

IBM Coursera Capstone

Introduction

Moving to work and live in a new city, one often wants to find a place that would correspond to the usual corner of life in the current city. I mean urban infrastructure and services.

To solve this problem, you can use Foursquare data by dividing the city you know into clusters and applying this clustering to the city you are going to move to.

Thus, you can get an idea of the most suitable areas for you.

I will solve this problem using the example of two capitals of my native country of Russia: Moscow and St. Petersburg. I live in Moscow and will try to find areas that suit my preferences in St. Petersburg.

```
In [339]: # pip install lxml
```

```
In [3]: import pandas as pd
import numpy as np
import re
```

```
In [4]: import requests # library to handle requests
from pandas.io.json import json_normalize # tranform JSON file into
a pandas dataframe
```

Data

Make sure that you provide adequate explanation and discussion, with examples, of the data that you will be using, even if it is only Foursquare location data.

Start with reading Moscow Neighbourhood names from wikipedia.

```
In [5]: msk= pd.read_html('https://ru.wikipedia.org/wiki/%D0%A1%D0%BF%D0%B8%
D1%81%D0%BE%D0%BA_%D1%80%D0%B0%D0%B9%D0%BE%D0%BD%D0%BE%D0%B2_%D0%B8
_%D0%BF%D0%BE%D1%81%D0%B5%D0%BB%D0%B5%D0%BD%D0%B8%D0%B9_%D0%9C%D0%B
E%D1%81%D0%BA%D0%B2%D1%8B')[0]
msk.drop(msk.columns[[0,1,2,6,7,8,9,10]], axis=1, inplace=True)
msk.columns = ['Borough', 'Neighbourhood', 'District']
```

```
In [68]: msk.head(3)
```

```
Out [68]:
```

	Borough	Neighbourhood	Distict	lat	lon
0	Академический	Академический	ЮЗАО	55.692091	37.589259
1	Алексеевский	Алексеевский	СВАО	55.821456	37.643961
2	Алтуфьевский	Алтуфьевский	СВАО	55.880255	37.581635

Then read Boroughs and Neiborhoods of Saint-Peterburg.

```

In [6]: spb_neig_raw= pd.read_html('https://ru.wikipedia.org/wiki/%D0%9A%D0%B0%D1%82%D0%B5%D0%B3%D0%BE%D1%80%D0%B8%D1%8F:%D0%9C%D1%83%D0%BD%D0%B8%D1%86%D0%B8%D0%BF%D0%B0%D0%BB%D1%8C%D0%BD%D1%8B%D0%B5_%D0%BE%D0%B1%D1%80%D0%B0%D0%B7%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F_%D0%A1%D0%B0%D0%BD%D0%BA%D1%82-%D0%9F%D0%B5%D1%82%D0%B5%D1%80%D0%B1%D1%83%D1%80%D0%B3%D0%B0')[1][1][0]
spb_neig= spb_neig_raw
spb_neig= re.sub('№ ', '№', spb_neig)
spb_neig= re.sub('[1 3 2 4 5 7 81234567890№]', '', spb_neig)
spb_neig= re.sub('округ', '', spb_neig)
spb_neig= re.sub('остров', '', spb_neig)
spb_neig= re.sub('речка', '', spb_neig)
spb_neig= re.sub('аэродром', '', spb_neig)
spb_neig= re.sub('застава', '', spb_neig)
spb_neig= re.sub('Остров', '', spb_neig)
spb_neig= re.sub('Озеро', '', spb_neig)
spb_neig= re.sub('меридиан', '', spb_neig)
spb_neig= re.sub('ворота', '', spb_neig)

spb_neig= re.sub(' ', ' ', spb_neig)
s = pd.Series(spb_neig)
spb_neig= s.str.split(expand=True).transpose().rename(columns={0: 'Neighbourhood'}, errors="raise")
spb_bor = pd.read_html('https://ru.wikipedia.org/wiki/%D0%90%D0%B4%D0%BC%D0%B8%D0%BD%D0%B8%D1%81%D1%82%D1%80%D0%B0%D1%82%D0%B8%D0%B2%D0%BD%D0%BE-%D1%82%D0%B5%D1%80%D1%80%D0%B8%D1%82%D0%BE%D1%80%D0%B8%D0%B0%D0%BB%D1%8C%D0%BD%D0%BE%D0%B5_%D0%B4%D0%B5%D0%BB%D0%B5%D0%BD%D0%B8%D0%B5_%D0%A1%D0%B0%D0%BD%D0%BA%D1%82-%D0%9F%D0%B5%D1%82%D0%B5%D1%80%D0%B1%D1%83%D1%80%D0%B3%D0%B0')[0]
spb_bor.drop(spb_bor.columns[[0,2,3,4]], axis=1, inplace=True)
spb_bor.rename(columns={"район": "Borough"}, inplace=True)

spb_neig_raw

