Table of contents

- Introduction: Business Problem
- Data
- Analysis
- Results and Discussion
- Conclusion

Capstone Project - The Battle of the Neighborhoods (Week 4)

Applied Data Science Capstone by IBM/Coursera

Introduction: Business Problem

In this project we will try to find an optimal location for a restaurant. Specifically, this report will be targeted to stakeholders interested in opening an **Italian restaurant** in **Houston**, Texas.

Since there are lots of restaurants in Houston we will try to detect **locations that are not already crowded with restaurants**. We are also particularly interested in **areas with no Italian restaurants in vicinity**. We would also prefer locations **as close to city center as possible**, assuming that first two conditions are met.

We will use our data science powers to generate a few most promissing neighborhoods based on this criteria. Advantages of each area will then be clearly expressed so that best possible final location can be chosen by stakeholders.

Data

Based on definition of our problem, factors that will influence our decission are:

- number of existing restaurants in the neighborhood (any type of restaurant)
- number of and distance to Italian restaurants in the neighborhood, if any
- distance of neighborhood from city center

We decided to use regularly spaced grid of locations, centered around city center, to define our neighborhoods.

Following data sources will be needed to extract/generate the required information:

- centers of candidate areas will be generated algorithmically and approximate addresses of centers of those areas will be obtained using **Google Maps API reverse geocoding**
- number of restaurants and their type and location in every neighborhood will be obtained using **Foursquare API**
- coordinate of Berlin center will be obtained using **Google Maps API geocoding** of well known Houston location (Midtown)

Neighborhood Candidates

Let's create latitude & longitude coordinates for centroids of our candidate neighborhoods. We will create a grid of cells covering our area of interest which is aprox. 12x12 killometers centered around Houston city center.

Let's first find the latitude & longitude of Houston city center, using specific, well known address and Google Maps geocoding API.

In [41]:
```
# The code was removed by Watson Studio for sharing.
google_api_key = 'AIzaSyALzGNYUljA-SMpE4x-pV_9fS0_VcezH2A'
```

In [42]:
```
import requests

def get_coordinates(api_key, address, verbose=False):
    try:
        url = 'https://maps.googleapis.com/maps/api/geocode/json?key={}&addres
s={}'.format(api_key, address)
        response = requests.get(url).json()
        if verbose:
            print('Google Maps API JSON result =>', response)
        results = response['results']
        geographical_data = results[0]['geometry']['location'] # get geographi
cal coordinates
        lat = geographical_data['lat']
        lon = geographical_data['lng']
        return [lat, lon]
    except:
        return [None, None]

address = 'Midtown, Houston, Texas'
houston_center = get_coordinates(google_api_key, address)
print('Coordinate of {}: {}'.format(address, houston_center))
```

```
Coordinate of Midtown, Houston, Texas: [29.740795, -95.3756344]
```

Now let's create a grid of area candidates, equaly spaced, centered around city center and within ~6km from Alexanderplatz. Our neighborhoods will be defined as circular areas with a radius of 300 meters, so our neighborhood centers will be 600 meters apart.

To accurately calculate distances we need to create our grid of locations in Cartesian 2D coordinate system which allows us to calculate distances in meters (not in latitude/longitude degrees). Then we'll project those coordinates back to latitude/longitude degrees to be shown on Folium map. So let's create functions to convert between WGS84 spherical coordinate system (latitude/longitude degrees) and UTM Cartesian coordinate system (X/Y coordinates in meters).

In [43]:
```
#!pip install shapely
import shapely.geometry

#!pip install pyproj
import pyproj

import math

def lonlat_to_xy(lon, lat):
    proj_latlon = pyproj.Proj(proj='latlong',datum='WGS84')
    proj_xy = pyproj.Proj(proj="utm", zone=33, datum='WGS84')
    xy = pyproj.transform(proj_latlon, proj_xy, lon, lat)
    return xy[0], xy[1]

def xy_to_lonlat(x, y):
    proj_latlon = pyproj.Proj(proj='latlong',datum='WGS84')
    proj_xy = pyproj.Proj(proj="utm", zone=33, datum='WGS84')
    lonlat = pyproj.transform(proj_xy, proj_latlon, x, y)
    return lonlat[0], lonlat[1]

def calc_xy_distance(x1, y1, x2, y2):
    dx = x2 - x1
    dy = y2 - y1
    return math.sqrt(dx*dx + dy*dy)

print('Coordinate transformation check')
print('-------------------------------')
print('Houston center longitude={}, latitude={}'.format(houston_center[1], hou
ston_center[0]))
x, y = lonlat_to_xy(houston_center[1], houston_center[0])
print('Houston center UTM X={}, Y={}'.format(x, y))
lo, la = xy_to_lonlat(x, y)
print('Houston center longitude={}, latitude={}'.format(lo, la))
```

```
Coordinate transformation check
-------------------------------
Houston center longitude=-95.3756344, latitude=29.740795
Houston center UTM X=-6760694.587620011, Y=13477124.781070549
Houston center longitude=-95.37563440000389, latitude=29.740794999994225
```

Let's create a **hexagonal grid of cells**: we offset every other row, and adjust vertical row spacing so that **every cell center is equally distant from all it's neighbors**.

