Datascience fundamentals

Fernando Lovera

Mario Verstraeten

Lecture 7 & 8:

Classification 2: kNN

&

Clustering 1: Kmeans

k-Nearest-Neighbours

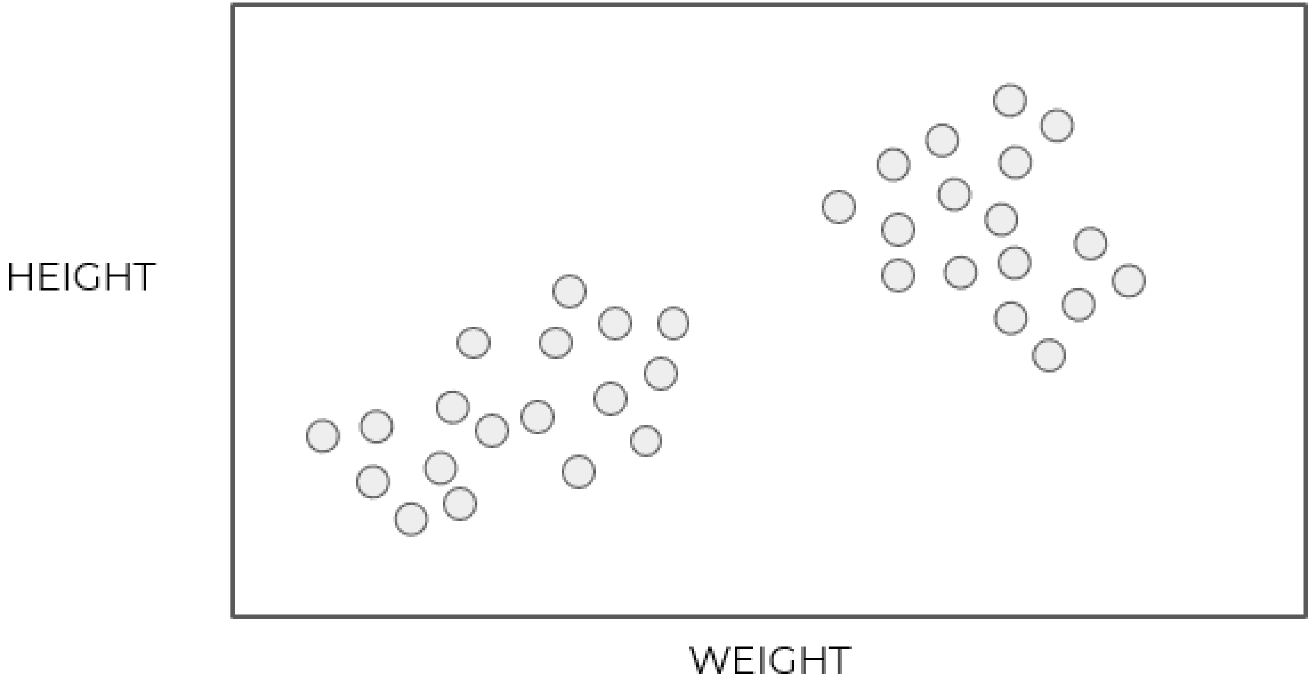
kNN

- While KNN can be used for regression tasks, its performance can be quite poor **and** less efficient than other algorithms, so we've decided not to exhibit its use for regression.
- However if you do want to use it for regression it is very easy to swap in the **KNNRegressor** model with scikit-learn.

- K nearest neighbors is one of the simplest machine learning algorithms.
- It simply assigns a label to new data based on the **distance** between the old data and new data.

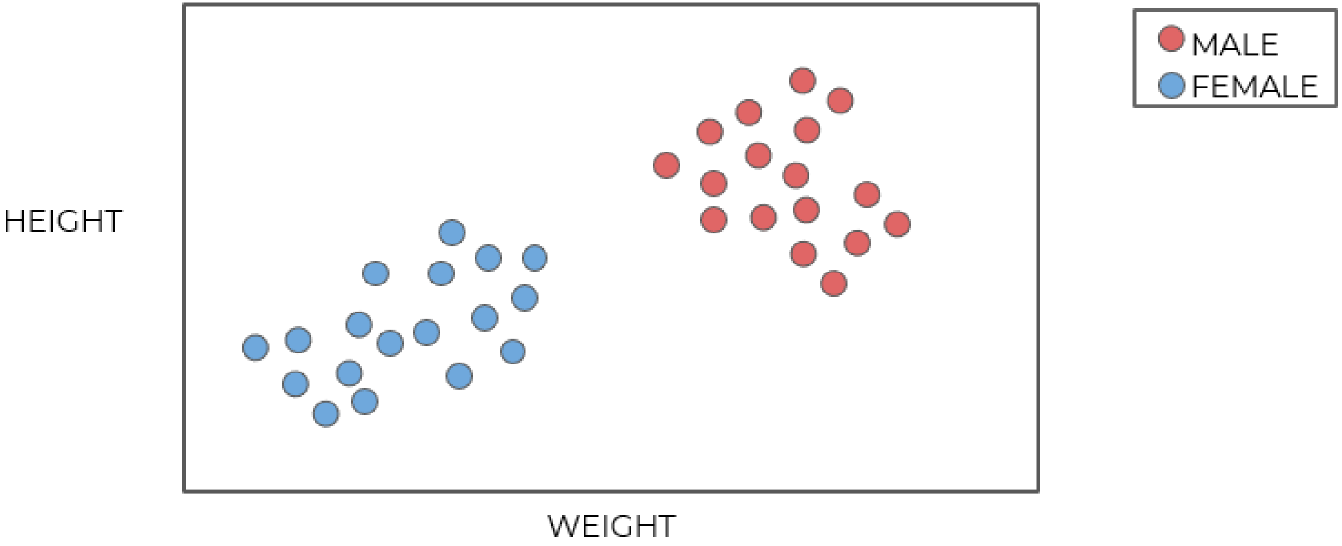
kNN

- Imagine a height and weight data set



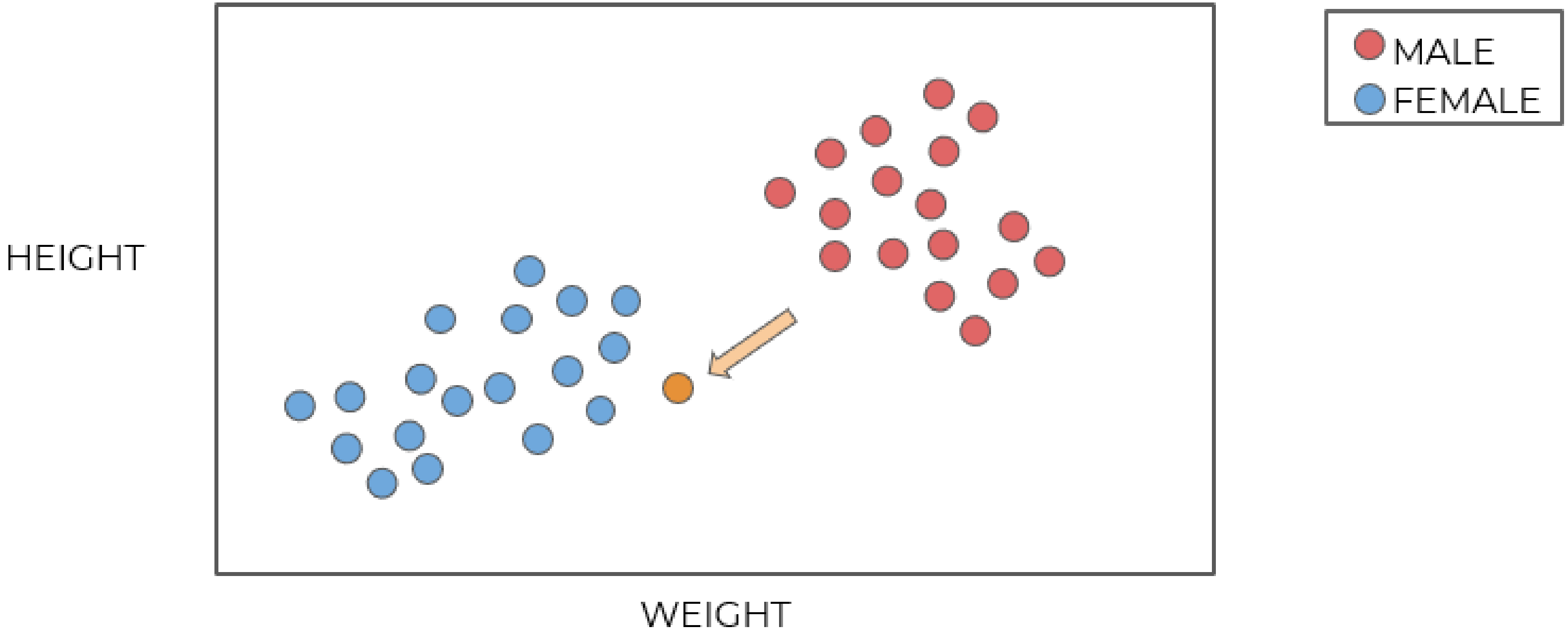
kNN

- We historically know the sex of the chicks:



