Unsupervised learning
Clustering.
k-means

While problems in Pattern Recognition and Machine Learning can be of various types, they can be broadly classified into three categories:

- **Supervised Learning:**

The system is presented with example inputs and their desired outputs, given by a “teacher”, and the goal is to learn a general rule that maps inputs to outputs.

- **Unsupervised Learning:**

No labels are given to the learning algorithm, leaving it on its own to find structure in its input. Unsupervised learning can be a goal in itself (discovering hidden patterns in data) or a means towards an end (feature learning).

- **Reinforcement Learning:**

A system interacts with a dynamic environment in which it must perform a certain goal (such as driving a vehicle or playing a game against an opponent). The system is provided feedback in terms of rewards and punishments as it navigates its problem space.

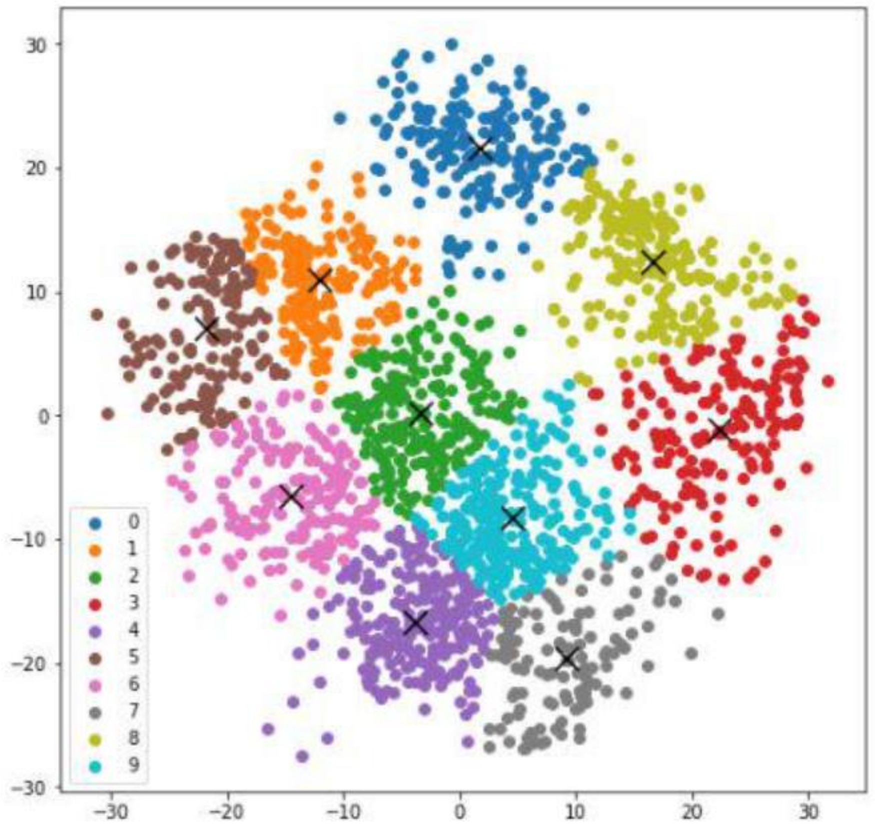
Unsupervised Learning and Data Clustering | by Sanatan Mishra | Towards Data Science

Applications of Clustering in different fields:

- Marketing.
- Biology.
- Libraries.
- Insurance.
- City Planning.
- Earthquake studies.
- Image Processing.
- Genetics.
- Finance.
- Customer Service.
- Manufacturing.
- Medical diagnosis.
- Fraud detection.
- Traffic analysis.
- Social network analysis.
- Cybersecurity.
- Climate analysis.
- Sports analysis.
- Crime analysis.

[Clustering in Machine Learning - GeeksforGeeks]

Output:



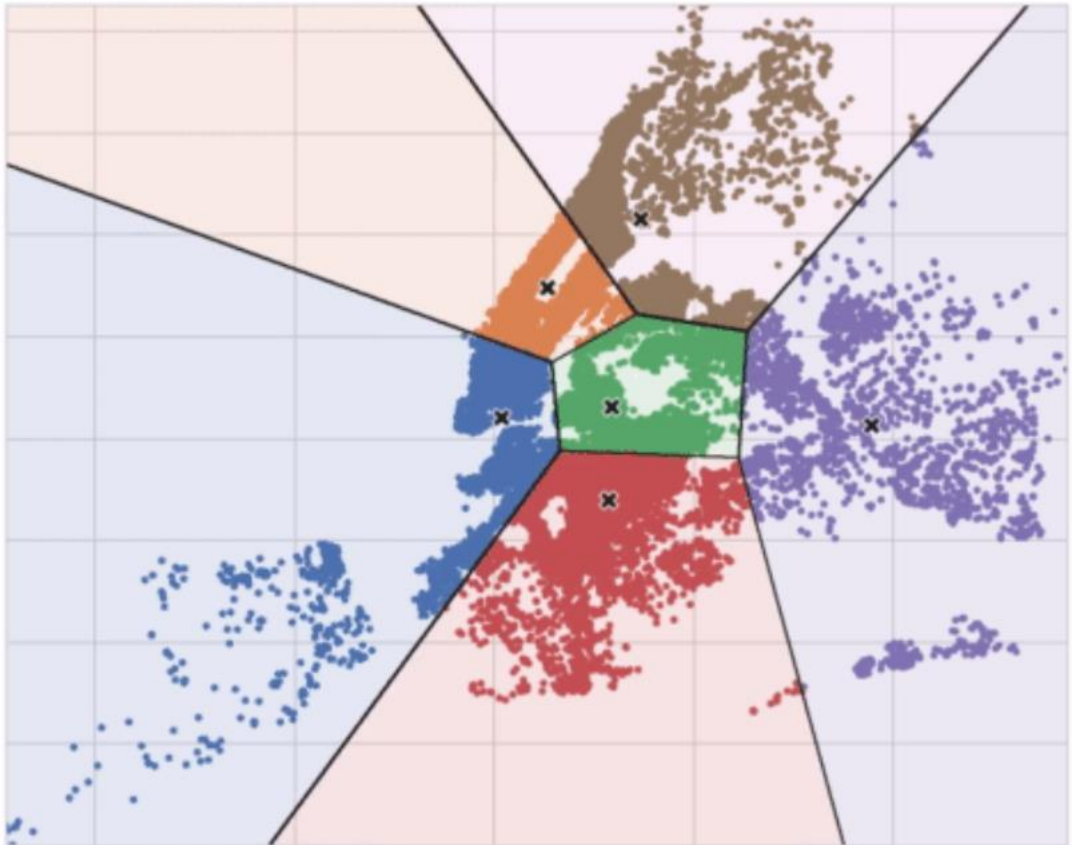
[K-Means clustering on the handwritten digits data using Scikit Learn in Python - GeeksforGeeks]

