POLITECNICO DI MILANO
Scuola di Ingegneria Industriale e dell'Informazione
Corso di Laurea Magistrale in Ingegneria Informatica

POLITECNICO
MILANO 1863

Integrating Machine Learning techniques into GIS software: Development of a comprehensive and versatile QGIS plugin for cluster analysis on geospatial data

Relatore:
Prof. Letizia Tanca
Correlatori:
Dott. Emilia Lenzi
Dott. Carlo Andrea Biraghi

Tesi di Laurea Magistrale di:
Andrea Folini, 920286

Anno Accademico 2020-2021

Acknowledgements

Innanzitutto grazie alla professoressa Letizia Tanca per la possibilità di lavorare a questo progetto, e per la professionalità dimostrata dall'inizio alla fine del lavoro. Grazie, inoltre, ai dottori Emilia Lenzi e Carlo Biraghi per l'aiuto fondamentale e l'immensa disponibilità, costanti per tutto il percorso di sviluppo, e senza i quali il completamento di questo lavoro non sarebbe stato possibile. Un ulteriore ringraziamento per avermi introdotto e guidato in questo appassionante, e per me nuovo, ambito di ricerca.

Grazie a tutta la mia famiglia per il supporto e l'affetto, dimostrati giorno per giorno, e per aver sempre creduto in me e nelle mie capacità. Soprattutto, un ringraziamento speciale ai miei genitori, che nonostante le difficoltà non mi hanno mai fatto mancare niente e hanno sempre assecondato le scelte personali, e sono il principale motivo per cui sono arrivato in fondo a questo lungo percorso.

Grazie, infine, a tutti gli amici, da chi è presente dai tempi delle elementari e con cui condivido con piacere la mia vita da ormai vent'anni, a chi ho conosciuto in questi ultimi anni di università, sia compagni di studio che di appartamento, con cui ho formato un legame altrettanto importante e profondo.

Come sapete, non mi è facile esprimere la mia gratitudine a parole, ma spero che queste poche righe possano far trasparire almeno in parte l'immensa riconoscenza per tutte le persone che mi sono vicine.

Abstract

As geospatial data continuously grows in complexity and size, the application of Machine Learning and Data Mining techniques to geospatial analysis is increasingly more essential to solve real world problems. Although, in the last two decades, the research in this field produced innovative methodologies, they are usually applied to specific situations and not automatized for general use. Therefore, both generalization and integration of these methods with Geographic Information Systems are necessary to support researchers and organizations in data exploration, pattern recognition, and prediction in the various applications of geospatial data. The lack of machine learning tools in GIS is especially clear for what concerns unsupervised learning and clustering. In this work we present a plugin, ready to be published, that we developed for the open-source software QGIS and offers functionalities for the entire cluster analysis process: from (i) pre-processing, to (ii) feature selection and clustering, and finally (iii) cluster evaluation. Our tool provides different improvements from the current solutions available in QGIS, but also in other widespread GIS. The expanded features provided by the plugin allow the users to deal with some of the most challenging problems of geospatial data, such as high dimensional space, poor quality of data, and large size of data. Another important objective of the research is the accessibility and ease of use of the plugin, since the general user of GIS is often lacking a machine learning and computer science background. To assess the strengths and weaknesses of the program, we will cover numerous experiments with real world situations on data from the city of Milan. The datasets for the experiments are of different nature (i.e., climatic, urban, and socio-demographic) and different sizes, ranging from less than 100 data points to almost 70000, and with a large number of numerical attributes, up to 109. Overall, the experimental phase shows good and adequate flexibility of the plugin, and outlines the possibilities for future developments that can be provided also by the QGIS community, given the open-source nature of the project.