```

```

Out[6]: 'Автово5 Адмиралтейский округ1 Академическое4 Аптекарский остров13 Балканский17 Большая Охта7 Васильевский2 Введенский13 Владимирский округ18 Волковское17 Гавань2 Гагаринское11 Георгиевский17 Горелово 8 Гражданка4 Дачное5 Дворцовый округ18 Екатерингофский1 Звёздное11 Ивановский12 Измайловское1 Княжево5 Коломна1 Коломяги15 Комендантский аэродром15 Константиновское8 Красненькая речка5 Кронверкское13 Купчино17 Ланское15 Лахта-Ольгино15 Лиговка-Ямская18 Литейный округ18 Малая Охта7 Морские ворота5 Морской2 Московская застава11 Нарвский округ5 Народный12 Невская застава12 Невский округ12 Новоизмайловское11 Обуховский12 Озеро Долгое15 Оккервиль12 округ Петровский13 Остров Декабристов2 Пискаревка4 Полюстрово7 Пороховые7 Посадский13 Правобережный12 Прометей4 Пулковский меридиан11 Ржевка7 Рыбацкое12 Сампсониевское3 Светлановское3 Северный4 Семёновский1 Сенной округ1 Сергиевское3 Смольнинское18 Сосновая Поляна8 Сосновское3 Ульянка5 Урицк8 Финляндский округ4 Чкаловское13 Шувалово-Озерки3 Юго-Запад8 Южно-Приморский8 Юнтолово15 № 72 № 153 № 214 № 5412 № 6515 № 7217 № 7517 № 7818'

```

```
In [7]: spb_neig.head(3)
```

```
Out[7]:
```

	Neighbourhood
0	Автово
1	Адмиралтейский
2	Академическое

Let's create a function that allow us to get Neighbourhood coordinates based on a string with the name.

```
In [7]: def getCoordByRegionName(regionName):
        url = "http://nominatim.openstreetmap.org/search?" + 'q=' + regionName + '&polygon_geojson=1&format=jsonv2'

        results = requests.get(url).json()
        # print(results)
        r= pd.DataFrame.from_dict( {'lat' : [None] , 'lon' : [None] })

        try:
            r= pd.DataFrame([[results[0]['lat'], results[0]['lon']]])
        except IndexError:
            print("Oops! That was an error with: "+regionName)

        return( r )
#getCoordByRegionName('округ Клёновское, поселение, Москва')
```

Then get coordinates of Moscow and Saint-Peterburg Neighbourhood.

```
In [8]: msk_c = pd.DataFrame.from_dict( {'lat' :[] , 'lon' :[]})

for index, x in msk.iterrows():
    coord = getCoordByRegionName('округ ' + x['Neighbourhood'] + ' ,
Москва')
    msk_c= msk_c.append({'lat' :coord.iloc[0][0] , 'lon' : coord.iloc
[0][1]} , ignore_index=True)
```

Oops! That was an error with: округ Новокосино , Москва
 Oops! That was an error with: округ Вороновское, поселение , Моск
 ва
 Oops! That was an error with: округ Клёновское, поселение , Москв
 а
 Oops! That was an error with: округ Кокошкино, поселение , Москва
 Oops! That was an error with: округ Краснопахорское, поселение ,
 Москва
 Oops! That was an error with: округ Михайлово-Ярцевское, поселени
 е , Москва
 Oops! That was an error with: округ Московский, поселение , Москв
 а
 Oops! That was an error with: округ Мосрентген, поселение , Москв
 а
 Oops! That was an error with: округ Первомайское, поселение , Мос
 ква
 Oops! That was an error with: округ Роговское, поселение , Москва
 Oops! That was an error with: округ Рязановское, поселение , Моск
 ва
 Oops! That was an error with: округ Троицк, городской округ , Мос
 ква
 Oops! That was an error with: округ Щербинка, городской округ , М
 осква

```
In [9]: msk= msk.join(msk_c)
msk= msk[msk.lat.notnull()]
```

```
In [10]: msk['lat']= msk['lat'].astype(float)
msk['lon']= msk['lon'].astype(float)
msk.head(3)
```

Out[10]:

	Borough	Neighbourhood	Distict	lat	lon
0	Академический	Академический	ЮЗАО	55.692091	37.589259
1	Алексеевский	Алексеевский	СВАО	55.821456	37.643961
2	Алтуфьевский	Алтуфьевский	СВАО	55.880255	37.581635

```
In [11]: spb_c= pd.DataFrame.from_dict( {'lat' :[] , 'lon' :[]})

for index, x in spb_neig.iterrows():
    coord = getCoordByRegionName('округ ' + x['Neighbourhood'] + ' ,
Санкт-Петербург')
    spb_c= spb_c.append({'lat' :coord.iloc[0][0] , 'lon' : coord.iloc
[0][1]} , ignore_index=True)
```

Oops! That was an error with: округ Аптекарский , Санкт-Петербург
 Oops! That was an error with: округ Красненькая , Санкт-Петербург
 Oops! That was an error with: округ Морские , Санкт-Петербург
 Oops! That was an error with: округ Невская , Санкт-Петербург
 Oops! That was an error with: округ Декабристов , Санкт-Петербург
 Oops! That was an error with: округ Сенной , Санкт-Петербург
 Oops! That was an error with: округ Сосновая , Санкт-Петербург
 Oops! That was an error with: округ Поляна , Санкт-Петербург

```
In [12]: spb_neig= spb_neig.join(spb_c)
spb_neig= spb_neig[spb_neig.lat.notnull()]
```

```
In [13]: spb_neig['lat']= spb_neig['lat'].astype(float)
spb_neig['lon']= spb_neig['lon'].astype(float)
spb_neig.head(3)
```

Out[13]:

	Neighbourhood	lat	lon
0	Автово	59.858637	30.278818
1	Адмиралтейский	59.931054	30.296798
2	Академическое	60.011526	30.394661

Let's visualise results on the folium maps

```
In [14]: # Matplotlib and associated plotting modules
#!conda install -c conda-forge matplotlib --yes
import matplotlib.cm as cm
import matplotlib.colors as colors
```

```
In [15]: #!conda install -c conda-forge folium=0.5.0 --yes
import folium # map rendering library
```

