Battle of the Neighborhoods

How to choose the best neighborhood to live in

Table of contents

Introduction

The objective of this project is to find the best location to move into the Abingdon-on-Thames area. Currently looking to move into the area of Abingdon-on-Thames and would like to take advantage of this project to use it in my favour to help me identify the best and the worst areas to live.

First step is to choose the safest borough by analysing **police crime data.** Also I need to understand the structure of the boroughs so would like to get the **Lower Layer Super Output Areas (LSOA)** for the area of interest. LSOA are a geographic hierarchy designed to improve the reporting of small area statistics in England and Wales.
Finally, then after having defined the area, get data from **FourSquare** to help us choose the best area for supermarket, leisure centres, good schools, parks and restaurants, etc.

Using data science tools, will help us analyse data and focus on the safest borough and explore its neighborhoods and the common venues in each neighborhood.

The success criteria of the project is simple: after having analysed such factors, we would then be able to make the best choice for the family.

Go to the beginning

Data

After having defined our problem, below are the factors that will help us make our decission:

- finding the safest area using crime data statistics
- finding the neaby venues around the preferred areas
- choosing the right neighbourhood within the borough

  We will be using the geographical coordinates of Abingdon to plot neighbourhoods in a borough that is safe and in the city's vicinity, and finally cluster the neighborhoods, plot the crime data, get venues and present our findings.

  Following data sources will be needed to get the required information:
- **Step 1**: Using a real world data set from Police Data UK, get crime data for last year: A dataset consisting of the crime statistics of each Neighbourhoof in Thames Valley along with type of crime.
- **Step 2**: Gathering LSOA and UK boundary information from Ordonance Survey of the list of boroughs around for Oxfordshire: Borough information will be used to map the crime data data and identify a the boroughs that are best and worst.
- **Step 3**: Adding the dataset from FourSquare with the most common venues and the respective Neighbourhood along with co-ordinates: This data will be fetched using Four Square API to explore the neighbourhood venues and to apply machine learning algorithm to cluster the neighbourhoods and present the findings by plotting it on maps using Folium.

Go to the beginning

Step 1: Using a real world data set from Police Data UK, get crime data for last year

Thames Valley Crime Report

Properties of the Crime Report

- CRIME ID - Crime type
- MONTH - Recorded month
- REPORTED BY - authority who reported it
- FALLS WITHIN - authority responsible
- LONGITUDE - GPS longtitude
- LATITUDE - GPS latitude
- LOCATION - where was the crime
- LSOA code - borough code where it falls
- LSOA name - borough name where it falls
- GroupedArea - Inner Abingdon or Outer Abingdon
- CRIME TYPE - type of crime

Data set URL: https://data.police.uk (https://data.police.uk)

Import all the libraries that are needed beforehand

In [1]:
```
import time
import numpy as np  # data vectors
import pandas as pd # data analysis
from collections import Counter
from pandas.io.json import json_normalize # transform json in
to pandas dataframe
import matplotlib.cm as cm #plotting
import matplotlib.colors as colors
import matplotlib.pyplot as py
import json
import requests
from geopy.geocoders import Nominatim #get lat and long
from sklearn.cluster import KMeans #clustering
import folium #visualise map
import os
import geopandas as gpd
import earthpy as et
from folium.plugins import HeatMap
import geopandas
import matplotlib.pyplot as plt
import matplotlib.lines as mlines
from matplotlib.colors import ListedColormap
from shapely.geometry import box # Load the box module from s
hapely to create box objects
from shapely.geometry import shape
import earthpy as et
import seaborn as sns
from configparser import ConfigParser
```

Reading from the Dataset

Due to the amount of data, for this project, I limited the data for one year to include the year of 2019. Get the CSV file for the Crime data from the Police database data.police.uk

In [2]:
```
CSV = pd.read_csv("ThamesValleyCRIMEDATA.csv")
CSV.head()
```

Out[2]:

	Crime ID	Month	Reported by	Falls within
0	3d294010dbca88ade8b95964f03d79dae86cbece156e32...	2019-08	Thames Valley Police	Thames Valley Police
1	dbcbf4f976221e1e202c7019f2803f9ba80a8e1c8881d9...	2019-08	Thames Valley Police	Thames Valley Police
2	95569239a93eb375ef1a30f147975303c0aaa322755be2...	2019-08	Thames Valley Police	Thames Valley Police
3	cdb82cca5ab21305455295afad2e04a4f2b4b2066d2709...	2019-08	Thames Valley Police	Thames Valley Police
4	f94e1c275753292c47a748c7241c75beb667817009ef96...	2019-08	Thames Valley Police	Thames Valley Police

