K-MEANS CLUSTERING

simplilearn

What is K-Means Clustering?

What is K-Means Clustering?

What is K-Means Clustering?

What is K-Means Clustering?

k-means performs division of objects into clusters which are “similar” between them and are “dissimilar” to the objects belonging to another cluster

simplilearn

What is K-Means Clustering?

Can you explain this with an example?

What is K-Means Clustering?

Can you explain this with an example?

Sure. For understanding K-Means in a better way, let's take an example of **Cricket**

simplilearn

What is K-Means Clustering?

Task: Identify bowlers and batsmen

What is K-Means Clustering?

Task: Identify bowlers and batsmen

- The data contains runs and wickets gained in the last 10 matches
- So, the bowler will have more wickets and the batsmen will have higher runs

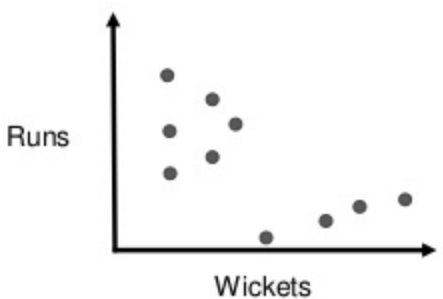
simplilearn

What is K-Means Clustering?

Assign data points

Here, we have our dataset with x and y coordinates

Now, we want to cluster this data using **K-Means**

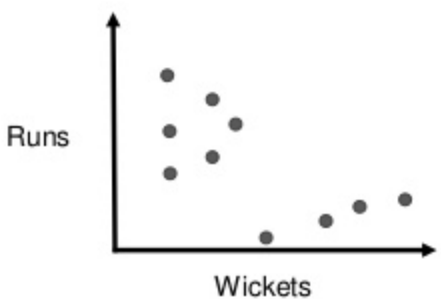


What is K-Means Clustering?

Assign data points

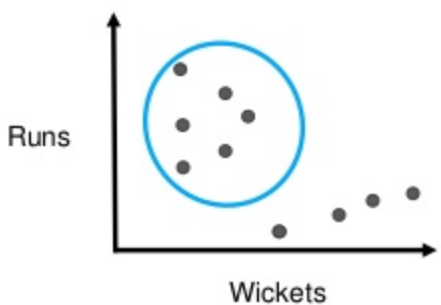
Here, we have our dataset with x and y coordinates

Now, we want to cluster this data using **K-Means**



Cluster 1

We can see that this cluster has players with high runs and low wickets

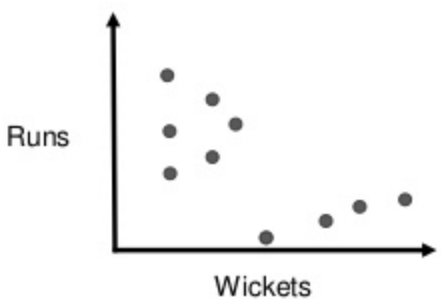


What is K-Means Clustering?

Assign data points

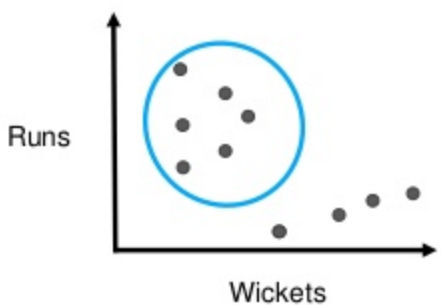
Here, we have our dataset with x and y coordinates

Now, we want to cluster this data using **K-Means**



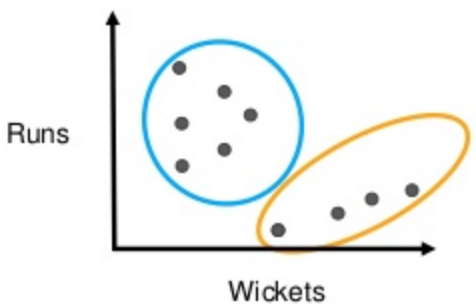
Cluster 1

We can see that this cluster has players with high runs and low wickets



Cluster 2

And here, we can see that this cluster has players with high wickets and low wickets



simplilearn

What is K-Means Clustering?

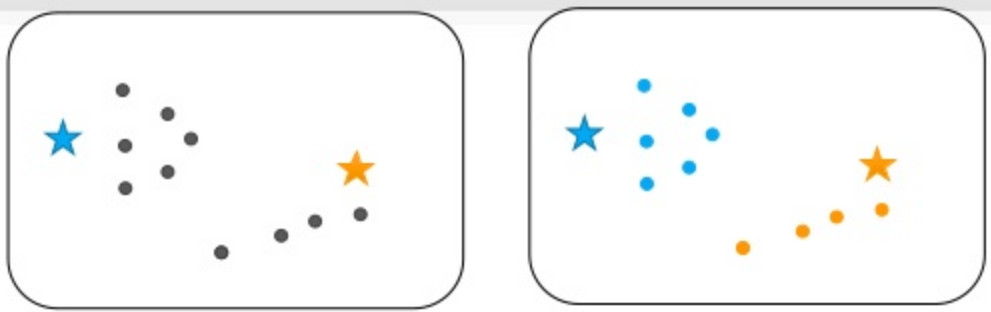
Consider the same data set of cricket

Solve the problem using K-Means

What is K-Means Clustering?

Initially, two centroids are assigned randomly

Euclidean distance to find out which centroid is closest to each data point and the data points are assigned to the corresponding centroids

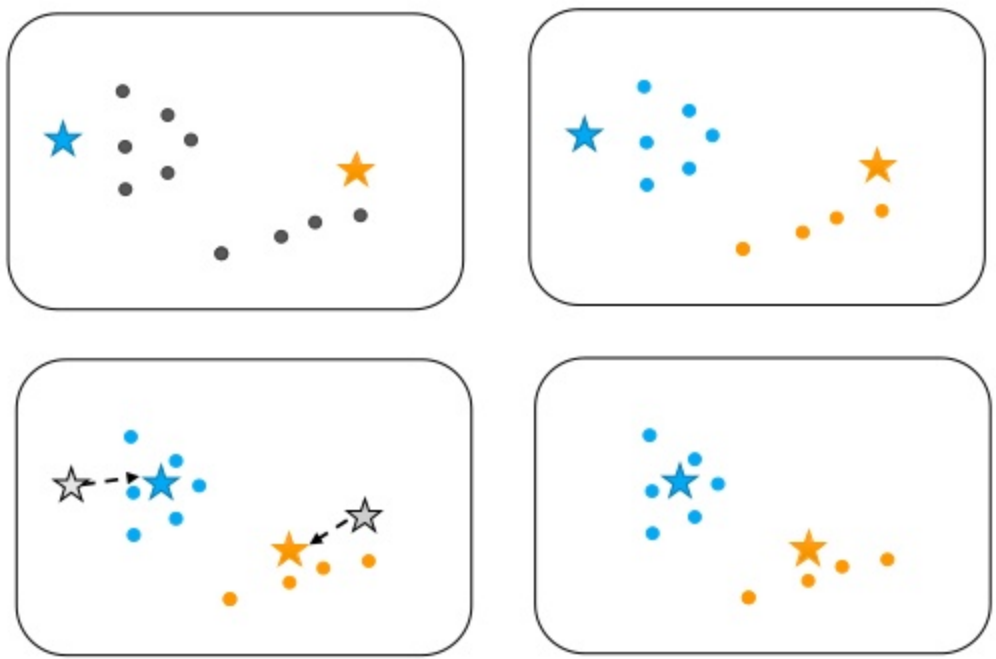


What is K-Means Clustering?

