Mapping Jerusalem – The ClusterOn app

A Data Science for Social Good (DSSG)[1] Project

By: Ameer Alayan[1], Anna Rubtsov[1], David Oriel[1], Amir Kleiner[2]
Supervision: Dan Pelleg[1]
[1] Hebrew University, Jerusalem, Israel
[2] Weizmann Institute of Science , Rehovot, Israel

Table of Contents

Introduction

Jerusalem is one of the most sensitive and complex cities in the world historically and culturally - being a spiritual center for the three major monotheistic religions, a city with a built and written history of over 3000 years.

In recent decades, Jerusalem has been under enormous development pressures (the population is expected to double over less than two decades), which constitutes a substantial threat to its unique qualities. Jerusalem, therefore, might also be seen as an 'accelerated time machine' of sorts.

These circumstances result in a pressing question: How can the city respond to the development pressures without losing its unique qualities?

Though originally formulated for the city of Jerusalem, this is an issue that is applicable to any city undergoing urban renewal.

In an attempt to provide a meaningful answer, a multidisciplinary approach has been employed, involving inquests and inciting public discussion regarding the question “What is my Jerusalem?” as well as a professional mapping of the urban sensitivity based on historical data, building typologies, cultural landscapes, community, and spiritual attributes. To complement these two aspects, we attempted a data-driven approach utilizing computational urban morphometrics methods to cluster the city into urban archetypes and perform statistical analyses on them.

Project goals

In this work we aim to make the computational tools available within the field of urban morphometrics to non-programmer professionals.

We aim to produce an application that accepts geographic data of a city either from file or from an online source, computes morphometrics that characterize the city, utilizes them to cluster the city into urban archetypes, and produces a comprehensive statistical analysis of each archetype.

Partner institutions

Michael Cidor, Yaara Rosner – The Jerusalem Development Authority

Yair Grinberger – Department of Geography, Hebrew University, Jerusalem, Israel

Alessandro Venerandini, Sergio Porta – Architecture department, University of Strathclyde, Glasgow, Ireland

Urban morphometrics

Urban morphometrics is the field of studies that focuses on the quantitative analysis of urban forms. The field focuses on the study of the physical form of the patterns of streets, plots, and buildings, called together the "urban grain"[2].

The Momepy project

Momepy[2] is a python library designed for the study of urban morphometrics. It is part of the Python Spatial Analysis Library (PySAL)[3], and builds upon other PySAL modules, as well as GeoPandas[4]c and networkX[5].

Momepy allows vast amounts of functionality, including measuring the dimensions and quantifying the shapes of geospatial geometries, capturing the spatial distribution and density of multiple kinds of elements, and tessellating the urban space while taking the morphological relations into consideration.

Geographic Information Systems (GIS)

Geographic Information Systems (GIS) is an umbrella term for multiple integrated hardware and software systems that deal with all aspects of geographic data, from storage and analysis to visualization.

In the context of this work, GIS refers to a spatial database (or geodatabase) that represents different elements of an urban settlement. More specifically we mean data, either open-sourced or private databases, that can be interacted with by means of the GeoPandas python module.

Data sources:

The geodatabases utilized can be exported from common GIS software in use in the architecture and urban planning communities, or from open sources.

GIS export - .gdb and .shp files

Standard GIS software can export data in multiple formats, between which are .gdb and .shp files. These are a collection of files sharing the same prefix that hold related geospatial data. These files hold vector information on the shape and geometries of elements, along with properties, or attributes of these elements. Collections of the data that represent a specific part of the geographic reality are bunched together and form a "layer".

Open sources – OpenStreetMap[6,7]

Though there are multiple open sources of geographic data, on of the most complete and easier to use is OpenStreetMap (OSM), thus we focus on it alone. OSM is a free, open

geographic database with a unique data structure without the formal concept of the layer, thus allowing for interconnection of thematically diverse data.
The OSM data can be acquired and pre-processed to get data frames representing elements such as buildings and graphs representing networks such as streets.

Map projections and local coordinate reference systems (CRS)

Our world is a three-dimensional (3D) body with an irregular shape called a geoid, while maps and other representations tend to be two-dimensional (2D). In order to accurately and precisely measure locations, distances, and areas in the 2D representations, the coordinate systems utilized have to be tailored to the region in which the measurements are being made. This results in local coordinate reference systems (CRS) for each region of interest which, though very accurate for their specific locations, could result in extremely inaccurate measurements in other locations on the planet.
There are thousands of such local CRS, some of which have been standardized in international specifications. Coordinates in different CRSs can be transformed from one to another, if their specifications are known. The GeoPandas python module can perform these operations between standardized CRS.

Morphometrics

Tessellations

Tessellation is the process of covering a whole surface by geometric shapes called “tiles” without any overlaps or gaps. This procedure can be performed by the Momepy library in two ways. The first method of “morphological tessellation” divides the whole space such that a single building is present inside each tessellation tile. The second method of “enclosed tessellation” first divides the surface sections enclosed by streets called “enclosures”, and then performs the morphological tessellation within each enclosure. While the morphological tessellation is a simpler procedure, the enclosed tessellation usually results in tiles that better represent the actual morphology of the urban landscape since, e.g. parks get their own tile instead of being divided by all the buildings that surround it.

Metric calculation

After creating the tessellation of the urban landscape, the Momepy library is capable of computing multiple metrics for all the present elements – the buildings, tessellation tiles, streets, and street junctions.
These range from relatively simple metrics such as the area of a building, the coverage of a tessellation cell by its building, and the linearity of a street, to more complex metrics taking

into account spatial distribution such as the length of shared walls between buildings, the interbuilding distances, and the width of street segments. Even more complex relational and connectivity analysis are possible, such as how well is a street segment connected within the city, and the degree into which street networks are aligned into a mesh.
All the metrics computed and utilized in our application are available in the metrics glossary at the end of this report.

Clustering

Using the computed metrics, we then proceed to classify the buildings into a number of clusters which, when performed correctly, should represent distinct urban archetypes.
The clustering procedure is performed automatically, but requires the number of clusters as an input. The choice of the number of clusters has a dramatic influence on the quality and properties of the resulting clusters. In order to make this process easier, we implemented a method that recommends the most adequate number of clusters, after which the user is free to choose the number of clusters.