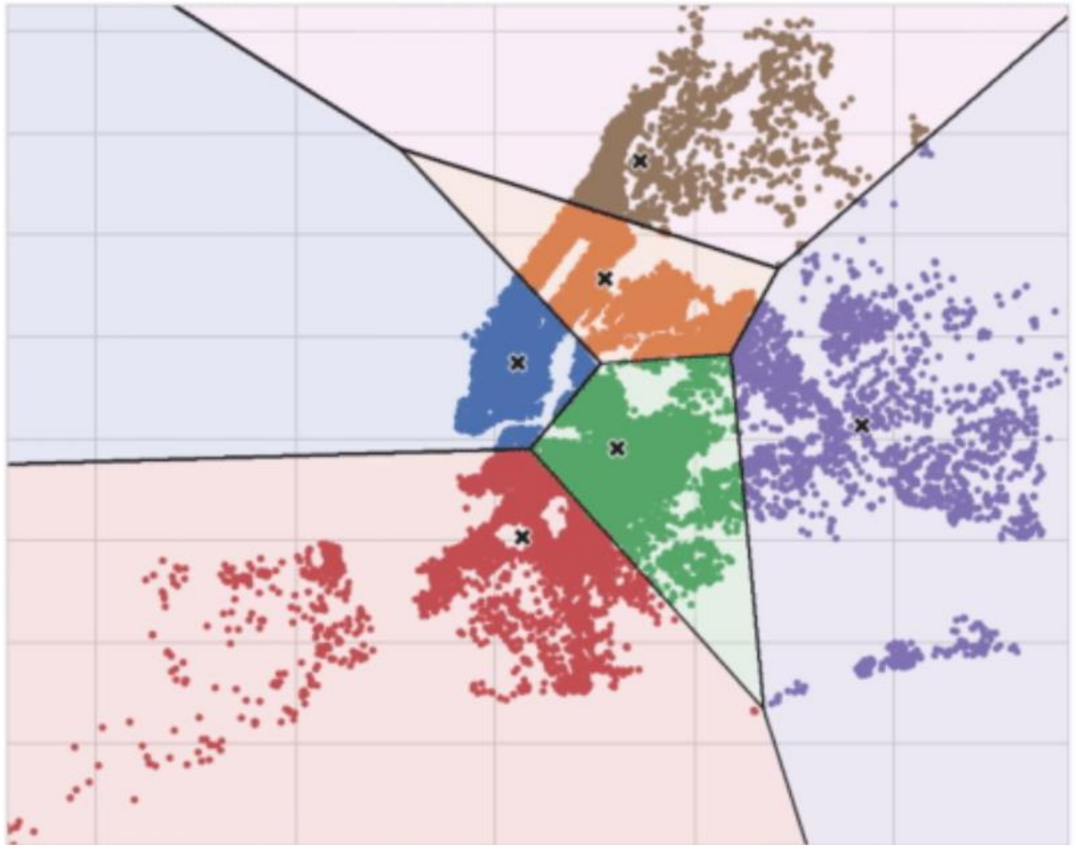
1. Select optimal value for *K* which will be the number of clusters
2. Randomly assign centroid values to each of clusters
3. Find the distance (Euclidean or Manhattan, special cases of Minkowski distance) between all the data points and centroids and assign the data point to the cluster of the nearest centroid
4. Calculate the mean values for all the coordinates of data points in all the clusters. Update the coordinate values for the centroids with the corresponding clusters mean values.
5. Repeat steps 3-4 until the new centroid values for all the clusters become the same as the previous centroid values.