Sommario

Con la continua crescita delle dimensioni e della complessità dei dati geospaziali, l'applicazione delle tecniche di Machine Learning e Data Mining all'analisi spaziale sta acquisendo un ruolo sempre più centrale nella risoluzione dei problemi reali. Nonostante, negli ultime due decenni, le ricerche in questo campo abbiano sviluppato metodologie innovative, queste vengono solitamente applicate a contesti specifici e non sono automatizzate per l'uso generale. Perciò, la generalizzazione e l'integrazione di questi metodi con i sistemi informativi geografici sono necessarie per sostenere i ricercatori e le organizzazioni nell'esplorazione dei dati, il riconoscimento di pattern, e la predizione nelle varie applicazioni dei dati geospaziali. La mancanza di strumenti per il machine learning nei GIS è evidente soprattutto per quanto riguarda l'apprendimento non supervisionato e il clustering. In questo lavoro presentiamo un plugin sviluppato per il software open-source QGIS, che include funzionalità per l'intero processo di clustering: dalla (i) pre-elaborazione, alla (ii) selezione delle caratteristiche e il clustering, e infine la (iii) valutazione del clustering. Il nostro programma introduce diverse innovazioni rispetto alle soluzioni presenti attualmente in QGIS e altri GIS molto diffusi. Le funzioni aggiuntive del plugin permettono agli utenti di affrontare le problematiche più comuni dei dati geospaziali, come l'alto numero di dimensioni, la scarsa qualità, e la grandezza dei dati. Altri obiettivi rilevanti del nostro lavoro sono l'accessibilità e la facilità di utilizzo, in quanto gli utenti dei GIS spesso non hanno molte conoscenze informatiche e di machine learning. Per valutare i punti di forza e di debolezza del programma, mostreremo numerosi esperimenti con situazioni reali su dati della città di Milano. I dati utilizzati per gli esperimenti sono di diversa natura (climatici, urbani e socio demografici) e di diverse grandezze, passando da meno di 100 osservazioni a quasi 70000, e con un ampio numero di attributi, fino a 109. Complessivamente, la fase sperimentale mostra l'ottima flessibilità del plugin, e delinea le possibilità per gli sviluppi futuri che possono essere apportati anche dalla comunità di QGIS, vista la natura open-source del progetto.

Contents

List of Figures

List of Tables

List of Algorithms

List of Acronyms

ML	Machine Learning
GIS	Geographic Information System
SIMBA	Systematic clusterIng-based methodology to support Built environment Analysis
IMM	Integrated Modification Methodology
CAS	Complex Adaptive System
KC	Key Categories
BED	Built Environvment Decomposition
FLC	First Level Clustering
SLC	Second Level Clustering
CRS	Coordinate Reference System
API	Application Programming Interface
AI	Artificial Intelligence
SSE	Sum of Squared Errors
WSS	Within clusters Sum of Squares
BSS	Between clusters Sum of Squares
CDS	Climate Data Source
NetCDF	Network Common Data Form
ACE	Area di Censimento
NIL	Nucleo di Identità Locale
PGT	Piano di Governo del Territorio
GUI	Graphical User Interface

LCZ	Local Climate Zones

Chapter 1

Introduction

With the exponential growth of the use of computer science in the last decades, the analysis of data and information is playing an increasingly critical role in every aspect of society. A particular type of data which is of great interest for a wide range of stakeholders, such as businesses or public organizations, is geospatial data, which combines location information (usually coordinates on earth), attribute information, and often temporal information. Geospatial data can be used to solve a large variety of real-world problems, from studies on customers to epidemiology, natural disasters, and urban planning. The software systems to perform spatial analysis on this kind of data are called Geographic Information Systems (GIS). GIS tools are essential to uncover meaning and insight in geospatial data with the creation and visualization of maps, graphs, and statistics. As geospatial data complexity, variety and volume increased over the years, the analysis required more and more advanced methods. To overcome these challenges, researchers started to apply the concepts of Machine Learning to spatial analysis and GIS to filter, interpret and predict information [1]. An example of this application is the recent introduction of the SIMBA system [2] as a support of the Investigation phase of the IMM methodology [3]. IMM is the acronym of Integrated Modification Methodology, an innovative design and analysis methodology for the built environment whose main goal is improving the energy performance of the cities. SIMBA is a clustering-based methodology which supports and systematises the analysis of the built environment. The methodology is divided in three different phases:

- Built Environment Decomposition;
- First Level Clustering;
- Second Level Clustering.

