Name: Madhura Bartakke

Problem statement: The dataset is a CSV of (timestamp (local time), lat, long) 3-tuples representing the location data of a single user over the course of a few days. Our goal is to gain insights into the travel behavior of the user.

In [1]:

```
# Import libraries

import pandas as pd
pd.set_option('precision', 10)

import geopy.distance

import warnings
warnings.filterwarnings("ignore")
```

In [2]:

```
# Read the data
df = pd.read_csv('ds_challenge_latlongtimestamp_data.csv')

# Create a copy of the data
mile_user_df = df.copy()
```

In [3]:

```
# structure of data
print("The dataset has {} records with {} fields.".format(*df.shape))
```

```
The dataset has 841 records with 3 fields.
```

In [4]:

```
# Columns in the dataset
df.columns
```

Out[4]:

```
Index(['timestamp', 'latitude', 'longitude'], dtype='object')
```

In [5]:

```
# Index, Datatype and Memory information
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 841 entries, 0 to 840
Data columns (total 3 columns):
 #   Column     Non-Null Count  Dtype  
---  ------     --------------  -----  
 0   timestamp  841 non-null    object 
 1   latitude   841 non-null    float64
 2   longitude  841 non-null    float64
dtypes: float64(2), object(1)
memory usage: 19.8+ KB
```

In [6]:

```
# Convert timestamp object to datetime object
df['timestamp'] = pd.to_datetime(df['timestamp'])
```

In [7]:

```
# Eyeballing the data
with pd.option_context('display.max_rows', 10, 'display.max_columns', df.shape[1]):
    display(df)
```

	timestamp	latitude	longitude
0	2018-03-22 09:20:04.360	37.46669407	-122.19646363
1	2018-03-22 09:20:18.072	37.46540552	-122.19783691
2	2018-03-22 09:21:02.410	37.46214200	-122.20028691
3	2018-03-22 09:21:49.000	37.46216399	-122.19986415
4	2018-03-22 09:22:28.000	37.46140431	-122.19890159
...	...	...	...
836	2018-03-24 22:48:31.616	37.87229943	-122.27442745
837	2018-03-24 22:52:25.916	37.87234094	-122.27438733
838	2018-03-24 23:32:54.991	37.87237579	-122.27432201
839	2018-03-24 23:56:31.033	37.87237579	-122.27432201
840	2018-03-24 23:57:00.033	37.87251247	-122.27425053

841 rows × 3 columns

In [8]:

```
# Analyzing summary statistics of variables
df.describe()
```

Out[8]:

	latitude	longitude
count	841.0000000000	841.0000000000
mean	37.4751552638	-122.1276732173
std	0.1532305279	0.1634997934
min	37.3291637200	-122.4317453200
25%	37.3703441400	-122.1961403300
50%	37.4532621200	-122.1762640700
75%	37.4734690800	-121.9830372300
max	37.8725124700	-121.8880907200

In [9]:

```
#Column wise count of null values
df.isnull().sum()
```

Out[9]:

```
timestamp    0
latitude     0
longitude    0
dtype: int64
```

Feature Engineering

In [10]:

```
# Calculates distance between two cumulative points using geodesic (in miles)
def distance_miles(row1, row2):

    coords_1 = (row1['latitude'], row1['longitude'])
    coords_2 = (row2['latitude'], row2['longitude'])
    return geopy.distance.geodesic(coords_1, coords_2).mi

df['distance_from_previous(in mi)'] = 0.0
for i in range(0, len(df)-1):
    df['distance_from_previous(in mi)'].iloc[i+1] = distance_miles(df.iloc[i], df.iloc[i+1])
```

In [11]:

```
# Calculates distance between two cumulative points using geodesic (in meters)

def distance_meters(row1, row2):

    coords_1 = (row1['latitude'], row1['longitude'])
    coords_2 = (row2['latitude'], row2['longitude'])
    return geopy.distance.geodesic(coords_1, coords_2).m

df['distance_from_previous(in m)'] = 0.0
for i in range(0, len(df)-1):
    df['distance_from_previous(in m)'].iloc[i+1] = distance_meters(df.iloc[i], df.iloc[i+1])
```

