Course notes: Cluster Analysis

Cluster analysis

Cluster analysis

Cluster analysis is a multivariate statistical technique that groups observations on the basis some of their features or variables that they are described by.

The goal of clustering is to maximize the similarity of observations within a cluster and maximize the dissimilarity between clusters.

Euclidean distance

The most intuitive way to measure the distance between them is by drawing a straight line from one to the other. That's also known as Euclidean distance.

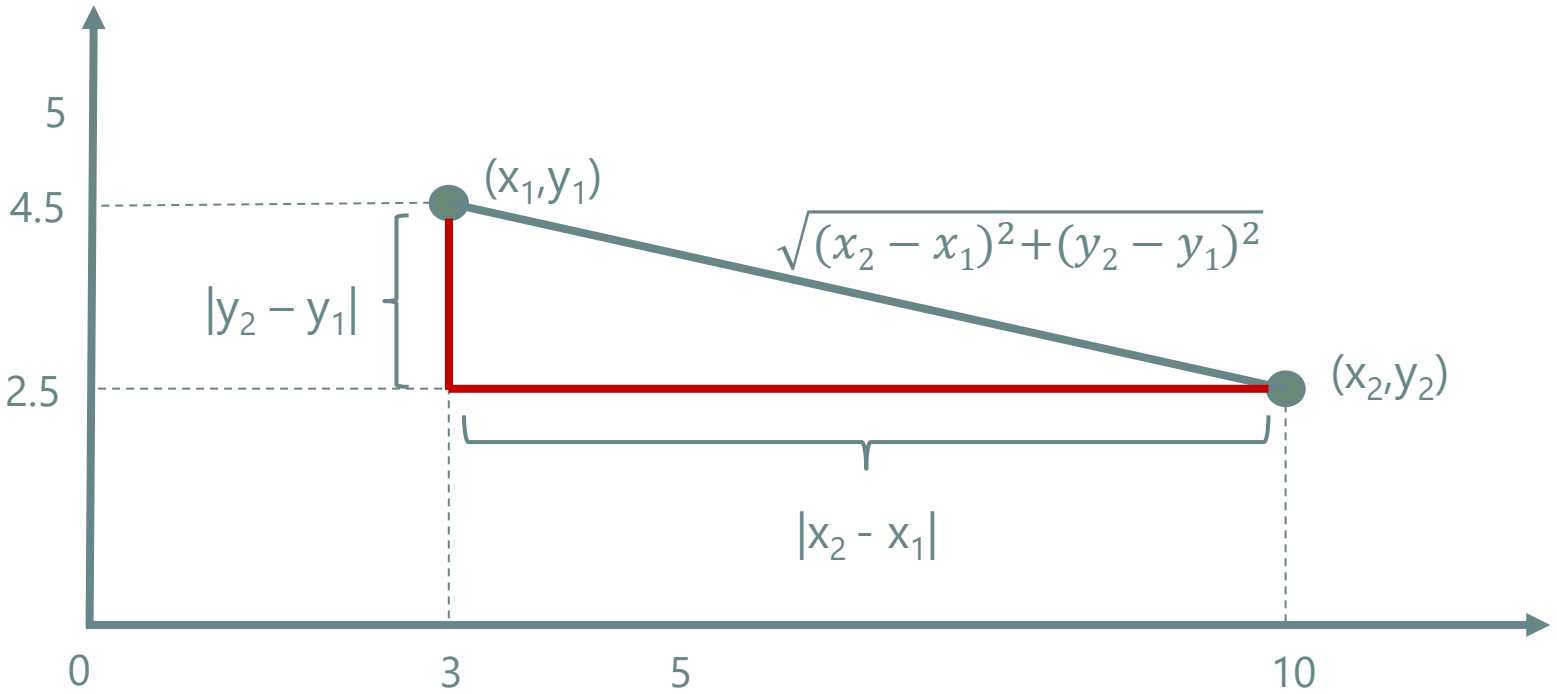
2D space: $d(A,B) = d(B,A) = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$

3D space: $d(A,B) = d(B,A) = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2 + (z_2 - z_1)^2}$

If the coordinates of A are $(a_1,a_2,\ldots,a_n)$ and of B are $(b_1,b_2,\ldots b_n)$

N-dim space:

$$d(A,B) = d(B,A) = \sqrt{(a_1 - b_1)^2 + (a_2 - b_2)^2 + \cdots + (a_n - bn)^2}$$



K-means clustering

1. Choose number of clusters

Number – K, chosen by the person performing the clustering

2. Specify the number of seeds

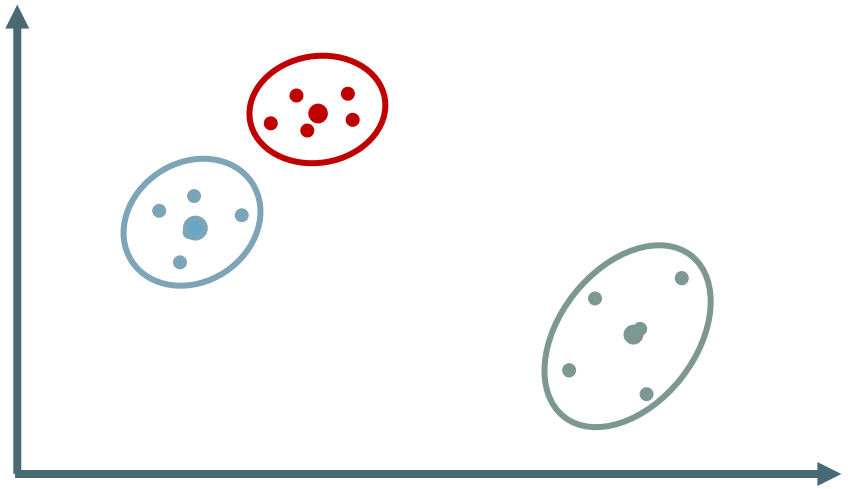
A seed is a starting centroid (can be chosen at random, with an algorithm or according to some prior knowledge)

3. Assign each point to a centroid

Based on proximity (measured by Euclidian distance)

4. Adjust the centroids

Repeat 2. and 3. until there is you can no longer find a better clustering solution



K-means clustering - pros and cons

PROS

- Simple to implement
  (so many people can use it)
- Computationally efficient
  (it takes considerably less time than any hierarchical clustering model)
- Widely used
  (popular, therefore, in demand)
- Always yields a result
  (also a con as it may be deceiving)

CONS

- We need to pick K
  (often, we don't know how many clusters we need)
- Sensitive to initialization
  (but we can use methods such as kmeans++ to determine the seeds)
- Sensitive to outliers
  (by far the biggest downside of k-means)
- Produces spherical solutions
  (thus, not as generalizable)

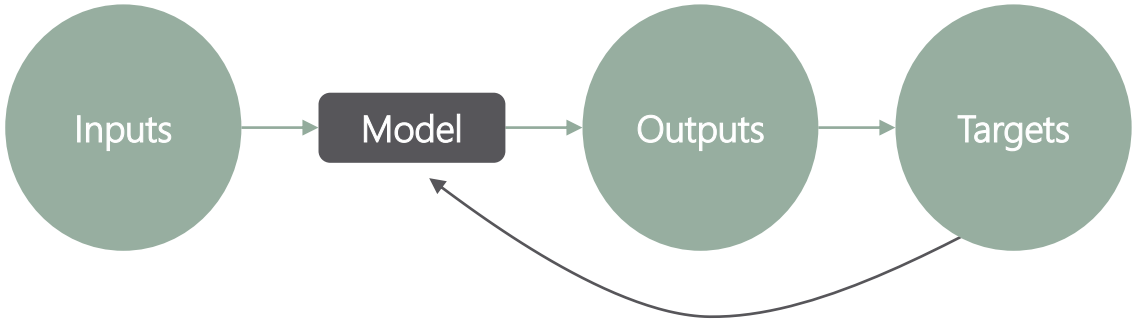
Classification vs Clustering

Classification is a typical example of **supervised learning**.

It is used whenever we have input data and the desired correct outcomes (targets). We train our data to find the patterns in the inputs that lead to the targets.

With classification we essentially need to know the correct class of each of the observations in our data, in order to apply the algorithm.

A logistic regression is a typical example of classification.

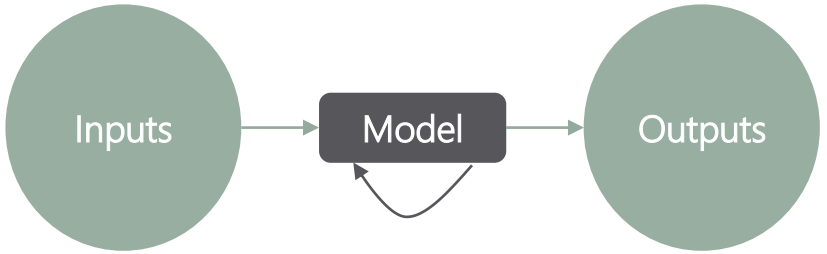


We use the targets (correct values) to adjust the model to get better outputs.

Cluster analysis is a typical example of **unsupervised learning**.

It is used whenever we have input data but have no clue what the correct outcomes are.

Clustering is about grouping data points together based on similarities among them and difference from others.



There is no feedback loop, therefore, the model simply finds the outputs it deems best.

Types of clustering

With flat methods there is no hierarchy, but rather the number of clusters are chosen prior to clustering.

Flat methods have been developed because hierarchical clustering is much slower and computationally expensive.

Nowadays, flat methods are preferred because of the volume of data we typically try to cluster.

Historically, lustering was developed first. An example hierarchical cof clustering with hierarchy is taxonomy of the animal kingdom.

It is superior to flat clustering in the fact that it explores (contains) all solutions.

With divisive clustering we start from a situation where all observations are in the same cluster, e.g. from the dinosaurs. Then we split this big cluster into 2 smaller ones. Then we continue with 3, 4, 5, and so on, until each observation is its separate cluster.

To find the best split, **we must explore all possibilities at each step**.

When it comes to agglomerative clustering, the approach is bottom up. We start from different dog and cat breeds, cluster them into dogs and cats respectively, and then we continue pairing up species, until we reach the animal cluster.

To find the combination of observations into a cluster, **we must explore all possibilities at each step.**

Introduction to
K Means Clustering

K Means Clustering

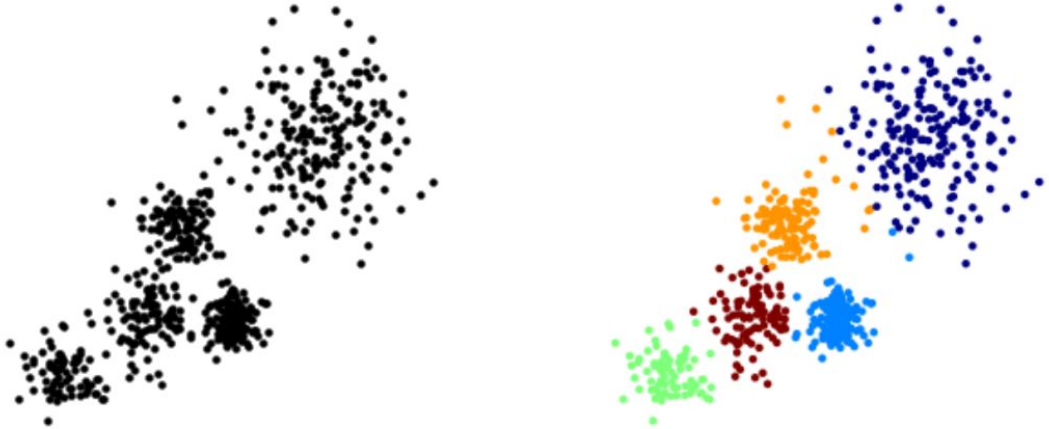
K Means Clustering is an unsupervised learning algorithm that will attempt to group similar clusters together in your data.

So what does a typical clustering problem look like?

- Cluster Similar Documents
- Cluster Customers based on Features
- Market Segmentation
- Identify similar physical groups

K Means Clustering

- The overall goal is to divide data into distinct groups such that observations within each group are similar

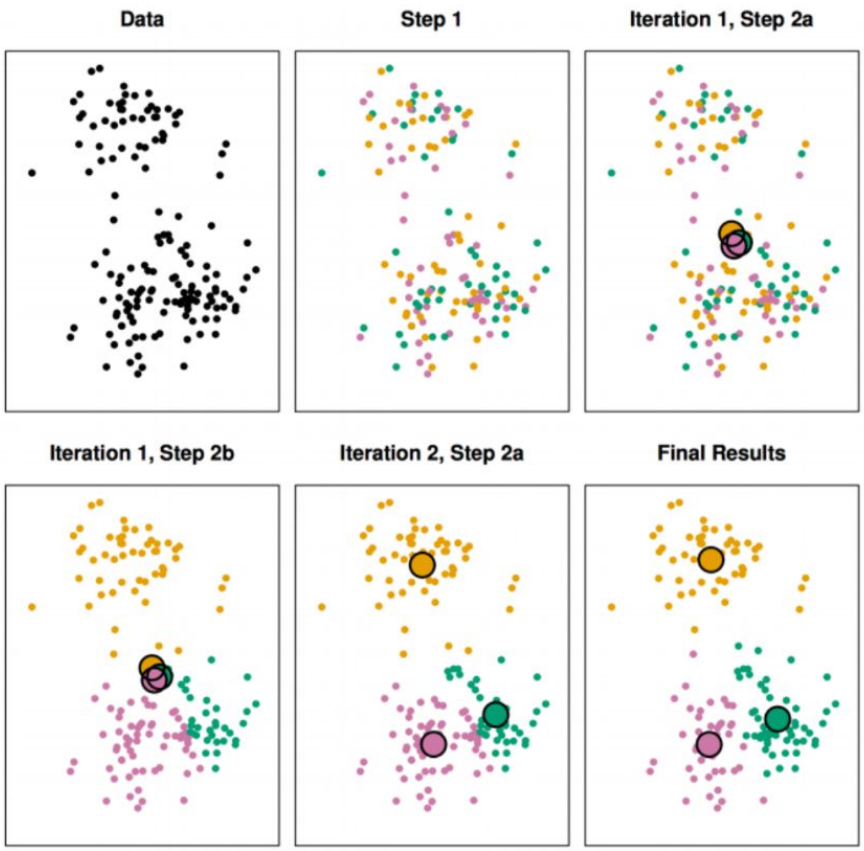


K Means Clustering

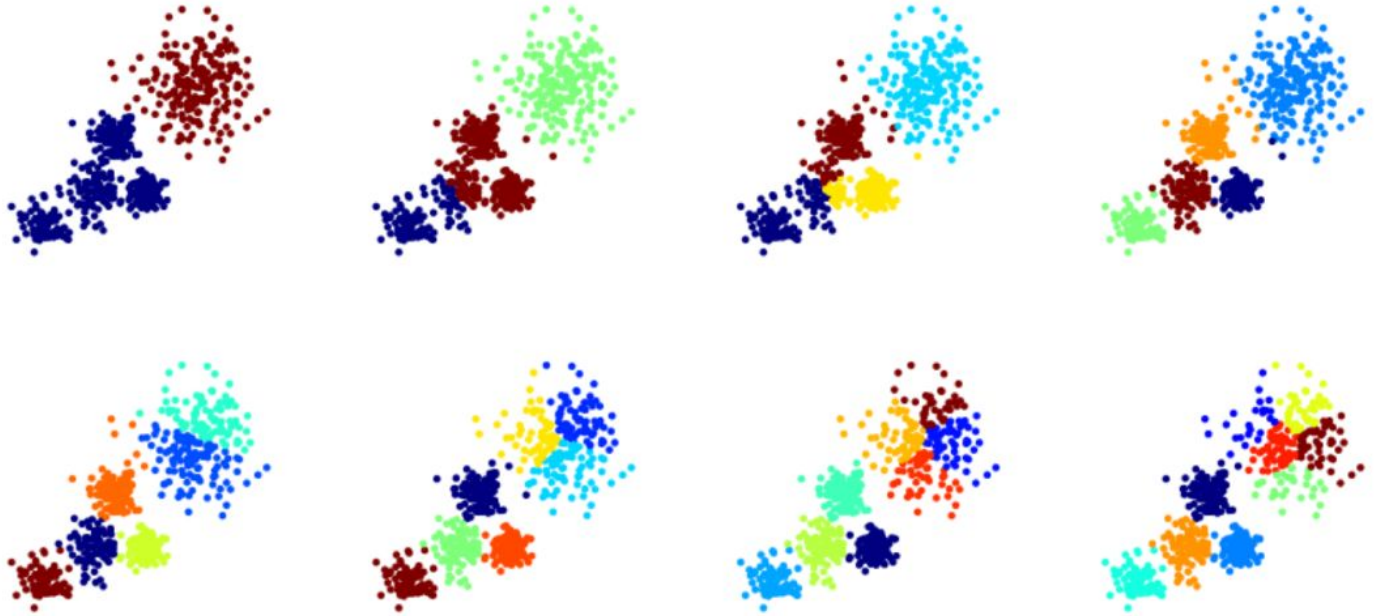
The K Means Algorithm

- Choose a number of Clusters “K”
- Randomly assign each point to a cluster
- Until clusters stop changing, repeat the following:
  - For each cluster, compute the cluster centroid by taking the mean vector of points in the cluster
  - Assign each data point to the cluster for which the centroid is the closest

K Means Clustering



Choosing a K Value



Choosing a K Value

- There is no easy answer for choosing a “best” K value
- One way is the elbow method

First of all, compute the sum of squared error (SSE) for some values of k (for example 2, 4, 6, 8, etc.).

The SSE is defined as the sum of the squared distance between each member of the cluster and its centroid.

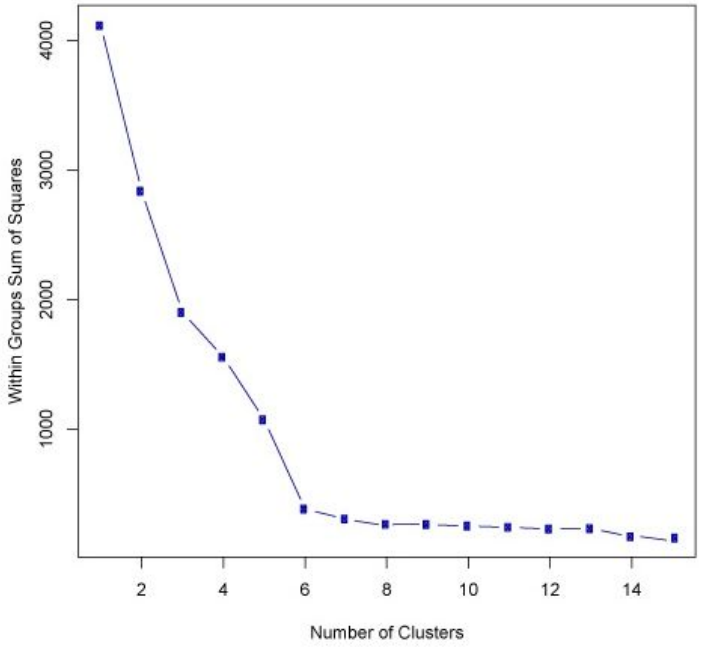
Choosing a K Value

If you plot k against the SSE, you will see that *the error decreases as k gets larger*; this is because when the number of clusters increases, they should be smaller, so distortion is also smaller.

The idea of the elbow method is to choose the k at which the SSE decreases abruptly.

This produces an "elbow effect" in the graph, as you can see in the following picture:

Choosing a K Value



MACHINE LEARNING

K-MEANS CLUSTERING

K-means clustering

- Very popular unsupervised learning algorithm in data mining
- Automatically divides the data into clusters / groupings of similar items + it does this without having been told what the groups should look like ahead of time !!!
- **Problem**: how could a computer possibly know where one group ends and another begins?
- Elements inside a cluster should be very similar to each other, but very different from those outside

K-means clustering

- **k-means** clustering aims to partition **n** observations into **k** clusters in which each observation belongs to the cluster with the nearest mean
- Can be done with graph algorithms: construct the minimum spanning tree...and remove the last **k** edges
- It is an **NP-hard** problem
- Lloyd-algorithm is very common nowadays

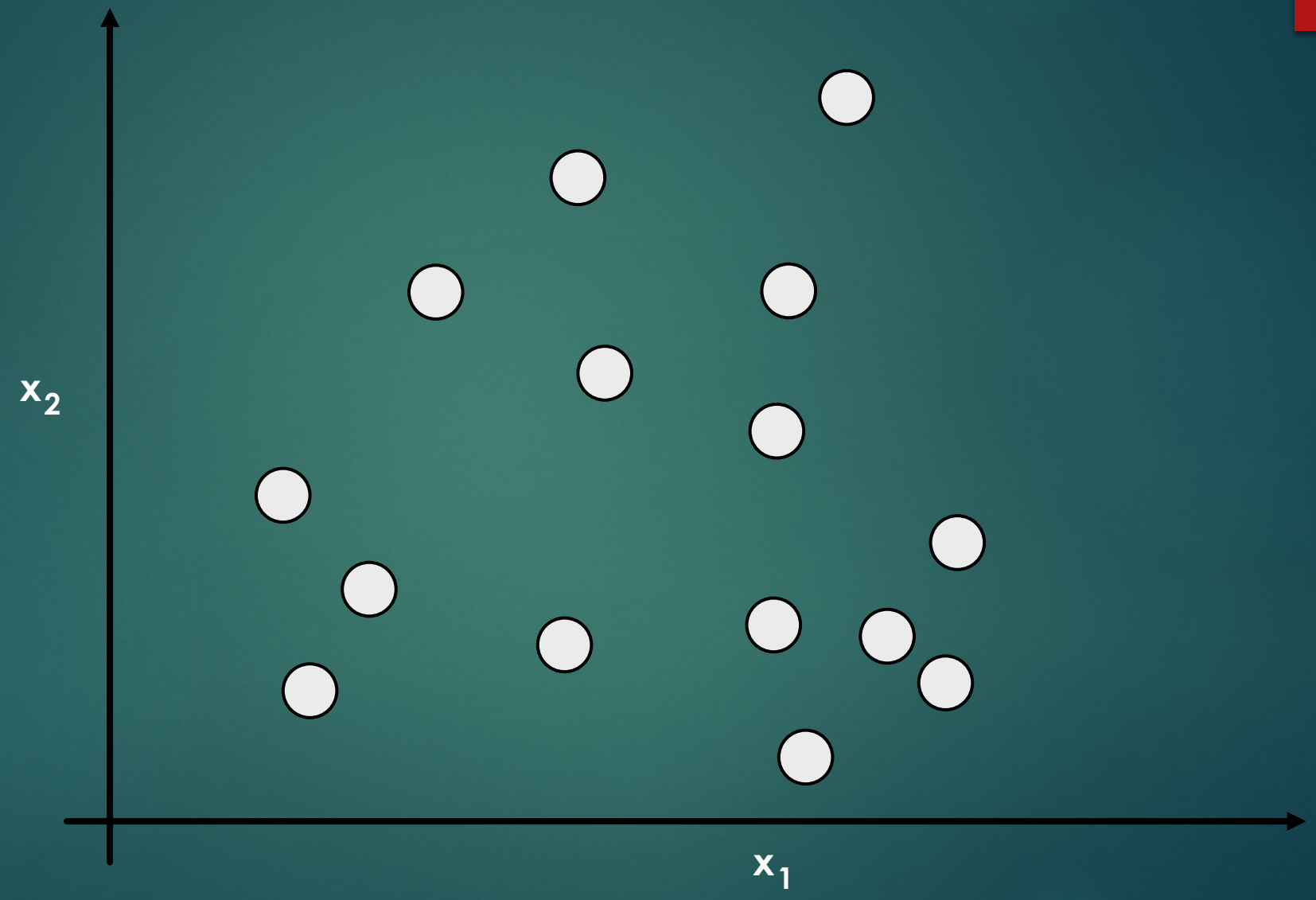
Lloyd-algorithm

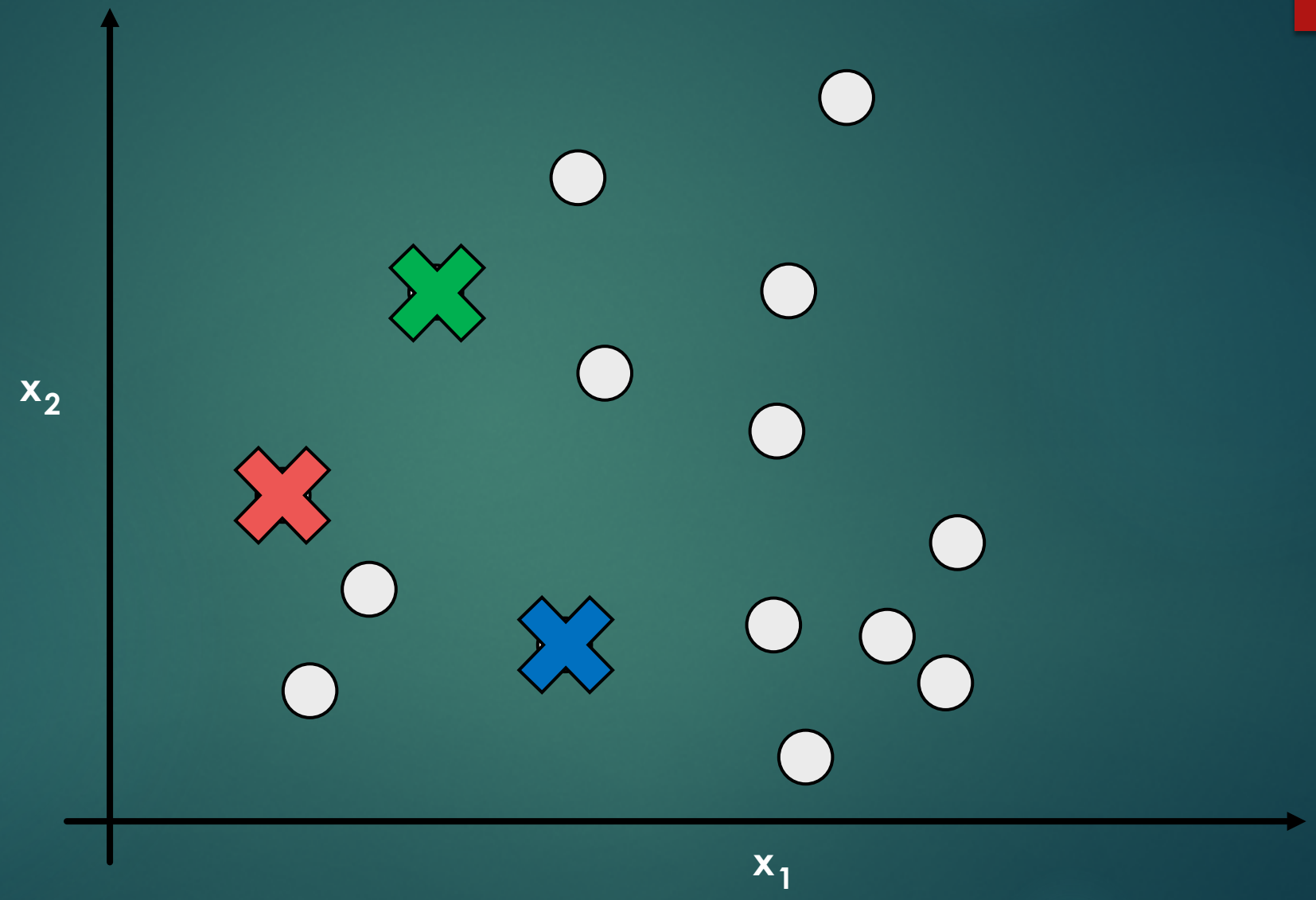
1.) initialize the centroids at random, these are the centers of a given cluster

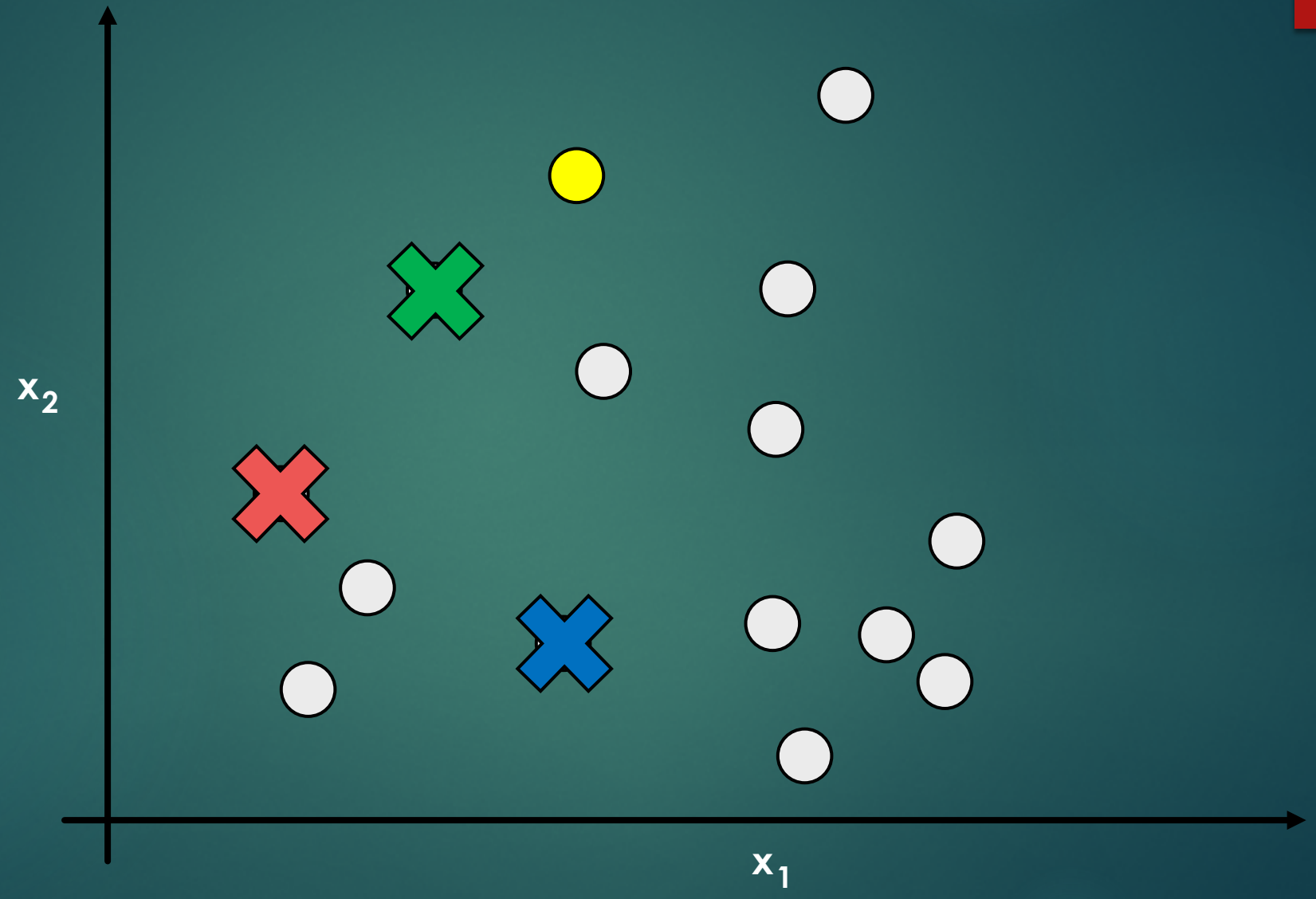
2.) decide for every point in our dataset → what centroid is the nearest to them

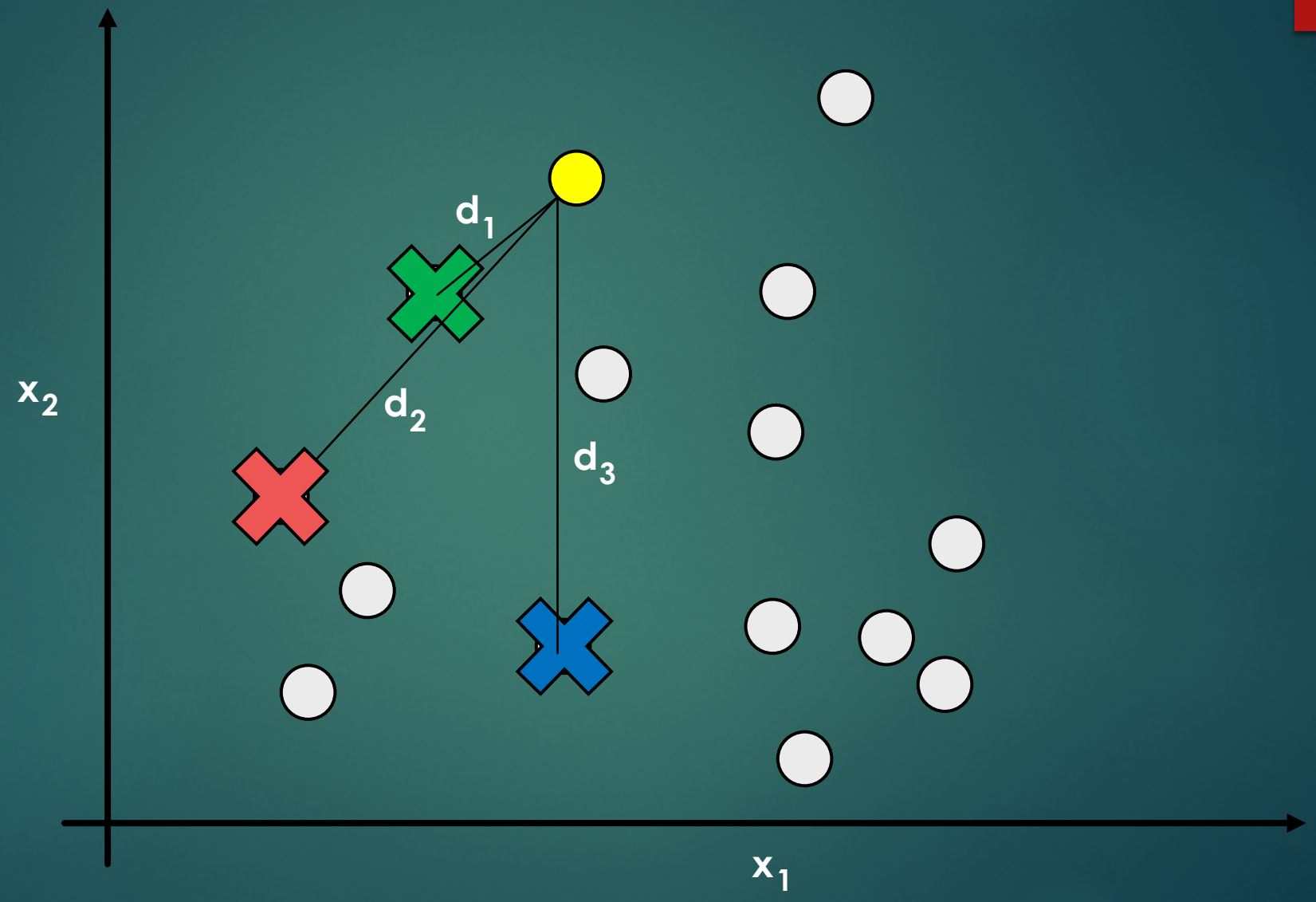
3.) calculate the new means of every distinct clusters

// run until convergence !!!

