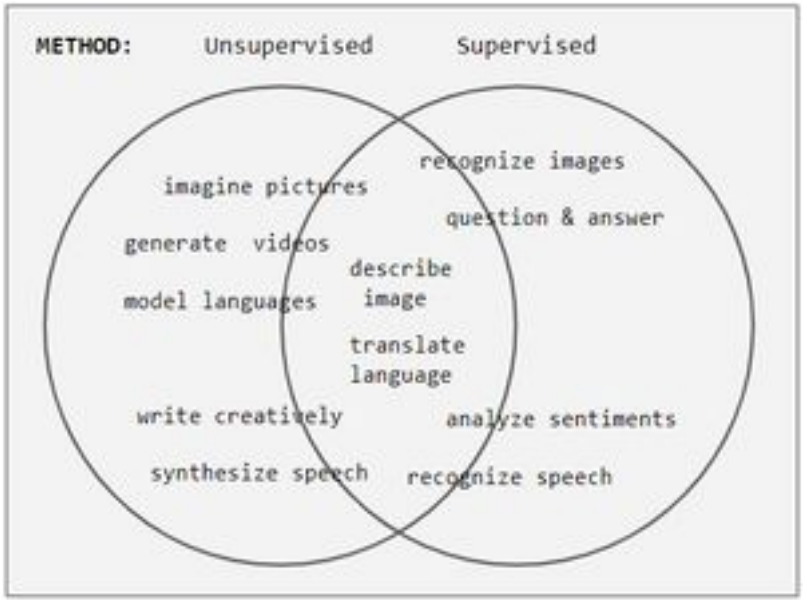
Unsupervised Learning Algorithms

Agenda - Schedule

1. Unsupervised Learning
2. Unsupervised Methods
3. Clustering
4. Break
5. TLAB #2

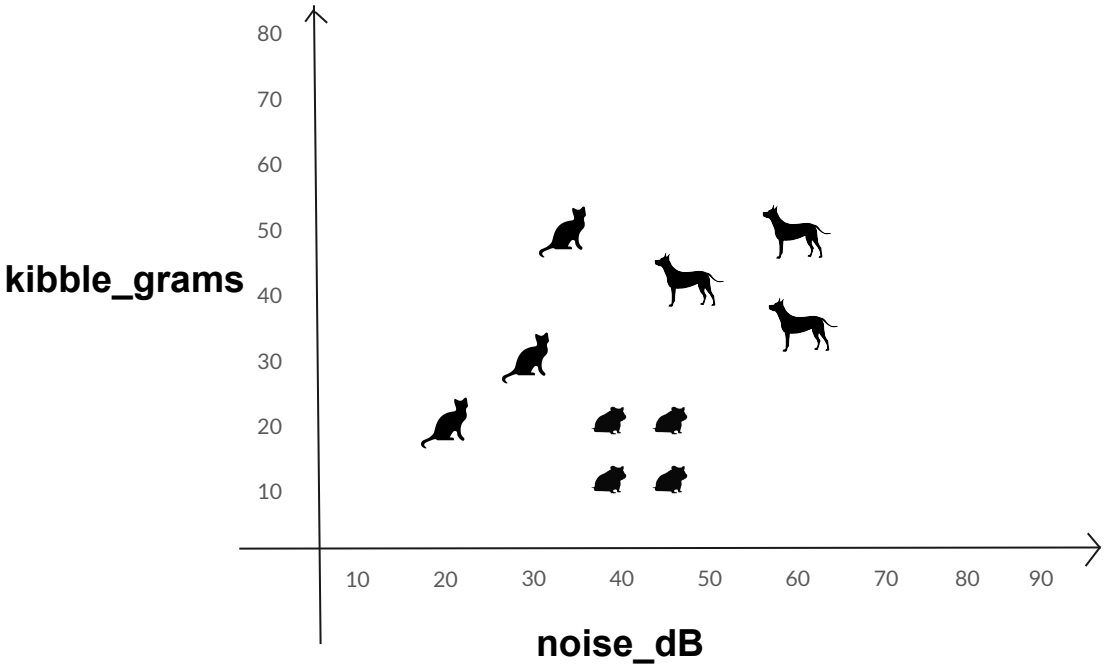


What differentiates the two?

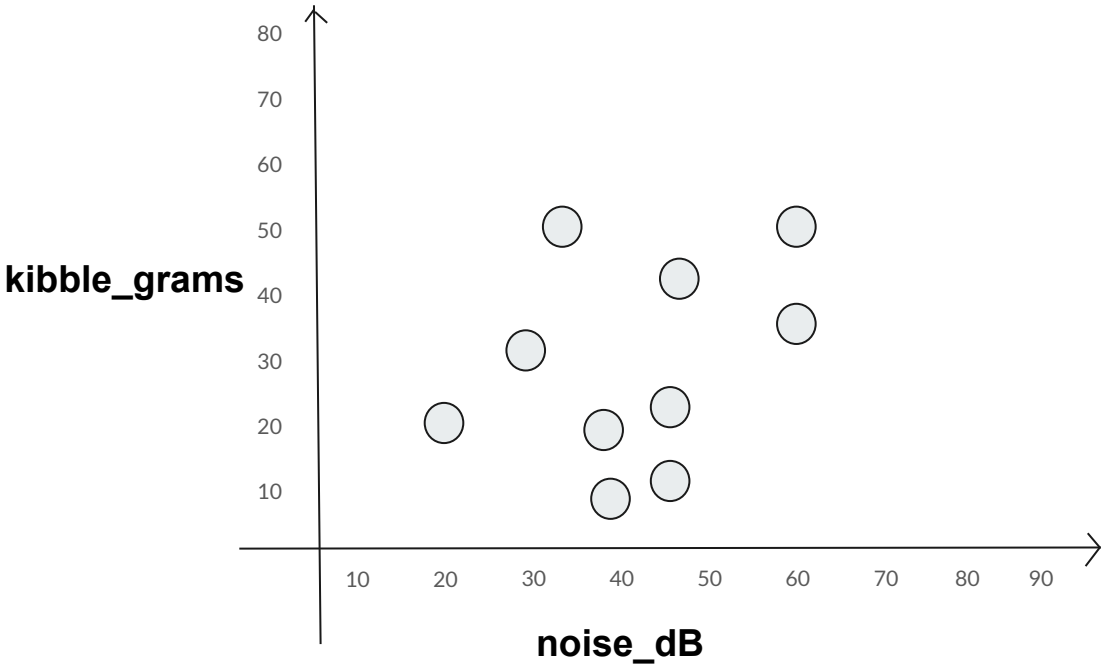
Agenda - Goals

-

Unsupervised Learning

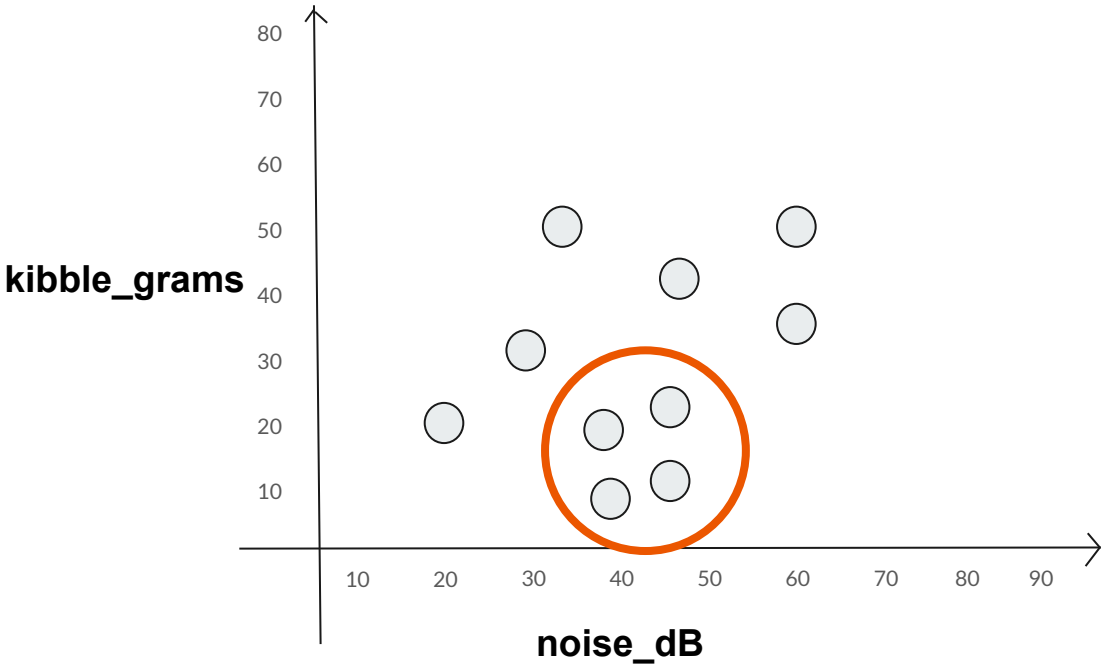


Let’s bring back this slide one last time. What if these data-points weren’t labeled…

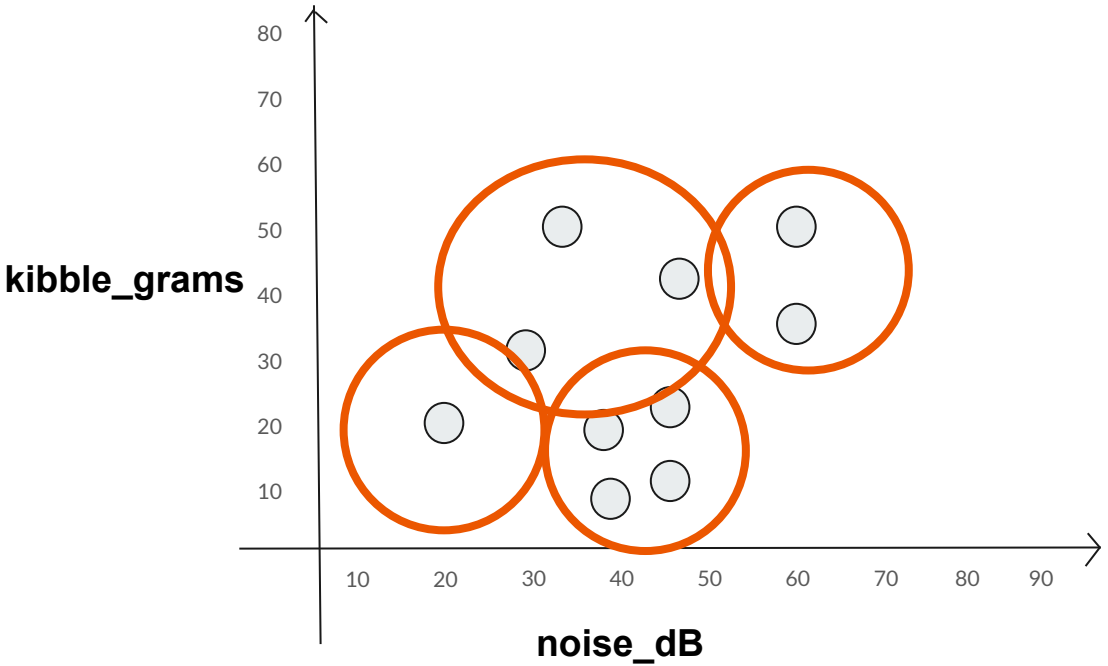


Would you still be able to determine which samples “**go together**?”

In abstract terms, it seems like the answer is yes!



It appears that the we could “cluster” our samples visually. But how do we speak about this in any definite terms?



Without target variables, we're basically on our own. How can we determine if these **are valid clusters**?

Unsupervised Learning

This field of machine learning presents a **unique challenge.**

Instead of being given an answer and “training” coefficients or recognizing training samples in order to best recreate a relationship,

We instead ask our learning algorithms to **recognize innate structure of datasets**!

Let’s review some useful applications of unsupervised learning…

Unsupervised Learning - Applications

Natural Language Processing: Both person A & person B used the "😤" emoji in their comment on our product. **Why?**

Image Recognition: We have 100 pixels in our image, which ones are the **most important** for multiclass classification?

Customer Segmentation: Both customer A & customer B are between 30-40 years old and spend $50 a month. Does this describe a **group**?