In [44]:
```
houston_center_x, houston_center_y = lonlat_to_xy(houston_center[1], houston_c
enter[0]) # City center in Cartesian coordinates

k = math.sqrt(3) / 2 # Vertical offset for hexagonal grid cells
x_min = houston_center_x - 6000
x_step = 600
y_min = houston_center_y - 6000 - (int(21/k)*k*600 - 12000)/2
y_step = 600 * k

latitudes = []
longitudes = []
distances_from_center = []
xs = []
ys = []
for i in range(0, int(21/k)):
    y = y_min + i * y_step
    x_offset = 300 if i%2==0 else 0
    for j in range(0, 21):
        x = x_min + j * x_step + x_offset
        distance_from_center = calc_xy_distance(houston_center_x, houston_cent
er_y, x, y)
        if (distance_from_center <= 6001):
            lon, lat = xy_to_lonlat(x, y)
            latitudes.append(lat)
            longitudes.append(lon)
            distances_from_center.append(distance_from_center)
            xs.append(x)
            ys.append(y)

print(len(latitudes), 'candidate neighborhood centers generated.')
```

```
364 candidate neighborhood centers generated.
```

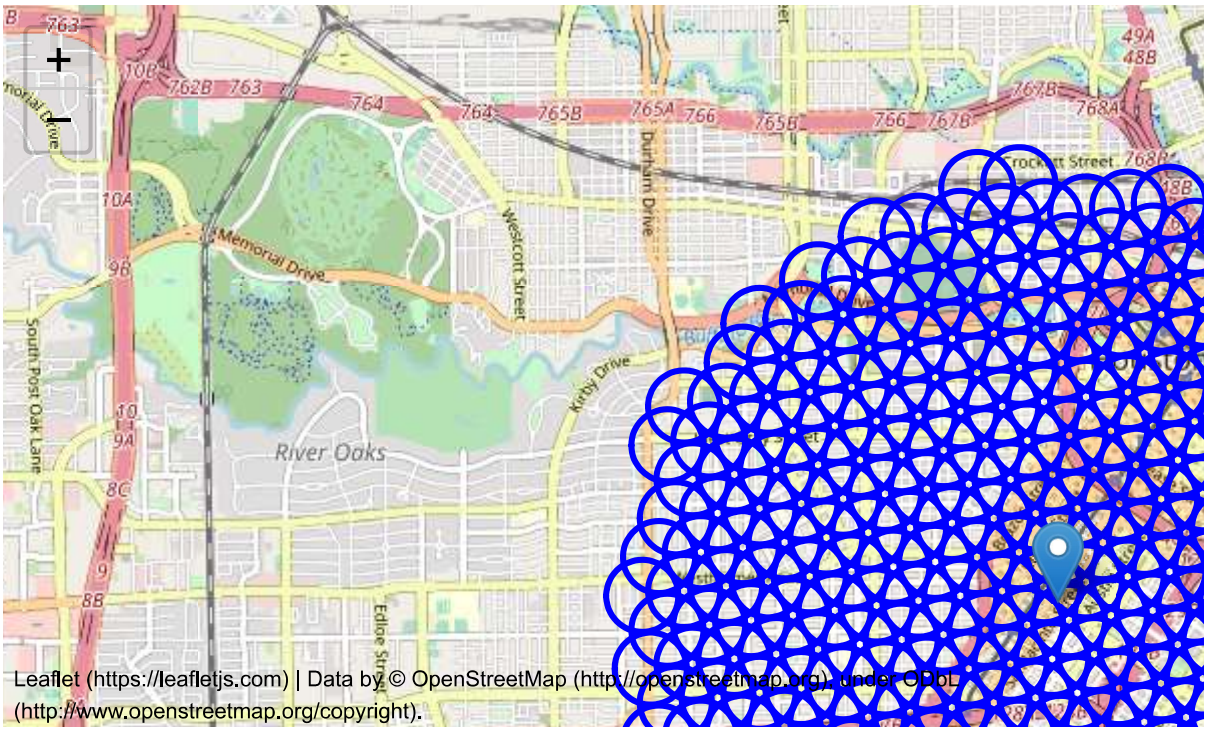
Let's visualize the data we have so far: city center location and candidate neighborhood centers:

In [45]:
```
#!pip install folium

import folium
```

In [46]:
```
map_houston = folium.Map(location=houston_center, zoom_start=13)
folium.Marker(houston_center, popup='Midtown').add_to(map_houston)
for lat, lon in zip(latitudes, longitudes):
    #folium.CircleMarker([lat, lon], radius=2, color='blue', fill=True, fill_c
olor='blue', fill_opacity=1).add_to(map_houston)
    folium.Circle([lat, lon], radius=300, color='blue', fill=False).add_to(map
_houston)
    #folium.Marker([lat, lon]).add_to(map_houston)
map_houston
```

Out[46]:



OK, we now have the coordinates of centers of neighborhoods/areas to be evaluated, equally spaced (distance from every point to it's neighbors is exactly the same) and within ~6km from Midtown.

Let's now use Google Maps API to get approximate addresses of those locations.

In [47]:
```
def get_address(api_key, latitude, longitude, verbose=False):
    try:
        url = 'https://maps.googleapis.com/maps/api/geocode/json?key={}&latlng
={},{}'.format(api_key, latitude, longitude)
        response = requests.get(url).json()
        if verbose:
            print('Google Maps API JSON result =>', response)
        results = response['results']
        address = results[0]['formatted_address']
        return address
    except:
        return None

addr = get_address(google_api_key, houston_center[0], houston_center[1])
print('Reverse geocoding check')
print('-----------------------')
print('Address of [{}, {}] is: {}'.format(houston_center[0], houston_center[1
], addr))
```

```
Reverse geocoding check
-----------------------
Address of [29.740795, -95.3756344] is: 1109 Elgin St, Houston, TX 77004, USA
```

In [48]:
```
print('Obtaining location addresses: ', end='')
addresses = []
for lat, lon in zip(latitudes, longitudes):
    address = get_address(google_api_key, lat, lon)
    if address is None:
        address = 'NO ADDRESS'
    address = address.replace(', USA', '') # We don't need country part of add
ress
    addresses.append(address)
    print(' .', end='')
print(' done.')
```

```
Obtaining location addresses:  . . . . . .
. . . . . .
. . . . . .
. . . . . .
. . . . . .
. . . . . .
. . . . . .
. . . . . .
. . . . . done.
```

In [49]:
```
addresses[150:170]
```

Out[49]:
```
['1795-1799 Memorial Dr, Houston, TX 77007',
 '2911 Washington Ave, Houston, TX 77007',
 '902 Memorial Heights Dr, Houston, TX 77007',
 '3710 Wheeler Ave, Houston, TX 77004',
 '3500 Cleburne St, Houston, TX 77004',
 '3893 Nettleton St, Houston, TX 77004',
 '3618 Briley St, Houston, TX 77004',
 '2820 Holman St, Houston, TX 77004',
 '3404 Live Oak St, Houston, TX 77004',
 '3217 Hutchins St, Houston, TX 77004',
 '1814 Rosalie St, Houston, TX 77004',
 '1521 Anita St, Houston, TX 77004',
 '2820 Caroline St, Houston, TX 77004',
 '2727 Travis St, Houston, TX 77006',
 'Smith St @ MC Gowen St, Houston, TX 77006',
 '9 Bremond St, Houston, TX 77006',
 '1417 Oneil St, Houston, TX 77019',
 '1323 Genesee St, Houston, TX 77019',
 'W Dallas St @ Taft St, Houston, TX 77019',
 '2002 Buffalo Terrace, Houston, TX 77019']
```

Looking good. Let's now place all this into a Pandas dataframe.

In [50]:
```
import pandas as pd

df_locations = pd.DataFrame({'Address': addresses,
                             'Latitude': latitudes,
                             'Longitude': longitudes,
                             'X': xs,
                             'Y': ys,
                             'Distance from center': distances_from_center})

df_locations.head(10)
```

Out[50]:

	Address	Latitude	Longitude	X	Y	Distance from center
0	215 Velasco St, Houston, TX 77003	29.751059	-95.341556	-6.762495e+06	1.347141e+07	5992.495307
1	2910 Commerce St, Houston, TX 77003	29.753591	-95.343702	-6.761895e+06	1.347141e+07	5840.376700
2	17 N St Charles St, Houston, TX 77003	29.756123	-95.345849	-6.761295e+06	1.347141e+07	5747.173218
3	2217 Canal St, Houston, TX 77003	29.758656	-95.347997	-6.760695e+06	1.347141e+07	5715.767665
4	Eastex Fwy, Houston, TX	29.761188	-95.350144	-6.760095e+06	1.347141e+07	5747.173218
5	511 McKee St, Houston, TX 77002	29.763721	-95.352292	-6.759495e+06	1.347141e+07	5840.376700
6	720 Walnut St, Houston, TX 77002	29.766254	-95.354440	-6.758895e+06	1.347141e+07	5992.495307
7	77001 Railroad, Houston, TX 77003	29.745639	-95.340849	-6.763395e+06	1.347193e+07	5855.766389
8	9363 Roberts St, Houston, TX 77003	29.748171	-95.342995	-6.762795e+06	1.347193e+07	5604.462508
9	3000 Hawkins St, Houston, TX 77003	29.750703	-95.345142	-6.762195e+06	1.347193e+07	5408.326913

...and let's now save/persist this data into local file.

