ECOM SCHOOL
המכללה למקצועות הדיגיטל וההייטק

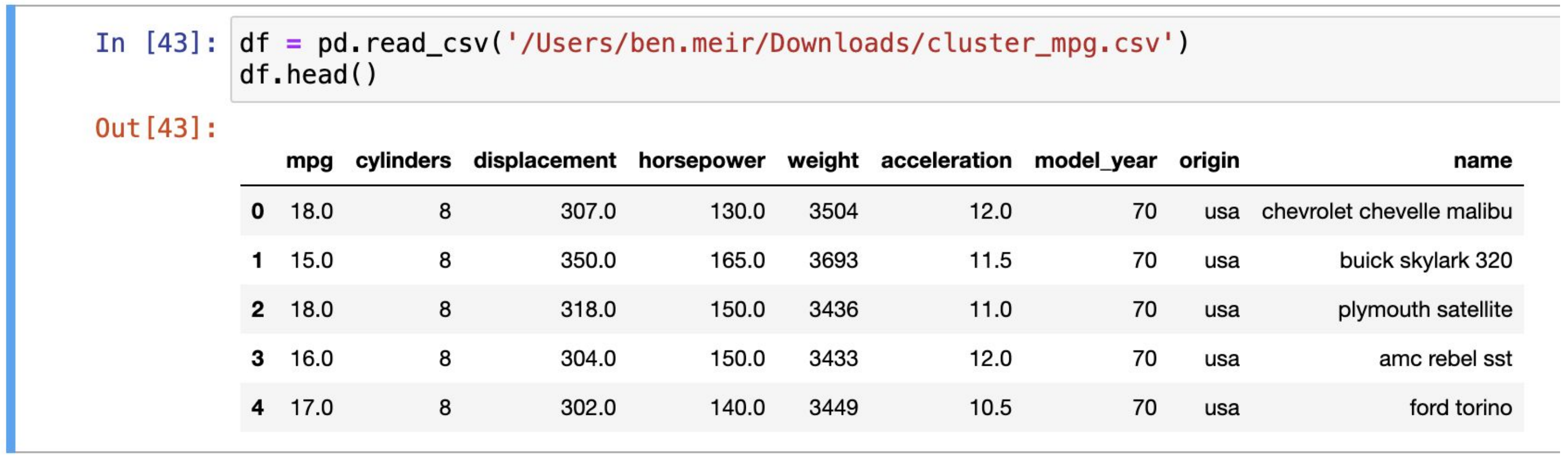
Last lecture reminder

We learned about:

- K-Mean Clustering - Image Quantization
- Image Quantization - Python Example
- Hierarchical Clustering - Introduction
- Hierarchical Clustering - Dendrogram
- Hierarchical Clustering - Distance Calculation (using euclidean distance)

Hierarchical Clustering - Python Example

For the example we will use the **'cluster_mpg.csv'** csv file that contain data about different car models. We already used this dataset in previous examples during the course.



```
In [43]: df = pd.read_csv('/Users/ben.meir/Downloads/cluster_mpg.csv')
         df.head()

Out[43]:
```

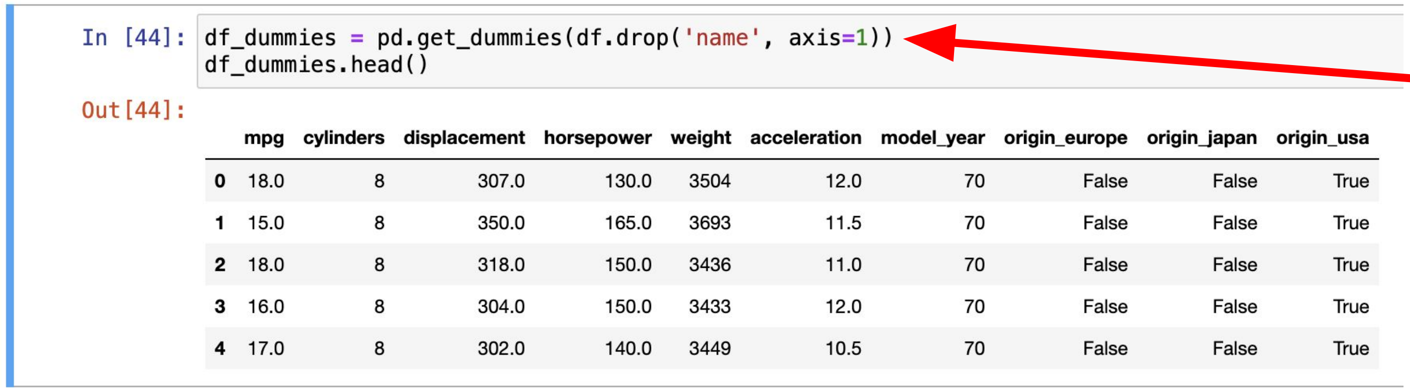
	mpg	cylinders	displacement	horsepower	weight	acceleration	model_year	origin	name
0	18.0	8	307.0	130.0	3504	12.0	70	usa	chevrolet chevelle malibu
1	15.0	8	350.0	165.0	3693	11.5	70	usa	buick skylark 320
2	18.0	8	318.0	150.0	3436	11.0	70	usa	plymouth satellite
3	16.0	8	304.0	150.0	3433	12.0	70	usa	amc rebel sst
4	17.0	8	302.0	140.0	3449	10.5	70	usa	ford torino

Because Hierarchical Clustering is a distance based algorithm we need to apply some data preparation that includes the following:

- Handle string type feature values by applying pd.get_dummies() method
- Applying feature scaling so the model won't be affected from larger feature values

Hierarchical Clustering - Python Example

For handling string type feature values:



```
In [44]: df_dummies = pd.get_dummies(df.drop('name', axis=1))
         df_dummies.head()
Out[44]:
```

	mpg	cylinders	displacement	horsepower	weight	acceleration	model_year	origin_europe	origin_japan	origin_usa
0	18.0	8	307.0	130.0	3504	12.0	70	False	False	True
1	15.0	8	350.0	165.0	3693	11.5	70	False	False	True
2	18.0	8	318.0	150.0	3436	11.0	70	False	False	True
3	16.0	8	304.0	150.0	3433	12.0	70	False	False	True
4	17.0	8	302.0	140.0	3449	10.5	70	False	False	True

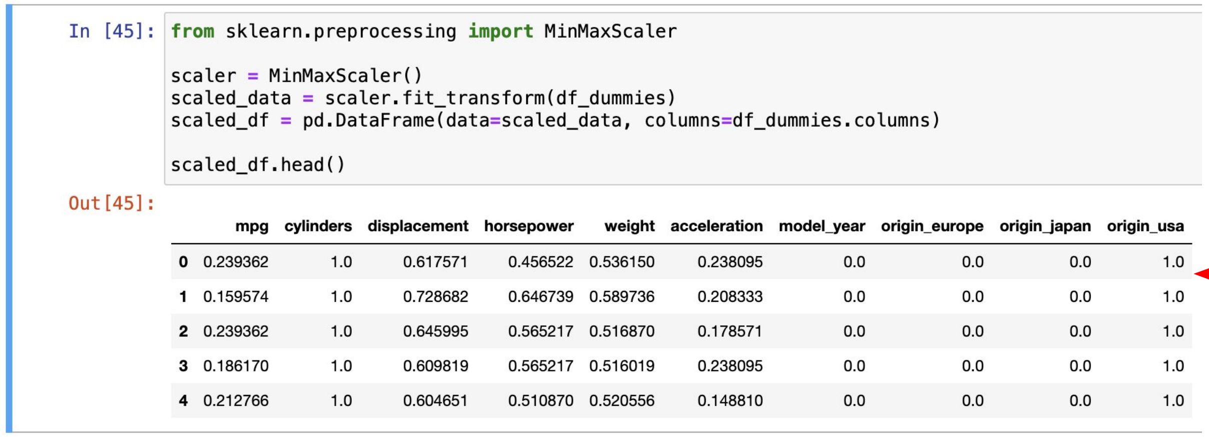
We removed the ‘name’ column because we don’t want a different True / False column for each car model name

For applying feature scaling we will use the **MinMaxScaler** which is a little different then the StandardScaler we used until now.

MinMaxScaler → MinMaxScaler scales the data to a fixed range (value between 0 to 1), which is achieved by subtracting the minimum value of the feature and then dividing by the range of the feature. It is useful when we need values in a bounded interval. However, this scaler is sensitive to outliers, so if there are outliers in the data, you might want to consider the StandardScaler.

Hierarchical Clustering - Python Example

For applying MinMaxScaler:



```
In [45]: from sklearn.preprocessing import MinMaxScaler

         scaler = MinMaxScaler()
         scaled_data = scaler.fit_transform(df_dummies)
         scaled_df = pd.DataFrame(data=scaled_data, columns=df_dummies.columns)

         scaled_df.head()
```

Out[45]:

	mpg	cylinders	displacement	horsepower	weight	acceleration	model_year	origin_europe	origin_japan	origin_usa
0	0.239362	1.0	0.617571	0.456522	0.536150	0.238095	0.0	0.0	0.0	1.0
1	0.159574	1.0	0.728682	0.646739	0.589736	0.208333	0.0	0.0	0.0	1.0
2	0.239362	1.0	0.645995	0.565217	0.516870	0.178571	0.0	0.0	0.0	1.0
3	0.186170	1.0	0.609819	0.565217	0.516019	0.238095	0.0	0.0	0.0	1.0
4	0.212766	1.0	0.604651	0.510870	0.520556	0.148810	0.0	0.0	0.0	1.0

We can see that all values in the scaled df is between 0 and 1 because we used the MinMaxScaler

Using MinMaxScaler in Hierarchical Clustering model can be very useful because it allowing us to limit the minimum and maximum distance between data points.

