Introduction to K-Means Algorithm

K Means Clustering is an unsupervised learning algorithm that tries to cluster data based on their similarity. Unsupervised learning means that there is no outcome to be predicted, and the algorithm just tries to find patterns in the data. In k means clustering, we have the specify the number of clusters we want the data to be grouped into. The algorithm randomly assigns each observation to a cluster, and finds the centroid of each cluster. Then, the algorithm iterates through two steps: Reassign data points to the cluster whose centroid is closest. Calculate new centroid of each cluster. These two steps are repeated till the within cluster variation cannot be reduced any further. The within cluster variation is calculated as the sum of the euclidean distance between the data points and their respective cluster centroids.

In this algorithm, the data points are assigned to a cluster in such a manner that the sum of the squared distance between the data points and centroid would be minimum. It is to be understood that less variation within the clusters will lead to more similar data points within same cluster.

Hence, K-means clustering algorithm computes the centroids and iterates until we it finds optimal centroid. It assumes that the number of clusters are already known. It is also called **flat clustering** algorithm. The number of clusters identified from data by algorithm is represented by ‘K’ in K-means.

Working of K-Means Algorithm

We can understand the working of K-Means clustering algorithm with the help of following steps −

Step 1 − First, we need to specify the number of clusters, K, need to be generated by this algorithm.

Step 2 − Next, randomly select K data points and assign each data point to a cluster. In simple words, classify the data based on the number of data points.

Step 3 − Now it will compute the cluster centroids.

Step 4 − Next, keep iterating the following until we find optimal centroid which is the assignment of data points to the clusters that are not changing any more

- **4.1** − First, the sum of squared distance between data points and centroids would be computed.
- **4.2** − Now, we have to assign each data point to the cluster that is closer than other cluster (centroid).
- **4.3** − At last compute the centroids for the clusters by taking the average of all data points of that cluster.

K-means follows **Expectation-Maximization** approach to solve the problem. The Expectation-step is used for assigning the data points to the closest cluster and the Maximization-step is used for computing the centroid of each cluster.

While working with K-means algorithm we need to take care of the following things −

- While working with clustering algorithms including K-Means, it is recommended to standardize the data because such algorithms use distance-based measurement to determine the similarity between data points.
- Due to the iterative nature of K-Means and random initialization of centroids, K-Means may stick in a local optimum and may not converge to global optimum. That is why it is recommended to use different initializations of centroids.

Implementing in Python

Importing Libraries

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

Create Some Data

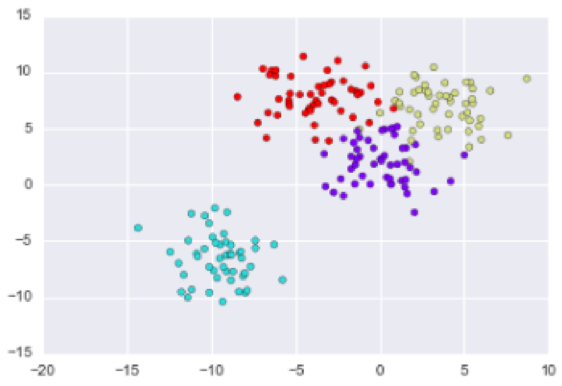
```
from sklearn.datasets import make_blobs
```

```
# Create Data
data = make_blobs(n_samples=200, n_features=2,
                           centers=4, cluster_std=1.8,random_state=101)
```

Visualizing Data

```
plt.scatter(data[0][:,0],data[0][:,1],c=data[1],cmap='rainbow')
```

```
<matplotlib.collections.PathCollection at 0x11ab34d30>
```



Creating Clusters

```
from sklearn.cluster import KMeans
```

```
kmeans = KMeans(n_clusters=4)
```

```
kmeans.fit(data[0])
```

```
KMeans(copy_x=True, init='k-means++', max_iter=300, n_clusters=4, n_init=10,
    n_jobs=1, precompute_distances='auto', random_state=None, tol=0.0001,
    verbose=0)
```

```
kmeans.cluster_centers_
```

