Final Project Report

Nancirose Piazza

May 6th 2020

Spring 2020 – Distributed and Scalable Data

Motivation:

I think geographical data is an example of large data that is in the interest of most people. It shows the clusters of people and possibly the behavior of people over a time series. My project is to complete the tasks adequately and establish methods of approaching data that are more efficient. Location is a primitive requirement to humans, regardless of how we measure it. Location is the reason why some businesses succeed, and others fail, as well as why many businesses move their headquarters (to evade higher business tax.) It's a requirement I believe all business data scientists should have insight about.

System Configuration:

Standard EMR cluster – m4.xlarge, 2 slave nodes, 1 master node

Documentation of Approach and Big Data Application Analysis:

Objective: Complete the tasks given in project final assignment, plot meaningful graphs and produce compiling time analysis. The order of the headings will correspond to the assignment.

Problem 2

Step 1

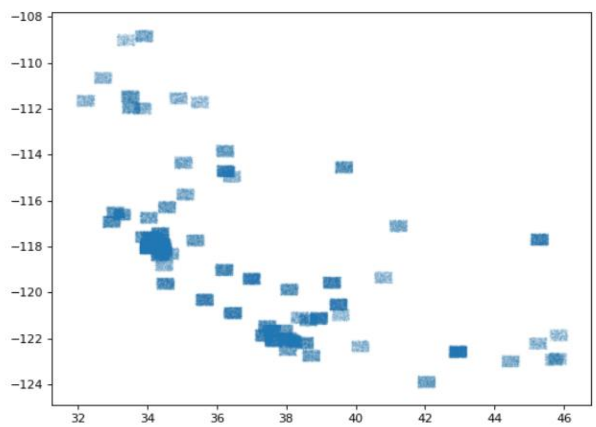
Data from devicestatus.txt required data scrubbing since (1) there were multiple delimiters such as |, comma, and forward slash, (2) the number of features varied among the samples. The following steps were taken:

1. We import the data from the S3 bucket
2. Use .flatMap and split the data on \n to separate the samples
3. Use regular expression library method re.sub and replace and of the delimiters with a comma
4. Use .split() on commas
5. Filter for the first two features and last two features. Conventionally this is where date, model, latitude, and longitude would be stored if correct.
6. Filter samples where the last two features are of type float. OR if it's still consider a string, consider using re.sub() and capture any patterns matching digits with a floating point and use backreference to return the value. Then check for float type.
7. We map attempt to cast the latitude and longitude to floats.
8. Filter entries where latitude or longitude are equal to 0
9. Split model into manufacturer and model_device using .split()[0] and .split()[1]
10. We use rdd.coalesce(1).saveAsTextFile() to save it to the bucket.

Step 2

Using the sample_geo.txt data, a bit of preprocessing was required since the incoming data was considered a string rather than numerical values. We use similarly the same steps of splitting on newlines, discarding the header line, splitting the features on \t and casting all features to numeric, floats or integer.

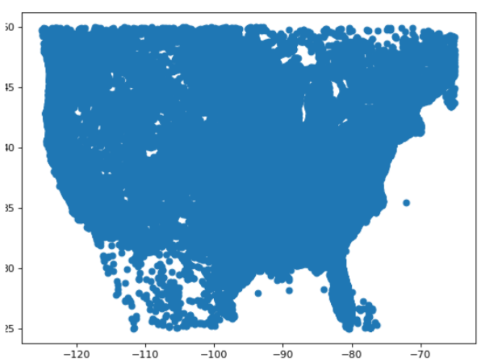
We use matplotlib.pyplot to plot our data, so we have to collect the data from the rdd using .collect(), though in practice, a finite number with .take() is better.

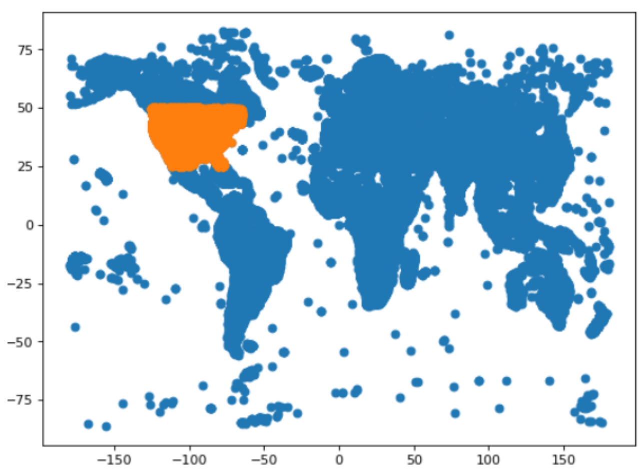


Visually this is a bit bizarre and I cannot recognize it as a continent; however, by evaluating a single sample, we can see that this is California State.

