k-Means Algorithm for Clustering

AcadView

June 7, 2018

---

1 Overview

k-means clustering is a type of unsupervised learning, which is used when you have unlabeled data (i.e., data without defined categories or groups). The goal of this algorithm is to find groups in the data, with the number of groups represented by the variable k. The algorithm works iteratively to assign each data point to one of k groups based on the features that are provided. Data points are clustered based on feature similarity. The results of the k-means clustering algorithm are:

- The centroids of the k clusters, which can be used to label new data
- Labels for the training data (each data point is assigned to a single cluster)

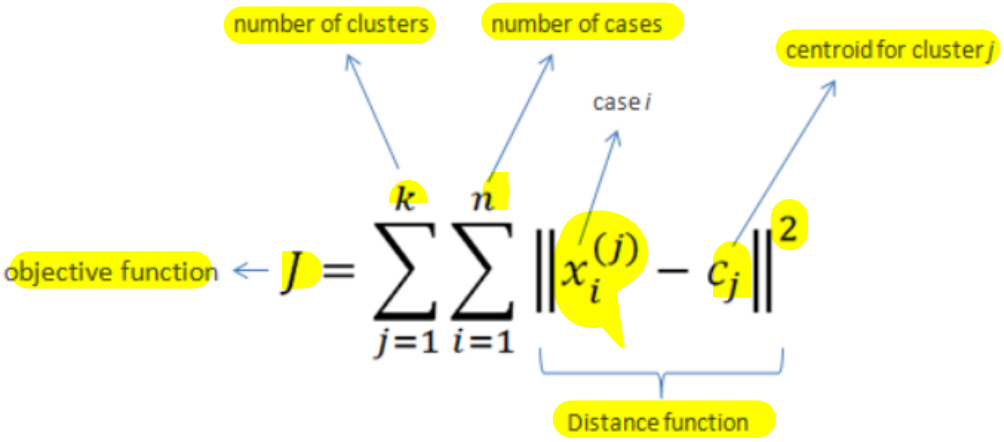
Rather than defining groups before looking at the data, clustering allows you to find and analyze the groups that have formed organically.

Each centroid of a cluster is a collection of feature values which define the resulting groups. Examining the centroid feature weights can be used to qualitatively interpret what kind of group each cluster represents.

2 k-means Algorithm

k-Means clustering intends to partition n objects into k clusters in which each object belongs to the cluster with the nearest mean. This method produces exactly k different clusters of greatest possible distinction. The best number of clusters k leading to the greatest separation (distance) is not known as a priori and must be computed from the data. The objective of k-Means clustering is to minimize total intra-cluster variance, or,

the squared error function:



2.1 Pseudocode

- Clusters the data into k groups where k is predefined.
- Select k points at random as cluster centers.
- Assign objects to their closest cluster center according to the Euclidean distance function.
- Calculate the centroid or mean of all objects in each cluster.
- Repeat steps 2, 3 and 4 until the same points are assigned to each cluster in consecutive rounds.

Or we can view the algorithm in a two step detailed way and iterate to get final clusters.

The k-means clustering algorithm uses iterative refinement to produce a final result. The algorithm inputs are the number of clusters and the data set. The data set is a collection of features for each data point. The algorithms starts with initial estimates for the centroids, which can either be randomly generated or randomly selected from the data set. The algorithm then iterates between two steps:

1.Data assigment step:

Each centroid defines one of the clusters. In this step, each data point is assigned to its nearest centroid, based on the squared Euclidean distance. More formally, if ci is the collection of centroids in set C, then each data point x is assigned to a cluster based on

$$\underset{\mathbf{c_i} \in \mathbf{C}}{\arg\min}\, \mathbf{dist}(\mathbf{c_i}, \mathbf{x})^{\mathbf{2}}$$

where dist( ) is the standard (L2) Euclidean distance. Let the set of data point assignments for each ith cluster centroid be Si.

2. Centroid update step:

In this step, the centroids are recomputed. This is done by taking the mean of all data points assigned to that centroid's cluster.

$$\mathbf{c_i} = \frac{\mathbf{1}}{|\mathbf{S_i}|} \sum_{\mathbf{x_i} \in \mathbf{S_i} \mathbf{x_i}}$$

The algorithm iterates between steps one and two until a stopping criteria is met (i.e., no data points change clusters, the sum of the distances is minimized, or some maximum number of iterations is reached).