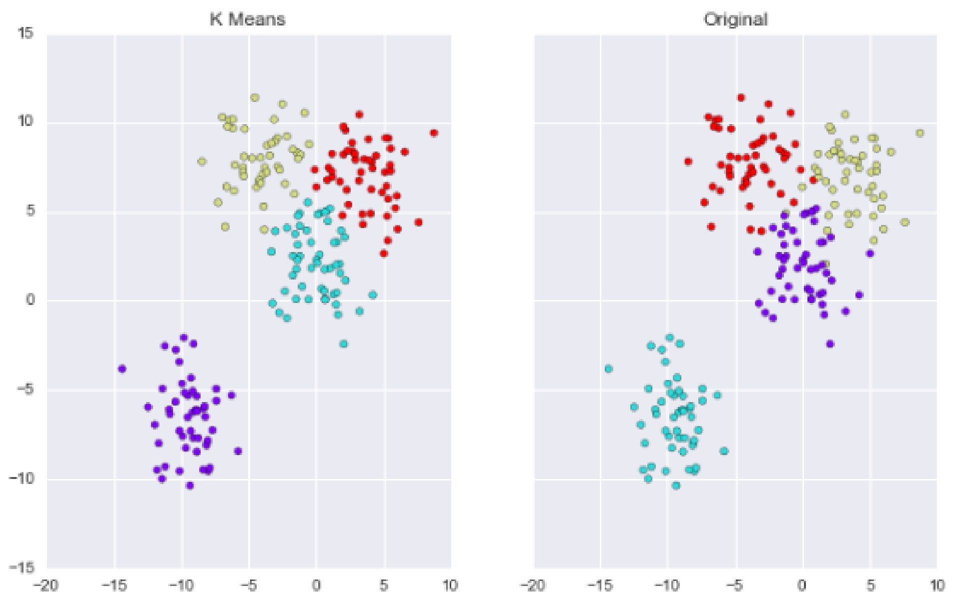
```
array([[-4.13591321,  7.95389851],
       [-9.46941837, -6.56081545],
       [-0.0123077 ,  2.13407664],
       [ 3.71749226,  7.01388735]])
```

```
kmeans.labels_
```

```
array([2, 3, 1, 3, 3, 0, 3, 1, 3, 1, 2, 1, 3, 3, 2, 1, 3, 1, 0, 2, 0, 1, 1,
       0, 2, 0, 0, 1, 3, 3, 2, 0, 3, 1, 1, 2, 0, 0, 0, 1, 0, 2, 2, 2, 1, 3,
       2, 1, 0, 1, 1, 2, 3, 1, 0, 2, 1, 1, 2, 3, 0, 3, 0, 2, 3, 1, 0, 3, 3,
       0, 3, 1, 0, 1, 0, 3, 3, 1, 2, 1, 1, 0, 3, 0, 1, 1, 1, 2, 1, 0, 0, 0,
       0, 1, 1, 0, 3, 2, 0, 3, 1, 0, 1, 1, 3, 1, 0, 3, 0, 0, 3, 2, 2, 3, 0,
       3, 2, 2, 3, 2, 1, 2, 1, 2, 1, 3, 2, 1, 0, 2, 2, 2, 1, 0, 0, 2, 3, 2,
       3, 1, 0, 3, 0, 2, 2, 3, 1, 0, 2, 2, 2, 2, 1, 3, 1, 2, 3, 3, 3, 1, 3,
       1, 1, 2, 0, 2, 1, 3, 2, 1, 3, 1, 2, 3, 1, 2, 3, 3, 0, 3, 2, 0, 0, 2,
       0, 0, 0, 0, 0, 1, 0, 3, 3, 2, 0, 1, 3, 3, 0, 1], dtype=int32)
```

```
f, (ax1, ax2) = plt.subplots(1, 2, sharey=True,figsize=(10,6))
ax1.set_title('K Means')
ax1.scatter(data[0][:,0],data[0][:,1],c=kmeans.labels_,cmap='rainbow')
ax2.set_title("Original")
ax2.scatter(data[0][:,0],data[0][:,1],c=data[1],cmap='rainbow')
```

```
<matplotlib.collections.PathCollection at 0x11da01be0>
```



You should note, the colors are meaningless in reference between the two plots.

Application of K-means Clustering

The concern of the fact is that the data is always complicated, mismanaged, and noisy. The conditions in the real world cast hardly the clear picture to which these types of algorithms can be applied. Let’s learn where we can implement k-means clustering.

1. K-means clustering is applied in the **Call Detail Record (CDR) Analysis**. It gives in-depth vision about customer requirements and satisfaction on the basis of call-traffic during the time of the day and demographic of a particular location.

2. It is used in the clustering of documents to identify the compatible documents in the same place

3. It is deployed to classify the sounds on the basis of their identical patterns and segregate malformation in them.

4. It serves as the **model of lossy images compression technique**, in the confinement of images, K-means makes clusters pixels of an image in order to decrease the total size of it.

5. It is helpful in the business sector for recognizing the portions of purchases made by customers, also to cluster movements on apps and websites.

6. In the field of insurance and fraud detection on the basis of prior data, it is plausible to cluster fraudulent consumers to demand based on their proximity to clusters as the patterns indicate.

Conclusion

K-means clustering is the unsupervised machine learning algorithm that is part of a much deep pool of data techniques and operations in the realm of Data Science. It is the fastest and most efficient algorithm to categorize data points into groups even when very little information is available about data.