# **Geo-Location Clustering with K-Mean**

*By:*

*Praveen Mandadi*

**Introduction and Motivation**

Clustering is a process of grouping a set of data points into clusters so that points that are put within the same cluster are like each other whereas points from different clusters are dissimilar. Clustering has many useful applications for marketing, logistics, and document classification. We clustered geo-location data, which will be naturally visualized.

We implemented the k-means algorithm in Spark to solve the clustering issue in a parallel fashion.

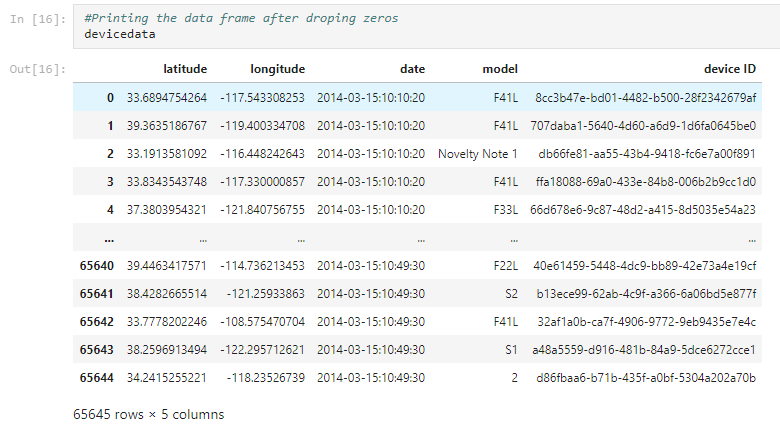
The algorithm iteratively updates the position of k cluster centroids as a distance-based approach until the shift in the mean of centroids converges to alpha=0.1 km where alpha is converged.

**Data Pre-Processing**

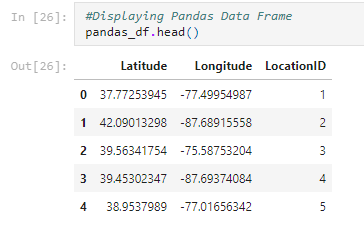
We went through pre-processing steps before implementing the real algorithm to translate the data for later processing into a standardized format. The pre-processing process for device status data is listed below.

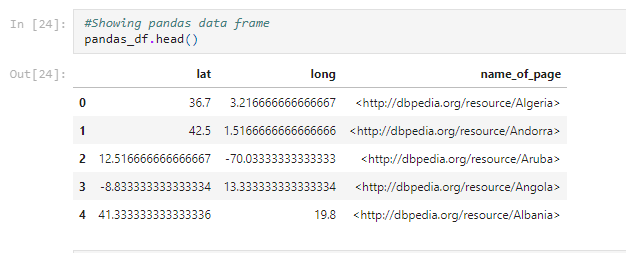
1. Load the dataset.
2. Determine which delimiter to use.
3. Filter out any records which do not parse correctly; each record should have exactly 14 values.
4. Extract the date, model, device ID, and latitude and longitude.
5. date: 1st field
6. model: 2nd field
7. device ID: 3rd field
8. latitude: 13th field
9. longitude: 14th field
10. Store latitude and longitude as the first two fields
11. Filter out locations that have a latitude and longitude of 0.
12. Split the model field that contains the device manufacturer and model name by spaces.
13. Save the extracted data as comma separated values file in the s3://datageo1/results/devicedata.csv directory on AWS S3.
14. Confirm the data in the file(s) was saved correctly.

