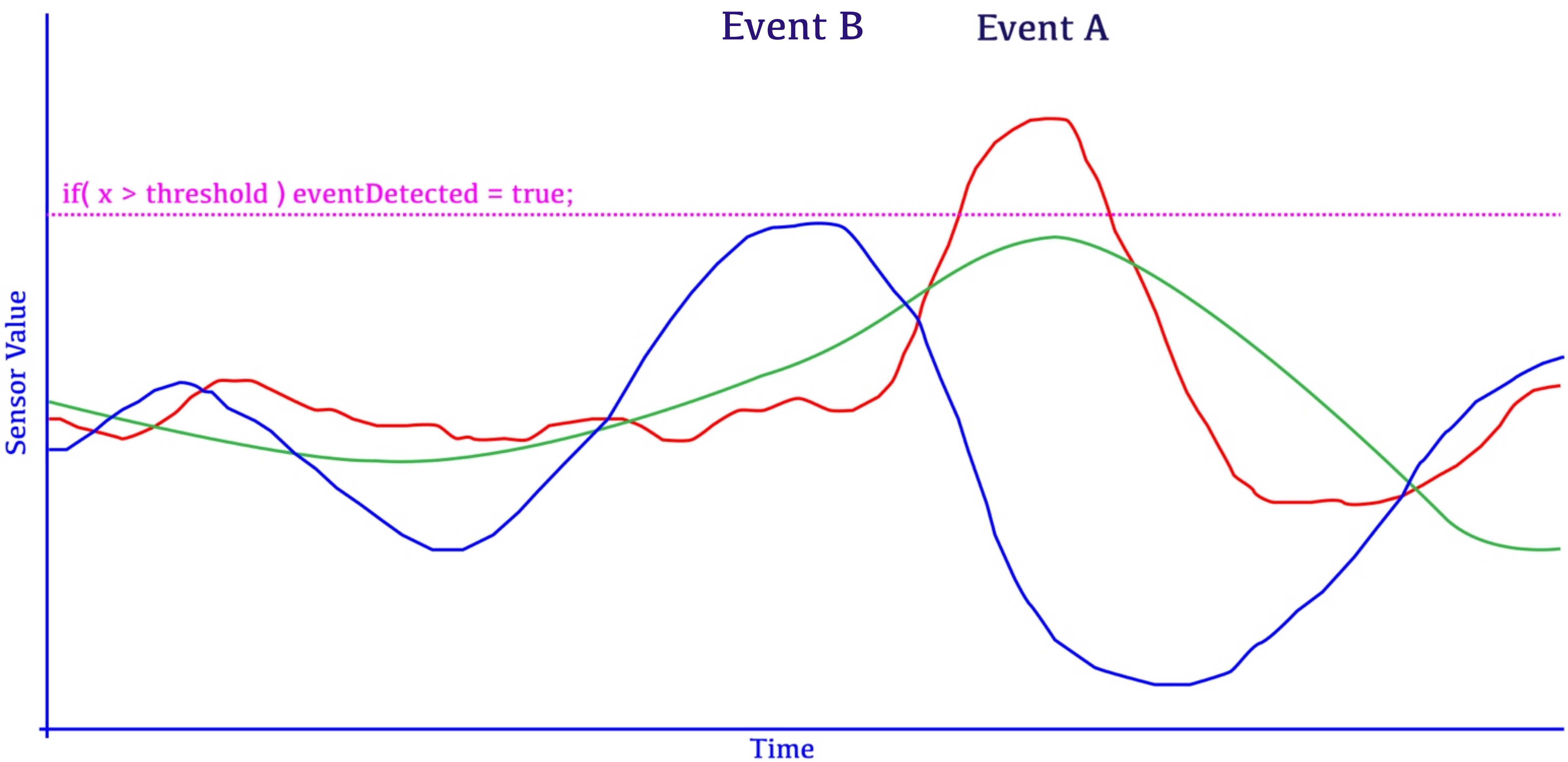

Might work for simple cases…

# Basic Pattern Recognition Problem




Can be more difficult with multidimensional data!

# Basic Pattern Recognition Problem

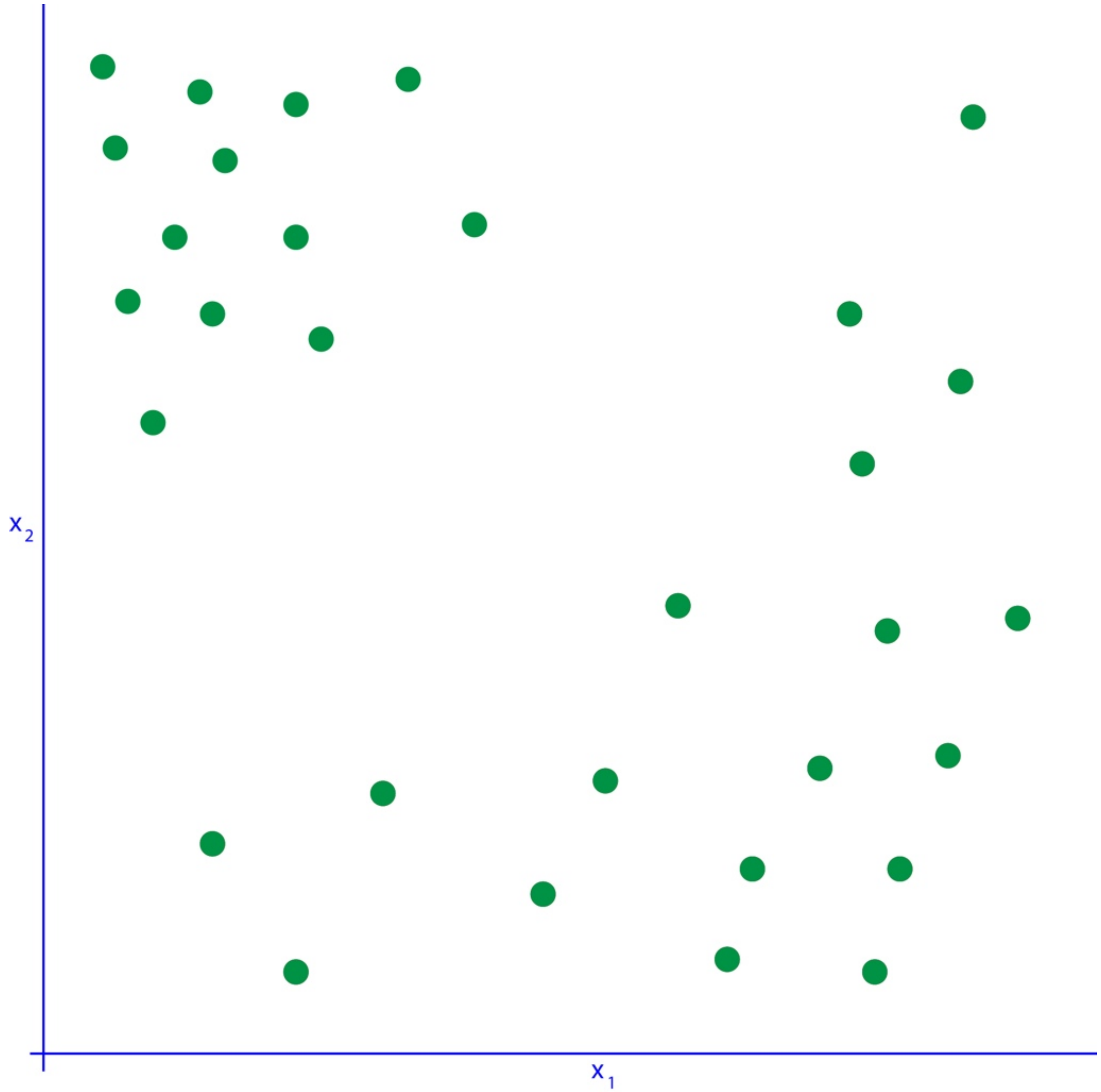



Can be more difficult with multiple events!

# Machine Learning

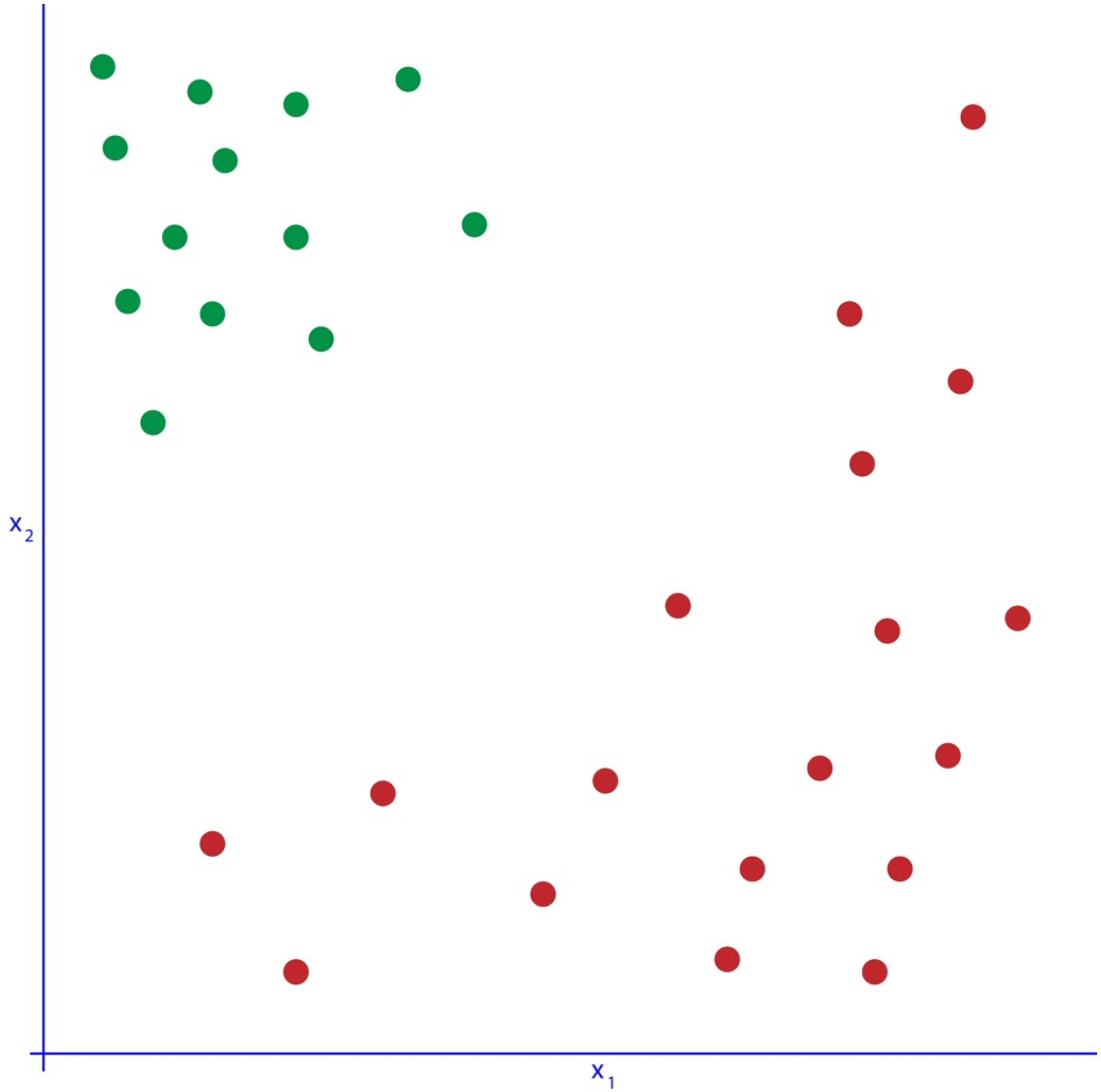
Machine Learning 101

# Machine Learning



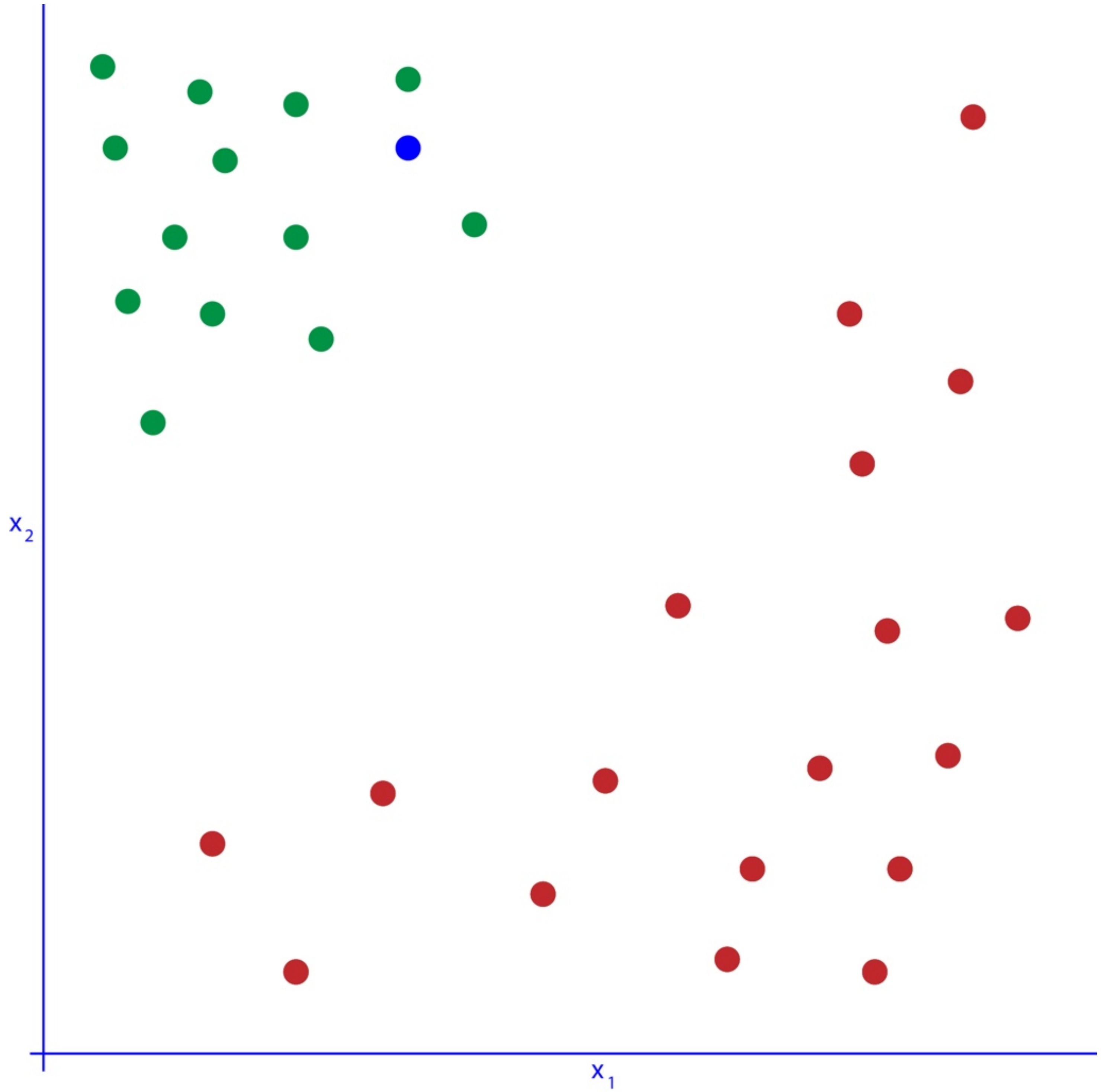

Dataset

# Machine Learning



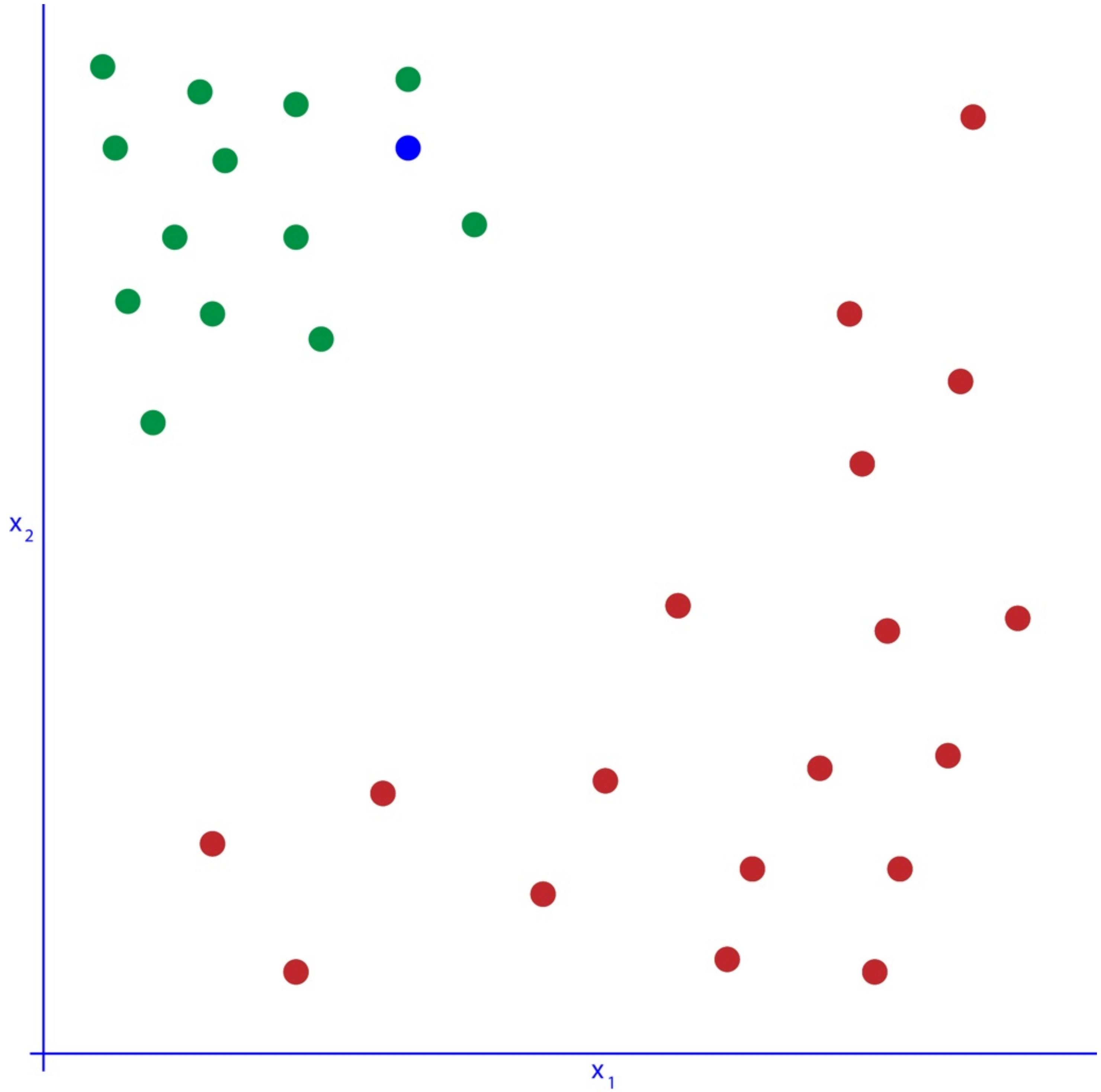

ML can automatically infer the underlying behavior/rules of this data

# Machine Learning



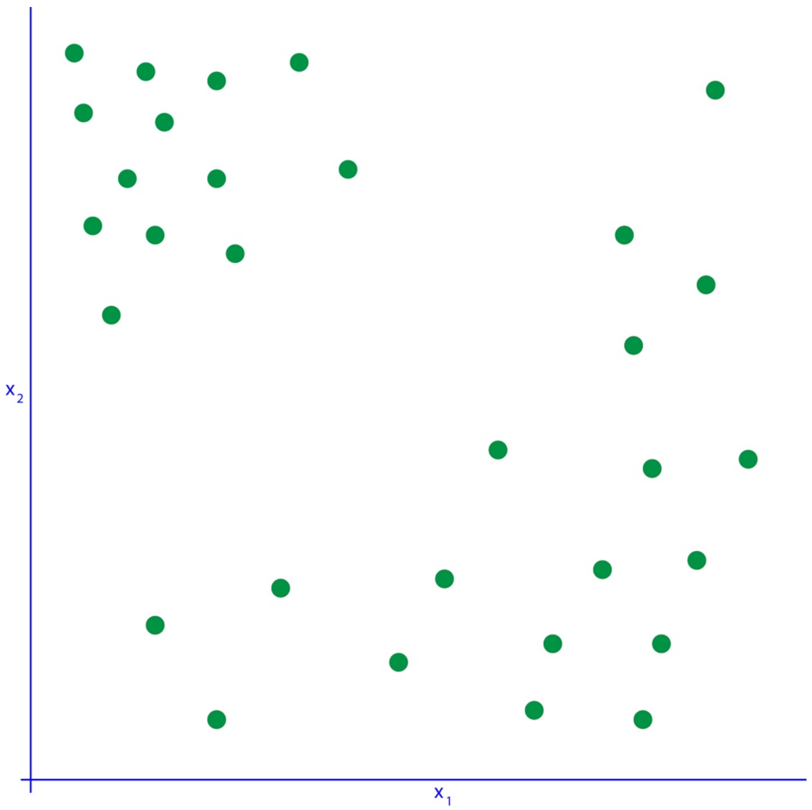

These rules can then be used to make predictions about future data

# Machine Learning

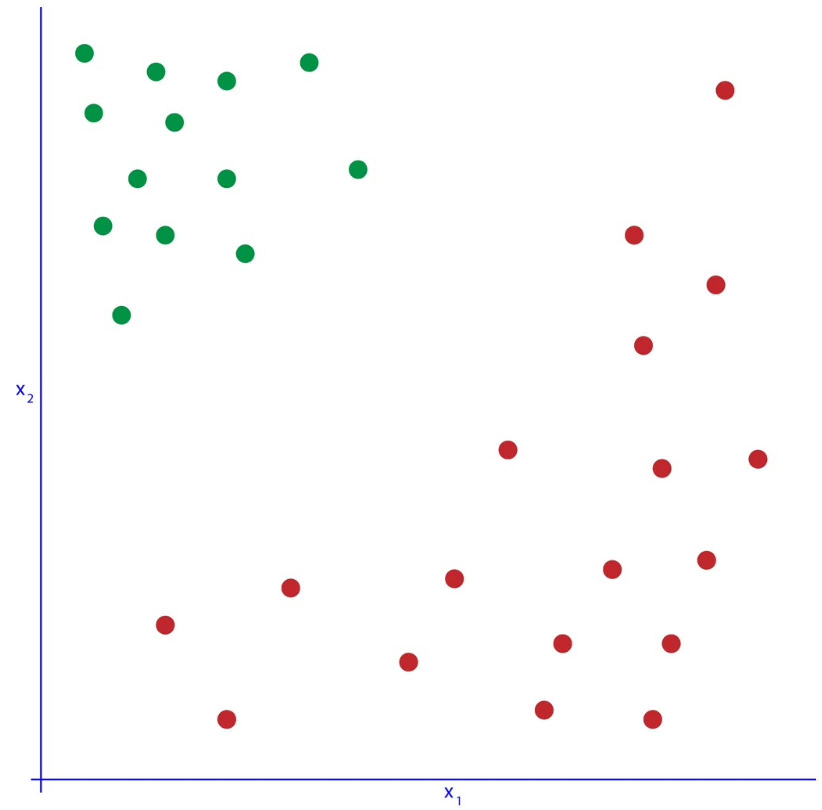


# Machine Learning

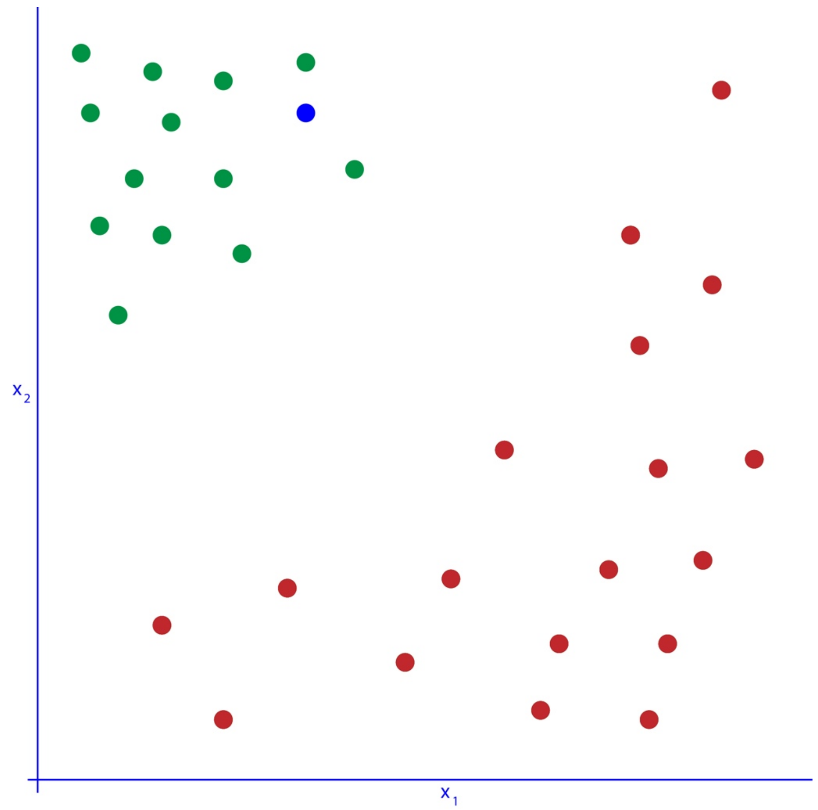
The three main phases of machine learning:




Data Collection




Learning




Prediction

# Machine Learning

Machine Learning is commonly used to solve two main problems:

# Machine Learning

Machine Learning is commonly used to solve two main problems:

CLASSIFICATION

# Machine Learning

Machine Learning is commonly used to solve two main problems:

CLASSIFICATION

# Machine Learning

Machine Learning is commonly used to solve two main problems:

CLASSIFICATION

# Machine Learning

Machine Learning is commonly used to solve two main problems:

CLASSIFICATION

# Machine Learning

Machine Learning is commonly used to solve two main problems:

CLASSIFICATION

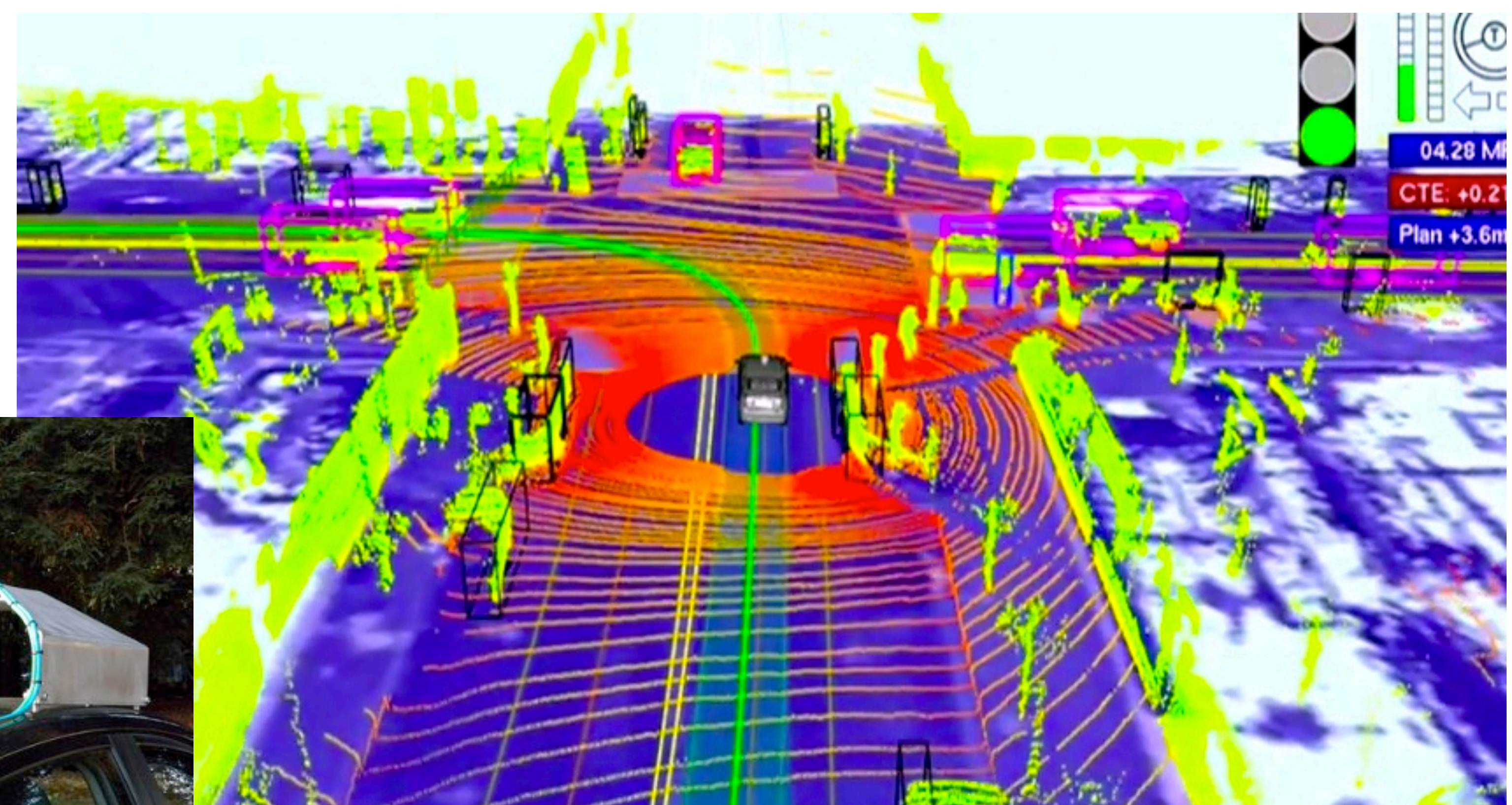
REGRESSION

# Machine Learning

Machine Learning is commonly used to solve two main problems:

CLASSIFICATION

REGRESSION



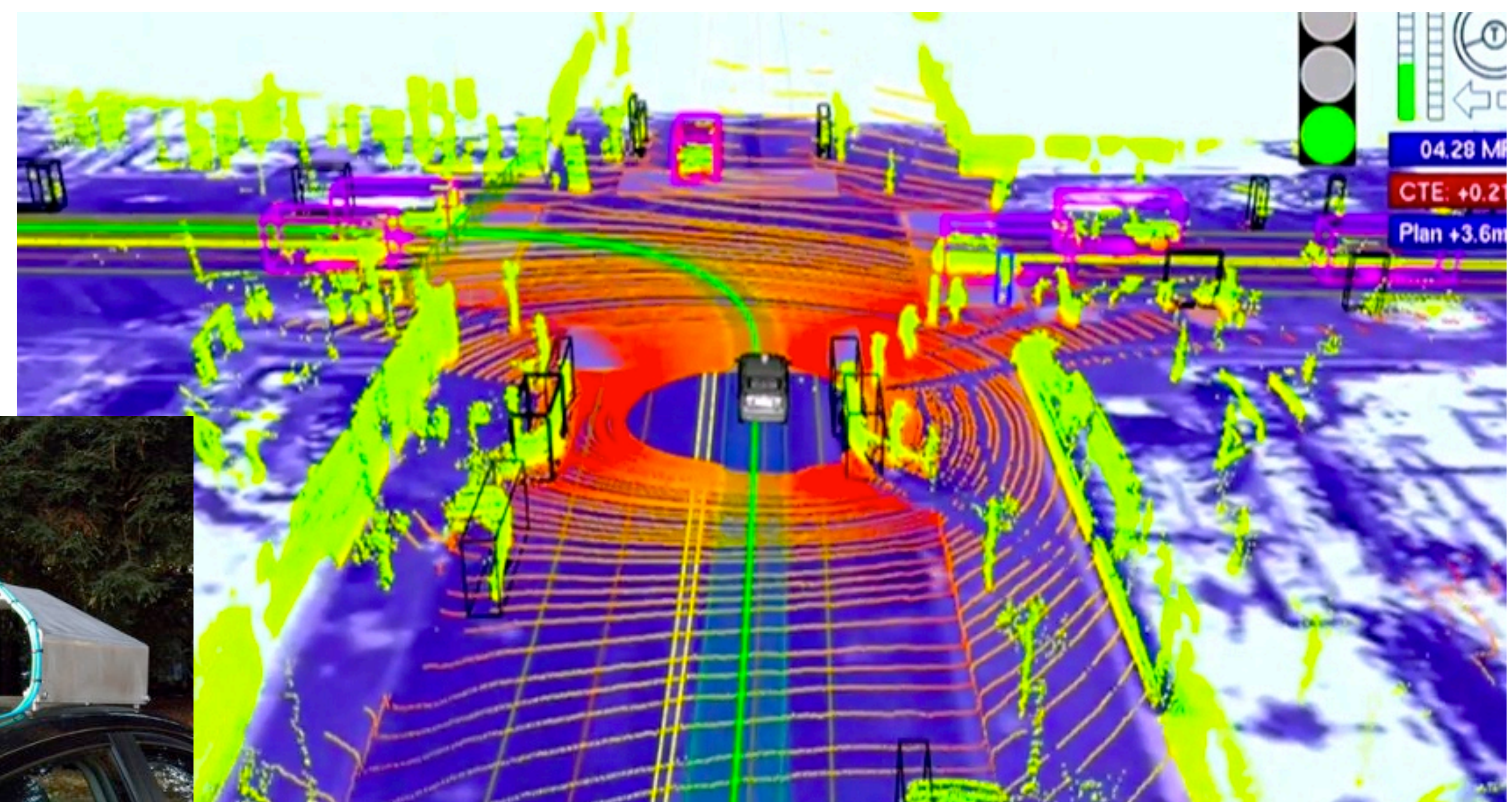
# Machine Learning

Machine Learning is commonly used to solve two main problems:

CLASSIFICATION $$f : \mathbf{x} \longmapsto [0, 1, 2, 3, \ldots, K]$$

Discrete Output, representing the most likely class that the input **x** belongs to

REGRESSION




# Machine Learning

Machine Learning is commonly used to solve two main problems:

CLASSIFICATION

$$f : \mathbf{x} \longmapsto [0, 1, 2, 3, \ldots, K]$$

Discrete Output, representing the most likely class that the input **x** belongs to

REGRESSION

$$f : \mathbf{x} \longmapsto [-1.4, 2.6, 5.2, \ldots]$$

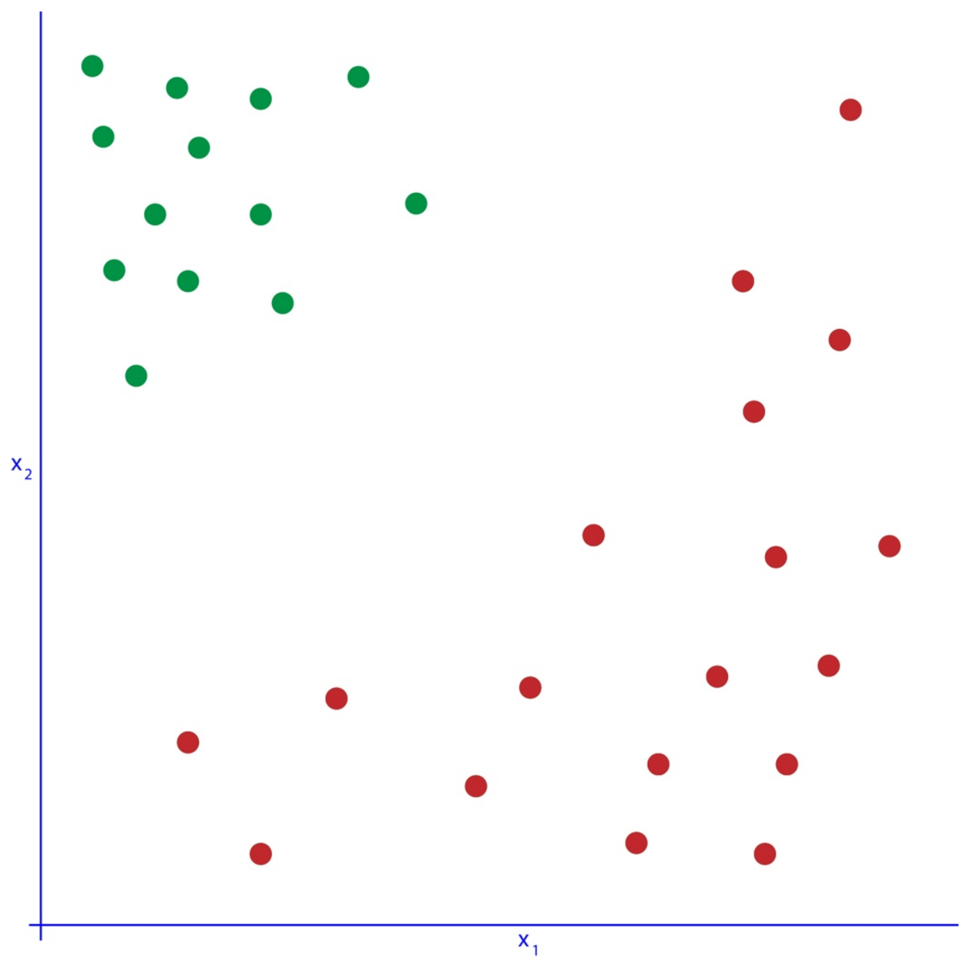
Continuous Output, mapping the ***N*** dimensional input vector **x** to an ***M*** dimensional vector **y**

# Machine Learning

Main types of learning:

# Machine Learning

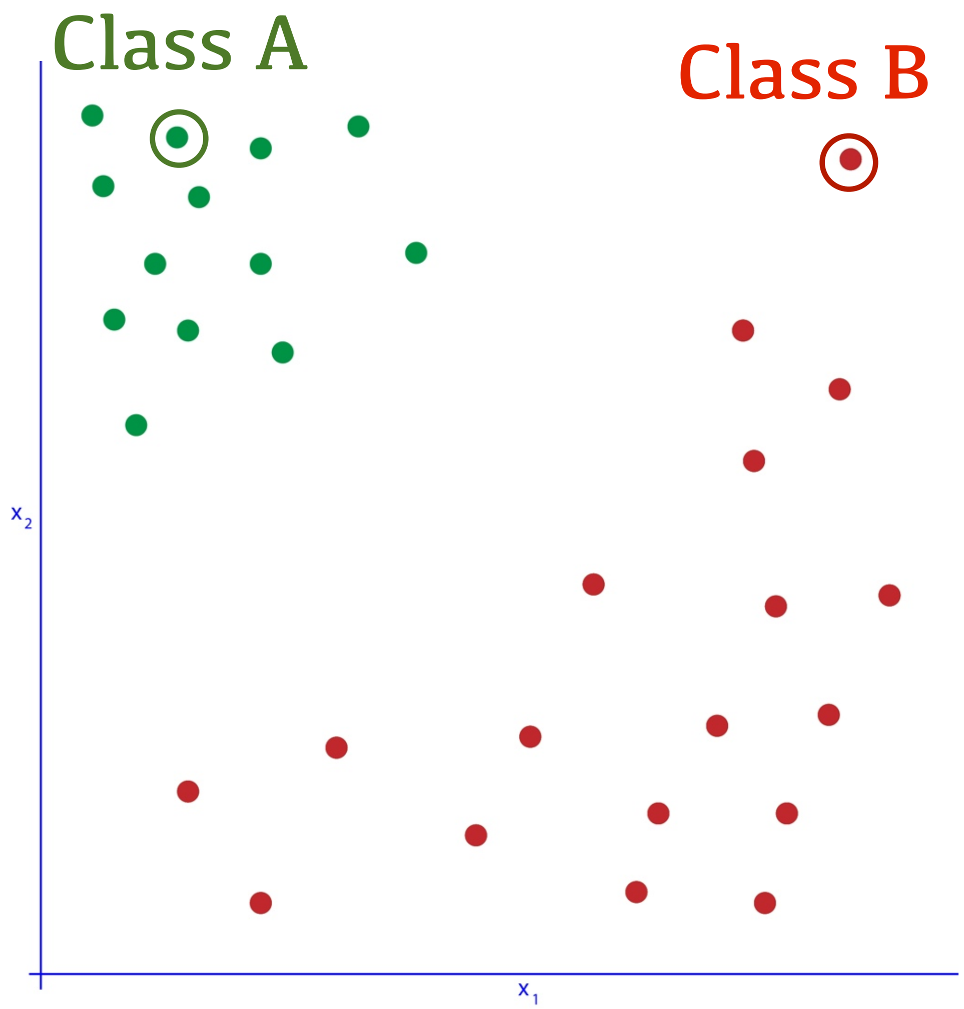
Main types of learning:




SUPERVISED LEARNING

# Machine Learning

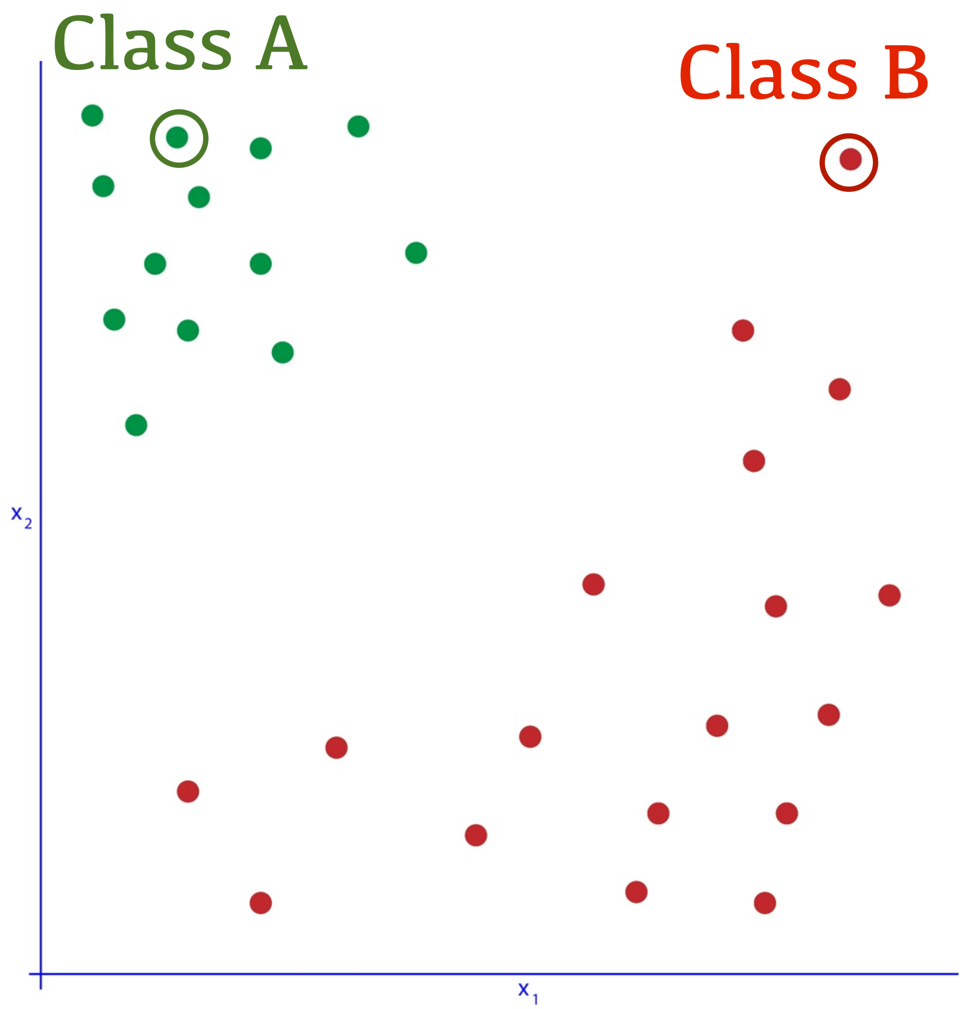
Main types of learning:



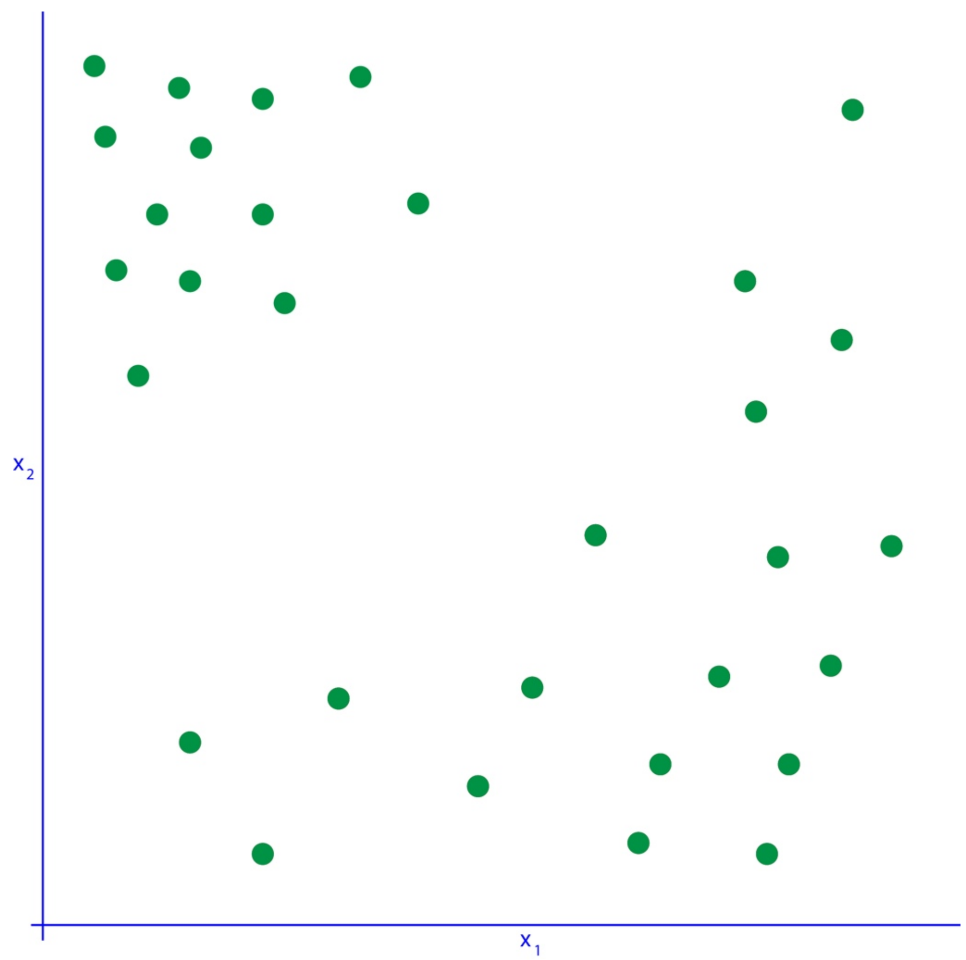

SUPERVISED LEARNING

# Machine Learning

Main types of learning:



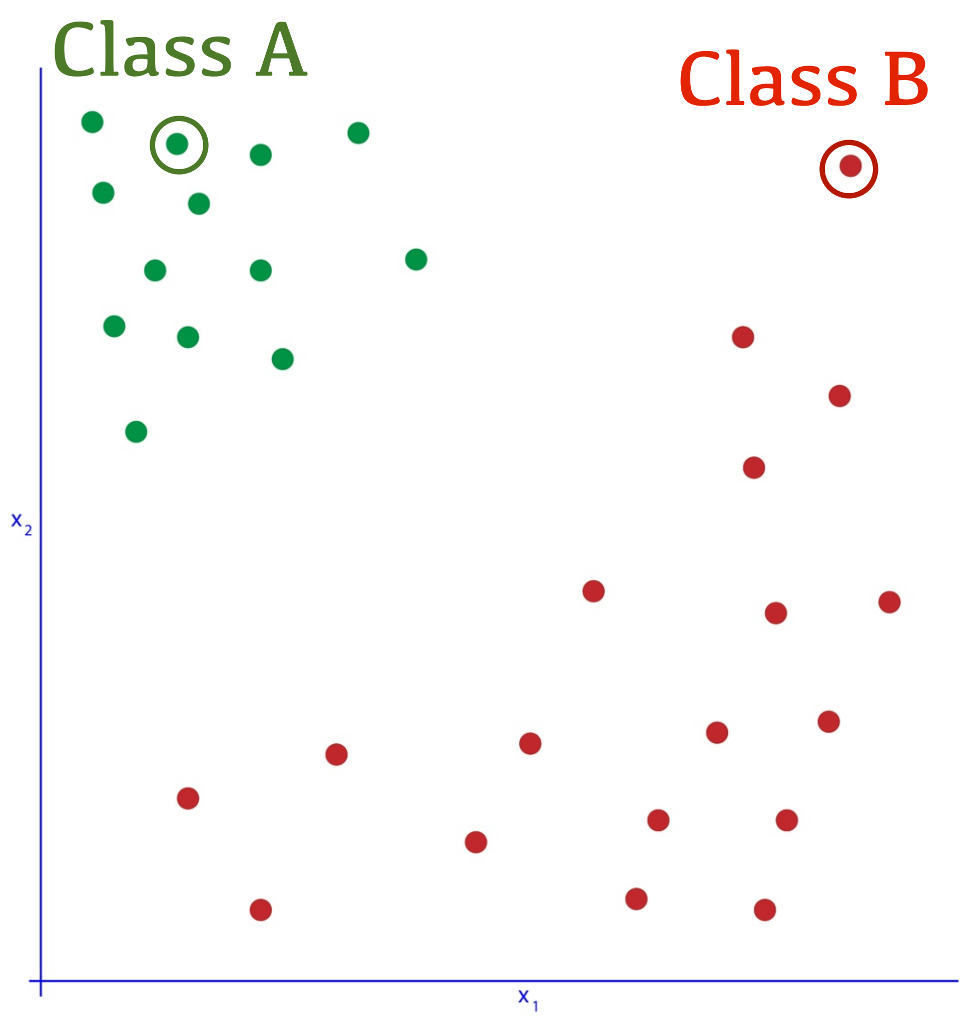

SUPERVISED LEARNING



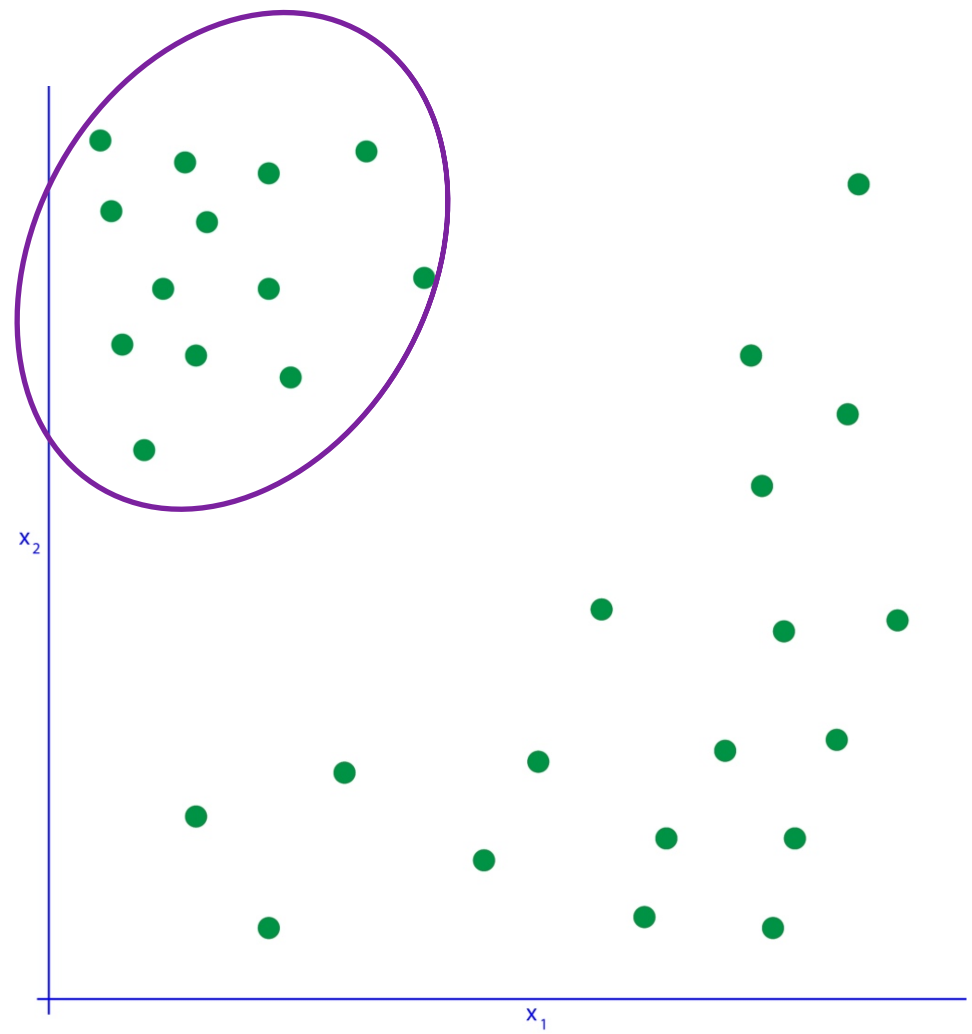

UNSUPERVISED LEARNING

# Machine Learning

Main types of learning:



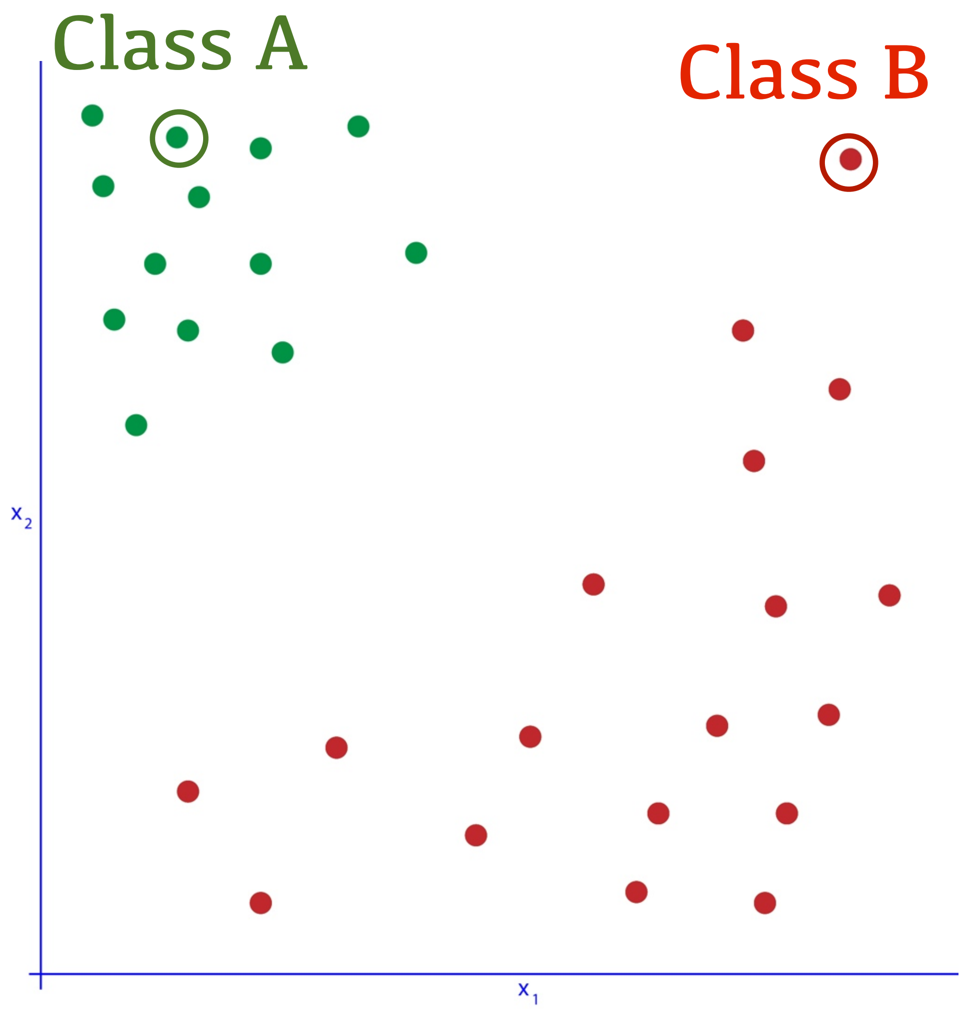

SUPERVISED LEARNING



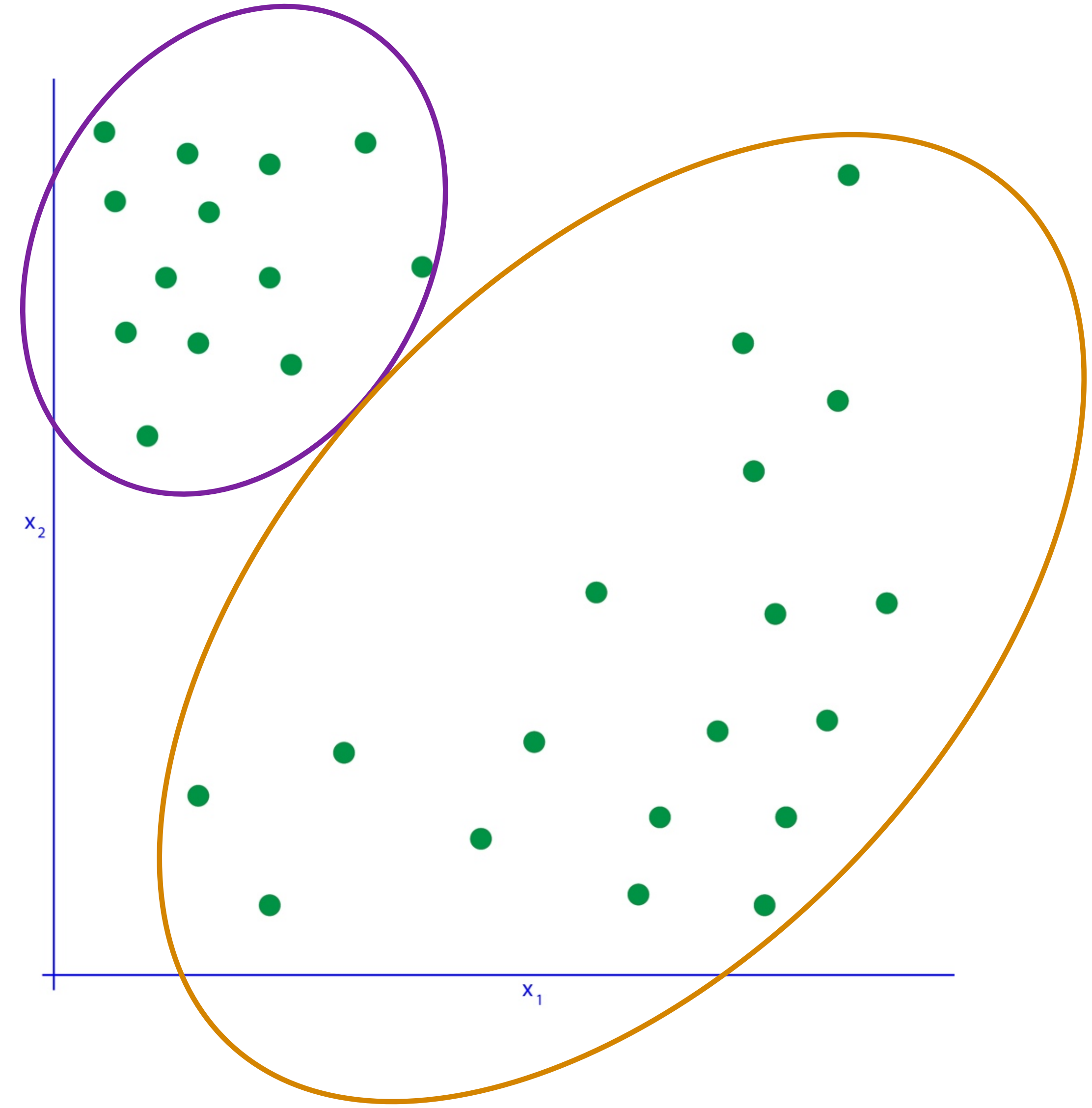

UNSUPERVISED LEARNING

# Machine Learning

Main types of learning:



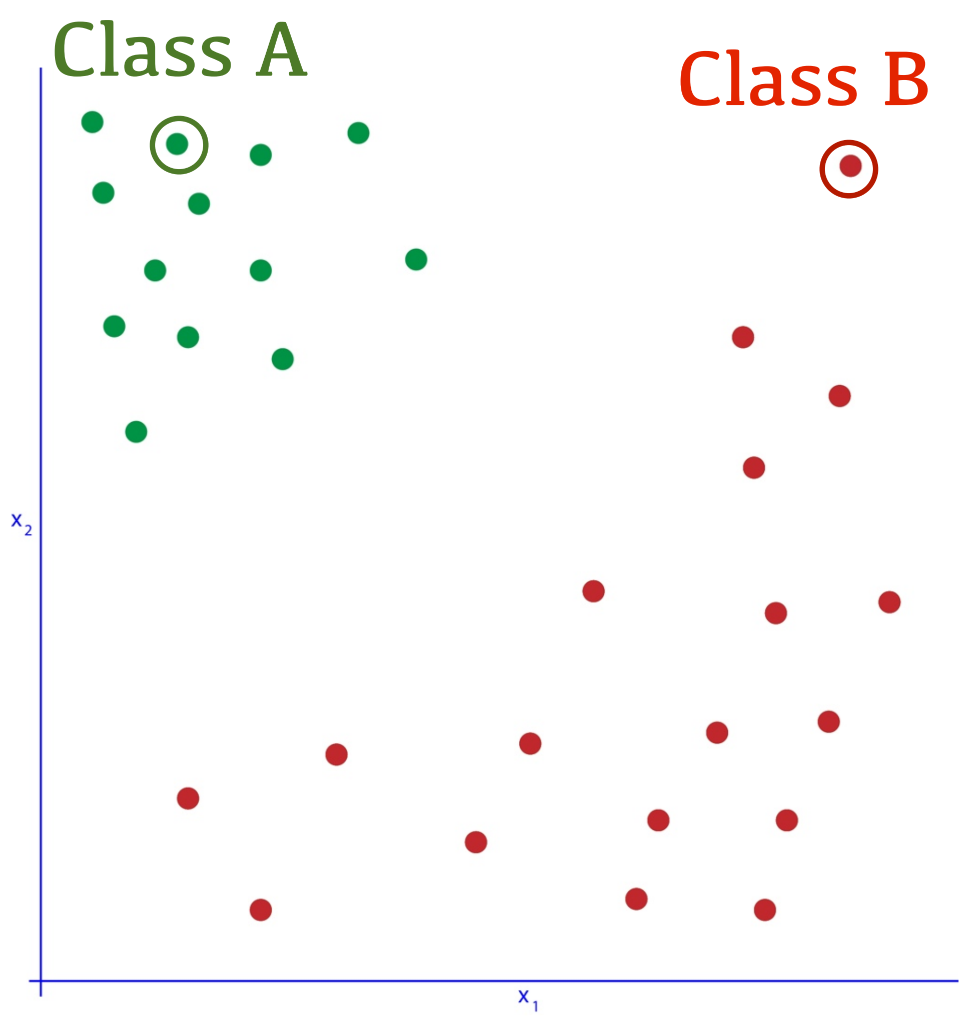

SUPERVISED LEARNING



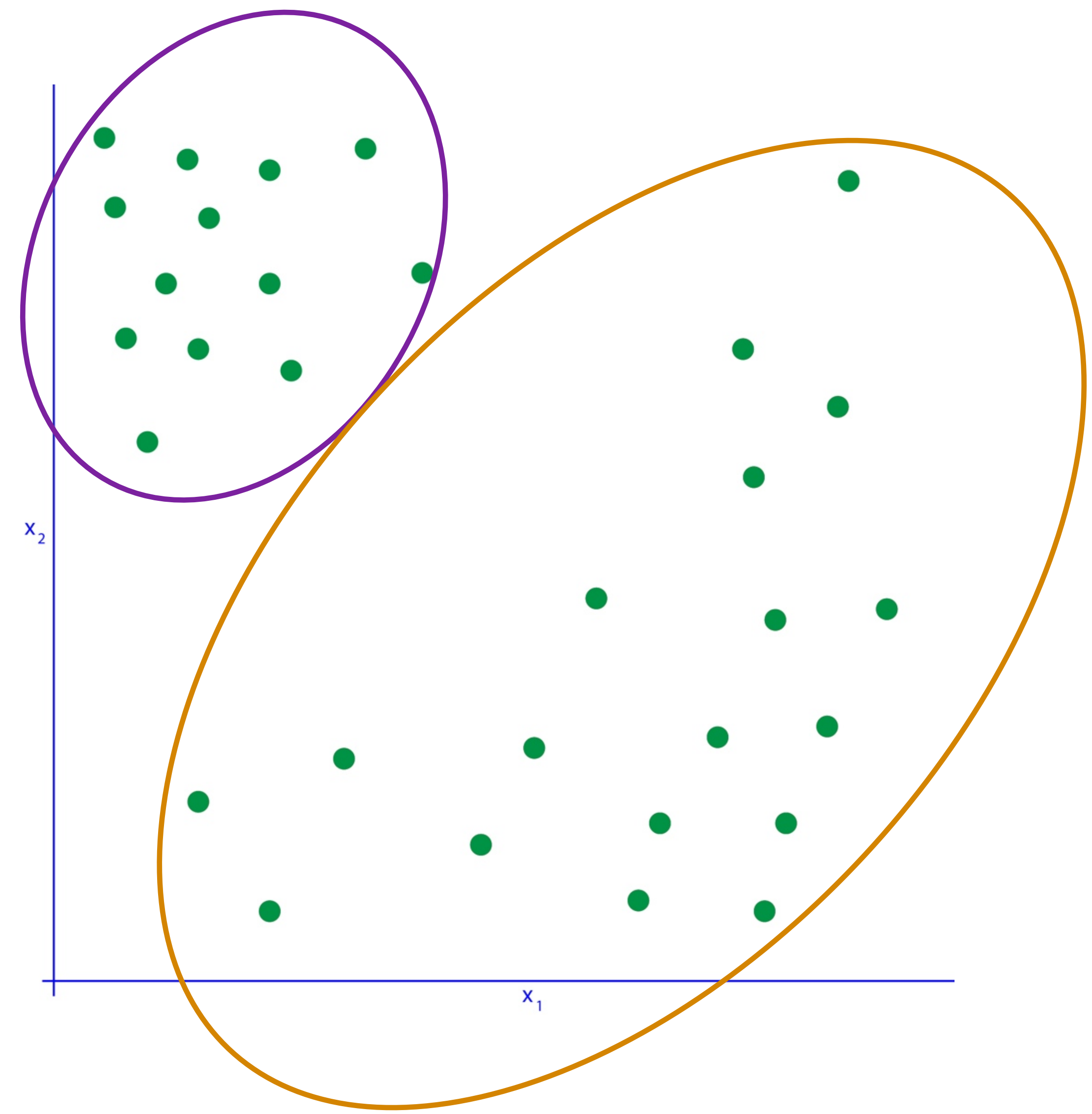

UNSUPERVISED LEARNING

# Machine Learning

Main types of learning:




SUPERVISED LEARNING

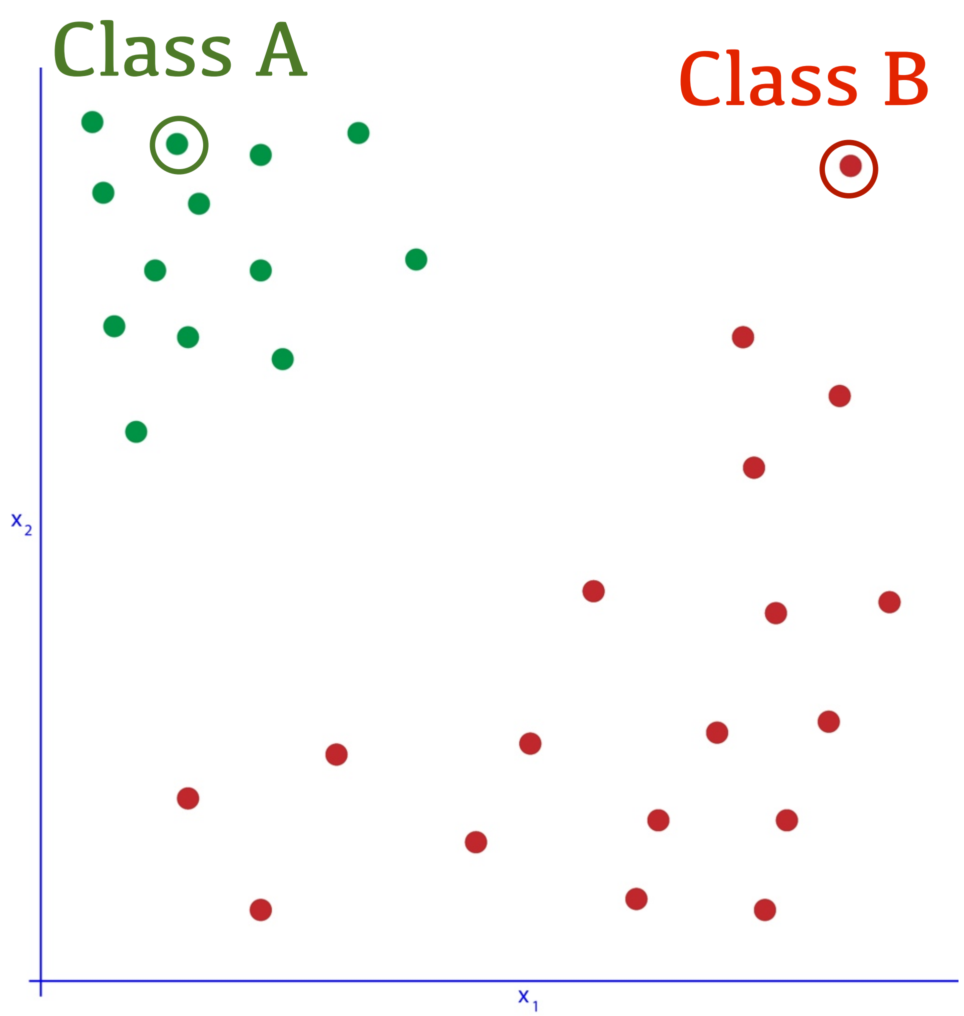



UNSUPERVISED LEARNING

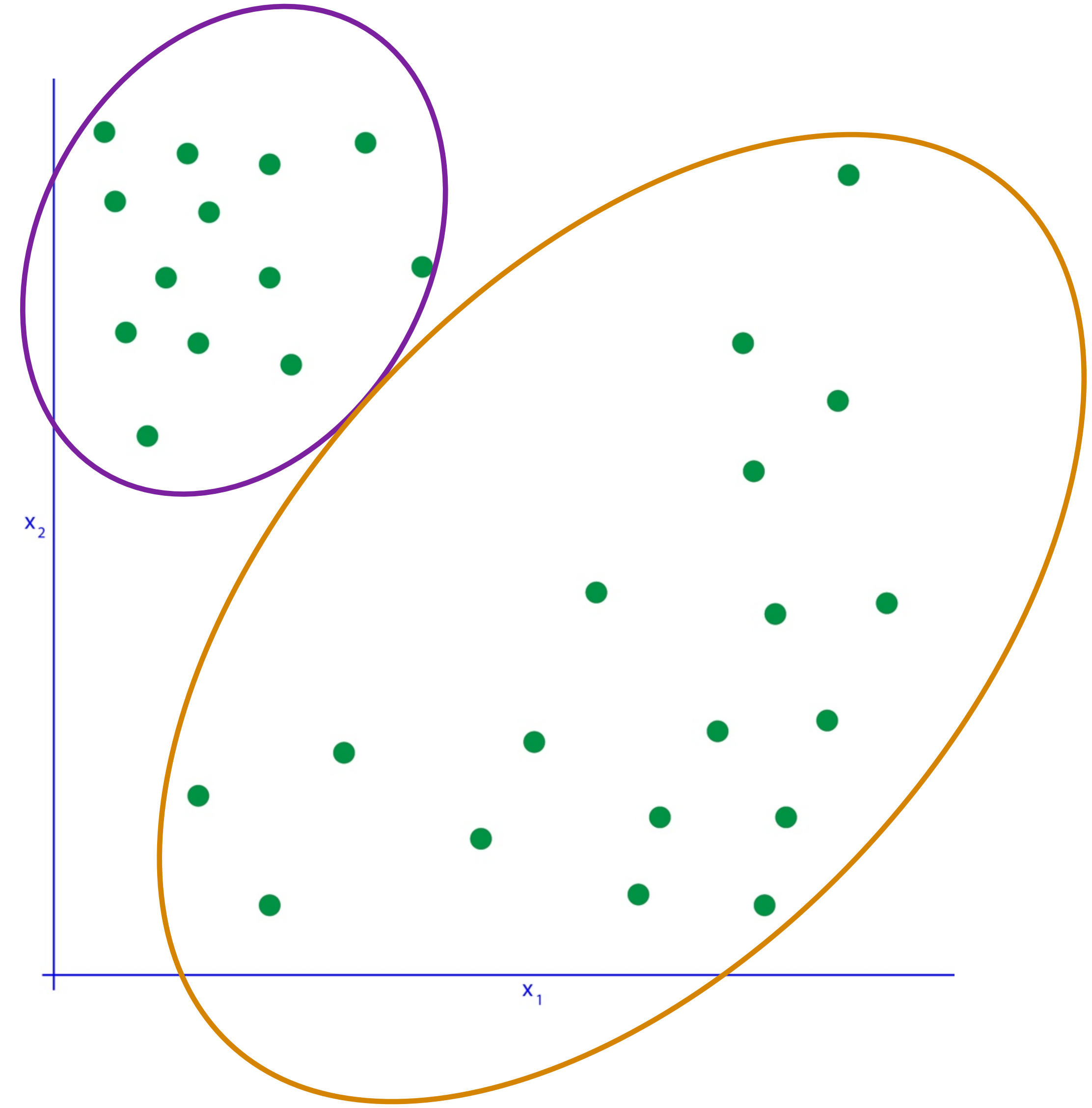
many others, such as semi-supervised learning, reinforcement learning, active learning, deep learning, etc..

# Machine Learning

Main types of learning:




SUPERVISED LEARNING




UNSUPERVISED LEARNING

many others, such as semi-supervised learning, reinforcement learning, active learning, deep learning, etc..

# Machine Learning

Supervised Learning

# Machine Learning

Training Data

# Machine Learning

Training Data

$$\mathbf{X} = \{\{\mathbf{x}_1, \mathbf{t}_1\}, \{\mathbf{x}_2, \mathbf{t}_2\}, \{\mathbf{x}_3, \mathbf{t}_3\}, \ldots, \{\mathbf{x}_M, \mathbf{t}_M\}\}^T$$

# Machine Learning

Training Data

$$\mathbf{X} = \{\{\mathbf{x}_1, \mathbf{t}_1\}, \{\mathbf{x}_2, \mathbf{t}_2\}, \{\mathbf{x}_3, \mathbf{t}_3\}, \ldots, \{\mathbf{x}_M, \mathbf{t}_M\}\}^T$$

$$\mathbf{x} = \{x_1, x_2, x_3, \ldots, x_N\}$$

Input Vector

# Machine Learning

Training Data

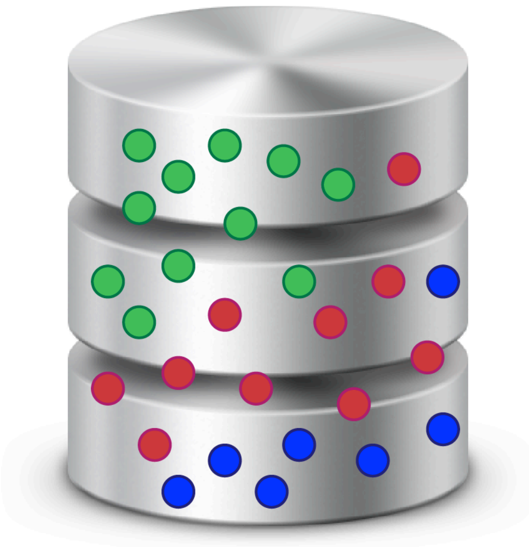
$$\mathbf{X} = \{\{\mathbf{x}_1, \mathbf{t}_1\}, \{\mathbf{x}_2, \mathbf{t}_2\}, \{\mathbf{x}_3, \mathbf{t}_3\}, \ldots, \{\mathbf{x}_M, \mathbf{t}_M\}\}^T$$

$$\mathbf{x} = \{x_1, x_2, x_3, \ldots, x_N\} \qquad \mathbf{t} = \{k\}$$

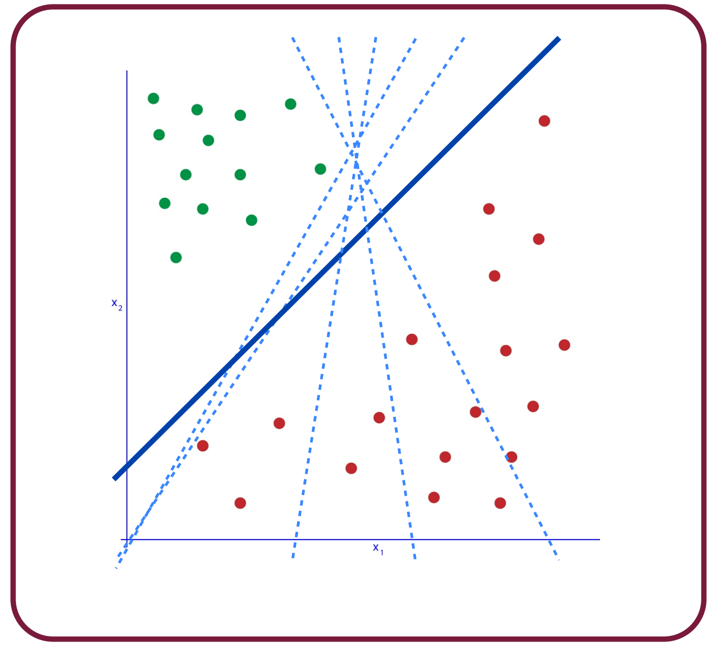
Input Vector

Target Vector

# Machine Learning



Training Data



Learning Algorithm

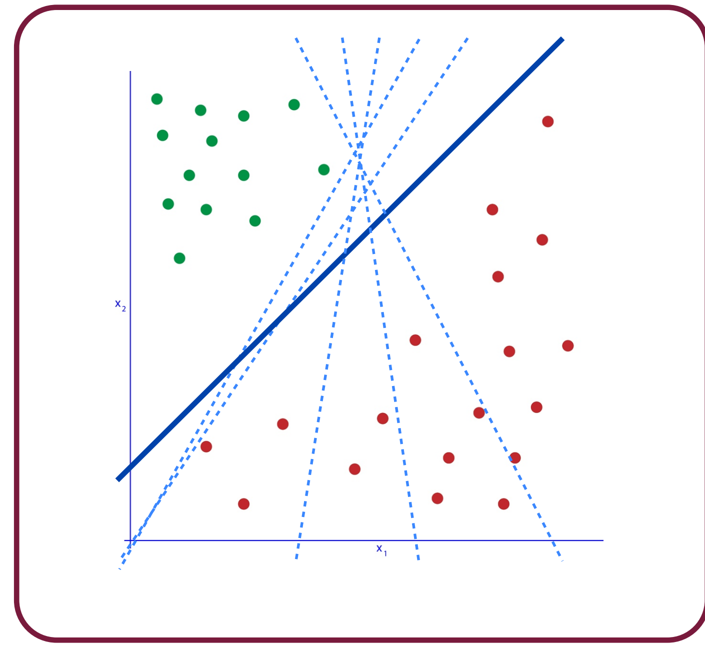
$$\mathbf{X} = \{\{\mathbf{x}_1, \mathbf{t}_1\}, \{\mathbf{x}_2, \mathbf{t}_2\}, \{\mathbf{x}_3, \mathbf{t}_3\}, \ldots, \{\mathbf{x}_M, \mathbf{t}_M\}\}^T$$

$$\mathbf{x} = \{x_1, x_2, x_3, \ldots, x_N\} \qquad \mathbf{t} = \{k\}$$

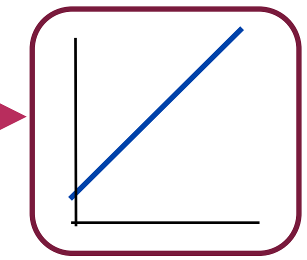
Input Vector

Target Vector

# Machine Learning



Model



Training Data

Learning Algorithm

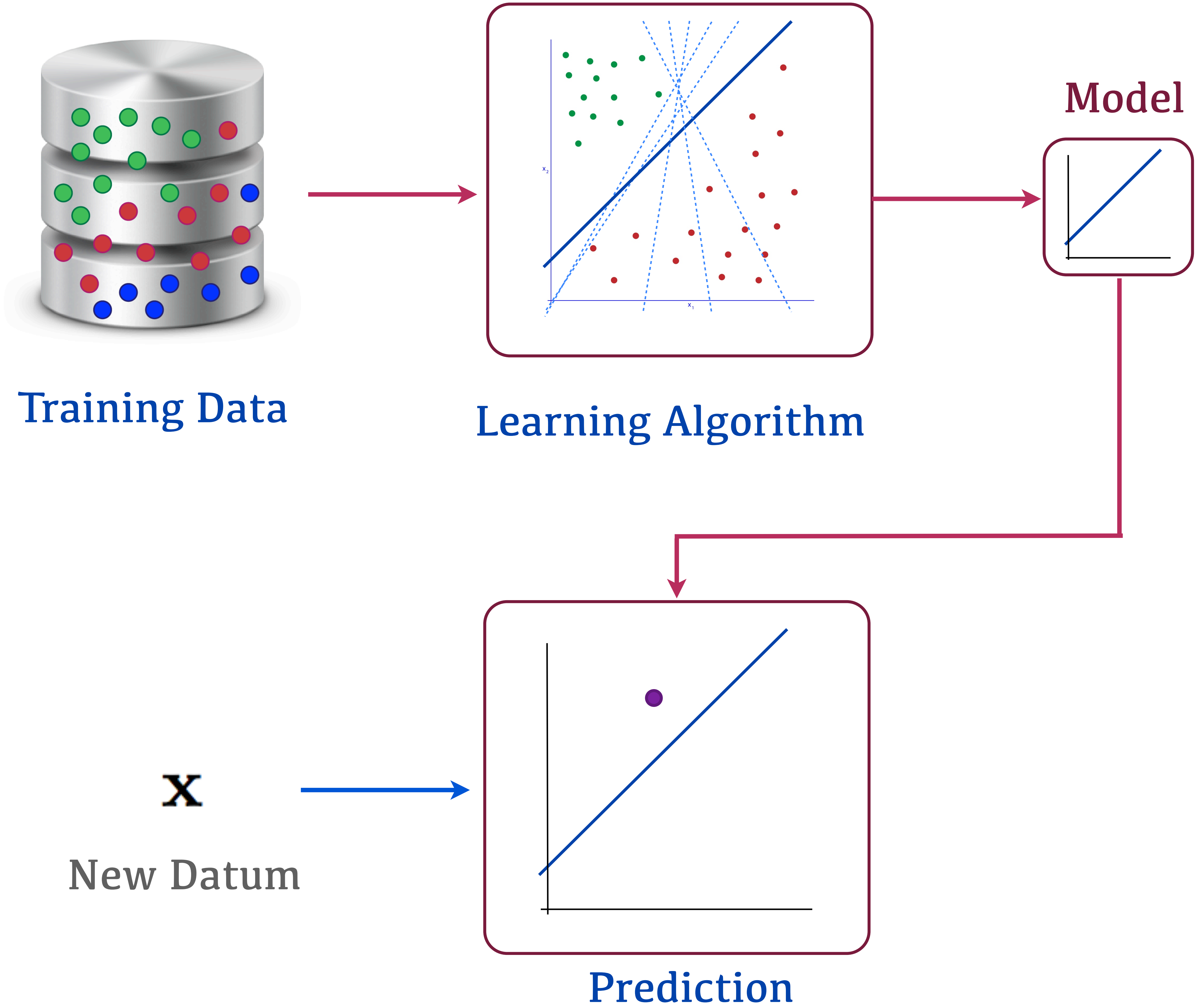
$$\mathbf{X} = \{\{\mathbf{x}_1, \mathbf{t}_1\}, \{\mathbf{x}_2, \mathbf{t}_2\}, \{\mathbf{x}_3, \mathbf{t}_3\}, \ldots, \{\mathbf{x}_M, \mathbf{t}_M\}\}^T$$

