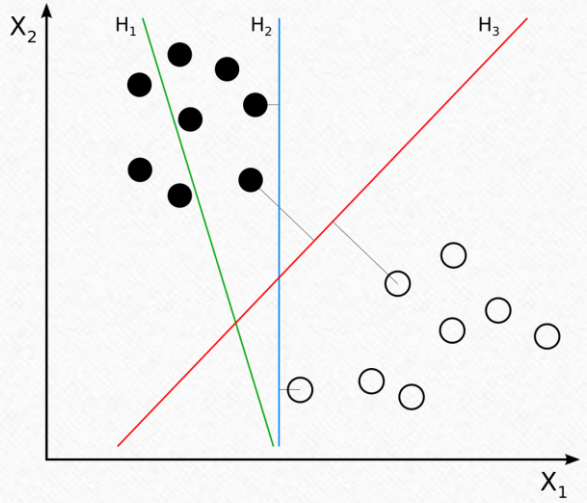
Artificial Intelligence and Machine Learning
Lecture #7

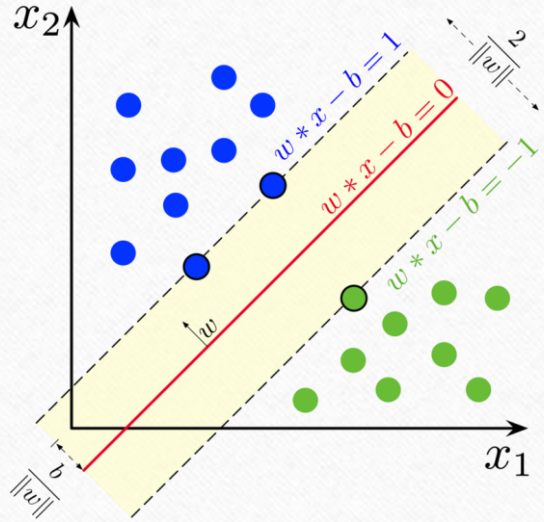
Amin Noroozi
University of Wolverhampton
✉ a.noroozifakhabi@wlv.ac.uk
https://www.linkedin.com/in/amin-n-148350218/

Classification part 3

Classification

- **SVM**
- The objective is to find a hyperplane in an n-dimensional space that separates the data points to their potential classes. The hyperplane should be positioned with the maximum distance to the data points
- The data points with the minimum distance to the hyperplane are called Support Vectors





Classification

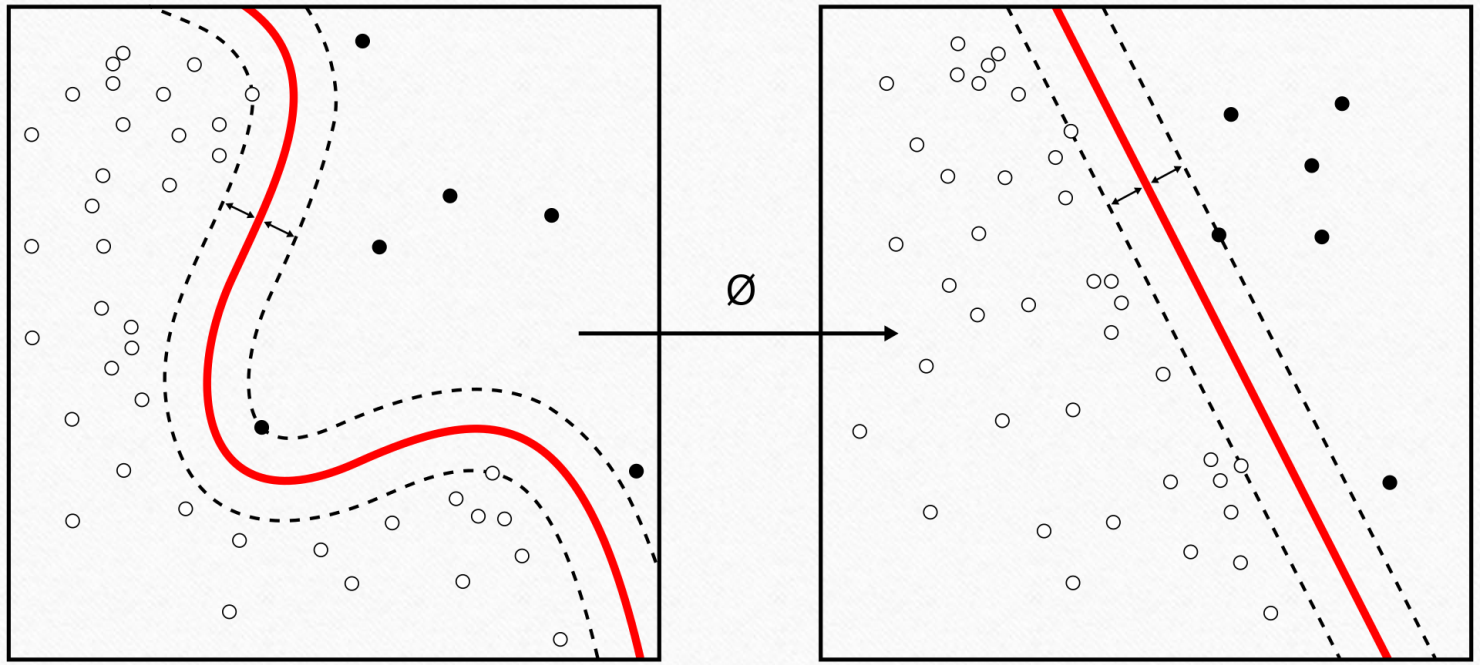
- **SVM**
- The original maximum-margin hyperplane algorithm proposed by Vapnik in 1963 constructed a linear classifier that works when data are linearly separable (using a line or hyperplane)

- For data that are not linearly separable, we need to use a kernel function and transform data to another space where they are linearly separable. This is called the kernel trick.

- There are different kernel functions including linear function, Polynomial function, Radial basis function (RBF), and Sigmoid function.

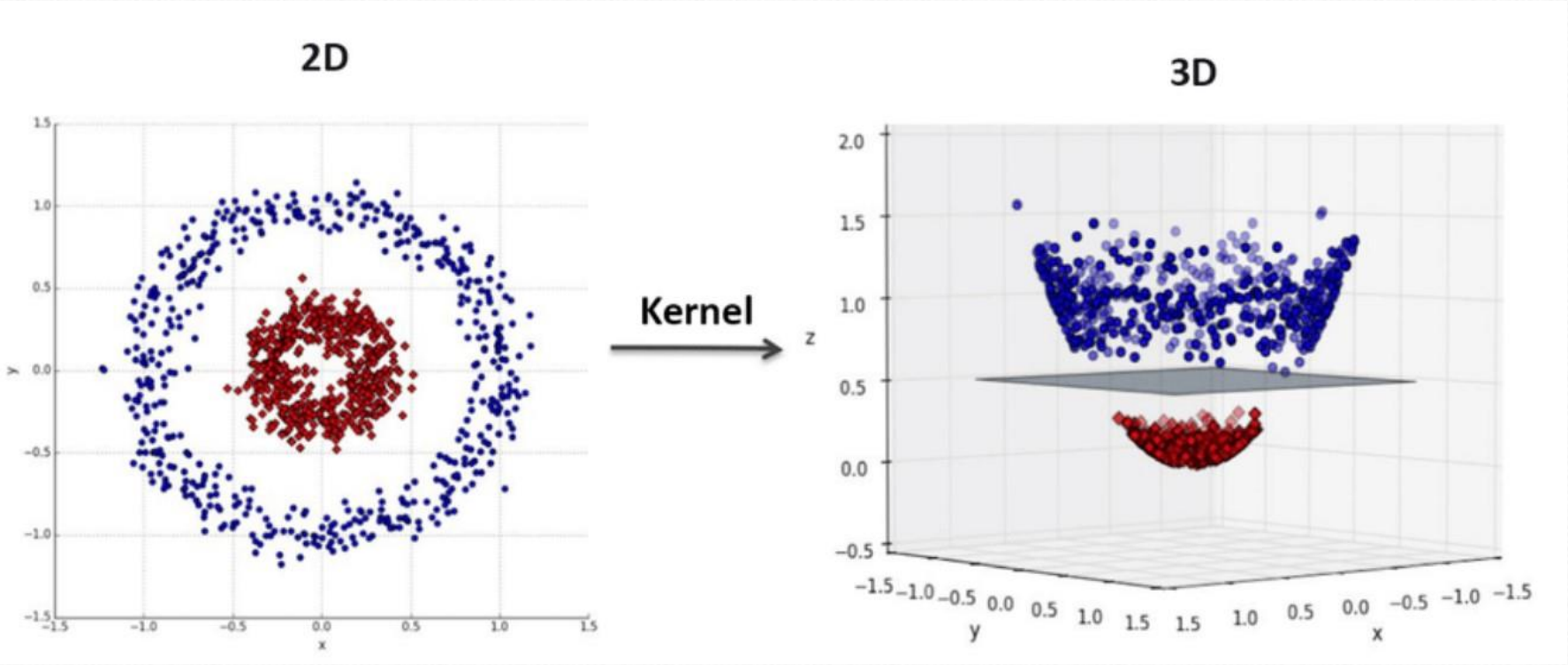
Classification

- **SVM**



Classification

- **SVM**



Classification

- **SVM**

- There are different kernel functions, including linear function, Polynomial function, Radial basis function (RBF), and Sigmoid function.

