Independent Project

Due date: 15th January 2024 by 4pm

Instructions

You will have the opportunity to test your Python skills by developing a "Crime Mapping User Interface". The purpose of this interface is to retrieve a information on crimes by neighbourhood during a specified year and month. The interface will also produce statistical summaries, data visualisation, and kernel density estimation on crime locations by neighbourhood searched.

Answer all questions, paying attention to the instructions on file and function names. In addition to the requested code files, some questions ask for text answers such as short explanations. Include these, indicating clearly which answers are relevant to which questions, in a single file answers.txt or answers.docx. If you'd like to create a more professional report you are welcome to include your graphs in this file as well. When you are done, zip all the requested files into an archive and hand in that zip file, using your candidate number and the module code as the name (e.g., `KTJA9_SECU0012.zip`).

You are allowed to work in pairs or alone. If you work in pairs, you may hand in the same code but your written answers must be done independently. You must indicate to us at beginning of the project who you are working with (using their name and not the candidate number), and you must indicate in your report who you worked with (using their candidate code and not their name) to ensure anonymity. In each answer, explain roughly what the contributions were of each group member.

Although in pairs it may happen that some programs may be written mainly by one person, it is important that both the members will be able to explain what the code does on their own.

Plagiarism

If your code for a question is identical (modulo variable renaming) to code we find on the Internet or to the code of your own previous submission (if this is your second attempt) or of another student not in your group, we will consider that plagiarism and proceed accordingly. There are some important differences between the previous assignment and this one. These differences would allow me to easily recognise if code is copied by previous submissions and would, therefore, be treated accordingly.

1 User Interface [70 marks]

This section of the project consists building a user interface. It will use the Data.Police.uk API to retrieve information on crimes or stop and searches.

1.1 User Input [5 marks]

Task 1. [**5 marks**] Create a file `interface.py`. This tasks consists on creating the first part of the user interface, where we ask the user some input parameters which will customize the API search. You will need to do the following:

- Ask the user for input on neighbourhoods' names. The interface should allow multiple neighbourhood names as input. You can decide to save the input neighbourhood names in a list or a dictionary, so every time a user inputs a name, it is saved on the container. You may find useful to check the API documentation on neighbourhoods part of the Metropolitan Police (as we will focus on London `https://data.police.uk/docs/method/neighbourhoods/`.
- You should also ask the user input for a specific year. Note the API takes only YYYY format

(i.e. '2023').

- You need to ask the user input for a specific month. You can give the user options for how to input the month, but consider the police API takes only MM format (i.e. '05' for May).
- You will also need to ask the user for a specific crime type ('burglary', 'robbery', etc.). You may find useful to check the API documentation on crime categories at:

  `https://data.police.uk/docs/method/crime-categories/`.

  This will allow you to be sure the user input is consistent with the crime type in the API.
- Now write some code that saves the search parameters input by the user as a dictionary `parameters`.
- Save the `parameters` dictionary using pickle.

Try the interface by input the following parameters:

- Victoria
- London Fields
- Hackney Wick
- Bayswater
- Year: 2023
- Month: 05
- Category: robbery

Save the `interface.py` file.

Make sure to take the necessary precautions so the input is valid (i.e. setting input to lowercase (not necessarily in the case of neighbourhoods), string, re-asking user input). You should save the user input into variables which will serve as search parameters in the following tasks.

1.2 API [60 marks]

In the previous task, we asked the user for some search parameters (neighbourhoods' names, year, month, and crime type). In this section, you will use the police API `https://data.police.uk` to retrieve the information requested by the user. For this you will need to complete the tasks below.

1.2.1 Get Neighbourhoods' ID

Task 2. **[15 marks]** Given that we want to retrieve crime data by neighbourhood, we will first need the coordinates of the vertices of the neighbourhood boundary. To get those, we will first need to get the neighbourhood unique identifier. The neighbourhood identifiers can be retrieved from `https://data.police.uk/api/metropolitan/neighbourhoods`.

Create a file `api.py` in which you do the following:

- Make an API search using `https://data.police.uk/api/metropolitan/neighbourhoods` as the URL.
- For the purposes of testing your code on this task, use the search parameters specified in task 1.
- Retrieve the neighbourhoods' 'id' if the neighbourhood name matches the user input.
- Once you have the neighbourhoods' id's, write some code which checks whether each neighbourhood input by the user was found or not. An output example would be: 'Neighbourhood : found'...

At the end of this task you should have the neighbourhoods' names along with their unique neighbourhood identifiers stored on a dictionary named `dict_nid`, with the names as keys and the identifiers as values.

1.2.2 Get Neighbourhoods' Boundary Coordinates

Task 3. [**15 marks**] Now that we have the neighbourhoods' identifiers, we can search for their boundary using the API. We will need the boundary coordinates to retrieve only the crimes that fall inside. To retrieve the boundary coordinates, for *each* neighbourhood on `dict_nid` you will do the following:

- Make a request on the police API, modifying the URL so the force is 'metropolitan' and the 'id' matches that of the neighbourhood of interest (stored in `dict_nid` as value).
- Retrieve the longitude and latitude coordinates of the neighbourhood's boundary. You can use lists for this.