In [12]:

```
# Checking the values
df[['distance_from_previous(in mi)', 'distance_from_previous(in m)']]
```

Out[12]:

	distance_from_previous(in mi)	distance_from_previous(in m)
0	0.0000000000	0.0000000000
1	0.1165987663	187.6475250062
2	0.2622838388	422.1049221922
3	0.0232897034	37.4811444612
4	0.0744629217	119.8364562449
...	...	...
836	0.0028934768	4.6565995553
837	0.0036065376	5.8041596143
838	0.0043046370	6.9276417579
839	0.0000000000	0.0000000000
840	0.0102045394	16.4226143094

841 rows × 2 columns

In [13]:

```
# Calculates time difference from previous data point (in seconds)
def time_sec(row1, row2):

    return pd.Timedelta(row2['timestamp'] - row1['timestamp']).seconds

df['time_from_previous(in sec)'] = 0
for i in range(0, len(df)-1):
    df['time_from_previous(in sec)'].iloc[i+1] = time_sec(df.iloc[i], df.iloc[i+1])
```

In [14]:

```
# Calculates time difference from previous data point (in minutes)
df['time_from_previous(in min)'] = df['time_from_previous(in sec)']/60.0
```

In [15]:

```
df[['time_from_previous(in sec)', 'time_from_previous(in min)']]
```

Out[15]:

	time_from_previous(in sec)	time_from_previous(in min)
0	0	0.0000000000
1	13	0.2166666667
2	44	0.7333333333
3	46	0.7666666667
4	39	0.6500000000
...	...	...
836	1243	20.7166666667
837	234	3.9000000000
838	2429	40.4833333333
839	1416	23.6000000000
840	29	0.4833333333

841 rows × 2 columns

In [16]:

```
# Calculates ratio of distance (in meters) and time (in minutes)
df['ratio(distance/time)'] = df['distance_from_previous(in m)']/df['time_from_previous(in min)']
```

Q1. How many unique “locations” does the user visit?

- To visit any location , the tendency of a user is to spend maximum time covering minimum distance. The same principle is applied here where the difference between two timesamp is maximum while the distance covered is minimum.
- The ratio of distance to time should be as small as possible.
- But these involve few cases, for example distance is 0.1 meters and time taken is 2 mins. Therefore, the decision to keep minimum time to visit a location is kept as minimum of 4.5 mins for this data.
- Therefore filtering criteria for location visited is time from last location should be more than 4.5 mins and ratio of distance from last location to time from last location should be less than 3.5.

(decided threshold based on experiments)

In [17]:

```
# Filtering data to get unique locations
unique_df = df[(df['ratio(distance/time)'] < 3.5) & (df['time_from_previous(in min)'] > 4.5) == True]
unique_df
```

Out[17]:

	timestamp	latitude	longitude	distance_from_previous(in mi)	distance_from_previous(in m)	time_from_previous(in sec)	time_from_previous(in min)
136	2018-03-22 11:06:41.244	37.33709935	-121.88962523	0.0154718142	24.8994713173	466	7.7666666667
138	2018-03-22 11:17:10.110	37.33697829	-121.88938045	0.0011469568	1.8458480848	428	7.1333333333
139	2018-03-22 11:29:00.690	37.33698209	-121.88937764	0.0003043274	0.4897674566	710	11.8333333333
140	2018-03-22 11:35:34.195	37.33697539	-121.88938709	0.0006958872	1.1199219412	393	6.5500000000
141	2018-03-22 18:26:40.001	37.33697539	-121.88938709	0.0000000000	0.0000000000	24665	411.0833333333