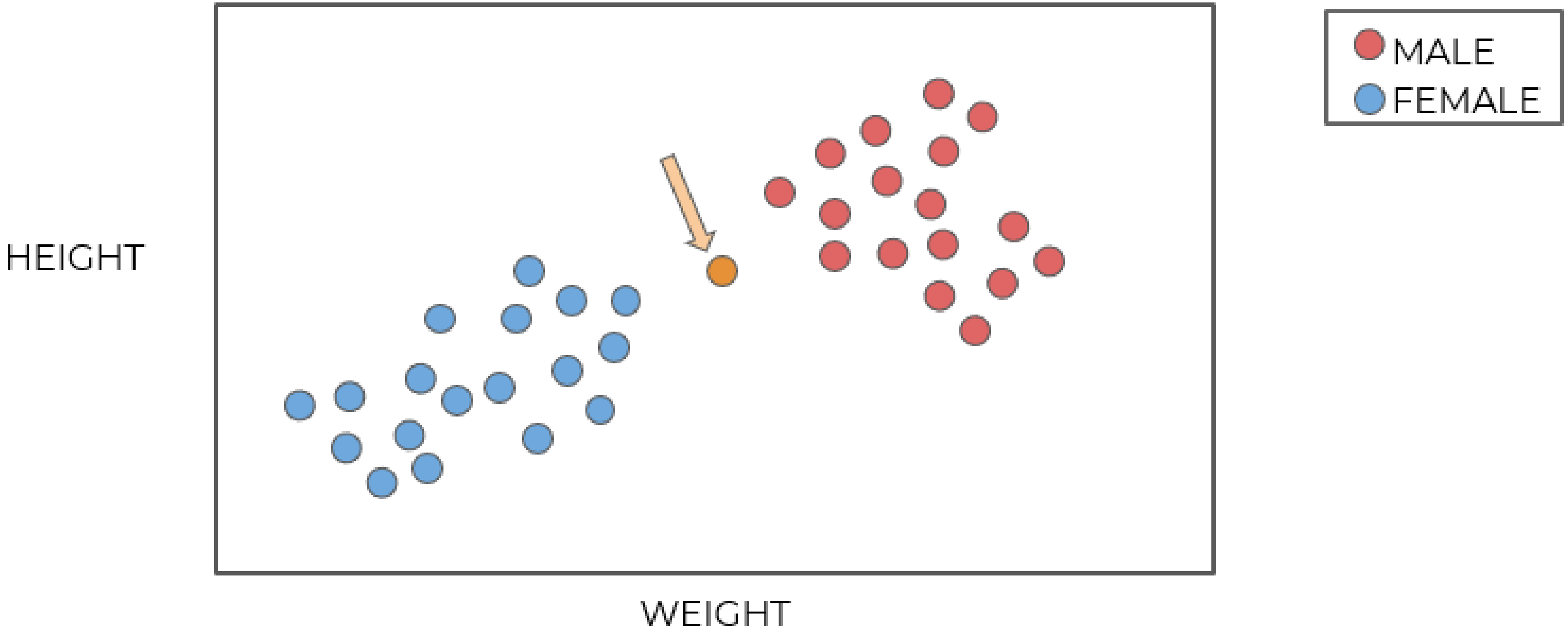
kNN

- How would we assign sex to a new point?
  - We intuitively “know” this is likely female.



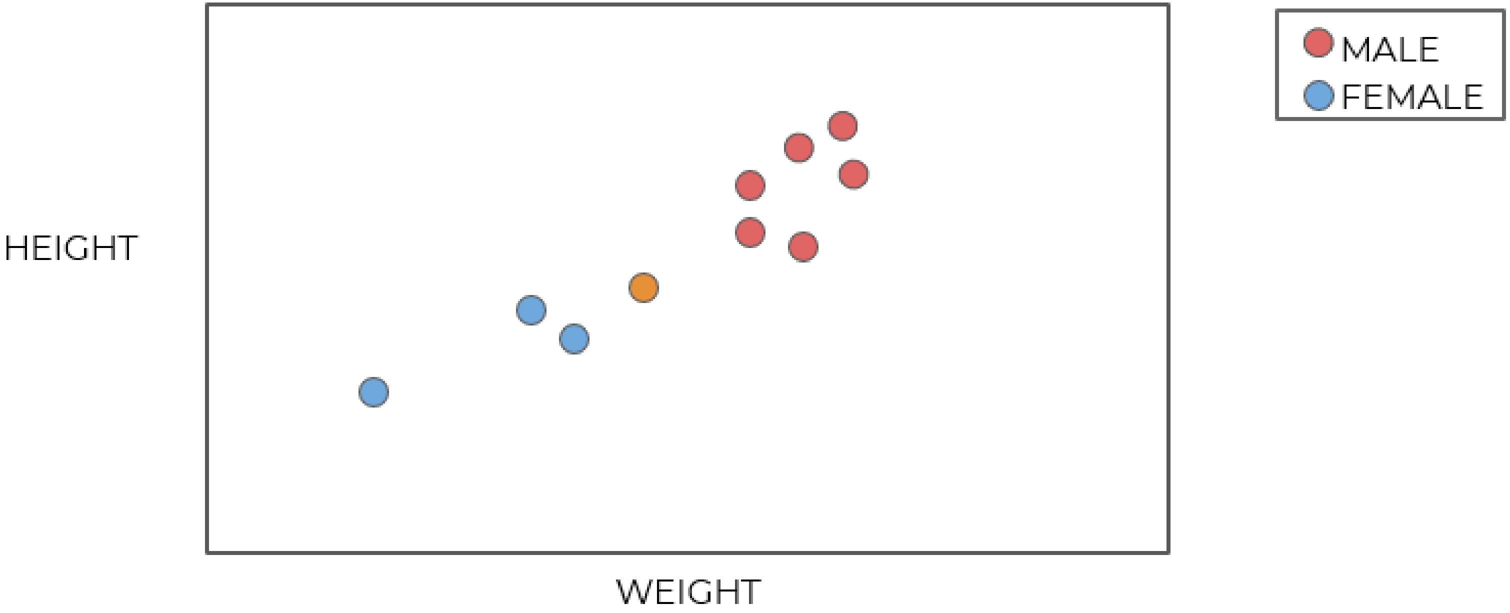
kNN

- What about a less obvious point?
  - How many points to we consider?



kNN

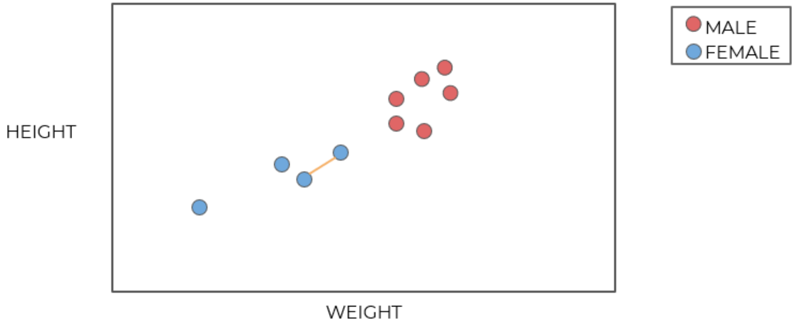
- Let's imagine a situation like this:



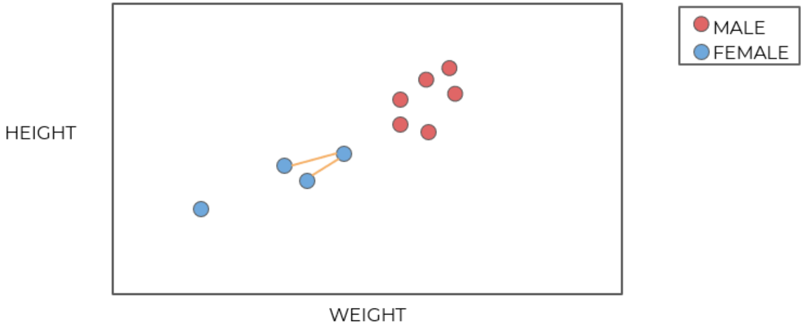
kNN

- Let’s imagine a situation like this:

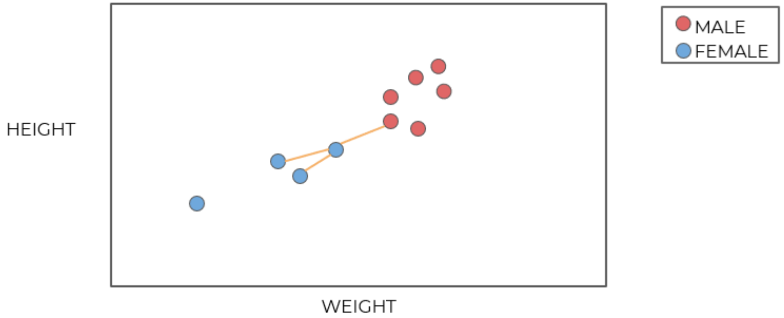
k = 1



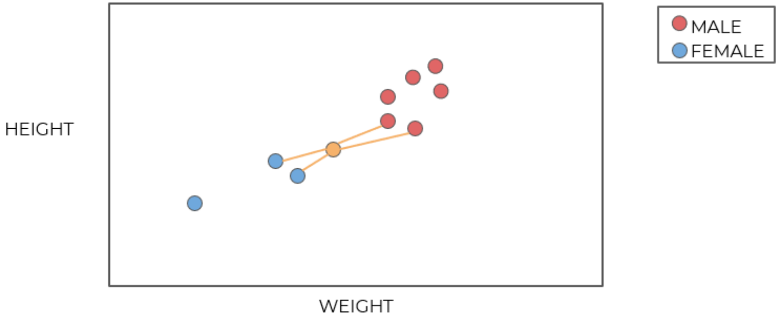
k = 2



k = 3



k = 4

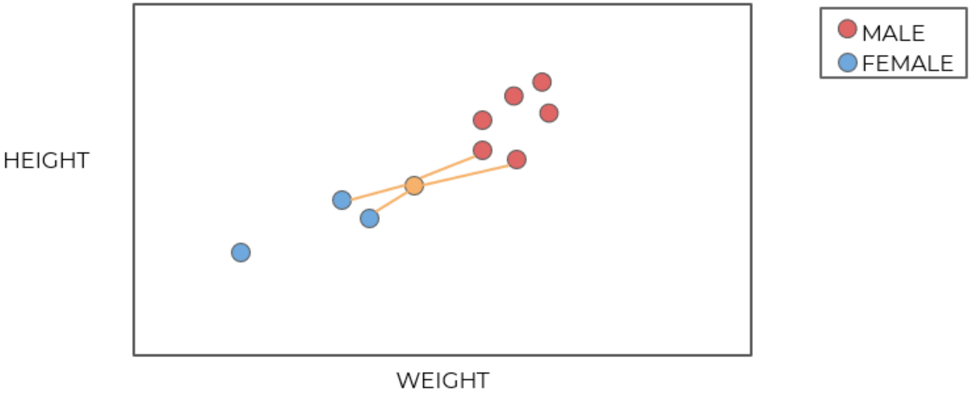


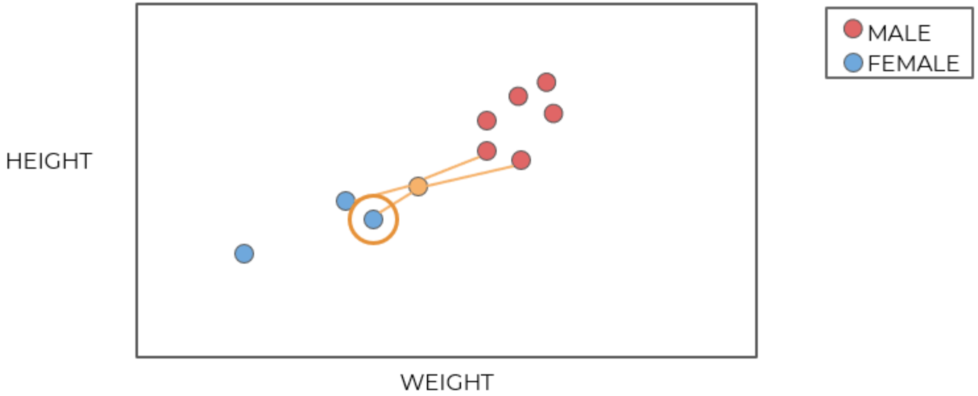
kNN

- Tie considerations and options:
  - Always choose an odd K.
  - In case of tie,simply reduce K by 1 until tie is broken.
  - Randomly break tie.
  - Choose nearest class point.

kNN

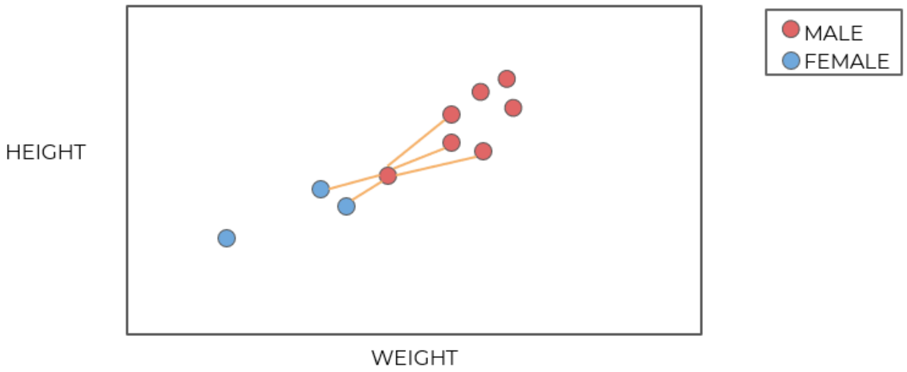
- K=4 leads to a tie!
    - Choose closest K





kNN

- K=5 causes a switch from previous K values.



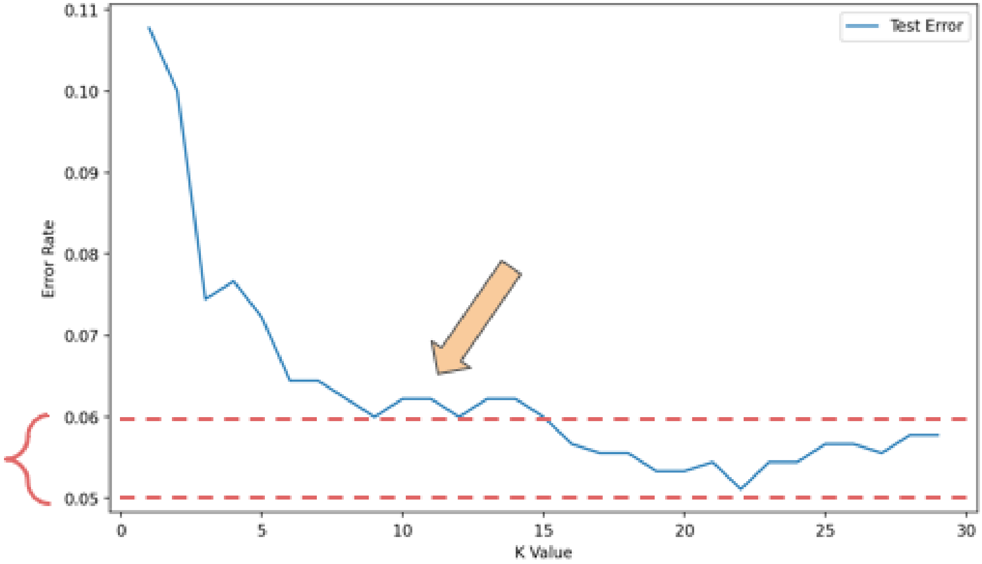
kNN

- How to choose best K value?

- We want a K value that **minimizes** error:
  - Error = 1 - Accuracy
- Two methods:
  - Elbow method.
  - Cross validate a grid search of multiple K values and choose K that results in lowest error or highest accuracy.

kNN

- Elbow method:



kNN

- Cross validation only takes into account the K value with the lowest error rate across multiple folds.
- This could result in a more complex model (higher value of K).
- Consider the context of the problem to decide if larger K values are an issue.

- KNN Algorithm
  - Choose K value.
  - Sort feature vectors (N dimensional space) by distance metric.
  - Choose class based on K nearest feature vectors.

kNN

- KNN Considerations:
  - Distance Metric
  - Many ways to measure distance:
    - Minkowski
    - Euclidean
    - Manhattan
    - Chebyshev

sklearn.metrics.pairwise.distance_metrics

sklearn.metrics.pairwise.**distance_metrics**() [source]

Valid metrics for pairwise_distances.

This function simply returns the valid pairwise distance metrics. It exists to allow for a description of the mapping for each of the valid strings.

The valid distance metrics, and the function they map to, are:

metric	Function
'cityblock'	metrics.pairwise.manhattan_distances
'cosine'	metrics.pairwise.cosine_distances
'euclidean'	metrics.pairwise.euclidean_distances
'haversine'	metrics.pairwise.haversine_distances
'l1'	metrics.pairwise.manhattan_distances
'l2'	metrics.pairwise.euclidean_distances
'manhattan'	metrics.pairwise.manhattan_distances
'nan_euclidean'	metrics.pairwise.nan_euclidean_distances

sklearn.neighbors.KNeighborsClassifier

class sklearn.neighbors.**KNeighborsClassifier**(*n_neighbors=5, *, weights='uniform', algorithm='auto', leaf_size=30, p=2, metric='minkowski', metric_params=None, n_jobs=None*) [source]

Kmeans Clustering

Kmeans

- ***Important Note:***
  - *Do not confuse K-Means with KNN!*
  - *The “K” is completely different in both algorithms, they solve completely different problems and are not related in any way!*

Clustering

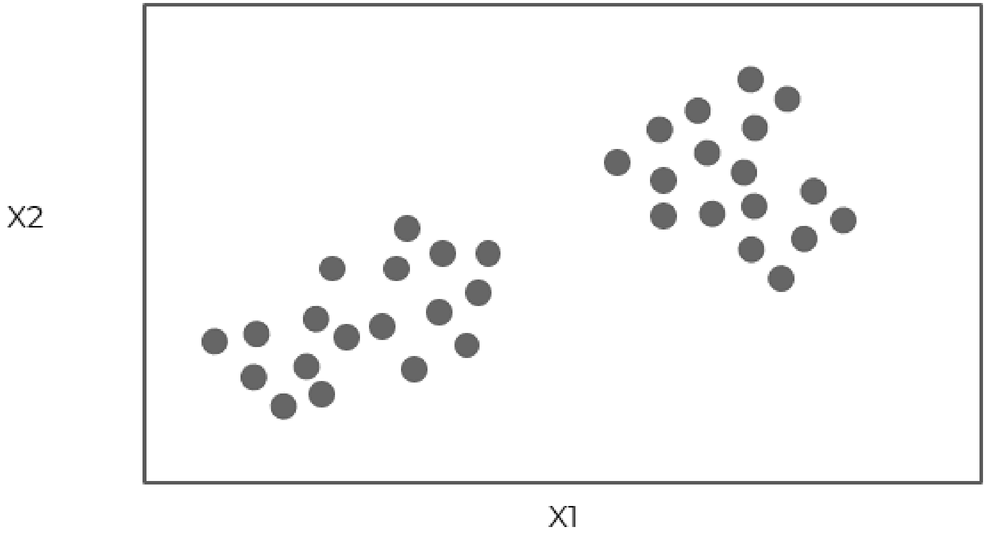
- Clustering uses **unlabeled data** and looks for similarities between groups (clusters) in order to attempt to segment the data into separate clusters.
- Keep in mind that we don't actually know the true correct label for this data!