Although SIMBA provides a useful methodology to select a representative and reasonable number of features and to measure the distance between elements in the built environment analysis, it still lacks a certain degree of automation. The automation and complete integration of this methodology within GIS tools are the motivations from which this work starts. The approach that we took to solve this problem consisted in the development of a plugin for QGIS [4], a widespread open-source GIS software.

During the development phase, we noticed the possibility to create a more general tool for cluster analysis that is suited for a wider range of use-cases than SIMBA. By doing this, we also extended the QGIS functionalities, as solutions of this kind are still underdeveloped in the most popular GIS. In particular, the objective of the work is to provide a complete tool for cluster analysis on geospatial data of different natures and different sizes. The final version of the tool we developed is composed by three main parts:

- data pre-processing for dimensionality reduction;
- feature selection and clustering;
- evaluation of the obtained clustering.

Along with the implementation, the research is integrated with a considerable experimental phase, both during and after the development phase. This phase is essential to highlight both the potential of the plugin and its limitations in real world scenarios. The great volume of experiments is conducted on data about the city of Milan, describing social-demographics, urban and climatic characteristics and with different granularities.

The structure of the thesis is the following:

- Chapter 2: provides a description of geospatial data analysis, SIMBA and IMM methodologies and the state of the art of the current solutions for clustering in GIS. Then we define the detailed goals and motivations of the work;
- Chapter 3: introduces the fundamental theoretical concepts about GIS data and plugins, machine learning and data mining relative to the functionalities of our plugin;
- Chapter 4: describes the composition of the datasets used in the experimental phase;
- Chapter 5: explains the details of the implementation of the plugin. Both from a higher level point of view and the detailed functionalities of each section.
- Chapter 6: provides a description of the settings of some of the experiments carried out during and after development;
- Chapter 7: reports the results of the experiments and discusses about the plugin's functionalities for evaluation;
- Chapter 8: first, includes the conclusions of the work and comments about the final version of the plugin. Then, provides some possibilities for future developments and expansion of the features.

Chapter 2

State of the art and goals

In this chapter we will talk about the context and the state of the art for our research work. Finally, we will give the motivation of the research and state the goals that we set. Some theoretic concepts introduced here will be covered in more detail in chapter 3.

2.1 Analysis of geospatial data

Geospatial analysis is a process of Geographical Information Systems data interpretation, exploration, and modelling, from acquisition to understanding results. The retrieved information is computer-processed with spatial analysis software and varies depending on the number of tasks and their complexity. The simplest one is visualization, while a more detailed approach suggests comprehensive analytics with specific tools to elaborate actionable insights. Typically, spatial analysis consists of five key stages: understanding your goal, preparing data, choosing suitable tools and techniques, performing the research, and estimating results [5].

Location, in the form of spatial data, is a key point for visualizing the current location, predicting events, and enhancing service delivery. Information about location can integrate and strengthen the complex analysis of the distribution of locations, events, and services. This provides many opportunities for improving government services in terms of best governmental segments, interacting with customers and optimizing processes. As cities get larger, spatial information becomes like a key tool in efficient urban service delivery, public safety, and overall resource management [6].

In the last years, one of the most interesting and popular fields of geospatial analysis has been the integration of machine learning and data mining techniques with spatial data. These algorithms have been applied to geoprocessing tools to solve problems in three broad categories. With classification, you can use vector machine algorithms to create land-cover classification layers [7, 8]. Another example is clustering, which lets you process large quantities of input point data, identify the meaningful clusters within them, and separate them from the sparse noise [9]. Prediction algorithms, such as geographically weighted regression, gives you the ability to model spatially varying relationships. These methods work well in several areas, and their results are interpretable, but they need experts to identify or feed in those factors (or features) that affect the outcome that we're trying to predict [10].

2.2 IMM and SIMBA

One of the areas of application of ML to spatial analysis is sustainability in building environments. In this field, we can find the research that contains the foundations for the functionalities and some of the experiments in our work. The SIMBA methodology, recently introduced by Emilia Lenzi [2], is a procedure created to support the analysis of built environments; in particular, it has been used as a support tool to the IMM process. In the next pages, we will cover this methodology and its structure, along with the basics of IMM.