- The kernel type is a hyperparameter and should be selected using a trial and error procedure.

- The SVM classification is used for binary classification by definition.

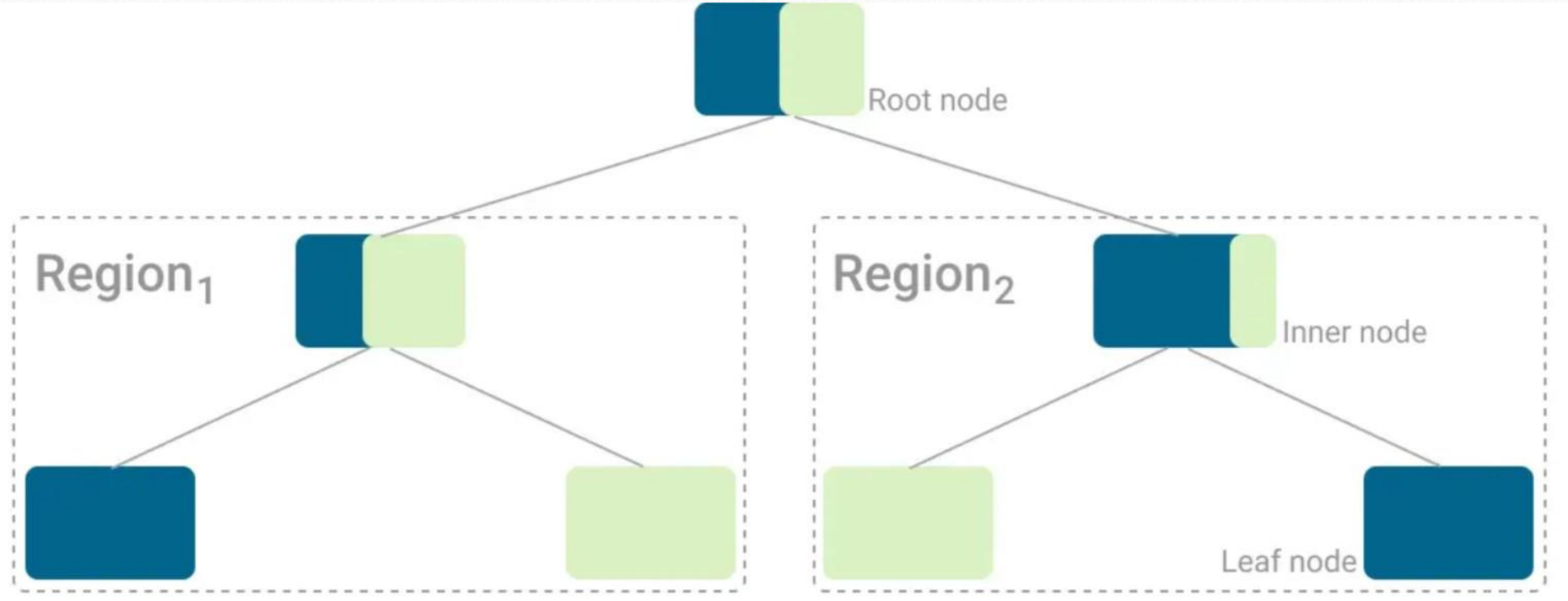
Classification

- **Decision tree**

- Decision Tree is a Supervised Machine Learning Algorithm that uses a set of rules to make decisions, similar to how humans make decisions.
- The intuition behind Decision Trees is that you use the dataset features to create yes/no questions and continually split the dataset until you isolate all data points belonging to each class.
- Every time you ask a question you're adding a **node** to the tree. And the first node is called the **root node**. Each node is connected to other nodes using **branches**.
- If you decide to stop the process after a split, the last nodes created are called **leaf nodes**.
- Every time you answer a question, you're also creating branches and segmenting the feature space into disjoint **regions**.

Classification

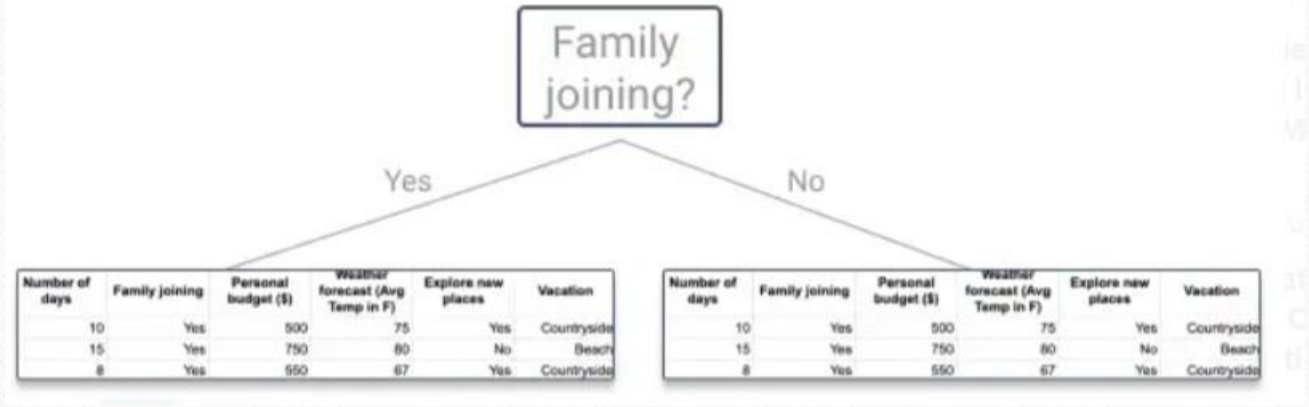
- **Decision tree**



Classification

- **Decision tree**

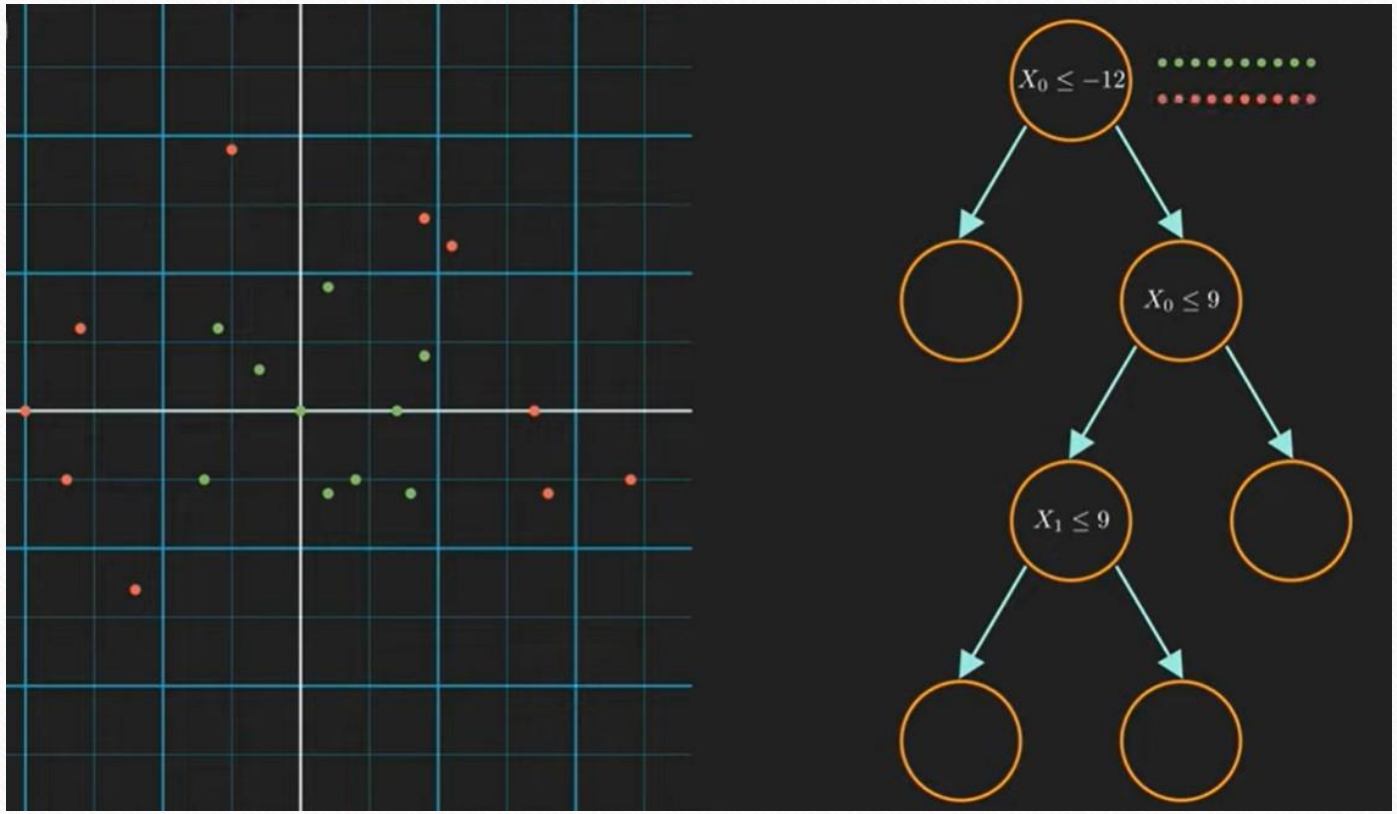
Number of days	Family joining	Personal budget ($)	Weather forecast (Avg Temp in F)	Explore new places	Vacation
10	Yes	500	75	Yes	Countryside
7	No	1000	78	Yes	Countryside
15	Yes	750	80	No	Beach
8	Yes	550	67	Yes	Countryside
10	No	800	73	No	Beach