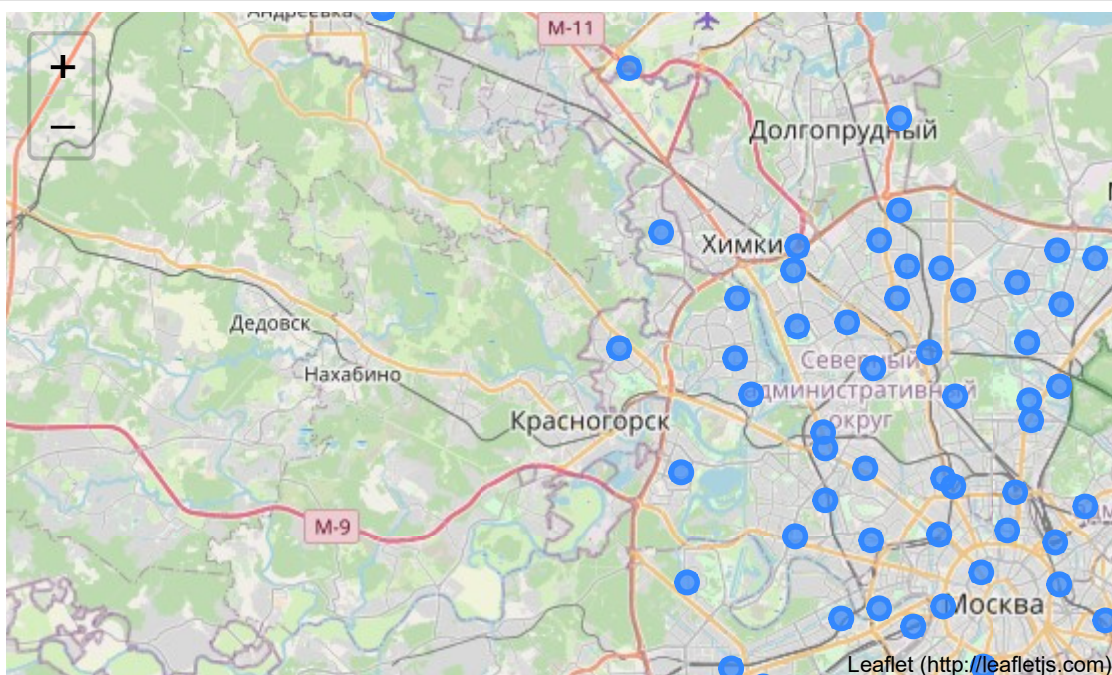
```
In [16]: msk_center= getCoordByRegionName('Москва')
spb_center= getCoordByRegionName('Санкт-Петербург')
```

```
In [17]: map_moscow = folium.Map(location=msk_center.astype(float).values.tolist()[0], zoom_start=10)

# add markers to the map
markers_colors = []
for lat, lon, poi in zip(msk['lat'], msk['lon'], msk['Neighbourhood']):
    label = folium.Popup(str(poi), parse_html=True)
    folium.CircleMarker(
        [lat, lon],
        radius=5,
        popup=label,
        #color=1,
        fill=True,
        #fill_color=1,
        fill_opacity=0.7).add_to(map_moscow)

map_moscow
```

Out[17]:

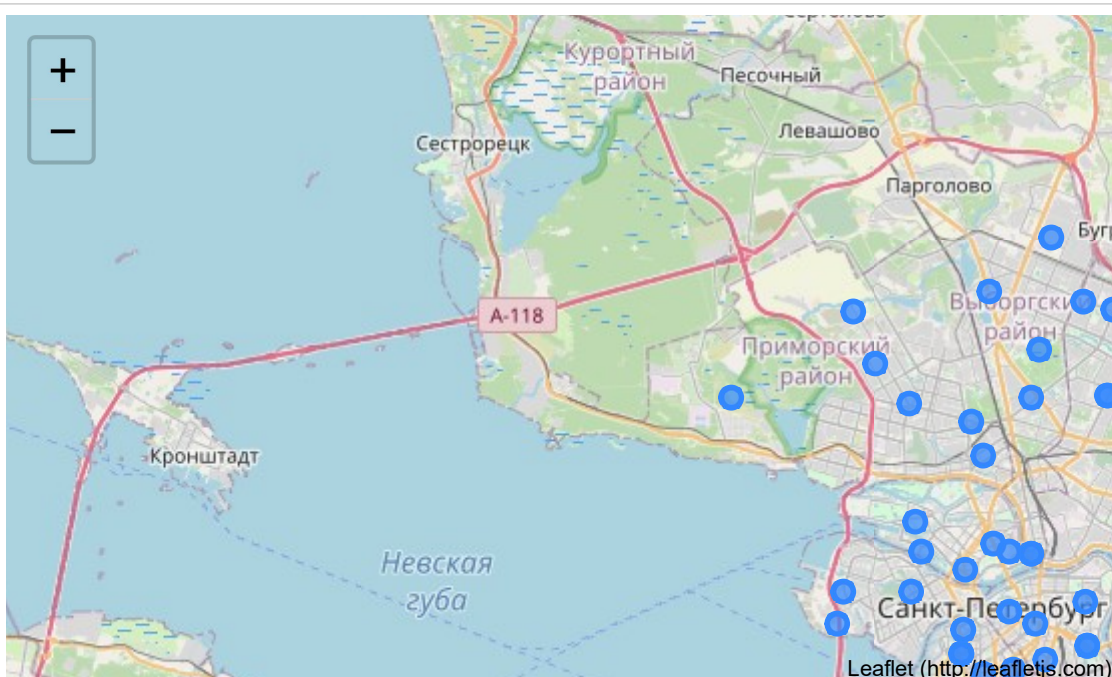



```
In [18]: map_spb = folium.Map(location=spb_center.astype(float).values.tolist()
[0], zoom_start=10)

markers_colors = []
for lat, lon, poi in zip(spb_neig['lat'], spb_neig['lon'], spb_neig
['Neighbourhood']):
    label = folium.Popup(str(poi), parse_html=True)
    folium.CircleMarker(
        [lat, lon],
        radius=5,
        popup=label,
        #color=1,
        fill=True,
        #fill_color=1,
        fill_opacity=0.7).add_to(map_spb)

map_spb
```

Out[18]:



Next step is to load data from foursquare.