In [51]:
```
df_locations.to_pickle('./locations.pkl')
```

Foursquare

Now that we have our location candidates, let's use Foursquare API to get info on restaurants in each neighborhood.

We're interested in venues in 'food' category, but only those that are proper restaurants - coffe shops, pizza places, bakeries etc. are not direct competitors so we don't care about those. So we will include in out list only venues that have 'restaurant' in category name, and we'll make sure to detect and include all the subcategories of specific 'Italian restaurant' category, as we need info on Italian restaurants in the neighborhood.

Foursquare credentials are defined in hidden cell bellow.

In [52]:
```
# The code was removed by Watson Studio for sharing.
foursquare_client_id, foursquare_client_secret='Q5PMIAXXGTKJ4NQZ4FMULS22UKAJYZ
FVE2OPUPGM2UNXTGK3','HHNBQ11JCSD3J15UBUEUMQKAV5RGZMPQ44O4IJWAR0YR3BTI'
```

In [53]:
```
# Category IDs corresponding to Italian restaurants were taken from Foursquare
web site (https://developer.foursquare.com/docs/resources/categories):

food_category = '4d4b7105d754a06374d81259' # 'Root' category for all food-rela
ted venues

italian_restaurant_categories = ['4bf58dd8d48988d110941735','55a5a1ebe4b013909
087cbb6','55a5a1ebe4b013909087cb7c',
                                 '55a5a1ebe4b013909087cba7','55a5a1ebe4b013909
087cba1','55a5a1ebe4b013909087cba4',
                                 '55a5a1ebe4b013909087cb95','55a5a1ebe4b013909
087cb89','55a5a1ebe4b013909087cb9b',
                                 '55a5a1ebe4b013909087cb98','55a5a1ebe4b013909
087cbbf','55a5a1ebe4b013909087cb79',
                                 '55a5a1ebe4b013909087cbb0','55a5a1ebe4b013909
087cbb3','55a5a1ebe4b013909087cb74',
                                 '55a5a1ebe4b013909087cbaa','55a5a1ebe4b013909
087cb83','55a5a1ebe4b013909087cb8c',
                                 '55a5a1ebe4b013909087cb92','55a5a1ebe4b013909
087cb8f','55a5a1ebe4b013909087cb86',
                                 '55a5a1ebe4b013909087cbb9','55a5a1ebe4b013909
087cb7f','55a5a1ebe4b013909087cbbc',
                                 '55a5a1ebe4b013909087cb9e','55a5a1ebe4b013909
087cbc2','55a5a1ebe4b013909087cbad']

def is_restaurant(categories, specific_filter=None):
    restaurant_words = ['restaurant', 'diner', 'taverna', 'steakhouse']
    restaurant = False
    specific = False
    for c in categories:
        category_name = c[0].lower()
        category_id = c[1]
        for r in restaurant_words:
            if r in category_name:
                restaurant = True
        if 'fast food' in category_name:
            restaurant = False
        if not(specific_filter is None) and (category_id in specific_filter):
            specific = True
            restaurant = True
    return restaurant, specific

def get_categories(categories):
    return [(cat['name'], cat['id']) for cat in categories]

def format_address(location):
    address = ', '.join(location['formattedAddress'])
    address = address.replace(', USA', '')
    return address

def get_venues_near_location(lat, lon, category, client_id, client_secret, rad
ius=500, limit=100):
    version = '20180724'
    url = 'https://api.foursquare.com/v2/venues/explore?client_id={}&client_se
cret={}&v={}&ll={},{}&categoryId={}&radius={}&limit={}'.format(
        client_id, client_secret, version, lat, lon, category, radius, limit)
```

```
    try:
        results = requests.get(url).json()['response']['groups'][0]['items']
        venues = [(item['venue']['id'],
                   item['venue']['name'],
                   get_categories(item['venue']['categories']),
                   (item['venue']['location']['lat'], item['venue']['location'
]['lng']),
                   format_address(item['venue']['location']),
                   item['venue']['location']['distance']) for item in results]
    except:
        venues = []
    return venues
```

