Introduction to Clustering

KMeans Algorithm

[Unsupervised Learning]

→ Module Intro

→ Clustering intro

→ Kmeans

CASE STUDY

You are a data scientist at Amazon, and you need to decide different marketing strategies for different types of customers.

How will you identify the different customer groups and assign a group to every customer ?

DATA:

ID	n_clicks	n_visits	amount_spent	amount_discount	days_since_registration
1476	130	65	213.905831	31.600751	233
1535	543	46	639.223004	5.689175	228
1807	520	102	1157.402763	844.321606	247
1727	702	83	1195.903634	850.041757	148
1324	221	84	180.754616	64.283300	243

→ Quiz

Q: What is the difference b/w this and a classification problem?

→ Here we have no target.

In binary classification, we had:

$$D = \left\{ (\vec{x}_i, y_i)_{i=1}^{\hat{}} \; ; \; \vec{x}_i \in \mathbb{R}^d \; ; \; y_i \in \{0, 1\} \right\}$$

multi-classification → $y_i \in \underbrace{S}_{\text{finite set of classes}}$

regression → $y_i \in \mathbb{R}$

Unsupervised Learning

However, we do not have a 'target' in our problem.

$$D = \{ \vec{x}_i ; \; x_i \in R^d \}$$

Such probems are called unsupervised learning

Q: Is this a valid use case then?

→ Offcourse.

Example:

→ customer / product segmentation

→ grouping similar images in galery

→ detecting similar stocks automatically

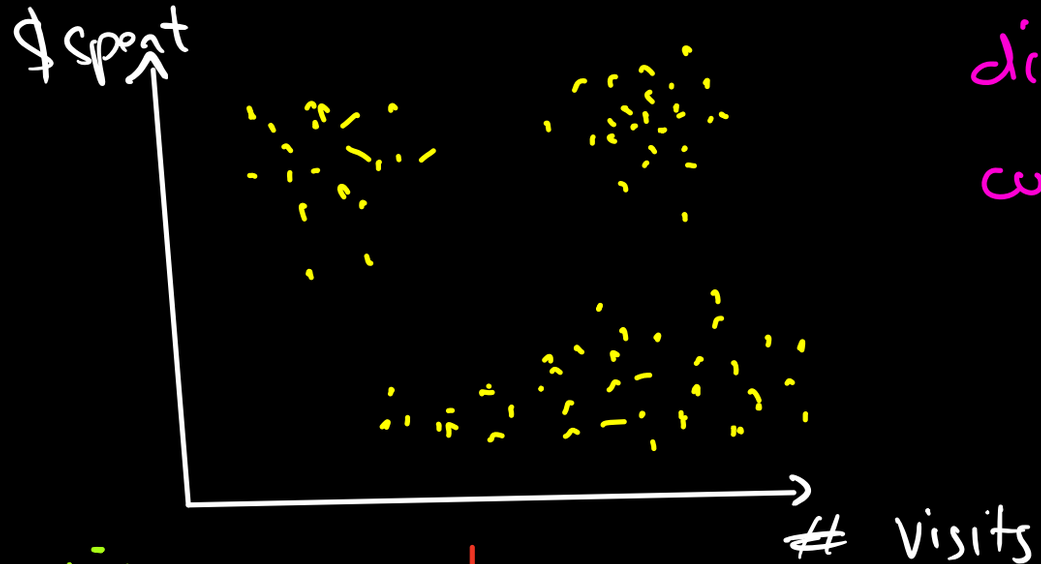
The process of grouping any kind of data based on similarity in their features, automatically, without human expertise, is called **Clustering**

This is one type of unsupervised - learning

Other examples of unsupervised learning:

- → Association rules / Recommenders
- → Dimensionality reduction (Eg: PCA)
- → Anomaly detection
- → Encoding (word-to-vec)

