University of New Haven
TAGLIATELA COLLEGE OF ENGINEERING
Electrical & Computer Engineering and Computer Science

Electrical & Computer Engineering & Computer Science (ECECS)

GEO-LOCATION CLUSTERING USING K MEANS ALGORITHM

SPRING 22

CONTENTS

Project Name

Executive Summary

Our project is geo-location clustering using k means algorithm, Geolocation data is information that can be used to identify an electronic device's physical location. Clustering is the process of grouping a set of data points into a set of similar groups or clusters. We used k-means clustering in this project.

Team Members:
Name 1:Naga Yaswanth Bodduluri
Name 2: Akash Raju Dasararaju
Name 3: Saismaran Thanneru

Questions?
Contact: Naga Yashwant Bodduluri

Title of Project

GEO-LOCATION CLUSTERING USING K-MEANS ALGORITHM

Highlights of Project

- Data scrubbing
- Using spark to implement algorithm
- Using k-means algorithm
- Finding Euclidean distance
- Finding Great circle distance

Submitted on:

05/03/2022

Abstract

Geolocation data is information that can be used to determine the physical location of an electronic device. Clustering is the process of categorizing a set of data points into similar groups or clusters. K means cluster was adopted for find distances. By using different distance methods like euclidean and great circle distances we get to know the distance between k points in a cluster. The great circle distance, unlike the Euclidean distance, takes into account the fact that two points are on the surface of a sphere. It was analyzed that great circle distance has least in distance when compared to euclidean distance.

Executive Summary

Introductory Section

Following are the datasets used in our projects

- devicestatus.txt
- lat_songs.txt
- sample_geo.txt

Task is to
use PySpark in Jupyter Notebooks to:

- Load the dataset
- Determine which delimiter to use
- Filter out any records which do not parse correctly
- Extract the date, model, device ID, and latitude

and longitude. You might want to store latitude
and longitude as the first two fields to make it consistent with the other two datasets.

- Filter out locations that have a latitude and longitude of 0.
- The model field contains the device manufacturer and model name

Split this field by spaces to separate the manufacturer from the model

- Save the extracted data to comma delimited text files in S3.
- Confirm that the data in the file

Methodology and results

DATA PREPROCESSING:

In this step we first installed and imported all necessary libraries

```
pip install geopandas
```

```
Requirement already satisfied: geopandas in /usr/local/lib/python3.6/dist-packages (0.8.1)
Requirement already satisfied: pyproj>=2.2.0 in /usr/local/lib/python3.6/dist-packages (from geopandas) (3.0.0.post1)
Requirement already satisfied: shapely in /usr/local/lib/python3.6/dist-packages (from geopandas) (1.7.1)
Requirement already satisfied: pandas>=0.23.0 in /usr/local/lib/python3.6/dist-packages (from geopandas) (1.1.4)
Requirement already satisfied: fiona in /usr/local/lib/python3.6/dist-packages (from geopandas) (1.8.18)
Requirement already satisfied: certifi in /usr/local/lib/python3.6/dist-packages (from pyproj>=2.2.0->geopandas) (2020.11.8)
Requirement already satisfied: python-dateutil>=2.7.3 in /usr/local/lib/python3.6/dist-packages (from pandas>=0.23.0->geopanda
s) (2.8.1)
Requirement already satisfied: pytz>=2017.2 in /usr/local/lib/python3.6/dist-packages (from pandas>=0.23.0->geopandas) (2018.9)
Requirement already satisfied: numpy>=1.15.4 in /usr/local/lib/python3.6/dist-packages (from pandas>=0.23.0->geopandas) (1.18.
5)
Requirement already satisfied: click<8,>=4.0 in /usr/local/lib/python3.6/dist-packages (from fiona->geopandas) (7.1.2)
Requirement already satisfied: cligj>=0.5 in /usr/local/lib/python3.6/dist-packages (from fiona->geopandas) (0.7.1)
Requirement already satisfied: click-plugins>=1.0 in /usr/local/lib/python3.6/dist-packages (from fiona->geopandas) (1.1.1)
Requirement already satisfied: six>=1.7 in /usr/local/lib/python3.6/dist-packages (from fiona->geopandas) (1.15.0)
Requirement already satisfied: munch in /usr/local/lib/python3.6/dist-packages (from fiona->geopandas) (2.5.0)
Requirement already satisfied: attrs>=17 in /usr/local/lib/python3.6/dist-packages (from fiona->geopandas) (20.3.0)
```

```
pip install pyspark
```

```
Collecting pyspark
  Downloading https://files.pythonhosted.org/packages/f0/26/198fc8c0b98580f617cb03cb298c6056587b8f0447e20fa40c5b634ced77/pyspar
k-3.0.1.tar.gz (204.2MB)
     |████████████████████████████████| 204.2MB 74kB/s
Collecting py4j==0.10.9
  Downloading https://files.pythonhosted.org/packages/9e/b6/6a4fb90cd235dc8e265a6a2067f2a2c99f0d91787f06aca4bcf7c23f3f80/py4j-
0.10.9-py2.py3-none-any.whl (198kB)
     |████████████████████████████████| 204kB 43.2MB/s
Building wheels for collected packages: pyspark
  Building wheel for pyspark (setup.py) ... done
  Created wheel for pyspark: filename=pyspark-3.0.1-py2.py3-none-any.whl size=204612243 sha256=6876921d54aa5bf75bf31bd059c56a99
048008cde72e6fafd66c44c40f1b5315
  Stored in directory: /root/.cache/pip/wheels/5e/bd/07/031766ca628adec8435bb40f0bd83bb676ce65ff4007f8e73f
Successfully built pyspark
Installing collected packages: py4j, pyspark
Successfully installed py4j-0.10.9 pyspark-3.0.1
```

```
import pandas as pd
import numpy as np
import plotly.express as px
import matplotlib.pyplot as plt
import geopandas
import math
import time
from pyspark.sql.functions import col
np.random_state=42
```

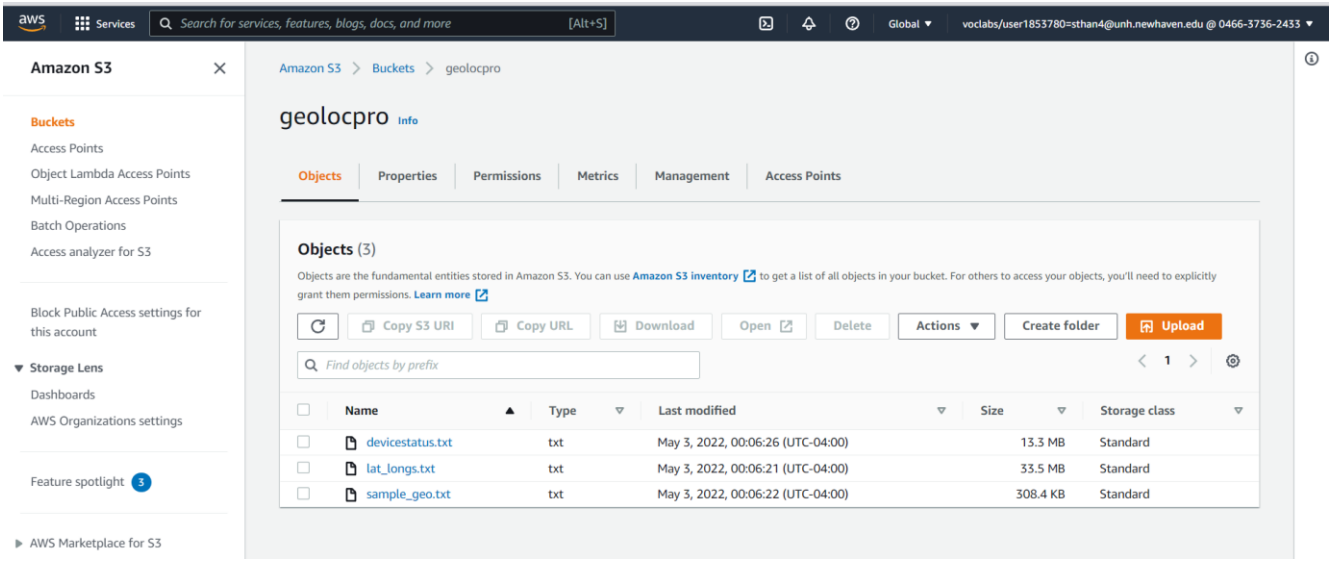
```
from pyspark import SparkContext,SQLContext
sc = SparkContext()
sqlContext = SQLContext(sc)
```

Importing datasets using pandas csv

```
data_extracted = pd.read_csv('data_extracted.csv')
sample_geo = pd.read_csv('sample_geo.txt',sep='\t')
lat_longs = pd.read_csv('lat_longs.csv')
```

```
data = sqlContext.read.csv('devicestatus.txt',inferSchema= True,header=None)
```

Uploading all datasets to s3 bucket



Important step in data preprocessing is data cleaning finding all the null values and filling them, removing unnecessary data to make data set more feasible to use.

```
for column in data.columns:
    data = data.where(col(column).isNotNull())
```

Data visualizing:

The graphical representation of information and data is known as data visualization. Data visualization tools, which use visual elements such as charts, graphs, and maps, provide an easy way to see and understand trends, outliers, and patterns in data.

```
fig, ax = plt.subplots()
fig.set_figheight(15)
fig.set_figwidth(15)
ax.set_aspect('equal')
world.plot(ax=ax,color='white', edgecolor='black')
data_extracted.plot(kind='scatter',x='longitude',y='latitude',ax=ax,color='r',marker='^')
plt.show()
```

Euclidean distance:

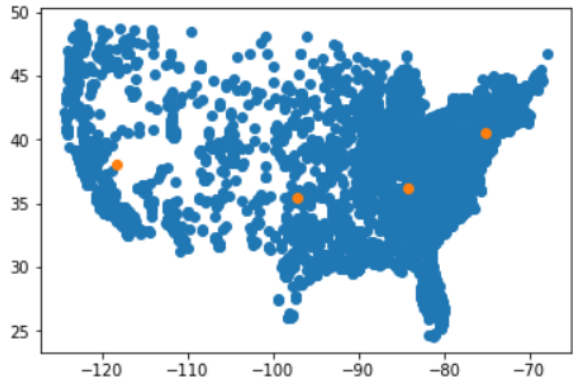
The Euclidean is often the default distance used in K-means clustering to find the k closest points of a particular sample point.

```
def EuclideanDistance(point1,point2):
    point1,point2 = np.array(point1),np.array(point2)
    dist = np.linalg.norm(point1 - point2)
    return dist
```

Euclidean distance for k =4

```
# Euclidean Distance(K = 4)
s_time = time.time()
centers = closestPoint([sample_geo.Latitude,sample_geo.Longitude],k=4,distance='Euclidean')
sampleGeoTime_4euc = time.time()-s_time
```

```
plot(centers,sample_geo.Longitude,sample_geo.Latitude)
```



Great circle distance:

It is the shortest distance measured along the surface of a sphere between two points on its surface.

```
def GreatCircleDistance(point1,point2):
    radians = math.pi/180.0

    phi1 = (90.0 - point1[0])*radians
    phi2 = (90.0 - point2[0])*radians

    theta1 = point1[1]*radians
    theta2 = point2[1]*radians

    cos = (math.sin(phi1)*math.sin(phi2)*math.cos(theta1 - theta2) +
           math.cos(phi1)*math.cos(phi2))
    dist = math.acos( cos )

    return dist
```

Comparing both distances

```
print("TIme for Euclidean Distance: {}\nTime for GreatCircle Distance: {}".format(sampleGeoTime_4euc,sampleGeoTime_4cir))
```

```
TIme for Euclidean Distance: 5.511715888977051
Time for GreatCircle Distance: 1.5558748245239258
```

Conclusion:

Since the distance between two points on a flat is greater than the distance between two points on a sphere, we can conclude that the great circle distance is less than the Euclidean distance.

Contributions/References:

https://sebastianraschka.com/faq/docs/euclidean-distance.html
https://towardsdatascience.com/spatial-distance-and-machine-learning-2cab72fc6284#:~:text=Unlike%20the%20Euclidean%20distance%2C%20the,the%20surface%20of%20a%20sphere.&text=Haversine%20Formula%20in%20KMs.,apply%20the%20Haversine%20Formula%20above.