Recommendation Engines: Both person A & person B watched ***Love is Blind*** on Netflix at 10 PM. **What will they binge next?**

Employee Classification: Both employee A & employee B needed access to folder Y, **which permissions do we give them?**

Unsupervised learning could be a **preprocessing step** in a supervised learning pipeline

Unsupervised Learning - Applications

Natural Language Processing: Both person A & person B used the "😤" emoji in their comment on our product. **Why?**

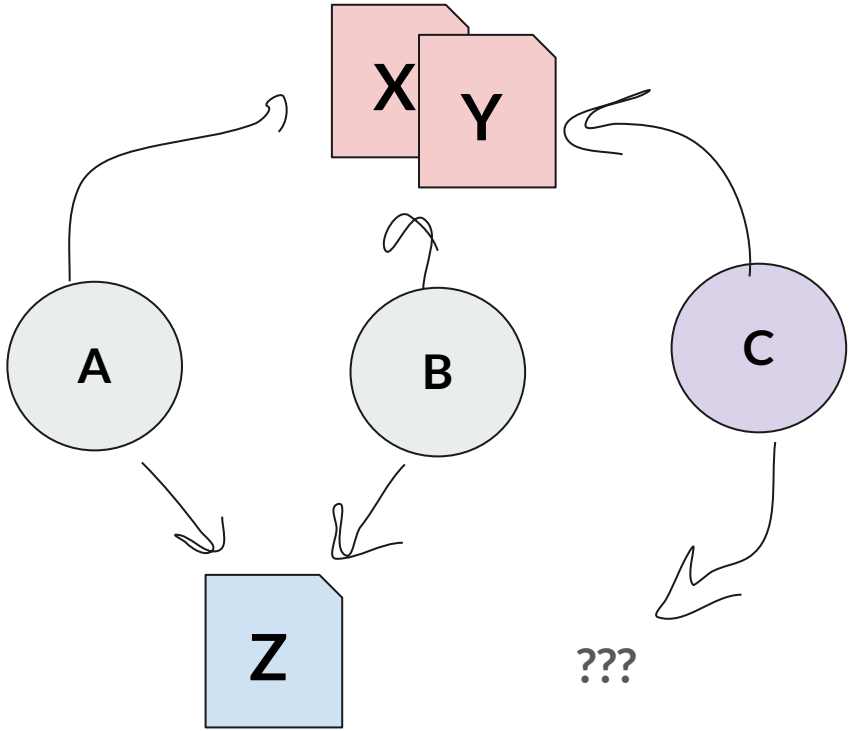
Image Recognition: We have 100 pixels in our image, which ones are the **most important** for multiclass classification?

Customer Segmentation: Both customer A & customer B are between 30-40 years old and spend $50 a month. Does this describe a **group**?

Recommendation Engines: Both person A & person B watched ***Love is Blind*** on Netflix at 10 PM. **What will they binge next?**

Employee Classification: Both employee A & employee B needed access to folder Y, **which permissions do we give them?**

Which article should we recommend to **UserC**? **Why**?

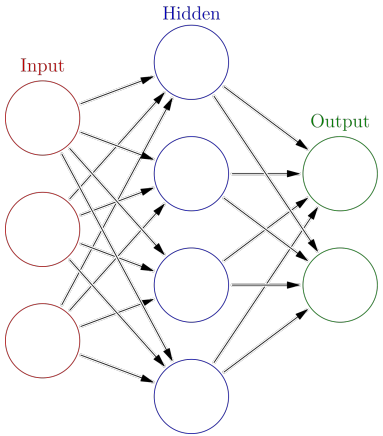


Unsupervised learning: a metaphor for interpretation. **UserA, UserB**, and **UserC** all read **ArticleX & ArticleY**. Shortly after, **UserA** & **UserB** both read "**ArticleZ**"…

Unsupervised Learning - Different Methods

Let’s review a few unsupervised learning methods:

- *PCA*
- *K-Means*
- *Hierarchical clustering*
- *DBScan*
- *Neural Networks …*



Unsupervised Machine Learning

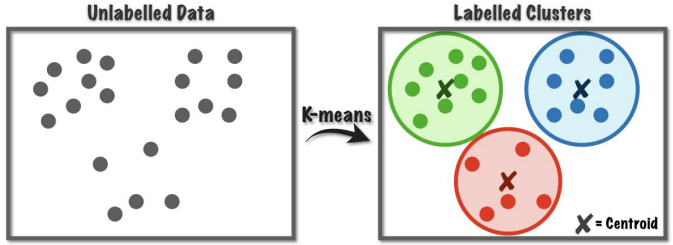
K-Means Clustering

We will discuss this algorithm today.

Although KNN and K-Means share the same letter, they are quite different!

K-means is our first unsupervised machine learning model. Instead of defining the groups exactly, we tell the model "*I think we have k groups*"

The model then calculate *k* number of means which center on each group

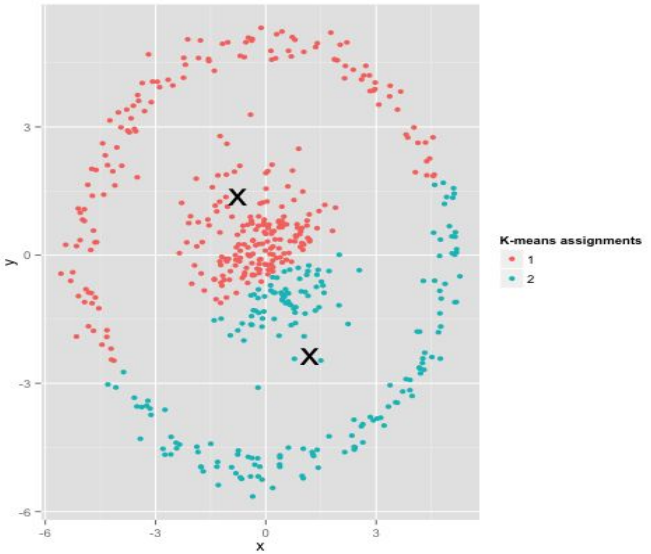


K-Means Clustering

It calculates *k* centroids which best fit the data based on the numerical values

These centroids are determined by a “mean” which has the smallest distance from all the groups

The downside is it struggles with complex datasets such as the one to the right since it is purely distance based



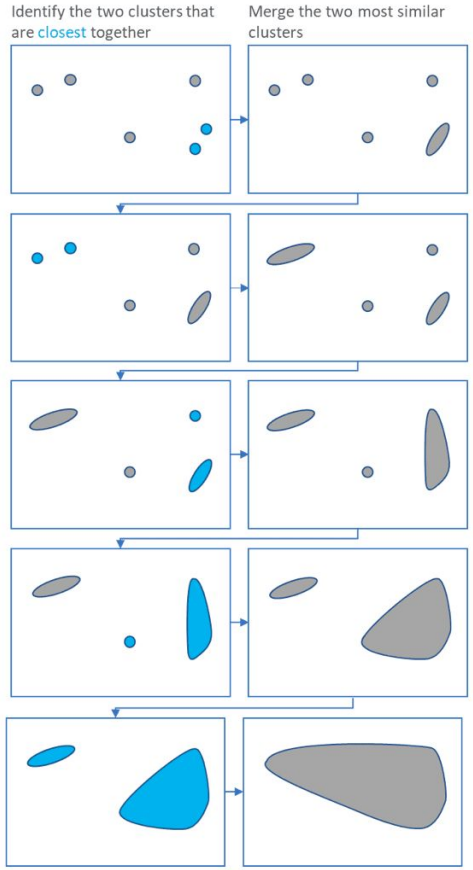
Hierarchical Clustering

Hierarchical clustering works backwards

It starts with assuming all data points are their own group

It then starts merging the closest data points together until they are all one group

We choose the point at which we feel the data is well clustered



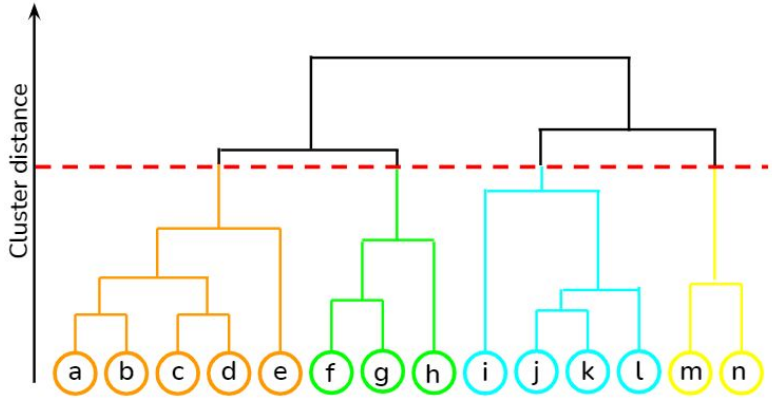
Hierarchical Clustering

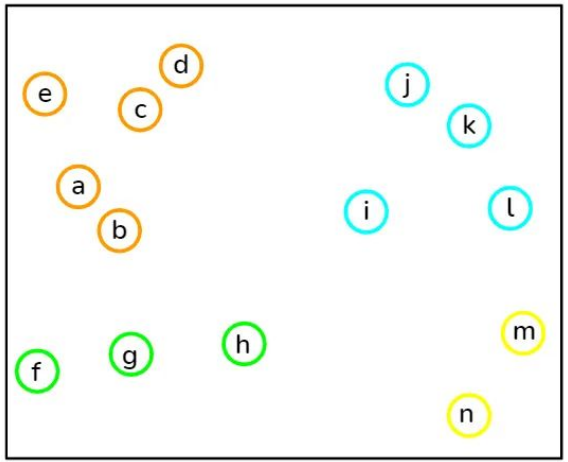
We visualise hierarchical clustering with a dendrogram

We choose a **cutoff point** based on distance

There are different ways to cluster based on our goal and we will discuss these when we reach hierarchical clustering!

It has an advantage of allowing us to deal with more complex shapes





DB SCAN

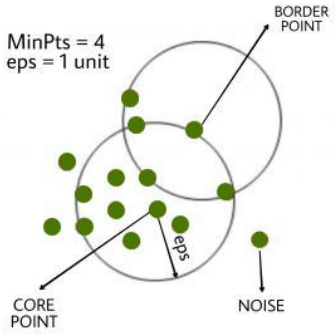
DBSCAN is a special algorithm standing for: Density-Based Spatial Clustering of Applications with Noise

It has 2 main parameters:

- **minpts** -> the minimum number of points clustered together for a region to be considered
- **epsilon** -> the distance which we will look for nearby points

We draw a circle with radius **epsilon** around each point and check to see if we have at least that many **minpts** surrounding that point in that circle

If yes, we consider them a **cluster**

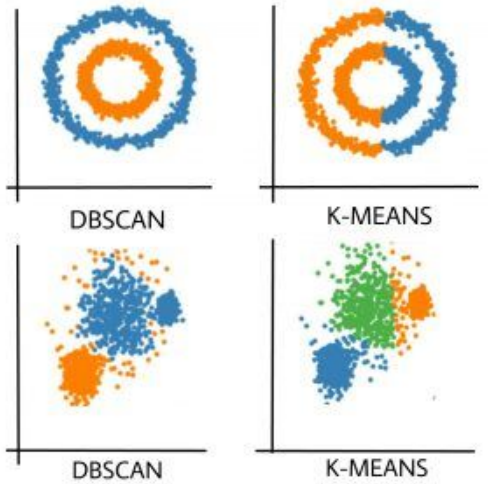


DB SCAN

If there are no points around a core point, then it is considered noise

DBSCAN is very good at discerning complicated shapes and finding patterns in non-linear data

It may be better understood through a visualization: https://www.naftaliharris.com/blog/visualizing-dbscan-clustering/



Neural Networks

Neural Networks

Neural networks are among the **most powerful and most complicated machine learning algorithms** we will discuss because they are not just one algorithm, but rather a network of algorithms

We will discuss them in great detail and their mathematics

It is a combination of **linear algebra and calculus**

They underlie many of the most advanced algorithms such as the model which runs **ChatGPT**

They are so named because they **parallel the neural connections in our mind**

Neural Networks

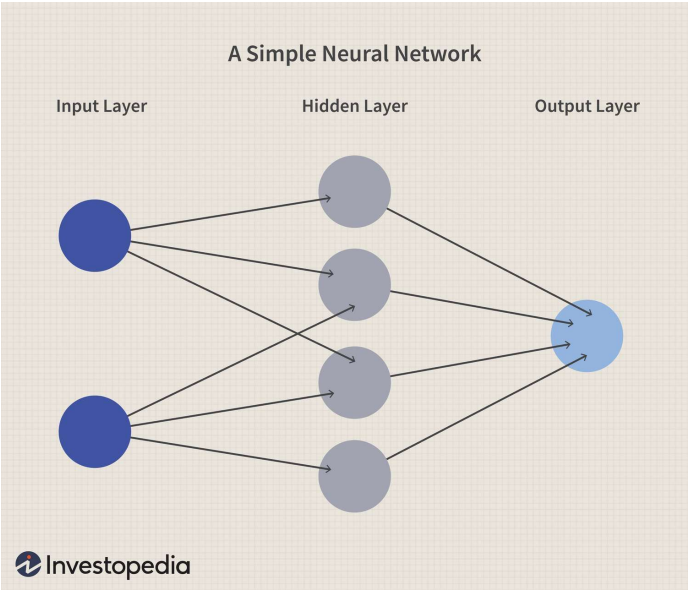
The key components to a neural network are layers made up of nodes:

Input Layer - the data we input

Hidden Layer - a series of calculation layers which take the data from the input layer

This is the magic, we do not determine what goes into the hidden layer, it simply works based on some predetermined mathematical properties we define

Output Layer - the final output from the network

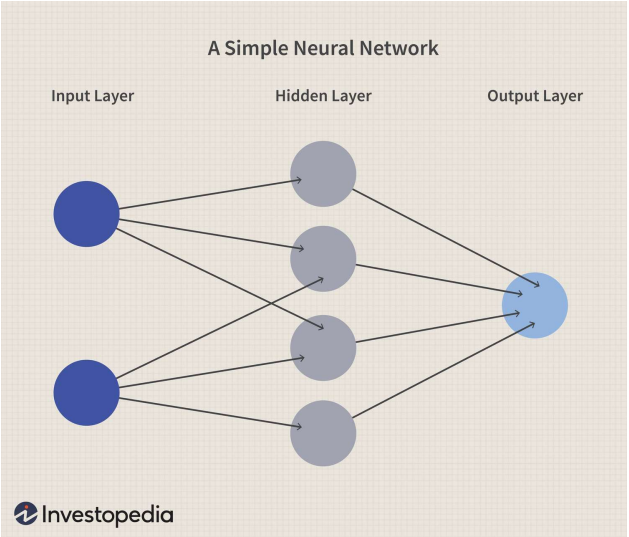


Neural Networks

The way these nodes learn are through mathematical calculations such as back propagation and forward propagation

These nodes in the **hidden layer** act like filters for information

They pick and choose what comes out in the output layer based on what is put in



Neural Networks

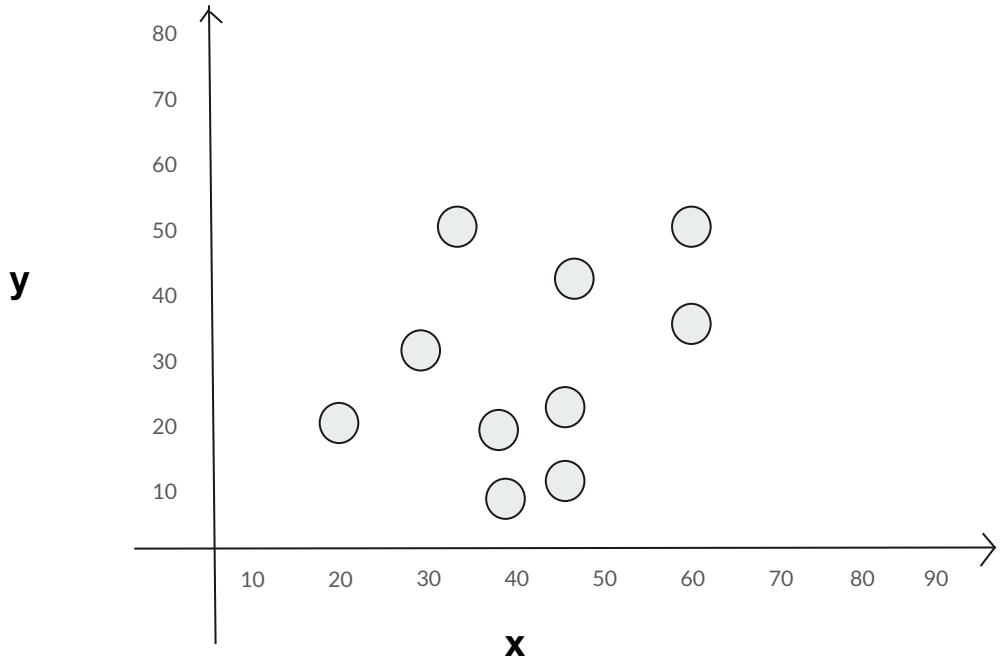
The hidden layer is truly the magic of our neural networks

The clever construction of these layers build the massive networks which define these neural network algorithms

Some other fancy neural networks include:

- ***Recurrent neural networks***
- ***Convolutional neural networks***
- ***Long Short-Term Memory networks***
- ***Large Language Models (transformers)***

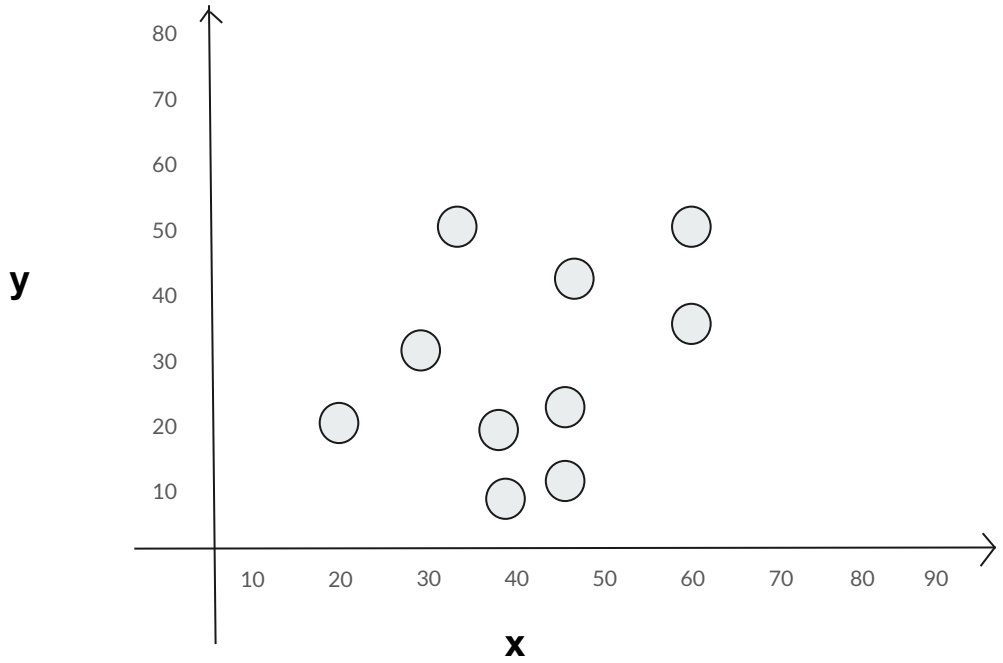
K-Means Clustering

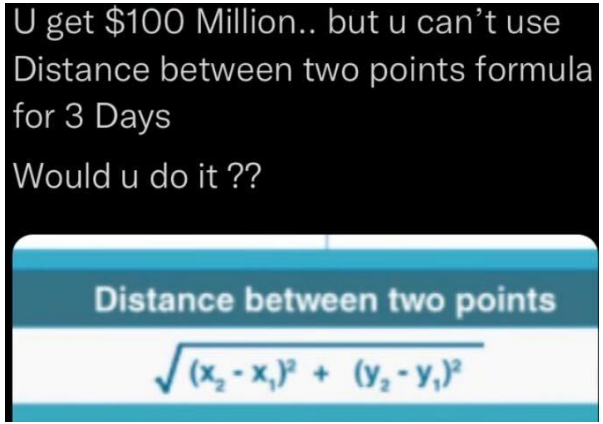


*Knowing what you know, how would YOU measure the “sameness” of two points in an unsupervised learning problem?

Let’s reevaluate the basics of K-Means Clustering through this visual example.

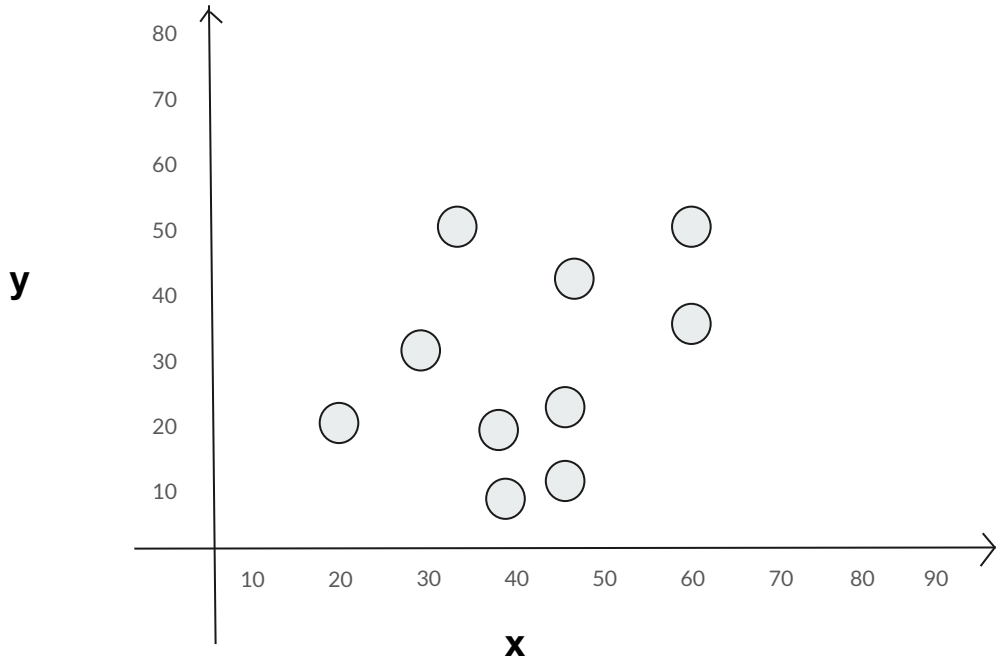
When it comes to clustering unlabeled samples, we need a measure of how **similar** or **different** two samples are.

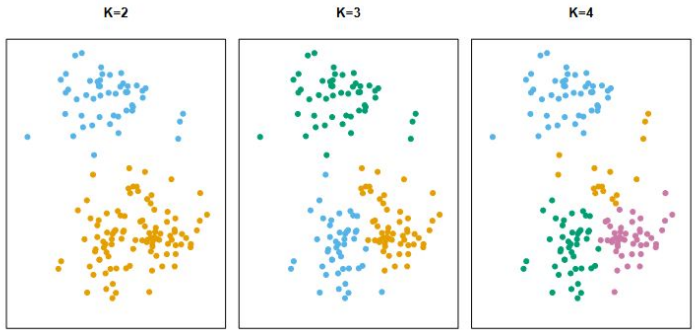




$$\frac{1}{|C_k|} \sum_{i,i' \in C_k} \sum_{j=1}^{p} (x_{ij} - x_{i'j})^2,$$

We use something like **euclidean distance!** **Specifically, "squared euclidean distance."** This is also known as **within-cluster-sum-of-squares**

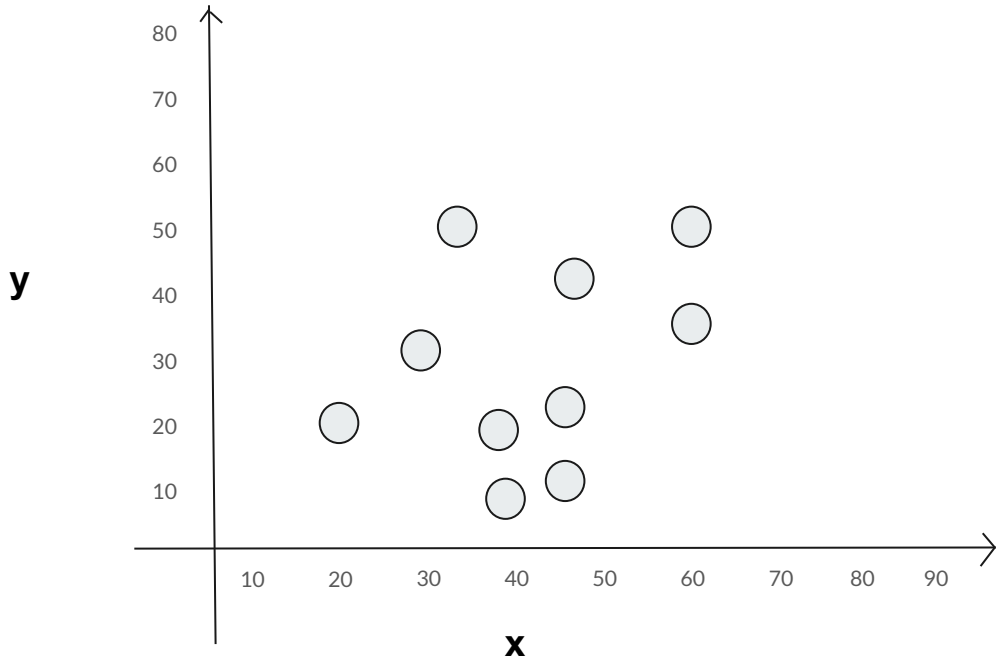




A varying *K* could result in different predictions. We will explore methods for determining the appropriate *K*.

However this isn't as simple as calculating distances between two points. After all, we **don't know** which samples belong to the same class.