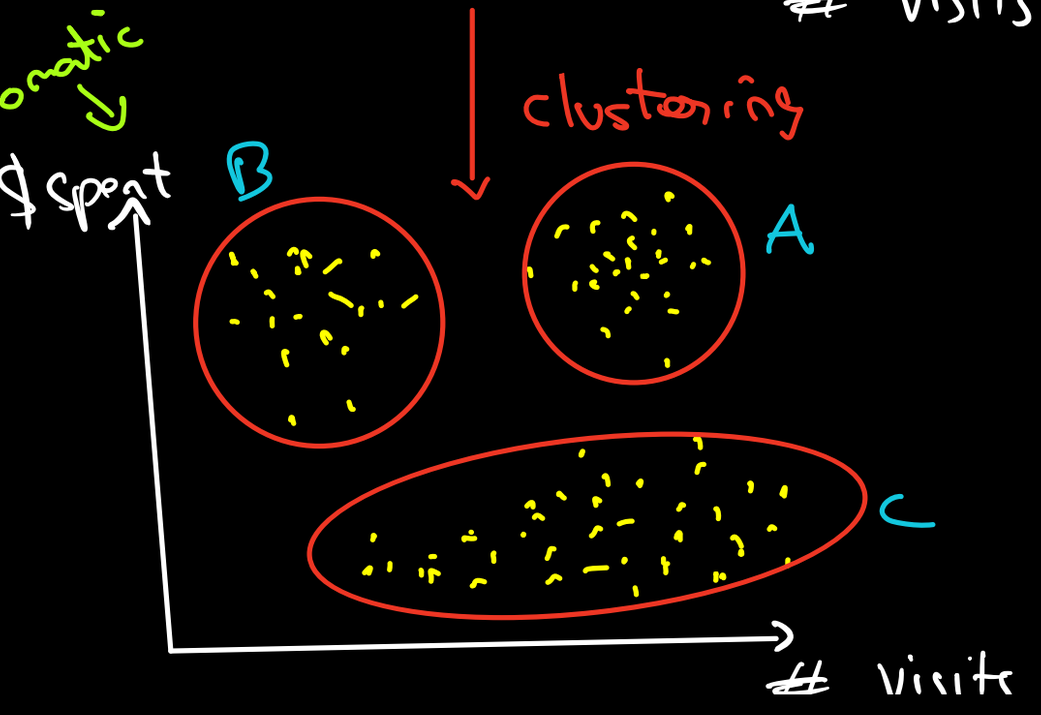
Clustering



Q: Can you spot different types of customers?

automatic

clustering



Intuitions

A → Heavy shoppers
Visit a lot / Spend a lot

B → Rich people
spend whenever they visit

C → Window shoppers

Q: What would you recommend for each group??

→ No ads needed for 'A'

→ Only ads needed for 'B'

→ Ads + discount for 'C'

Intuitively, clustering is dividing a population into groups such that the pts in one group are similar to each other.

Each group is called a **cluster**

So the task is
- → group similar points
- → define "similarity"
  - ↓
  - Eg: distance on a scatter plot

Q: How do you decide if a cluster is good or bad??

→ There is no ground truth data!
Hence nothing to compare with

→ It should simply make 'business sense'

Advanced applications of clustering

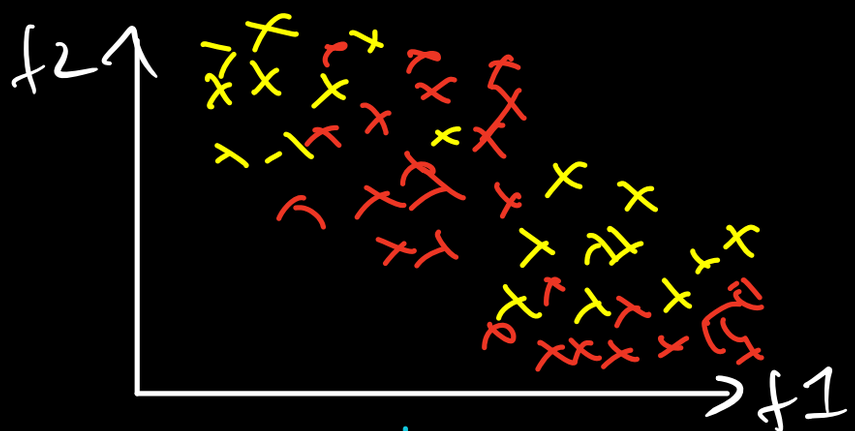
→ Validation of domain knowledge / human opinion

→ Search algorithms and recommender sys.

→ Feature creation

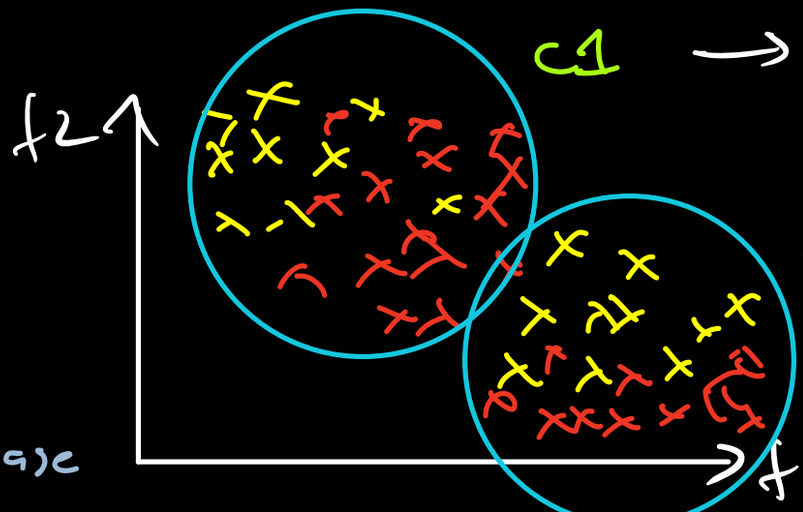
→ Clustering with supervised learning to improve model accuracy!!

Example :



Build a classifier for this problem.

Q: Any guesses??



C1 → Logistic Regression 1

C2 → Logistic Regression 2

→ Our use case

[cascading]

input → Cluster → LR1 → Yellow / Red

Cluster → LR2 → Yellow / Red

How to perform clustering

Q: Any ideas ??

So far we have seen a pattern

Geometrical idea

↳ Convert to math

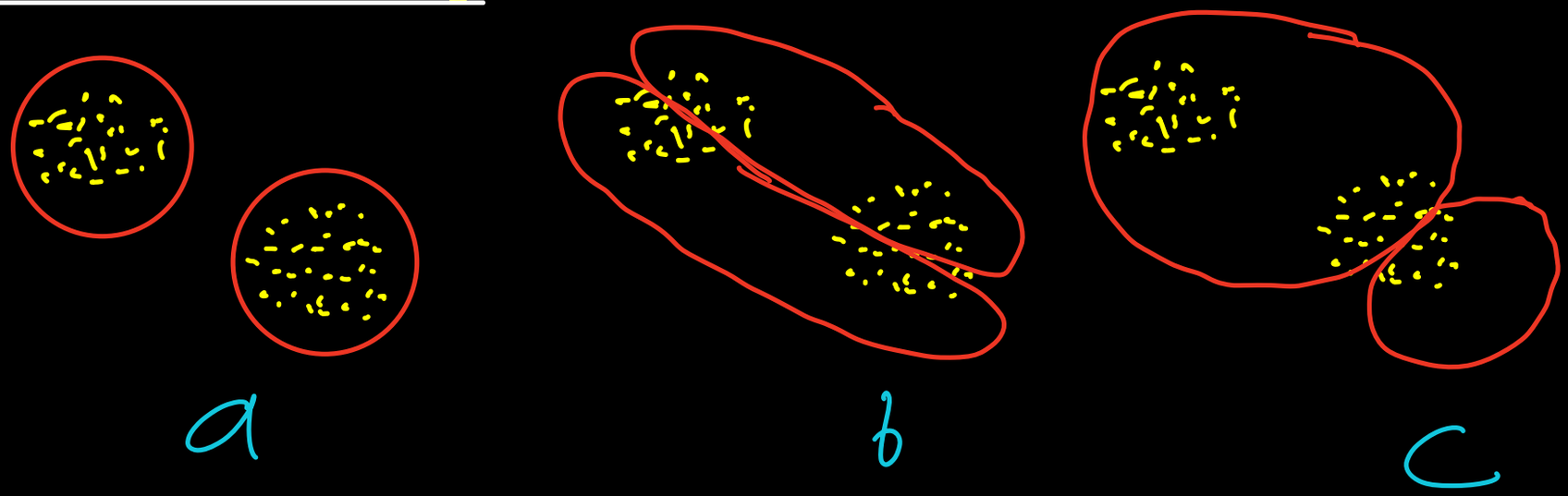
↓

Design an optimisation function

↓

optimize

Good Clustering:



How to express this mathematically ??

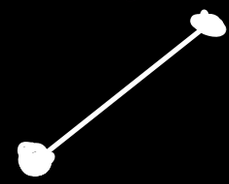
→ within a cluster variance / distance should be as low as possible

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

Q: What is distance mathematically ??

→ Euclidian distance

→ Low dimensions



→ Manhattan distance

→ Medium dimensions

→ Cosine distance

→ high dimensions

→ etc..

