Kmeans ++, Heirarchical

[Clustering]

- -> Problem recap
- -> Kmeans algorithm
  - -> WCSS
- -> Code from scratch
- -> Kmeans ++
- -> Heirarchical clustering

Recap

↳ Amazon customer segmentation

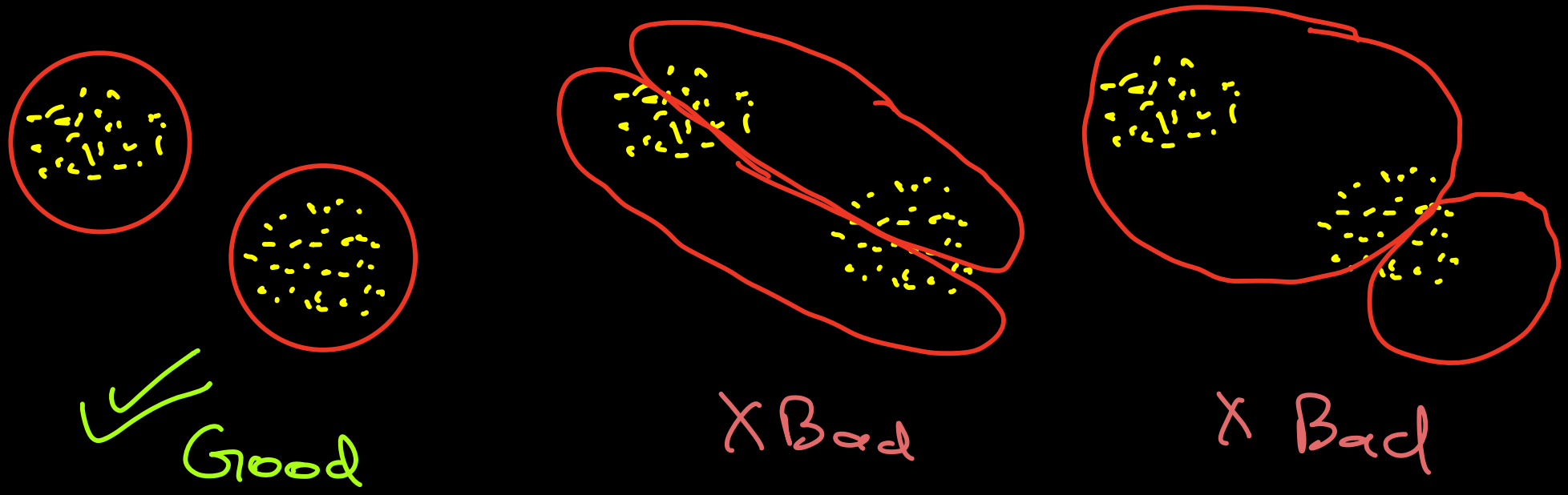
	ID	n_clicks	n_visits	amount_spent	amount_discount	days_since_registration	profile_information
0	1476	130	65	213.905831	31.600751	233	235
1	1535	543	46	639.223004	5.689175	228	170
2	1807	520	102	1157.402763	844.321606	247	409
3	1727	702	83	1195.903634	850.041757	148	200
4	1324	221	84	180.754616	64.283300	243	259

→ Clustering :

↳ Points belong to different populations

→ A good clustering algorithm will be able to detect these.

Good Clustering:



How to express this mathematically??

→ within a cluster variance / distance should be as low as possible

→ Between cluster distance (var of centroids) should be as high as possible

Hence, we can come up with metrics that can measure a clustering o/p.

$$M \rightarrow \frac{\text{inter cluster distance}}{\text{intra cluster distance}} \quad \begin{matrix}\text{(between)} \\ \text{(within)}\end{matrix}$$

Possible metrics:

F-statistic / Dunn index / WCSS / Silhouette, $\bar{c}_{ty}$

F-statistic ↓ kind of an avg metric

Dunn index ↓ extreme metric

WCSS / Silhouette, $\bar{c}_{ty}$ — will study

$$\frac{\sum(\bar{x} - \bar{\bar{x}})^2 \cdot n / df_1}{\sum\sum(x - \bar{x})^2 / df_2} \qquad \frac{\min(\text{inter})}{\max(\text{intra})}$$

↓

✗ not used much

→ degrees of freedom

→ complex looking

→ tied to one distance metric

So we can use any good measure of clustering and proceed.

Optimisation

params: cluster centers, metric: F, dunn, WCSS, etc

$$\max_{C} \text{metric}(\text{assignment}(X, C))$$

maximise over centroids, some metric measured on some assignment of points to closest centroid.

Problem:

↳ "assignment of nearest cluster center" is not differentiable

→ in some cases the loss metric may not be differentiable. eg: dunn / silhoette

Hence we cannot solve this using gradient descent.

Lloyd's Algorithm (K Means)

→ animation

→ random initialise 'K' clusters
  ↳ manually decide

```
while True:
    y = closest(X)
    for center in y.unique():
        new_center = X.loc[y == c].mean()

    loss = metric(X, c)
    if prev_loss - loss < 1e-4:
        break
```

Complexity: $n^{\star} c^{\star} i^{\star} d$ (n: pts, c: K, i: iters, d: dim)

→ code

WCSS (intertia) → Most popular

Within cluster sum of squares.

$$WCSS = \sum_{i=1}^{K} \sum_{j=1}^{m_K} (x_{ij} - \bar{x}_i)^2$$

- x_{ij}: point
- K: all clusters
- m_K: all pts in i^{th} cluster
- $\bar{x}_i$: center of i^{th} cluster

idea: **Minimize** the within cluster variance/distance (intra)

→ Here we are simplifying by completely ignoring between (inter) cluster distance.

Variation :

$$wcss = \sum_{i=1}^{K} \sum_{j=1}^{mk} \text{distance}(x_{ij}, C_i)$$

- distance → Any distance func of your choice
- C_i → center of i^{th} cluster

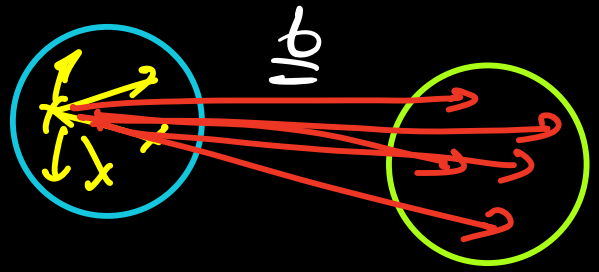
Silloutte Score

$$S(x_i) = \frac{b - a}{\max(b, a)}$$

a = avg distance of x_i with points in its own cluster

b = avg distance of all point of nearest

cluster with x_i

a =  = b

$\frac{b-a}{\max(a,b)}$ ↙ calculated for a single point.

How close is the point to the points neighbouring cluster compared to own cluster.

$\max \sum_i s(x_i)$

[Bounded]

$S_i(x_i) = 1$,

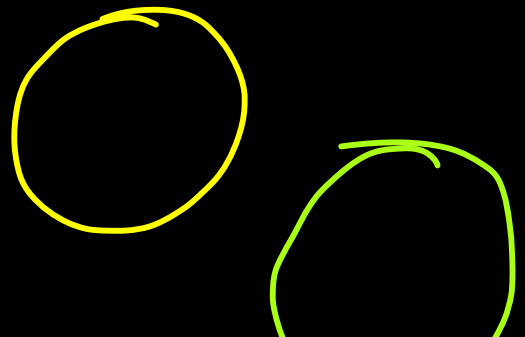
$S_i(x_i) = -1$, $a > b$ & $b = 0$

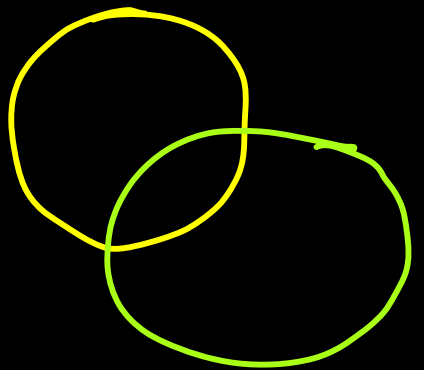
(incorrectly alotted clusters)

x_i → x

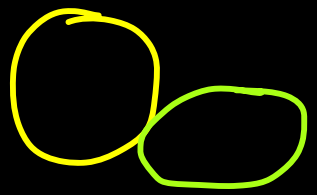
$0 > S_i > 1$

$-1 < S_i < 0$





Si ~ 0



Determining the best 'K'

→ Visually

→ Domain knowledge

└ Slack groups

→ avg reply time, # users, # msgs

→ support channels (sales / ops)

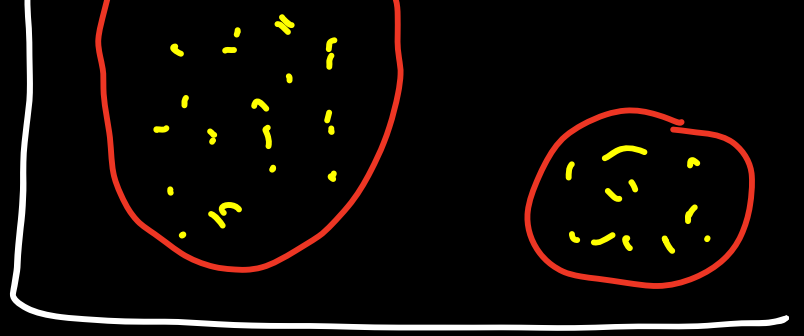
→ discussion forums (students)

→ internal teams (employees)

→ Elbow method

Q: which is best K??

a) 3



b) 2

c) 14

→ We can visually see that its a) 3

Is How do you tell that numerically?

=> Metrics!!

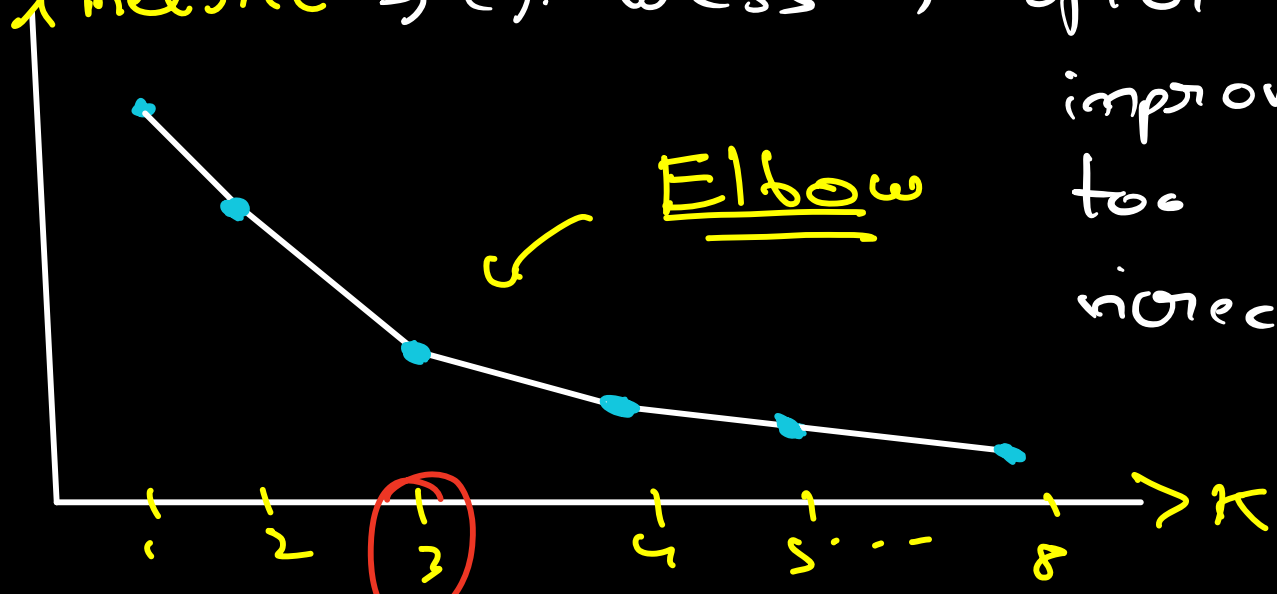
1) WCSS

2) Silhoutte Score

3) Dunn, etc.

Elbow method

```
for K in range (P):
    res.append (metric (clustering (K)))
plt.plot (res)
```



after a while improvement Δ becomes too small to increase 'K' further

→ code

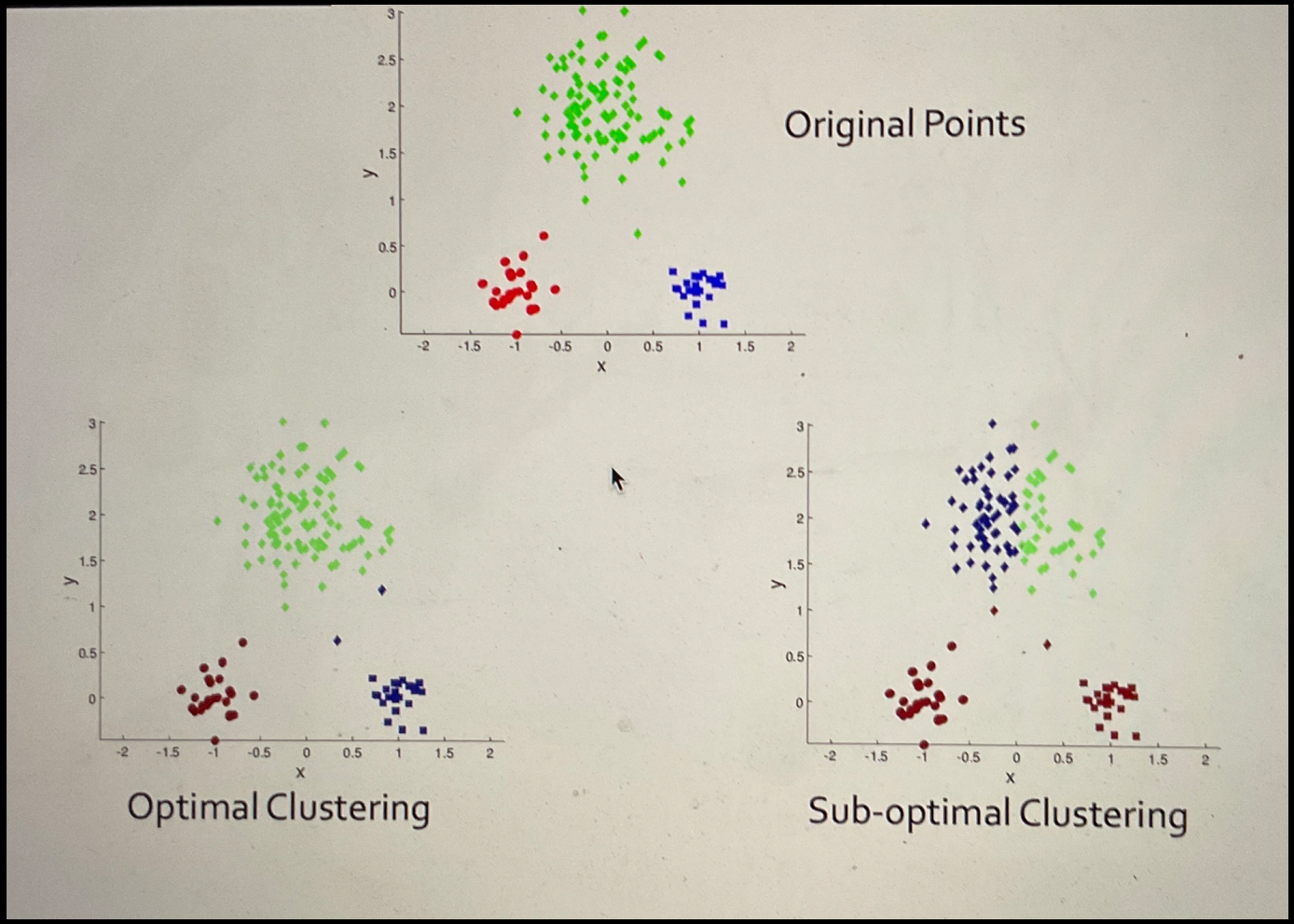
→ optimum K

Kmeans ++

There is an issue with the lloyd's algo.

→ Because, centroids are randomly initialized we may get different results some times. (bad)

→ animation



Solution

→ choose centers smartly.

Idea:

→ 1st centroid initialised randomly

→ then choose 2nd center as the furthest point from first

→ 3rd to be far from 1st and 2nd

→ animation

What about outliers?

→ Instead of choosing outliers we could choose furthest point as centroid with prob proportional to distance.

→ Remove outliers for learning centers.