$$\mathbf{x} = \{x_1, x_2, x_3, \ldots, x_N\} \qquad \mathbf{t} = \{k\}$$

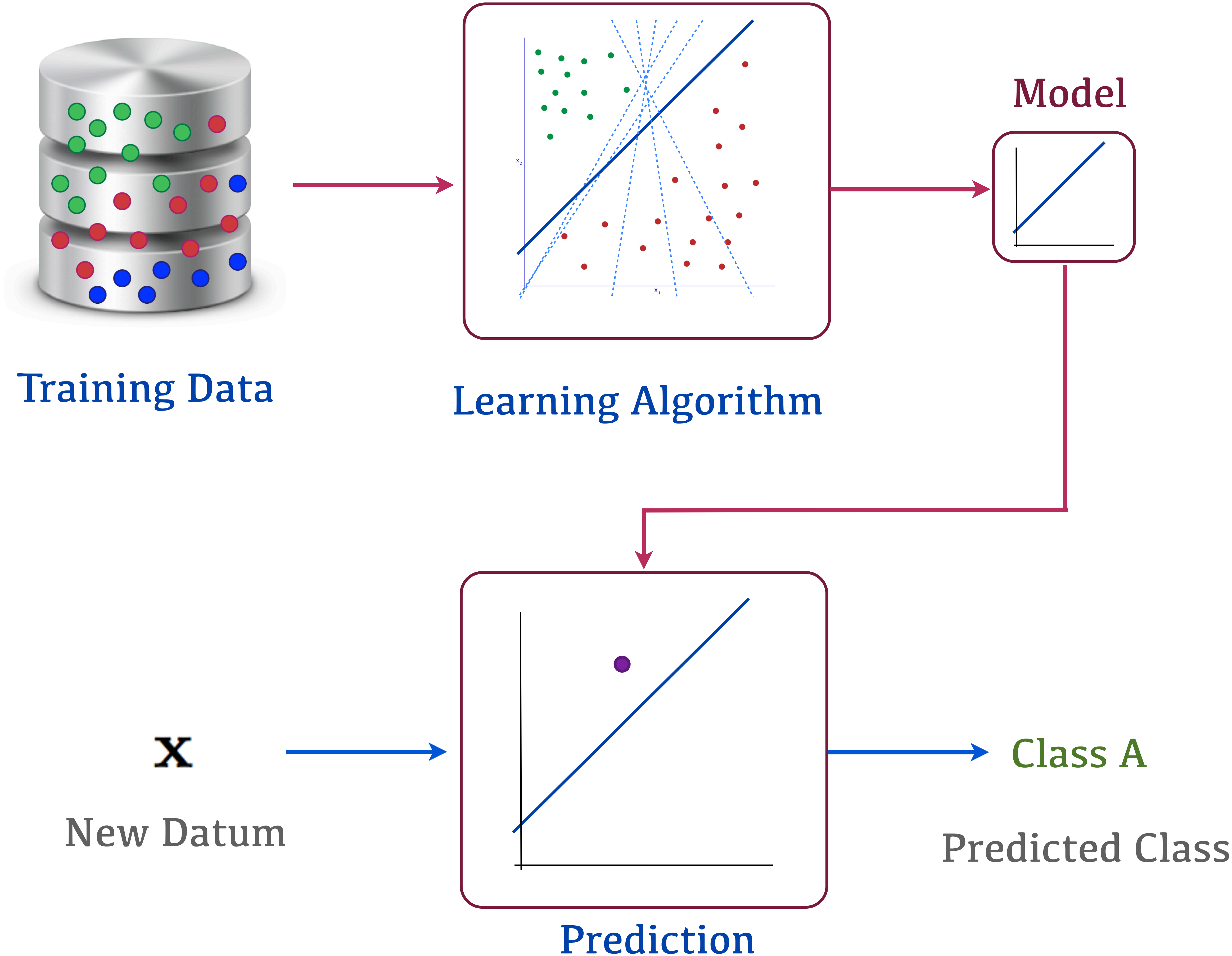
Input Vector

Target Vector

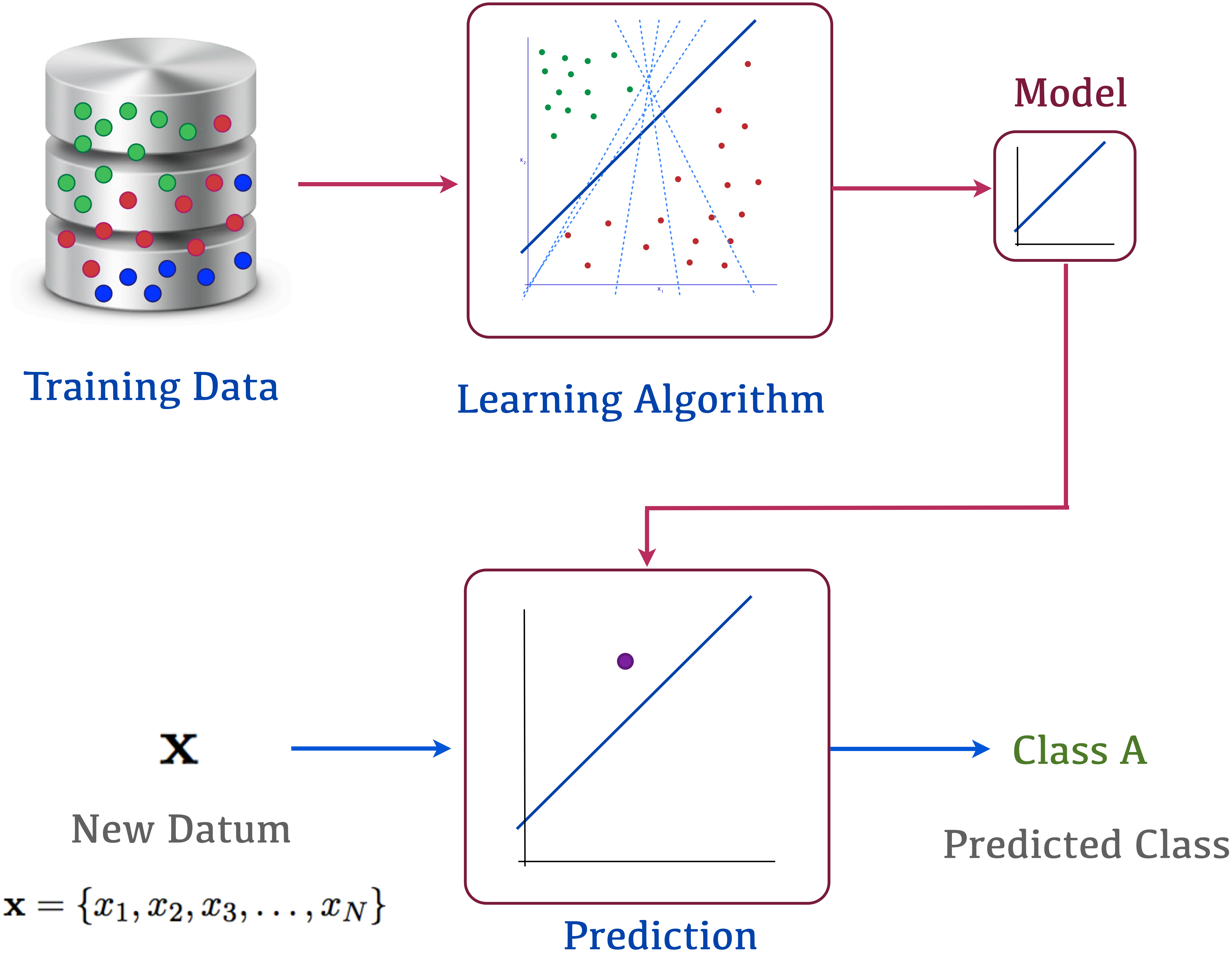
# Machine Learning



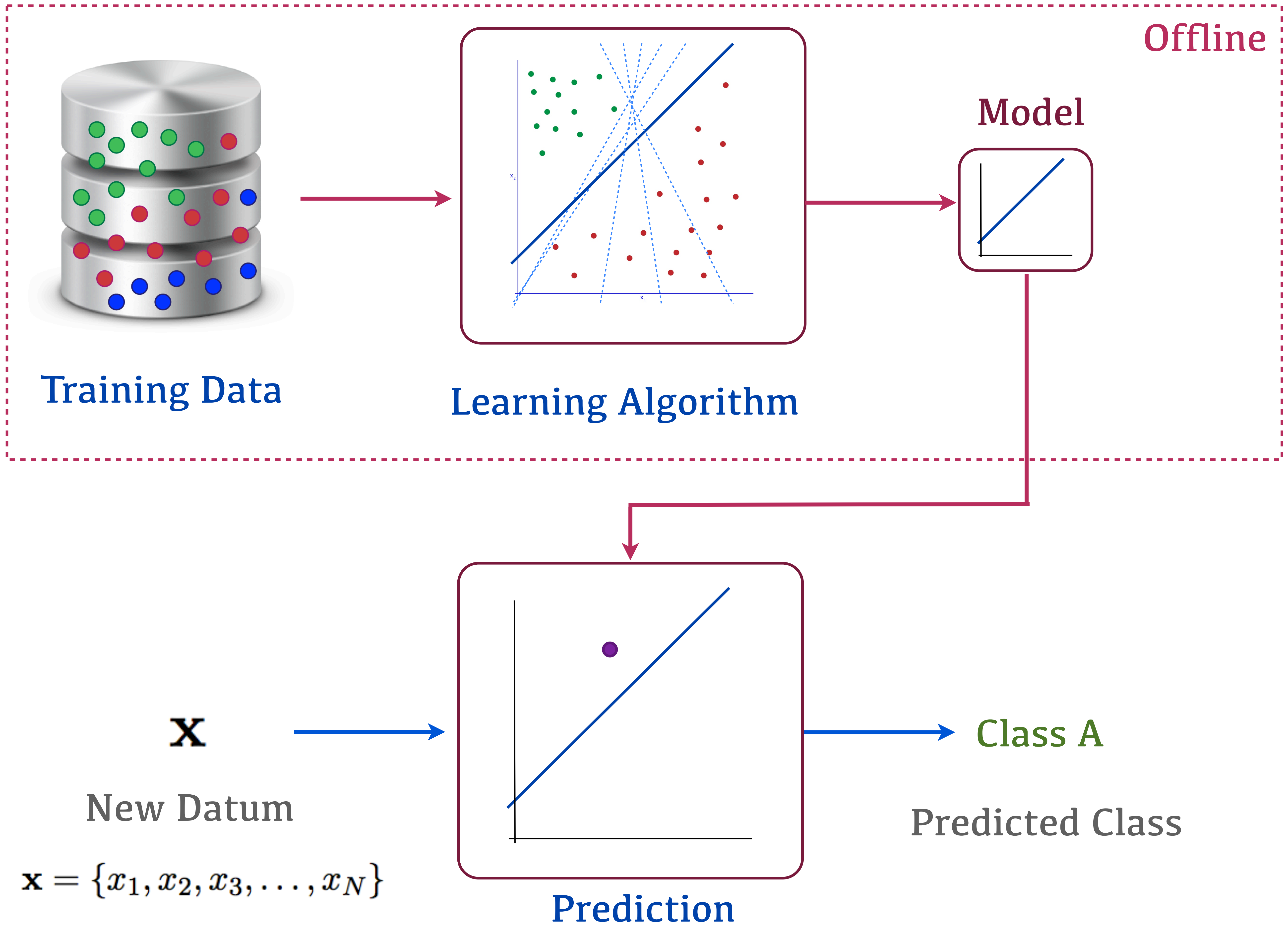
# Machine Learning



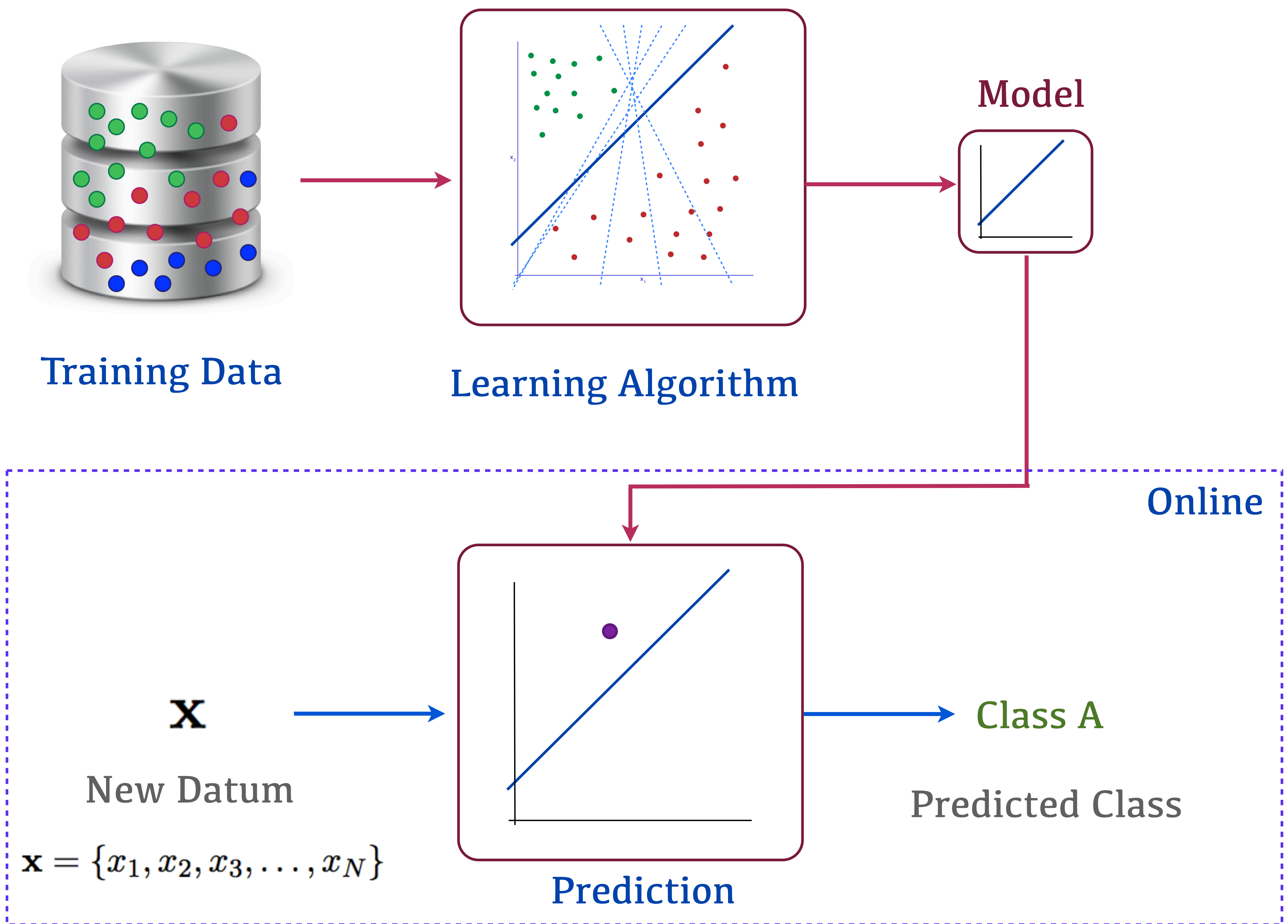
# Machine Learning



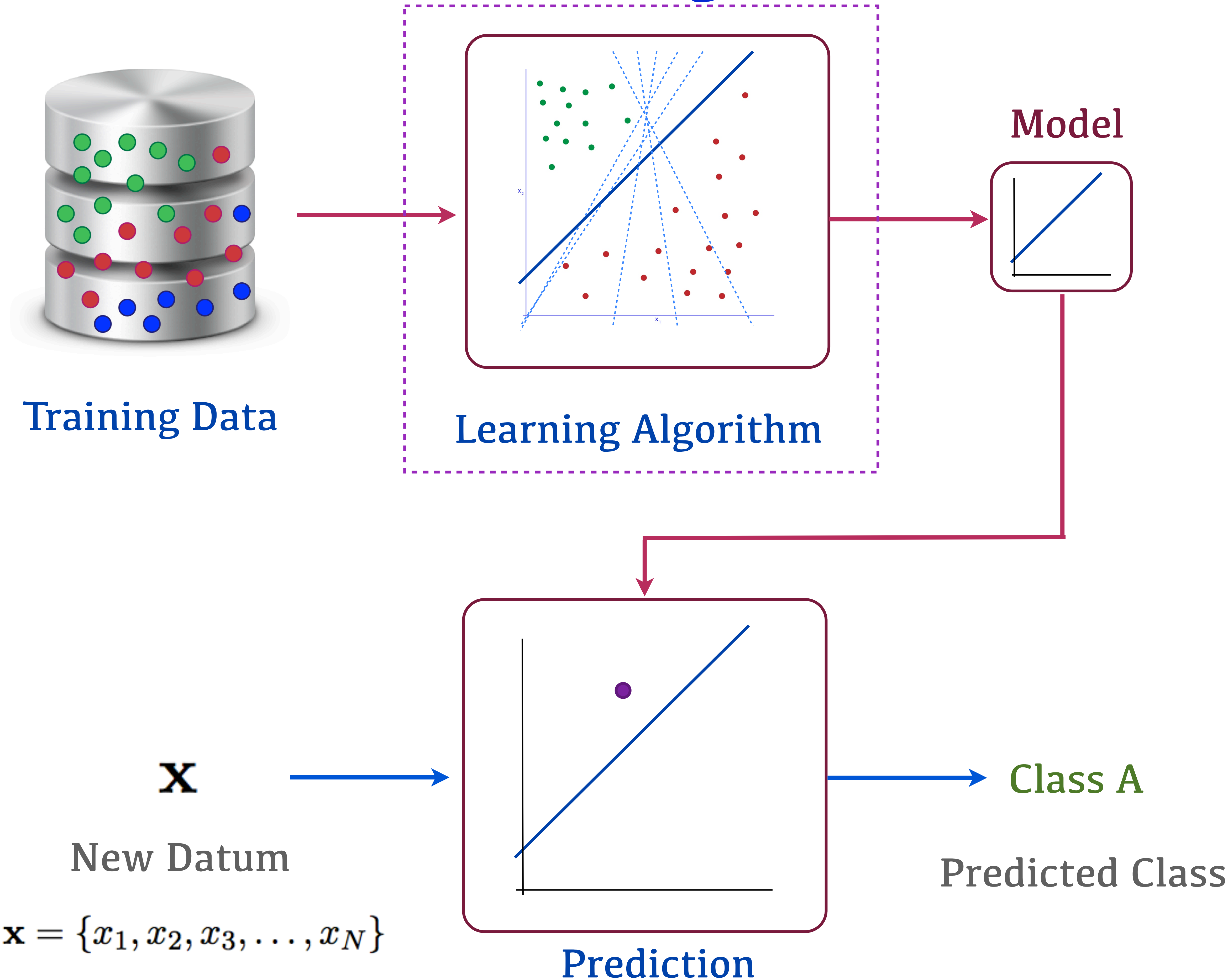
# Machine Learning



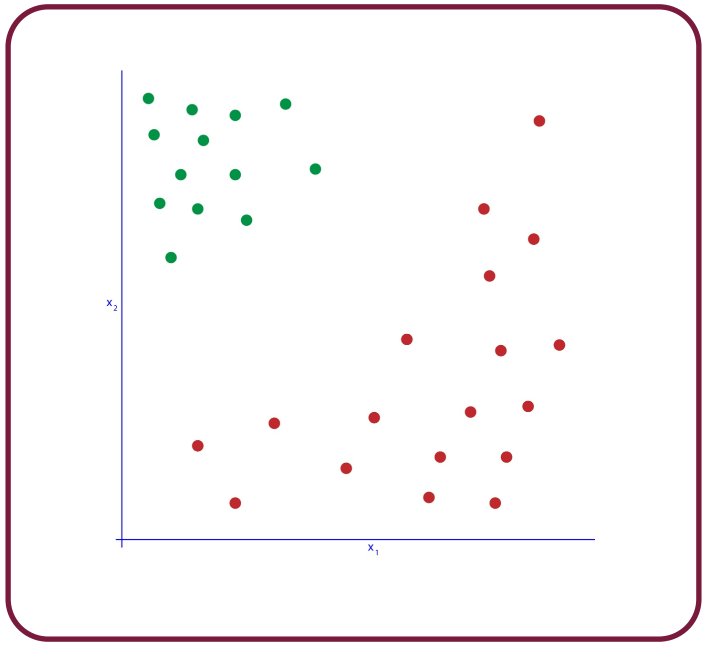
# Machine Learning



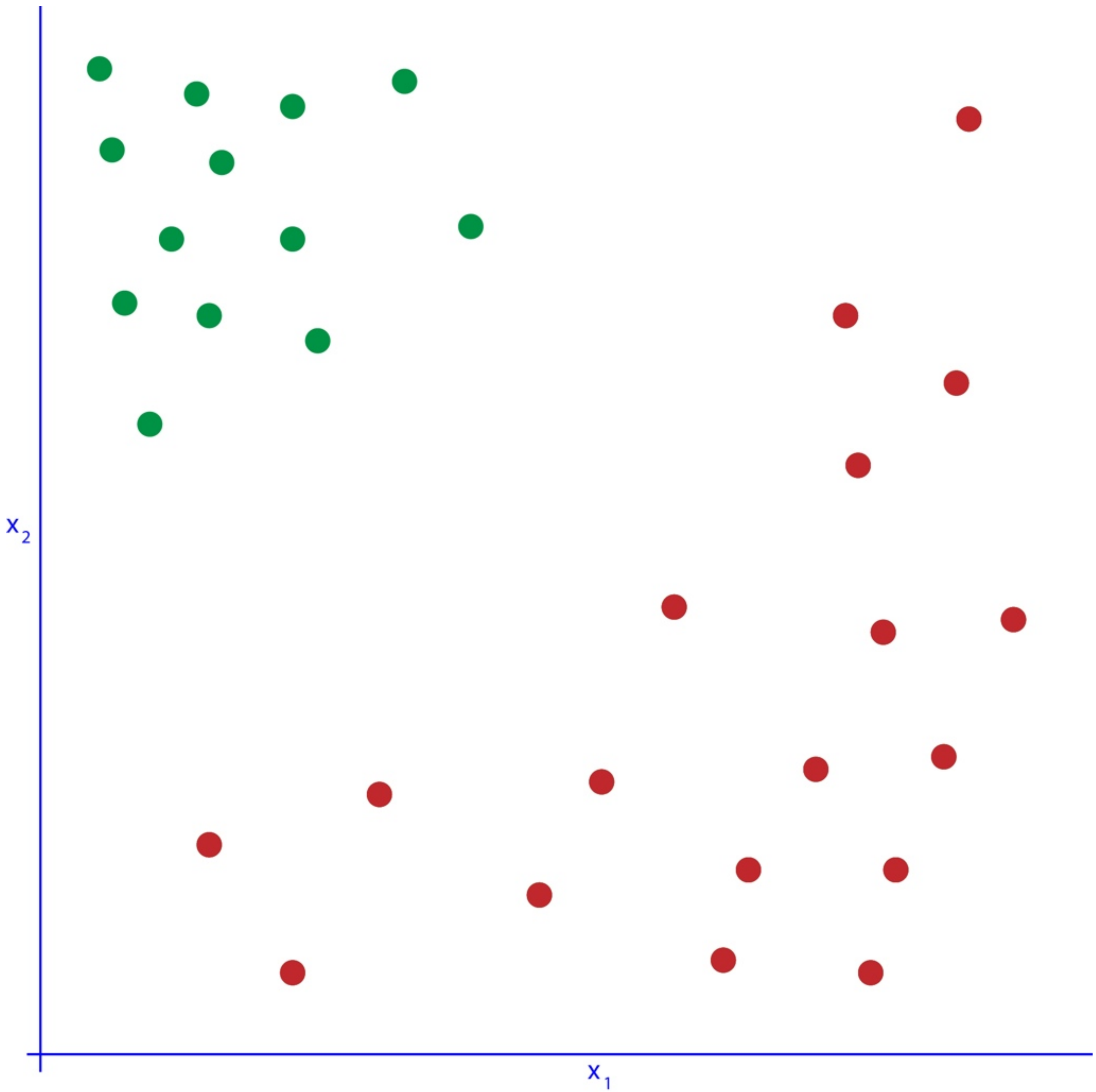
# The Learning Process



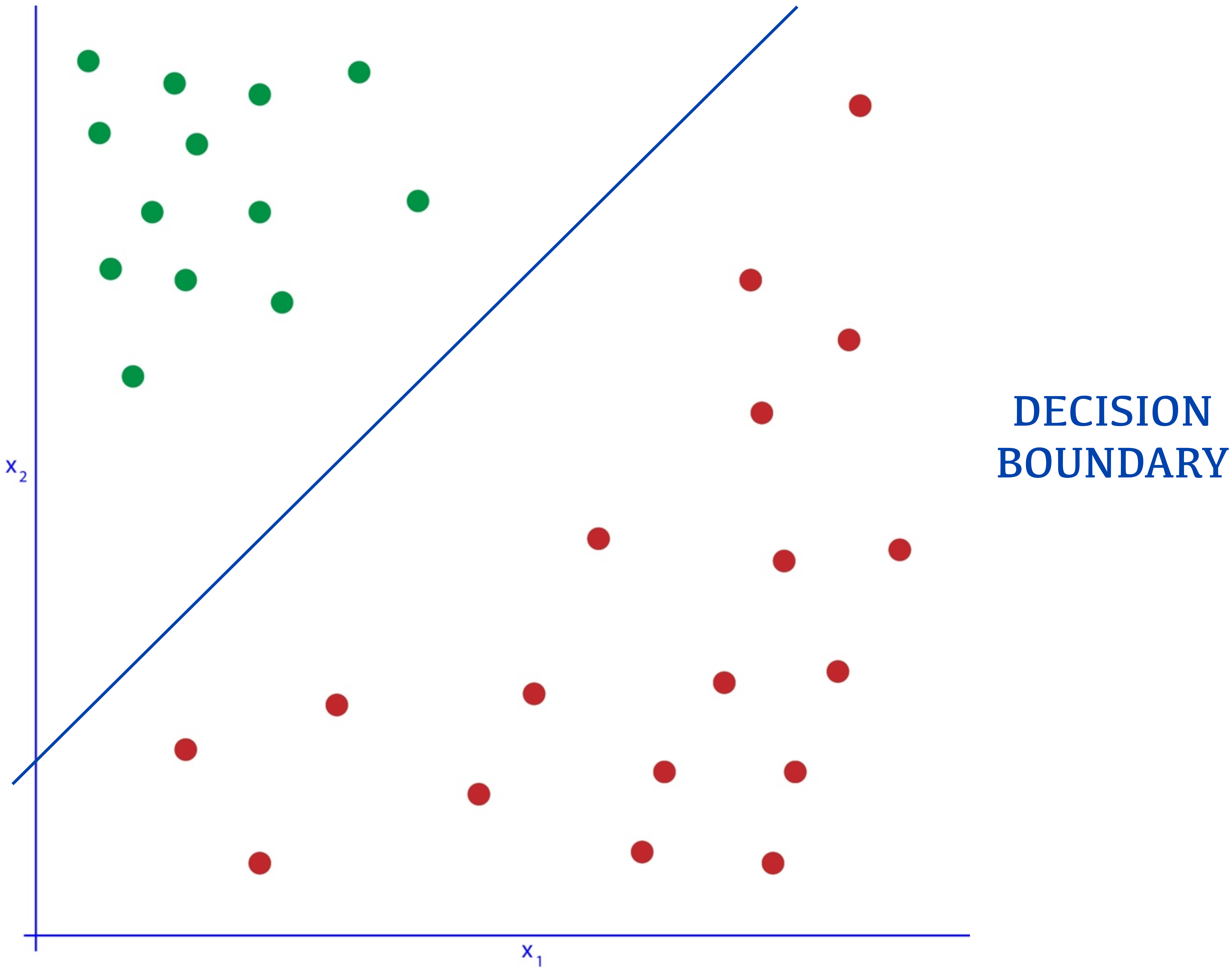
# The Learning Process



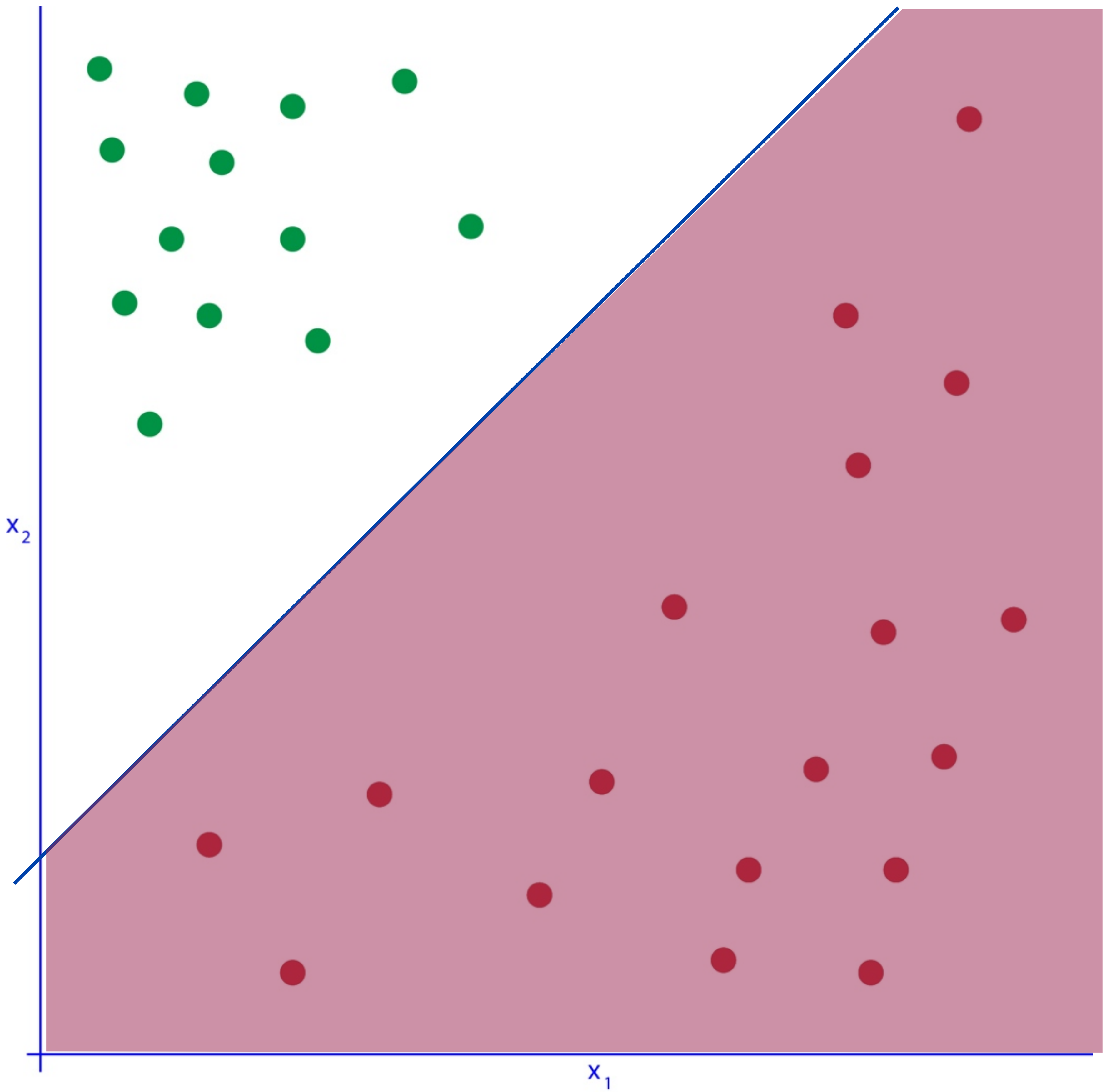
# The Learning Process



# The Learning Process

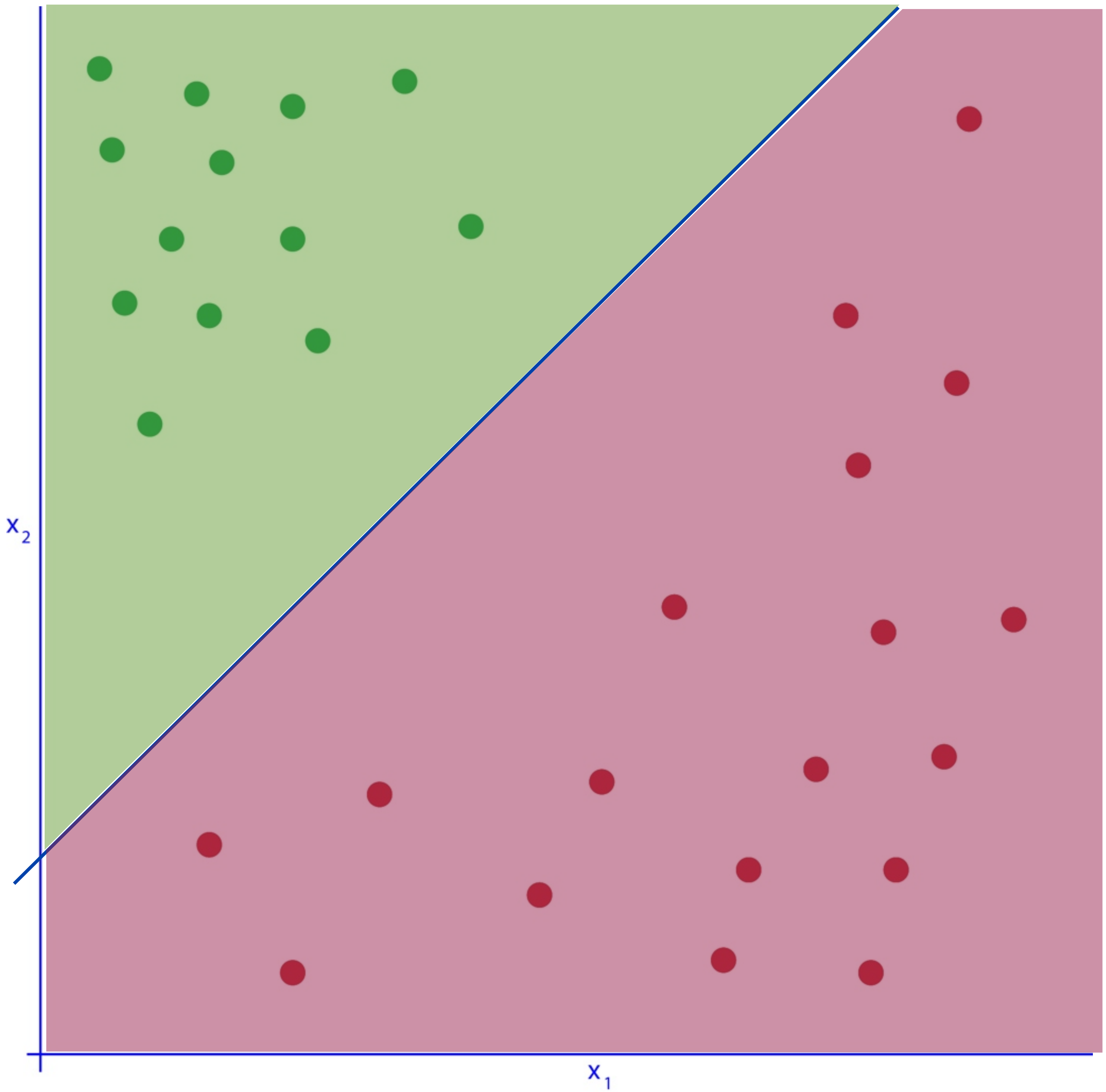


# The Learning Process



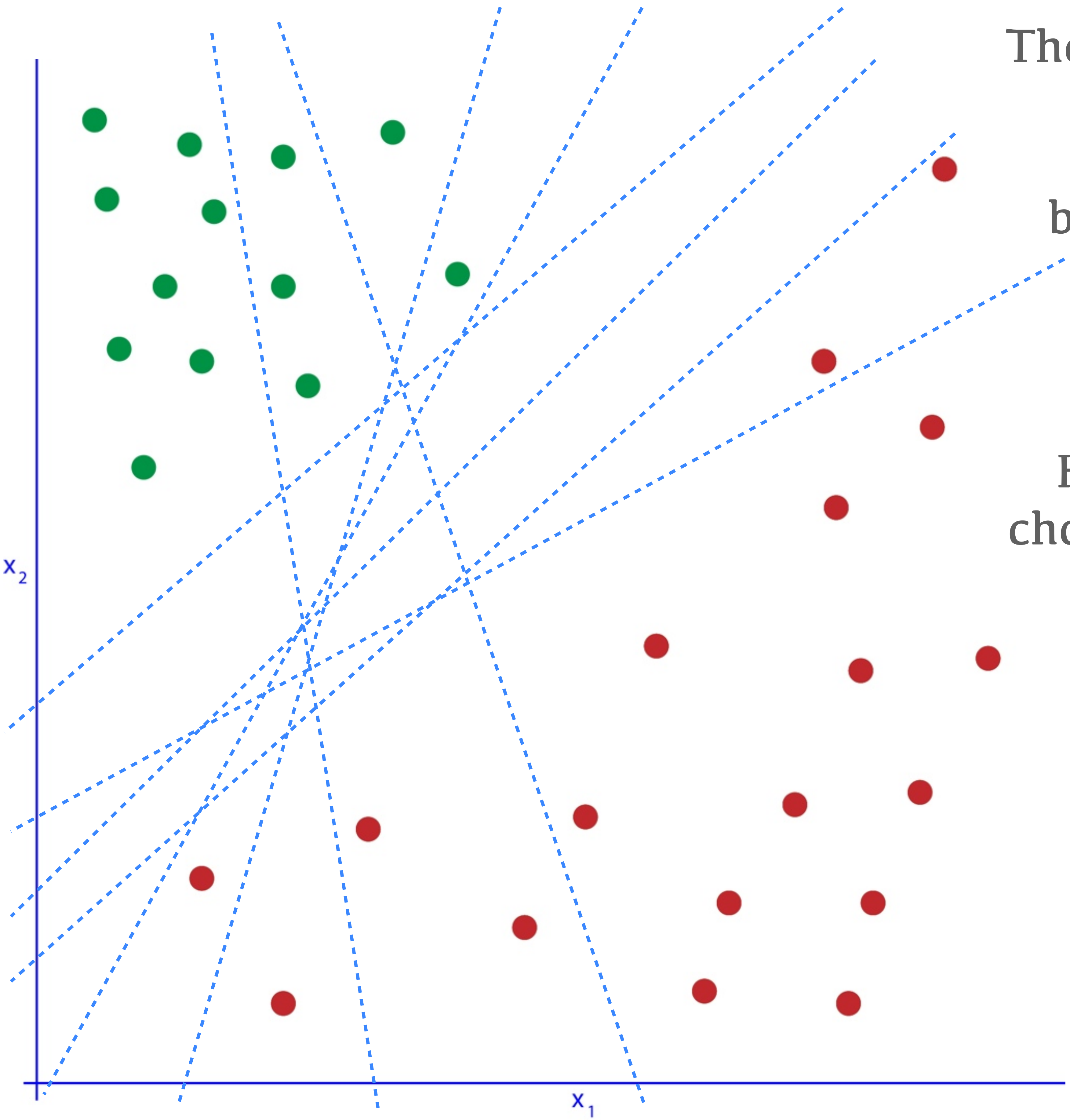

DECISION BOUNDARY

# The Learning Process




DECISION BOUNDARY

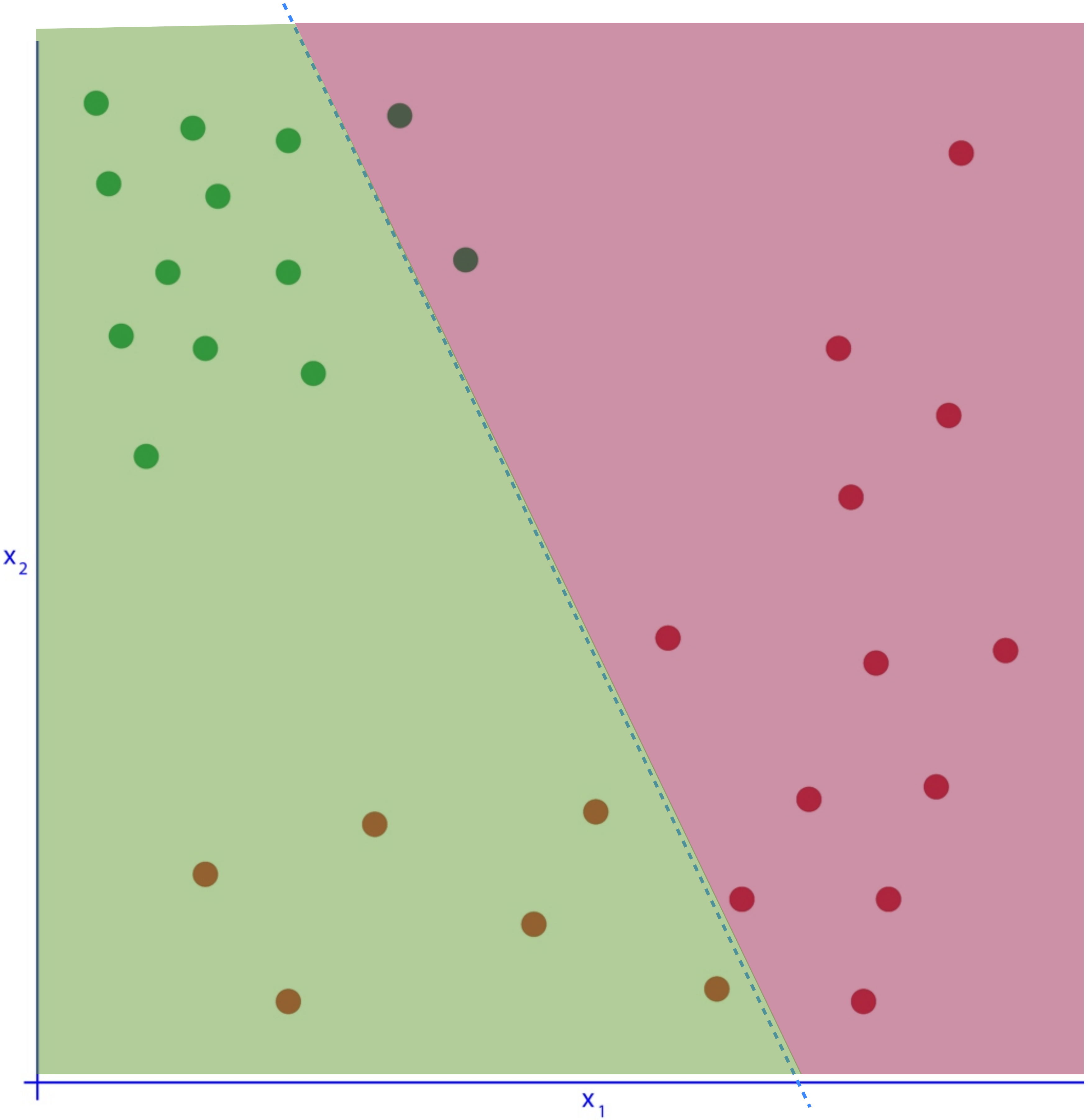
# The Learning Process




There are many possible decision boundaries!

How do we choose the best one?

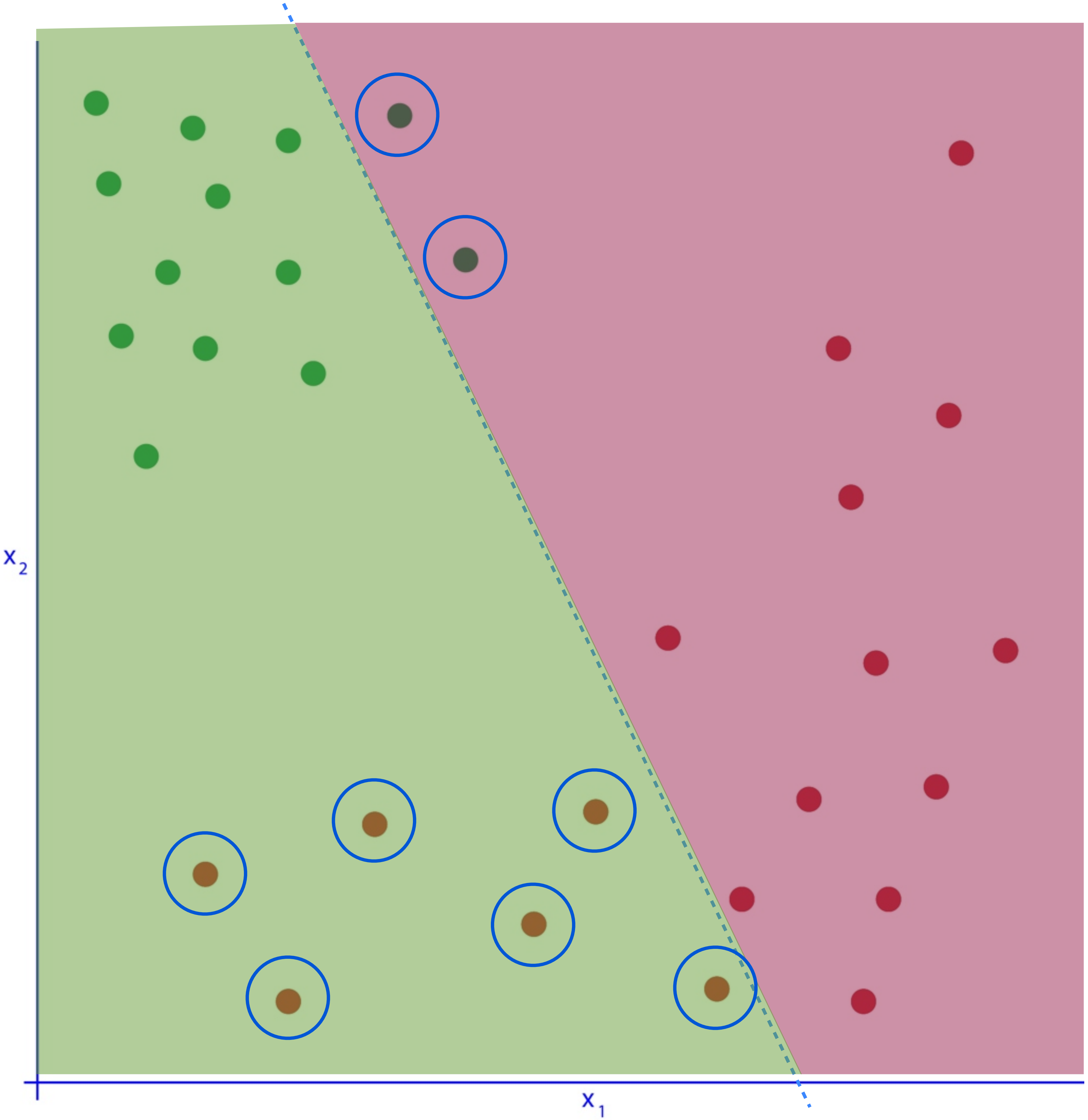
# The Learning Process




Minimize some error:

$$\frac{\text{Num Correctly Classified Examples}}{\text{Num Examples}}$$

# The Learning Process



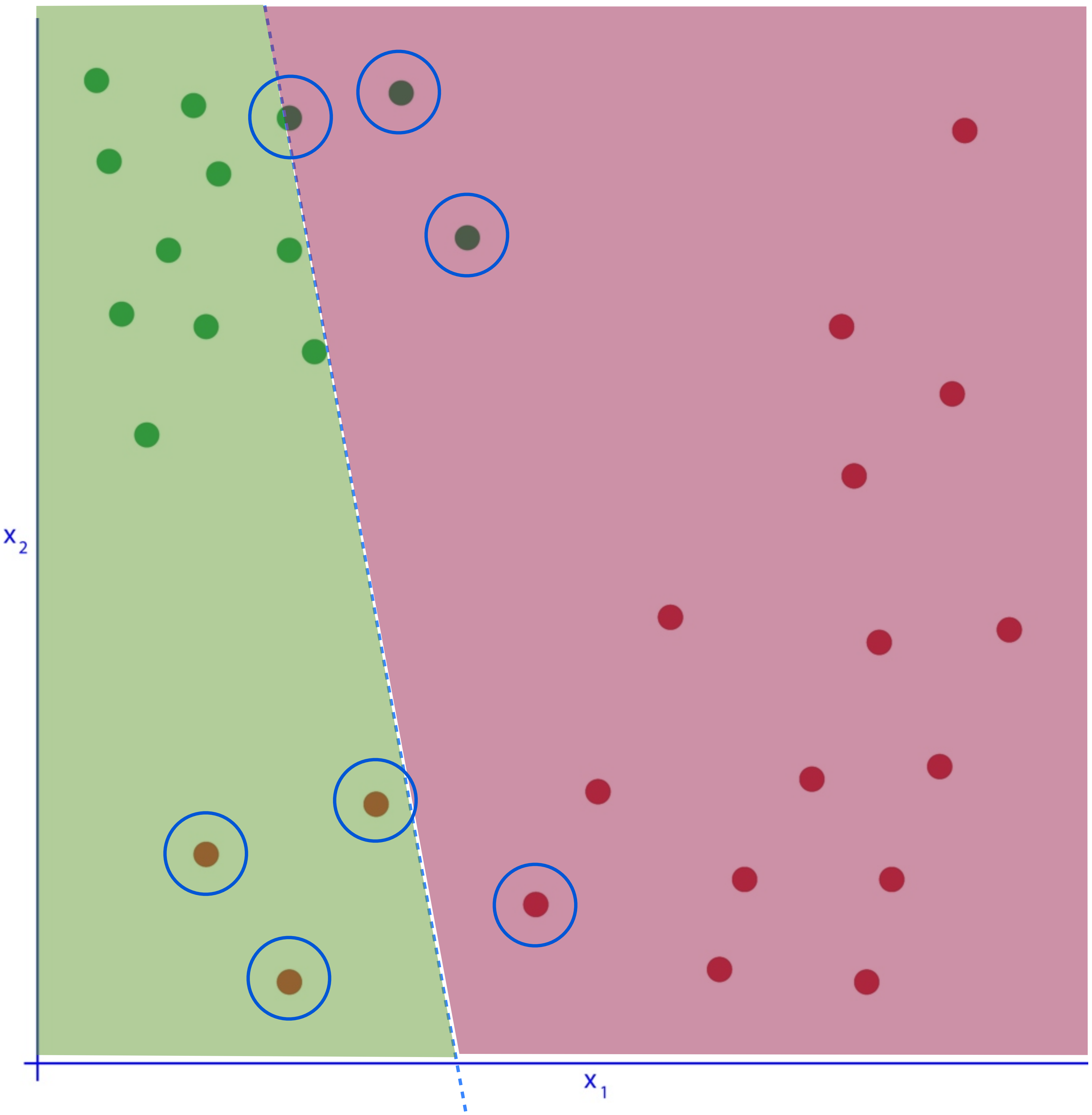

Minimize some error:

$$\frac{\text{Num Correctly Classified Examples}}{\text{Num Examples}}$$

$$\frac{22}{32}$$

Error = 0.31

# The Learning Process



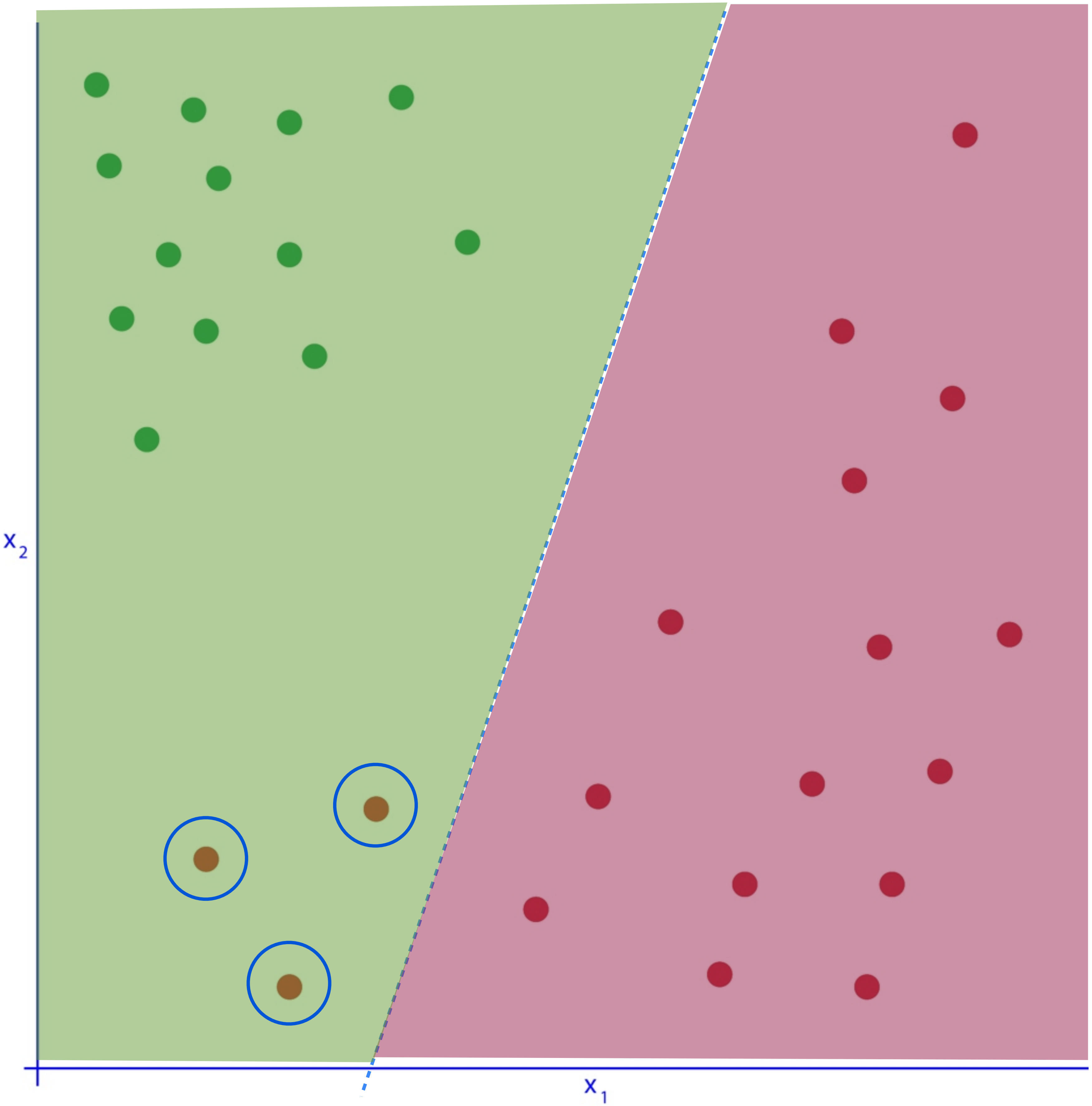

Minimize some error:

$$\frac{\text{Num Correctly Classified Examples}}{\text{Num Examples}}$$

$$\frac{25}{32}$$

Error = 0.22

# The Learning Process



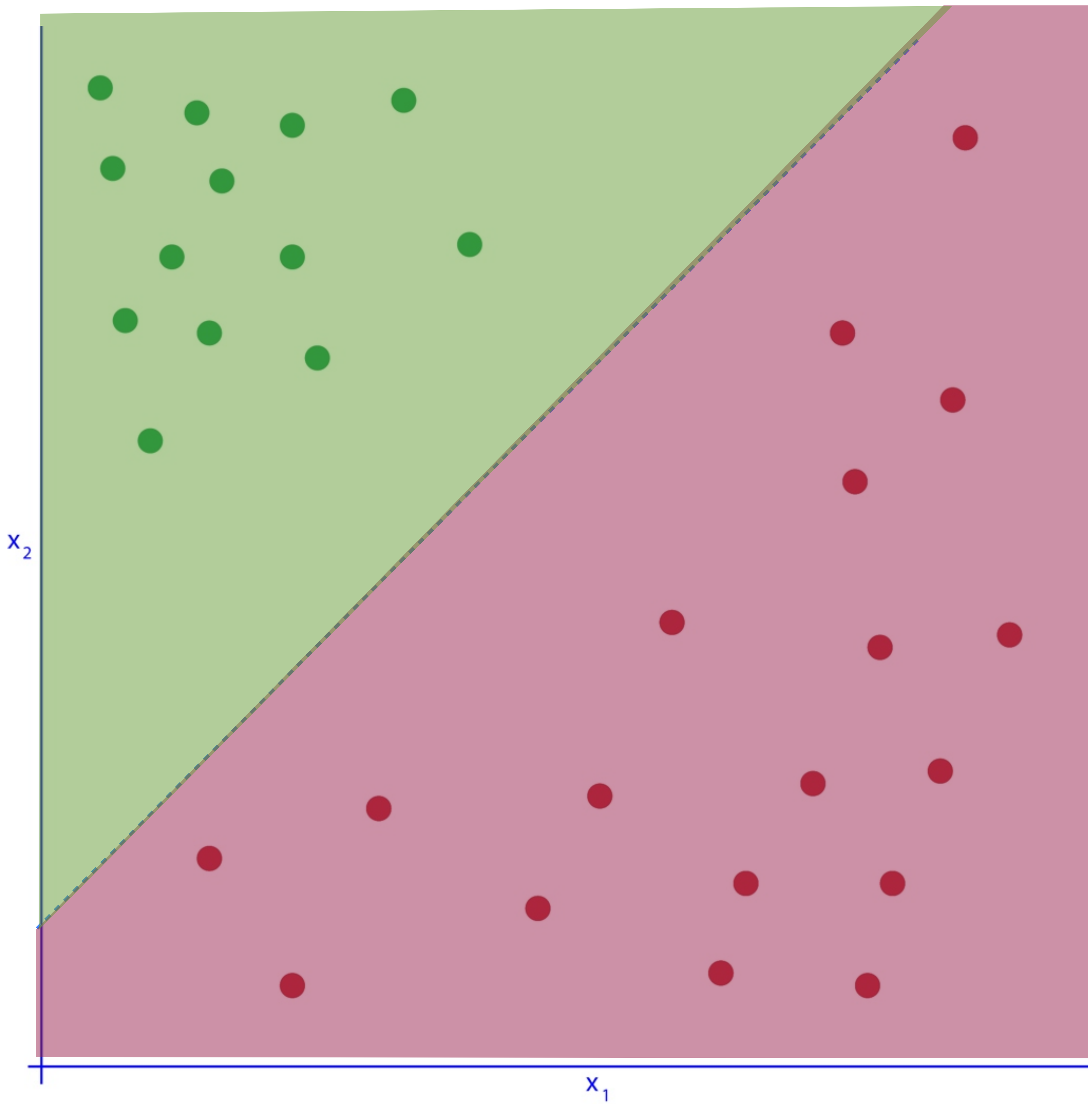

Minimize some error:

$$\frac{\text{Num Correctly Classified Examples}}{\text{Num Examples}}$$

$$\frac{28}{32}$$

Error = 0.12

# The Learning Process



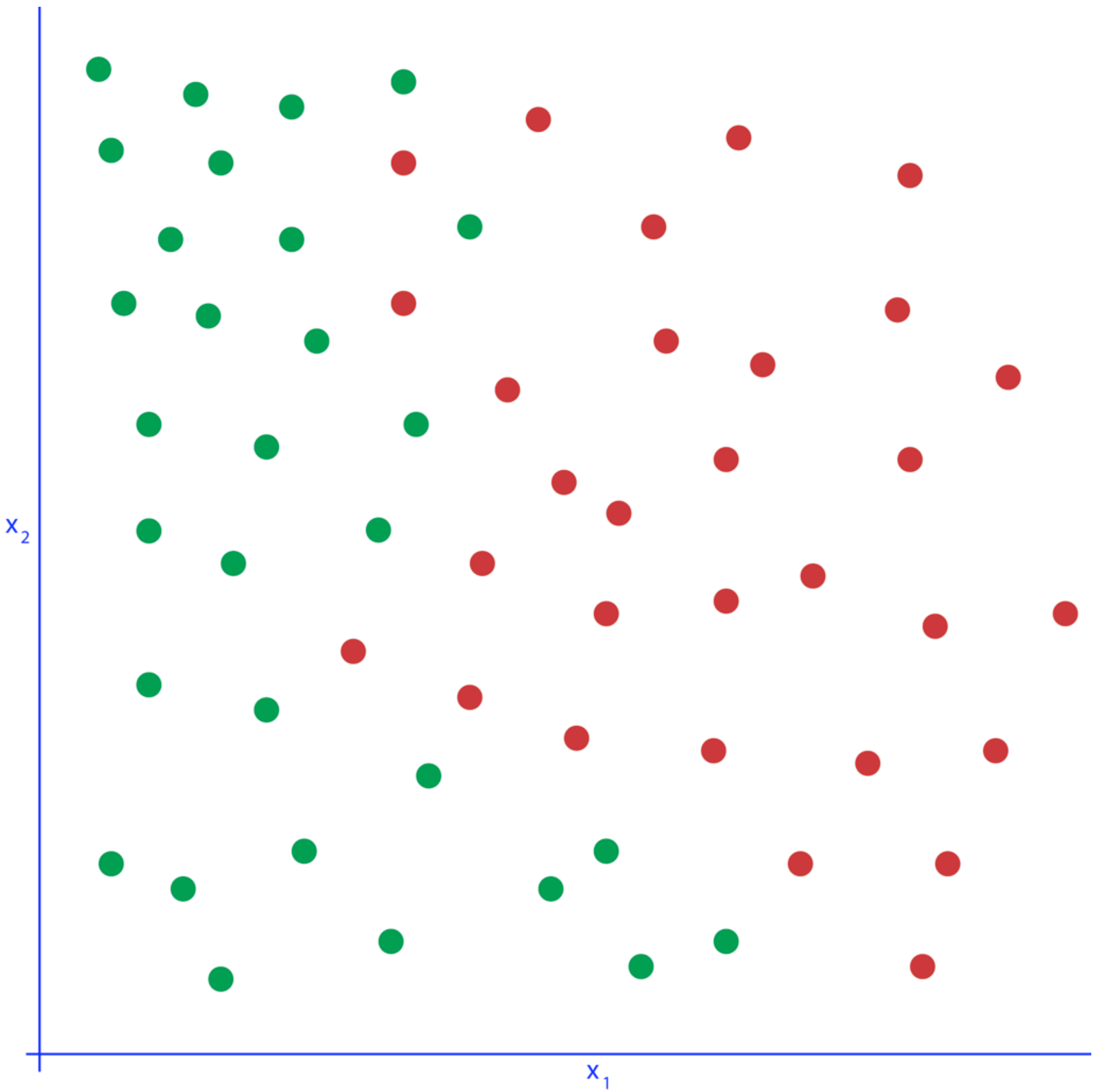

Stop when this error is small

$$\frac{32}{32}$$

Error = 0

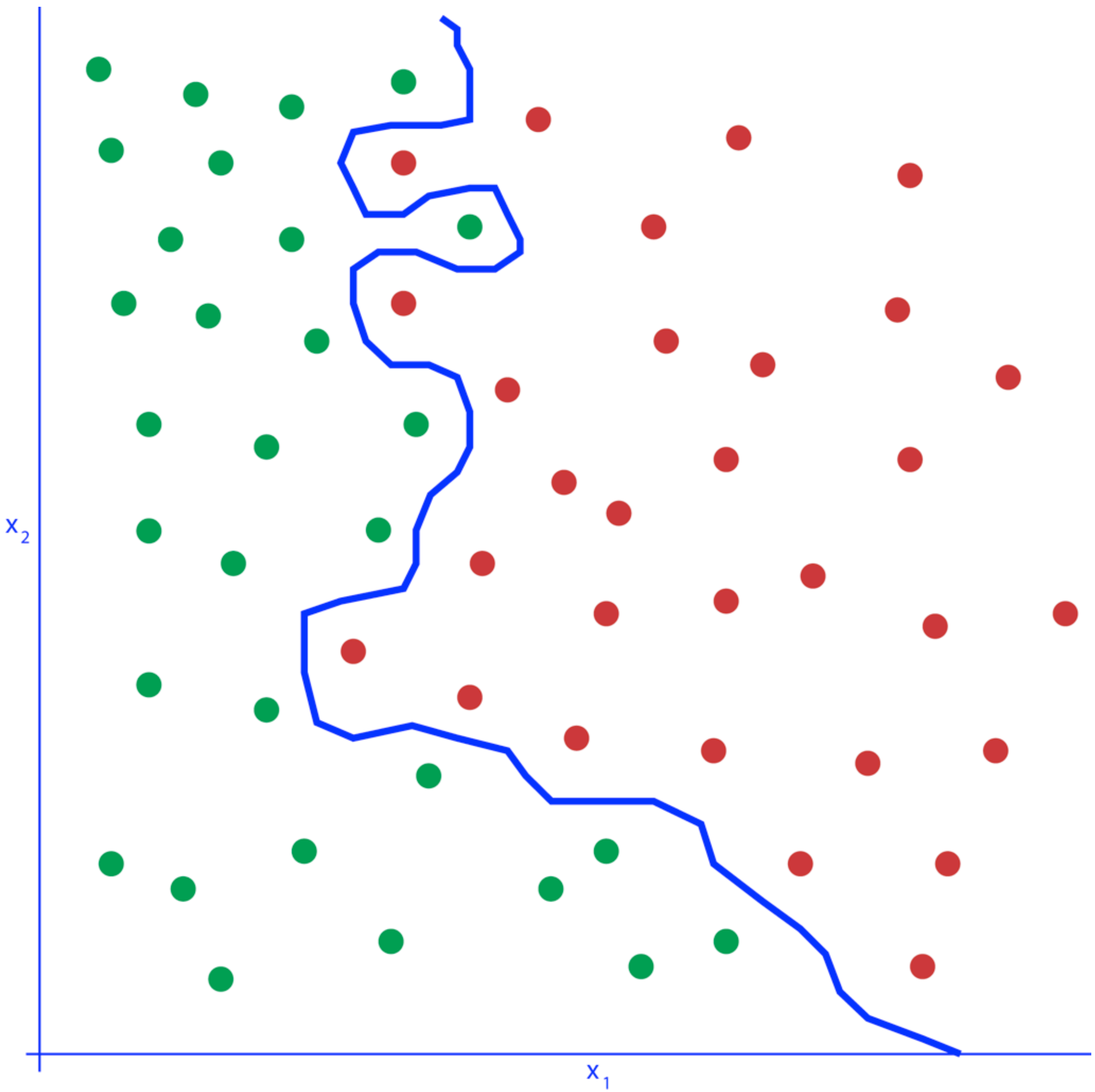
# The Learning Process

Need to be careful that we don't overtrain the model…



# The Learning Process

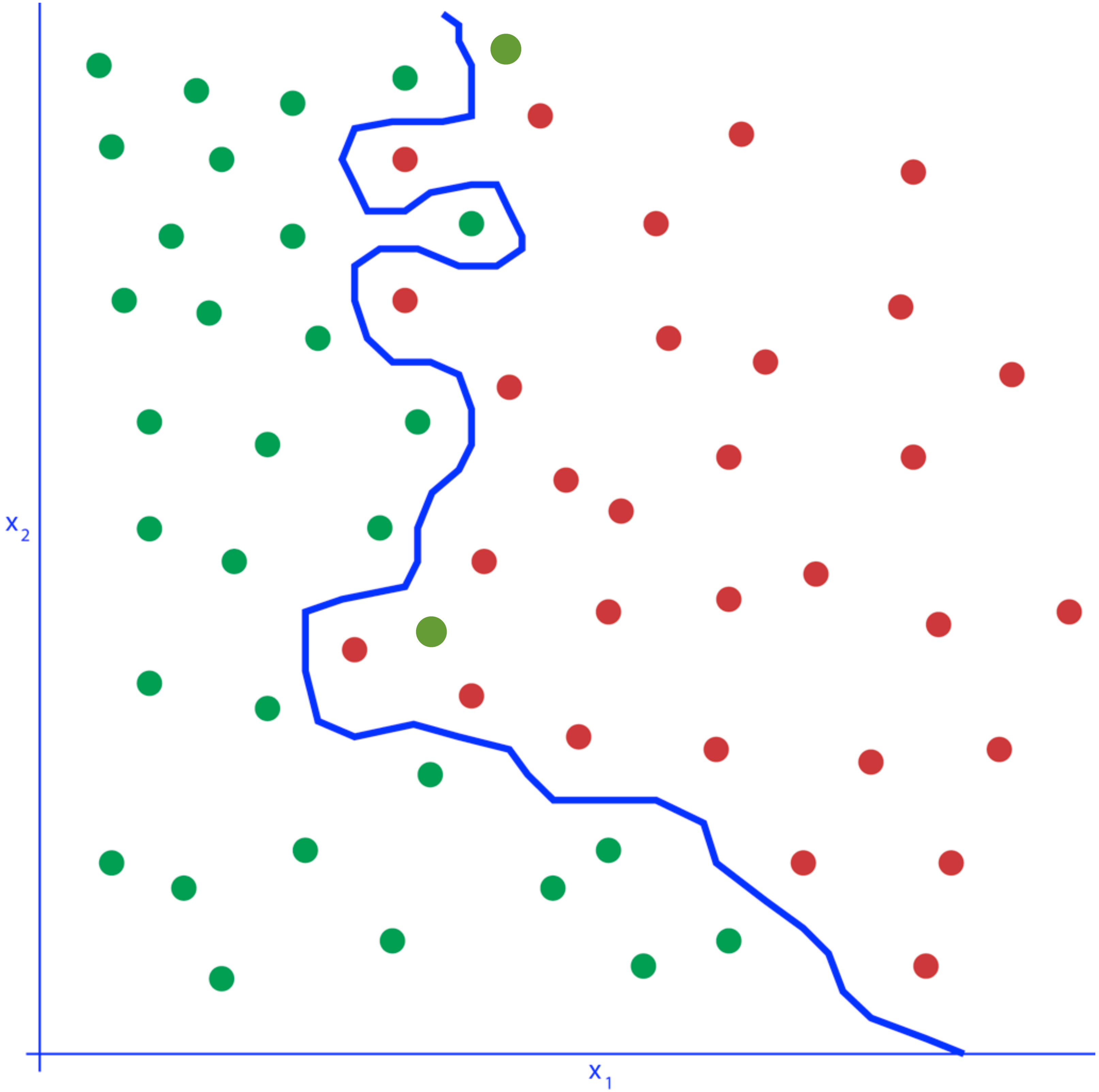
Need to be careful that we don't overtrain the model…




Complex decision boundary gets a perfect result on the training data

# The Learning Process

Need to be careful that we don't overtrain the model…

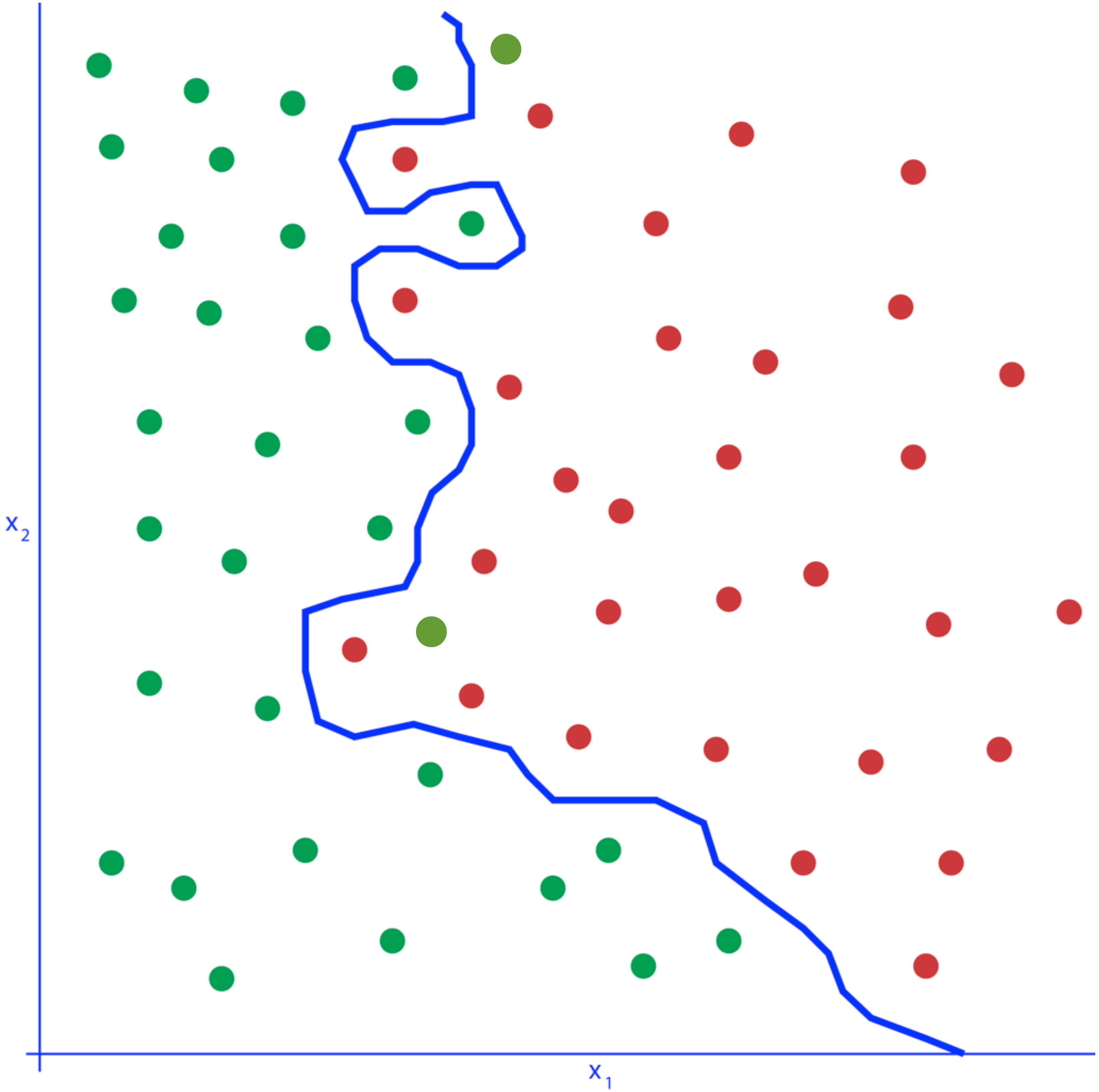



Complex decision boundary gets a perfect result on the training data

But it might fail terribly with new data

# The Learning Process

Need to be careful that we don't overtrain the model…



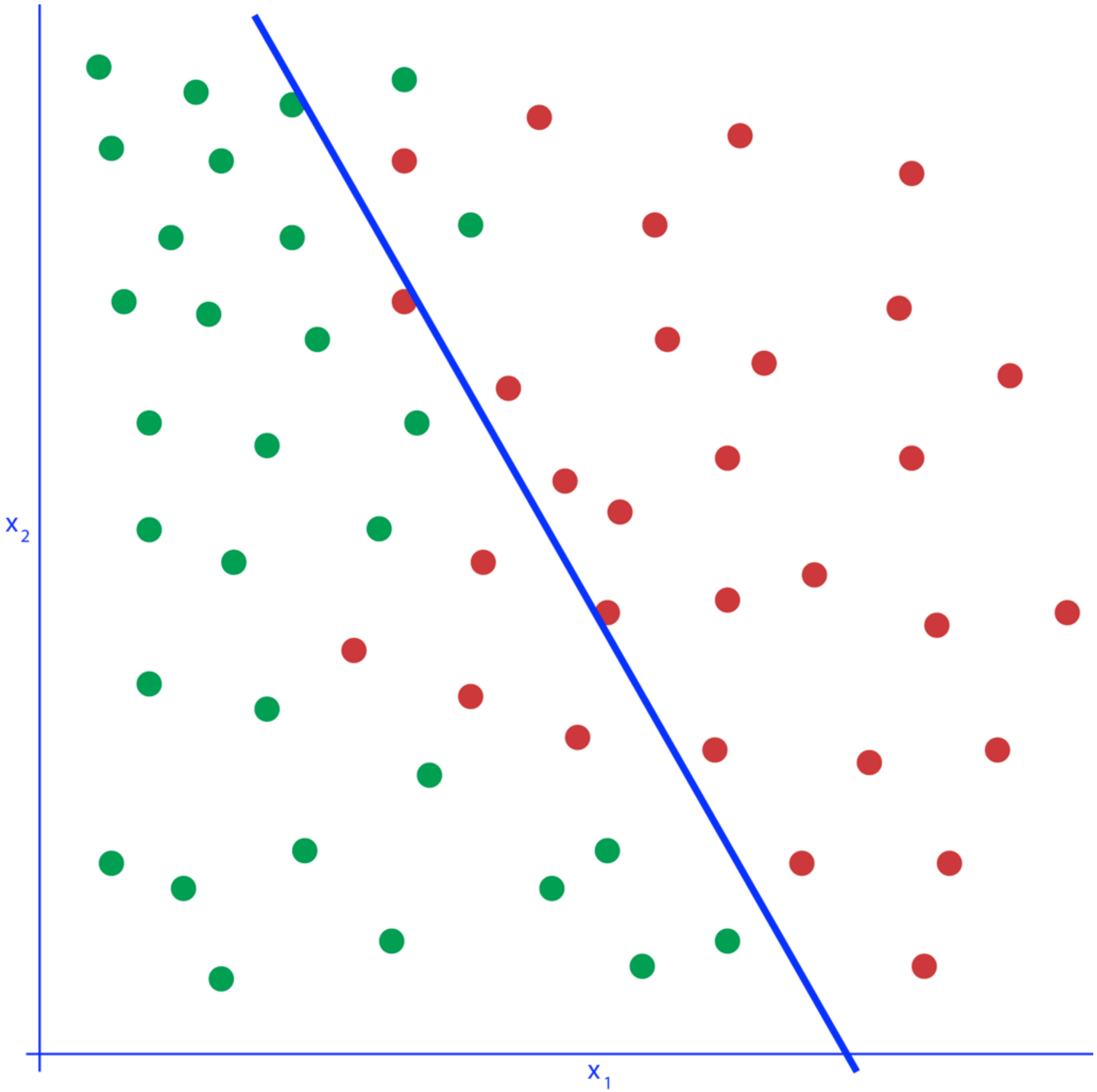

Complex decision boundary gets a perfect result on the training data

But it might fail terribly with new data

This is know as **OVERFITTING**

# The Learning Process

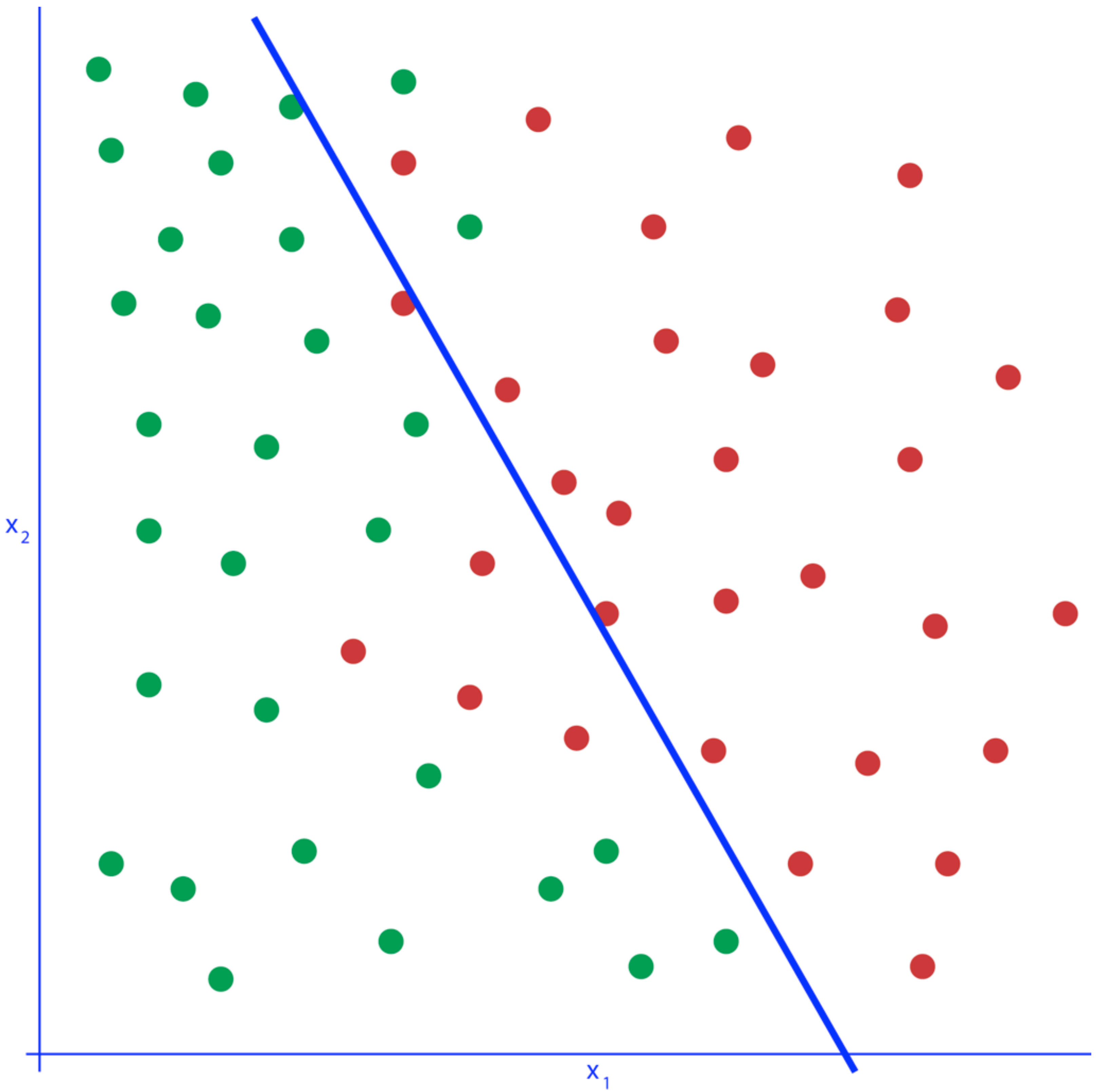
Need to be careful that we don't overtrain the model…




A very simple decision boundary might not work either

# The Learning Process

Need to be careful that we don't overtrain the model…

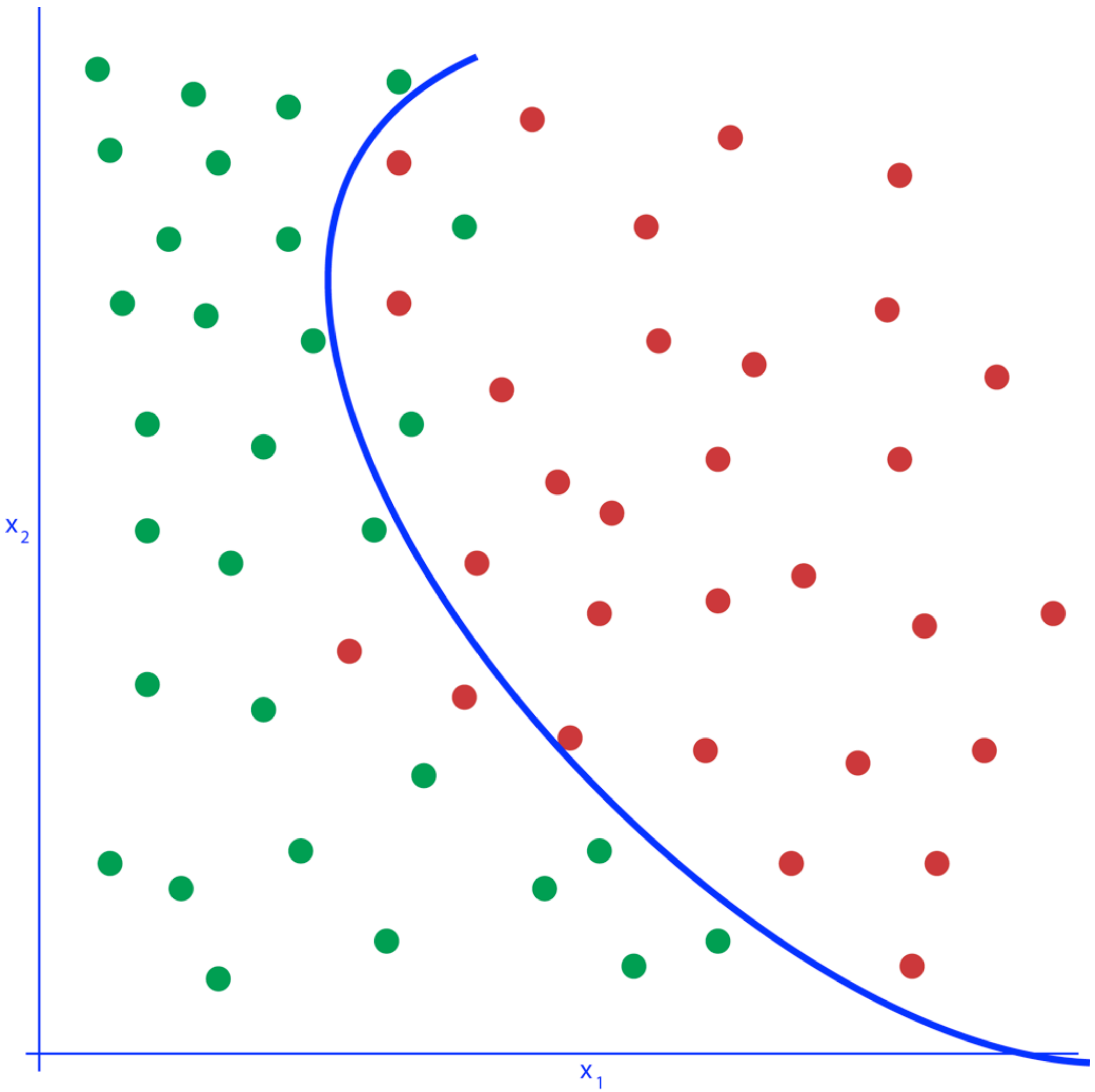



A very simple decision boundary might not work either

This is know as **UNDERFITTING**

# The Learning Process

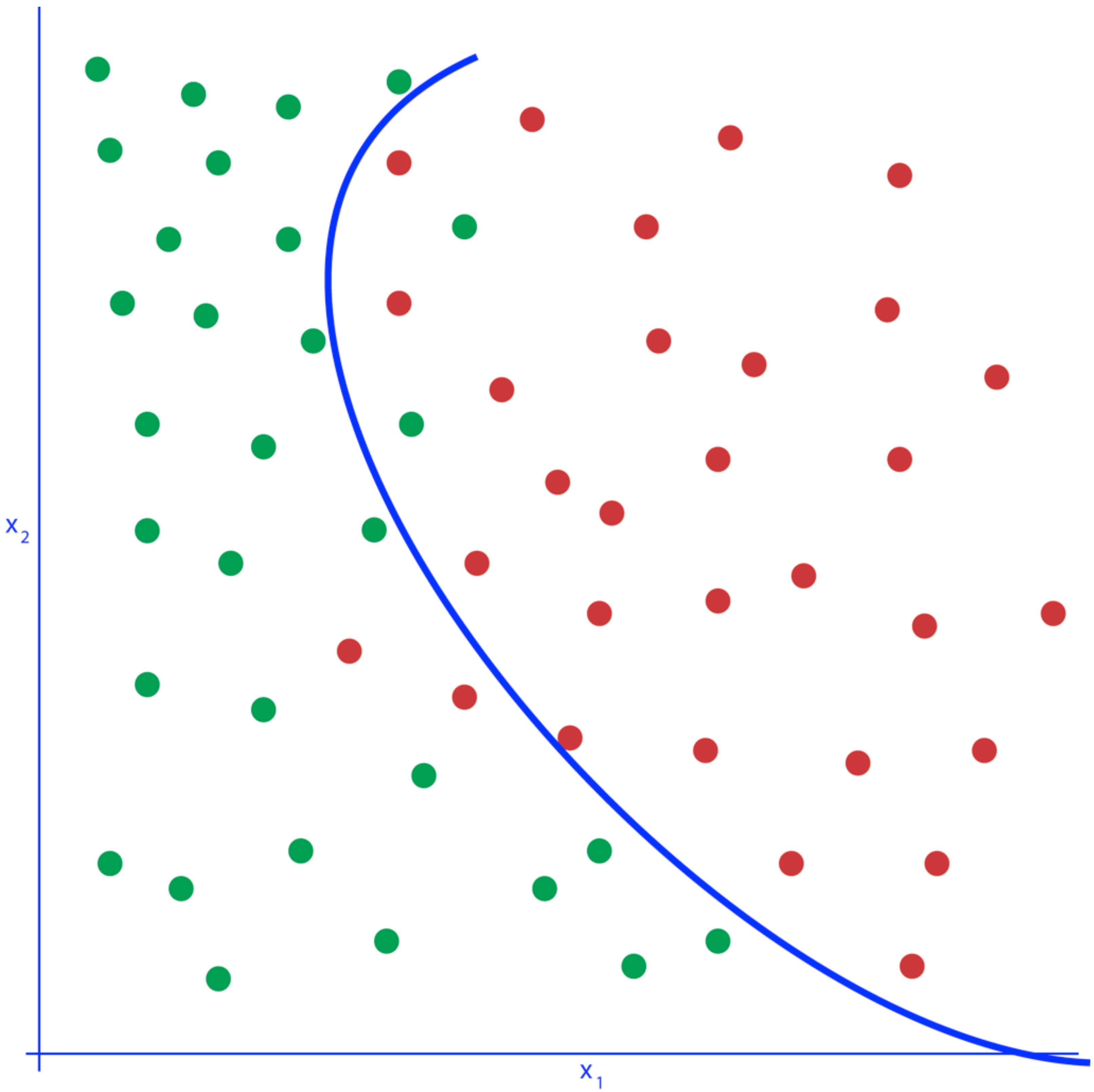
Need to be careful that we don't overtrain the model…




Instead, a less complex decision boundary might work much better, even if it does not perfectly reduce the error on the training data

# The Learning Process

Need to be careful that we don't overtrain the model...




Instead, a less complex decision boundary might work much better, even if it does not perfectly reduce the error on the training data

A model's ability to correctly predict the values of unseen data is know as **GENERALIZATION**

# Testing a Model's Generalization Ability

# Testing a Model's Generalization Ability

Important not to use the training data to test a model!

# Testing a Model's Generalization Ability

Important not to use the training data to test a model!

Instead use a test dataset

# Testing a Model's Generalization Ability

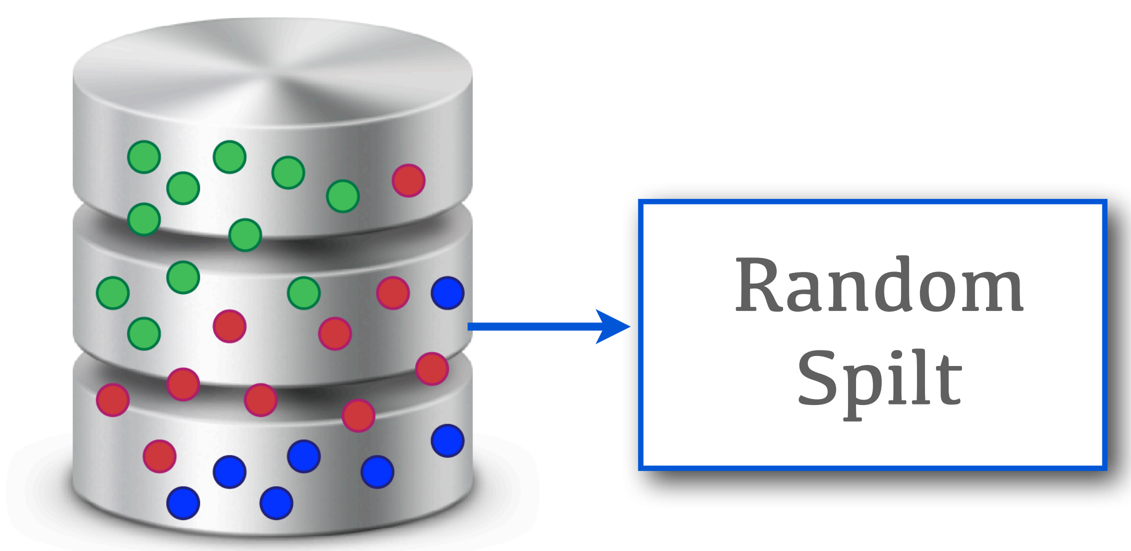
Important not to use the training data to test a model!

Instead use a test dataset

# Testing a Model’s Generalization Ability

Important not to use the training data to test a model!

Instead use a test dataset

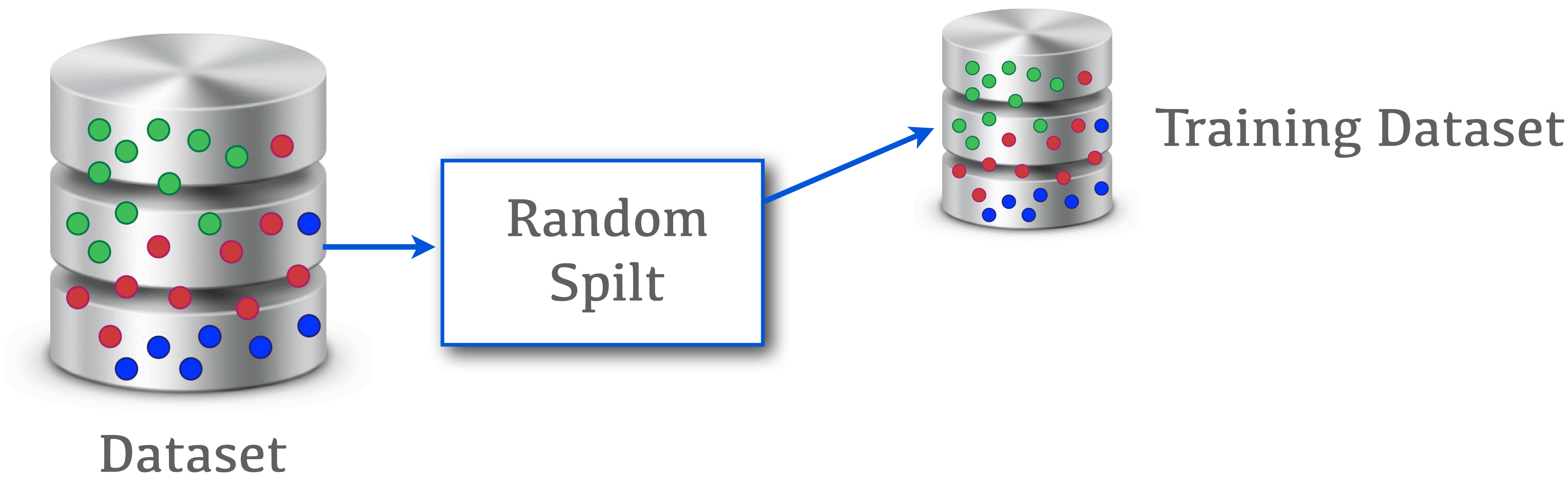



Dataset

# Testing a Model's Generalization Ability

Important not to use the training data to test a model!

