PROJECT
IoT using GPS,Data Visualization, and Anomaly Detection
Amanraj Lnu

GROUP
Amanraj Lnu

INTRODUCTION
We are finding the stops that the GPS stopped for more than 15 minutes ,more than 45 minutes and more than 24 hours during the course of 5 months during which it recorded data when it was turned on.We are using clustering to reduce the number of stops that are near to each other and Also we are using data cleaning methods to remove redundant data.We are using stored stop data to kml file to show them on a google earth map.

BACKGROUND
GPS:
GPS satellites circle the Earth twice a day in a precise orbit. Each satellite transmits a unique signal and orbital parameters that allow GPS devices to decode and compute the precise location of the satellite.
GPS receivers use this information and trilateration to calculate a user's exact location. Essentially, the GPS receiver measures the distance to each satellite by the amount of time it takes to receive a transmitted signal.
With distance measurements from a few more satellites, the receiver can determine a user's position and display it electronically .
To calculate your 2D position (latitude and longitude) and track movement:

1.)A GPS receiver must be locked onto the signal of at least three satellites.

For 3D position (latitude, longitude and altitude):

1.) Four or more satellites in view for receiver

A GPS receiver will generally track eight or more satellites, depending on the time of day and where you are on the Earth.

Once your position has been determined, the GPS unit can calculate other information, such as:

- Speed
- Bearing
- Track
- Trip distance

- Distance to destination
- Sunrise and sunset times

References from garmin.com

KML:

KML is a file format used to display geographic data in an Earth browser such as Google Earth. KML uses a tag-based structure with nested elements and attributes and is based on the XML standard.

Placemarks

A Placemark is one of the most commonly used features in Google Earth. It marks a position on the Earth's surface, using a yellow,red or green pushpin as the icon. The simplest Placemark includes only a <Point> element, which specifies the location of the Placemark. You can specify a name and a custom icon for the Placemark.

Paths

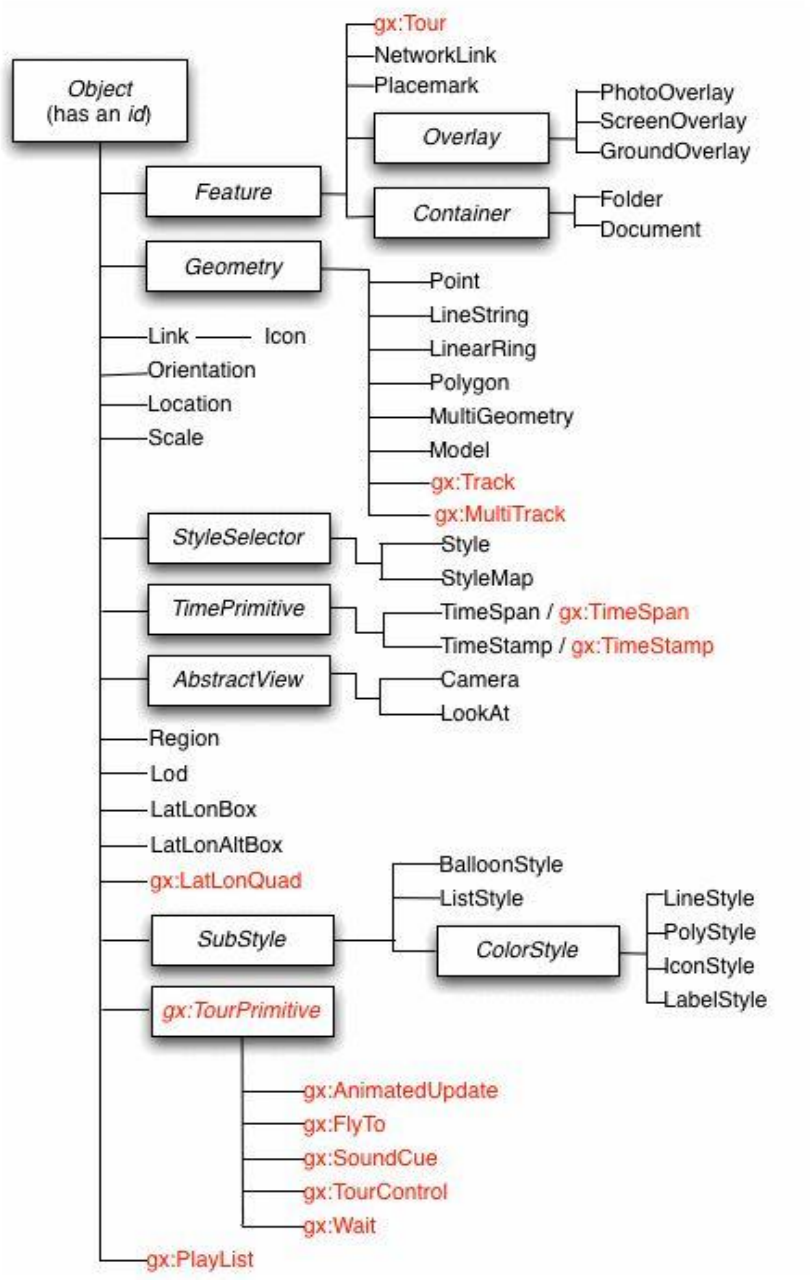
Many different types of paths can be created in Google Earth, and it is easy to be very creative with your data. In KML, a path is created by a <LineString> element