Classification

- **Decision tree**

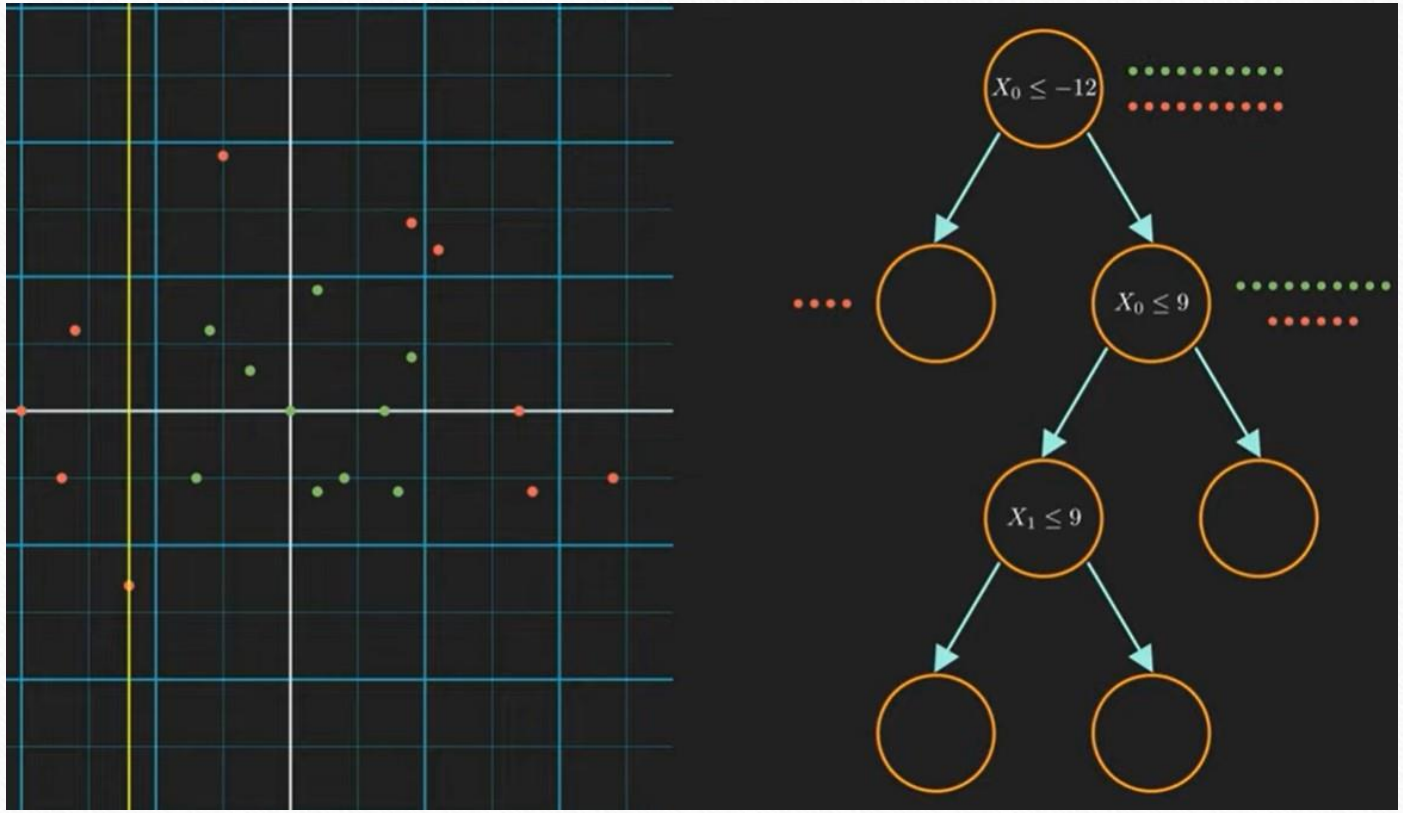
Example



Classification

- **Decision tree**

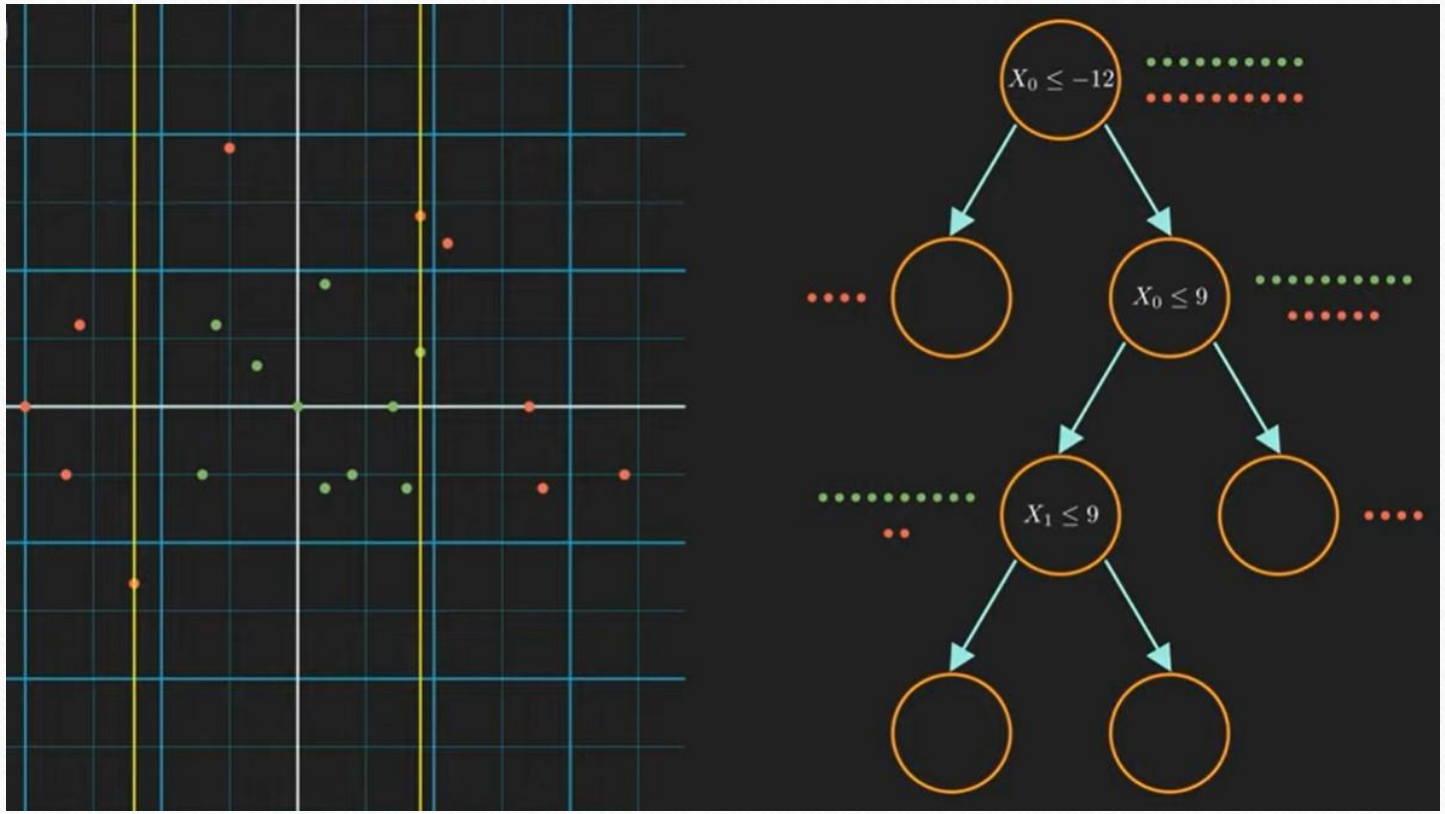
Example



Classification

- **Decision tree**

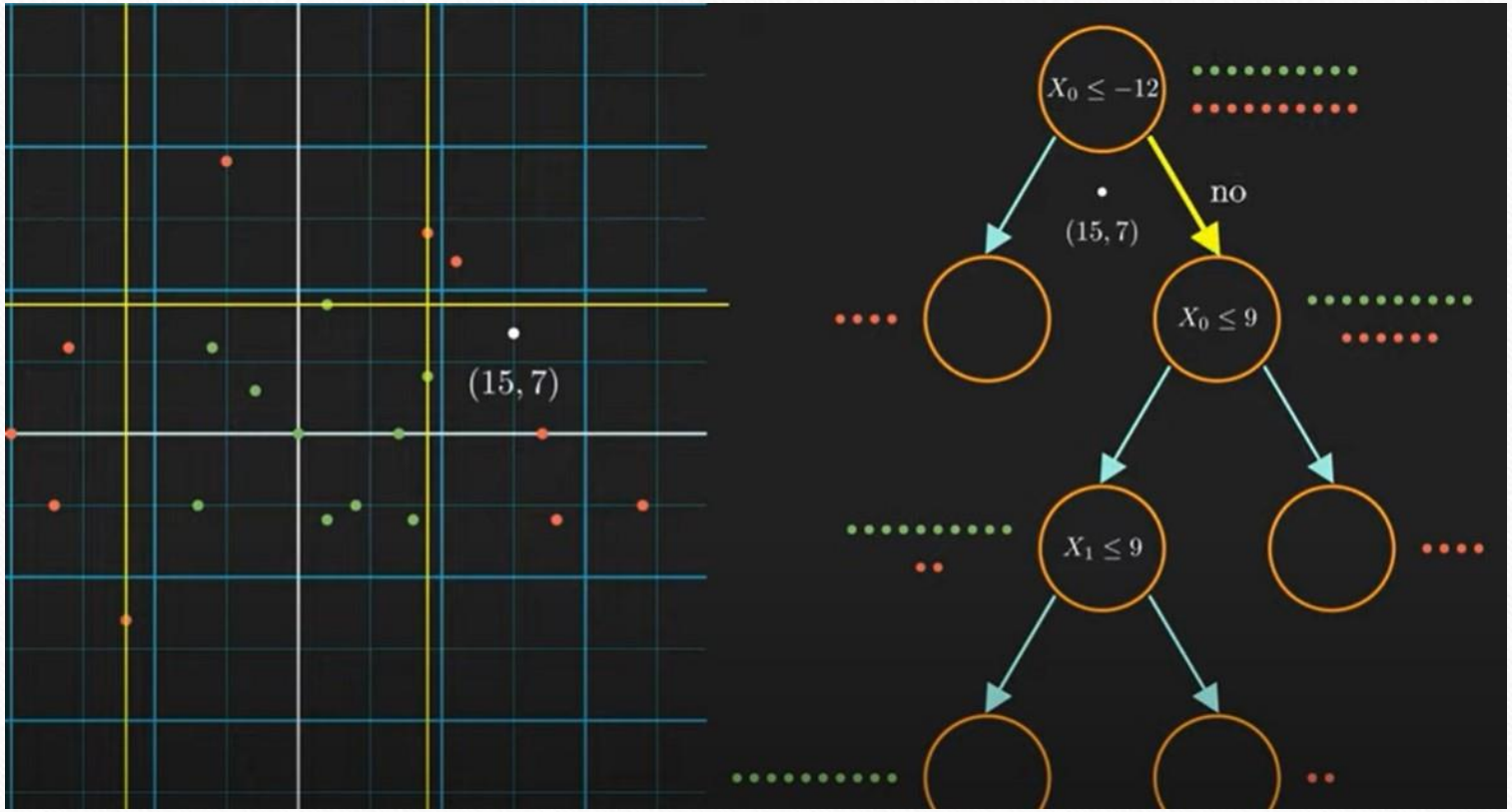
Example



Classification

- **Decision tree**

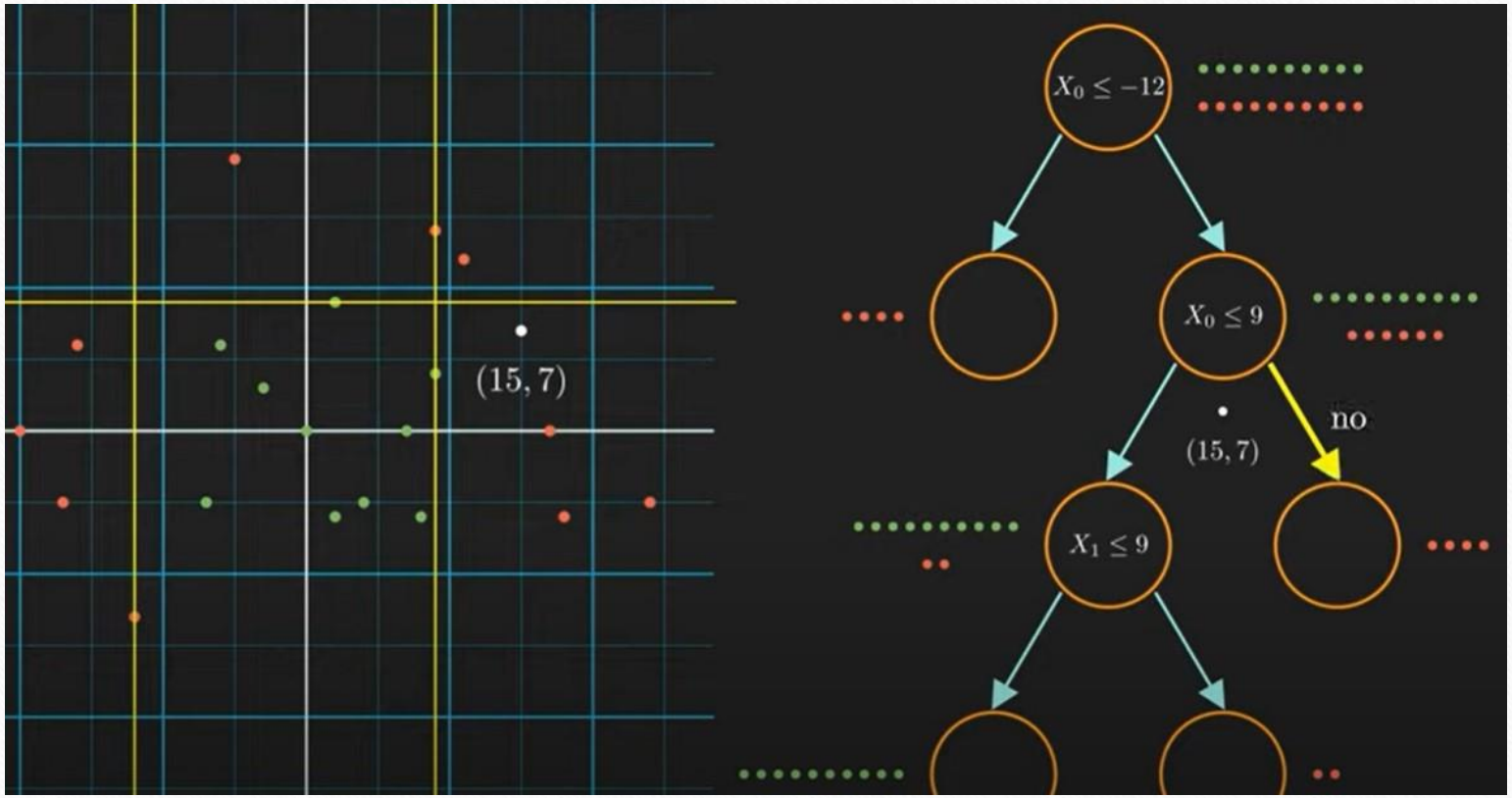