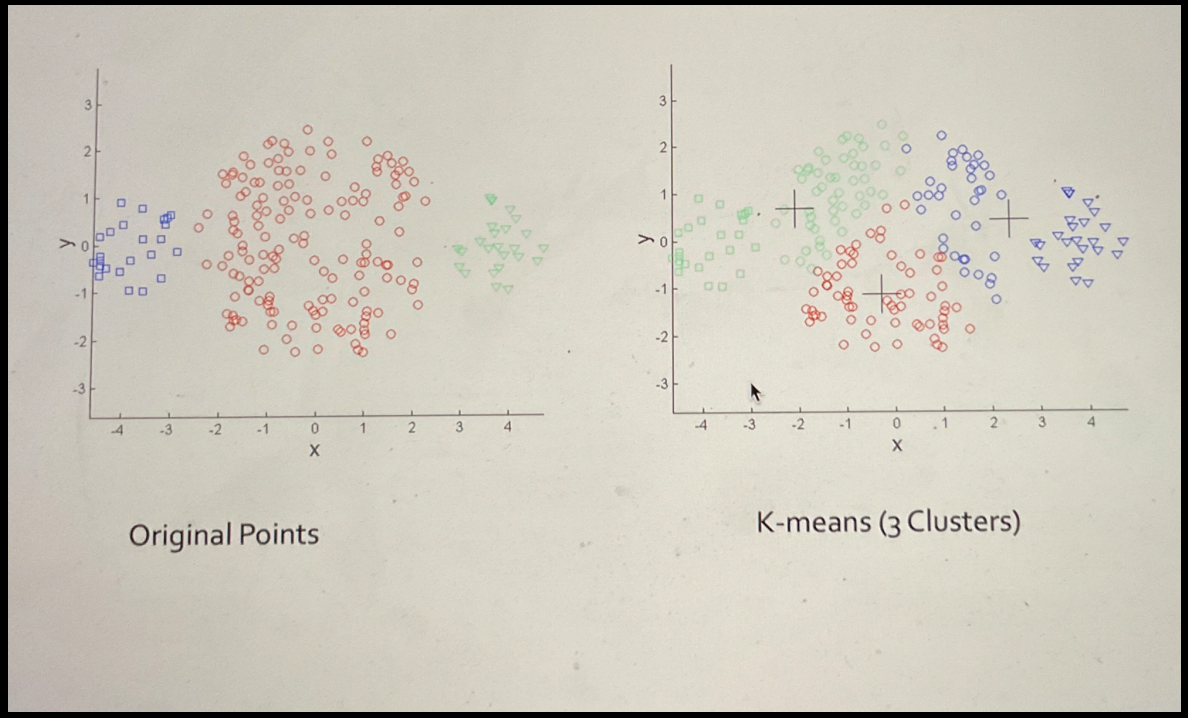
→ sklearn.