So the choice of the distance metric is a hyperparameter for many clustering algorithms.

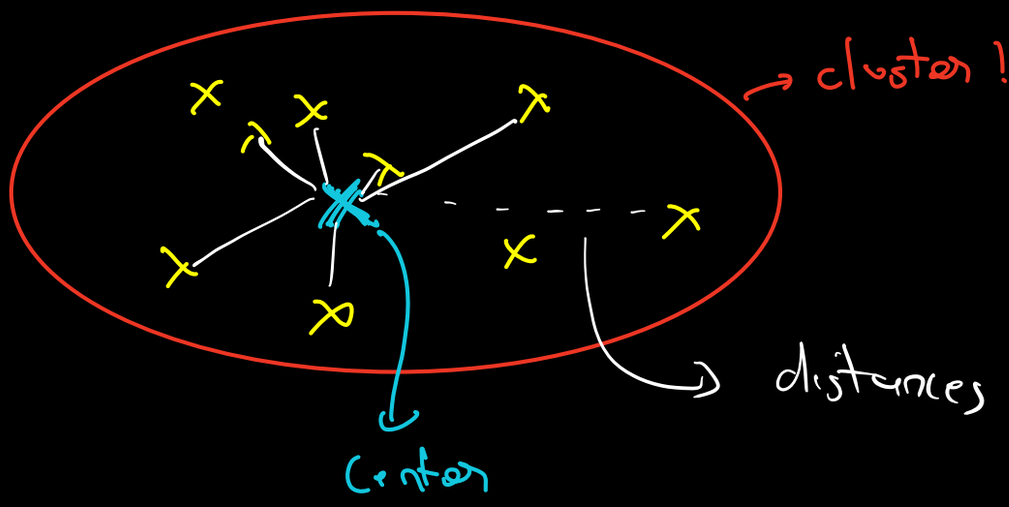
→ pre-read.

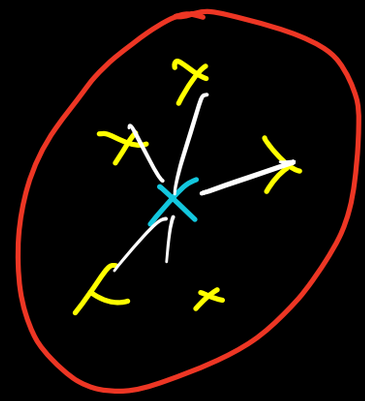
So, we want to come up with a metric which can tell us which clustering is best!

Idea:

→ Minimise within cluster distance.

Within Cluster Sum of Squares (inertia)





$$\sum dist_{c_1} + \sum dist_{c_2}$$

$$WCSS = \sum_{j=1}^{K} \sum_{i \in C_k} (x_i - \bar{x}_j)^2$$

point

all clusters

all pts in j^{th} cluster

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 :

$$WCSS = \sum_{i=1}^{K} \sum_{j=1}^{m_K} \text{distance}(x_{ij}, C_i)^2$$

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

Lloyd's Algorithm (K-Means Clustering)

Steps:

→ Randomly initialize K centers

→ assign points to nearest center to get your clusters

→ find the centroids of these clusters → because this will reduce WCSS

→ Re-assign points

→ Repeat until new centers = prev centers

→ animation

→ 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: $n^{\text{pts}} \cdot c^{K} \cdot i^{\text{iters}} \cdot d^{\text{dim}}$

→ code

Determining the best 'K'

→ Visually

→ Domain knowledge

  └ Slack groups

  → avg reply time, # users, # msgs

    → support channels (sales / ops)

