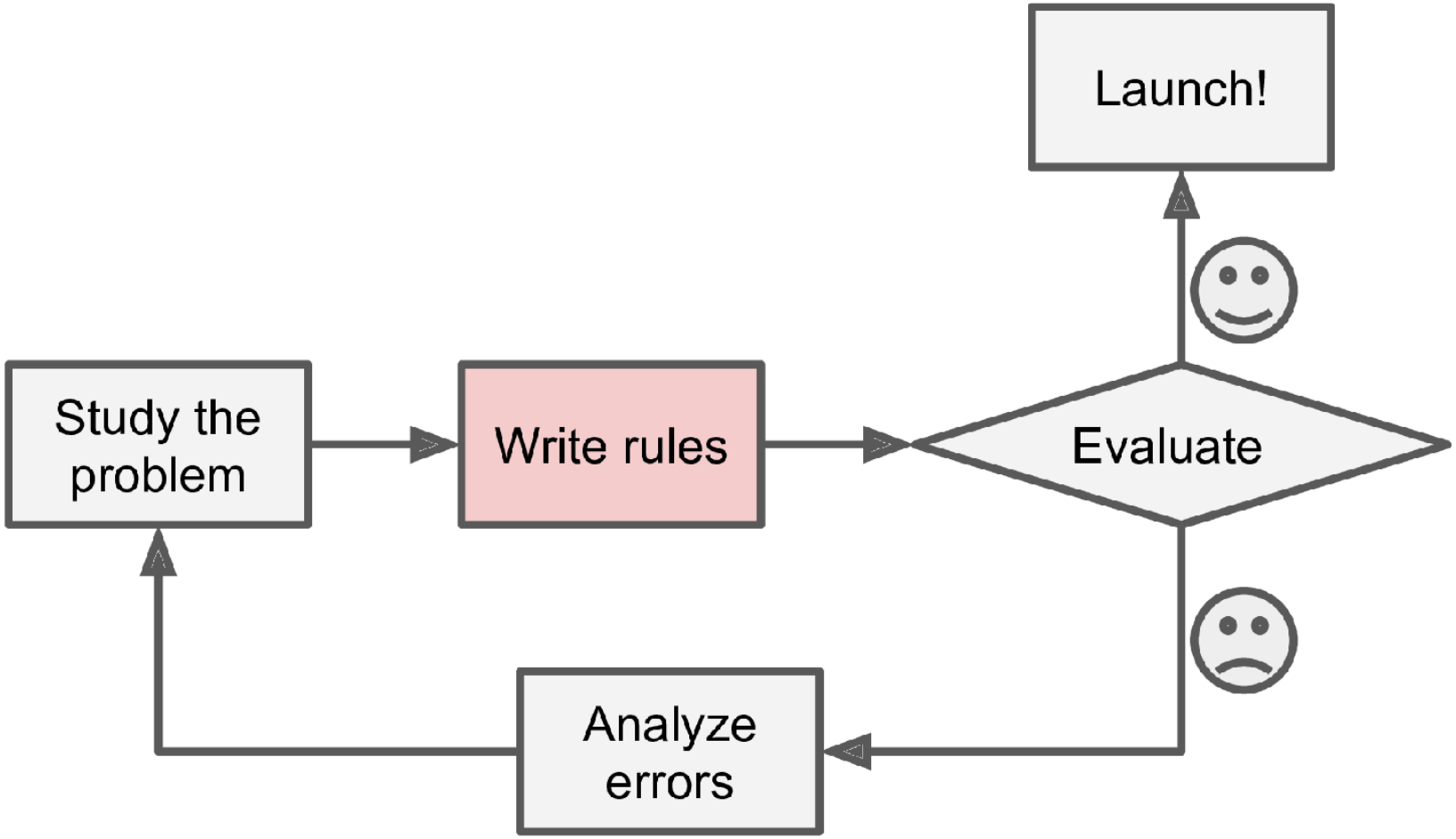
Machine Learning

Introduction

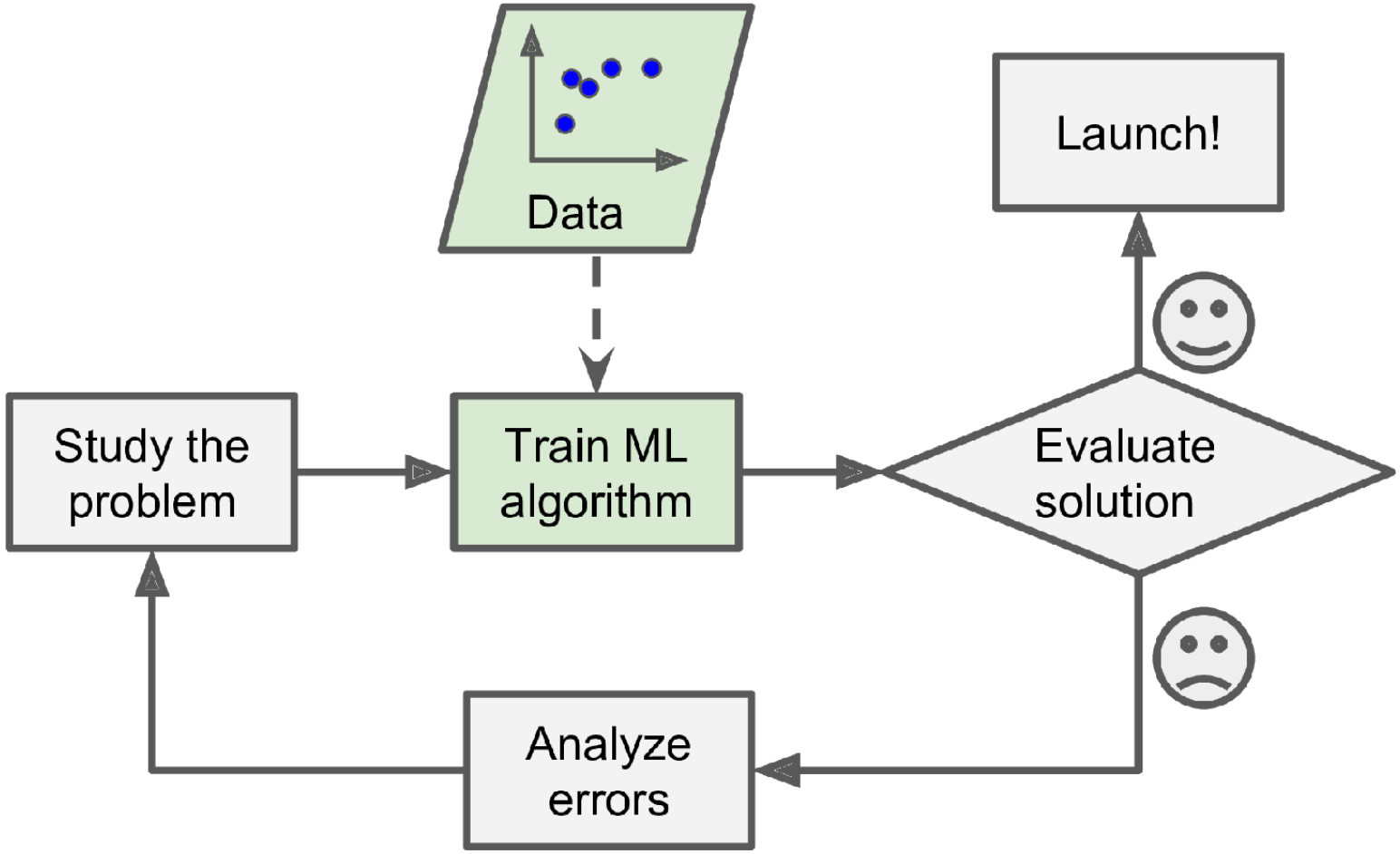
Programming computers so they can *learn from data*.

A field of study that gives computers the ability to learn without being explicitly programmed.

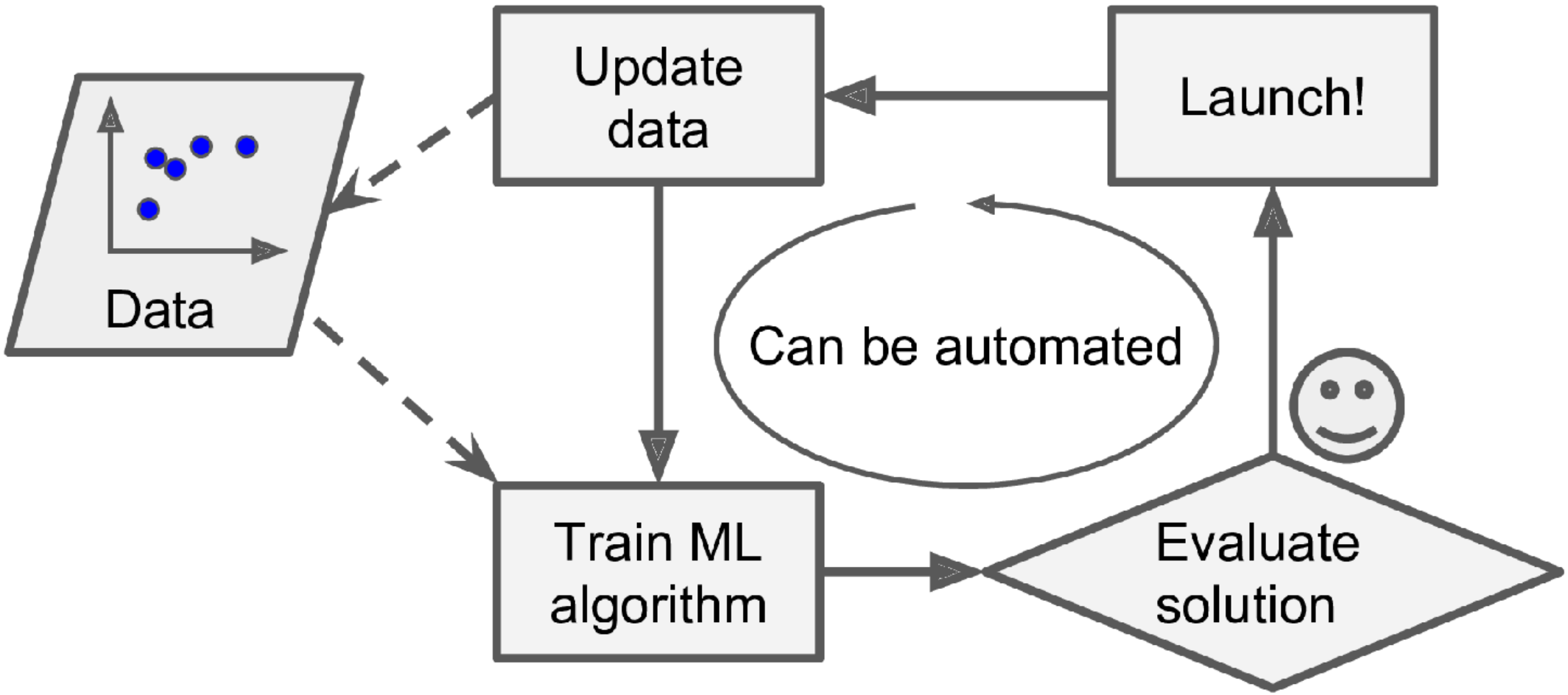
The traditional approach



Machine learning approach



Automatically adapting to change



Machine Learning is great for:

- Problems for which existing solutions require a **lot of fine-tuning** or **long lists of rules**: one Machine Learning algorithm can often simplify code and perform better than the traditional approach.
- Complex problems for which using a traditional approach yields **no good solution**: the best Machine Learning techniques can perhaps find a solution.
- **Fluctuating environments**: a Machine Learning system can adapt to new data.
- Getting insights about **complex problems and large amounts of data**.

Examples of Applications

- Analyzing images of products on a production line to automatically classify them
  - image classification, typically performed using CNN
- Detecting tumors in brain scans
  - This is semantic segmentation, where each pixel in the image is classified, typically performed using CNN
- Automatically classifying news articles
  - This is natural language processing (NLP), and more specifically, text classification, which can be tackled using recurrent neural networks (RNNs), CNNs, or Transformers
- Automatically flagging offensive comments on discussion forums
  - This is also text classification, using the same NLP tools
- Creating a chatbot or a personal assistant
  - NLP

Examples of Applications

- Forecasting your company's revenue next year, based on many performance metrics
  - This is a regression task (i.e., predicting values) that may be tackled using any regression model, such as a Linear Regression or Polynomial Regression model, a regression SVM, a regression Random Forest, or an artificial neural network.
  - If you want to take into account sequences of past performance metrics, you may want to use RNNs, CNNs, or Transformers.
- Making your app react to voice commands
  - This is speech recognition, which requires processing audio samples: since they are long and complex sequences, they are typically processed using RNNs, CNNs, or Transformers
- Detecting credit card fraud
  - This is anomaly detection
- Segmenting clients based on their purchases so that you can design a different marketing strategy for each segment
  - This is clustering

Types of Machine Learning Systems

- Whether or not they are trained with human supervision **(supervised, unsupervised, semi supervised, and Reinforcement Learning)**
- Whether or not they can learn incrementally on the fly **(online versus batch learning)**
- Whether they work by simply comparing new data points to known data points, or instead by detecting patterns in the training data and building a predictive model, much like scientists do **(instance-based versus model-based learning)**

Supervised/Unsupervised Learning

Machine Learning systems can be classified according to the amount and type of supervision they get during training.

- Supervised learning,
- Unsupervised learning,
- Semi-supervised learning, and
- Reinforcement Learning.

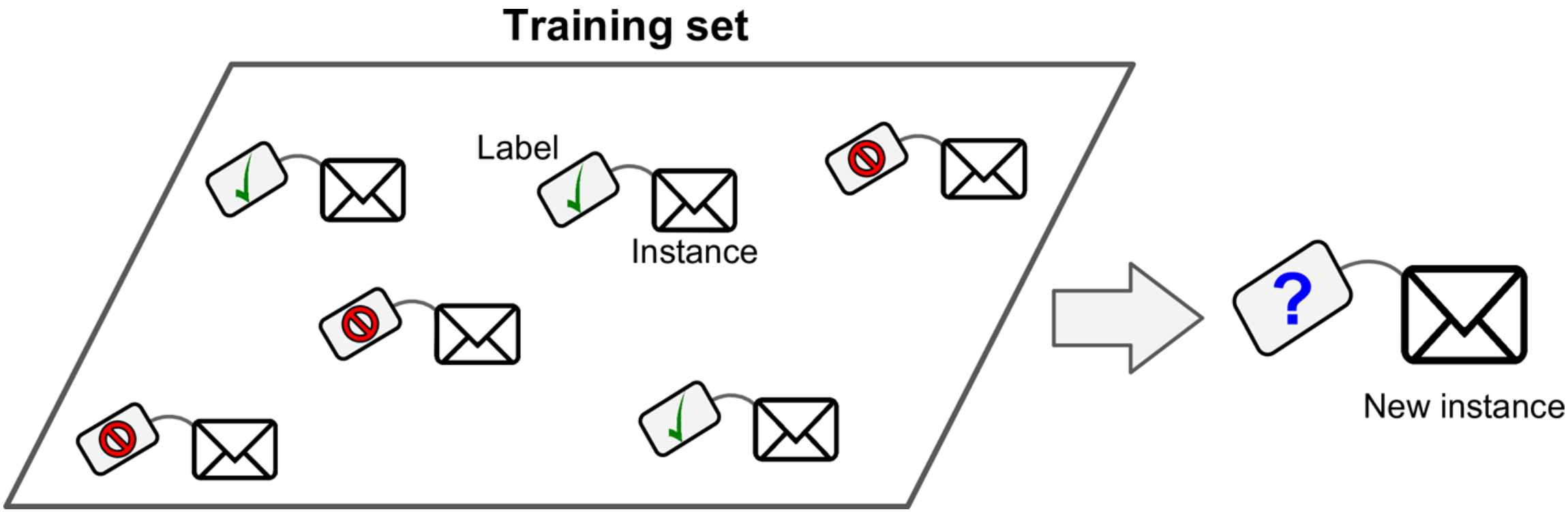
Supervised learning

- In *supervised learning*, the training set you feed to the algorithm includes the desired solutions, called *labels*.
- Some of the most important supervised learning algorithms
  - k-Nearest Neighbors
  - Linear Regression
  - Logistic Regression
  - Support Vector Machines (SVMs)
  - Decision Trees and Random Forests
  - Neural networks