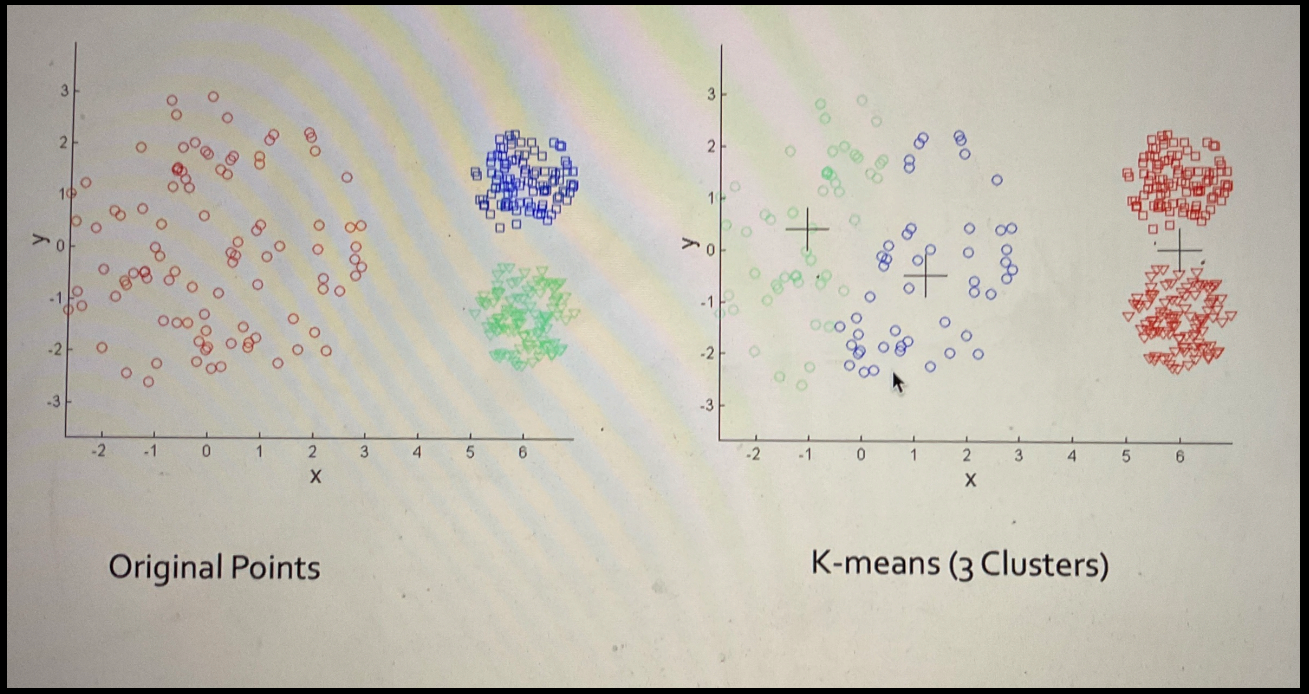
Limitations of K means

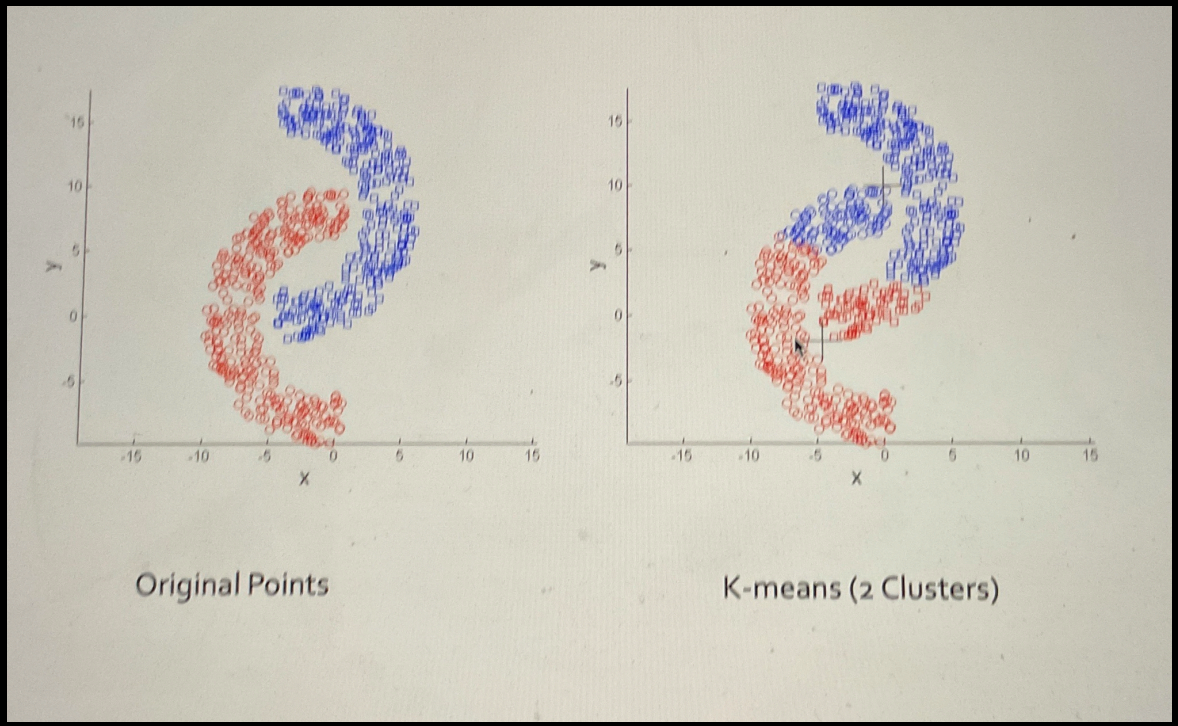
→ Need to decide "K"

→ May not be best for ___ clusters.

- → Diff size
- → diff density
- → non globular (hyper-spheres)




Original Points
K-means (3 Clusters)


Original Points
K-means (2 Clusters)