Example



Classification

- **Decision tree**

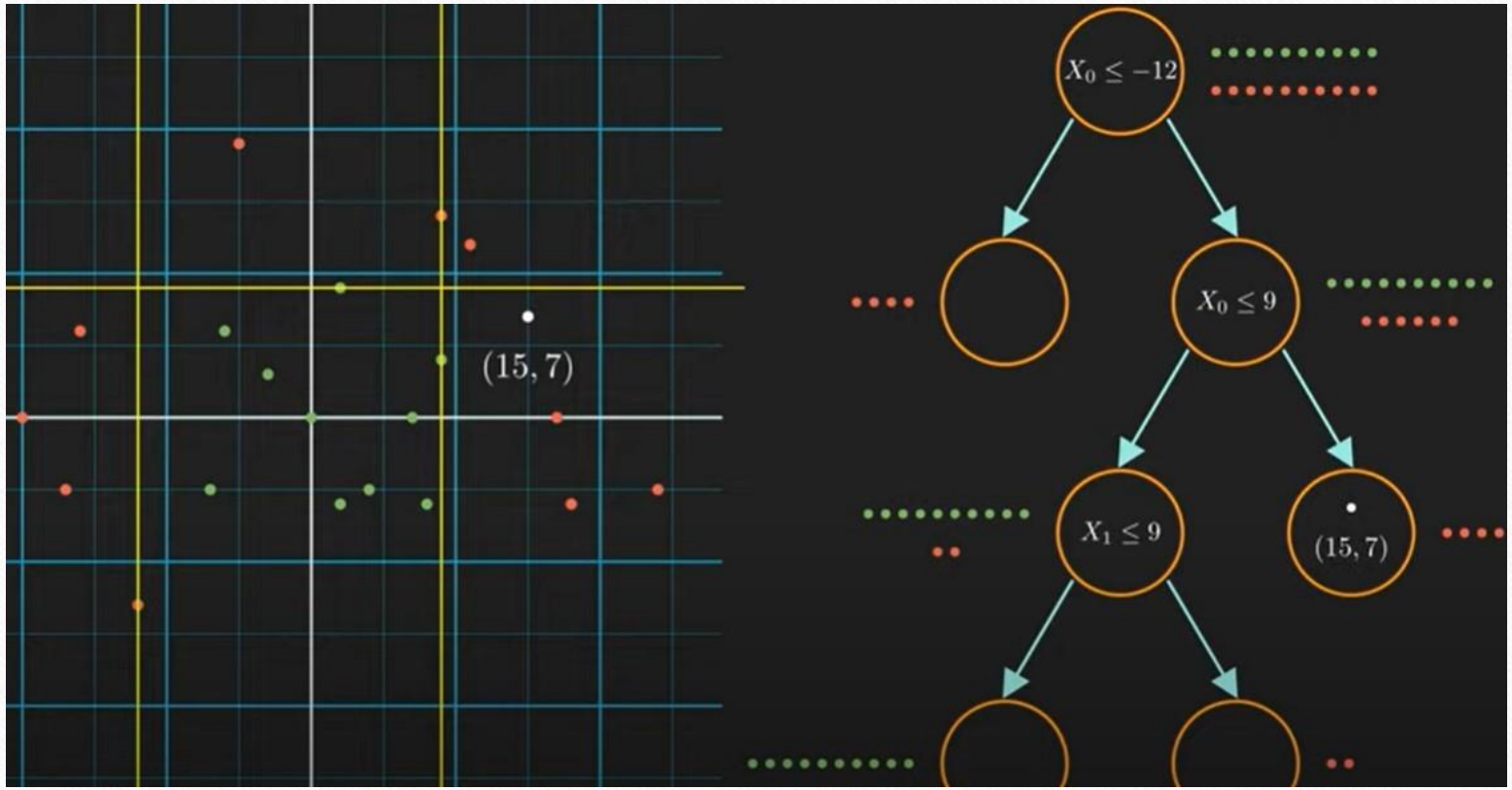
Example



Classification

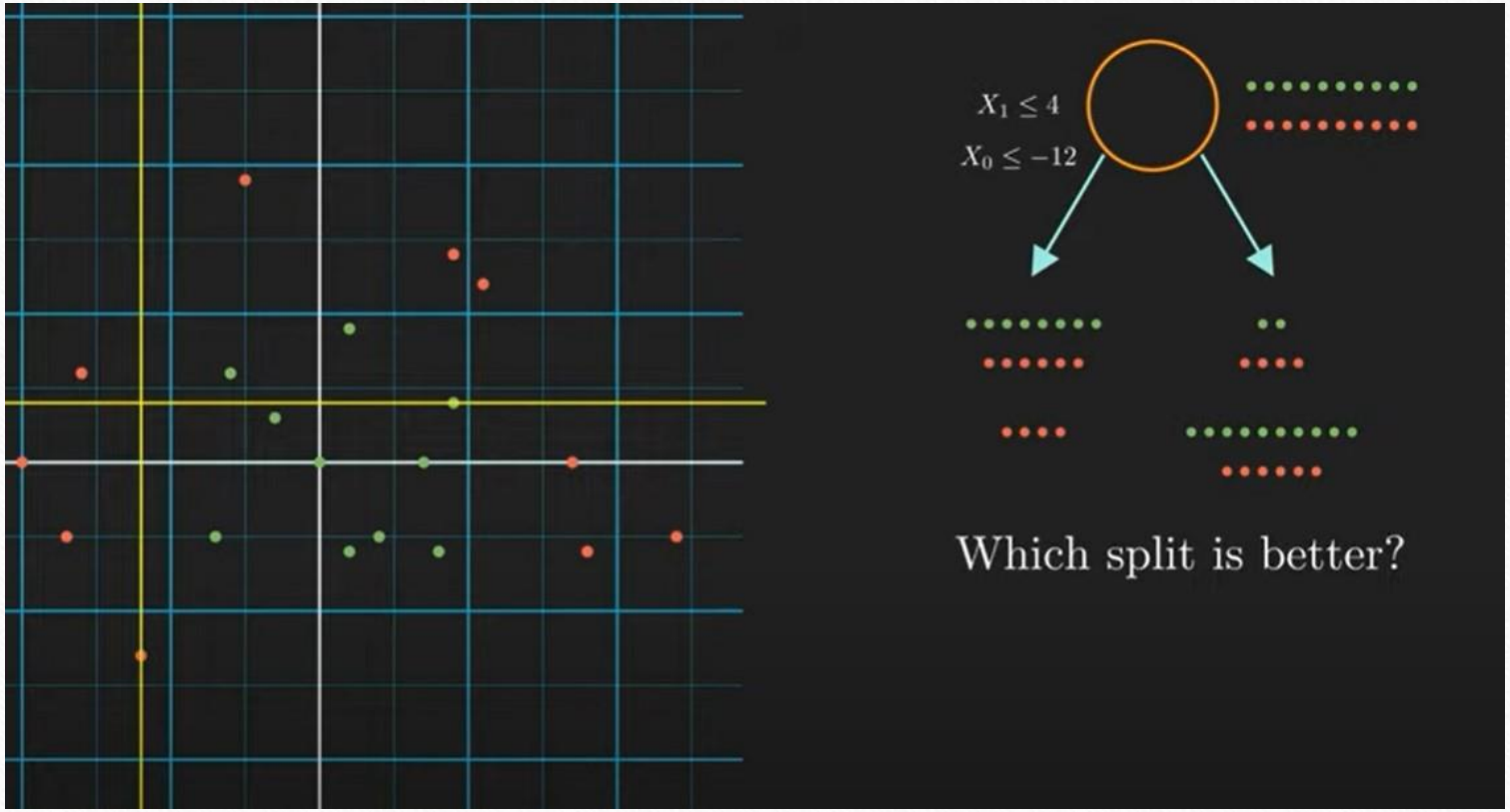
- **Decision tree**

Example



Classification

- **Decision tree**
- But how can we determine the optimal split and condition?



Classification

- **Decision tree**