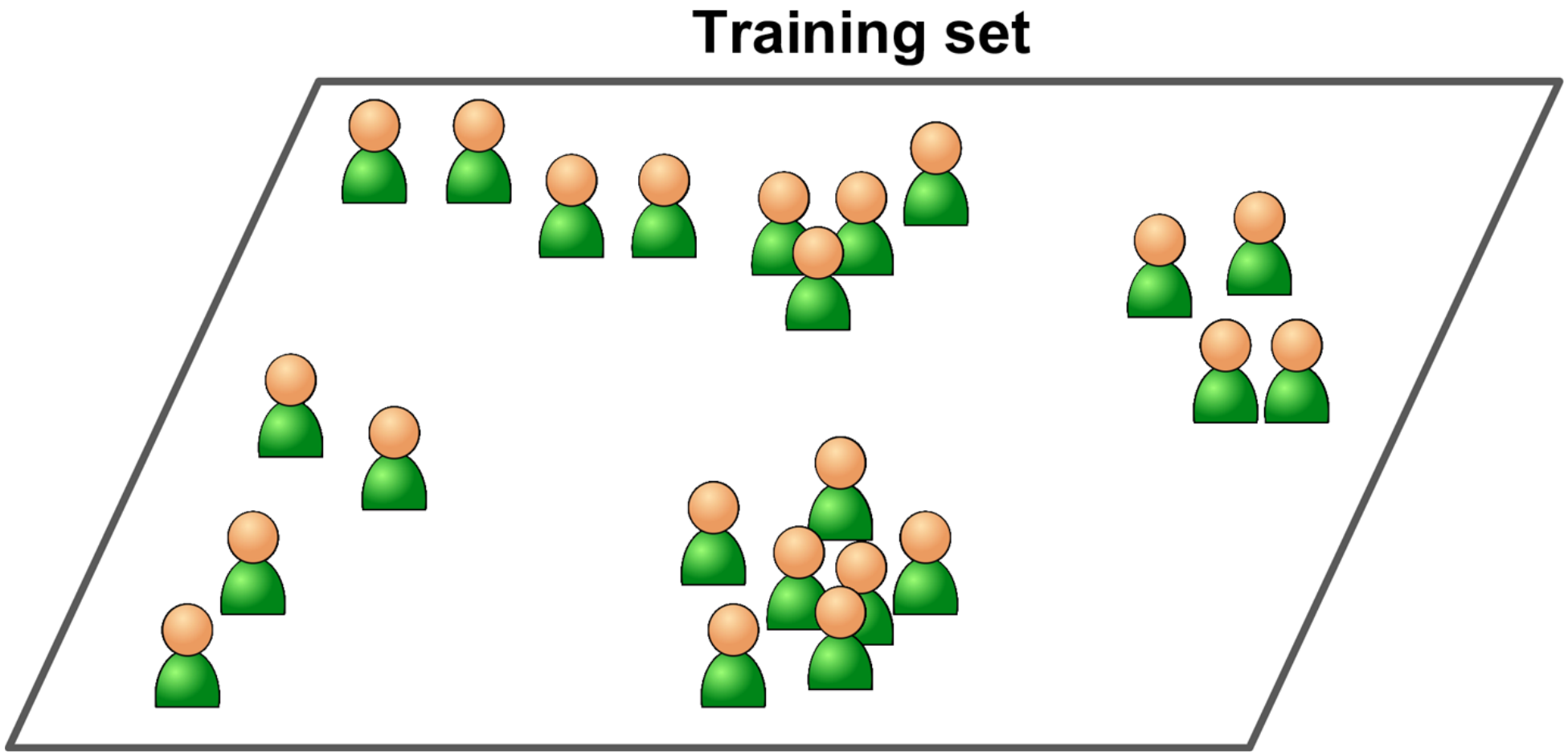
Unsupervised learning

- Clustering
  - K-Means
  - DBSCAN
  - Hierarchical Cluster Analysis (HCA)
  - Gaussian Mixture Models (GMM)
- Anomaly detection and novelty detection
  - One-class SVM
  - Isolation Forest
- Visualization and dimensionality reduction
  - Principal Component Analysis (PCA)
  - Kernel PCA
  - Locally Linear Embedding (LLE)
  - t-Distributed Stochastic Neighbor Embedding (t-SNE)
- Association rule learning
  - Apriori
  - Eclat

Supervised learning

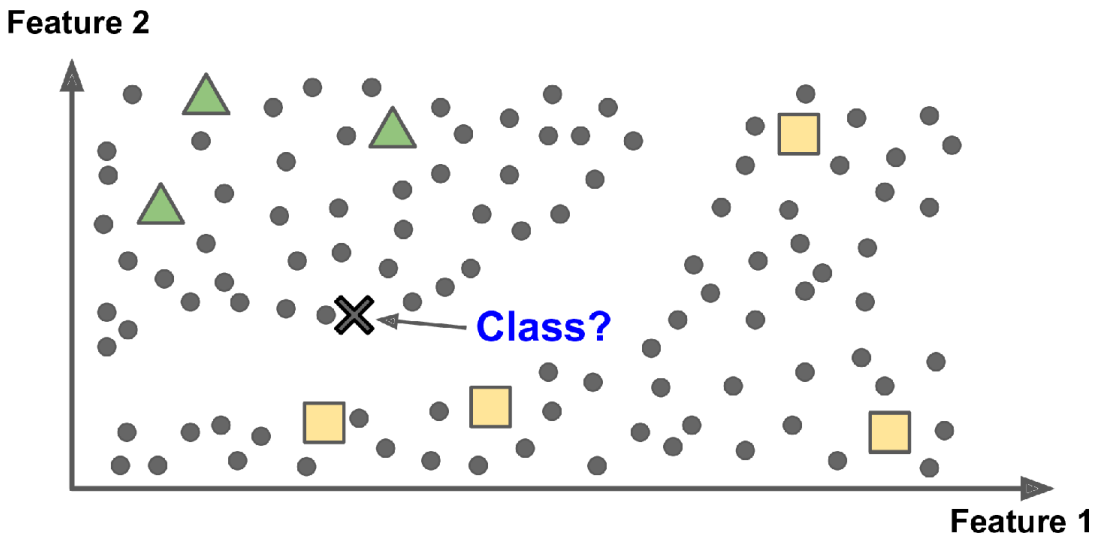


Unsupervised learning



Semi Supervised Learning

- Since labeling data is usually time-consuming and costly, you will often have plenty of unlabeled instances, and few labeled instances

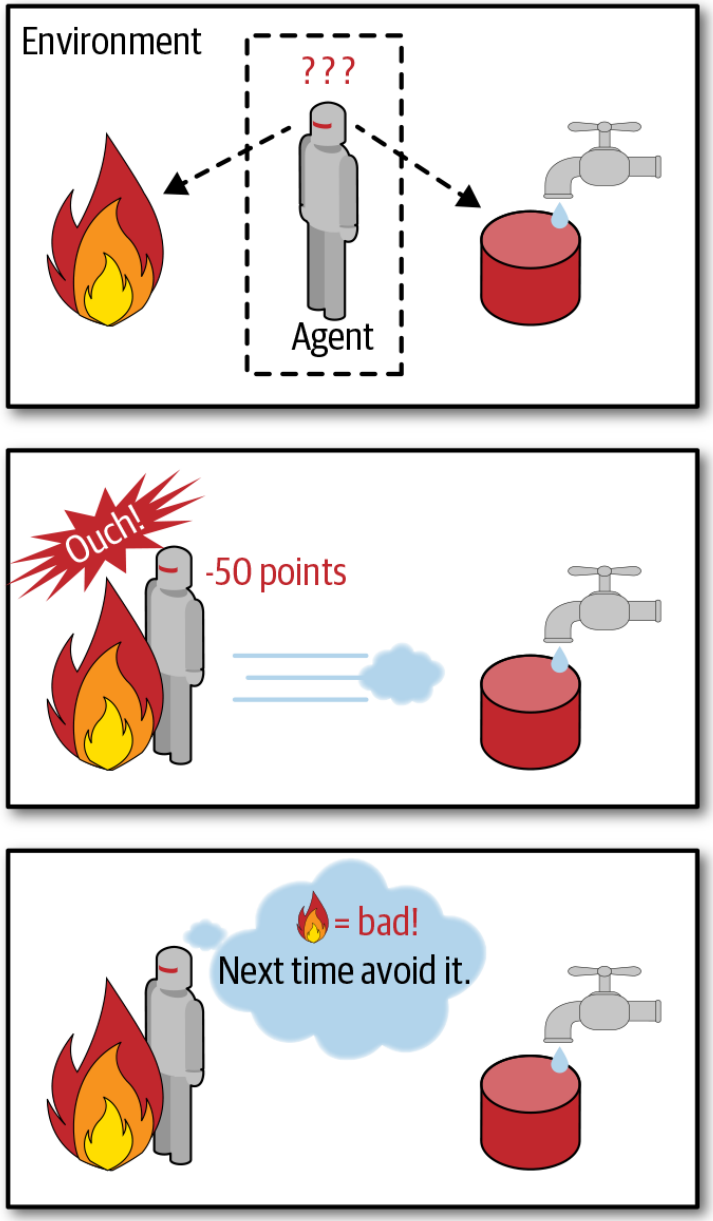


Reinforcement Learning

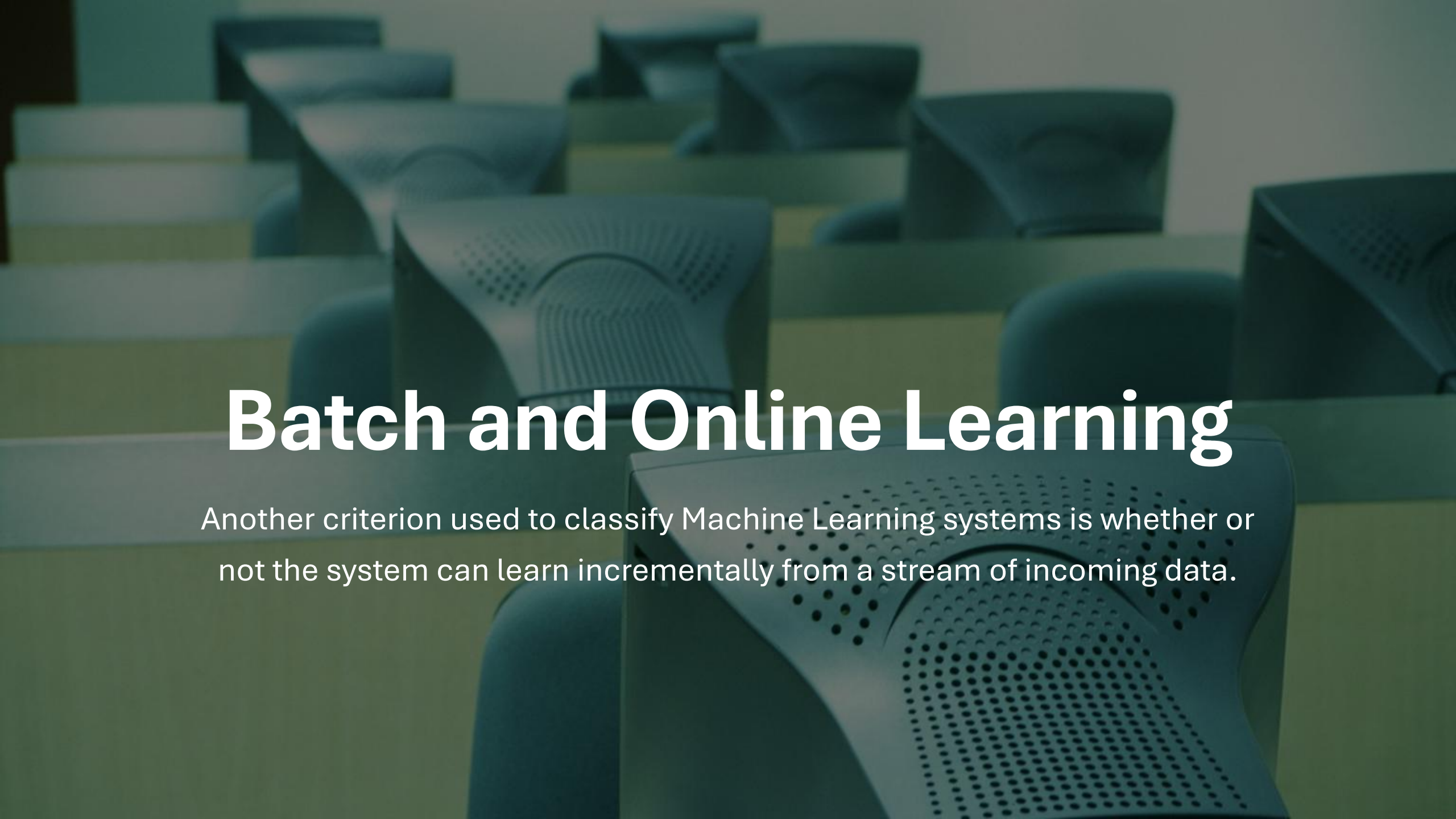
- The learning system (an *agent)* can observe the environment, select and perform actions, and
  - get *rewards* in return or
  - *penalties* in the form of negative rewards,

- It must then learn by itself what is the best strategy, called a *policy*, to get the most reward over time.

Reinforcement Learning



1. Observe
2. Select action using policy
3. Action!
4. Get reward or penalty
5. Update policy (learning step)
6. Iterate until an optimal policy is found



Batch and Online Learning

Another criterion used to classify Machine Learning systems is whether or not the system can learn incrementally from a stream of incoming data.

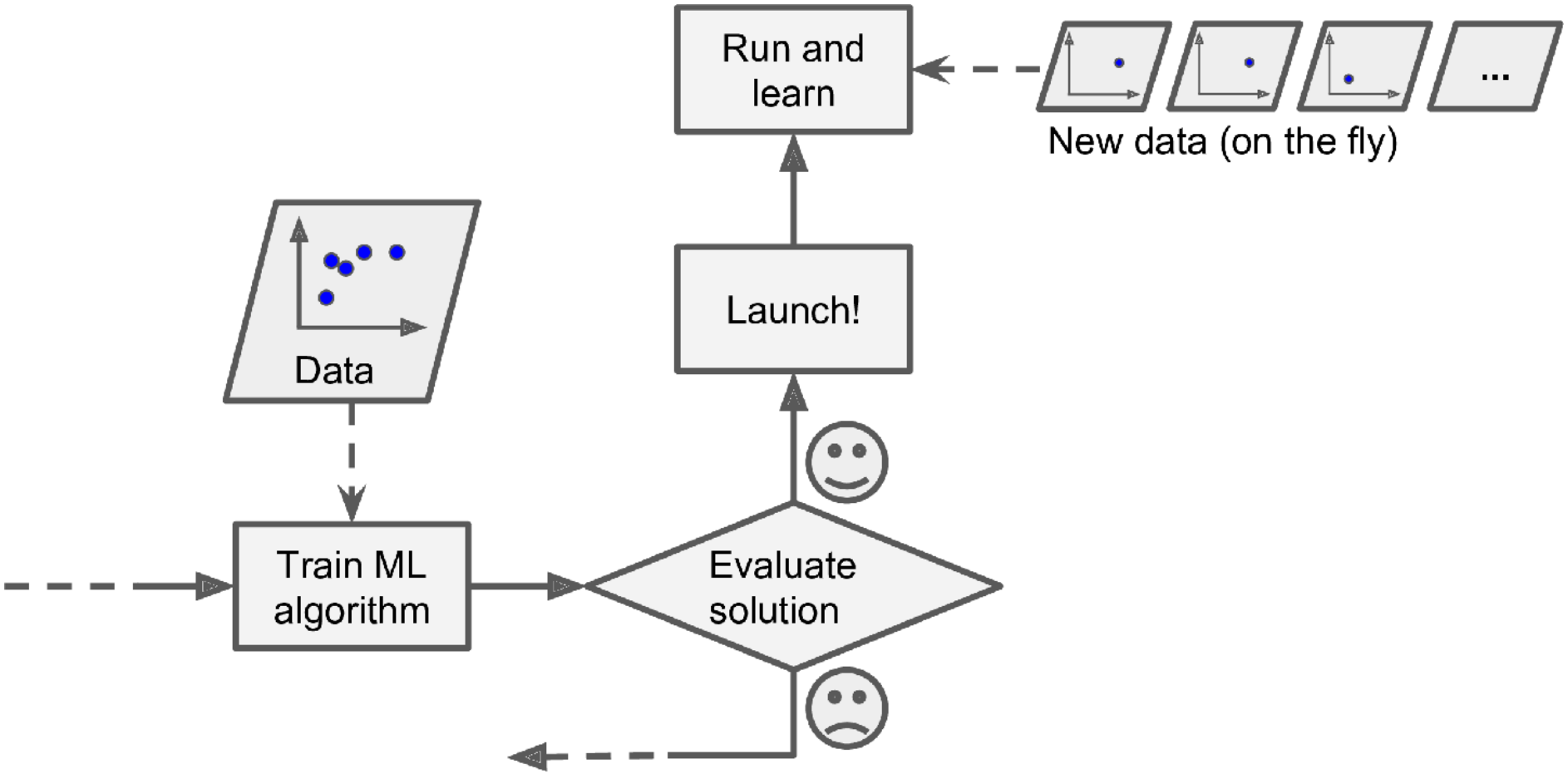
Batch learning

- In *batch learning*, the system is incapable of learning incrementally: it must be trained using all the available data.
- If you want a batch learning system to know about new data (such as a new type of spam), you need to train a new version of the system from scratch on the full dataset (not just the new data, but also the old data), then stop the old system and replace it with the new one.