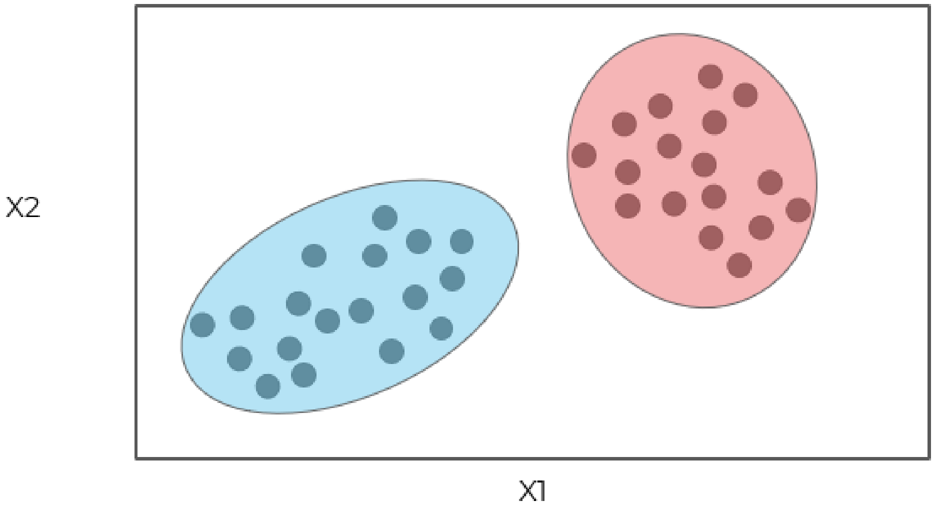
- Imagine an example data set:
  - Notice again we only have features!
  - How could we cluster this data together?

X1	X2
2	4
6	3
...	...
1	2

Clustering

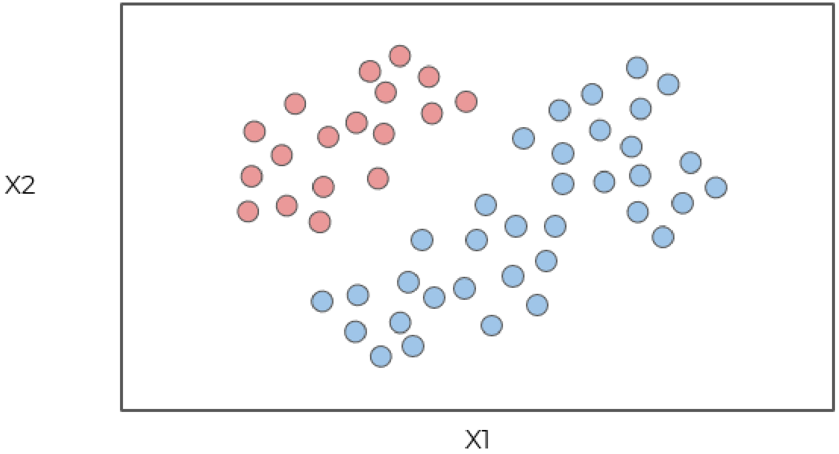
- Could simply plot and discover patterns:
  - Note how distance is the intuitive metric, which we can use to assign clusters

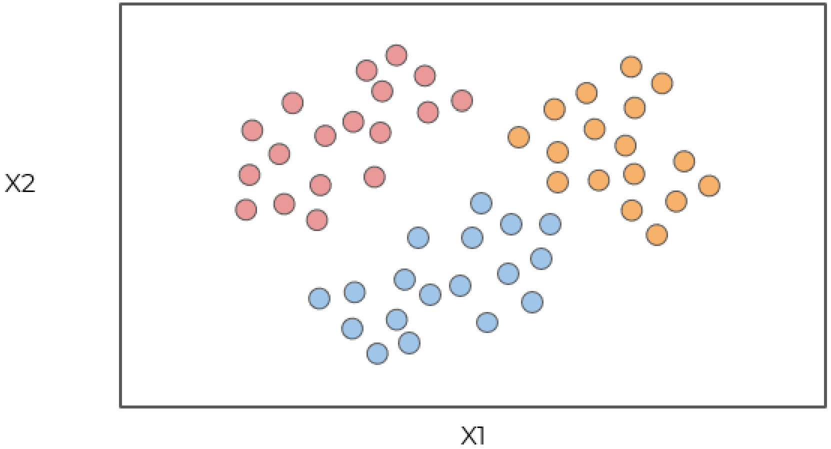




Clustering

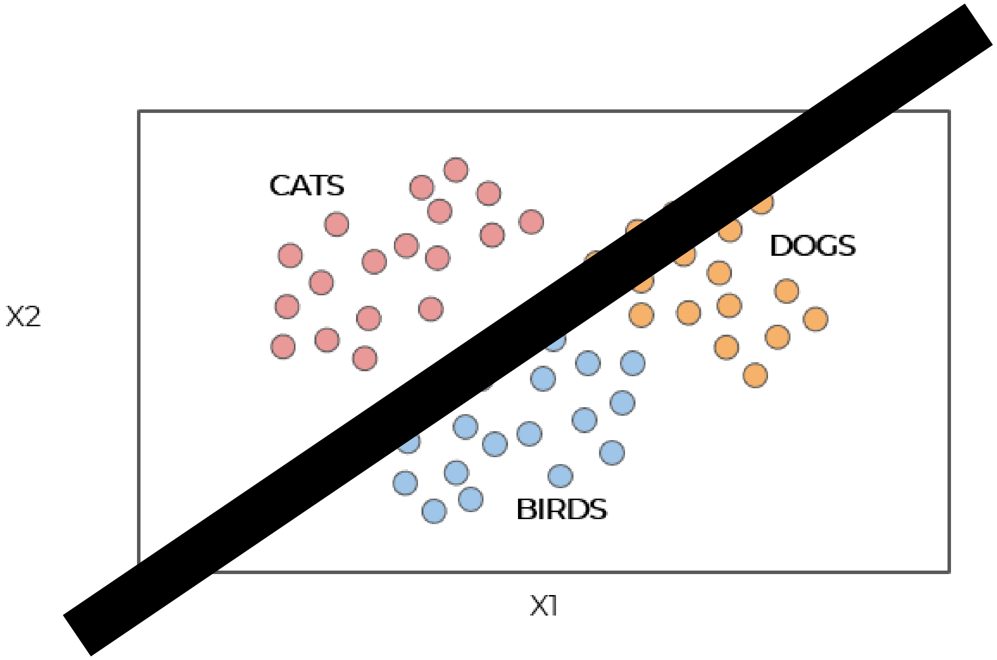
- Notice how we don't actually know for sure if this is a correct way of grouping together these data points, there was no correct label to begin with!
- And what about situations that are not so obvious or multi-dimensional?
- F.e. 2 or 3 clusters could both be reasonable in the following example:

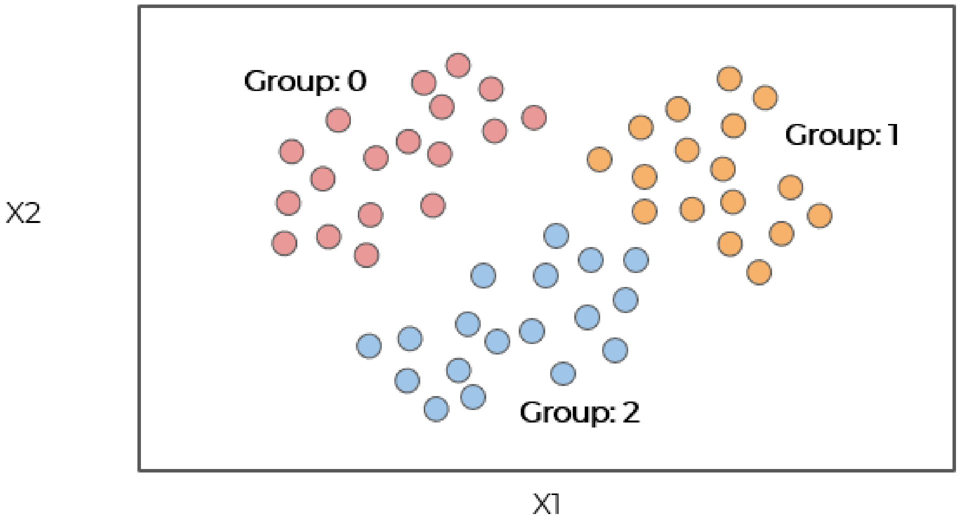




Clustering

- Clustering doesn’t “label” these for you!





Clustering

- Main Clustering Ideas:
  - Use features to decide which points are most similar to other points.
  - Realize that there is no final correct **y** label to compare cluster results to.
  - We can think of clustering as an unsupervised learning process that "discovers" potential labels.

- Unsupervised Learning Paradigm Shift:
  - *How do we assign a new data point to a cluster?*
    - Different approaches depending on the unsupervised learning algorithm used.
    - Use features to assign most appropriate cluster.

Clustering

- Unsupervised Learning Paradigm Shift:
    - *If we've discovered these new cluster labels, could we use that as a **y** for supervised training?*
        - Yes! We can use unsupervised learning to discover possible labels, then apply supervised learning on new data points.
        - BUT: Clustering doesn't tell you what these new cluster labels represent, no real way of knowing if these clusters are truly significant.

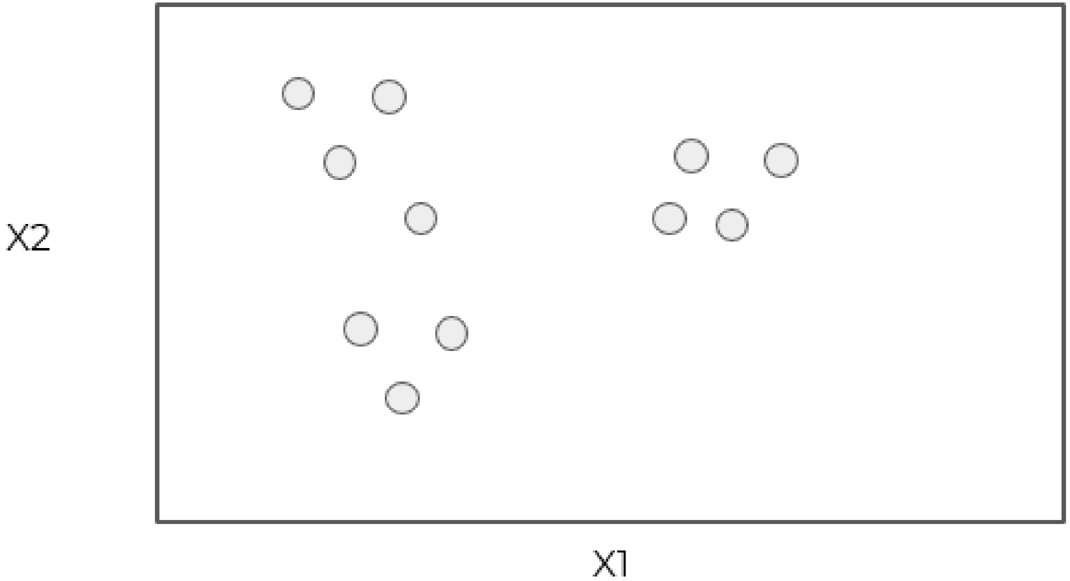
Kmeans Clustering

- So how does K-Means clustering actually work?
  - The main concept is actually very simple!
    - Let’s walk through an example of clustering unlabeled data.

- First a set of properties each point must satisfy:
  - Each point must belong to a cluster.
  - Each point can only belong to one cluster (no single point can belong to multiple clusters).

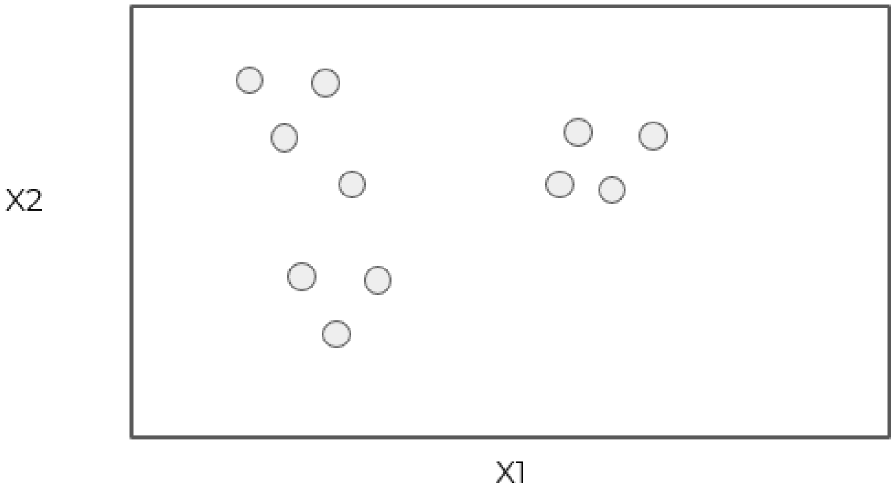
Kmeans Clustering

- Step 0: Start with unlabeled data (only features).



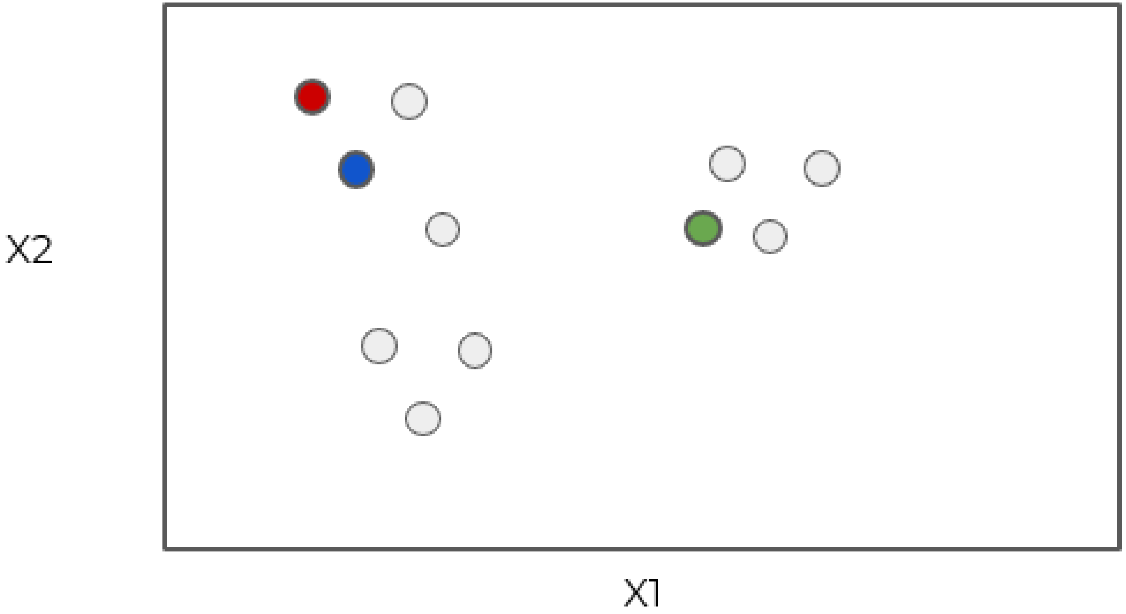
Kmeans Clustering

- Step 1: Choose the number of clusters to create (this is the K value).
    - We'll choose K=3. Note in most situations you won't visualize the data!



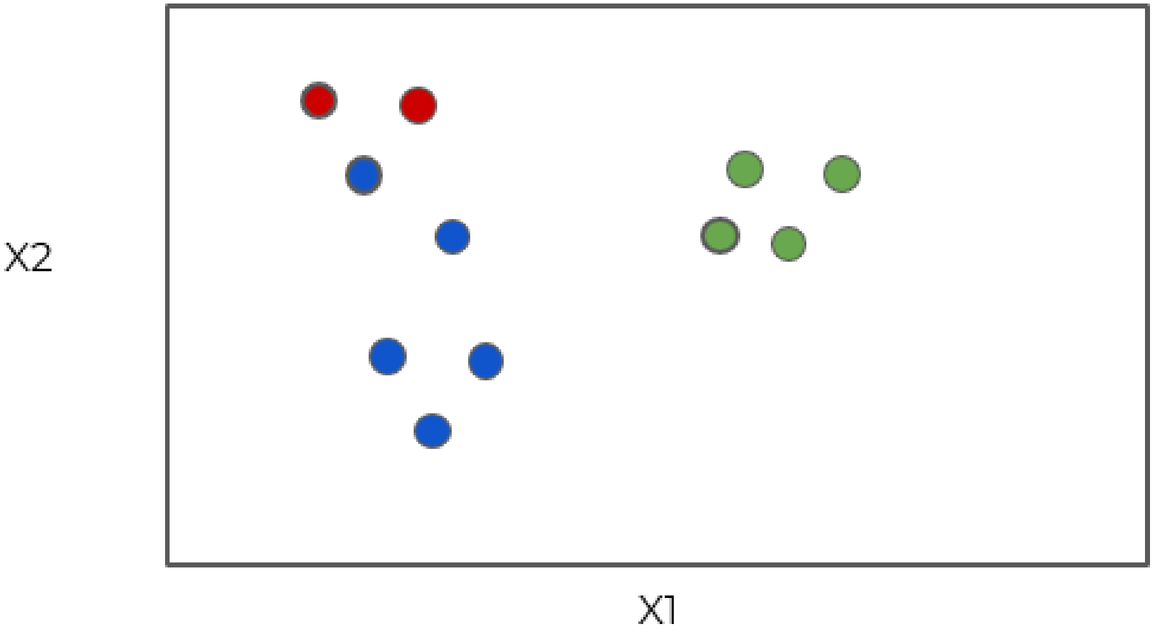
Kmeans Clustering

- Step 2: Randomly select K distinct data points. Our K=3:
  - We'll treat these new K points as our "cluster" points.