159	2018-03-22 18:52:56.005	37.33057792	-121.90296792	0.0095959959	15.4432584212	293	4.8833333333
290	2018-03-22 21:13:26.308	37.51224213	-122.26618395	0.0028695438	4.6180830810	975	16.2500000000
291	2018-03-22 21:19:55.307	37.51228052	-122.26616350	0.0028760364	4.6285319045	388	6.4666666667
292	2018-03-22 21:35:26.311	37.51217826	-122.26634195	0.0120764336	19.4351360146	931	15.5166666667
294	2018-03-22 21:51:11.133	37.51229557	-122.26620113	0.0023236341	3.7395265841	728	12.1333333333
297	2018-03-22 21:59:27.012	37.51228120	-122.26611677	0.0031622032	5.0890726975	287	4.7833333333
341	2018-03-22 22:51:07.008	37.47337106	-122.19473200	0.0067823013	10.9150559514	1070	17.8333333333
347	2018-03-23 09:06:31.000	37.47336789	-122.19462405	0.0000000000	0.0000000000	36458	607.6333333333
453	2018-03-23 10:08:02.524	37.33697850	-121.88938360	0.0007160902	1.1524354344	326	5.4333333333
470	2018-03-23 18:52:03.586	37.33550905	-121.89190423	0.1711701131	275.4715945028	30771	512.8500000000
639	2018-03-23 21:57:02.309	37.47333786	-122.19472756	0.0148109385	23.8358949817	6021	100.3500000000
666	2018-03-24 18:11:07.956	37.47071621	-122.19569068	0.1992462088	320.6556906780	72169	1202.8166666667
744	2018-03-24 19:45:37.219	37.78626147	-122.42997822	0.0007537406	1.2130279689	544	9.0666666667
748	2018-03-24 19:57:36.856	37.78647509	-122.43003300	0.0159634191	25.6906327235	587	9.7833333333
749	2018-03-24 21:28:46.011	37.78644637	-122.43147366	0.0788785851	126.9427776759	5469	91.1500000000
836	2018-03-24 22:48:31.616	37.87229943	-122.27442745	0.0028934768	4.6565995553	1243	20.7166666667
838	2018-03-24 23:32:54.991	37.87237579	-122.27432201	0.0043046370	6.9276417579	2429	40.4833333333
839	2018-03-24 23:56:31.033	37.87237579	-122.27432201	0.0000000000	0.0000000000	1416	23.6000000000

In [18]:

```
unique_df.shape
```

Out[18]:

```
(23, 8)
```

In [19]:

```
unique_df = unique_df.reset_index()
```

In [20]:

```
# Calculates distance between two destinations
def distance_destinations(df):
    for i in range(0, len(df)):
        row1 = df.iloc[i]
        coords_1 = (row1['latitude'], row1['longitude'])
        column_name = 'distance_from_'+str(i)
        df[column_name] = 0
        for j in range(0, len(df)):
            row2 = df.iloc[j]
            coords_2 = (row2['latitude'], row2['longitude'])
            df[column_name].iloc[j] = geopy.distance.geodesic(coords_1, coords_2).m
    return df

unique_df = distance_destinations(unique_df)
```

In [21]:

```
unique_df
```

Out[21]:

	index	timestamp	latitude	longitude	distance_from_previous(in mi)	distance_from_previous(in m)	time_from_previous(in sec)	time_from_prev
0	136	2018-03-22 11:06:41.244	37.33709935	-121.88962523	0.0154718142	24.8994713173	466	7.7666