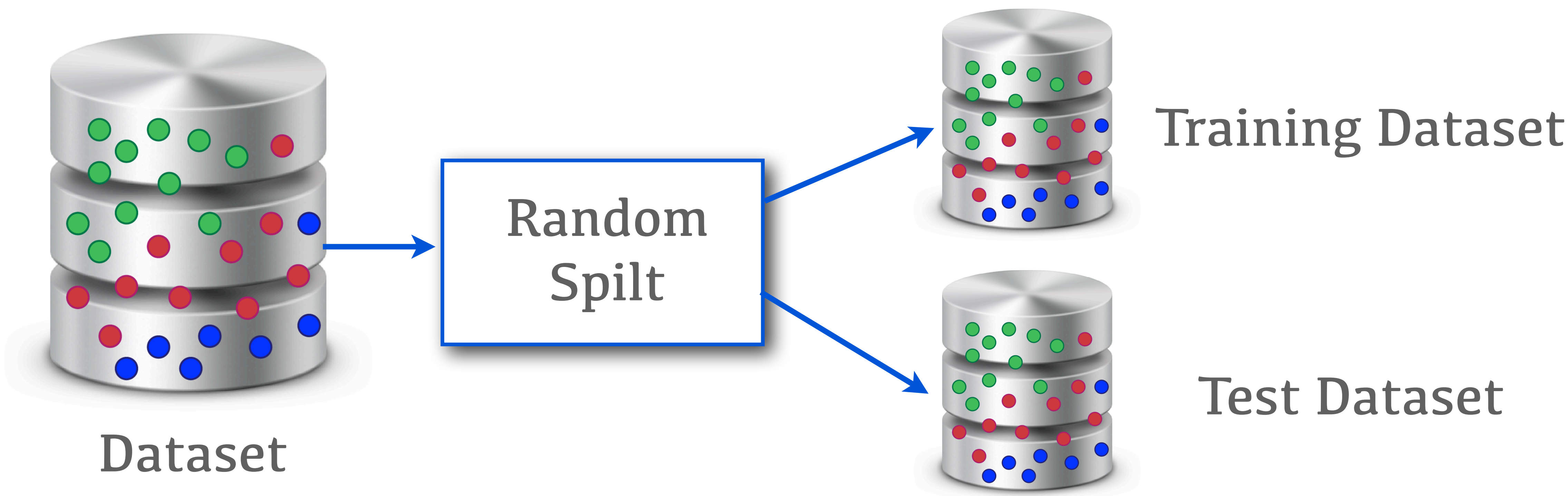
Instead use a test dataset



# Testing a Model's Generalization Ability

Important not to use the training data to test a model!

Instead use a test dataset



# Testing a Model's Generalization Ability

Sometimes there is not enough data to create a test dataset

# Testing a Model's Generalization Ability

Sometimes there is not enough data to create a test dataset

Instead use K-FOLD CROSS VALIDATION

# Testing a Model's Generalization Ability

Sometimes there is not enough data to create a test dataset

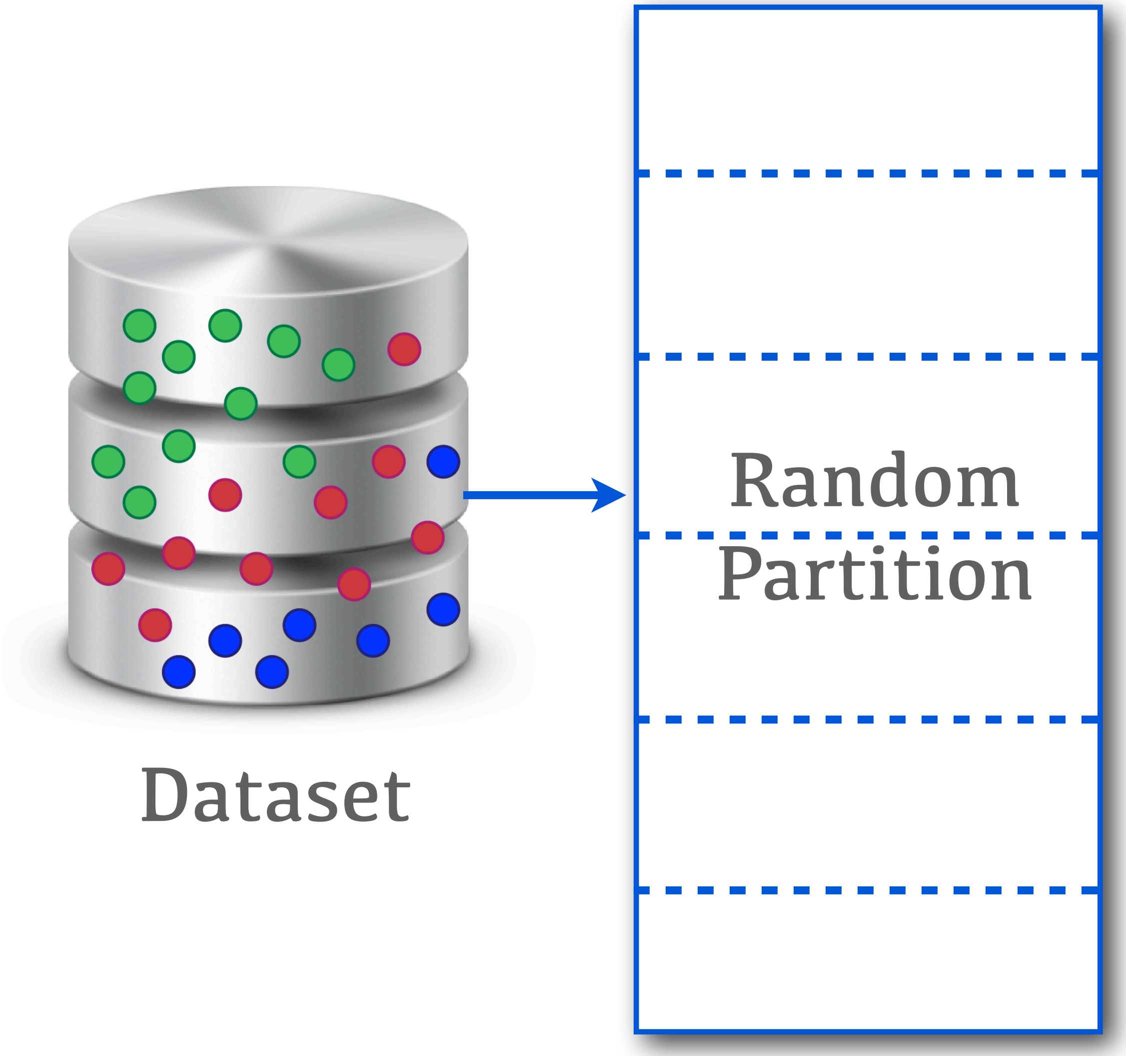
Instead use K-FOLD CROSS VALIDATION

Dataset

# Testing a Model's Generalization Ability

Sometimes there is not enough data to create a test dataset

Instead use K-FOLD CROSS VALIDATION

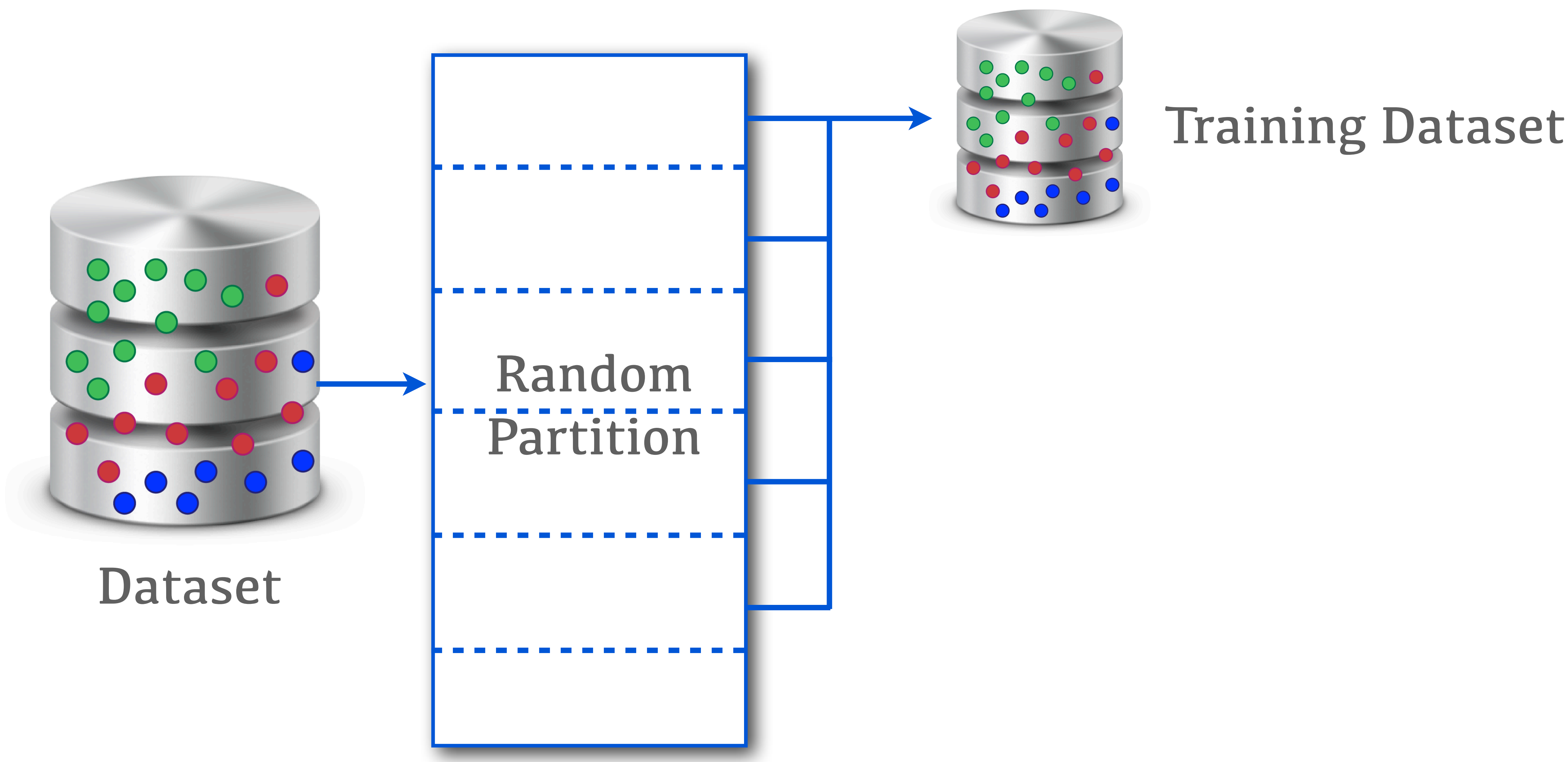



Partition Data into K Folds

# Testing a Model's Generalization Ability

Sometimes there is not enough data to create a test dataset

Instead use K-FOLD CROSS VALIDATION

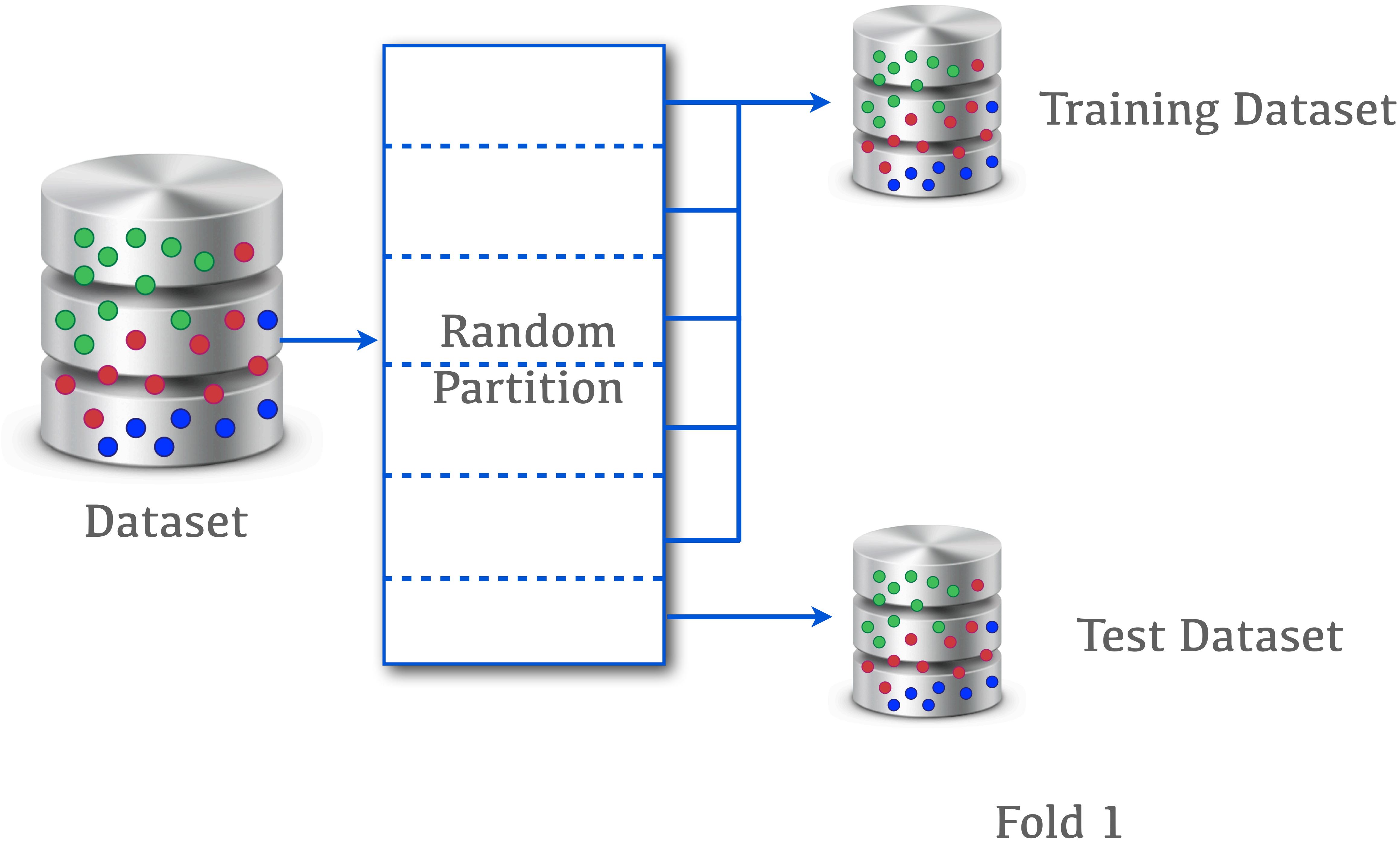



Fold 1

# Testing a Model's Generalization Ability

Sometimes there is not enough data to create a test dataset

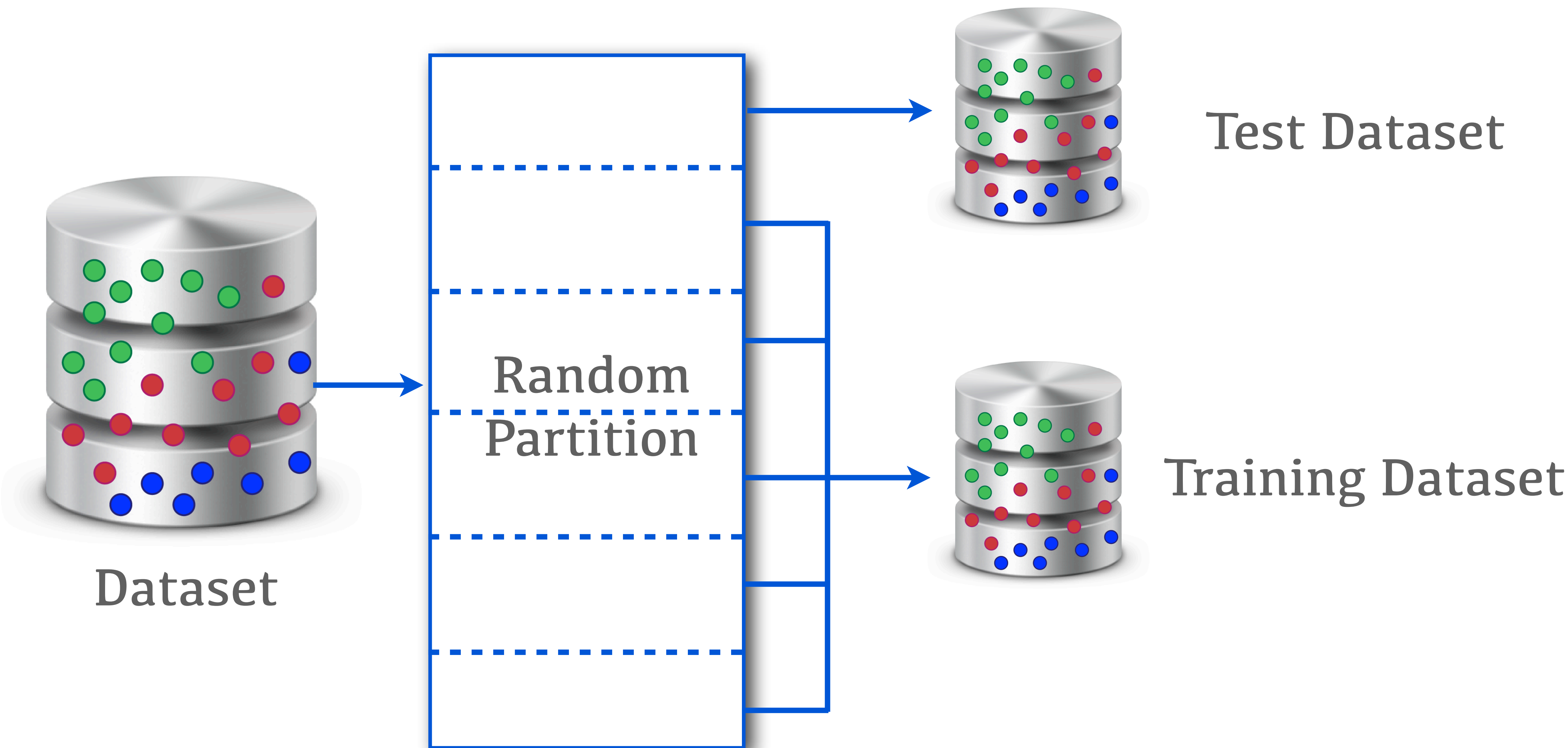
Instead use K-FOLD CROSS VALIDATION



# Testing a Model's Generalization Ability

Sometimes there is not enough data to create a test dataset

Instead use K-FOLD CROSS VALIDATION

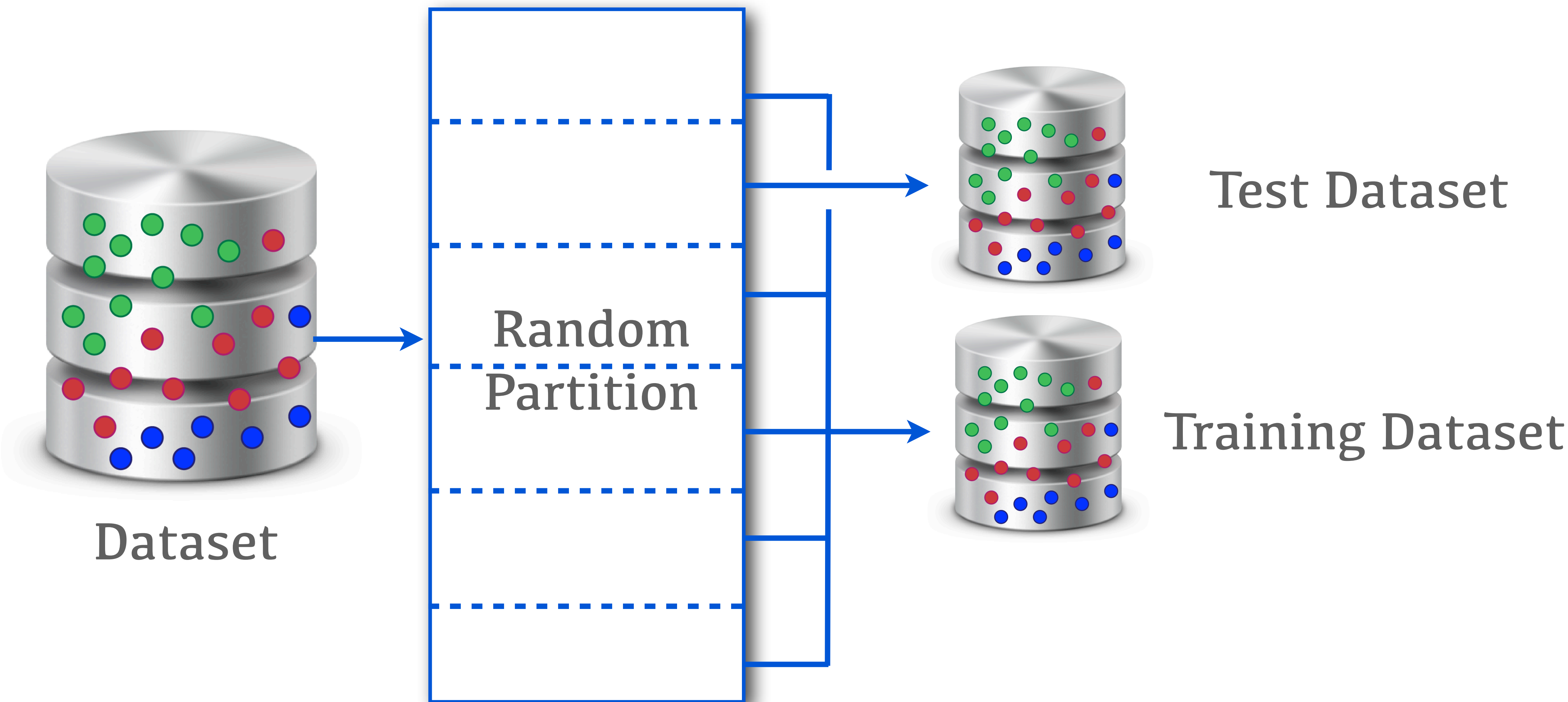



Fold 2

# Testing a Model's Generalization Ability

Sometimes there is not enough data to create a test dataset

Instead use K-FOLD CROSS VALIDATION




Fold 3

# Testing a Model's Generalization Ability

$$\text{Classification Accuracy} = \frac{\text{Num Correctly Classified Examples}}{\text{Num Test Examples}}$$

# Testing a Model's Generalization Ability

$$\text{Classification Accuracy} = \frac{\text{Num Correctly Classified Examples}}{\text{Num Test Examples}}$$

$$\text{Precision}_k = \frac{\text{Num Correctly Classified Examples for Class } k}{\text{Num Examples Classified as Class } k}$$

# Testing a Model's Generalization Ability

$$\text{Classification Accuracy} = \frac{\text{Num Correctly Classified Examples}}{\text{Num Test Examples}}$$

$$\text{Precision}_k = \frac{\text{Num Correctly Classified Examples for Class } k}{\text{Num Examples Classified as Class } k}$$

$$\text{Recall}_k = \frac{\text{Num Correctly Classified Examples for Class } k}{\text{Num Class } k \text{ Examples}}$$

# Testing a Model's Generalization Ability

$$\text{Classification Accuracy} = \frac{\text{Num Correctly Classified Examples}}{\text{Num Test Examples}}$$

$$\text{Precision}_k = \frac{\text{Num Correctly Classified Examples for Class } k}{\text{Num Examples Classified as Class } k}$$

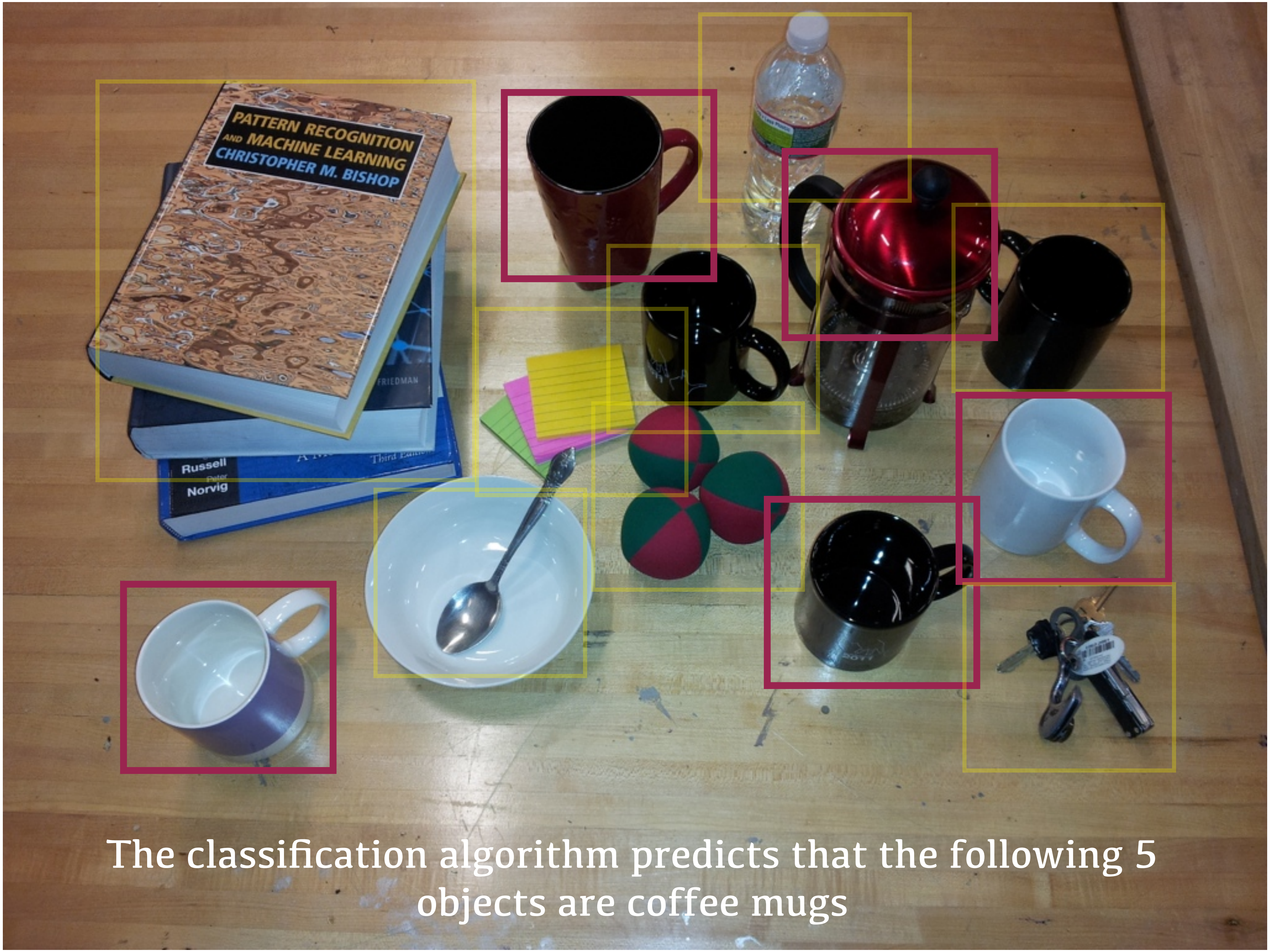
$$\text{Recall}_k = \frac{\text{Num Correctly Classified Examples for Class } k}{\text{Num Class } k \text{ Examples}}$$

$$\text{F-measure}_k = 2 * \frac{\text{Precision}_k * \text{Recall}_k}{\text{Precision}_k + \text{Recall}_k}$$

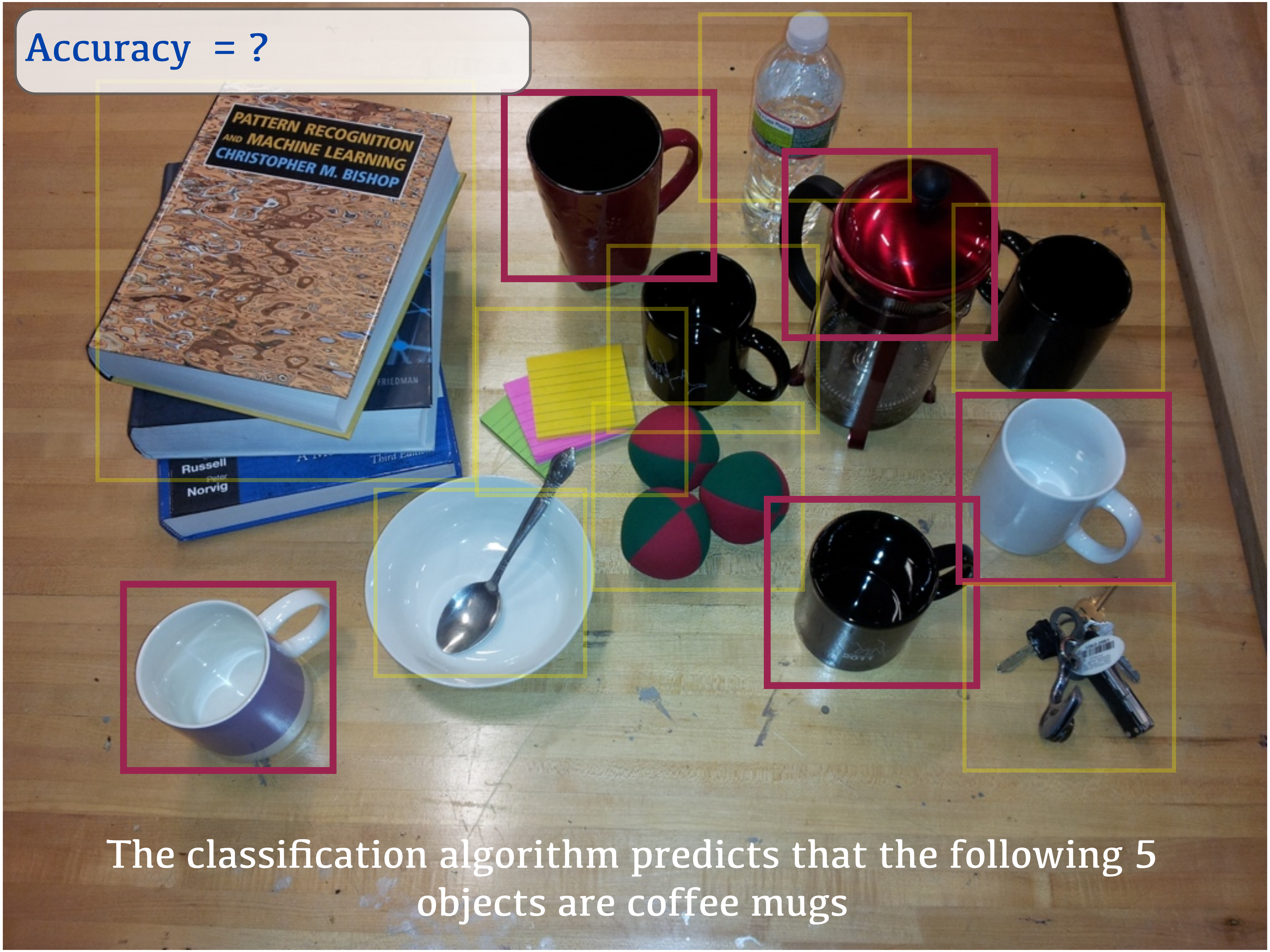
# Testing a Model's Generalization Ability

# Testing a Model's Generalization Ability

# Testing a Model's Generalization Ability



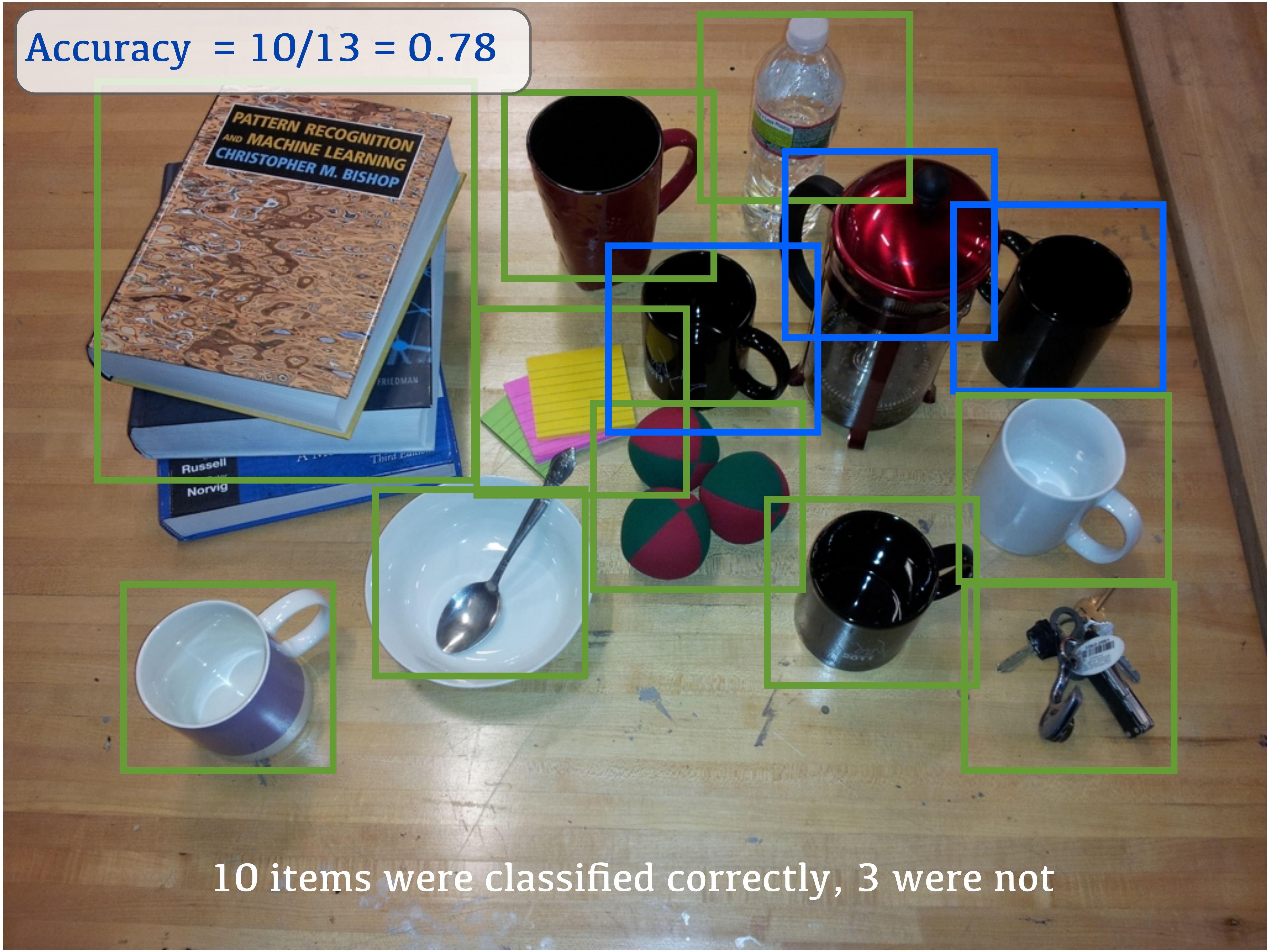
# Testing a Model's Generalization Ability



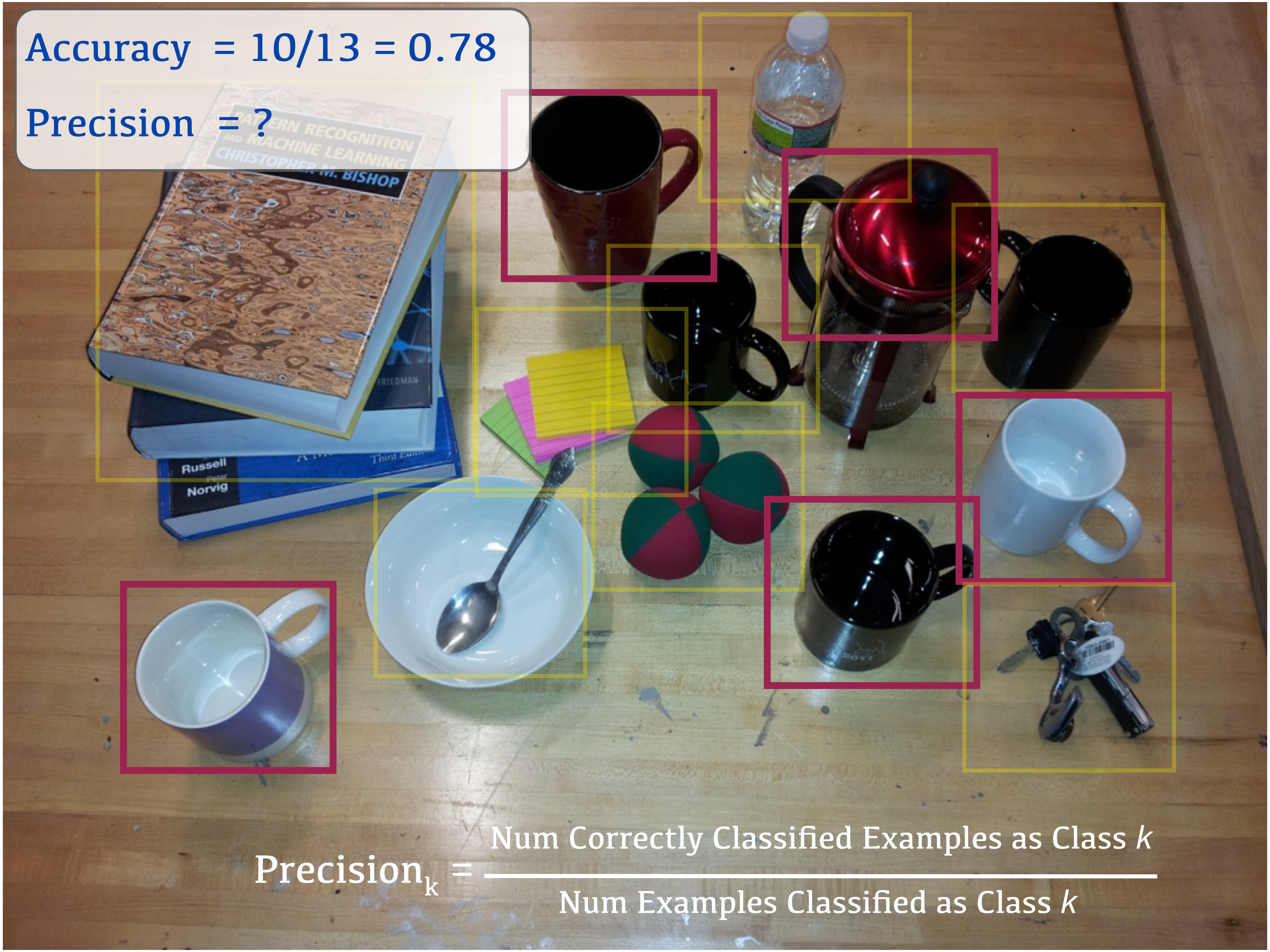

Accuracy = ?

The classification algorithm predicts that the following 5 objects are coffee mugs

# Testing a Model's Generalization Ability

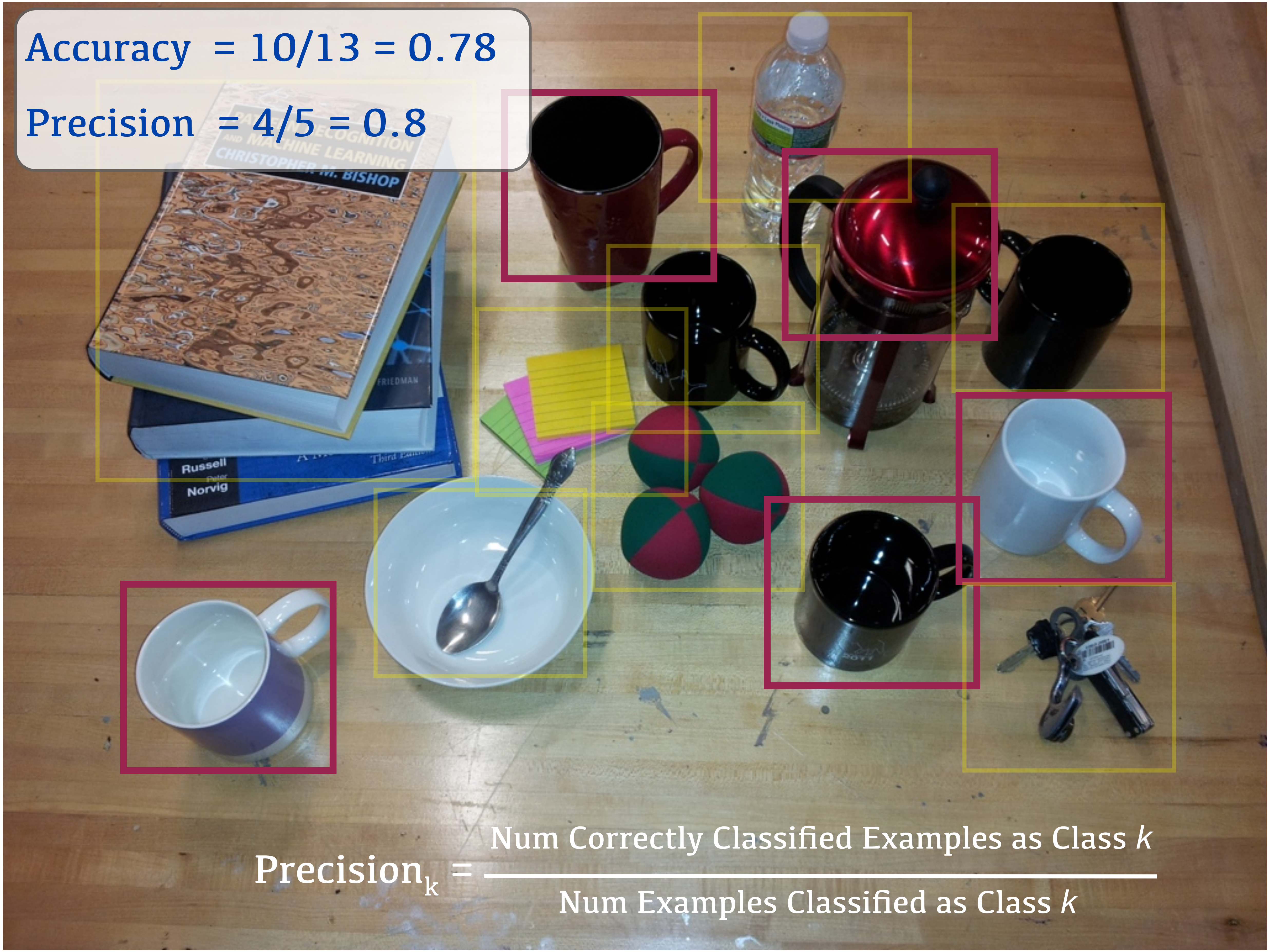


# Testing a Model's Generalization Ability

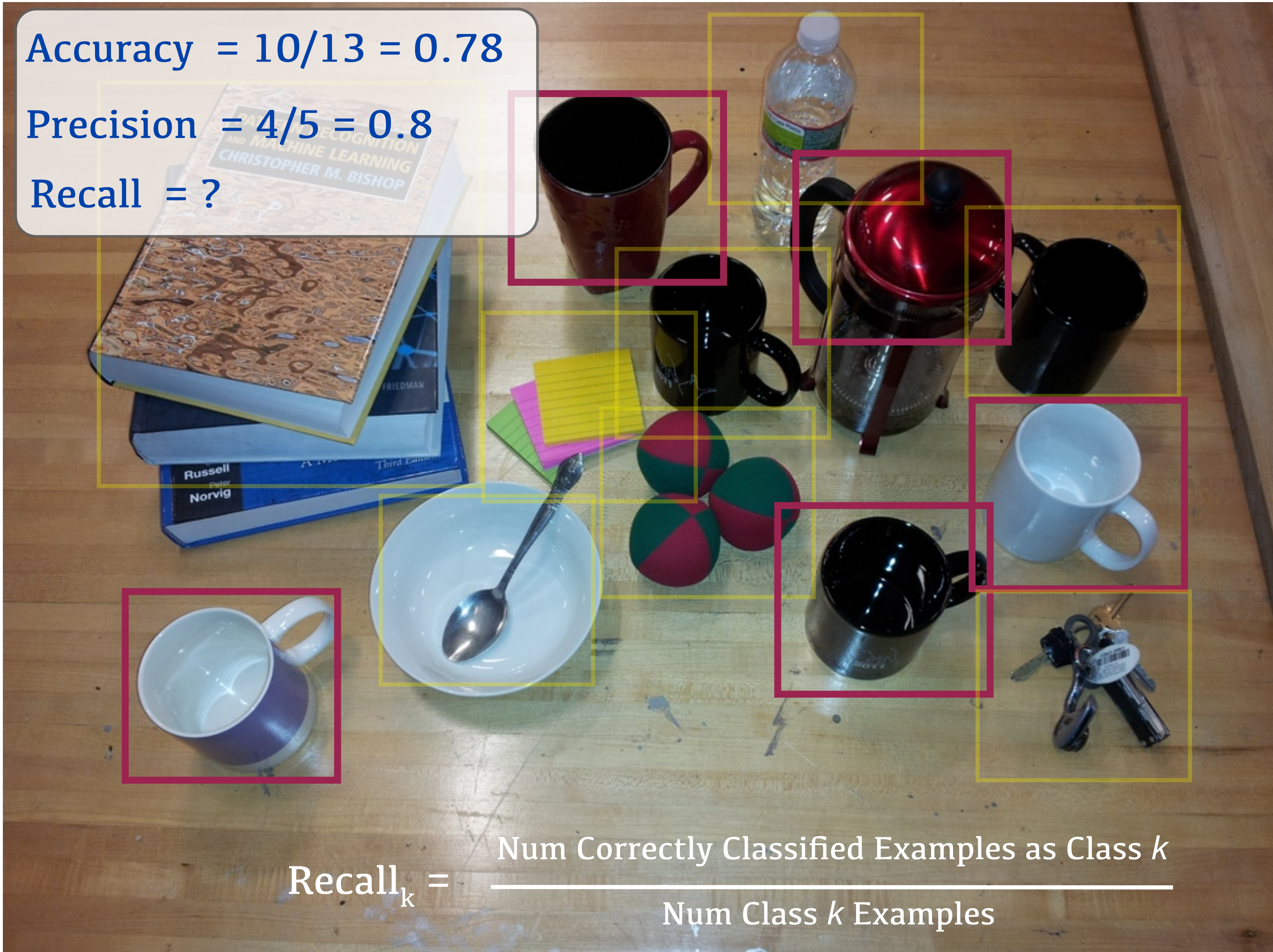



$$\text{Precision}_k = \frac{\text{Num Correctly Classified Examples as Class } k}{\text{Num Examples Classified as Class } k}$$