In [54]:
```
# Let's now go over our neighborhood locations and get nearby restaurants; w
e'll also maintain a dictionary of all found restaurants and all found italian
restaurants

import pickle

def get_restaurants(lats, lons):
    restaurants = {}
    italian_restaurants = {}
    location_restaurants = []

    print('Obtaining venues around candidate locations:', end='')
    for lat, lon in zip(lats, lons):
        # Using radius=350 to meke sure we have overlaps/full coverage so we d
on't miss any restaurant (we're using dictionaries to remove any duplicates re
sulting from area overlaps)
        venues = get_venues_near_location(lat, lon, food_category, foursquare_
client_id, foursquare_client_secret, radius=350, limit=100)
        area_restaurants = []
        for venue in venues:
            venue_id = venue[0]
            venue_name = venue[1]
            venue_categories = venue[2]
            venue_latlon = venue[3]
            venue_address = venue[4]
            venue_distance = venue[5]
            is_res, is_italian = is_restaurant(venue_categories, specific_filt
er=italian_restaurant_categories)
            if is_res:
                x, y = lonlat_to_xy(venue_latlon[1], venue_latlon[0])
                restaurant = (venue_id, venue_name, venue_latlon[0], venue_lat
lon[1], venue_address, venue_distance, is_italian, x, y)
                if venue_distance<=300:
                    area_restaurants.append(restaurant)
                restaurants[venue_id] = restaurant
                if is_italian:
                    italian_restaurants[venue_id] = restaurant
        location_restaurants.append(area_restaurants)
        print(' .', end='')
    print(' done.')
    return restaurants, italian_restaurants, location_restaurants

# Try to load from local file system in case we did this before
restaurants = {}
italian_restaurants = {}
location_restaurants = []
loaded = False
try:
    with open('restaurants_350.pkl', 'rb') as f:
        restaurants = pickle.load(f)
    with open('italian_restaurants_350.pkl', 'rb') as f:
        italian_restaurants = pickle.load(f)
    with open('location_restaurants_350.pkl', 'rb') as f:
        location_restaurants = pickle.load(f)
    print('Restaurant data loaded.')
    loaded = True
```

```
except:
    pass

# If load failed use the Foursquare API to get the data
if not loaded:
    restaurants, italian_restaurants, location_restaurants = get_restaurants(l
atitudes, longitudes)

    # Let's persists this in local file system
    with open('restaurants_350.pkl', 'wb') as f:
        pickle.dump(restaurants, f)
    with open('italian_restaurants_350.pkl', 'wb') as f:
        pickle.dump(italian_restaurants, f)
    with open('location_restaurants_350.pkl', 'wb') as f:
        pickle.dump(location_restaurants, f)
```

```
Restaurant data loaded.
```

In [55]:

```
import numpy as np

print('Total number of restaurants:', len(restaurants))
print('Total number of Italian restaurants:', len(italian_restaurants))
print('Percentage of Italian restaurants: {:.2f}%'.format(len(italian_restaura
nts) / len(restaurants) * 100))
print('Average number of restaurants in neighborhood:', np.array([len(r) for r
in location_restaurants]).mean())
```

```
Total number of restaurants: 417
Total number of Italian restaurants: 32
Percentage of Italian restaurants: 7.67%
Average number of restaurants in neighborhood: 2.6483516483516483
```

In [56]:
```
print('List of all restaurants')
print('-----------------------')
for r in list(restaurants.values())[:10]:
    print(r)
print('...')
print('Total:', len(restaurants))
```

```
List of all restaurants
-----------------------
('4c61942779d1e21eba0bd415', 'Champ Burger', 29.749795653429068, -95.34034977
76047, '304 Sampson St (Garrow), Houston, TX 77003, United States', 313, Fals
e, -6762807.250639517, 13471390.98087352)
('4e60ecb3c65b2dc89ef32613', 'Carino Taco Truck', 29.751084895739087, -95.342
4810629325, 'Houston, TX, United States', 260, False, -6762399.097309045, 134
71529.744116662)
('52ceb507498e403c09326dc7', 'Andes Cafe', 29.758162158656575, -95.3462051502
4655, '2311 Canal St Ste 104, Houston, TX 77003, United States', 181, False,
-6760947.219556388, 13471225.608930212)
('4a3af801f964a52079a01fe3', "The Original Ninfa's on Navigation", 29.7568592
43005565, -95.34260583136786, '2704 Navigation Blvd (at N Delano St), Housto
n, TX 77003, United States', 323, False, -6761502.469417095, 13470892.2841225
08)
('4bbf614d85fbb713ed137267', 'Dona Maria Mexican Cafe', 29.757914652074994, -
95.34322994122218, '2601 Navigation Blvd (live oak), Houston, TX 77003, Unite
d States', 322, False, -6761279.166958111, 13470856.13152513)
('4bb8a31c1261d13a5bc2e898', 'Alamo Tamale & Taco', 29.758526697553105, -95.3
4696765868095, '2310 Navigation Blvd (at Jensen Dr), Houston, TX 77003, Unite
d States', 100, False, -6760816.039892961, 13471286.187137429)
('4c3384f9452620a16dbc230f', 'Merida Mexican Restaurant', 29.758249905896466,
-95.34384913417873, '2509 Navigation Blvd, Houston, TX 77003, United States',
305, False, -6761166.634308094, 13470900.890620211)
('52386ba511d269a17aa4bffd', '2nd ward', 29.757387254792956, -95.343252387843
5, 'Houston, TX, United States', 287, False, -6761357.710753477, 13470918.868
057627)
('4be1af29edbb0f472251a615', "Irma's", 29.7603, -95.352099, '22 N Chenevert S
t (at Ruiz St), Houston, TX 77002, United States', 168, False, -6760037.38109
6342, 13471770.781580744)
('4f4a9c6ce4b0458782c6cdb3', 'la familia meat market', 29.761497663305892, -9
5.35019988547644, 'Houston, TX 77003, United States', 319, False, -6760041.72
5000174, 13471381.445204645)
...
Total: 417
```

