What is Clustering?

Cluster analysis is a technique used in data mining and machine learning to group similar objects into clusters. K-means clustering is a widely used method for cluster analysis where the aim is to partition a set of objects into K clusters in such a way that the sum of the squared distances between the objects and their assigned cluster mean is minimized.

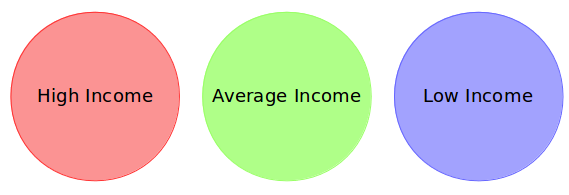
Hierarchical clustering and k-means clustering are two popular techniques in the field of unsupervised learning used for clustering data points into distinct groups. While k-means clustering divides data into a predefined number of clusters, hierarchical clustering creates a hierarchical tree-like structure to represent the relationships between the clusters.

#### **Example**

#### A bank wants to give credit card offers to its customers. Currently, they look at the details of each customer and, based on this information, decide which offer should be given to which customer.

Now, the bank can potentially have millions of customers. Does it make sense to look at the details of each customer separately and then make a decision? Certainly not! It is a manual process and will take a huge amount of time.

So what can the bank do? One option is to segment its customers into different groups. For instance, the bank can group the customers based on their income:

[](https://cdn.analyticsvidhya.com/wp-content/uploads/2019/08/Screenshot-from-2019-08-07-15-19-27.png)

Can you see where I’m going with this? The bank can now make three different strategies or offers, one for each group. Here, instead of creating different strategies for individual customers, they only have to make 3 strategies. This will reduce the effort as well as the time.

**The groups I have shown above are known as clusters, and the process of creating these groups is known as clustering.** Formally, we can say that:

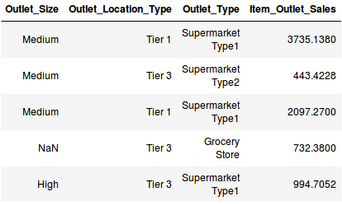
Clustering is the process of dividing the entire data into groups (also known as clusters) based on the patterns in the data.

Can you guess which type of learning problem clustering is? Is it a supervised or unsupervised learning problem?

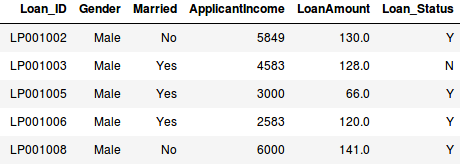
Think about it for a moment and use the example we just saw. Got it? Clustering is an unsupervised learning problem!

## How is Clustering an Unsupervised Learning Problem?

Let’s say you are working on a project where you need to predict the sales of a big mart:

[](https://cdn.analyticsvidhya.com/wp-content/uploads/2019/08/Screenshot-from-2019-08-07-16-25-31.png)

Or, a project where your task is to predict whether a loan will be approved or not:

[](https://cdn.analyticsvidhya.com/wp-content/uploads/2019/08/Screenshot-from-2019-08-07-16-29-45.png)

We have a fixed target to predict in both of these situations. In the sales prediction problem, we have to predict the Item\_Outlet\_Sales based on outlet\_size, outlet\_location\_type, etc., and in the loan approval problem, we have to predict the Loan\_Status depending on the Gender, marital status, the income of the customers, etc.

So, when we have a target variable to predict based on a given set of predictors or independent variables, such problems are called supervised learning problems.

Now, there might be situations where we do not have any target variable to predict.

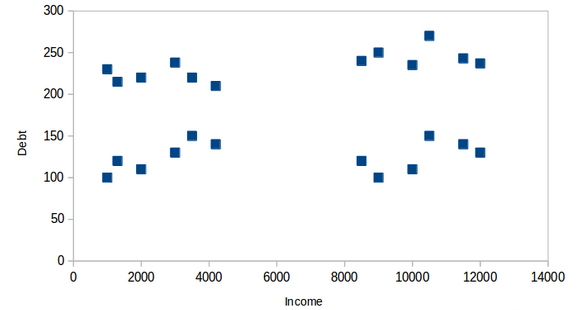
Such problems, without any fixed target variable, are known as unsupervised learning problems. In these problems, we only have the independent variables and no target/dependent variable.