Clustering algorithms

for clustering we used clustergram which supported different clustering algorithms which we could switch between them:

1. **KMeans**: A fast and widely used clustering algorithm that partitions data into k clusters by minimizing within-cluster variance.
2. **GMM (Gaussian Mixture Model)**: A probabilistic clustering model that assumes data is generated from a mixture of several Gaussian distributions, providing more flexibility than KMeans.
3. **Hierarchical Clustering**: Builds a tree-like structure of clusters by either agglomerating or dividing data, useful for visualizing data relationships at different levels.
4. **MiniBatchKMeans**: A scalable variant of KMeans that processes small, random data batches, making it efficient for large datasets.

Choosing optimal number of clusters

For KMeans algorithm we run KMeans multiple times and calculate the Davies-Bouldin (DB) score for each run, then average these scores, which can provide more reliable clustering results due to KMeans' sensitivity to initial centroid placement. Since KMeans is initialized randomly, and for other clustering algorithms we calculate and plot different scores and metrics which may help in choosing the optimal number of clusters, which are:

1. **elbow method:** helps find the optimal number of clusters by plotting the within-cluster sum of squares (WCSS) for different k. The "elbow" point, where the WCSS reduction slows, indicates the best number of clusters.
2. **David Bouldin score:** evaluates clustering by comparing intra-cluster compactness and inter-cluster separation. Lower scores represent better-defined clusters, with minimal overlap between them.
3. **silhouette score:** The Silhouette Score measures how well each point fits within its cluster, ranging from -1 (poor) to 1 (excellent). Higher scores indicate that clusters are cohesive and well-separated.
4. **calinski_harabasz score:** The Calinski-Harabasz Score assesses cluster quality by comparing between-cluster and within-cluster dispersion. Higher scores indicate well-separated, compact clusters.

A lot of times you may get that the best number of clusters is the minimum number of clusters you asked for which is by default 2, in that occurrence you may need to consider the second, third or fourth best point according to the scores.

Statistical analysis

After classifying the buildings into clusters, we can perform a deeper analysis to gain more insights into the buildings and the metrics we've gathered. The statistical analysis consists of two main parts, cluster level and global level analysis.

Cluster level analysis

For each cluster, we will compute fundamental statistics such as mean, variance, and other key indicators. Additionally, we will calculate the **Flexibility Score**, which measures how much variation exists within a specific metric across the buildings in the cluster. This score helps determine whether a particular metric should be carefully considered in future planning (if it's inflexible) or if there is more freedom for variation. We will also calculate the **Variance Inflation Factor (VIF)**, which quantifies how much a specific metric is influenced by other metrics, helping to identify multicollinearity. Moreover, we will identify **exceptional buildings, or outliers**, which are buildings in the cluster that significantly differ from most of the others.

Global level analysis

In the second part, we focus on understanding the data and classification from a global

perspective. Using a statistical model, we will identify the metrics that have the greatest influence on the classification, using a feature importance method to highlight which variables are driving the clustering process.

Bibliography

1. Data Science for Social Good. *CIDR* https://cidr.huji.ac.il/en/data-science-for-social-good/.
2. Fleischmann, M. momepy: Urban Morphology Measuring Toolkit. *J. Open Source Softw.* **4**, 1807 (2019).
3. Rey, S. J. & Anselin, L. PySAL: A Python Library of Spatial Analytical Methods. *Rev. Reg. Stud.* **37**, 5–27 (2007).
4. Jordahl, K. *et al.* geopandas/geopandas: v0.8.1. Zenodo https://doi.org/10.5281/zenodo.3946761 (2020).
5. Proceedings of the Python in Science Conference (SciPy): Exploring Network Structure, Dynamics, and Function using NetworkX. http://conference.scipy.org.s3-website-us-east-1.amazonaws.com/proceedings/SciPy2008/paper_2/.
6. OpenStreetMap. *OpenStreetMap* https://www.openstreetmap.org/.
7. Planet OSM. https://planet.openstreetmap.org/.
8. Vanderhaegen, S. & Canters, F. Mapping urban form and function at city block level using spatial metrics. *Landsc. Urban Plan.* **167**, 399–409 (2017).
9. Dibble, J. *et al.* On the origin of spaces: Morphometric foundations of urban form evolution. *Environ. Plan. B Urban Anal. City Sci.* **46**, 707–730 (2019).
10. Schirmer, P. M. & Axhausen, K. W. A multiscale classification of urban morphology. *J. Transp. Land Use* **9**, (2016).
11. Gil, J., Beirão, J. N., Montenegro, N. & Duarte, J. P. On the discovery of urban typologies: data mining the many dimensions of urban form. *Urban Morphol.* **16**, 27–40 (2012).
12. Basaraner, M. & Cetinkaya, S. Performance of shape indices and classification schemes for characterising perceptual shape complexity of building footprints in GIS. *Int. J. Geogr. Inf. Sci.* **31**, 1952–1977 (2017).

13. McGarigal, K. & Marks, B. J. FRAGSTATS: spatial pattern analysis program for quantifying landscape structure. *Gen Tech Rep PNW-GTR-351 Portland US Dep. Agric. For. Serv. Pac. Northwest Res. Stn. 122 P* (2022) doi:10.2737/PNW-GTR-351.
14. Hamaina, R., Leduc, T. & Moreau, G. Towards Urban Fabrics Characterization Based on Buildings Footprints. in *Bridging the Geographic Information Sciences: International AGILE'2012 Conference, Avignon (France), April, 24-27, 2012* (eds. Gensel, J., Josselin, D. & Vandenbroucke, D.) 327–346 (Springer, Berlin, Heidelberg, 2012). doi:10.1007/978-3-642-29063-3_18.
15. Feliciotti, A. Resilience and urban design : a systems approach to the study of resilience in urban form : learning from the case of the Gorbals.
16. Araldi, A. & Fusco, G. From the street to the metropolitan region: Pedestrian perspective in urban fabric analysis. *Environ. Plan. B Urban Anal. City Sci.* **46**, 1243–1263 (2019).
17. Bourdic, L., Salat, S. & Nowacki, C. Assessing cities: a new system of cross-scale spatial indicators. *Build. Res. Inf.* **40**, 592–605 (2012).