As we learned in euclidean distance, the total distance between two data points is calculated by the following formula → $\boxed{D = sqrt(X1 - Y1)^2 + (X2 - Y2)^2 + … + (Xn - Yn)^2)}$

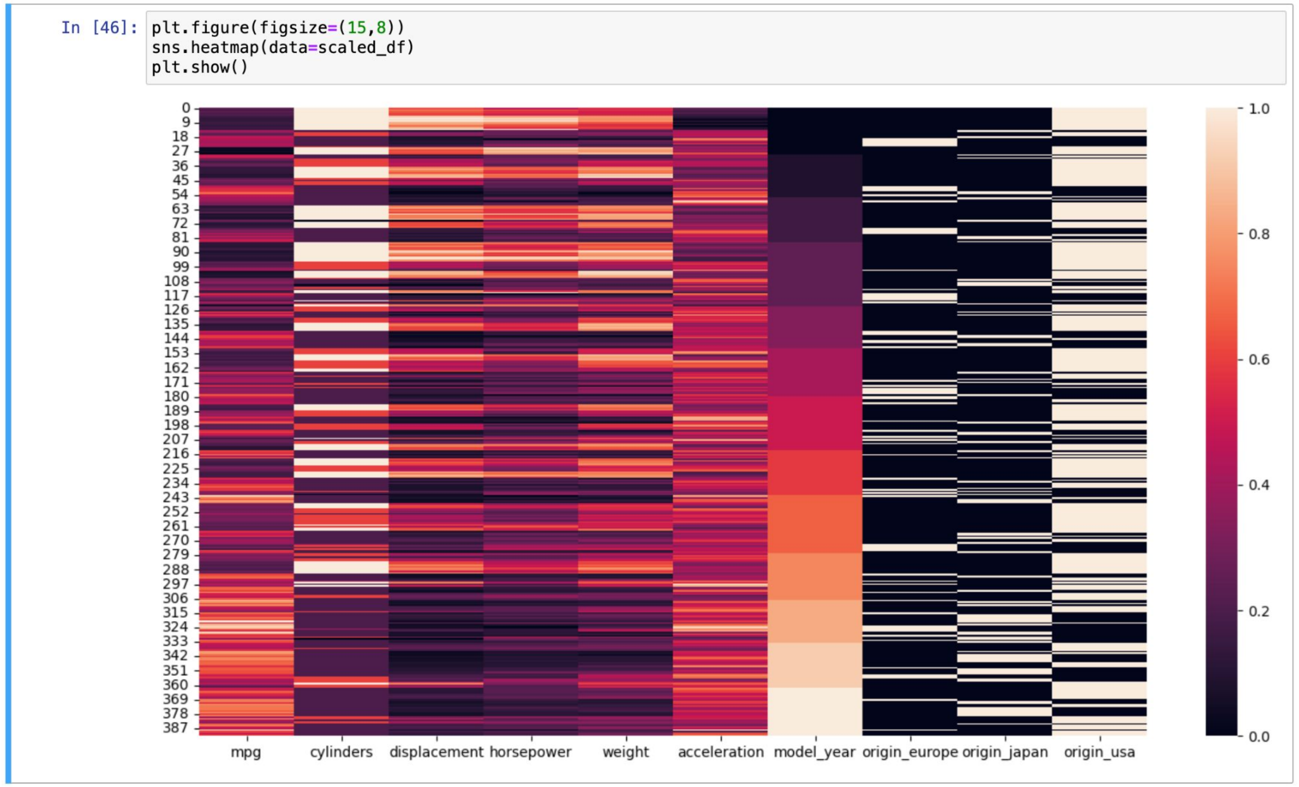
So with MinMaxScaler the maximum difference between the feature values can be 1

and that's mean that → **Max(D) = sqrt(1^2 + 1^2 + … + 1^2) = sqrt(N_featurees)**

Hierarchical Clustering - Python Example

We can visualize the scaled data results using a **heatmap** from seaborn library.

sns.heatmap() → A heatmap is a graphical representation of data where values are depicted by color. Heatmaps make it easy to visualize complex data and understand it at a glance. Darker shades usually denote lower values, and lighter shades denote higher values.



```
In [46]: plt.figure(figsize=(15,8))
         sns.heatmap(data=scaled_df)
         plt.show()
```

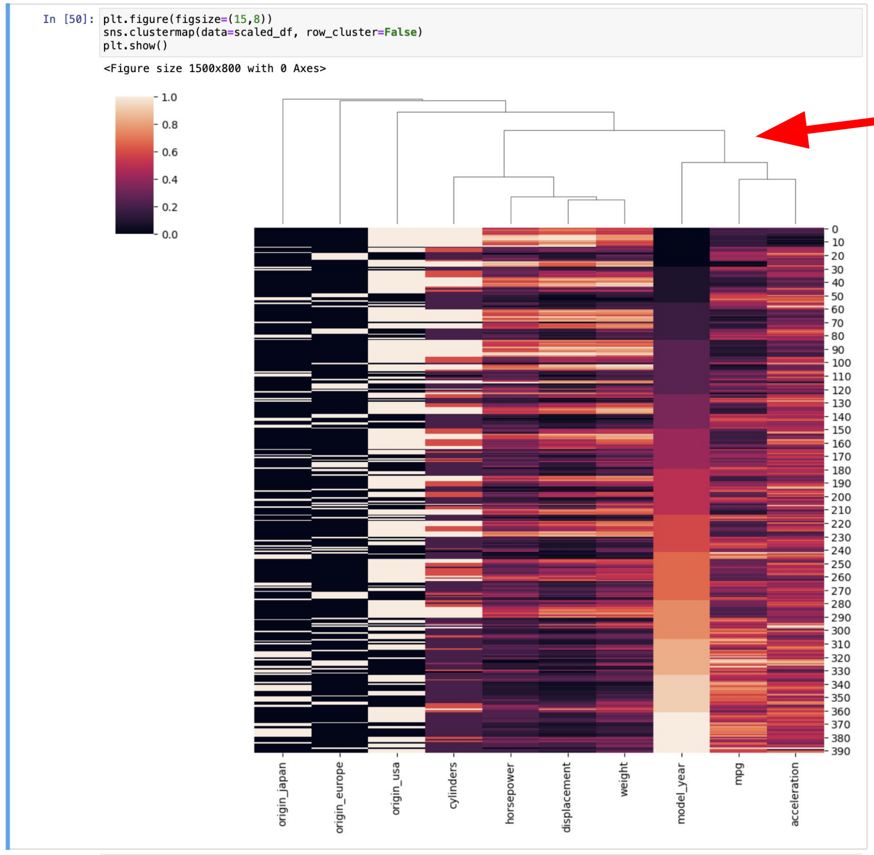
The heatmap Y-axis contain all the rows from the scaled dataset and the X-axis contain all the columns.
The color represent the value of the column for each row in the data set scaled from black (0 value) to white (1 value)

Hierarchical Clustering - Python Example

We can also combine a heatmap with dendrogram by using **clustermap** from Seaborn library.

sns.clustermap() → Provide a heatmap of the provided dataset and combine it with dendrogram.

We can choose to include dendrogram on the rows, columns or both.

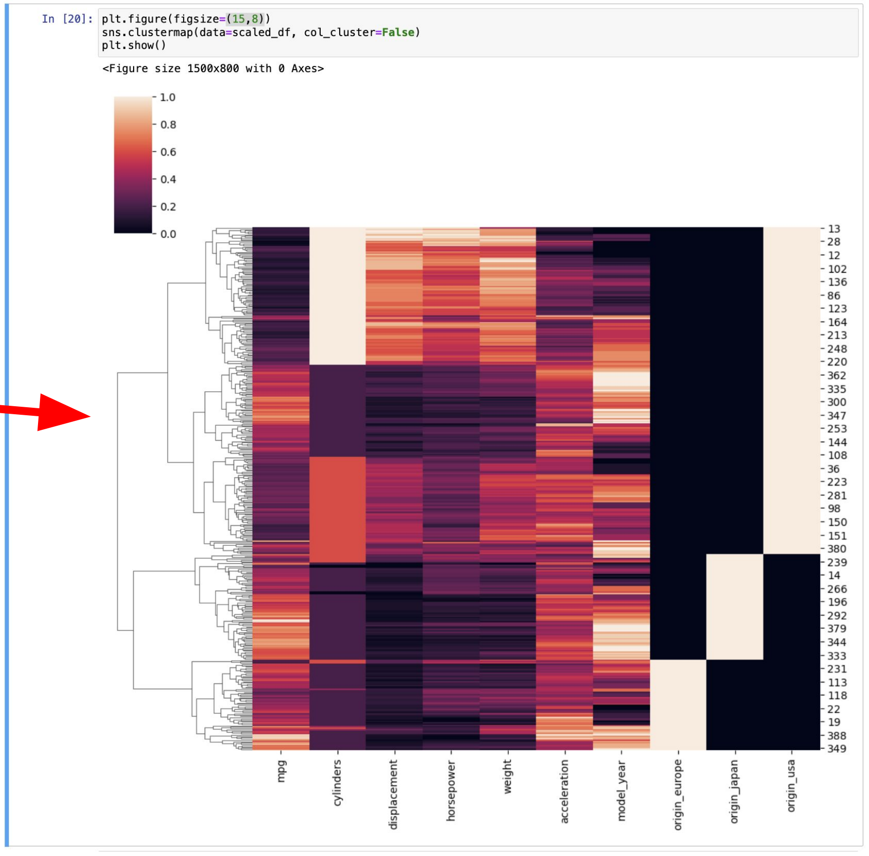


```
In [50]: plt.figure(figsize=(15,8))
         sns.clustermap(data=scaled_df, row_cluster=False)
         plt.show()
```

```
<Figure size 1500x800 with 0 Axes>
```

