|  |
| --- |
|  |
| GFI_logo1 |

***CoE Challenge***

***UK Post Code agglomerative clustering algorithm***

*GFl’s Proposal*

INDEX

1. Document revision history 3

2. PURPOSE 3

3. PRE-Requisites 3

4. CONTEXT OF THE SOLUTION 3

5. testing 4

5.1 Controlled datasets 4

5.1.1 Test I: Chosing 2 clusters 5

5.1.1.1 Testing case A 5

5.1.1.2 Testing case B 5

5.1.2 Test II: Chosing 3 clusters 6

5.2 Uncontrolled datasets 7

5.2.1 Test I: Chosing 2 clusters 7

5.2.2 Test II: Chosing 5 clusters 8

6. The importance of chosing clusters 9

6.1 Dendrograms for cluster choices 9

6.2 How to choose the “best” cluster choice 10

7. how to use the script 11

7.1 Folders 11

7.1.1 Inputs folder 11

7.1.2 Script folder 11

7.1.3 Output folder 11

7.2 How to tweak the program 12

7.2.1 Normal run: 12

7.2.2 Controlled Testing runs: 13

7.2.2.1 First dataset: 13

7.2.2.2 Second dataset: 13

7.2.3 Uncontrolled testing runs: 14

1. Document revision history

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Version** | **Date** | **Changed By** | **Reviewed By** | **Comments** |
| 1.0 | 10/10/2019 | GFI | Adrián Romero-Dapena | Initial version |

1. PURPOSE

The purpose of this document is to propose a solution for the challenge sent by David Torrecilla where an aggrupation of post-codes by the minimum distances among them is requested.

The conditions where the following ones:

* Create an algorithm that groups a number of post codes by the shorter distance between each other.
* The software on which is based the solution has to be free.
* Any programming language is allowed.

1. PRE-Requisites

In order to run this script you will need:

* + Python 3.7 or similar
  + Excel or similar
  + The following libraries to be installed in your local machine:
    - Numpy
    - Pandas
    - Geopandas
    - Shapely
    - Matplotlib
    - Sklearn

**Please, notice:** The installation of some of this libraries may be difficult depending on the machine. We recommend to do it on a Linux machine if possible or with Condas frame in case of having a Windows machine.

1. CONTEXT OF THE SOLUTION

The solution applied to solve the problem is to make use of an Clustering Agglomerative Algorithm based on Euclidean distances.

The approach of the solution is to load all the post codes of UK with a relation of longitudes and latitudes.

This will serve as the database of the script from which we will take information for the list of post codes that the user wants to check.

The script will calculate POINT objects from those latitudes and longitudes and then will normalise the data on a unique array of points with the geographical details.

With the details, the script allows us to visualise a dendrogram to choose the number of clusters that we want on our output. Check the link below for better understanding on how to use the dendrogram.

<https://towardsdatascience.com/machine-learning-algorithms-part-12-hierarchical-agglomerative-clustering-example-in-python-1e18e0075019>

This means that, if we have 8 post codes, if we choose to have 2 clusters the script will try to make fit the postcodes in 2 groups if possible.

The script will respect the number of clusters chosen by the user whenever the choice is logical. However, it might not respect the homogeneity of the number of postcodes among clusters.

For example: if we have a distribution of 100 postcodes and we want the clusters to have 4 post codes (grouped by the closest distance) the logic tells us to use 25 clusters (100 / 4). We expect each cluster to have 4 postcodes and we also expect to have 25 clusters in our output.

This is not always like is seems. In a random input of postcodes, some of the clusters may group 6 postcodes in them, other 4, other 8 and others even 1 (imagine to have 98 postcodes in London, one in Leeds and one from Edingburgh, for example. Even if your choice of cluster was 2, the logic tells us that 3 might be the best option. The script would opt for that).

However, the algorithm is always going try to respect the users’ decisions on the number of clusters (25 in the example above).

1. TESTING

We have performed 3 tests:

* 2 with a controlled set of data, this is a fixed set of post-codes.
* 1 with a non-controlled set of data, this is a random set of post-codes.
  1. Controlled datasets

Our intention was to see, clearly, whether the script was making the right choices or not when grouping the postcodes. To do so, we have produced two datasets from Edinburgh locations.

Both datasets have 8 postcodes.

The first dataset (PY0013\_Dataset\_1\_8PostCodes\_2\_clusters.csv) contains:

* 4 postcodes that are really close to each other (around the Castle area).
* 4 postcodes are also quite close to each other (neighborhood behind Arthur’s Seat), but the two groups of 4 are separated from each other.

The second dataset (PY0013\_Dataset\_2\_8PostCodes\_3\_clusters.csv) contains:

* 4 postcodes quite close to each other (neighborhood behid Arthur’s Seat).
* 3 postcodes quite close to each other (around the Castle area).
* 1 postcode relatively far from the locations of the other groups of postcodes (Leith Port area).
  + 1. Test I: Chosing 2 clusters

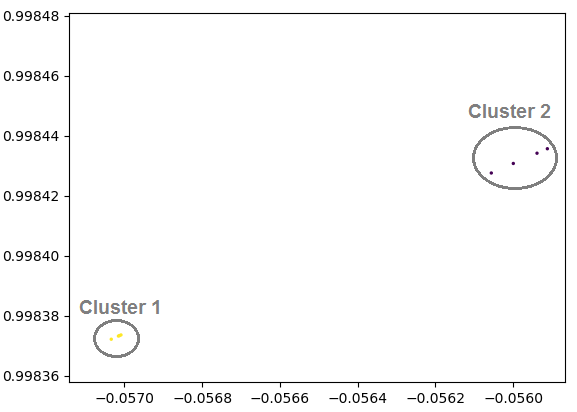
For both datasets, we run the script obtaining positions grouped by the minimum distance.

* + - 1. Testing case A

In the first axis chart, we used the input file PY0013\_Dataset\_1\_8PostCodes\_2\_clusters.csv, with a two separated group of chosen postcodes.

* Cluster 1 is the group of postcodes around Edinburgh’s Castle.
* Cluster 2 is the group of postcodes from the neighborhood between Arthur’s Seat and Portobello.

The script is able to categorise them correctly.

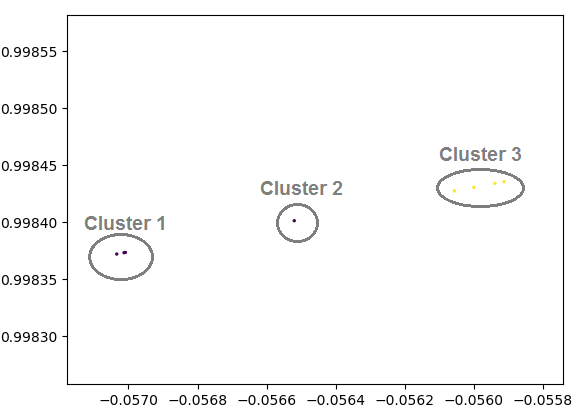


**Figure 1**: Graphical cluster representation of normalized array following Test I.A approach with a controlled dataset. **Source**: Produced by script.

* + - 1. Testing case B

We have also run the same test (2 clusters choice) with the input file PY0013\_Dataset\_2\_8PostCodes\_3\_clusters.csv, that has, as mentioned, one post code is kind of far from the others.

The outcome looks like this:



**Figure 2**: Graphical cluster representation of normalized array following Test I.B approach with a controlled dataset. **Source**: Produced by script.

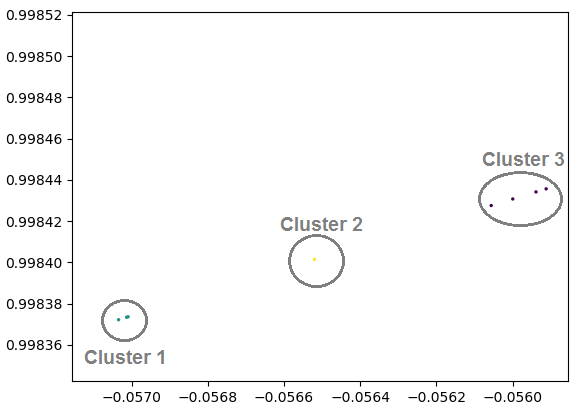
What happened is that, geometrically speaking, to group the postcodes in only 2 clusters is an imbalanced solution. Therefore, the algorithm is going for the best render solution: swapping to 3 clusters on its own.

* Cluster 1 is the group of postcodes around Edinburgh’s Castle.
* Cluster 2 is the far away one, in Leith Port. That’s why it has been classified in its own cluster.
* Cluster 3 is the group of postcodes from the neighborhood between Arthur’s Seat and Portobello.
  + 1. Test II: Chosing 3 clusters

The dataset used for this testing case is PY0013\_Dataset\_2\_8PostCodes\_3\_clusters.csv.

As we can see in the axis chart below, the script recognises correctly the 3 closest groups:

* Cluster 1 is the group of postcodes around Edinburgh’s Castle.
* Cluster 2 is the far away one, in Leith Port. That’s why it has been classified in its own cluster.
* Cluster 3 is the group of postcodes from the neighborhood between Arthur’s Seat and Portobello.



**Figure 3**: Graphical cluster representation of normalized array following Test II approach with a controlled dataset. **Source**: Produced by script.

* 1. Uncontrolled datasets

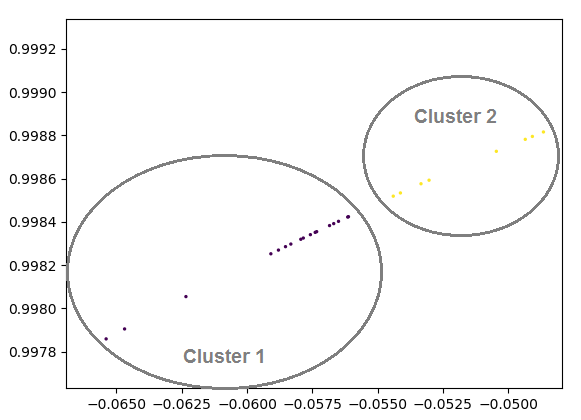
We have also generated a random set of 25 postcodes from Edinburgh city.

The script itself will choose a shuffled set of postcodes and will start clustering the input postcodes.

For this uncontrolled dataset, we have performed 2 tests.

* + 1. Test I: Chosing 2 clusters

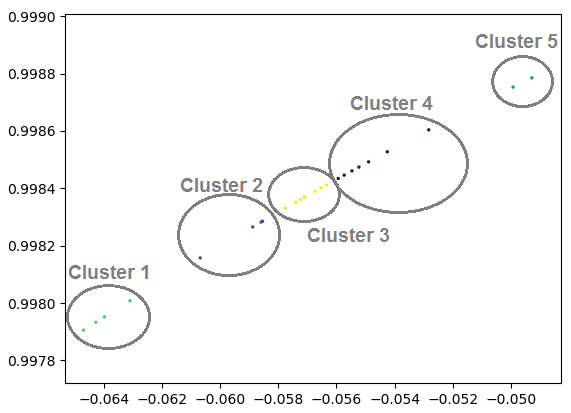
As we can see and understand (for the similarities with the axis charts above) the script is able to group the data in 2 clusters:



**Figure 4**: Graphical cluster representation of normalized array following Test I approach with an uncontrolled dataset. **Source**: Produced by script.

* + 1. Test II: Chosing 5 clusters

Similar to the above, here they are the evidences:



**Figure 5**: Graphical cluster representation of normalized array following Test II approach with an uncontrolled dataset. **Source**: Produced by script.