Reposition the two centroids for optimization.

What is K-Means Clustering?

The process is iteratively repeated until our centroids become static



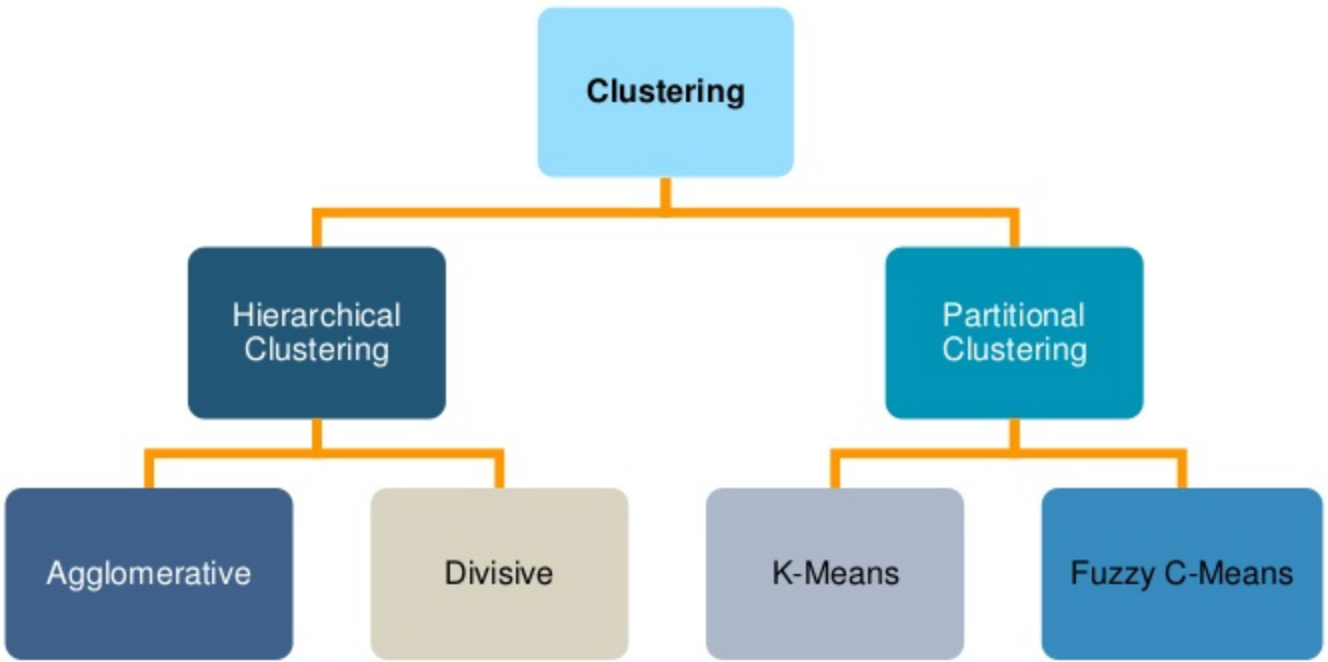
simplilearn

What's in it for you?

- Types of Clustering
- What is K-Means Clustering?
- Applications of K-Means clustering
- Common distance measure
- How does K-Means clustering work?
- K-Means Clustering Algorithm
- Demo: K-Means Clustering
- Use Case: Color Compression

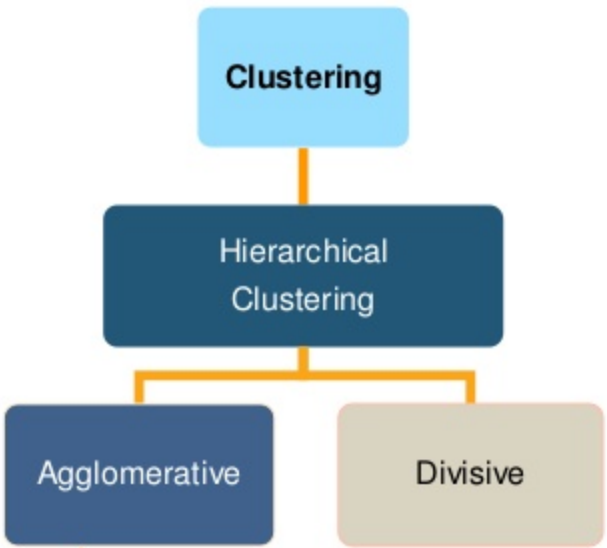
simplilearn

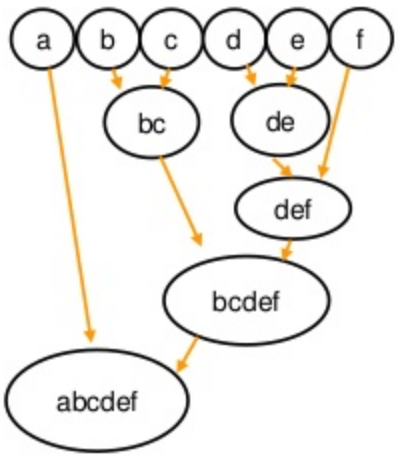
Types of Clustering



Types of Clustering

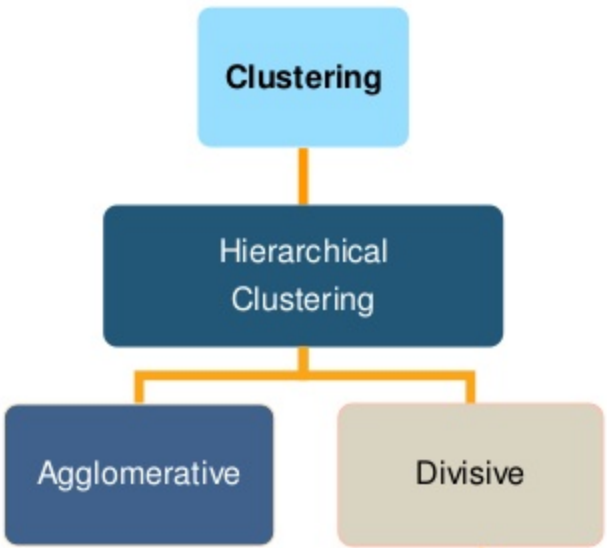
Types of Clustering

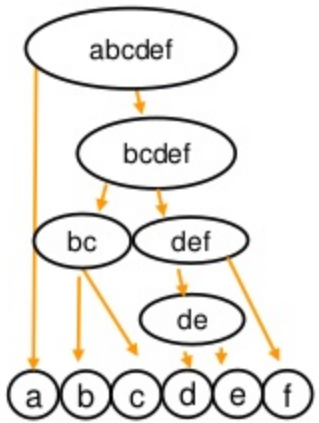




"Bottom up" approach: Begin with each element as a separate cluster and merge them into successively larger clusters