1	138	2018-03-22 11:17:10.110	37.33697829	-121.88938045	0.0011469568	1.8458480848	428	7.1333
2	139	2018-03-22 11:29:00.690	37.33698209	-121.88937764	0.0003043274	0.4897674566	710	11.8333
3	140	2018-03-22 11:35:34.195	37.33697539	-121.88938709	0.0006958872	1.1199219412	393	6.5500
4	141	2018-03-22 18:26:40.001	37.33697539	-121.88938709	0.0000000000	0.0000000000	24665	411.0833
5	159	2018-03-22 18:52:56.005	37.33057792	-121.90296792	0.0095959959	15.4432584212	293	4.8833
6	290	2018-03-22 21:13:26.308	37.51224213	-122.26618395	0.0028695438	4.6180830810	975	16.2500
7	291	2018-03-22 21:19:55.307	37.51228052	-122.26616350	0.0028760364	4.6285319045	388	6.4666
8	292	2018-03-22 21:35:26.311	37.51217826	-122.26634195	0.0120764336	19.4351360146	931	15.5166
9	294	2018-03-22 21:51:11.133	37.51229557	-122.26620113	0.0023236341	3.7395265841	728	12.1333
10	297	2018-03-22 21:59:27.012	37.51228120	-122.26611677	0.0031622032	5.0890726975	287	4.7833
11	341	2018-03-22 22:51:07.008	37.47337106	-122.19473200	0.0067823013	10.9150559514	1070	17.8333
12	347	2018-03-23 09:06:31.000	37.47336789	-122.19462405	0.0000000000	0.0000000000	36458	607.6333
13	453	2018-03-23 10:08:02.524	37.33697850	-121.88938360	0.0007160902	1.1524354344	326	5.4333
14	470	2018-03-23 18:52:03.586	37.33550905	-121.89190423	0.1711701131	275.4715945028	30771	512.8500
15	639	2018-03-23 21:57:02.309	37.47333786	-122.19472756	0.0148109385	23.8358949817	6021	100.3500
16	666	2018-03-24 18:11:07.956	37.47071621	-122.19569068	0.1992462088	320.6556906780	72169	1202.8166
17	744	2018-03-24 19:45:37.219	37.78626147	-122.42997822	0.0007537406	1.2130279689	544	9.0666
18	748	2018-03-24 19:57:36.856	37.78647509	-122.43003300	0.0159634191	25.6906327235	587	9.7833
19	749	2018-03-24 21:28:46.011	37.78644637	-122.43147366	0.0788785851	126.9427776759	5469	91.1500
20	836	2018-03-24 22:48:31.616	37.87229943	-122.27442745	0.0028934768	4.6565995553	1243	20.7166
21	838	2018-03-24 23:32:54.991	37.87237579	-122.27432201	0.0043046370	6.9276417579	2429	40.4833
22	839	2018-03-24 23:56:31.033	37.87237579	-122.27432201	0.0000000000	0.0000000000	1416	23.6000

23 rows × 32 columns

In [22]:

```
def similar_destinations(df):
    df['total_similar_destinations'] = None
    for i in range(0, 23):
        column_name = 'distance_from_'+str(i)
        des_list = []
        for j in range(0, 23):
            if df[column_name].iloc[j] < 10.0:
                des_list.append(j)
        df['total_similar_destinations'].iloc[i] = des_list
    return df

unique_df = similar_destinations(unique_df)
```

- The criteria for location to be similar is that the distnace between two locations should be less than 10 meters.
- For this I calculated the distance between to all the location from each location.
- We have 23 unique locations (from 0 to 22), therefore the above step created 23x23 matrix where diagonal is 0 (distance from self)
- Filtered out similar location (within 10 m) and added it to the list. Now the list contains all the locations which are within 10 meters of distance.

- Top 3 visited location is the list with most location indices.

In [23]:

```
unique_df
```

Out[23]:

	index	timestamp	latitude	longitude	distance_from_previous(in mi)	distance_from_previous(in m)	time_from_previous(in sec)	time_from_prev
0	136	2018-03-22 11:06:41.244	37.33709935	-121.88962523	0.0154718142	24.8994713173	466	7.7666
1	138	2018-03-22 11:17:10.110	37.33697829	-121.88938045	0.0011469568	1.8458480848	428	7.1333
2	139	2018-03-22 11:29:00.690	37.33698209	-121.88937764	0.0003043274	0.4897674566	710	11.8333
3	140	2018-03-22 11:35:34.195	37.33697539	-121.88938709	0.0006958872	1.1199219412	393	6.5500
4	141	2018-03-22 18:26:40.001	37.33697539	-121.88938709	0.0000000000	0.0000000000	24665	411.0833
5	159	2018-03-22 18:52:56.005	37.33057792	-121.90296792	0.0095959959	15.4432584212	293	4.8833
6	290	2018-03-22 21:13:26.308	37.51224213	-122.26618395	0.0028695438	4.6180830810	975	16.2500
7	291	2018-03-22 21:19:55.307	37.51228052	-122.26616350	0.0028760364	4.6285319045	388	6.4666
8	292	2018-03-22 21:35:26.311	37.51217826	-122.26634195	0.0120764336	19.4351360146	931	15.5166
9	294	2018-03-22 21:51:11.133	37.51229557	-122.26620113	0.0023236341	3.7395265841	728	12.1333
10	297	2018-03-22 21:59:27.012	37.51228120	-122.26611677	0.0031622032	5.0890726975	287	4.7833
11	341	2018-03-22 22:51:07.008	37.47337106	-122.19473200	0.0067823013	10.9150559514	1070	17.8333
12	347	2018-03-23 09:06:31.000	37.47336789	-122.19462405	0.0000000000	0.0000000000	36458	607.6333
13	453	2018-03-23 10:08:02.524	37.33697850	-121.88938360	0.0007160902	1.1524354344	326	5.4333
14	470	2018-03-23 18:52:03.586	37.33550905	-121.89190423	0.1711701131	275.4715945028	30771	512.8500
15	639	2018-03-23 21:57:02.309	37.47333786	-122.19472756	0.0148109385	23.8358949817	6021	100.3500
16	666	2018-03-24 18:11:07.956	37.47071621	-122.19569068	0.1992462088	320.6556906780	72169	1202.8166
17	744	2018-03-24 19:45:37.219	37.78626147	-122.42997822	0.0007537406	1.2130279689	544	9.0666
18	748	2018-03-24 19:57:36.856	37.78647509	-122.43003300	0.0159634191	25.6906327235	587	9.7833
19	749	2018-03-24 21:28:46.011	37.78644637	-122.43147366	0.0788785851	126.9427776759	5469	91.1500
20	836	2018-03-24 22:48:31.616	37.87229943	-122.27442745	0.0028934768	4.6565995553	1243	20.7166
21	838	2018-03-24 23:32:54.991	37.87237579	-122.27432201	0.0043046370	6.9276417579	2429	40.4833
22	839	2018-03-24 23:56:31.033	37.87237579	-122.27432201	0.0000000000	0.0000000000	1416	23.6000

23 rows × 33 columns

Ans1: From the column 'total similar destinations' there are 13 unique locations the user visits.

What are the top 3 locations visited by this user?

Ans2: The top 3 locations visited by the user are as follows:

In [24]:

```
print("First most visited location is\n", unique_df[['latitude', 'longitude']].iloc[1])
print("Second most visited location is\n", unique_df[['latitude', 'longitude']].iloc[6])
print("Third most visited location is\n", unique_df[['latitude', 'longitude']].iloc[11])
```

```
First most visited location is
 latitude      37.33697829
longitude   -121.88938045
Name: 1, dtype: float64
Second most visited location is
 latitude      37.51224213
longitude   -122.26618395
Name: 6, dtype: float64
Third most visited location is
 latitude      37.47337106
longitude   -122.19473200
Name: 11, dtype: float64
```

Bonus: what other insights can you obtain about the travel behavior of this user?

There are few data points where the users traveled very less distance and the time taken is more (>5 hours). In this case we can analyse the user behaviour

User behaviour 1

In [25]:

```
df.iloc[141]
```

Out[25]:

```
timestamp                      2018-03-22 18:26:40.001000
latitude                                      37.33697539
longitude                                    -121.88938709
distance_from_previous(in mi)                         0.0
distance_from_previous(in m)                          0.0
time_from_previous(in sec)                          24665
time_from_previous(in min)                   411.0833333333
ratio(distance/time)                                  0.0
Name: 141, dtype: object
```

The above datapoint shows that the user has not moved since approximately 7 hours. The time is 6.26 pm. This shows tha the user has been working for past 7 hours with very less displacement. Lets see what the person does after working in office (assumption).