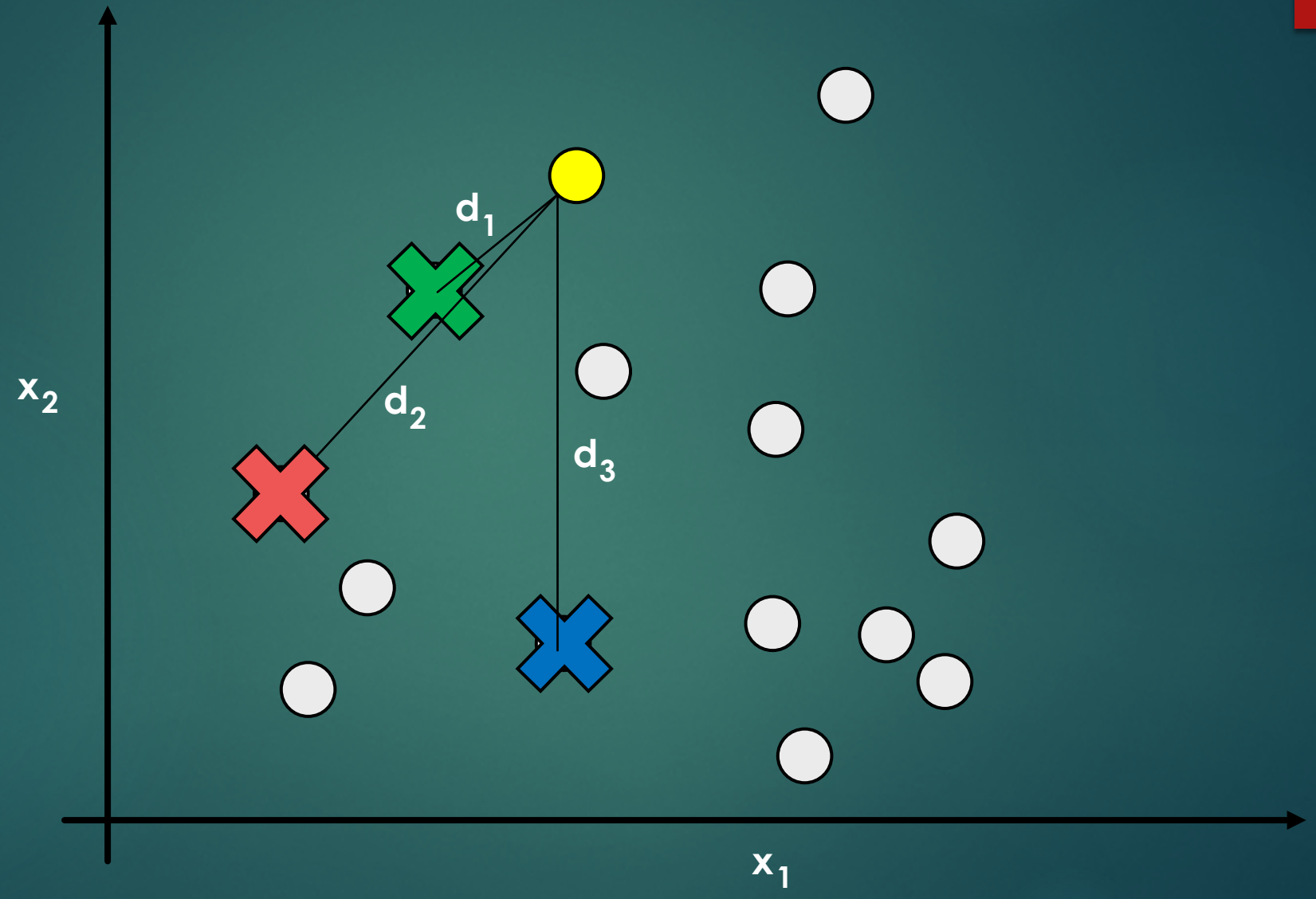

x_2
x_1


x_2
x_1


x_2
x_1

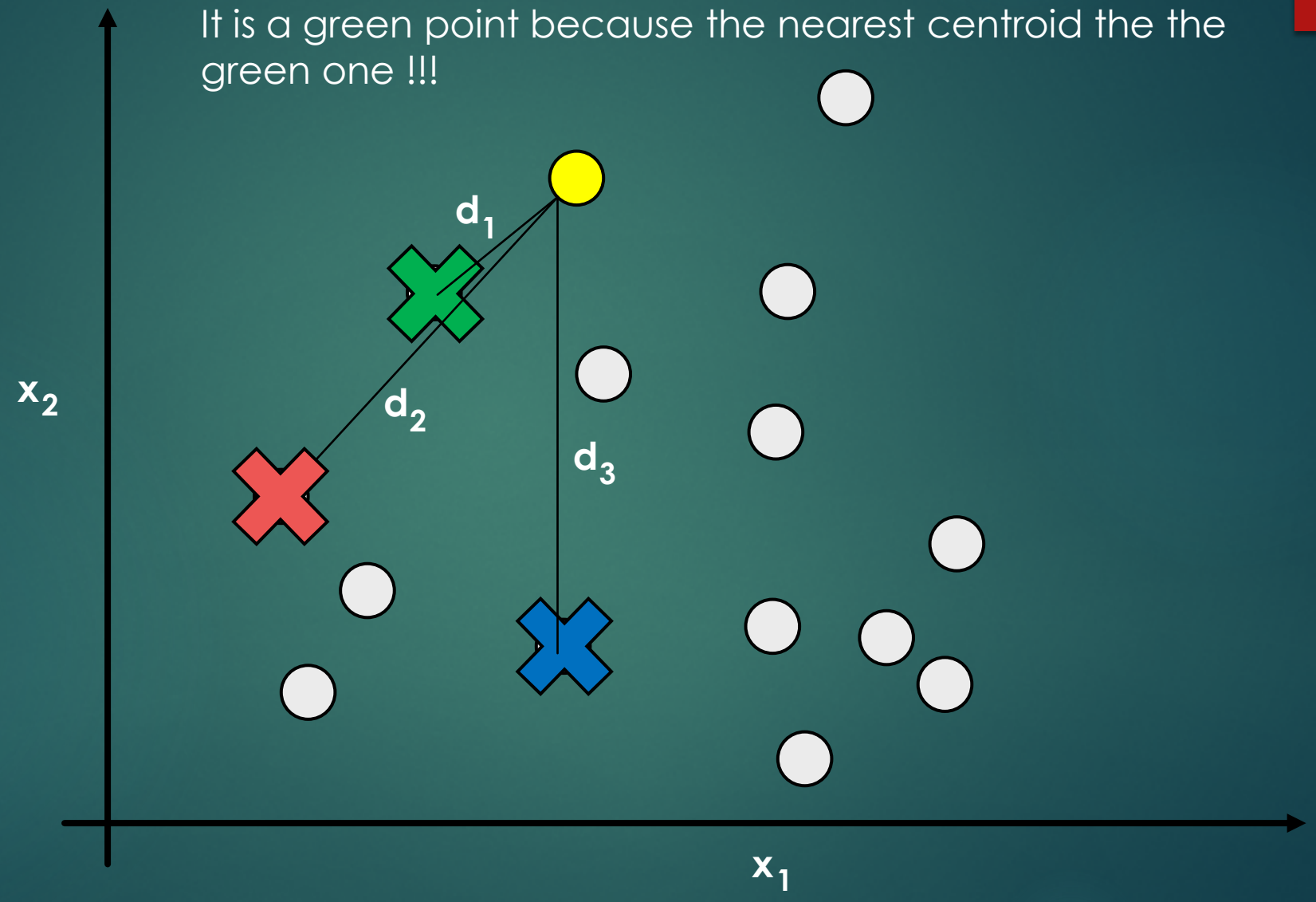

x_2
d_1
d_2
d_3
x_1

Calculate the min $\{ d_1, d_2, d_3 \}$!!!



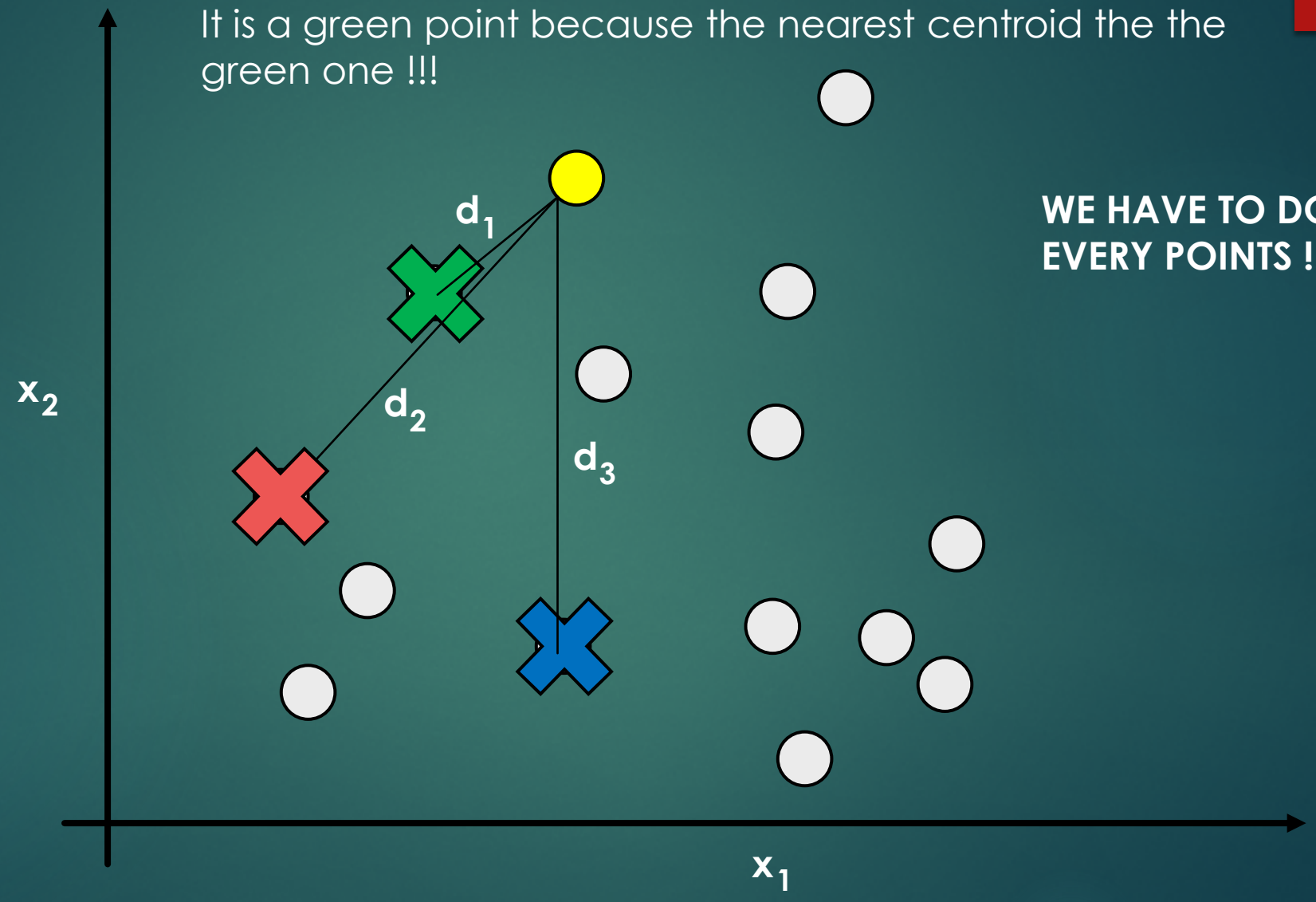
Calculate the min $\{ d_1, d_2, d_3 \}$!!!

It is a green point because the nearest centroid the the green one !!!

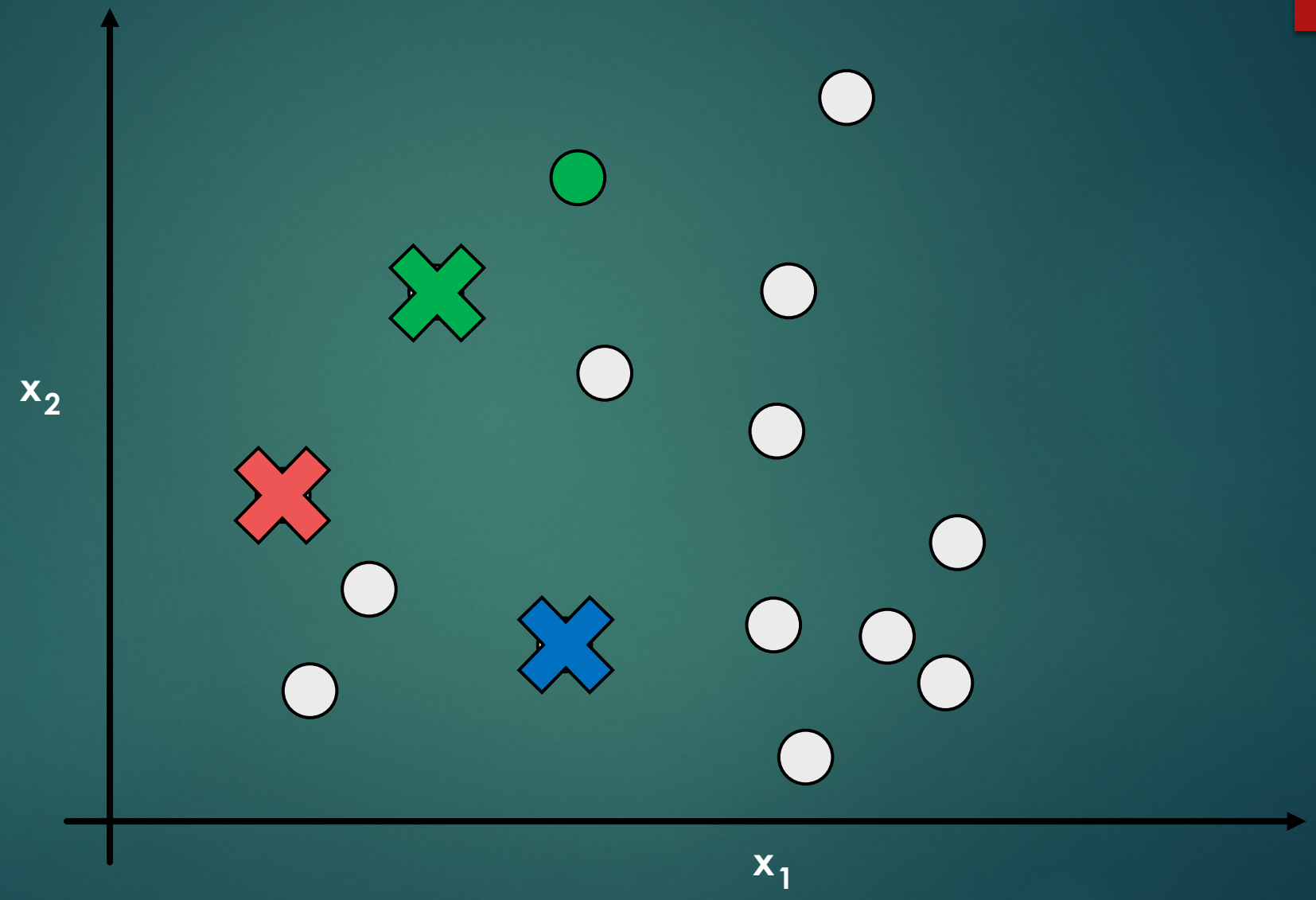


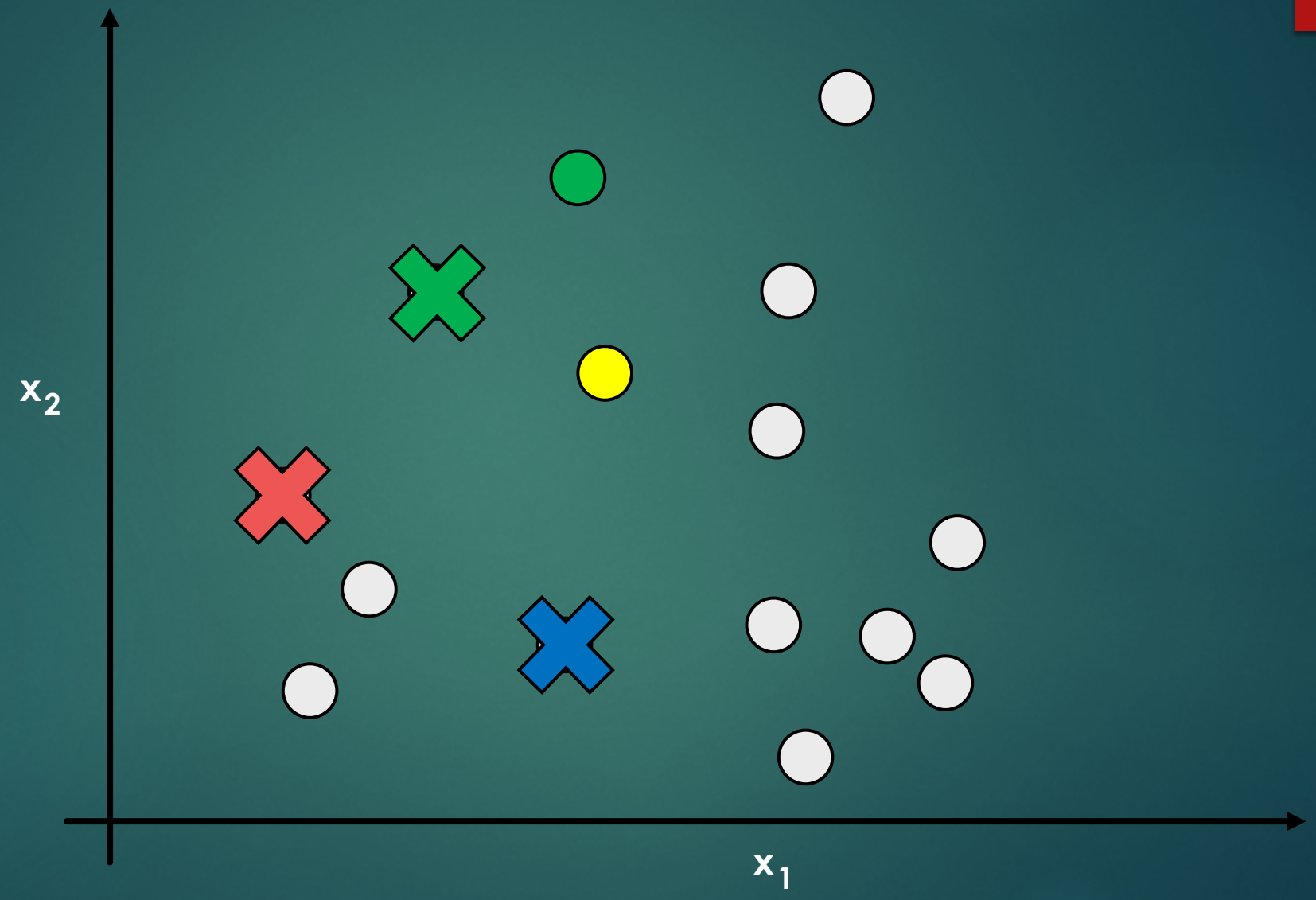
Calculate the min $\{ d_1, d_2, d_3 \}$!!!

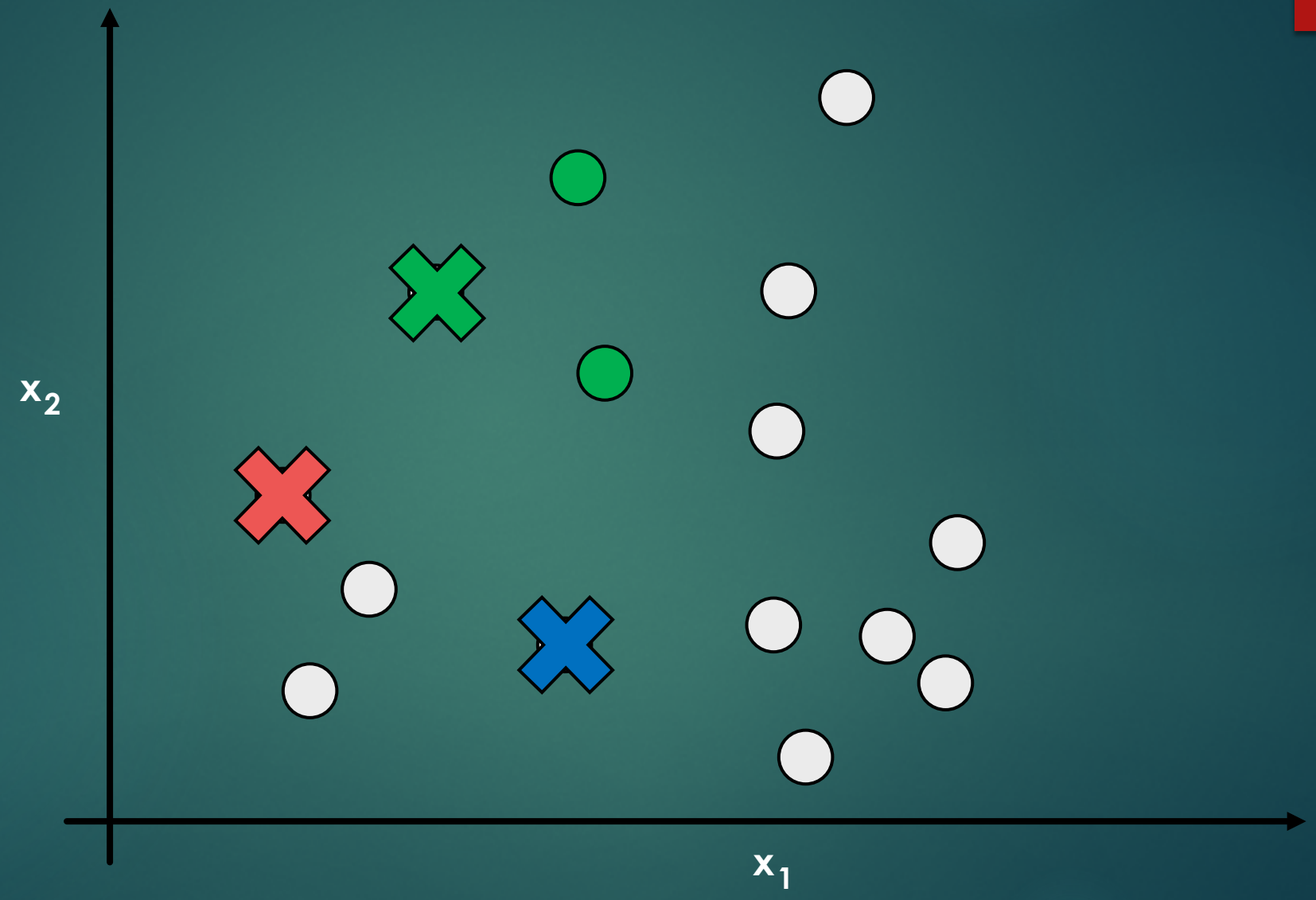
It is a green point because the nearest centroid the the green one !!!

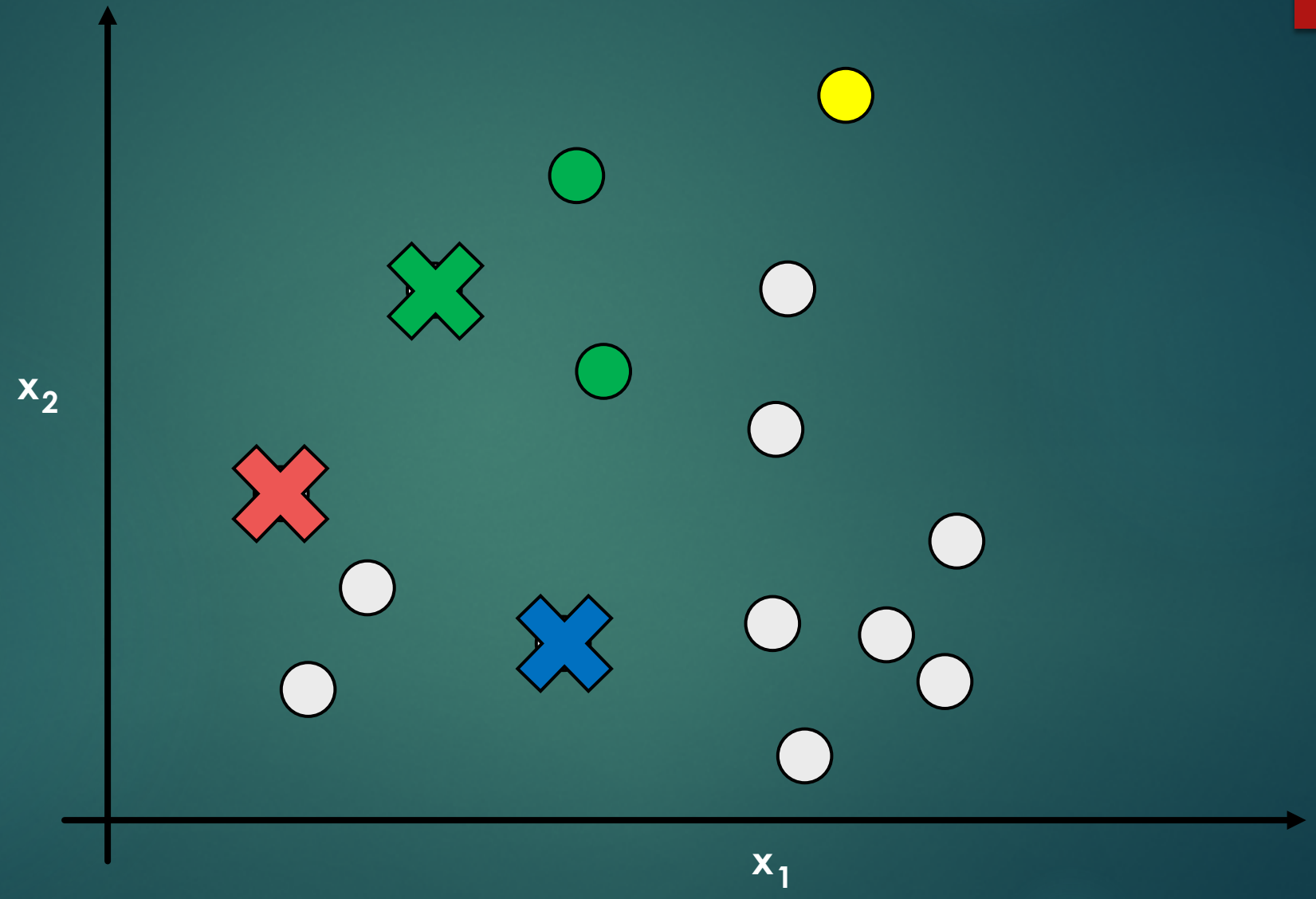


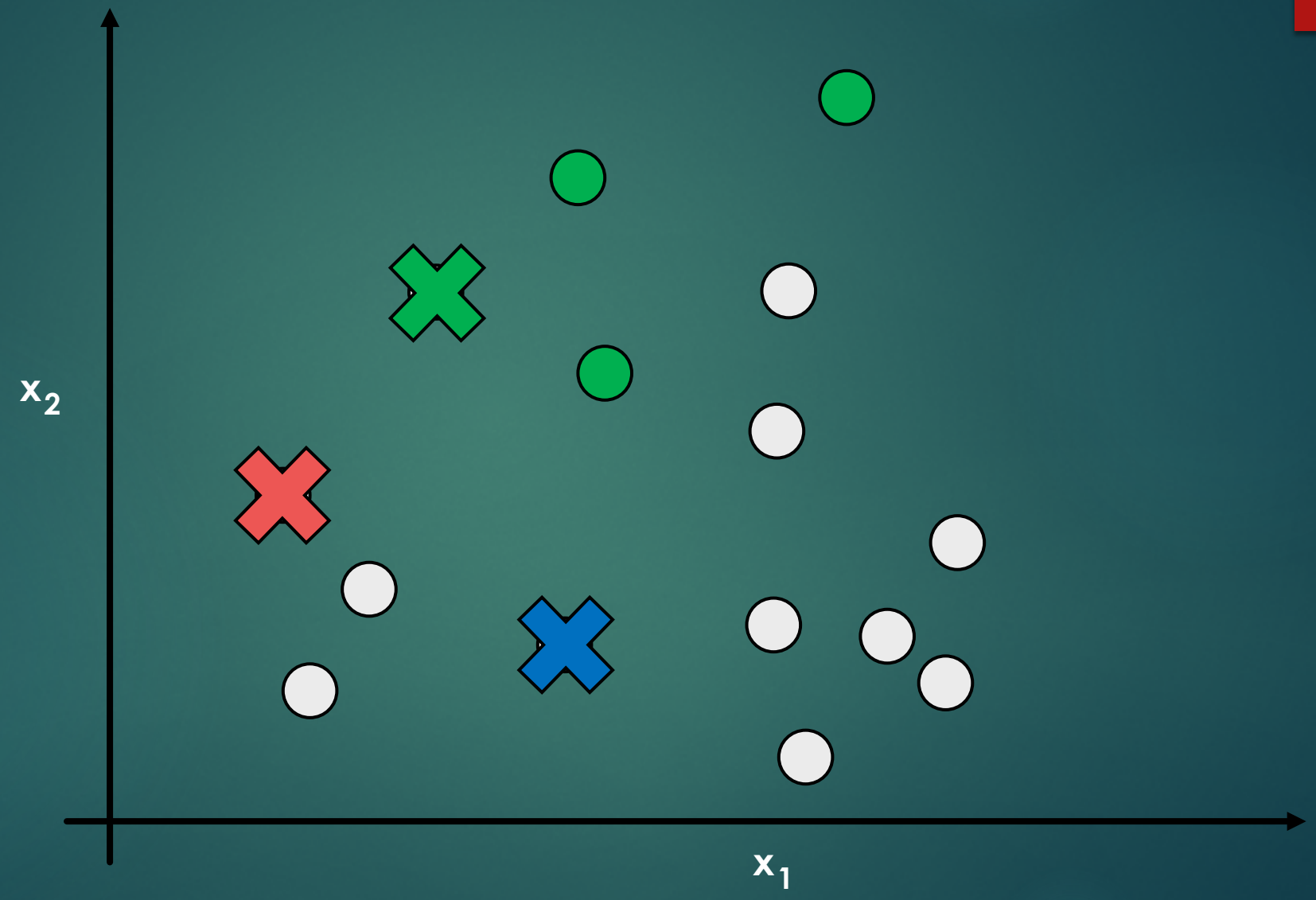
WE HAVE TO DO IT TO EVERY POINTS !!!