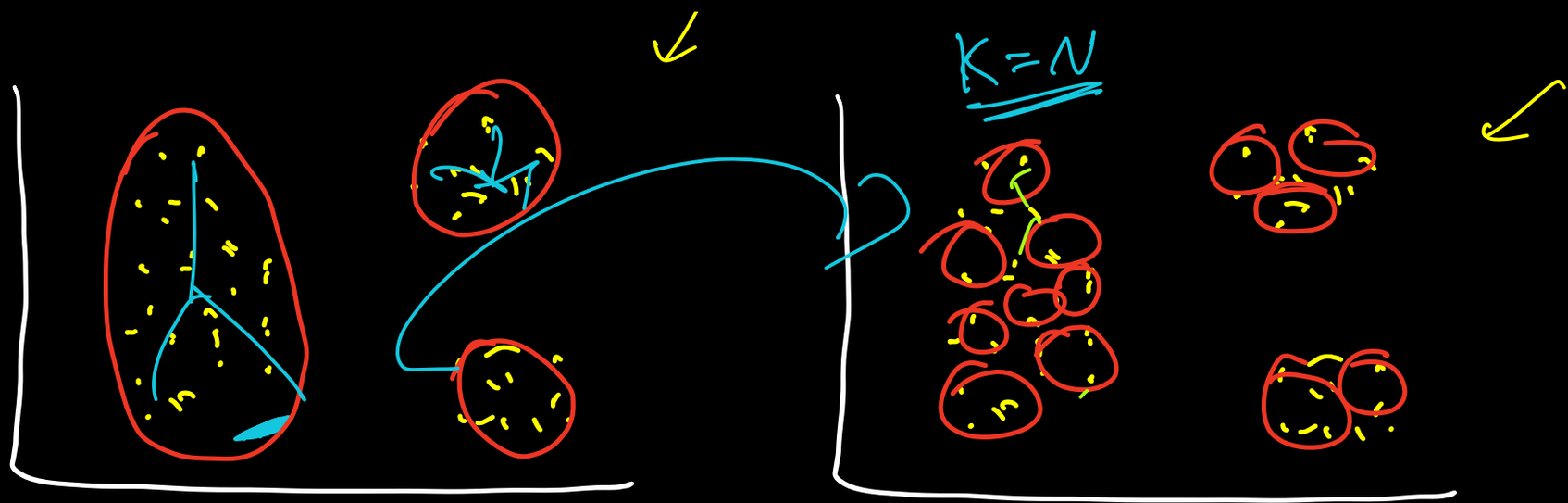
    → discussion forums (students)

    → internal teams (employees)

→ Elbow method

Q: which is best K ??

a) 3



→ We can visually see that its a) 3

Is How do you tell that numerically?

=> Metrics!!

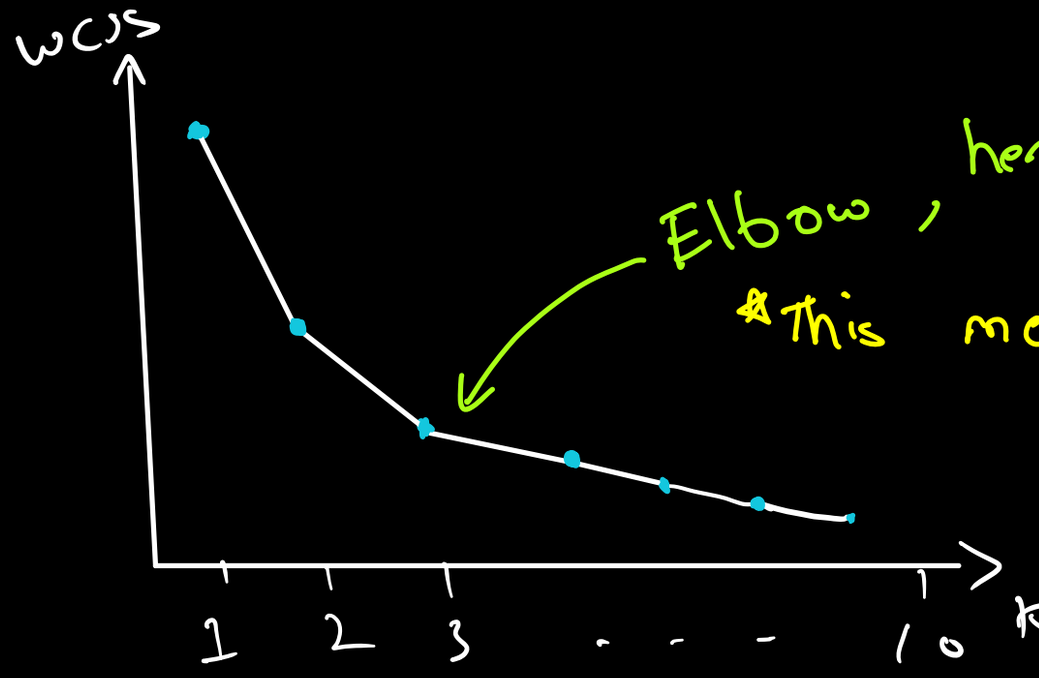
1) WCSS

2) Silhoutte Score

3) Dunn Index

--.. others

Elbow Method



hence K = 3

* this method is just a guideline. Final decision is based on "business logic".

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