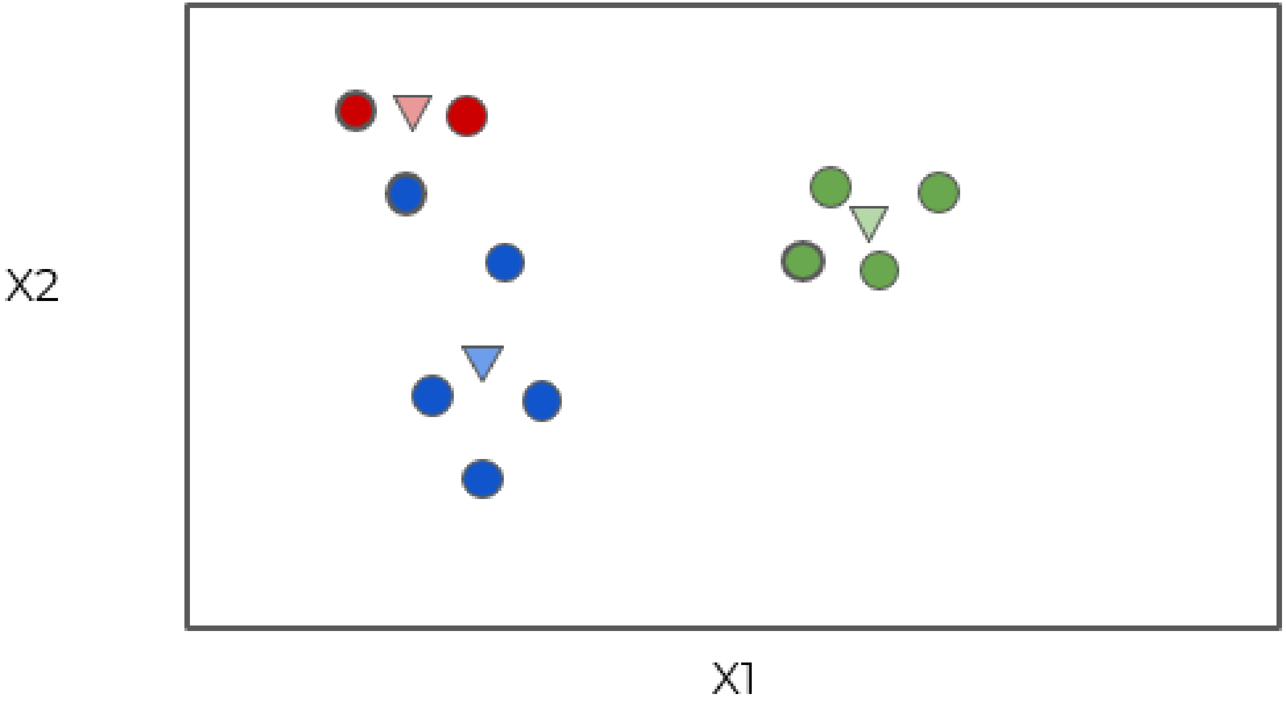
Kmeans Clustering

- Step 3: Assign each remaining point to the nearest “cluster” point.
  - Note how this is using a distance metric to judge the nearest point.



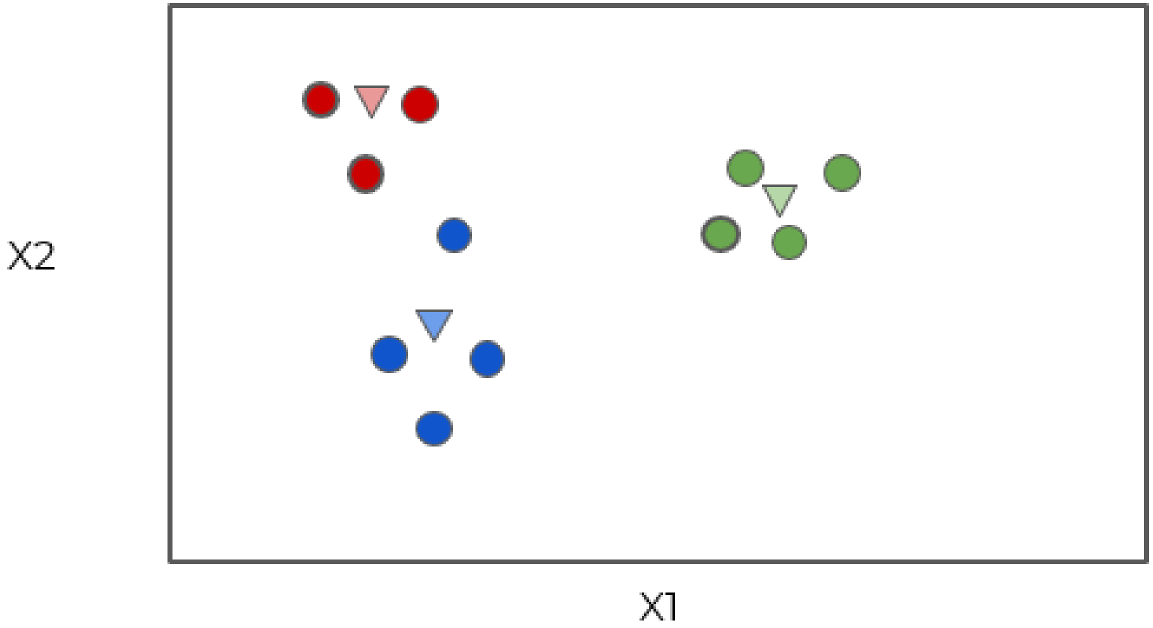
Kmeans Clustering

- Step 4: Calculate the center of the cluster points (mean value of each point vector).



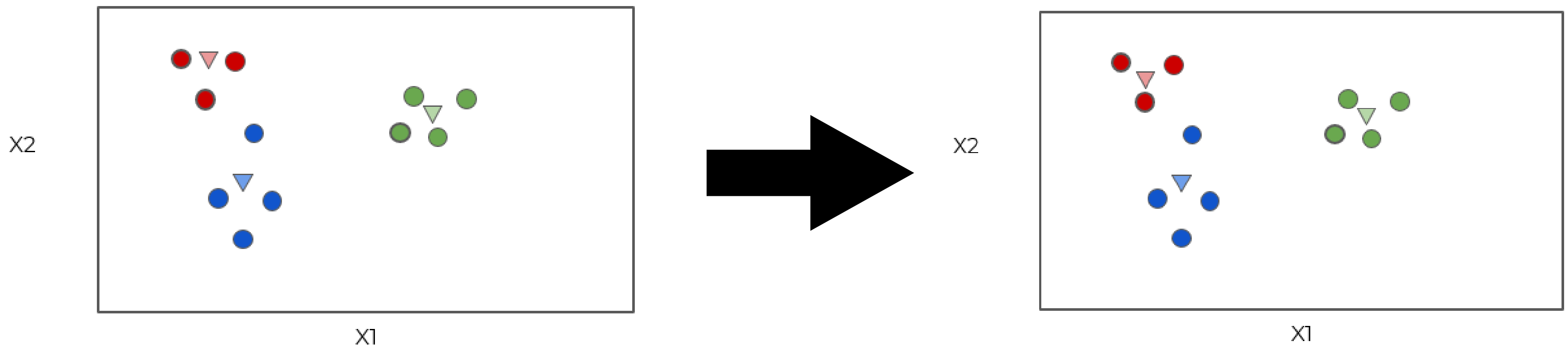
Kmeans Clustering

- Step 5: Now assign each point to the nearest cluster center.



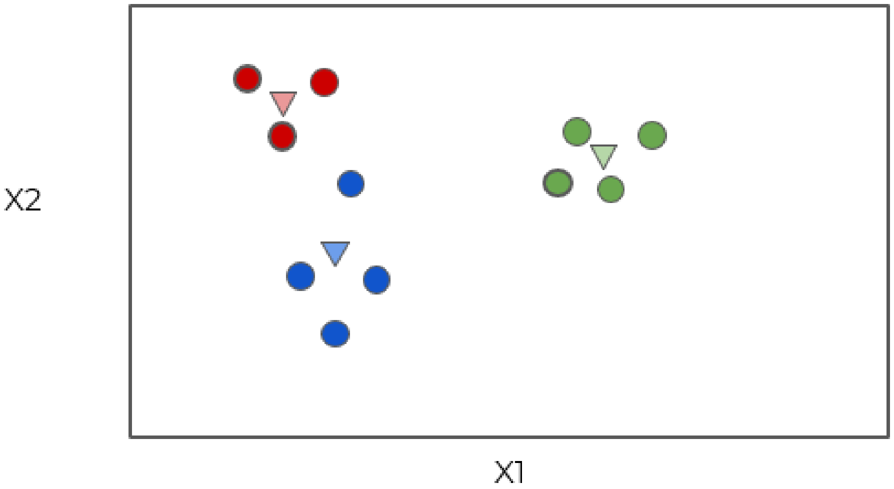
Kmeans Clustering

- We repeat steps 4 and 5 until there are no more cluster reassignments.
- Step 4b: Recalculate new cluster centers:



Kmeans Clustering

- Step 5b: Assign points to nearest cluster center.
- If there are no more reassignments, we're done! The clusters have been found.