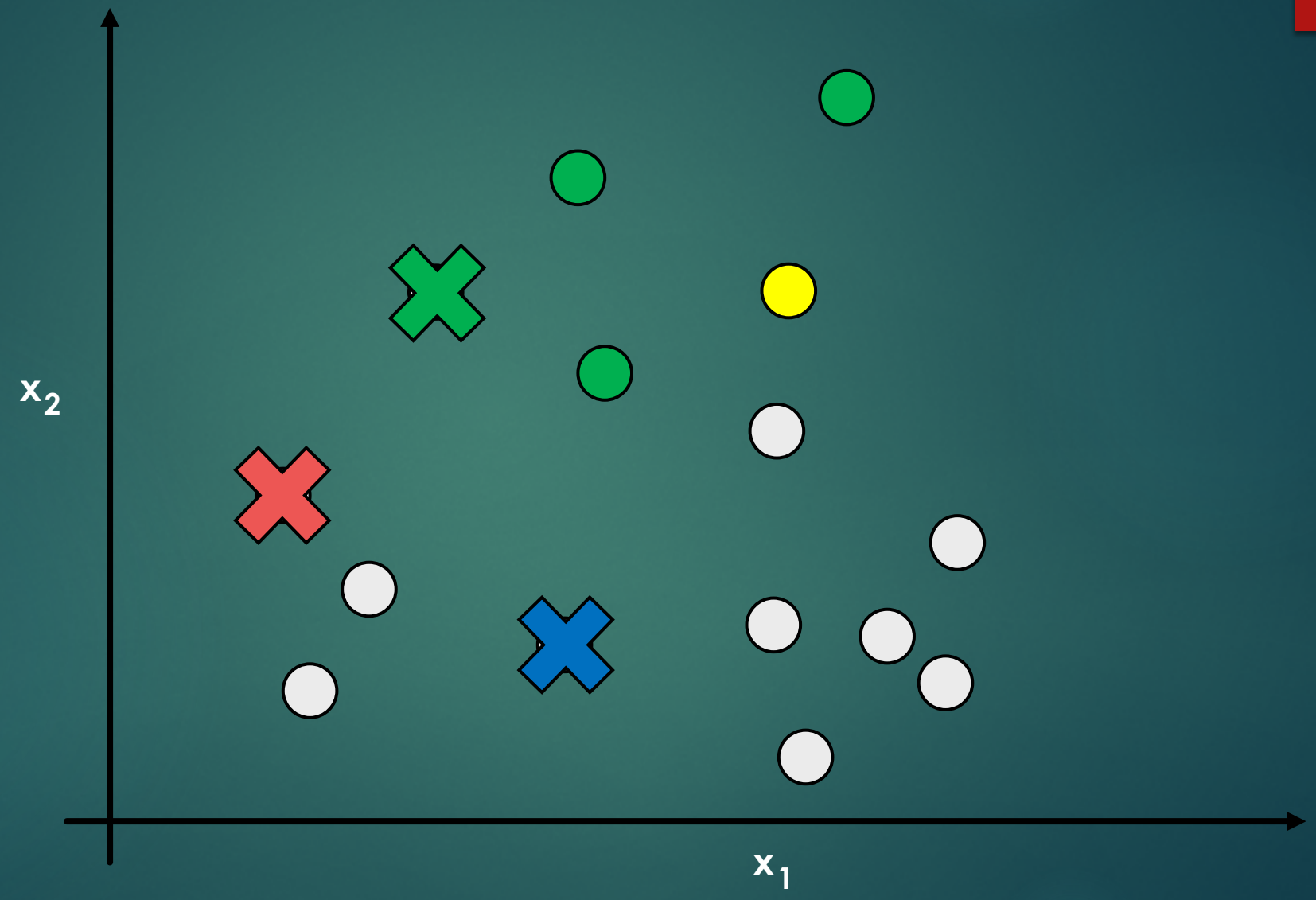

x_2
x_1

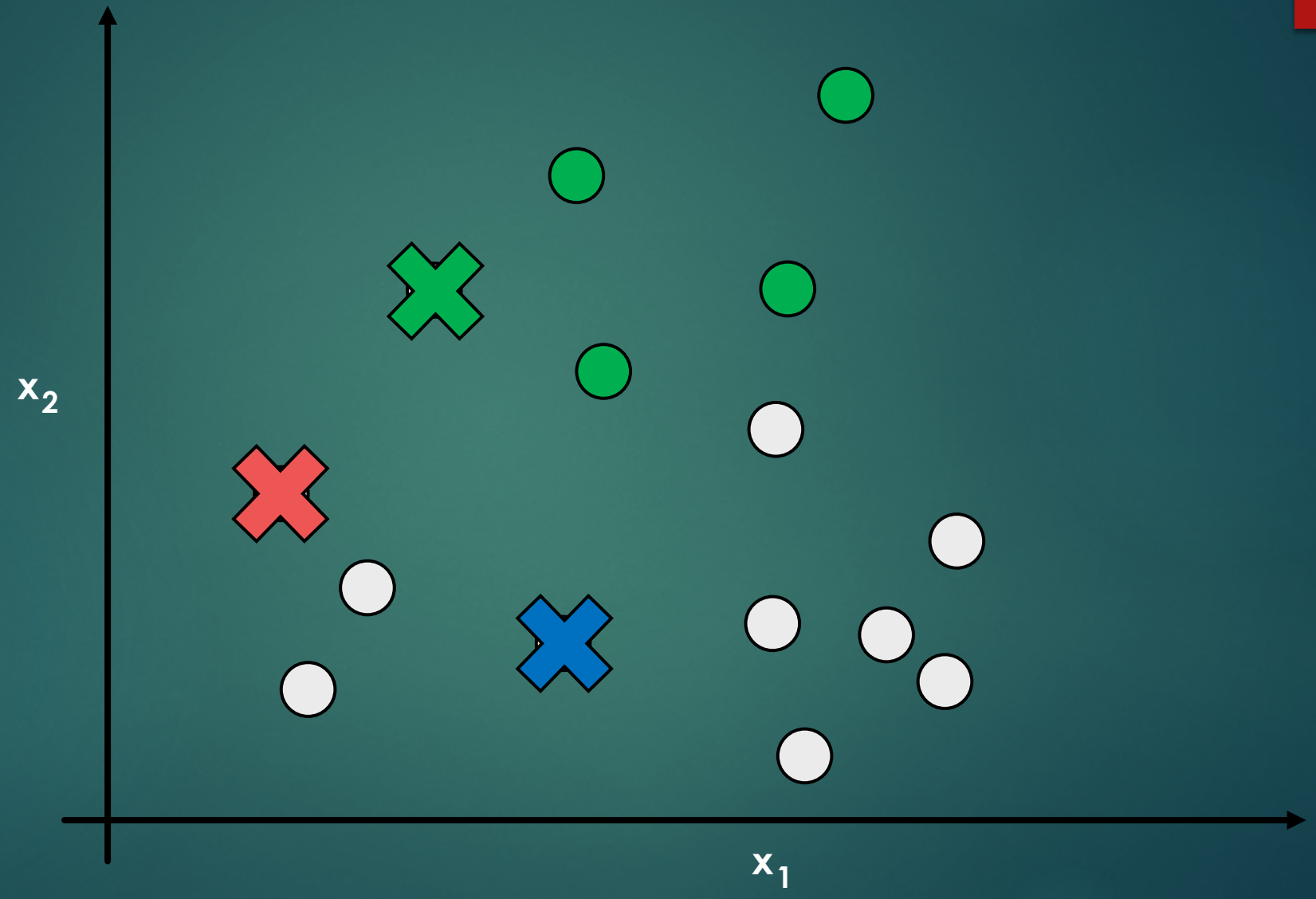

x_2
x_1

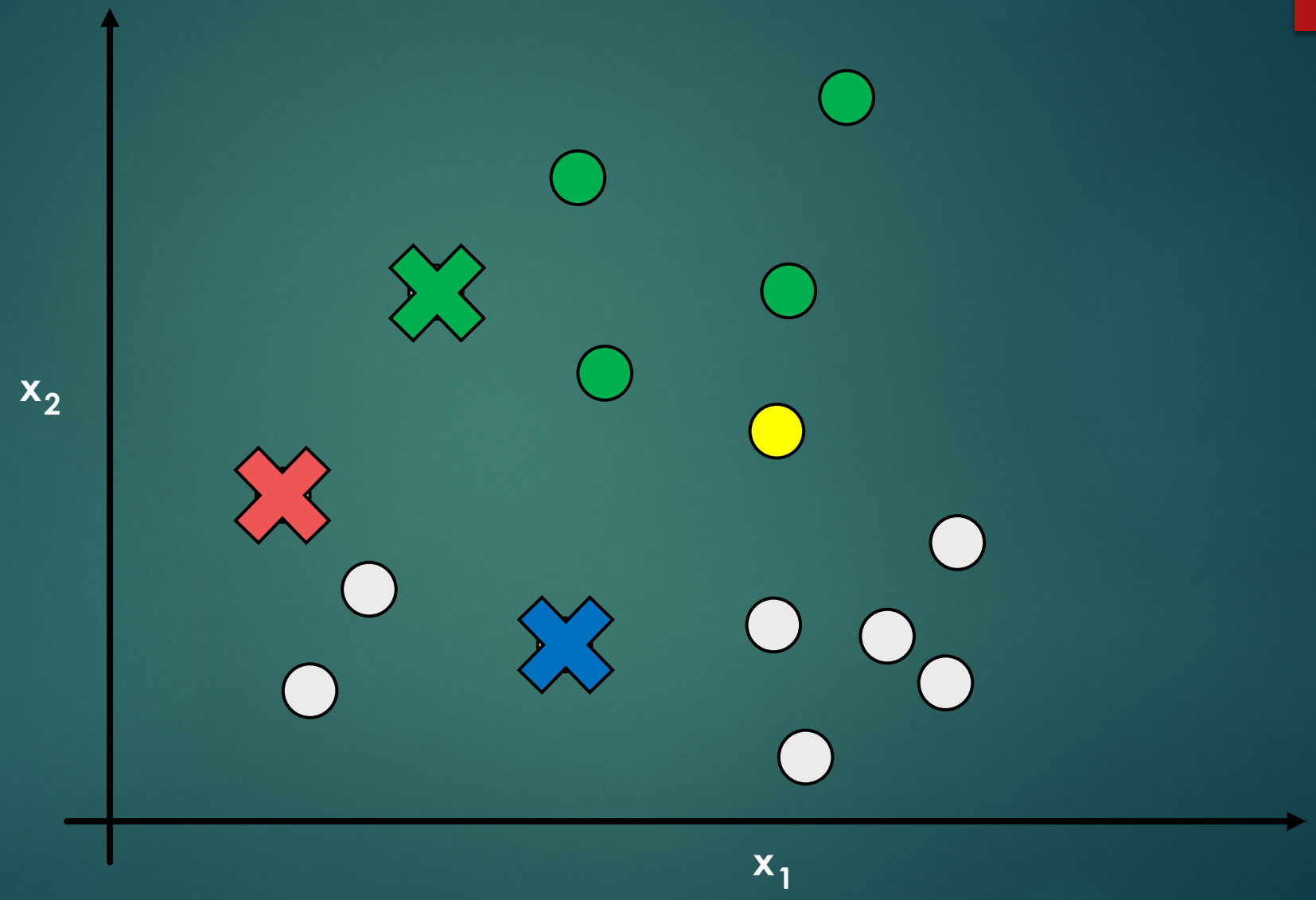

x_2
x_1

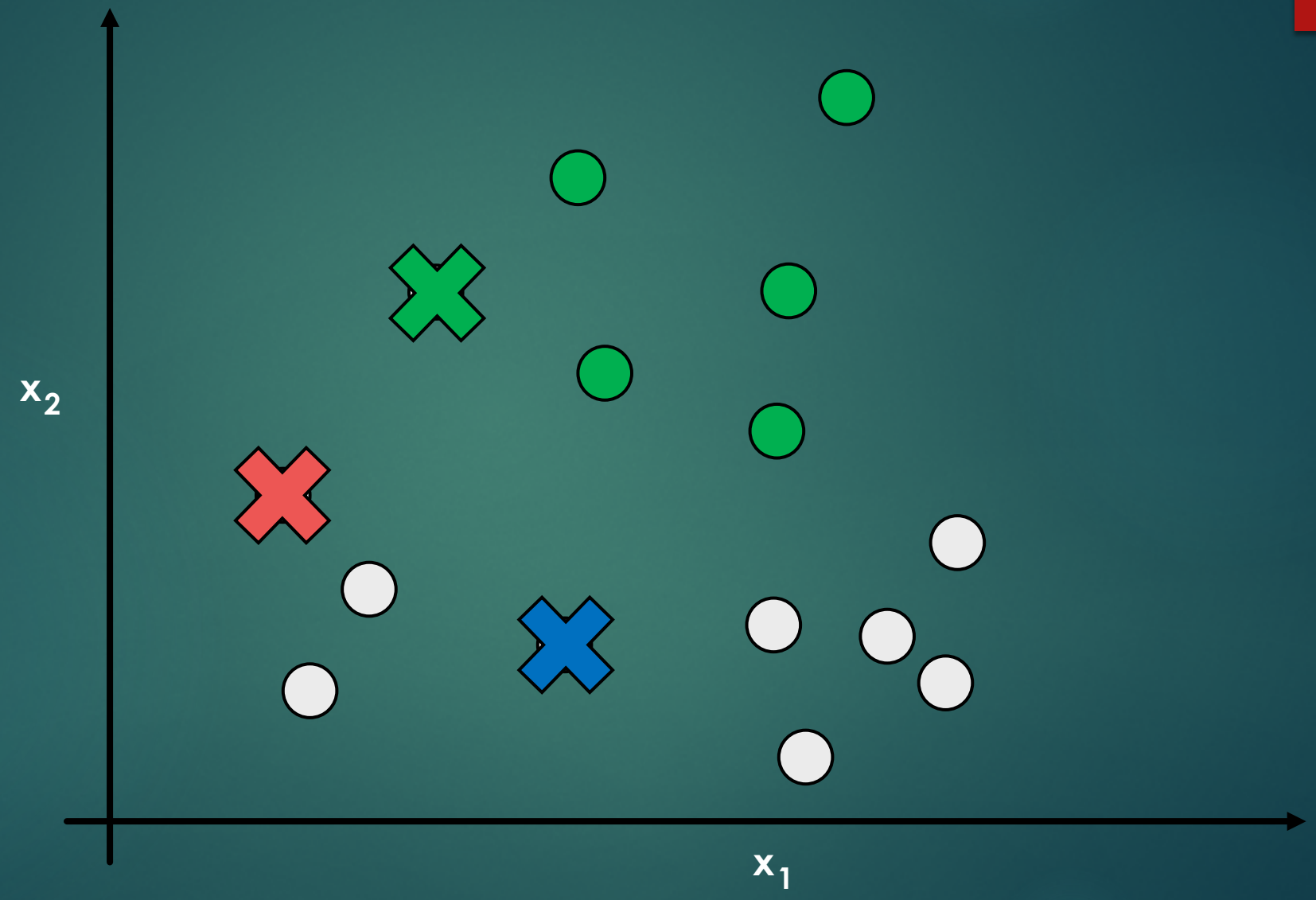

x_2
x_1

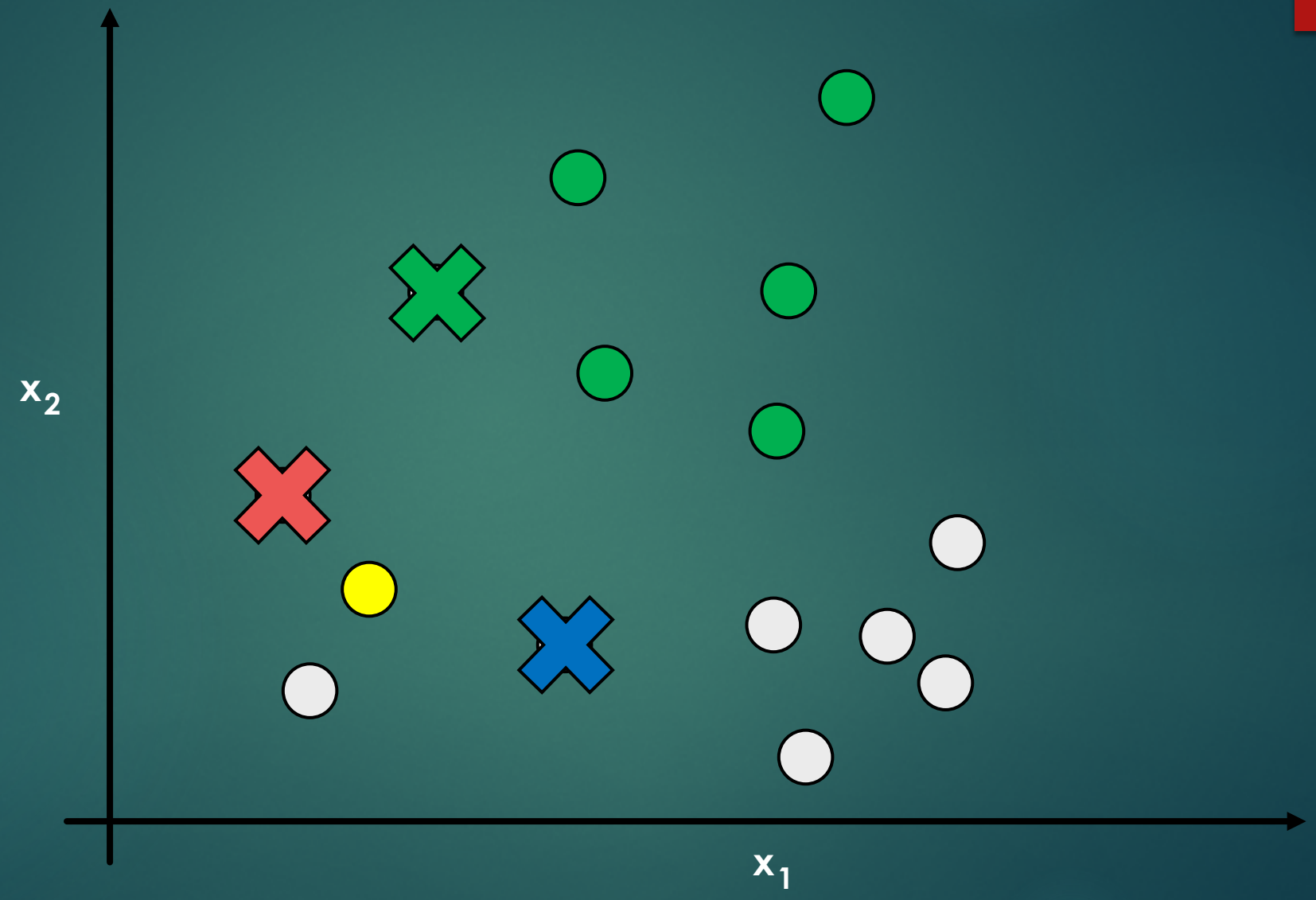

x_2
x_1

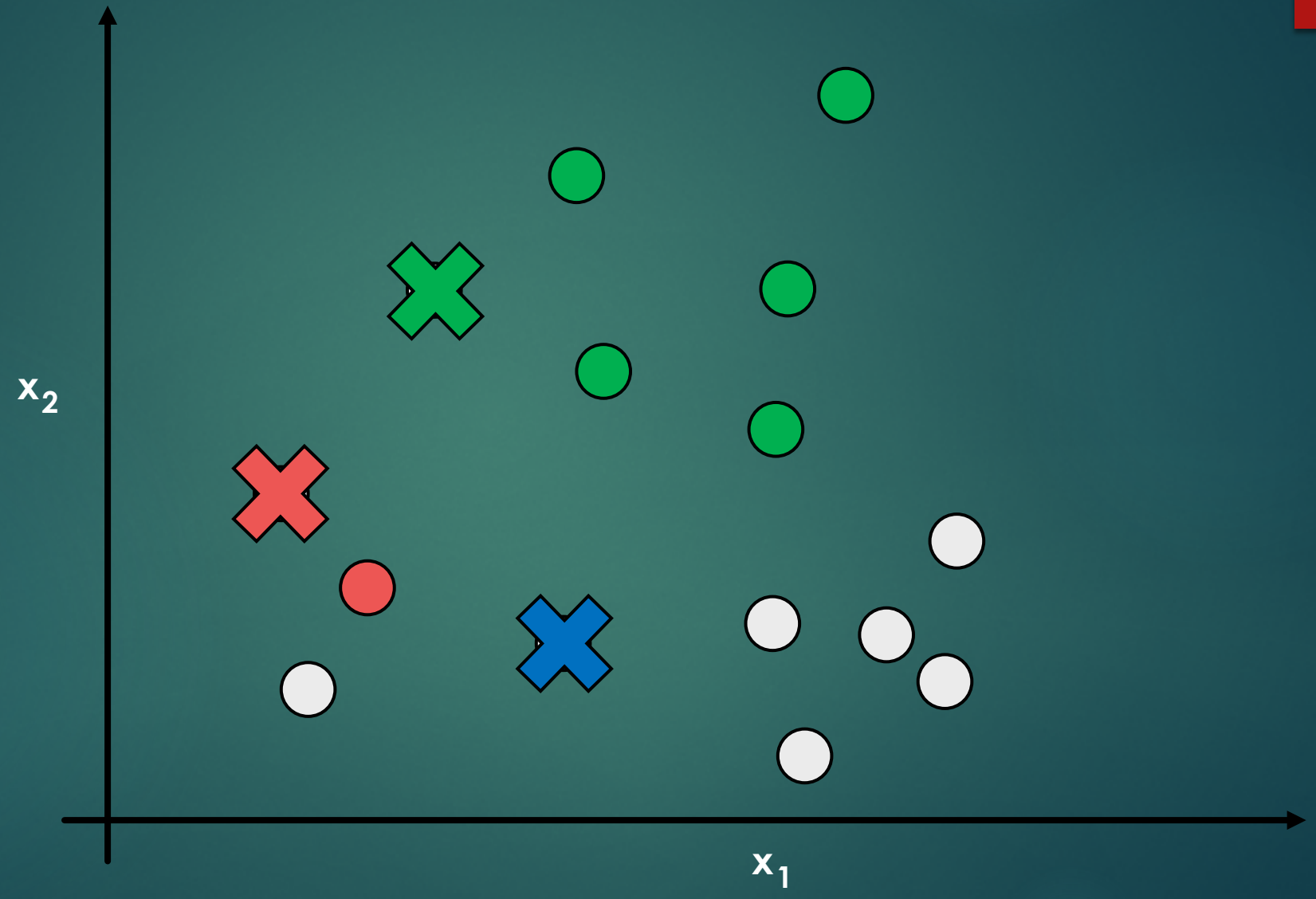

x_2
x_1

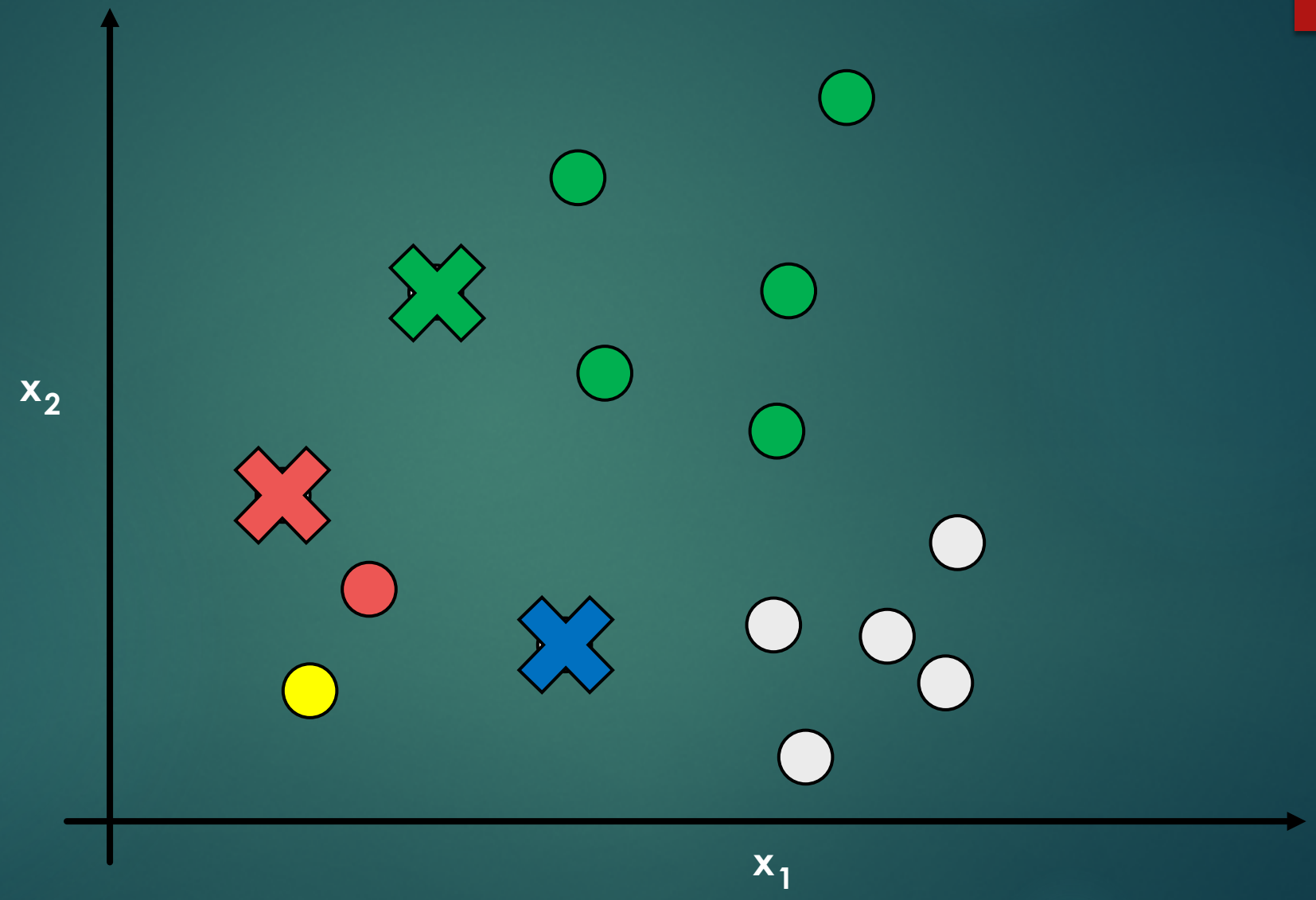

x_2
x_1

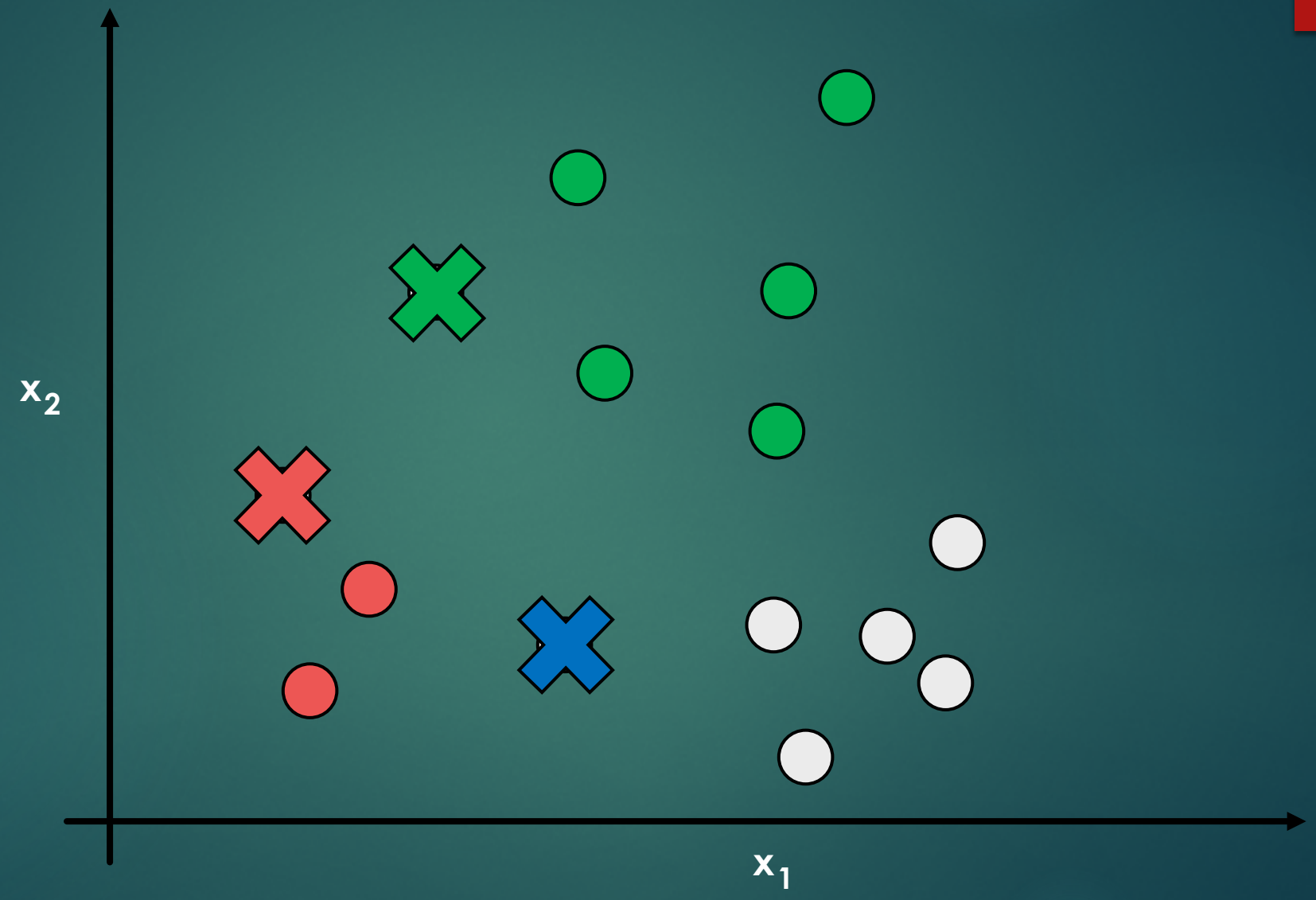

x_2
x_1

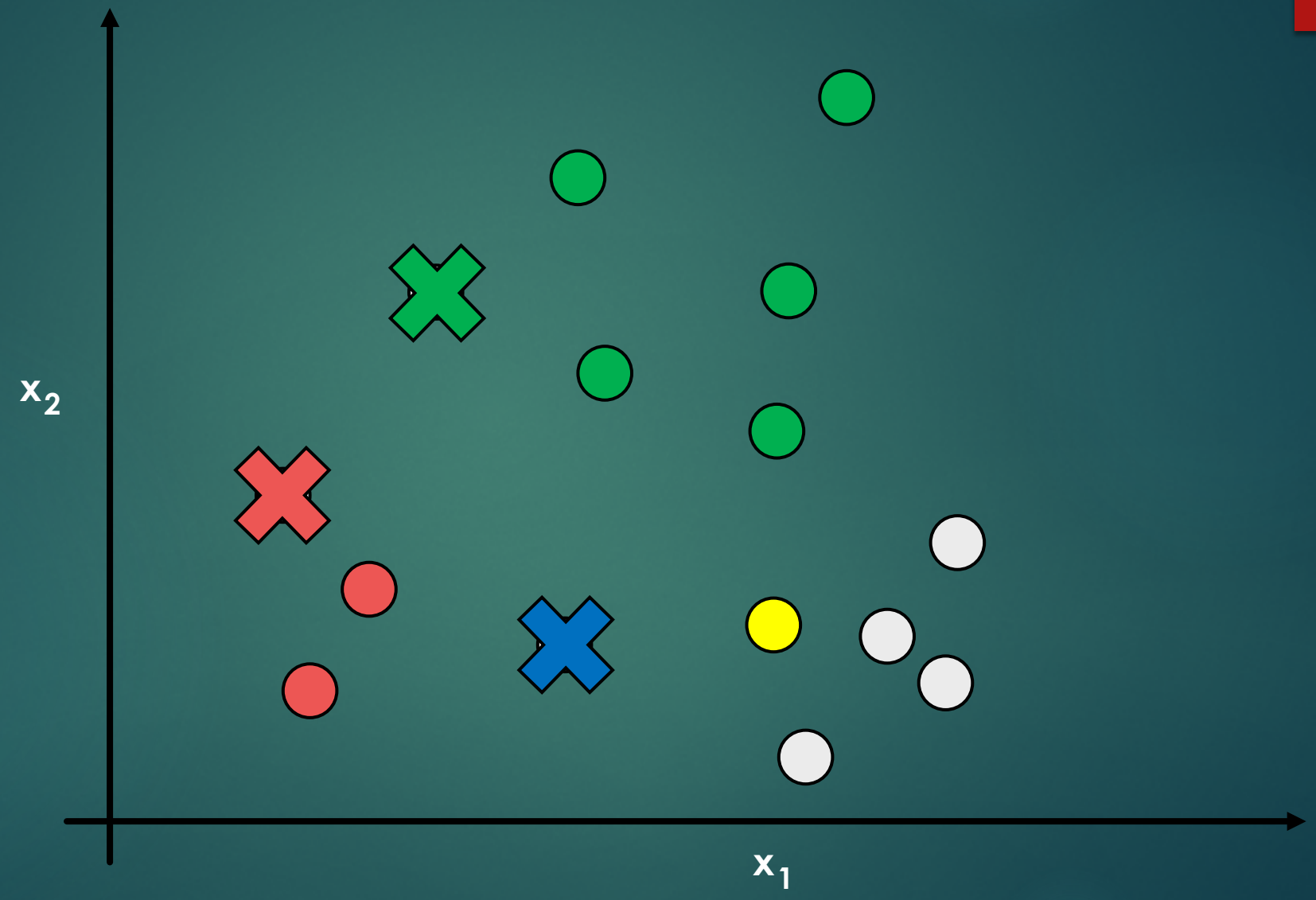

x_2
x_1

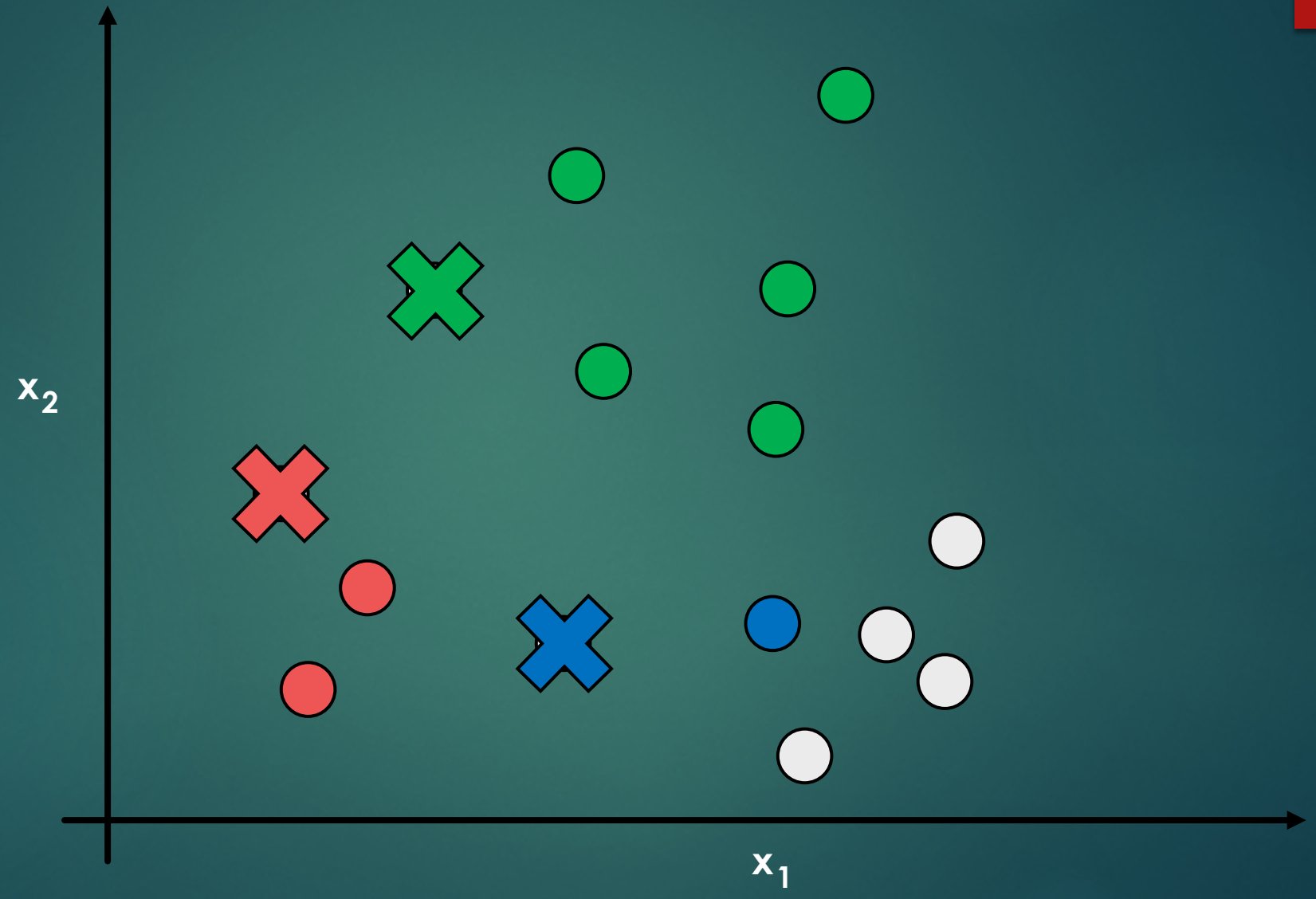

x_2
x_1

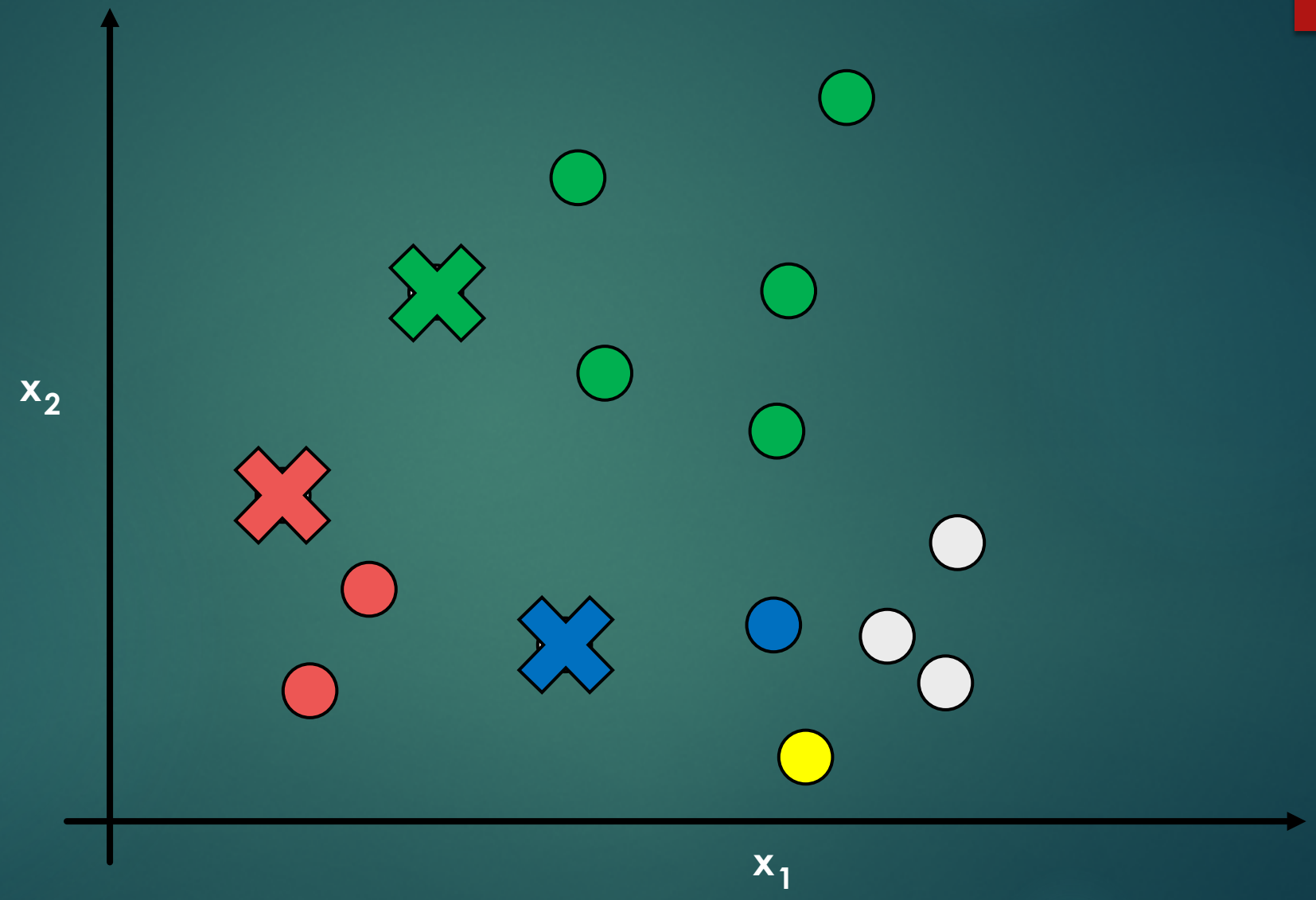

x_2
x_1

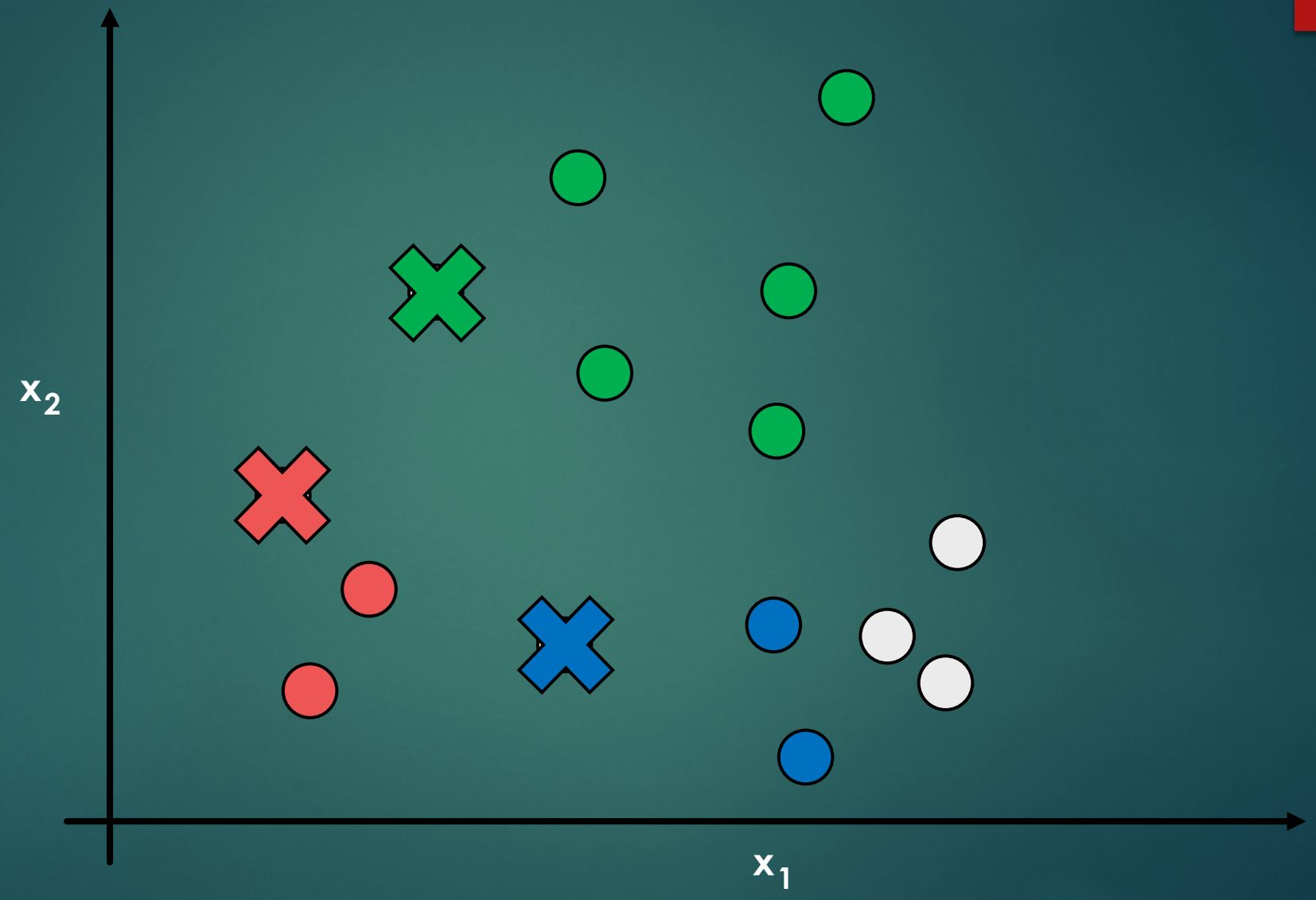

x_2
x_1

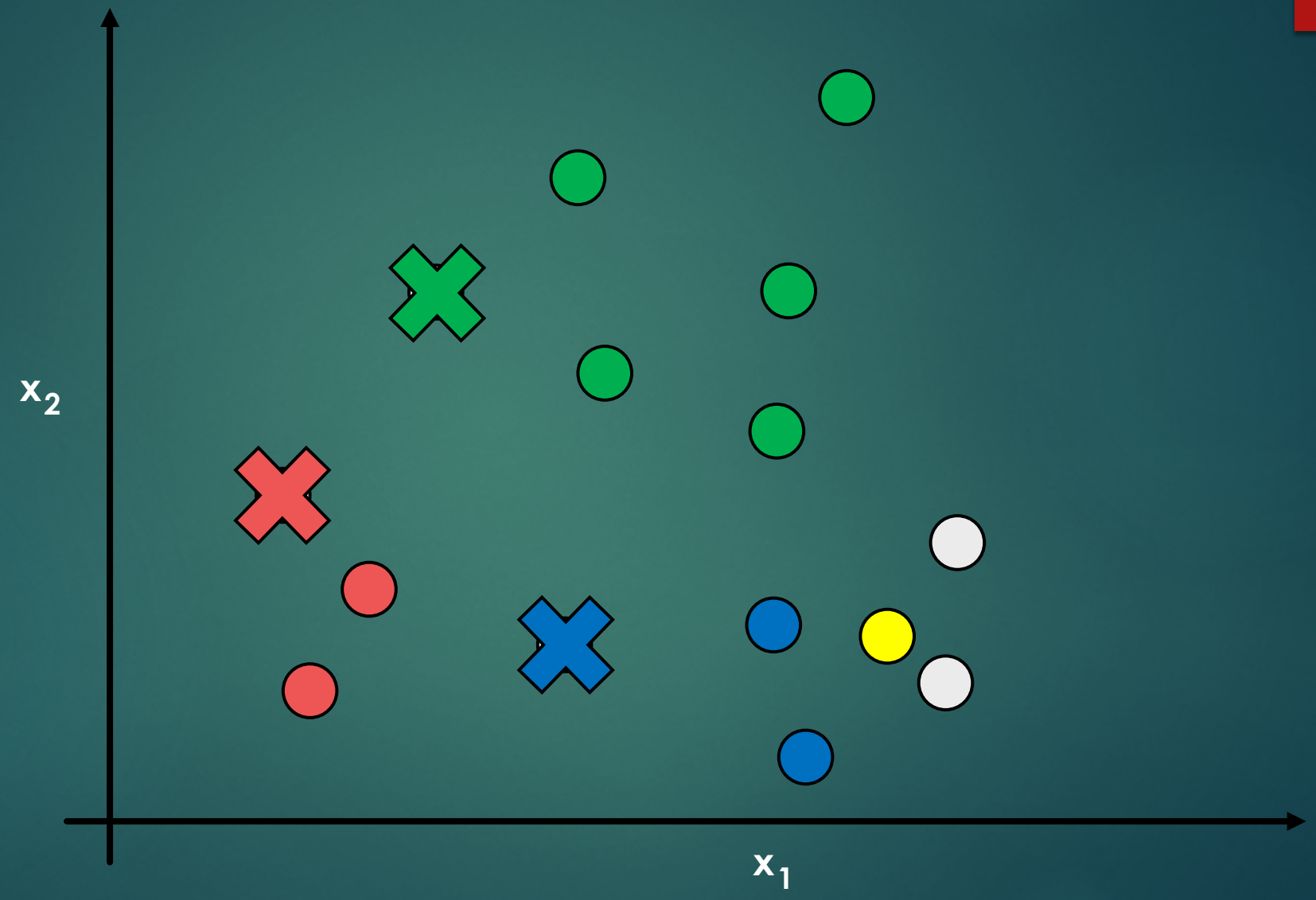

x_2
x_1

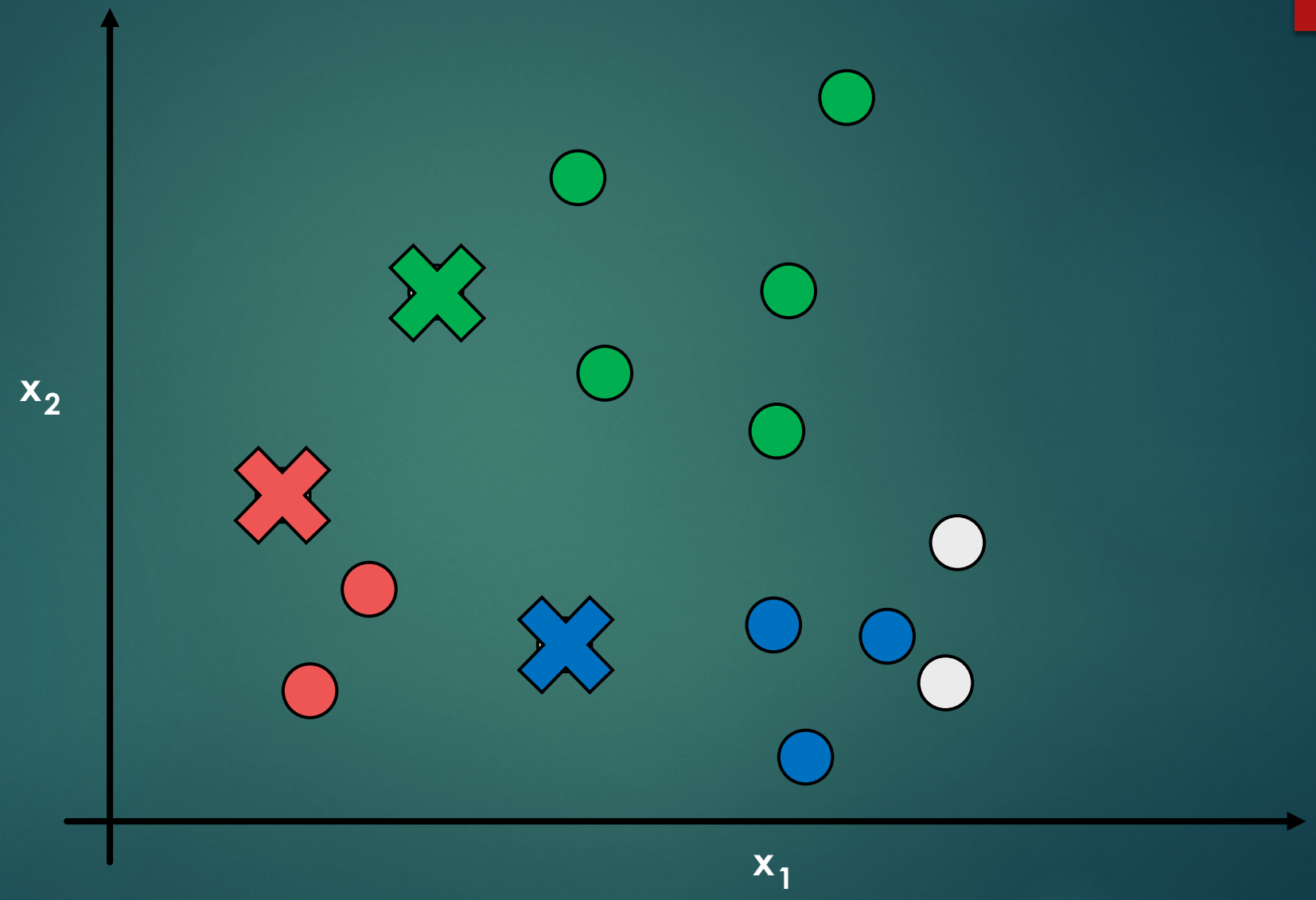

x_2
x_1

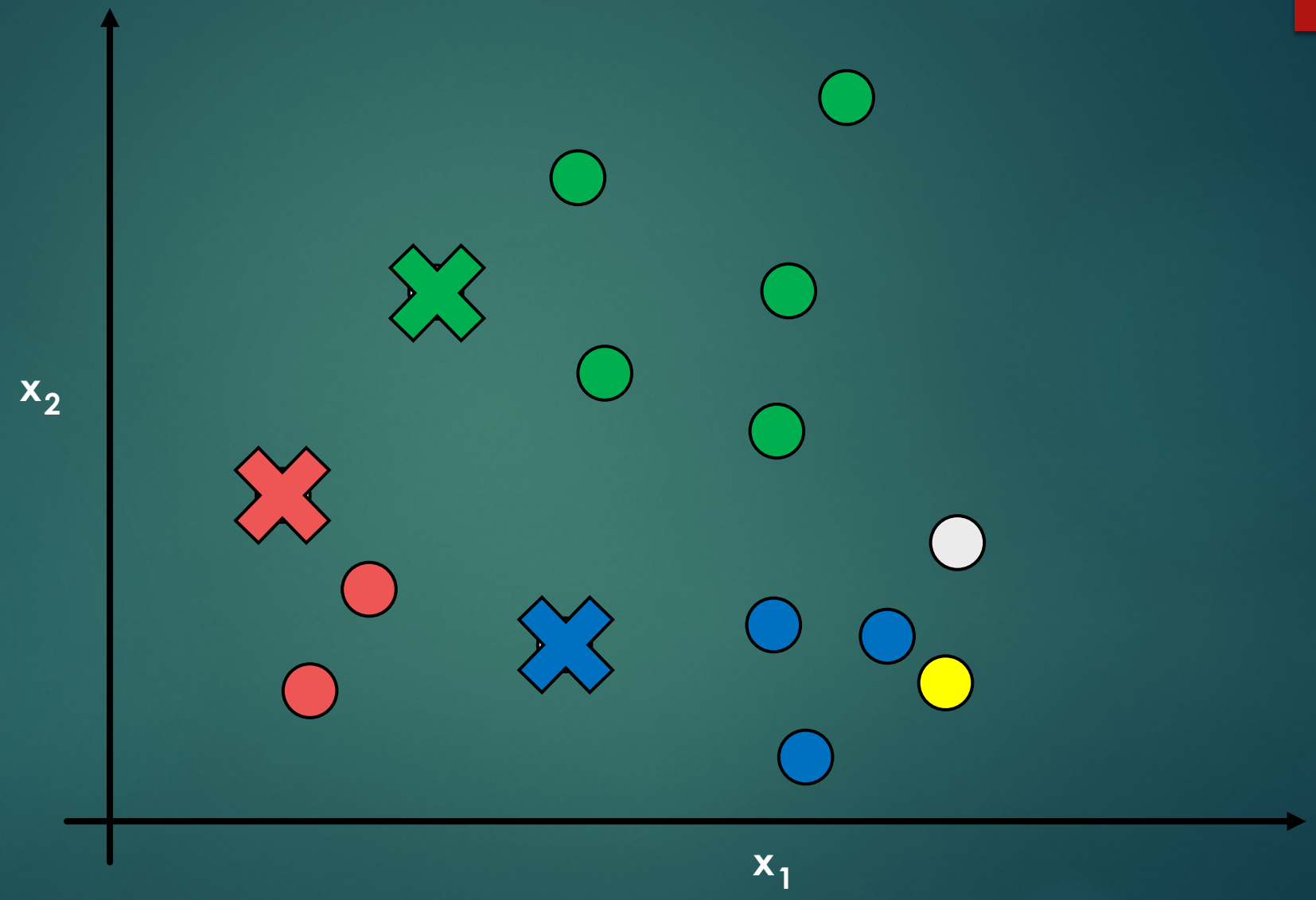

x_2
x_1

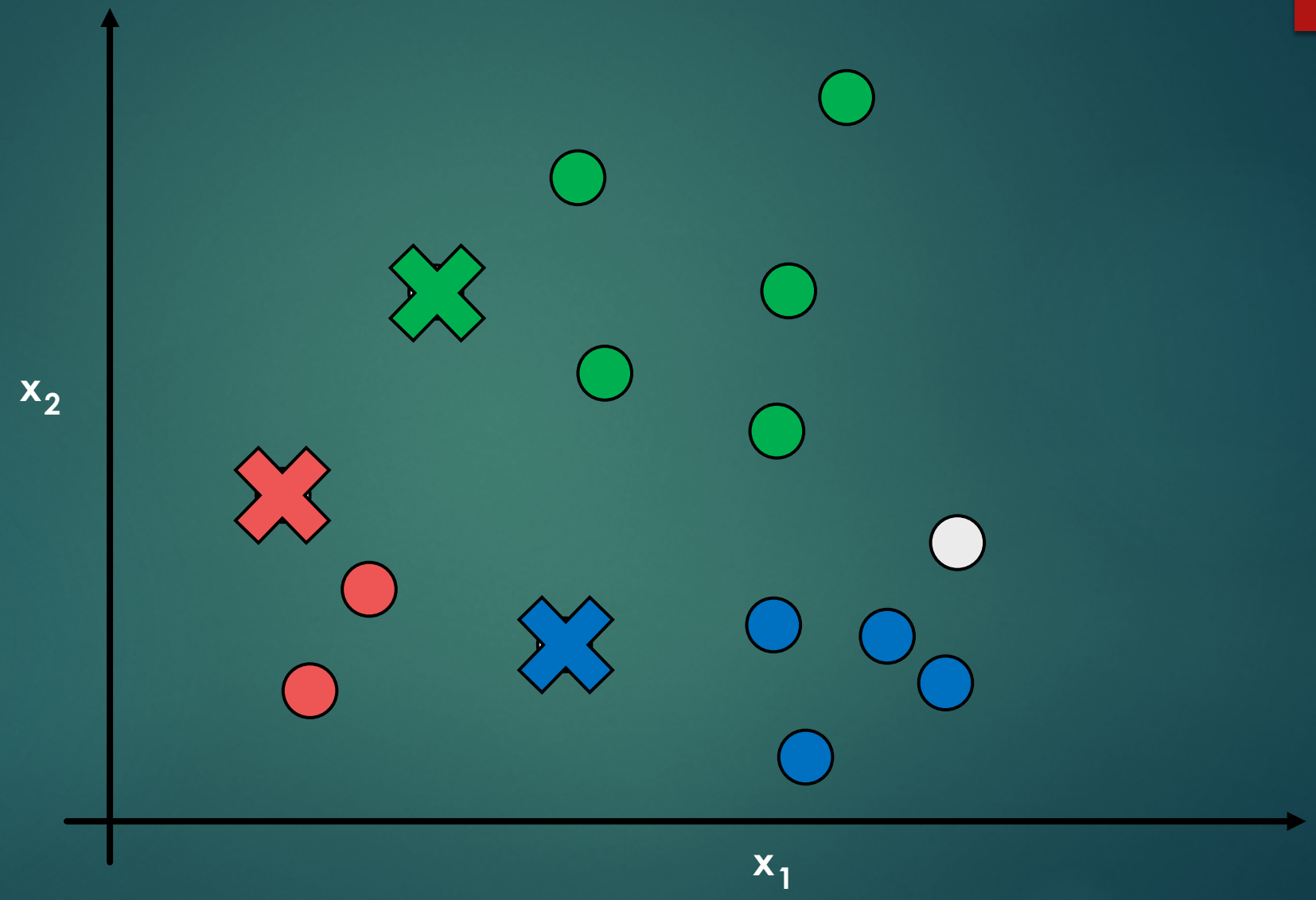

x_2
x_1

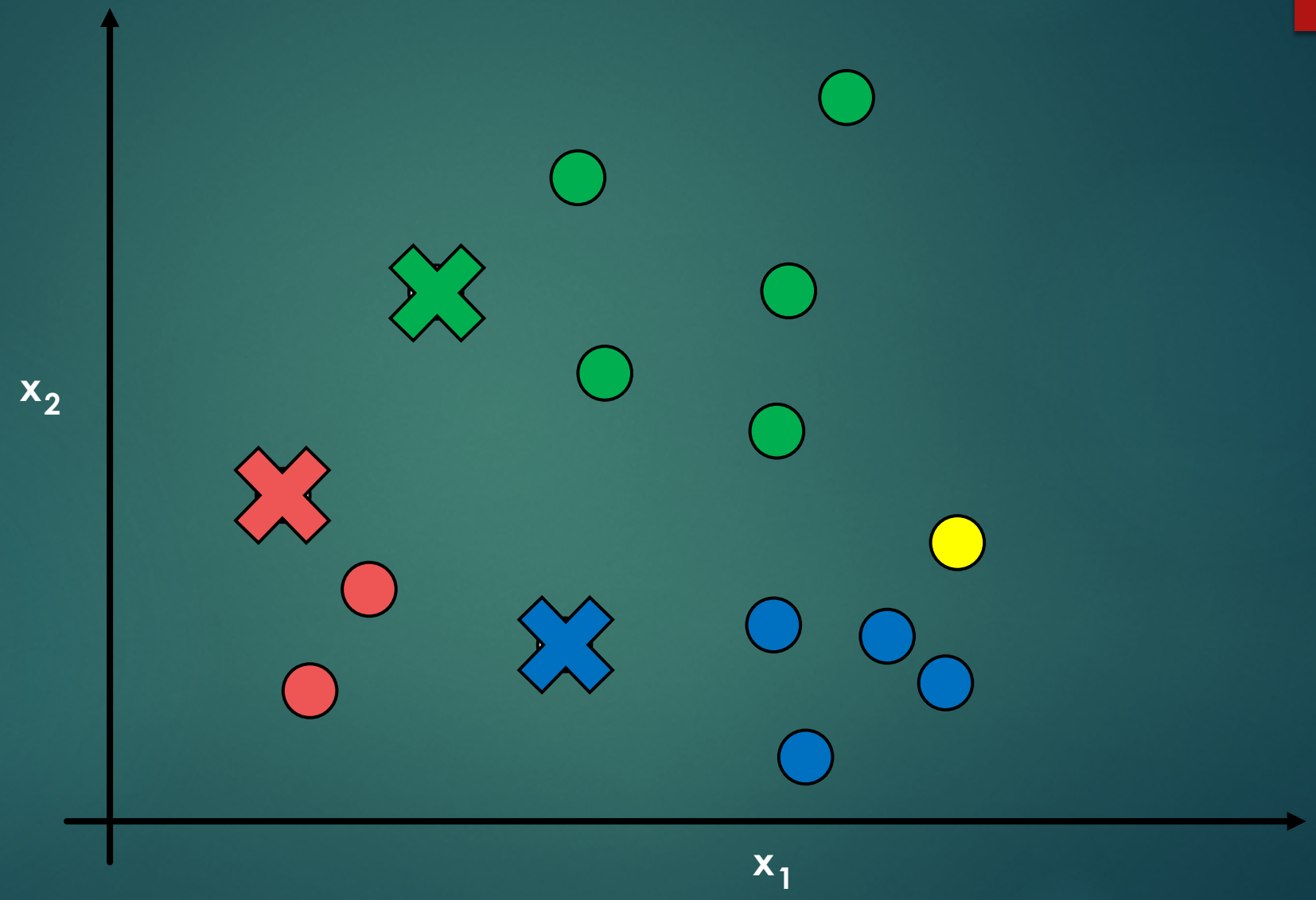

x_2
x_1

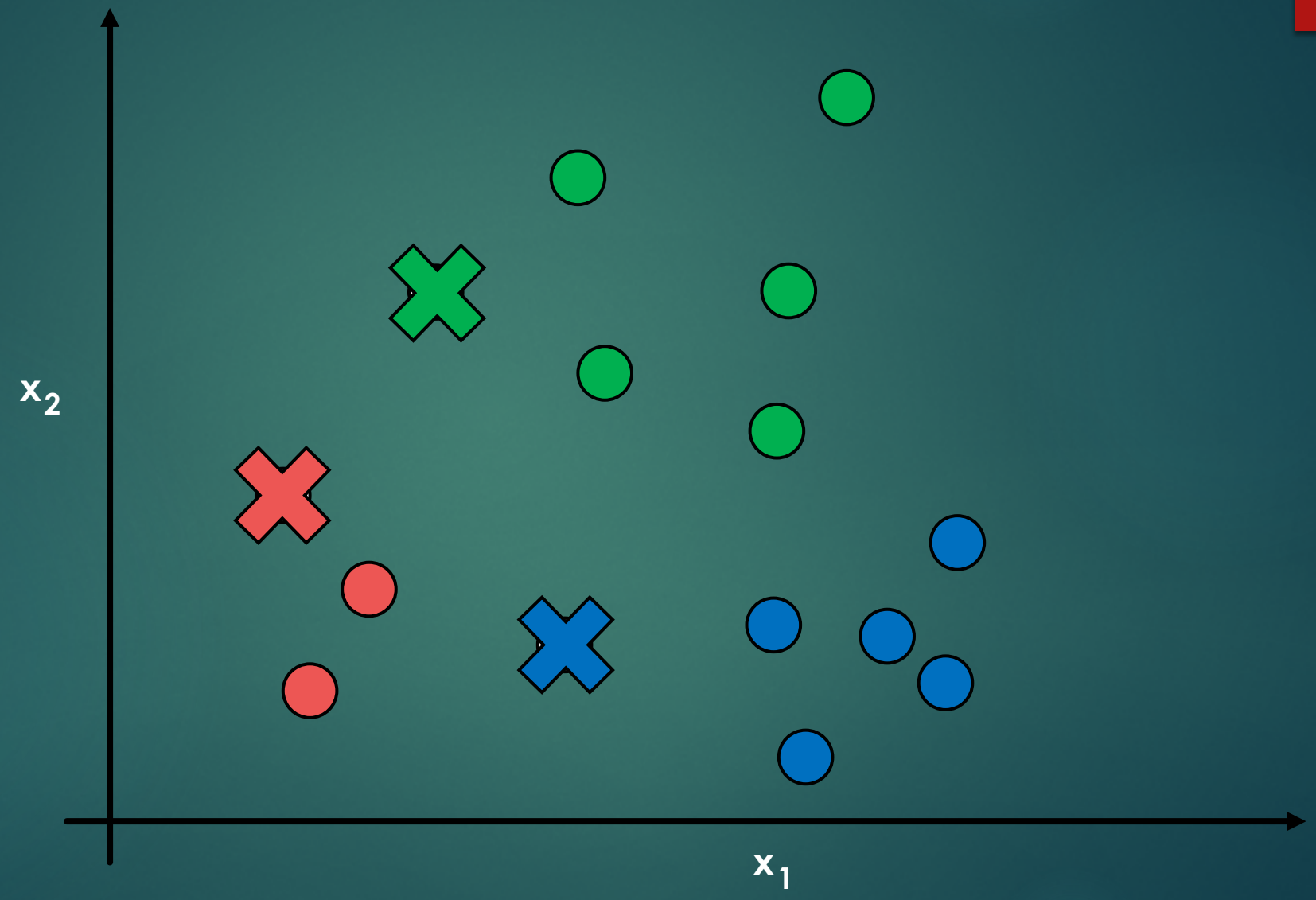

x_2
x_1


x_2
x_1

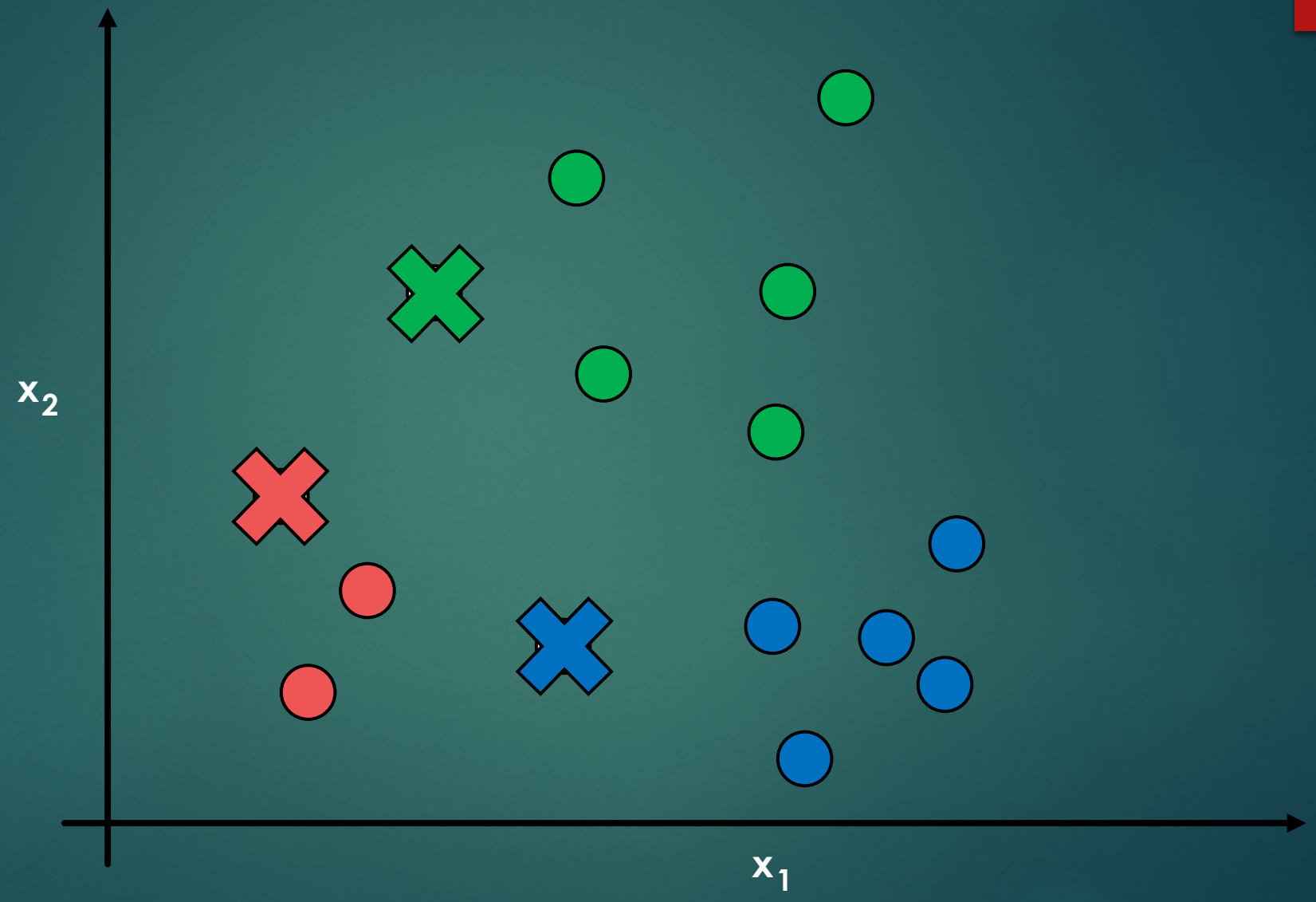

x_2
x_1

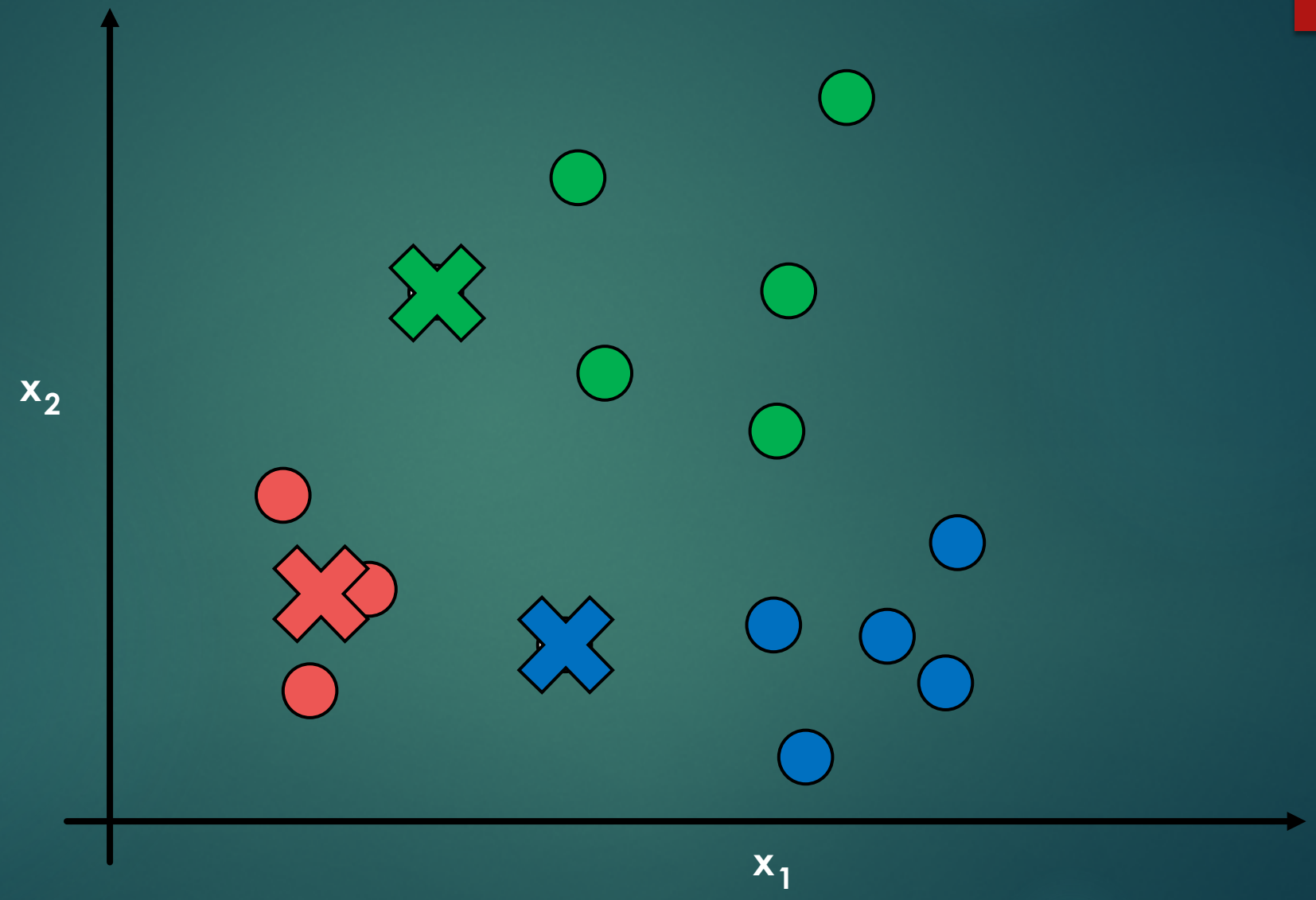

x_2
x_1

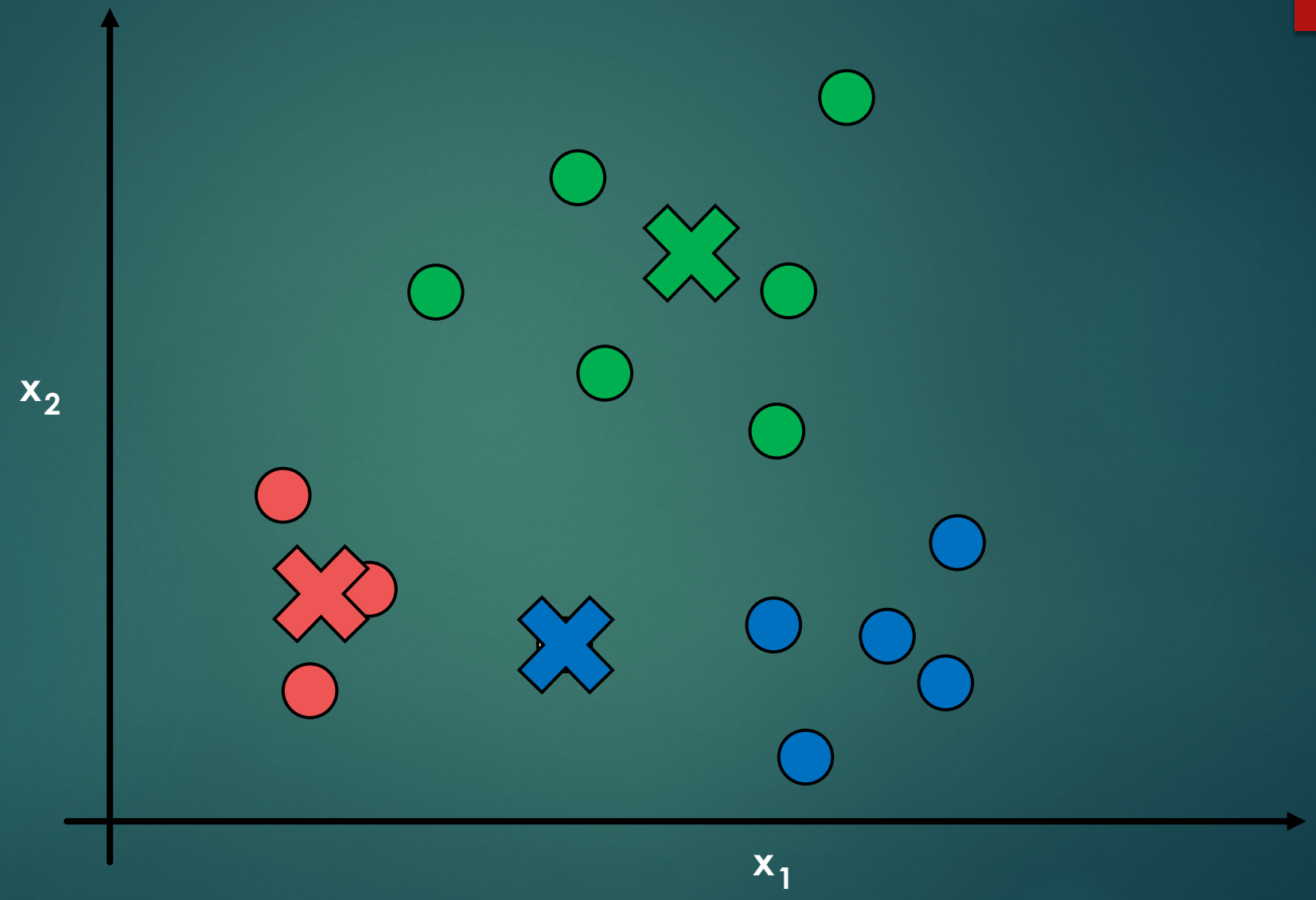

x_2
x_1

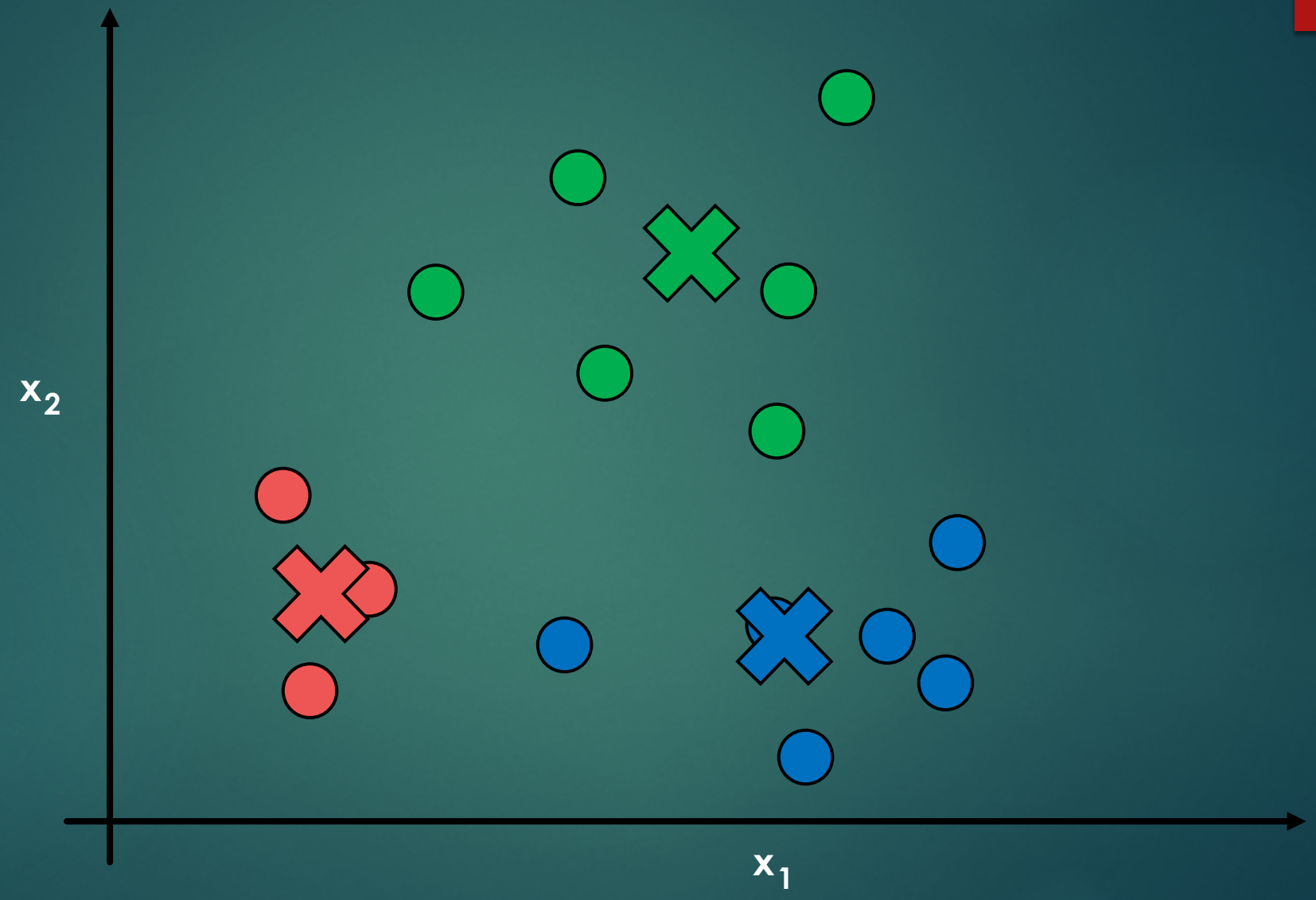

x_2
x_1

