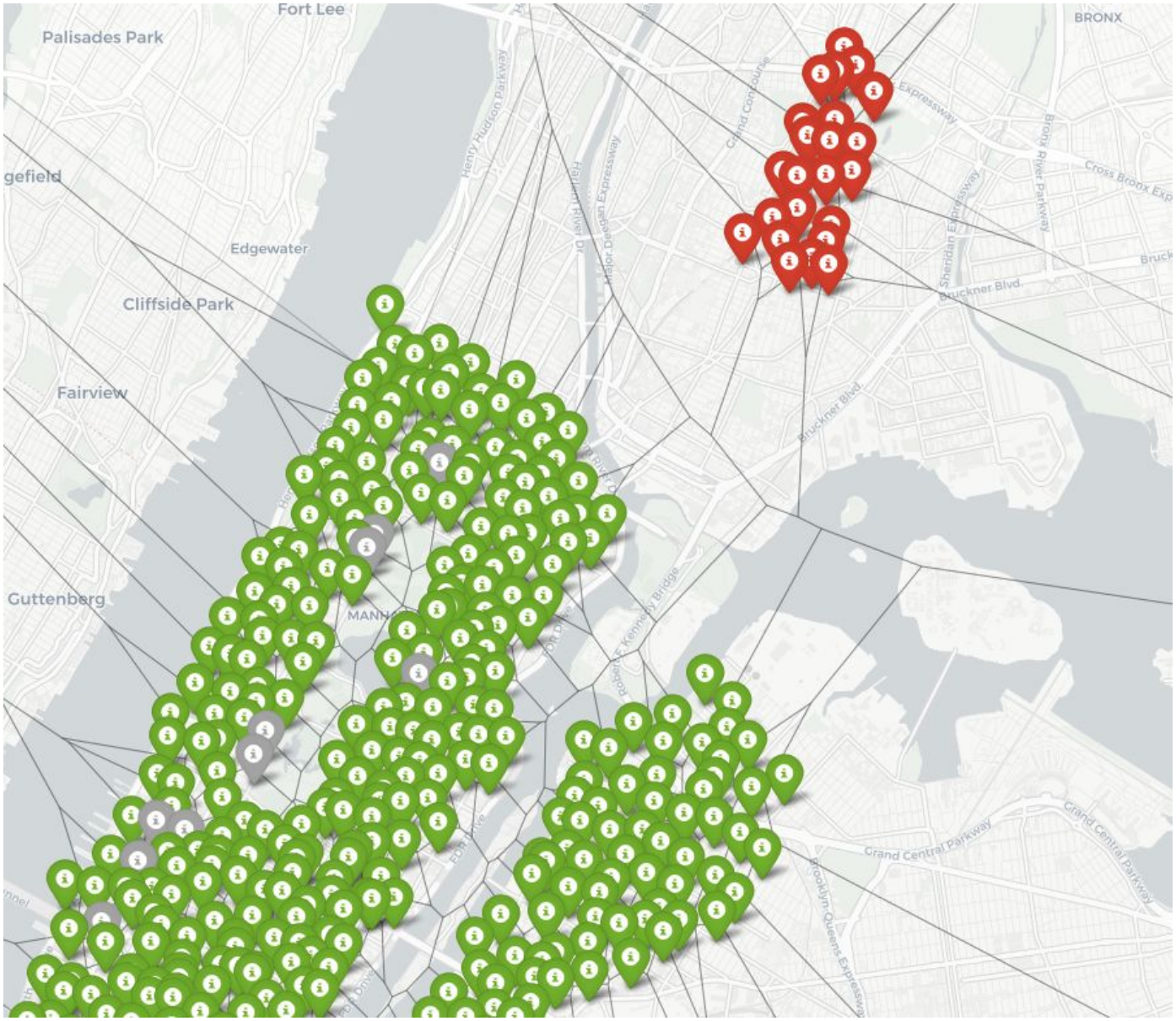
Yusef Wray - 23693925
Catchments for Proposed Bike Locations (HC12)
Neighborhood: Morrisania, Bronx



```
import pandas as pd
from scipy.spatial import Voronoi, voronoi_plot_2d
import folium

#File with library locations in NYC:
#libs = pd.read_csv('LIBRARY.csv')

#Read directly from the citibike feed (can be replaced by static file):
stat_json = pd.read_json('https://feeds.citibikenyc.com/stations/stations.json')
stat_csv = pd.read_csv('stationLocations.csv', skiprows = 1)

#a datagrame that will hold relevant series extracted from the json
stations = pd.DataFrame(columns=['stationName', 'latitude', 'longitude', 'statusValue'])

data = pd.DataFrame({
'stationName':['Washington Ave and Cross Bronx Expressway Service Rd','3rd Ave and East 174th St','Washington Av and East
173rd St','Park Ave and Between East 173rd St and East 172nd St','Crotona Ave','Crotona Pl and East 171st St','Park Ave and St
Pauls Pl','Washington Ave and East 170th St','Fulton Ave and East 170th St','Crotona Ave and Cortona Park S','Union Ave and
Boston Rd','Franklin Ave and East 169th St','3rd Ave and East 168th St','Park Ave and East 168th St','Melrose Ave and East 163rd
St','Washington Ave and East 165th St','Franklin Ave and East 166th St','Tinton Ave and East 166th St','3rd Ave and Boston
Rd','Tinton Ave and East 165th St','Union Ave and East 163rd St','Cauldwell Ave and Between East 163rd St and East 161st
St','Between Trinity Ave and Tinton Ave and East 163rd St'],
'latitude':[40.8441,40.8423,40.8417,40.8414,40.8396,40.8368,40.8365,40.8353,40.8347,40.8346,40.8318,40.8314,40.8312,40.8318
,40.8255,40.827,40.828,40.8263,40.8249,40.8247,40.8224,40.8227,40.823],
'longitude':[-73.8998,-73.8983,-73.9011,-73.9029,-73.8959,-73.9011,-73.9054,-73.9047,-73.9018,-73.8981,-73.8988,-73.9022,-73.90
6,-73.908,-73.9132,-73.9093,-73.9061,-73.9016,-73.9083,-73.9023,-73.902,-73.9071,-73.9041],
'statusValue':['Proposed','Proposed','Proposed','Proposed','Proposed','Proposed','Proposed','Proposed','Proposed','Proposed','Propo
sed','Proposed','Proposed','Proposed','Proposed','Proposed','Proposed','Proposed','Proposed','Proposed','Proposed','Proposed','Pro
posed']    })

#Create a map object, focused on NYC:
mapVor = folium.Map(location=[40.75, -73.96],tiles="Cartodb Positron",zoom_start=14)

#Pull out data from JSON beanlists into the stations dataframe
for i,row in stat_json.iterrows():
    #Pull out the station info:
    beanList = row['stationBeanList']
    #Create new row with desired station information
    newRow = {'stationName': beanList['stationName'], 'latitude': beanList['latitude'], 'longitude':