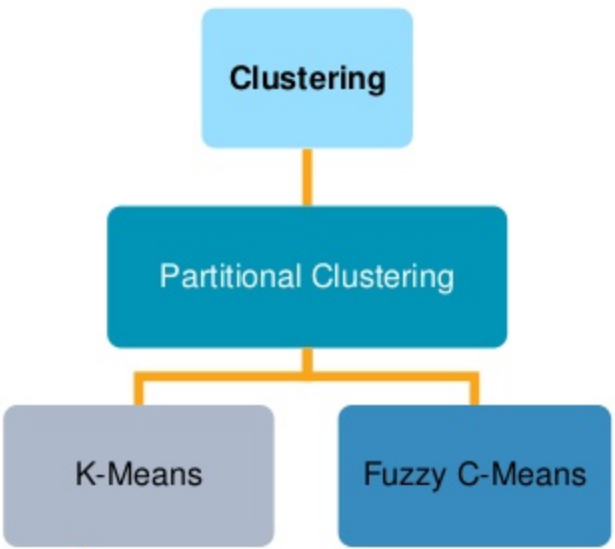
Types of Clustering

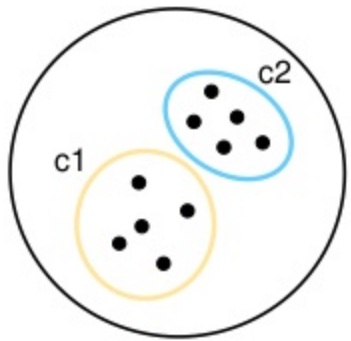




"Top down" approach begin with the whole set and proceed to divide it into successively smaller clusters.

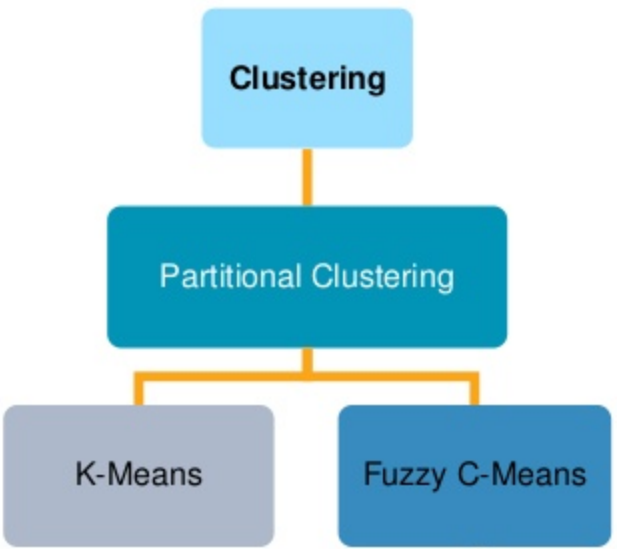
Types of Clustering

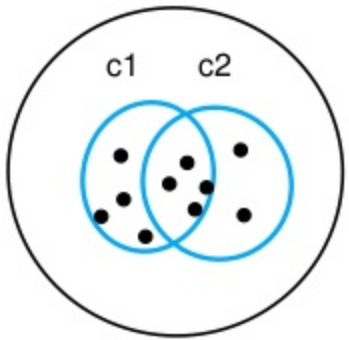




Division of objects into clusters such that each object is in exactly one cluster, not several

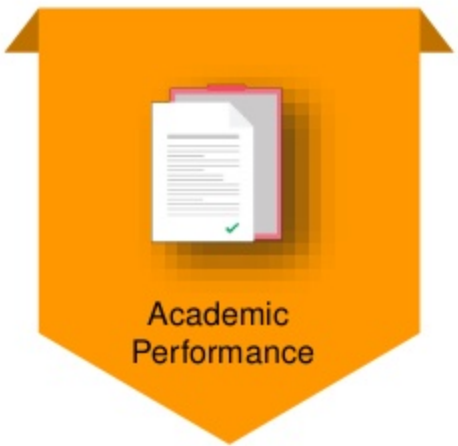
Types of Clustering



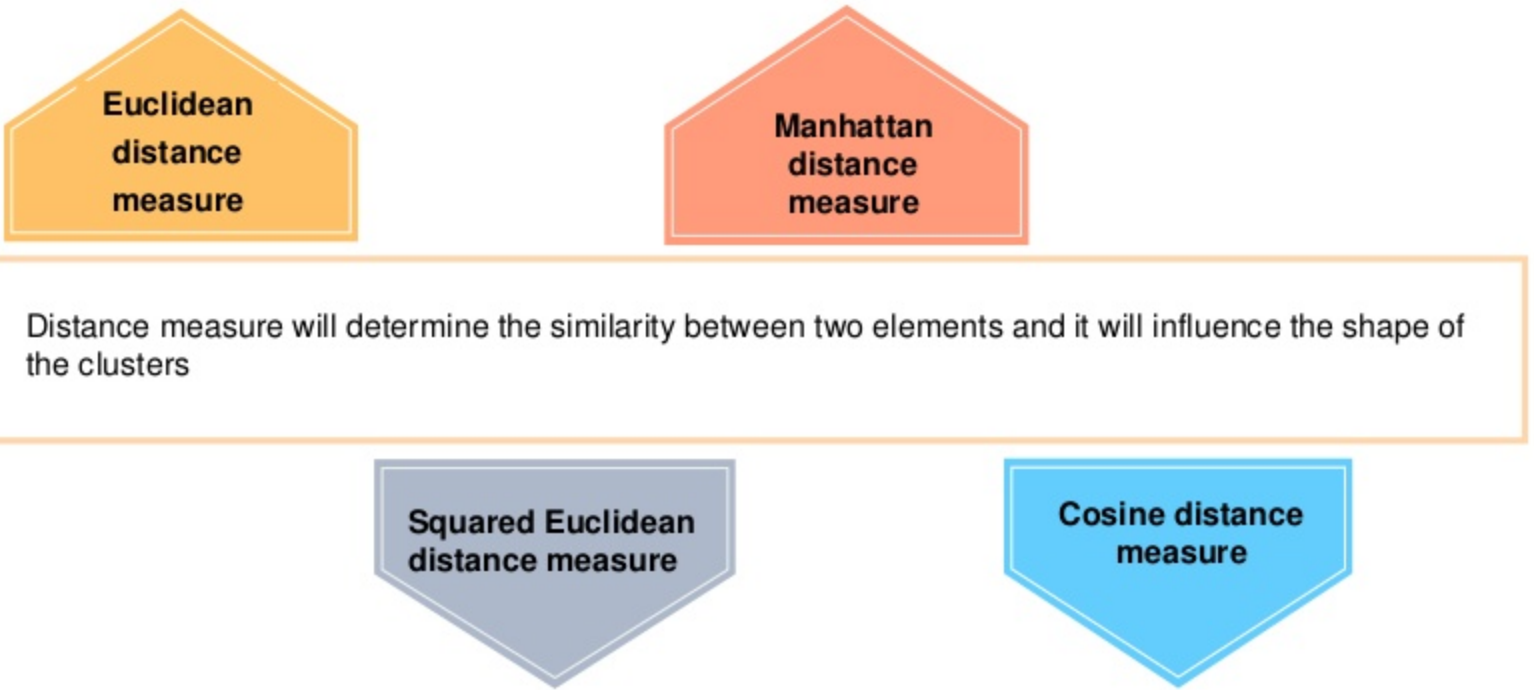


Division of objects into clusters such that each object can belong to multiple clusters

Applications of K-Means Clustering



Distance Measure



Euclidean Distance Measure

01 **Euclidean distance measure**

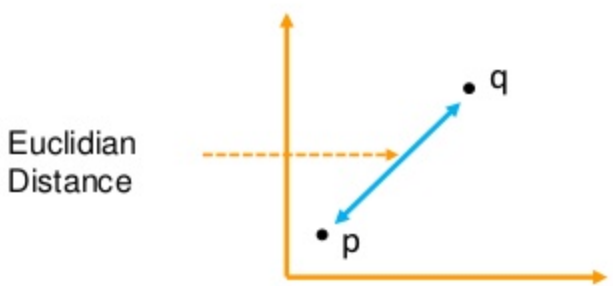
02 Squared euclidean distance measure

03 Manhattan distance measure

04 Cosine distance measure

- The Euclidean distance is the "ordinary" straight line
- It is the distance between two points in Euclidean space

$$\mathbf{d}=\sqrt{\sum_{i=1}^{n}(q_i-p_i)^2}$$



Squared Euclidean Distance Measure

01 **Euclidean distance measure**

02 Squared euclidean distance measure

03 Manhattan distance measure

04 Cosine distance measure

The **Euclidean squared distance** metric uses the same equation as the **Euclidean distance** metric, but does not take the square root.

$$\mathbf{d} = \sum_{i=1}^{n} (q_i - p_i)^2$$

Manhattan Distance Measure

01 **Euclidean distance measure**

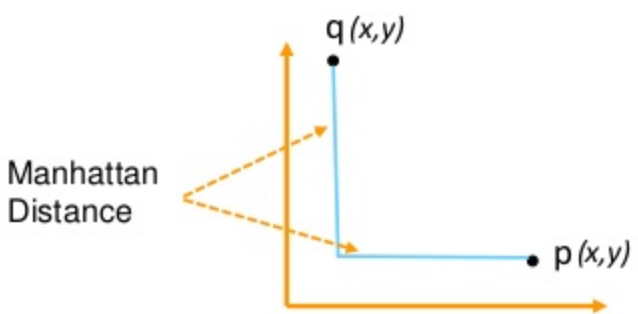
02 Squared euclidean distance measure

03 Manhattan distance measure

04 Cosine distance measure

The Manhattan distance is the simple sum of the horizontal and vertical components or the distance between two points measured along axes at right angles

$$\mathbf{d}= \sum_{i=1}^{n} | q_x - p_x | + | q_x - p_y |$$



simplilearn

Cosine Distance Measure

01 **Euclidean distance measure**

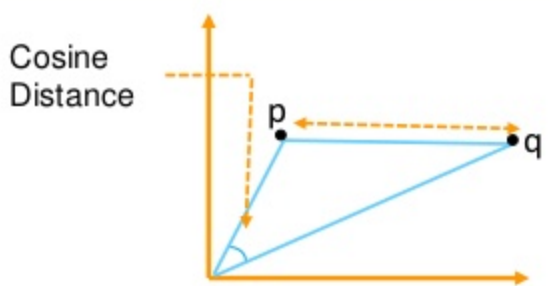
02 Squared euclidean distance measure

03 Manhattan distance measure

04 Cosine distance measure

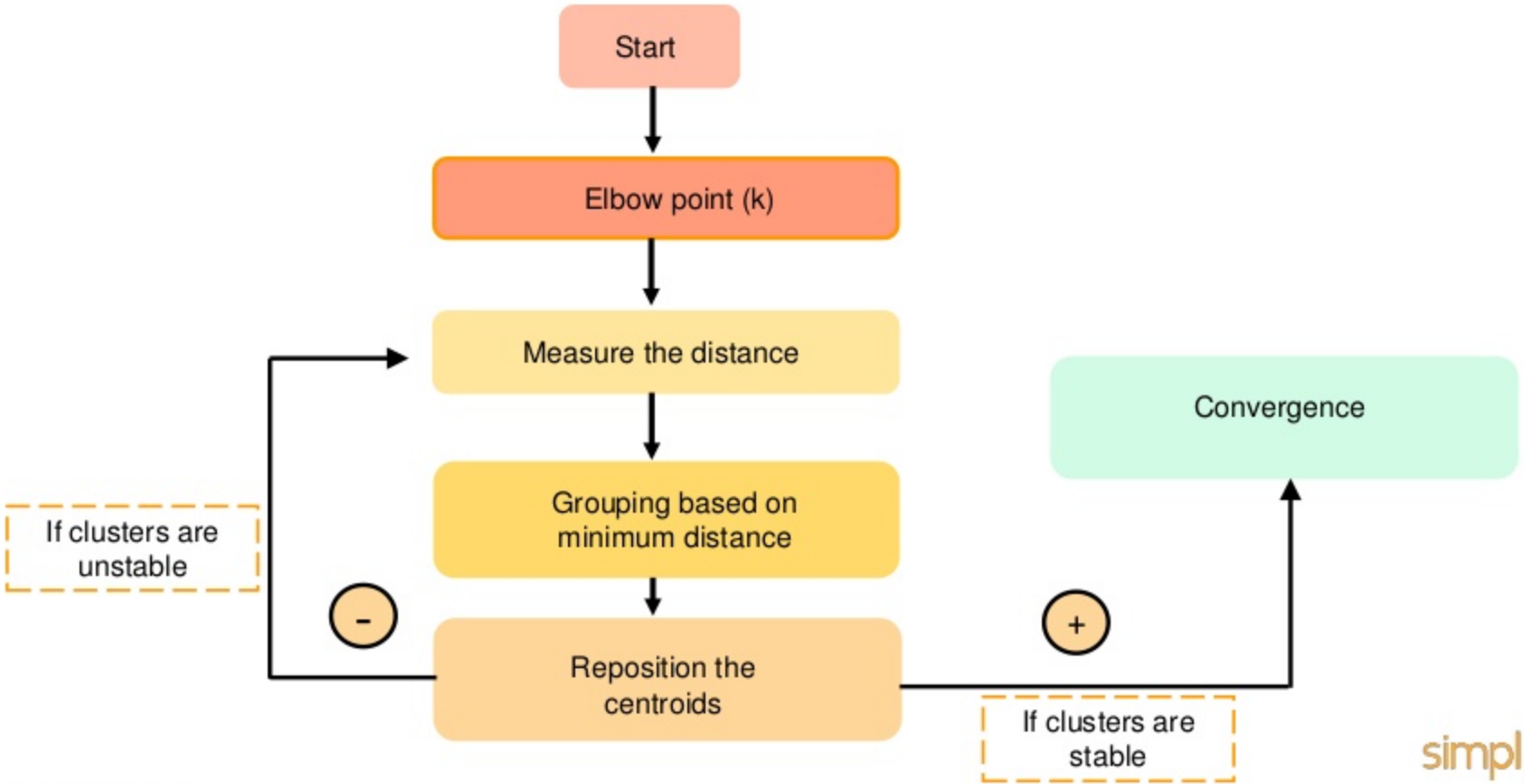
The cosine distance similarity measures the angle between the two vectors

$$\mathbf{d}= \frac{\sum_{i=0}^{n-1} q_{i-} p_x}{\sum_{i=0}^{n-1}(q_i)^2 \times \sum_{i=0}^{n-1}(p_i)^2}$$



simplilearn

How does K-Means clustering work?