Total Crimes in different Locations

In [3]:
```
CSV['Location'].value_counts()
```

Out[3]:
```
On or near Supermarket             267
On or near Police Station          235
On or near Parking Area            187
On or near Petrol Station          101
On or near Sports/Recreation Area   90
                                  ...
On or near Hampden Road              1
On or near Alfreds Place             1
On or near Ginge Road                1
On or near The Pound                 1
On or near Conduit Road              1
Name: Location, Length: 1069, dtype: int64
```

Creating a pivot table to display the crimes by ward and by type

In [4]:

```
crime_cat_ward =pd.pivot_table(CSV,
                               values=['Count'],
                               index=['WARD'],
                               columns=['Crime type'],
                               aggfunc=len,
                               fill_value=0,
                               margins=True)
crime_cat_ward.head()
```

Out[4]:

	Count							
Crime type	Anti-social behaviour	Bicycle theft	Burglary	Criminal damage and arson	Drugs	Other crime	Other theft	Possessic of weapons
WARD								
Abingdon Abbey Northcourt	171	31	18	98	24	10	39	
Abingdon Caldecott	104	16	14	65	18	7	28	
Abingdon Dunmore	35	3	9	14	6	1	2	
Abingdon Fitzharris	107	15	13	99	96	10	28	1
Abingdon Peachcroft	56	5	18	18	6	7	12	

Creating a pivot table to display the crimes by major areas and by type

In [5]:
```
crime_cat_area = pd.pivot_table(CSV,
                               values=['Count'],
                               index=['GroupedArea'],
                               columns=['Crime type'],
                               aggfunc=len,
                               fill_value=0,
                               margins=True)
crime_cat_area.head()
```

Out[5]:

	Count							
Crime type	Anti-social behaviour	Bicycle theft	Burglary	Criminal damage and arson	Drugs	Other crime	Other theft	Posses of weapor
GroupedArea								
Abingdon Inner	473	70	72	294	150	35	109	
Abingdon Outer	542	54	291	451	128	80	315	
All	1015	124	363	745	278	115	424	

Pandas describe() is used to view some basic statistical details like percentile, mean, std etc. of a data frame or a series of numeric values.

In [6]:
```
crime_cat_area.describe()
```

Out[6]:

	Count					
Crime type	Anti-social behaviour	Bicycle theft	Burglary	Criminal damage and arson	Drugs	Other crime
count	3.000000	3.000000	3.000000	3.000000	3.000000	3.000000
mean	676.666667	82.666667	242.000000	496.666667	185.333333	76.666667
std	295.029377	36.678786	151.561869	228.941768	81.002058	40.104031
min	473.000000	54.000000	72.000000	294.000000	128.000000	35.000000
25%	507.500000	62.000000	181.500000	372.500000	139.000000	57.500000
50%	542.000000	70.000000	291.000000	451.000000	150.000000	80.000000
75%	778.500000	97.000000	327.000000	598.000000	214.000000	97.500000
max	1015.000000	124.000000	363.000000	745.000000	278.000000	115.000000

Merging the data by area and renaming columns

In [7]:

```
crime_cat_area.reset_index(inplace = True)
crime_cat_area.columns = crime_cat_area.columns.map(''.join)
crime_cat_area.rename(columns={'CountAll':'Total',
                                'CountAnti-social behaviour':'Antis
ocial',
                                'CountBicycle theft': 'Bike Theft',
                               'CountBurglary':'Burglary',
                               'CountCriminal damage and arson':'Cr
iminal damage',
                               'CountDrugs':'Drugs',
                               'CountOther crime':'Other crime',
                               'CountOther theft':'Other theft',
                               'CountPossession of weapons':'Posses
sion of weapons',
                               'CountPublic order':'Public order',
                               'CountRobbery':'Robbery',
                               'CountShoplifting':'Shoplifting',
                               'CountTheft from the person':'Theft
from the person',
                               'CountVehicle crime':'Vehicle crime
',
                               'CountViolence and sexual offences
':'Violence and sexual offences'}, inplace=True)
crime_cat_area.head()
```

Out[7]:

	GroupedArea	Antisocial	Bike Theft	Burglary	Criminal damage	Drugs	Other crime	Other theft	Posse: wea
0	Abingdon Inner	473	70	72	294	150	35	109	
1	Abingdon Outer	542	54	291	451	128	80	315	
2	All	1015	124	363	745	278	115	424	