Therefore we start with a **guess**. Namely, we "guess" that **K** many groups exist within this dataset. This is the "**K**" in "**K-Means Clustering!**"

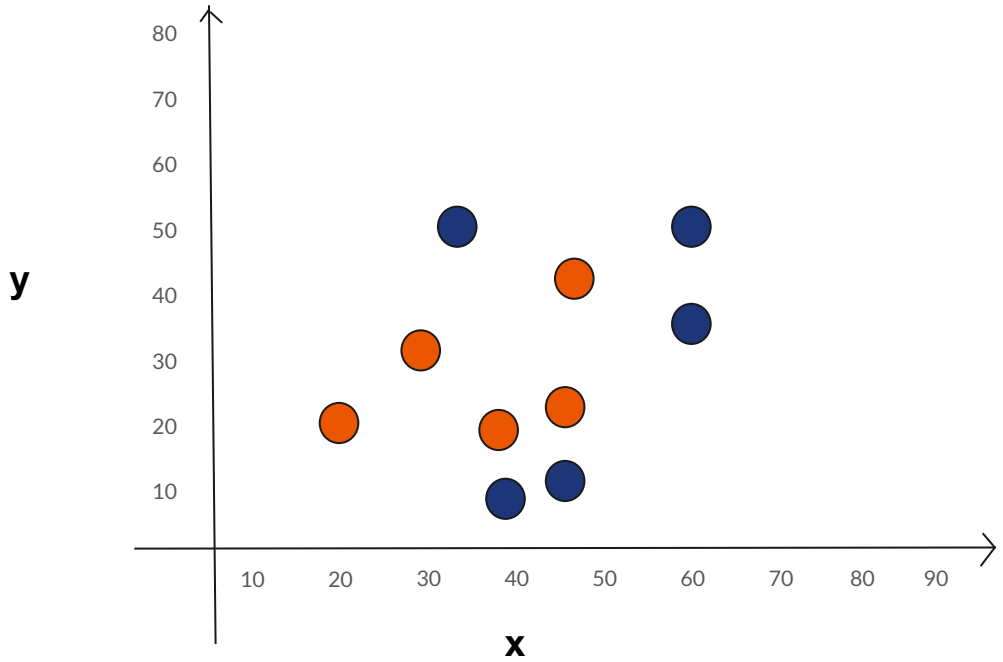


Step 1

Randomly assign each data point as belonging to cluster “1” or cluster “2”

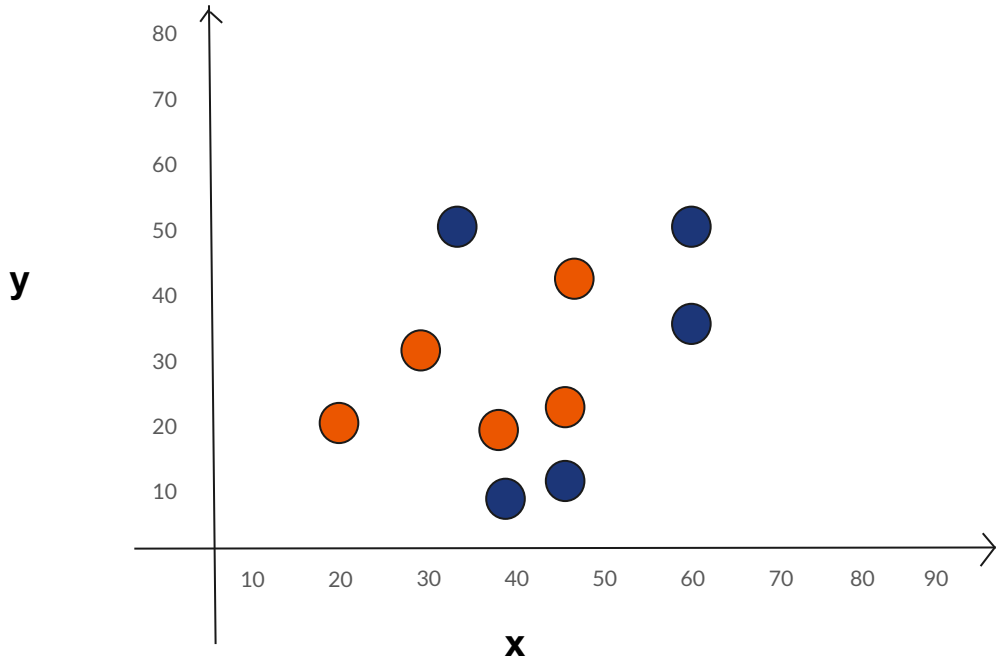
Once we choose this *K*, we **randomly** assign our data-points to belong to these *K* number of clusters.

For this example, let’s say that there are 2 clusters.

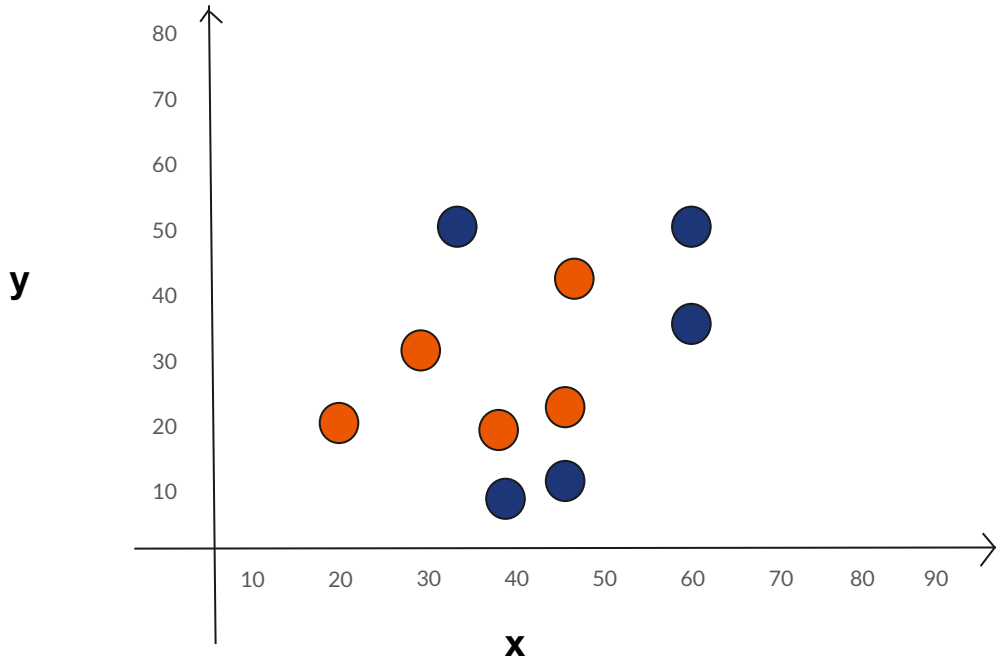


Do these clusters seem to make sense?

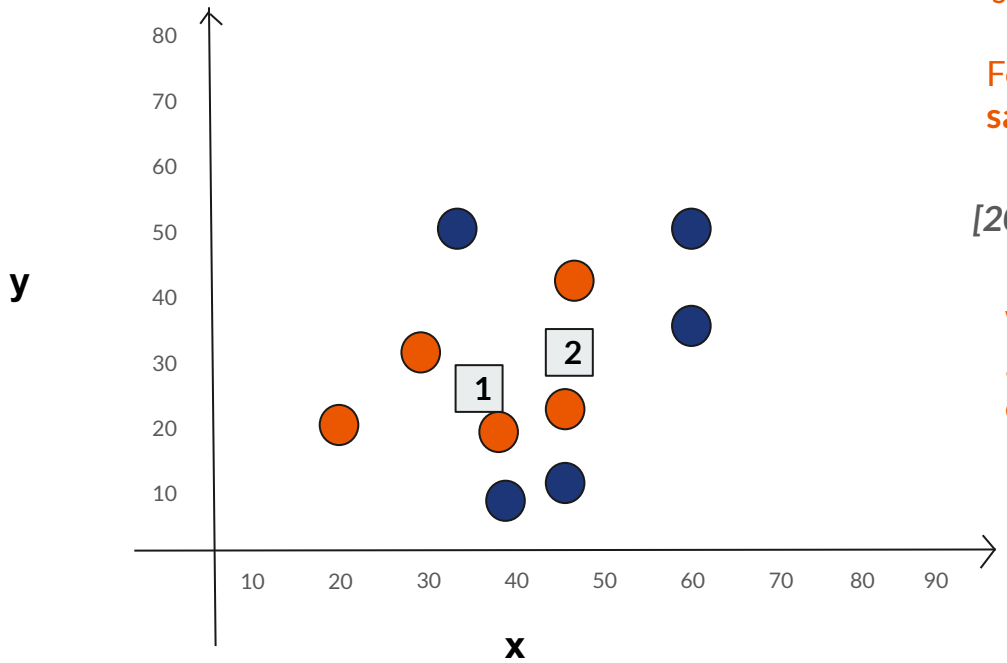
If your answer is no, what exactly seems to be the problem with them?



They seem to be **quite spread apart**. How do we measure “spread” in the world of stats?



We calculate the clusters **variance**, but to get variance we need some sort of **“mean”** value. Visually speaking, where would the “mean” value of a cluster exist?



*Simply put, the centroid is the average value (across all dimensions) of each sample in an assigned cluster

For example, we have the **following samples** in our "cluster 1"

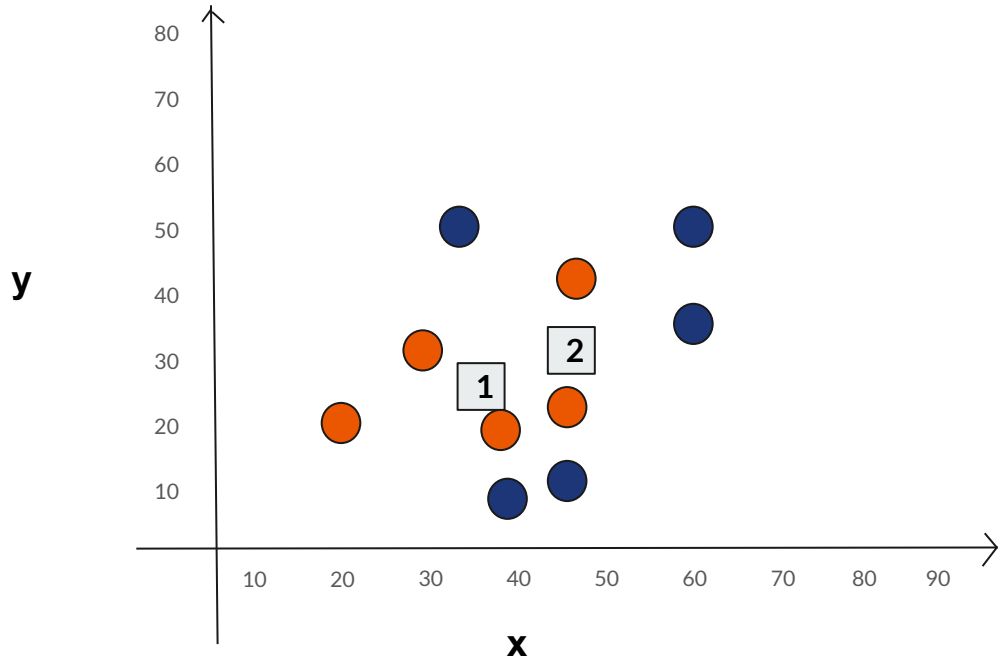
[20, 20], [30, 30], [40, 20], [50, 20], [50, 50]

We would calculate the average value of an x & y variable to get the **following centroid**

[38, 28]

These "mean" values of clusters are what we know as **centroids.** We define a centroid as a *vector of p feature means for the observations in the k clusters*