2.2.1 IMM

IMM is the acronym of Integrated Modification Methodology [11], an innovative design methodology based on a specific process with the main goal of improving the energy performance of cities. In this methodology, the city is considered as a dynamic Complex Adaptive System comprised of the synergic integration of a number of elementary parts which, through their arrangement and the architecture of their ligands, provide a certain physical and provisional arrangement of the CAS. In IMM the emergence process of interaction between elementary parts to form a synergy is named Key categories. Key categories are the products of the synergy between elementary parts, a new organization that emerges not simply as an additive result of the proprieties of the elementary parts. According to this view, the city is not solely a mere aggregation of disconnected energy consumers, and the total energy consumption of the city is different from the sum of all of the buildings' consumption. This considerable gap between the total energy consumption of the city and the sum of all consumers is concealed from the urban morphology and urban form of the city. The IMM investigates the relationships between urban morphology energy consumption and environmental performances by focusing mostly on the 'Subsystems' characterized by physical characters and arrangements. The main object of this design process is to address a more sustainable and better performing urban arrangement. IMM methodology is based on a multi-stage process composed of four different but fully integrated phases, as shown in figure 2.1.

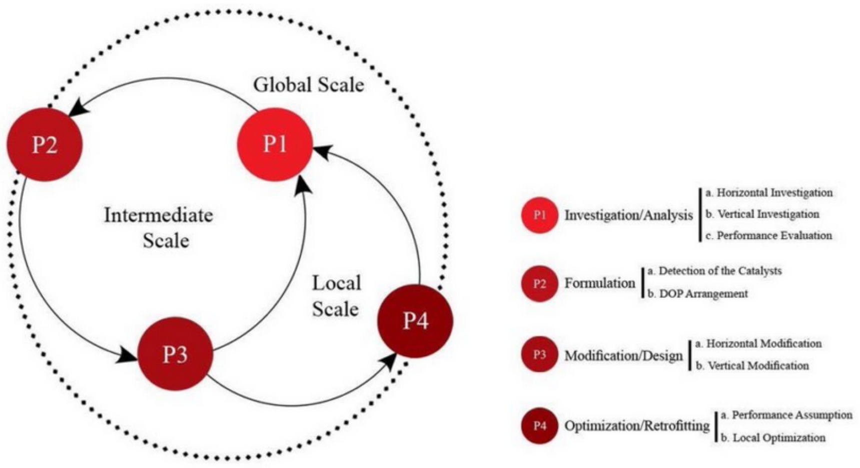

Global Scale
P1
P2
P3
P4
Intermediate Scale
Local Scale
P1 Investigation/Analysis
a. Horizontal Investigation
b. Vertical Investigation
c. Performance Evaluation
P2 Formulation
a. Detection of the Catalysts
b. DOP Arrangement
P3 Modification/Design
a. Horizontal Modification
b. Vertical Modification
P4 Optimization/Retrofitting
a. Performance Assumption
b. Local Optimization

Figure 2.1: IMM phases

In the first part, the Investigation phase, the actual state of the system is dismantled into its Components (Volume, Void, Network, Type of Uses) and reassembled into Key Categories (KC) in order to assess the system Determinants (Compactness, Complexity and Connectivity) with the goal of achieving an efficient urban form. Now we describe only the elements of IMM that are relative to our experiments using the definitions provided by Carlo Biraghi in his work "Multi-Scale Modelling Approach for Urban Optimization: Urban Compactness Environmental Implications" [12]:

- Components: despite their heterogeneity, cities can be dismantled into four components, whose unpredictable and continuous interaction over times gave birth to contemporary urban areas. These components are Volume (the built part), Void (empty spaces), Function (activities performed by citizens) and Network (networks of different modalities) [13]. Components are the least set of elements to be considered when dealing with urban environment. People are agents whose behaviour is affected by the configuration and interaction of these components; with their lives, they affect and reflect the performances of the city;
- Key Categories: we have already introduced Key Categories, mentioning their representing role of the synergy between parts of the built environment. More precisely, Key Categories used in IMM (up to now) are:
  - Urban Porosity: the spatial relationship between urban built-ups and voids;
  - Proximity: the structural relationships driven by the distances between basic land-uses;
  - Diversity: the structural relationship derived from the different typologies of land-uses;

    - Accessibility: the mobility patterns driven by dynamic characteristics of origins and destinations;
    - Effectiveness: the static effect of urban characteristics on the functioning order of mobility systems;
    - Interface: the characteristics of the street network that influence overall connectivity;
    - Permeability: the relationship between the street network and spatial component influencing overall connectivity.