Appendixes

Privacy disclaimer

Our application utilizes a web app built with the **streamlit** library. The app runs completely on the user's local machine, with the only connection to the network being the possibility to fetch data from OpenStreetMaps (OSM).
Except for the fetching data from OSM, the app is fully utilizable without any internet connection. No data is intentionally set to any 3rd party.
Despite the above statement, the developers take no responsibility for the security or privacy of the application.

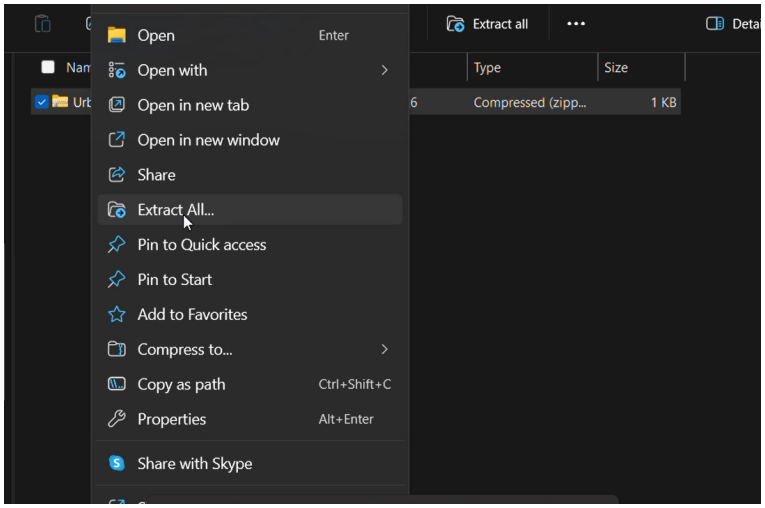
ClusterOn - User manual

Installation

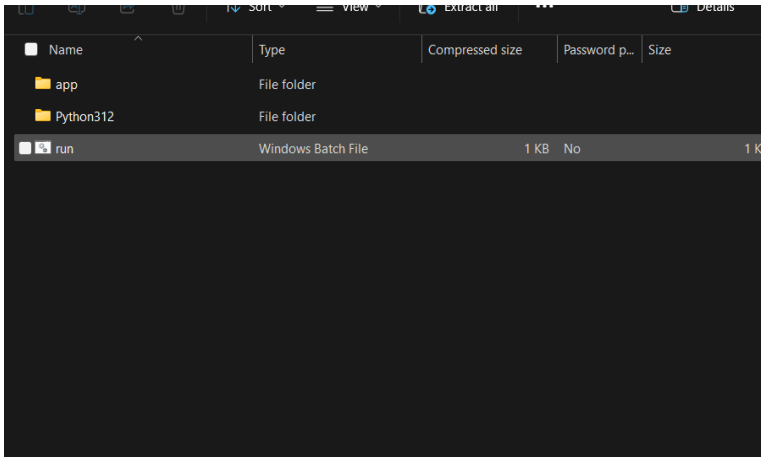
The Application is available on the github repository https://github.com/amirkl91/ClusterOn
For windows (10/11) the application files are available at
https://drive.google.com/drive/folders/1H_L4Z8-4u4spILFSY_iYLom680190FbZ?usp=drive_link

On Windows (10/11)

download the latest version of the ClusterOnx64 zip file and extract it (right click on the file then extract all)