Cluster 1 Variance = 272.0
Cluster 2 Variance = 853.0



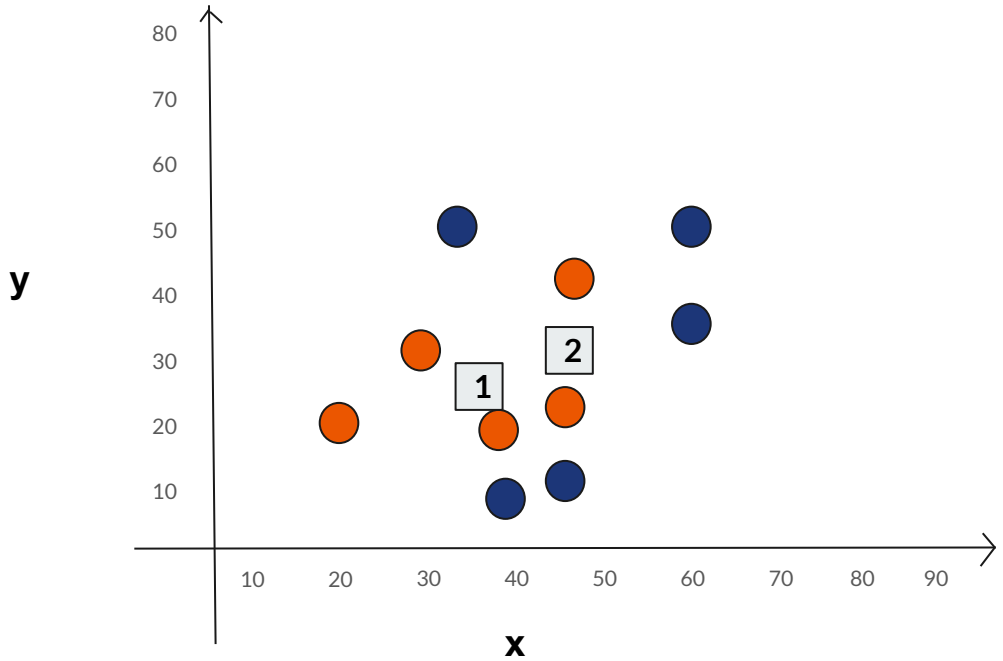
Step 2

Now that we have our centroid values, compute the variance for each cluster **using squared euclidean distance.**

$$\frac{1}{|C_k|} \sum_{i,i' \in C_k} \sum_{j=1}^{p} (x_{ij} - x_{i'j})^2,$$

Recall that the square heavily penalizes errors (and removes negatives)

We've calculated our variance. Should we **maximize or minimize** these variances? Another way to ask this question: *do we want **spread out** clusters or **tight-knit** clusters?*



$$\underset{C_1,\ldots,C_K}{\text{minimize}} \left\{ \sum_{k=1}^{K} \frac{1}{|C_k|} \sum_{i,i' \in C_k} \sum_{j=1}^{p} (x_{ij} - x_{i'j})^2 \right\}.$$

To exhaustively solve this calculation when partitioning **100 samples:**

K = 1

$1^{100} = 1$

K = 2

$2^{100} = 1.2676506e+30$

K = 3

$3^{100} = 5.1537752e+47$

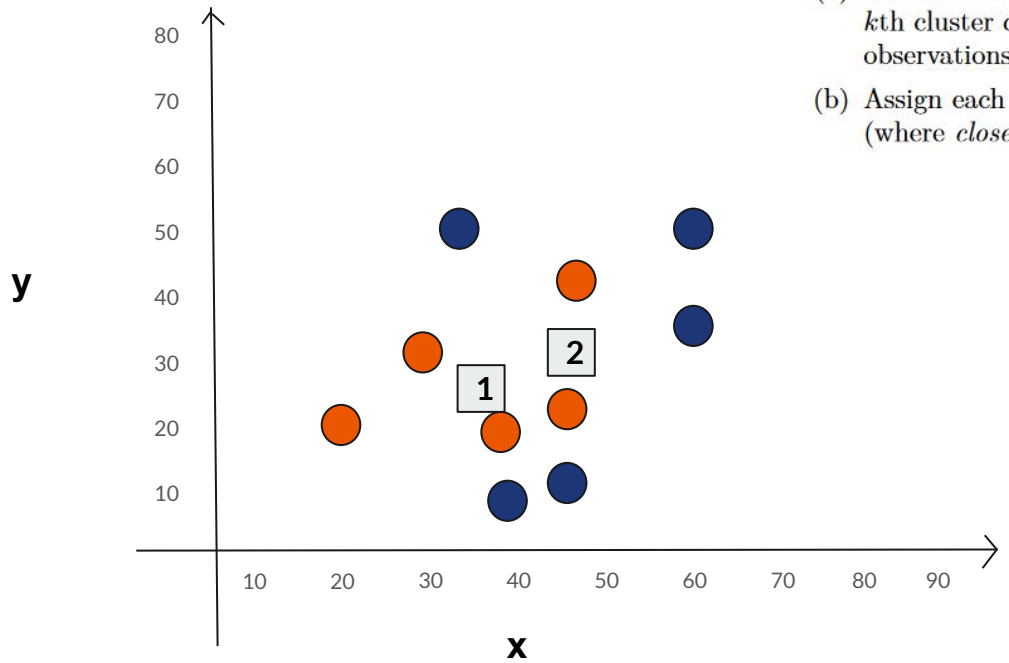
The art of CS/maths is getting a "best guess" without actually doing the calculations.

We absolutely want to **minimize these variances**. This is actually an **extremely expensive** task as we have K^n ways to partition our dataset…

2. Iterate until the cluster assignments stop changing:

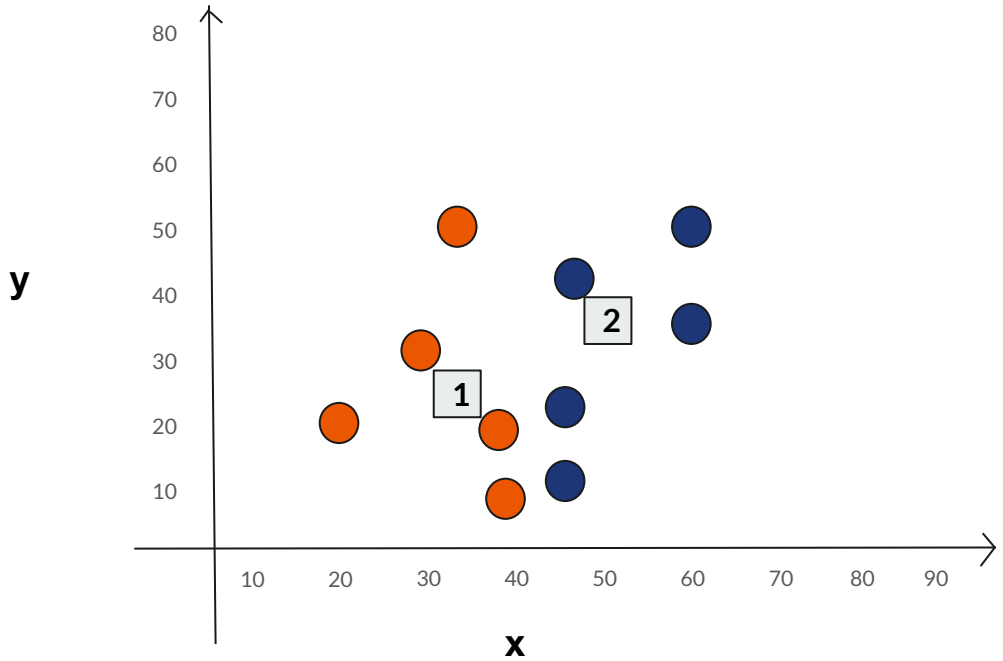
   (a) For each of the K clusters, compute the cluster *centroid*. The kth cluster centroid is the vector of the p feature means for the observations in the kth cluster.

   (b) Assign each observation to the cluster whose centroid is closest (where *closest* is defined using Euclidean distance).



So instead, we utilize a for-loop where we iteratively assign our samples to their **closest centroid using euclidean distance.** After assigning our samples, we calculate our variance again to check if we've created "better" clusters.

Cluster 1 Variance = 150.0
Cluster 2 Variance = 300.0



Step 3

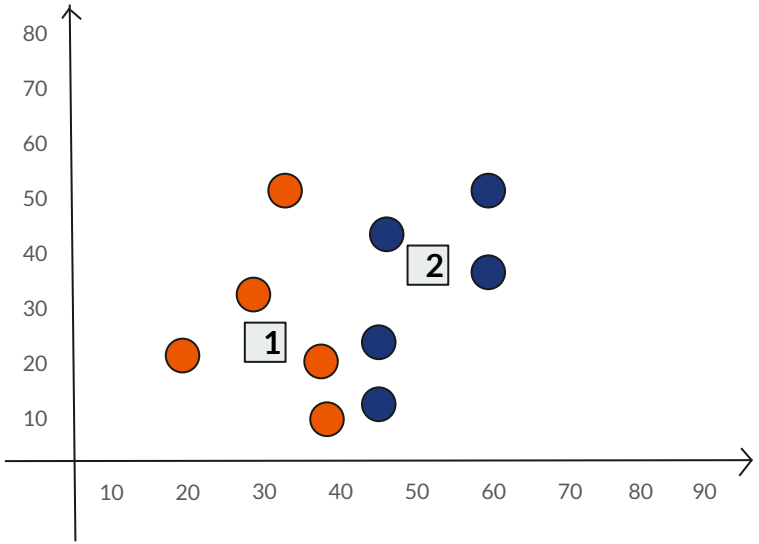
Assign each sample to their closest **centroid** using euclidean distance.

Repeat steps 2-3 until new **cluster assignments stop.**

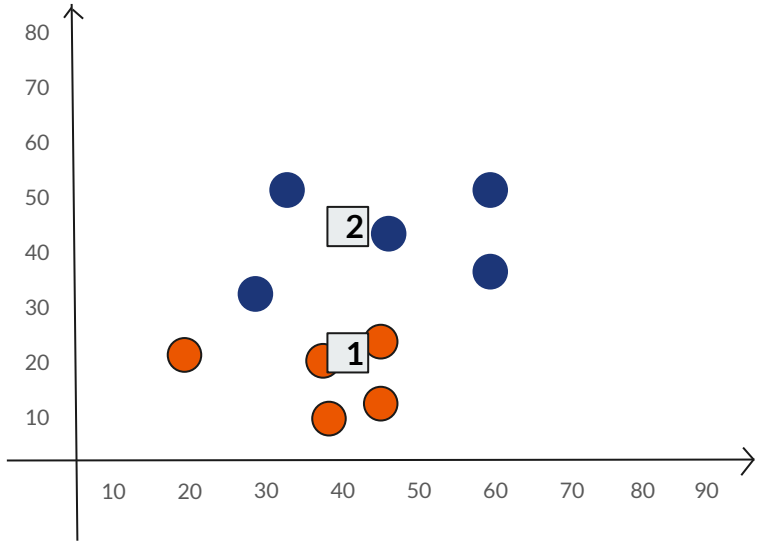
Now that we've reassigned our dataset, we repeat this process until new cluster assignments stop occurring. (that is we found the densest clusters available in this dataset)

It's basically kNN without the labels.

Universe 1

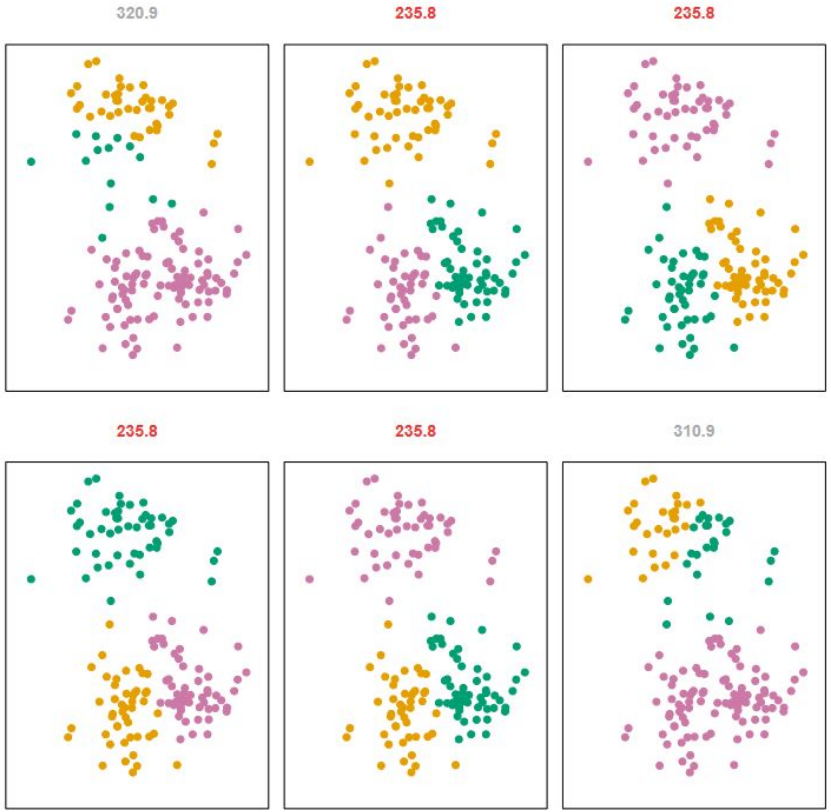


Universe 2

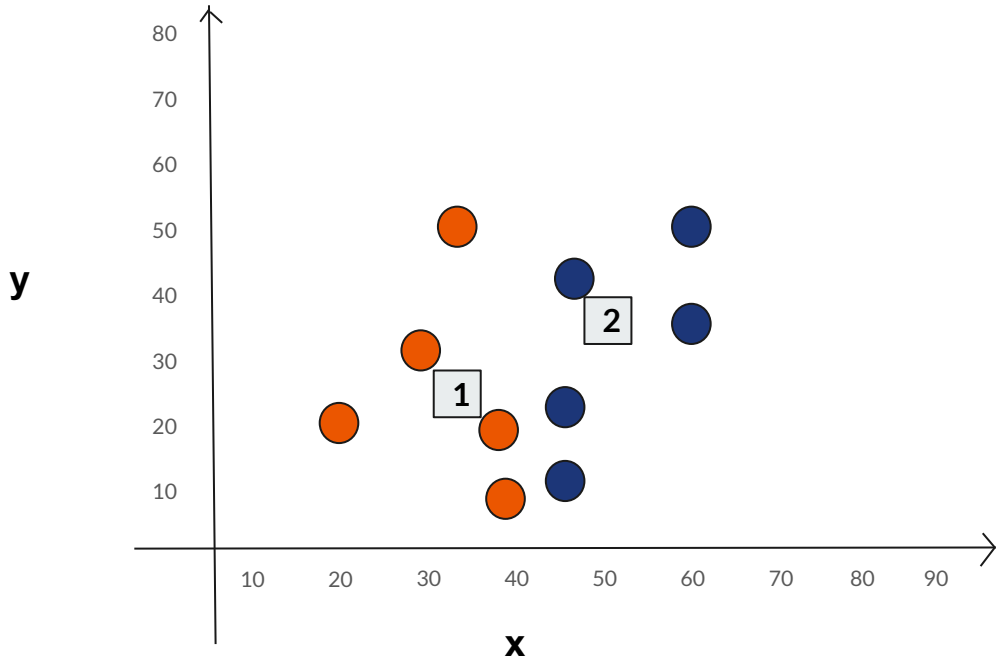


One issue however is that the clusters we find are **local optimums (not global).**

The random assignment that we performed at the very beginning of this process determines the clusters we find. How do we make sure we didn't just "settle" on a local optimum?



