A Project Report

on

GEO-LOCATION CLUSTERING USING THE K-MEANS ALGORITHM

by

BHAVANA SRUSHTI
DURGA PRASAD NELAKURTHI
JYOTSHNA AITIPAMULA

Under the Supervision of
Prof. ARDRIANA SULA

Contents

INTRODUCTION

- In this project we will use SPARK to implement an iterative algorithm that solves theclustering problem in an efficient distributed fashion.
- Clustering is the process of grouping a set of objects (or data points) into a set of k clusters of similar objects.
- Thus, objects that are similar should be in the same cluster and objects that are dissimilar should be in different clusters.
- Clustering has many useful applications such as finding a group of consumers with common preferences, grouping documents based on the similarity of their contents, or finding spatial clusters of customers to improve logistics.
- More specific use cases are

  Marketing: given a large set of customer transactions, find customers with similar purchasing behaviors.

  Document classification: cluster web log data to discover groups of similar access patterns.

  Logistics: find the best locations for warehouses or shipping centers to minimize shipping times.
- We will approach the clustering problem by implementing the k-means algorithm.
- k- means is a distance-based method that iteratively updates the location of k cluster centroids until convergence.
- The main user-defined ingredients of the k-means algorithm are the distance function (often Euclidean distance) and the number of clusters k.
- This parameter needs to be set according to the application or problem domain.
- In a nutshell, k-means groups the data by minimizing the sum of squared distances between the data points and their respective closest centroid.

ENVIRONMENT SETUP:

For this Project we have used Aws because local machine takes lot of time to compute. Following are thesteps we followed for the Environment set up

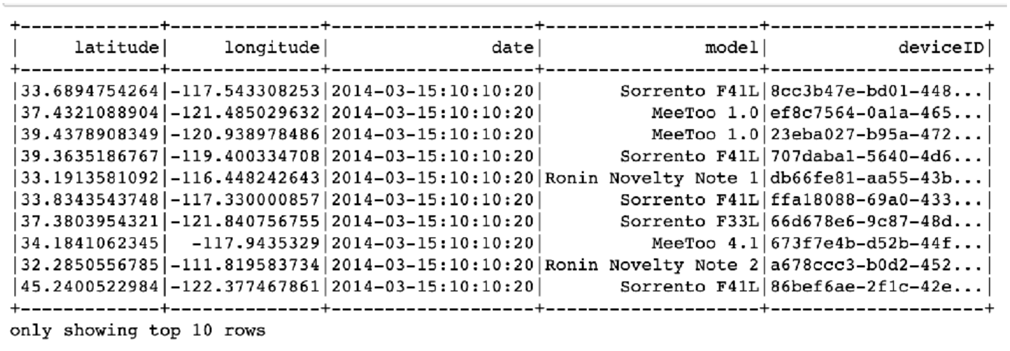
1. Created a S3 bucket, loaded all the data given in the created bucket.
2. Created an EC2 key pair.
3. Created a SPARK cluster, with m4x large instance.
4. In master security group of cluster added a SSH 22 and Custom TCP 8888 ports for accessing the applications.
5. SSH into EMR cluster
6. In super, installed jupyter by using "python3 -m pip install pyyaml ipython jupyter ipyparallel pandas boto -U "
7. By using export PYSPARK_DRIVER_PYTHON=/usr/local/bin/jupyter, set pyspark driver to the location of jupyter.
8. By using export PYSPARK_DRIVER_PYTHON_OPTS="notebook --no-browser --ip=0.0.0.0 --port=8888" set pyspark driver python opts.
9. With the help of pyspark command accessed jupyter notebook on the port 8888.
10. In jupyter installed matplotlib, geopandas , s3fs.

DATA PREPARATION

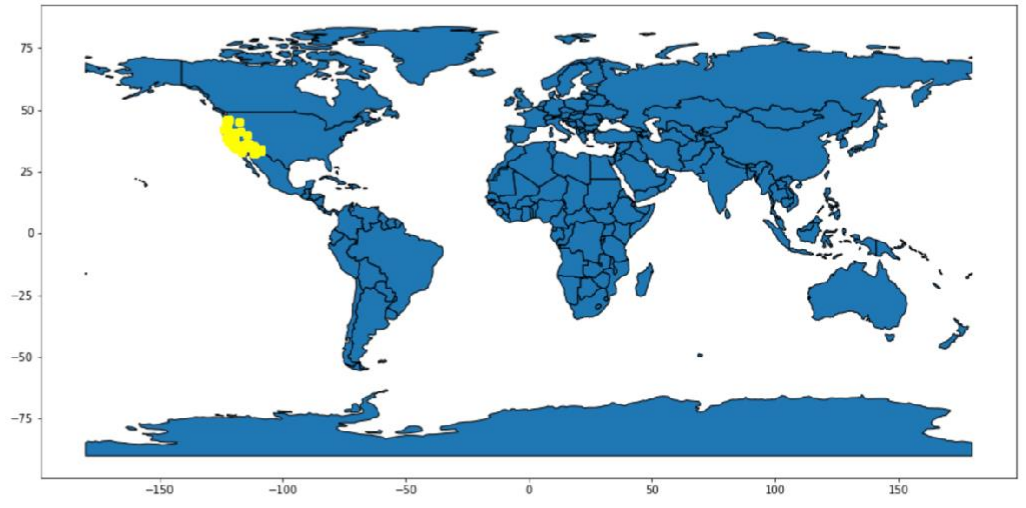
Device status data preparation:

1. Uploaded all the data sets to the S3 bucket.
2. Loaded the data set using “sc.textFile” command.
3. Keenly observed each data set and figured out a regex to use for splitting the data set.
4. From datasets, removed all the unwanted data.
5. Selected the records which have exactly 14 values.
6. Extracted the date (first field), model (second field), device ID (third field), and latitudeand longitude (13th and 14th fields respectively).
7. Selected the latitude and longitude which are not equal to 0.
8. Model field is split into model name and manufacturer.
9. Converted into CSV file and stored the file in the bucket.
10. Plotted all latitude and longitude points using geopandas.

