Capstone: Italian Options in Houston, TX

Daniel Eads

Background & Problem

Company XYZ will be sending sales representatives to a convention in the City of Houston, Texas in the Southwestern United States with a specific senior financial executive in mind. They know the executive loves Italian Food, but this is XYZ's first trip to Texas.

The Sales team of company XYZ wants to make sure they select an Italian Restaurant and the legal department wants to be sure crime statistics impact that decision.

Data Acquisition

The City of Houston OpenData Website

This resource provides a GeoJSON for the individual areas of operation of the various departments of the Houston Police Departments. In this case divided up by "beats". The use of these map boundaries is significant as publicly available crime data for Houston is only categorized by beats.

Houston Police Department Crime Data

This resource is only available as an xls file and ,again, only available by beat. The file comprised over 60,000 crimes between 01 Jan 2020 and 04 25 2020 and is the only official release for crime data for Houston.

The Foursquare API

Unfortunately, my company is uncomfortable with the idea of use-limits and I will be limited to sandbox access to foursquare. In order to establish restaurants of interest, I will explore Italian restaurants in areas deemed safest for that purpose. This will reduce the number of calls needed and satisfy company requirements.

Workflow

Initial Manipulations

We can see below that there are no coordinates in the crimes DataFrame, but it is populated by column beats. Due to the rate limits of Foursquare we need to prepare the DataFrame by removing unnecessary columns, and narrowing the categories we will use for analysis as this is too much to geocode without additional cost.

```
crimes_shape = crimes.shape
beats_shape = beats.shape
print("Shape of crimes Dataframe = {}\nShape of beats GeoDataframe = {}".format(crimes_shape, beats_shape))
```

```
Shape of crimes Dataframe = (65499, 7)
Shape of beats GeoDataframe = (118, 10)
```