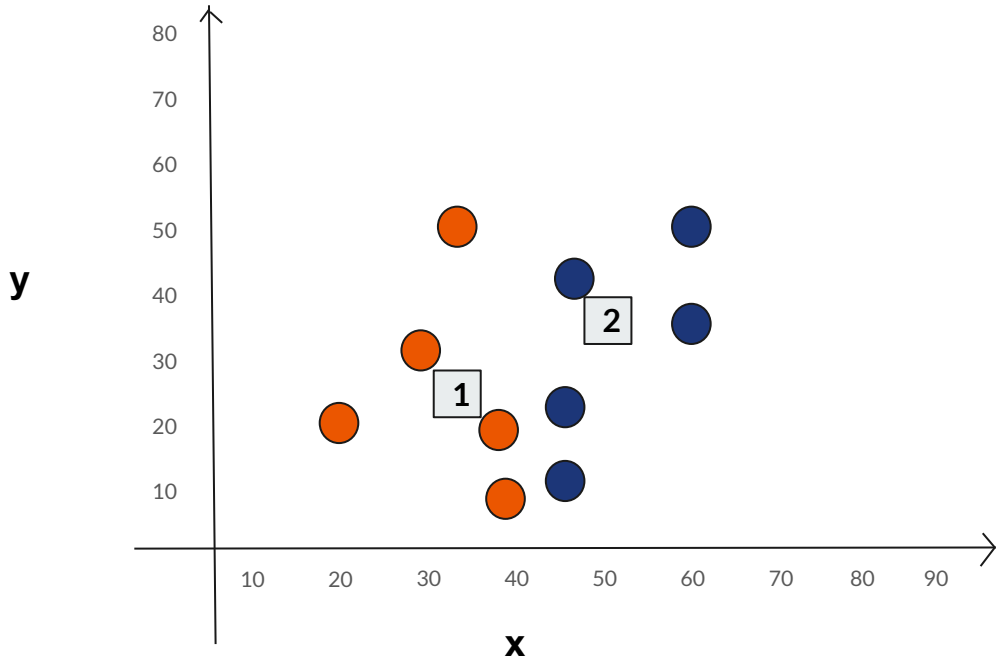
Think of a shoe-store analogy: try multiple times until you find the random assignment that results in the **minimum objective function (minimum variance)**.



What are some next questions you have for this technique (related to its hyperparameters & other classical issues we've seen already)

K-Means clustering is an excellent technique to cluster data when labels are not available.

But we're not done yet.



1) How can we find the ***best* number of clusters?** We know we originally had 3 clusters, but 2 seems to work here as well…
   a) I.e. what is the measure of error?

2) Is it **important to standardize our features as well?**

Not only do we have to concern ourselves with the classical problems of the kNN algorithm, **the unlabeled samples add another layer of ambiguity.** Any ideas on how to answer these two questions?

Classic Issues in Unsupervised Clustering

How do I find optimal clusters?

There's no quick & easy answer to this. There are 30 published & proposed methods to determine best clusters. We will explore the **elbow method, average silhouette,** and the **gap statistic**.

Do I need to do dimensionality reduction/feature standardization?

The answer to this is similar to all sage advice, "*the answer depends on what context you're operating in.*"

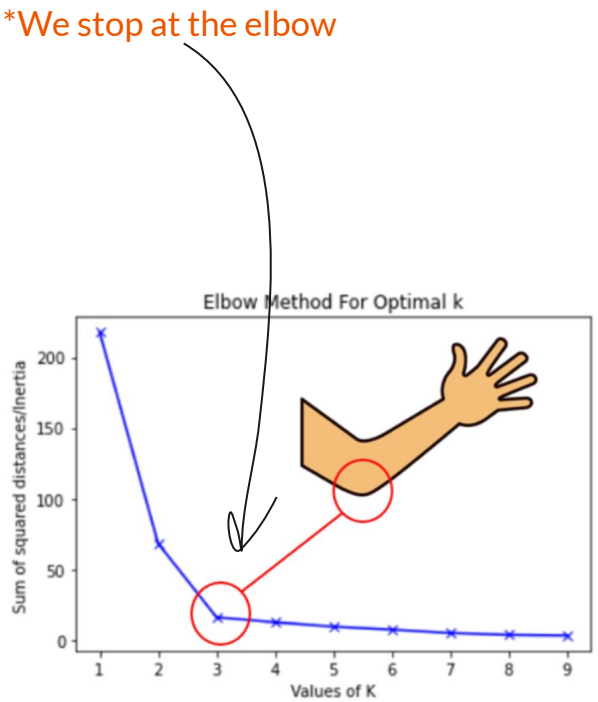
Elbow Method

Recall how we “trained” optimal clusters at some fixed ***K***.

We measured **within-cluster-sum-of-squares** (WSS) to measure variance.

Using cross-validation, we can find the best possible number of clusters by iteratively increasing our ***K*** until we no longer see rapid decrease in WSS.

When recognizing this visually, we call this the “**elbow method**.”



Average Silhouette

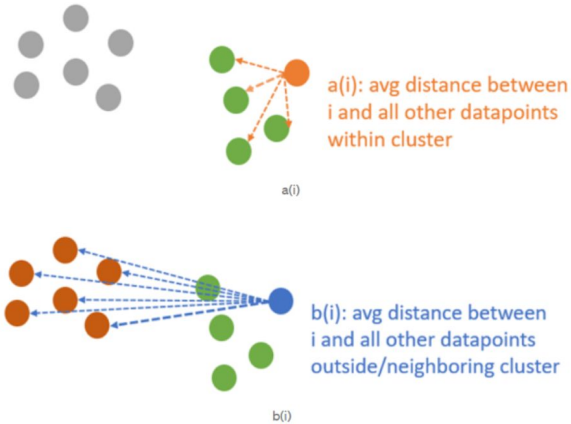
$$S(i) = \frac{b(i) - a(i)}{\max\{a(i), b(i)\}}$$

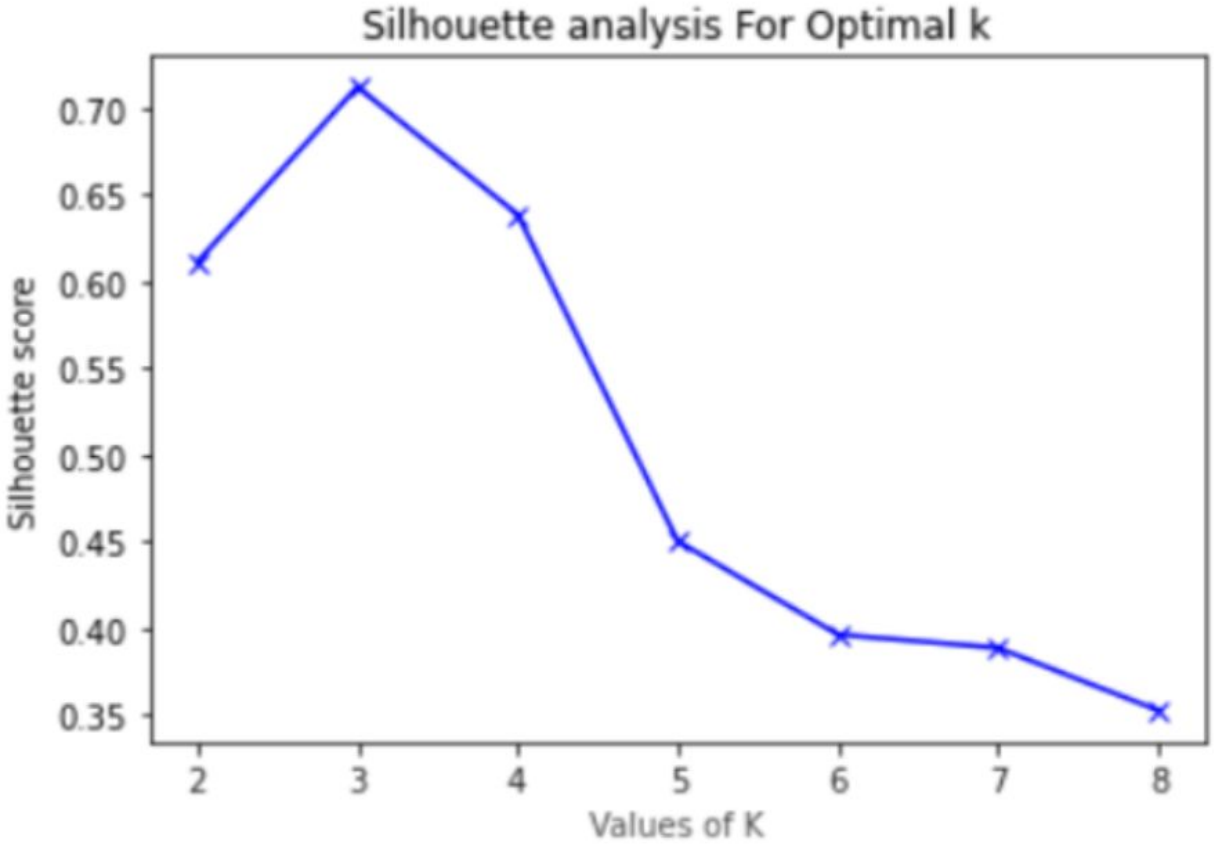
This is similar to the elbow method, where we iterate on a number of clusters.

Except this time instead of calculating WSS, we calculate the **average silhouette coefficient**.

This calculates the **ratio** of average distance **between samples that belong to the same cluster,** and samples that **don't belong to the same cluster.**

This value ranges from 1 to -1. A value of 1 denotes that **clusters are compact**, whereas a value of -1 indicates otherwise.