Polygons

Polygons are used to create buildings and other shapes on the map.

References from https://developers.google.com/kml/documentation/kml_tut

KML References



ETHICAL ANALYSIS

What we are doing is a gray area as we are using somebody's data to get the location of them for a specific point of time thus observing things which a person considers private.We are essentially invading their privacy and these observations could be used by malicious people to harm the person whose data we are using. We could even tell the residence of someone who doesn't want his residence known publicly by following this data which is dangerous in today's world.

DATA CLEANING

Data is cleaned by using tolerance for stops and tolerance for gps to move in a straight line.

Example :

For a stop the gps could have a difference of less than 0.00002 for both latitude and longitude for consecutive data points .

For a straight line the gps difference for any one of latitude and longitude for a consecutive data point should be less than 0.00004.

We also remove invalid or erroneous data by checking their validity and number of datapoints.

```
def remove_redundant_GPS_points():
    # Get a list of all of the keys for the dictionary of GPS info.
    key_list    = list(gps_info.keys())

    # Iterate through the list of keys.
    for key_index in range(0, len(key_list) - 1):
        # Get the coordinates of the current and next entries in the dictionary.

        this_latitude   = gps_info[key_list[key_index]]['latitude']
        next_latitude   = gps_info[key_list[key_index + 1]]['latitude']
        this_longitude  = gps_info[key_list[key_index]]['longitude']
        next_longitude  = gps_info[key_list[key_index + 1]]['longitude']

        # If the coordinates are the same (rounded down a bit, to account for GPS skew)
        # meaning we haven't moved, remove one of the values.
        if my_round(this_latitude,next_latitude) and my_round(this_longitude,next_longitude):
            gps_info.pop(key_list[key_index])
            continue


        # If we didn't remove the data for the key right before this one, and this isn't the first key.
        if key_index != 0 and key_list[key_index - 1] in gps_info:

            # Get coordinates of the preceding entry.
            last_latitude   = gps_info[key_list[key_index - 1]]['latitude']
            last_longitude  = gps_info[key_list[key_index - 1]]['longitude']

            # Get the latitude and longitude differences in both directions.
            lat_diff_from_last  = abs(this_latitude - last_latitude)
            long_diff_from_last = abs(this_longitude - last_longitude)
            lat_diff_from_next  = abs(this_latitude - next_latitude)
            long_diff_from_next = abs(this_longitude - next_longitude)

            # If our speed is constant and we're moving in a straight line, remove the unnecessary point in between.
            tolerance   = 0.00004   # This is WAY TOO BIG!!
            if abs(lat_diff_from_last - lat_diff_from_next) <= tolerance and \
                    abs(long_diff_from_last - long_diff_from_next) <= tolerance :
                gps_info.pop(key_list[key_index])

                continue

            # If the last entry has speed data (some lines may not if they were only GGA).
            if 'knots' in gps_info[key_list[key_index - 1]]:

                # Get the speed of the last entry.
                last_speed  = float(gps_info[key_list[key_index - 1]]['knots'])

                # Remove the entry if the difference in latitude or longitude is too high compared to the speed.
                # This manages errors where a GPS reading is way off (e.g. in the middle of the ocean.)
                if long_diff_from_last/10 > last_speed or lat_diff_from_next/10 > last_speed:
                    gps_info.pop(key_list[key_index])
                    continue
```