Visualizations:



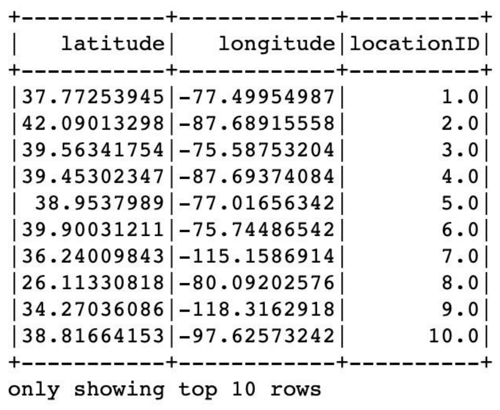
```
+-------------+--------------+-------------------+--------------------+--------------------+
|     latitude|     longitude|               date|               model|            deviceID|
+-------------+--------------+-------------------+--------------------+--------------------+
|33.6894754264|-117.543308253|2014-03-15:10:10:20|       Sorrento F41L|8cc3b47e-bd01-448...|
|37.4321088904|-121.485029632|2014-03-15:10:10:20|          MeeToo 1.0|ef8c7564-0a1a-465...|
|39.4378908349|-120.938978486|2014-03-15:10:10:20|          MeeToo 1.0|23eba027-b95a-472...|
|39.3635186767|-119.400334708|2014-03-15:10:10:20|       Sorrento F41L|707daba1-5640-4d6...|
|33.1913581092|-116.448242643|2014-03-15:10:10:20|Ronin Novelty Note 1|db66fe81-aa55-43b...|
|33.8343543748|-117.330000857|2014-03-15:10:10:20|       Sorrento F41L|ffa18088-69a0-433...|
|37.3803954321|-121.840756755|2014-03-15:10:10:20|       Sorrento F33L|66d678e6-9c87-48d...|
|34.1841062345|  -117.9435329|2014-03-15:10:10:20|          MeeToo 4.1|673f7e4b-d52b-44f...|
|32.2850556785|-111.819583734|2014-03-15:10:10:20|Ronin Novelty Note 2|a678ccc3-b0d2-452...|
|45.2400522984|-122.377467861|2014-03-15:10:10:20|       Sorrento F41L|86bef6ae-2f1c-42e...|
+-------------+--------------+-------------------+--------------------+--------------------+
only showing top 10 rows
```



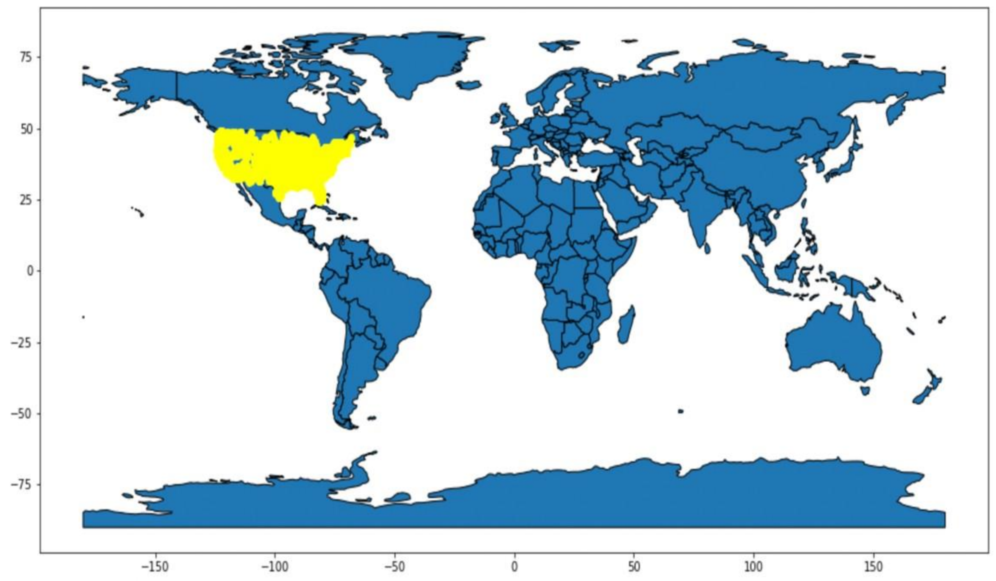
Synthetic clustering data preparation:

1. Uploaded all the data sets to the S3 bucket.
2. Loaded the data set using "sc.textFile" command.
3. Keenly observed each data set and figured out a regex to use for splitting the data set.
4. From datasets, removed all the unwanted data.
5. Extracted the latitude, longitude, and location ID
6. Selected the latitude and longitude which are not equal to 0.
7. Converted into CSV file and stored the file in the bucket.
8. Plotted all latitude and longitude points using geopandas.

Visualizations:



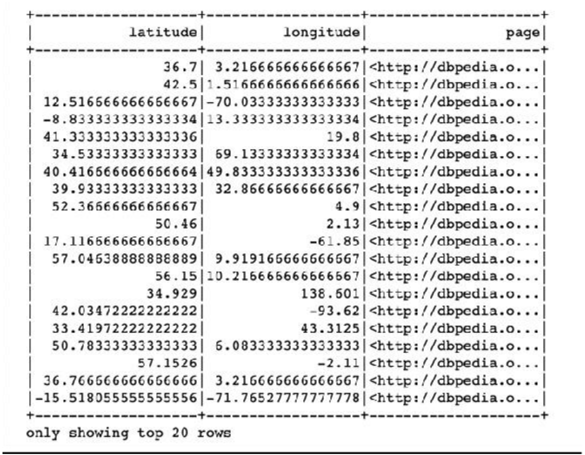
```
+-----------+------------+----------+
|   latitude|   longitude|locationID|
+-----------+------------+----------+
|37.77253945|-77.49954987|       1.0|
|42.09013298|-87.68915558|       2.0|
|39.56341754|-75.58753204|       3.0|
|39.45302347|-87.69374084|       4.0|
| 38.9537989|-77.01656342|       5.0|
|39.90031211|-75.74486542|       6.0|
|36.24009843|-115.1586914|       7.0|
|26.11330818|-80.09202576|       8.0|
|34.27036086|-118.3162918|       9.0|
|38.81664153|-97.62573242|      10.0|
+-----------+------------+----------+
only showing top 10 rows
```