Merging the data by ward and renaming columns

In [8]:
```
crime_cat_ward.reset_index(inplace = True)
crime_cat_ward.columns = crime_cat_ward.columns.map(''.join)
crime_cat_ward.rename(columns={'CountAll':'Total',
                                'CountAnti-social behaviour':'Antis
ocial',
                                'CountBicycle theft': 'Bike Theft',
                               'CountBurglary':'Burglary',
                               'CountCriminal damage and arson':'Cr
iminal damage',
                               'CountDrugs':'Drugs',
                               'CountOther crime':'Other crime',
                               'CountOther theft':'Other theft',
                               'CountPossession of weapons':'Posses
sion of weapons',
                               'CountPublic order':'Public order',
                               'CountRobbery':'Robbery',
                               'CountShoplifting':'Shoplifting',
                               'CountTheft from the person':'Theft
from the person',
                               'CountVehicle crime':'Vehicle crime
',
                               'CountViolence and sexual offences
':'Violence and sexual offences'}, inplace=True)
crime_cat_ward.head()
```

Out[8]:

	WARD	Antisocial	Bike Theft	Burglary	Criminal damage	Drugs	Other crime	Other theft	Possessio weapo
0	Abingdon Abbey Northcourt	171	31	18	98	24	10	39	
1	Abingdon Caldecott	104	16	14	65	18	7	28	
2	Abingdon Dunmore	35	3	9	14	6	1	2	
3	Abingdon Fitzharris	107	15	13	99	96	10	28	
4	Abingdon Peachcroft	56	5	18	18	6	7	12	

Sorting the data by crimes per ward

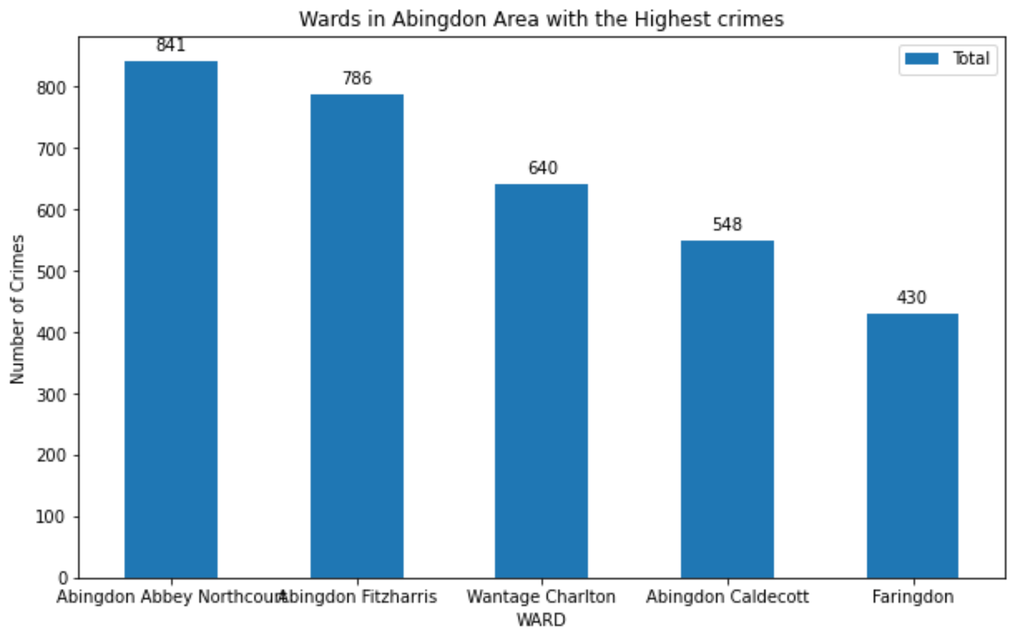
In [9]:
```
crime_cat_ward.sort_values(['Total'], ascending = False, axis
= 0, inplace = True )

crime_neigh_top5 = crime_cat_ward.iloc[1:6]
crime_neigh_top5
```

Out[9]:

	WARD	Antisocial	Bike Theft	Burglary	Criminal damage	Drugs	Other crime	Other theft	Possess weap
0	Abingdon Abbey Northcourt	171	31	18	98	24	10	39	
3	Abingdon Fitzharris	107	15	13	99	96	10	28	
21	Wantage Charlton	145	2	17	72	18	6	35	
1	Abingdon Caldecott	104	16	14	65	18	7	28	
9	Faringdon	53	3	20	40	21	6	31	

Five neighborhoods with highest crime

In [10]:
```
per_neigh = crime_neigh_top5[['WARD','Total']]

per_neigh.set_index('WARD',inplace = True)

ax = per_neigh.plot(kind='bar', figsize=(10, 6), rot=0)

ax.set_ylabel('Number of Crimes')
ax.set_xlabel('WARD')
ax.set_title('Wards in Abingdon Area with the Highest crimes
')

for p in ax.patches:
    ax.annotate(np.round(p.get_height(),decimals=2),
                (p.get_x()+p.get_width()/2., p.get_height()),
                ha='center',
                va='center',
                xytext=(0, 10),
                textcoords='offset points',
                fontsize = 10,
               )

plt.show()
```



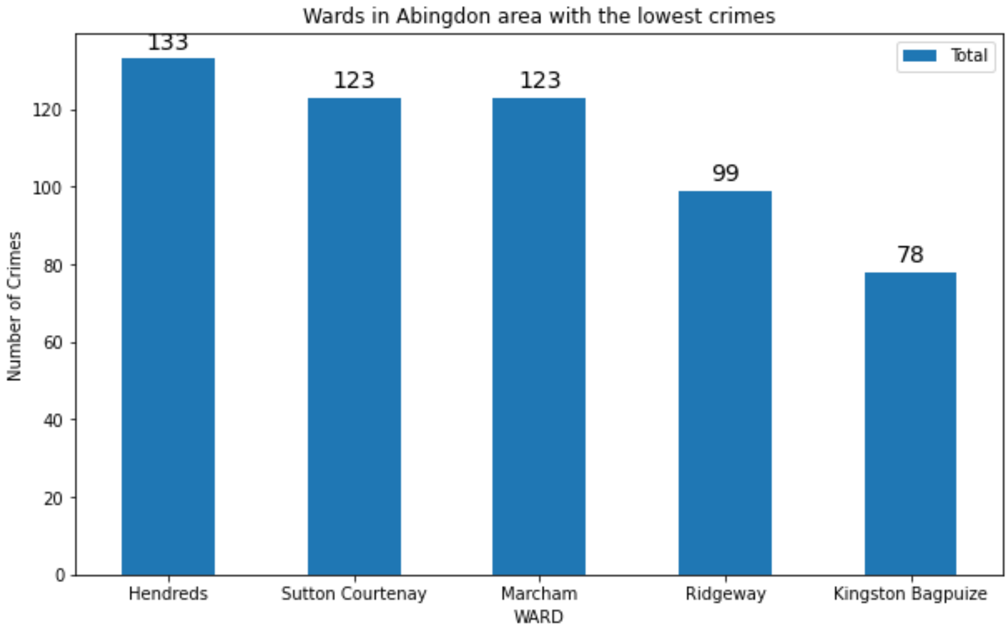
Five Neighborhoods with lowest crime

In [11]:
```
crime_neigh_low = crime_cat_ward.tail(5)

crime_neigh_low
```

Out[11]:

	WARD	Antisocial	Bike Theft	Burglary	Criminal damage	Drugs	Other crime	Other theft	Possessi weapo
11	Hendreds	13	2	10	14	4	1	16	
18	Sutton Courtenay	15	2	3	10	2	4	4	
14	Marcham	5	2	3	19	4	1	9	
15	Ridgeway	14	0	10	15	2	1	7	
13	Kingston Bagpuize	7	2	1	5	0	0	8	

In [12]:
```
per_neigh = crime_neigh_low[['WARD','Total']]

per_neigh.set_index('WARD',inplace = True)

ax = per_neigh.plot(kind='bar', figsize=(10, 6), rot=0)

ax.set_ylabel('Number of Crimes')
ax.set_xlabel('WARD')
ax.set_title('Wards in Abingdon area with the lowest crimes')

for p in ax.patches:
    ax.annotate(np.round(p.get_height(),decimals=2),
                (p.get_x()+p.get_width()/2., p.get_height()),
                ha='center',
                va='center',
                xytext=(0, 10),
                textcoords='offset points',
                fontsize = 14,
               )

plt.show()
```