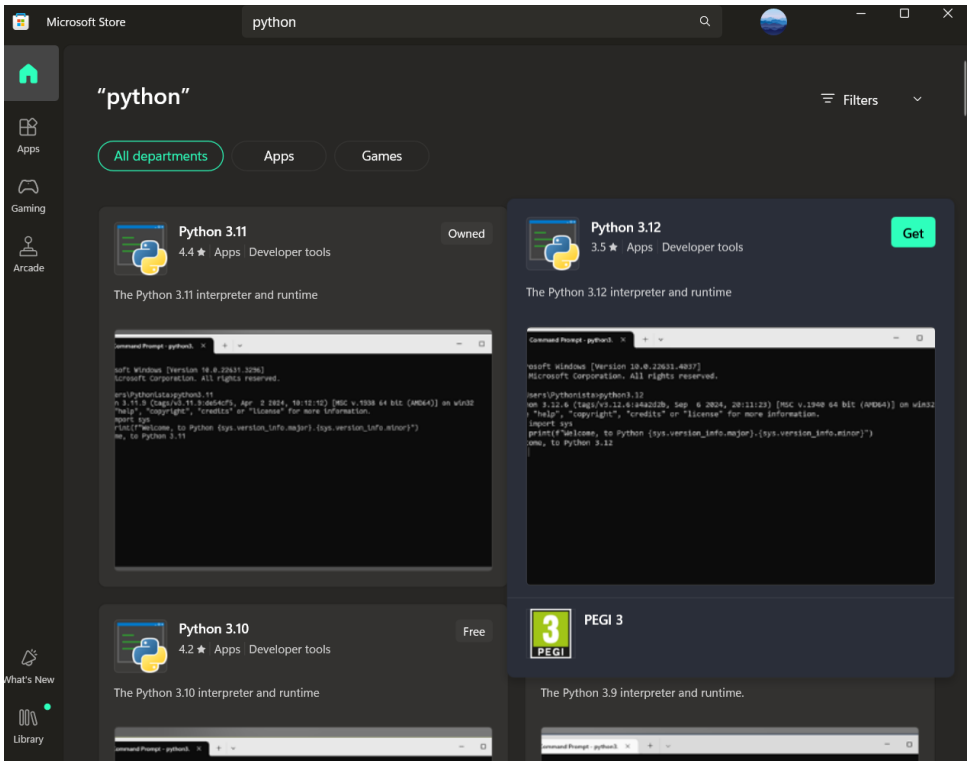


click on “run” then allow

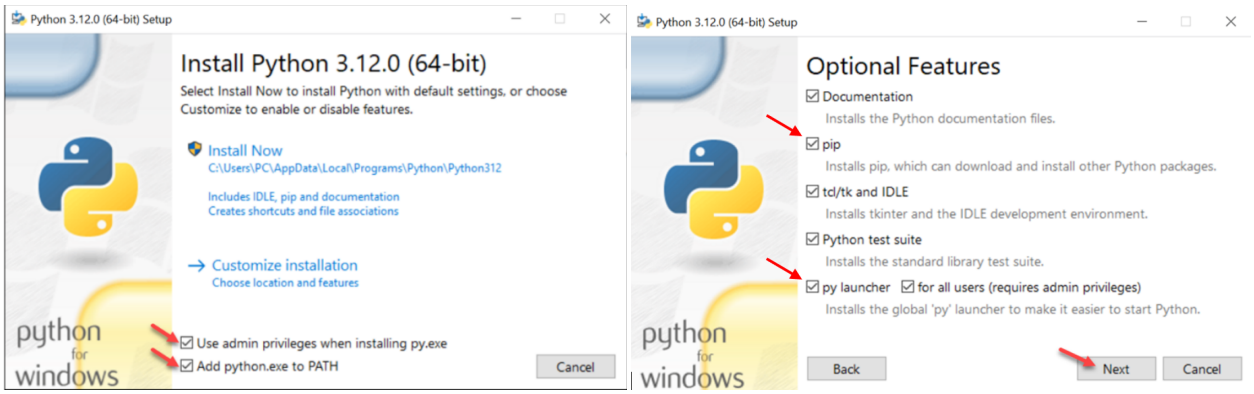


For Python users (any OS)

- install python3.12 from https://www.python.org/downloads/ or from Microsoft store
  Option 1: from Microsoft store (Windows)
  search for python3.12 and install it



Option 2: from the official python website https://www.python.org/downloads/
you can follow the tutorial for installing python here , but make sure to mark the following options during the installation



- install the ClusterOn zip file and extract it
- run the program in one of these ways:
    - (Only on windows) double click on “run”
    - Through the command line (Terminal)
        1. **Step 1:** Open your terminal (Command Prompt on Windows, Terminal on macOS/Linux).
        2. **Step 2:** Navigate to the directory where your Python file is located using the cd (change directory) command

```
cd path_to_ClusterOn_folder/app
```

        3. **Step 3:** Run your Python program by typing:

```
python streamlit_app.py
```

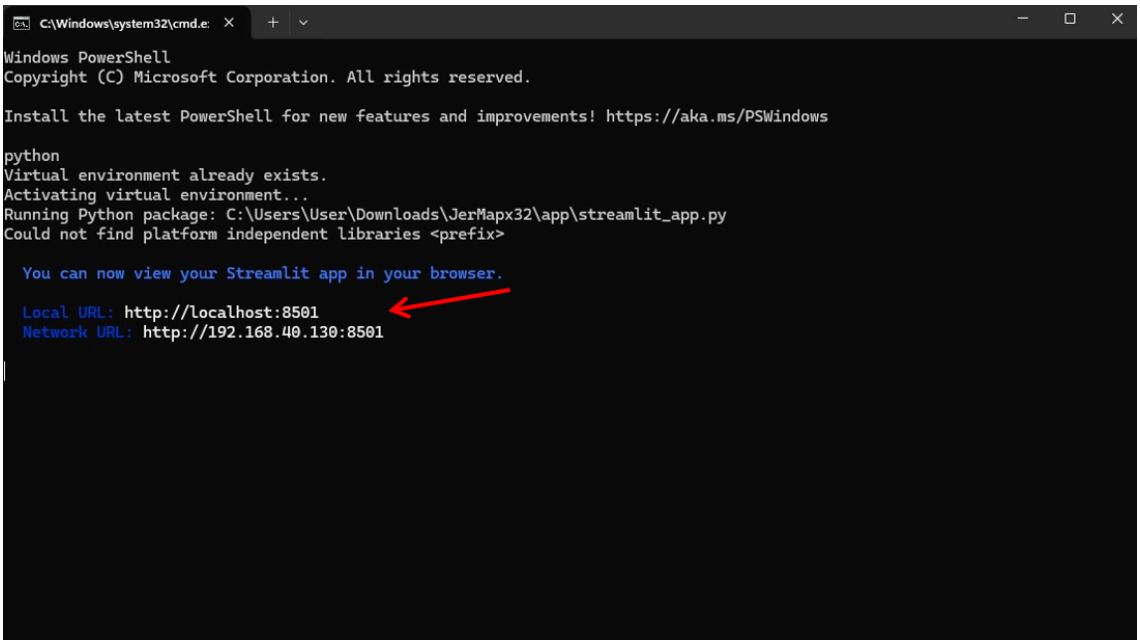
        4. Alternatively, you might need to use python3 or py:

```
Python3 streamlit_app.py
```

        5.

Launch the UI

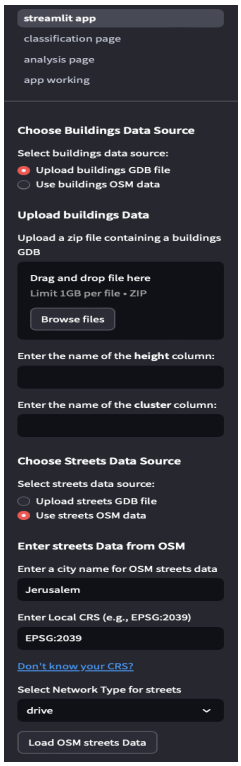
- In General the UI should open a webpage on the browser automatically
- if you run the program for the first time you may get a message in the command line to write your email, if you get it just press enter
- if the UI didn’t launch automatically, copy the URL in the terminal and paste it in the browser (you can use either Local or Network URL, both should work)


C:\Windows\system32\cmd.e:
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.
Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows
python
Virtual environment already exists.
Activating virtual environment...
Running Python package: C:\Users\User\Downloads\JerMapx32\app\streamlit_app.py
Could not find platform independent libraries <prefix>
You can now view your Streamlit app in your browser.
Local URL: http://localhost:8501
Network URL: http://192.168.40.130:8501

Manual

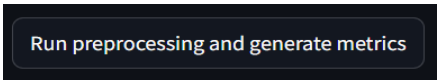
Part 1: Uploading the data

1. Go to the "streamlit app" page
   In the top left corner
2. Below, you will need to upload your buildings data. You can choose between uploading a file from your local directory or retrieve data from Open Streets Maps.
   If you want to use a .gdb file - **Upload a Zip file** containing your GeoDatabase (.gdb file).
   If your data contains building heights, please add the name of the **height** column. If none is specified, By default the column name will be "height", and if there isn't any height column, the metrics requiring buildings heights will be ignored.
3. Do the same for the streets data, if you choose to use data from Open Streets Maps, you can choose between the network type: drive, walk or bike.

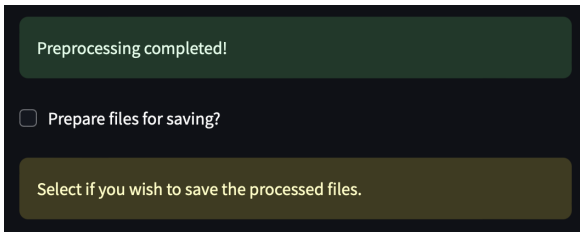


4. (Optional) After you have loaded the data, in the middle of the page you can choose which metrics you want to use for the preprocessing and the analysis of the data. Explanation over each metric is attached to the final report.

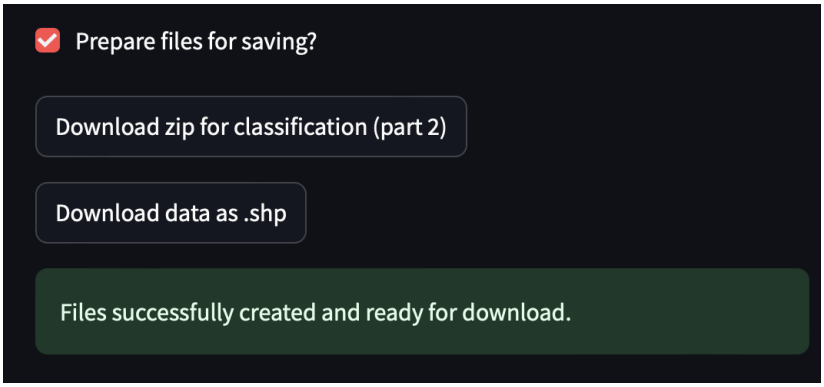
5. Finally, press on "run preprocessing and generate metrics". This will load the data, verify it, and calculate the metrics.



6. If the process has been completed successfully, you will get a green message.



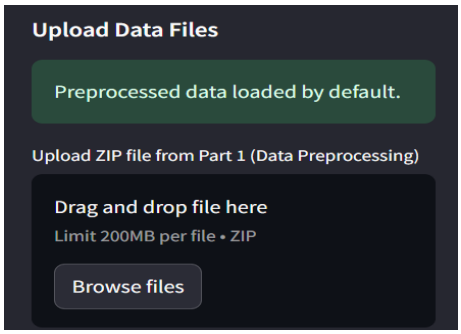
7. Once you select Prepare files for saving you will get options to download the .zip and .shp files:
   CAUTION! This step makes the app slower! Only do it when you want to download files, and after downloading un-select the checkbox.



8. You can see some of the metrics plotted on top of the map on the right side of the window. Just choose the desired data type (buildings or streets) and the requested metric.

Part 2: Classification of City Textures

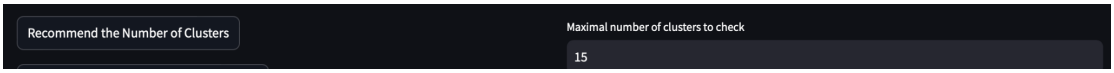
In this section, the application will analyze your data and classify it into various textures. If you have just finished preprocessing your data in Part 1, it will automatically be loaded for further processing. If not, you can easily upload your data using the options on the left side of the page. Since we have already loaded the data from the previous step, you will see a green message confirming that the data is being loaded by default.



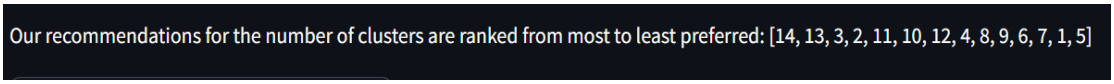
1. Optional – In the middle of the page, if you want, you can get a recommendation for the optimal number of clusters to divide the city.

   CAUTION! This step may take a while to complete!

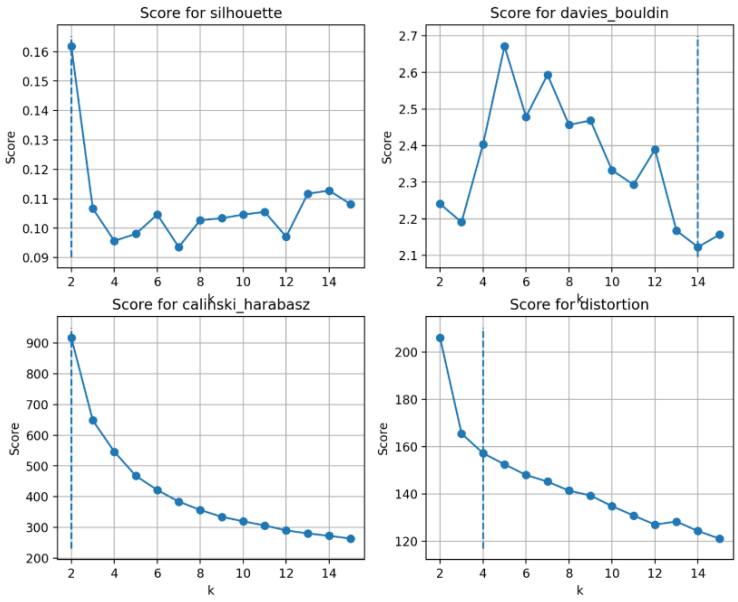
   1. You can modify the maximal number of clusters to check (defaults to 15).



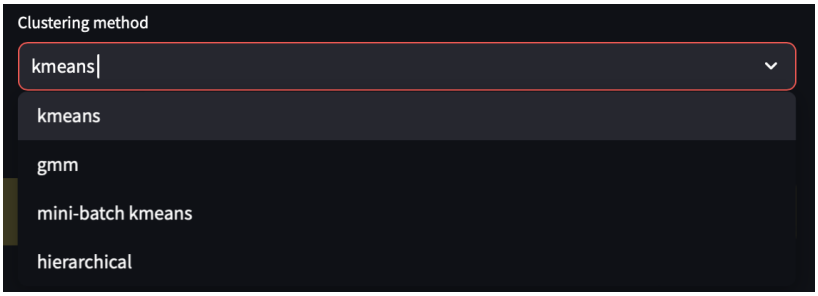
   2. You will get a recommendation for the optimal amount of clusters, with the 1st number (on the left) being the most likely and the last number (on the right) the least likely.



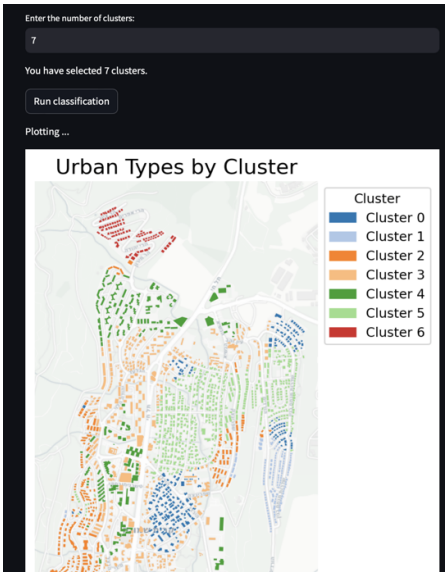
2. Optional –you can plot the metrics used to determine the recommendations



3. You can choose the clustering method to calculate the clusters with:



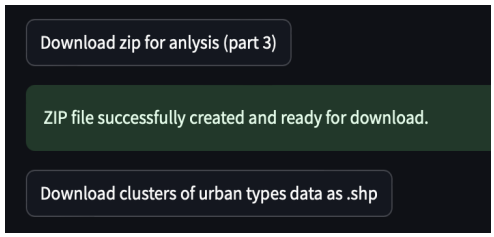
4. Finally, choose the desired number of clusters and press "Run classification". A plot of the buildings in the city divided into the clusters will appear.



5. Step 5 can be repeated with different values for the number of clusters until satisfactory classification is achieved.

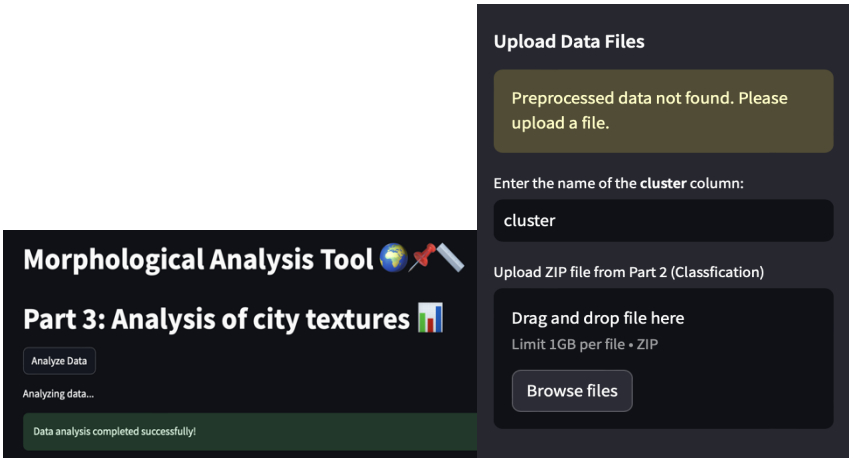
   - Warning! The classification has a random element. Reclassifying multiple times with the same number of clusters **may** result in some differences in clustering result.
   - Warning! The order and color of the clusters is random. Reclustering with the same number of clusters **will** result in different cluster order and color.

6. You can download the results for the next part or save a .shp file.

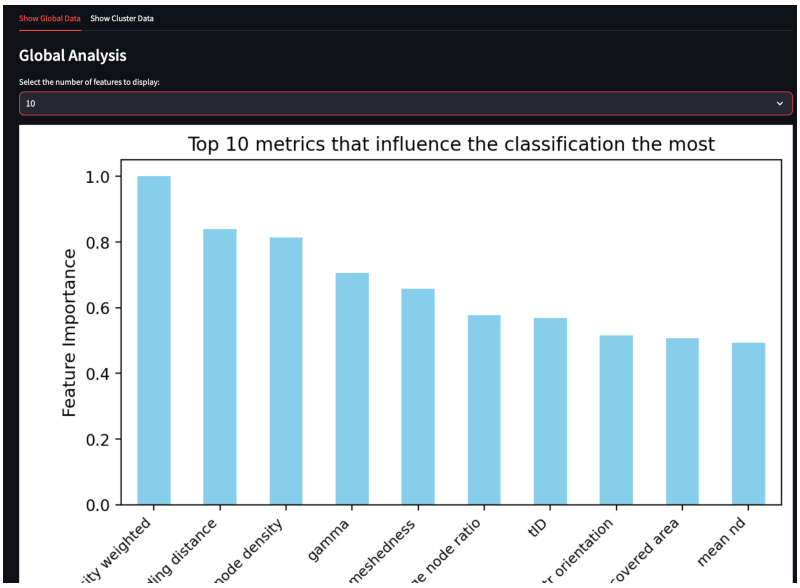


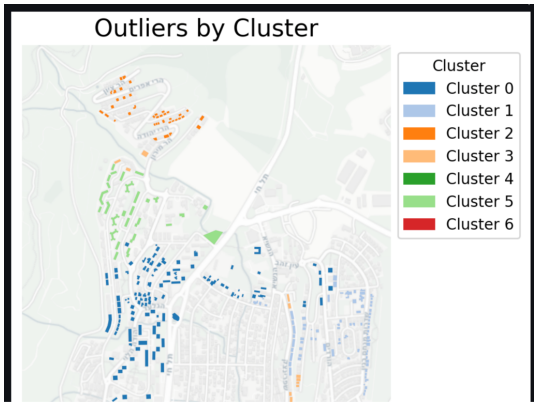
Part 3: Analysis of city textures

1. As before, you can either continue working with the data from Part 2 or upload the data you downloaded from Part 2
2. Then you can run analyze data to compute statistics



3. You can choose the “Global data” display to explore and analyze the entire dataset:

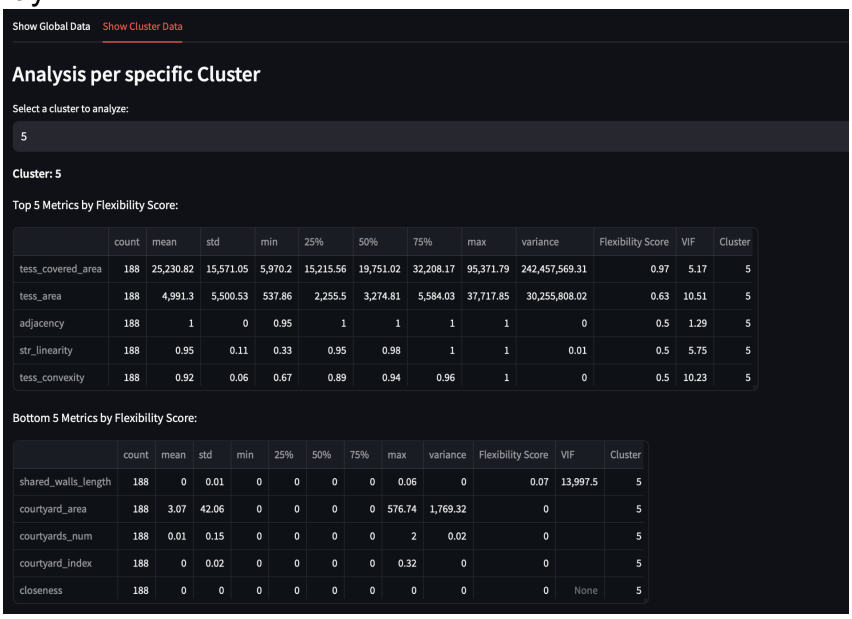




Overall statistics

	count	mean	std	min	25%	50%	75%	max	variance	Flexibility Score	VIF
tess_covered_area	2,140	12,048.32	9,787.18	2,263.96	5,934.52	8,920.06	15,042.48	101,521.75	95,788,857.75	0.98	4.37
tess_area	2,140	2,018.96	2,473.53	166.54	862.46	1,299.59	2,236.02	37,717.85	6,118,351.82	0.6	4.82
tID	2,140	1,069.5	617.91	0	534.75	1,069.5	1,604.25	2,139	381,811.67	0.52	1.67
adjacency	2,140	1	0	0.95	1	1	1	1	0	0.5	1.05
tess_convexity	2,140	0.94	0.05	0.52	0.92	0.95	0.97	1	0	0.5	3.88
tess_equivalent_rect_index	2,140	0.99	0.05	0.6	0.96	0.99	1.02	1.1	0	0.5	5.19
equivalent_rectangular_index	2,140	0.95	0.08	0.38	0.93	1	1	1.12	0.01	0.5	110.07
rectangularity	2,140	0.92	0.13	0.29	0.85	1	1	1	0.02	0.5	35.25
square_compactness	2,140	0.86	0.15	0.14	0.78	0.92	0.98	1.26	0.02	0.5	248.02
shape_index	2,140	0.73	0.08	0.37	0.69	0.75	0.79	0.99	0.01	0.5	1,807.98

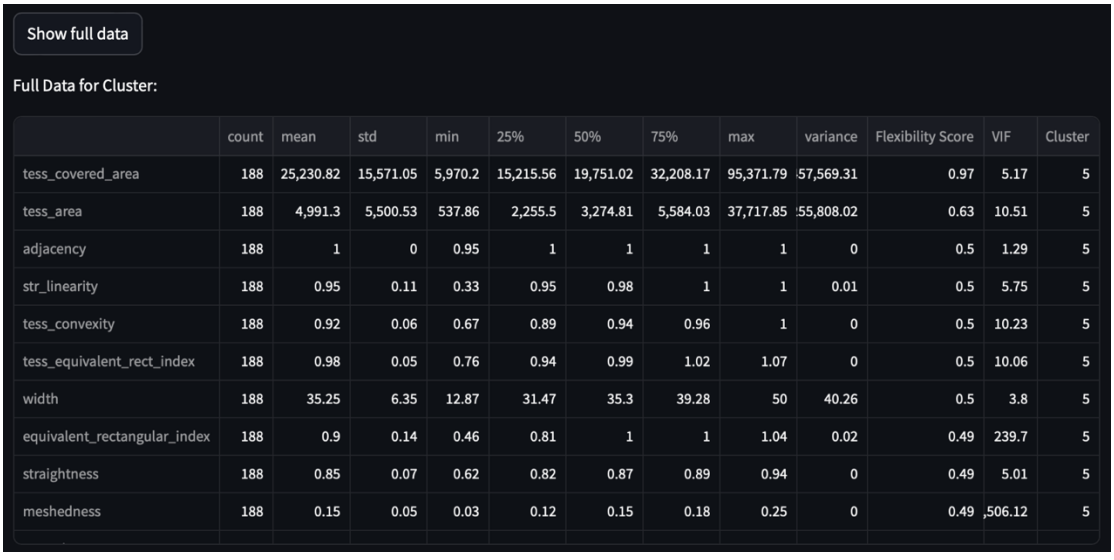
4. Alternatively, you can focus on a specific cluster and explore it using the “Show Cluster” display –



	count	mean	std	min	25%	50%	75%	max	variance	Flexibility Score	VIF	Cluster
tess_covered_area	188	25,230.82	15,571.05	5,970.2	15,215.56	19,751.02	32,208.17	95,371.79	242,457,569.31	0.97	5.17	5
tess_area	188	4,991.3	5,500.53	537.86	2,255.5	3,274.81	5,584.03	37,717.85	30,255,808.02	0.63	10.51	5
adjacency	188	1	0	0.95	1	1	1	1	0	0.5	1.29	5
str_linearity	188	0.95	0.11	0.33	0.95	0.98	1	1	0.01	0.5	5.75	5
tess_convexity	188	0.92	0.06	0.67	0.89	0.94	0.96	1	0	0.5	10.23	5

	count	mean	std	min	25%	50%	75%	max	variance	Flexibility Score	VIF	Cluster
shared_walls_length	188	0	0.01	0	0	0	0	0.06	0	0.07	13,997.5	5
courtyard_area	188	3.07	42.06	0	0	0	0	576.74	1,769.32	0		5
courtyards_num	188	0.01	0.15	0	0	0	0	2	0.02	0		5
courtyard_index	188	0	0.02	0	0	0	0	0.32	0	0		5
closeness	188	0	0	0	0	0	0	0	0	0	None	5