In [57]:
```
print('List of Italian restaurants')
print('---------------------------')
for r in list(italian_restaurants.values())[:10]:
    print(r)
print('...')
print('Total:', len(italian_restaurants))
```

```
List of Italian restaurants
---------------------------
('4a6f3427f964a5209ad51fe3', "Mandola's Deli", 29.73745357701867, -95.3395870
4477517, '4105 Leeland St (at Cullen), Houston, TX 77023, United States', 19
3, True, -6764773.096430207, 13472687.650772782)
('58a3b9ed51666a278d92af9e', 'Potente', 29.757237886603125, -95.3574474900741
5, '1515 Texas St (Crawford St.), Houston, TX 77002, United States', 132, Tru
e, -6759977.412849978, 13472832.062630547)
('5807b8d338faaedf9e01192b', 'Osso & Kristalla', 29.7574509, -95.3579026, '15
15 Texas St (Crawford), Houston, TX 77002, United States', 84, True, -675989
9.813779492, 13472868.709354943)
('4f42ccd5e4b0c5334de17a11', 'Quatro Bar & Restaurant', 29.763022097691675, -
95.36229359342164, 'Houston, TX 77002, United States', 84, True, -6758613.072
172567, 13472823.859249126)
('5815e4be38fab8bf7b564739', 'Grotto Downtown', 29.7539178, -95.3571966, '100
1 Avenida de las Americas Ste A, Houston, TX 77010, United States', 302, Tru
e, -6760510.419725645, 13473174.762345675)
('4b11cb7ff964a520ca8423e3', "Birraporetti's", 29.762461731403604, -95.364669
77857937, '500 Louisiana St (Prairie St), Houston, TX 77002, United States',
191, True, -6758463.934960416, 13473204.609357985)
('4b510fe4f964a520554027e3', 'Perbacco', 29.760257437720647, -95.364773395163
25, '700 Milam St, Houston, TX 77002, United States', 333, True, -6758791.003
323279, 13473468.204478525)
('41326e00f964a52084121fe3', 'Mingalone Italian Bar & Grill', 29.762381, -95.
366655, '540 Texas St (btw Bagby St and Smith St), Houston, TX 77002, United
States', 349, True, -6758280.033575494, 13473478.801656555)
('4b7c8437f964a5203b982fe3', 'Quattro', 29.754122193587296, -95.3624689688728
6, '1300 Lamar St (Austin St), Houston, TX 77010, United States', 129, True,
-6759957.774785714, 13473855.749917114)
('4f316bbee4b08a0e6d7dbf3d', 'Brooklyn Meatball Company', 29.75732460146048,
-95.36526623748942, '930 Main St Ste T240C (Tunnel @ Travis & McKinney), Hous
ton, TX 77002, United States', 82, True, -6759191.0773128085, 13473866.353732
675)
...
Total: 32
```