Let's create a function that allow as get information about Moscow and St.Petersburg neighborhoods venues.

```
In [19]: CLIENT_ID = 'ZMTK2WDYCDINQIHLP5BVTN03U1NY41AGZVIJPGDZSGPDTPX' # you
r Foursquare ID
CLIENT_SECRET = '4PNTRNEXTCD53E3C0ZMWFPQP2WAU50QGQ3KF2CMUSW4COSRT' #
your Foursquare Secret
VERSION = '20180605' # Foursquare API version
LIMIT = 100 # A default Foursquare API limit value

print('Your credentails:')
print('CLIENT_ID: ' + CLIENT_ID)
print('CLIENT_SECRET: ' + CLIENT_SECRET)
```

```
Your credentails:
CLIENT_ID: ZMTK2WDYCDINQIHLP5BVTN03U1NY41AGZVIJPGDZSGPDTPX
CLIENT_SECRET: 4PNTRNEXTCD53E3C0ZMWFPQP2WAU50QGQ3KF2CMUSW4COSRT
```



```

In [20]: def getNearbyVenues(names, latitudes, longitudes, radius=1000):

    venues_list=[]
    for name, lat, lng in zip(names, latitudes, longitudes):
        print(name)

        # create the API request URL
        url = 'https://api.foursquare.com/v2/venues/explore?&client_
id={}&client_secret={}&v={}&ll={},{}&radius={}&limit={}'.format(
            CLIENT_ID,
            CLIENT_SECRET,
            VERSION,
            lat,
            lng,
            radius,
            LIMIT)

        # make the GET request
        results = requests.get(url).json()["response"]['groups
'] [0] ['items']

        # return only relevant information for each nearby venue
        venues_list.append([ (
            name,
            lat,
            lng,
            v['venue']['name'],
            v['venue']['location']['lat'],
            v['venue']['location']['lng'],
            v['venue']['categories'][0]['name']) for v in results])

    nearby_venues = pd.DataFrame([item for venue_list in venues_list
for item in venue_list])
    nearby_venues.columns = ['Neighborhood',
                            'Neighborhood Latitude',
                            'Neighborhood Longitude',
                            'Venue',
                            'Venue Latitude',
                            'Venue Longitude',
                            'Venue Category']

    return(nearby_venues)

```

Now write the code to run the above function on each neighborhood and create a new dataframe for Moscow and St.Peterburg.

```
In [22]: msk_venues = getNearbyVenues(names=msk['Neighbourhood'],  
                                         latitudes=msk['lat'],  
                                         longitudes=msk['lon']  
                                         )
```

Академический
Алексеевский
Алтуфьевский
Арбат
Аэропорт
Бабушкинский
Басманный
Беговой
Бескудниковский
Бибирево
Бирюлёво Восточное
Бирюлёво Западное
Богородское
Братеево
Бутырский
Вешняки
Внуково
Войковский
Восточное Дегунино
Восточное Измайлово
Восточный
Выхино-Жулебино
Гагаринский
Головинский
Гольяново
Даниловский
Дмитровский
Донской
Дорогомилово
Замоскворечье
Западное Дегунино
Зюзино
Зябликово
Ивановское
Измайлово
Капотня
Коньково
Коптево
Косино-Ухтомский
Котловка
Красносельский
Крылатское
Крюково
Кузьминки
Кунцево
Куркино
Левобережный
Лефортово
Лианозово
Ломоносовский
Лосиноостровский
Люблино
Марфино
Марьино Роща
Марьино
Матушкино
Метрогородок
Мещанский
Митино

Можайский
Молжаниновский
Москворечье-Сабурово
Нагатино-Садовники
Нагатинский Затон
Нагорный
Некрасовка
Нижегородский
Новогиреево
Новокосино
Ново-Переделкино
Обручевский
Орехово-Борисово Северное
Орехово-Борисово Южное
Останкинский
Отрадное
Очаково-Матвеевское
Перово
Печатники
Покровское-Стрешнево
Преображенское
Пресненский
Прспект Вернадского
Раменки
Ростокино
Рязанский
Савёлки
Савёловский
Свиблово
Северное Бутово
Северное Измайлово
Северное Медведково
Северное Тушино
Северный
Силино
Сокол
Соколиная Гора
Сокольники
Солнцево
Старое Крюково
Строгино
Таганский
Тверской
Текстильщики
Тёплый Стан
Тимирязевский
Тропарёво-Никулино
Филёвский Парк
Фили-Давыдково
Хамовники
Ховрино
Хорошёво-Мнёвники
Хорошёвский
Царицыно
Черёмушки
Чертаново Северное
Чертаново Центральное
Чертаново Южное
Щукино
Южное Бутово

Южное Медведково
Южное Тушино
Южнопортовый
Якиманка
Ярославский
Ясенево
Внуковское, поселение
Воскресенское, поселение
Десёновское, поселение
Киевский, поселение
Марушкинское, поселение
Новофёдоровское, поселение
Сосенское, поселение
Филимонковское, поселение

```
In [25]: spb_venues = getNearbyVenues(names=spb_neig['Neighbourhood'],  
                                       latitudes=spb_neig['lat'],  
                                       longitudes=spb_neig['lon']  
                                       )
```