- We can decide the best split by calculating the information gain resulting from each split using the following formula (it is log_2):

$$Entropy = \sum - p_i \ \log(p_i)$$

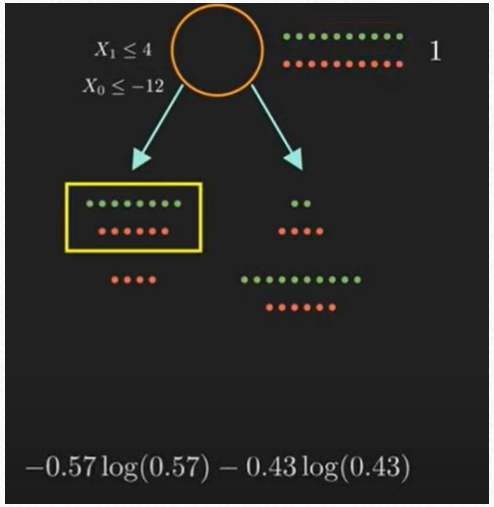
p_i = probability of class i

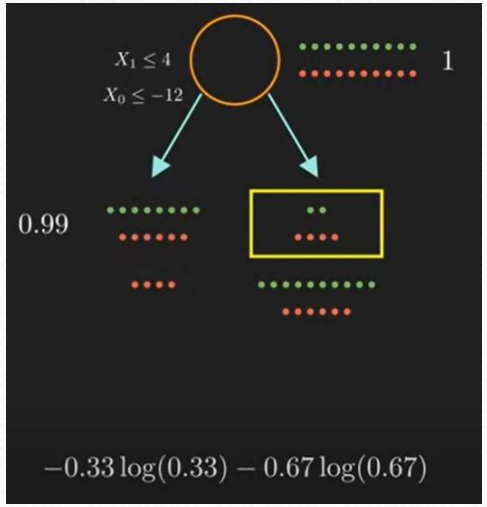
$$\text{IG} = E(parent) - \sum w_i \ E(child_i)$$

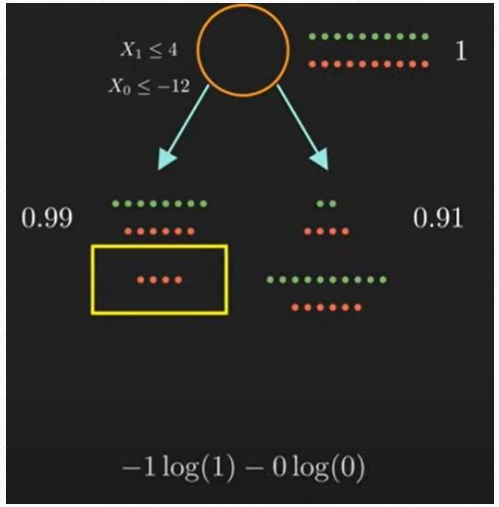
Classification

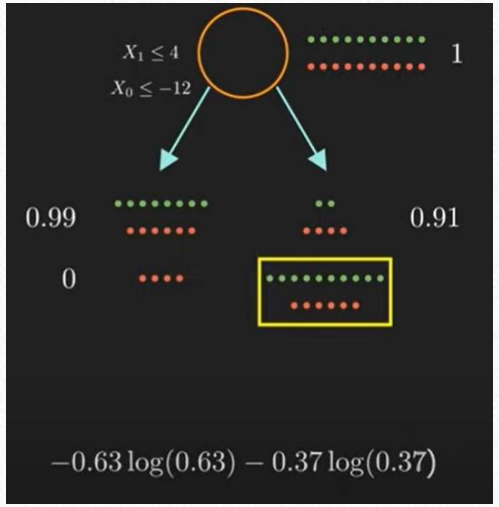
- **Decision tree**

Example





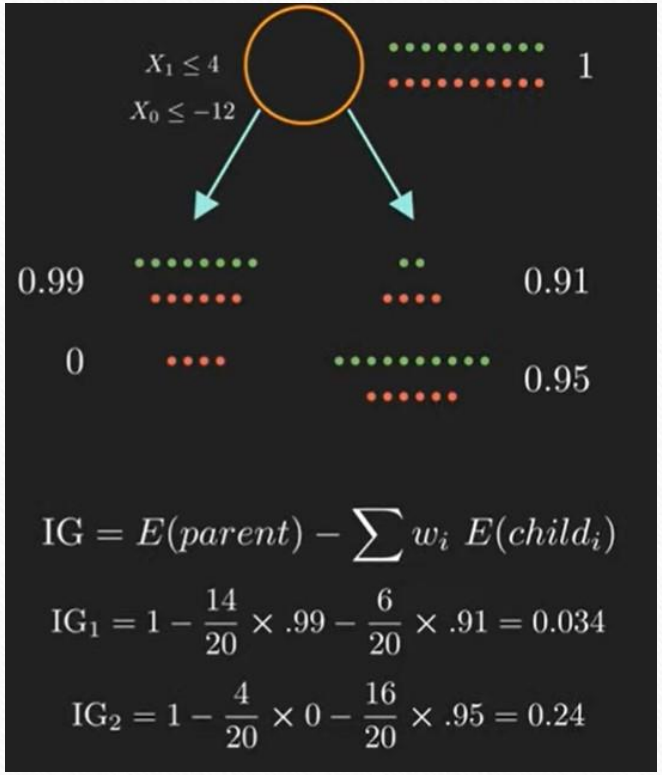


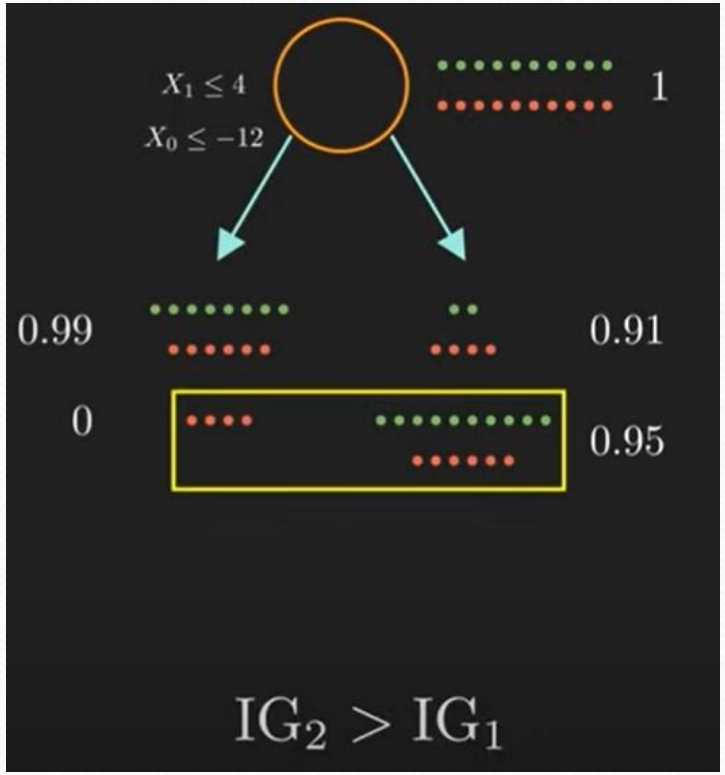


Classification

- **Decision tree**

Example





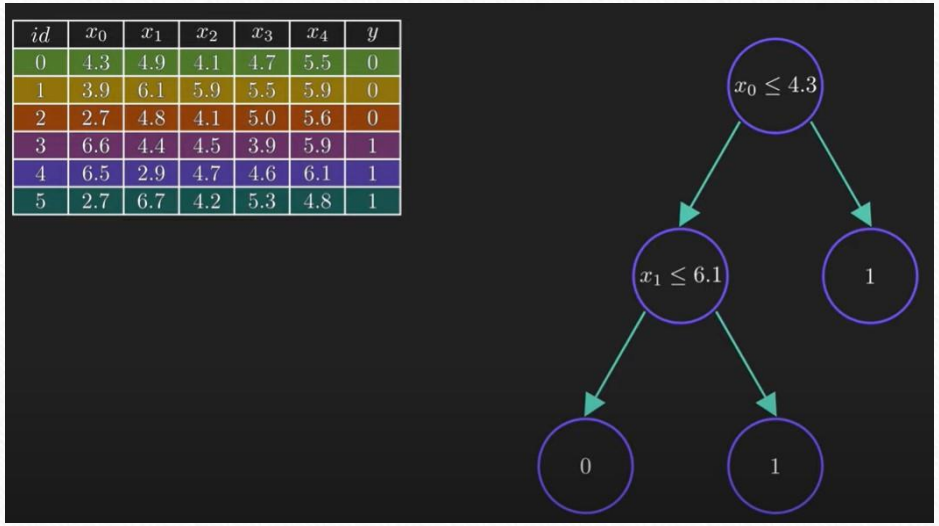
Classification

- **Decision tree**
  - In reality, the model traverses through every possible feature and feature value to find the best split.
  - A pure leaf node is a leaf node that contains data from only one class.
  - Although it's desirable to have data from only one class in each leaf node, this will not happen in most cases. If we have data from different classes in leaf nodes, we can use the **majority voting** technique to decide which class the leaf node belongs to.
  - According to the majority voting, the class of the leaf node is the same class as the class of the majority of data inside the leaf.

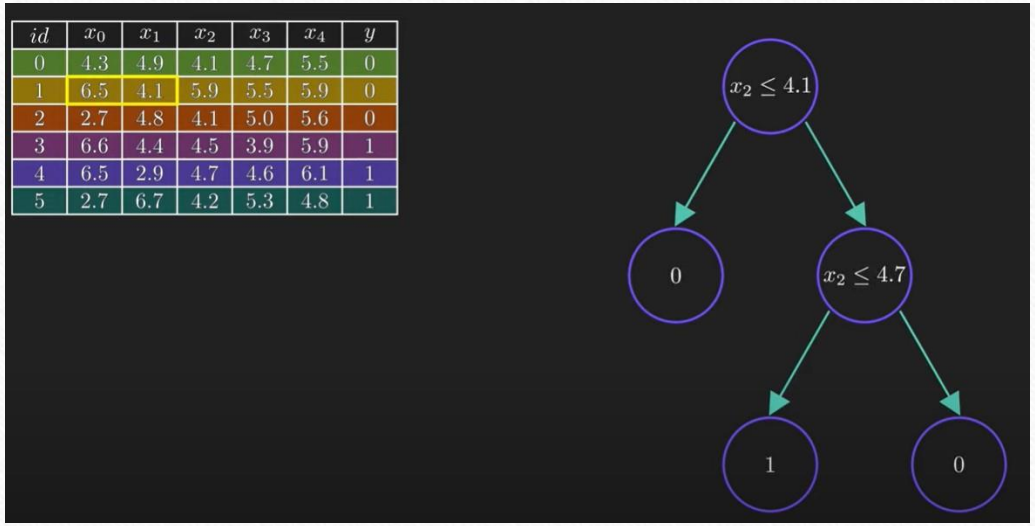
Classification

- **Random Forest**
- Decision trees are highly sensitive to the training data resulting in a high variance
- This problem will limit the generalization of the trained model

id	x_0	x_1	x_2	x_3	x_4	y
0	4.3	4.9	4.1	4.7	5.5	0
1	3.9	6.1	5.9	5.5	5.9	0
2	2.7	4.8	4.1	5.0	5.6	0
3	6.6	4.4	4.5	3.9	5.9	1
4	6.5	2.9	4.7	4.6	6.1	1
5	2.7	6.7	4.2	5.3	4.8	1

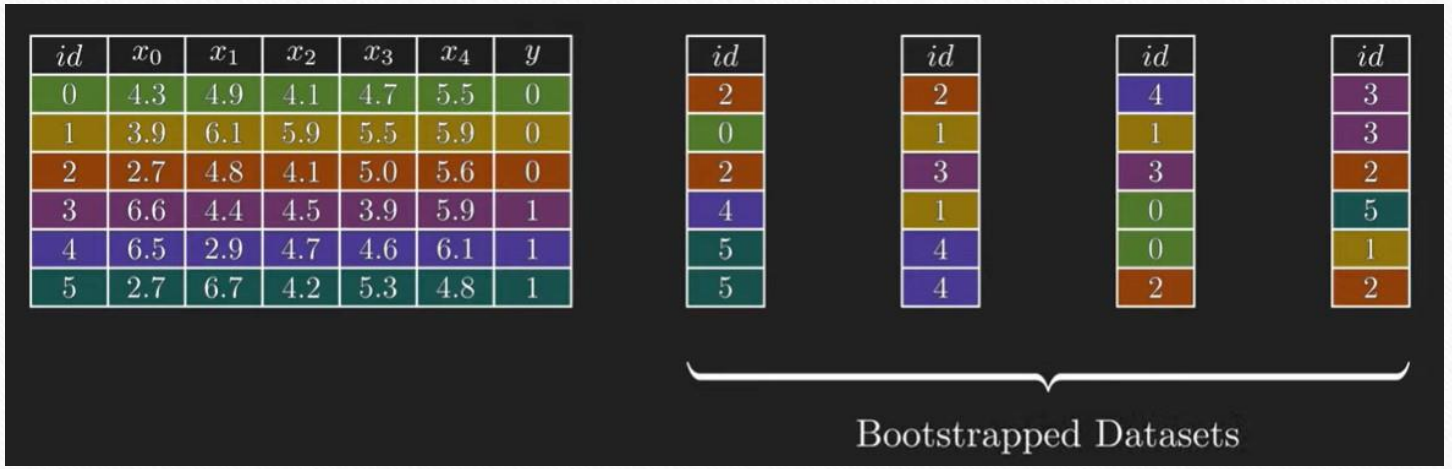


id	x_0	x_1	x_2	x_3	x_4	y
0	4.3	4.9	4.1	4.7	5.5	0
1	6.5	4.1	5.9	5.5	5.9	0
2	2.7	4.8	4.1	5.0	5.6	0
3	6.6	4.4	4.5	3.9	5.9	1
4	6.5	2.9	4.7	4.6	6.1	1
5	2.7	6.7	4.2	5.3	4.8	1



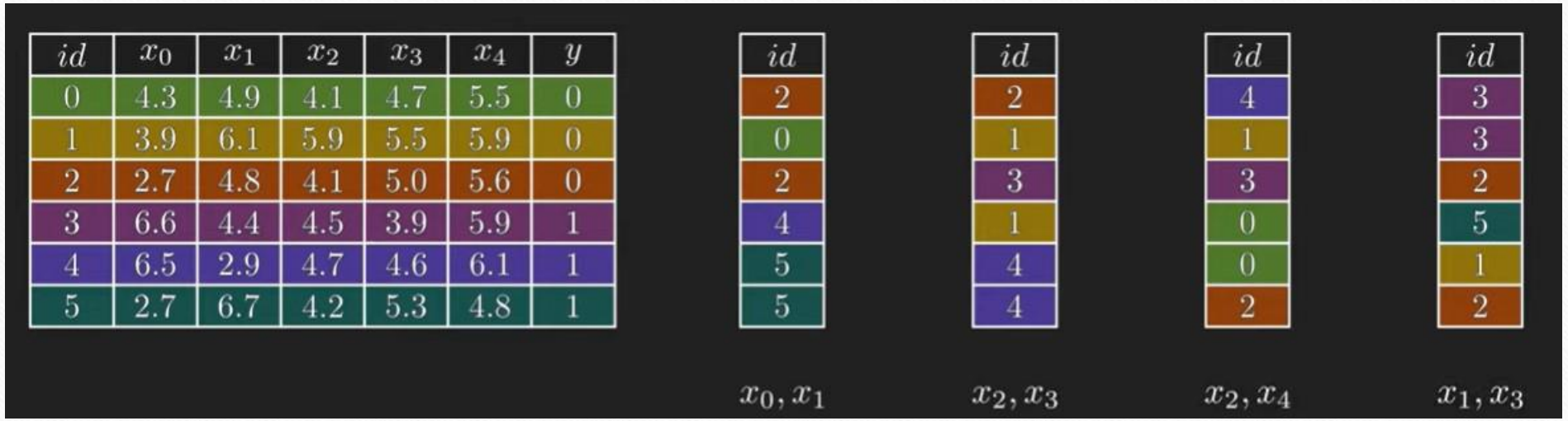
Classification

- **Random Forest**
- Random forest is a collection of multiple random decision trees
- Random forest algorithm can be summarized as follows:
- **Step1**: Choose random subsets of the original dataset with replacement so that the total number of rows in the selected subset is the same as the original dataset. This process is called bootstrapping.



Classification

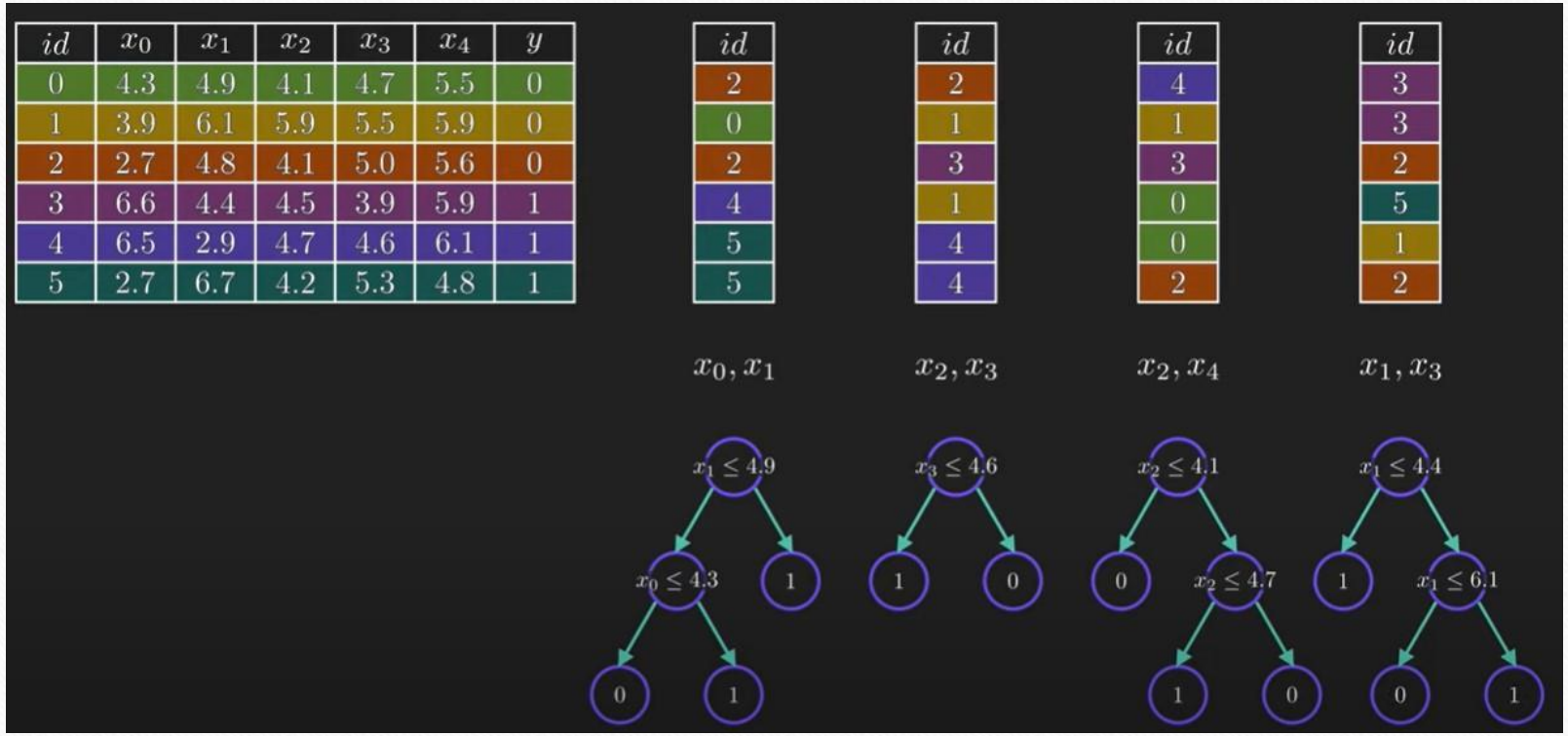
- **Random Forest**
- NOTE: The number of subsets is a hyperparameter. Here we chose four subsets. However, it is usually set to 100.
- **Step2**: Randomly select a subset of features for each dataset. The number of selected features is a hyperparameter usually set to the square root or log of the total number of features.



