I. K-means++

K-means++ is an improved way to initialize the cluster centers for K-means to avoid bad starting points (since you could get a poor results if the starting k centers are positioned badly).

1. Pick the first center **randomly**.

2. For each remaining datapoint x, compute D(x): the **distance from x** to the **nearest chosen center**.

3. Pick the second center…. how: assign each remaining point a probability of being picked where the probability for a datapoint is $D(x)^2$ (the distance from the point to the closest centroid –right now there's only 1 option– but squared so that bigger distances are favored, raising the probability). Squaring distances gives higher weight to points that are far from existing centers, encouraging centers to spread more.

4. Repeat until you have k centers, then run normal K-means.

The Main Idea: Spread out the initial centers intelligently instead of purely random, which leads to faster convergence and better (lower-cost) clusterings.

**There's no reason ever to use k-means over k-means++

II. How to Choose the Right k

1. Elbow method: try multiple values for k with the goal to minimize the cost function
2. Using domain knowledge
3. Metric for evaluating a clustering output

III. Evaluation

The K-means cost function (also called inertia or SSE — sum of squared errors) measures how tight each cluster is…. but it says nothing about how far apart the clusters are from each other (inter-cluster distance).

This is important because we don't want clusters that are too close together/overlapping. But the k-means cost function can't tell you if the clusters are well separated.

Method to evaluate between-cluster distance:

a = average within-cluster distance
b = average between-cluster distance

If **b − a = 0**→ points are just as close to other clusters as to their own → bad separation (clusters overlap).

If **b − a** is large → clusters are well separated (good).

BUT …… the raw difference b − a depends on the scale of your data — so it's hard to compare across datasets or features.

SO …… **normalize it with the silhouette formula** ……. $s = \frac{b - a}{max(a, b)}$

We can formalize this idea by computing these values for **each data point**.

For every point i:

- a_i = mean distance to other points in its own cluster
- b_i = mean distance to points in the nearest other cluster

Then we can calculate the **Silhouette Score** $s_i = \frac{(b_i - a_i)}{max(a_i, b_i)}$ for each point which measures how well that point fits within its assigned cluster.

- s ≈ 1: point is well clustered (a ≪ b → far from other clusters)
- s ≈ 0: point is on or near a cluster boundary
- s ≈ −1: point might be assigned to the wrong cluster

FINALLY …… once we get all s scores, we have two options to evaluate the entire clustering:

1. Plot all s_i values in a silhouette plot … shows how well each cluster is separated
2. Or simply take the mean silhouette score across all data points