Heath_AI_Near_Me_Map_Simulation_K_Foo_8th_Oct 2020

In [ ]:

```python
import math
import numpy as np
import pandas as pd
## for plotting
import matplotlib.pyplot as plt
import seaborn as sns
## for geospatial
import folium
import geopy
## for machine learning
from sklearn import preprocessing, cluster
import scipy
## for deep learning
import minisom
```

In [44]:

```python
dtf = pd.read_csv('stores_10_8_2020.csv')
```

In [45]:

```python
filter = "Las Vegas"
dtf = dtf[dtf["City"]==filter][["City","Street Address","Longitude","Latitud
dtf = dtf.reset_index().rename(columns={"index":"id"})
dtf.head()
```

Out[45]:

```
   id       City                      Street Address  Longitude  Latitude
0   0  Las Vegas                    4507 Flamingo Rd    -115.20     36.12
1   1  Las Vegas   475 E Windmill Lane, Fashion Show    -115.15     36.04
2   2  Las Vegas  3200 LAS VEGAS BLVD. S., STE 1795    -115.17     36.13
3   3  Las Vegas                 8350 W Cheyenne Ave    -115.28     36.22
4   4  Las Vegas                3730 LAS VEGAS BLVD S    -115.18     36.11
```

In [46]:

```
#Tells you the data above is how far Relative to 2255 E Centennial Parkway -

Locations_less_than_5_miles=[]
for i in dtf['Longitude']:
    for x in dtf['Latitude']:
        radiusEarth= 3961 #miles. If you want km, put in 6373 km in instead
        lat1 = math.radians(36.14)  #3301 W. Sahara Ave.
        lon1 = math.radians(-115.19)  #3301 W. Sahara Ave.
        lat2 = math.radians(x)  #input latitude
        lon2 = math.radians(i)  #input longitude
        dlon = lon2 - lon1
        dlat = lat2 - lat1
        a = math.sin(dlat / 2)**2 + math.cos(lat1) * math.cos(lat2) * math.s
        c = 2 * math.atan2(math.sqrt(a), math.sqrt(1 - a))
        distance = round(radiusEarth * c,2) #in miles
        print(distance)
```

```
1.49
6.94
0.89
5.56
2.15
2.15
1.49
5.56
0.89
9.69
2.15
2.82
0.89
2.82
1.49
2.15
2.15
5.56
3.5
```

In [47]:

```
#Tells you the data above is within 5 miles or not by outputting: True or Fa

Locations_less_than_5_miles=[]
for i in dtf['Longitude']:
    for x in dtf['Latitude']:
        radiusEarth= 3961 #miles. If you want km, put in 6373 km in instead
        lat1 = math.radians(36.14)
        lon1 = math.radians(-115.19)
        lat2 = math.radians(x)  #plug and chug
        lon2 = math.radians(i)  #plug and chug
        dlon = lon2 - lon1
        dlat = lat2 - lat1
        a = math.sin(dlat / 2)**2 + math.cos(lat1) * math.cos(lat2) * math.s
        c = 2 * math.atan2(math.sqrt(a), math.sqrt(1 - a))
        distance = round(radiusEarth * c,2) #in miles
        if distance < 5:  #5 mile threshold
            Locations_less_than_5_miles.append(True)
        else:
            Locations_less_than_5_miles.append(False)
Locations_less_than_5_miles[:]
```

Out[47]:

```
[True,
 False,
 True,
 False,
 True,
 True,
 True,
 False,
 True,
 False,
 True,
 True,
 True,
 True,
 True,
 True,
 True,
 False,
 True,
```

In [48]:

```
#Combine the data and the corresponding

ID_Locations_less_than_5_miles=pd.Series(Locations_less_than_5_miles)
dtf["Within_5_miles"] = ID_Locations_less_than_5_miles[:] #Index the true an

#This is a random string generator for the # of evaluations
dtf["% of People Infected"] = np.random.choice(["%80 > Infected","%60 > Infe

#dtf["# of People Processed"] = np.random.choice(["People using app:Greater

# The fraction of people infected from the # of evaluations
dtf["# of People Processed"] = np.random.randint(low=0, high=100, size=len(d
#dtf["Infected%"] = np.random.randint(low=0, high=66, size=len(dtf))

#Toggle to play with # of data points will output
#dtf.head(10)
```

In [49]:

```
#Display only data points within 5 miles only!
dtf[dtf.Within_5_miles==True]
```

Out[49]:

	id	City	Street Address	Longitude	Latitude	Within_5_miles	% of People Infected	# of People Processed
0	0	Las Vegas	4507 Flamingo Rd	-115.20	36.12	True	%80 > Infected	99
2	2	Las Vegas	3200 LAS VEGAS BLVD. S., STE 1795	-115.17	36.13	True	%80 > Infected	16
4	4	Las Vegas	3730 LAS VEGAS BLVD S	-115.18	36.11	True	%20 > Infected	90
5	5	Las Vegas	129 East Fremont Street	-115.14	36.17	True	%80 > Infected	35
6	6	Las Vegas	3475 Las Vegas Blvd S	-115.17	36.12	True	%20 > Infected	89
...	...	...	...	...	...	...	...	...
150	150	Las Vegas	3900 Las Vegas Blvd	-115.18	36.09	True	%20 > Infected	26
151	151	Las Vegas	3850 Las Vegas Blvd So	-115.18	36.10	True	%20 > Infected	3
153	153	Las Vegas	9701 W. Flamingo Road, #1	-115.30	36.11	True	%20 < Infected	58
154	154	Las Vegas	3950 Las Vegas Blvd So	-115.17	36.09	True	%80 > Infected	36
155	155	Las Vegas	9151 West Sahara, Suite 110	-115.30	36.14	True	%20 > Infected	17