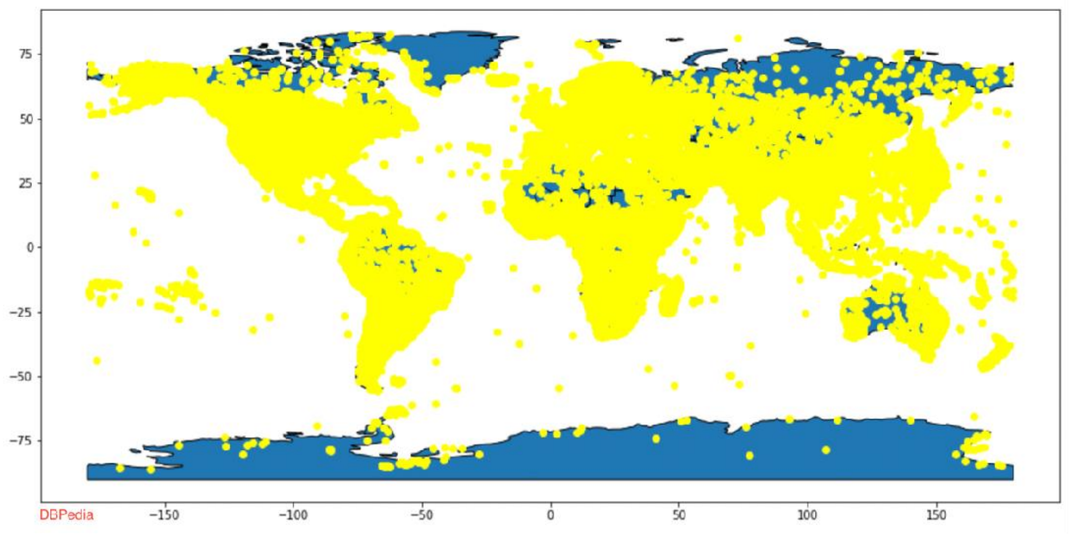
DB pedia location data preparation:

1. Uploaded all the data sets to the S3 bucket.
2. Loaded the data set using “sc.textFile” command.
3. Keenly observed each data set and figured out a regex to use for splitting the data set.
4. Extracted the latitude, longitude, and page
5. Selected the latitude and longitude which are not equal to 0.
6. Converted into CSV file and stored the file in the bucket.
7. Plotted all latitude and longitude points using geopandas.

Visualizations:



```
+-------------------+------------------+--------------------+
|           latitude|         longitude|                page|
+-------------------+------------------+--------------------+
|               36.7| 3.216666666666667|<http://dbpedia.o...|
|               42.5|1.5166666666666666|<http://dbpedia.o...|
| 12.516666666666667|-70.03333333333333|<http://dbpedia.o...|
| -8.833333333333334|13.333333333333334|<http://dbpedia.o...|
| 41.333333333333336|              19.8|<http://dbpedia.o...|
|  34.53333333333333| 69.13333333333334|<http://dbpedia.o...|
| 40.416666666666664|49.833333333333336|<http://dbpedia.o...|
|  39.93333333333333| 32.86666666666667|<http://dbpedia.o...|
|  52.36666666666667|               4.9|<http://dbpedia.o...|
|              50.46|              2.13|<http://dbpedia.o...|
| 17.116666666666667|            -61.85|<http://dbpedia.o...|
|  57.04638888888889| 9.919166666666667|<http://dbpedia.o...|
|              56.15|10.216666666666667|<http://dbpedia.o...|
|             34.929|           138.601|<http://dbpedia.o...|
|  42.03472222222222|            -93.62|<http://dbpedia.o...|
|  33.41972222222222|           43.3125|<http://dbpedia.o...|
|  50.78333333333333| 6.083333333333333|<http://dbpedia.o...|
|            57.1526|             -2.11|<http://dbpedia.o...|
| 36.766666666666666| 3.216666666666667|<http://dbpedia.o...|
|-15.518055555555556|-71.76527777777778|<http://dbpedia.o...|
+-------------------+------------------+--------------------+
only showing top 20 rows
```