However, we need to specify the number of clusters, in advance and the final results are sensitive to initialization and often terminates at a local optimum. Unfortunately there is no global theoretical method to find the optimal number of clusters. A practical approach is to compare the outcomes of multiple runs with different k and choose the best one based on a predefined criterion. In general, a large k probably decreases the error but increases the risk of overfitting.

2.2 Illustration with an toy example

Suppose we want to group the visitors to a website using just their age (one-dimensional space) as follows:

$$n = 19$$

$$\mathbf{15, 15, 16, 19, 19, 20, 20, 21, 22, 28, 35, 40, 41, 42, 43, 44, 60, 61, 65}$$

Initial clusters (random centroid or average):

$$\mathbf{k = 2}$$

$$Distance\ 1 = \sqrt{(x_i - c_1)^2}$$

$$Distance\ 2 = \sqrt{(x_i - c_2)^2}$$

$$c1 = 16\ c2 = 22$$

Iteration 1:

$$c1 = 15.33 \; c2 = 36.25$$

-

x_i	c_1	c_2	Distance 1	Distance 2	Nearest Cluster	New Centroid
15	16	22	1	7	1	
15	16	22	1	7	1	**15.33**
16	16	22	0	6	1	
19	16	22	9	3	2	
19	16	22	9	3	2	
20	16	22	16	2	2	
20	16	22	16	2	2	
21	16	22	25	1	2	
22	16	22	36	0	2	
28	16	22	12	6	2	
35	16	22	19	13	2	
40	16	22	24	18	2	**36.25**
41	16	22	25	19	2	
42	16	22	26	20	2	
43	16	22	27	21	2	
44	16	22	28	22	2	
60	16	22	44	38	2	
61	16	22	45	39	2	
65	16	22	49	43	2	

Iteration 2:

$$c1 = 18.56 \; c2 = 45.90$$

x_i	c_1	c_2	Distance 1	Distance 2	Nearest Cluster	New Centroid
15	15.33	36.25	0.33	21.25	1	
15	15.33	36.25	0.33	21.25	1	
16	15.33	36.25	0.67	20.25	1	
19	15.33	36.25	3.67	17.25	1	
19	15.33	36.25	3.67	17.25	1	**18.56**
20	15.33	36.25	4.67	16.25	1	
20	15.33	36.25	4.67	16.25	1	
21	15.33	36.25	5.67	15.25	1	
22	15.33	36.25	6.67	14.25	1	
28	15.33	36.25	12.67	8.25	2	
35	15.33	36.25	19.67	1.25	2	
40	15.33	36.25	24.67	3.75	2	
41	15.33	36.25	25.67	4.75	2	
42	15.33	36.25	26.67	5.75	2	**45.9**
43	15.33	36.25	27.67	6.75	2	
44	15.33	36.25	28.67	7.75	2	
60	15.33	36.25	44.67	23.75	2	
61	15.33	36.25	45.67	24.75	2	
65	15.33	36.25	49.67	28.75	2	

Iteration 3:

$$c1 = 19.50 \; c2 = 47.89$$

x_i	c_1	c_2	Distance 1	Distance 2	Nearest Cluster	New Centroid
15	18.56	45.9	3.56	30.9	1	
15	18.56	45.9	3.56	30.9	1	
16	18.56	45.9	2.56	29.9	1	
19	18.56	45.9	0.44	26.9	1	
19	18.56	45.9	0.44	26.9	1	
20	18.56	45.9	1.44	25.9	1	**19.50**
20	18.56	45.9	1.44	25.9	1	
21	18.56	45.9	2.44	24.9	1	
22	18.56	45.9	3.44	23.9	1	
28	18.56	45.9	9.44	17.9	1	
35	18.56	45.9	16.44	10.9	2	
40	18.56	45.9	21.44	5.9	2	
41	18.56	45.9	22.44	4.9	2	
42	18.56	45.9	23.44	3.9	2	
43	18.56	45.9	24.44	2.9	2	**47.89**
44	18.56	45.9	25.44	1.9	2	
60	18.56	45.9	41.44	14.1	2	
61	18.56	45.9	42.44	15.1	2	
65	18.56	45.9	46.44	19.1	2	

Iteration 4

$$c1 = 19.50 \; c2 = 47.89$$

x_i	c_1	c_2	Distance 1	Distance 2	Nearest Cluster	New Centroid
15	19.5	47.89	4.50	32.89	1	
15	19.5	47.89	4.50	32.89	1	
16	19.5	47.89	3.50	31.89	1	
19	19.5	47.89	0.50	28.89	1	
19	19.5	47.89	0.50	28.89	1	
20	19.5	47.89	0.50	27.89	1	**19.50**
20	19.5	47.89	0.50	27.89	1	
21	19.5	47.89	1.50	26.89	1	
22	19.5	47.89	2.50	25.89	1	
28	19.5	47.89	8.50	19.89	1	
35	19.5	47.89	15.50	12.89	2	
40	19.5	47.89	20.50	7.89	2	
41	19.5	47.89	21.50	6.89	2	
42	19.5	47.89	22.50	5.89	2	
43	19.5	47.89	23.50	4.89	2	**47.89**
44	19.5	47.89	24.50	3.89	2	
60	19.5	47.89	40.50	12.11	2	
61	19.5	47.89	41.50	13.11	2	
65	19.5	47.89	45.50	17.11	2	

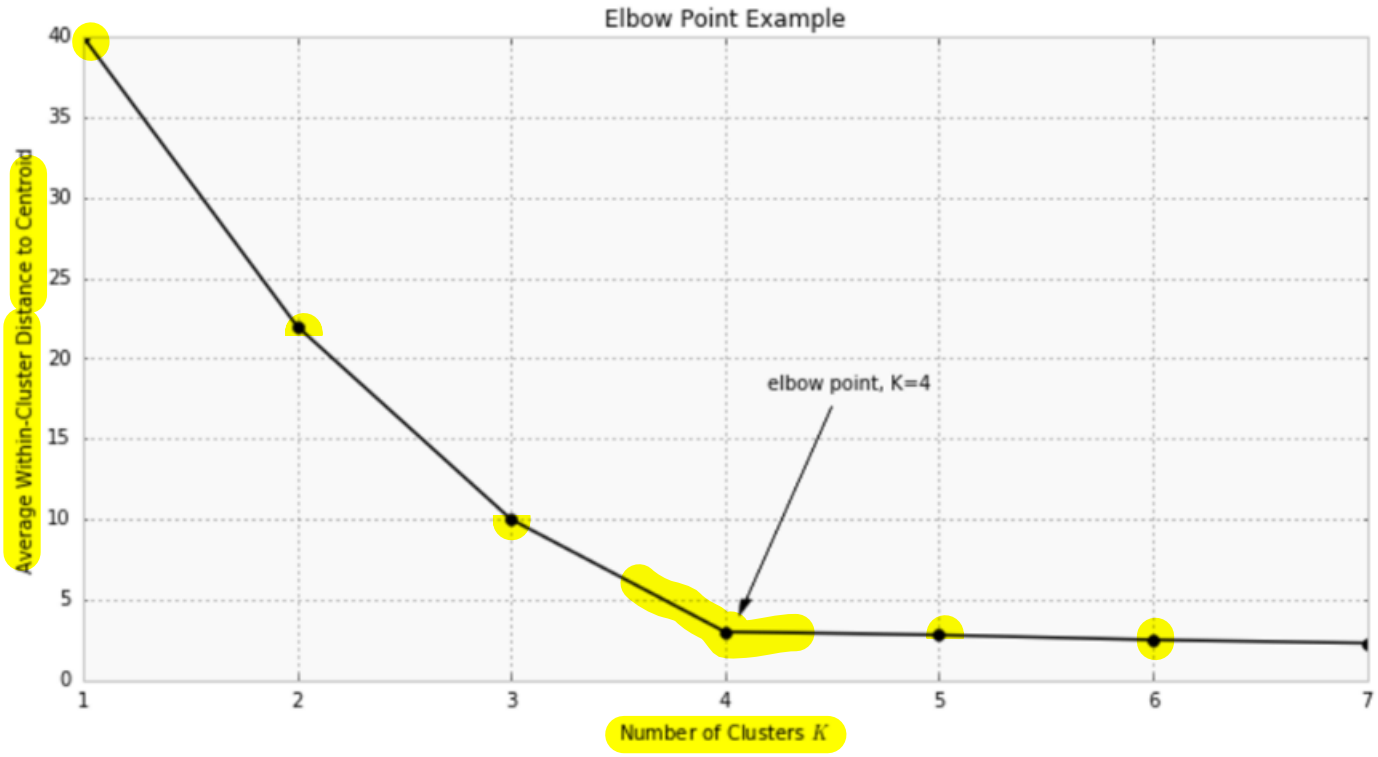
Note: No change between iterations 3 and 4 has been noted. By using clustering, 2 groups have been identified 15-28 and 35-65. The initial choice of centroids can affect the output clusters, so the algorithm is often run multiple times with different starting conditions in order to get a fair view of what the clusters should be.