K-Means Clustering Explained - neptune.ai

Python k-means clustering with scikit-learn - wellsr.com

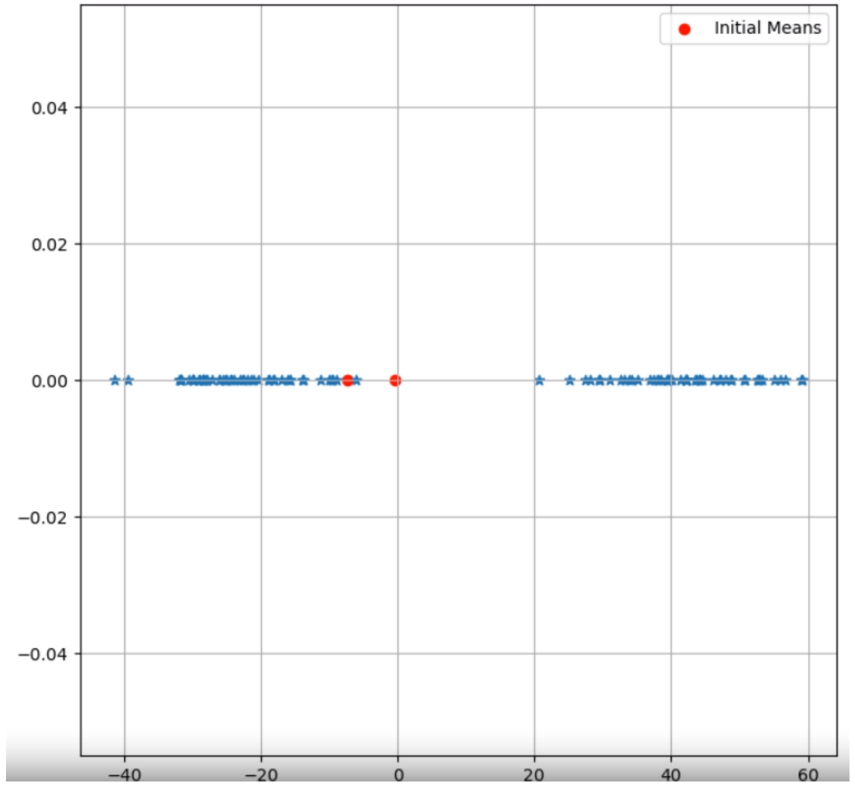
CLUSTERING WITH K-MEANS

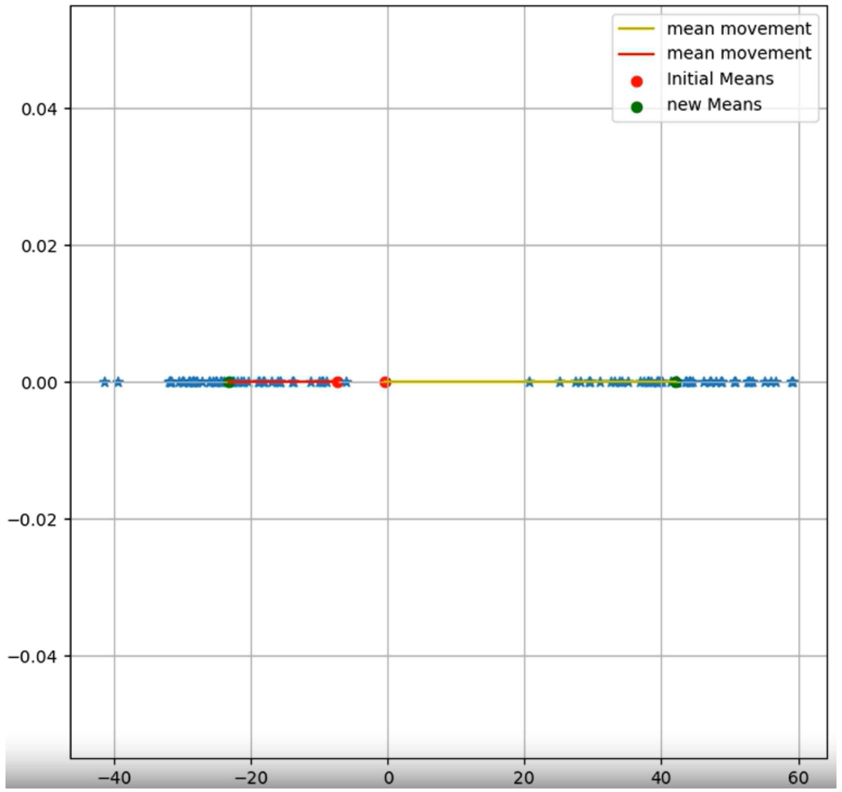


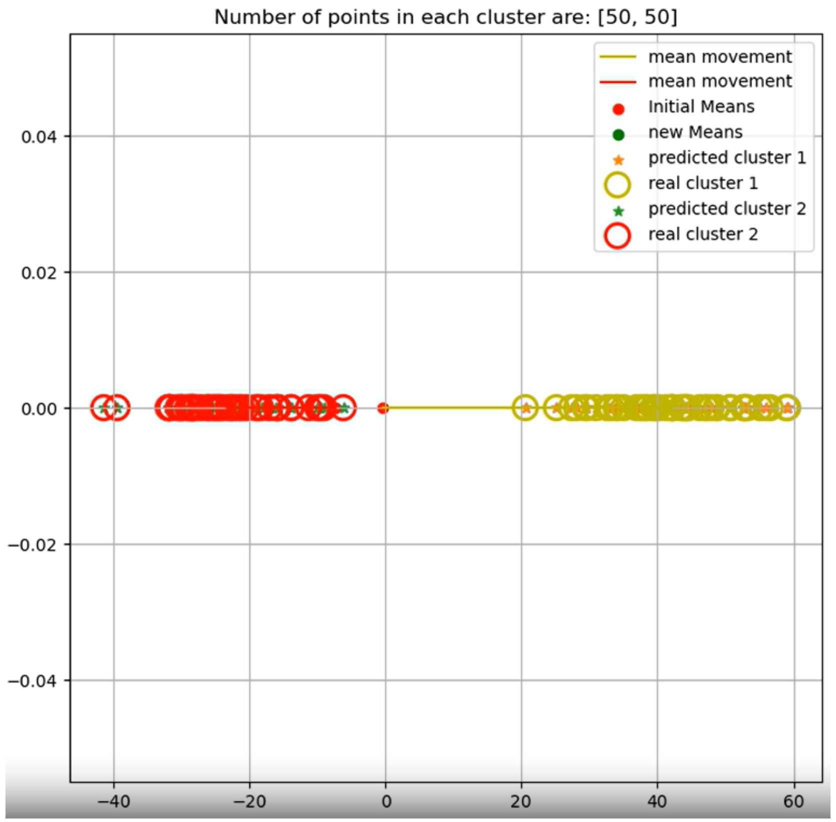


[Clustering With K-Means | Kaggle]

In this particular case the K-Means algorithm clusters accurately with 50 1D data samples of each of the two clusters.