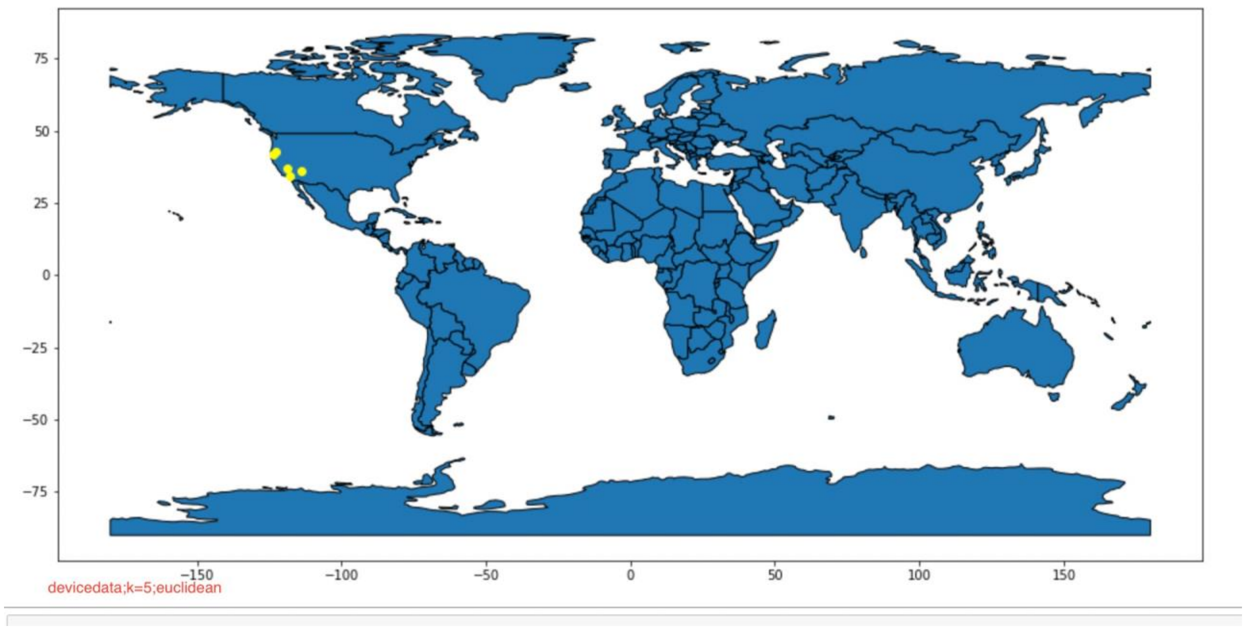


<u>**KMEANS IMPLEMENTATION IN PYSPARK**</u>

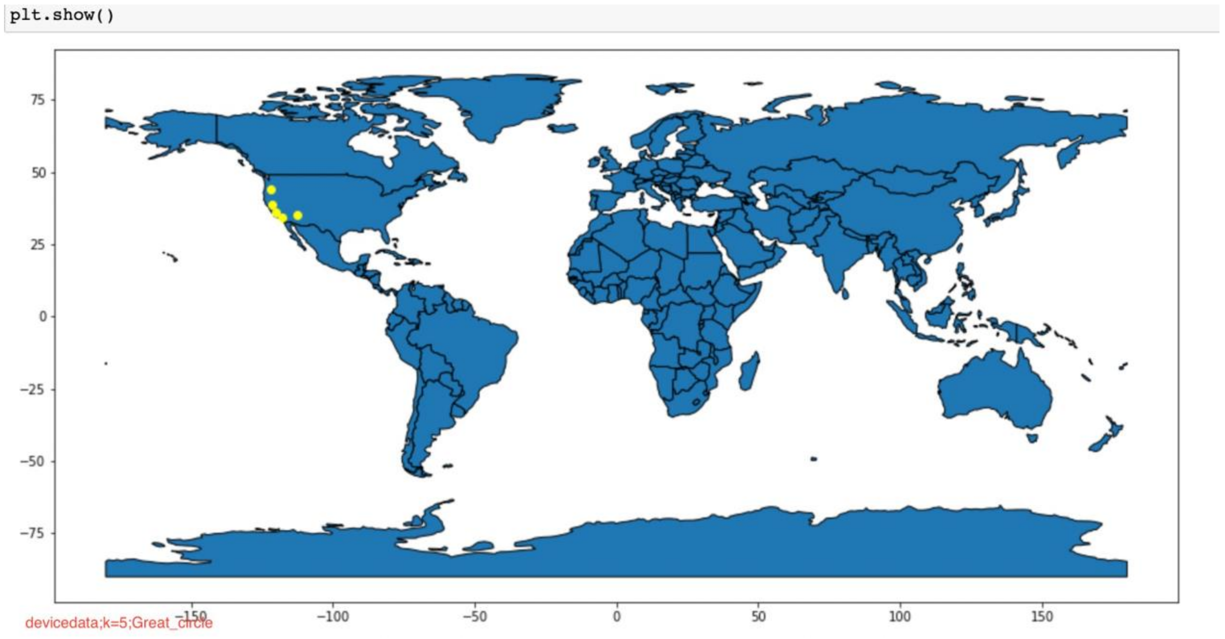
1. Data is in the form of latitude-longitude pairs
2. Here we are comparing clusters using Euclidean distance and great circle distance
3. Spark matplotlib has an inbuilt function to calculate Euclidean distance but not great circledistance
4. So, we defined four functions called Euclidean, Great circle, Equality, Centroid.In each of the function definition calculated using respective formula
5. Initialized the functions by giving number of clusters (k), number of iterations, a distance metric, Convergence distance and by selecting the initial centroids.
6. Computed the distance from each pair to the number of clusters
7. Assigned labels to the latitude-longitude pairs, which have shortest distance from thepoints to the centroid
8. Based on the labels computed the new centroids.
9. Computed the distance between new centroids and old centroids
10. Continued the process if the old and new centroids were not equal, until the number of iterations comes to an end, or if the distance between them is less than the convergence until they converge
11. After finishing this entire process, we get new centroids which are the clusters of our data.

<u>RESULTS</u>:

DEVICE DATA LOCATION, EUCLIDEAN DISTANCE USING, K =5

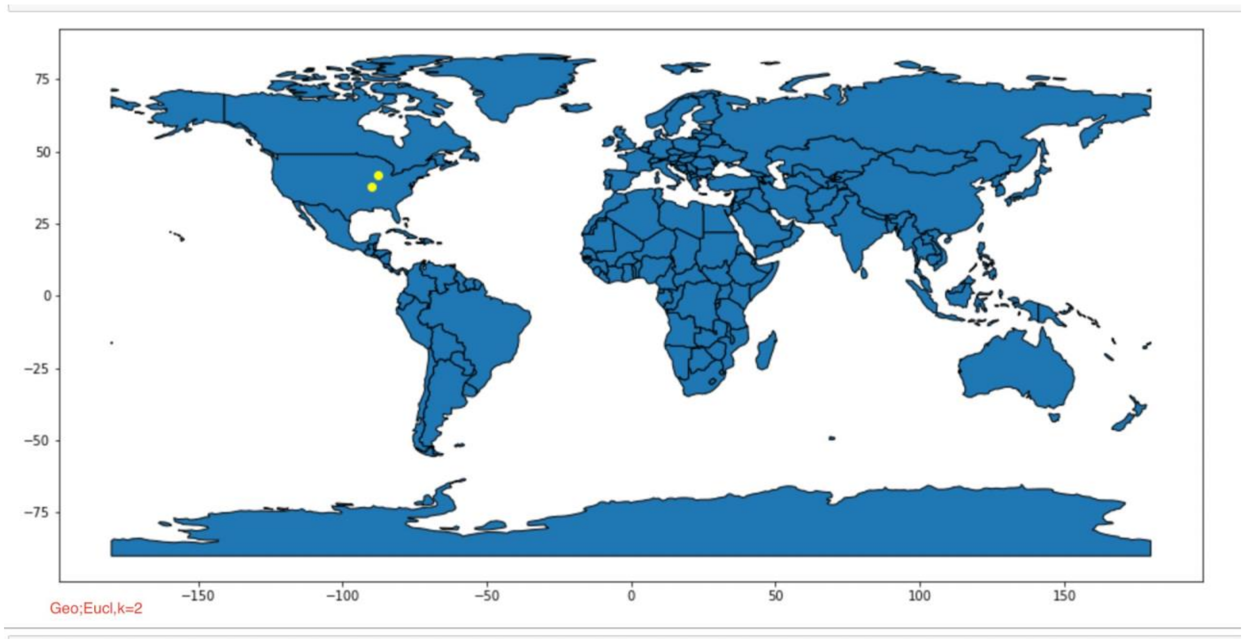


DEVICE DATA LOCATION, GREAT CIRCLE DISTANCE USING, K=5

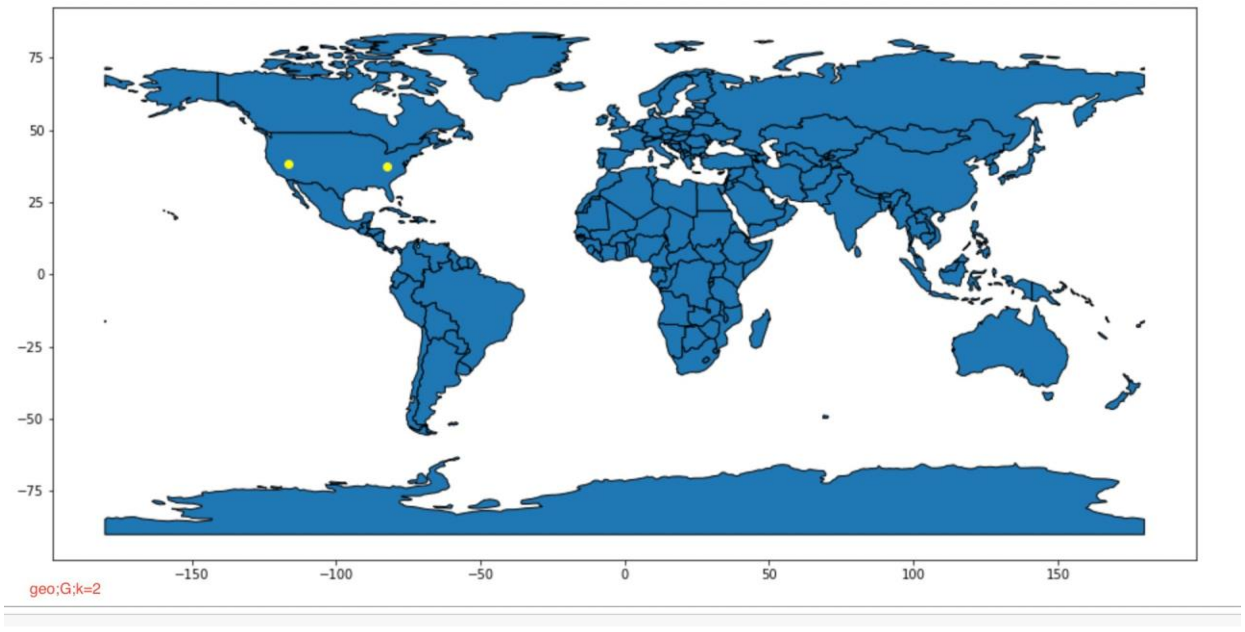


```
plt.show()
```

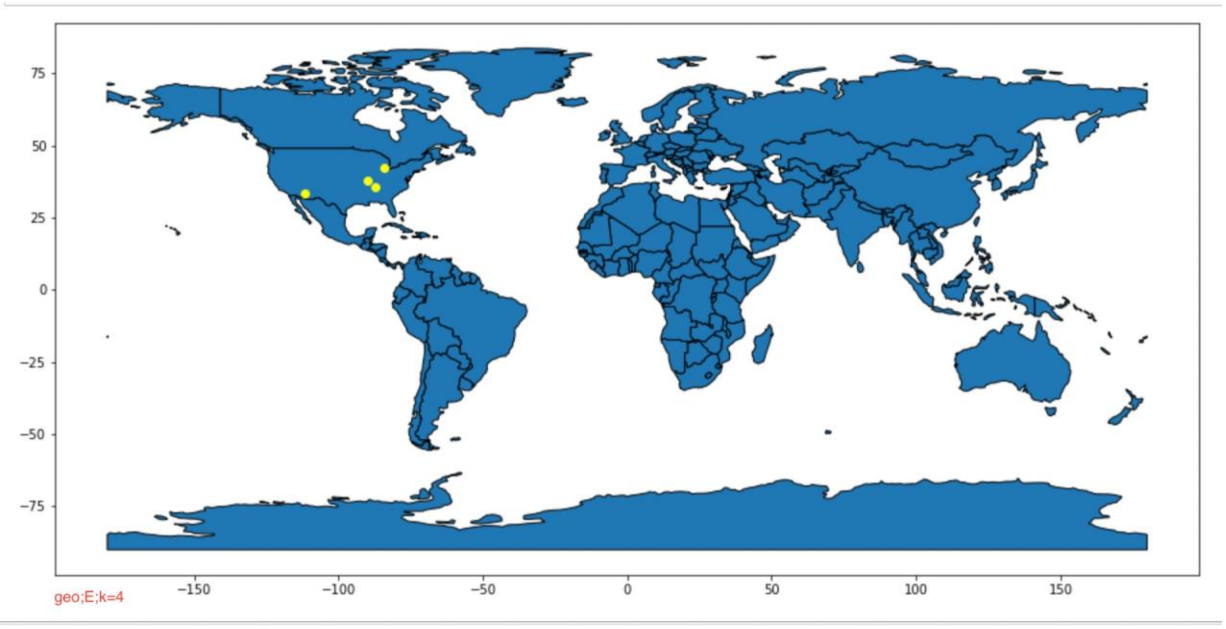
SYNTHETIC LOCATION DATA, EUCLIDEAN DISTANCE USING, USING K=2



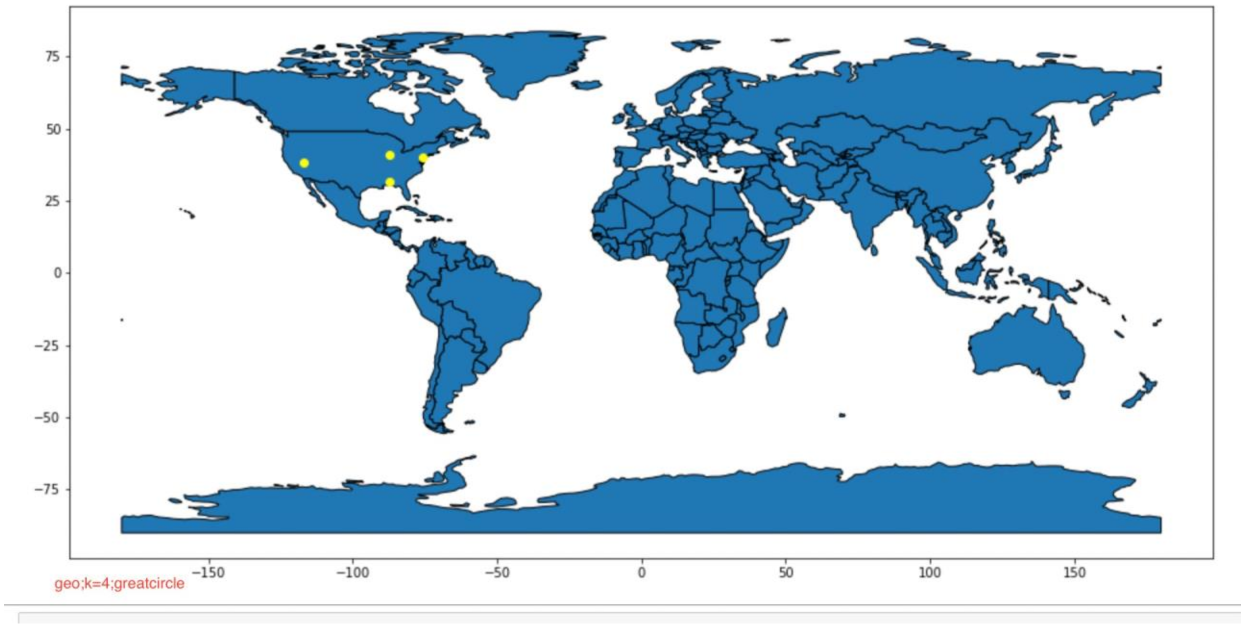
SYNTHETIC LOCATION DATA, GREAT CIRCLE DISTANCE USING, USING K=2



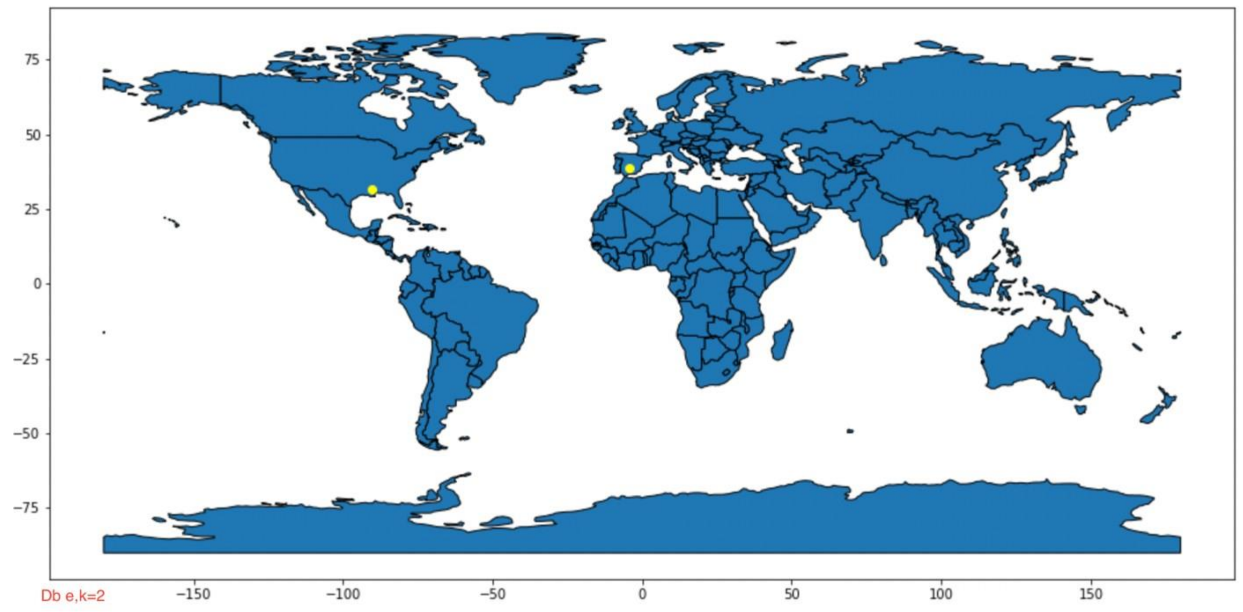
SYNTHETIC LOCATION DATA, EUCLIDEAN DISTANCE USING, USING K=4



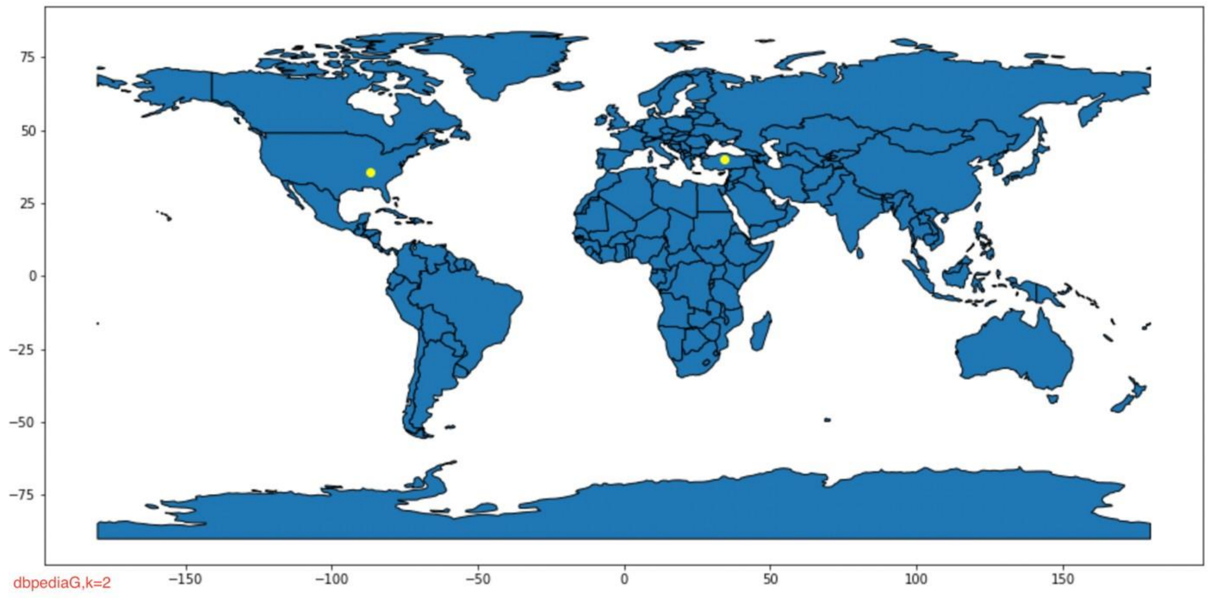
SYNTHETIC LOCATION DATA, GREAT CIRCLE DISTANCE USING, USING K=4



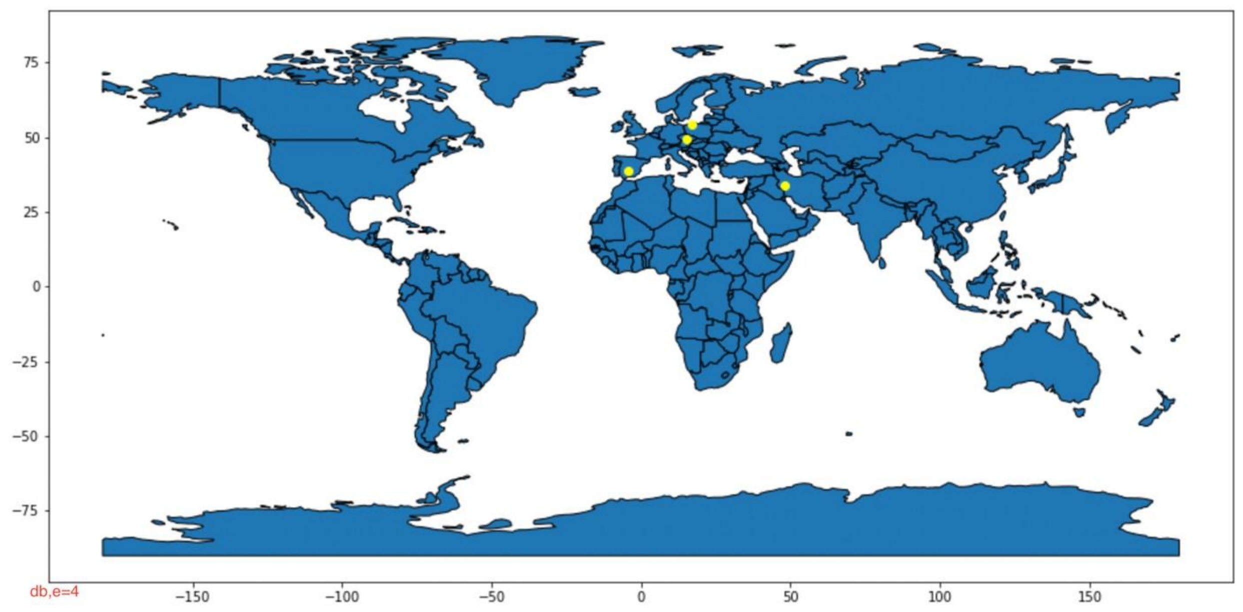
DB PEDIA LOCATION DATA EUCLIDEAN DISTANCE USING, USING K=2



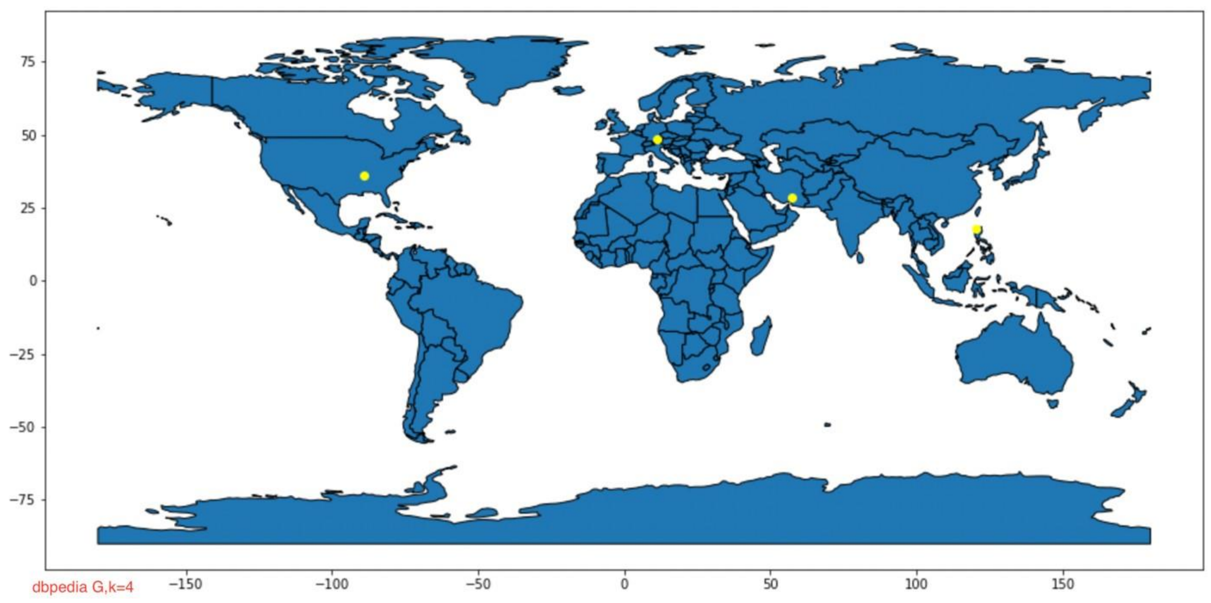
DB PEDIA LOCATION DATA GREAT CIRCLE DISTANCE USING, USING K=2



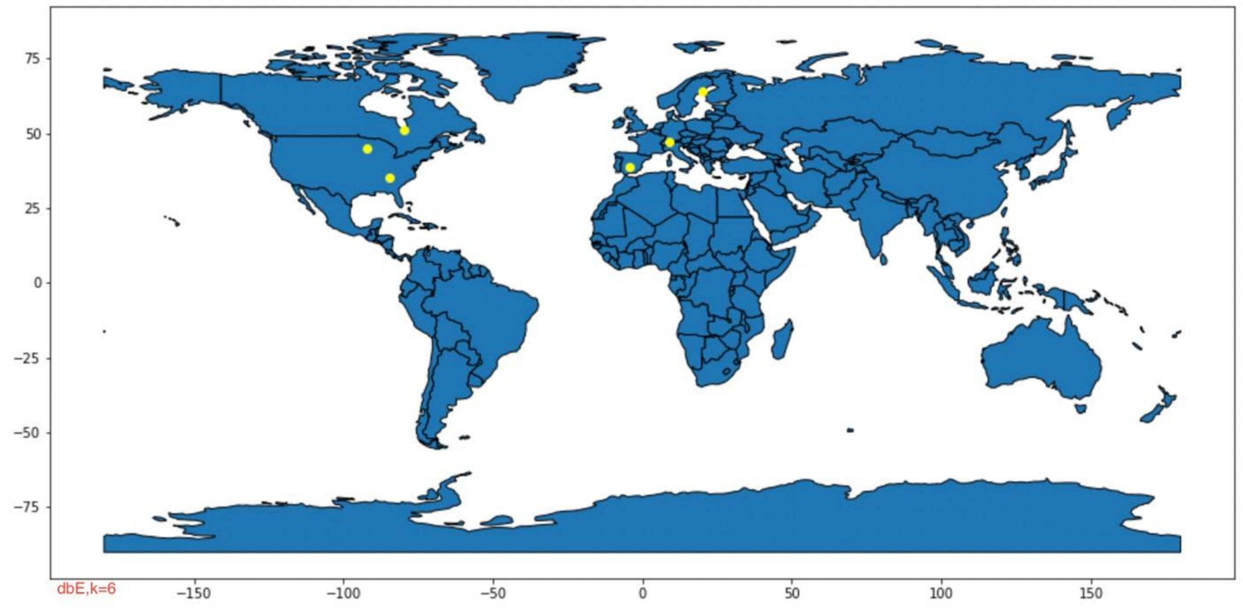
DB PEDIA LOCATION DATA EUCLIDEAN DISTANCE USING, USING K=4



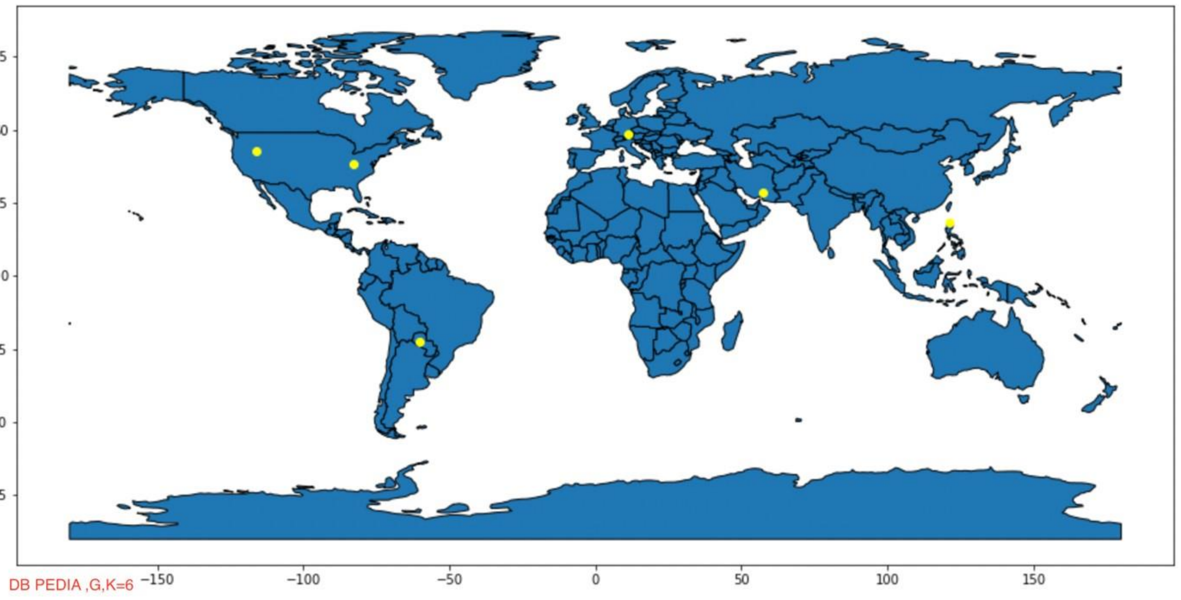
DB PEDIA LOCATION DATA GREAT CIRCLE DISTANCE USING, USING K=4



DB PEDIA LOCATION DATA EUCLIDEAN DISTANCE USING, USING K=6



DB PEDIA LOCATION DATA GREAT CIRCLE DISTANCE USING, USING K=6



RUNTIME ANALYSIS:

Dataset	**Persist**	**k**	**Distance metric**	**Run time**
Device data	yes	5	Great_circle	2.7m
Device data	no	5	Great_circle	2.9m
Synthetic Data	yes	4	Euclidean	1.8m
Synthetic Data	no	4	Euclidean	1.8m
DBpedia data	yes	4	Euclidean	2.9m
DBpedia Data	no	4	Euclidean	3.2m

OBSERVATIONS:

There is a relationship between time and persist. The one which is cached took less time compared to the one which is cached.

CONCLUSION:

As per our point of view, we observed that certain k values are not perfect, few overlapped. K-means algorithm might be reliable for this project, but we don’t think it will be robust enough forothers.