Автово
Адмиралтейский
Академическое
Балканский
Большая
Охта
Васильевский
Введенский
Владимирский
Волковское
Гавань
Гагаринское
Георгиевский
Горелово
Гражданка
Дачное
Дворцовый
Екатерингофский
Звёздное
Ивановский
Измайловское
Княжево
Коломна
Коломяги
Комендантский
Константиновское
Кронверкское
Купчино
Ланское
Лахта-Ольгино
Лиговка-Ямская
Литейный
Малая
Охта
Морской
Московская
Нарвский
Народный
Невский
Новоизмайловское
Обуховский
Долгое
Оккервиль
Петровский
Пискарёвка
Полюстрово
Пороховые
Посадский
Правобережный
Прометей
Пулковский
Ржевка
Рыбацкое
Сампсониевское
Светлановское
Северный
Семёновский
Сергиевское
Смольнинское

Сосновское
Ульянка
Урицк
Финляндский
Чкаловское
Шувалово-Озерки
Юго-Запад
Южно-Приморский
Юнтолово

Let's look how is data looks like.

In [23]: msk_venues

Out [23]:

	Neighborhood	Neighborhood Latitude	Neighborhood Longitude	Venue	Venue Latitude	Venue Longitude
0	Академический	55.692091	37.589259	Вкусвилл	55.691800	37.588235
1	Академический	55.692091	37.589259	Парк «Новые Черёмушки»	55.693547	37.589786
2	Академический	55.692091	37.589259	Spirit Fitness	55.690272	37.601362
3	Академический	55.692091	37.589259	Подружка	55.690292	37.601388
4	Академический	55.692091	37.589259	Cats & Dogs	55.689592	37.601016
...
6818	Филимонковское, поселение	55.586240	37.265358	Рейс 463 VKO-TJM	55.593191	37.258526
6819	Щаповское, поселение	55.352489	37.443953	Atlantic City	55.355982	37.450608
6820	Щаповское, поселение	55.352489	37.443953	Nikulino Village 	55.344541	37.443891
6821	Щаповское, поселение	55.352489	37.443953	Магазин "Никулинский"	55.344530	37.444056
6822	Щаповское, поселение	55.352489	37.443953	Sysoev Serega 25th Birthday Party	55.348515	37.456081

6823 rows × 7 columns

In [26]: spb_venues

Out[26]:

	Neighborhood	Neighborhood Latitude	Neighborhood Longitude	Venue	Venue Latitude	Venue Longitude
0	АВТОВО	59.858637	30.278818	Fitness One	59.853499	30.282053
1	АВТОВО	59.858637	30.278818	Кода	59.852220	30.283060
2	АВТОВО	59.858637	30.278818	Подружка	59.852164	30.273006
3	АВТОВО	59.858637	30.278818	Лэнд	59.852476	30.281625
4	АВТОВО	59.858637	30.278818	IMPERIAL SHOPS	59.852355	30.268517
...
4332	Юнтолово	60.022153	30.235951	Воздух	60.027077	30.224183
4333	Юнтолово	60.022153	30.235951	Остановка «Долгоозёрная ул.»	60.017072	30.248387
4334	Юнтолово	60.022153	30.235951	Зоомакет в Юбилейном	60.030244	30.235912
4335	Юнтолово	60.022153	30.235951	Продукты	60.030493	30.237980
4336	Юнтолово	60.022153	30.235951	Булочная Ф. Вольчека	60.030578	30.238352

4337 rows × 7 columns

So, we have insights in a data, next step is to analyse neighbourhoods and find out similar clusters in to cities.

Exploratory data analysis

Let's find out how many unique categories can be curated from all the returned venues

```
In [27]: print('There are {} uniques categories in Moscow.'.format(len(msk_venues['Venue Category'].unique())))
print('There are {} uniques categories in St.Peterburg.'.format(len(spb_venues['Venue Category'].unique())))
```

There are 369 uniques categories in Moscow.

There are 331 uniques categories in St.Peterburg.

Then find most frequent categories for each neighbourhoods

```
In [28]: # one hot encoding
msk_onehot = pd.get_dummies(msk_venues[['Venue Category']], prefix
=" ", prefix_sep=" ")
spb_onehot = pd.get_dummies(spb_venues[['Venue Category']], prefix
=" ", prefix_sep=" ")

# add neighborhood column back to dataframe
msk_onehot['Neighborhood'] = msk_venues['Neighborhood']
spb_onehot['Neighborhood'] = spb_venues['Neighborhood']

# move neighborhood column to the first column
fixed_columns = [msk_onehot.columns[-1]] + list(msk_onehot.columns
[:-1])
msk_onehot = msk_onehot[fixed_columns]
fixed_columns = [spb_onehot.columns[-1]] + list(spb_onehot.columns
[:-1])
spb_onehot = spb_onehot[fixed_columns]

msk_onehot.tail()
```

Out [28]:

	Neighborhood	ATM	Accessories Store	Adult Boutique	American Restaurant	Arcade	Arepa Restaurant	Gall
6818	Филимонковское, поселение	0	0	0	0	0	0	
6819	Щаповское, поселение	0	0	0	0	0	0	
6820	Щаповское, поселение	0	0	0	0	0	0	
6821	Щаповское, поселение	0	0	0	0	0	0	
6822	Щаповское, поселение	0	0	0	0	0	0	