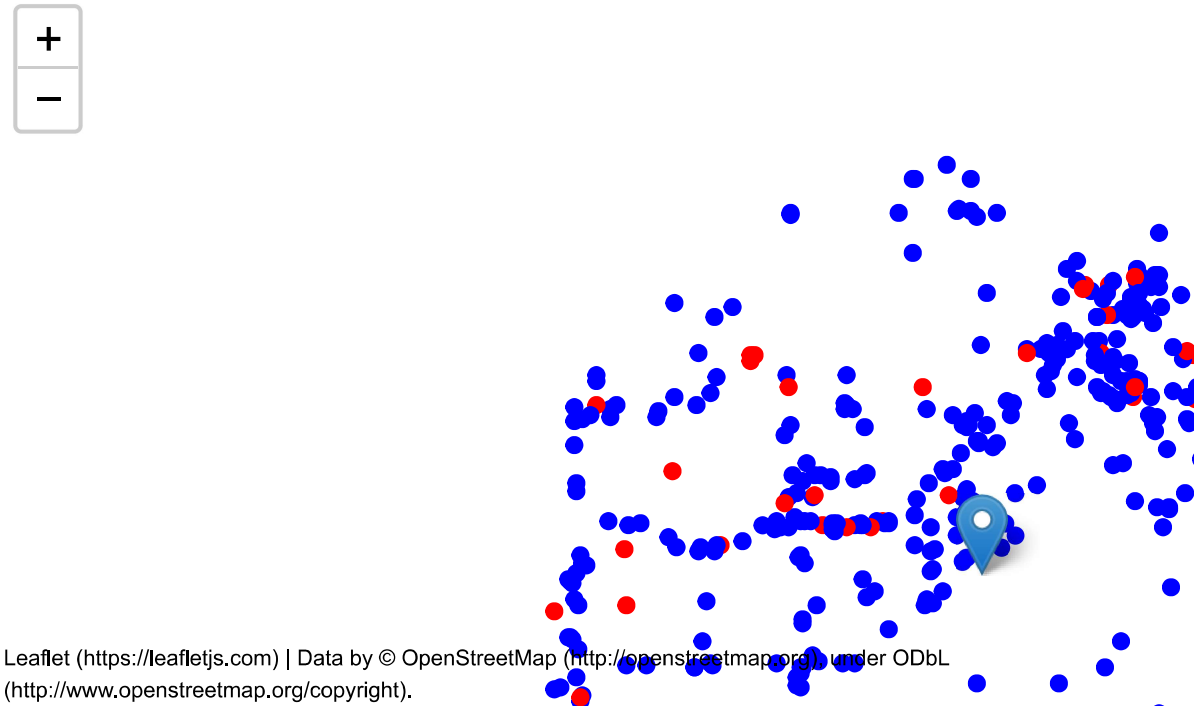
In [58]:
```
print('Restaurants around location')
print('---------------------------')
for i in range(100, 110):
    rs = location_restaurants[i][:8]
    names = ', '.join([r[1] for r in rs])
    print('Restaurants around location {}: {}'.format(i+1, names))
```

```
Restaurants around location
---------------------------
Restaurants around location 101:
Restaurants around location 102: Mikeys Supreme Burgers And More
Restaurants around location 103: Mikeys Supreme Burgers And More
Restaurants around location 104: The Cajun Stop
Restaurants around location 105: Natto Japanese & Chinese
Restaurants around location 106:
Restaurants around location 107: Zydeco Louisiana Diner, The Mediterranean Gr
ill
Restaurants around location 108: Benihana, The 43rd, Brazos Restaurant & Bar,
Bouray's Burrito Bar and Vietnamese Restaurant, Edgar's Hermano, Jefferson Ca
fe
Restaurants around location 109: Trofi Restaurant, Brazos Restaurant & Bar, B
ouray's Burrito Bar and Vietnamese Restaurant, Rosalies, Edgar's Hermano
Restaurants around location 110: South Bank Seafood Bar
```

Let's now see all the collected restaurants in our area of interest on map, and let's also show Italian restaurants in different color.

In [59]:
```
map_houston = folium.Map(location=houston_center, zoom_start=13)
folium.Marker(houston_center, popup='Midtown').add_to(map_houston)
for res in restaurants.values():
    lat = res[2]; lon = res[3]
    is_italian = res[6]
    color = 'red' if is_italian else 'blue'
    folium.CircleMarker([lat, lon], radius=3, color=color, fill=True, fill_col
or=color, fill_opacity=1).add_to(map_houston)
map_houston
```

Out[59]:



Looking good. So now we have all the restaurants in area within few kilometers from Midtown, and we know which ones are Italian restaurants! We also know which restaurants exactly are in vicinity of every neighborhood candidate center.

This concludes the data gathering phase - we're now ready to use this data for analysis to produce the report on optimal locations for a new Italian restaurant!