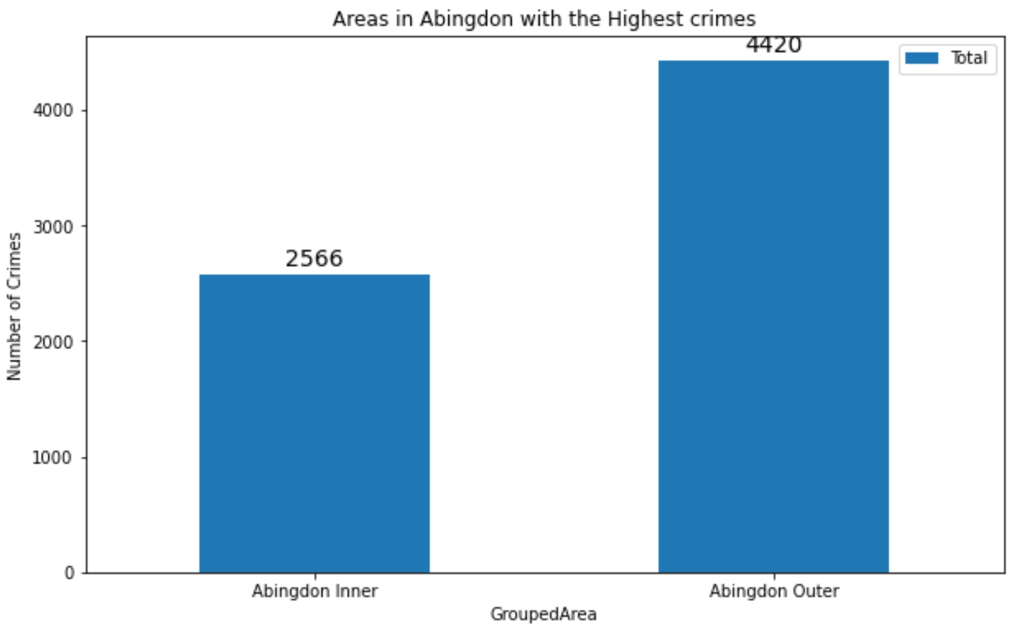
Abingdon Areas with Highest Crime

In [13]:
```
crime_only = crime_cat_area.drop([2])
crime_only
```

Out[13]:

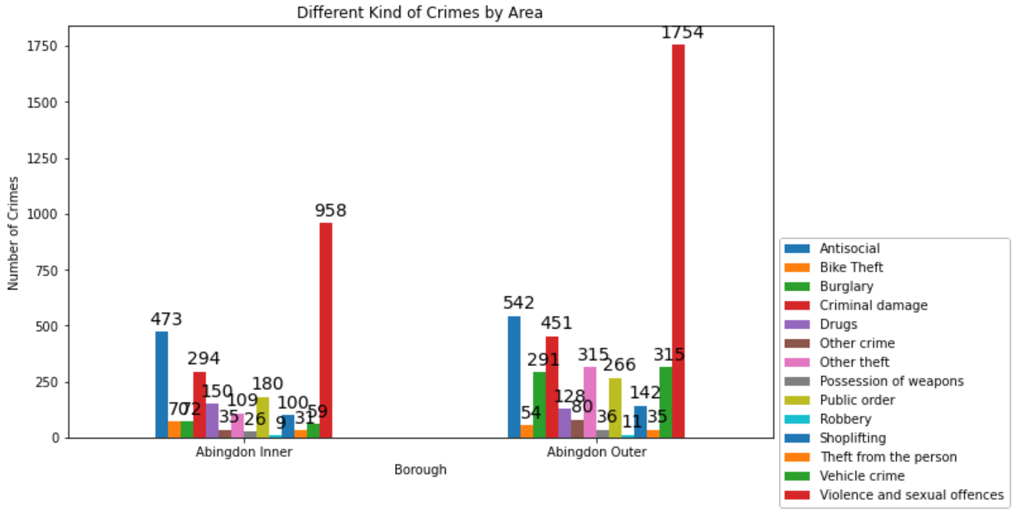
	GroupedArea	Antisocial	Bike Theft	Burglary	Criminal damage	Drugs	Other crime	Other theft	Posse wea
0	Abingdon Inner	473	70	72	294	150	35	109	
1	Abingdon Outer	542	54	291	451	128	80	315	

In [14]:
```
per_area = crime_only[['GroupedArea','Total']]

per_area.set_index('GroupedArea',inplace = True)

ax = per_area.plot(kind='bar', figsize=(10, 6), rot=0)

ax.set_ylabel('Number of Crimes')
ax.set_xlabel('GroupedArea')
ax.set_title('Areas in Abingdon with the Highest crimes')

for p in ax.patches:
    ax.annotate(np.round(p.get_height(),decimals=2),
                (p.get_x()+p.get_width()/2., p.get_height()),
                ha='center',
                va='center',
                xytext=(0, 10),
                textcoords='offset points',
                fontsize = 14,
               )

plt.show()
```