5. Choose the desired cluster number and get the analyzed data.
   5.1. If you want to show the full metrics table – press ‘Show full data’



	count	mean	std	min	25%	50%	75%	max	variance	Flexibility Score	VIF	Cluster
tess_covered_area	188	25,230.82	15,571.05	5,970.2	15,215.56	19,751.02	32,208.17	95,371.79	57,569.31	0.97	5.17	5
tess_area	188	4,991.3	5,500.53	537.86	2,255.5	3,274.81	5,584.03	37,717.85	55,808.02	0.63	10.51	5
adjacency	188	1	0	0.95	1	1	1	1	0	0.5	1.29	5
str_linearity	188	0.95	0.11	0.33	0.95	0.98	1	1	0.01	0.5	5.75	5
tess_convexity	188	0.92	0.06	0.67	0.89	0.94	0.96	1	0	0.5	10.23	5
tess_equivalent_rect_index	188	0.98	0.05	0.76	0.94	0.99	1.02	1.07	0	0.5	10.06	5
width	188	35.25	6.35	12.87	31.47	35.3	39.28	50	40.26	0.5	3.8	5
equivalent_rectangular_index	188	0.9	0.14	0.46	0.81	1	1	1.04	0.02	0.49	239.7	5
straightness	188	0.85	0.07	0.62	0.82	0.87	0.89	0.94	0	0.49	5.01	5
meshedness	188	0.15	0.05	0.03	0.12	0.15	0.18	0.25	0	0.49	,506.12	5

6. Following is an explanation of the performed statistical:
   6.1. **Count**: how many buildings (in our case) there are in the cluster.
   6.2. **Mean**: the mean value of the buildings under a specific metric.
   6.3. **Std**: standard deviation, how much a building under a specific metric is tend to be close or far from the mean value of that metric.
   6.4. **Min**: minimum value of that metric.

6.5. **25%, 50%, 75%**: these are the quantiles, they represent the values of the buildings that are bigger then 25% of the buildings, 50% of the buildings (the median) and 75% of the building in relation to the metric.
6.6. **Max**: maximum value of that metric.
6.7. **Variance**: the variance of the metric, how the buildings under that metric are "spreading".
6.8. **Flexibility score**: this score is calculated by combining the variance and weighted entropy of each metric. It measures how "flexible" or "variable" a metric is within a cluster. The higher the score, the more this metrics is playable in future planning.
6.9. **VIF**: variance inflation factor, measures how much a specific metric is influenced by other metrics. A high VIF value means that this metric is strongly dependent on other metrics. In other words, it shows how much the metric's value will change if other metrics in the model also change. A higher VIF indicates a higher risk of multicollinearity, meaning the metric doesn't provide unique information.
6.10. **Feature importance**: Key metrics that influence the classification of the data into clusters the most. A leading metric will typically show significant variation across different clusters, meaning it might have low values in one cluster and high values in another.

Metrics glossary

- **Building metrics**
  - **Adjacency**[8] - How much a building tend to join together with its neighbors
  - **Alignment** - Assesses how similar the orientation of a building is to its neighboring buildings
  - **Area** - Total area of the building.
  - **Circular compactness**[9] - How much the shape resembles a circle, how well the shape fills space with minimal perimeter.
  - **Corners** - Calculate the number of corners in the building.
  - **Courtyard area** - The area of the courtyard within the shape.
  - **Courtyard index**[10] - Courtyard area divided by the total area of the shape.
  - **Courtyards num**[10] - Number of courtyards within a structure.
  - **Elongation**[11] - Measures how long or stretched the shape is.
  - **Equivalent rectangular index**[12] - Measures how close the building to a rectangular.
  - **Facade ratio**[10] - Area divided by the perimeter.
  - **Fractal dimension**[13] - Measures how much irregularity the boundary of the shape has.
  - **Longest axis length** - Diameter of minimal bounding circle around the geometry.
  - **Mean interbuilding distance** - Mean distance between a shape and its adjacent buildings defined by an adjacency graph
  - **Neighbor distance** - Mean distance between a shape and its neighbors.
  - **Orientation**[10] - orientation of a shape in relation to the cardinal directions.
  - **Perimeter** - The total length of the boundary of the shape.
  - **Perimeter wall** - The total length of the boundary of the shape.
  - **Rectangularity**[9] - Area devided by the area of minimum bounded rectangle
  - **Shape index** - How much the shape is compact or elongated ("streched").
  - **Shared walls length**[14] - Calculates the length of the shared walls of the object.
  - **Square compactness**[15] - How much the shape resemples a square.
  - **Squareness** - How much the shape resembles a square or rectangular by evaluating the angles.
  - **Street alignment** - Measures the deviation of the building orientation from the street orientation
  - **Weighted axis compactness** - Measures the compactness-weighted axis, how the shape is compact.
- **Tessellation metrics**
  - **tess alignment** - How the tessellation is aligned with its tessellation neighbors.
  - **tess area** - Area of the tesselation.
  - **tess build area ratio** - Building area divided by the tesselation area.

- **tess building orientation** - Orientation of the tesselation minus the orientation of the building.
- **tess circ compactness**[9] - How much the tesselation resembles a circle.
- **tess convexity** - Area of the tesselation devided by the area of the convexed hull.
- **tess covered area** - Calculates the area covered by neighbours.
- **tess equivalent rect index**[12] - How the tessellation resembles a rectangle.
- **tess neighbors** - Number of tesselation neighbors.
- **tess orientation**[10] - Orientation in relation to the cardinal directions.
- **Meanlen** - Mean of the length of the streets of a subgraph around the junction.

- **Street metrics**
  - **Openness**[16] - How much the street feels "open".
  - **str length** - Calculates the length of the street.
  - **str linearity**[16] - Measures the linearity of the street.
  - **str longest axis** - Length of the longest axis.
  - **str orientation** - Orientation in relation to the cardinal directions.
  - **Width** - Calculates the width of the street.
  - **Width dev** - Measures the standart deviation of the width.
- **Junction metrics**
  - **Closeness** - How close the junction is to all other junctions.
  - **Cyclomatic**[17] - Calculates cyclomatic complexity (number of loops) around each junction.
  - **Degree** - Degree of the junction.
  - **Edge node ratio**[9] - Calculate the ratio of streets to junctions in a subgraph around the junction.
  - **Gamma**[9] - Calculates connectivity gamma index for subgraph around the junction.
  - **Mean nd** - Mean node degree of a subgraph around the junction.
  - **Meshedness**[15] - Measures how "meshed" or interconnected the graph around the junction is.
  - **Node density**[9] - Number of neighboring junctions divided by the cumulative street length.
  - **Node density weighted**[9] - Number of neighboring junctions degree minus one, divided by the cumulative street length.
  - **Straightness**[9] - How straight the streets are from the junction to its neighbors.