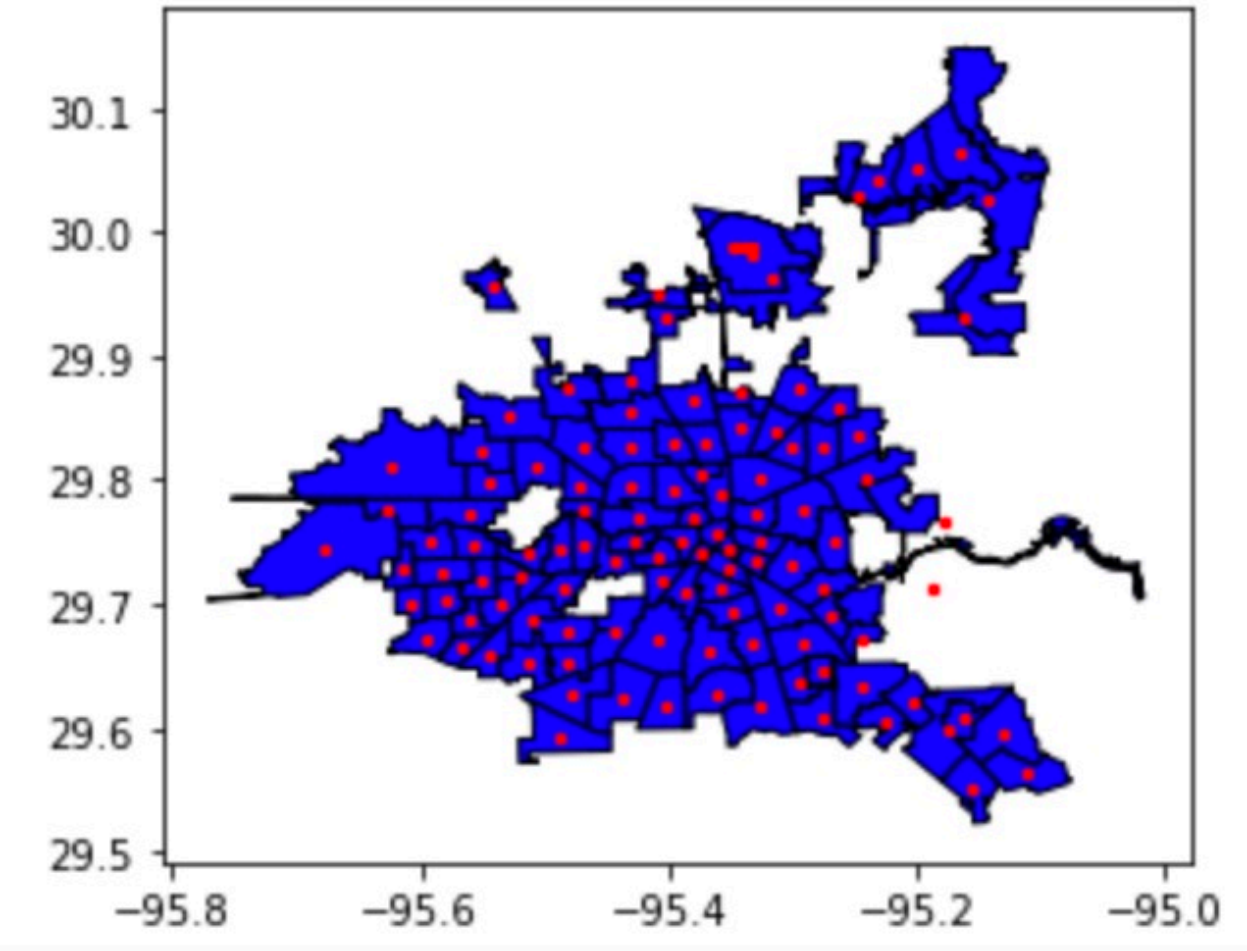
	Incident	Occurrence Date	Occurrence Hour	NIBRS Class	NIBRSDescription	Offense Count	Beat	Premise	Block Range	StreetName	Street Type	Suffix	City	ZIP Code
0	8220	2020-01-01	0	23G	Theft of motor vehicle parts or accessory	1	8C50	Residence, Home (Includes Apartment)	9311	BELLA PINE	CT	NaN	HOUSTON	77078
1	18920	2020-01-01	0	13A	Aggravated Assault	1	17E30	Residence, Home (Includes Apartment)	8701	GUSTINE	LN	NaN	HOUSTON	77031

We can use the focus of restaurants as that is what we are most interested in. Furthermore we will limit our datetime range to make the data more manageable. To do this we will use the pandas loc method on the date and premise columns of the crimes dataframe,

```
crimes['Date'] = pd.to_datetime(crimes['Date'])
start_date = '03-01-2020'
end_date = '04-25-2020'
mask = (crimes['Date'] > start_date) & (crimes['Date']  <= end_date)
crimes = crimes.loc[mask]
crimes = crimes.loc[crimes['Premise'] == "Restaurant"]
crimes.reset_index(drop=True)
crimes.head(2)
```

Test Maps and Merge

I made a test map to ensure my police beats GeoJSON works properly.




```
# Create and test to make sure the data works
beats_p = beats.copy()
beats_p['geometry'] = beats_p['geometry'].centroid
base = beats.plot(color='blue', edgecolor='black')
beats_p.plot(ax=base, marker='o', color='red', markersize=5);
beats_p.head()
```