Different types of crimes recorded in both areas

In [15]:
```
#area_df = crime_only[['GroupedArea']]

#area_df = area_df.sort_values(['GroupedArea'], ascending = T
rue, axis = 0)

abi_ws = crime_only[['GroupedArea','Antisocial','Bike Theft
','Burglary','Criminal damage','Drugs','Other crime','Other t
heft','Possession of weapons',
                  'Public order','Robbery','Shoplifting','The
ft from the person','Vehicle crime','Violence and sexual offe
nces']]


abi_ws.set_index('GroupedArea',inplace = True)

ax = abi_ws.plot(kind='bar', figsize=(10, 6), rot=0)

ax.set_ylabel('Number of Crimes')
ax.set_xlabel('Borough')
ax.set_title('Different Kind of Crimes by Area')

for p in ax.patches:
    ax.annotate(np.round(p.get_height(),decimals=3),
                (p.get_x()+p.get_width()/3., p.get_height()),
                ha='center',
                va='center',
                xytext=(5, 10),
                textcoords='offset points',
                fontsize = 14
               )
    ax.legend(loc='upper left', bbox_to_anchor=(1.00, 0.5))

plt.show()
```



Go to the beginning

Part 2: Gathering LSOA and UK boundary information from Ordonance Survey of the list of boroughs around for Oxfordshire

In order to understand the boroughs, so we will get first get the shapefile for the UK Boundary from the Ordonance Survey. The way the boundaries are structured is by what is called the Lower Layer Super Output Areas (LSOA)

Get the shapefile for the UK Boundary from Ordnance Survey website

In [16]:
```
data = gpd.read_file('Sectors.shp')
ox_index = data[data.name == "OX"].index
ox_geom = data.loc[ox_index,'geometry']
ox_geom.head()
```

Out[16]:
```
GeoSeries([], Name: geometry, dtype: geometry)
```

In [17]:
```
shapefile = gpd.read_file("Sectors.shp")
shapefile.head()
```

Out[17]:

	name	geometry
0	AB10 1	POLYGON ((-2.11645 57.14656, -2.11655 57.14663...
1	AB10 6	MULTIPOLYGON (((-2.12239 57.12887, -2.12279 57...
2	AB10 7	POLYGON ((-2.12239 57.12887, -2.12119 57.12972...
3	AB11 5	POLYGON ((-2.05528 57.14547, -2.05841 57.14103...
4	AB11 6	POLYGON ((-2.09818 57.13769, -2.09803 57.13852...

Grouped the boundary data and the crime data to make an array to be displayed later in the map

In [18]:

```
#use only for LSOA grouping
Grouped = pd.DataFrame({'Value' : CSV.groupby( ['LSOA code
']).size()}).reset_index()

temp1 = Grouped.to_numpy()

newdf = pd.DataFrame(data=temp1, index=None, columns=["LSOA c
ode", "Value"])
#print(newdf)
#merge right to get back LSOA and one 'central' lat/long

df3 = CSV.drop_duplicates(subset='LSOA code', keep="first")


#newdf.insert(2, 'Latitude', newdf['LSOA code'].map(df_abiC.s
et_index('LSOA code')['Latitude']))
Temp = newdf.merge(df3, left_on='LSOA code', right_on='LSOA c
ode', how='right')


#Temp.drop_duplicates(subset="LSOA code",keep="first", inplac
e=True)
Temp['Latitude'] = Temp['Latitude'].astype(float)
Temp['Longitude'] = Temp['Longitude'].astype(float)
Temp['Value'] = Temp['Value'].astype(int)


#Making a simple table for values as with the dataframe foliu
m wasn't happy
Mapme = Temp[['Value','Latitude','Longitude']].copy()
Mapme.head()
```

Out[18]:

	Value	Latitude	Longitude
0	34	51.673533	-1.269892
1	108	51.675202	-1.260667
2	100	51.661178	-1.284287
3	102	51.659692	-1.289920
4	130	51.663491	-1.290685

Go to the beginning

Part 3: Adding the dataset from FourSquare with the most common venues and the respective Neighbourhood along with co-ordinates

Define Foursquare Credentials and Version

In [19]:
```
address = 'Abingdon-on-Thames, United Kingdom'
geolocator = Nominatim(user_agent="abi_explorer")
location = geolocator.geocode(address)
latitude = location.latitude
longitude = location.longitude
print('The geographical coordinates of Abingdon-on-Thames, Uni
ted Kingdom are {}, {}.'.format(latitude, longitude))
```

```
The geographical coordinates of Abingdon-on-Thames, United Ki
ngdom are 51.6714842, -1.2779715.
```

Now we are going to get data from FourSquare

In [20]:
```
parser = ConfigParser()
_ = parser.read('Credentials.cfg')
```

Used a parser file to hide credentials and confidential information

In [21]:
```
CLIENT_ID = parser.get('Credentials','CLIENT_ID') # your Four
square ID
CLIENT_SECRET = parser.get('Credentials','CLIENT_SECRET') # y
our Foursquare Secret
VERSION = parser.get('Credentials','VERSION') # Foursquare AP
I version
LIMIT = parser.get('Credentials','LIMIT') # A default Foursqu
are API limit value
```