5 rows × 370 columns

Next, let's group rows by neighborhood and by taking the mean of the frequency of occurrence of each category

```
In [29]: msk_grouped = msk_onehot.groupby('Neighborhood').mean().reset_index()
msk_grouped.head()
spb_grouped = spb_onehot.groupby('Neighborhood').mean().reset_index()
spb_grouped.head()
```

Out[29]:

	Neighborhood	ATM	Accessories Store	Adult Boutique	Advertising Agency	American Restaurant	Antique Shop	Aq
0	Автово	0.000000	0.0	0.0	0.0	0.0	0.0	
1	Адмиралтейский	0.000000	0.0	0.0	0.0	0.0	0.0	
2	Академическое	0.000000	0.0	0.0	0.0	0.0	0.0	
3	Балканский	0.011905	0.0	0.0	0.0	0.0	0.0	
4	Большая	0.000000	0.0	0.0	0.0	0.0	0.0	

5 rows × 332 columns

Let's get top most venues for each neighborhood into a pandas dataframe

```
In [30]: # First, let's write a function to sort the venues in descending order.
def return_most_common_venues(row, num_top_venues):
    row_categories = row.iloc[1:]
    row_categories_sorted = row_categories.sort_values(ascending=False)

    return row_categories_sorted.index.values[0:num_top_venues]
```

```
In [31]: # Now let's create the new dataframe and display the top 10 venues f
or each neighborhood.
num_top_venues = 10

indicators = ['st', 'nd', 'rd']

# create columns according to number of top venues
columns = ['Neighborhood']
for ind in np.arange(num_top_venues):
    try:
        columns.append('{}{} Most Common Venue'.format(ind+1, indica
tors[ind]))
    except:
        columns.append('{}th Most Common Venue'.format(ind+1))

# create a new dataframe for Moscow
msk_neighborhoods_venues_sorted = pd.DataFrame(columns=columns)
msk_neighborhoods_venues_sorted['Neighborhood'] = msk_grouped['Neigh
borhood']

for ind in np.arange(msk_grouped.shape[0]):
    msk_neighborhoods_venues_sorted.iloc[ind, 1:] = return_most_comm
on_venues(msk_grouped.iloc[ind, :], num_top_venues)

msk_neighborhoods_venues_sorted.head()
```

Out[31]:

	Neighborhood	1st Most Common Venue	2nd Most Common Venue	3rd Most Common Venue	4th Most Common Venue	5th Most Common Venue	6th Mo: Commo Venu
0	Академический	Bakery	Park	Coffee Shop	Gym / Fitness Center	Pharmacy	Supermark
1	Алексеевский	Pizza Place	Coffee Shop	Pet Store	Caucasian Restaurant	Park	Hot
2	Алтуфьевский	Supermarket	Convenience Store	Zoo Exhibit	Auto Workshop	Bus Stop	Burger Joi
3	Арбат	Coffee Shop	Hotel	Yoga Studio	Museum	Russian Restaurant	Flower Shc
4	Аэропорт	Cosmetics Shop	Coffee Shop	Café	Convenience Store	Pharmacy	Bakei

```
In [32]: # create a new dataframe for St.Peterburg
spb_neighborhoods_venues_sorted = pd.DataFrame(columns=columns)
spb_neighborhoods_venues_sorted['Neighborhood'] = spb_grouped['Neighborhood']

for ind in np.arange(spb_grouped.shape[0]):
    spb_neighborhoods_venues_sorted.iloc[ind, 1:] = return_most_common_venues(spb_grouped.iloc[ind, :], num_top_venues)

spb_neighborhoods_venues_sorted.head()
```

Out[32]:

	Neighborhood	1st Most Common Venue	2nd Most Common Venue	3rd Most Common Venue	4th Most Common Venue	5th Most Common Venue	6th Most Common Venue	7 C
0	Автово	Gym / Fitness Center	Furniture / Home Store	Pharmacy	Dance Studio	Auto Workshop	Restaurant	E
1	Адмиралтейский	Hotel	Restaurant	Bar	Park	Concert Hall	Opera House	Ca Re
2	Академическое	Bakery	Dessert Shop	Gym / Fitness Center	Playground	Bus Stop	Sporting Goods Shop	S
3	Балканский	Clothing Store	Mobile Phone Shop	Shopping Mall	Shawarma Place	Gym / Fitness Center	Electronics Store	Fa Re
4	Большая	Bakery	Park	Playground	Pharmacy	Coffee Shop	Pet Store	

Cluster Neighborhoods

Run k-means to cluster the Moscow neighborhoods into some number of clusters.

```
In [33]: # import k-means from clustering stage
from sklearn.cluster import KMeans
```

First of all let's leave the same venues that can be found in Moscow and St.Petersburg.

```
In [94]: same_names= list(set(list(spb_grouped.drop('Neighborhood', 1)) & set(msk_grouped.drop('Neighborhood', 1))))
len(same_names)
```

Out[94]: 280

And after that get the Moscow Clusters

```
In [90]: # set number of clusters
kclusters = 4

msk_grouped_clustering = msk_grouped[same_names]

# run k-means clustering
kmeans = KMeans(n_clusters=kclusters, random_state=0).fit(msk_grouped_clustering)

# add clustering labels
# msk_neighborhoods_venues_sorted.drop(columns=['Cluster Labels'], inplace=True)
msk_neighborhoods_venues_sorted.insert(0, 'Cluster Labels', kmeans.labels_)

unique, counts = np.unique(kmeans.labels_, return_counts=True)
print(unique, counts)

[0 1 2 3] [28  5 60 39]
```

```
In [91]: msk_neighborhoods_venues_sorted.head()
```

```
Out[91]:
```