How does K-Means clustering work?

- Elbow point
- Measure the distance
- Grouping
- Reposition the centroids
- Convergence

- Let's say, you have a dataset for a **Grocery shop**

- Now, the important question is, "***how would you choose the optimum number of clusters?***"

simplilearn

How does K-Means clustering work?

- Elbow point
- Measure the distance
- Grouping
- Reposition the centroids
- Convergence

- The best way to do this is by **Elbow method**
- The idea of the elbow method is to run K-Means clustering on the dataset where 'k' is referred as number of clusters
- Within sum of squares (WSS) is defined as the sum of the squared distance between each member of the cluster and its centroid

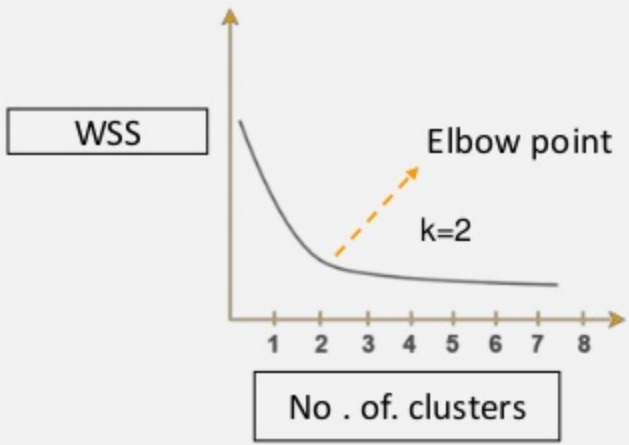
$$WSS = \sum_{i=1}^{m} (x_i - c_i)^2$$

Where x_i = data point and c_i = closest point to centroid

simplilearn

How does K-Means clustering work?

- Elbow point
- Measure the distance
- Grouping
- Reposition the centroids
- Convergence



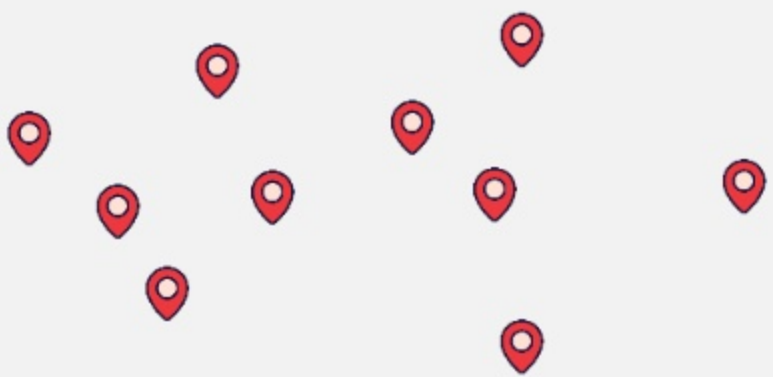
- Now, we draw a curve between ***WSS*** (within sum of squares) and the ***number of clusters***
- Here, we can see a very slow change in the value of WSS after k=2, so you should take that elbow point value as the final number of clusters

simplilearn

How does K-Means clustering work?

- Elbow point
- Measure the distance
- Grouping
- Reposition the centroids
- Convergence

Step 1: The given data points below are assumed as **delivery points**



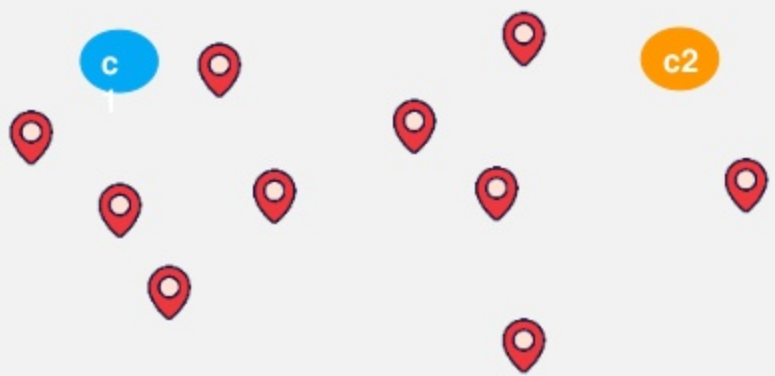
simplilearn

How does K-Means clustering work?

- Elbow point
- Measure the distance
- Grouping
- Reposition the centroids
- Convergence

Step 2: We can randomly initialize two points called the cluster centroids,

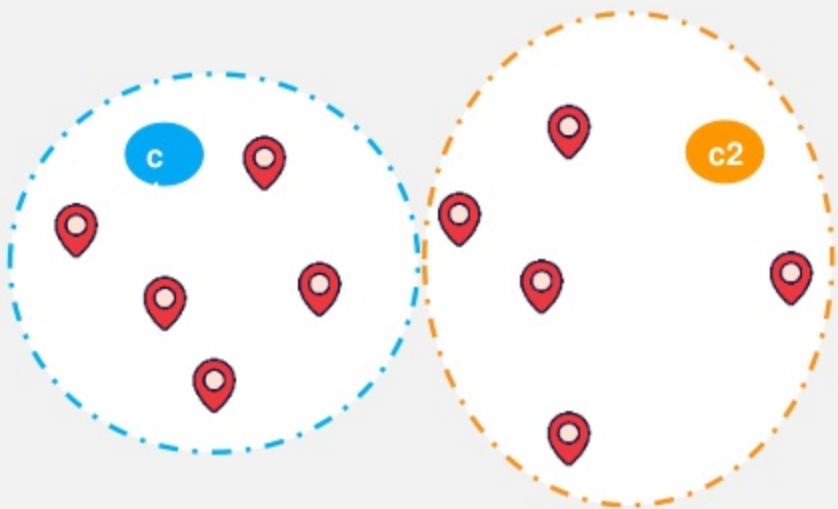
Euclidean distance is a distance measure used to find out which data point is closest to our centroids



How does K-Means clustering work?

- Elbow point
- Measure the distance
- Grouping
- Reposition the centroids
- Convergence

Step 3: Based upon the distance from c1 and c2 centroids, the data points will group itself into clusters



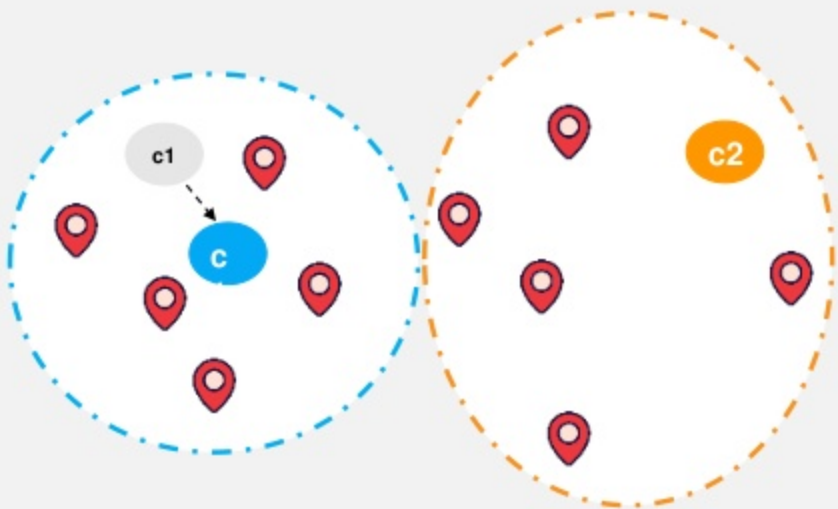
simplilearn

How does K-Means clustering work?

- Elbow point
- Measure the distance
- Grouping
- Reposition the centroids
- Convergence

Step 4: Compute the centroid of data points inside blue cluster

Step 5: Reposition the centroid of the blue cluster to the new centroid



simplilearn

How does K-Means clustering work?

- Elbow point
- Measure the distance
- Grouping
- Reposition the centroids
- Convergence

Step 6: Now, compute the centroid of data points inside orange cluster

Step 7: Reposition the centroid of the orange cluster to the new centroid

simplilearn

How does K-Means clustering work?

- Elbow point
- Measure the distance
- Grouping
- Reposition the centroids
- Convergence

Step 8: Once the clusters become static, K-Means clustering algorithm is said to be converged

simplilearn

K-Means Clustering Algorithm

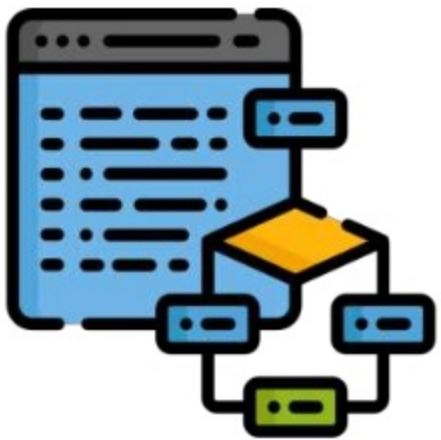
Assuming we have inputs x1,x2,x3,…, and value of **K,**

Step 1 : Pick K random points as cluster centers called centroids

Step 2 : Assign each xi to nearest cluster by calculating its distance to each centroid

Step 3 : Find new cluster center by taking the average of the assigned points

Step 4 : Repeat Step 2 and 3 until none of the cluster assignments change



simplilearn

K-Means Clustering Algorithm

Step 1 :

We randomly pick **K** cluster centers (centroids). Let's assume these are c1,c2,…,ckc1,c2,…,ck, and we can say that;

$$C = c_1, c_2, \ldots . c_k$$

C is the set of all centroids.

Step 2:

In this step, we assign each data point to closest center, this is done by calculating Euclidean distance

$$\arg\min_{c_i \in C} dist(\quad, x_i)^2$$

Where **dist()** is the Euclidean distance.

K-Means Clustering Algorithm

Step 3:

In this step, we find the new centroid by taking the average of all the points assigned to that cluster.

$$c_i = \frac{1}{|Si|} \sum_{xi \in Si} x_i$$

S_i is the set of all points assigned to the i th cluster

Step 4:

In this step, we repeat step 2 and 3 until none of the cluster assignments change

That means until our clusters remain stable, we repeat the algorithm

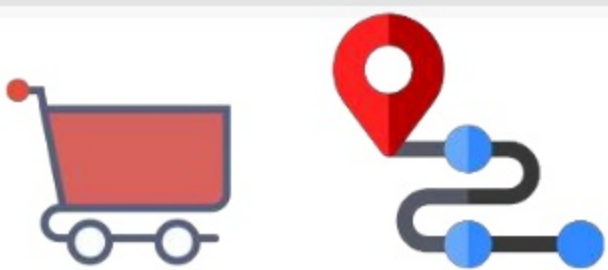
Demo: K-Means Clustering

Problem Statement

- Walmart wants to open a chain of stores across Florida and wants to find out optimal store locations to maximize revenue

Solution

- Walmart already has a strong e-commerce presence
- Walmart can use its online customer data to analyze the customer locations along with the monthly sales

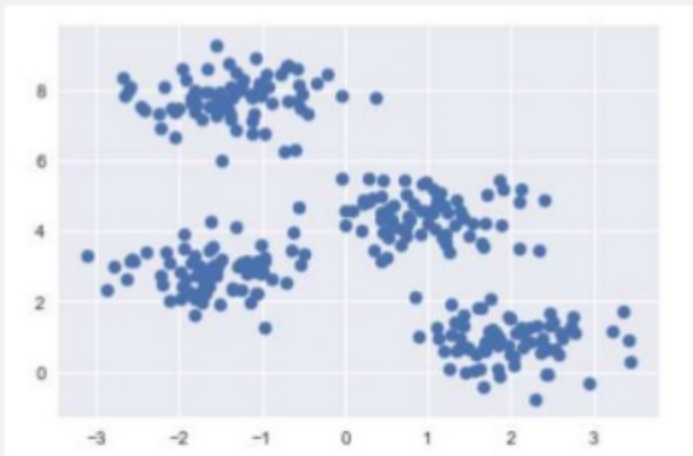


Demo: K-Means Clustering

```
%matplotlib inline
import matplotlib.pyplot as plt

# for plot styling
import seaborn as sns; sns.set()
import numpy as np
from sklearn.datasets.samples_generator import make_blobs
X, y_true = make_blobs(n_samples=300, centers=4, cluster_std=0.60, random_state=0)
plt.scatter(X[:, 0], X[:, 1], s=50);
```

Demo: K-Means Clustering

```
# output
```



simplilearn

Demo: K-Means Clustering

```
# assign four clusters
from sklearn.cluster import KMeans
kmeans = KMeans(n_clusters=4)
kmeans.fit(X)
y_kmeans = kmeans.predict(X)

# import library
from sklearn.metrics import pairwise_distances_argmin
def find_clusters(X, n_clusters, rseed=2):

# 1. randomly choose clusters
    rng = np.random.RandomState(rseed)
    i = rng.permutation(X.shape[0])[:n_clusters]
    centers = X[i]

    while True:
```

Demo: K-Means Clustering

```
# 2. assign labels based on closest center
     labels = pairwise_distances_argmin(X, centers)

# 3. find new centers from means of points
        new_centers = np.array([X[labels == i].mean(0)
                                for i in range(n_clusters)])

centers, labels = find_clusters(X, 4)
plt.scatter(X[:, 0], X[:, 1], c=labels,
            s=50, cmap='viridis')
```

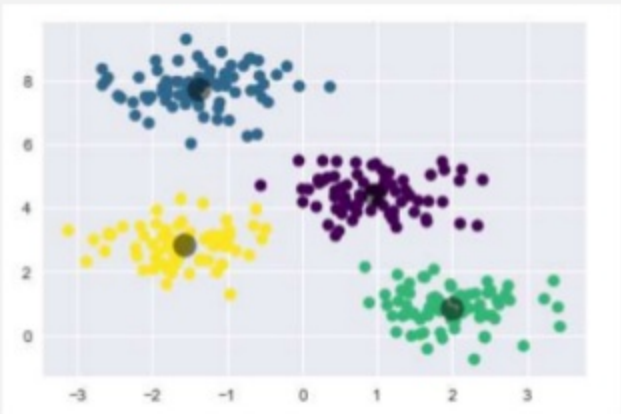
Demo: K-Means Clustering

```
# 4. check for convergence
   if np.all(centers == new_centers):
            break
   centers = new_centers
return centers, labels
centers, labels = find_clusters(X, 4)
plt.scatter(X[:, 0], X[:, 1], c=y_kmeans, s=50, cmap='viridis')

plt.scatter(centers[:, 0], centers[:, 1], c='black', s=200, alpha=0.5);
```

simplilearn

Demo: K-Means Clustering

 # output:



Conclusion

Congratulations!