PROBLEM DEFINITION

We need to find the stops where the professor stopped for more than 15 minutes ,more than 45 minutes and more than 24 hours using Red ,Yellow and Green pin.We also the agglomerate the Stops which are clustered together meaning they are very close.

```
def findStops():
    red=[]
    yellow=[]
    green=[]


    key_list    = list(gps_info.keys())
    st=0
    stop=0
    for key_index in range(0, len(key_list) - 1):
        # if gps_filename=='2022_06/2022_06_05__212220_gps_file.txt':
        #     print("Integer"+str(key_list[key_index]))
        # Get the coordinates of the current and next entries in the dictionary.
        this_latitude   = gps_info[key_list[key_index]]['latitude']
        next_latitude   = gps_info[key_list[key_index + 1]]['latitude']
        this_longitude  = gps_info[key_list[key_index]]['longitude']
        next_longitude  = gps_info[key_list[key_index + 1]]['longitude']
        lat=this_latitude-next_latitude
        long=this_longitude-next_longitude
        # print(lat,long)

        if lat<0.00002 and long<0.00002:
            if stop==0:
                st=key_list[key_index]
                stop=1
            else:
                pass
        else:
            if stop==1:
                t1=converttime(st)
                t2=converttime(key_list[key_index])
                dt=t2-t1
                time=dt.total_seconds()
                # print(time)
                if abs(time)>86400:
                    red.append([gps_info[st]['latitude'],gps_info[st]['longitude']])
                    gRed[st]=gps_info[st]
                elif abs(time)>2700:
                    yellow.append([gps_info[st]['latitude'],gps_info[st]['longitude']])
                    gYellow[st]=gps_info[st]
                elif abs(time)>900:
                    green.append([gps_info[st]['latitude'],gps_info[st]['longitude']])
                    gGreen[st]=gps_info[st]
                else:
                    continue
                stop=0
            else:
                pass

    return [red,yellow,green]
```

IMPLEMENTATION

Step 1
We first get all points stored in a dictionary in python using different timestamps as keys and I added 1 before the timestamp which exceeded more than 24 hours of data recordings.

Step 2
I then removed redundant points like multiple points having stops and error made points .I also Removed points where the gps was moving in straight line

Step 3
Then we find the stops using tolerance of range of points that can attributed as same stop
Here we have taken 0.00002 as tolerance of stops for both longitude and latitude.

Step 4
We then write these stop points in a kml file ordering them in red ,yellow and green pins according to the duration of the stop.

Step 5
We also use agglomerative clustering to cluster stops which are close to each other to reduce the number of stops irrespective of duration of stops.

KML FILE CREATION

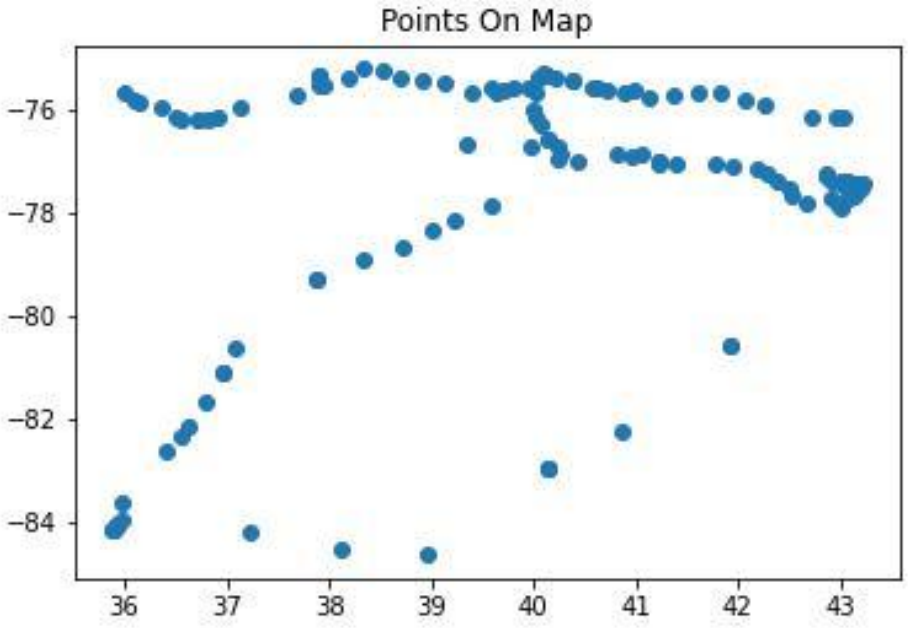
We then use the write_out_KML_file(stops,pin,color) function to write the stops as placemark in our kml file with Red ,Yellow and Green pin for different duration stops in data points.

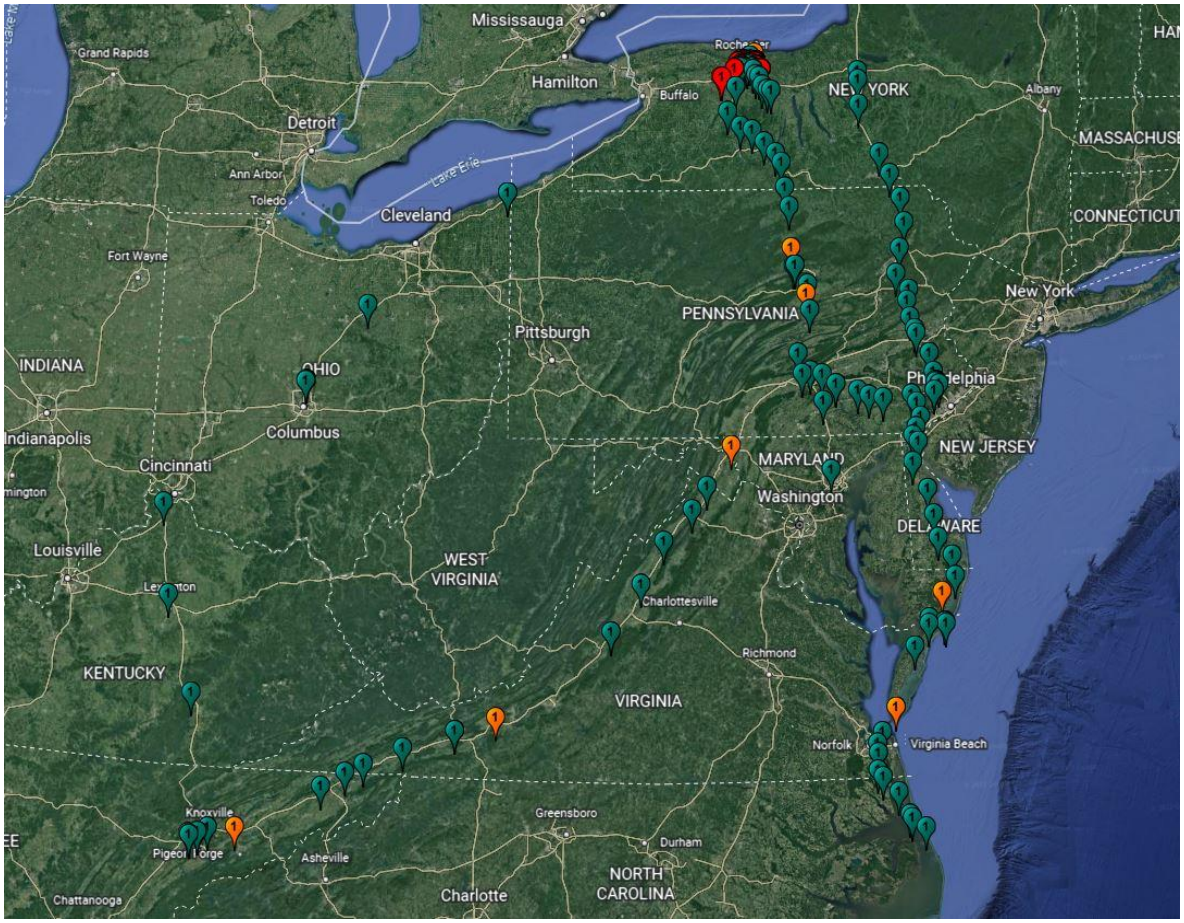
Example of the Written file:

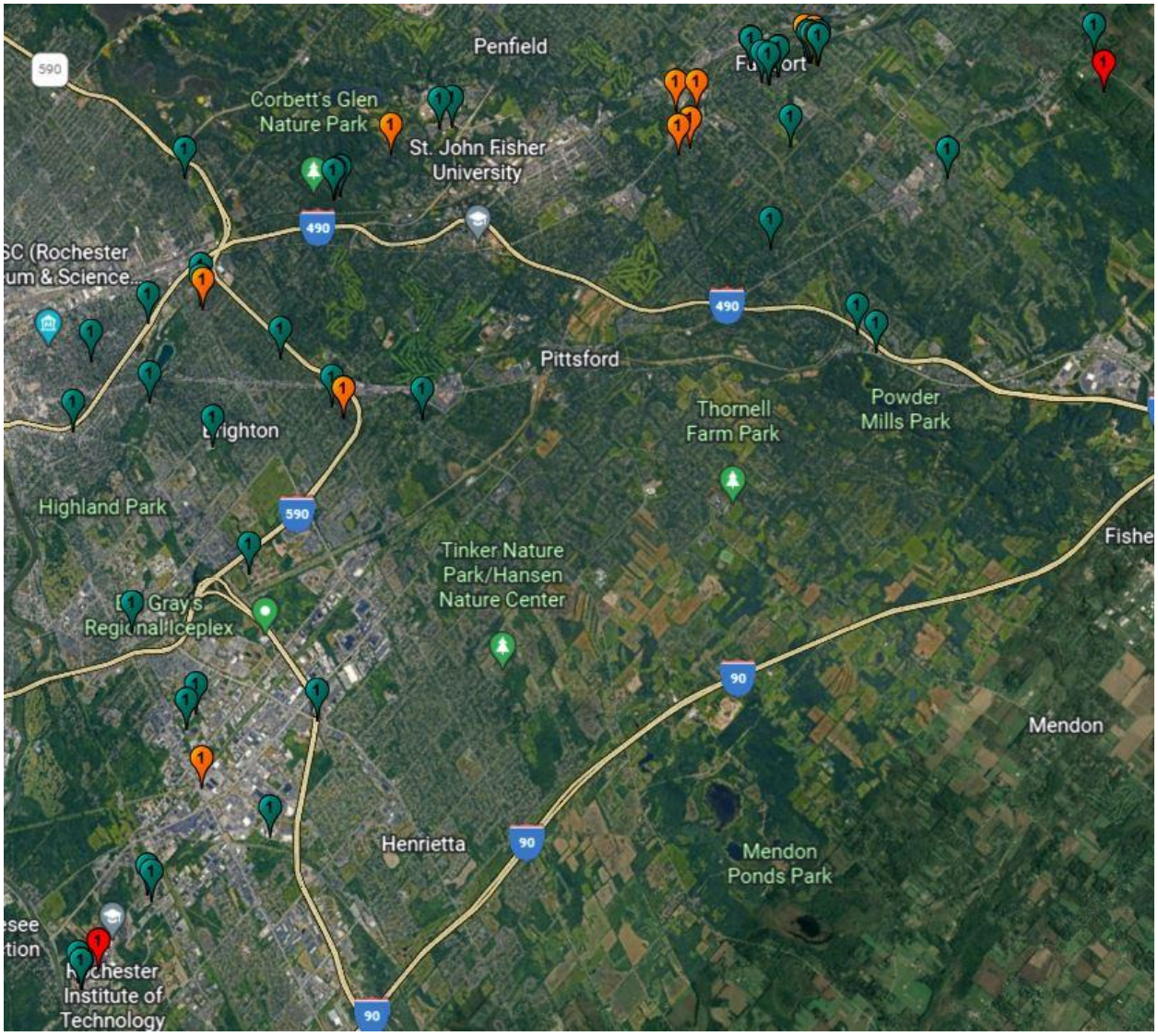
```
<?xml version="1.0" encoding="UTF-8"?>
<kml xmlns="http://www.opengis.net/kml/2.2">
<Document>
<Placemark>
	<description>Red PIN for A Stop</description>
	<Style>
		<IconStyle>
			<color>ff0000ff</color>
			<Icon>
				<href>http://maps.google.com/mapfiles/kml/paddle/1.png</href>
			</Icon>
		</IconStyle>
	</Style>
	<Point>
		<coordinates>-77.43225,43.095585,0</coordinates>
	</Point>
</Placemark>
<Placemark>
	<description>Yellow PIN for A Stop</description>
	<Style>
		<IconStyle>
			<color>ff00ffff</color>
			<Icon>
				<href>http://maps.google.com/mapfiles/kml/paddle/1.png</href>
			</Icon>
		</IconStyle>
	</Style>
	<Point>
		<coordinates>-77.437787,43.138652,0</coordinates>
	</Point>
</Placemark>

<Placemark>
	<description>Green PIN for A Stop</description>
	<Style>
```

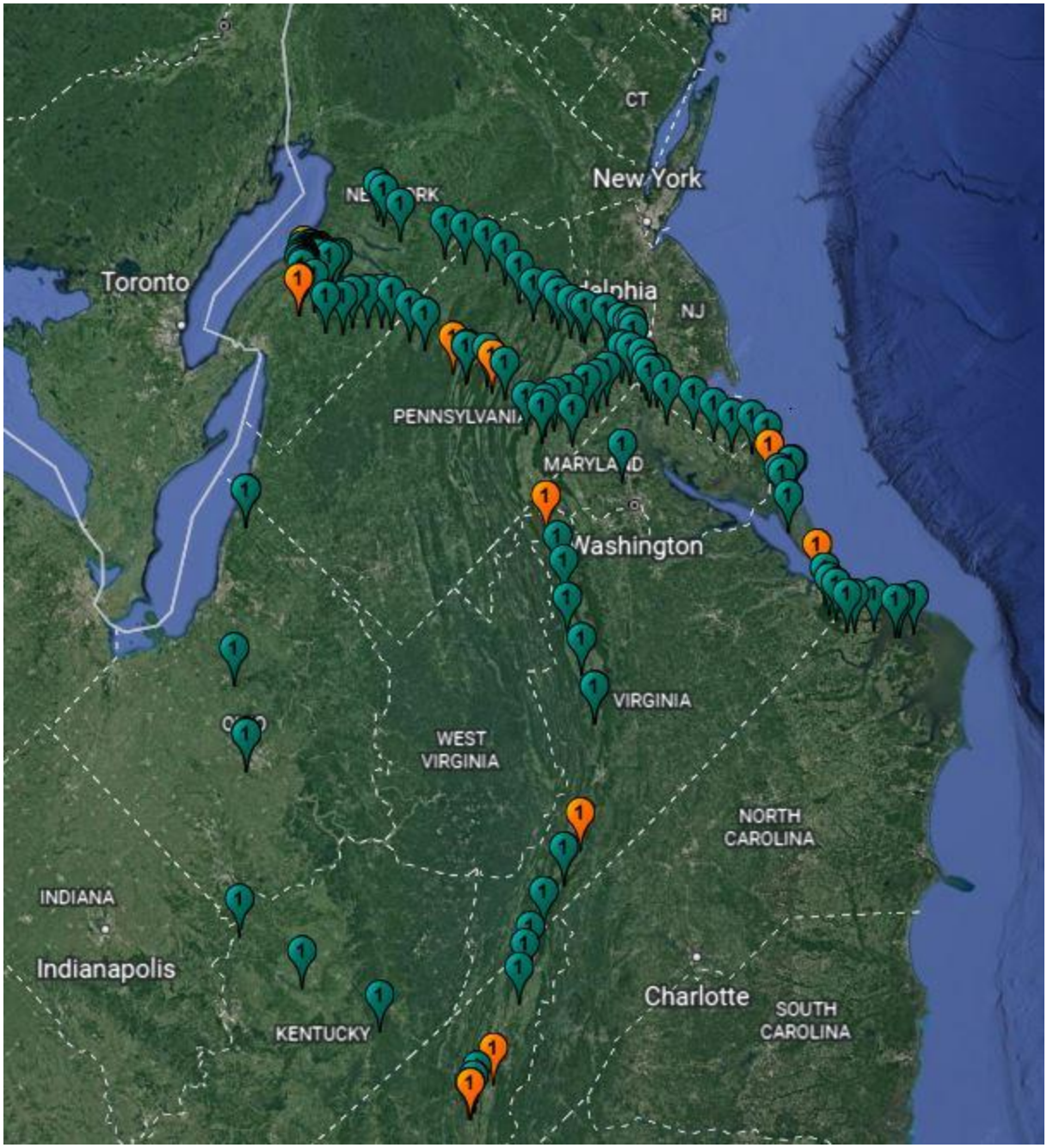
```
                <IconStyle>
                        <color>ffffff00</color>
                        <Icon>
                                <href>http://maps.google.com/mapfiles/kml/paddle/1.png</href>
                        </Icon>
                </IconStyle>
        </Style>
        <Point>
                <coordinates>-77.437622,43.138477,162.6</coordinates>
        </Point>
</Placemark>
  </Document>
</kml>
```