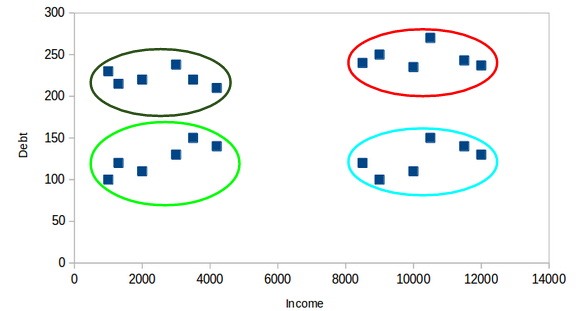
**In clustering, we do not have a target to predict. We look at the data, try to club similar observations, and form different groups. Hence it is an unsupervised learning problem.**

We now know what clusters are and the concept of clustering. Next, let’s look at the properties of these clusters, which we must consider while forming the clusters.

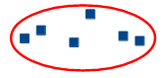
## Properties of Clusters

How about another example? We’ll take the same bank as before, which wants to segment its customers. For simplicity purposes, let’s say the bank only wants to use the income and debt to make the segmentation. They collected the customer data and used a scatter plot to visualize it:

[](https://cdn.analyticsvidhya.com/wp-content/uploads/2019/08/Screenshot-from-2019-08-08-14-46-17.png)  
On the X-axis, we have the income of the customer, and the y-axis represents the amount of debt. Here, we can clearly visualize that these customers can be segmented into 4 different clusters, as shown below:

[](https://cdn.analyticsvidhya.com/wp-content/uploads/2019/08/Screenshot-from-2019-08-08-14-47-20.png)  
This is how clustering helps to create segments (clusters) from the data. The bank can further use these clusters to make strategies and offer discounts to its customers. So let’s look at the properties of these clusters.

### **First Property of K-Means Clustering Algorithm**

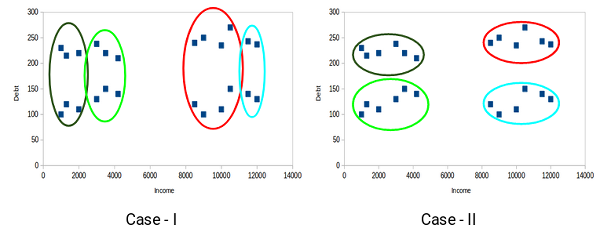
**All the data points in a cluster should be similar to each other.** Let me illustrate it using the above example:[](https://cdn.analyticsvidhya.com/wp-content/uploads/2019/08/Screenshot-from-2019-08-08-14-49-11.png)

If the customers in a particular cluster are not similar to each other, then their requirements might vary, right? If the bank gives them the same offer, they might not like it, and their interest in the bank might reduce. Not ideal.

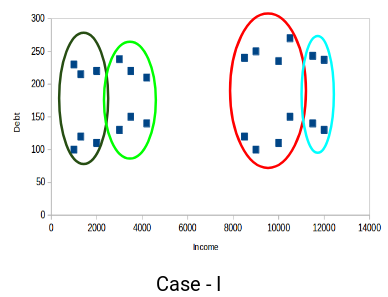
Having similar data points within the same cluster helps the bank to use targeted marketing. You can think of similar examples from your everyday life and consider how clustering will (or already does) impact the business strategy.