WE HAVE TO UPDATE THE CENTROIDS !!!

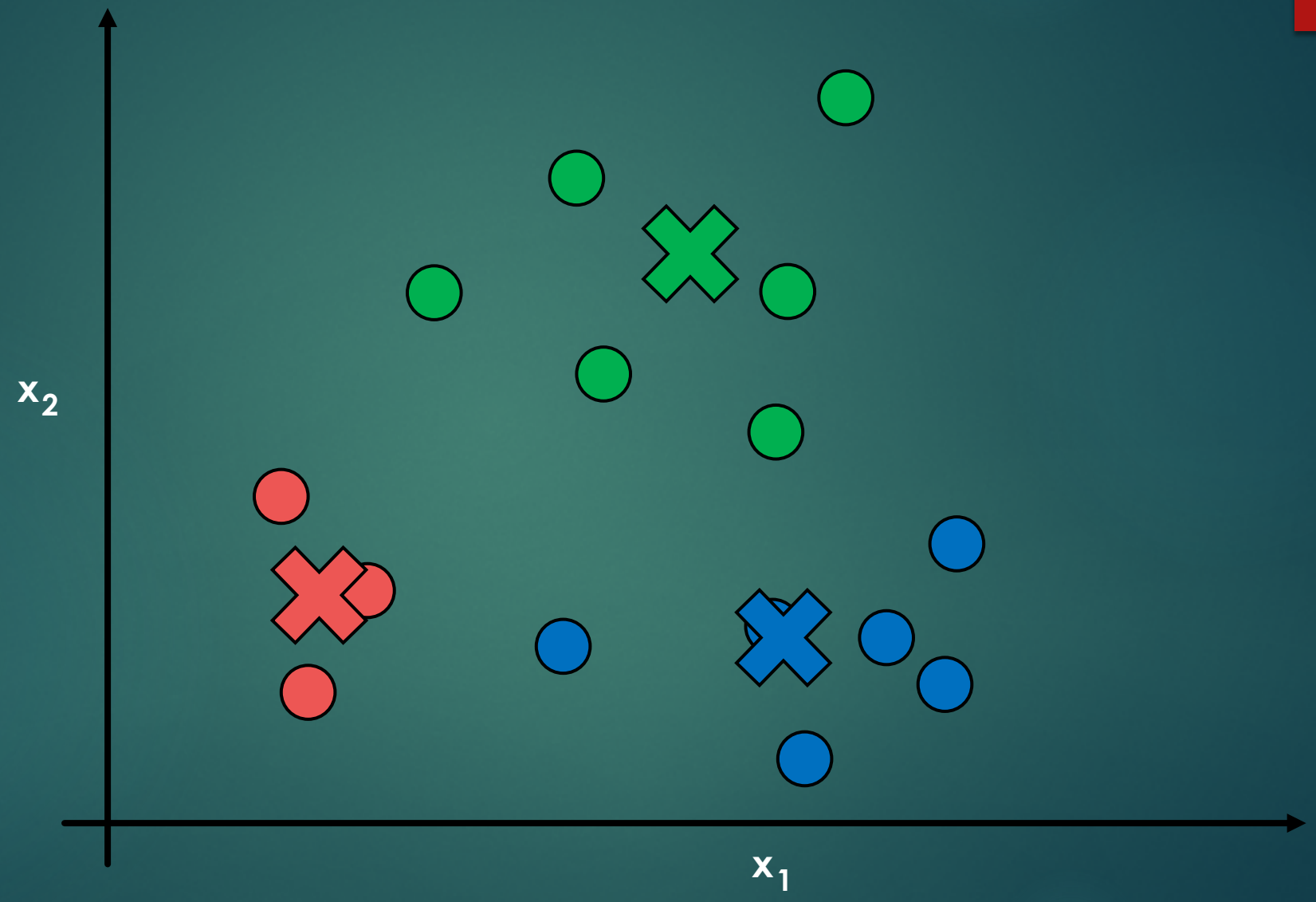



x_2
x_1

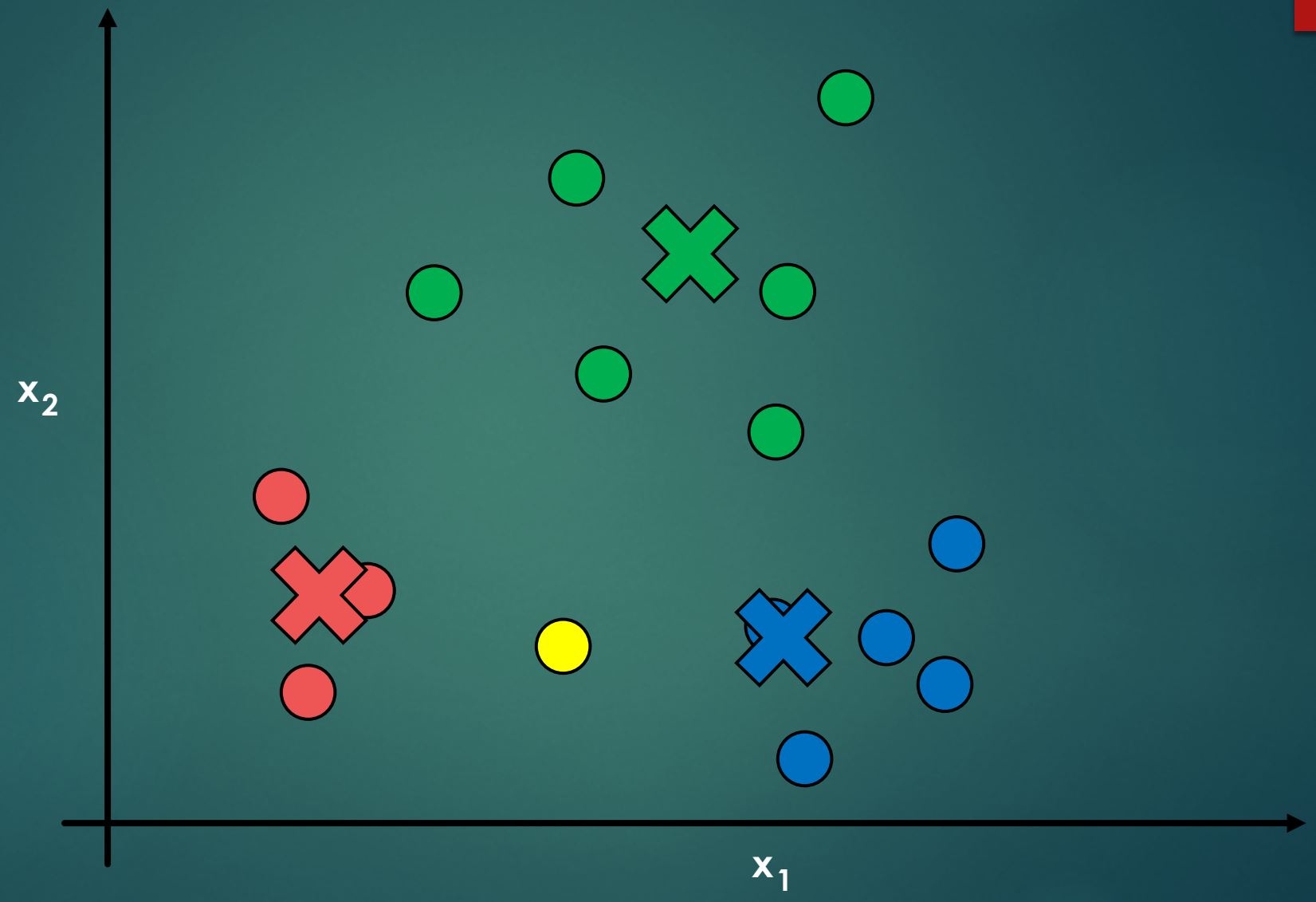

x_2
x_1


x_2
x_1

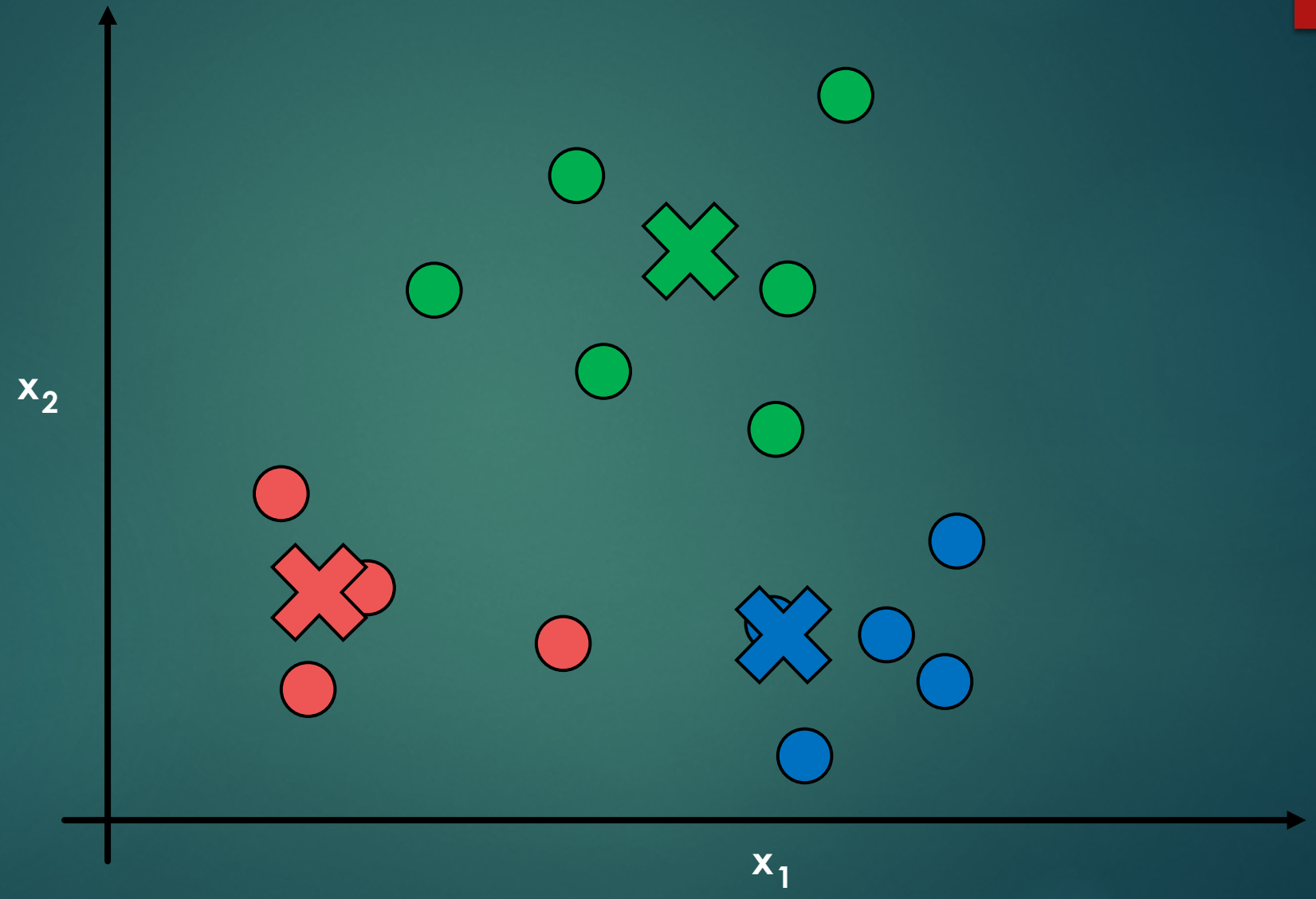
AND AGAIN WE HAVE TO CONSIDER THE DOTS !!!



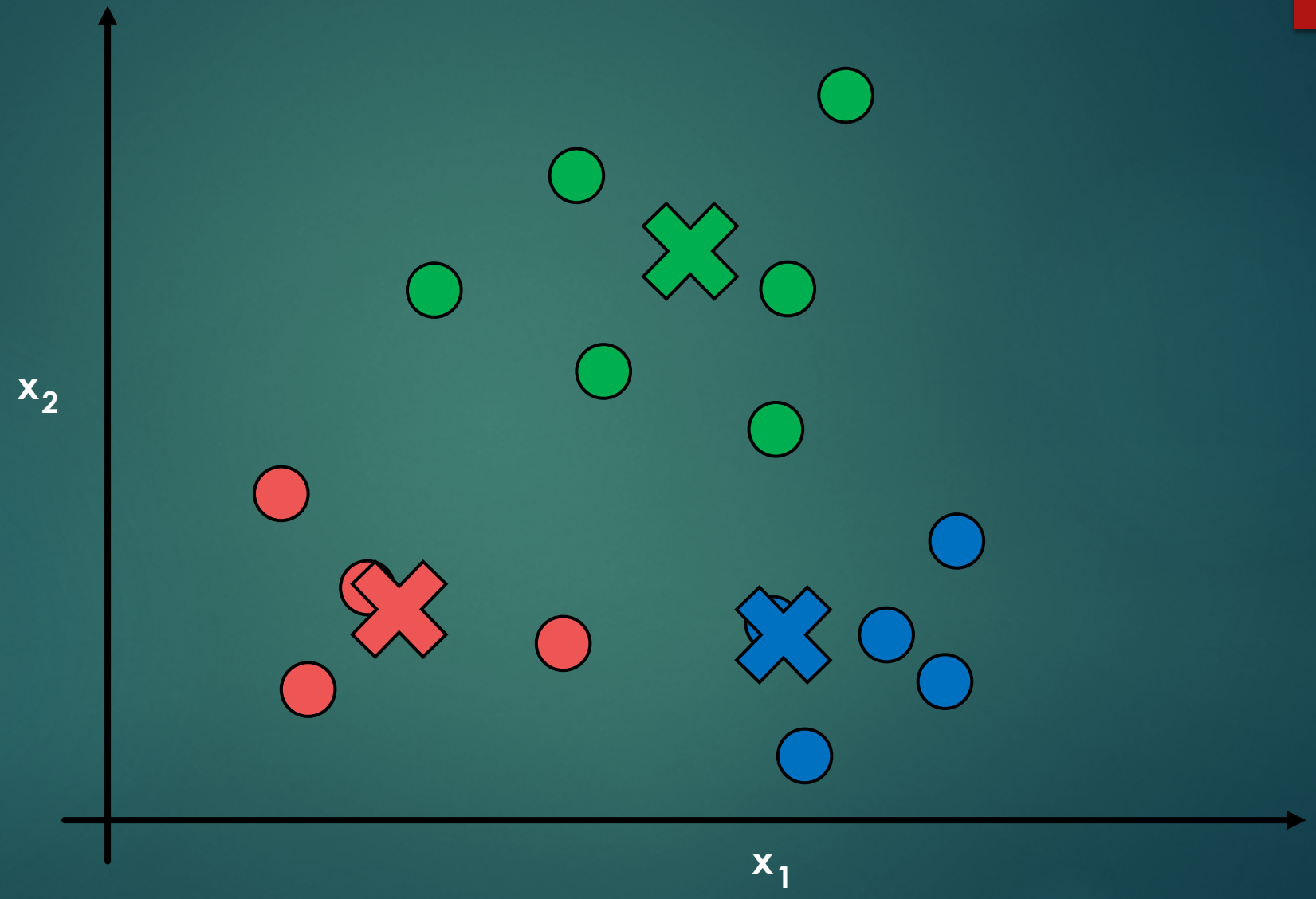
AND AGAIN WE HAVE TO CONSIDER THE DOTS !!!



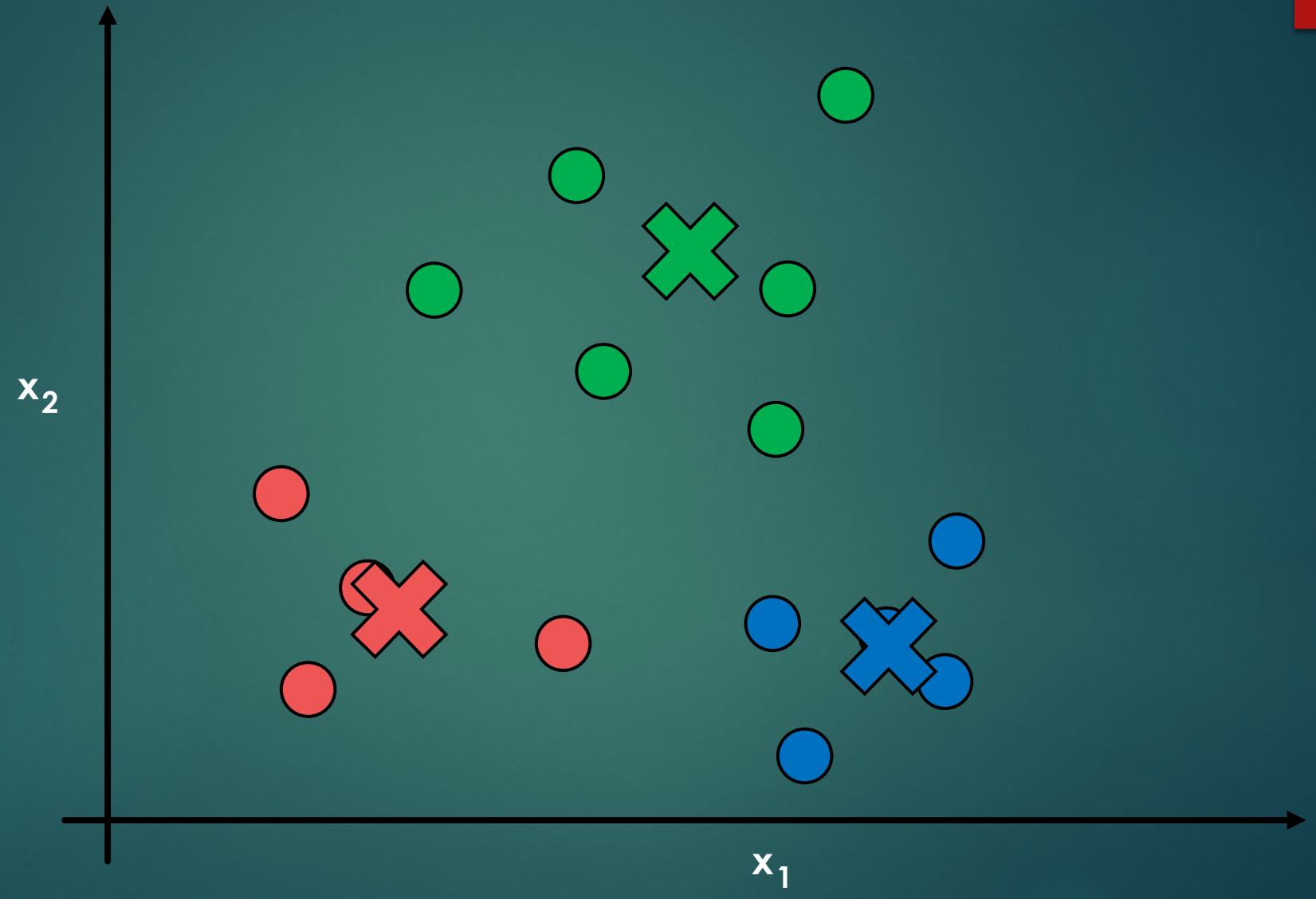
AND AGAIN WE HAVE TO CONSIDER THE DOTS !!!



AND AGAIN WE HAVE TO CONSIDER THE DOTS !!!



AND AGAIN WE HAVE TO CONSIDER THE DOTS !!!

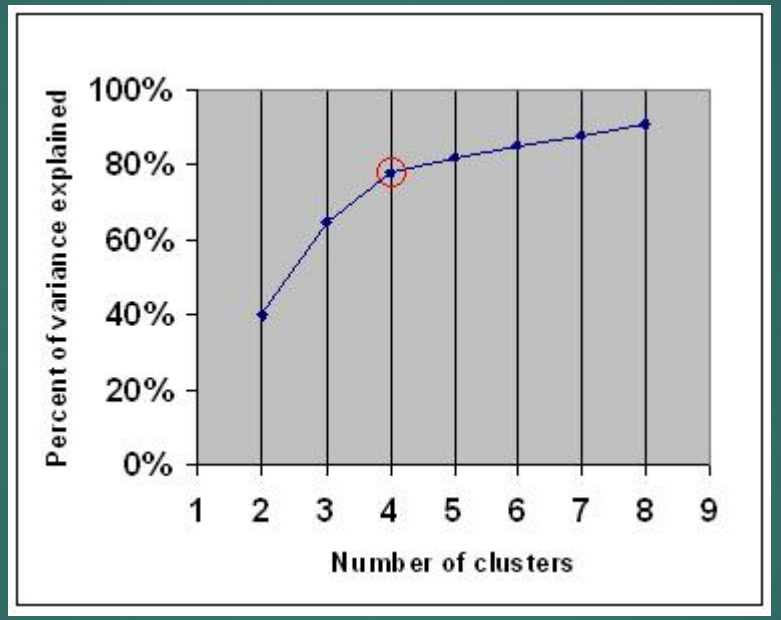


Finding k parameter

- Sometimes we have some a priori knowledge: we know how many clusters we want to construct
- Without any a priori knowledge: **k** is approximately equal to the square root of $\frac{\mathbf{n}}{\mathbf{2}}$

  // **n** is the number of elements in the dataset
- Elbow method: we monitor the change of homogeneity within the clusters with different **k** values
- It looks at the percentage of variance explained as a function of the number of clusters: one should choose a number of clusters so that adding another cluster does not give much better modeling of the data