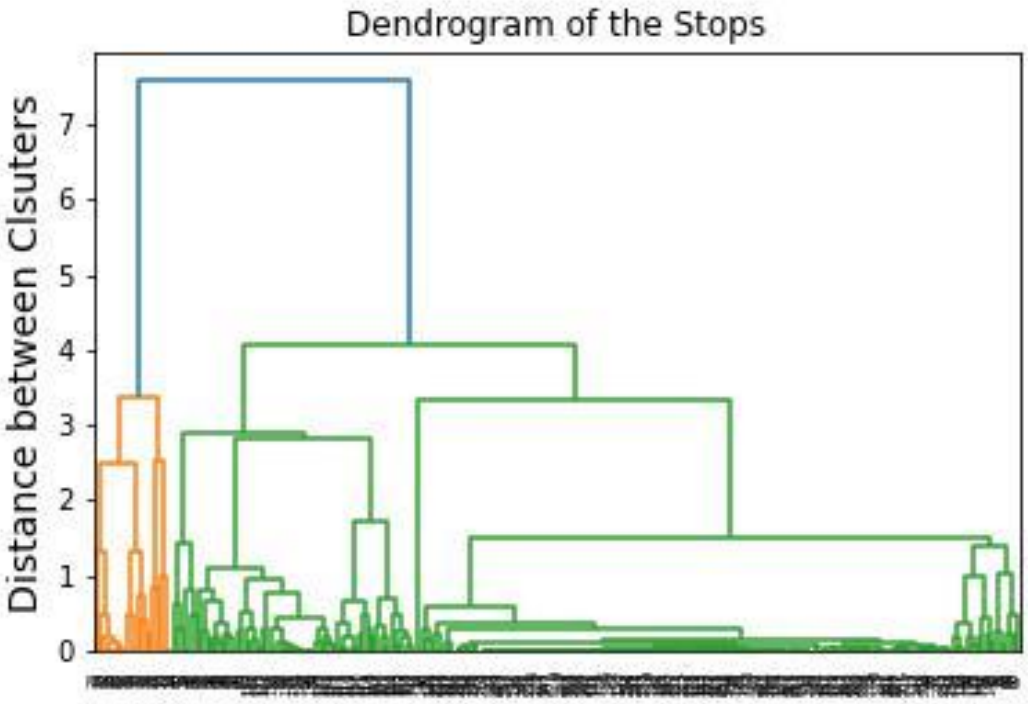
SCREEN CAPTURE

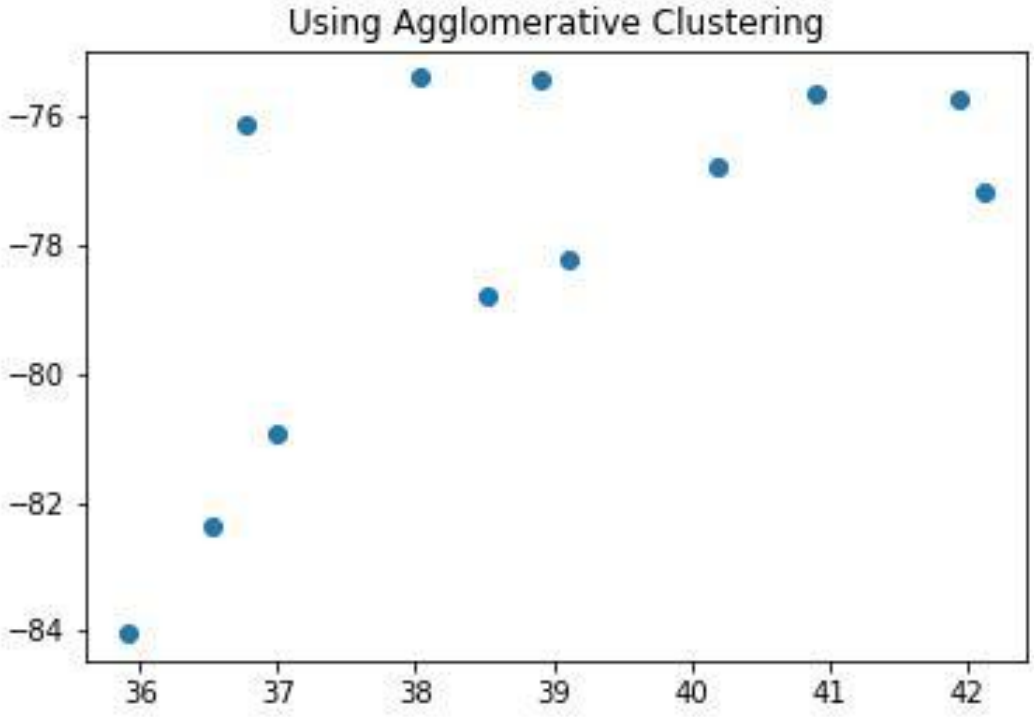





Penfield
Corbett's Glen Nature Park
St. John Fisher University
490
Pittsford
Thornell Farm Park
Powder Mills Park
Highland Park
590
Tinker Nature Park/Hansen Nature Center
Gray's Regional Iceplex
90
Mendon
Henrietta
Mendon Ponds Park
Institute of Technology


RI
CT
New York
Toronto
NJ
PENNSYLVANIA
MARYLAND
Washington
VIRGINIA
WEST
VIRGINIA
NORTH
CAROLINA
INDIANA
Indianapolis
KENTUCKY
Charlotte
SOUTH
CAROLINA


Dendrogram of the Stops
Distance between Clsuters
0
1
2
3
4
5
6
7


Using Agglomerative Clustering
−76
−78
−80
−82
−84
36
37
38
39
40
41
42

CONCLUSION

We implemented a real world application of clustering that was taught in our course.It is not very ethical as we are finding out about what a person is doing in his private life thus invading his privacy which in some countries is also illegal.We have to carefully do our implementation and observations on our data as not give details about a person's life when seeing our observation. This project showed us how we can gain knowledge about things which we didn't need like if a person stops at location for more than 24 hours it's high chance that it is residence of person of data we are analyzing We also get to apply agglomerative clustering in our this project to reduce repeating of stops for different dates.

This project helped me read and manipulate map data.I also performed data analysis on the data provided.I was particularly interested in the kml file and its projection on google map.
I also saw how Google and other companies use location data for commercial purposes for things like targeted ads,etc.