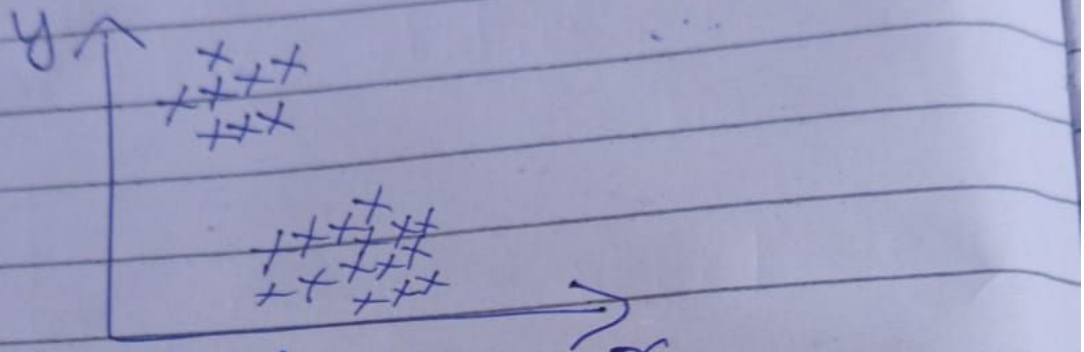
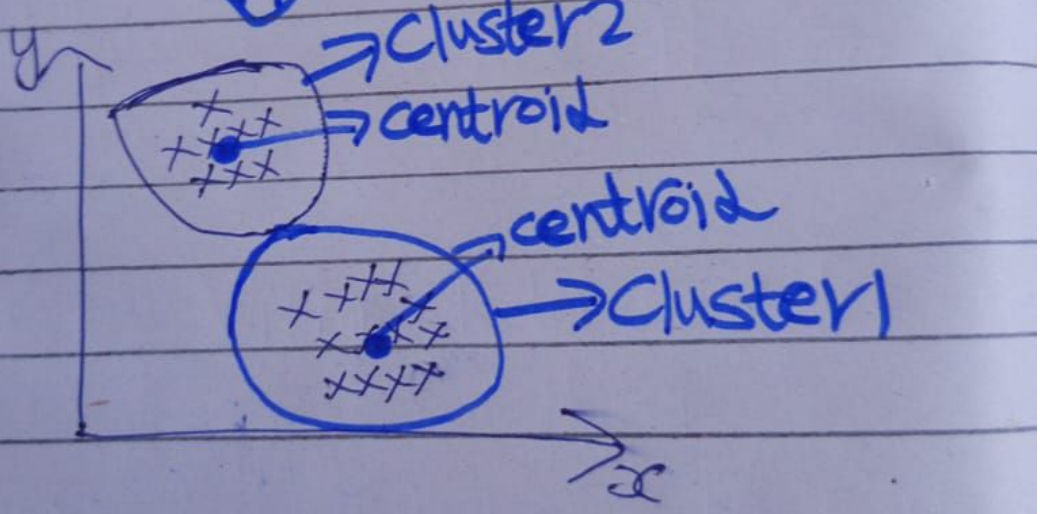
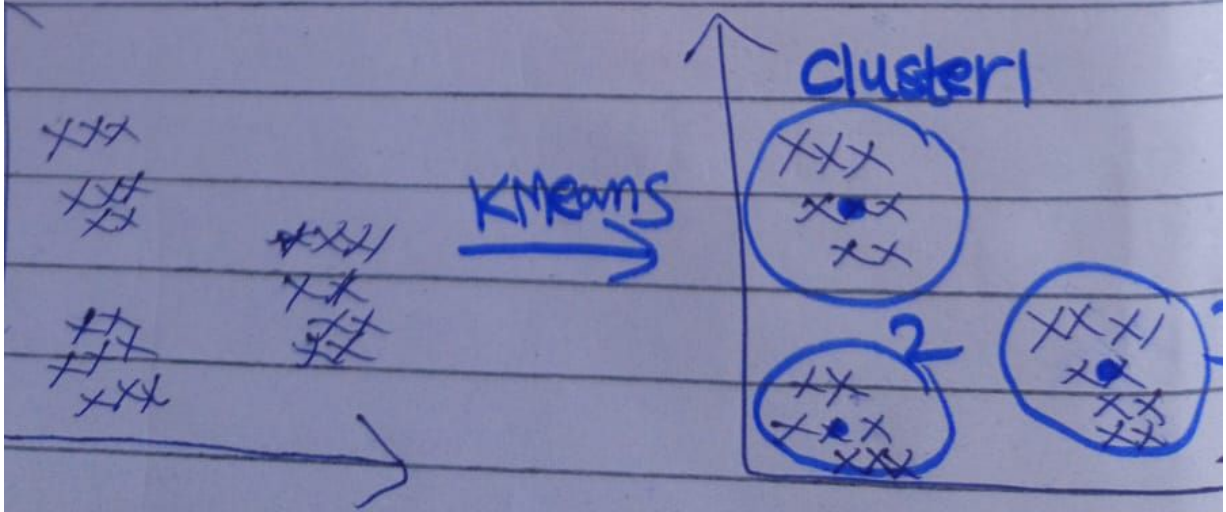
⇒ K Means Clustering Algorithm:

- Geometric Intuition:

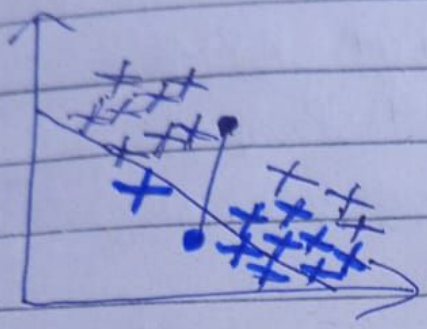


After applying KMeans





Let's suppose we have datapoints



Step 01: Initialize some K values ↓ centroids

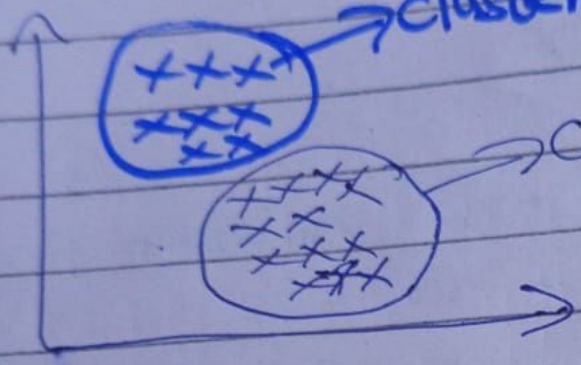
Let K=2 and we just select 2 random centroids

Step 02: Now we will find distance of all points from centroid1 and centroid2 Points that are nearest to centroid 1 mark them blue in that group

Step 03: Move the centroids to the average of the points in their group

And then again repeat step 2 and 3

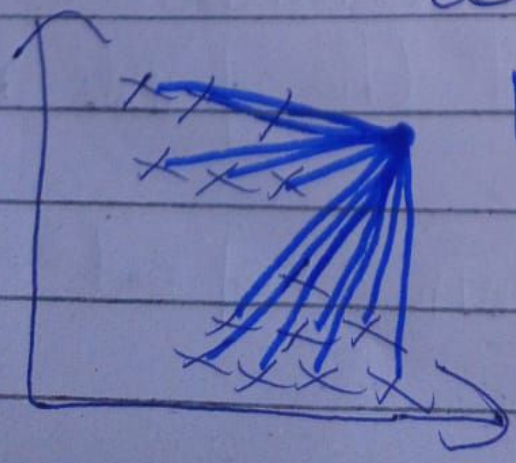
We will do these steps repeatedly until we not get 2 seperate groups or clusters and then we just mark clusters which is goal.



⇒ How to find the K-value:

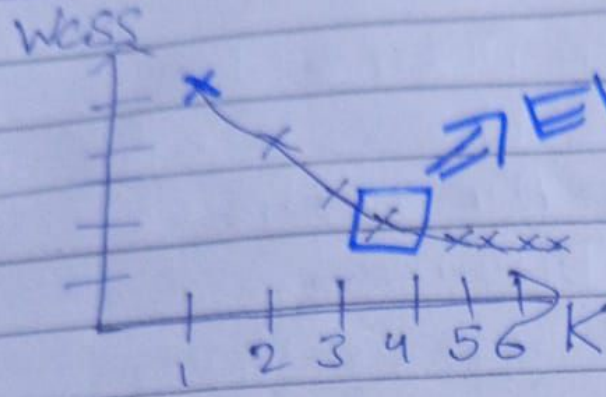
WCSS ⇒ Within cluster sum of squares

$$WCSS \Rightarrow \sum_{i=1}^{n} (\text{distance b/w points to nearest centroid})^2$$



K=1 (only 1 centroi

WCSS will be high for K=1 as distance is more



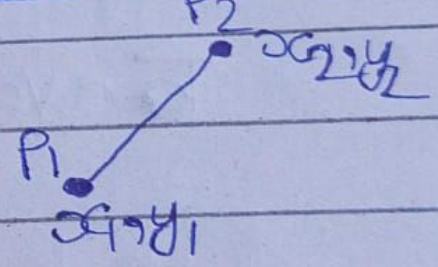
Elbow Method (We select K-value where wcss abruptly decrease and then stablize)

For K=2 it wcss decreases

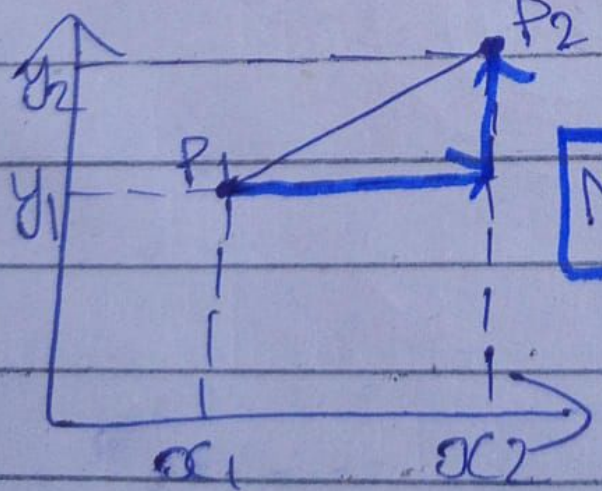
To find distance between points and centroid we use

① Euclidean Distance:

$$\text{Euclidean Distance} = \sqrt{(x_1-x_2)^2+(y_2-y_1)^2}$$



② Manhaton Distance



$$\text{Manhaton Dist} = |x_2-x_1|+|y_2-y_1|$$

- We use Manhaton Distance for road things as there are building
- In case of Air traffic we use Euclidean Distance

⇒ **Random Intialization Trap (KMeans++):**

As for now we select centroids randomly but selection goes wrong i.e centroids are very near to each other. This is called **Random Initialization Trap**

- To avoid this we use **KMeans++ Initialization** it ensures that centroids are at maximum distance from each other.