***Device Status dataset***



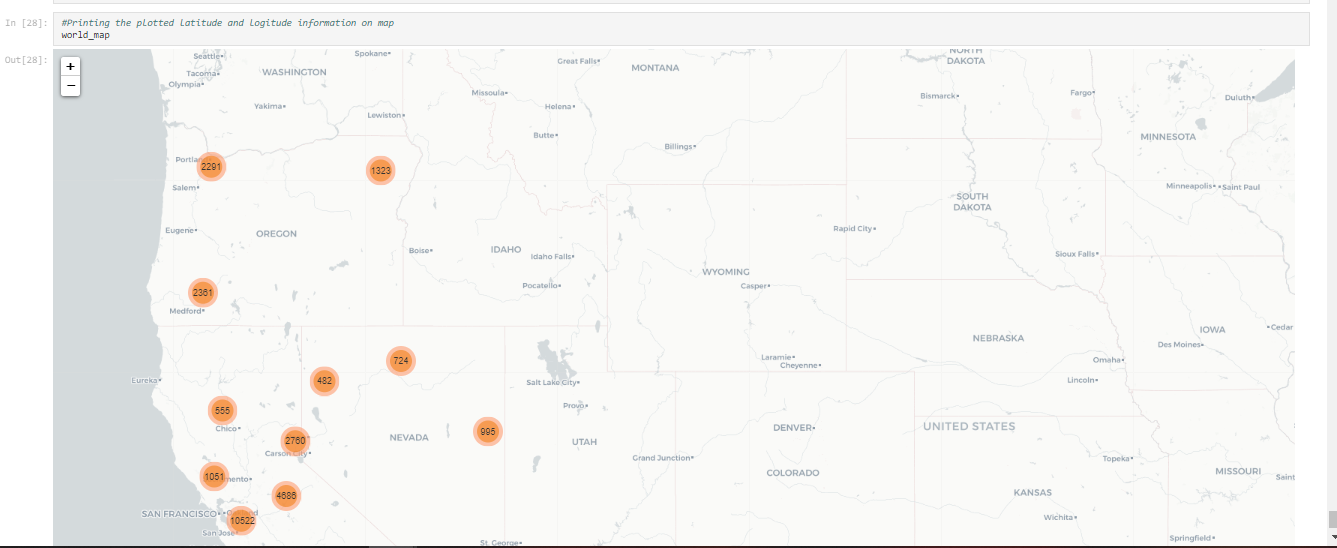
***Synthetic Dataset***



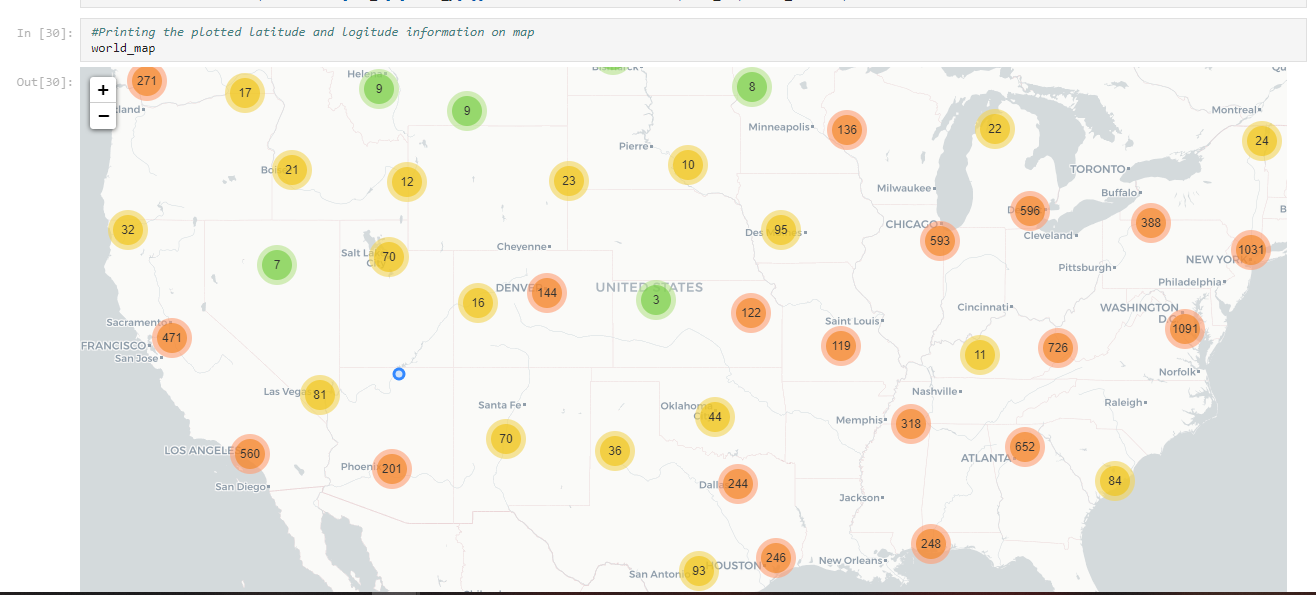
***DBPedia Dataset***

**Visualization**

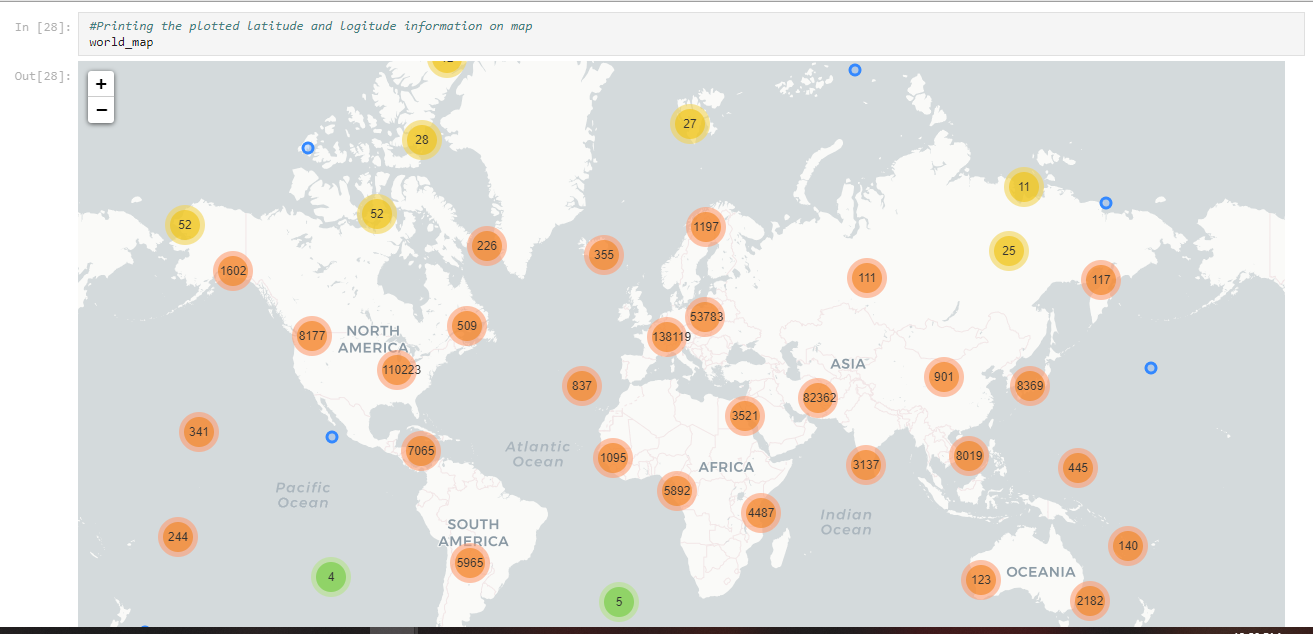
***Mobile Net data***



***Synthetic Location Data***



***DBPedia Location Data***



**Clustering Approach**

In the dataset, the initial centroids are a k-sized random sample of all items. The algorithm assigns each point to its nearest centroid for each iteration, then calculates the new centroids by taking the average of all points in the cluster of that centroid. Using either Euclidean distance or Great Circle distance, the distance between points and centroids is determined - the user sets this parameter. The key difference between the two measurements is that the former measures a straight-line distance between the points in 3D space, while the latter measures the distance around the spherical surface of the Planet.

Though a “perfect” algorithm would iterate until the change in centroid locations converges to 0, this algorithm continues iterating until the sum of all changes in centroid locations converges to α=0.1 km. That is, the algorithm calculates the distance between the current position of each centroid and the former location (using the user-specified distance measure) for each iteration and proceeds to iterate until the sum of these distances is less than 0.1 km for all centroids. This requires larger values of k to converge more precisely than smaller values. Because data this algorithm runs on covers at least an entire continent, we determined that the alpha of 0.1 km or 100 m was a small enough value for this purpose.