We have demonstrated K-Means clustering by establishing Walmart stores across Florida in the most optimized way

Use case – Color compression

simplilearn

Use Case: K-Means for Color Compression

Problem Statement

To perform color compression on images using K-Means algorithm

Use Case: K-Means for Color Compression

```
# example 1:

# note: this requires the ``pillow`` package to be installed
from sklearn.datasets import load_sample_image
china = load_sample_image("flower.jpg")
ax = plt.axes(xticks=[], yticks=[])
ax.imshow(china);
```

#Output:

Use Case: K-Means for Color Compression

```
# returns the dimensions of the array
china.shape
```

```
Out[26]: (427, 640, 3)
```

```
# reshape the data to [n_samples x n_features], and rescale the colors so that they lie between 0 and 1
data = china / 255.0 # use 0...1 scale
data = data.reshape(427 * 640, 3)
data.shape
```

```
Out[27]: (273280, 3)
```

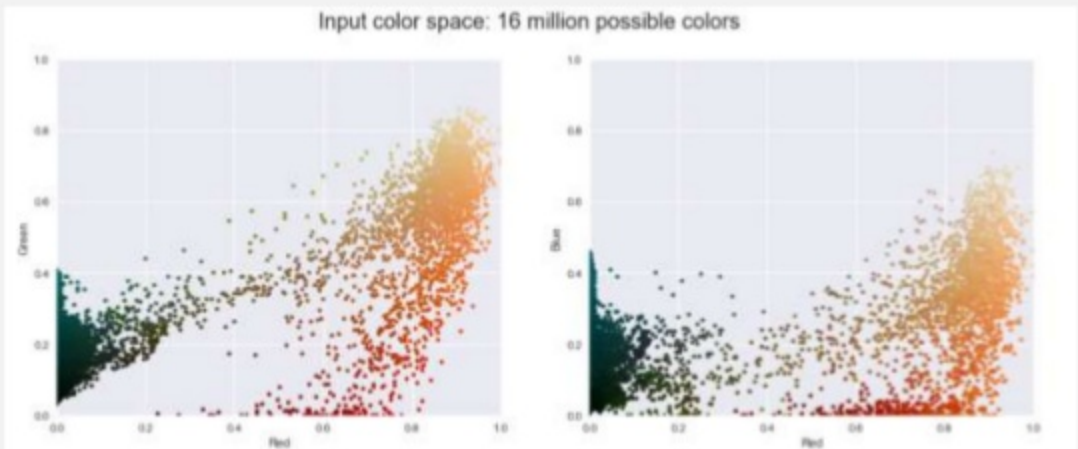
```
# visualize these pixels in this color space, using a subset of 10,000 pixels for efficiency
def plot_pixels(data, title, colors=None, N=10000):
    if colors is None:
        colors = data
```

Use Case: K-Means for Color Compression

```
# choose a random subset
    rng = np.random.RandomState(0)
    i = rng.permutation(data.shape[0])[:N]
    colors = colors[i]
    R, G, B = data[i].T

    fig, ax = plt.subplots(1, 2, figsize=(16, 6))
    ax[0].scatter(R, G, color=colors, marker='.')
    ax[0].set(xlabel='Red', ylabel='Green', xlim=(0, 1), ylim=(0, 1))

    ax[1].scatter(R, B, color=colors, marker='.')
    ax[1].set(xlabel='Red', ylabel='Blue', xlim=(0, 1), ylim=(0, 1))

    fig.suptitle(title, size=20);
```

Use Case: K-Means for Color Compression

```
plot_pixels(data, title='Input color space: 16 million possible colors')
```



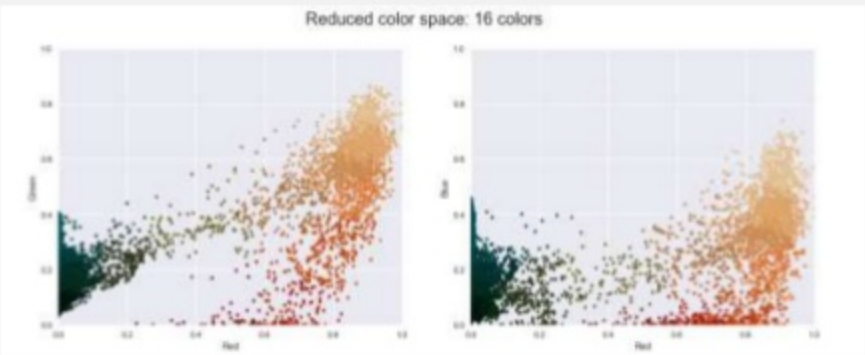
Use Case: K-Means for Color Compression

```
# fix numPy issues
import warnings; warnings.simplefilter('ignore')

# reducing these 16 million colors to just 16 colors
from sklearn.cluster import MiniBatchKMeans
kmeans = MiniBatchKMeans(16)
kmeans.fit(data)
new_colors = kmeans.cluster_centers_[kmeans.predict(data)]

plot_pixels(data, colors=new_colors,
            title="Reduced color space: 16 colors")
```

Use Case: K-Means for Color Compression

the result is re-coloring of the original pixels, where each pixel is assigned the color of its closest cluster center

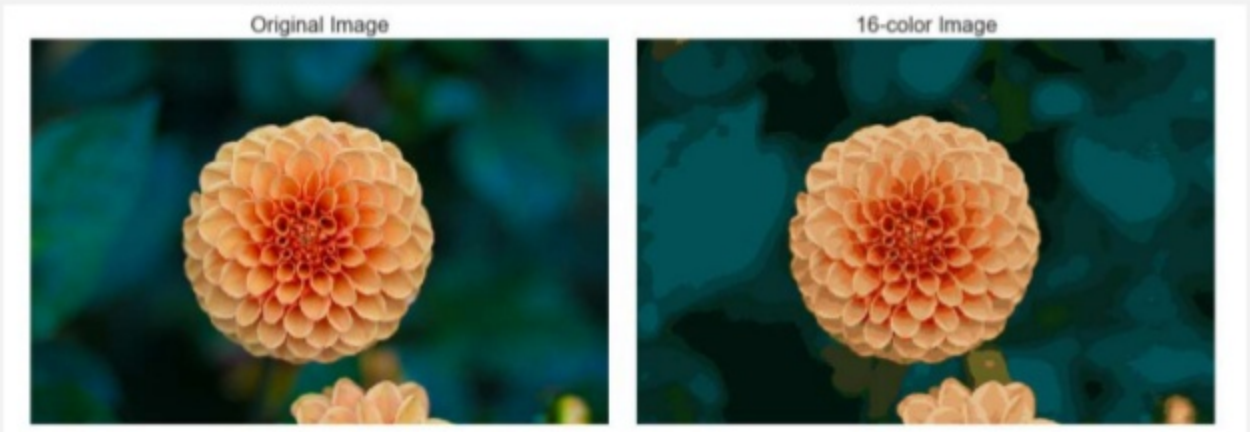
output:




```
china_recolored = new_colors.reshape(china.shape)
fig, ax = plt.subplots(1, 2, figsize=(16, 6), subplot_kw=dict(xticks=[], yticks=[]))
fig.subplots_adjust(wspace=0.05)
ax[0].imshow(china)
ax[0].set_title('Original Image', size=16)
ax[1].imshow(china_recolored)
ax[1].set_title('16-color Image', size=16);
```