Online learning

- In *online learning*, you train the system incrementally by feeding it data instances sequentially, either individually or in small groups called *minibatches*.
- Each learning step is fast and cheap, so the system can learn about new data on the fly, as it arrives

Online learning



Instance-Based Versus Model-Based Learning

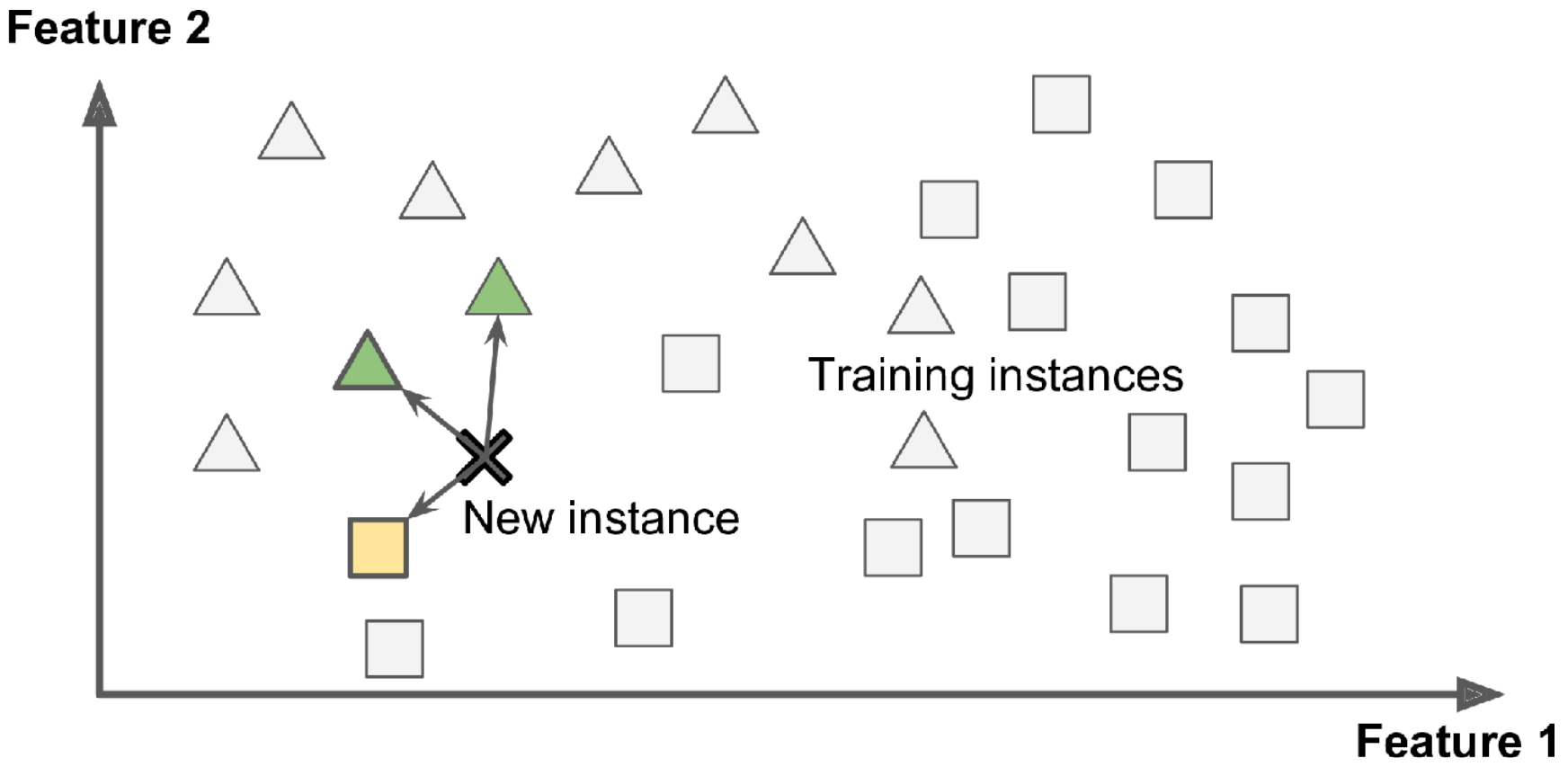
One more way to categorize Machine Learning systems is by how they *generalize*.

There are two main approaches to generalization:

instance-based learning	and model-based learning.

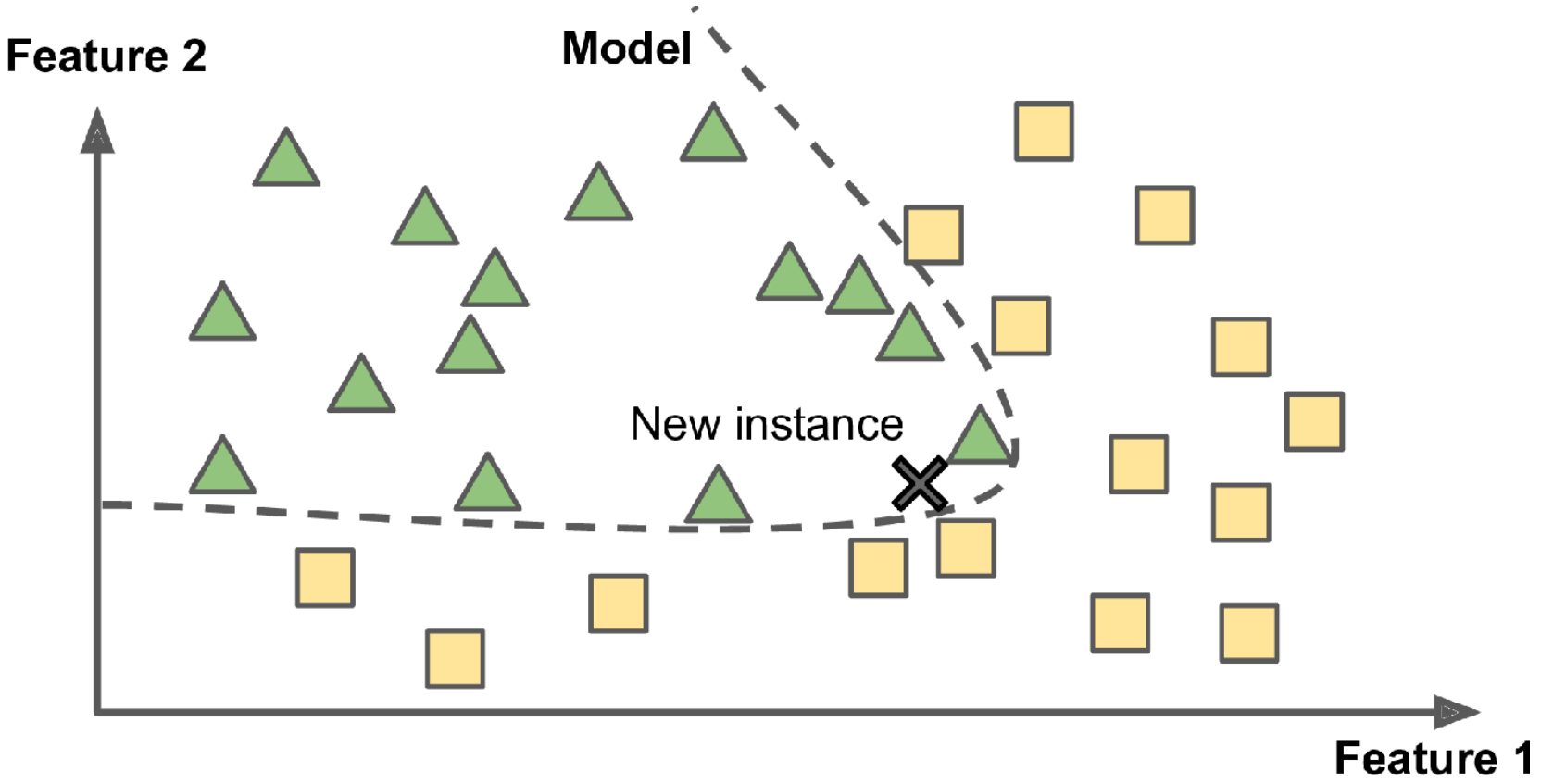
Instance-based learning

- The system learns the examples by heart, then generalizes to new cases by using a similarity measure to compare them to the learned examples (or a subset of them).



Model-based learning

- Another way to generalize from a set of examples is to build a model of these examples and then use that model to make *predictions*.



Main Challenges of Machine Learning

- **Insufficient Quantity of Training Data**
- **Nonrepresentative Training Data**
- **Poor-Quality Data**
- **Irrelevant Features**
- **Overfitting the Training Data**
- **Underfitting the Training Data**

Supervised Learning

Simple linear regression

Linear Regression is a statistical method to model the relationship between a dependent variable (target) and one or more independent variables (features). It is commonly used for prediction and finding trends.

- $y = mx + c$

Where:

- y = Dependent Variable (Target)
- x = Independent Variable (Feature)
- m = Slope of the line
- c = Intercept (value of y when x=0)

- Formula for Slope:

$$m = \frac{n(\sum xy) - (\sum x)(\sum y)}{n(\sum x^2) - (\sum x)^2}$$

- Formula for Intercept:

$$c = \frac{\sum y - m(\sum x)}{n}$$

Regression Metrics

Some common regression metrics are

- **Mean Absolute Error (MAE):** $MAE = \frac{1}{n}\sum_{i=1}^{n}|x_i - y_i|$
- **Mean Squared Error (MSE):** $MSE = \frac{1}{n}\sum_{i=1}^{n}(x_i - y_i)^2$
- **Root Mean Squared Error (RMSE):** $RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_i - y_i)^2}$
- **R-squared (R^2) Score:** $R^2 = 1 - (SSR / SST)$

$$\begin{aligned} \text{r2\_score} &= 1 - \frac{\text{total\_error\_model}}{\text{total\_error\_baseline}} \\ &= 1 - \frac{\sum_{i=1}^{N}(\text{predicted}_i - \text{actual}_i)^2}{\sum_{i=1}^{N}(\text{average\_value} - \text{actual}_i)^2} \end{aligned}$$

where,

- x_i represents the actual or observed value for the i-th data point.
- y_i represents the predicted value for the i-th data point.
- SSR (Sum of Squared Residuals) and SST (Total Sum of Squares).

Regression Metrics

Q. A real estate company is trying to predict the selling price of houses based on their size (in square feet). They trained a regression model and obtained the following predicted prices and actual selling prices for a sample of five houses:

Calculate the MAE, MSE, RMSE, R2 Score.

House	Actual Price (in $1000)	Predicted Price (in $1000)
1	300	280
2	350	360
3	420	410
4	280	310
5	500	480

Regression metrics

Mean Absolute Error (MAE)

- Lower is better; indicates how far off predictions are, on average.

Mean Squared Error (MSE)

- Penalizes larger errors more than MAE; useful for emphasizing outliers.

Root Mean Squared Error (RMSE)

- Easier to interpret than MSE; lower is better.

R-squared (R^2) – Coefficient of Determination

- $R^2 = 1$ → perfect prediction
- $R^2 = 0$ → model predicts no better than the mean
- Can be negative if the model performs worse than a constant predictor.

Logistic Regression

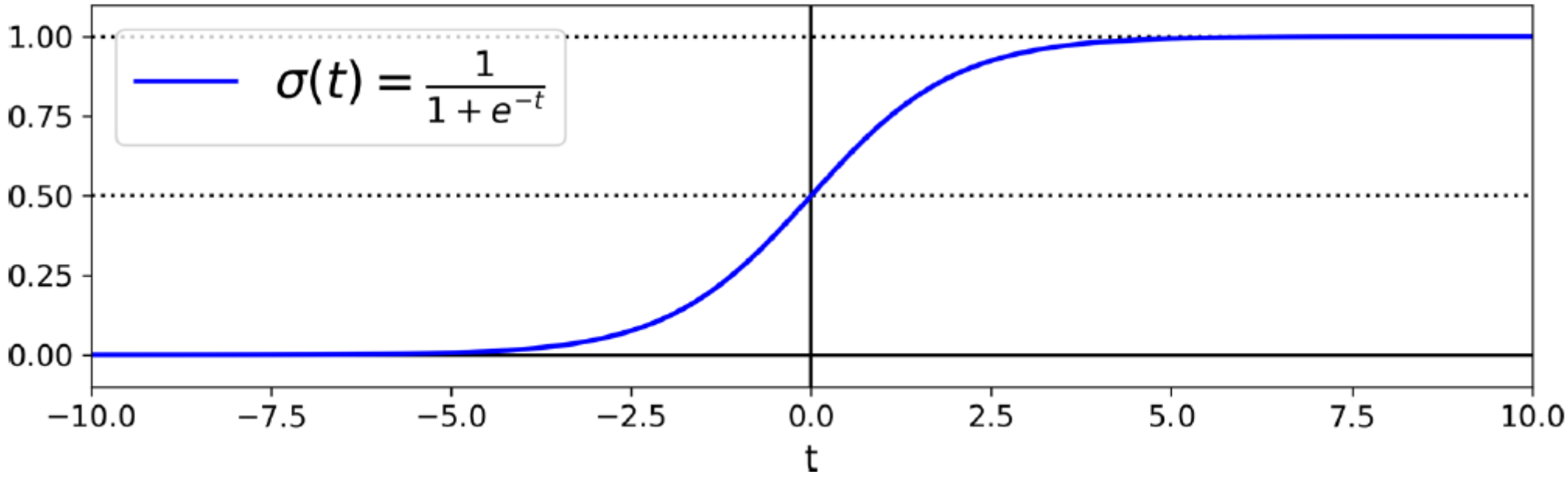
- *Logistic Regression* (also called *Logit Regression*) is commonly used to estimate the probability that an instance belongs to a particular class.
- e.g., what is the probability that the email is spam?.
- If the estimated probability is greater than 50%, then the model predicts that the instance belongs to that class
  - *positive class*, labeled “1”,
- Otherwise, it predicts that it does not
  - *negative class*, labeled “0”
- This makes it a binary classifier.
- Logistic Regression model computes a weighted sum of the input features (plus a bias term),

$$P(Y = 1|X) = \frac{1}{1 + e^{-(\beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_n X_n)}}$$

- The output of logistic regression is a probability between 0 and 1
- If the probability > 0.5 (threshold), it predicts class 1; else, class 0