2.3 Choosing K

The algorithm described above finds the clusters and data set labels for a particular pre-chosen K. To find the number of clusters in the data, the user needs to run the K-means clustering algorithm for a range of K values and compare the results. In general, there is no method for determining exact value of K, but an accurate estimate can be obtained using the following techniques.

One of the metrics that is commonly used to compare results across different values of K is the mean distance between data points and their cluster centroid. Since increasing the number of clusters will always reduce the distance to data points, increasing K will always decrease this metric, to the extreme of reaching zero when K is the same as the number of data points. Thus, this metric cannot be used as the sole target. Instead, mean distance to the centroid as a function of K is plotted and the "elbow point," where the rate of decrease sharply shifts, can be used to roughly determine K.

A number of other techniques exist for validating K, including cross-validation, information criteria, the information theoretic jump method, the silhouette method, and the G-means algorithm. In addition, monitoring the distribution of data points across groups provides insight into how the algorithm is splitting the data for each K.



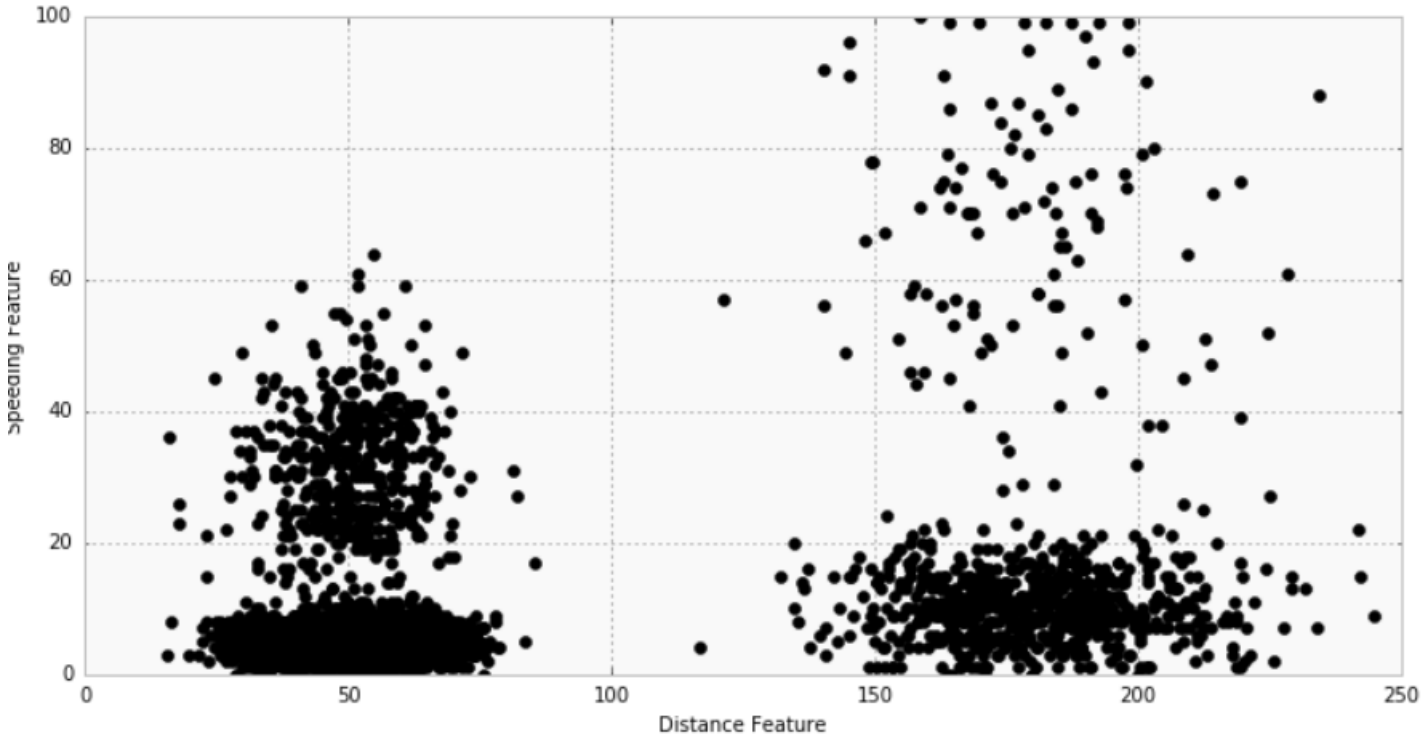
3 Implementation of K-means through scikit-learning

Applying K-Means Clustering to Delivery Fleet Data:
As an example, we'll show how the K-means algorithm works with a sample dataset of delivery fleet driver data.
`https://raw.githubusercontent.com/datascienceinc/learn-data-science/master/Introduction-to-K-means-Clustering/Data/data_1024.csv`
For the sake of simplicity, we'll only be looking at two driver features: mean distance driven per day and the mean percentage of time a driver was ¿5 mph over the speed limit. In general, this algorithm can be used for any number of features, so long as the number of data samples is much greater than the number of features.