Get the venues for the latitude desired

In [22]:
```
LIMIT = 100 # limit of number of venues returned by Foursquar
e API
radius = 1000 # define radius
 # create URL
url = 'https://api.foursquare.com/v2/venues/explore?&client_i
d={}&client_secret={}&v={}&ll={},{}&radius={}&limit={}'.forma
t(
    CLIENT_ID,
    CLIENT_SECRET,
    VERSION,
    latitude,
    longitude,
    radius,
    LIMIT)
url
```

Out[22]:
```
'https://api.foursquare.com/v2/venues/explore?&client_id=GOI
03BVANIMQZ54ZM0T5XJTFQSHWG43CDOWVLGVKDZFRXMUA&client_secret=
G5CADWKCNYVXYY54Z1ARDUINRHBAKSCAHFCV3HMPH00BVUZH&v=20180604&
ll=51.6714842,-1.2779715&radius=1000&limit=100'
```

In [23]:
```
results = requests.get(url).json()
```

In [24]:
```
# function that extracts the category of the venue from Fours
qure json response
def get_category_type(row):
    try:
        categories_list = row['categories']
    except:
        categories_list = row['venue.categories']

    if len(categories_list) == 0:
        return None
    else:
        return categories_list[0]['name']
```

In [25]:

```
venues = results['response']['groups'][0]['items']

nearby_venues = json_normalize(venues) # flatten JSON

# filter columns
filtered_columns = ['venue.name', 'venue.categories', 'venue.
location.lat', 'venue.location.lng']
nearby_venues =nearby_venues.loc[:, filtered_columns]

# filter the category for each row
nearby_venues['venue.categories'] = nearby_venues.apply(get_c
ategory_type, axis=1)

# clean columns
nearby_venues.columns = [col.split(".")[-1] for col in nearby
_venues.columns]

nearby_venues.head()
```

```
<ipython-input-25-561c05f0fdd1>:3: FutureWarning: pandas.io.
json.json_normalize is deprecated, use pandas.json_normalize
instead
  nearby_venues = json_normalize(venues) # flatten JSON
```

Out[25]:

	name	categories	lat	lng
0	Crown and Thistle	Pub	51.669604	-1.280457
1	The Nags Head	Pub	51.668649	-1.279253
2	Costa Coffee	Coffee Shop	51.670426	-1.281812
3	The Broad Face	Pub	51.669315	-1.280289
4	Waitrose & Partners	Supermarket	51.672038	-1.279792

In [26]:

```
print('{} venues were returned by Foursquare.'.format(nearby_
venues.shape[0]))
```

```
24 venues were returned by Foursquare.
```

In [27]:
```
NVgdf = geopandas.GeoDataFrame(nearby_venues, geometry=geopan
das.points_from_xy(nearby_venues.lng, nearby_venues.lat))

NVgdf
```

Out[27]:

	name	categories	lat	lng	geometry
0	Crown and Thistle	Pub	51.669604	-1.280457	POINT (-1.28046 51.66960)
1	The Nags Head	Pub	51.668649	-1.279253	POINT (-1.27925 51.66865)
2	Costa Coffee	Coffee Shop	51.670426	-1.281812	POINT (-1.28181 51.67043)
3	The Broad Face	Pub	51.669315	-1.280289	POINT (-1.28029 51.66931)
4	Waitrose & Partners	Supermarket	51.672038	-1.279792	POINT (-1.27979 51.67204)
5	Chaba	Thai Restaurant	51.670013	-1.283104	POINT (-1.28310 51.67001)
6	The Kings Head & Bell	Pub	51.669603	-1.281333	POINT (-1.28133 51.66960)
7	The Brewery Tap	Pub	51.669936	-1.286519	POINT (-1.28652 51.66994)
8	Abingdon Lock	Canal Lock	51.670538	-1.269276	POINT (-1.26928 51.67054)
9	ASK Italian	Italian Restaurant	51.670432	-1.284324	POINT (-1.28432 51.67043)
10	PizzaExpress	Pizza Place	51.670651	-1.281020	POINT (-1.28102 51.67065)
11	Dil Raj	Indian Restaurant	51.670179	-1.284381	POINT (-1.28438 51.67018)
12	Java&Co	Café	51.670439	-1.281238	POINT (-1.28124 51.67044)
13	Throwing Buns	Tea Room	51.670161	-1.281673	POINT (-1.28167 51.67016)
14	BP	Gas Station	51.673320	-1.280400	POINT (-1.28040 51.67332)
15	Boots	Pharmacy	51.670591	-1.282602	POINT (-1.28260 51.67059)
16	Greggs	Bakery	51.670566	-1.282471	POINT (-1.28247 51.67057)
17	The Narrows (Wetherspoon)	Pub	51.670291	-1.283370	POINT (-1.28337 51.67029)
18	WHSmith	Stationery Store	51.670857	-1.282824	POINT (-1.28282 51.67086)

	name	categories	lat	lng	geometry
19	Abbey Gardens	Park	51.670692	-1.278674	POINT (-1.27867 51.67069)
20	R&R	Coffee Shop	51.670588	-1.281345	POINT (-1.28134 51.67059)