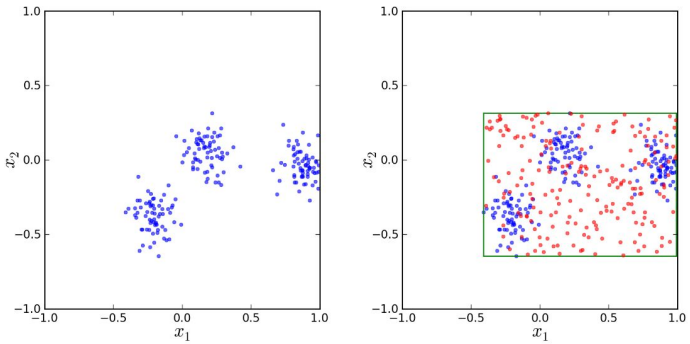


We select the K **where the silhouette score is the largest.**

Gap Statistic

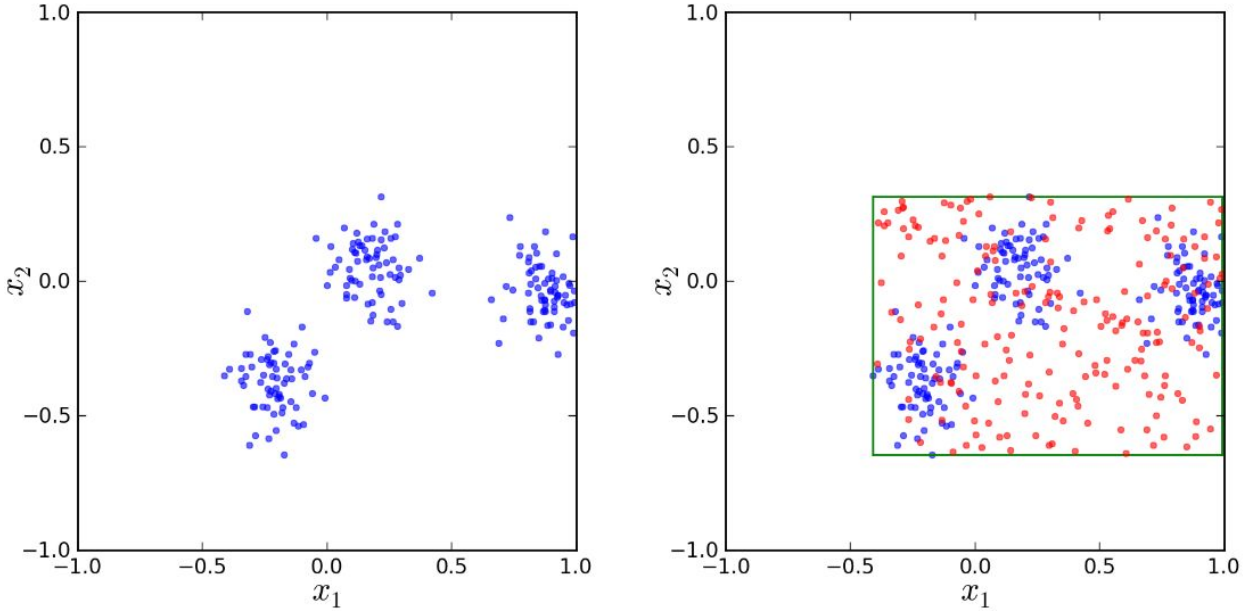
As its name suggests, this measure compares the “gap” between **the actual WSS** & **the expected WSS** *under the null reference distribution of data*.

Remember, **concepts in statistics are re-used time and time again**. What do we mean by the null distribution of data? Feel free to guess.



Think back to:

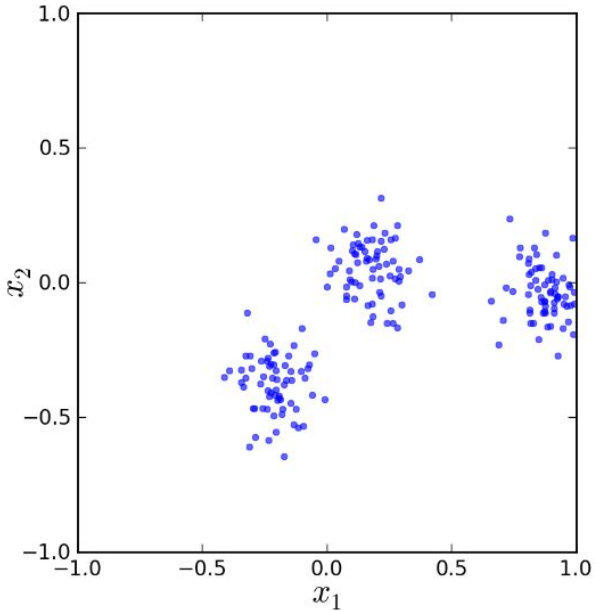
- ***Null Hypothesis***
- ***Null Accuracy***

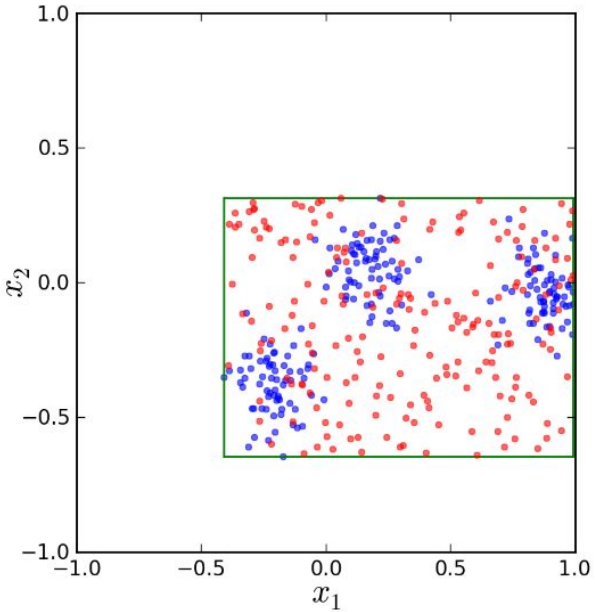


Whenever we discuss the **null** of some dataset, experiment, etc, we are always talking about the **uninteresting case**. The null is always our assumption that what we are searching for **does not exist.**

Null hypothesis: There is no effect
Null accuracy: The model captured no relationships
Null distribution: Clusters do not exist

Original dataset



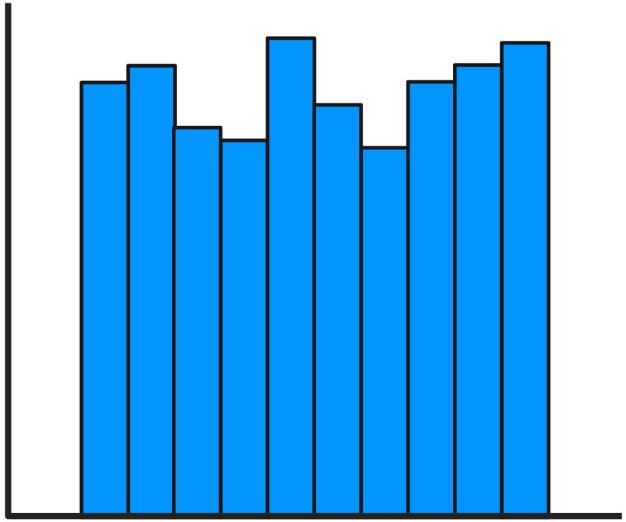


Simulated dataset

In the case of null distribution, we simulate a dataset similar to our original, but now we have a **equal probability of selecting a sample from the original range.** (i.e. we select values from a **uniform distribution**!)

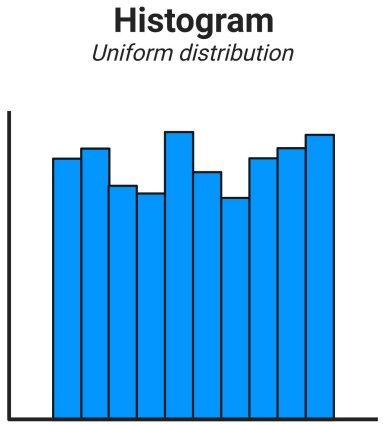
Histogram

Uniform distribution

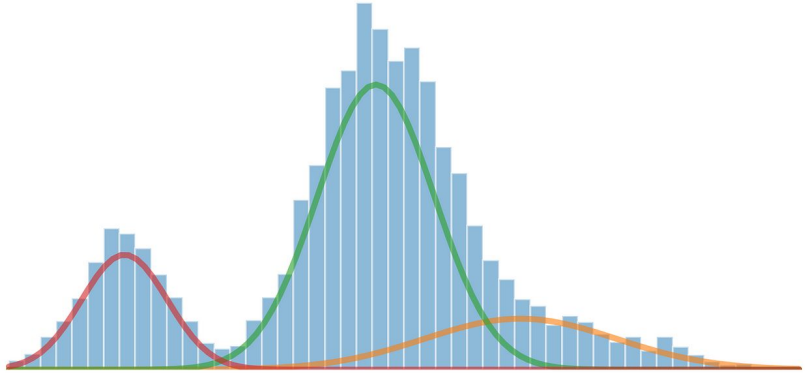


What do you think, is a **uniform distribution good** for predictability???

This is what **unpredictability** looks like



This is **beautiful predictability.**



Absolutely not

Remember, uniformity is good for gambling and games of chance because there is a degree of unpredictability.

We do not want to try to model chance, we want our clustering algorithm to **model structure.**

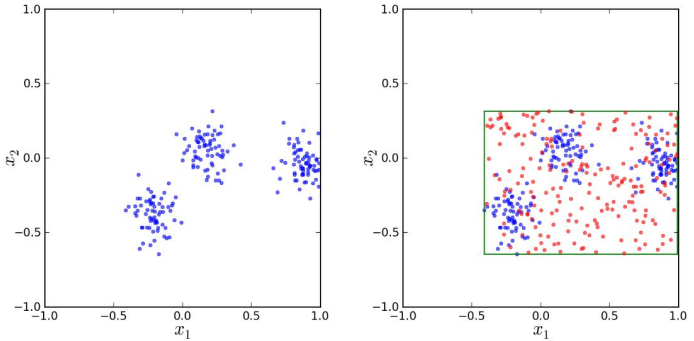
Our goal for the gap statistic is to find the **K that produces the largest difference from our null distribution.**

Gap Statistic

Using these two datasets, we calculate the **log of average variance across both datasets** and find the difference between these two values.

What do you think a **large difference indicates** for a certain K # of clusters?

What about a s**mall difference**?



$$Gap_n(k) = E_n^* log(W_k) - log(W_k)$$

*Whenever you see us take the log of a value, this is almost always for ease of interpretation

Gap Statistic

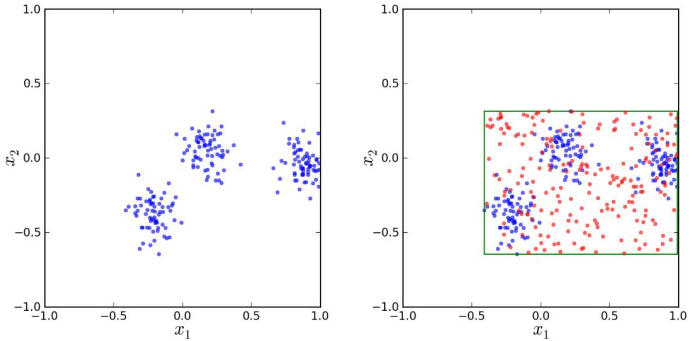
What do you think a **large difference indicates** for a certain K # of clusters?

> Clusters are **less likely** to come from a uniform distribution.

What about a s**mall difference**?

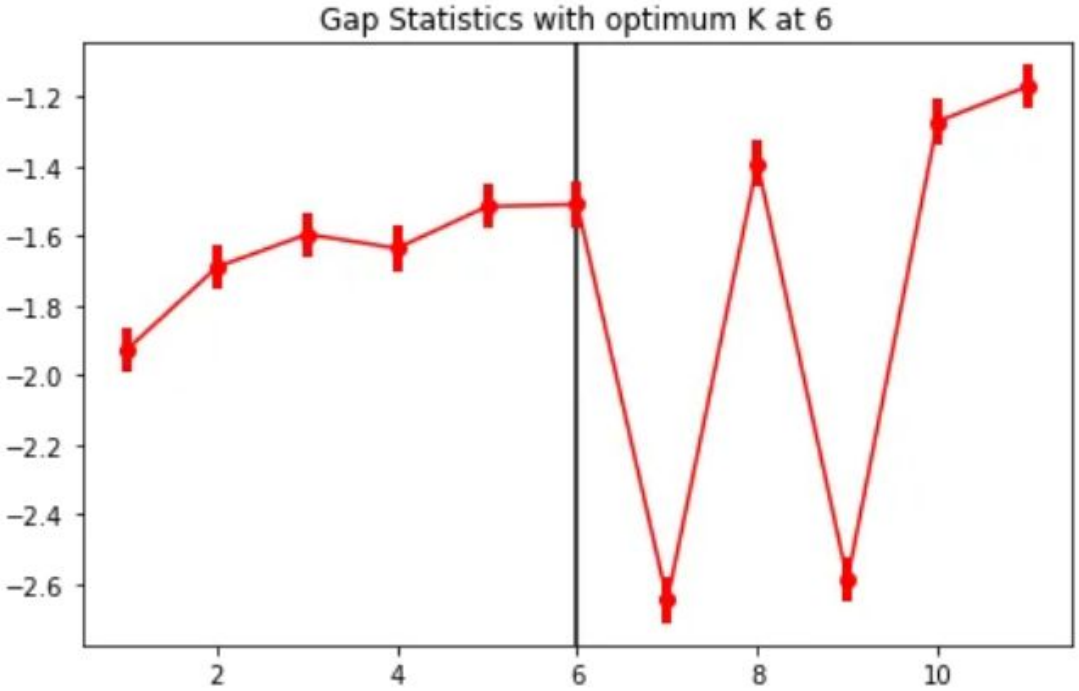
> Clusters are **more likely** to come from a uniform distribution.

Our goal is to find the **smallest k** which contains the **largest difference (gap statistic)**.