# Testing a Model's Generalization Ability



# Testing a Model's Generalization Ability



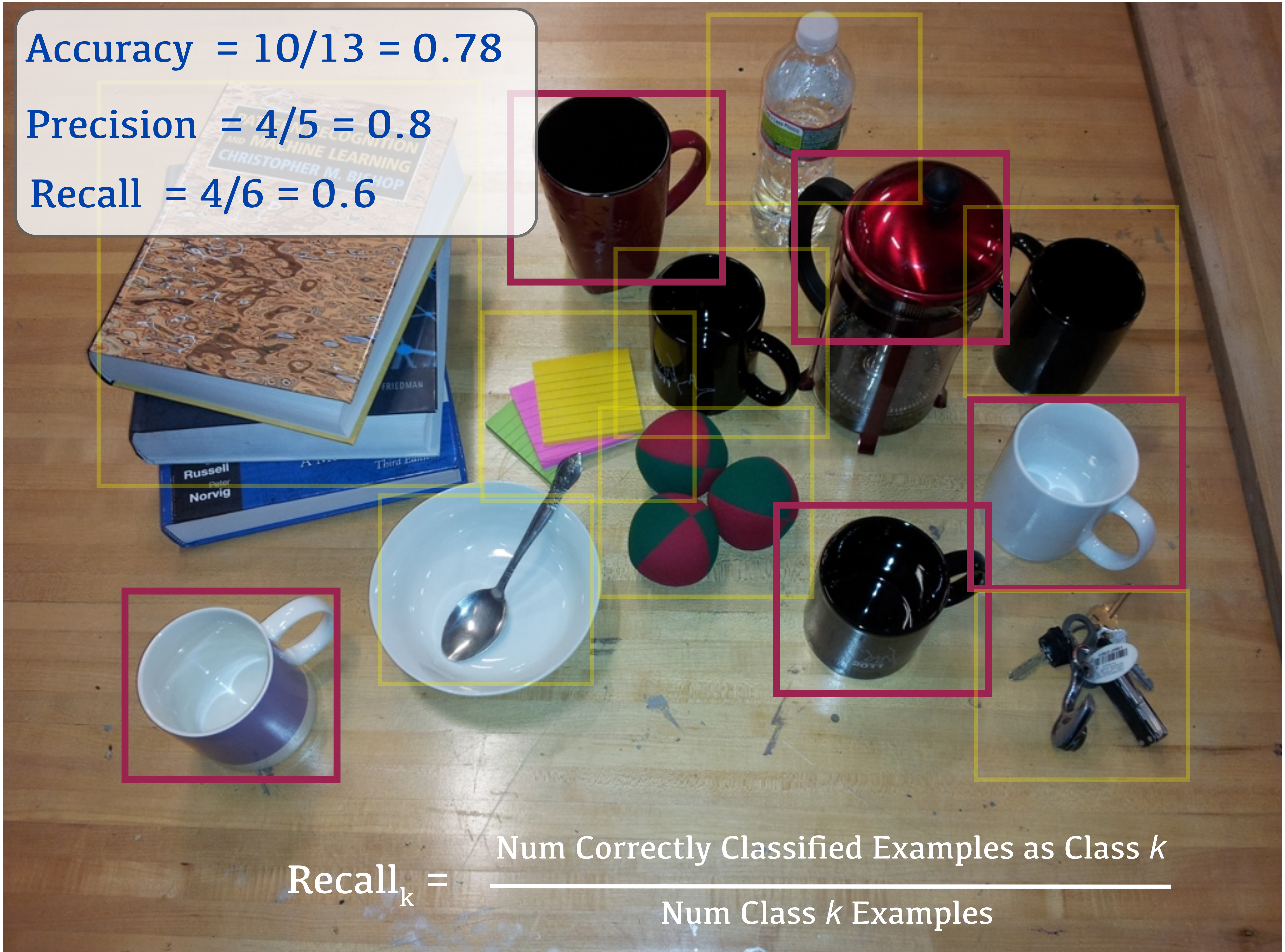

Accuracy = 10/13 = 0.78

Precision = 4/5 = 0.8

Recall = ?

$$\text{Recall}_k = \frac{\text{Num Correctly Classified Examples as Class } k}{\text{Num Class } k \text{ Examples}}$$

# Testing a Model's Generalization Ability



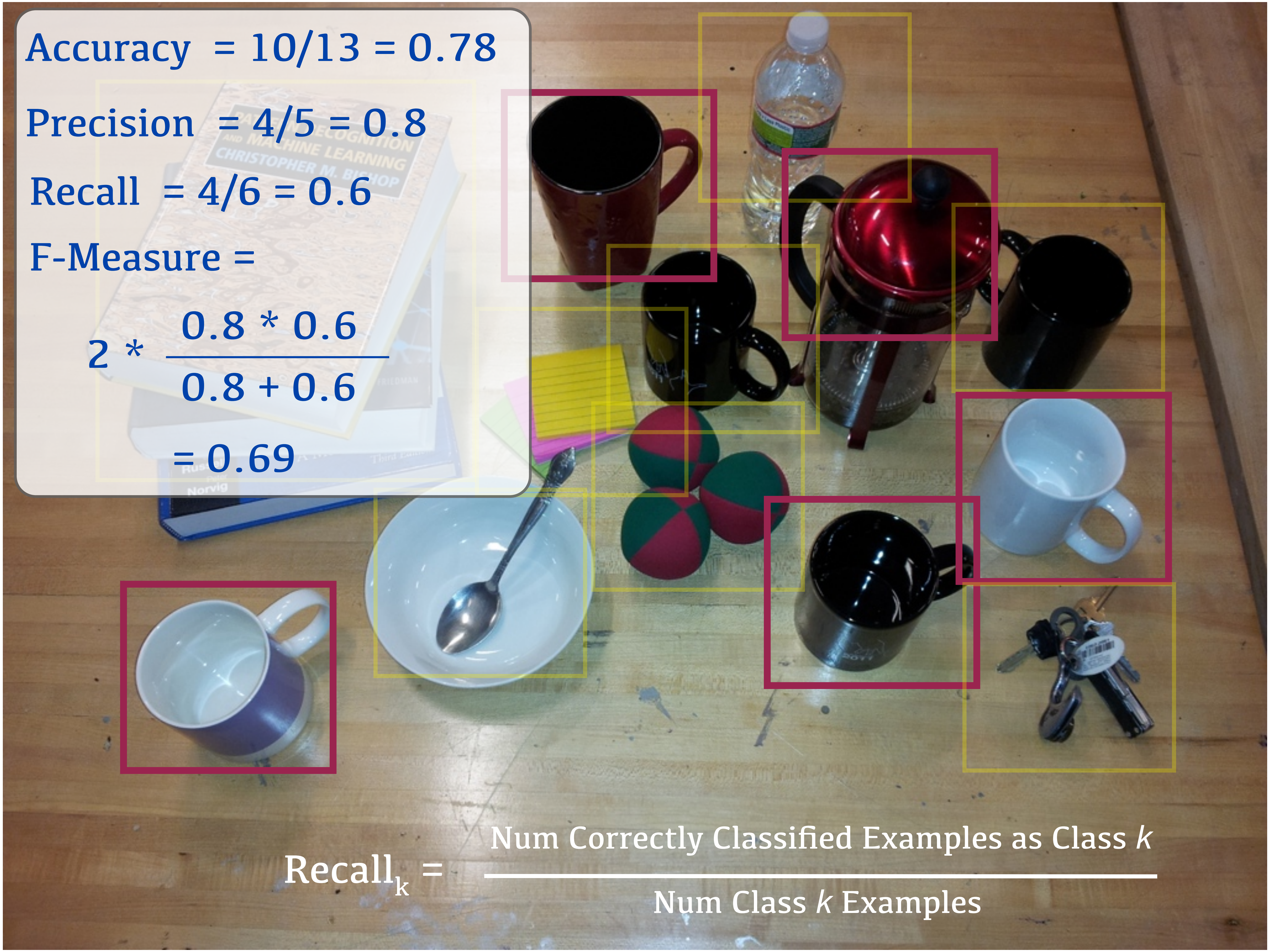

Accuracy $= 10/13 = 0.78$

Precision $= 4/5 = 0.8$

Recall $= 4/6 = 0.6$

$$\text{Recall}_k = \frac{\text{Num Correctly Classified Examples as Class } k}{\text{Num Class } k \text{ Examples}}$$

# Testing a Model's Generalization Ability



Accuracy $= 10/13 = 0.78$

Precision $= 4/5 = 0.8$

Recall $= 4/6 = 0.6$

F-Measure =

$$2 * \frac{0.8 * 0.6}{0.8 + 0.6}$$

$$= 0.69$$

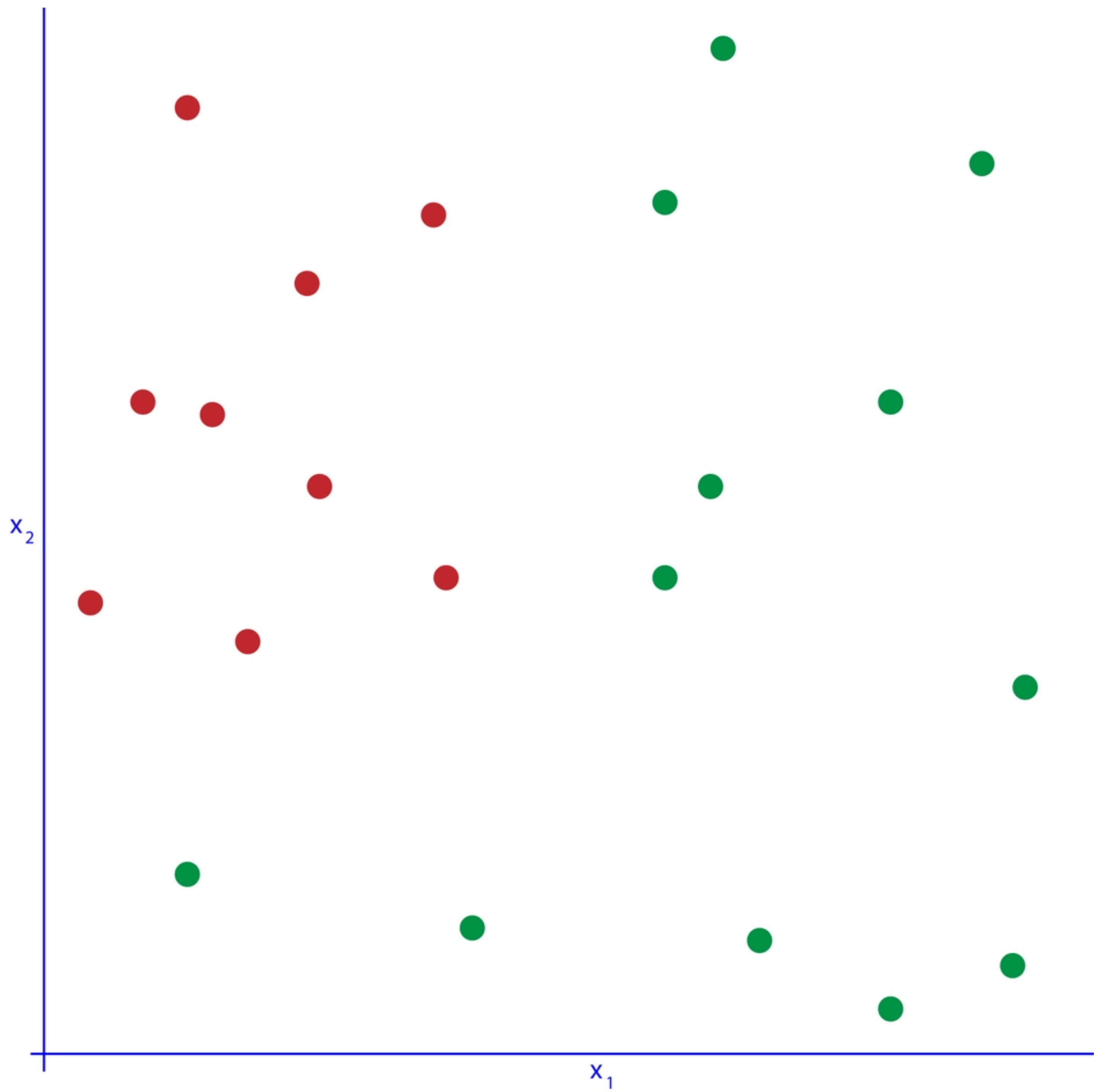
$$\text{Recall}_k = \frac{\text{Num Correctly Classified Examples as Class } k}{\text{Num Class } k \text{ Examples}}$$

# A Simple Classifier Example

# K-Nearest Neighbor Classifier (KNN)

# K-Nearest Neighbor Classifier (KNN)

Training Data

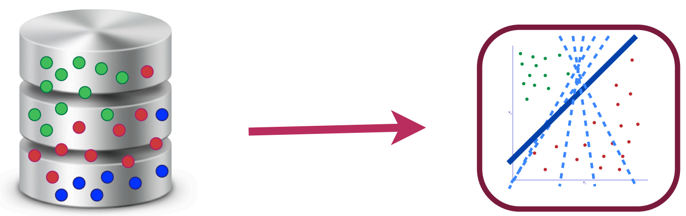



Training Data:

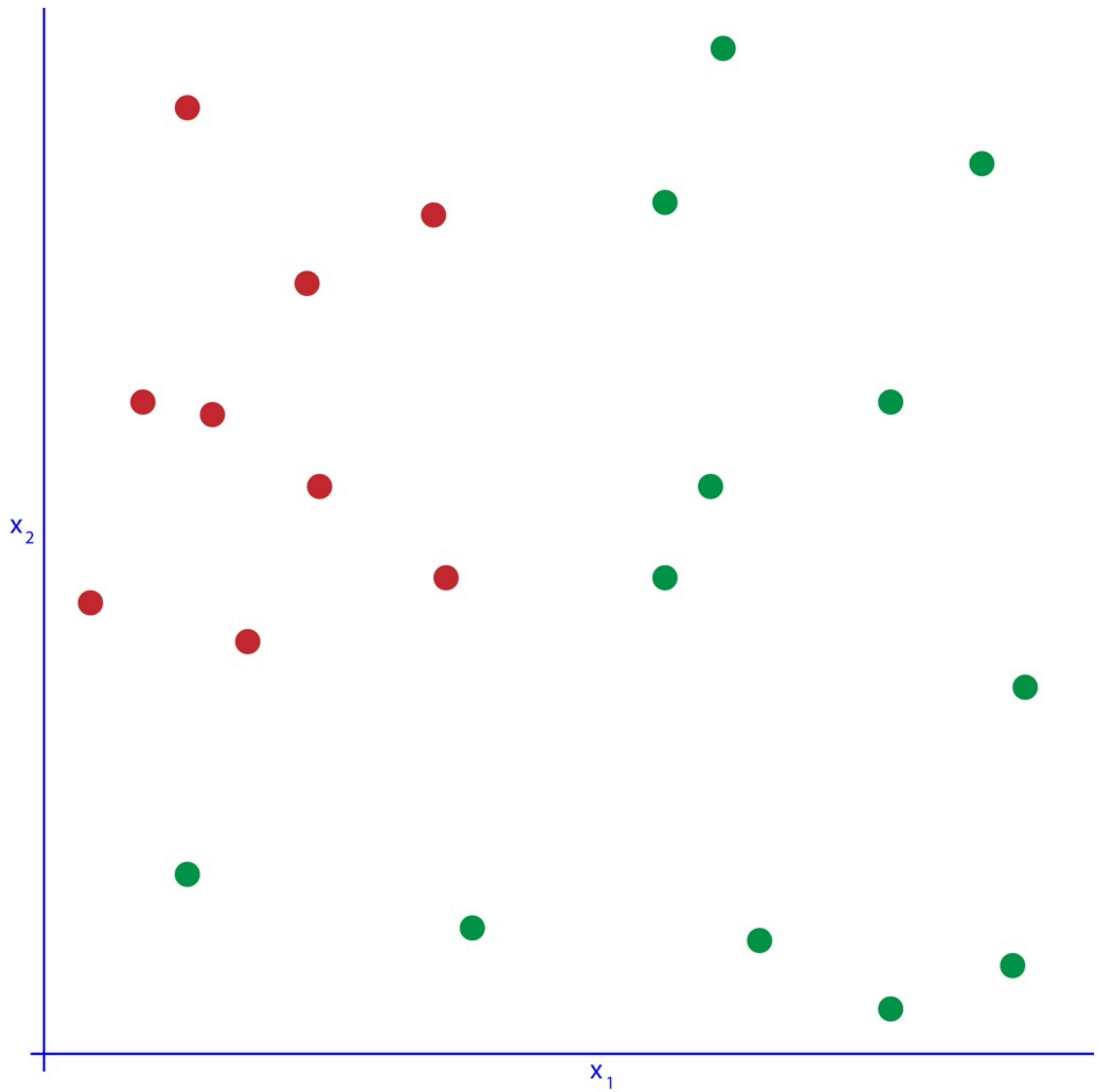
- M Labelled Training Examples

- Each example is an N-Dimensional Vector

# K-Nearest Neighbor Classifier (KNN)



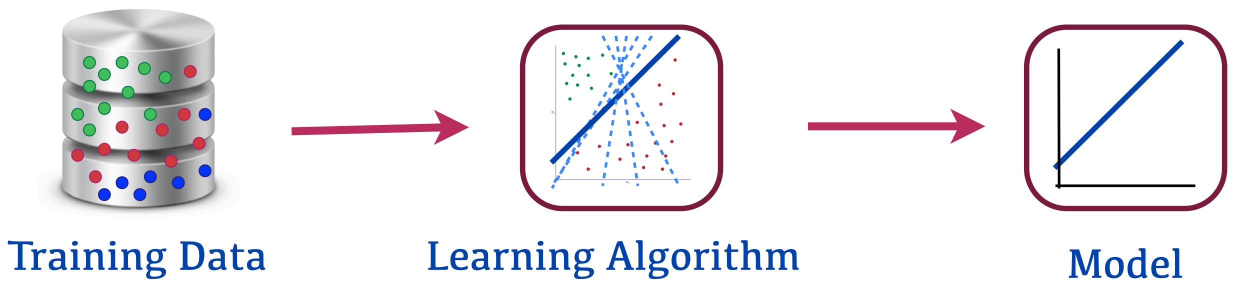
Training Data    Learning Algorithm

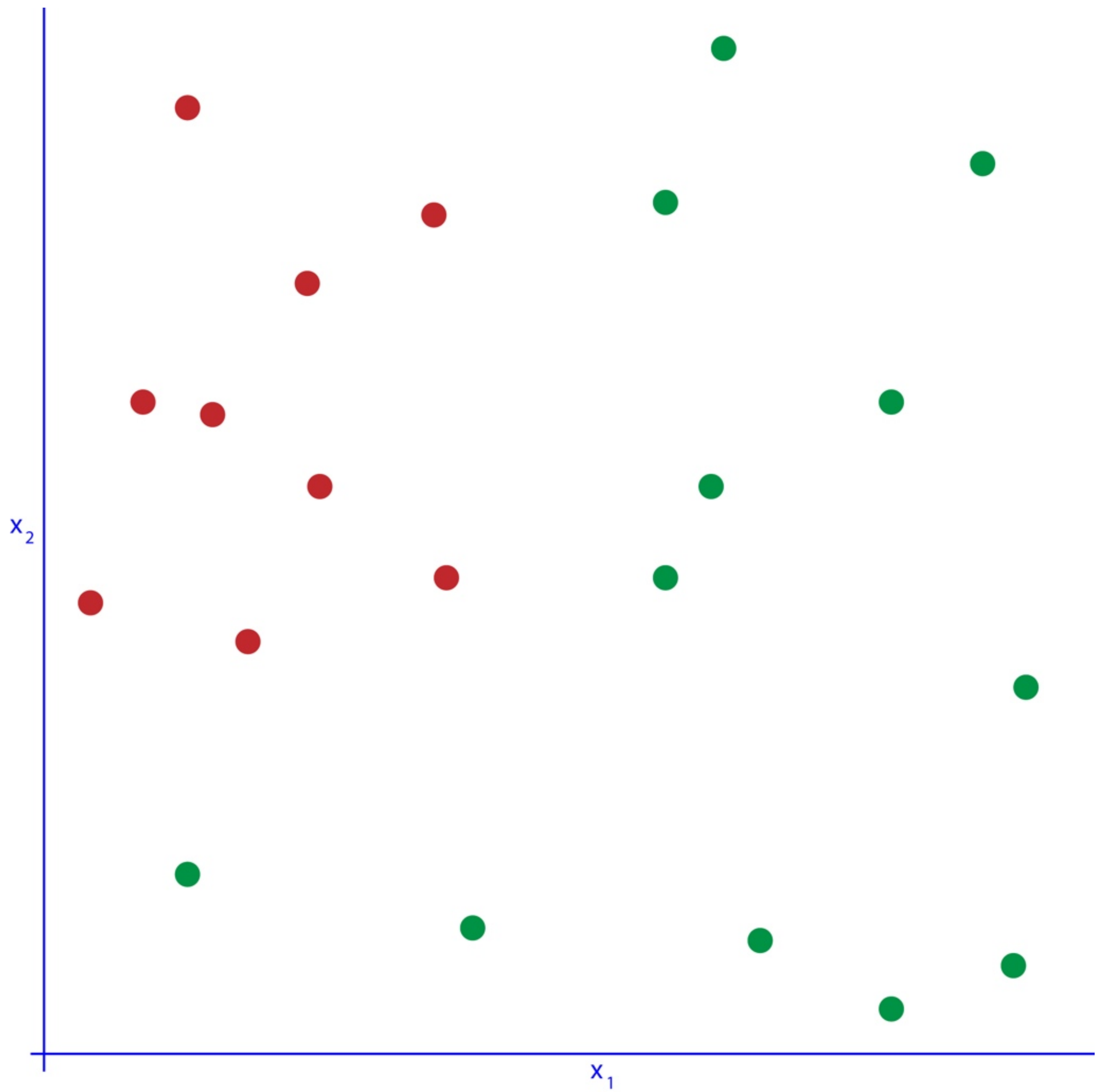



Training Phase:

- Simply save the labelled training examples

# K-Nearest Neighbor Classifier (KNN)

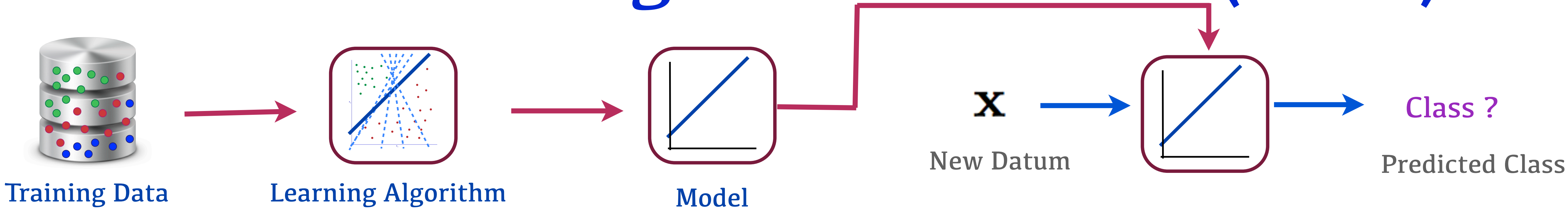


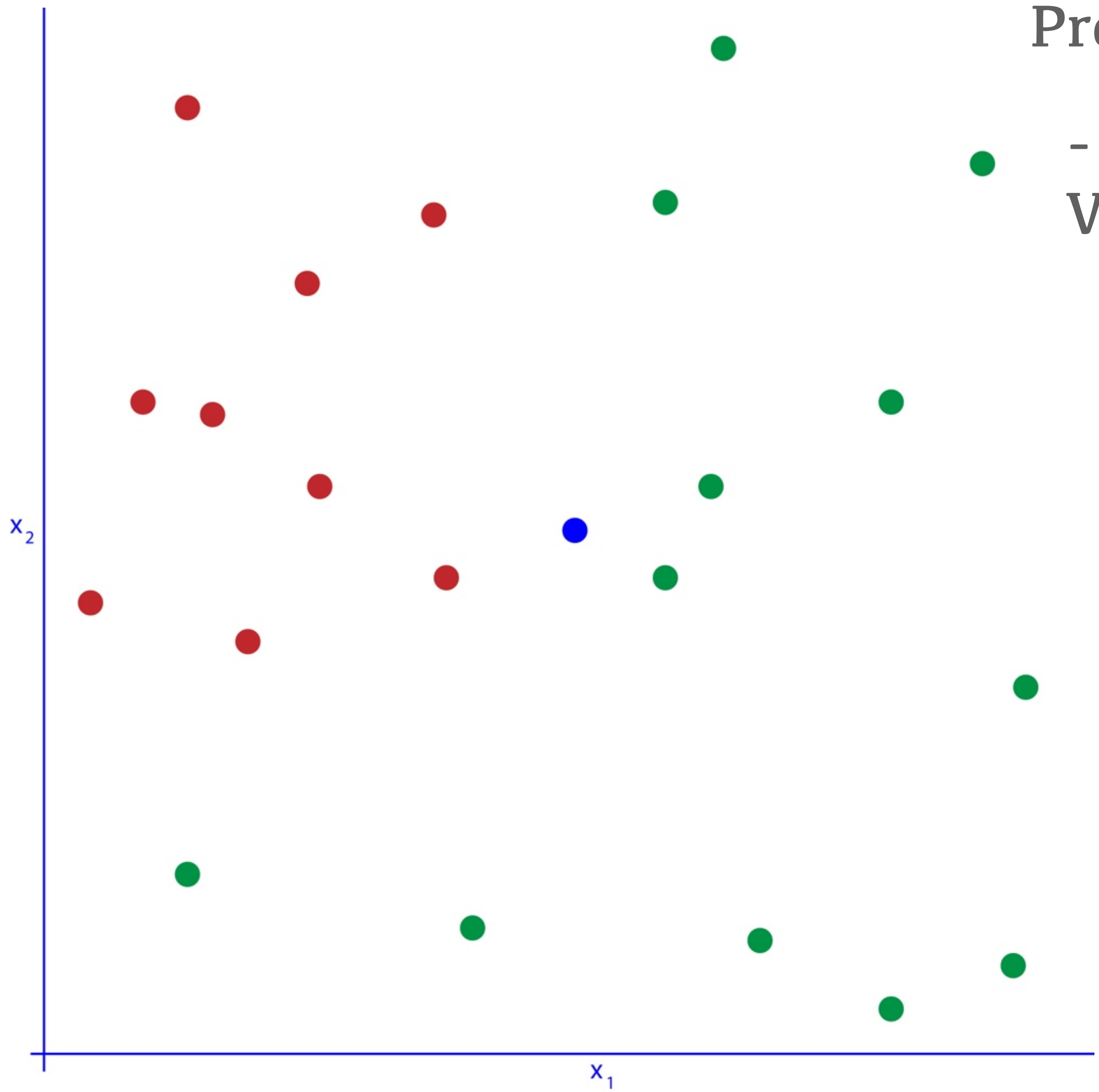




Model:

- Labelled training examples

# K-Nearest Neighbor Classifier (KNN)

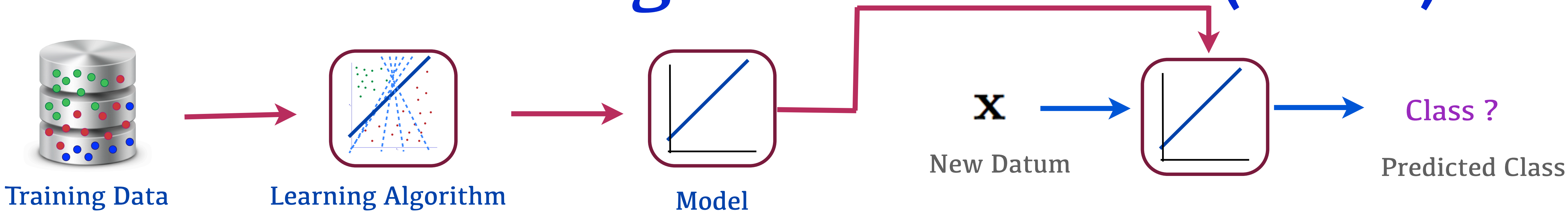


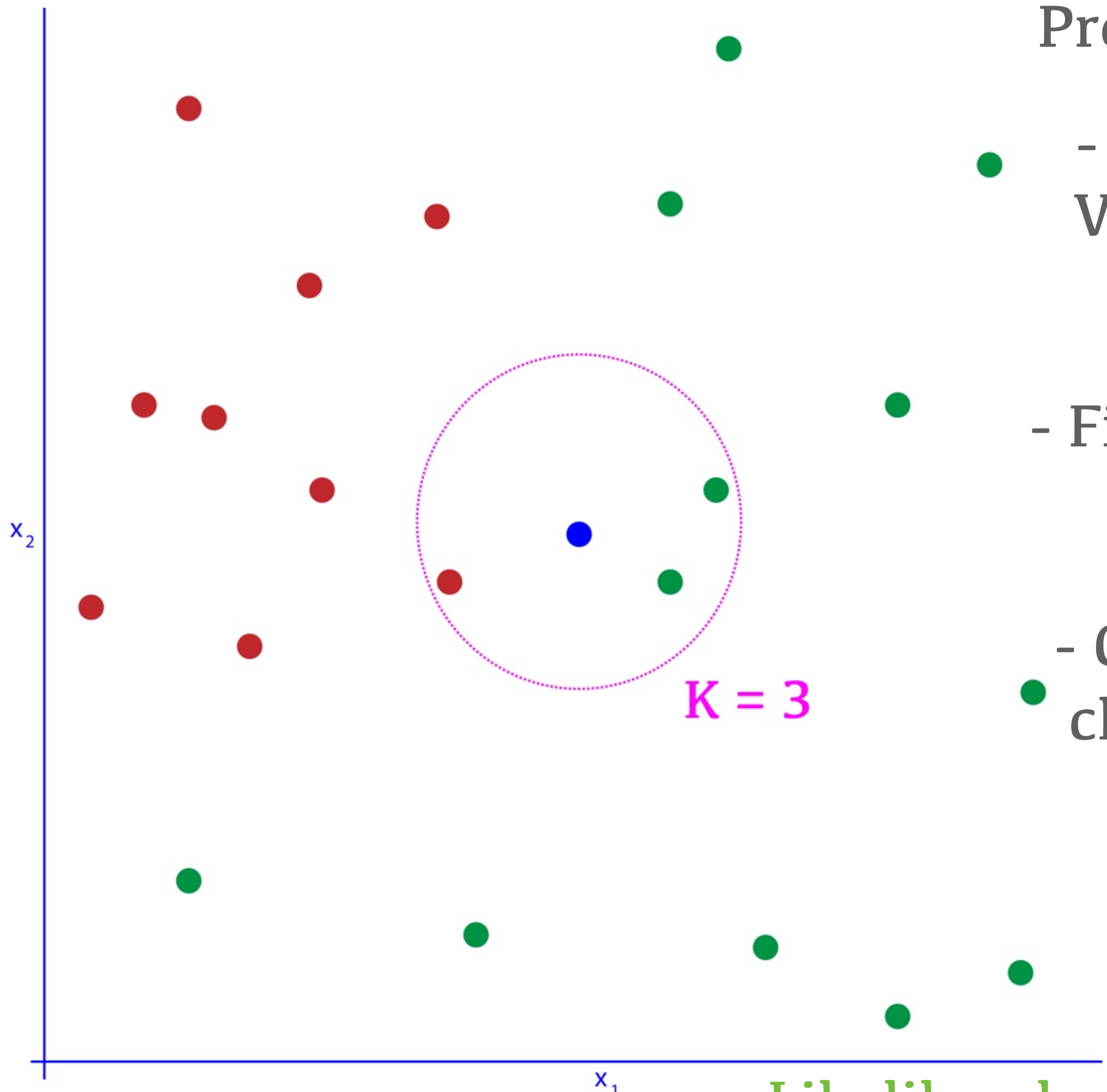




Prediction Phase:

- Given a new N-Dimensional Vector, predict which class it belongs to

# K-Nearest Neighbor Classifier (KNN)




Prediction Phase:

- Given a new N-Dimensional Vector, predict which class it belongs to

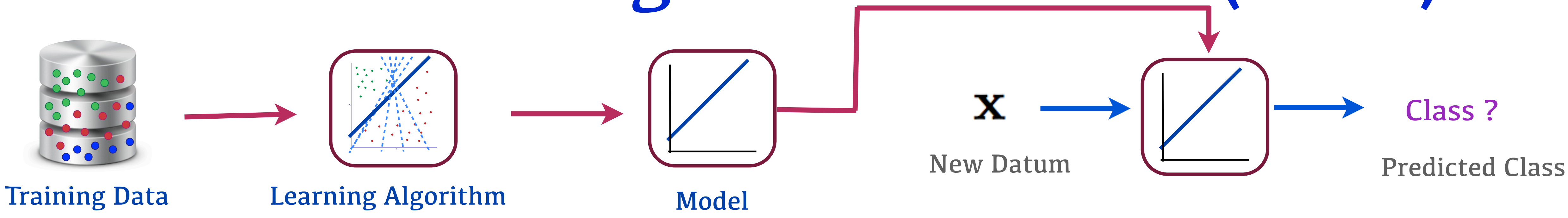
- Find the K Nearest Neighbors in the training examples

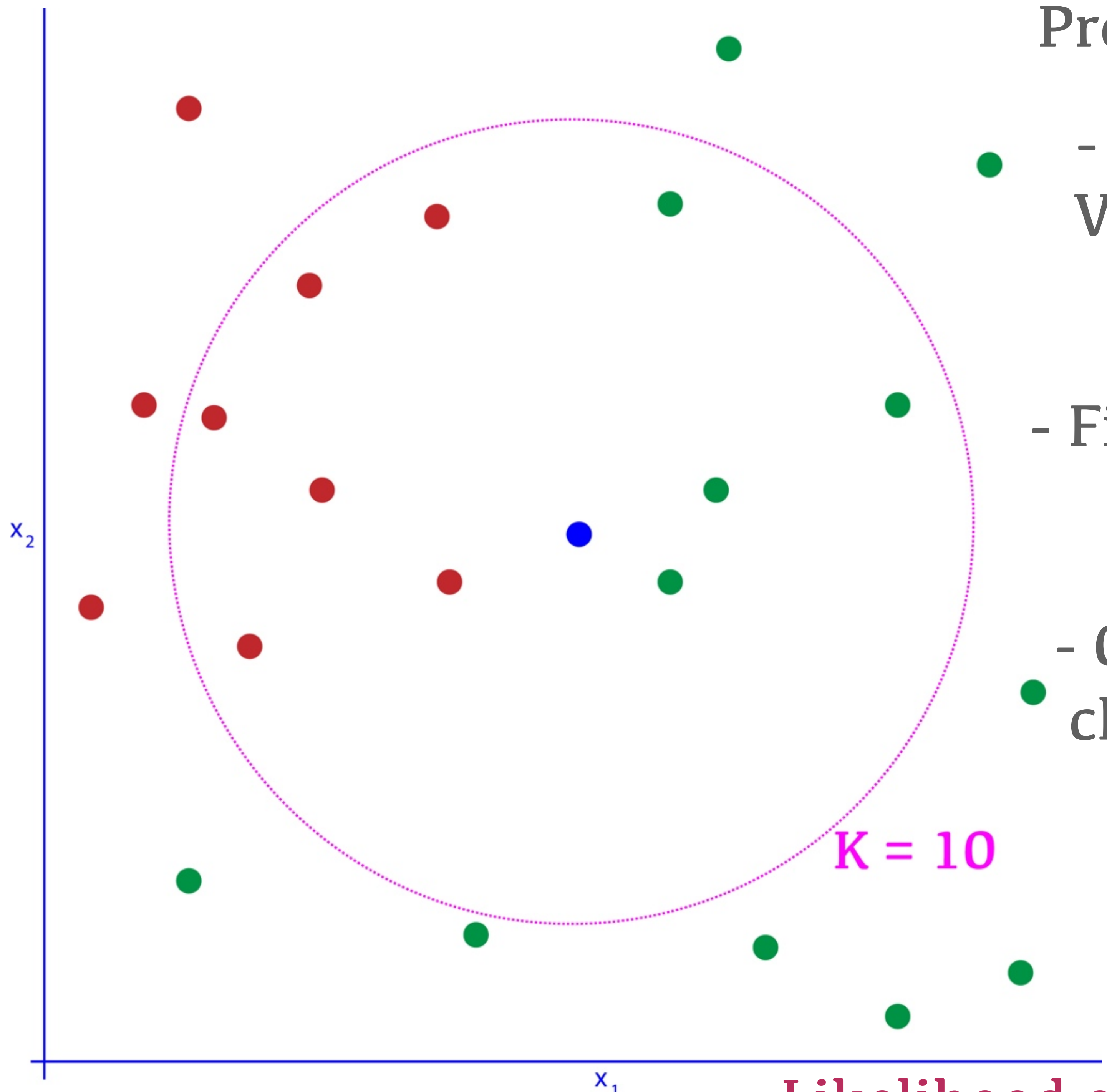
- Classify **x** as the most likely class (i.e. the most common class in the K Nearest Neighbors)

Class A: 2 Class B: 1

Likelihood of belonging to Class A = 0.6

# K-Nearest Neighbor Classifier (KNN)




Prediction Phase:

- Given a new N-Dimensional Vector, predict which class it belongs to

- Find the K Nearest Neighbors in the training examples

- Classify **x** as the most likely class (i.e. the most common class in the K Nearest Neighbors)

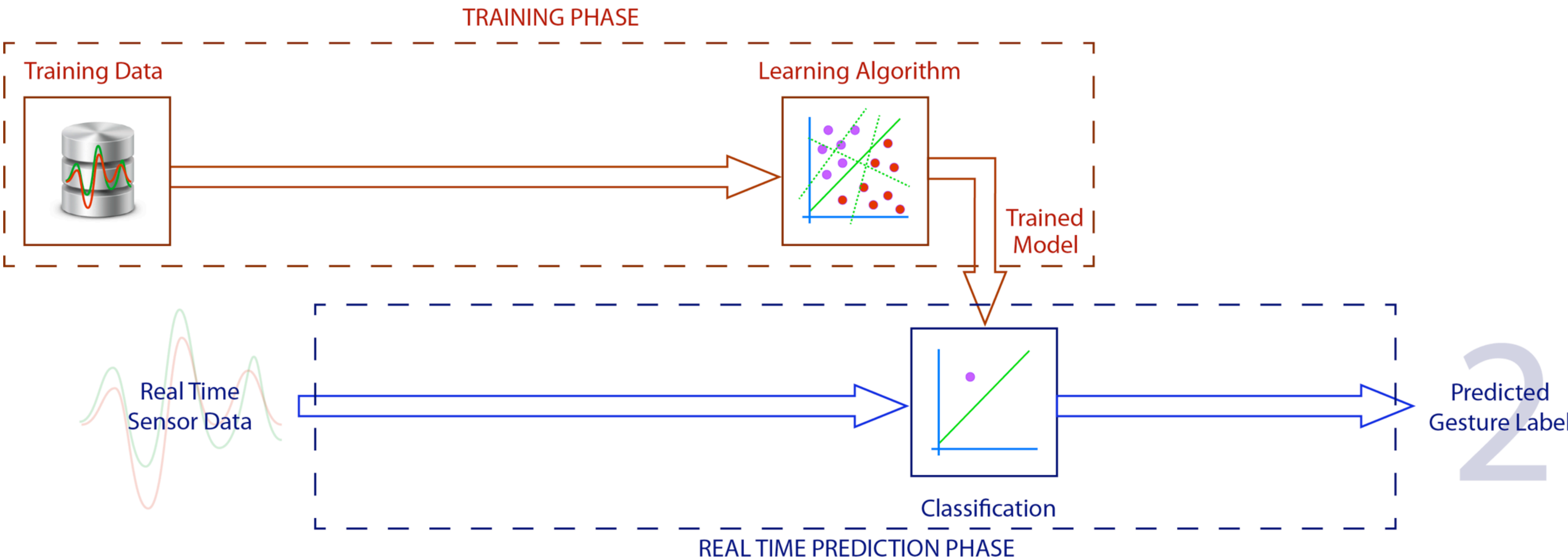
Class A: 4 Class B: 6

Likelihood of belonging to Class B = 0.6

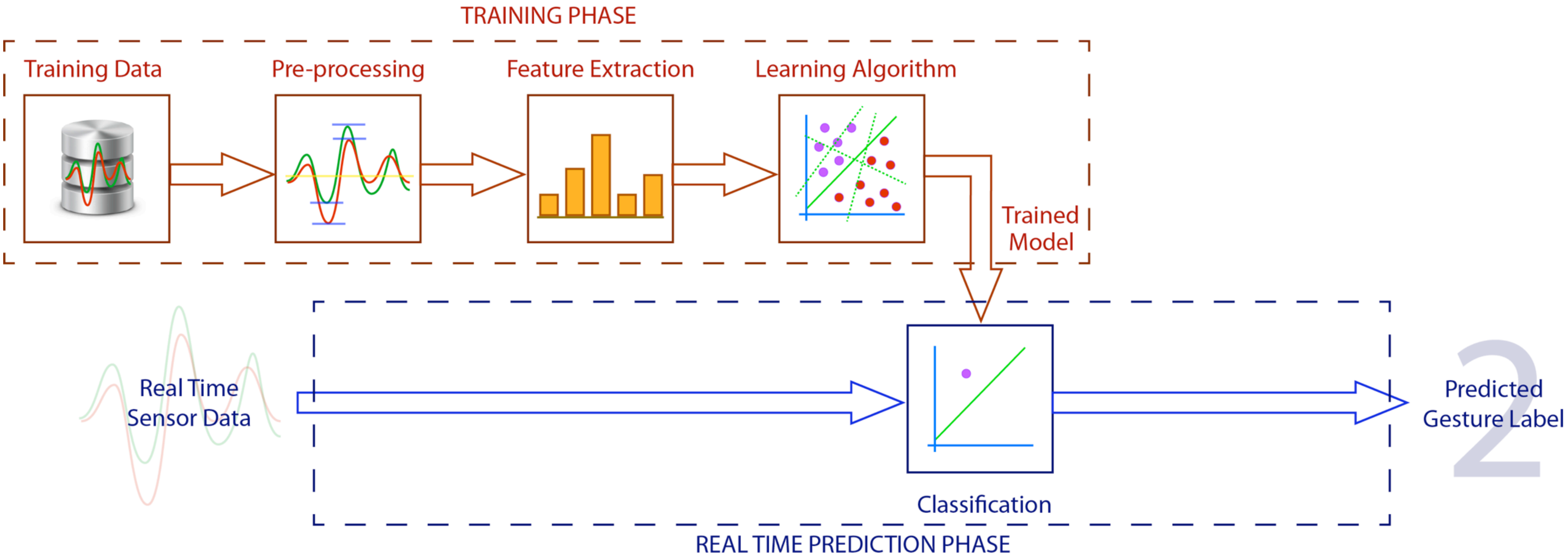
# Hello World - KNN Demo

# Gesture Recognition

# Gesture Recognition

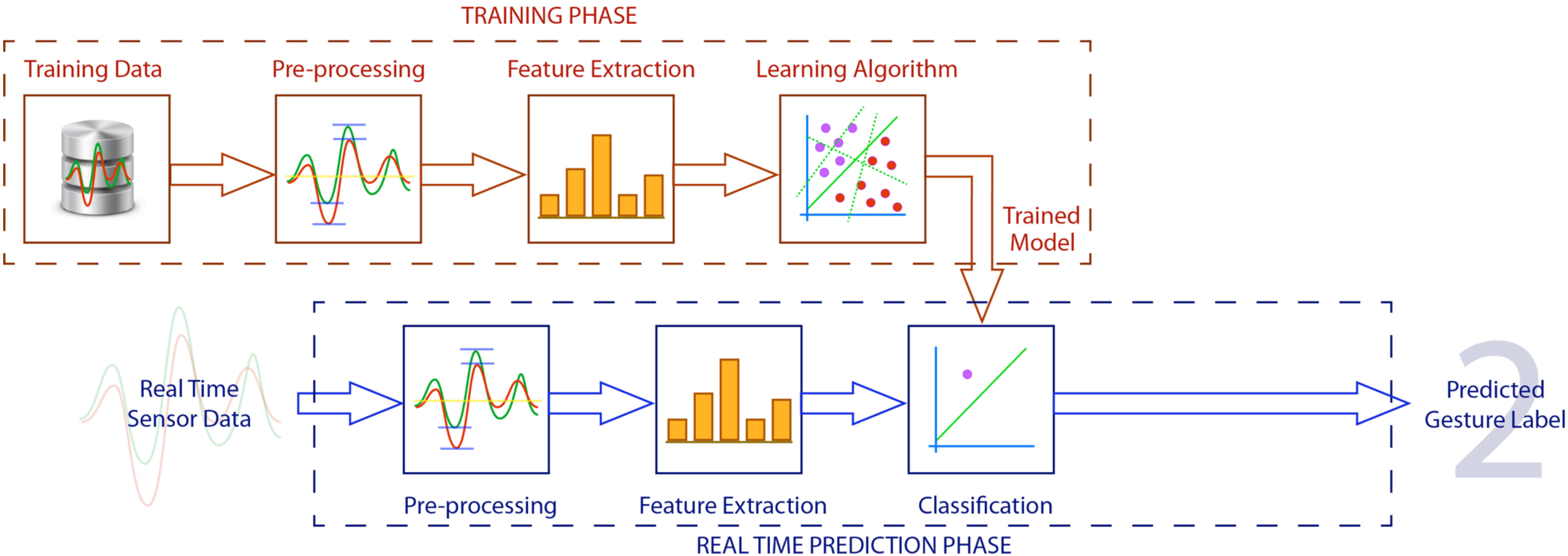


# Gesture Recognition



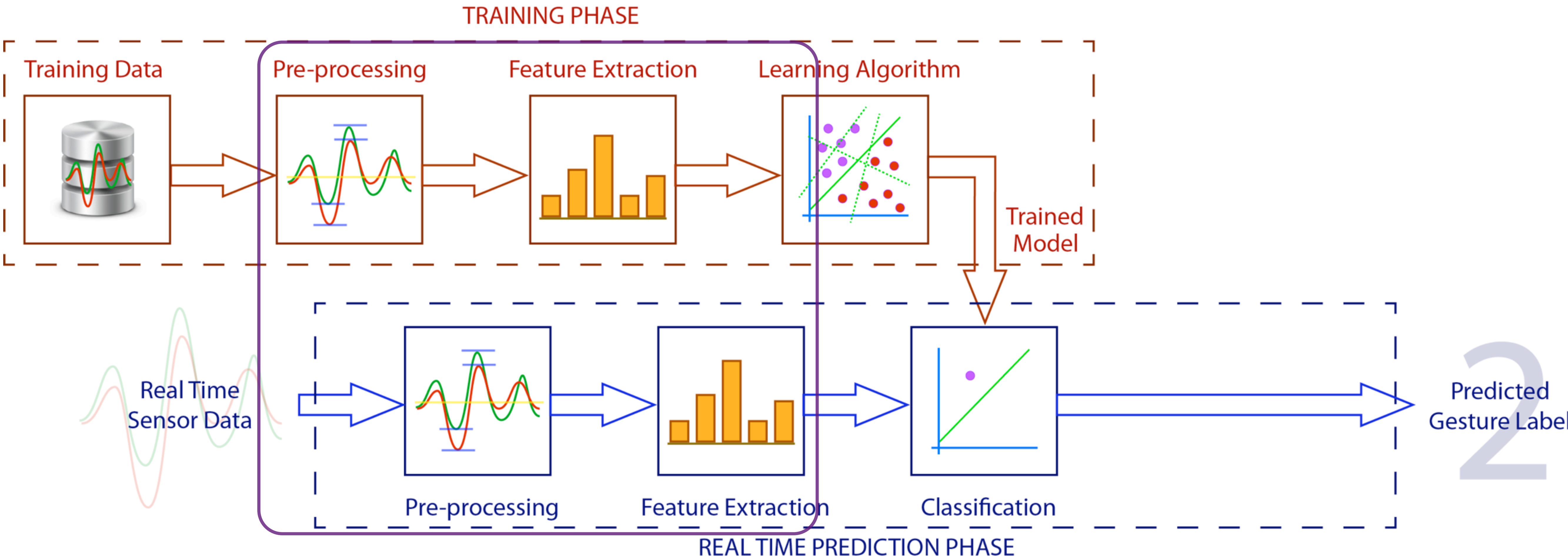

Instead of using the raw data as input to the learning algorithm, we might want to pre-process the data (i.e. scale it, smooth it) and also compute some features from the data which make the classification task easier for the machine-learning algorithm