Use Case: K-Means for Color Compression

```
# output
```



Use Case: K-Means for Color Compression

```
# example 2:

from sklearn.datasets import load_sample_image
china = load_sample_image("china.jpg")
ax = plt.axes(xticks=[], yticks=[])
ax.imshow(china);
```

Use Case: K-Means for Color Compression

```
# returns the dimensions of the array
china.shape
```

```
Out[26]: (427, 640, 3)
```

```
# reshape the data to [n_samples x n_features], and rescale the colors so that they lie between 0 and 1
data = china / 255.0 # use 0...1 scale
data = data.reshape(427 * 640, 3)
data.shape
```

```
Out[27]: (273280, 3)
```

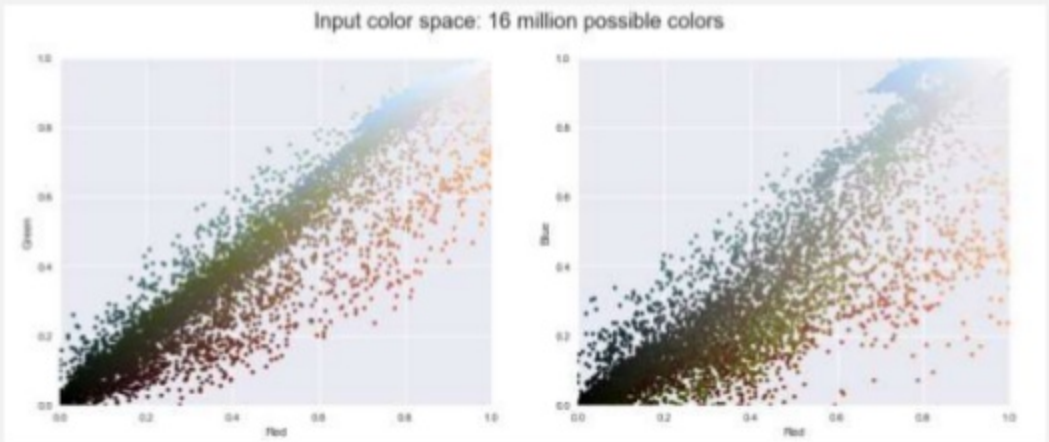
```
# visualize these pixels in this color space, using a subset of 10,000 pixels for efficiency
def plot_pixels(data, title, colors=None, N=10000):
    if colors is None:
        colors = data
```

Use Case: K-Means for Color Compression

```
# choose a random subset
    rng = np.random.RandomState(0)
    i = rng.permutation(data.shape[0])[:N]
    colors = colors[i]
    R, G, B = data[i].T

    fig, ax = plt.subplots(1, 2, figsize=(16, 6))
    ax[0].scatter(R, G, color=colors, marker='.')
    ax[0].set(xlabel='Red', ylabel='Green', xlim=(0, 1), ylim=(0, 1))

    ax[1].scatter(R, B, color=colors, marker='.')
    ax[1].set(xlabel='Red', ylabel='Blue', xlim=(0, 1), ylim=(0, 1))

    fig.suptitle(title, size=20);
```

Use Case: K-Means for Color Compression

```
plot_pixels(data, title='Input color space: 16 million possible colors')
```

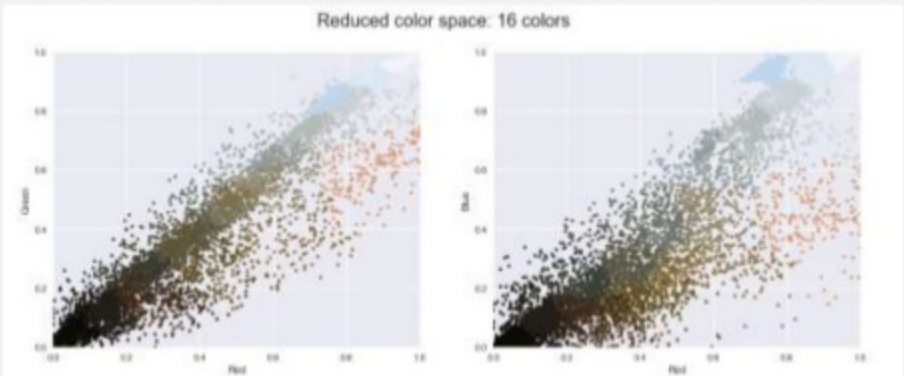


Use Case: K-Means for Color Compression

```
# fix NumPy issues
import warnings; warnings.simplefilter('ignore')

# reducing these 16 million colors to just 16 colors
from sklearn.cluster import MiniBatchKMeans
kmeans = MiniBatchKMeans(16)
kmeans.fit(data)
new_colors = kmeans.cluster_centers_[kmeans.predict(data)]

plot_pixels(data, colors=new_colors,
            title="Reduced color space: 16 colors")
```

Use Case: K-Means for Color Compression

```
# the result is a re-coloring of the original pixels, where each pixel is assigned the color of its closest cluster center
# Output
```




```
china_recolored = new_colors.reshape(china.shape)
fig, ax = plt.subplots(1, 2, figsize=(16, 6), subplot_kw=dict(xticks=[], yticks=[]))
fig.subplots_adjust(wspace=0.05)
ax[0].imshow(china)
ax[0].set_title('Original Image', size=16)
ax[1].imshow(china_recolored)
ax[1].set_title('16-color Image', size=16);
```

simplilearn

Use Case: K-Means for Color Compression

```
# output
```

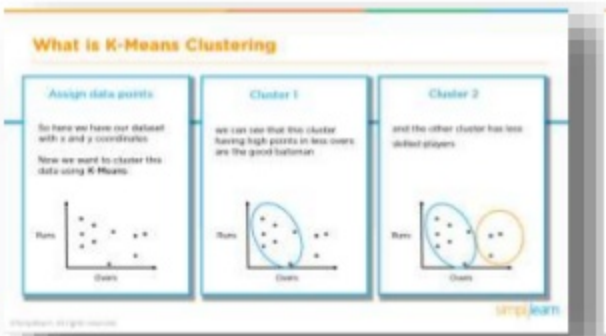
Conclusion

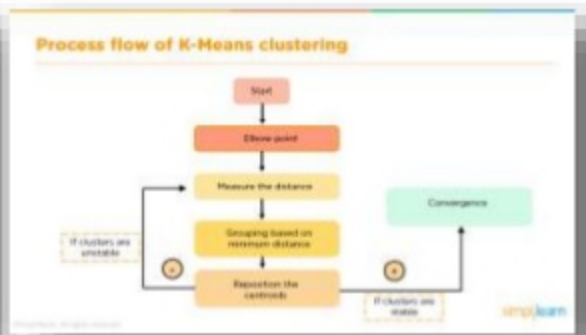
Congratulations!

We have demonstrated K-Means in color compression.

The hands on example will help you to encounter any K-Means project in future.

Key Takeaways





simplilearn


THANK YOU
For more information, visit
www.simplilearn.com
simplilearn