Let's perform some basic explanatory data analysis and derive some additional info from our raw data. First let's count the **number of restaurants in every area candidate**:

Analysis

In [60]:
```
location_restaurants_count = [len(res) for res in location_restaurants]

df_locations['Restaurants in area'] = location_restaurants_count

print('Average number of restaurants in every area with radius=300m:', np.arra
y(location_restaurants_count).mean())

df_locations.head(10)
```

```
Average number of restaurants in every area with radius=300m: 2.6483516483516
483
```

Out[60]:

	Address	Latitude	Longitude	X	Y	Distance from center	Restaura in a
0	215 Velasco St, Houston, TX 77003	29.751059	-95.341556	-6.762495e+06	1.347141e+07	5992.495307	2
1	2910 Commerce St, Houston, TX 77003	29.753591	-95.343702	-6.761895e+06	1.347141e+07	5840.376700	0
2	17 N St Charles St, Houston, TX 77003	29.756123	-95.345849	-6.761295e+06	1.347141e+07	5747.173218	3
3	2217 Canal St, Houston, TX 77003	29.758656	-95.347997	-6.760695e+06	1.347141e+07	5715.767665	2
4	Eastex Fwy, Houston, TX	29.761188	-95.350144	-6.760095e+06	1.347141e+07	5747.173218	2
5	511 McKee St, Houston, TX 77002	29.763721	-95.352292	-6.759495e+06	1.347141e+07	5840.376700	0
6	720 Walnut St, Houston, TX 77002	29.766254	-95.354440	-6.758895e+06	1.347141e+07	5992.495307	2
7	77001 Railroad, Houston, TX 77003	29.745639	-95.340849	-6.763395e+06	1.347193e+07	5855.766389	1
8	9363 Roberts St, Houston, TX 77003	29.748171	-95.342995	-6.762795e+06	1.347193e+07	5604.462508	0

	Address	Latitude	Longitude	X	Y	Distance from center	Restaura in a
9	3000 Hawkins St, Houston, TX 77003	29.750703	-95.345142	-6.762195e+06	1.347193e+07	5408.326913	1

OK, now let's calculate the **distance to nearest Italian restaurant from every area candidate center** (not only those within 300m - we want distance to closest one, regardless of how distant it is).

In [61]:
```
distances_to_italian_restaurant = []

for area_x, area_y in zip(xs, ys):
    min_distance = 10000
    for res in italian_restaurants.values():
        res_x = res[7]
        res_y = res[8]
        d = calc_xy_distance(area_x, area_y, res_x, res_y)
        if d<min_distance:
            min_distance = d
    distances_to_italian_restaurant.append(min_distance)

df_locations['Distance to Italian restaurant'] = distances_to_italian_restaura
nt
```

In [62]:
```
df_locations.head(10)
```

Out[62]:

	Address	Latitude	Longitude	X	Y	Distance from center	Restaura in a
0	215 Velasco St, Houston, TX 77003	29.751059	-95.341556	-6.762495e+06	1.347141e+07	5992.495307	2
1	2910 Commerce St, Houston, TX 77003	29.753591	-95.343702	-6.761895e+06	1.347141e+07	5840.376700	0
2	17 N St Charles St, Houston, TX 77003	29.756123	-95.345849	-6.761295e+06	1.347141e+07	5747.173218	3
3	2217 Canal St, Houston, TX 77003	29.758656	-95.347997	-6.760695e+06	1.347141e+07	5715.767665	2
4	Eastex Fwy, Houston, TX	29.761188	-95.350144	-6.760095e+06	1.347141e+07	5747.173218	2
5	511 McKee St, Houston, TX 77002	29.763721	-95.352292	-6.759495e+06	1.347141e+07	5840.376700	0
6	720 Walnut St, Houston, TX 77002	29.766254	-95.354440	-6.758895e+06	1.347141e+07	5992.495307	2
7	77001 Railroad, Houston, TX 77003	29.745639	-95.340849	-6.763395e+06	1.347193e+07	5855.766389	1
8	9363 Roberts St, Houston, TX 77003	29.748171	-95.342995	-6.762795e+06	1.347193e+07	5604.462508	0

	Address	Latitude	Longitude	X	Y	Distance from center	Restaura in a
9	3000 Hawkins St, Houston, TX 77003	29.750703	-95.345142	-6.762195e+06	1.347193e+07	5408.326913	1

In [63]:
```
print('Average distance to closest Italian restaurant from each area center:',
df_locations['Distance to Italian restaurant'].mean())
```

```
Average distance to closest Italian restaurant from each area center: 2090.07
2793071814
```

OK, so **on average Italian restaurant can be found within ~2100m or 1.3 miles** from every area center candidate. That's fairly close, so we need to filter our areas carefully!

Results and Discussion

Our analysis shows that although there is a great number of restaurants in Houston (~2100 in our initial area of interest which was 12x12km around Midtown), there are pockets of low restaurant density fairly close to city center.

Conclusion

Purpose of this project was to identify Houston areas close to center with low number of restaurants (particularly Italian restaurants) in order to aid stakeholders in narrowing down the search for optimal location for a new Italian restaurant.

Final decission on optimal restaurant location will be made by stakeholders based on specific characteristics of neighborhoods and locations in every recommended zone, taking into consideration additional factors like attractiveness of each location (proximity to park or water), levels of noise / proximity to major roads, real estate availability, prices, social and economic dynamics of every neighborhood etc.