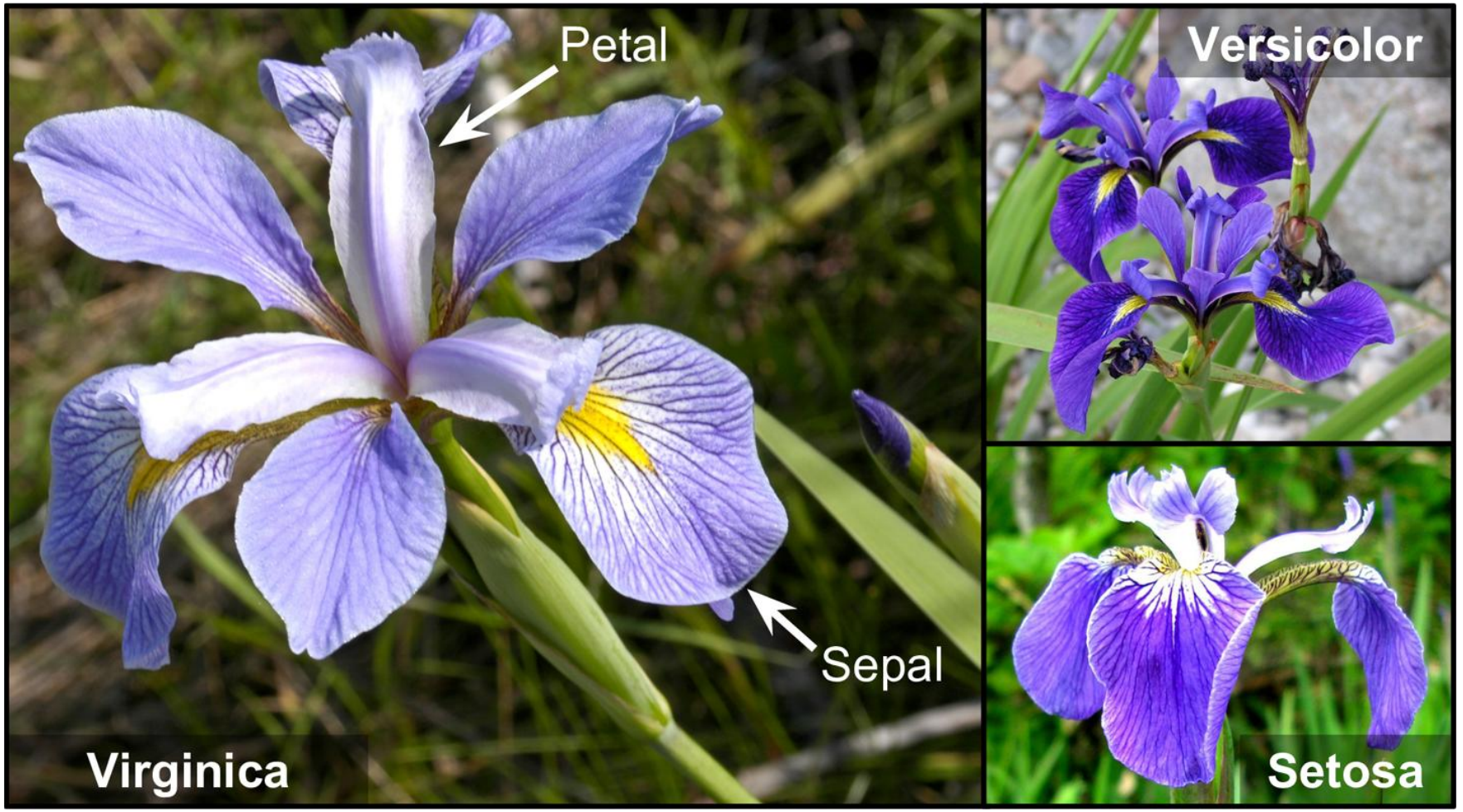
Logistic Regression

$$\sigma(t) = \frac{1}{1 + \exp(-t)}$$



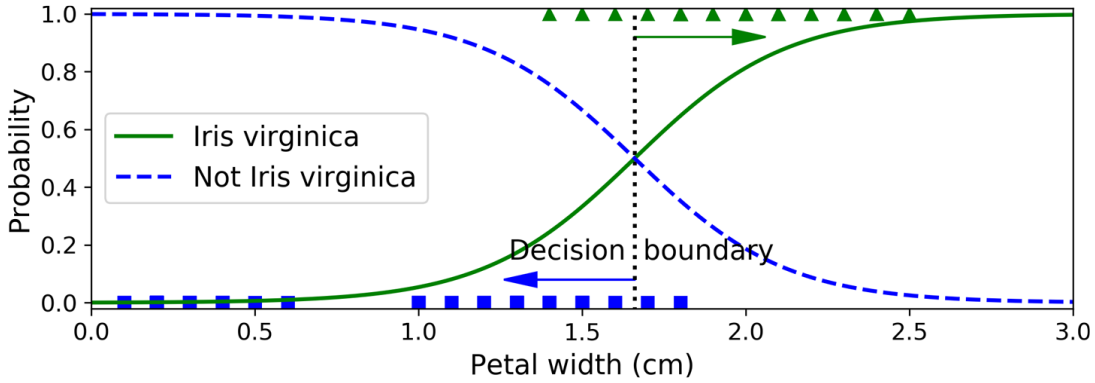
Decision Boundaries

- Classifier to detect the *Iris virginica* type based only on the petal width feature



Steps

- Load the data
- Train a Logistic Regression Model
- Estimate probabilities for flowers with petal widths varying from 0 cm to 3 cm

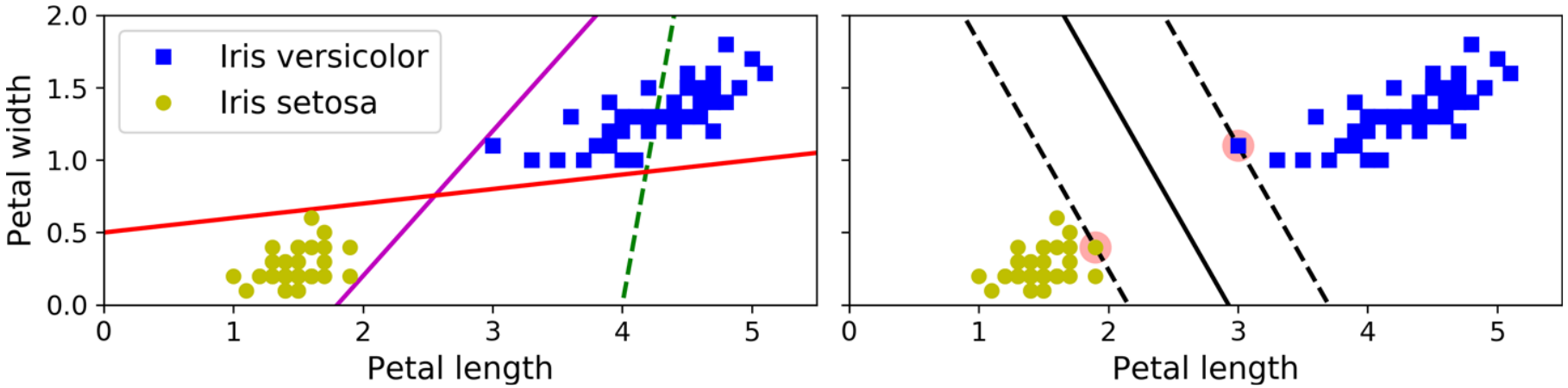


Support Vector Machines

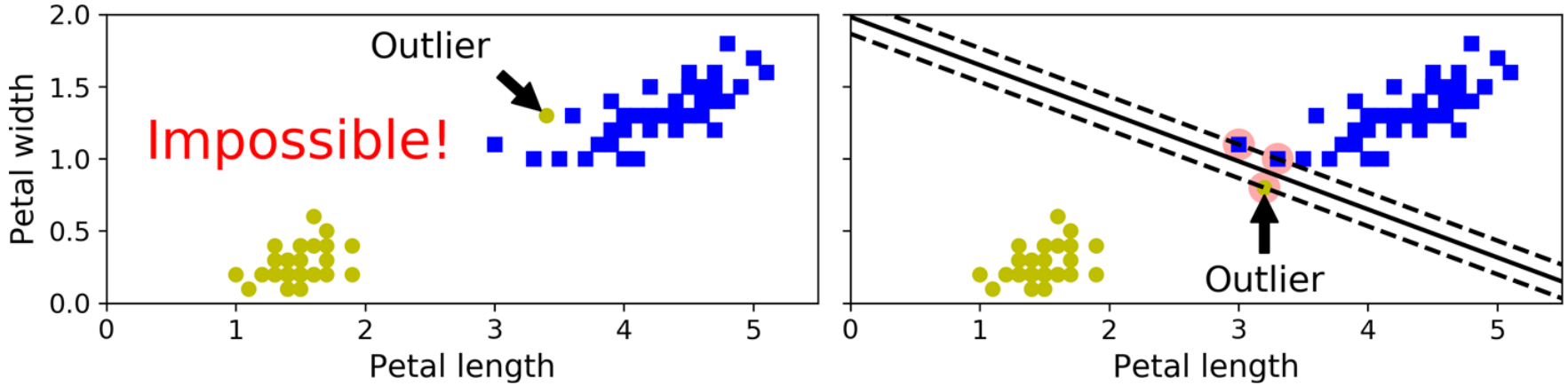
- Capable of performing linear or nonlinear classification, regression, and even outlier detection
- SVMs are particularly well suited for classification of complex small- or medium-sized datasets.
- Support vector machines classify data by finding the hyperplane that maximizes the margin between the classes in the training data.