1. The importance of chosing clusters

As per the evidences brough to this point, you might have realised that the choice of the number of clusters is critical in the performance of this machine learning algorithm.

The agglomerative clustering technique used in this script allows the user to have the option of choosing the quantity regarding the clusters choice.

To do it in a wise way, you might want to have a look at dendrograms.

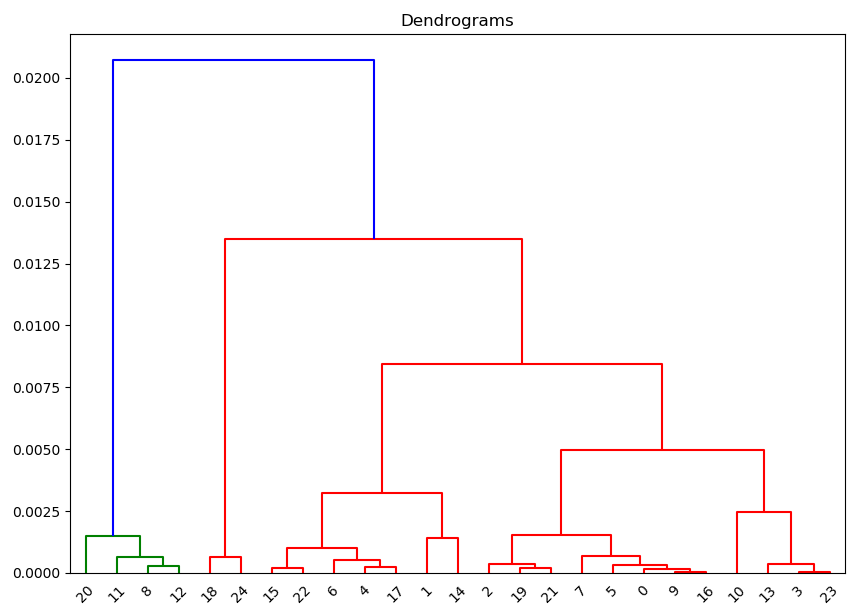
* 1. Dendrograms for cluster choices

Dendrograms will help you choosing the best number of clusters for your data set.

Since the algorithm starts agglomerating clusters by grouping points in pairs, the distribution of the clusters agglomeration can be really different depending on the number of clusters and the randomless of data provided.

In the figure below we can see how the algorithm plans the agglomeration for the 5-cluster dataset of random Edinburgh’s test case.

Please, notice that – first – pairs of data are put together. Later, those clusters of 2 get agglomerated and so on.



**Figure 6**: Dendrogram example of uncontrolled dataset. **Source**: Produced by script.

* 1. How to choose the “best” cluster choice

The purpose of dendrograms is to help us on this decision.

We should trace an horizontal line between the two horizontal lines which are the farthest from each other (see **orange line** in Figure 6).

If we do so, we see that we are going to cut two vertical lines (the two blue vertical ones). The number of cuts is the most suitable number of clusters.

This might not be always like this. This is why is left to the user the choice on the number of clusters.

There is plenty information about it on the internet.

1. HOW TO USE THE SCRIPT

You need to follow certain steps to set up the script on your machine.

1. We are going to assume that you have already fulfil all the previous requirements (Section 3).
2. Paste folder that you’ll get after the .ZIP extraction (named “Python”) on your “Documents” local folder.
3. Inside the PY0013\_UK\_PostCodes you will find several subfolders.
   1. Folders
      1. Inputs folder

* File “ukpostcodes.csv” is the one that holds all the post codes of UK. Used as DB file.
* File “input.csv” is a file with only one column where the user can paste a list of post codes to be checked.
  + 1. Script folder

The script folder holds only one file: PY0013\_UK\_PostCodes.py.

That one is the script proposed for the solution.

* + 1. Output folder

Each run of the script will overwrite the output file “cluster\_postcodes.xlsx”.