- We have to find the „elbow point" at a plot

Finding k parameter



advantages	**disadvantages**
Relies on simple principles to identify clusters Flexible Efficient	Not so sophisticated Because it uses an element of random chance, it is not guaranteed to find the optimal set of clusters **k** parameter → we have to know in advance how many clusters we want to find

Clustering and classification

- Clustering is different from classification or numerical predictions
- Classification / regression: the result is a model that relates features to an outcome
- Clustering: creates new data !!!
- Unlabeled examples are given a cluster label and inferred entirely from the relationships within the data

MACHINE LEARNING

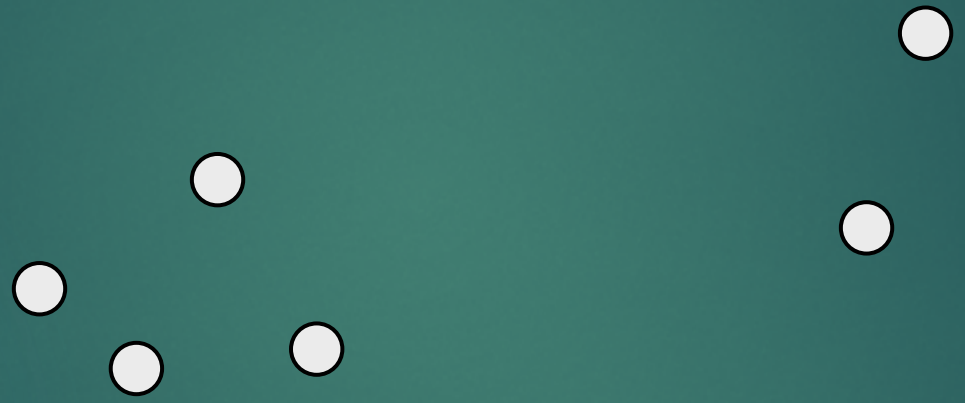
DBSCAN CLUSTERING

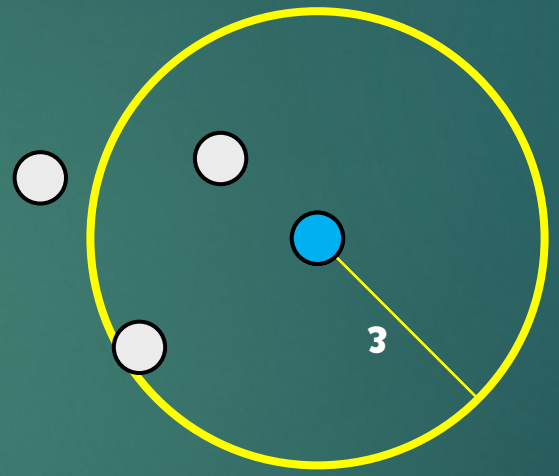
DBSCAN

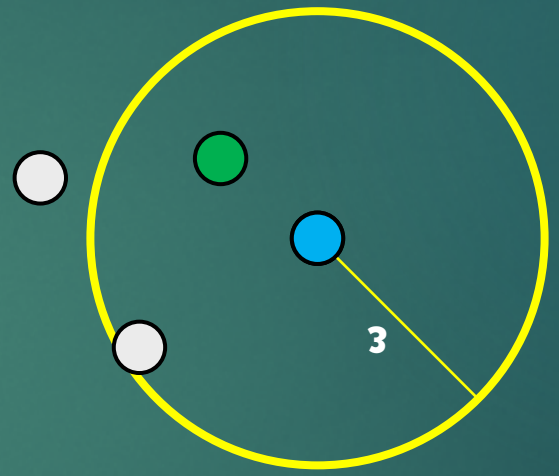
- **D**ensity-**B**ased **S**patial **C**lustering of **A**pplications with **N**oise (**DBSCAN**)
- Data clustering algorithms such as K-means
- Density-based → given a set of points in some space, it groups together points that are closely packed together
- Very common clustering algrithm !!!

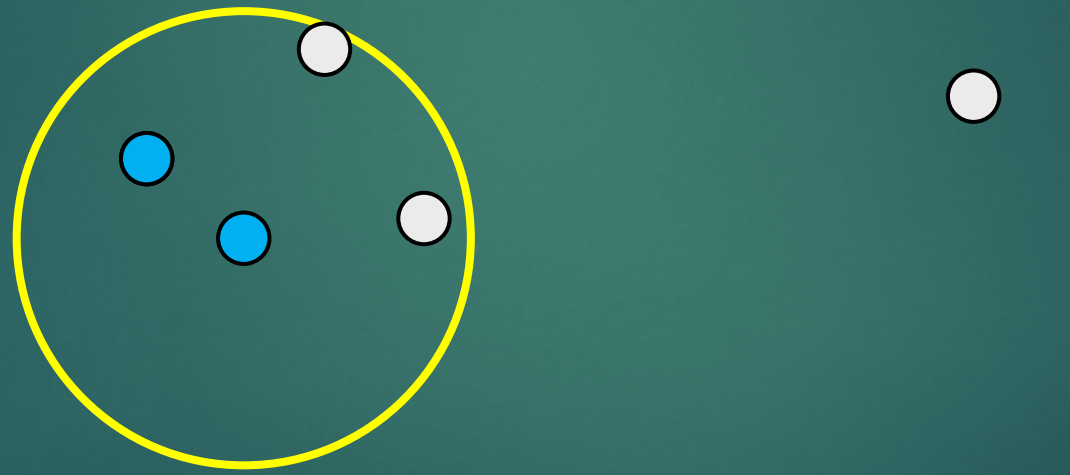
DBSCAN algorithm

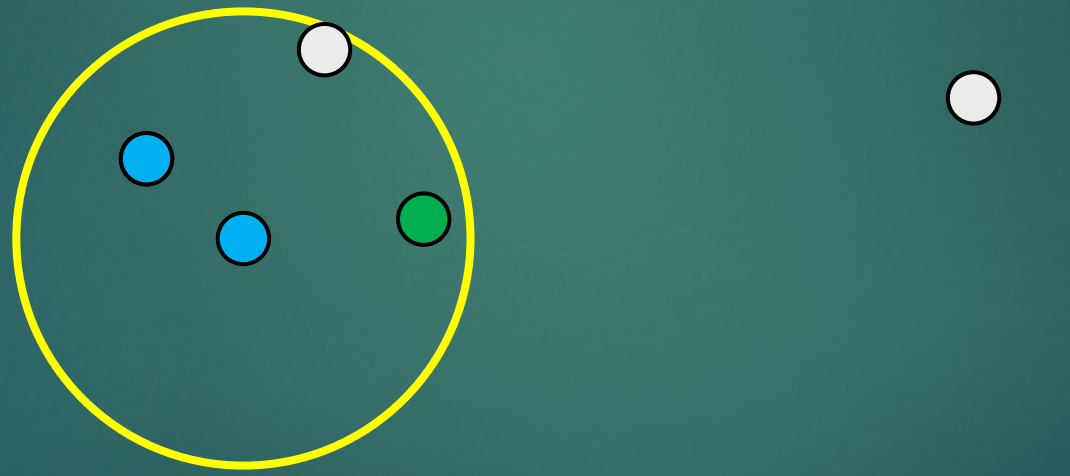
- There are given points in the 2 dimensional space
- Try to find every points → that are separated by a distance no more than a given **ε** epsilon (the threshold distance)
- Same clusters: we can hop from a given node to another by hopping no more than **ε** epsilon → the points are in the same cluster

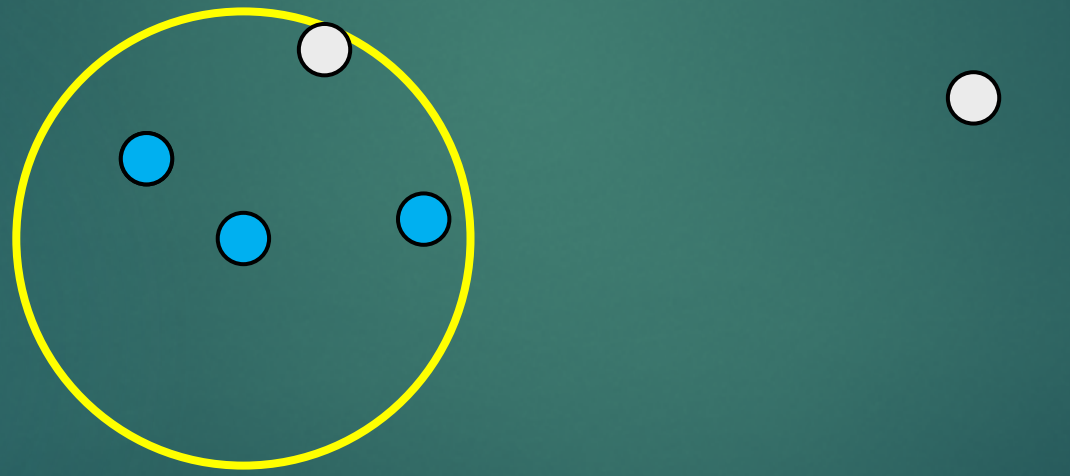


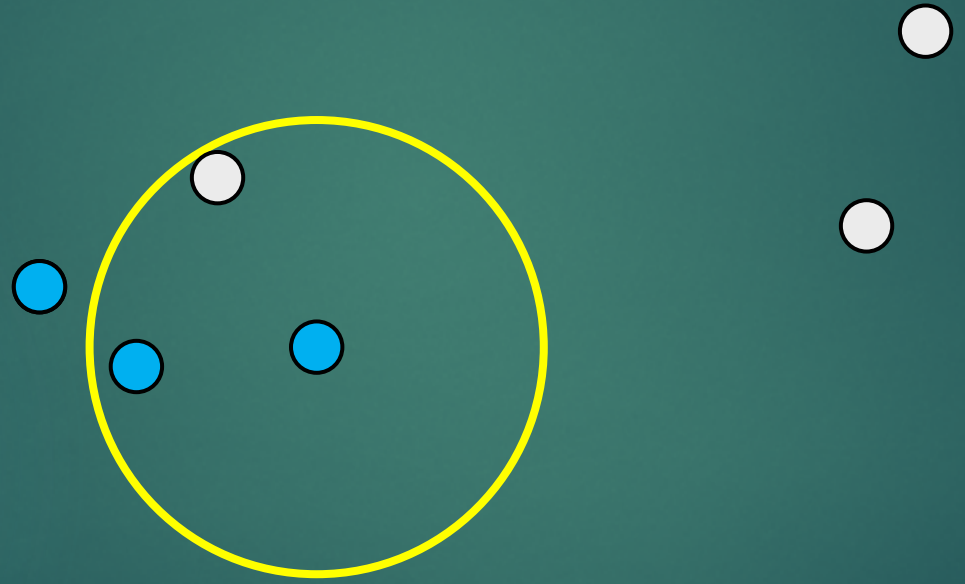

3

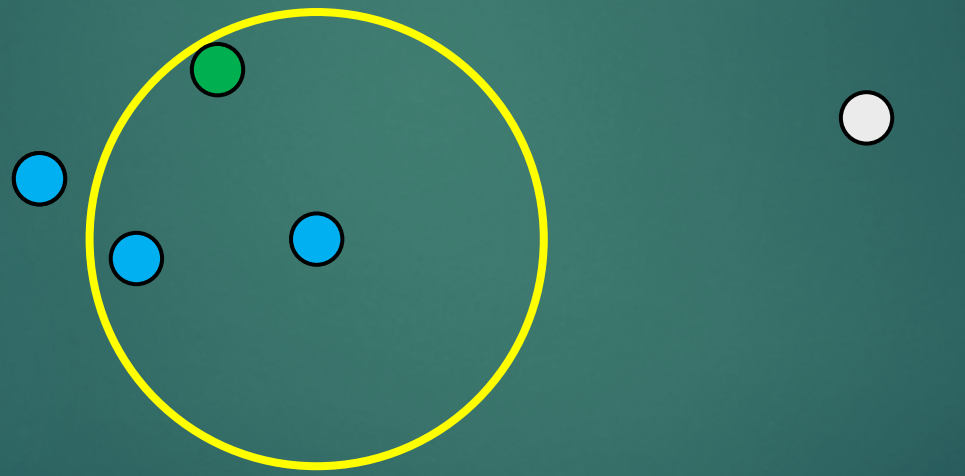

3

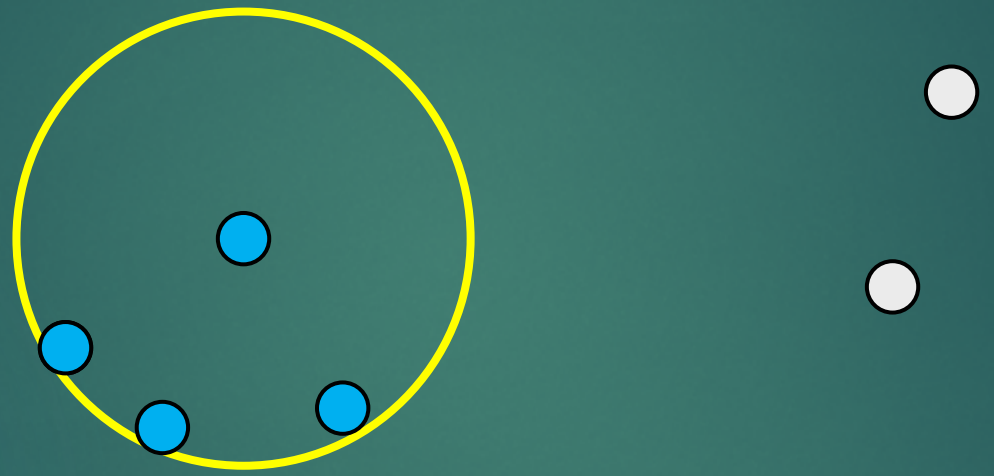


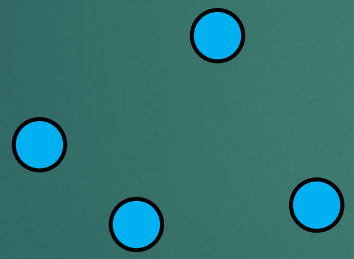


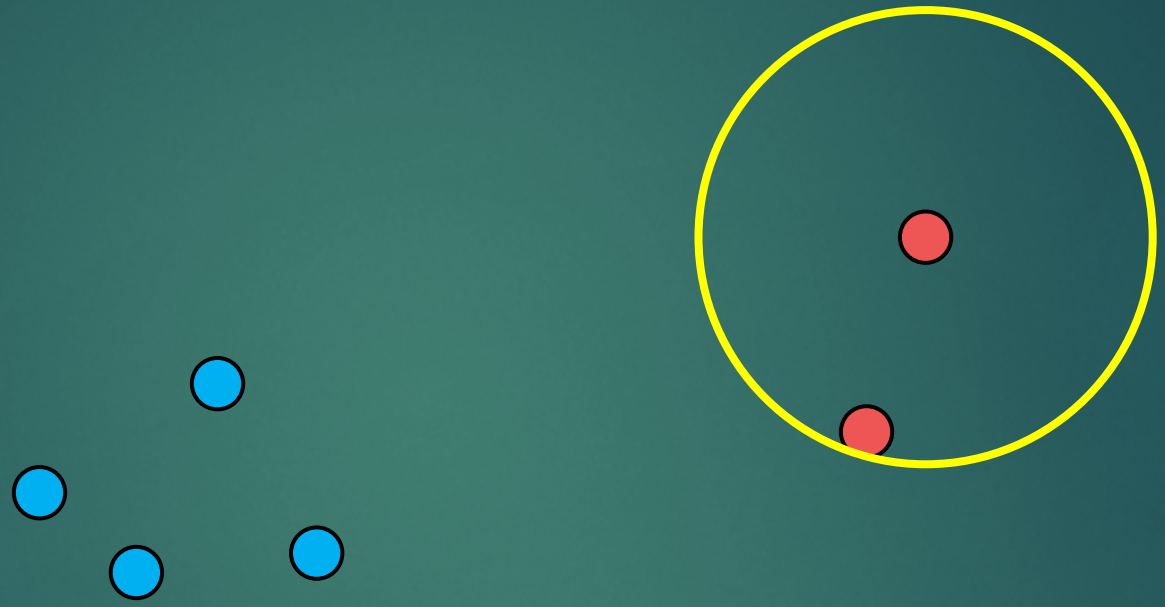


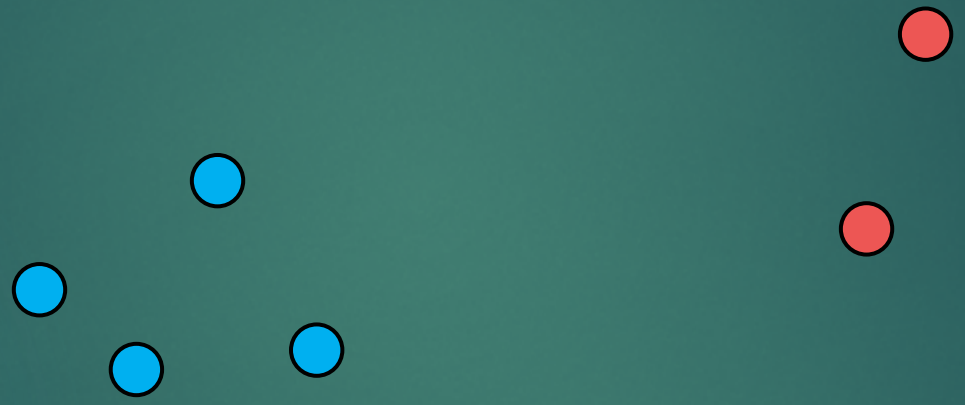












Advantages

- Finds non-linearly separable clusters (arbitrarily shaped clusters) !!!
- For K-means we have to specify the number of clusters we want to find → here we do not need to do so
- Very robust to outliers
- The result does not depend on the starting conditions
- Parameters: **ε** epsilon (distance threshold) + minimum number of neighbors
- **O(N logN)** running time !!!

Disadvantages

- **DBSCAN** is not entirely deterministic !!!
- Border points that are reachable from more than one cluster can be part of either cluster depending on the order the data is processed
- Relies heavily on a distance measure: Euclidean-measure. In higher dimensons it is very hard to find a good value for **ε** epsilon
- **„curse of dimensionality"**
- If the data and scale are not well understood → choosing a meaningful distance threshold **ε** epsilon can be difficult

MACHINE LEARNING

HIERARCHICAL CLUSTERING

Hierarchical clustering

- Huge disadvantage of **k-means** clustering: we have to specify the **k** parameter in advance
- Here we do not have to do so !!!
- We build a tree like structure out of the data which „contains" all the **k** parameters

Hierarchical clustering

- In data mining, hierarchical clustering is a method of cluster analysis which seeks to build a hierarchy of clusters
- Agglomerative approach → this is a "bottom up" approach, each observation starts in its own cluster, and pairs of clusters are merged as one moves up the hierarchy
- In general, the merges and splits are determined in a greedy manner.
- The results of hierarchical clustering are usually presented in a dendrogram

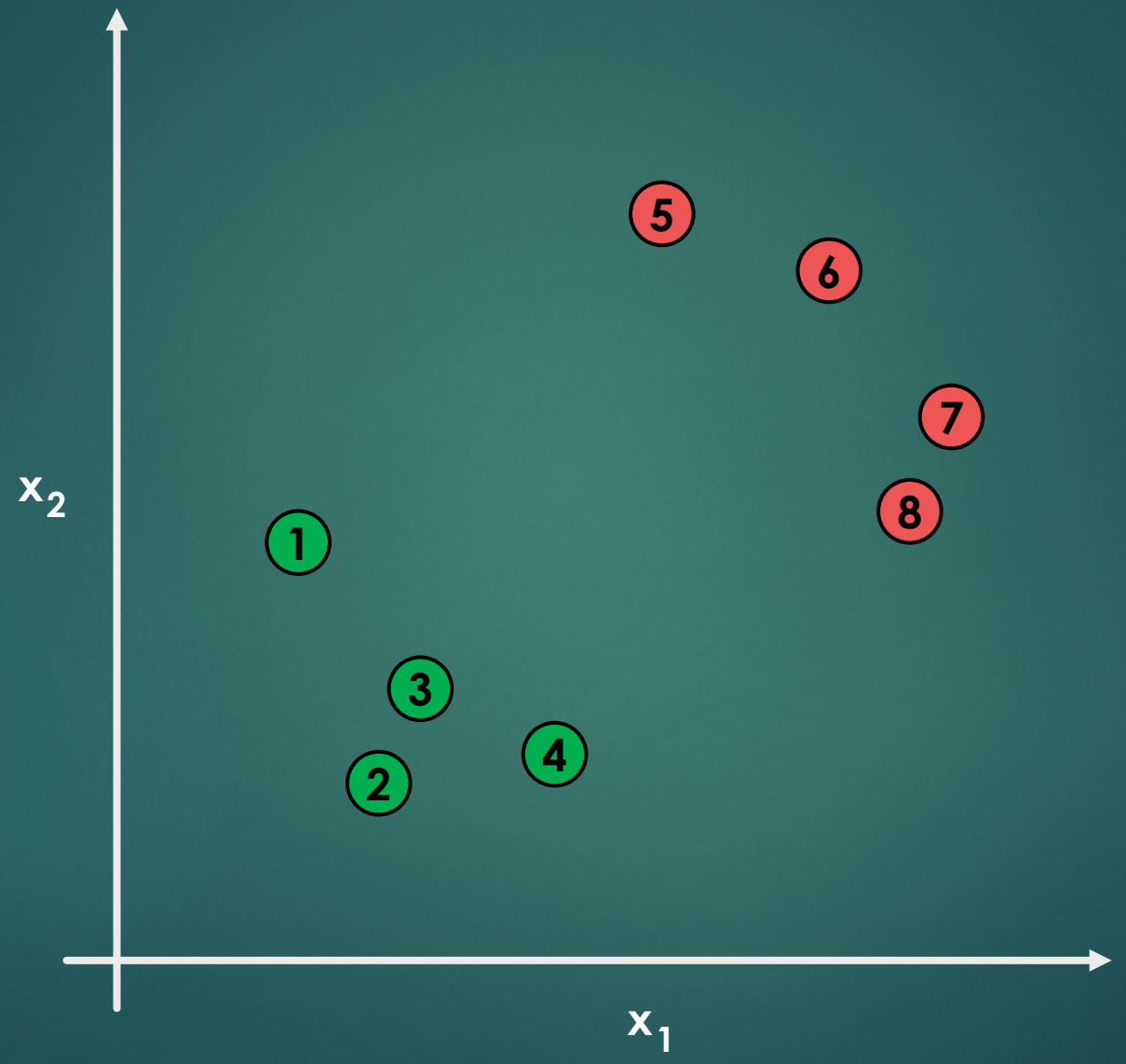
Algorithm

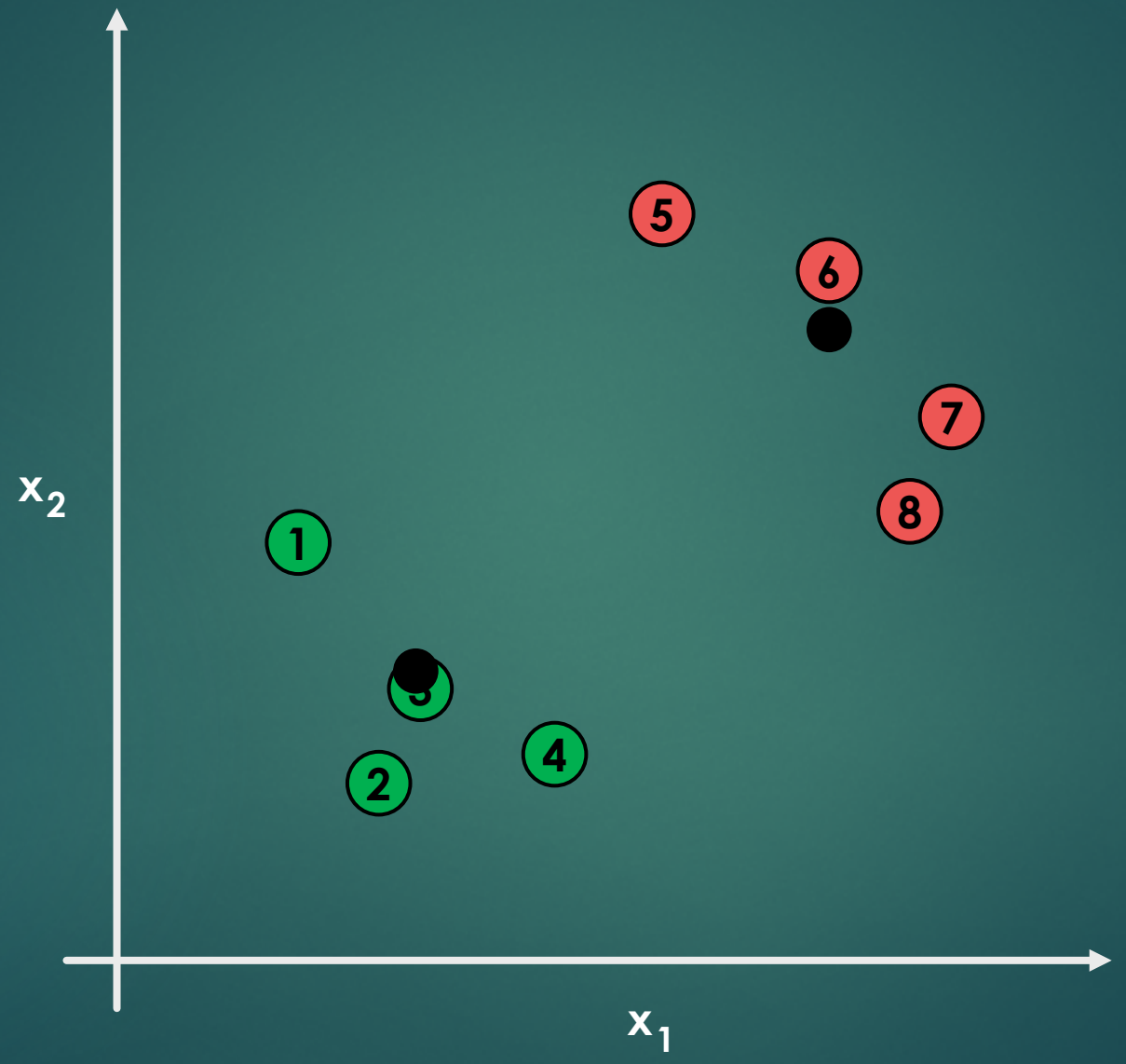
- **1.)** start each node in its own cluster ... this is sort of an initialization phase
- **2.)** find the two closest clusters and merge them together
- **3.)** repeat the algorithm until all the points are in the same cluster

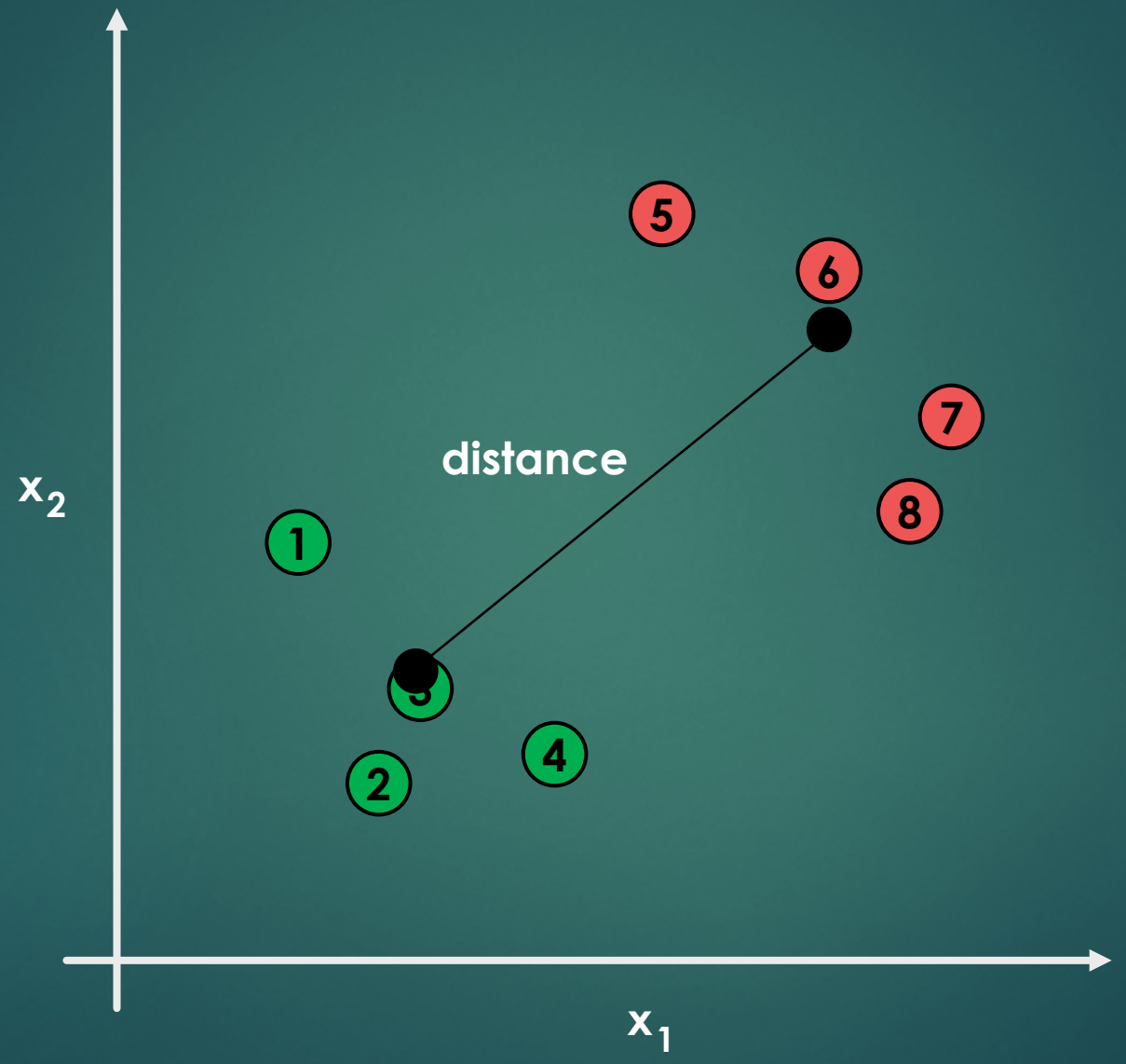
  // so there is only a single cluster left

Algorithm

- How are we able to measure the distance of two clusters?
- We usually calculate the distance of the avarages of the clusters' elements

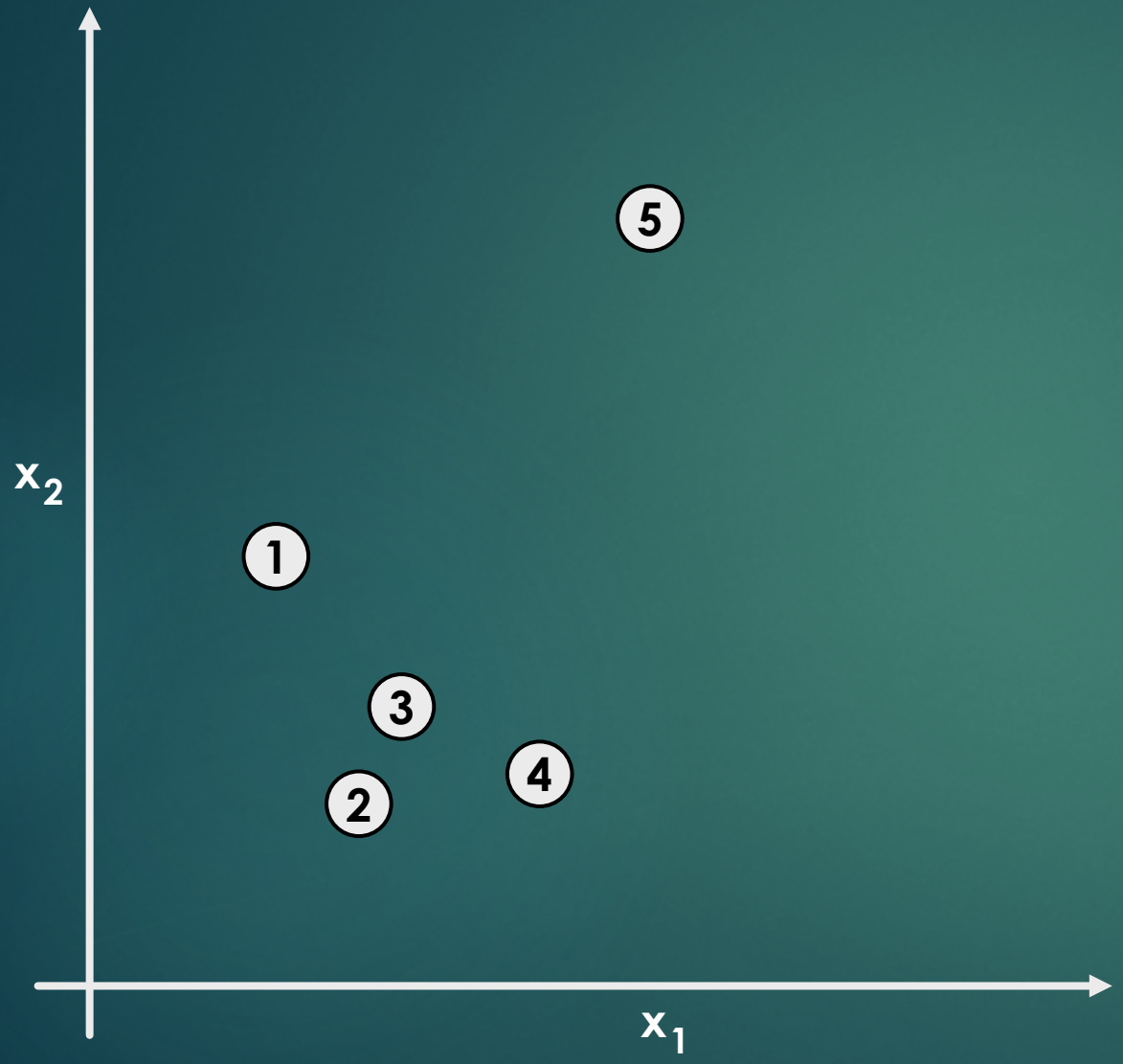

5
6
7
8
1
3
4
2
x_2
x_1

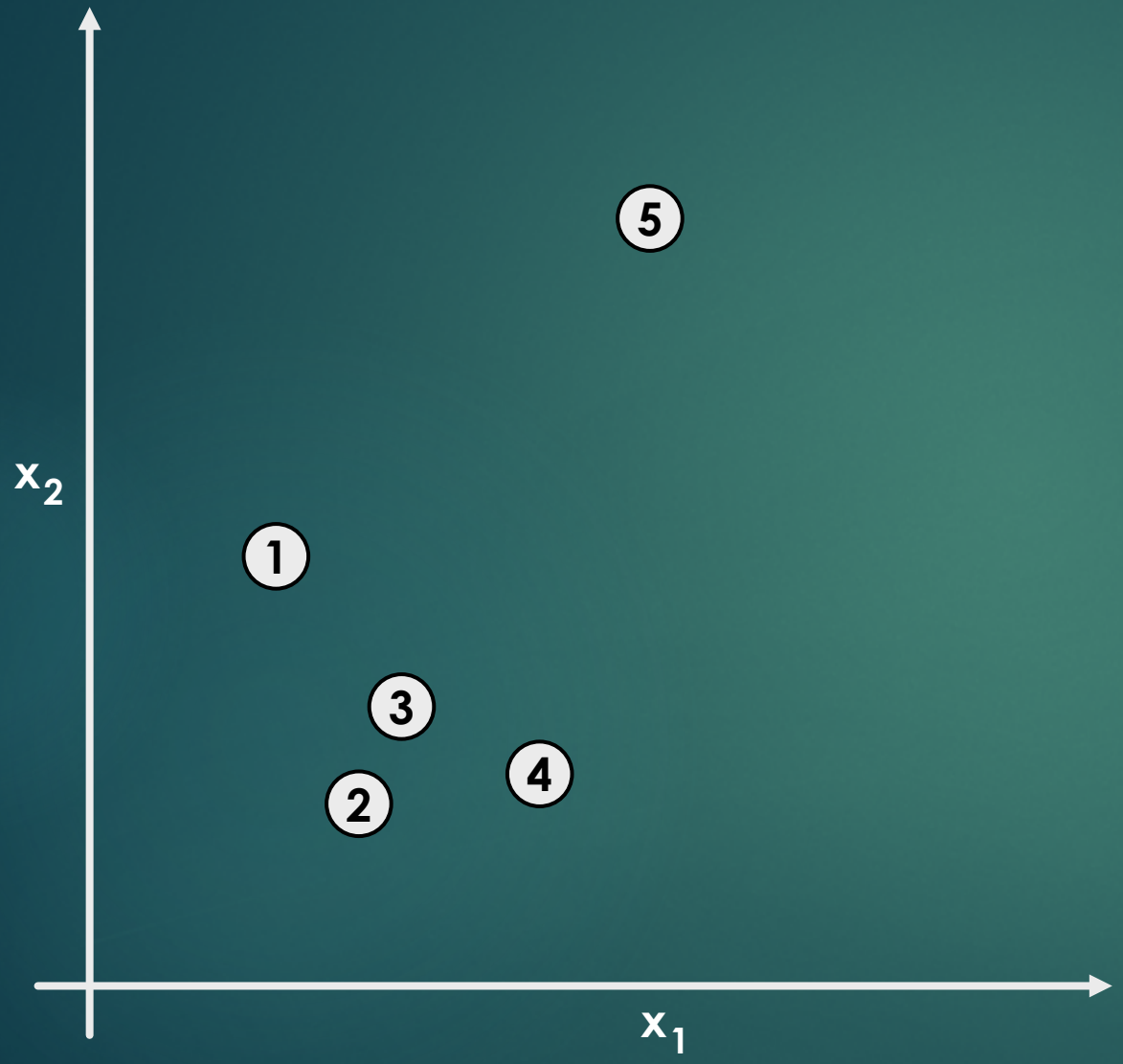

5
6
7
8
1
3
4
2
x_2
x_1


5
6
7
8
distance
1
3
4
2
x_2
x_1

Hierarchical clustering

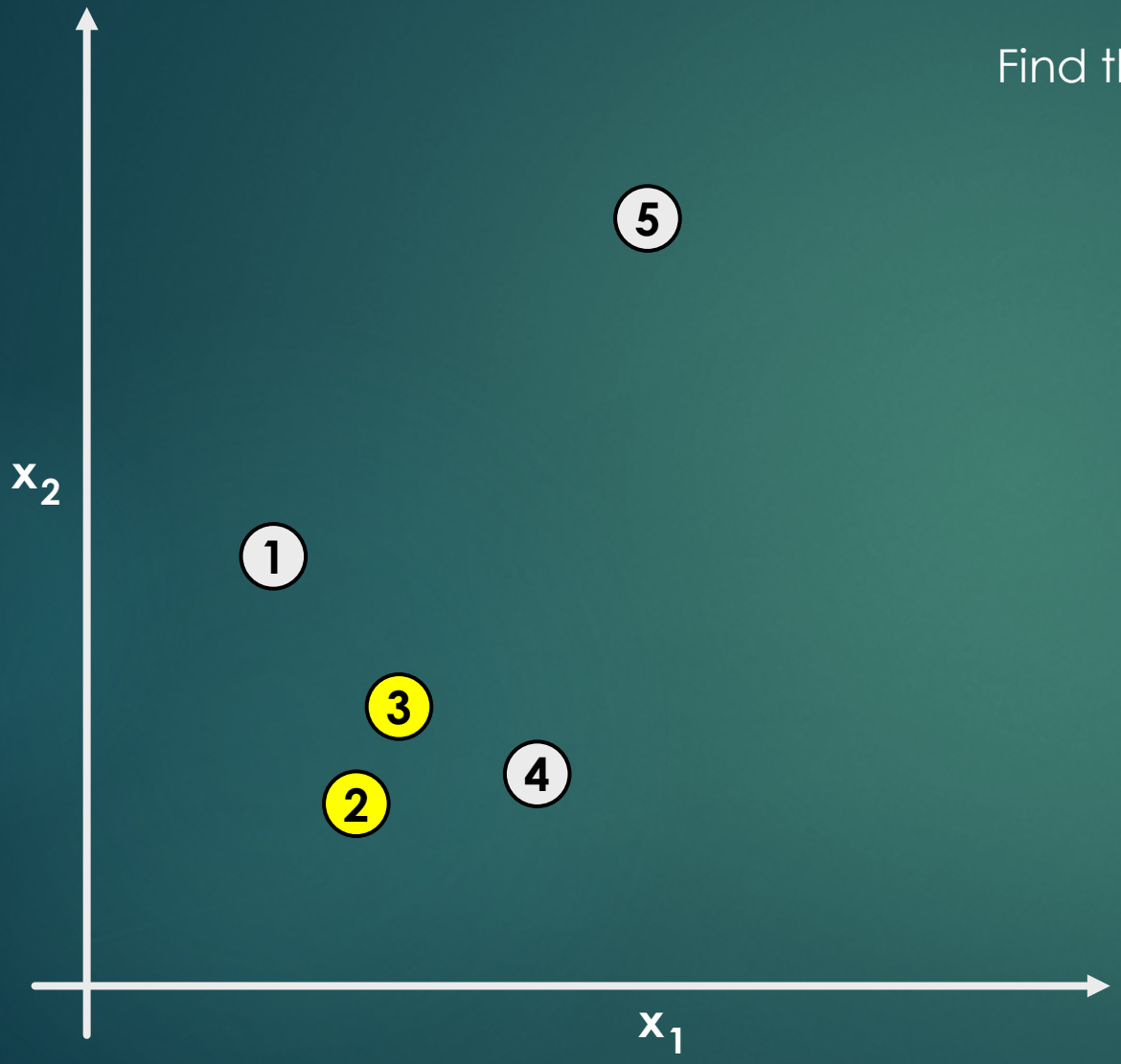
- How are we able to measure the distance of two clusters?
- We usually calculate the distance of the avarages of the clusters' elements
- We usually use Euclidean-distance: two observations are similar if the calculated distance is small
- Correlation based distance: two observations are similar if their features are highly correlated

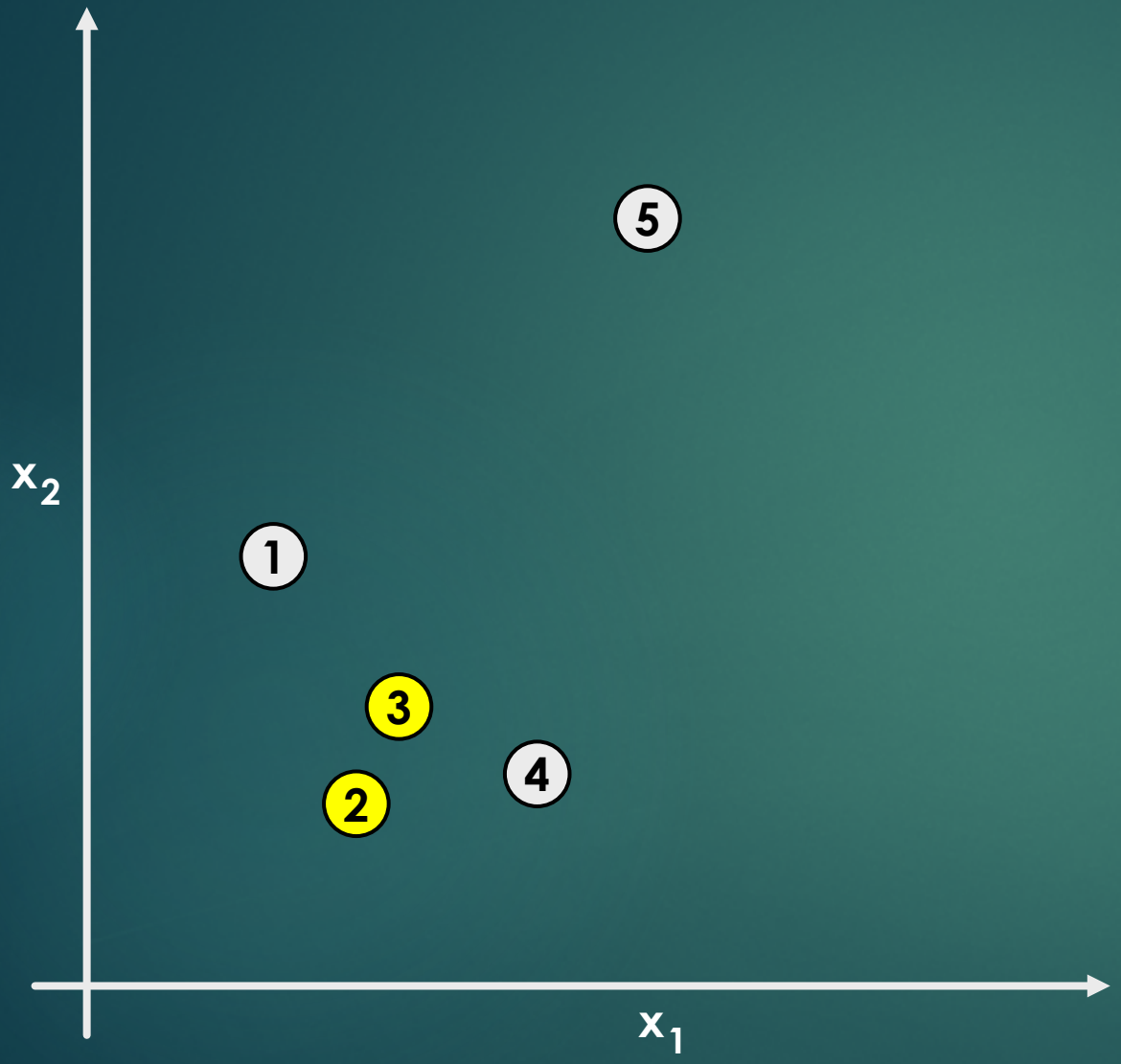

5
x_2
1
3
4
2
x_1


5
x_2
1
3
4
2
x_1

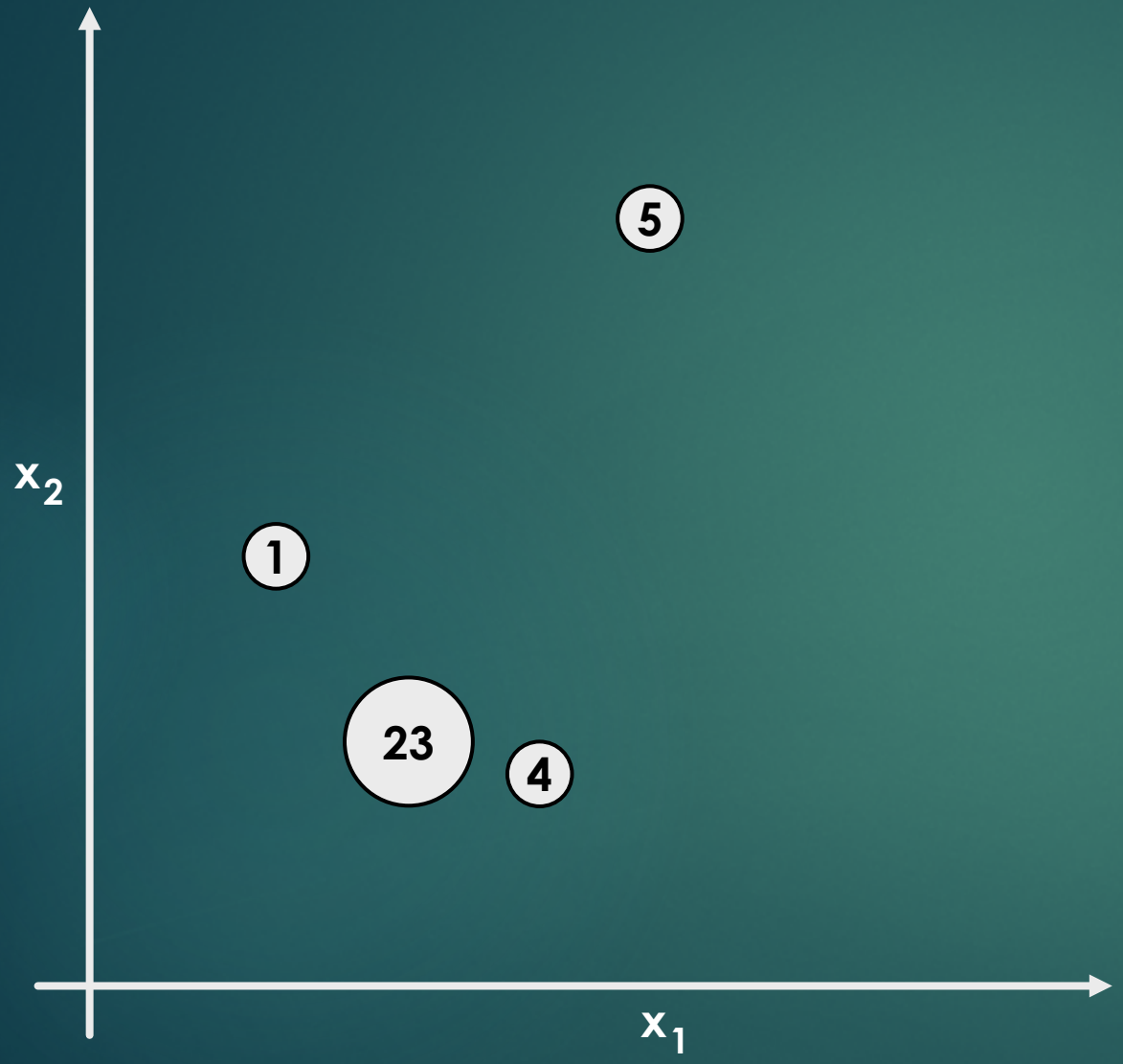

1
2
3
4
5

Find the closest pair of items in the dataset

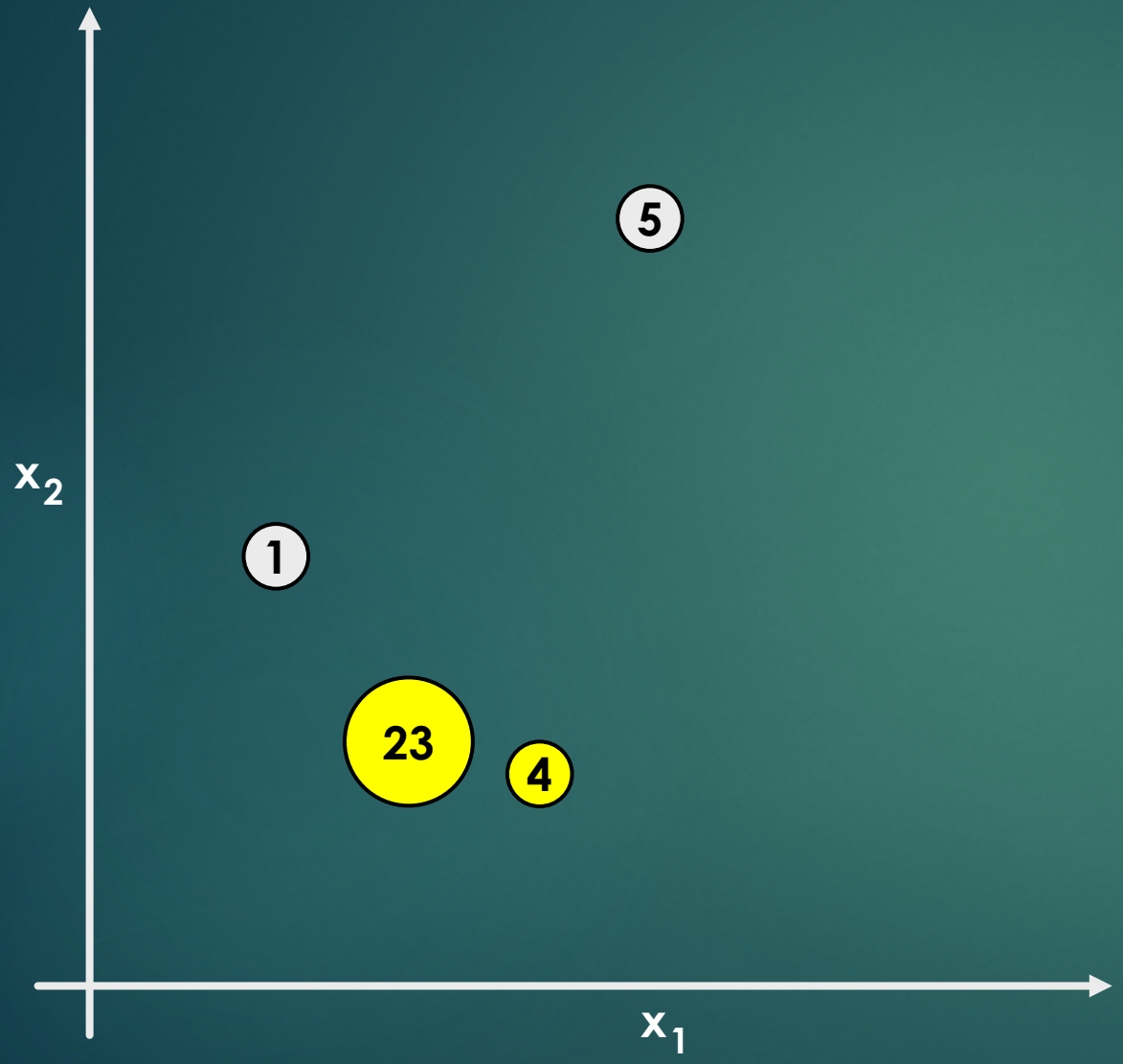



x2
x1
1
2
3
4
5

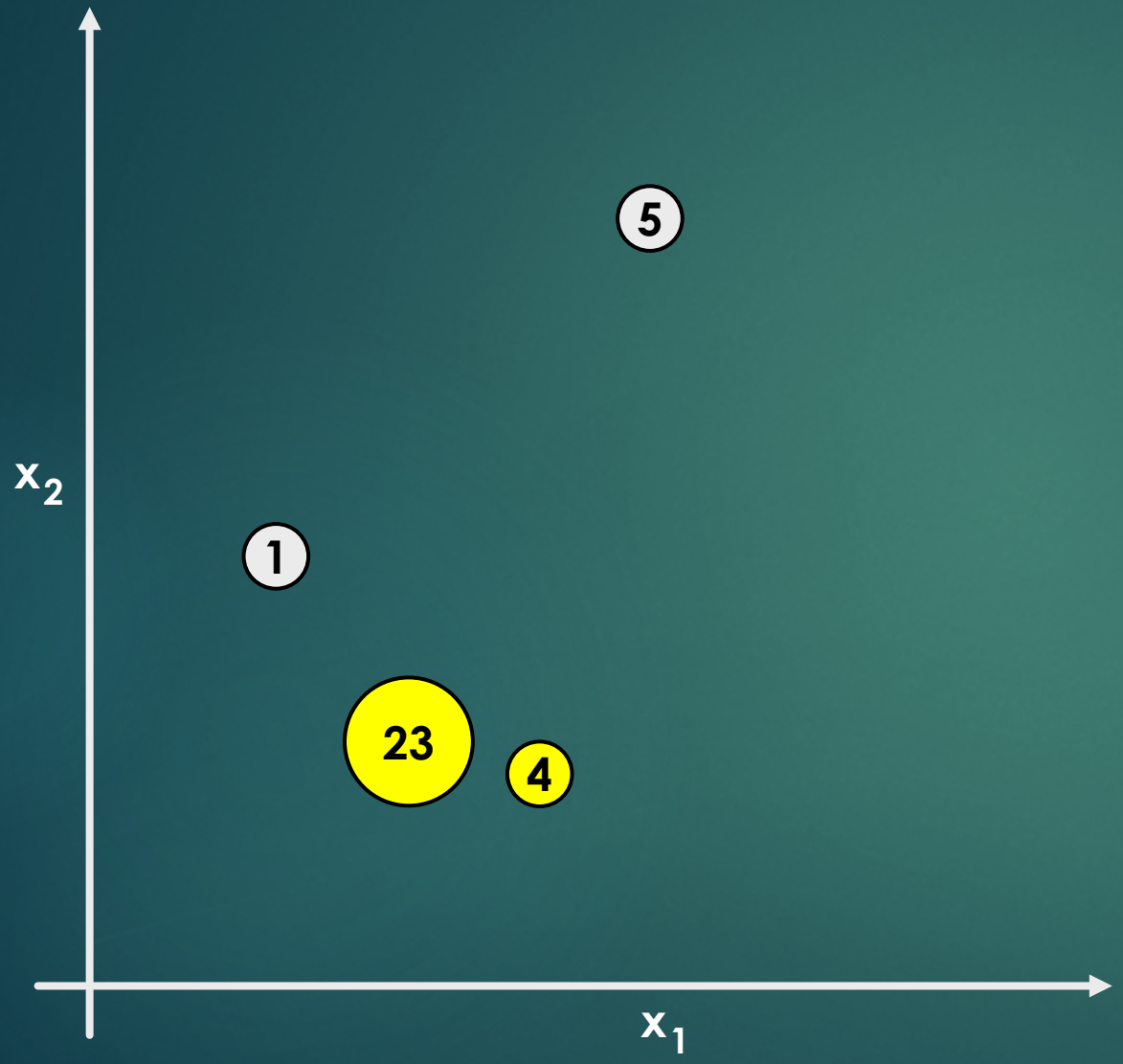

1
2
3
4
5

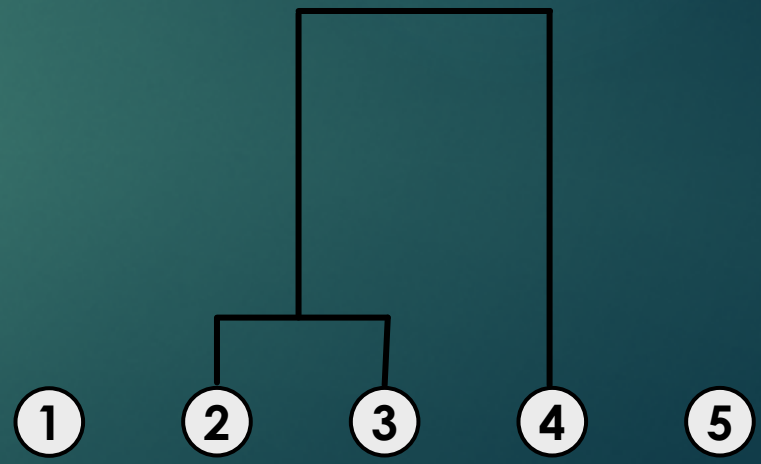

5
x_2
1
23
4
x_1

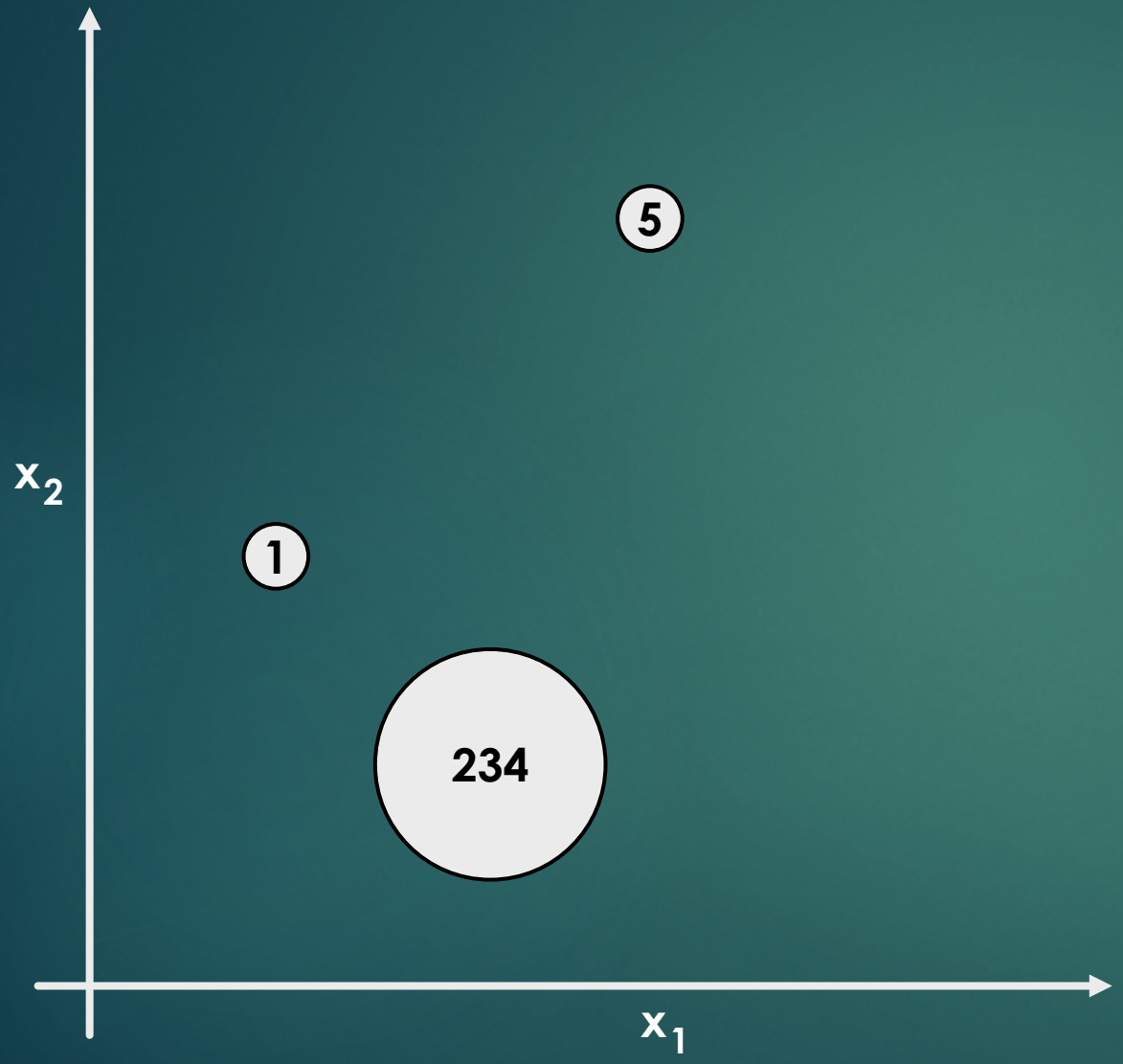

1
2
3
4
5

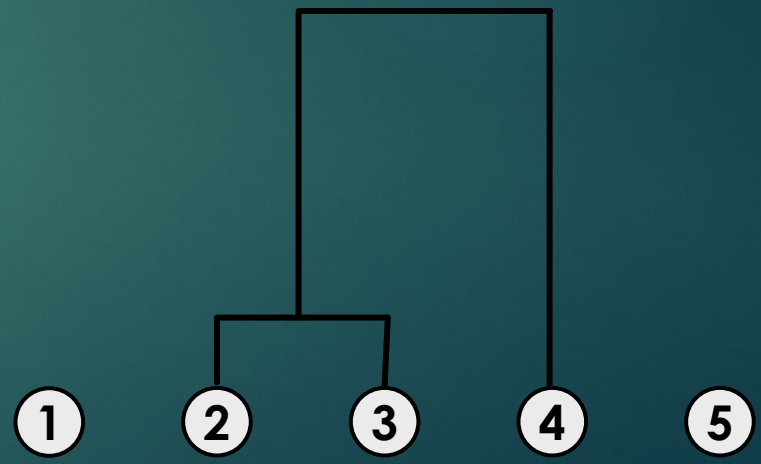

x2
x1
5
1
23
4

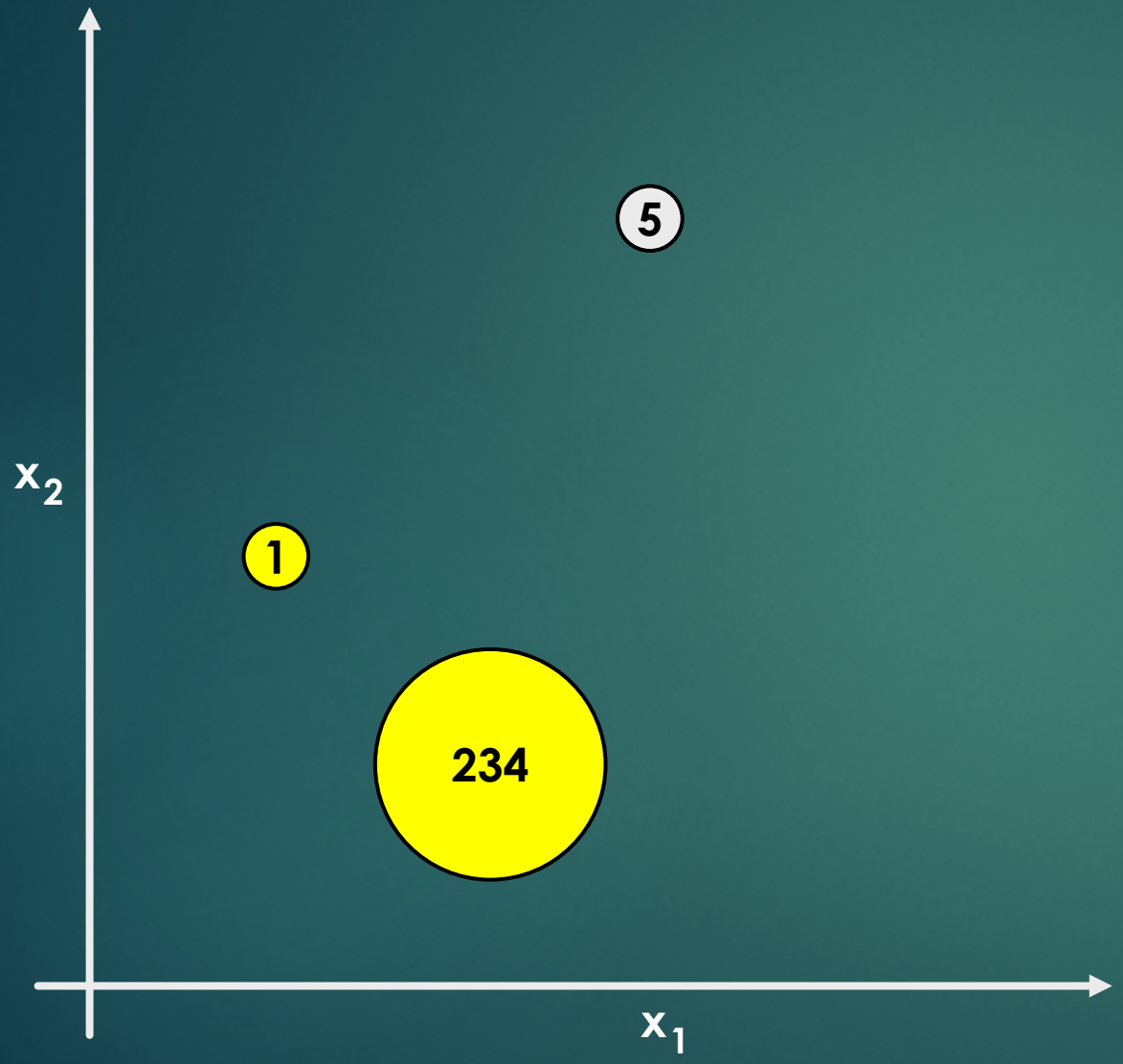

1
2
3
4
5

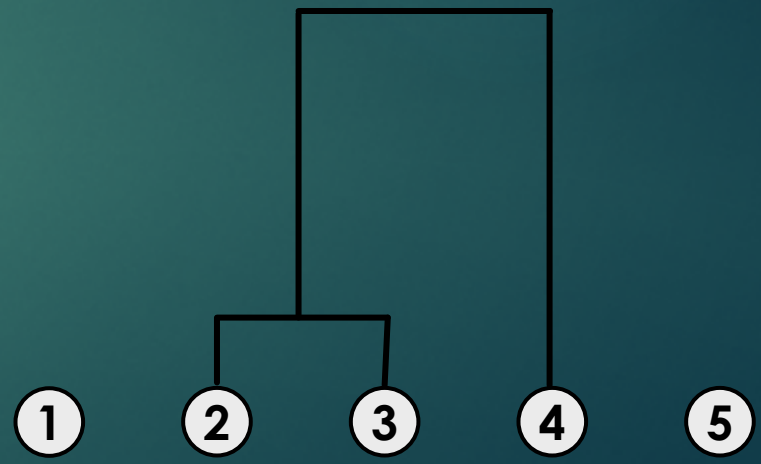

5
x_2
1
23
4
x_1

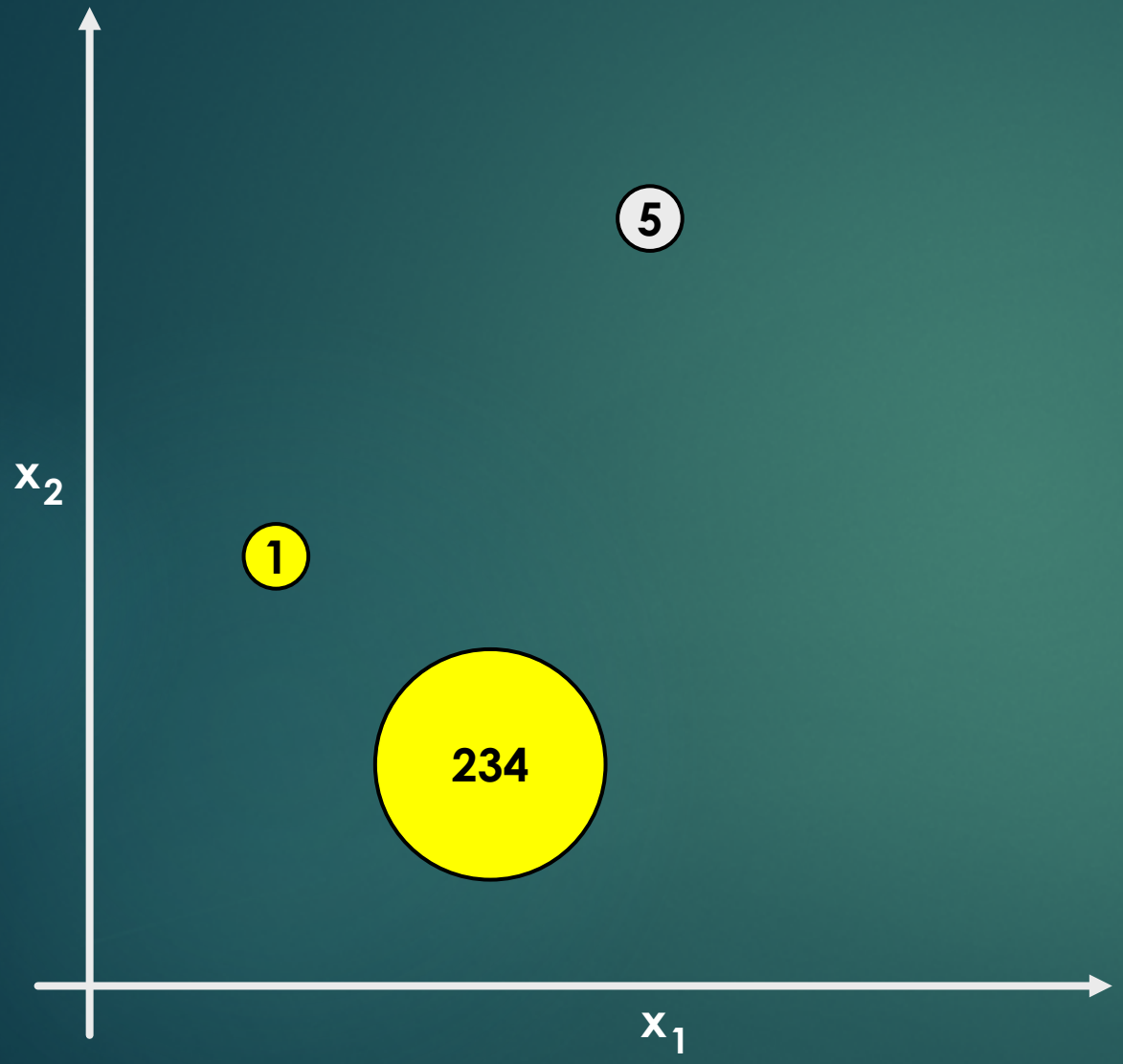

1
2
3
4
5

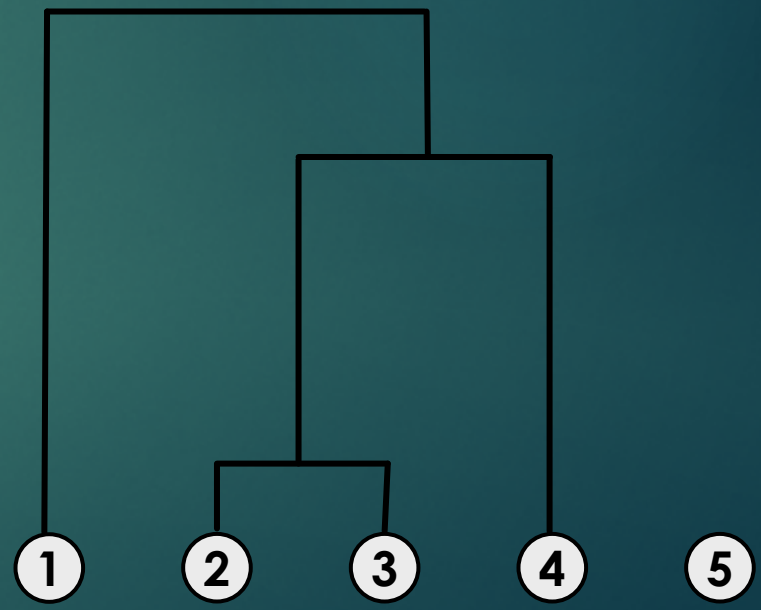

5
x_2
1
234
x_1

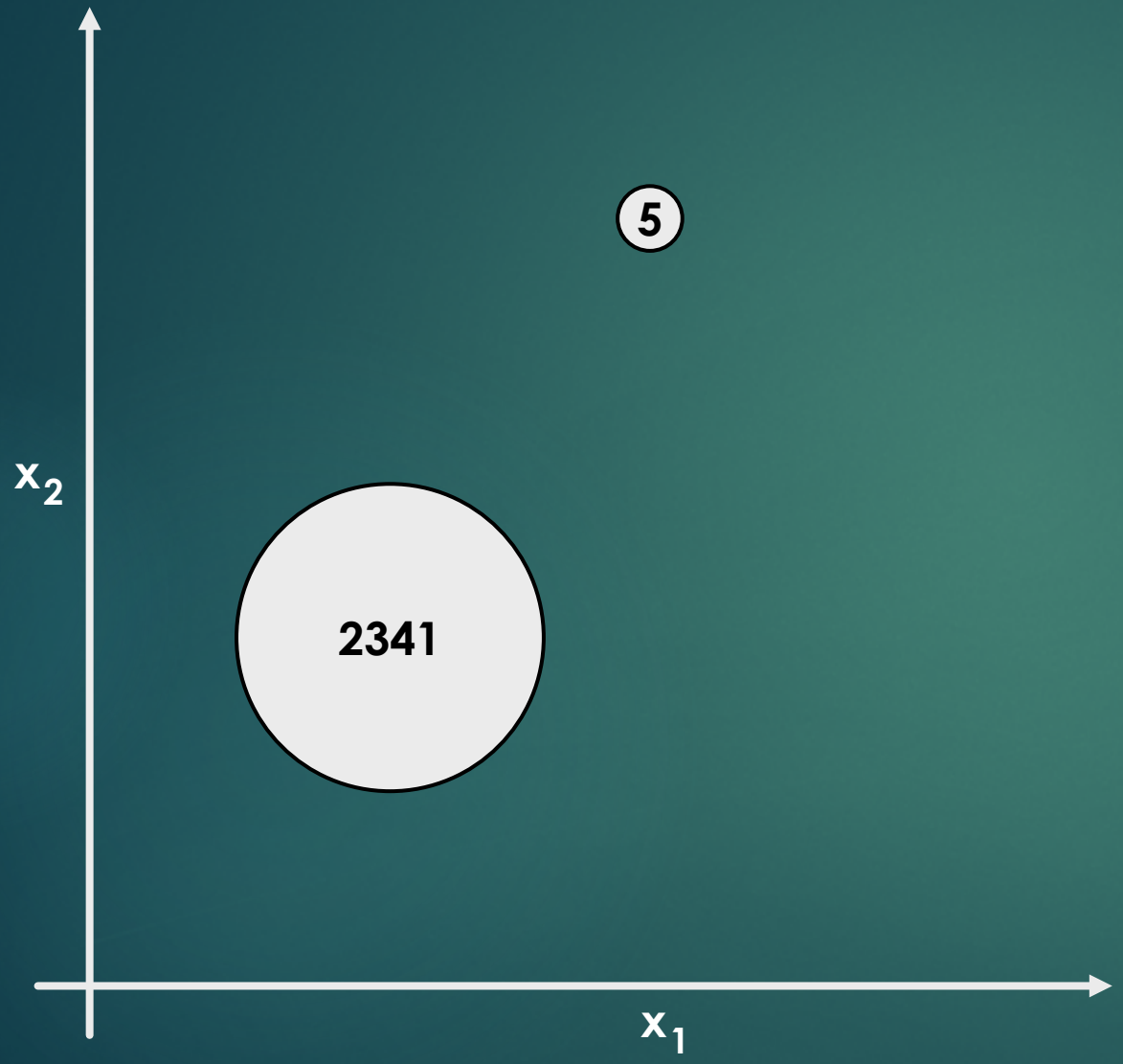

1
2
3
4
5

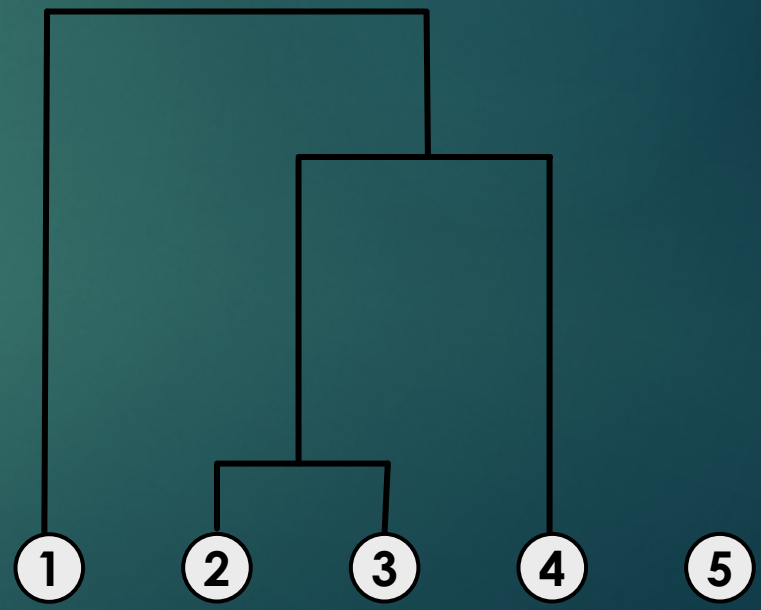

5
x_2
1
234
x_1

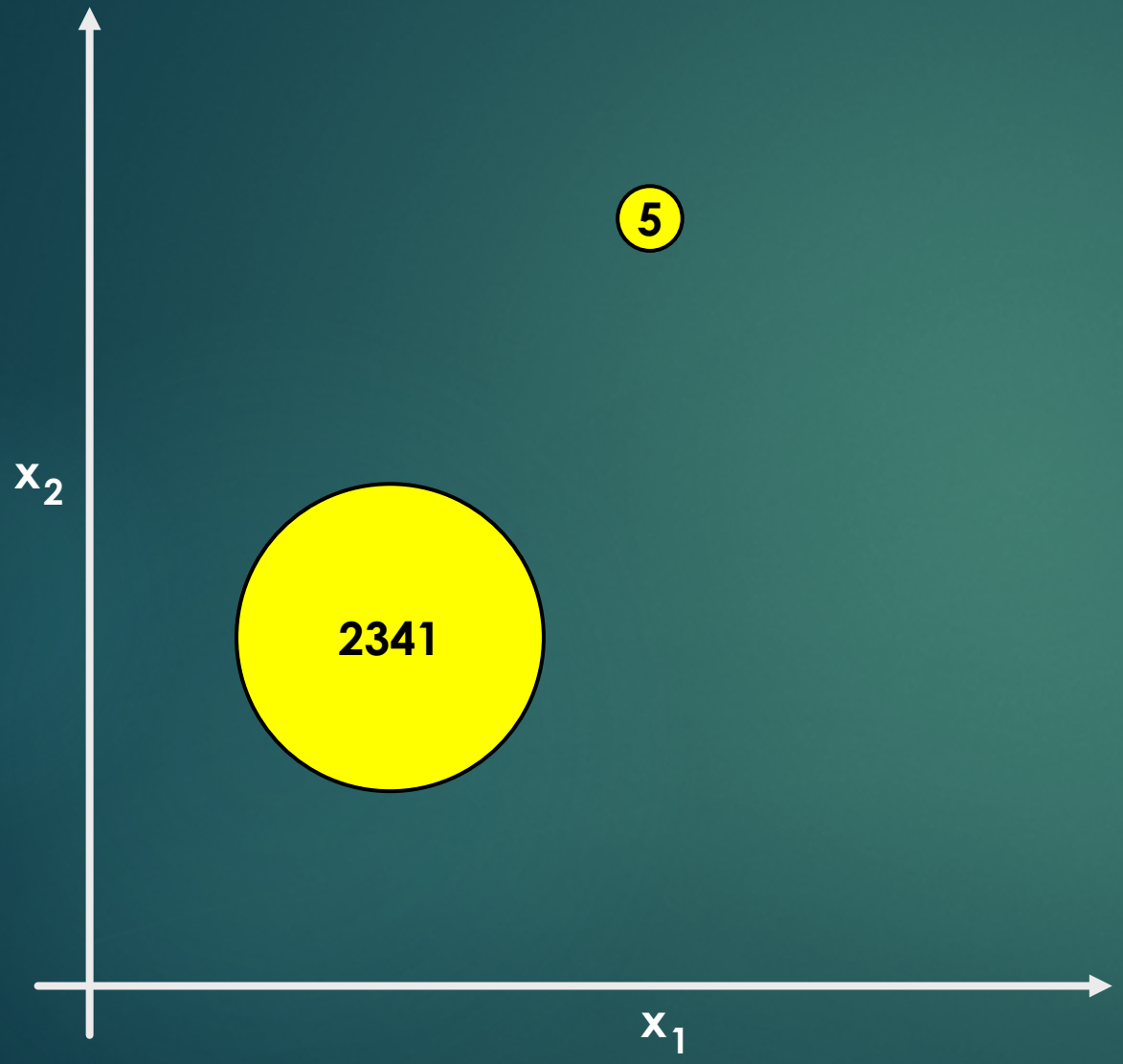