# Gesture Recognition



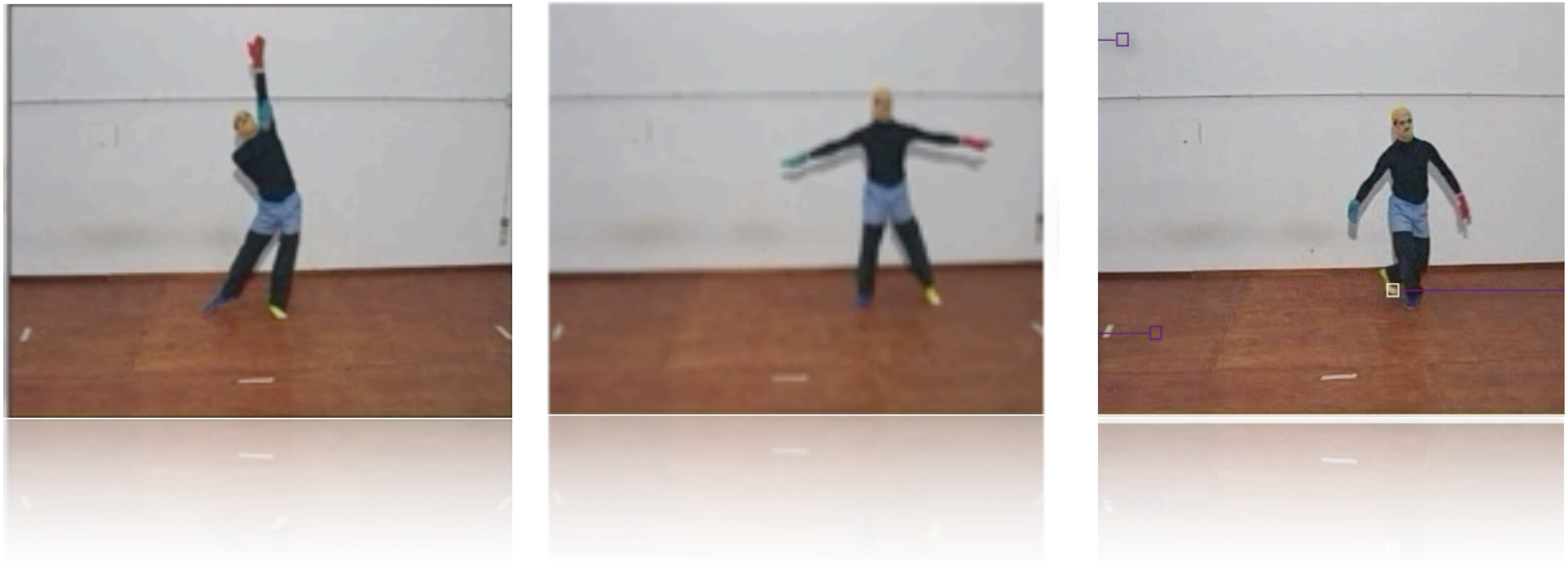

Important that we also use the same pre-processing and feature extraction methods when predicting the new data!

# Gesture Recognition




Important that we also use the same pre-processing and feature extraction methods when predicting the new data!

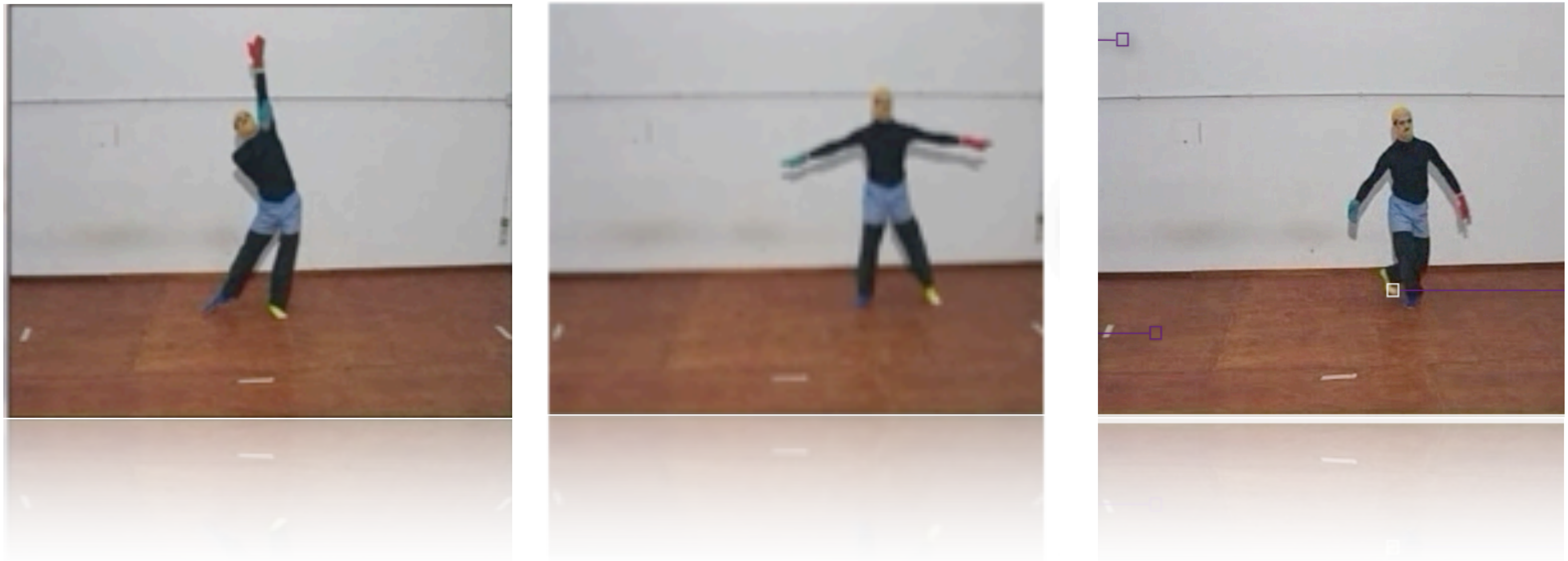
# Gesture Recognition



Classification Task:

Recognize different postures of a dancer

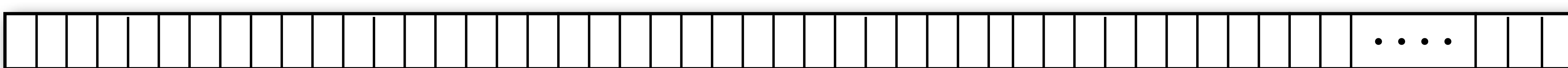
# Gesture Recognition



## Classification Task:

Recognize different postures of a dancer

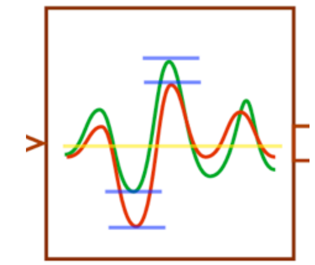
## Input Vector:



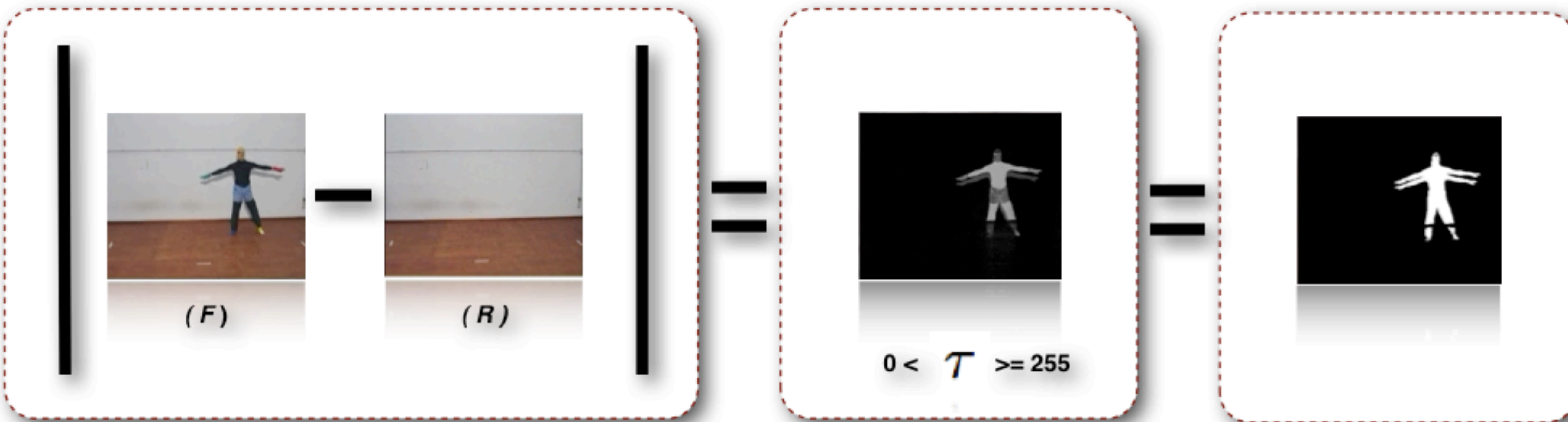
$$640 * 480 * 3 = 921600$$

# Gesture Recognition

Pre-processing



## Preprocessing: Background Subtraction

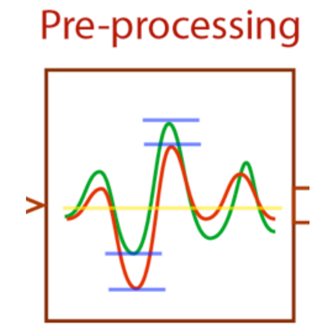



**Absolute Subtraction of new frame ( *F* ) from reference frame ( *R* )**

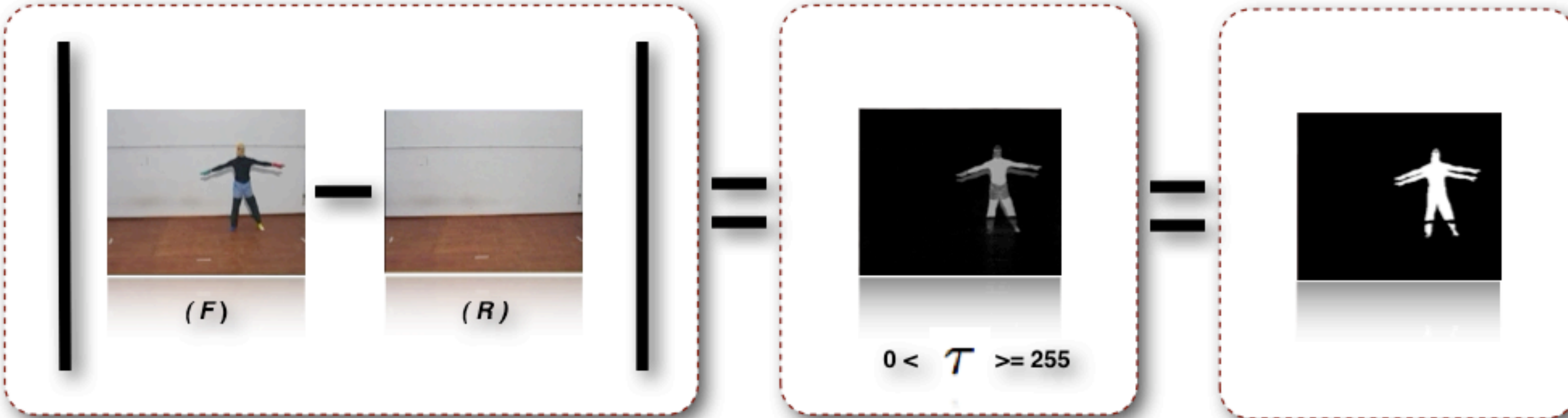
**Thresholding: any value less than threshold is set to black, any value above threshold is set to white**

**New Background Subtracted Frame after thresholding**

# Gesture Recognition




## Preprocessing: Background Subtraction

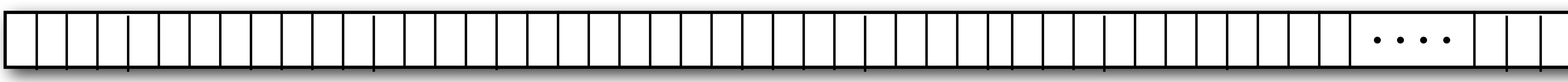



Absolute Subtraction of new frame ( *F* ) from reference frame ( *R* )

Thresholding: any value less than threshold is set to black, any value above threshold is set to white

New Background Subtracted Frame after thresholding

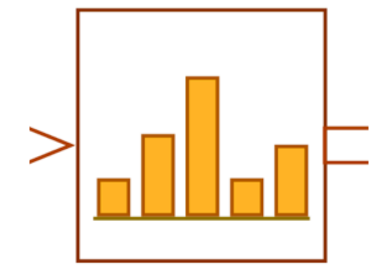
## Input Vector:



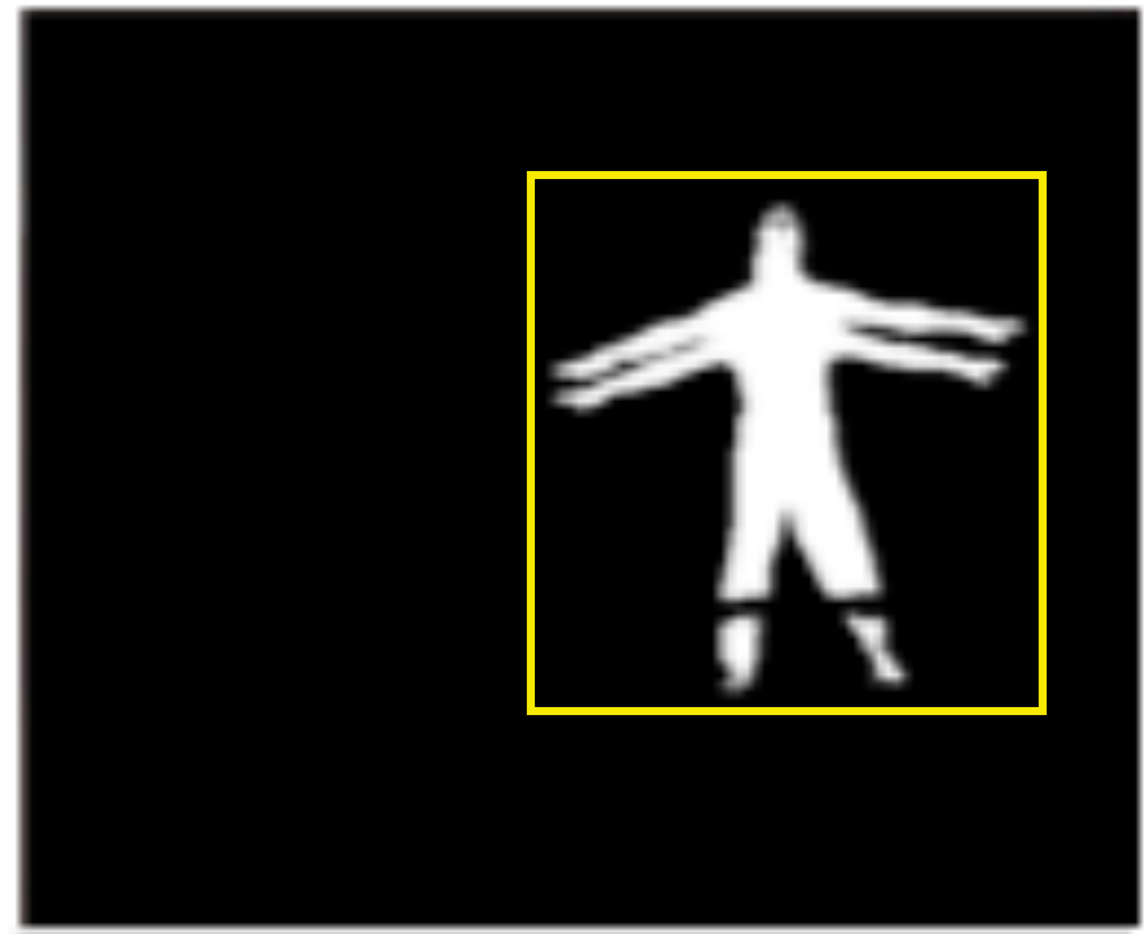
$$640 * 480 = 307200$$

# Gesture Recognition

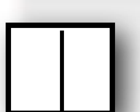
Feature Extraction



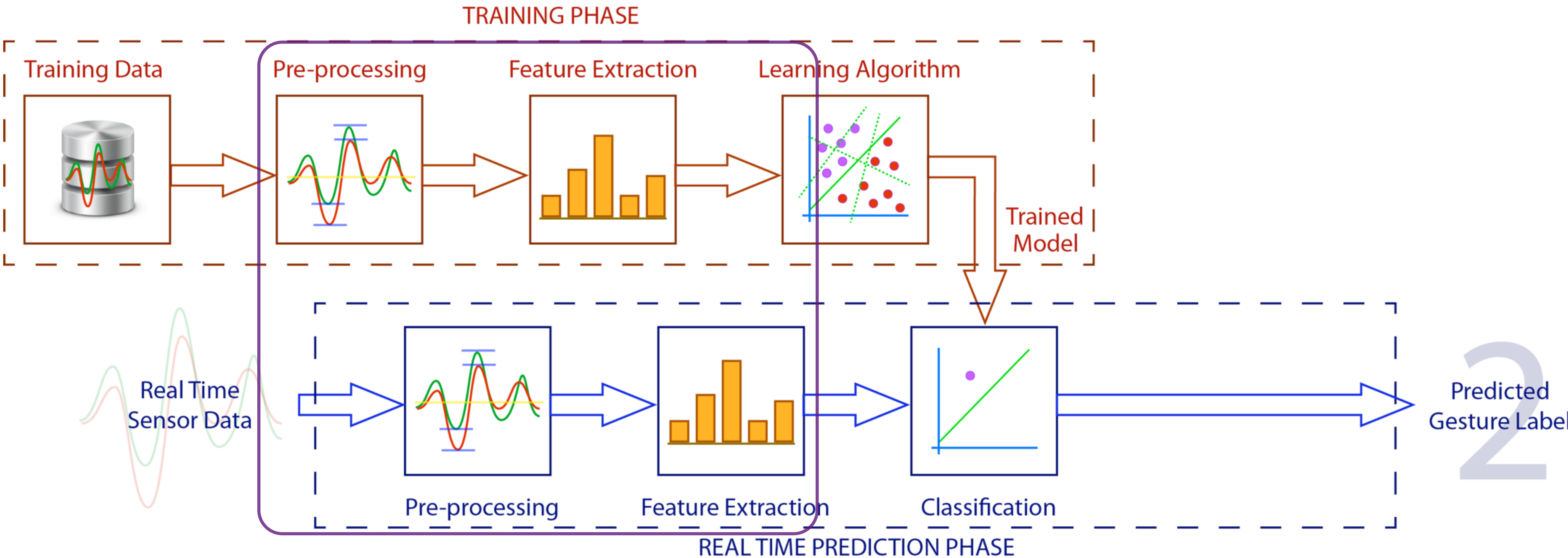
## Feature Extraction: Bounding Box



Input Vector:

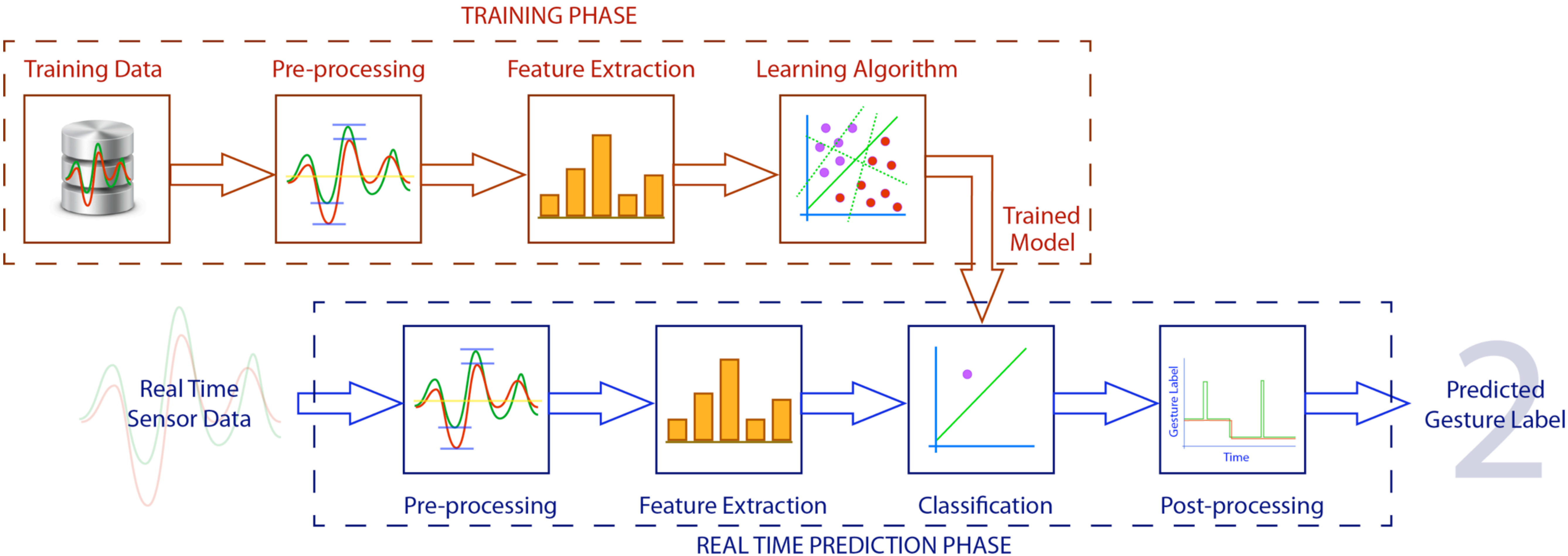
 = 2

# Gesture Recognition



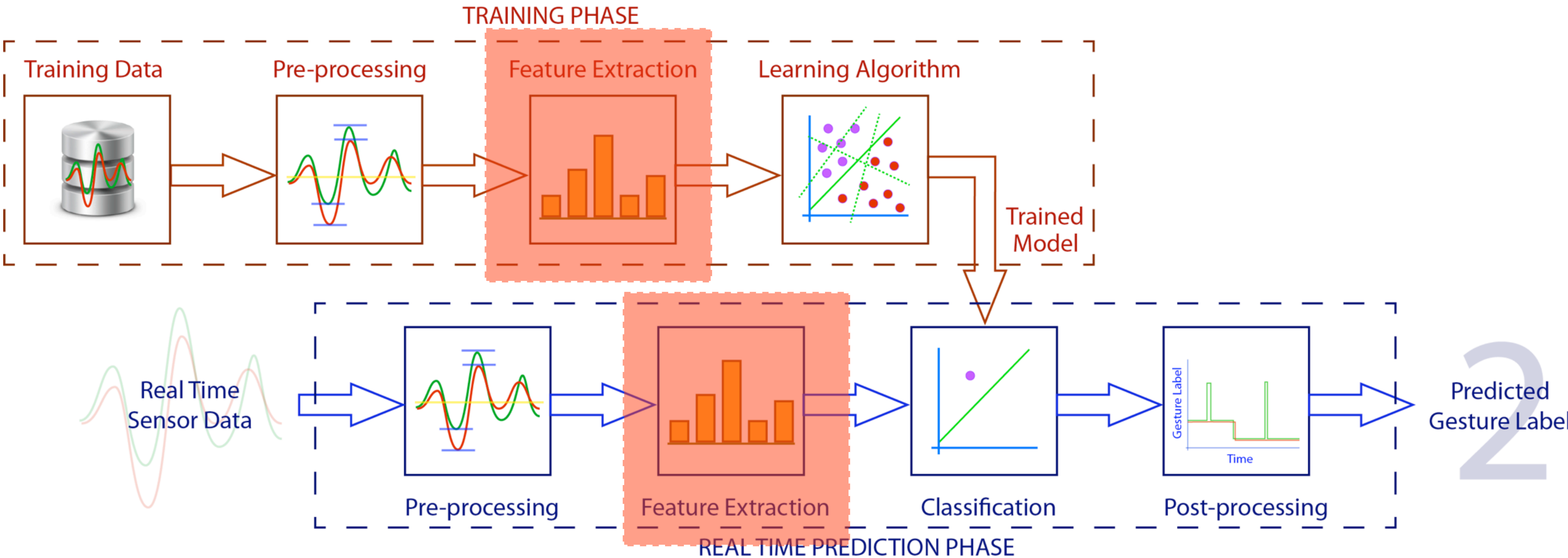

Important that we also use the same pre-processing and feature extraction methods when predicting the new data!

# Gesture Recognition




As well as pre-processing the input to the classification algorithm, we might also want to process the output of the classifier

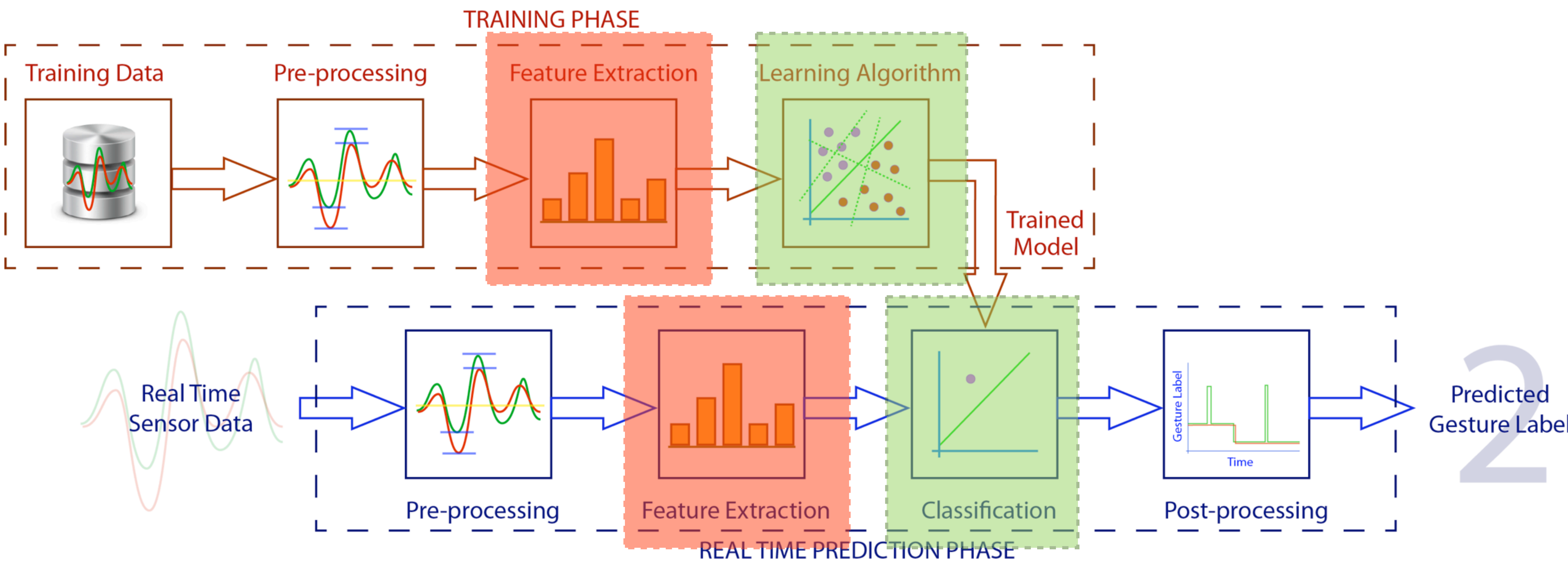
# Gesture Recognition




2

Choosing the right features is REALLY IMPORTANT!

# Gesture Recognition




Choosing the right features is REALLY IMPORTANT!

Choosing the right ML algorithm is also REALLY IMPORTANT!

# Gesture Recognition

Choosing the right algorithm to solve **your** problem:

# Gesture Recognition

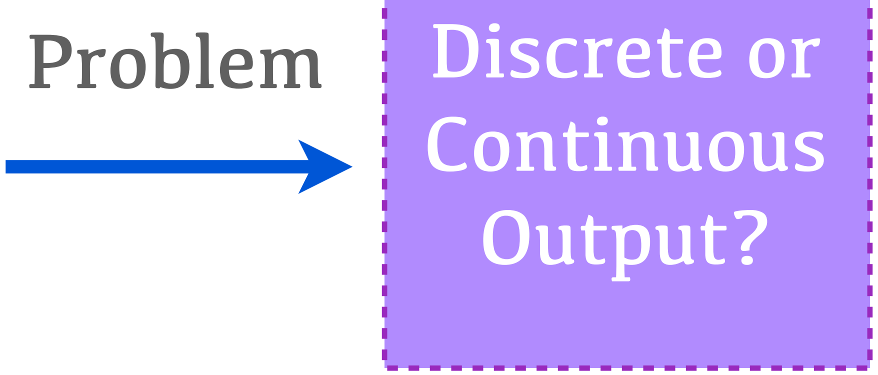
Choosing the right algorithm to solve **your** problem:

First you need to categorize **your** problem:

# Gesture Recognition

Choosing the right algorithm to solve **your** problem:

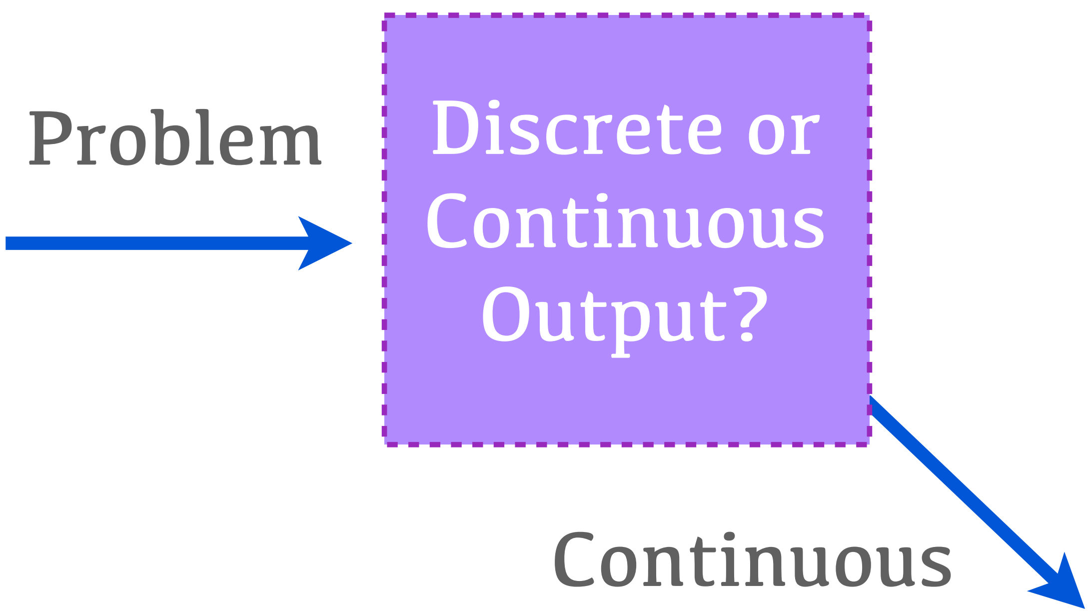
First you need to categorize **your** problem:



# Gesture Recognition

Choosing the right algorithm to solve **your** problem:

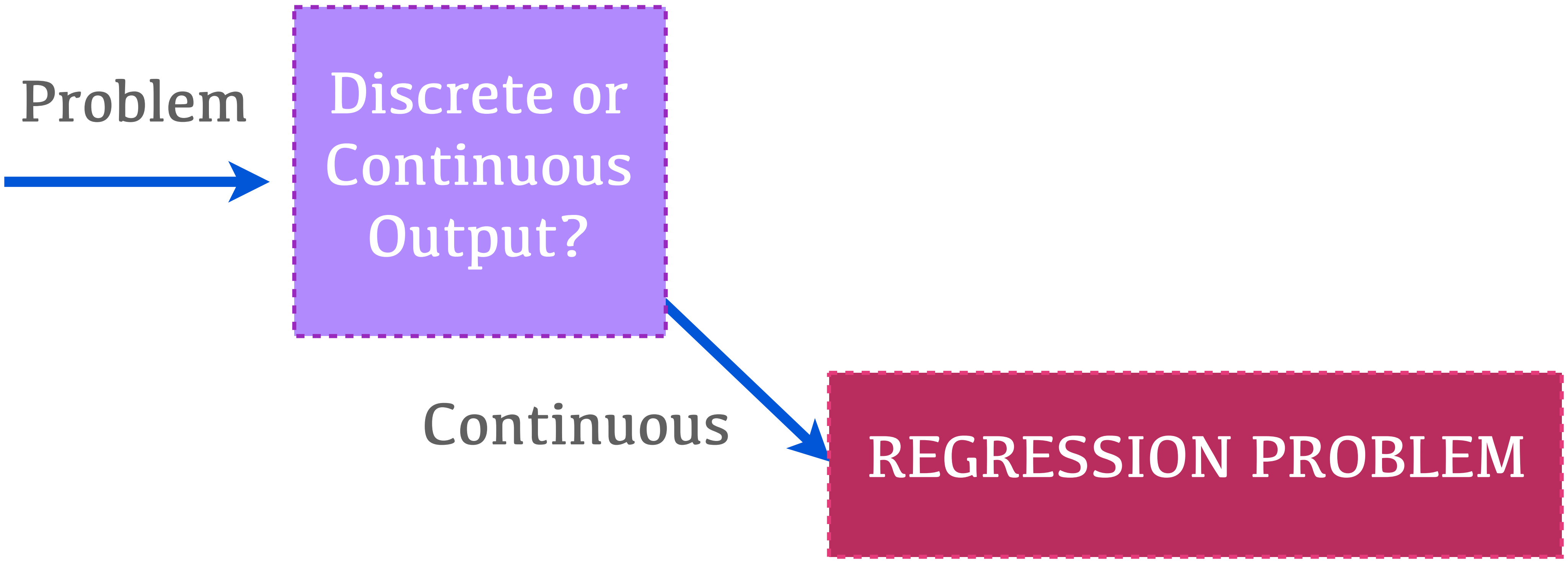
First you need to categorize **your** problem:



# Gesture Recognition

Choosing the right algorithm to solve **your** problem:

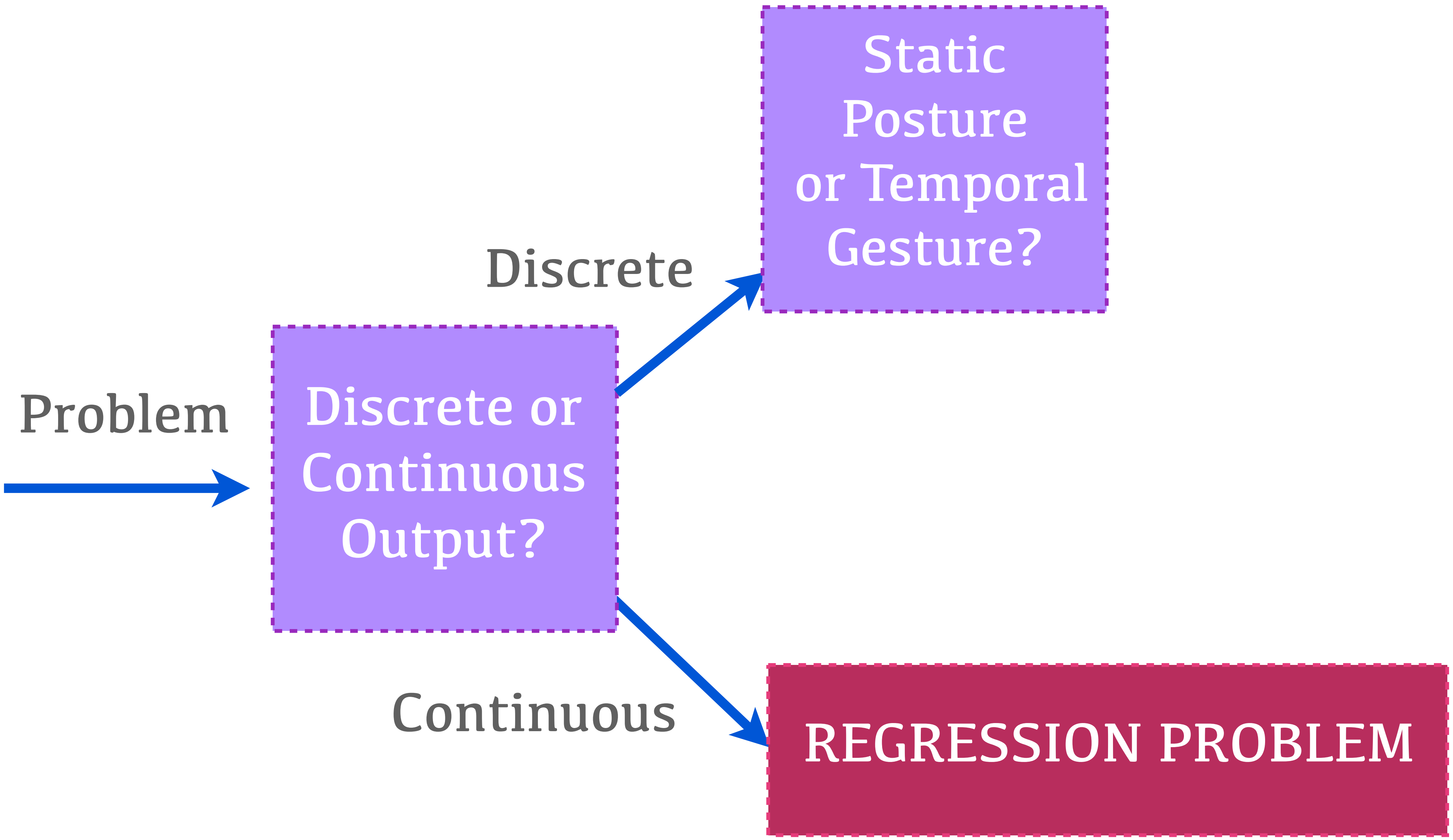
First you need to categorize **your** problem:



# Gesture Recognition

Choosing the right algorithm to solve **your** problem:

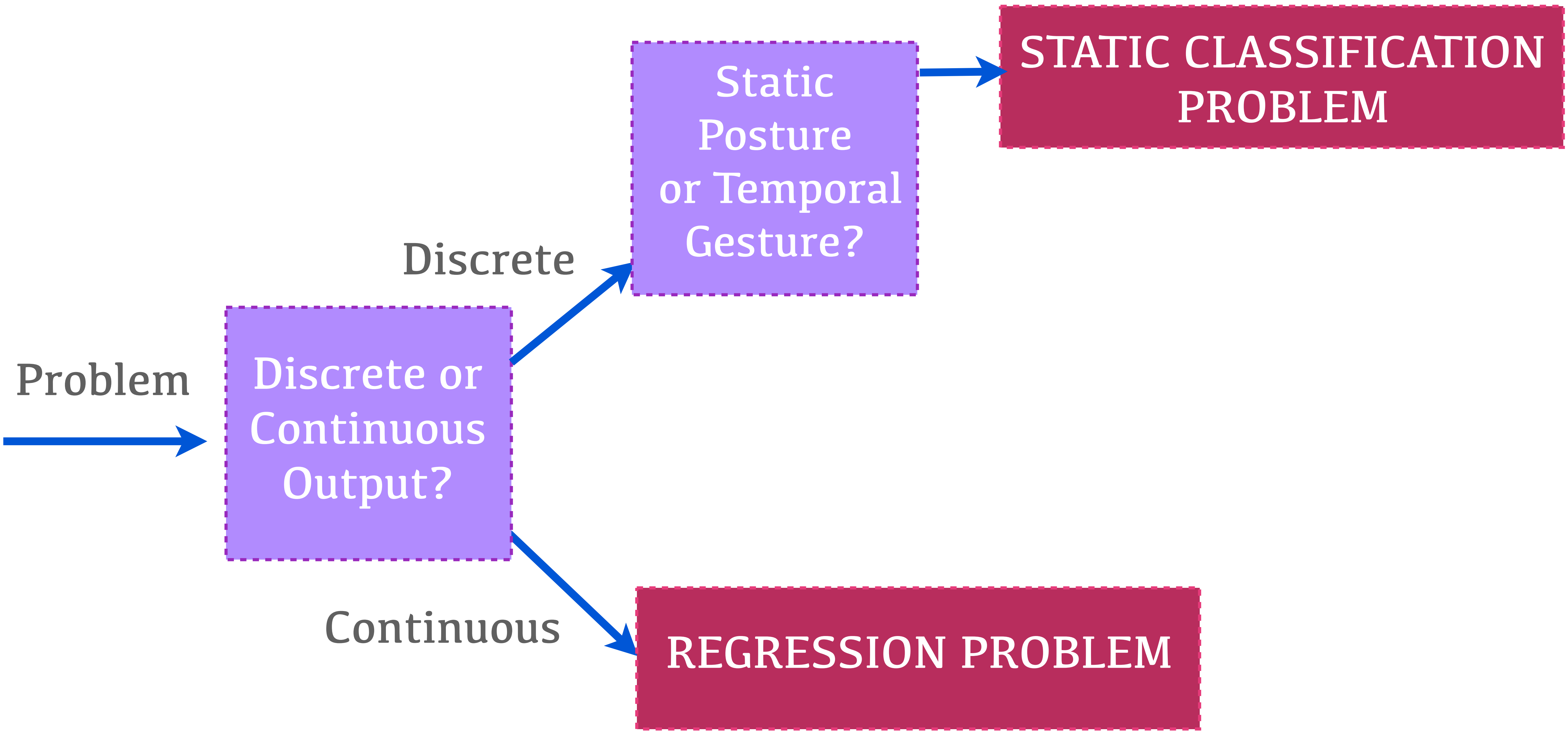
First you need to categorize **your** problem:



# Gesture Recognition

Choosing the right algorithm to solve **your** problem:

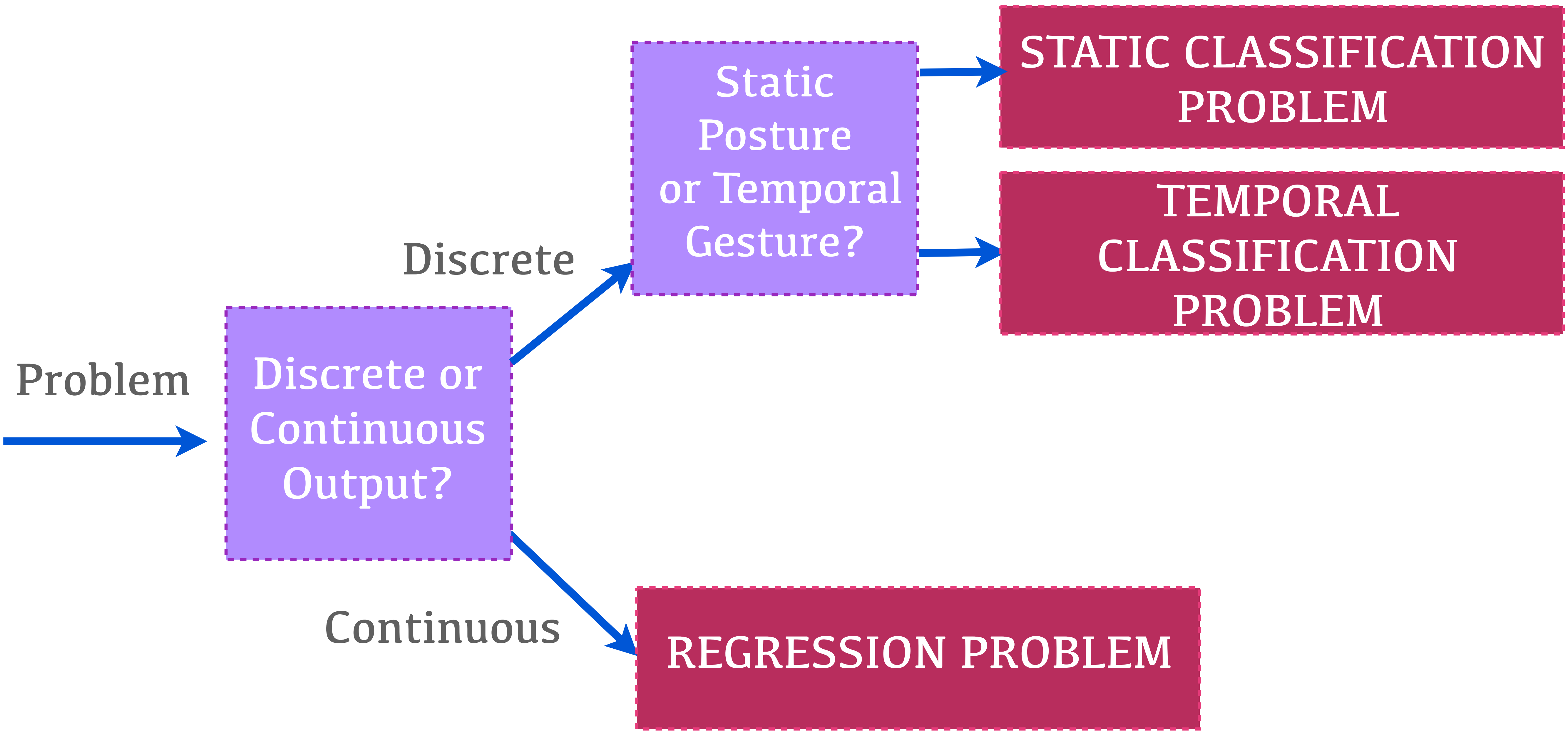
First you need to categorize **your** problem:



# Gesture Recognition

Choosing the right algorithm to solve **your** problem:

First you need to categorize **your** problem:



# Gesture Recognition

Choosing the right algorithm to solve **your** problem:

STATIC CLASSIFICATION PROBLEM

TEMPORAL CLASSIFICATION PROBLEM

PROBLEM

# Gesture Recognition

Choosing the right algorithm to solve **your** problem:

STATIC CLASSIFICATION PROBLEM

Adaptive Naive Bayes Classifier (ANBC)

TEMPORAL CLASSIFICATION PROBLEM

PROBLEM

# Gesture Recognition

Choosing the right algorithm to solve **your** problem:

STATIC CLASSIFICATION PROBLEM

- Adaptive Naive Bayes Classifier (ANBC)
- K-Nearest Neighbor (KNN)

TEMPORAL CLASSIFICATION PROBLEM

PROBLEM

# Gesture Recognition

Choosing the right algorithm to solve **your** problem:

STATIC CLASSIFICATION PROBLEM

- Adaptive Naive Bayes Classifier (ANBC)
- K-Nearest Neighbor (KNN)
- AdaBoost

TEMPORAL CLASSIFICATION PROBLEM

PROBLEM

# Gesture Recognition

Choosing the right algorithm to solve **your** problem:

STATIC CLASSIFICATION PROBLEM

TEMPORAL CLASSIFICATION PROBLEM

- Adaptive Naive Bayes Classifier (ANBC)
- K-Nearest Neighbor (KNN)
- AdaBoost
- Decision Trees

PROBLEM

# Gesture Recognition

Choosing the right algorithm to solve **your** problem:

STATIC CLASSIFICATION PROBLEM

TEMPORAL CLASSIFICATION PROBLEM

- Adaptive Naive Bayes Classifier (ANBC)
- K-Nearest Neighbor (KNN)
- AdaBoost
- Decision Trees
- Support Vector Machine (SVM)

PROBLEM

# Gesture Recognition

Choosing the right algorithm to solve **your** problem:

STATIC CLASSIFICATION PROBLEM

TEMPORAL CLASSIFICATION PROBLEM

PROBLEM

# Gesture Recognition

Choosing the right algorithm to solve **your** problem:

STATIC CLASSIFICATION PROBLEM

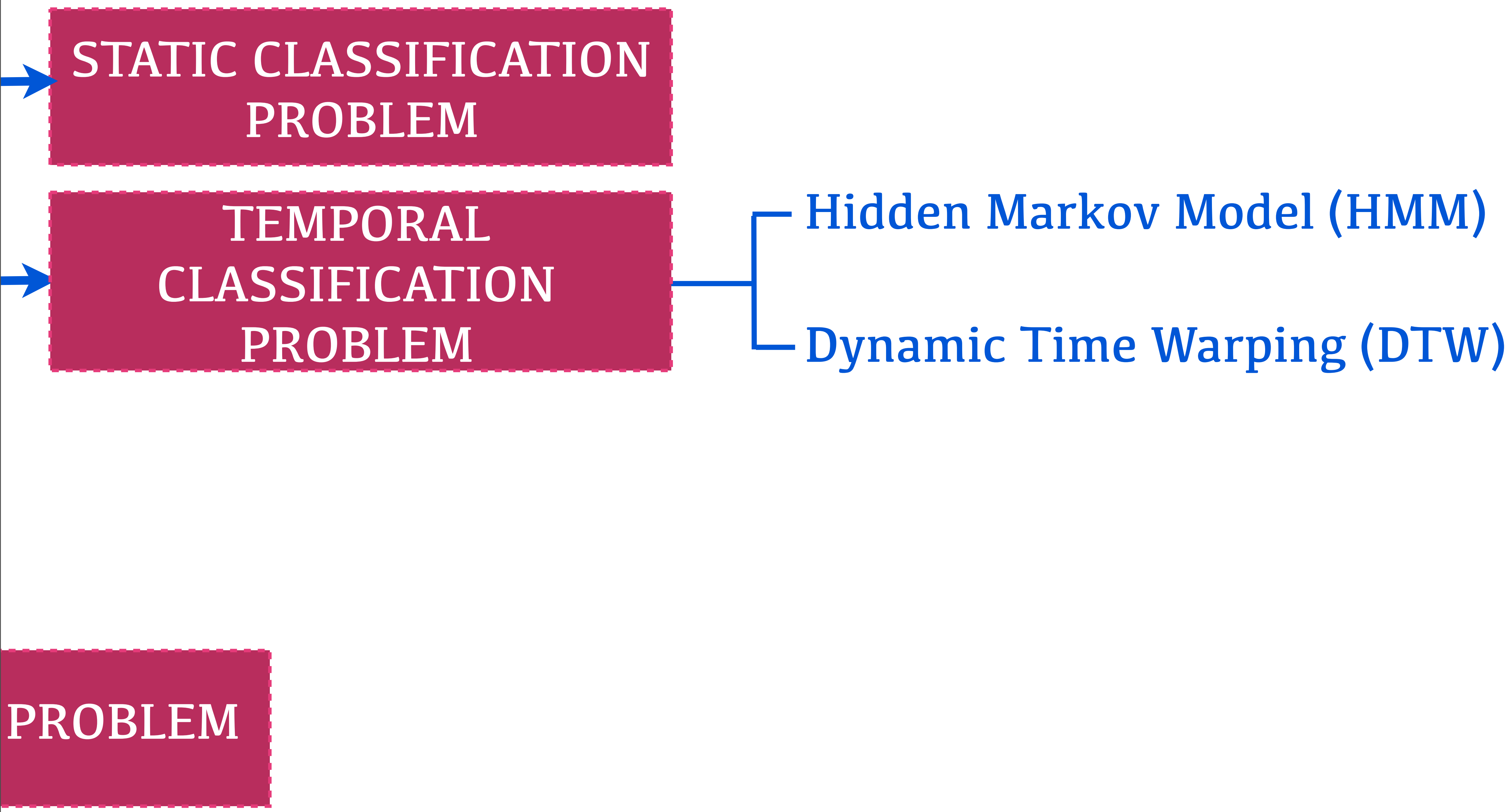
TEMPORAL CLASSIFICATION PROBLEM

Hidden Markov Model (HMM)

PROBLEM

# Gesture Recognition

Choosing the right algorithm to solve **your** problem:



# Gesture Recognition

Choosing the right algorithm to solve **your** problem:

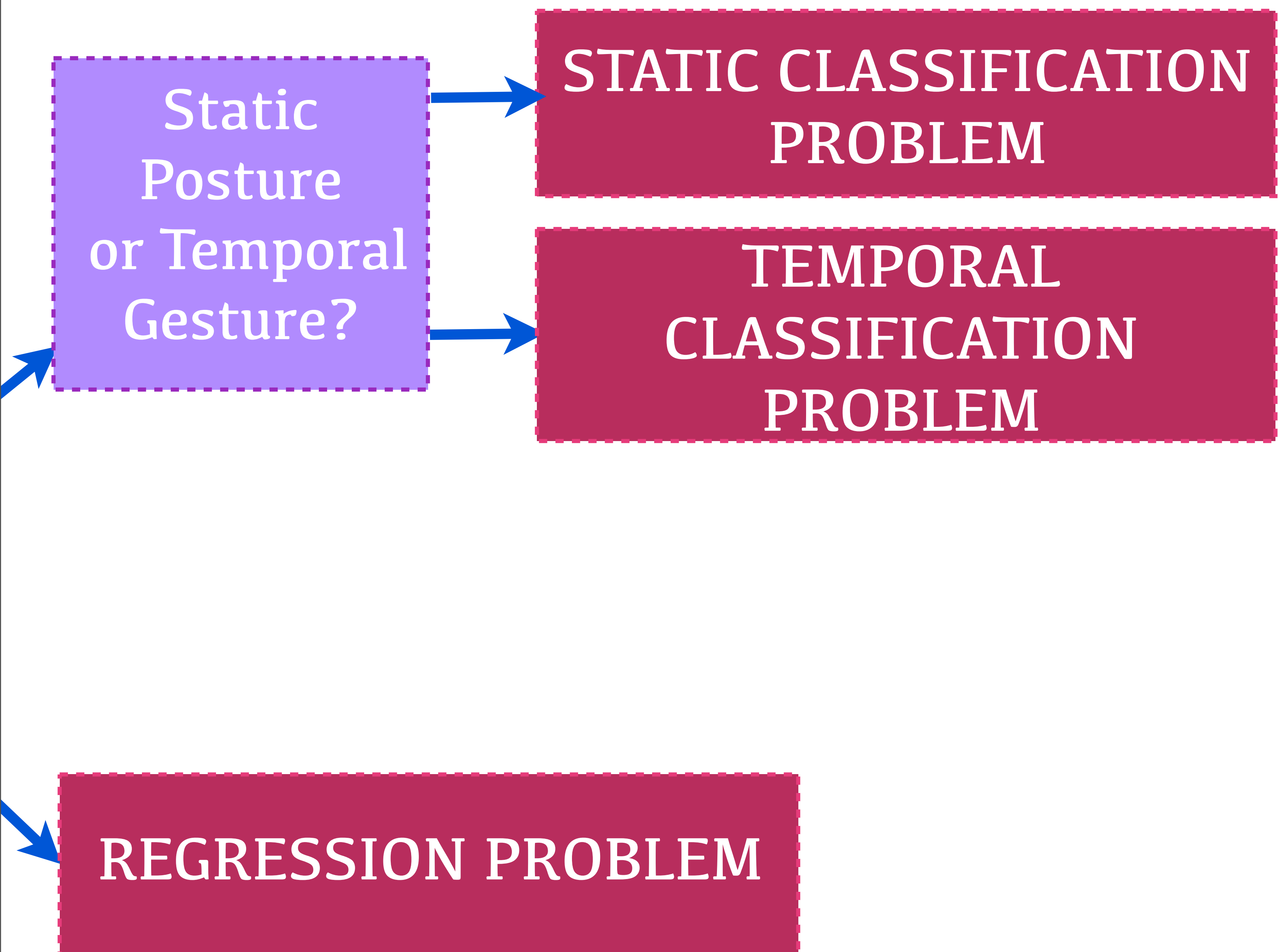
STATIC CLASSIFICATION PROBLEM

TEMPORAL CLASSIFICATION PROBLEM

PROBLEM

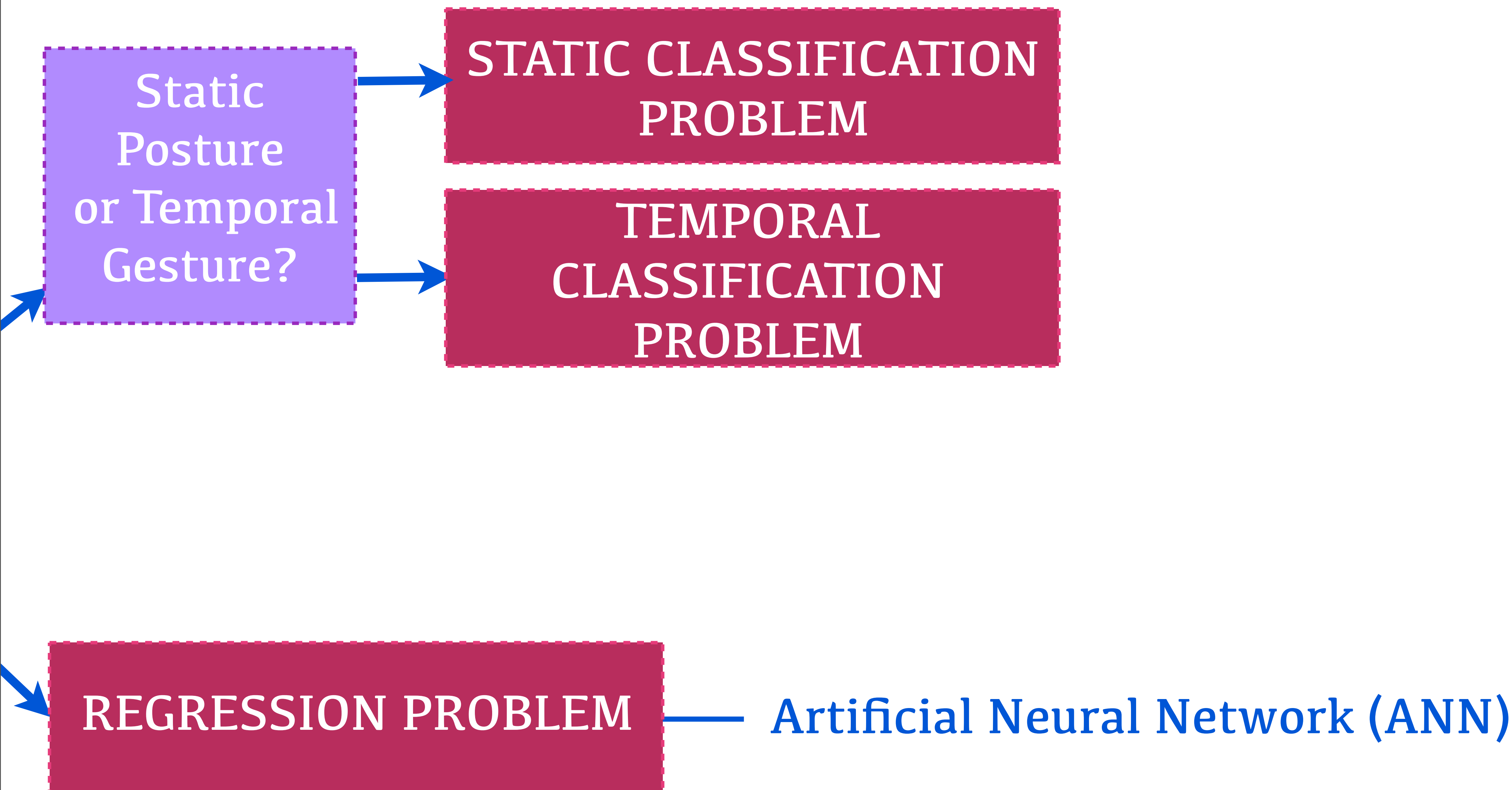
# Gesture Recognition

Choosing the right algorithm to solve **your** problem:



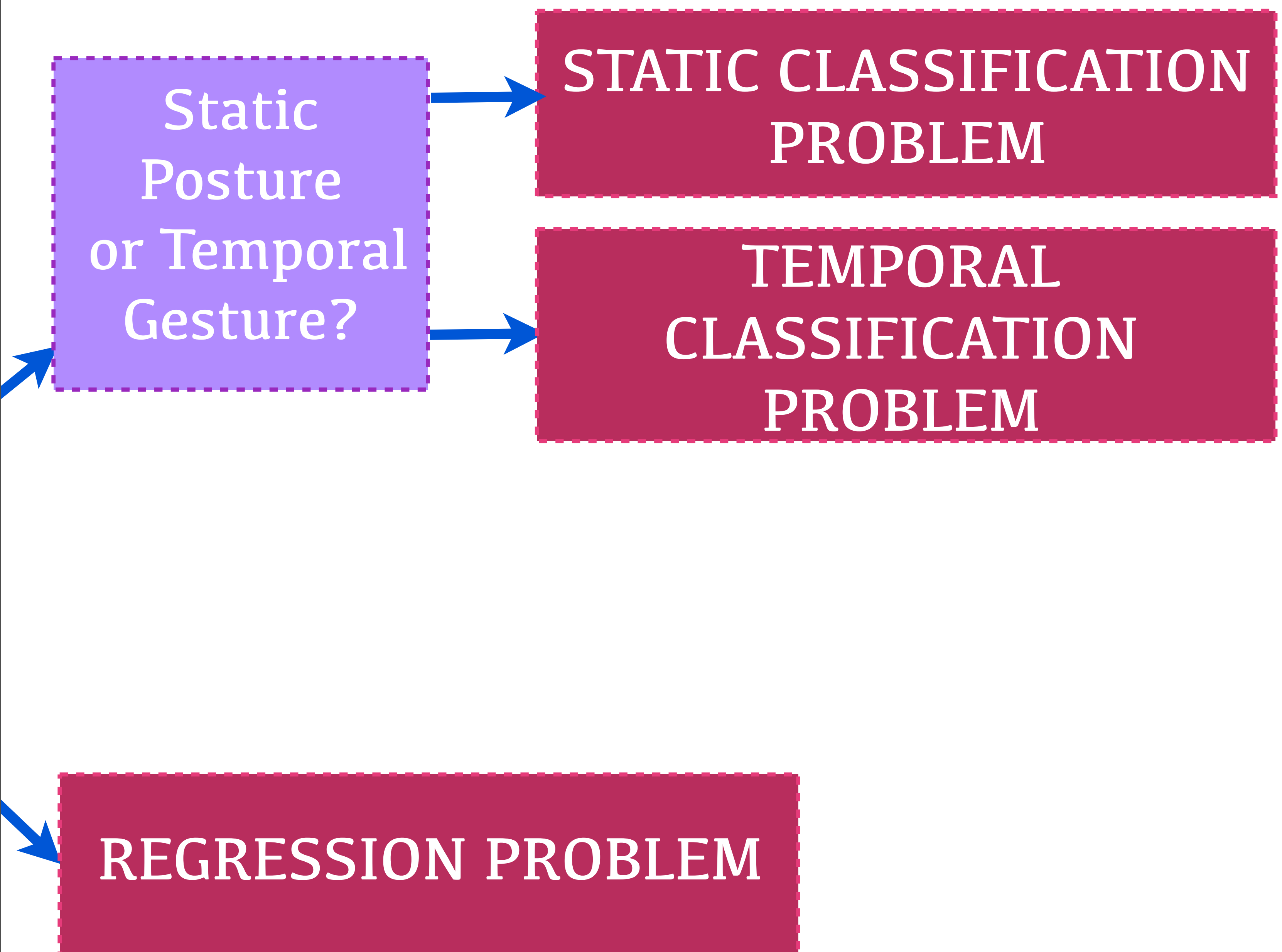
# Gesture Recognition

Choosing the right algorithm to solve **your** problem:



# Gesture Recognition

Choosing the right algorithm to solve **your** problem:



# Gesture Recognition

Choosing the right algorithm to solve **your** problem:

# Machine Learning Resources

- Great books to get started:

Marsland (2009): Machine Learning: An Algorithmic Perspective

Witten (2011): Data Mining: Practical Machine Learning Tools and Techniques

- More detailed books:

Bishop (2007): Pattern Recognition and Machine Learning

Duda (2001): Pattern Classification

- Online Lectures:

Prof. Andrew Ng (Stanford University), Machine Learning Lectures
(search for Machine Learning (Stanford) in youtube)

# Gesture Recognition Toolkit

# Gesture Recognition Toolkit

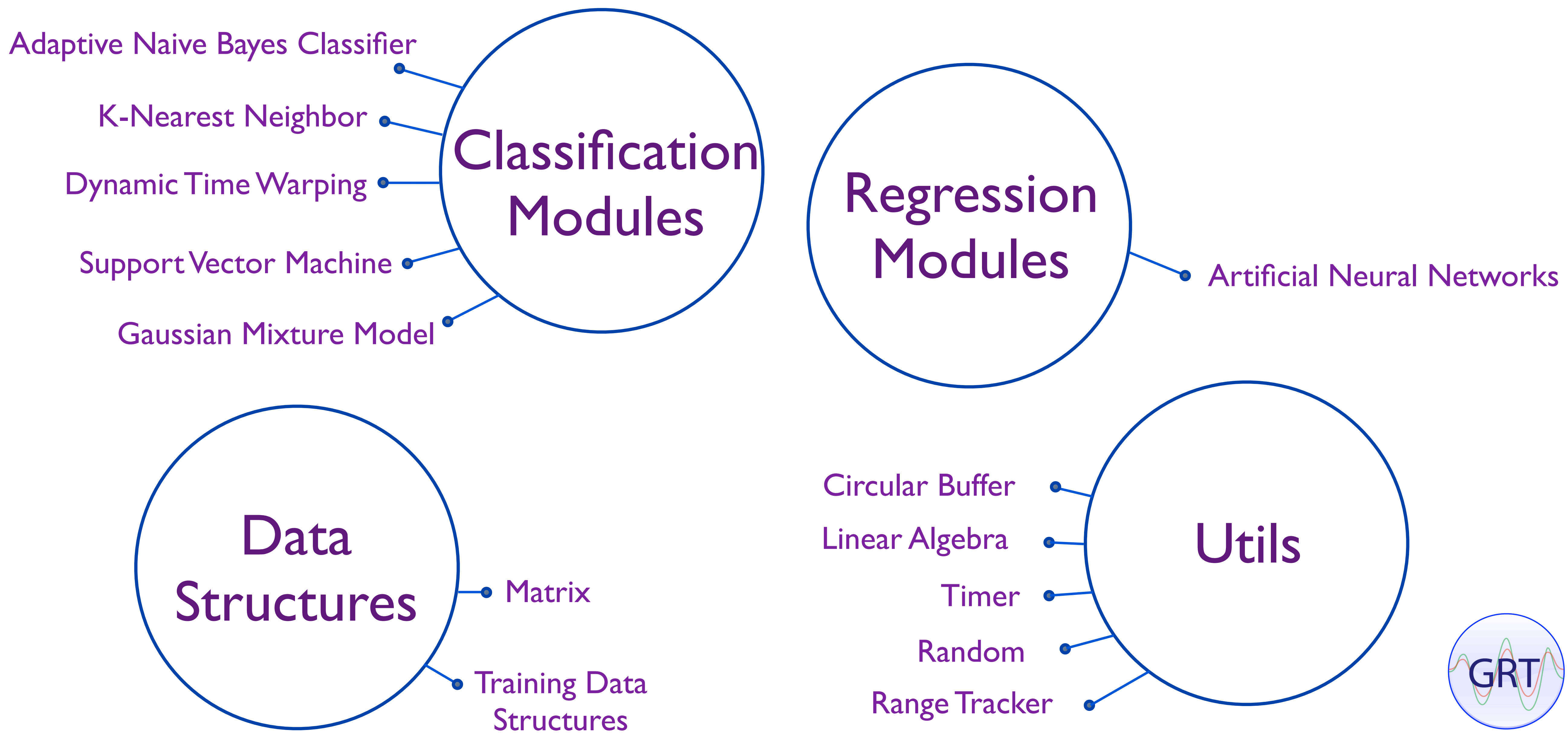
**Classification Modules**

- Adaptive Naive Bayes Classifier
- K-Nearest Neighbor
- Dynamic Time Warping
- Support Vector Machine
- Gaussian Mixture Model

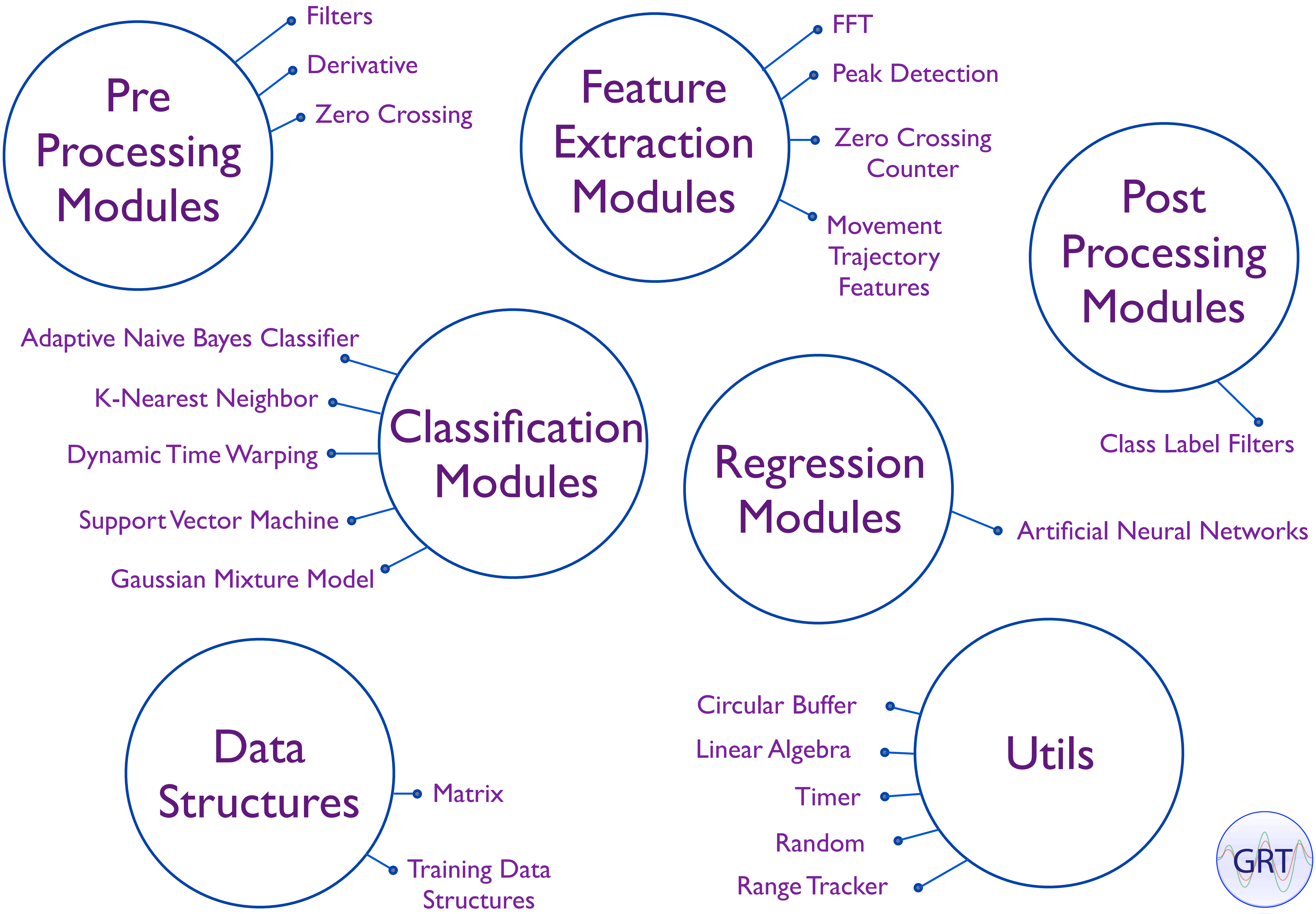
**Regression Modules**

- Artificial Neural Networks

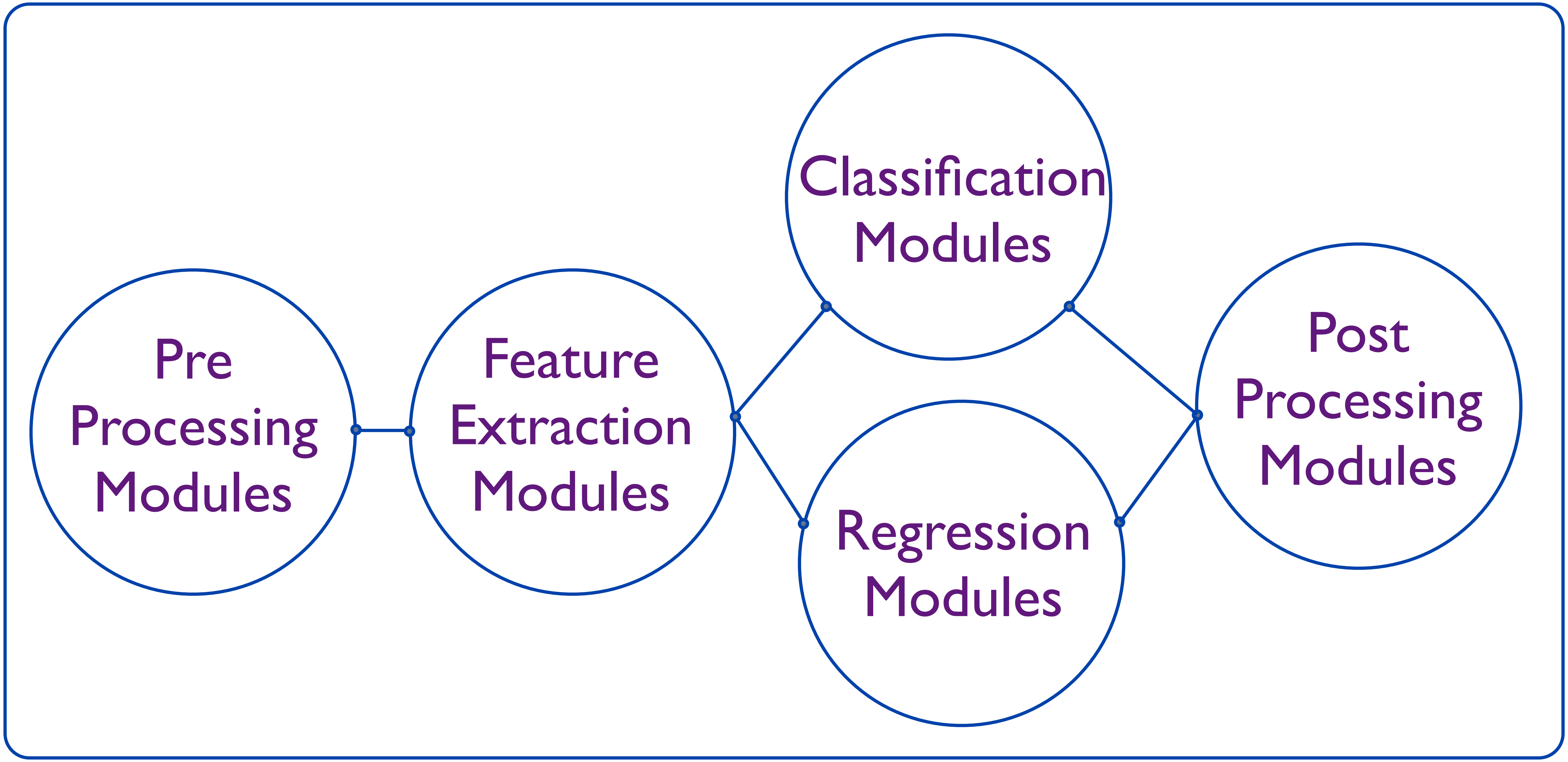
# Gesture Recognition Toolkit

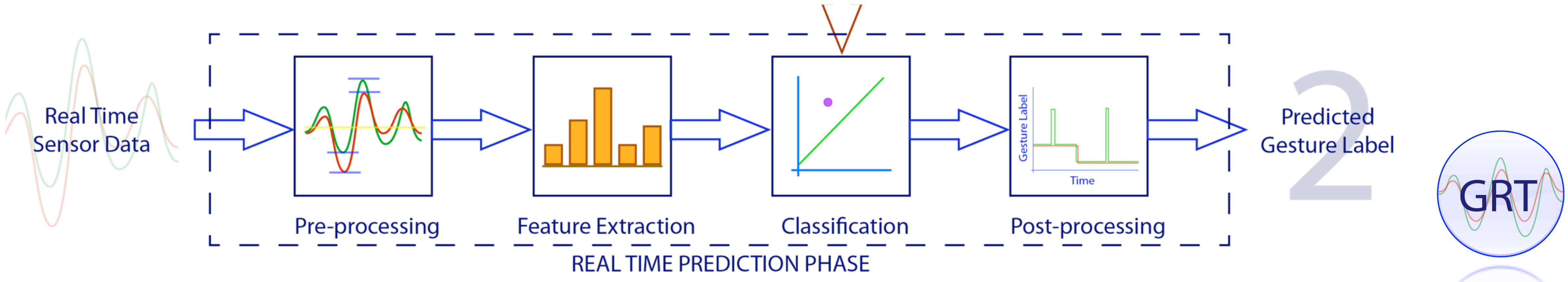


# Gesture Recognition Toolkit

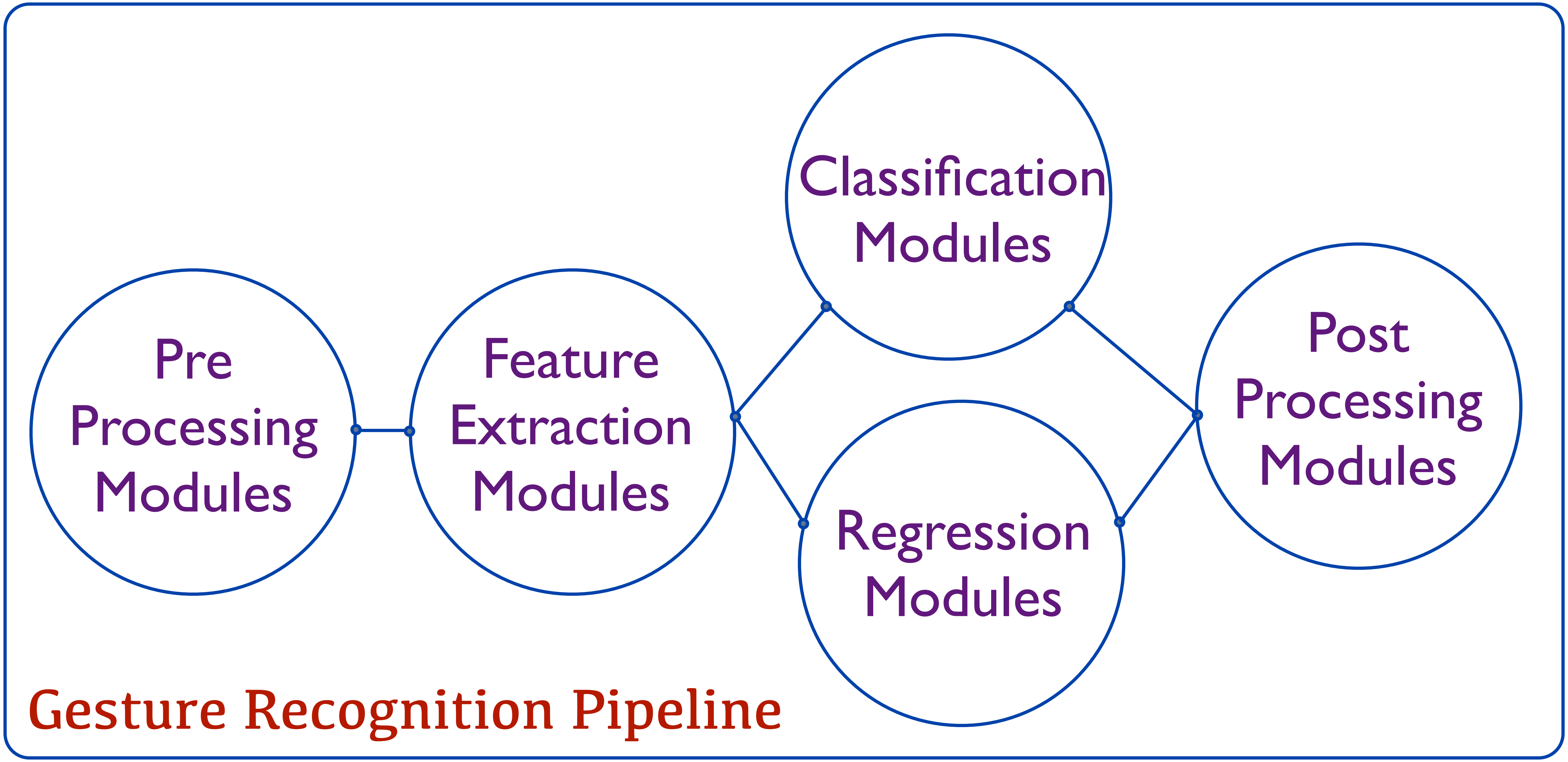


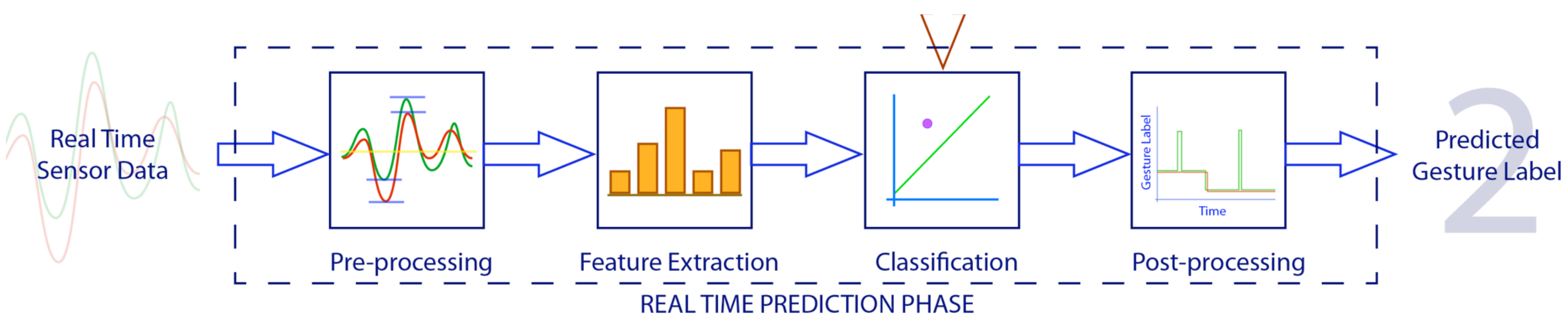
# Gesture Recognition Toolkit



# Gesture Recognition Toolkit



# Gesture Recognition Toolkit

```
//Create a new GestureRecognitionPipeline
GestureRecognitionPipeline pipeline;

//Set the classifier at the core of the pipeline
pipeline.setClassifier( ANBC() );
```

This is how you setup a new pipeline and set the classifier

GRT

# Gesture Recognition Toolkit

```
//Create a new GestureRecognitionPipeline
GestureRecognitionPipeline pipeline;

//Set the classifier at the core of the pipeline
pipeline.setClassifier( ANBC() );
```

```
//Set the classifier at the core of the pipeline
pipeline.setClassifier( SVM() );
```

This is how you would change the classifier

# Gesture Recognition Toolkit

```
//Create a new GestureRecognitionPipeline
GestureRecognitionPipeline pipeline;

//Add a moving average filter as a pre-processing module
//With a buffer size of 5 and for a 1 dimensional signal
pipeline.addPreProcessingModule( MovingAverageFilter(5,1) );

//Add an FFT as a feature-extraction module
pipeline.addFeatureExtractionModule( FFT(1024,1) );

//Add a custom feature module to the pipeline
pipeline.addFeatureExtractionModule( MyOwnFeatureMethod() );

//Set the classifier at the core of the pipeline
pipeline.setClassifier( ANBC() );

//Add a class label timeout filter to the end of the pipeline
pipeline.addPostProcessingModule( ClassLabelTimeoutFilter(1000) );
```

This is how you setup a more complex pipeline

# Gesture Recognition Toolkit

```
//Train the pipeline
bool trainSuccess = pipeline.train( trainingData );
```

This is how you train the algorithm at the core of the pipeline

# Gesture Recognition Toolkit

```
//Perform the prediction
bool testSuccess = pipeline.test( testData );
```

This is how you test the accuracy of the pipeline

# Gesture Recognition Toolkit

```
//Perform the prediction
bool testSuccess = pipeline.test( testData );

//Get the test accuracy
double accuracy = pipeline.getTestAccuracy();

//Get the F-Measure, Precision and Recall for gesture 1
double fMeasure = pipeline.getTestFMeasure( 1 );
double precision = pipeline.getTestPrecision( 1 );
double recall = pipeline.getTestRecall( 1 );
```

You can then easily access the accuracy, precision, recall, etc.

GRT

# Gesture Recognition Toolkit

```
//Perform the prediction
bool trainSuccess = pipeline.train( trainingData, 10 );

//Get then get the cross validation accuracy
double accuracy = pipeline.getCrossValidationAccuracy();
```

If you want to run k-fold cross validation, then simply state the k-value when you call the train method and the pipeline will do the rest

# Gesture Recognition Toolkit

```
//Perform the prediction
bool predictionSuccess = pipeline.predict( inputVector );
```

This is how you perform real-time classification

GRT

# Gesture Recognition Toolkit

```
//Perform the prediction
bool predictionSuccess = pipeline.predict( inputVector );

//You can then get the predicted class label from the pipeline
UINT predictedClassLabel = pipeline.getPredictedClassLabel();

//Get the likelihood of the most likely class
double bestLoglikelihood = pipeline.getMaximumLikelihood();

//Get the likelihood of all the classes in the model
vector<double> classLikelihoods = pipeline.getClassLikelihoods();

//Use the predicted class label to trigger the action associated with that gesture
if( predictedClassLabel == 1 ){
    //Trigger the action associated with gesture 1
}
if( predictedClassLabel == 2 ){
    //Trigger the action associated with gesture 2
}
```

After the prediction you can then get the predicted class label, predication likelihoods, etc.

GRT