### **Second Property of K-Means Clustering Algorithm**

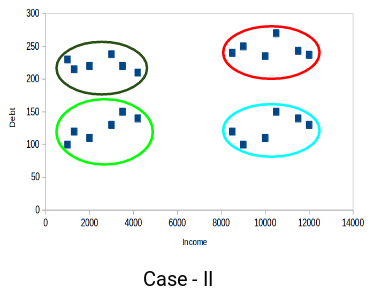
**The data points from different clusters should be as different as possible.** This will intuitively make sense if you’ve grasped the above property. Let’s again take the same example to understand this property:

[](https://cdn.analyticsvidhya.com/wp-content/uploads/2019/08/Screenshot-from-2019-08-08-14-51-31.png)

Which of these cases do you think will give us the better clusters? If you look at case I:

[](https://cdn.analyticsvidhya.com/wp-content/uploads/2019/08/Screenshot-from-2019-08-08-14-52-26.png)

Customers in the red and blue clusters are quite similar to each other. The top four points in the red cluster share similar properties to those of the blue cluster’s top two customers. They have high incomes and high debt values. Here, we have clustered them differently. Whereas, if you look at case II:

[](https://cdn.analyticsvidhya.com/wp-content/uploads/2019/08/Screenshot-from-2019-08-08-14-52-58.png)

Points in the red cluster completely differ from the customers in the blue cluster. All the customers in the red cluster have high income and high debt, while the customers in the blue cluster have high income and low debt value. Clearly, we have a better clustering of customers in this case.

Hence, data points from different clusters should be as different from each other as possible to have more meaningful clusters. The k-means algorithm uses an iterative approach to find the optimal cluster assignments by minimizing the sum of squared distances between data points and their assigned cluster centroid.

So far, we have understood what clustering is and the different properties of clusters. But why do we even need clustering? Let’s clear this doubt in the next section and look at some applications of clustering.

## Applications of Clustering in Real-World Scenarios

Clustering is a widely used technique in the industry. It is being used in almost every domain, from banking and recommendation engines to document clustering and image segmentation.

### **Customer Segmentation**

We covered this earlier – one of the most common applications of clustering is customer segmentation. And it isn’t just limited to banking. This strategy is across functions, including telecom, e-commerce, sports, advertising, sales, etc.

### **Document Clustering**

This is another common application of clustering. Let’s say you have multiple documents and you need to cluster similar documents together. Clustering helps us group these documents such that similar documents are in the same clusters.

#### **[document clustering](https://cdn.analyticsvidhya.com/wp-content/uploads/2019/08/Screenshot-from-2019-08-08-15-02-44.png)**

### **Image Segmentation**

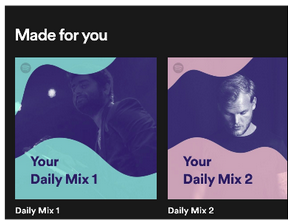
We can also use clustering to perform image segmentation. Here, we try to club similar pixels in the image together. We can apply clustering to create clusters having similar pixels in the same group.

[](https://cdn.analyticsvidhya.com/wp-content/uploads/2019/08/Screenshot-from-2019-08-08-15-03-22.png)

.

### **Recommendation Engines**

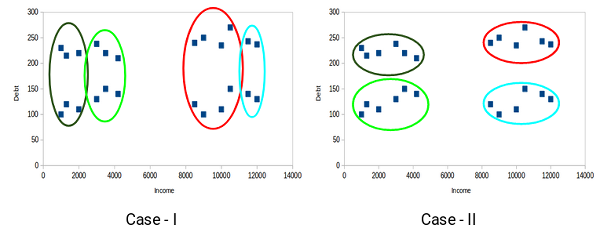
Clustering can also be used in recommendation engines. Let’s say you want to recommend songs to your friends. You can look at the songs liked by that person and then use clustering to find similar songs and finally recommend the most similar songs.

[](https://cdn.analyticsvidhya.com/wp-content/uploads/2019/08/Screenshot-from-2019-08-08-15-05-03.png)

There are many more applications that I’m sure you have already thought of. Next, let’s look at how we can evaluate our clusters.

## Understanding the Different Evaluation Metrics for Clustering

The primary aim of clustering is not just to make clusters but to make good and meaningful ones. We saw this in the below example:

[](https://cdn.analyticsvidhya.com/wp-content/uploads/2019/08/Screenshot-from-2019-08-08-14-51-31.png)

Here, we used only two features, and hence it was easy for us to visualize and decide which of these clusters was better.

