" K-means and K-means ++ "

Supervised → Label → Target / Dependent variable

Unsupervised → No label

↳ " CLUSTERING "

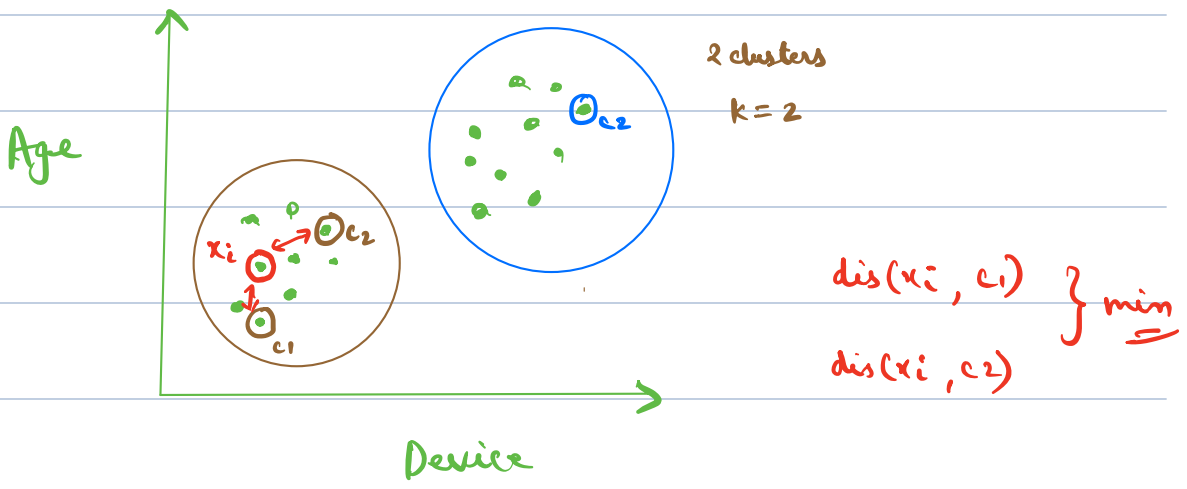
" JIOHOTSTAR "

Person	Age	Number of views	Duration (mins)	Device
A	20	4	180	Mobile
→ B	48	10	360	TV
C	30	1	20	iPad
D	25	15	10	Mobile

2 categories / clusters

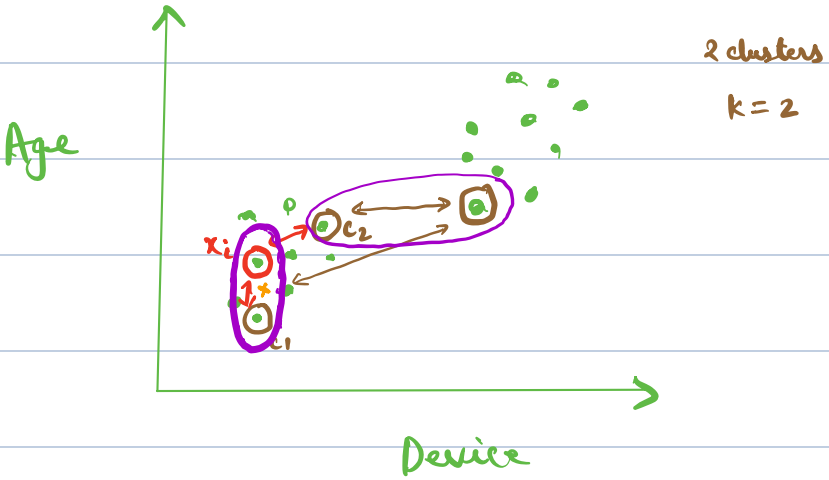
① Young viewers and mobile users

② Senior adults and TV



Steps of K-means:

1. Initialize 'k' centroids.
2. Calculate distance of each point from all centroids.
3. Assign that data point to nearest centroid.
4. Re-calculate the new centroid.
5. Repeat "Steps 2 to 4"
6. Till the point of convergence [same centroids are getting calculated]