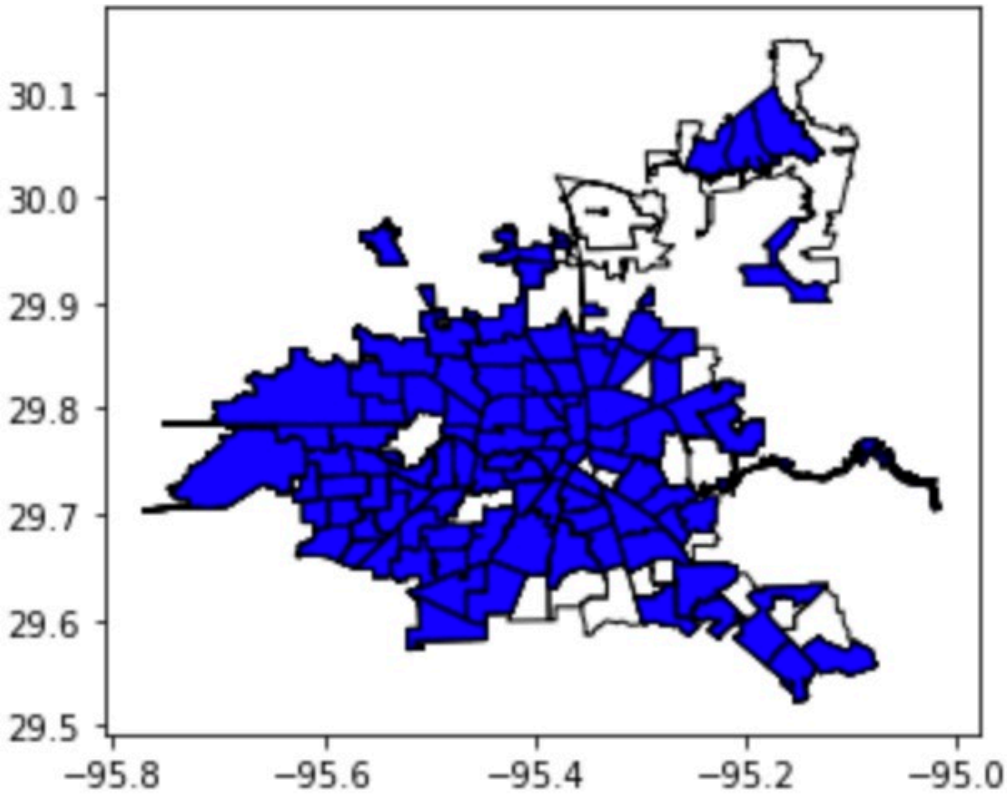
I format my crime data so as to make it easier to interpret and use a merge so the crime data has geographic polygons. I merge on right so we keep the GeoDataFrame as opposed to the Dataframe. The merge didn't return any errors, but head wouldn't tell us what the conversion did. "isinstance" checks if a variable is what we think it is.

```
# Use the polygons of Geodataframe beats to assign geographic values to our reastaurant crime DataFrame, crimes.
# Establish common attribute
beats = beats.rename(columns={'Beats': 'Beat'})
# Merge on right with the GeoDataFrame on the left side of the argument so we keep the GeoDataFrame as opposed to the Dataframe.
cri_geo = beats.merge(crimes, how="right", on='Beat')
# The merge didn't return any errors, but head wouldn't tell us what the conversion did. "isinstance" checks if a variable is what we think it is.
isinstance(cri_geo, gpd.GeoDataFrame)
```

```
)]: True
```

Geometry Types and Comparison Overlay

A test map of the restaurant crime data shows multiple police beats with no restaurant related crime in 2020. Additional manipulation is required to identify these beats based on the map data.



1. Geopandas geometry types must be formatted as MultiPolygon.
2. Empty geometries must be dropped.
3. The Geopandas Overlay method is used to identify polygons with no restaurant crime data.

Italian Food, Spatial Comparison, and Final Choice

I called the Foursquare API using the coordinates of Houston, TX. Calls are made to the Foursquare API to return results for Italian as long as they fit into the food category in a radius of 100000 meters.

```
url2 = 'https://api.foursquare.com/v2/venues/search?client_id={}&client_secret={}&ll={},{}&v={}&radius={}&query={}&limit={}&categoryId={}'.format(
    CLIENT_ID, CLIENT_SECRET, latitude, longitude, VERSION, radius, search_query, LIMIT, VENUE_CAT)
```

I then create a GeoDataFrame out of the Foursquare venues data so it can be incorporated into a map. Using Geopandas methods unary join and contains, only one restaurant met the criteria outlined at the beginning, Carrabbas Italian Grill.