1
2
3
4
5

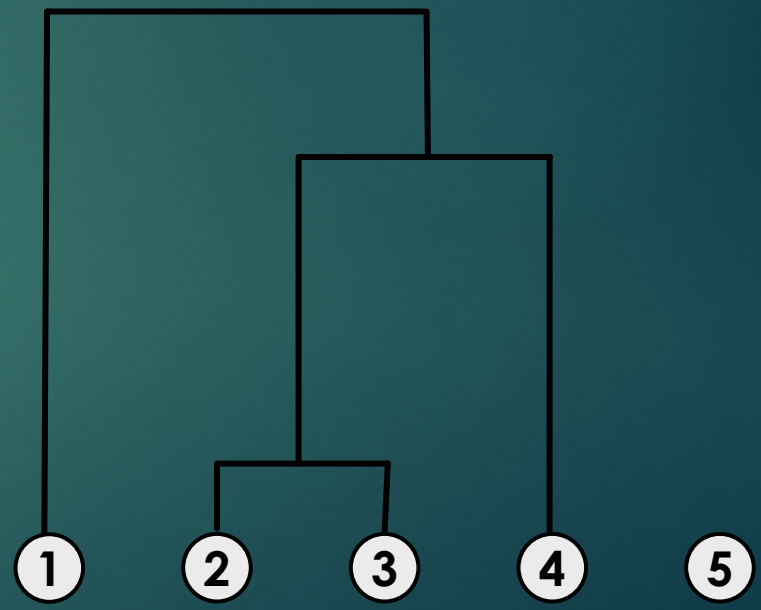

5
x_2
1
234
x_1

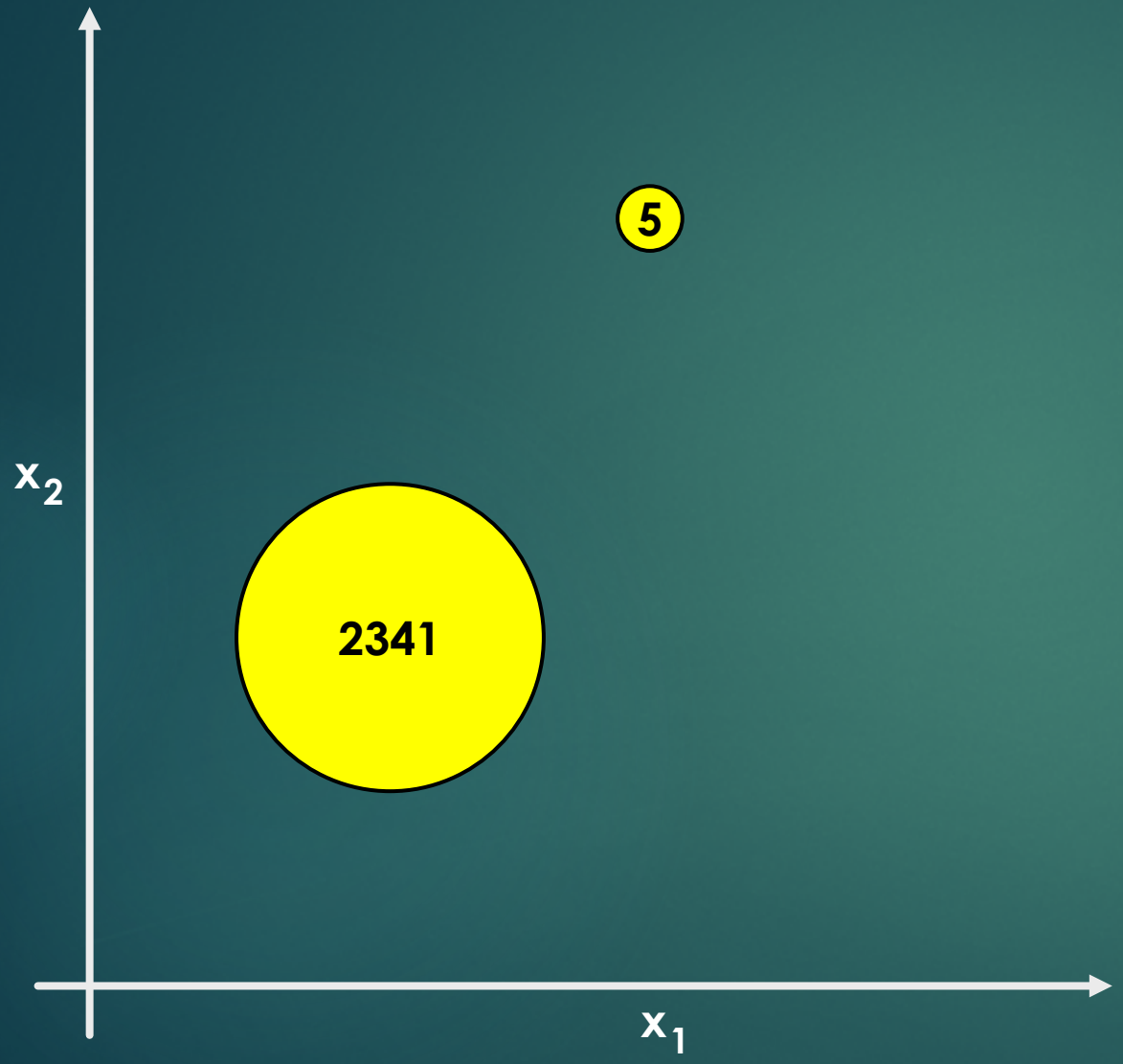

1
2
3
4
5

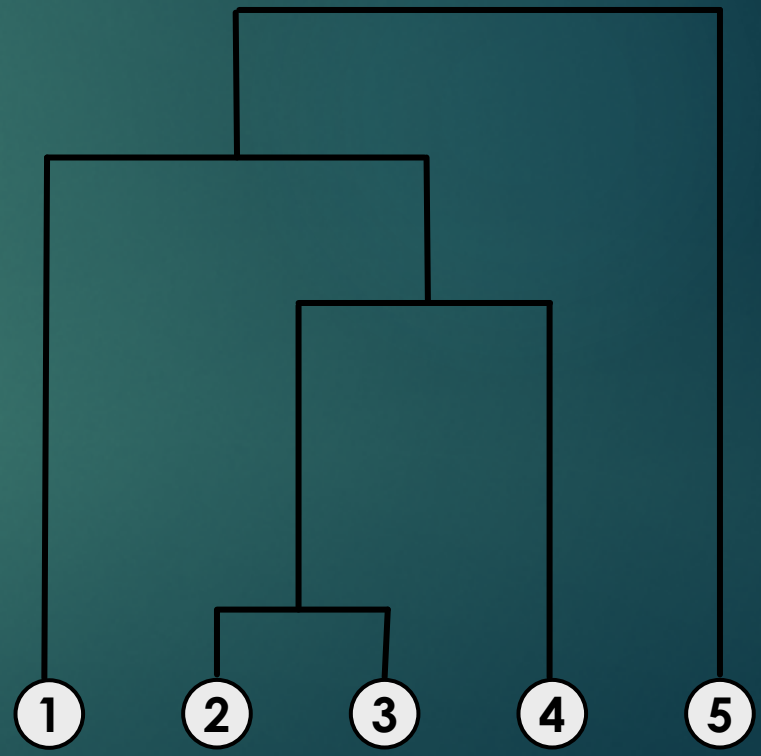

5
2341
x2
x1

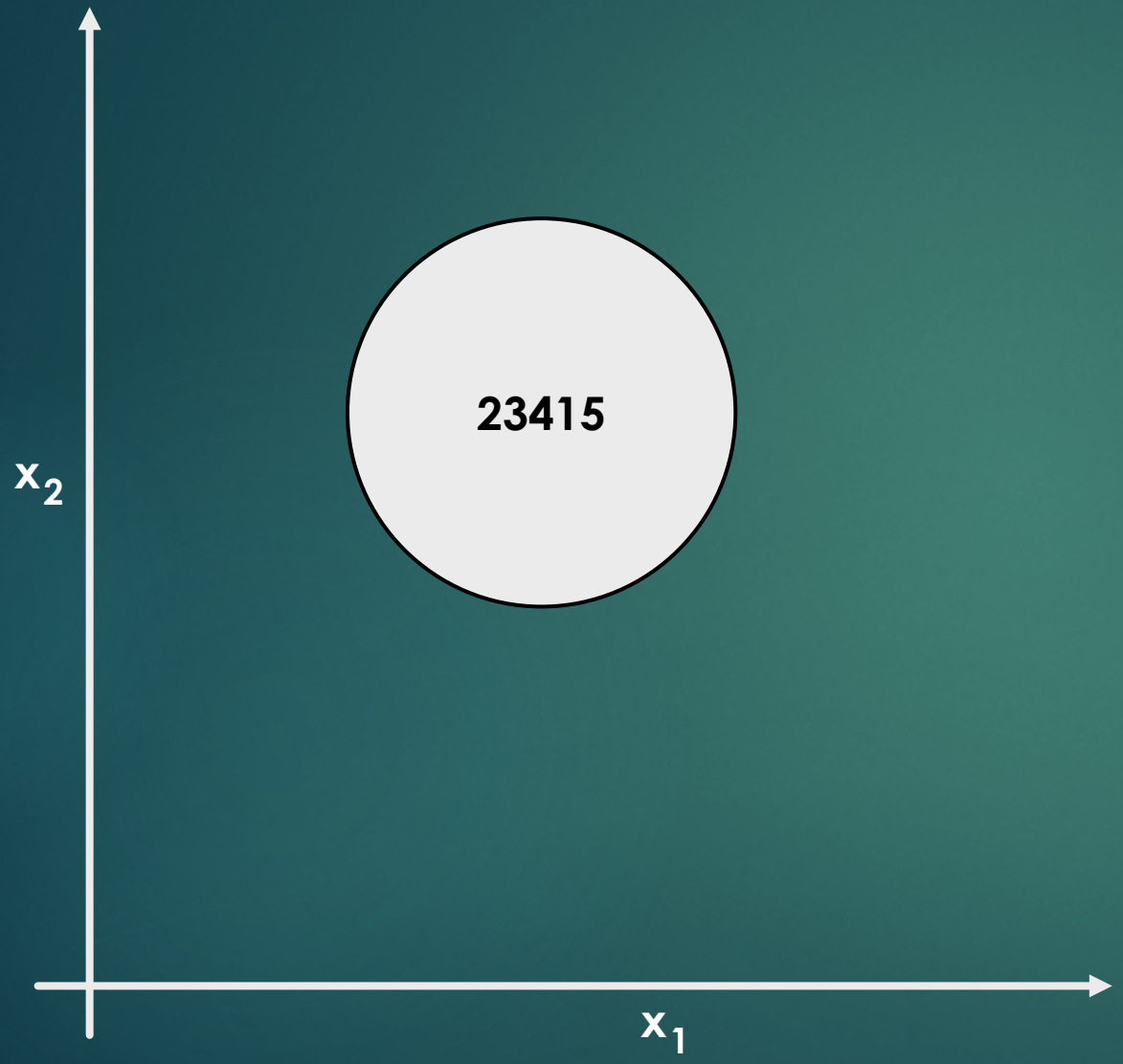

1
2
3
4
5

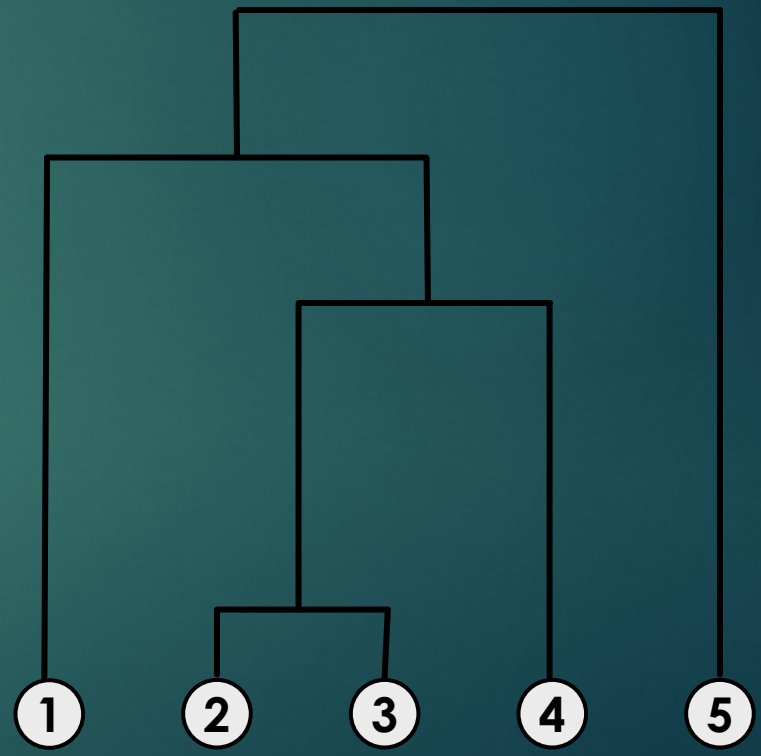

5
2341
x2
x1


1
2
3
4
5

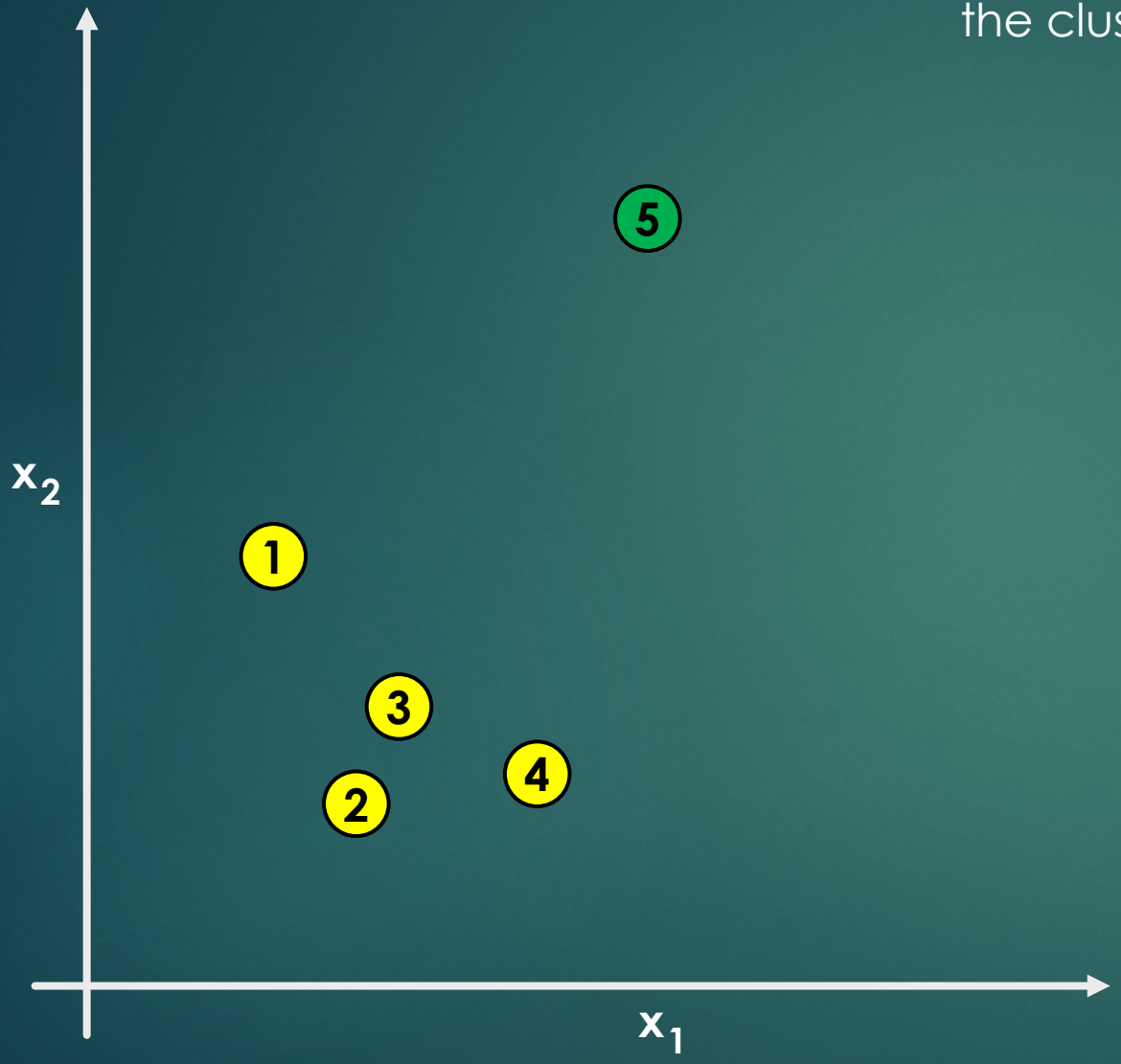

5
2341
x2
x1

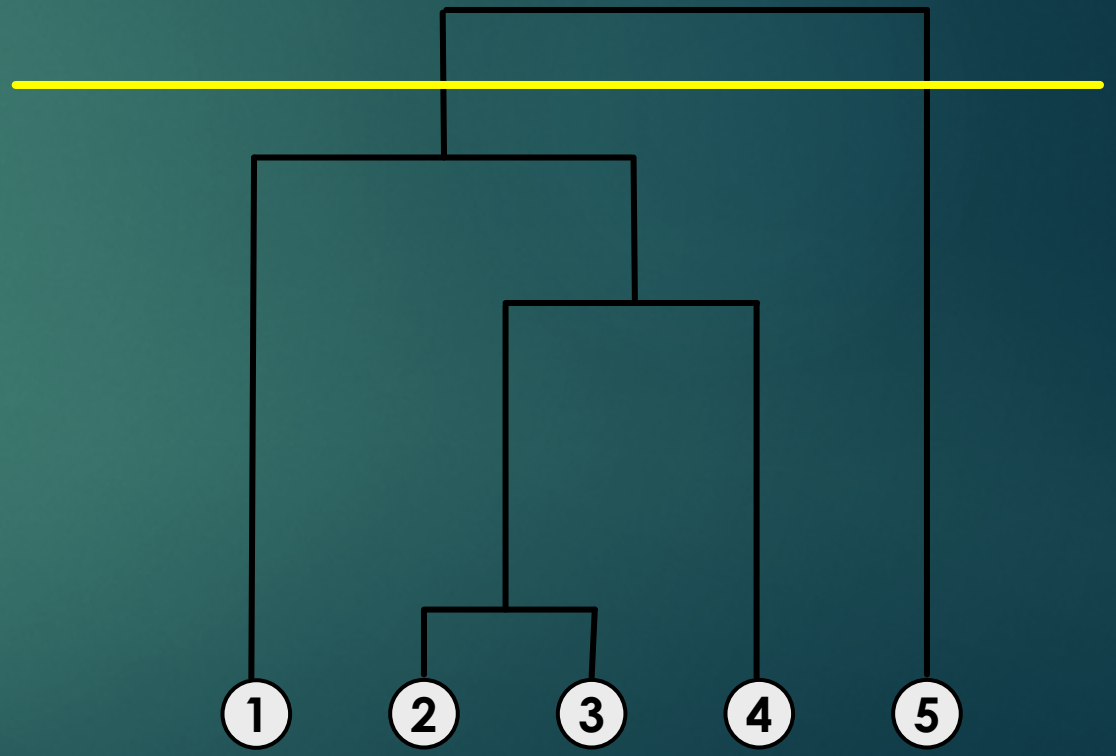

1
2
3
4
5

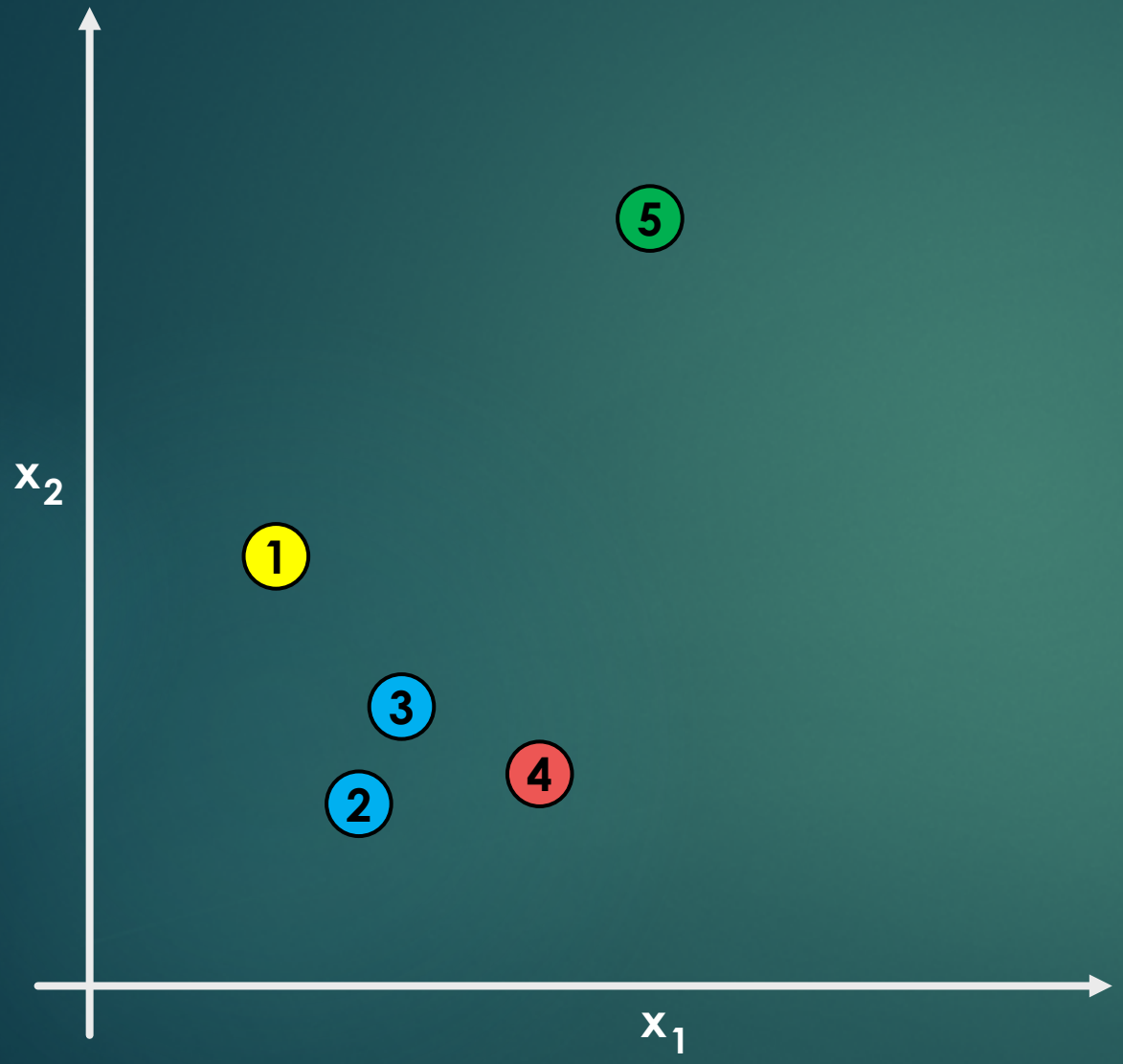

23415
x_2
x_1

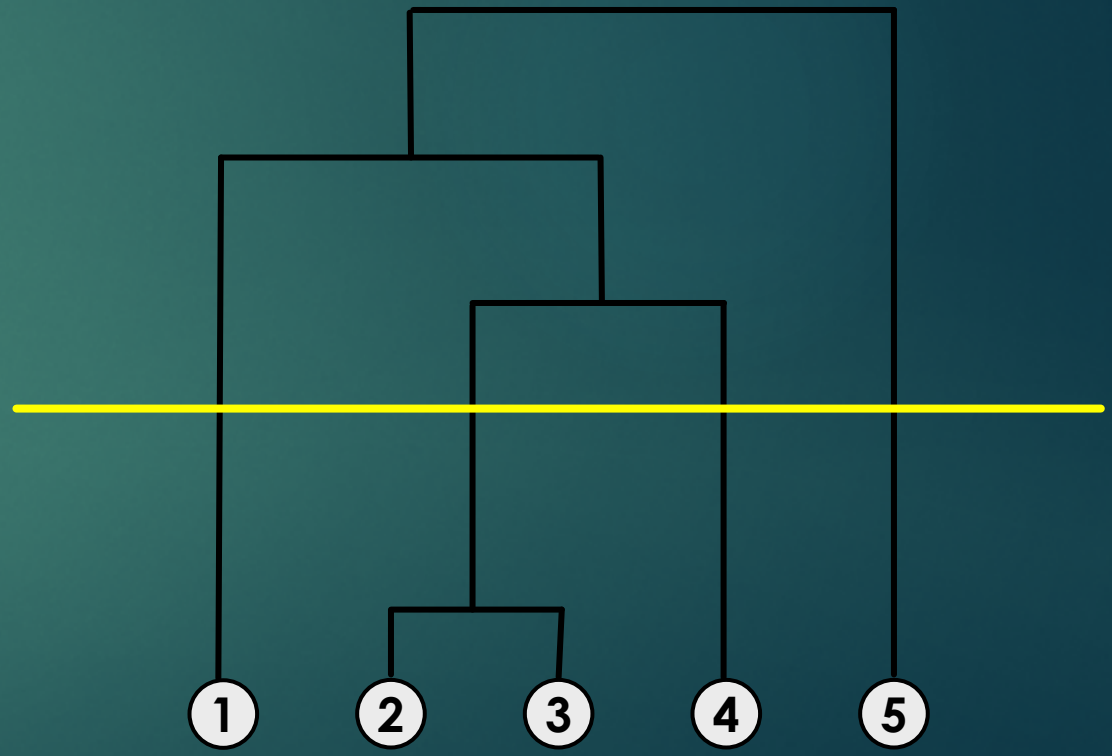

1
2
3
4
5

We cut the dendogram in order the get the clusters !!!






x2
x1
1
2
3
4
5


1
2
3
4
5

Hierarchical clustering

- Important: scaling of variables matters
- We should use some way of standardization
- Variables should be centered to have mean **0** or scaled to have standard deviation **1**