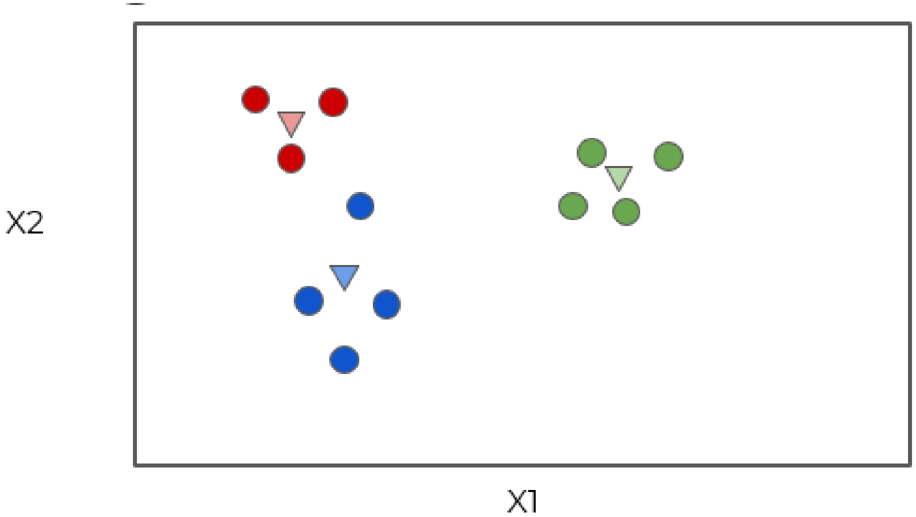


Kmeans Clustering

- How do we choose a reasonable value for K number of clusters?
- Is there any way we can evaluate how good our current K value is at determining clusters?

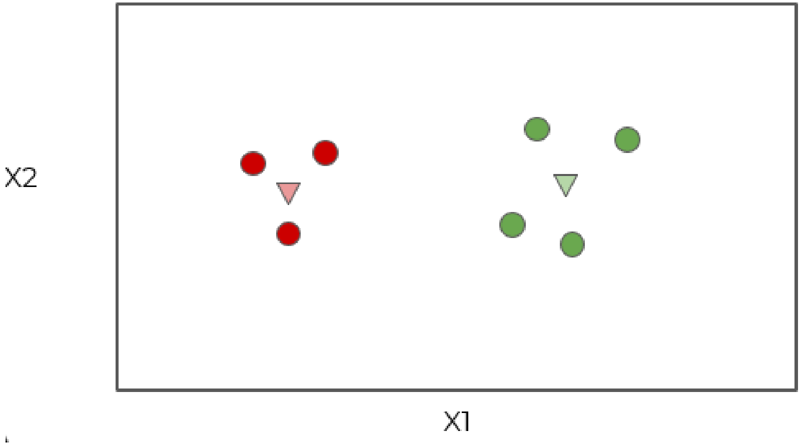
Kmeans Clustering

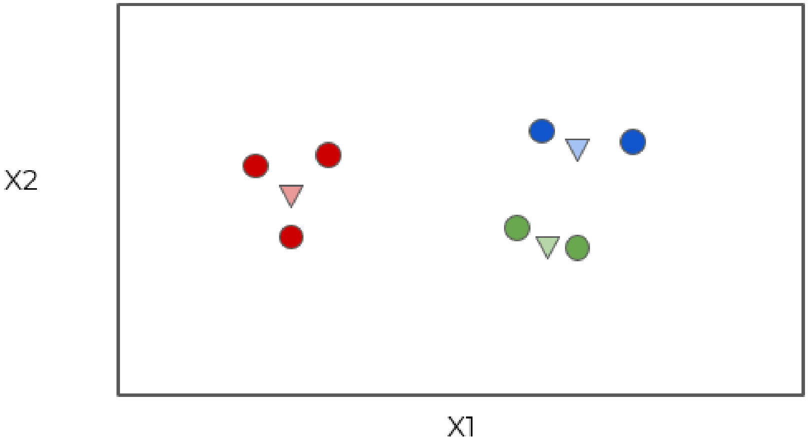
- Here we have 3 clusters, how can we measure “goodness of fit”?
  - We could measure the sum of the distances from points to cluster centers.



Kmeans Clustering

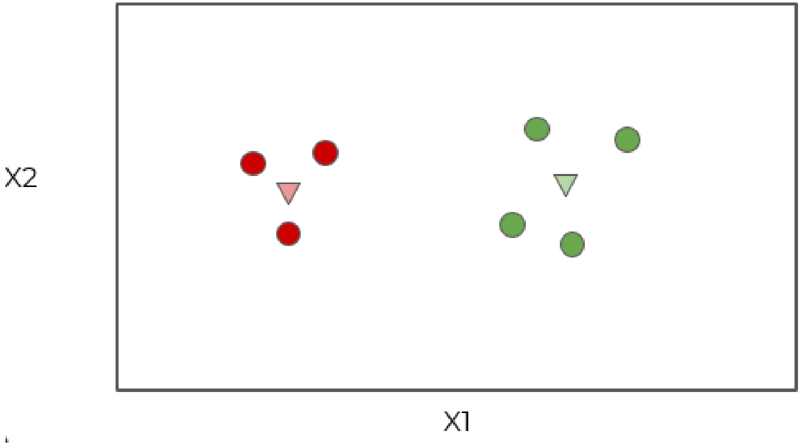
- Imagine a simple example starting with K=2.
  - We measure the sum of the squared distances from points to the cluster center:
  - Then we fit an entirely new KMeans model with K+1:

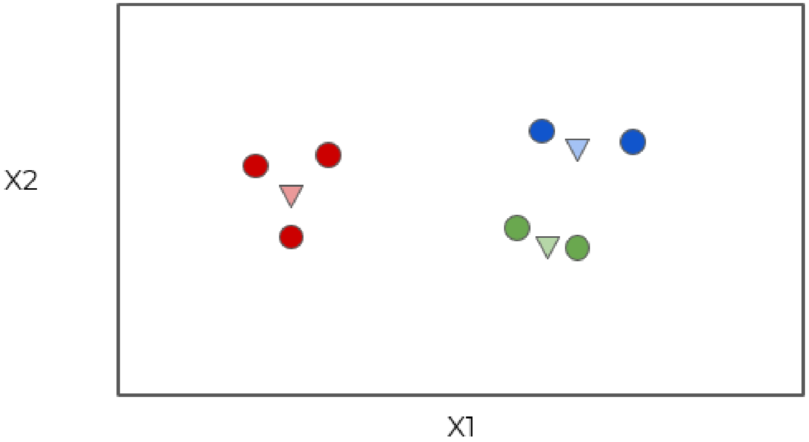




Kmeans Clustering

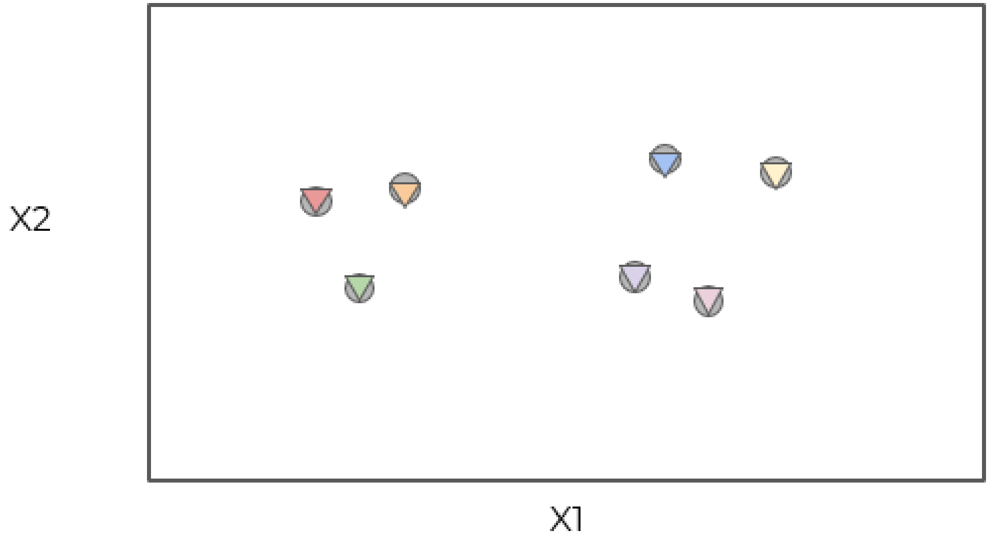
- Imagine a simple example starting with K=2.
  - We measure the sum of the squared distances from points to the cluster center:
  - Then we fit an entirely new KMeans model with K+1:





Kmeans Clustering

- Then measure again the sum of the squared distance (SSD) to center.
- In theory this SSD would go to zero once K is equal to the number of points.
  - You would have a cluster for each point! SSD would be perfect at 0!