You can find two samples more of testing outputs from the uncontrolled dataset scenarios:



The outputs file will provide a set of information. However, the important columns will be: POSTCODE & CLUSTER.

In the screenshot below you can appreciate that there is a set of 25 postcodes grouped in 5 clusters (from 0 to 4) rated in column “cluster”.



**Figure 7**: Ouput example after cluster classification following the uncontrolled dataset approach. **Source**: Produced by script.

* 1. How to tweak the program

Just commenting or uncommenting some lines you will activate the different scenarios that the script has.

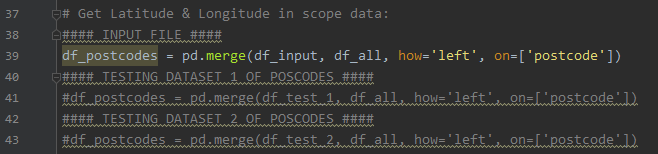
To uncomment one line, delete the ‘#’ leading character on the line.

To comment back the line, write a ‘#’ leading character on the line.

Only one of these scenarios should be set up.

* + 1. Normal run

Uncomment line 39 by deleting the ‘#’ if there is any.

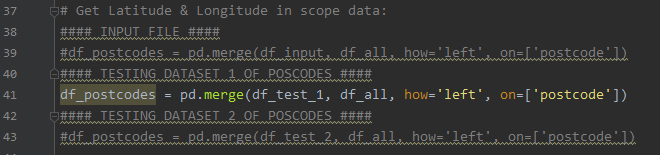


**Figure 8**: Regular execution of script. **Source**: Snippet from IDE PyCharm.

* + 1. Controlled Testing runs:
       1. First dataset:

The first dataset (PY0013\_Dataset\_1\_8PostCodes\_2\_clusters.csv) contains:

* 4 postcodes that are really close to each other (around the Castle area)
* 4 postcodes are also quite close to each other (neighborhood behid Arthur’s Seat), but the two groups of 4 are separated from each other.

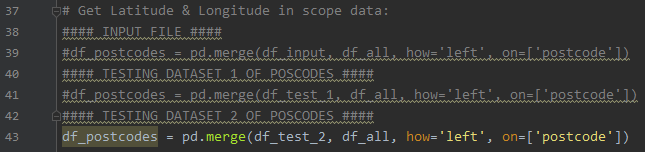


**Figure 9**: Scripts execution for test I. **Source**: Snippet from IDE PyCharm.

* + - 1. Second dataset:

The second dataset (PY0013\_Dataset\_2\_8PostCodes\_3\_clusters.csv) contains:

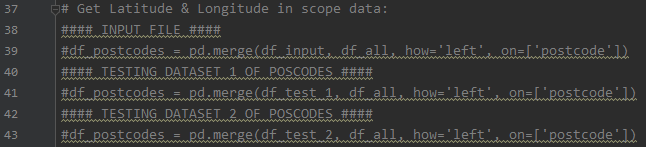
* 4 postcodes quite close to each other (neighborhood behid Arthur’s Seat)
* 3 postcodes quite close to each other (around the Castle area)
* 1 postcode relatively far from the locations of the other groups of postcodes (Leith Port area)



**Figure 10**: Scripts execution for test II. **Source**: Snippet from IDE PyCharm.

* + 1. Uncontrolled testing runs:

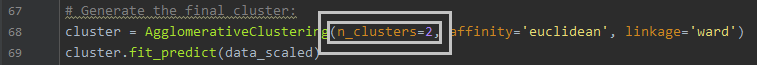
The script itself will choose 25 shuffled postcodes and will start clustering them.



**Figure 11**: Scripts execution for uncontrolled datasets. **Source**: Snippet from IDE PyCharm.

* + 1. Change the value for the number of clusters.

To choose the number of clusters, just change the value surrounded by the rectangles:



**Figure 12**: Adjustments to choose the number of clusters. **Source**: Snippet from IDE PyCharm.