Dendrogram on the dataset columns

Dendrogram on the dataset rows



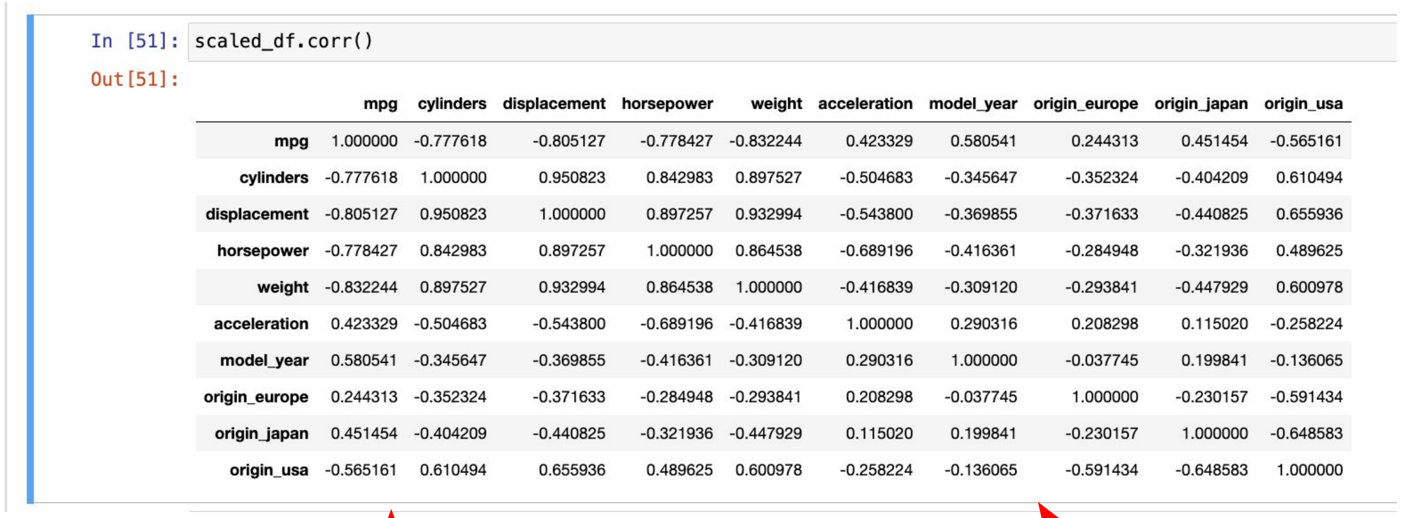
```
In [20]: plt.figure(figsize=(15,8))
         sns.clustermap(data=scaled_df, col_cluster=False)
         plt.show()
```

```
<Figure size 1500x800 with 0 Axes>
```

Hierarchical Clustering - Python Example

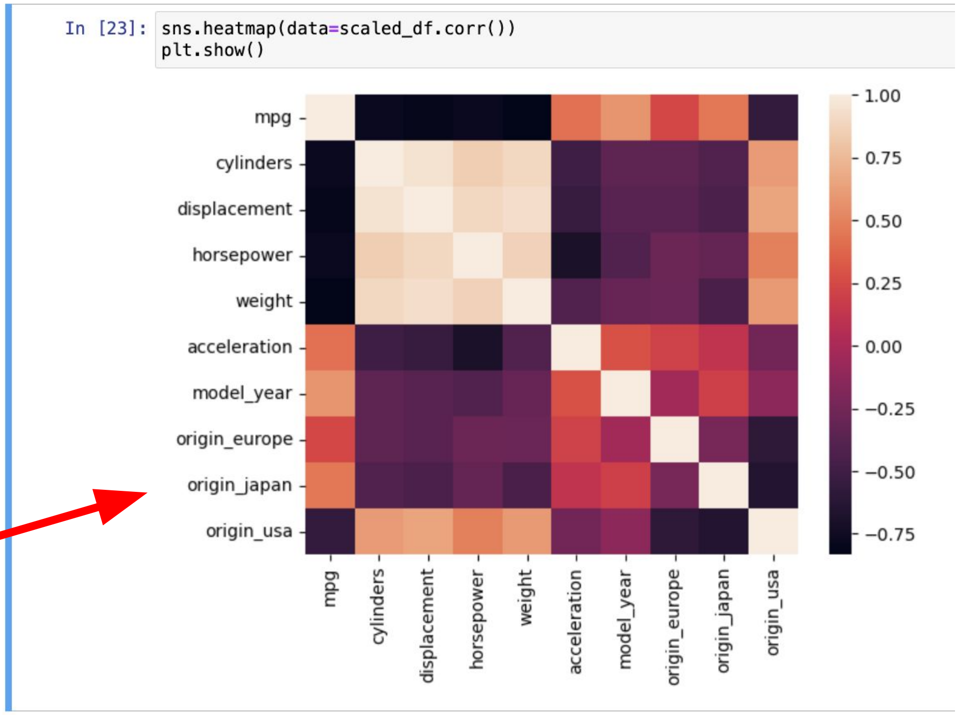
The dendrogram on the dataset columns can help us find correlated features.

This functionality is not directly belong to Hierarchical Clustering but it just use the same dendrogram chart. We can also achieve the same result by using the **df.corr()** method.



```
In [51]: scaled_df.corr()
Out[51]:
```

	mpg	cylinders	displacement	horsepower	weight	acceleration	model_year	origin_europe	origin_japan	origin_usa
mpg	1.000000	-0.777618	-0.805127	-0.778427	-0.832244	0.423329	0.580541	0.244313	0.451454	-0.565161
cylinders	-0.777618	1.000000	0.950823	0.842983	0.897527	-0.504683	-0.345647	-0.352324	-0.404209	0.610494
displacement	-0.805127	0.950823	1.000000	0.897257	0.932994	-0.543800	-0.369855	-0.371633	-0.440825	0.655936
horsepower	-0.778427	0.842983	0.897257	1.000000	0.864538	-0.689196	-0.416361	-0.284948	-0.321936	0.489625
weight	-0.832244	0.897527	0.932994	0.864538	1.000000	-0.416839	-0.309120	-0.293841	-0.447929	0.600978
acceleration	0.423329	-0.504683	-0.543800	-0.689196	-0.416839	1.000000	0.290316	0.208298	0.115020	-0.258224
model_year	0.580541	-0.345647	-0.369855	-0.416361	-0.309120	0.290316	1.000000	-0.037745	0.199841	-0.136065
origin_europe	0.244313	-0.352324	-0.371633	-0.284948	-0.293841	0.208298	-0.037745	1.000000	-0.230157	-0.591434
origin_japan	0.451454	-0.404209	-0.440825	-0.321936	-0.447929	0.115020	0.199841	-0.230157	1.000000	-0.648583
origin_usa	-0.565161	0.610494	0.655936	0.489625	0.600978	-0.258224	-0.136065	-0.591434	-0.648583	1.000000



```
In [23]: sns.heatmap(data=scaled_df.corr())
         plt.show()
```

df.corr() method return a dataframe for the correlation between -1 and 1 of each column with each other column

We can also combine the correlation dataframe with heatmap for better correlation visualization

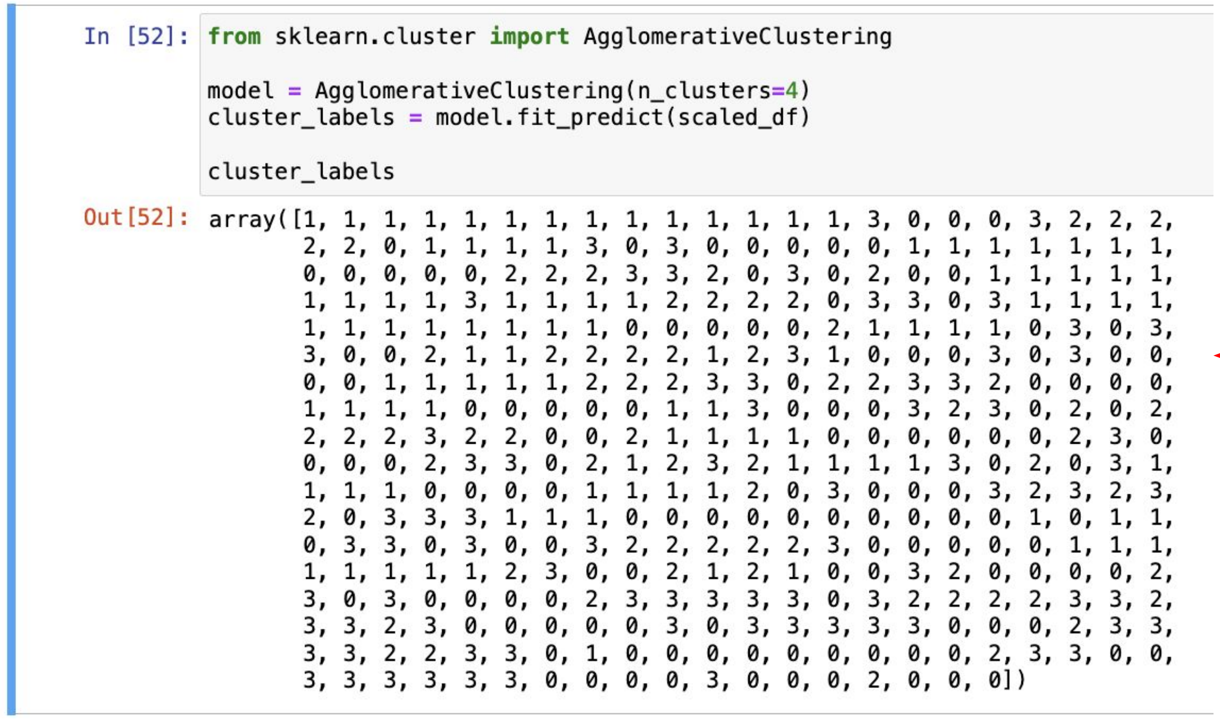
Hierarchical Clustering - Python Example

Now that we explored the dataset, we can now start to train our Hierarchical Clustering model.