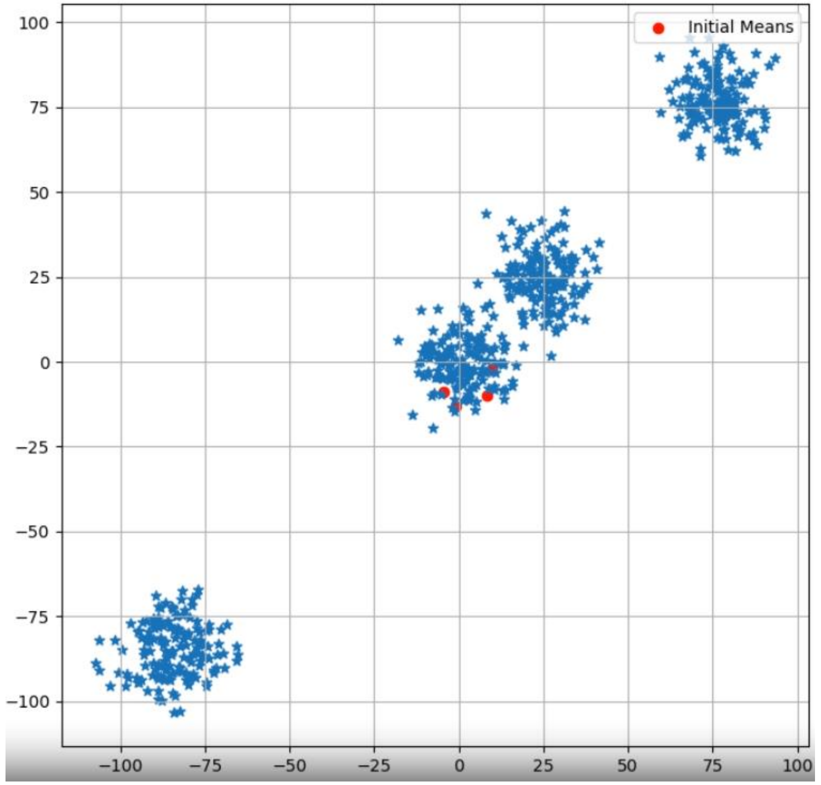


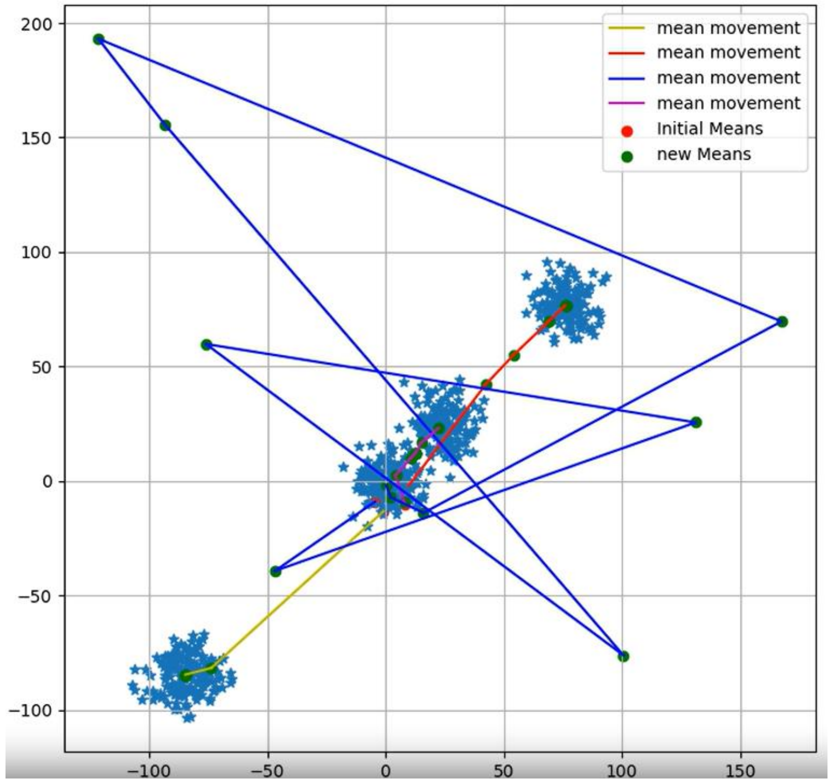


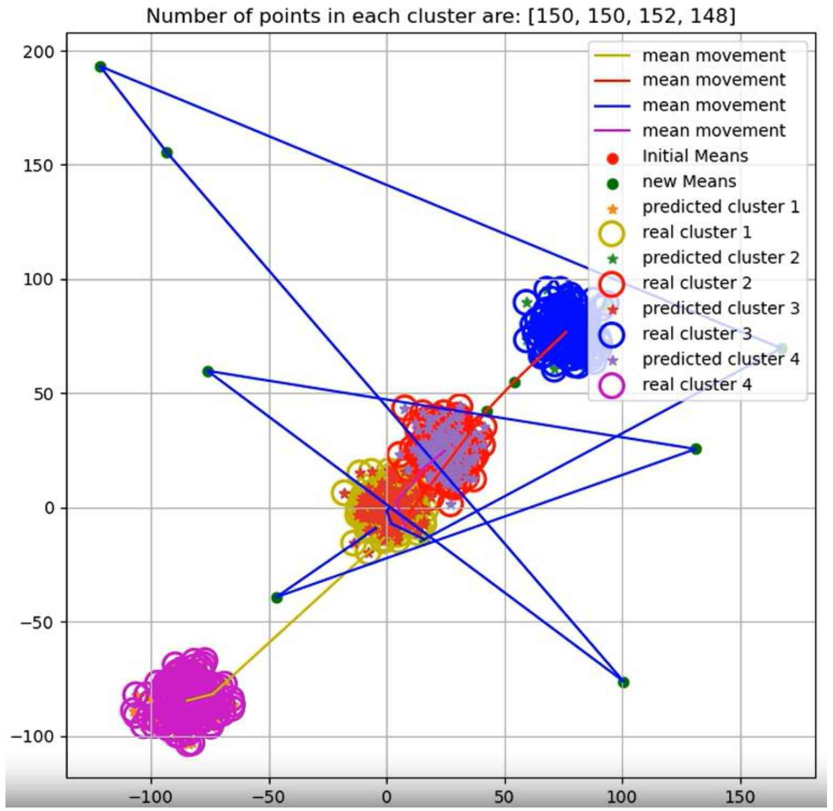


K-Means from scratch visualised with 1D, 2D and 3D data | Scratchpad (logicatcore.github.io)

2D data samples are being used to cluster with the objective of identifying 4 clusters in the data.