Step 1: Clean and Transform Your Data
For this example, we've already cleaned and completed some simple data transformations. A sample of the data as a pandas DataFrame is shown below.

	Driver_ID	Distance_Feature	Speeding_Feature
0	3423311935	71.24	28
1	3423313212	52.53	25
2	3423313724	64.54	27
3	3423311373	55.69	22
4	3423310999	54.58	25

The chart below shows the dataset for 4,000 drivers, with the distance feature on the x-axis and speeding feature on the y-axis.



Step 2: Choose K and Run the Algorithm

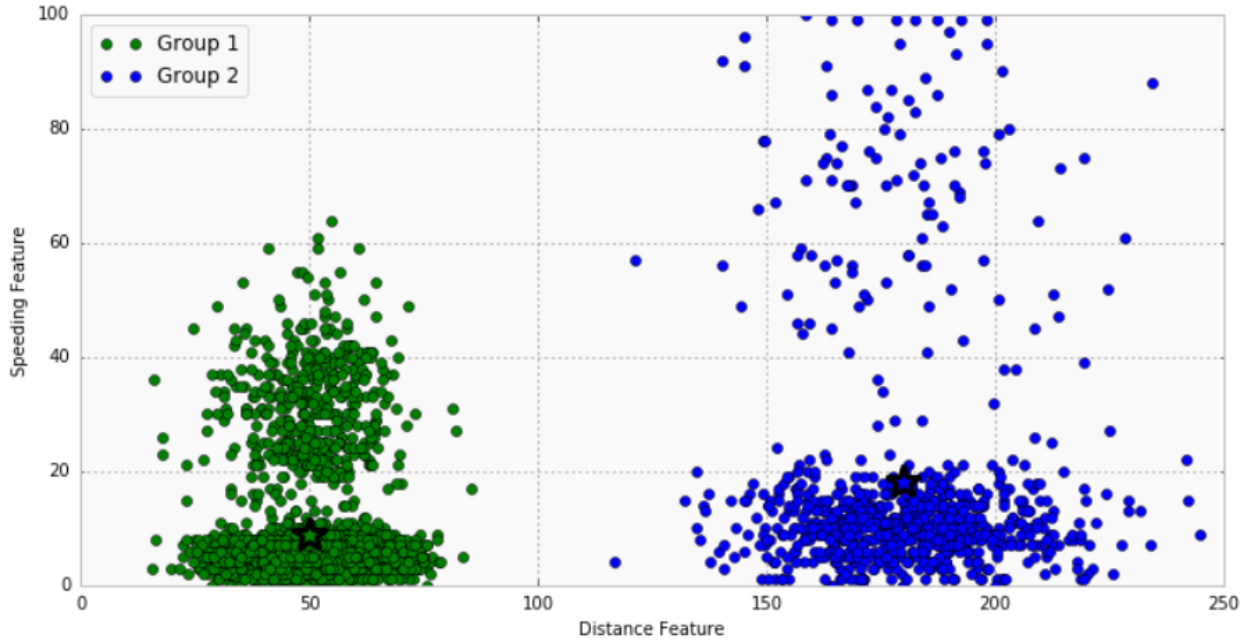
Start by choosing K=2. For this example, use the Python packages scikit-learn and NumPy for computations as shown below:

```
import numpy as np
from sklearn.cluster import KMeans

### For the purposes of this example, we store feature data from our
### dataframe `df`, in the `f1` and `f2` arrays. We combine this into
### a feature matrix `X` before entering it into the algorithm.
f1 = df['Distance_Feature'].values
f2 = df['Speeding_Feature'].values

X=np.matrix(zip(f1,f2))
kmeans = KMeans(n_clusters=2).fit(X)
```

Step 3: Review the Results The chart below shows the results. Visually, you can see that the K-means algorithm splits the two groups based on the distance feature. Each cluster centroid is marked with a star.