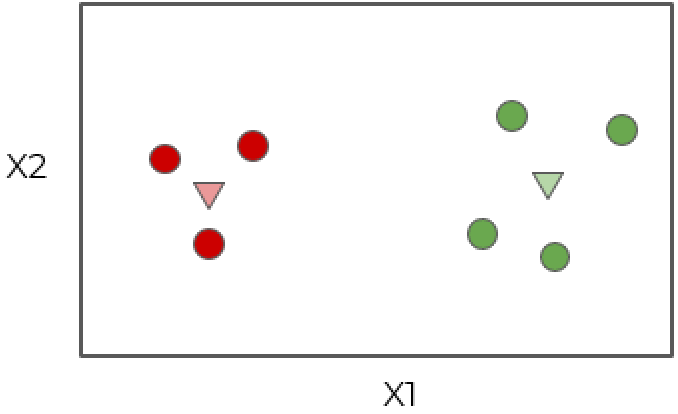
Kmeans Clustering

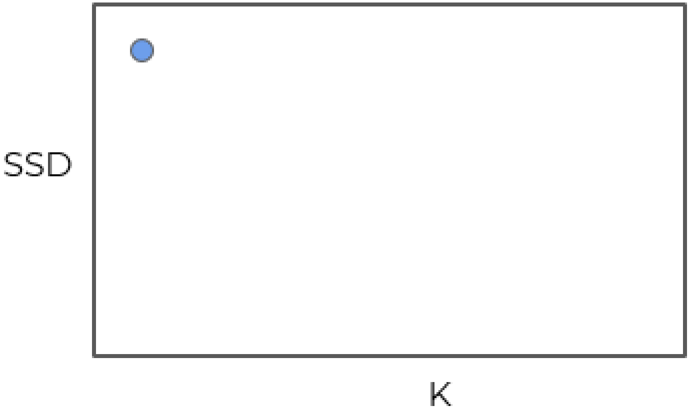
- We keep track of this SSD value for a range of different K values.
- We then look for a K value where **rate of reduction in SSD** begins to decline.
- This signifies that adding an extra cluster is **not** obtaining enough clarity of cluster separation to justify increasing K.
    - This is known as the “elbow” method since we will track where decrease in SSD begins to flatten out compared to increasing K values.

    Let’s walk through what this chart would look like...

Kmeans Clustering

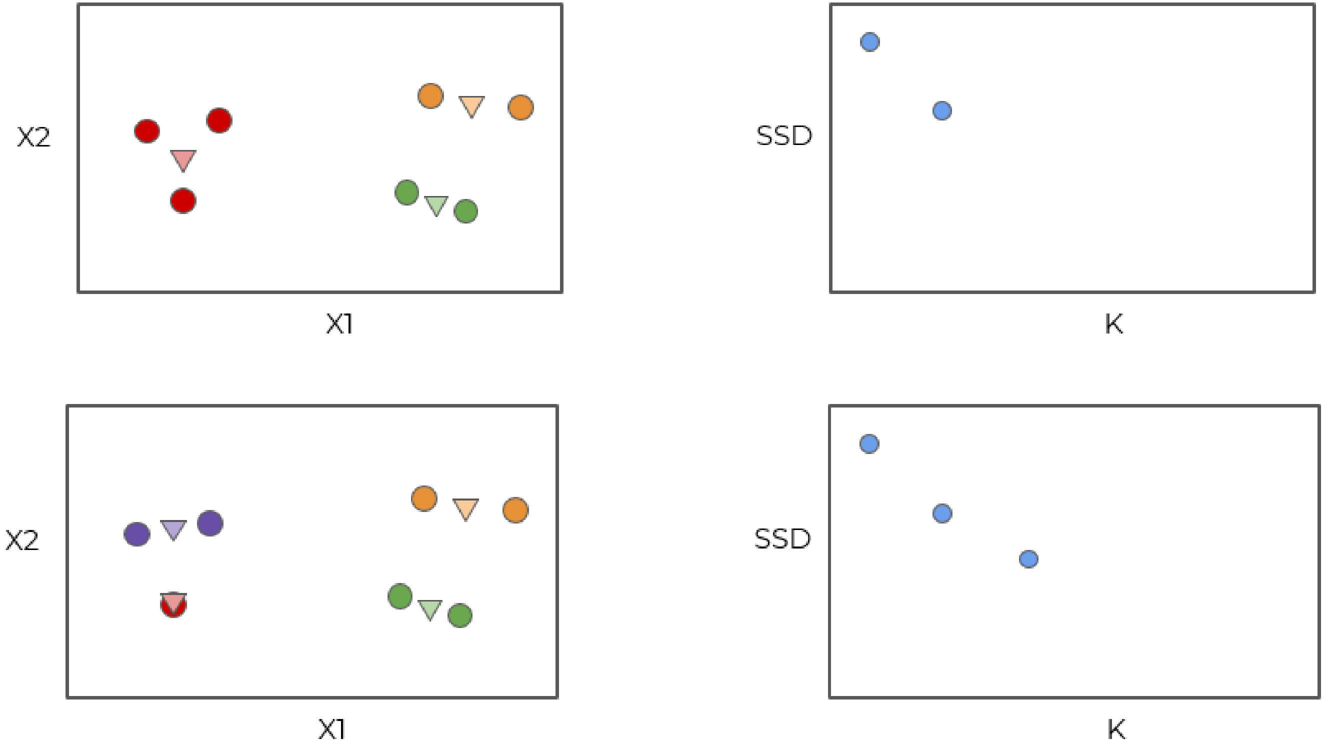
- Start with K=2:





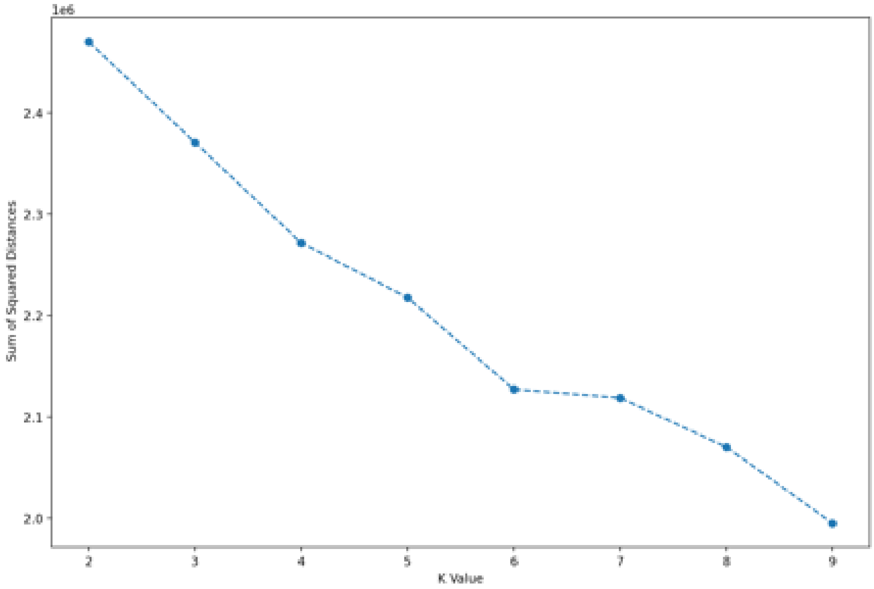
Kmeans Clustering

- Increase K and measure SSD:



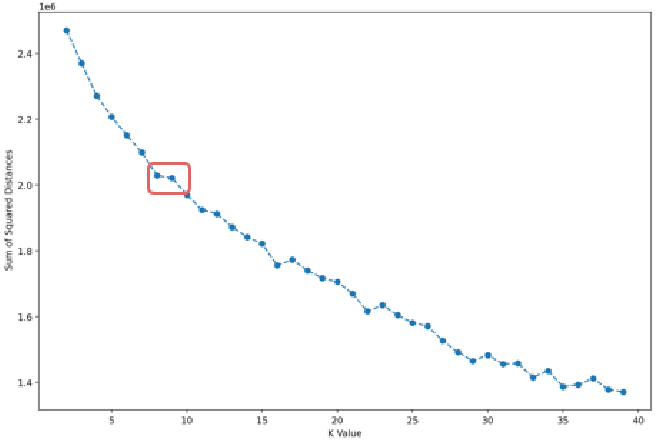
Kmeans Clustering

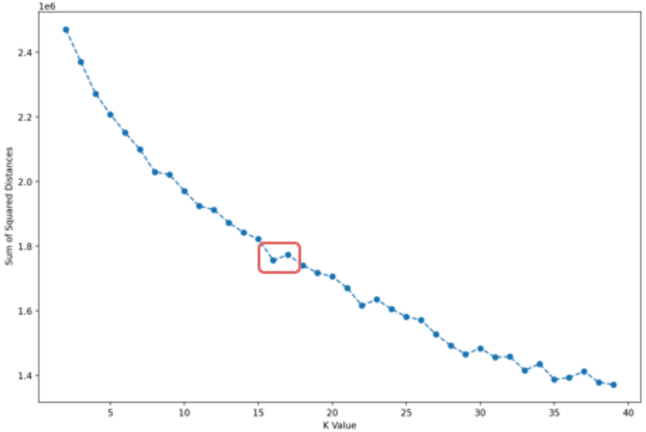
- You will see a continuous decline.

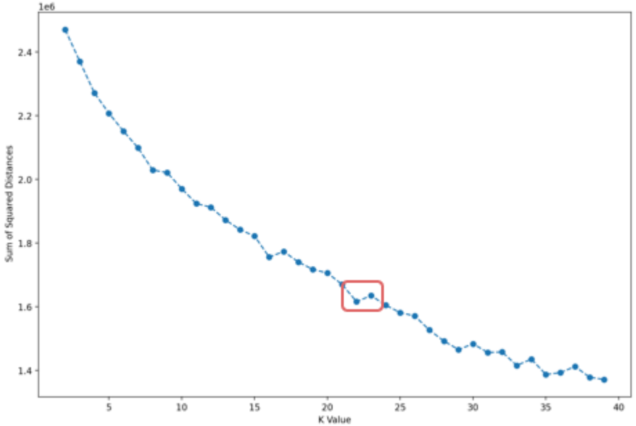


Kmeans Clustering

- Eventually you will see “elbow” points:







- These points are strong indicators that increasing K further is no longer justified as it is not revealing more “signal”.
- You can also measure out this SSD in a barplot. Let's explore this in the notebooks