Linear SVM Classification

Large margin classification

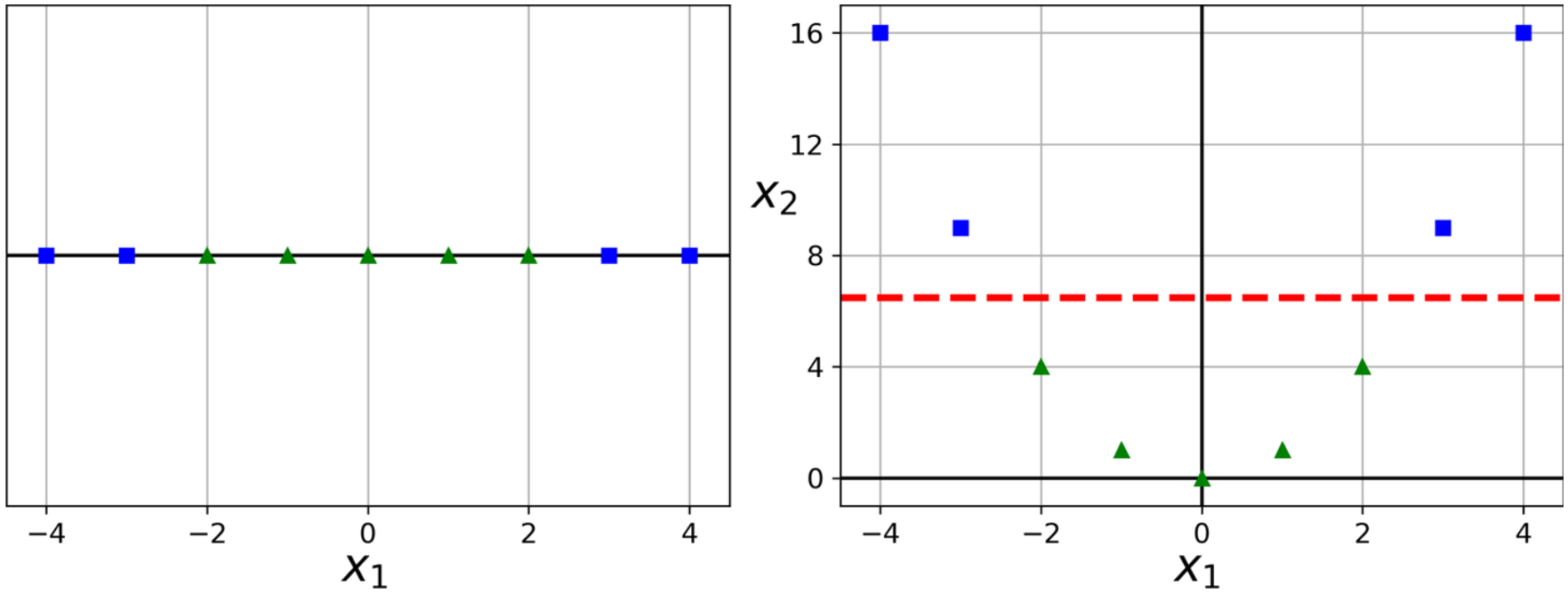


Hard Margin Classification



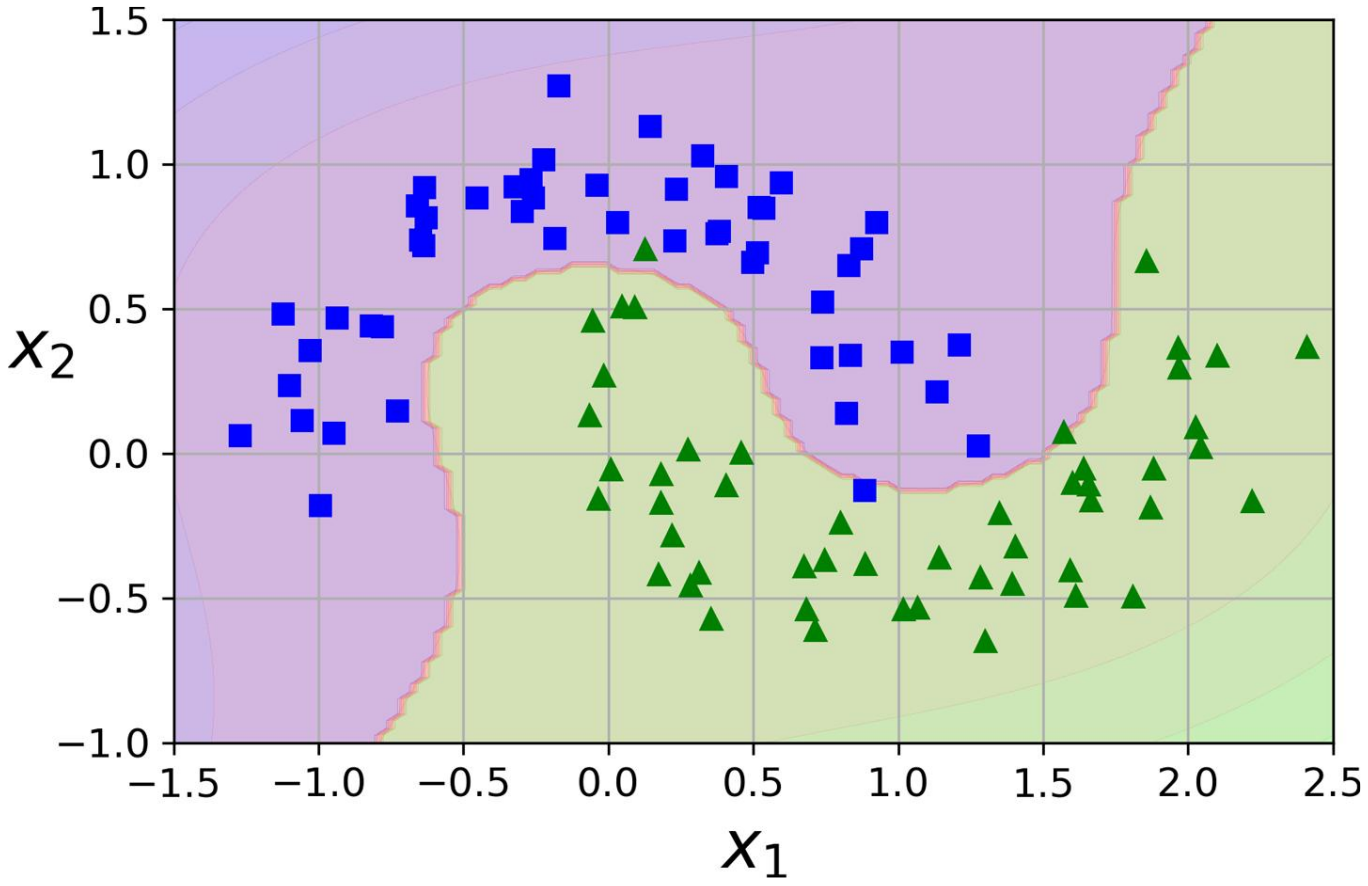
Non Linear SVM Classification

- *Adding features to make a dataset linearly separable*



Linear SVM classifier using polynomial features

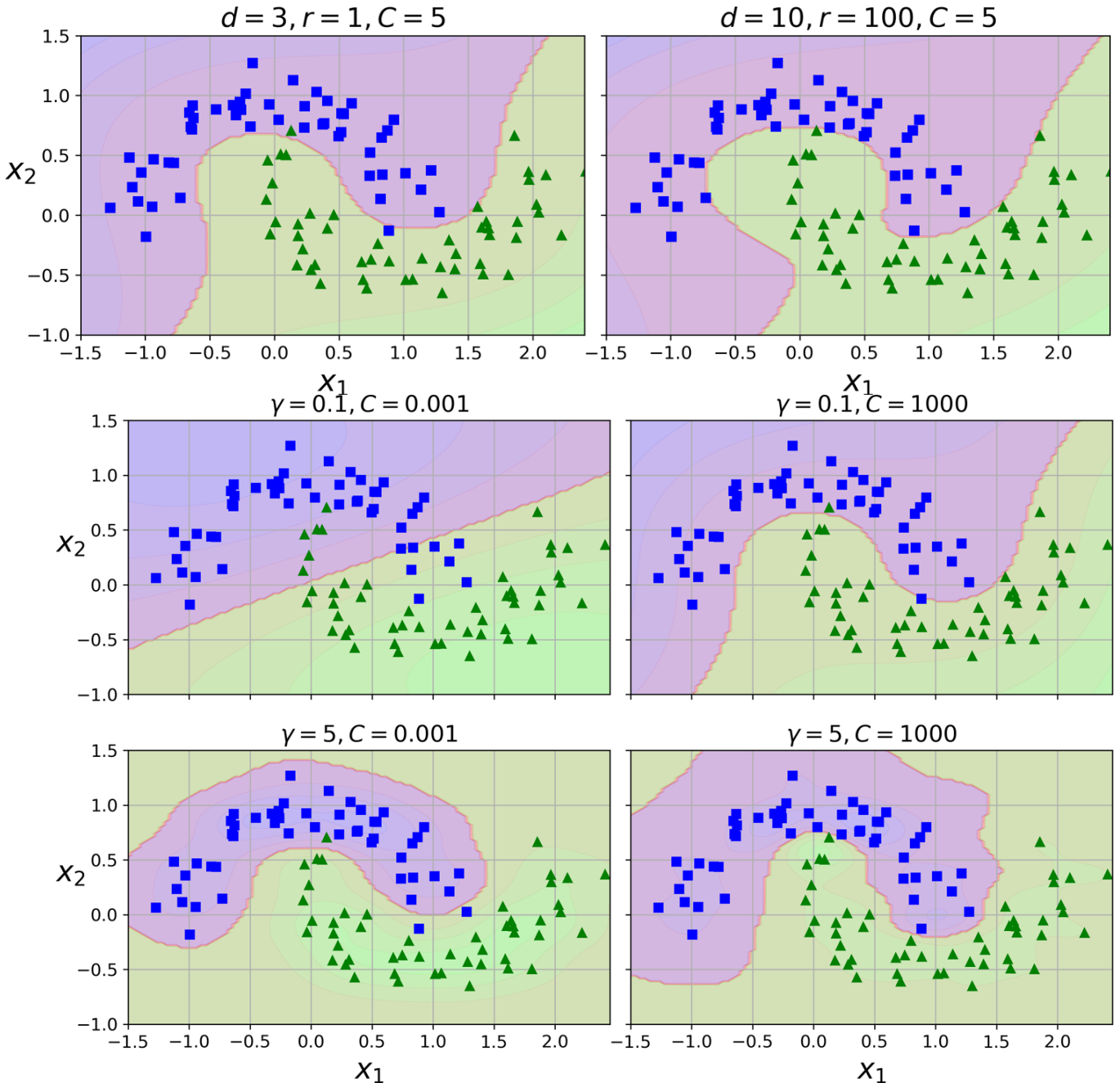
- moons dataset: this is a toy dataset for binary classification in which the data points are shaped as two interleaving half circles



Polynomial Kernel

- Adding polynomial features is simple to implement and can work great with all sorts of Machine Learning algorithms (not just SVMs).
- **SVM classifiers with a polynomial kernel**

Gaussian RBF Kernel



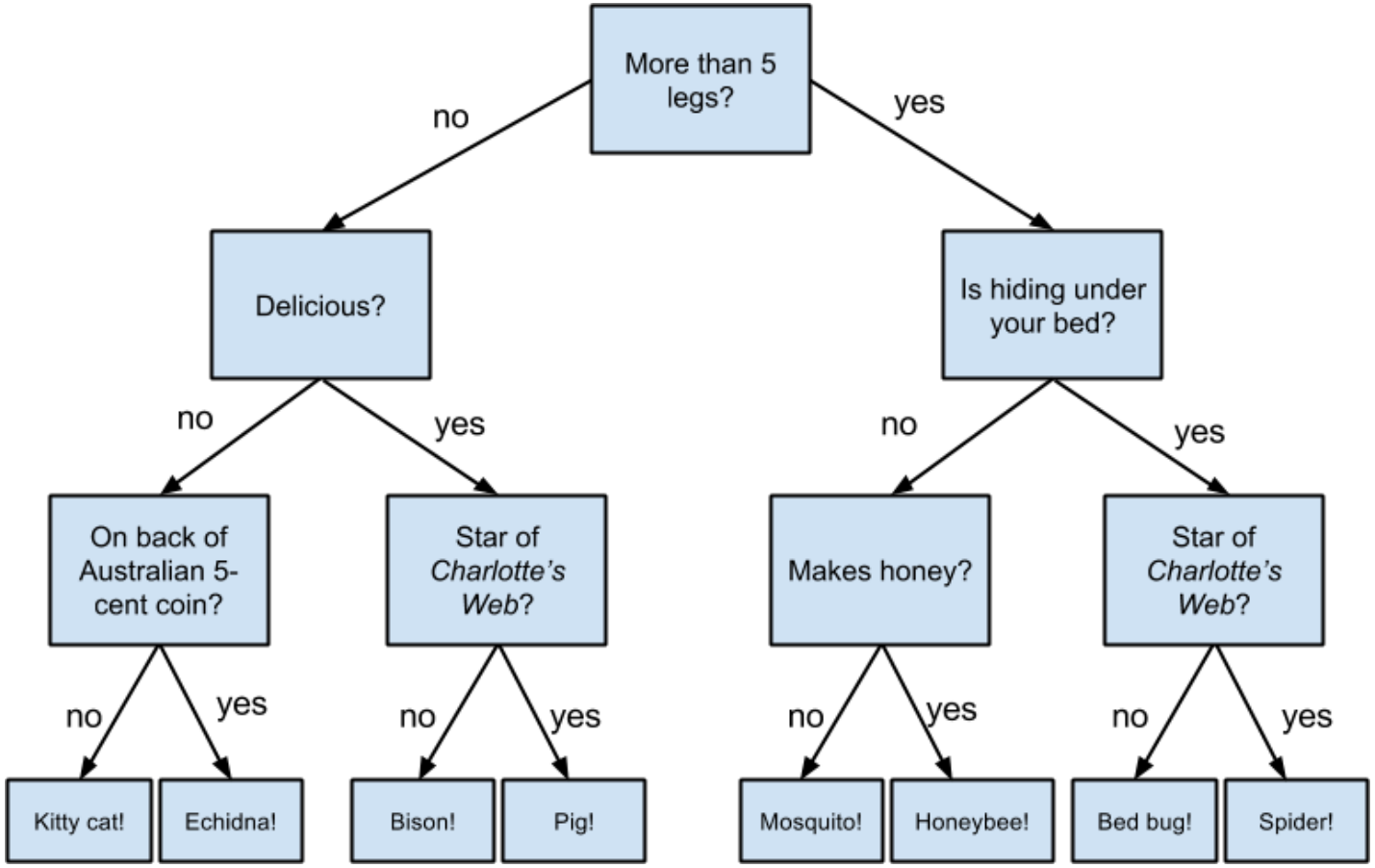
Decision Tree

- *Decision Trees* are versatile Machine Learning algorithms that can perform both classification and regression tasks, and even multioutput tasks.
- In a decision tree, every decision rule occurs at a decision node, with the rule creating branches leading to new nodes. A branch without a decision rule at the end is called a *leaf*.
- Decision trees can easily handle a mix of numeric (e.g., number of legs) and categorical (e.g., delicious/not delicious) attributes and can even classify data for which attributes are missing.
- Finding an “optimal” decision tree for a set of training data is computationally a very hard problem.

Q & A game - “guess the animal”

- “I am thinking of an animal.”
- “Does it have more than five legs?”
- “No.”
- “Is it delicious?”
- “No.”
- “Does it appear on the back of the Australian five-cent coin?”
- “Yes.”
- “Is it an echidna?”
- “Yes, it is!”,

“guess the animal” decision tree



Decision Tree

To build a decision tree,

- we will need to decide what questions to ask and in what order.

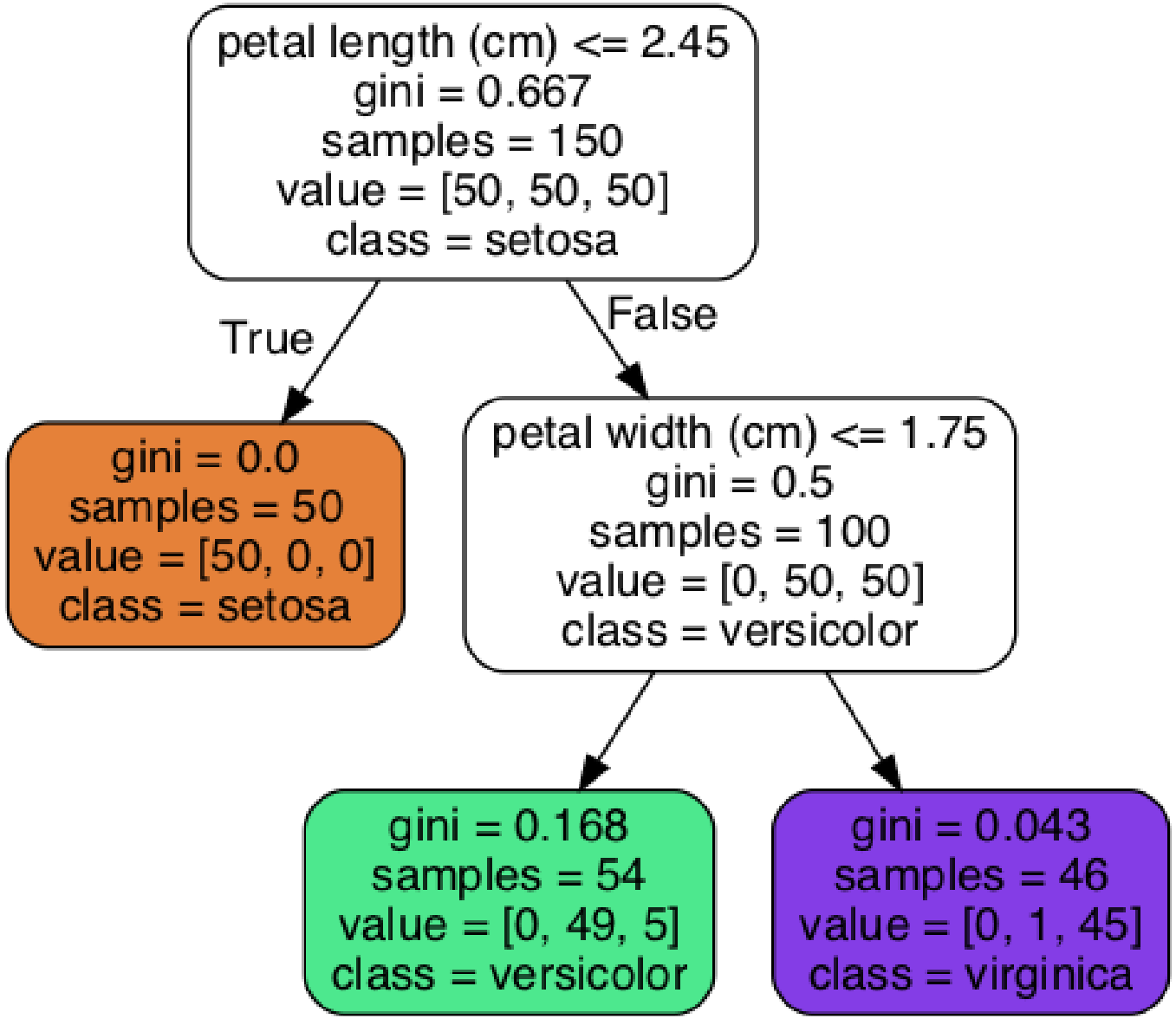
At each stage of the tree there are some possibilities

- eliminated and some that we haven't.

Entropy

- "how much information"

Iris Decision Tree

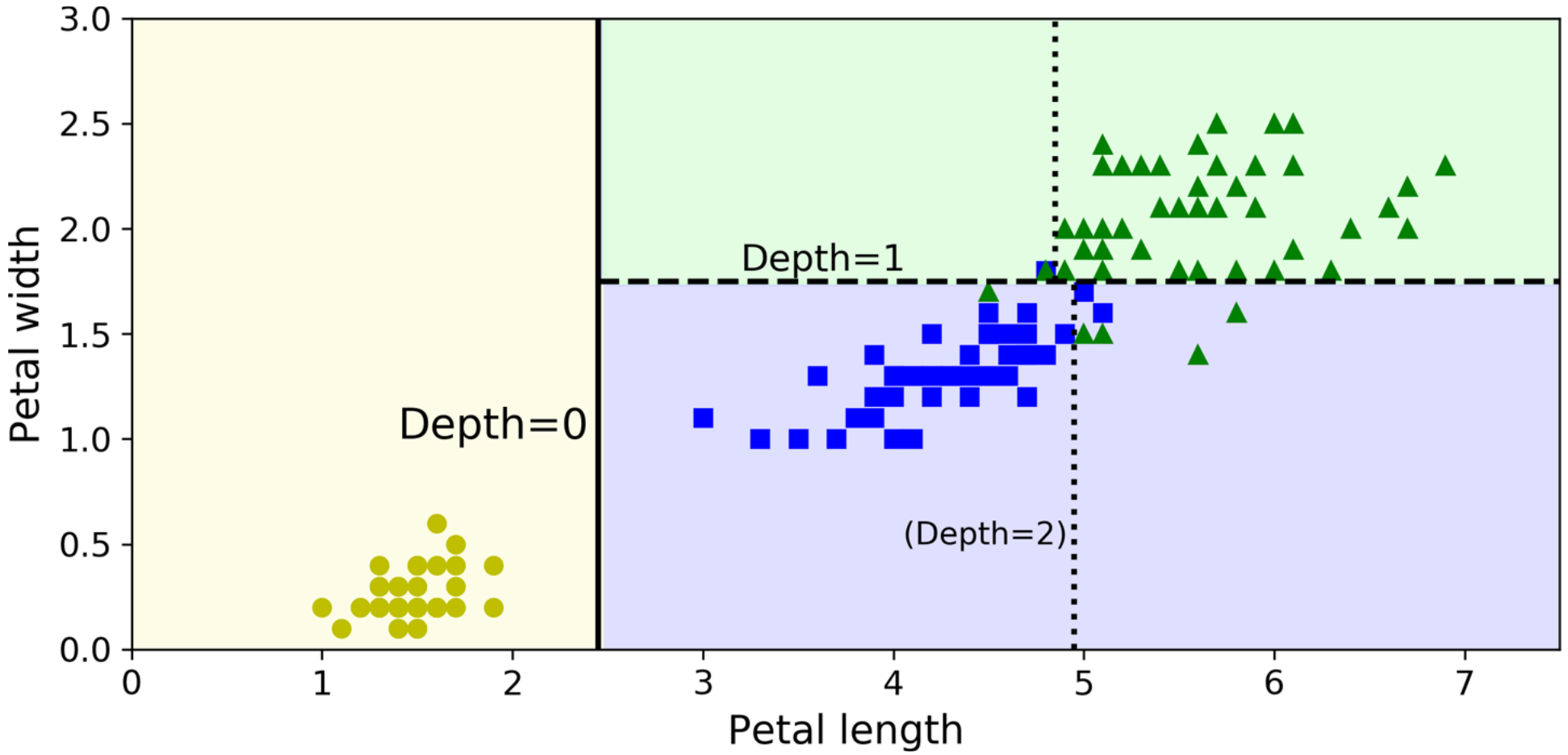


Gini impurity

- a node's gini attribute measures its *impurity*:
  - a node is "pure" (gini=0) if all training instances it applies to belong to the same class.
- For example, since the depth-1 left node applies only to *Iris setosa* training instances, it is pure and its gini score is 0.
- he depth-2 left node has a gini score equal to 1 – (0/54) – (49/54) – (5/54) ≈ 0.168.
- *Pi,k* is the ratio of class *k* instances among the training instances in the i *th* node.

$$G_i = 1 - \sum_{k=1}^{n} {p_{i,k}}^2$$

Decision Tree decision boundaries



Why Is SVM Effective on High Dimensional Data?

- The **complexity** of trained classifier is characterized by the # of support vectors rather than the dimensionality of the data
- The **support vectors** are the essential or critical training examples —they lie closest to the decision boundary
- If all other training examples are removed and the training is repeated, the same separating hyperplane would be found
- The number of support vectors found can be used to compute an (upper) bound on the expected error rate of the SVM classifier, which is independent of the data dimensionality
- Thus, an SVM with a small number of support vectors can have good generalization, even when the dimensionality of the data is high

Model Evaluation and Selection

- Evaluation metrics: How can we measure accuracy? Other metrics to consider?
- Use **validation/test set** of class-labeled tuples instead of training set when assessing accuracy
- Methods for estimating a classifier's accuracy:
  - Holdout method, random subsampling
  - Cross-validation
  - Bootstrap

Holdout Method / Random Subsampling

Split the dataset into two parts:

- Training set (e.g., 70–80%)
- Test set (e.g., 20–30%)

Train on the training set, evaluate on the test set.