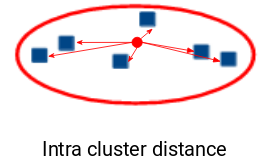
Unfortunately, that’s not how real-world scenarios work. We will have a ton of features to work with. Let’s take the customer segmentation example again – we will have features like customers’ income, occupation, gender, age, and many more. We would not be able to visualize all these features together and decide on better and more meaningful clusters.

This is where we can make use of evaluation metrics. Let’s discuss a few of them and understand how we can use them to evaluate the quality of our clusters.

### **Inertia**

Recall the first property of clusters we covered above. This is what inertia evaluates. It tells us how far the points within a cluster are. So,**inertia actually calculates the sum of distances of all the points within a cluster from the centroid of that cluster.**Normally, we use Euclidean distance as the distance metric, as long as most of the features are numeric; otherwise, Manhattan distance in case most of the features are categorical.

We calculate this for all the clusters; the final inertial value is the sum of all these distances. This distance within the clusters is known as **intracluster distance**. So, inertia gives us the sum of intracluster distances:

[](https://cdn.analyticsvidhya.com/wp-content/uploads/2019/08/Screenshot-from-2019-08-08-15-32-17.png)

Now, what do you think should be the value of inertia for a good cluster? Is a small inertial value good, or do we need a larger value? We want the points within the same cluster to be similar to each other, right? Hence, **the distance between them should be as low as possible**.

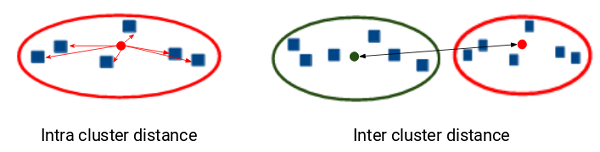
Keeping this in mind, we can say that the lesser the inertia value, the better our clusters are.

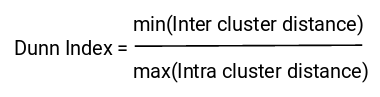
### **Dunn Index**

We now know that inertia tries to minimize the intracluster distance. It is trying to make more compact clusters.

Let me put it this way – if the distance between the centroid of a cluster and the points in that cluster is small, it means that the points are closer to each other. So, inertia makes sure that the first property of clusters is satisfied. But it does not care about the second property – that different clusters should be as different from each other as possible.

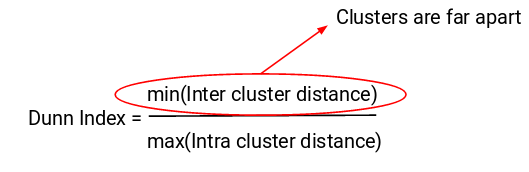
This is where the Dunn index comes into action.

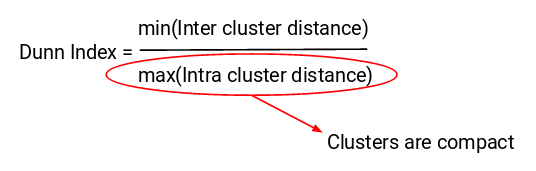
[](https://cdn.analyticsvidhya.com/wp-content/uploads/2019/08/Screenshot-from-2019-08-08-15-37-02.png)  
Along with the distance between the centroid and points,**the Dunn index also takes into account the distance between two clusters**. This distance between the centroids of two different clusters is known as **inter-cluster distance**. Let’s look at the formula of the Dunn index:

[](https://cdn.analyticsvidhya.com/wp-content/uploads/2019/08/Screenshot-from-2019-08-08-15-37-22.png)

Dunn index is the ratio of the minimum of inter-cluster distances and maximum of intracluster distances.

We want to maximize the Dunn index. The more the value of the Dunn index, the better the clusters will be. Let’s understand the intuition behind the Dunn index:

[](https://cdn.analyticsvidhya.com/wp-content/uploads/2019/08/Screenshot-from-2019-08-08-15-39-19.png)  
In order to maximize the value of the Dunn index, the numerator should be maximum. Here, we are taking the minimum of the inter-cluster distances. So, the distance between even the closest clusters should be more which will eventually make sure that the clusters are far away from each other.