Step 3

Similarly for geo_sample.txt, we process the string latitude and longitude to numerics again and plot. We filtered out the longitude and latitude regions of the continuous United States.





Problem 3

Step 2

The documentation here: https://spark.apache.org/docs/latest/mllib-clustering.html#k-means describes that the parameter epsilon is the Convergence threshold; however, with KMeans from pyspark.mllib.clustering, it is not required to be set. The alternative conclusion is that it is calculated internally. Still, considering the distribution of points and many outliers, we should expect by running kmeans first and calculating the distance metrics, then consider the standard deviation of the distance metrics. Then, take the standard deviation and divide it by the sum of the distance metric to get a small value less than 1. This is would be our proportional epsilon to the sum of the distance metric which we may attempt to apply to other k-clusters.

Step 3

Using the **device location data** and clustering for 5 groups we get a column called predictions. Then, we use the prediction as positional values to obtain the center cluster's longitude and latitude:

```
+-----------------+------------------+----------+---------------+----------------+
|original_latitude|original_longitude|prediction|center_latitude|center_longitude|
+-----------------+------------------+----------+---------------+----------------+
|        33.689476|       -117.543304|         1|      34.297184|      -117.78653|
|         37.43211|        -121.48503|         0|       38.02865|      -121.23352|
|         39.43789|        -120.93898|         0|       38.02865|      -121.23352|
+-----------------+------------------+----------+---------------+----------------+
only showing top 3 rows
```

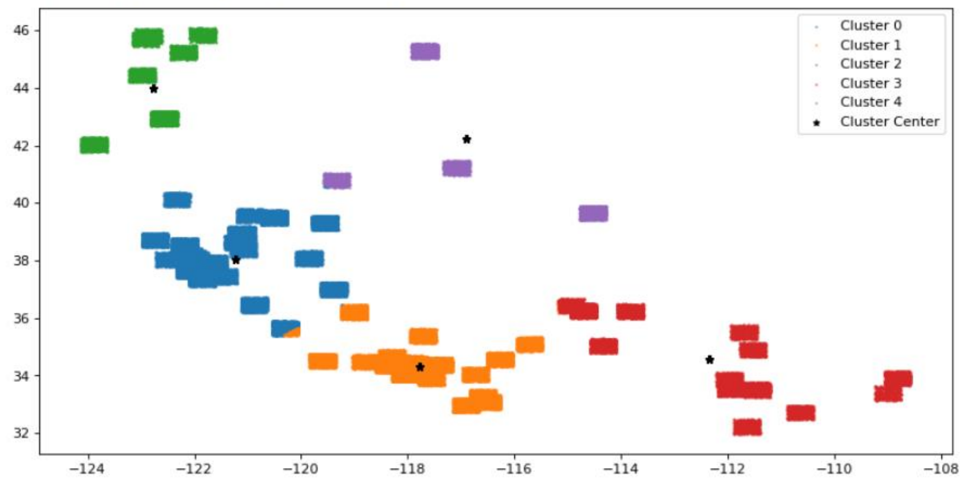
We manually calculated the great circle distance and Euclidean distance through creating a column from aggregating the latitude and longitude columns.

```
+-----------------+------------------+---------------+----------------+-----------------+-------------------+
|original_latitude|original_longitude|center_latitude|center_longitude|          gc_dist|            eu_dist|
+-----------------+------------------+---------------+----------------+-----------------+-------------------+
|        33.689476|       -117.543304|      34.297184|      -117.78653|71197.25683187979|0.42846743379777763|
|         37.43211|        -121.48503|       38.02865|      -121.23352|69922.61967336302|0.41911582615284715|
|         39.43789|        -120.93898|       38.02865|      -121.23352|158769.1391050153| 2.0727134647313505|
+-----------------+------------------+---------------+----------------+-----------------+-------------------+
only showing top 3 rows
```

We can use spark SQL commands to aggregate the average of the distance mtetrics.