In **random subsampling**, this is done **multiple times** with different splits.

Simple to implement
Can give **high variance** results if the dataset is small.

Cross-Validation

The data is split into **k-folds** (commonly 5 or 10).

Each fold gets a chance to be the test set once; remaining folds form the training set.

Final score is the **average of all iterations**.

More stable and reliable evaluation
Computationally more expensive

Bootstrap

Samples are drawn **with replacement** to create multiple datasets of the same size.

Each model is trained on a bootstrap sample and tested on the **out-of-bag** instances (not selected in that sample).

Useful for estimating the **distribution of model performance**
More complex to implement

Classifier Evaluation Metrics: Confusion Matrix

Confusion Matrix:

Actual class\Predicted class	C_1	¬ C_1
C_1	**True Positives (TP)**	**False Negatives (FN)**
¬ C_1	**False Positives (FP)**	**True Negatives (TN)**

Example of Confusion Matrix:

Actual class\Predicted class	buy_computer = yes	buy_computer = no	Total
buy_computer = yes	**6954**	**46**	7000
buy_computer = no	**412**	**2588**	3000
Total	7366	2634	10000

- Given *m* classes, an entry, $\boldsymbol{CM_{i,j}}$ in a **confusion matrix** indicates # of tuples in class *i* that were labeled by the classifier as class *j*
- May have extra rows/columns to provide totals

Classifier Evaluation Metrics

- **Classifier Accuracy,** or recognition rate: percentage of test set tuples that are correctly classified.

  Accuracy = (TP + TN)/All

- **Error rate = *1 – accuracy* = (FP + FN)/All**

- **Class Imbalance Problem**:
  - One class may be *rare*, e.g. fraud, or HIV-positive
  - Significant *majority of the negative class* and minority of the positive class

- **Precision**: exactness – what % of tuples that the classifier labeled as positive are actually positive.
- **Recall:** completeness – what % of positive tuples did the classifier label as positive?
- Perfect score is 1.0

$$precision = \frac{TP}{TP + FP} \qquad recall = \frac{TP}{TP + FN}$$

- Inverse relationship between precision & recall
- ***F* measure (*F_1*- score)**: harmonic mean of precision and recall.

$$F = \frac{2 \times precision \times recall}{precision + recall}$$

Classifier Evaluation Metrics

Classification Metrics Examples

	Predicted		
Actual	Positive	Negative	Row Totals
Positive	60	10	70
Negative	5	25	30
Col Totals	65	35	100

$$\text{Recall} = \frac{60}{70} = 0.857$$

$$\text{Specificity} = \frac{25}{30} = 0.833$$

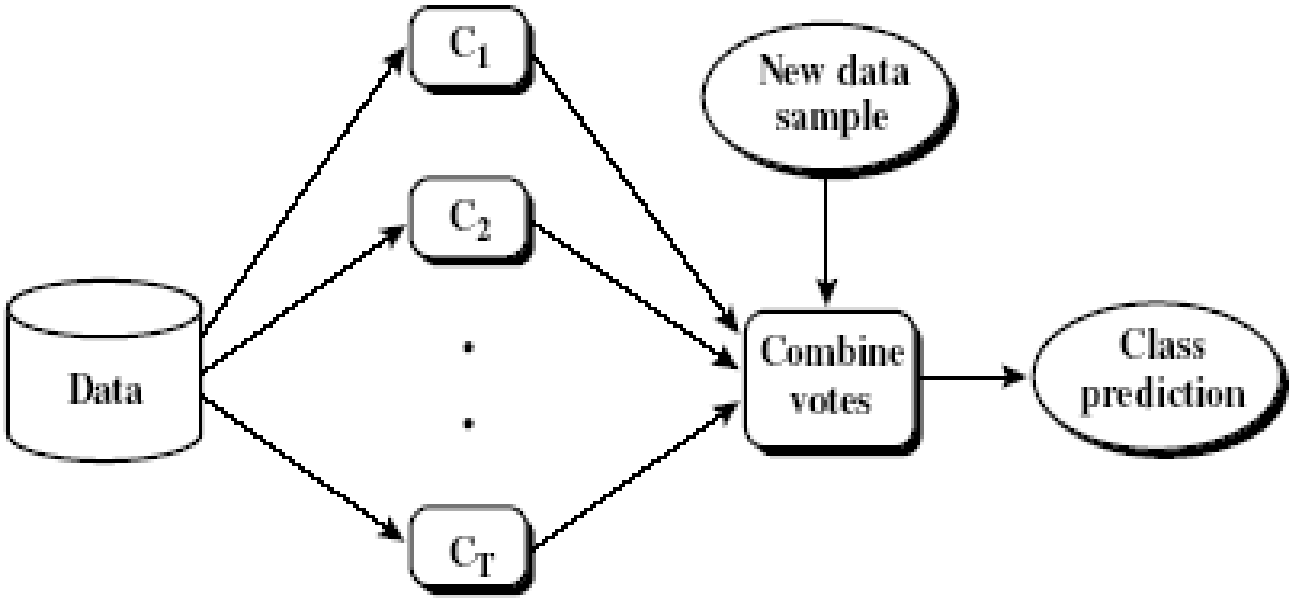
$$\text{Error} = \frac{15}{100} = 15\%$$

$$\text{Accuracy} = \frac{85}{100} = 85\%$$

$$\text{Precision} = \frac{60}{65} = 0.923$$

$$F = 2 * \frac{0.857 * 0.923}{0.857 + 0.923} = 0.889$$

Ensemble Methods: Increasing the Accuracy

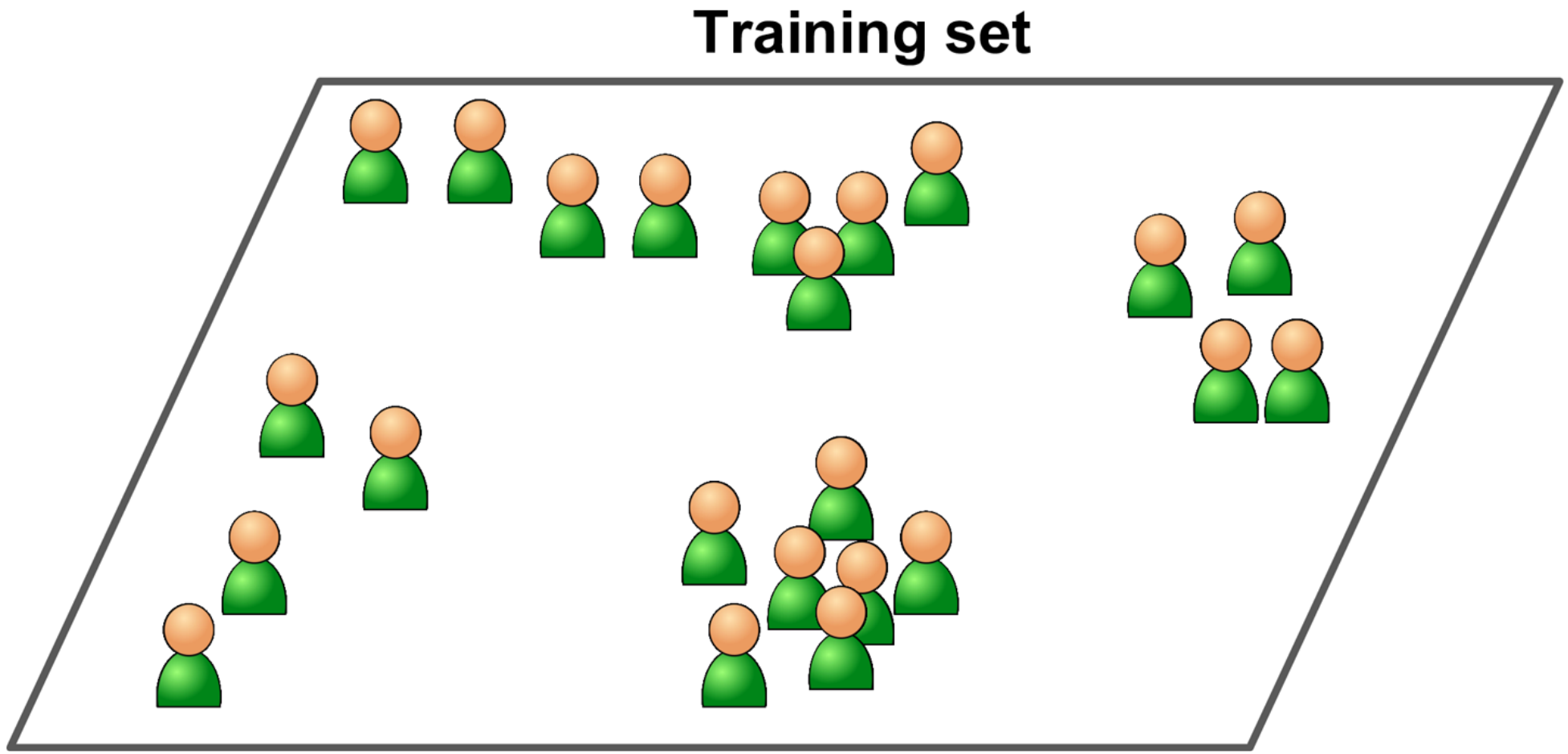


- Ensemble methods
  - Use a combination of models to increase accuracy
  - Combine a series of k learned models, M_1, M_2, …, M_k, with the aim of creating an improved model M*
- Popular ensemble methods
  - Bagging: averaging the prediction over a collection of classifiers
  - Boosting: weighted vote with a collection of classifiers
  - Ensemble: combining a set of heterogeneous classifiers

Random Forest

- Random Forest:
  - Each classifier in the ensemble is a *decision tree* classifier and is generated using a random selection of attributes at each node to determine the split
  - During classification, each tree votes and the most popular class is returned
- Two Methods to construct Random Forest:
  - Forest-RI (*random input selection*): Randomly select, at each node, F attributes as candidates for the split at the node. The CART methodology is used to grow the trees to maximum size
  - Forest-RC (*random linear combinations*): Creates new attributes (or features) that are a linear combination of the existing attributes (reduces the correlation between individual classifiers)
- Insensitive to the number of attributes selected for consideration at each split, and faster than bagging or boosting

Unsupervised Learning

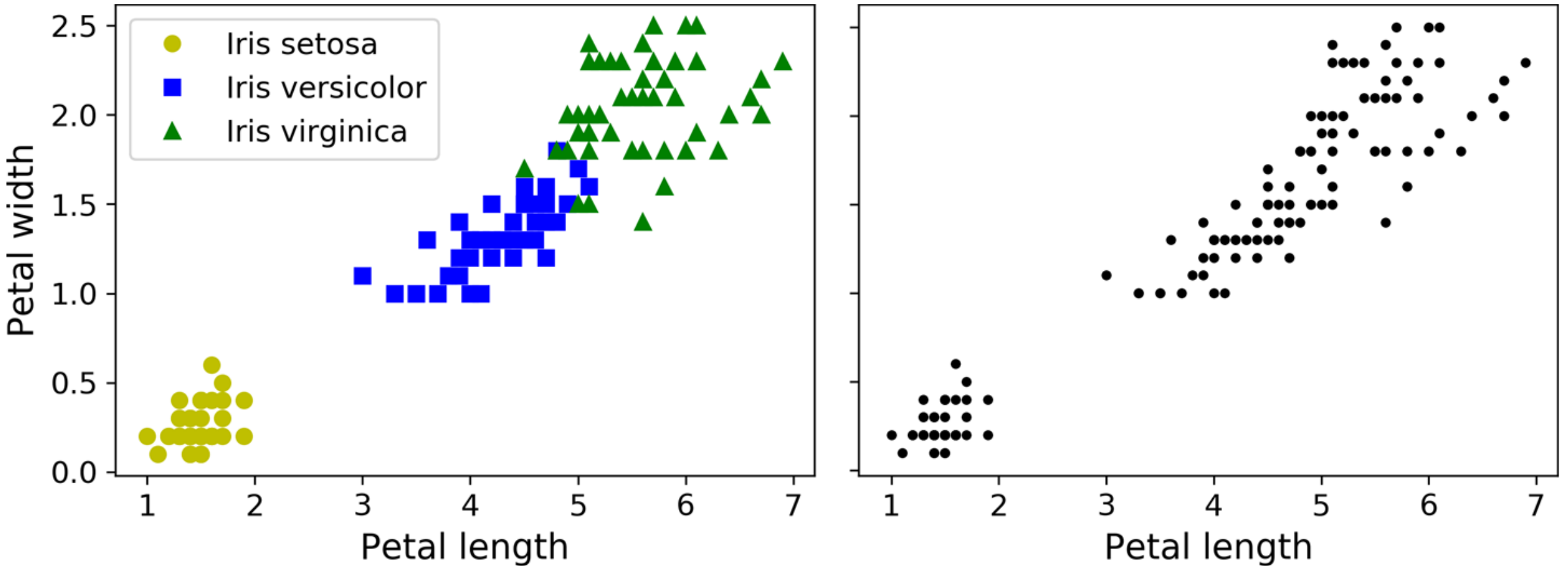

Training set

Unsupervised learning tasks and algorithms

- Anomaly detection
  - The objective is to learn what “normal” data looks like, and then use that to detect abnormal instances, such as defective items on a production line or a new trend in a time series.
- Density estimation
  - This is the task of estimating the *probability density function* (PDF) of the random process that generated the dataset.
  - Density estimation is commonly used for anomaly detection: instances located in very low-density regions are likely to be anomalies. It is also useful for data analysis and visualization.

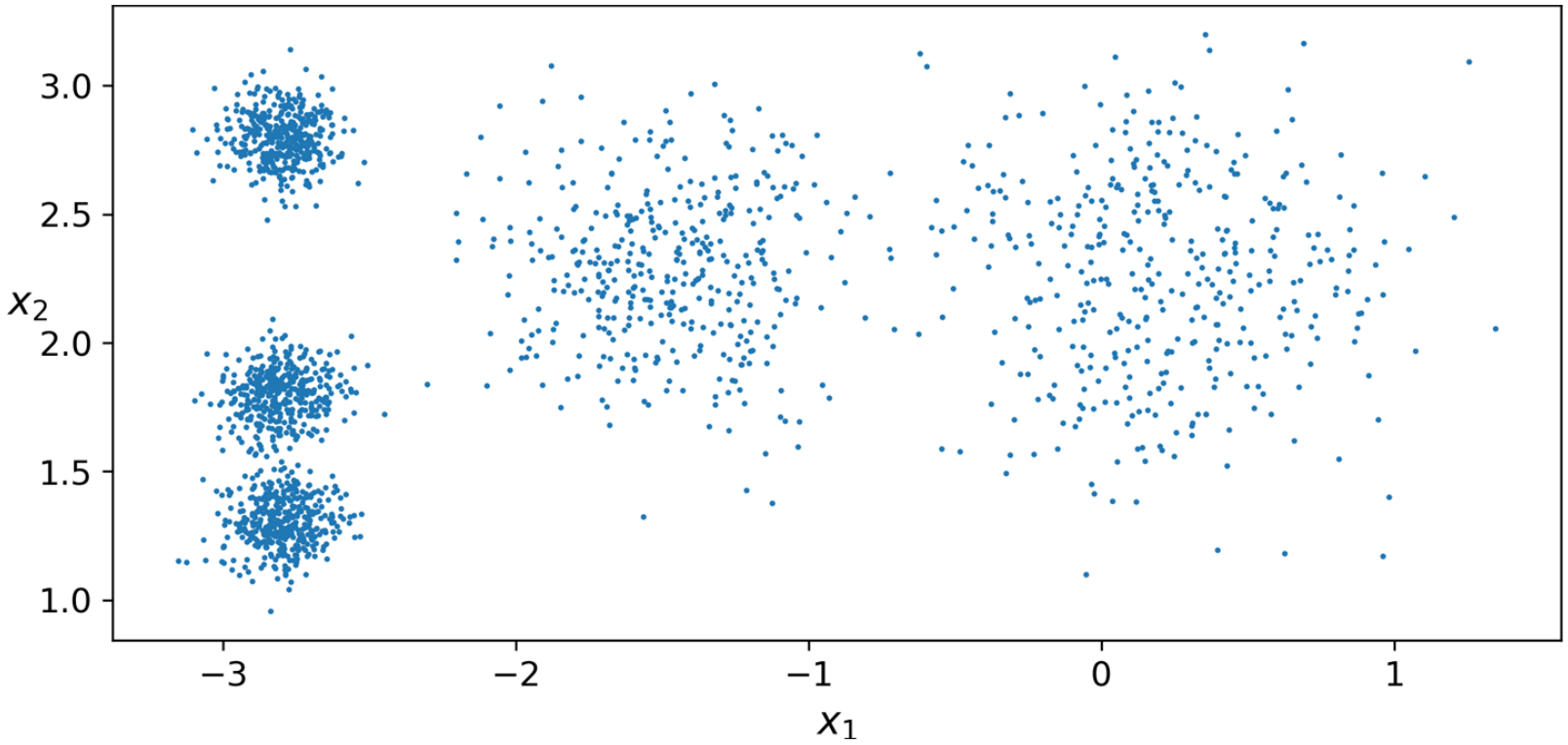
- Dimensionality Reduction
- Clustering
  - The goal is to group similar instances together into *clusters*.
  - Clustering is a great tool for
  - data analysis,
  - customer segmentation,
  - Recommender systems,
  - search engines,
  - image segmentation,
  - semi-supervised
  - learning, dimensionality reduction, and more.