beanList['longitude'],'statusValue':beanList['statusValue']}
    #add row to dataframe
    stations = stations.append(newRow, ignore_index=True, sort=False)
stations = stations.append(data)

#Create lists to hold coordinates for vornoi diagram:
coords = []

#Add markers for each station to map
for index, row in stations.iterrows():
    name = row['stationName'] #repeat for LAT, LON, STATUS
    lat = float(row['latitude']) #repeat for others
    lon = float(row['longitude'])
    status = row['statusValue']

    #Add the [lat,lon] to list of coordinates:
    coords.append([lat,lon])
```

```
    if (status == 'Not In Service'):
        i = folium.Icon(color='lightgray')
    elif (status == 'Proposed'):
        i = folium.Icon(color='red')
    else:
        i = folium.Icon(color='green')
    #Add a marker to the map:
    folium.Marker([lat,lon] ,popup = name, icon= i).add_to(mapVor)
    print("Processing:", name)

#Use scipy to make the voronoi diagram:
vor = Voronoi(coords)
#Plot with matplotlib to check that it's working:
import matplotlib.pyplot as plt
voronoi_plot_2d(vor)
plt.show()


#Use geojson file to write out the features
from geojson import FeatureCollection, Feature, Polygon

#The output file, to contain the Voronoi diagram we computed:
vorJSON = open('libVorBike.json', 'w')
point_voronoi_list = []
feature_list = []
for region in range(len(vor.regions)-1):
#for region in range(9):
    vertex_list = []
    for x in vor.regions[region]:
        #Not sure how to map the "infinite" point, so, leave off those regions for now:
        if x == -1:
            break;
        else:
            #Get the vertex out of the list, and flip the order for folium:
            vertex = vor.vertices[x]
            vertex = (vertex[1], vertex[0])
        vertex_list.append(vertex)
    #Save the vertex list as a polygon and then add to the feature_list:
    polygon = Polygon([vertex_list])
    feature = Feature(geometry=polygon, properties={})
    feature_list.append(feature)

#Write the features to the new file:
feature_collection = FeatureCollection(feature_list)
print (feature_collection, file=vorJSON)
vorJSON.close()
#Add the voronoi layer to the map:
#mapVor.geo_json(geo_path= 'libVor.json', fill_color = "BuPu",
#                fill_opacity=0.01, line_opacity=0.25)
#lib_geo = os.path.join('data', 'libVor.json')
mapVor.choropleth(geo_data='libVorBike.json', fill_color = "BuPu",
                  fill_opacity=0.01, line_opacity=0.25)
#folium.LayerControl().add_to(mapVor)

mapVor.save(outfile='libVorBike.html')
```