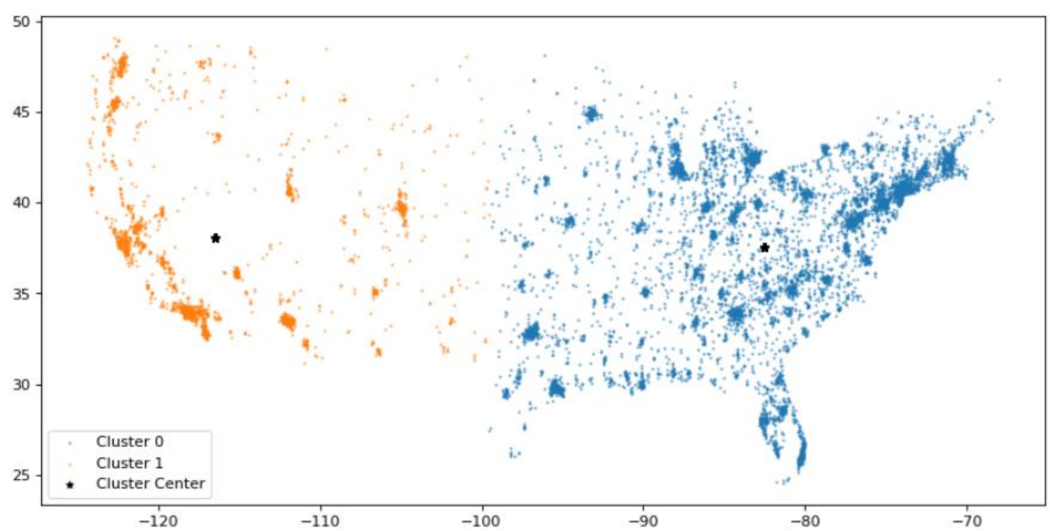
```
+--------------------------------------------+--------------------------------------------+
|(sum(gc_dist) / CAST(count(gc_dist) AS DOUBLE))|(sum(eu_dist) / CAST(count(eu_dist) AS DOUBLE))|
+--------------------------------------------+--------------------------------------------+
|                           132700.5067084632|                           2.536413567676747|
+--------------------------------------------+--------------------------------------------+
```

We can also plot the original coordinates and plot them by color based on their prediction classification.



Using the **synthetic location data** and clustering for 2 and 4 groups:

K=2:



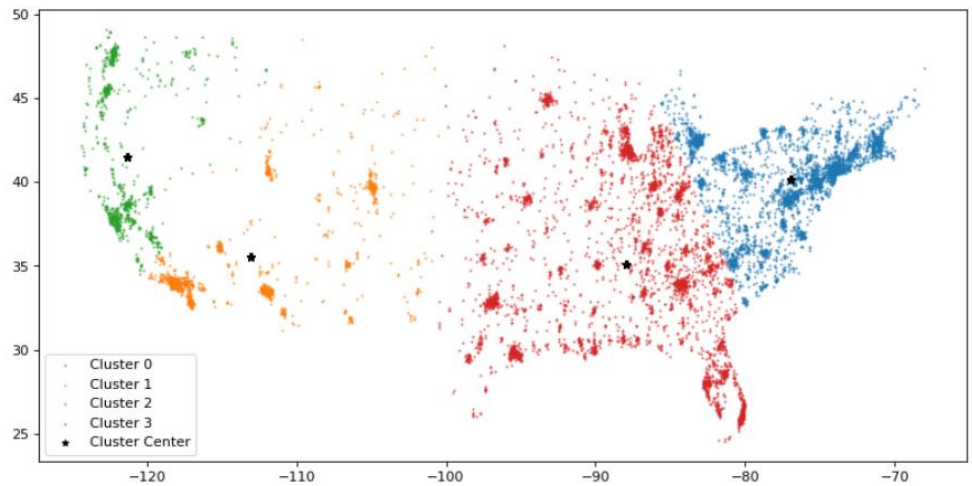
original_latitude	original_longitude	center_latitude	center_longitude	gc_dist	eu_dist
37.77254	-77.49955	37.564747	-82.55711	445697.46533880796	25.622129831521306
42.090134	-87.689156	37.564747	-82.55711	667018.4268019216	46.816980165967834
39.56342	-75.58753	37.564747	-82.55711	645216.0229611602	52.569759438571054

only showing top 3 rows

Mean distance metrics:

(sum(gc_dist) / CAST(count(gc_dist) AS DOUBLE))	(sum(eu_dist) / CAST(count(eu_dist) AS DOUBLE))
730104.5231404818	70.63638292632837

K=4:

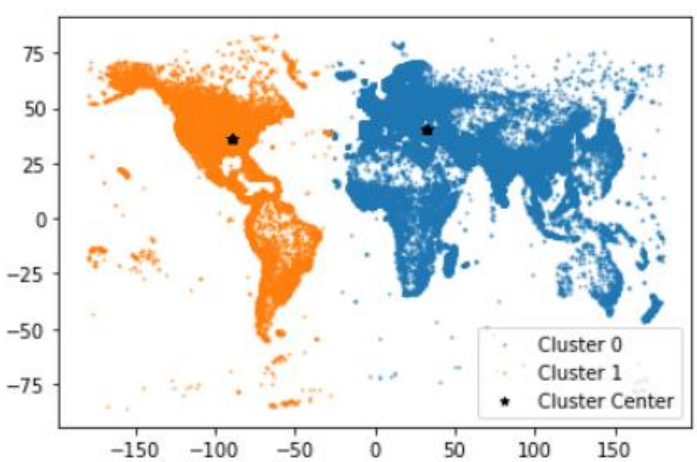


(sum(gc_dist) / CAST(count(gc_dist) AS DOUBLE))	(sum(eu_dist) / CAST(count(eu_dist) AS DOUBLE))
532013.1570450937	36.665836915395765

We see that many of the clusterings are recognizibly similar to how we'd partition the United States. Eg. Northeast, Southeast, the Middle West, and West Coast.

Using the **DBpedia** data for 2,4,6 clusters:

K=2:



Metrics:

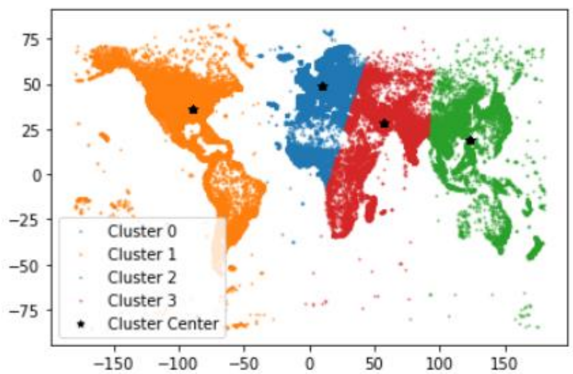
original_latitude	original_longitude	center_latitude	center_longitude	gc_dist	eu_dist
36.7	3.2166667	40.182056	32.483273	2566289.6920982013	868.6589019522216
42.5	1.5166667	40.182056	32.483273	2583587.1626723097	964.3035582459561
12.516666	-70.03333	35.941673	-88.83209	3214123.721362224	902.1243039452966

only showing top 3 rows

Mean distances:

(sum(gc_dist) / CAST(count(gc_dist) AS DOUBLE))	(sum(eu_dist) / CAST(count(eu_dist) AS DOUBLE))
2526786.4716782477	1242.8234952762402

K=4:

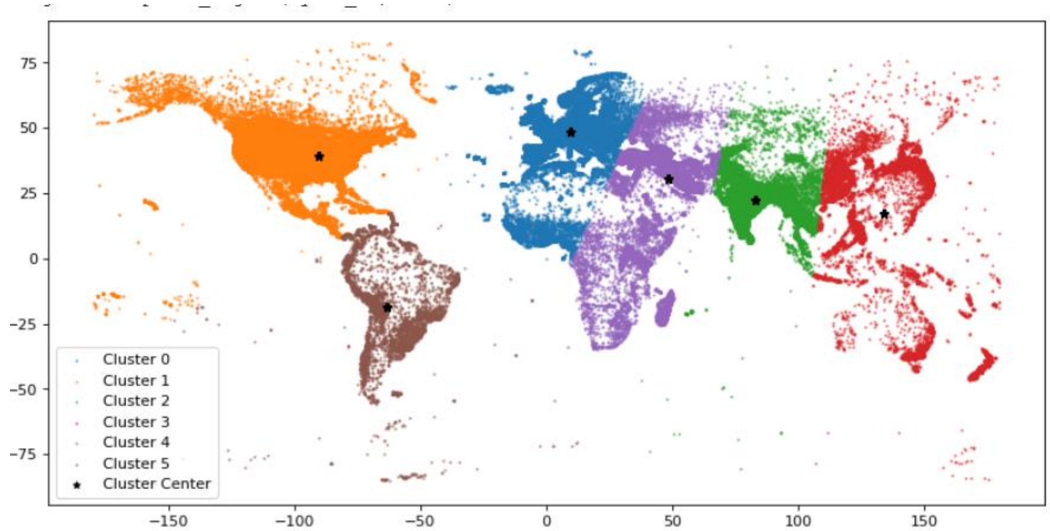


original_latitude	original_longitude	center_latitude	center_longitude	gc_dist	eu_dist
36.7	3.2166667	48.436523	9.999369	1416923.7066886013	183.75101049274076
42.5	1.5166667	48.436523	9.999369	933439.1195545978	107.19854807519368
12.516666	-70.03333	35.934708	-88.915085	3218406.521958237	904.9252141390025

only showing top 3 rows

(sum(gc_dist) / CAST(count(gc_dist) AS DOUBLE))	(sum(eu_dist) / CAST(count(eu_dist) AS DOUBLE))
1431525.8149716915	385.8855322130194

K=6:

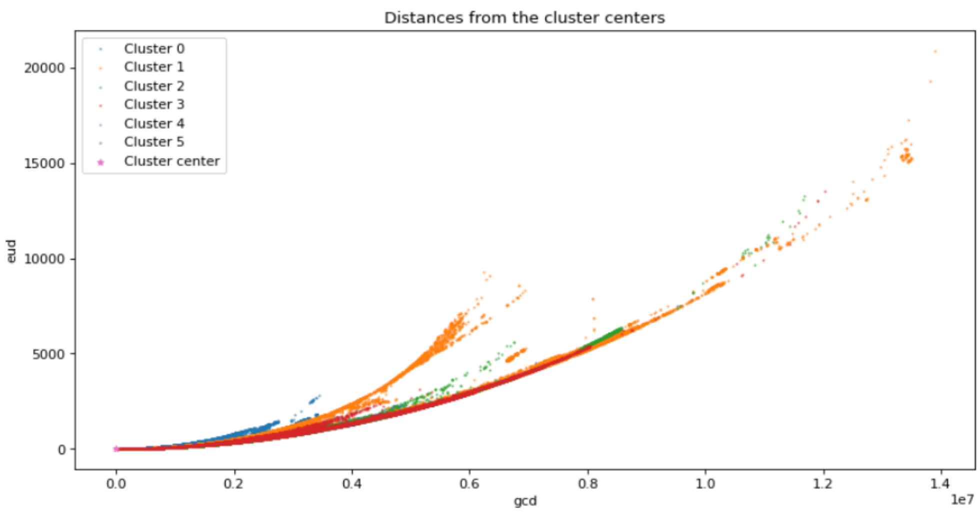


original_latitude	original_longitude	center_latitude	center_longitude	gc_dist	eu_dist
36.7	3.2166667	48.59884	9.7733	1426288.6588699806	184.57178927056543
42.5	1.5166667	48.59884	9.7733	933432.3362596794	105.36783580617339
12.516666	-70.03333	-18.668272	-63.107796	3549762.674538302	1020.4634491852521

only showing top 3 rows

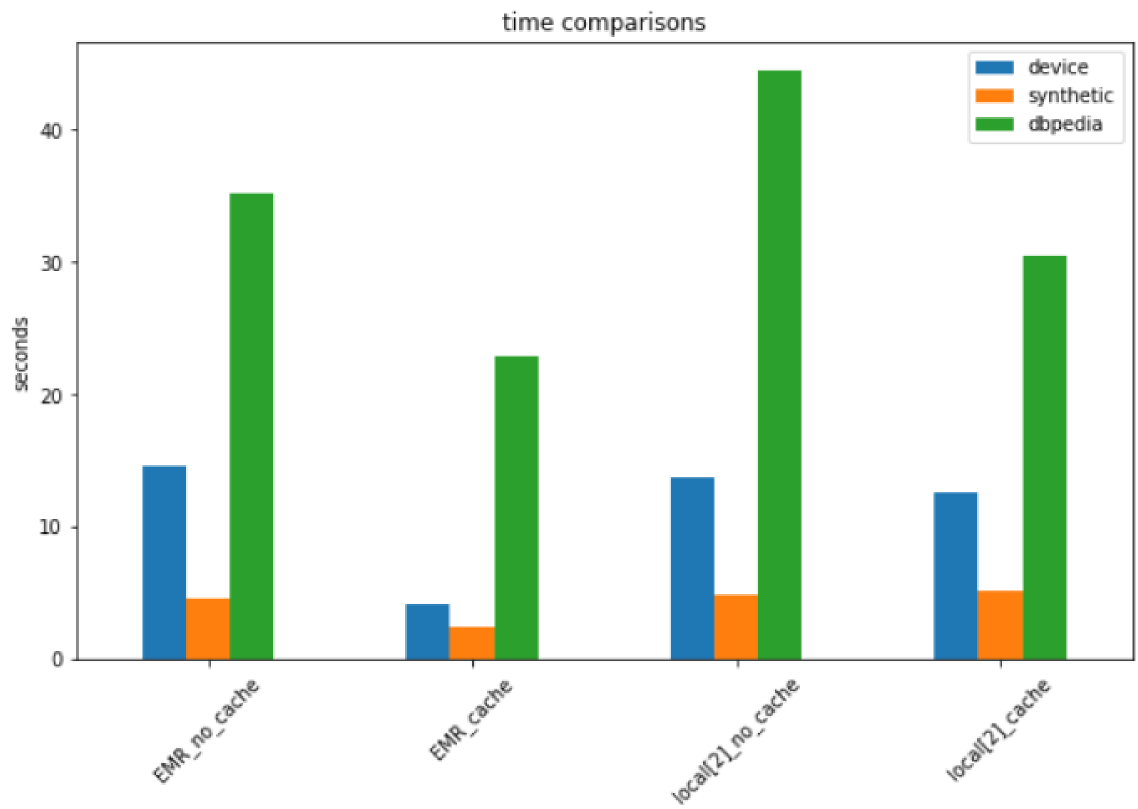
(sum(gc_dist) / CAST(count(gc_dist) AS DOUBLE))	(sum(eu_dist) / CAST(count(eu_dist) AS DOUBLE))
1192528.1209456117	264.35301051973346

Recognizibly we see that with lower k clusters, more points partitioned by their longitude value than latitude. This generally makes sense since longitude contains the most variance out of the two coordinates. It seems that grouping by 4 or 6 clusters represents generally our continents and it’s amazing that, as people with pattern recognition, we’ve established continents as such before being able to compute clusters numerically.



This is a graph of the 6-mean cluster distance metrics. We can see that certain clusters contain more outliers than others. The reason we should consider this plot is because any 2D plot of the Earth is a manifold, a flat representation of the dimensions in Euclidean distance despite having a curve to the topology; however, since longitude and latitude are treated orthogonal from one another, our clustering suffers from being unable to estimate the true distances from the clusters using Euclidean distance. Instead, by looking the comparison between Euclidean and great circle distance from their cluster center located at 0,0 on the graph above, we get a better idea of where Euclidean distance falls short in representing the distance. Eg. Many of the plots in orange have higher Euclidean distance, but only finitely few have higher great circle distance. This is probably due to Euclidean distance’s incapability to represent fundamental regions of the Earth’s topology. Both distances agree that outliers like islands are numerically further away from their centers.

Step 4:



Here is a graph that compares four different environments for running the kmeans clustering for all three datasets, respectively K=5, K=2,4 and K=2,4,6. As we can see, there is an exponential relationship to time when able to cache/persist the rdds and have a distributed system for parallel computation. Of course, this supports that non-pseudo clusters are better because pseudo-clusters are sequential, and caching reduces the time taken assuming some repetition.

This part of the assignment could've done through spark—submit where the times would've been appended to a textfile, which would've made the process more documentary from the Cluster Manager page and extra statistics; however it wasn't until after doing it manually did I realize that would've been a better option. (The ipynb could easily be converted to a spark-submit job since the ipynbs are modified to run using the runall option.) Still, the task was completed and since I didn't use spark-submit, the times recorded are more accurate to the time taken to run the kmean models separately from the preprocessing steps.

Final Conclusions:

If we consider the type of data the devicestatus.txt is, if we were a service provider, we would know where data would probably be bottlenecked if our services went down. We would mitigate this by stationing more equipment or generators in those locations. But, back to the theorical concept of clustering of n dimensions, we want to consider the nature of the data like the topology when determining the distance metric. Geographical data is easier to understand, but it's a manifold, we may want data of higher dimensions that we do not have intuition about. We do however, understand that humans are good at pattern recognition, including clustering since we’ve established many of the clusters to be partitions of continents we can agree makes sense in their k-cluster. I believe further research in geographical data is dependable on the domain, but is fundamental to all domains.