Clustering


Iris setosa
Iris versicolor
Iris virginica
Petal width
Petal length
Petal length

K-Means

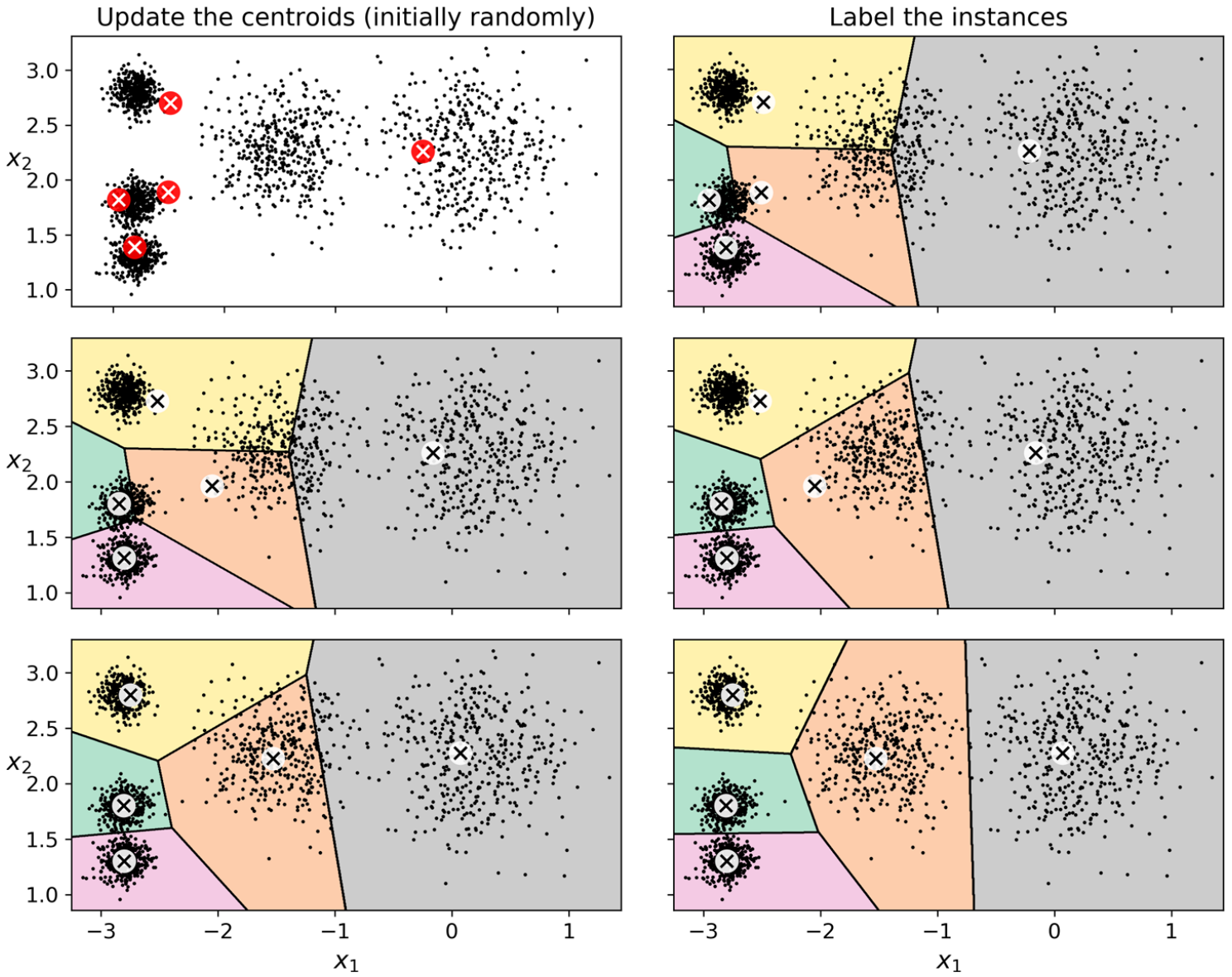
An unlabeled dataset composed of five blobs of instances



K-Means Algorithm

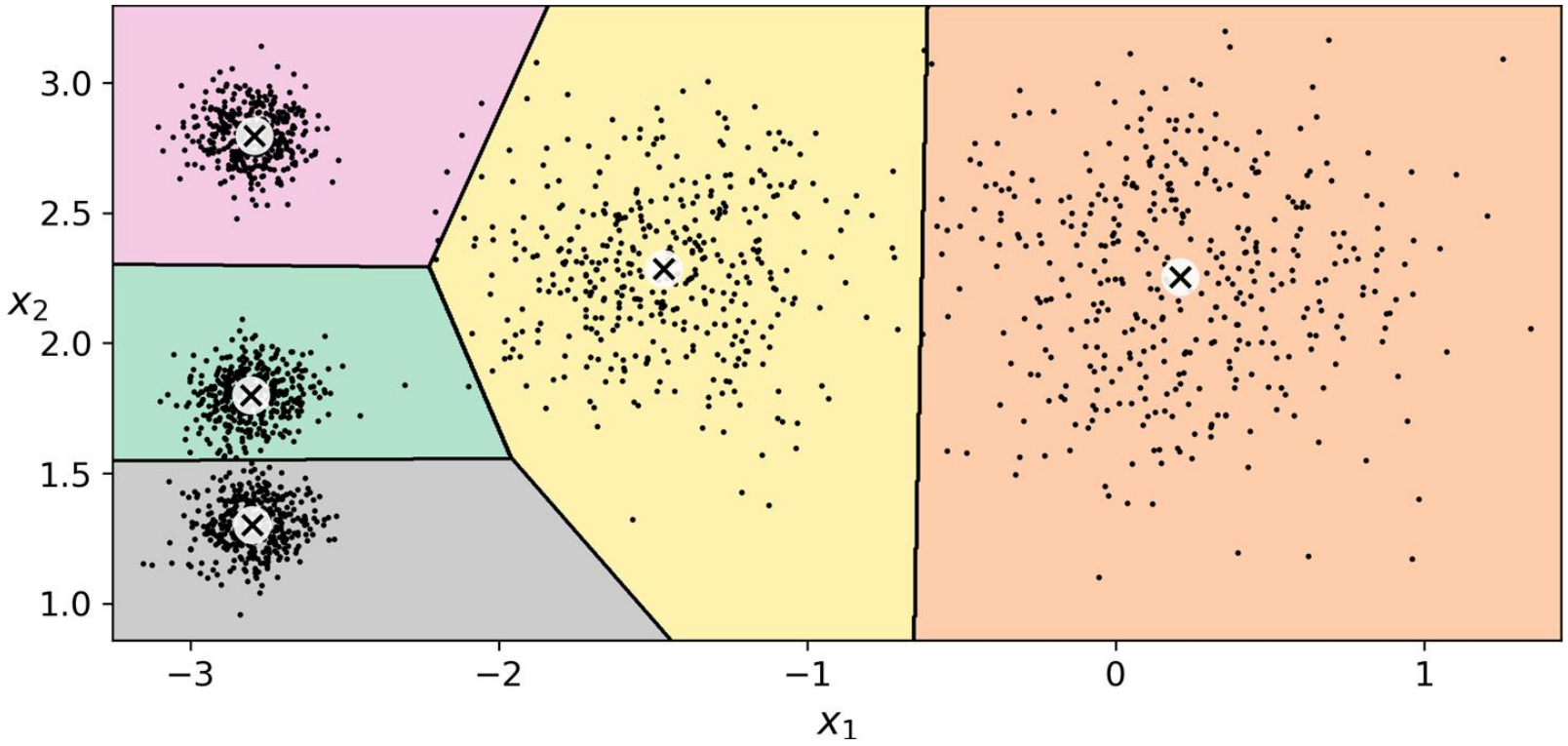
- *k* cluster “center” points are created at random locations.
- For each observation:
  - a. The distance between each observation and the *k* center points is calculated.
  - b. The observation is assigned to the cluster of the nearest center point.
- The center points are moved to the means (i.e., centers) of their respective clusters.
- Steps 2 and 3 are repeated until no observation changes in cluster membership

The K-Means algorithm



K-Means

K-Means decision boundaries (Voronoi tessellation)



DBSCAN

- This algorithm defines clusters as continuous regions of high density.
- how it works:
  - For each instance, the algorithm counts how many instances are located within a small distance ε (epsilon) from it. This region is called the instance's *ε-neighborhood*.
  - If an instance has at least `min_samples` instances in its ε-neighborhood (including itself), then it is considered a *core instance*. In other words, core instances are those that are located in dense regions.
  - All instances in the neighborhood of a core instance belong to the same cluster. This neighborhood may include other core instances; therefore, a long sequence of neighboring core instances forms a single cluster.
  - Any instance that is not a core instance and does not have one in its neighborhood is considered an anomaly.

Question

Given the points A(3, 7), B(4, 6), C(5, 5), D(6, 4), E(7, 3), F(6, 2), G(7, 2) and H(8, 4), Find the core points and outliers using DBSCAN. Take Eps = 2.5 and MinPts = 3.

Data Points	X	Y
A	3	7
B	4	6
C	5	5
D	6	4
E	7	3
F	6	2
G	7	2
H	8	4

core points, outliers and clusters

Step 1:

Calculating distance between data point A and other data points as:

$d(A, B) = \sqrt{(4-3)^2 + (6-7)^2} = \sqrt{(1)^2 + (-1)^2} = \sqrt{1+1} = \sqrt{2} = 1.4142$

$d(A, C) = \sqrt{(5-3)^2 + (5-7)^2} = \sqrt{(2)^2 + (-2)^2} = \sqrt{4+4} = \sqrt{8} = 2.8284$

$d(A, D) = \sqrt{(6-3)^2 + (4-7)^2} = \sqrt{(3)^2 + (-3)^2} = \sqrt{9+9} = \sqrt{18} = 4.2426$

$d(A, E) = \sqrt{(7-3)^2 + (3-7)^2} = \sqrt{(4)^2 + (-4)^2} = \sqrt{16+16} = \sqrt{32} = 5.6569$

$d(A, F) = \sqrt{(6-3)^2 + (2-7)^2} = \sqrt{(3)^2 + (-5)^2} = \sqrt{9+25} = \sqrt{34} = 5.8310$

$d(A, G) = \sqrt{(7-3)^2 + (2-7)^2} = \sqrt{(4)^2 + (-5)^2} = \sqrt{16+25} = \sqrt{41} = 6.4031$

$d(A, H) = \sqrt{(8-3)^2 + (4-7)^2} = \sqrt{(5)^2 + (-3)^2} = \sqrt{25+9} = \sqrt{34} = 5.8310$

Calculating distance between data point B and other data points as:

$d(B, C) = \sqrt{(5-4)^2 + (5-6)^2} = \sqrt{(1)^2 + (-1)^2} = \sqrt{1+1} = \sqrt{2} = 1.4142$

$d(B, D) = \sqrt{(6-4)^2 + (4-6)^2} = \sqrt{(2)^2 + (-2)^2} = \sqrt{4+4} = \sqrt{8} = 2.8284$

$d(B, E) = \sqrt{(7-4)^2 + (3-6)^2} = \sqrt{(3)^2 + (-3)^2} = \sqrt{9+9} = \sqrt{18} = 4.2426$

$d(B, F) = \sqrt{(6-4)^2 + (2-6)^2} = \sqrt{(2)^2 + (-4)^2} = \sqrt{4+16} = \sqrt{20} = 4.4721$

$d(B, G) = \sqrt{(7-4)^2 + (2-6)^2} = \sqrt{(3)^2 + (-4)^2} = \sqrt{9+16} = \sqrt{25} = 5$

$d(B, H) = \sqrt{(8-4)^2 + (4-6)^2} = \sqrt{(4)^2 + (-2)^2} = \sqrt{16+4} = \sqrt{20} = 4.4721$

Calculating distance between data point C and other data points as:

$d(C, D) = \sqrt{(6-5)^2 + (4-5)^2} = \sqrt{(1)^2 + (-1)^2} = \sqrt{1+1} = \sqrt{2} = 1.4142$

$d(C, E) = \sqrt{(7-5)^2 + (3-5)^2} = \sqrt{(2)^2 + (-2)^2} = \sqrt{4+4} = \sqrt{8} = 2.8284$

$d(C, F) = \sqrt{(6-5)^2 + (2-5)^2} = \sqrt{(1)^2 + (-3)^2} = \sqrt{1+9} = \sqrt{10} = 3.1623$

$d(C, G) = \sqrt{(7-5)^2 + (2-5)^2} = \sqrt{(2)^2 + (-3)^2} = \sqrt{4+9} = \sqrt{13} = 3.6056$

$d(C, H) = \sqrt{(8-5)^2 + (4-5)^2} = \sqrt{(3)^2 + (-1)^2} = \sqrt{9+1} = \sqrt{10} = 3.1623$

Calculating distance between data point D and other data points as:

$d(D, E) = \sqrt{(7-6)^2 + (3-4)^2} = \sqrt{(1)^2 + (-1)^2} = \sqrt{1+1} = \sqrt{2} = 1.4142$

$d(D, F) = \sqrt{(6-6)^2 + (2-4)^2} = \sqrt{(0)^2 + (-2)^2} = \sqrt{0+4} = \sqrt{4} = 2$

$d(D, G) = \sqrt{(7-6)^2 + (2-4)^2} = \sqrt{(1)^2 + (-2)^2} = \sqrt{1+4} = \sqrt{5} = 2.2361$

$d(D, H) = \sqrt{(8-6)^2 + (4-4)^2} = \sqrt{(2)^2 + (0)^2} = \sqrt{4+0} = \sqrt{4} = 2$

Calculating distance between data point E and other data points as:

$d(E, F) = \sqrt{(6-7)^2 + (2-3)^2} = \sqrt{(-1)^2 + (-1)^2} = \sqrt{1+1} = \sqrt{2} = 1.4142$

$d(E, G) = \sqrt{(7-7)^2 + (2-3)^2} = \sqrt{(0)^2 + (-1)^2} = \sqrt{0+1} = \sqrt{1} = 1$

$d(E, H) = \sqrt{(8-7)^2 + (4-3)^2} = \sqrt{(1)^2 + (1)^2} = \sqrt{1+1} = \sqrt{2} = 1.4142$

Calculating distance between data point F and other data points as:

$d(F, G) = \sqrt{(7-6)^2 + (2-2)^2} = \sqrt{(1)^2 + (0)^2} = \sqrt{1+0} = \sqrt{1} = 1$

$d(F, H) = \sqrt{(8-6)^2 + (4-2)^2} = \sqrt{(2)^2 + (2)^2} = \sqrt{4+4} = \sqrt{8} = 2.8284$

Calculating distance between data point G and other data points as:

$d(G, H) = \sqrt{(8-7)^2 + (4-2)^2} = \sqrt{(1)^2 + (2)^2} = \sqrt{1+4} = \sqrt{5} = 2.2361$

The final distance matrix becomes as shown below:

Data Points	A	B	C	D	E	F	G	H
A	0	1.4142	2.8284	4.2426	5.6569	5.831	6.4031	5.831
B		0	1.4142	2.8284	4.2426	4.4721	5	4.4721
C			0	1.4142	2.8284	3.1623	3.6056	3.1623
D				0	1.4142	2	2.2361	2
E					0	1.4142	1	1.4142
F						0	1	2.8284
G							0	2.2361
H								0

Step 2: Now, finding all the data points that lie in the *Eps-neighborhood* of each data points. That is, put all the points in the neighborhood set of each data point whose distance is <=2.5.