When creating the model instance we need to choose the number of clusters it will divide our data to.

The dufault value of clusters is 2 meaning it will perform bottom-up merging until getting 2 final clusters.

In our example we will determine the ‘n_clusters’ parameter to be 4.



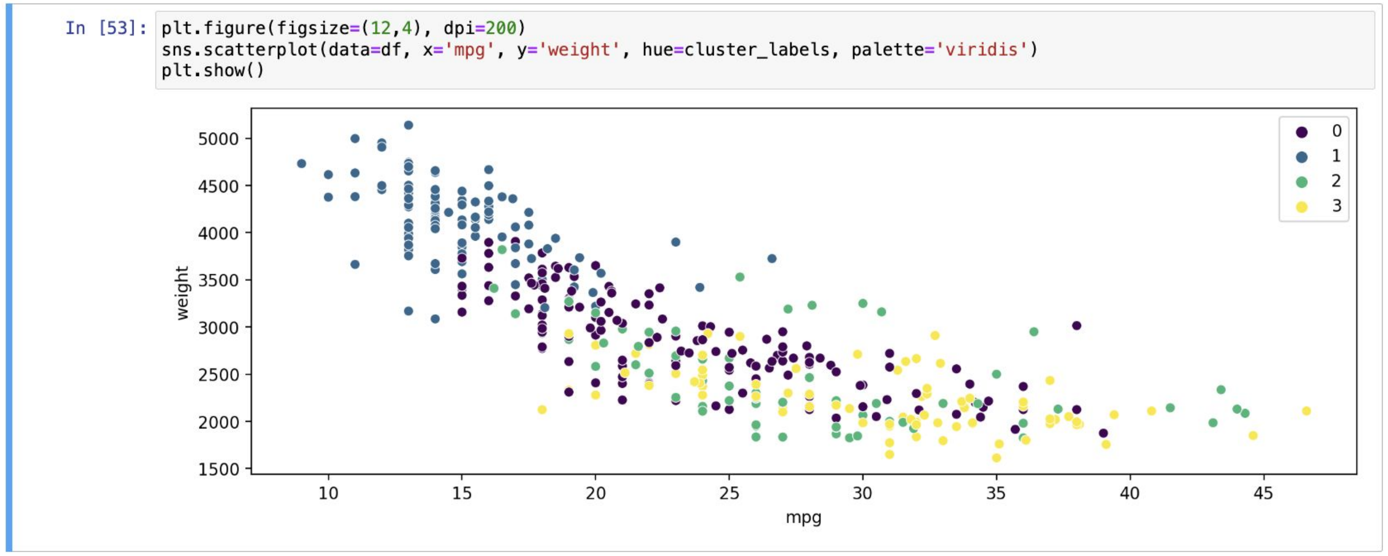
```
In [52]: from sklearn.cluster import AgglomerativeClustering

         model = AgglomerativeClustering(n_clusters=4)
         cluster_labels = model.fit_predict(scaled_df)

         cluster_labels

Out[52]: array([1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 3, 0, 0, 0, 3, 2, 2, 2,
                2, 2, 0, 1, 1, 1, 1, 3, 0, 3, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1,
                0, 0, 0, 0, 0, 2, 2, 2, 3, 3, 2, 0, 3, 0, 2, 0, 0, 1, 1, 1, 1, 1,
                1, 1, 1, 1, 3, 1, 1, 1, 1, 2, 2, 2, 2, 0, 3, 3, 0, 3, 1, 1, 1, 1,
                1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 2, 1, 1, 1, 1, 0, 3, 0, 3,
                3, 0, 0, 2, 1, 1, 2, 2, 2, 2, 1, 2, 3, 1, 0, 0, 0, 3, 0, 3, 0, 0,
                0, 0, 1, 1, 1, 1, 1, 2, 2, 2, 3, 3, 0, 2, 2, 3, 3, 2, 0, 0, 0, 0,
                1, 1, 1, 1, 0, 0, 0, 0, 0, 1, 1, 3, 0, 0, 0, 3, 2, 3, 0, 2, 0, 2,
                2, 2, 2, 3, 2, 2, 0, 0, 2, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 2, 3, 0,
                0, 0, 0, 2, 3, 3, 0, 2, 1, 2, 3, 2, 1, 1, 1, 1, 3, 0, 2, 0, 3, 1,
                1, 1, 1, 0, 0, 0, 0, 1, 1, 1, 1, 2, 0, 3, 0, 0, 0, 3, 2, 3, 2, 3,
                2, 0, 3, 3, 3, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1,
                0, 3, 3, 0, 3, 0, 0, 3, 2, 2, 2, 2, 2, 3, 0, 0, 0, 0, 0, 1, 1, 1,
                1, 1, 1, 1, 1, 2, 3, 0, 0, 2, 1, 2, 1, 0, 0, 3, 2, 0, 0, 0, 0, 2,
                3, 0, 3, 0, 0, 0, 0, 2, 3, 3, 3, 3, 3, 0, 3, 2, 2, 2, 2, 3, 3, 2,
                3, 3, 2, 3, 0, 0, 0, 0, 0, 3, 0, 3, 3, 3, 3, 3, 0, 0, 0, 2, 3, 3,
                3, 3, 2, 2, 3, 3, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 3, 3, 0, 0,
                3, 3, 3, 3, 3, 3, 0, 0, 0, 0, 3, 0, 0, 0, 2, 0, 0, 0])
```

We can see the labels our model gave for each datapoint in our dataset.
We can see that in total there are 4 different labels.

Hierarchical Clustering - Python Example

Remember, because this is unsupervised learning model the labels are just different groups that contain different data points. we don't know from the model what those groups actually represent.
However, what we can do is use those label predictions in order to try and understand the group meaning.

For example → we can create a scatterplot of correlated features and use the predicted labels as hue.



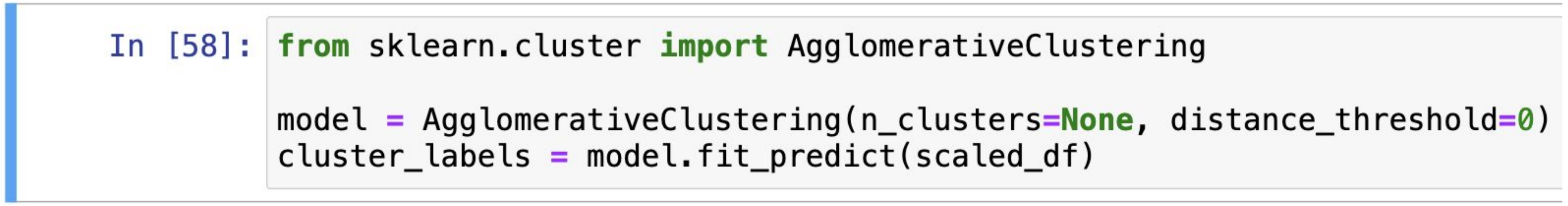
```
In [53]: plt.figure(figsize=(12,4), dpi=200)
         sns.scatterplot(data=df, x='mpg', y='weight', hue=cluster_labels, palette='viridis')
         plt.show()
```

Hierarchical Clustering - Python Example

Another model parameter we can use is ‘distance_threshold’ which determine the maximum distance that can have between different clusters in the hierarchy.

The ‘distance_threshold’ parameter can’t work in parallel with the ‘n_clusters’ parameter, so in order to use it we need to pass n_clusters = None.