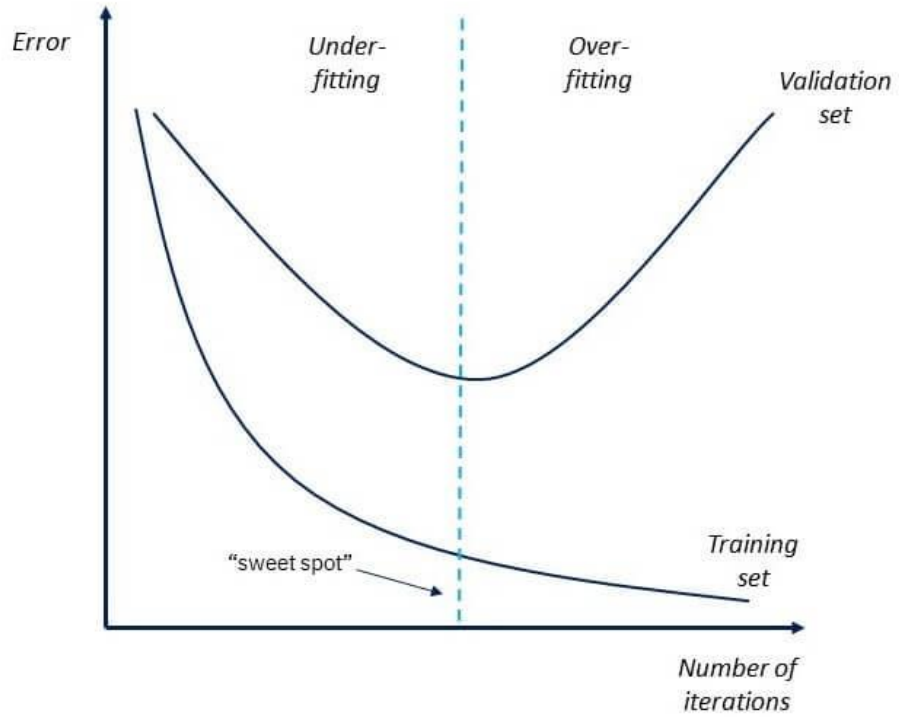


$$Gap_n(k) = E_n^* log(W_k) - log(W_k)$$

*Whenever you see us take the log of a value, this is almost always for ease of interpretation



What is the smallest **k** that experiences the largest gap? A better question, why don't we want to just select the largest **possible k** that maximizes the difference?



Even though we are working in the domain of unsupervised learning, concerns of overfitting still apply. Generally speaking, you want to be **as stingy as possible with your features, dimensions, hyperparameters.**

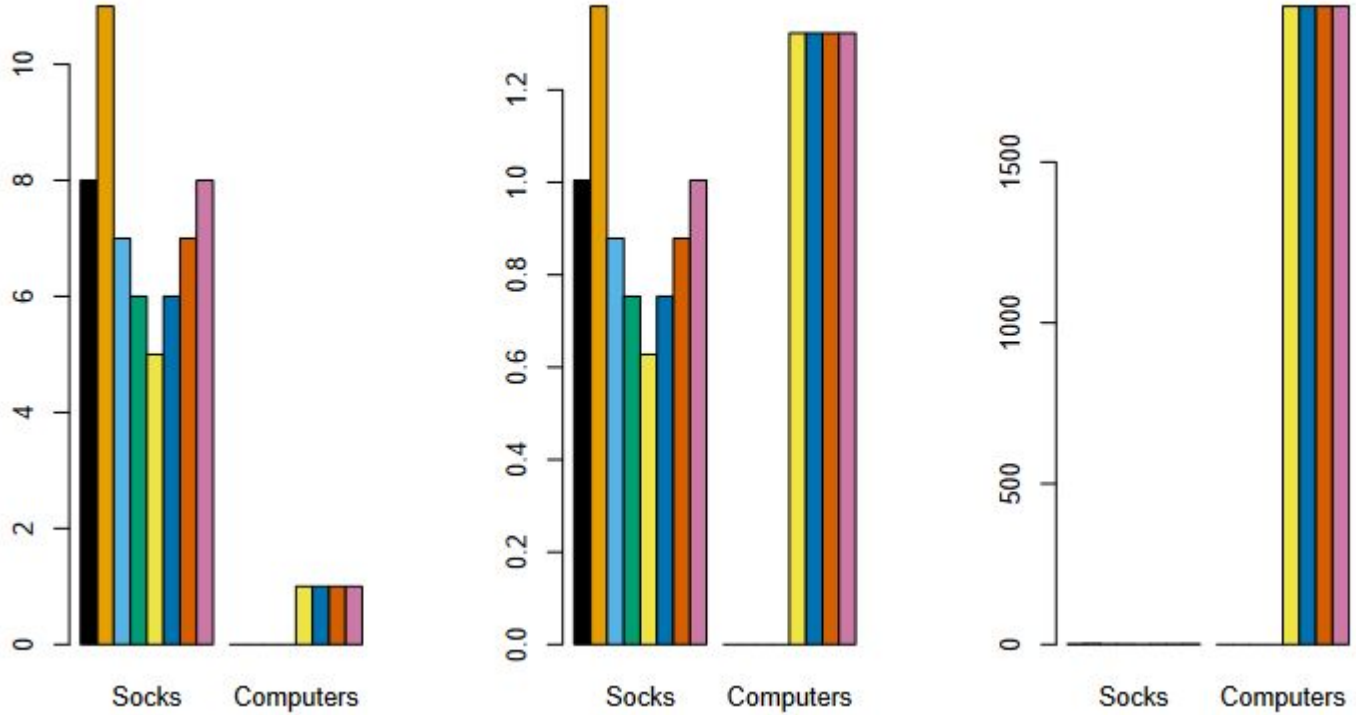
Standardization/Dim Reduction

In **supervised learning** we told you:

> *"**Always** standardize your data (for most supervised learning algorithms)"*

However in **unsupervised learning** we will tell you instead:

> *"Standardization is dependent on which insights you'd like to capture, so be intentional about standardizing your dataset. Consider the question: **should all columns have equal weights?"***



We are interested in discovering customer insights on a retailer that only **sells socks and computers**. Which column do you think is **most important** when **predicting customer behavior**? Do we standardize and assign equal weights to socks & computers?

K-Means

To conclude our conversation on the **unsupervised** learning classifier K-Means, it is a powerful non-parametric learning algorithm that measures the spread of clusters to determine labels.

Pros

- Always **converges** to optimal clusters
- **Scales well** to large datasets

Cons

- Often **captures noise**
- Must choose **k manually**

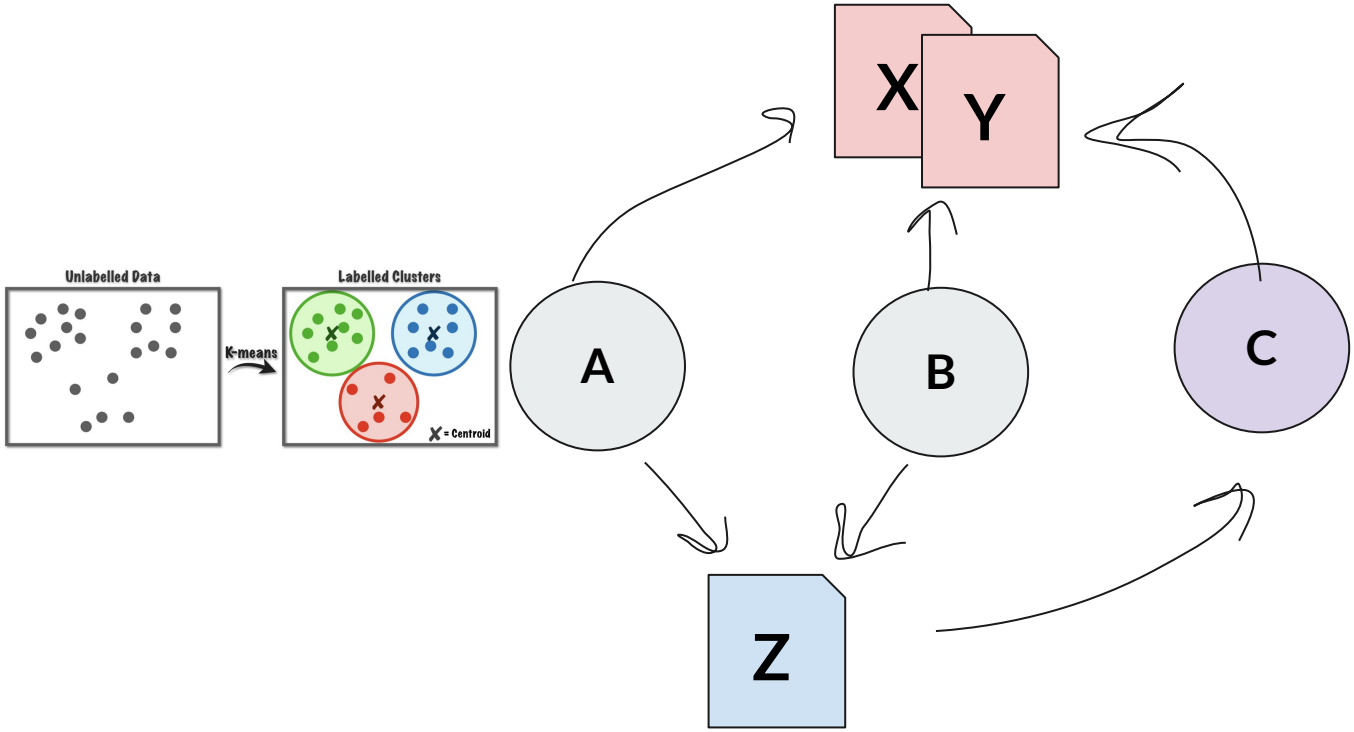
*Keep in mind that **k-means would only be part of this system**, not the entire system itself!

K-Means Application

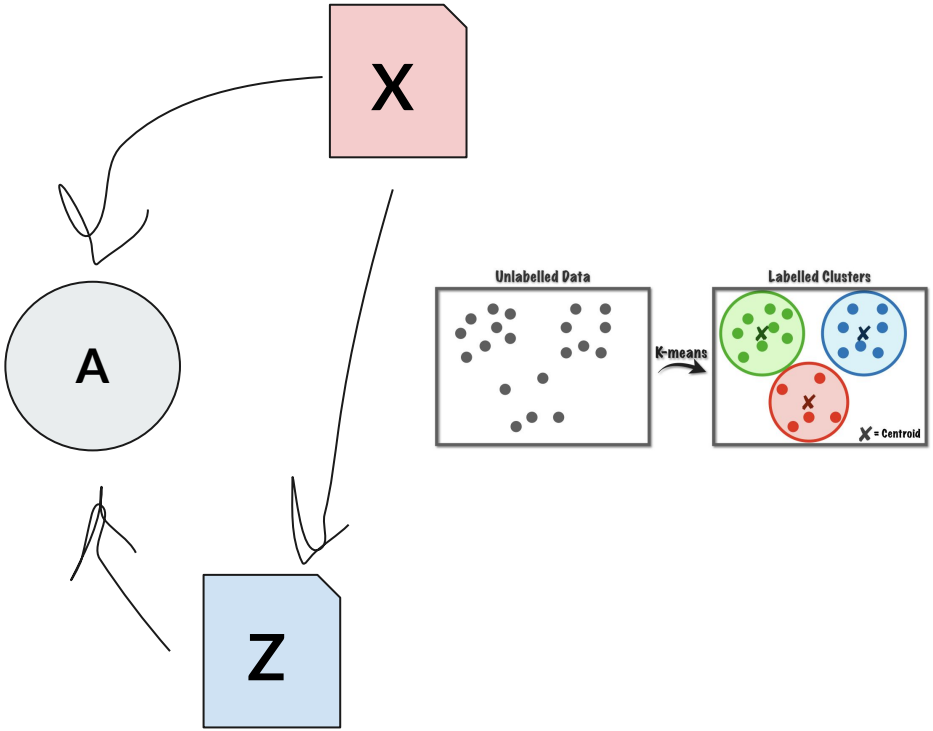
When implementing **unsupervised clustering algorithms** to implement recommendation-based systems, there are two main types of recommenders we will see:

Collaborative Filtering : Recommend a new piece of content based on consumed content **from similar users.**

Content-Based Filtering: Recommend a new piece of content based on **similarity with previously consumed content.**



Collaborative Filtering. Cluster **users** based on similarity in content they **consume, rate, buy**. Recommend next piece of media based on user-similarities.



Content Based Filtering. Cluster **content** based on similarity in subject-material, and content features. Recommend next piece of content based on content-similarities.

End of Class Announcements

Lab (Due 7/23)

Zurich, Switzerland

You are a data scientist working for a Zurich-based international bank called Caishen. The company announced in an all-hands meeting that they are aiming to develop a classification algorithm **that can identify 99% of all fraudulent activity within customer-facing bank accounts.**

For this project, you will use a **dataset of 1 million bank transactions to create a classifier** that will detect if fraudulent activity has occurred for a transaction.

Next Week

Introduction to Neural Networks

- Dimensionality Reduction
- Neural Networks
- Deep Learning