N(A) = {B}; — — — — — — -→ because distance of B is <= 2.5 with A

N(B) = {A, C}; — — — — — → because distance of A and C is <= 2.5 with B

N(C) = {B, D}; — — — — —→ because distance of B and D is <=2.5 with C

N(D) = {C, E, F, G, H}; — → because distance of C, E, F,G and H is <=2.5 with D

N(E) = {D, F, G, H}; — — → because distance of D, F, G and H is <=2.5 with E

N(F) = {D, E, G}; — — — — → because distance of D, E and G is <=2.5 with F

N(G) = {D, E, F, H}; — — -→ because distance of D, E, F and H is <=2.5 with G

N(H) = {D, E, G}; — — — — → because distance of D, E and G is <=2.5 with H

Here, data points A, B and C have neighbors <= **MinPts (i.e. 3)** so can't be considered as core points. Since they belong to the neighborhood of other data points, hence there exist *no outliers* in the given set of data points.
Data points D, E, F, G and H have neighbors >= **MinPts (i.e. 3)** and hence are the *core data points*.

Association Rule Mining

- Unsupervised Non-linear algorithm to uncover how the items are associated with each other.
- Frequent Mining shows which items appear together in a transaction or relation.
- Majorly used by retailers, grocery stores, an online marketplace that has a large transactional database.
- Common ways to measure association:
  - Support
  - Confidence
  - Lift

Association Rule Mining Measures

- **Support** says how popular an item is, as measured in the proportion of transactions in which an item set appears.

$$\text{Support}(A \Rightarrow B) = P(A \cup B)$$

- **Confidence** says how likely item Y is purchased when item X is purchased, expressed as {X -> Y}.

$$\text{Confidence}(A => B) = P(B \mid A) = \frac{P(A \cup B)}{P(A)}$$

- **Lift** says how likely item Y is purchased when item X is purchased while controlling for how popular item Y is.

$$\text{Lift}(A \Rightarrow B) = \frac{\text{Confidence}(A \Rightarrow B)}{P(B)} = \frac{P(A \cup B)}{P(A)P(B)}$$

Apriori Algorithm

- **The Apriori Algorithm** is a foundational method in data mining used for **discovering frequent itemsets and generating association rules.**
- Its significance lies in its ability to **identify relationships between items in large datasets** which is particularly valuable in market basket analysis.
- Eg: if a grocery store finds that customers who buy **bread** often also buy **butter**, it can use this information to optimise product placement or marketing strategies.

How the Apriori Algorithm Works?

- **Identifying frequent itemsets**
- **Creating possible item group**
- **Removing infrequent item groups**
- **Generating Association Rules**
  - using metrics like **support, confidence,** and **lift** to evaluate the strength of these relationships.

$$\text{Support}(A) = \frac{\text{Number of transactions containing itemset } A}{\text{Total number of transactions}}$$

$$\text{Confidence}(X \rightarrow Y) = \frac{\text{Support}_{(X \cup Y)}}{\text{Support}_{(X)}}$$

Transaction ID	Items
T1	Hot Dogs, Buns, Ketchup
T2	Hot Dogs, Buns
T3	Hot Dogs, Coke, Chips
T4	Chips, Coke
T5	Chips, Ketchup
T6	Hot Dogs, Coke, Chips

- Find the **frequent itemsets** and generate **association rules** on this. Assume that minimum support threshold (s = 33.33%) and minimum confident threshold (c = 60%) using Apriori algorithm.

Item set	Sup-count
Hot Dogs	4
Buns	2
Ketchup	2
Coke	3
Chips	4

Item set	Sup-count
Hot Dogs	4
Buns	2
Ketchup	2
Coke	3
Chips	4

$$\text{minimum support count} = \frac{33.33}{100} \times 6 = 2$$

Item set	Sup-count
Hot Dogs, Buns	2
Hot Dogs, Ketchup	1
Hot Dogs, Coke	2
Hot Dogs, Chips	2
Buns, Ketchup	1
Buns, Coke	0
Buns, Chips	0
Ketchup, Coke	0
Ketchup, Chips	1
Coke, Chips	3

Item set	Sup-count
Hot Dogs, Buns	2
Hot Dogs, Coke	2
Hot Dogs, Chips	2
Coke, Chips	3

Item set	Sup-count
Hot Dogs, Buns, Coke	0
Hot Dogs, Buns, Chips	0
Hot Dogs, Coke, Chips	2

Item set	Sup-count
Hot Dogs, Coke, Chips	2

Frequent Itemset (I) = {Hot Dogs, Coke, Chips}

Association rules

- [Hot Dogs^Coke]=>[Chips]
- confidence = sup(Hot Dogs^Coke^Chips)/sup(Hot Dogs^Coke)

  = 2/2*100

  =100% **//Selected**
- [Hot Dogs^Chips]=>[Coke]
- confidence = sup(Hot Dogs^Coke^Chips)/sup(Hot Dogs^Chips)

  = 2/2*100

  =100% **//Selected**

- [Coke^Chips]=>[Hot Dogs]
- confidence = sup(Hot Dogs^Coke^Chips)/sup(Coke^Chips)
- = 2/3*100
- =66.67% //**Selected**

- [Hot Dogs]=>[Coke^Chips] //confidence
- = sup(Hot Dogs^Coke^Chips)/sup(Hot Dogs)
- = 2/4*100=50% //**Rejected**

- [Coke]=>[Hot Dogs^Chips]
- confidence = sup(Hot Dogs^Coke^Chips)/sup(Coke)
- = 2/3*100
- =66.67% **//Selected**

- [Chips]=>[Hot Dogs^Coke]
- confidence = sup(Hot Dogs^Coke^Chips)/sup(Chips)
- = 2/4*100=50% **//Rejected**

Example 2

Transaction ID	Items
T1	Hot Dogs, Buns, Ketchup
T2	Hot Dogs, Buns
T3	Hot Dogs, Coke, Chips
T4	Chips, Coke
T5	Chips, Ketchup
T6	Hot Dogs, Coke, Chips

Find the **frequent itemsets** on this. Assume that minimum support (s = 3)

Drawbacks of the Apriori Algorithm

The two primary are:

1. At each step, candidate sets have to be built.
2. To build the candidate sets, the algorithm has to repeatedly scan the database.

Frequent Pattern Growth Algorithm

- It overcomes the disadvantages of the Apriori algorithm by storing all the transactions in a **Tree Data Structure**.

How it works

- **Data Compression**
  - Smaller structure called the **Frequent Pattern Tree (FP-Tree) -** item sets (collections of items) and their frequencies,
- **Mining the Tree**
  - Identify patterns that appear frequently, based on a minimum support threshold.
- **Generating Patterns**:
  - generates the frequent patterns (itemsets) and the rules that describe relationships between items

Example

Transaction ID	Items
T1	{E,K,M,N,O,Y}
T2	{D,E,K,N,O,Y}
T3	{A,E,K,M}
T4	{K,M,Y}
T5	{C,E,I,K,O,O}

Item	Frequency
A	1
C	2
D	1
E	4
I	1
K	5
M	3
N	2
O	4
U	1
Y	3

Let the minimum support be 3

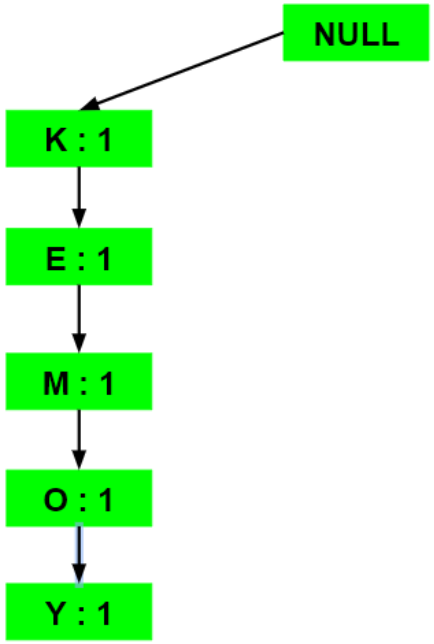
Frequent Pattern set :
L = {K : 5, E : 4, M : 3, O : 4, Y : 3}

Ordered-Item set

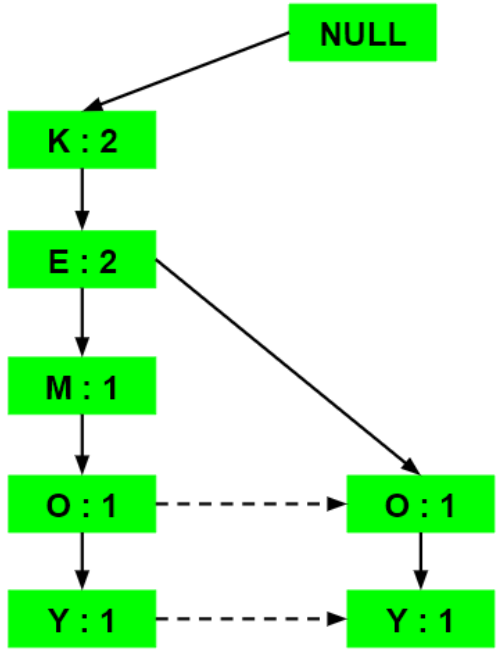
Transaction ID	Items	Ordered-Item-Set
T1	{E,K,M,N,O,Y}	{K,E,M,O,Y}
T2	{D,E,K,N,O,Y}	{K,E,O,Y}
T3	{A,E,K,M}	{K,E,M}
T4	{C,K,M,U,Y}	{K,M,Y}
T5	{C,E,I,K,O,O}	{K,E,O}

Ordered-Item sets are inserted into a Tree Data Structure

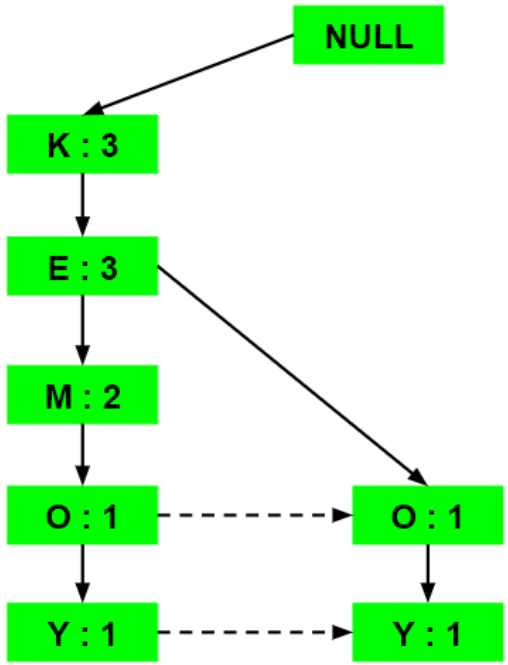
a) **Inserting the set {K, E, M, O, Y}:**



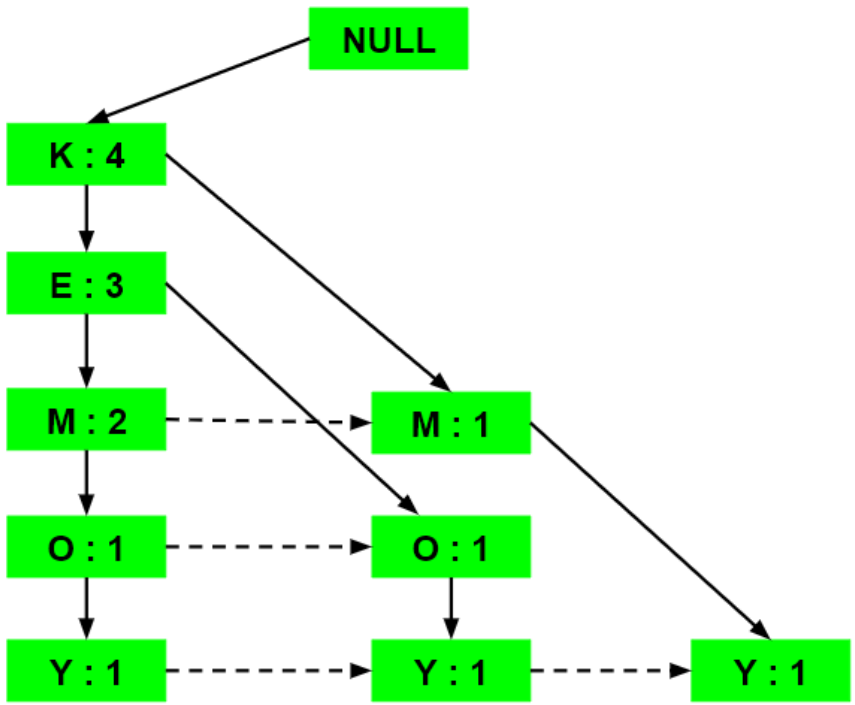
b) **Inserting the set {K, E, O, Y}:**



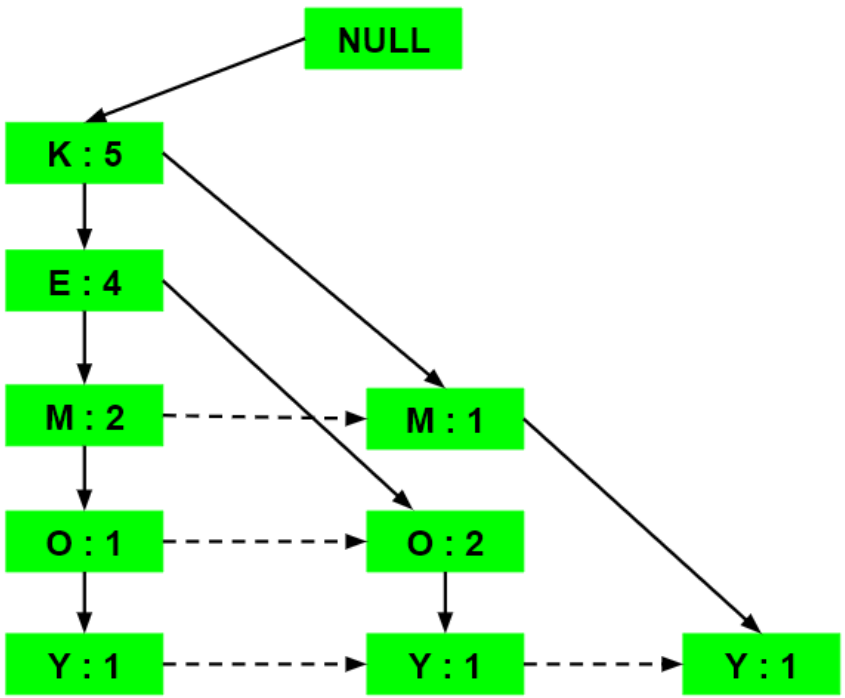
c) **Inserting the set {K, E, M}:**



d) **Inserting the set {K, M, Y}:**



e) **Inserting the set {K, E, O}:**



Conditional Pattern Base

Items	Conditional Pattern Base
Y	{{K,E,M,O : 1}, {K,E,O : 1}, {K,M : 1}}
O	{{K,E,M : 1}, {K,E : 2}}
M	{{K,E : 2}, {K : 1}}
E	{K : 4}
K	

Conditional Frequent Pattern Tree is built

Items	Conditional Pattern Base	Conditional Frequent Pattern Tree
Y	{{K,E,M,O : 1}, {K,E,O : 1}, {K,M : 1}}	{K : 3}
O	{{K,E,M : 1}, {K,E : 2}}	{K,E : 3}
M	{{K,E : 2}, {K : 1}}	{K : 3}
E	{K : 4}	{K : 4}
K		

Frequent Pattern rules

Items	Frequent Pattern Generated
Y	{<K,Y : 3>}
O	{<K,O : 3>, <E,O : 3>, <E,K,O : 3>}
M	{<K,M : 3>}
E	{<E,K : 4>}
K	

Anomaly Detection

- **Anomaly Detection** is the technique of identifying rare events or observations which can raise suspicions by being statistically different from the rest of the observations.
- Credit card fraud, failing machine in a server, a cyber attack, etc.
- An anomaly can be broadly categorised into three categories –
  - **Point Anomaly:** A tuple in a dataset is said to be a Point Anomaly if it is far off from the rest of the data.
  - **Contextual Anomaly:** An observation is a Contextual Anomaly if it is an anomaly because of the context of the observation.
  - **Collective Anomaly:** A set of data instances helps in finding an anomaly.

Anomaly detection using Machine Learning

- **Supervised Anomaly Detection**
  - This method requires a labelled dataset containing both normal and anomalous samples to construct a predictive model to classify future data points.
- **Unsupervised Anomaly Detection**
  - This method does require any training data and instead assumes two things about the data ie Only a small percentage of data is anomalous and Any anomaly is statistically different from the normal samples.
  - Based on the above assumptions, the data is then clustered using a similarity measure and the data points which are far off from the cluster are considered to be anomalies.