117 rows × 8 columns

In [62]:

```
#Initialize and center the map to be simulated in Las Vegas

city = "Las Vegas"
## get location
locator = geopy.geocoders.Nominatim(user_agent="MyCoder")
#location = locator.geocode(city)
print(location)
## keep latitude and longitude only
location = [36.14, -115.19]  #3301 W. Sahara Ave
print("[lat, long]:", location)
```

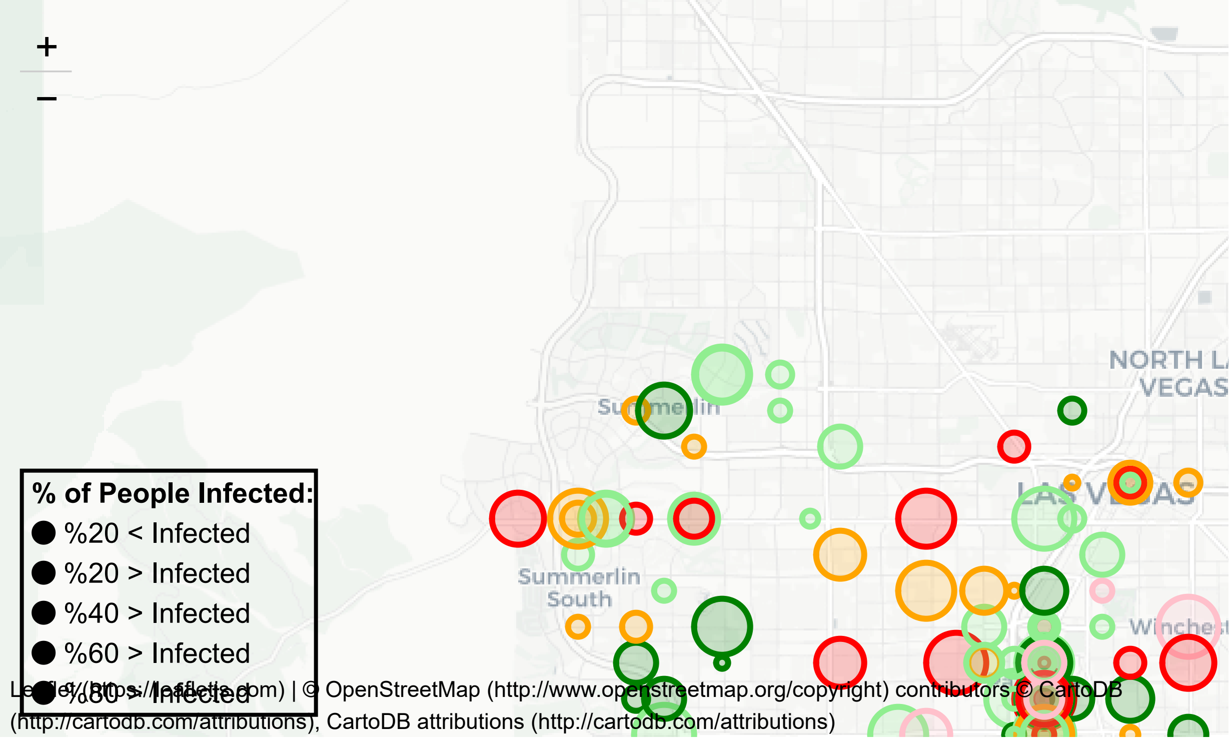
```
[36.1672559, -115.1485163]
[lat, long]: [36.14, -115.19]
```

In [64]:

```
x, y = "Latitude", "Longitude"
color = "% of People Infected"
size = "# of People Processed"
#popup = "Cost" & "Street Address"

popup = "Street Address"
data = dtf[dtf.Within_5_miles==True]

## create color column
lst_colors=["green","lightgreen","pink","orange","red"]

#lst_colors=["red","pink","orange","darkblue","black"]

lst_elements = sorted(list(dtf[color].unique()))
data["color"] = data[color].apply(lambda x:
                lst_colors[lst_elements.index(x)])
## create size column (scaled)
scaler = preprocessing.MinMaxScaler(feature_range=(3,15))
data["size"] = scaler.fit_transform(
               data[size].values.reshape(-1,1)).reshape(-1)

## initialize the map with the starting location
map_ = folium.Map(location=location, tiles="cartodbpositron",
                  zoom_start=11)
## add points
data.apply(lambda row: folium.CircleMarker(
           location=[row[x],row[y]], popup=row[popup],
           color=row["color"], fill=True,
           radius=row["size"]).add_to(map_), axis=1)
## add html legend
legend_html = """<div style="position:fixed; bottom:10px; left:10px; border:
for i in lst_elements:
     legend_html = legend_html+"""&nbsp;<i class="fa fa-circle
     fa-1x" style="color:"""+lst_colors[lst_elements.index(i)]+""">
     </i>&nbsp;"""+str(i)+"""<br>"""
legend_html = legend_html+"""</div>"""
map_.get_root().html.add_child(folium.Element(legend_html))

## plot the map
map_
```

```
<ipython-input-64-1271a957e97f>:15: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/s
table/user_guide/indexing.html#returning-a-view-versus-a-copy (https://panda
s.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-ver
sus-a-copy)
  data["color"] = data[color].apply(lambda x:
<ipython-input-64-1271a957e97f>:19: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/s
table/user_guide/indexing.html#returning-a-view-versus-a-copy (https://panda
s.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-ver
sus-a-copy)
  data["size"] = scaler.fit_transform(
```

Out[64]:



Leaflet (https://leafletjs.com) | © OpenStreetMap (http://www.openstreetmap.org/copyright) contributors © CartoDB (http://cartodb.com/attributions), CartoDB attributions (http://cartodb.com/attributions)

In [ ]:

In [ ]: