# LAB # 12

## Statement Purpose:

This lab will introduce students to clustering using K-means algorithm.

## Introduction:

k-means clustering is a method of vector quantization, originally from signal processing, that aims to partition n observations into k clusters in which each observation belongs to the cluster with the nearest mean (cluster centers or cluster centroid), serving as a prototype of the cluster. This results in a partitioning of the data space into Voronoi cells. k-means clustering minimizes within-cluster variances (squared Euclidean distances), but not regular Euclidean distances, which would be the more difficult Weber problem: the mean optimizes squared errors, whereas only the geometric median minimizes Euclidean distances. For instance, better Euclidean solutions can be found using k-medians and k-medoids.

## Lab Activities:

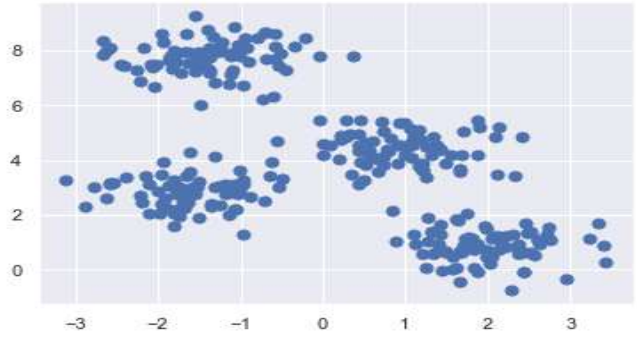
**Activity 1:**

**Using k means for unsupervised learning using sample data**

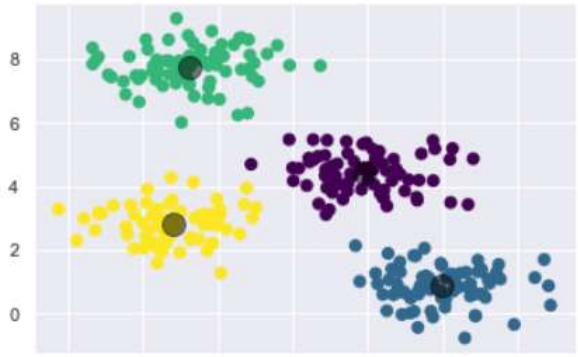
**Solution:**

We will first generate some random points using and cluster them. For visualization we use matplotlib and seaborn

```
]: import matplotlib.pyplot as plt
   import seaborn as sns; sns.set()  # for plot styling
   import numpy as np
   from sklearn.cluster import KMeans
   from sklearn.datasets.samples_generator import make_blobs

   # Generating data
   X, y_true = make_blobs(n_samples=300, centers=4,
                          cluster_std=0.60, random_state=0)

   # Generating plot
   plt.scatter(X[:, 0], X[:, 1], s=50);
```



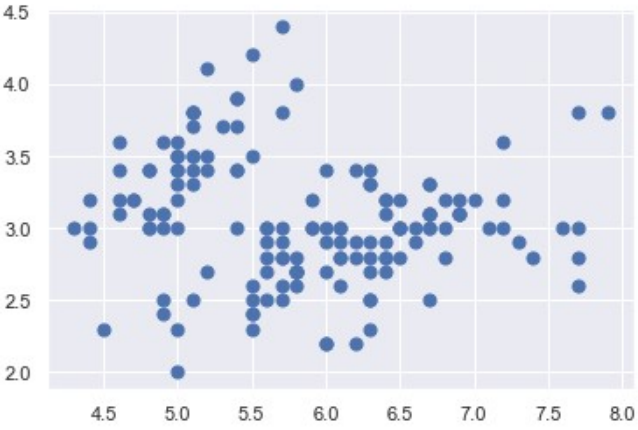
```
# Running k-means
kmeans = KMeans(n_clusters=4)
kmeans.fit(X)
y_kmeans = kmeans.predict(X)

# Generating result plot
plt.scatter(X[:, 0], X[:, 1], c=y_kmeans, s=50, cmap='viridis')

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




**Activity 2:**

K means on iris – without using labels

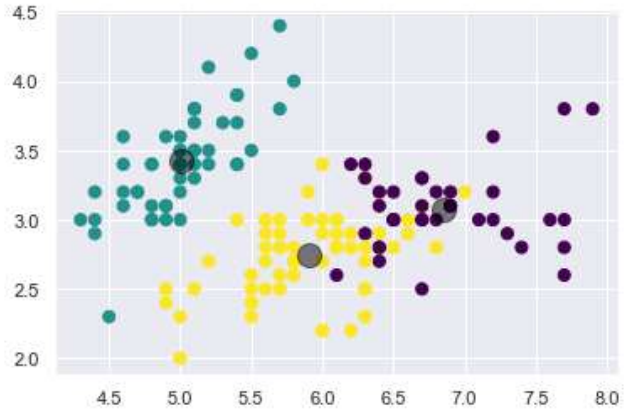
```
import matplotlib.pyplot as plt
import seaborn as sns; sns.set()  # for plot styling
import numpy as np
from sklearn.cluster import KMeans

from sklearn.datasets import load_iris
iris = load_iris()
iris.data.shape

plt.scatter(iris.data[:, 0], iris.data[:, 1], s=50);
```



```
kmeans = KMeans(n_clusters=3, random_state=0)
y_kmeans = kmeans.fit_predict(iris.data)
kmeans.cluster_centers_.shape
```

```
(3, 4)
```

```
# Generating result plot
plt.scatter(X[:, 0], X[:, 1], c=y_kmeans, s=50, cmap='viridis')

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




**Activity 3:**

Reading form own csv file and running k-means on it. MNIST is used in this example, which is a digits dataset – 28x28 sized images of 0-9 digits. Each digit is changed into 784 sized vector

In [40]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sn
from numpy import genfromtxt

dataset = genfromtxt('C:/Users/DELL/Google Drive/FA21-AI/Artificial Intelligence/rollno_lab5.tar/rollno_lab5/d

data = dataset[1:,:]
# print(dataset)
print(dataset.shape)
```

```
(5000, 784)
```

In [42]:
```
kmeans = KMeans(n_clusters=10, random_state=0)
y_kmeans = kmeans.fit_predict(data)
kmeans.cluster_centers_.shape
```

Out[42]:
```
(10, 784)
```

**Bonus:**
**Example 2 given on following colab link:**

https://colab.research.google.com/github/jakevdp/PythonDataScienceHandbook/blob/master/notebooks/05.11-K-Means.ipynb#scrollTo=YTMaSQiwux9S

## Pre-Lab Activity:

**Use the iris dataset given in last class and run k-means algorithm on it on paper. Do at-least 3 iterations.**

**These hand-written solutions have to be SUBMITTED.**