- Group 1 Centroid = (50, 5.2)
- Group 2 Centroid = (180.3, 10.5)

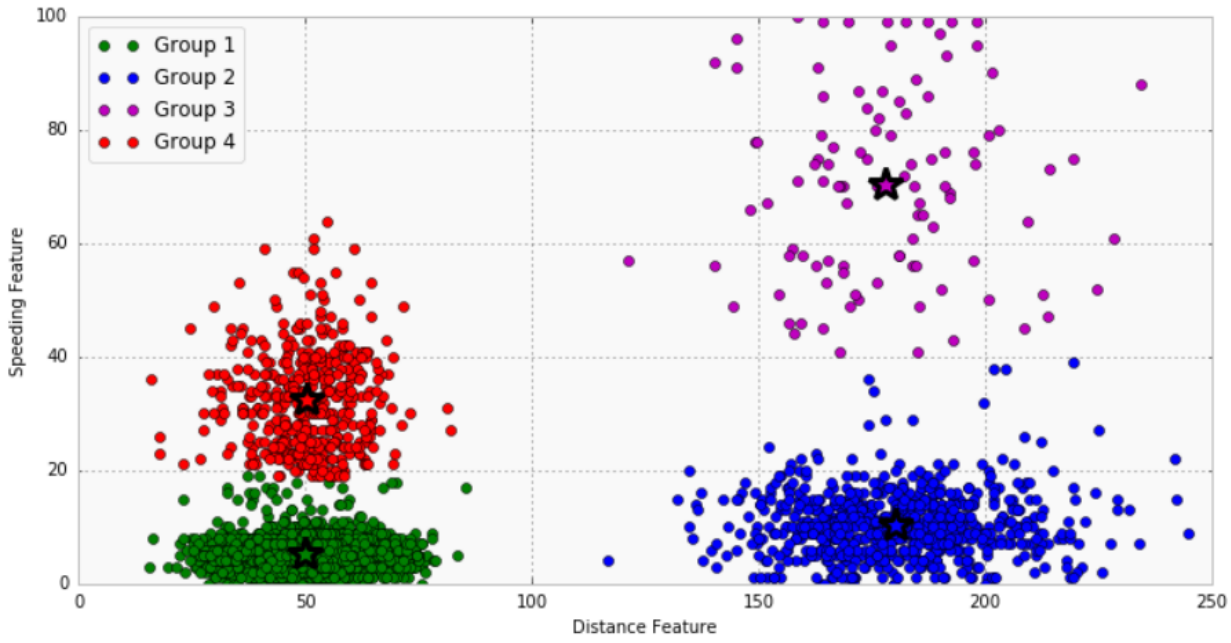
Using domain knowledge of the dataset, we can infer that Group 1 is urban drivers and Group 2 is rural drivers.



Step 4: Iterate Over Several Values of K Test how the results look for K=4. To do this, all you need to change is the target number of clusters in the KMeans() function.

```
kmeans = KMeans(n_clusters=4).fit(X)
```

The chart below shows the resulting clusters. We see that four distinct groups have been identified by the algorithm; now speeding drivers have been separated from those who follow speed limits, in addition to the rural vs. urban divide. The threshold for speeding is lower with the urban driver group than for the rural drivers, likely due to urban drivers spending more time in intersections and stop-and-go traffic.



3.1 Feature Engineering(To be kept in mind)

Feature engineering is the process of using domain knowledge to choose which data metrics to input as features into a machine learning algorithm. Feature engineering plays a key role in K-means clustering; using meaningful features that capture the variability of the data is essential for the algorithm to find all of the naturally-occurring groups.

Categorical data (i.e., category labels such as gender, country, browser type) needs to be encoded or separated in a way that can still work with the algorithm.

Feature transformations, particularly to represent rates rather than measurements, can

help to normalize the data. For example, in the delivery fleet example above, if total distance driven had been used rather than mean distance per day, then drivers would have been grouped by how long they had been driving for the company rather than rural vs. urban.

4 Advantages and Disadvantages of K-Means

K-Means Advantages :

- If variables are huge, then K-Means most of the times computationally faster than hierarchical clustering, if we keep k smalls.
- K-Means produce tighter clusters than hierarchical clustering, especially if the clusters are globular.

K-Means Disadvantages :

- Difficult to predict K-Value.
- With global cluster, it didn't work well.
- Different initial partitions can result in different final clusters.
- It does not work well with clusters (in the original data) of Different size and Different density