For example → Try to understand how many clusters should we get if we pass distance_threshold = 0?



```
In [58]: from sklearn.cluster import AgglomerativeClustering

         model = AgglomerativeClustering(n_clusters=None, distance_threshold=0)
         cluster_labels = model.fit_predict(scaled_df)
```

ECOM SCHOOL
המכללה למקצועות הדיגיטל וההייטק

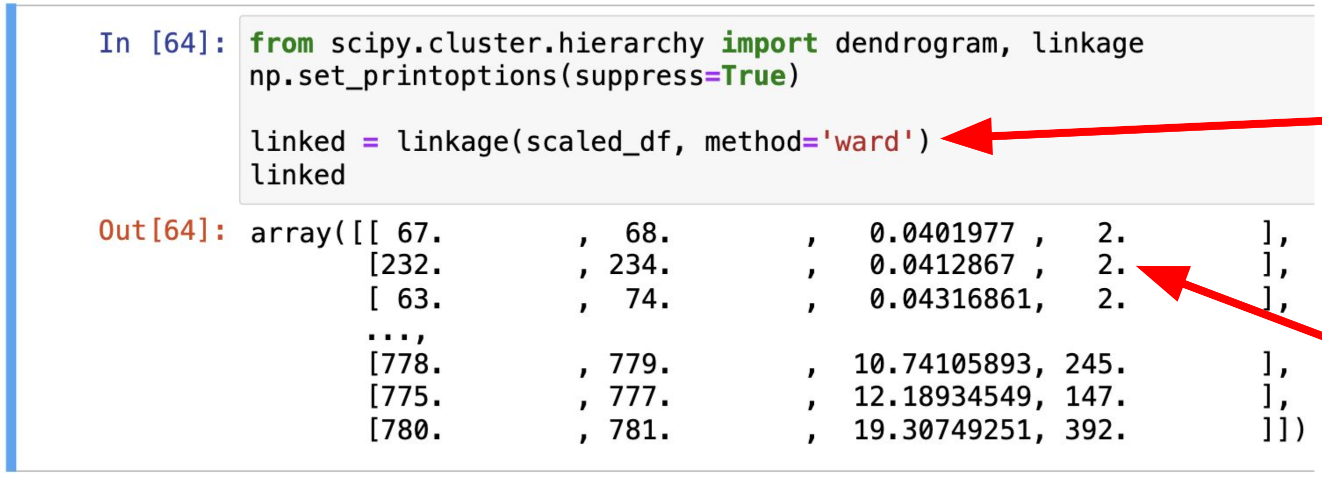
Hierarchical Clustering - Visualization

In order to visualize the Hierarchical Clustering itself we can use the **linkage()** function from Scikit-Learn.

Linkage() → A function that perform Hierarchical Clustering modeling on the given dataframe.

By default the linkage function use euclidean distance calculation and ‘single’ as the linkage method.

The result of the Linkage() function is the linkage matrix which is a 2D array that contain for each cluster merge the number of datapoints that been merged to the cluster and the distance between those points.



```
In [64]: from scipy.cluster.hierarchy import dendrogram, linkage
         np.set_printoptions(suppress=True)

         linked = linkage(scaled_df, method='ward')
         linked

Out[64]: array([[ 67.        ,  68.        ,   0.0401977 ,   2.        ],
                [232.        , 234.        ,   0.0412867 ,   2.        ],
                [ 63.        ,  74.        ,   0.04316861,   2.        ],
                ...,
                [778.        , 779.        ,  10.74105893, 245.        ],
                [775.        , 777.        ,  12.18934549, 147.        ],
                [780.        , 781.        ,  19.30749251, 392.        ]])
```

Here we use the ‘ward’ method which calculate the distance between the cluster centroids

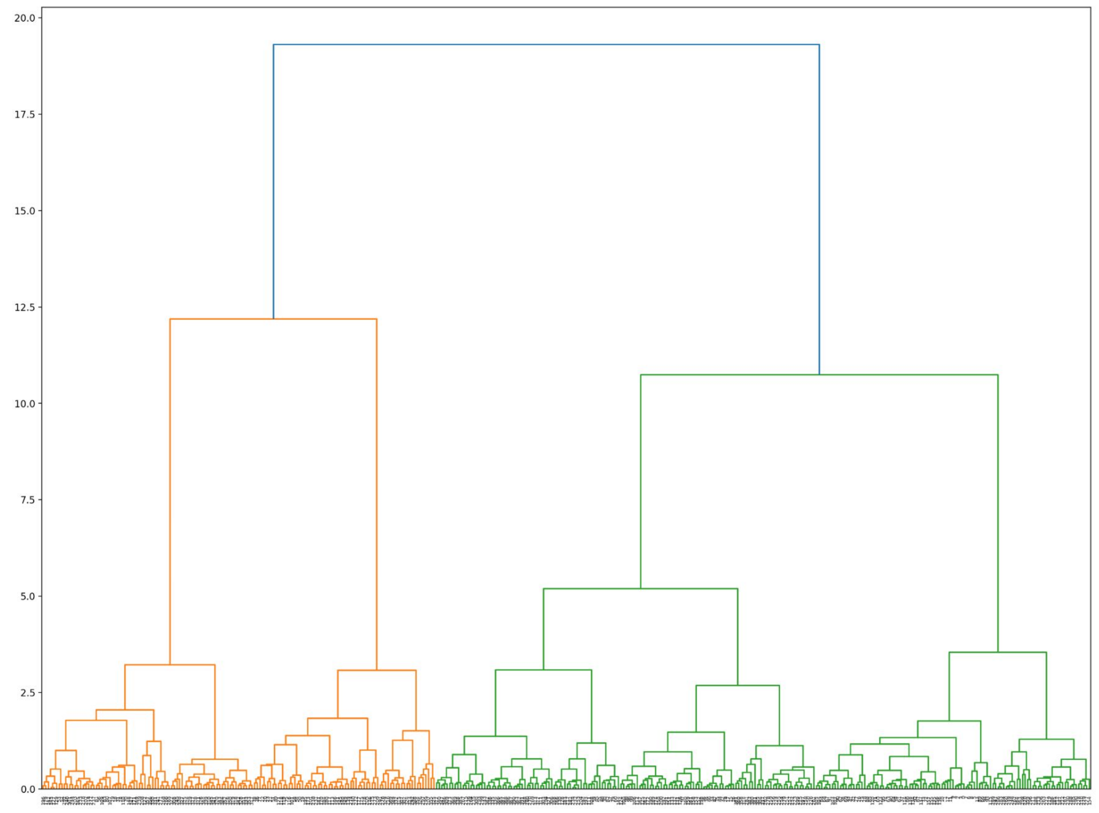
According the the linkage matrix, the model first merged data points in indexes 67 and 68, and the distance between those data points was 0.0402

ECOM SCHOOL
המכללה למקצועות הדיגיטל וההייטק

Hierarchical Clustering - Visualization

Once we have the linkage matrix we can also visualize the diandogram itself:

```
In [74]: from scipy.cluster.hierarchy import dendrogram, linkage

         plt.figure(figsize=(20,15), dpi=200)
         dendrogram(linked,
                    orientation='top',
                    labels=range(0, len(scaled_df)),
                    distance_sort='descending')
         plt.show()
```



Class Exercise - Hierarchical Clustering

Instructions:

Use the **'iris_unlabled.csv'** file and implement the following:

- Apply feature scaling using MinMaxScaler, apply other data preparation actions if needed.
- Generate a clustermap for the entire dataset and visualize a dendrogram of the columns and rows.
- Use the column dendrogram from the previous exercise and identify 2 features that have the highest correlation with each other. Print the correlation value of those 2 features.
- Generate a linkage matrix with 'ward' method and visualize the dendrogram as a plot
- Train a Hierarchical Clustering model with n_clusters = 3 and visualize the cluster prediction on the dataset with scatterplot (each cluster should have it's own unique color in the scatterplot).

ECOM SCHOOL
המכללה למקצועות הדיגיטל וההייטק

Class Exercise Solution - Hierarchical Clustering

DBSCAN - Introduction

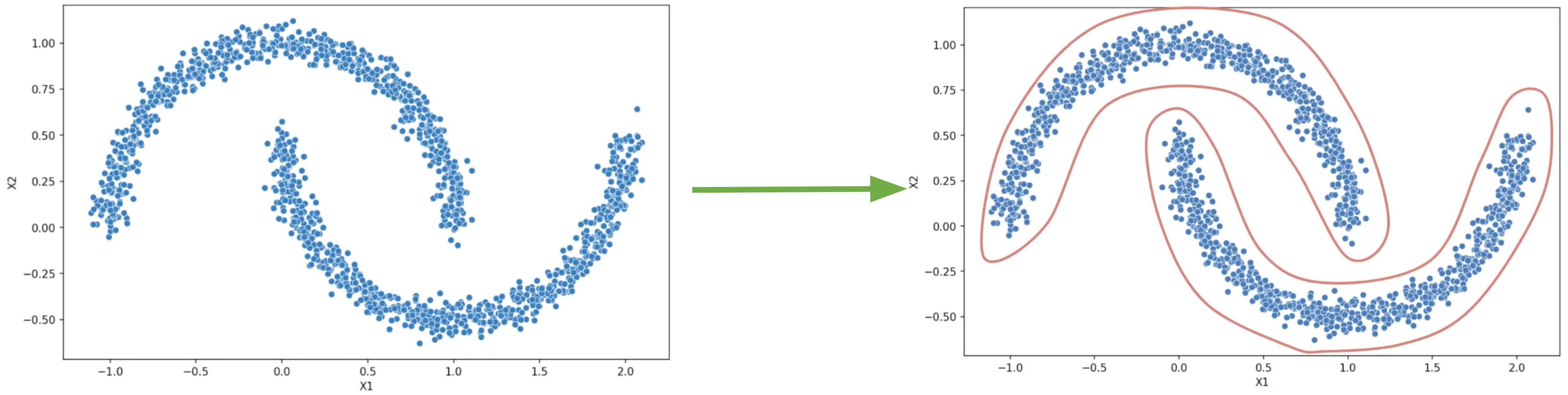
DBSCAN → DBSCAN algorithm stand for Density-Based Spatial Clustering of Applications with Noise. This is a widely-used clustering algorithm in data mining and machine learning. It is particularly effective for discovering clusters of arbitrary shape in datasets containing noise and outliers.
DBSCAN clusters data points based on their density, which makes it fundamentally different from centroid-based clustering algorithms like K-Means Clustering or Hierarchical Clustering.