You can find more information on how to retrieve the neighbourhoods' boundaries on:
`https://data.police.uk/docs/method/neighbourhood-boundary/`.

1.2.3 Create a bounding box

Task 4. [**15 marks**] The Police API requires the polygon coordinates in the format:
`[lat],[lng]:[lat],[lng]:[lat],[lng]`
As you probably noticed, if we were to input all the coordinates we would get a 400 status code because the URL would be longer than 4094 characters (the limit on this API). A way to overcome this issue is to create a bounding box with only 4 vertices/coordinate pairs. The bounding box we will use is a rhomboid containing the borough of interest.

- With the coordinates retrieved, get the maximum, minimum, and medium latitude (y) and longitude (x) coordinates and store these in (6) different variables. If you stored the latitude and longitude coordinates on separate lists, this part is easily accomplished using the max/min functions.
- Create 4 new variables `top`, `bottom`, `left`, and `right`. Each representing a vertex of the bounding box. The values of each of these vertices will be given by so that `top` for example, would be a concatenation of the maximum latitude + ',' + medium longitude, `left` would be a concatenation of the minimum longitude and medium latitude. Make sure the max/min/medium lat/lng variables are strings.
- The previous mini task results in 4 lat-lng pairs. You will now concatenate these 4 pairs into one new variable `polygon`, where each pair is separated by a ':'. This creates a unique long string representing the 4 vertices of the bounding box of the neighbourhood of interest.
- For each neighbourhood key, save the polygon variable as value. You can overwrite the neighbourhood 'id' as it is no longer needed.

Task 5. [**15 marks**] Now that you have the bounding box coordinates for each neighbourhood, you can search for crime data using the police API.

- You will have to modify the URL on each API request to meet the specifications of each independent search. This means changing the polygon, the year, the month, and the crime type in the URL request as appropriate.
- For each search, for each crime event, you must retrieve: category, latitude, and longitude.
- Store the results of your search on a list of lists `results`. Each individual list should contain the neighbourhood, crime type, latitude, and longitude information of an individual crime

event. For example: ['Kentish Town', 'robbery', '51.556690', '-0.130231'].

You will find more information on how to make this search on:
`https://data.police.uk/docs/method/crime-street/`

- Now merge `interface.py` and `api.py` into a single file called `run_interface.py`.
- Make the necesary changes in the code (if applicable) so that when `run_interface` is run accomplishes the goals of the previous tasks in one go.

1.3 CSV Output [5 marks]

Task 6. [**5 marks**] You will now write some code in `run_interface.py` to output the search results as a CSV file. You need to do the following:

- Write some code that output a CSV containing the information on the `results` list. Each row in the CSV should contain all the information (neighbourhood, category, latitude, longitude) of an individual crime.
- Note that if the user were to input different parameters, on a new search, `interface.py` should output (if accepted by the user) a new CSV with the corresponding search information.

At this point, you should have a Python script which when run uses the police API to retrieve crime data with the search parameters specified by the user. The script is also able to generate a CSV file.

2 Data Analysis and Visualisation [30 marks]

With the interface operational, you will now focus on creating numerical and graphical outputs.

For the next tasks, you will need the following modules:

- scipy.stats
- numpy
- matplotlib.pyplot

Task 7. [**10 marks**] You will now make two searches, one for robbery and one for burglary, on the following neighbourhoods:

- Bexleyheath
- West Ham
- Peckham
- Golders Green
- Soho
- King's Cross

In both searches, use 2023 as the year, and May as the month.

The optimal output would be only two CSV files, one per robbery and one per burglary. Alternatively, you may choose to output a file per crime and borough.

- Write a function `plot` that takes as arguments a neighbourhood and a CSV file (produced by the search), and produces a scatter plot of the crimes in that neighbourhood.
- The y-axis is given by the latitude coordinates, while the x-axis is given by the longitude coordinates.

Task 8. **[10 marks]** The distribution of crime locations is not random. Using the CSV's from task 7, you will now compute a kernel density estimation (KDE) test. KDE is a technique which calculates a smooth probability surface based on the density of points in space. In other words, KDE maps are useful to locate crime hotspots.

- Write a function `kde` which takes as arguments a neighbourhood and a dataset (the CSV datasets), and creates a KDE map for that neighbourhood.
- Make KDE maps for all 6 neighbourhoods, and for both, robberies and burglaries.

Task 9. **[10 marks]** Create a file `answers` (it can be a txt or a word document according to your preference) in which you reflect on your findings by answering the following questions:

- Based on your analysis, to what extent are robbery hotspots similar to burglary hotspots?
- What visualisation do you consider to be a better at convening information and why?
- What do you consider to be some of the limitations of KDE-generated maps?