[](https://cdn.analyticsvidhya.com/wp-content/uploads/2019/08/Screenshot-from-2019-08-08-15-39-34.png)

Also, the denominator should be minimum to maximize the Dunn index. Here, we are taking the maximum of all intracluster distances. Again, the intuition is the same here. The maximum distance between the cluster centroids and the points should be minimum, eventually ensuring that the clusters are compact.

### **Silhouette Score**

The silhouette score and plot are used to evaluate the quality of a clustering solution produced by the k-means algorithm. The silhouette score measures the similarity of each point to its own cluster compared to other clusters, and the silhouette plot visualizes these scores for each sample. A high silhouette score indicates that the clusters are well separated, and each sample is more similar to the samples in its own cluster than to samples in other clusters. A silhouette score close to 0 suggests overlapping clusters, and a negative score suggests poor clustering solutions.

## What Is K-Means Clustering?

K-means clustering is a method for grouping n observations into K clusters. It uses vector quantization and aims to assign each observation to the cluster with the nearest mean or centroid, which serves as a prototype for the cluster. Originally developed for signal processing, K-means clustering is now widely used in machine learning to partition data points into K clusters based on their similarity. The goal is to minimize the sum of squared distances between the data points and their corresponding cluster centroids, resulting in clusters that are internally homogeneous and distinct from each other.

Recall the first property of clusters – it states that the points within a cluster should be similar to each other. So,**our aim here is to minimize the distance between the points within a cluster.**

There is an algorithm that tries to minimize the distance of the points in a cluster with their centroid – the k-means clustering technique.

K-means is a centroid-based algorithm or a distance-based algorithm, where we calculate the distances to assign a point to a cluster. In K-Means, each cluster is associated with a centroid.

**The main objective of the K-Means algorithm is to minimize the sum of distances between the points and their respective cluster centroid.**

Optimization plays a crucial role in the k-means clustering algorithm. The goal of the optimization process is to find the best set of centroids that minimizes the sum of squared distances between each data point and its closest centroid. This process is repeated multiple times until convergence, resulting in the optimal clustering solution.

## How to Apply K-Means Clustering Algorithm?

Let’s now take an example to understand how K-Means actually works:

[](https://cdn.analyticsvidhya.com/wp-content/uploads/2019/08/Screenshot-from-2019-08-09-12-21-43.png)

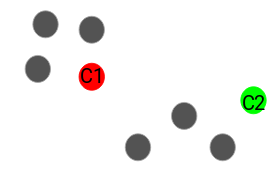
Time needed: 10 minutes

We have these 8 points, and we want to apply k-means to create clusters for these points. Here’s how we can do it.

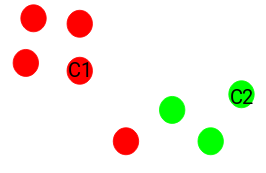
1. **Choose the number of clusters k**

The first step in k-means is to pick the number of clusters, k.

1. **Select k random points from the data as centroids**

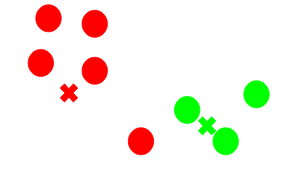
Next, we randomly select the centroid for each cluster. Let’s say we want to have 2 clusters, so k is equal to 2 here. We then randomly select the centroid:  
  
Here, the red and green circles represent the centroid for these clusters.

1. **Assign all the points to the closest cluster centroid**

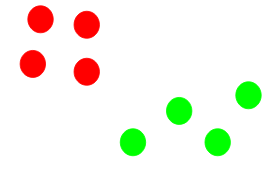
Once we have initialized the centroids, we assign each point to the closest cluster centroid:   


Here you can see that the points closer to the red point are assigned to the red cluster, whereas the points closer to the green point are assigned to the green cluster.

1. **Recompute the centroids of newly formed clusters**

Now, once we have assigned all of the points to either cluster, the next step is to compute the centroids of newly formed clusters:  
  
Here, the red and green crosses are the new centroids.

1. **Repeat steps 3 and 4**

We then repeat steps 3 and 4:  
  
The step of computing the centroid and assigning all the points to the cluster based on their distance from the centroid is a single iteration. But wait – when should we stop this process? It can’t run till eternity, right?

#### **Stopping Criteria for K-Means Clustering**

There are essentially three stopping criteria that can be adopted to stop the K-means algorithm:

1. Centroids of newly formed clusters do not change
2. Points remain in the same cluster
3. Maximum number of iterations is reached

We can stop the algorithm if the centroids of newly formed clusters are not changing. Even after multiple iterations, if we are getting the same centroids for all the clusters, we can say that the algorithm is not learning any new pattern, and it is a sign to stop the training.

Another clear sign that we should stop the training process is if the points remain in the same cluster even after training the algorithm for multiple iterations.

Finally, we can stop the training if the maximum number of iterations is reached. Suppose we have set the number of iterations as 100. The process will repeat for 100 iterations before stopping.

#import libraries

import pandas as pd

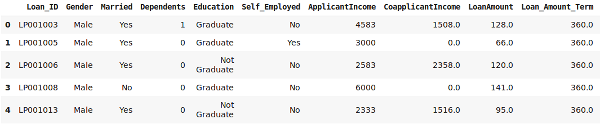
import numpy as np

import random as rd

import matplotlib.pyplot as plt

data = pd.read\_csv('clustering.csv')

data.head()



X = data[["LoanAmount","ApplicantIncome"]]

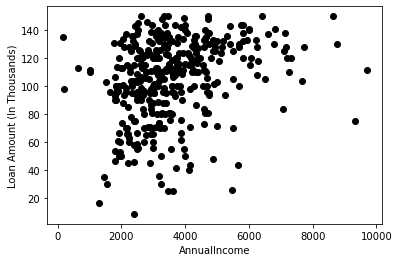
#Visualise data points

plt.scatter(X["ApplicantIncome"],X["LoanAmount"],c='black')

plt.xlabel('AnnualIncome')

plt.ylabel('Loan Amount (In Thousands)')

plt.show()



# Randomly initialize cluster centroids

K = 3 # Specify the number of clusters

Centroids = (X.sample(n=K))

# Step 3 - Assign all the points to the closest cluster centroid

# Step 4 - Recompute centroids of newly formed clusters

# Step 5 - Repeat step 3 and 4

diff = 1

j=0

while(diff!=0):

XD=X

i=1

for index1,row\_c in Centroids.iterrows():

ED=[]

for index2,row\_d in XD.iterrows():

d1=(row\_c["ApplicantIncome"]-row\_d["ApplicantIncome"])\*\*2

d2=(row\_c["LoanAmount"]-row\_d["LoanAmount"])\*\*2

d=np.sqrt(d1+d2)

ED.append(d)

X[i]=ED

i=i+1

C=[]

for index,row in X.iterrows():

min\_dist=row[1]

pos=1

for i in range(K):

if row[i+1] < min\_dist:

min\_dist = row[i+1]

pos=i+1

C.append(pos)

X["Cluster"]=C

Centroids\_new = X.groupby(["Cluster"]).mean()[["LoanAmount","ApplicantIncome"]]

if j == 0:

diff=1

j=j+1

else:

diff = (Centroids\_new['LoanAmount'] - Centroids['LoanAmount']).sum() + (Centroids\_new['ApplicantIncome'] - Centroids['ApplicantIncome']).sum()

print(diff.sum())

Centroids = X.groupby(["Cluster"]).mean()[["LoanAmount","ApplicantIncome"]]

217.4992641591473

193.25510336836837

198.8254734312786

97.96657416663327

80.42253001270922

99.50821192263949

27.38200063816943

18.274686272279013

9.21023994083339

18.345487493007468

46.27013250786139

0.0

color=['blue','green','cyan']

for k in range(K):

data=X[X["Cluster"]==k+1]

plt.scatter(data["ApplicantIncome"],data["LoanAmount"],c=color[k])

plt.scatter(Centroids["ApplicantIncome"],Centroids["LoanAmount"],c='red')

plt.xlabel('Income')

plt.ylabel('Loan Amount (In Thousands)')

plt.show()