Why should we use density based clustering model?
The main problem with the distance based models is that they can’t identify clusters that are not in spherical shapes. Meaning that in case our data points combine from an arbitrary shapes the distance based models won’t be able to effectively recognize the different clusters.
In addition, density based clustering models knows how to identify outliers and noise.

DBSCAN - Introduction

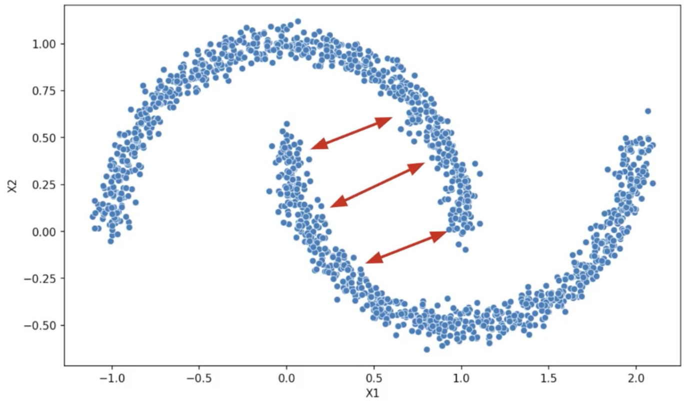
For example → Let’s consider the following scatterplot as our dataset.

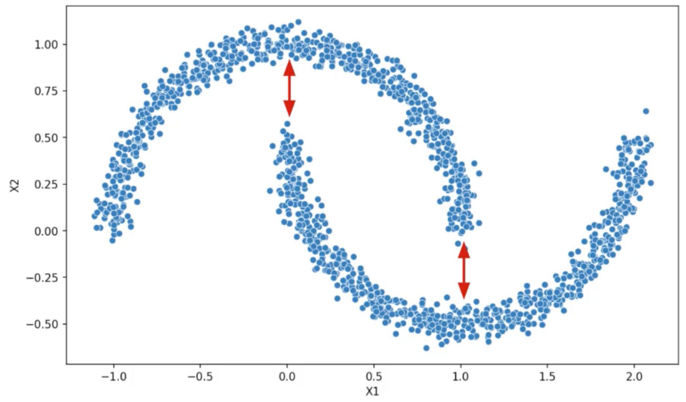
We as humans can easily spot the two main clusters and divide the data points accordingly

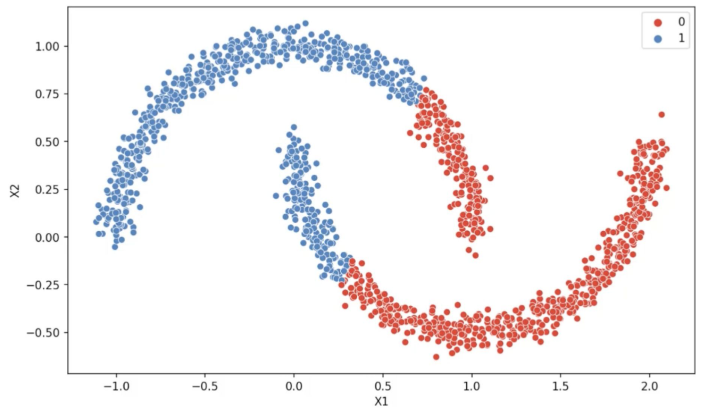


DBSCAN - Introduction

The problem is that the distance based algorithms will not be able to identify the two clusters because the distance between the points is changing within the cluster itself.





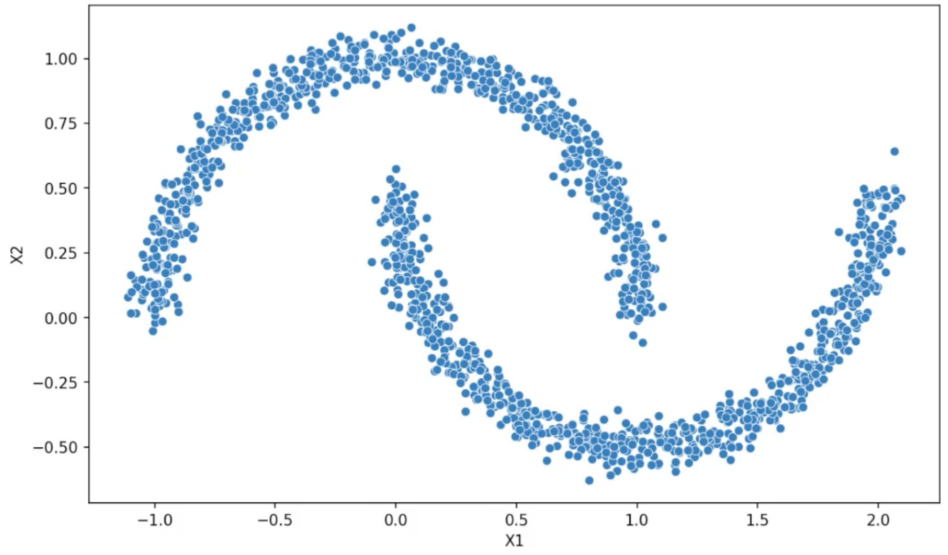


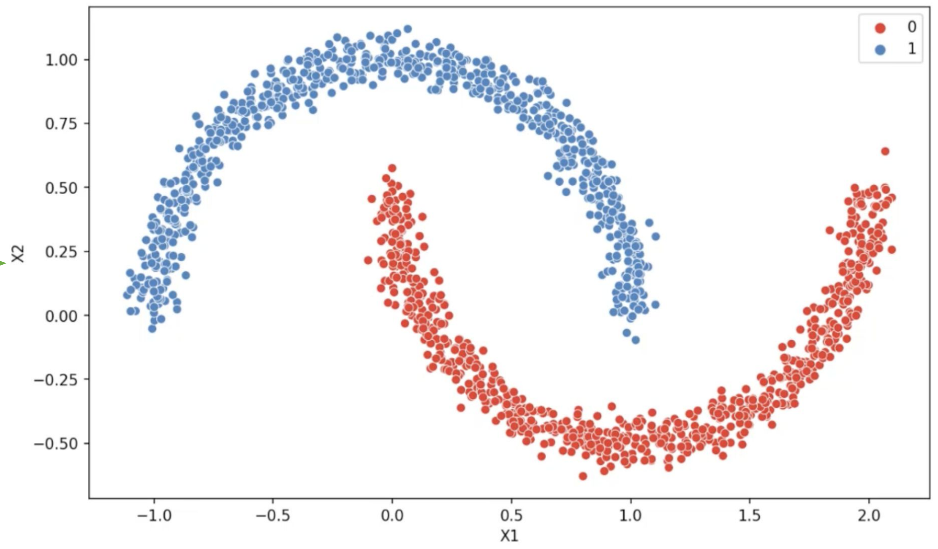
Distance based clustering models outcome

ECOM SCHOOL
המכללה למקצועות הדיגיטל וההייטק

DBSCAN - Introduction

However, if we will apply density based algorithm (like DBSCAN) on the same dataset the result will be the following:





DBSCAN - Key Concepts

Before we try to understand how DBSCAN algorithm is working, we first need to be familiar with some key concepts regarding the DBSCAn algorithm:

- **Epsilon (ε)** → A parameter that determine the maximum radius of the neighborhood around a point.
- **Minimum number of points** → A parameter that determine The minimum number of points required to form a dense region (a cluster).
- **Density Reachability** → A point ‘p’ is directly density-reachable from a point ‘q’ if ‘p’ is within a certain distance ε (epsilon) from ‘q’, and ‘q’ has enough nearby points (at least a minimum number of points) within its ε-radius.
- **Density Connectivity (core point)** → A point ‘p’ is density-connected to a point ‘q’ if there is a sequence of points p1, p2,…, pn where p1 = p and pn = q and each point in the sequence is directly density-reachable from the next point in the sequence.
- **Border point** → A point ‘p’ is a border point if it density-reachable from a point ‘q’ but doesn’t have enough number of points to be considered core point.
- **Outlier point** → A point ‘p’ is an outlier point if it can’t be consider as density reachable to any cluster

DBSCAN - Theory

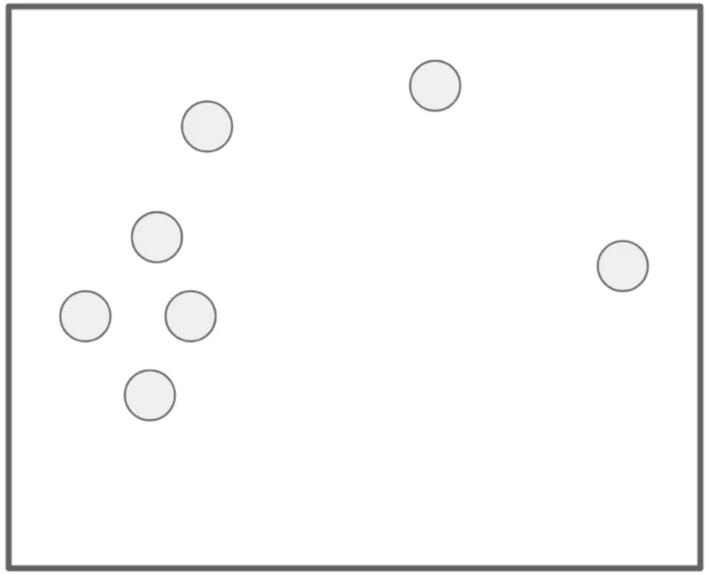
Now that we understand the key concepts of the density based algorithm we can discuss on how the DBSCAN algorithm is working:

1. Label all data points as unvisited.
2. Select a random unvisited data point. If it contains more than MinPts points within its ε-radius, a new cluster is started.
3. Expand the cluster by iteratively adding all directly density-reachable points to the cluster. This process continues until no new points can be added.
4. Mark visited points as part of the cluster. If a point does not have enough neighbors to form a cluster, it is labeled as noise.
5. Repeat the process until all points are visited.