	Cluster Labels	Neighborhood	1st Most Common Venue	2nd Most Common Venue	3rd Most Common Venue	4th Most Common Venue	5th Most Common Venue	
0	2	Академический	Bakery	Park	Coffee Shop	Gym / Fitness Center	Pharmacy	S
1	3	Алексеевский	Pizza Place	Coffee Shop	Pet Store	Caucasian Restaurant	Park	
2	0	Алтуфьевский	Supermarket	Convenience Store	Zoo Exhibit	Auto Workshop	Bus Stop	F
3	3	Арбат	Coffee Shop	Hotel	Yoga Studio	Museum	Russian Restaurant	F
4	2	Аэропорт	Cosmetics Shop	Coffee Shop	Café	Convenience Store	Pharmacy	

Let's visualize clusters


```
In [92]: msk_merged= pd.merge(msk.rename(columns = {'Neighbourhood': 'Neighborhood'}), msk_neighborhoods_venues_sorted, on="Neighborhood")
msk_merged.head(3)
```

Out[92]:

	Borough	Neighborhood	Distict	lat	lon	Cluster Labels	1st Most Common Venue	2nd Most Common Venue
0	Академический	Академический	ЮЗАО	55.692091	37.589259	2	Bakery	Coffee Shop
1	Алексеевский	Алексеевский	СВАО	55.821456	37.643961	3	Pizza Place	Coffee Shop
2	Алтуфьевский	Алтуфьевский	СВАО	55.880255	37.581635	0	Supermarket	Convenience Store

```

In [106]: # create map
map_msk_clusters = folium.Map(location=msk_center.astype(float).values.tolist()[0], zoom_start=10)

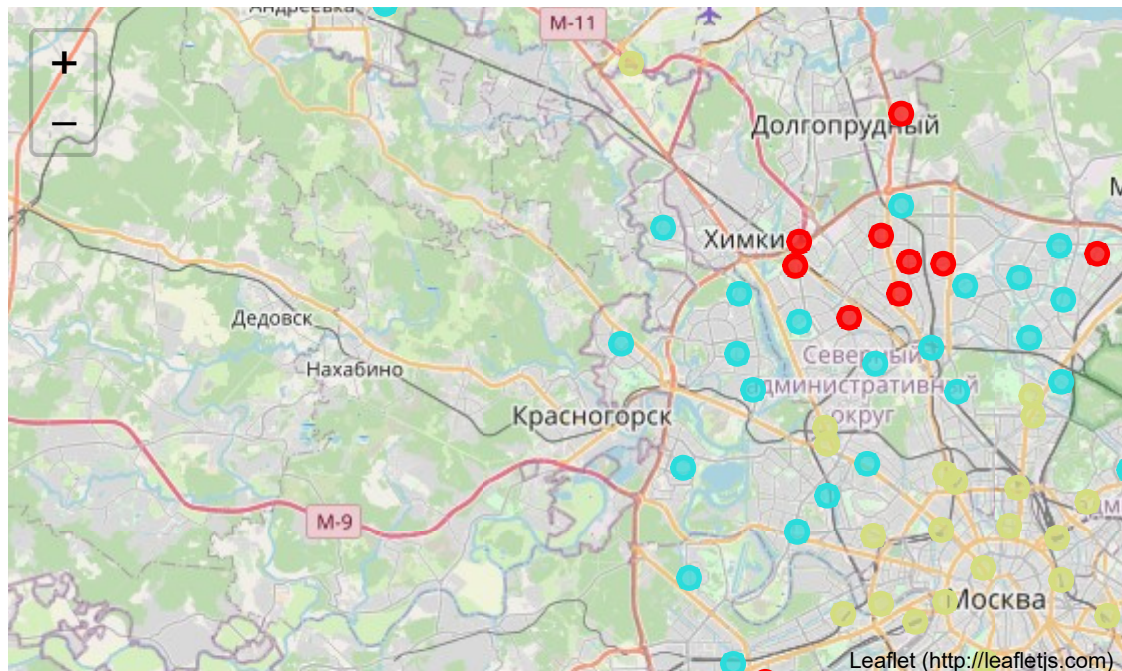
# set color scheme for the clusters
x = np.arange(kclusters)
ys = [i + x + (i*x)**2 for i in range(kclusters)]
colors_array = cm.rainbow(np.linspace(0, 1, len(ys)))
rainbow = [colors.rgb2hex(i) for i in colors_array]

# add markers to the map
markers_colors = []
for lat, lon, poi, cluster, v1, v2, v3 in zip(msk_merged['lat'], msk_merged['lon'], msk_merged['Neighborhood'], msk_merged['Cluster Labels'],
                                              msk_merged['1st Most Common Venue'], msk_merged['2nd Most Common Venue'], msk_merged['3rd Most Common Venue']):
    label = folium.Popup(poi + ' Cluster ' + str(cluster) + ' top venues: ' + str(v1) + ', ' + str(v2) + ', ' + str(v3) + ', ', parse_html=True)
    folium.CircleMarker(
        [lat, lon],
        radius=5,
        popup=label,
        color=rainbow[cluster-1],
        fill=True,
        fill_color=rainbow[cluster-1],
        fill_opacity=0.7).add_to(map_msk_clusters)

map_msk_clusters

```

Out[106]:



And than let's predict same clusters in St.Petersbug and plot it on the map.

```
In [98]: spb_clustering = kmeans.predict(spb_grouped[same_names])

# add clustering labels
#spb_neighborhoods_venues_sorted.drop(columns=['Cluster Labels'], in
place=True)
spb_neighborhoods_venues_sorted.insert(0, 'Cluster Labels', spb_clustering)

unique, counts = np.unique(spb_clustering, return_counts=True)
print (unique, counts)

[0 2 3] [ 2 34 31]
```

```
In [103]: spb_merged= pd.merge(spb_neig.rename(columns = {'Neighbourhood': 'Neighborhood'}), spb_neighborhoods_venues_sorted, on="Neighborhood")
spb_merged.head(3)
```

Out[103]:

	Neighborhood	lat	lon	Cluster Labels	1st Most Common Venue	2nd Most Common Venue	3rd Most Common Venue	4th Most Common Venue
0	Автово	59.858637	30.278818	2	Gym / Fitness Center	Furniture / Home Store	Pharmacy	District
1	Адмиралтейский	59.931054	30.296798	3	Hotel	Restaurant	Bar	Food
2	Академическое	60.011526	30.394661	2	Bakery	Dessert Shop	Gym / Fitness Center	Playground

```

In [105]: # create map
map_spb_clusters = folium.Map(location=spb_center.astype(float).values.tolist()[0], zoom_start=10)

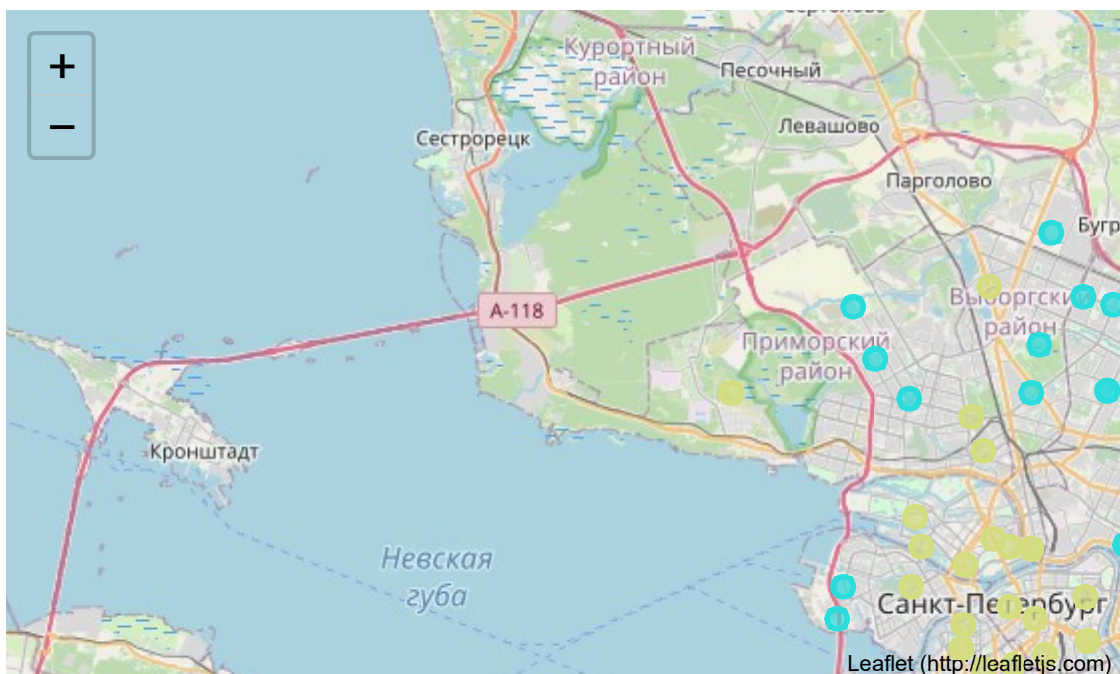
# set color scheme for the clusters
#x = np.arange(spb_clustering)
#ys = [i + x + (i*x)**2 for i in range(spb_clustering)]
#colors_array = cm.rainbow(np.linspace(0, 1, len(ys)))
#rainbow = [colors.rgb2hex(i) for i in colors_array]

# add markers to the map
markers_colors = []
for lat, lon, poi, cluster, v1, v2, v3 in zip(spb_merged['lat'], ms
k_merged['lon'], spb_merged['Neighborhood'], spb_merged['Cluster La
bels'],
                                             spb_merged['1st Most
Common Venue'], spb_merged['2nd Most Common Venue'], spb_merged['3r
d Most Common Venue']):
    label = folium.Popup(poi + ' Cluster ' + str(cluster) + ' top v
en: ' + str(v1) + ', ' + str(v2) + ', ' + str(v3) + ', ', parse_html=True)
    folium.CircleMarker(
        [lat, lon],
        radius=5,
        popup=label,
        color=rainbow[cluster-1],
        fill=True,
        fill_color=rainbow[cluster-1],
        fill_opacity=0.7).add_to(map_spb_clusters)

map_spb_clusters

```

Out[105]:



Conclusions and recommendations

So,

1. We managed to collect information about the coordinates of the districts of the two cities, for subsequent comparison.
2. Collect POI data and statistics for each area using Foursquare.
3. Conduct clustering of one area and determine the closest and most distant areas in terms of POI composition.
4. Mark out the districts of another city based on the proximity to one of the clusters of the first.

A person who is going to move from Moscow to St. Petersburg can now identify preferred districts in a familiar city and identify similar ones to this district in another city.

As a direction for the development of this topic, we can propose a statistical study of clusters and the interpretation of statistics in the form of a description of the properties of a cluster in terms that are understandable to the layman. This will allow you to explore areas not only by statistical similarity, but to form expectations based on a readable description of the properties of the area or cluster.