In [26]:

```
df.iloc[141:160]
```

Out[26]:

	timestamp	latitude	longitude	distance_from_previous(in mi)	distance_from_previous(in m)	time_from_previous(in sec)	time_from_previous(in min)
141	2018-03-22 18:26:40.001	37.33697539	-121.88938709	0.0000000000	0.0000000000	24665	411.0833333333
142	2018-03-22 18:26:40.001	37.33697539	-121.88938709	0.0000000000	0.0000000000	0	0.0000000000
143	2018-03-22 18:28:40.001	37.33537740	-121.89136578	0.1549700417	249.4001068182	120	2.0000000000
144	2018-03-22 18:29:25.000	37.33550342	-121.89195025	0.0333367749	53.6503387387	44	0.7333333333
145	2018-03-22 18:30:10.000	37.33493349	-121.89340585	0.0892711757	143.6680309951	45	0.7500000000
146	2018-03-22 18:31:01.000	37.33463455	-121.89444940	0.0610500581	98.2505446506	51	0.8500000000
147	2018-03-22 18:31:46.000	37.33383726	-121.89417732	0.0569876315	91.7127027631	45	0.7500000000
148	2018-03-22 18:32:24.999	37.33432828	-121.89453456	0.0391612952	63.0239953893	38	0.6333333333
149	2018-03-22 18:33:08.000	37.33392666	-121.89504040	0.0392808996	63.2164801399	43	0.7166666667
150	2018-03-22 18:33:49.999	37.33420929	-121.89540409	0.0279459334	44.9746202692	41	0.6833333333
151	2018-03-22 18:34:39.000	37.33356359	-121.89536344	0.0445851233	71.7528006661	49	0.8166666667
152	2018-03-22 18:35:21.000	37.33374942	-121.89586124	0.0302598545	48.6985152752	42	0.7000000000
153	2018-03-22 18:40:20.002	37.33217274	-121.89965607	0.2355657272	379.1062896795	299	4.9833333333
154	2018-03-22 18:41:04.000	37.33206348	-121.90088645	0.0681719971	109.7121945409	43	0.7166666667

155	2018-03-22 18:42:49.001	37.33195636	-121.90180980	0.0513807459	82.6892950909	105	1.7500000000
156	2018-03-22 18:44:01.001	37.33126192	-121.90202848	0.0493809515	79.4709380746	72	1.2000000000
157	2018-03-22 18:44:44.000	37.33073977	-121.90204340	0.0360179701	57.9653040150	42	0.7000000000
158	2018-03-22 18:48:03.003	37.33043882	-121.90296331	0.0547449862	88.1035151437	199	3.3166666667
159	2018-03-22 18:52:56.005	37.33057792	-121.90296792	0.0095959959	15.4432584212	293	4.8833333333

An average person walks 80-90 meters in a minute, looking at the data after the user leaves office, and walks towards home. His walking data is captured from data indices 142-158. Finally he reaches home at index 159 where he dispplaces 15 meters in 4 minutes.

User behaviour 2

In [27]:

```
df.iloc[347]
```

Out[27]:

```
timestamp                          2018-03-23 09:06:31
latitude                                   37.47336789
longitude                                -122.19462405
distance_from_previous(in mi)                      0.0
distance_from_previous(in m)                       0.0
time_from_previous(in sec)                       36458
time_from_previous(in min)                607.6333333333
ratio(distance/time)                               0.0
Name: 347, dtype: object
```

The above datapoint tells us that the user has not moved for the past 10 hours. The time of the day of this datapoint is 9 am. This tells us that the user was asleep for the past 10 hours.

Questions

1. There are few instances where the user has not movd (0 meters) and time recorded is 0 minutes.
2. There are few instances where the displacement of the user is very less but the time recorded from previous point is high (in hours)

- Q1. How is the sensor data captured?
- Q2. If the user leaves or forgets it's phone somewhere, does the sensor data capture that?