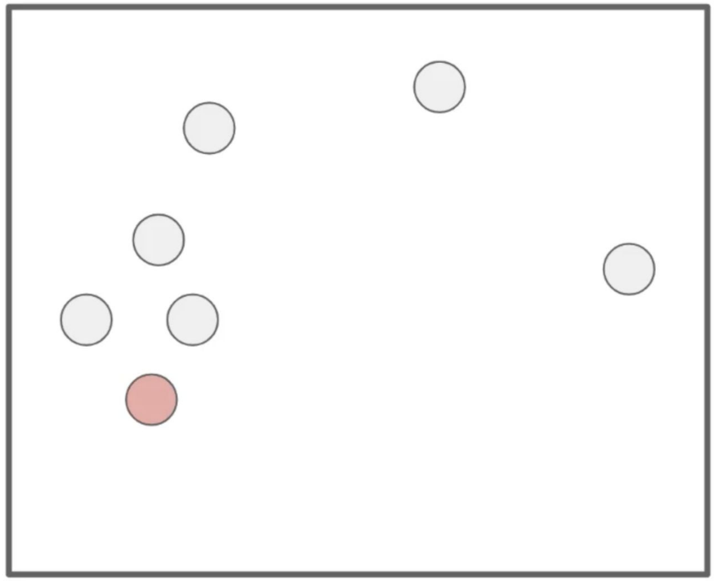
DBSCAN - Theory

Let’s visualize the algorithm process in order to better understand each step.

For the example we will choose the following parameters → **ε = 1** and **Min points = 3**



Step 1 - All data points are mark with as unvisited

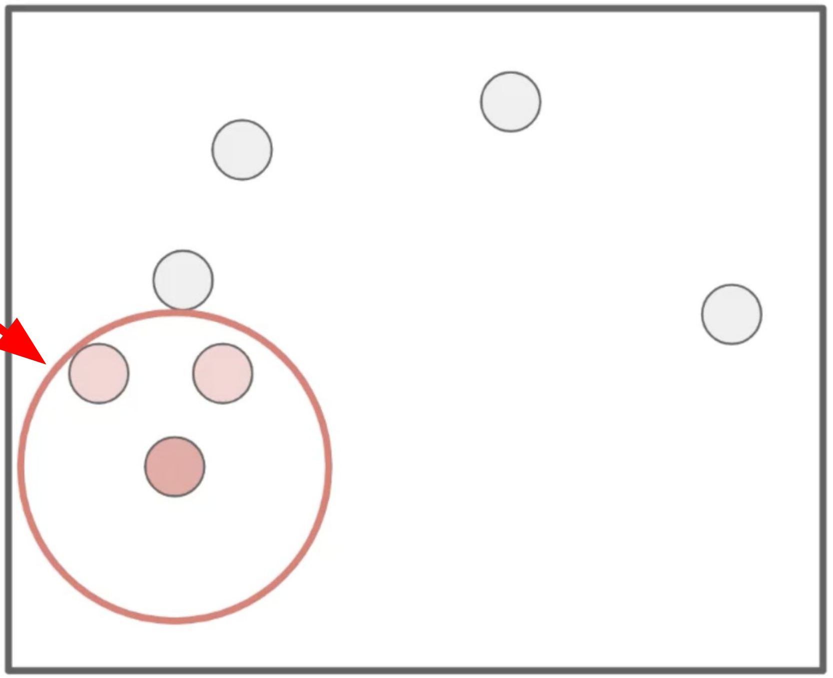


Step 2 - The algorithm randomly select a starting data point

DBSCAN - Theory

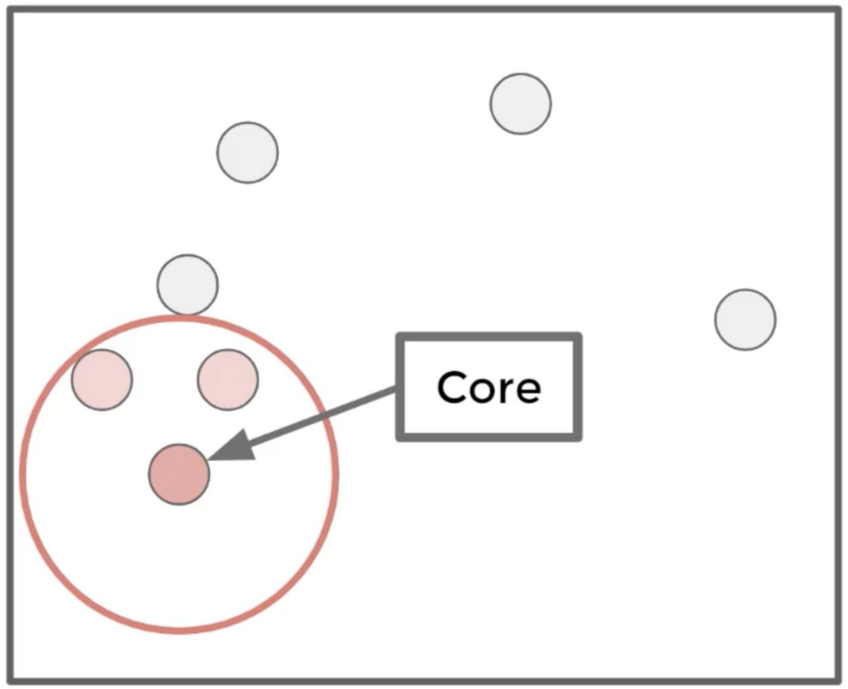
ε = 1 Min Points = 3

The ε radius



Step 3 - The algorithm try to expand the cluster by reaching to all points within the ε radius

ε = 1 Min Points = 3

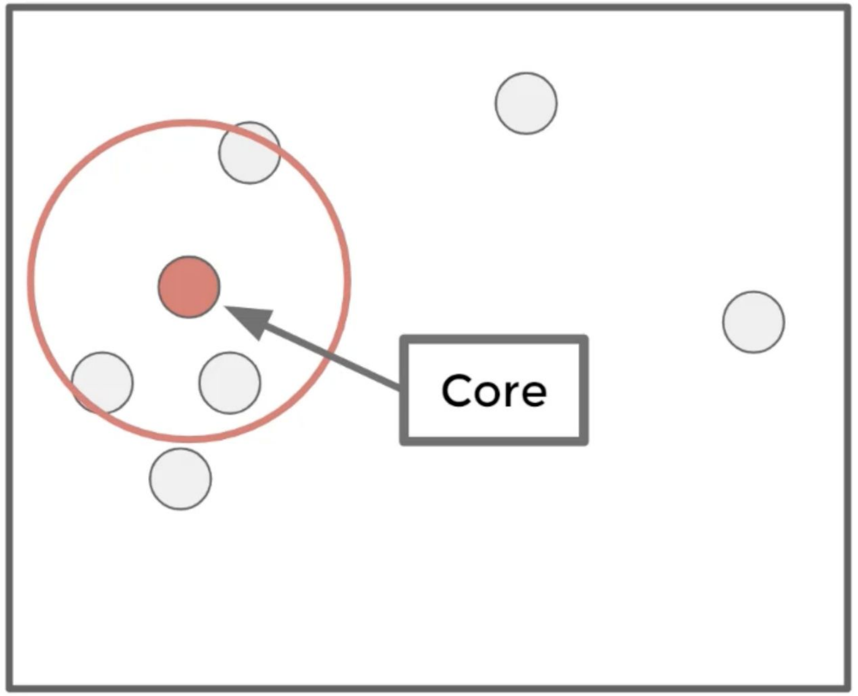


The point consider a core point because it has enough points within the ε radius.
Min Points = 3 and there are 3 points in total within the ε radius including the core point itself.