- Metric: IMM aims at providing quantitative measures (metrics) that can pinpoint significant features of the spatial organization of the urban elements, in order to characterize the concept of Key Categories. Metrics describe the properties of the sample area. It is possible to create almost an infinite number of metrics even if many could result as redundant because built on the same parameters. The ones we will use in the datasets described in chapter 4 are related to porosity and permeability.

- Attributes: Attributes are data related to morphological characteristics of the territory and are used to compute metrics. Attributes could be both geometrical properties and additional information. More attributes can be obtained by numerical or spatial operation on the existing ones.

2.2.2 SIMBA methodology

SIMBA is a systematic clustering-based methodology with the goal of supporting and improving the Investigation phase of IMM, by understanding and analysing urban environments and the relationships among their parts. The methodology, shown in figure 2.2, is composed of three main phases:

1. BUILT ENVIRONMENT DECOMPOSITION (BED phase)
2. FIRST LEVEL CLUSTERING (FLC phase, one for each dataset)
3. SECOND LEVEL CLUSTERING (SLC phase)

The input is a built environment of any Dimension. In the first step of the BED phase, the granularity at which the analysis is conducted must be chosen. In other words, the samples that will be clustered are defined. After the granularity, there is the definition of the Dimensions, which represent the different aspects to analyse. They can be of any type and categories, e.g., performances, morphology characteristics, demographic data and so on. Once the datasets with the wanted characteristics are created, they can move to the FLC phase. In these part, after a pre-processing step and a feature selection step, clustering is performed on each dataset separately. The feature selection is a process where the attributes of the data used for clustering are chosen. In SIMBA, this selection is usually done twice for every dataset, once manually using a set of attributes selected by experts and once with an automatic entropy-based procedure. After clustering with both

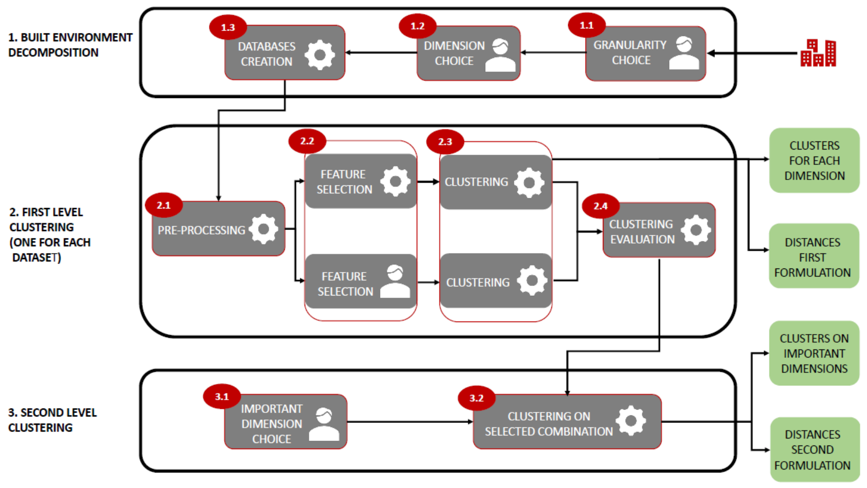

1. BUILT ENVIRONMENT DECOMPOSITION
1.3 DATABASES CREATION
1.2 DIMENSION CHOICE
1.1 GRANULARITY CHOICE
2. FIRST LEVEL CLUSTERING (ONE FOR EACH DATASET)
2.1 PRE-PROCESSING
2.2 FEATURE SELECTION
FEATURE SELECTION
2.3 CLUSTERING
CLUSTERING
2.4 CLUSTERING EVALUATION
3. SECOND LEVEL CLUSTERING
3.1 IMPORTANT DIMENSION CHOICE
3.2 CLUSTERING ON SELECTED COMBINATION
CLUSTERS FOR EACH DIMENSION
DISTANCES FIRST FORMULATION
CLUSTERS ON IMPORTANT DIMENSIONS
DISTANCES SECOND FORMULATION