id	x_0	x_1	x_2	x_3	x_4	y
0	4.3	4.9	4.1	4.7	5.5	0
1	3.9	6.1	5.9	5.5	5.9	0
2	2.7	4.8	4.1	5.0	5.6	0
3	6.6	4.4	4.5	3.9	5.9	1
4	6.5	2.9	4.7	4.6	6.1	1
5	2.7	6.7	4.2	5.3	4.8	1

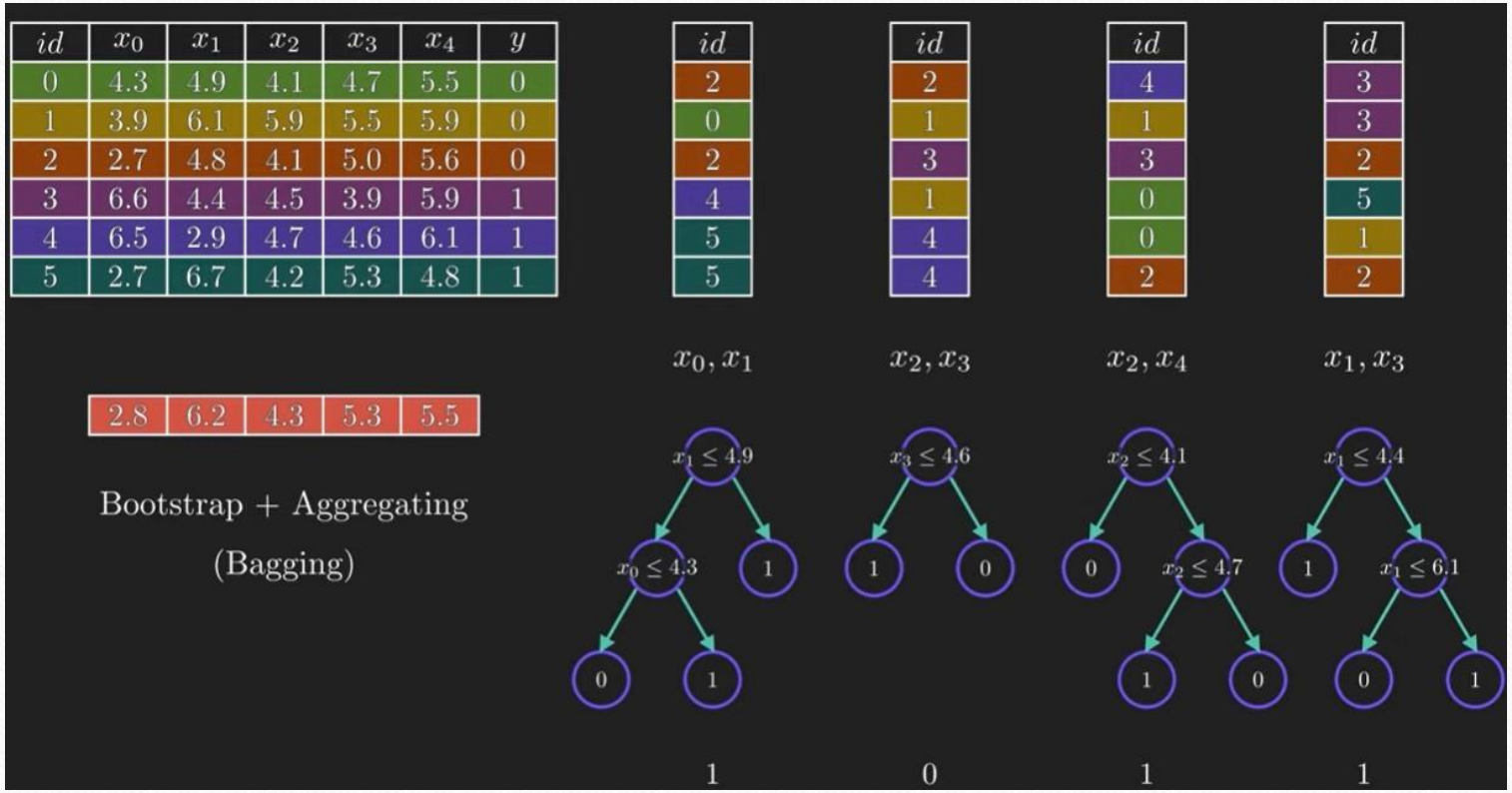
Classification

- **Random Forest**
- **Step3**: Train one tree for each selected dataset and set of features.



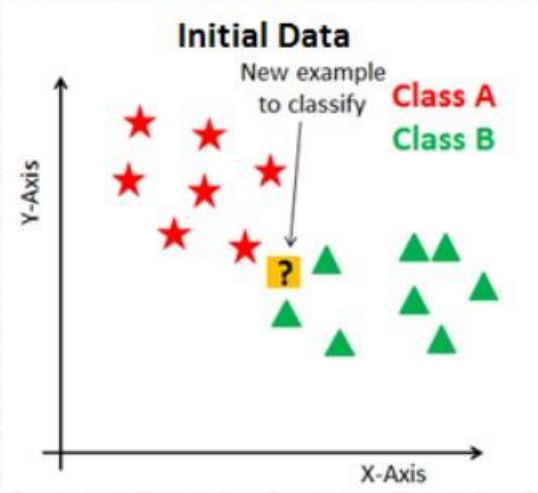
Classification

- **Random Forest**
- **Step4**: For new unseen data, make predictions using the majority voting



Classification

- **K-nearest neighbors**
- Suppose P1 is the point, for which label needs to predict. First, you find the k closest point to P1 and then classify points by majority vote of its k neighbors. Each object votes for their class and the class with the most votes is taken as the prediction



Classification

- **K-nearest neighbors (KNN)**
- For finding closest similar points, you find the distance between points using distance measures such as Euclidean distance, Hamming distance, Manhattan distance and Minkowski distance.

- KNN algorithm has the following three steps:

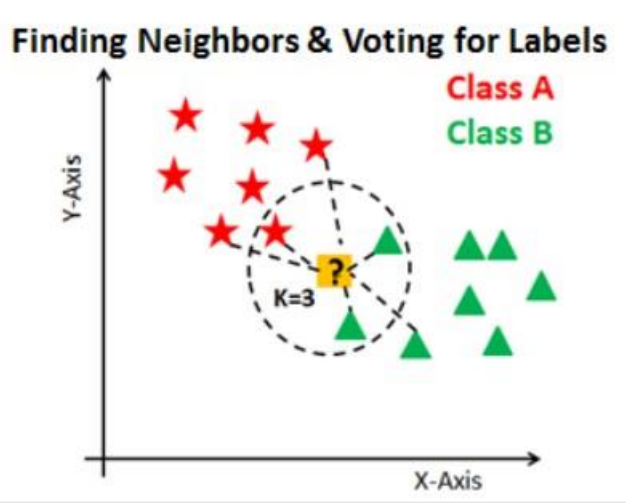
Step1: Calculate the distance between the new data point and K neighboring data

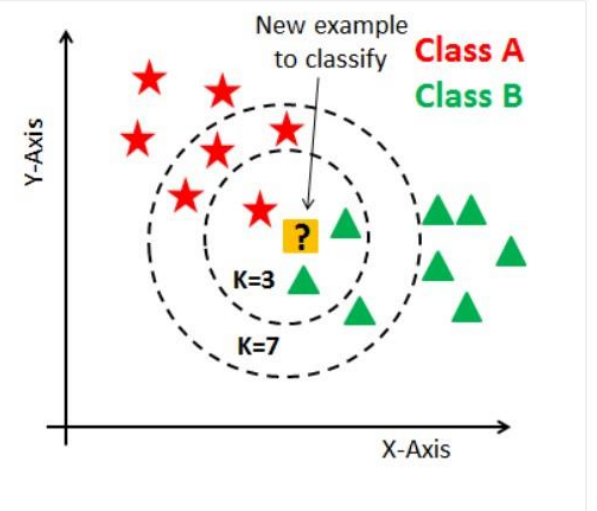
Step2: Find closest neighbors

Step3: Use majority voting to predict the class of the new data

Classification

- **K-nearest neighbors (KNN)**





NOTE: K is a hyperparameter and is usually set to the square root of the total number of samples. You may get different results using different K.

Classification

- **K-nearest neighbors (KNN)**

NOTE: KNN and RF can be used for multiclass classification

NOTE: There are also KNN and RF regression algorithms that use similar procedures to the classifiers but are used for regression.

Classification

- **Examples**

Please see Lecture #7_examples notebook for examples of classification models.