ECOM SCHOOL
המכללה למקצועות הדיגיטל וההייטק

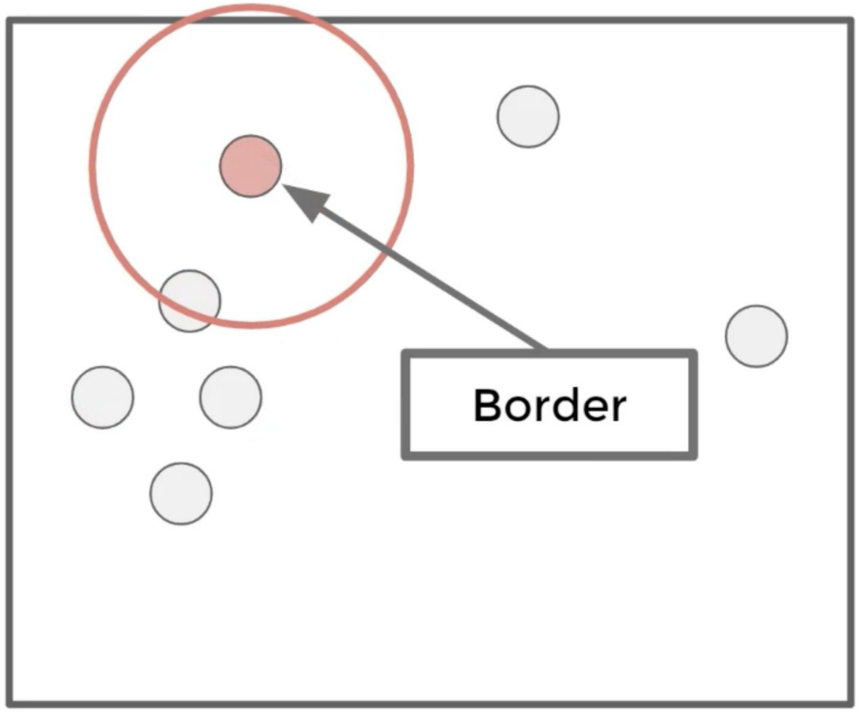
DBSCAN - Theory

ε = 1 Min Points = 3



The following data point also consider as core point for the same reason

ε = 1 Min Points = 3

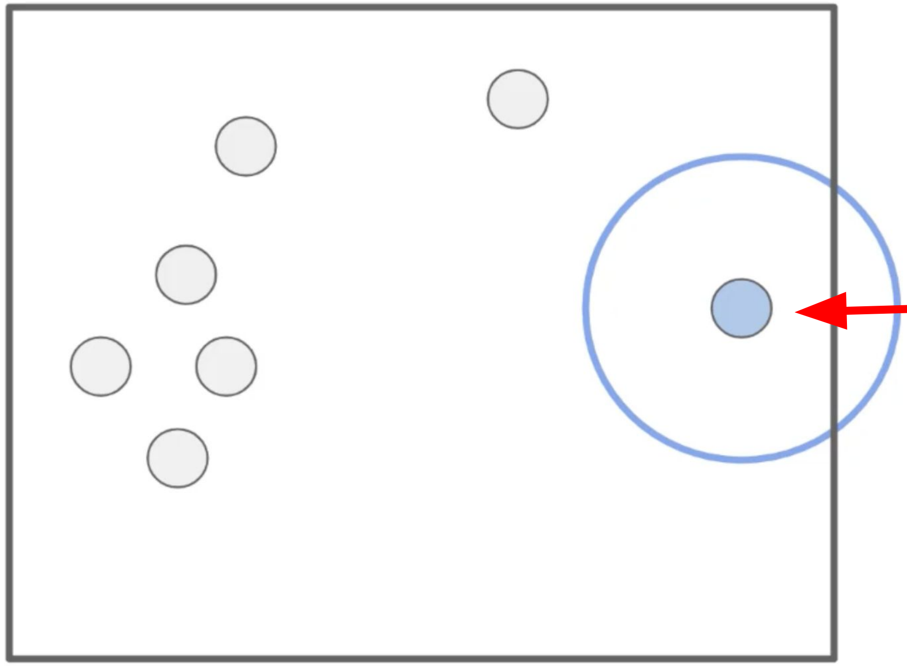


The following data point consider as border point because its has Density Reachability to the other points in the cluster but not has enough data points within its ε radius

ECOM SCHOOL
המכללה למקצועות הדיגיטל וההייטק

DBSCAN - Theory

ε = 1 Min Points = 3



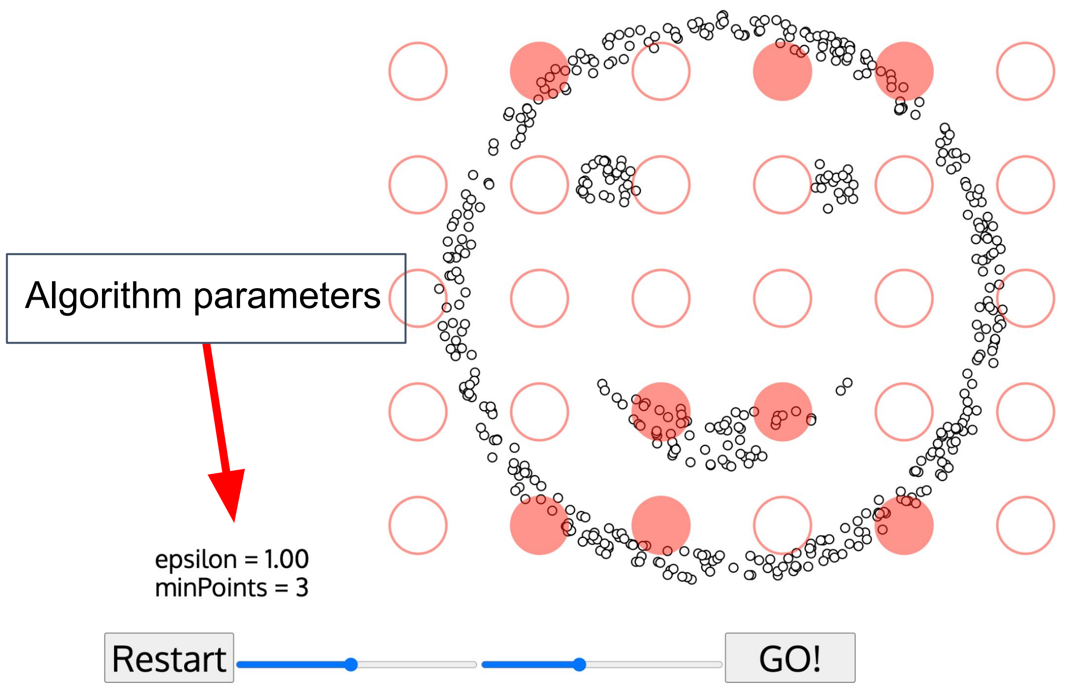
This point will consider as outlier point because it can't be reached by any other point from any cluster

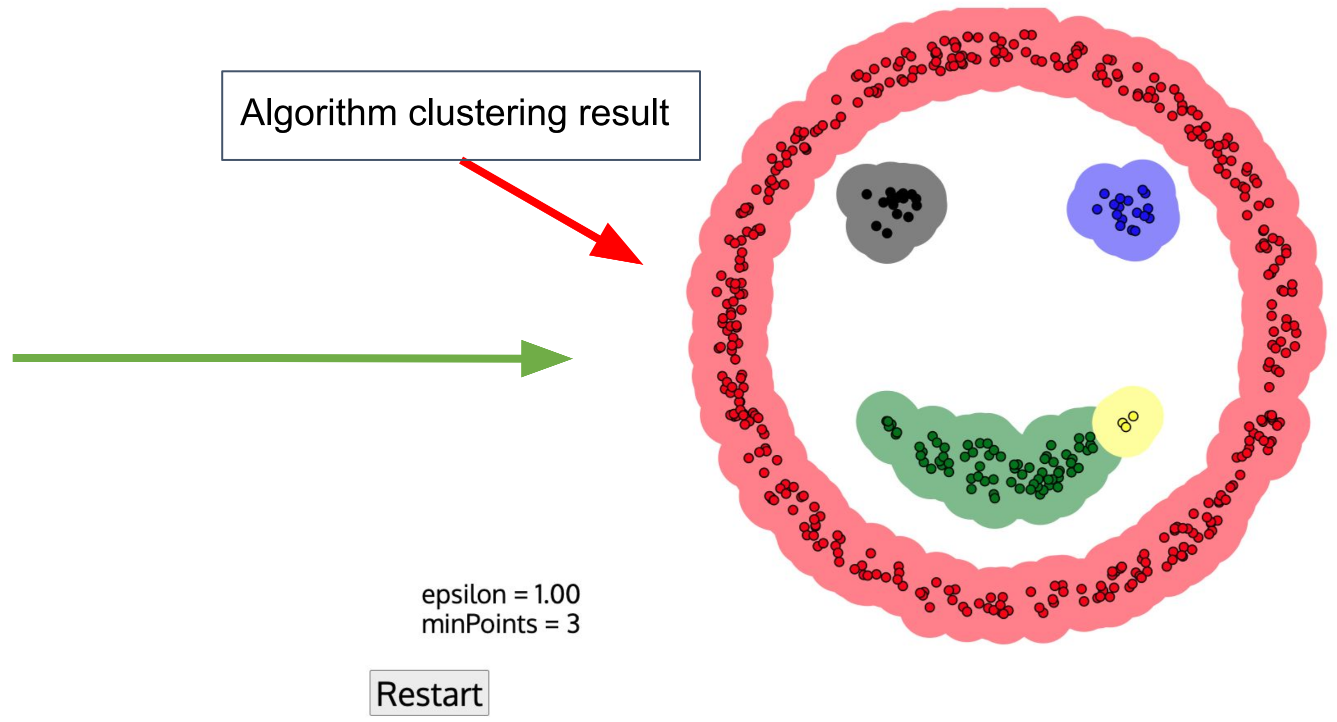
The assignment process will continue until all points been marked as visited and labeled as core / border / outlier.

DBSCAN - Visualization

We can visit the following free website to see the DBSCAN algorithm in action as a simulation:

https://www.naftaliharris.com/blog/visualizing-dbscan-clustering/ (Choose Smiley Face)





ECOM SCHOOL
המכללה למקצועות הדיגיטל וההייטק