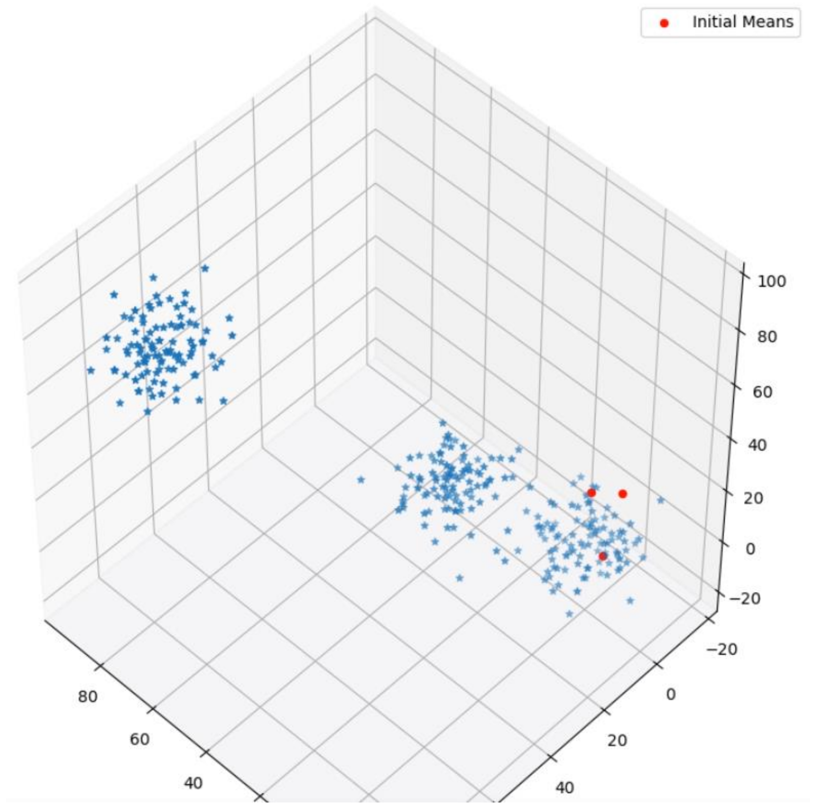


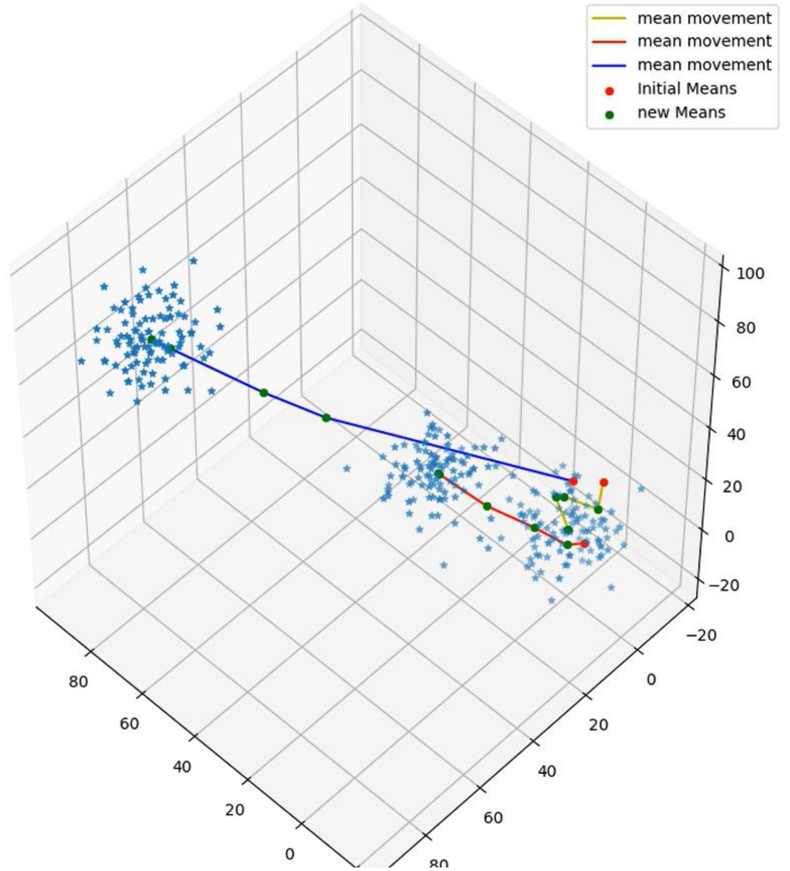


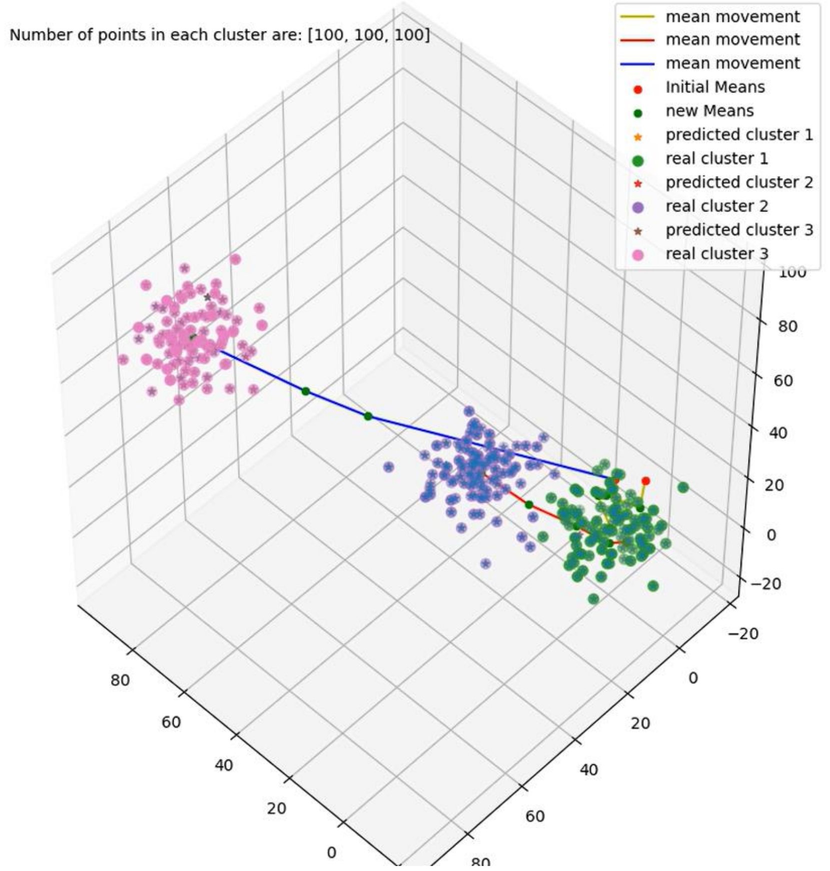


[K-Means from scratch visualised with 1D, 2D and 3D data | Scratchpad (logicatcore.github.io)]

3D data samples are being used to cluster with the objective of identifying 3 clusters in the data







K-Means from scratch visualised with 1D, 2D and 3D data | Scratchpad (logicatcore.github.io)

Initial partition:

- Pick *k* center points at random
- Pick *k* objects at random
- Construct a random partition

[An Animated Guide to KMeans Algorithm You Always Wanted to See (dailydoseofds.com)]

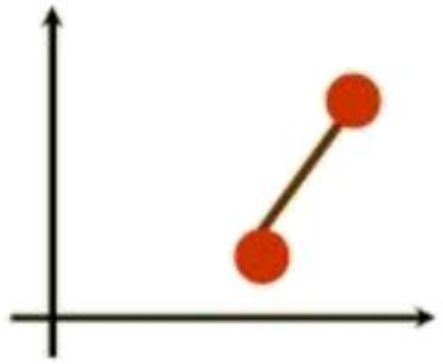
Conditions:

- Repeat until (almost) no object changes cluster
- Repeat a predetermined number of times
- Repeat until a predetermined quality is reached

[10 Clustering Algorithms With Python - MachineLearningMastery.com]

- ***Euclidean distance:***

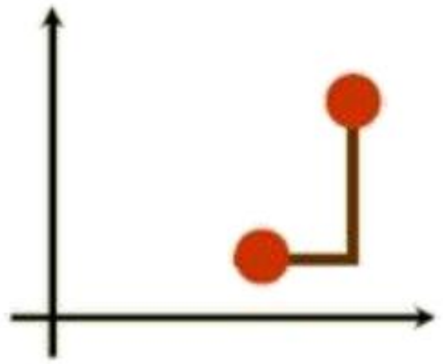
$$d(x_i, x_j) = \sqrt{\sum_{k=1}^{m} (x_i^{(k)} - x_j^{(k)})^2}$$



translation invariant

- ***Manhattan*** **(*city block*)** ***distance:***

$$d(x_i, x_j) = \sum_{k=1}^{m} \left| x_i^{(k)} - x_j^{(k)} \right|$$



approximation to Euclidian distance, cheaper to compute

- ***Minkowski distance:***

$$d_p(x_i, x_j) = \left(\sum_{k=1}^{m} \left| x_i^{(k)} - x_j^{(k)} \right|^p \right)^{\frac{1}{p}}$$

(*p* is a positive integer)

[Unsupervised Learning: Clustering (mit.edu)]

• ***Cluster inertia:*** This refers to the Sum of Squared Errors in the cluster:

$$SSE = \sum_{i=1}^{n} \sum_{j=1}^{k} w^{(i,j)} \left\| x^{(i)} - \mu^{(j)} \right\|_2^2$$

In the equation above, $\mu^{(j)}$ represents cluster *j* centroid. If $x^{(i)}$ is in this cluster (j), then $w^{(i,j)}$ =1. If it’s not, then $w^{(i,j)}$ =0.

The *k*-means algorithm aims at keeping the cluster inertia at a minimum level.

Example:

This performs 10 runs of the *k*-means algorithm on the data with a maximum of 300 iterations per run:

```
kmeans = KMeans(init="random", n_clusters=3, n_init=10,
                                        max_iter=300, random_state=42)
```

Features are scales for *k*-means with function:

```
kmeans.fit(scaled_features)
```

K-Means Clustering in Python: A Practical Guide – Real Python

PARAMETERS IN *K*-MEANS:

1. n_clusters — Number of clusters.

2. init — *k*-means algorithm has a major issue when it comes to selecting the random initial centroid location.

3. max_iter — This is to define how many iterations at most it's possible to adjust the centroids, if this doesn't limit this, then the centroids adjustment process will be performed over and over again, which might take a very long time.

4. n_init — Number of time the k-means algorithm will be run with different centroid seeds. This means the *k*-means model will test 10 different centroid location and finally among the 10 find the optimal initial location.

Then we have stored the cluster of the records in y_pred variable (prediction output).

Unsupervised Machine Learning (KMeans Clustering) with Scikit-Learn | by Charles Rajendran | Ascentic Technology | Medium

k-Means Clustering in Python. Results

```
#view cluster assignments for each observation
kmeans.labels_
```

K-Means Clustering in Python: Step-by-Step Example - Statology

Code for getting all objects in different clusters:

Painless Kmeans in Python – Step-by-Step with Sklearn (onestopdataanalysis.com)

```
#centroids of clusters
centroids = kmeans.cluster_centers_
```

Example of K-Means Clustering in Python – Data to Fish

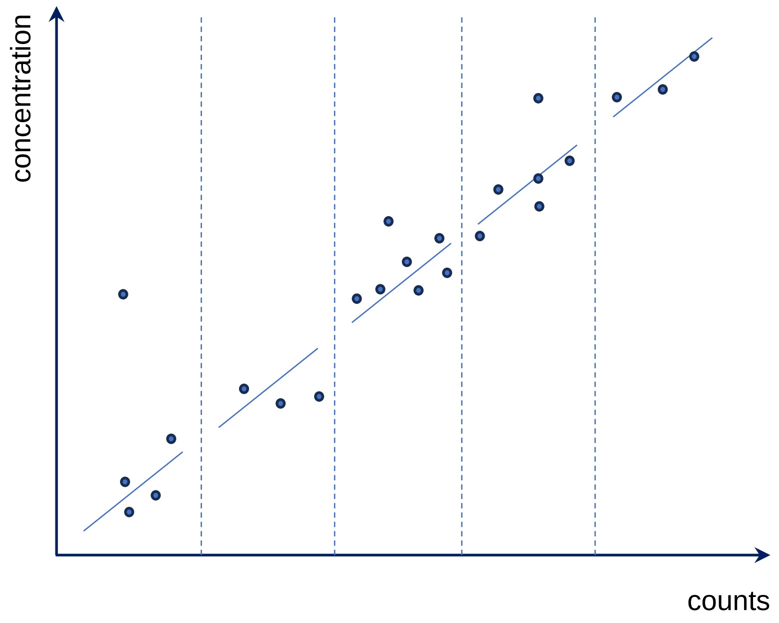
Python k-means clustering with scikit-learn - wellsr.com

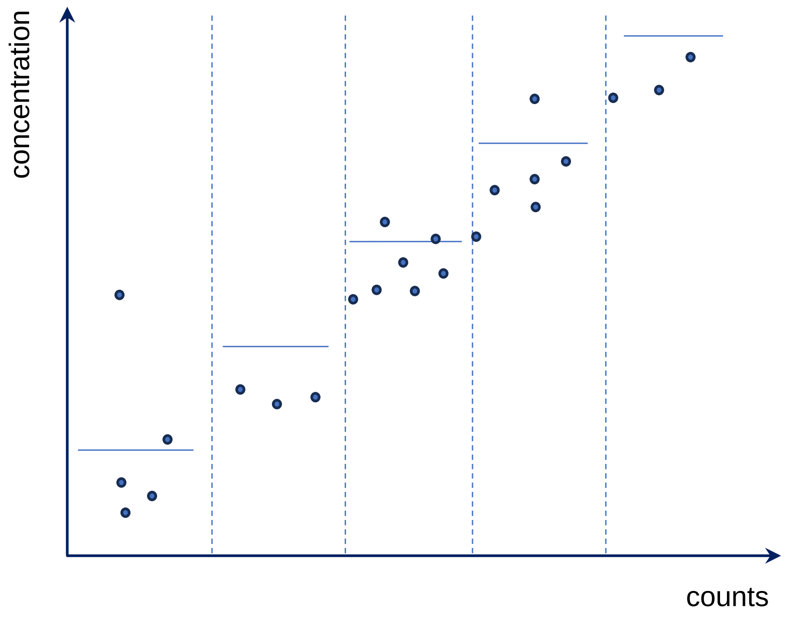
K-means application

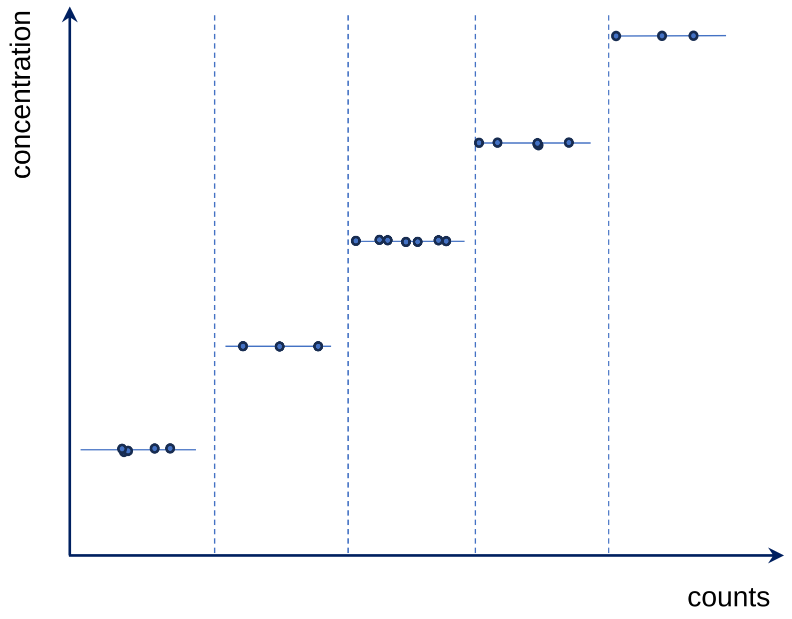
For cases when results of regressions were rounded to developing clusters:

- Well specified boundary conditions
- Applicability for small data sets
- Fast in training
- Well developed for new similar cases
- Direct application without data recalculation