Go to the beginning

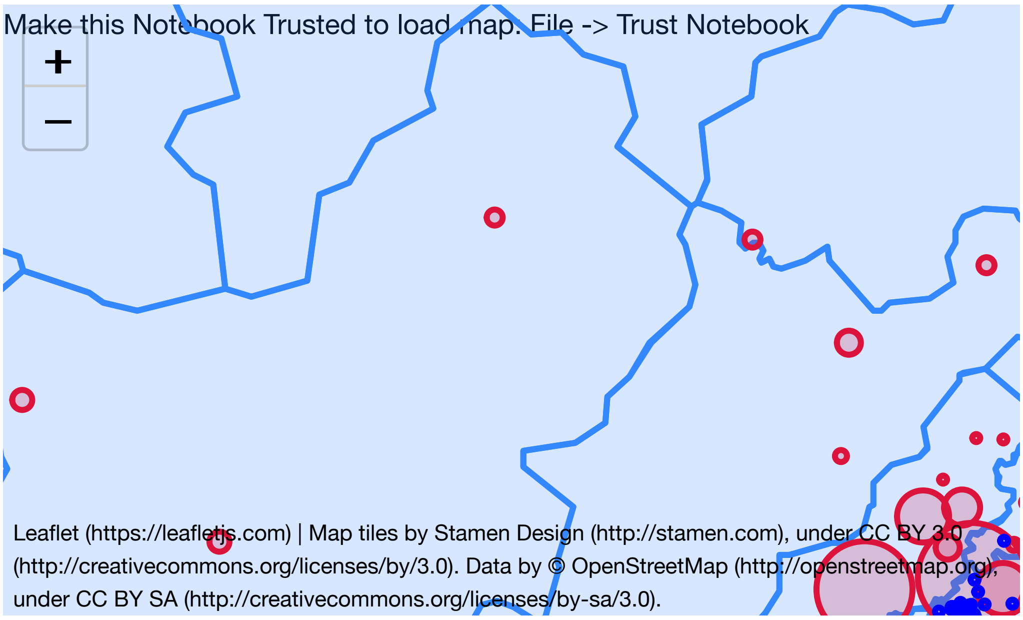
Mapping and Analysis

This part will allow us to explore more about the neighbourhood, venues and the crime around the area of interest by plotting it on maps using Folium and perform exploratory data analysis.

In [28]:

```
abi_map = folium.Map(location=[latitude, longitude], zoom_sta
rt=12, tiles='stamenterrain')

#add the boundary first
boundary = folium.features.GeoJson(shapefile)
abi_map.add_child(boundary)

# Add crime to the map one by one sized by the grouping info
for i in range(0,len(Mapme)):
   folium.Circle(
      #popup=Mapme.iloc[i]['Value'],
      location=[Mapme.iloc[i]['Latitude'],Mapme.iloc[i]['Long
itude']],
      radius=Mapme.iloc[i]['Value']*1.5,
      fill=True,
      color='crimson',
      fill_color='crimson'
   ).add_to(abi_map)

#add foursquare venues
for j in range(0,len(nearby_venues)):
   #print([nearby_venues.iloc[j]['lat']])
   folium.Circle(
      popup=nearby_venues.iloc[j]['categories'],
      location=[nearby_venues.iloc[j]['lat'],nearby_venues.il
oc[j]['lng']],
      radius=50,
      fill=True,
      color='blue',
      fill_color='blue'
   ).add_to(abi_map)

abi_map
```

Out[28]:



Go to the beginning

Conclusion

Using a combination of crime data from Police Data UK grouped per ward, together with the boundary areas and the common venues data from FourSquare helped visualise both to take an informed decision.

By visual inspection, as expected, there are more venues in the Abingdon Inner area but also more crime too.

Also, we can see that the outer Abingdon has fewer crimes and it is therefore safer to move to. The limitation of fewer venues, is not prime concern for our family and as we are willing to compromise to have to travel a bit.