- EVALUATION METRICS:
  1. Silhoutte Score
  2. Elbow Curve
     - ↳ WCSS

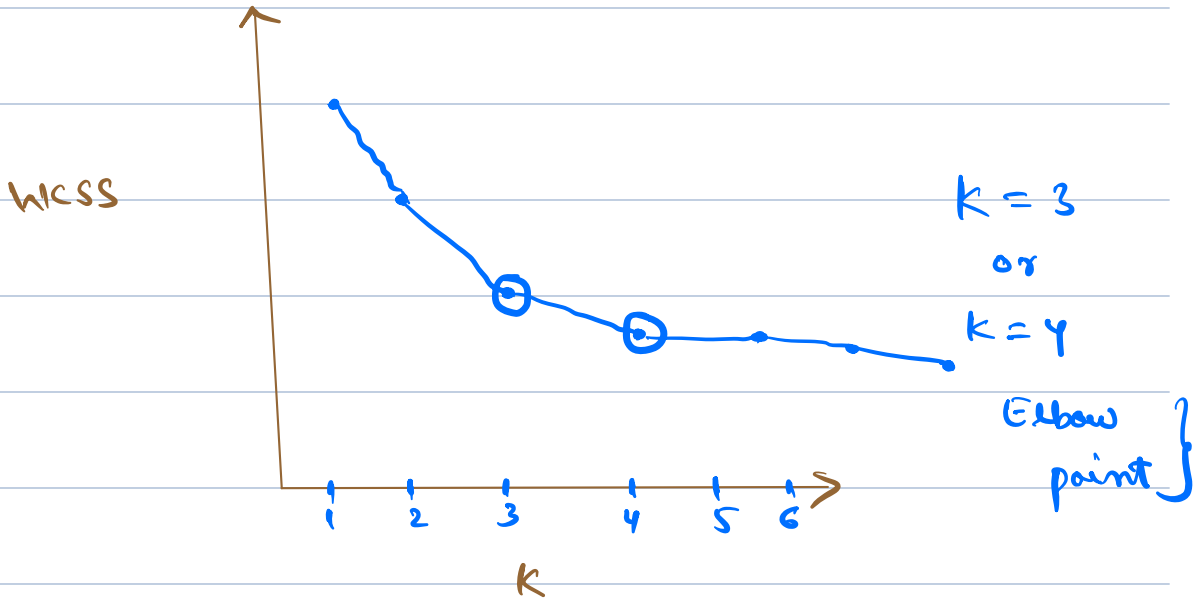
"Within Clusters Sum of Squares"

3 clusters

↳ Foreach cluster → intra-cluster distances (SUM)

↳ SUM of all Summed Up Distances of all clusters.

Python → inertia_



k = [1 to 10]

WCSS = [Model1, Model2 ..... Model10]

To decide the value of 'k':

1. Silhouette Score → [-1, +1] → "Best Model" ↳ SS as close to +1
2. Elbow Curve
3. Domain Knowledge

Limitations of Kmeans:

1. Not robust to outliers
2. Scaling of data.

③ Because of random centroids initialization, may not get correct clusters.

④ Mostly gives spherical clusters

"K-means ++"

↓

Another case of "Kmeans" where:

- instead of initializing random data points as centroids, we take farthest points as centroids.

Categories:

① Low spenders, medium, high

② Frequent visitors v/s non-frequent

③ High discount users v/s low

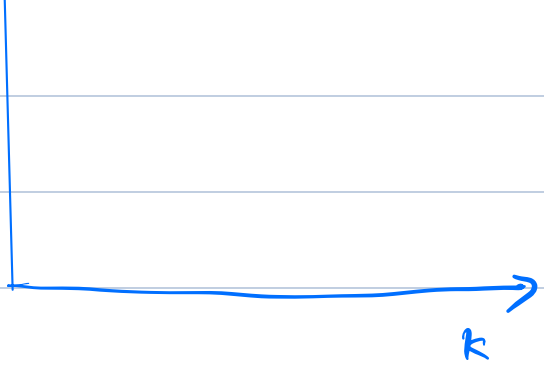
④ New users v/s Old users

$K = [1, 2, 3 \ldots \ldots 9] \rightarrow SS = [SS_1, SS_2 \ldots SS_9]$

→ K-models = [Model_1, Model 2 . . . . . Model 9]

→ inertia = [WCSS, , WCSS2 . . . . . WCSS 9]

WCSS ↑



① Visualization → $k = 3$

② Elbow Curve → $k = 3$ or 4

③ Silhouette Score → $k = 2$ or $k = 3$ or $k = 4$

+ Domain Knowledge

" $k = 3$ "