Example: Concentration levels detection of different small molecules





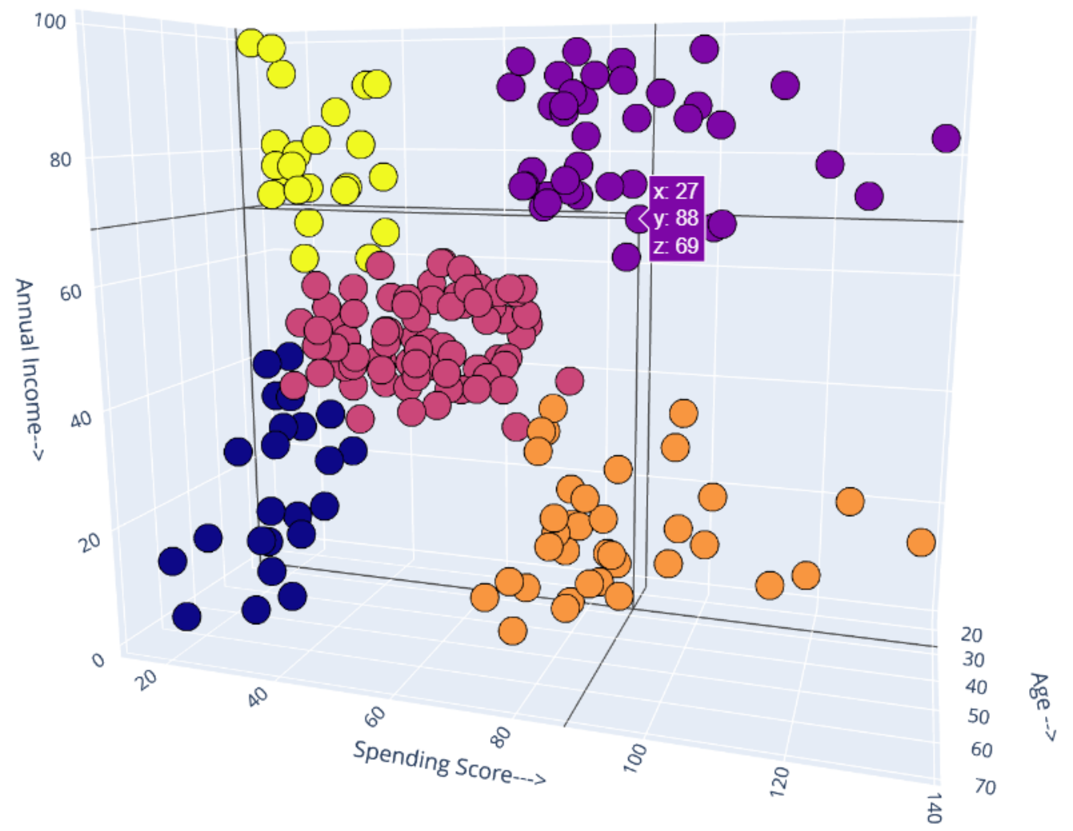


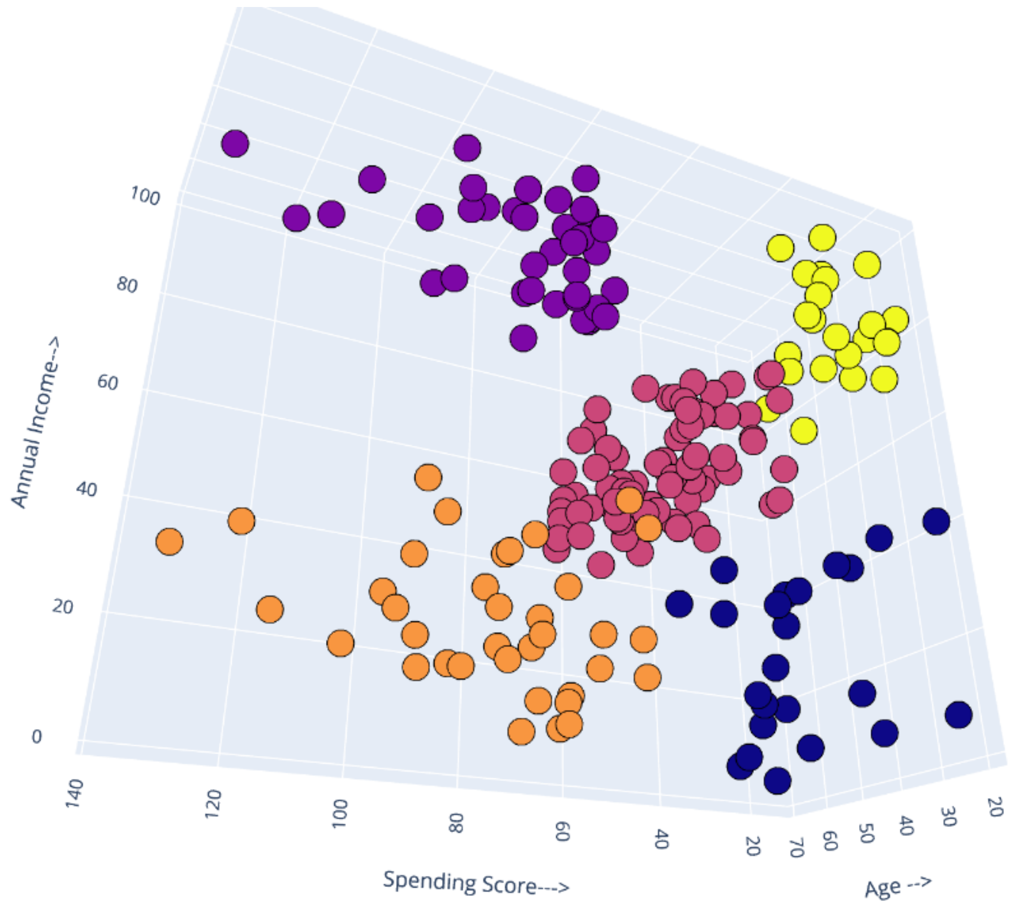
If deviation from general trend is large, the influence of the most important features can be difficult to detect

K-means advantages	K-means drawbacks
It is straightforward to understand and apply.	It’s necessary to set the number of clusters – the value of *k*.
It is applicable to clusters of different shapes and dimensions. With a large number of variables, k-means performs faster than hierarchical clustering.	It’s sensitive to rescaling. For example, the outcome will be dramatically different if one rescales the data using normalization.
It provides tighter clusters than hierarchical clustering.	The input, such as the number of clusters in a network, significantly impacts output (value of *k*).
	Clustering doesn’t work if somebody needs the clusters to have a complex geometric shape.

K-Means Clustering Algorithm in ML (serokell.io)

k-means clustering and cluster visualization in 3D





Links of codes to apply *k*-means:

Unsupervised Learning with k-Means Clustering - Atmosera

K-Means Clustering with Python and Scikit-Learn (github.com)

Python Machine Learning - K-means (w3schools.com)

8 Clustering Algorithms in Machine Learning that All Data Scientists Should Know (freecodecamp.org)

Unsupervised Learning with Python: A Beginner's Guide | Built In

Comparing usage of Scikit tools and coding math algorithm of *k*-means algorithm:

Definitive Guide to K-Means Clustering with Scikit-Learn (stackabuse.com)

Parameters selection for coding by *k*-means algorithm:

sklearn.cluster.KMeans — scikit-learn 1.2.2 documentation