Figure 2.2: SIMBA methodology flow

the manual and automatic sets, the two results are compared and evaluated. In the third and last phase, the SLC one, the evaluation of the obtained results together with the IMM expert's knowledge and needs are combined and used to select which are the Dimensions, and thus, features, that will be used in the Second Level Clustering. The outputs of the procedure are:

- clusters for each Dimension;
- distances between elements for each Dimension;
- clusters and distances calculated combining only the selected Dimensions, using the features selected for each one of them in the FLC phase.

In our tool, we implemented most of the functionalities and settings used for SIMBA, along with additional ones that better suit our goals, as explained in detail in the following chapters.

2.3 Clustering tools in ArcGIS and QGIS

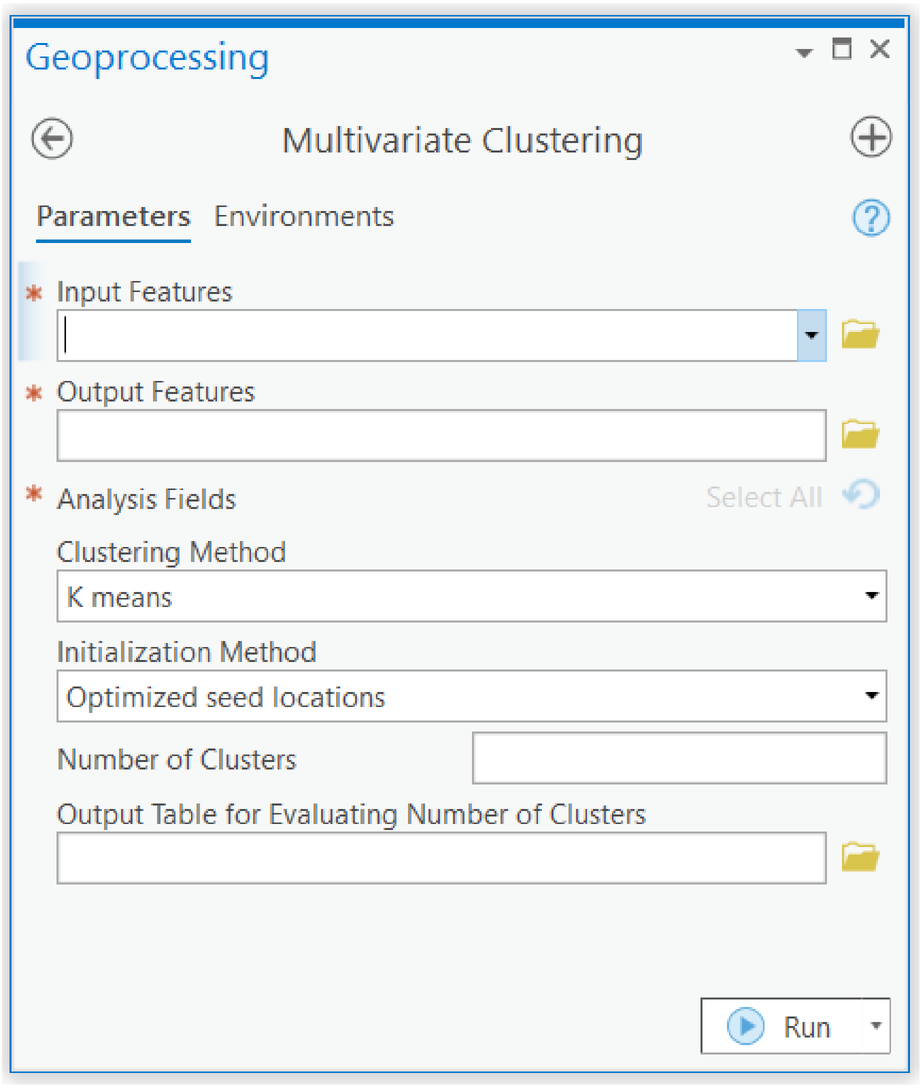
Here we will analyse the tools available for clustering in ESRI ArcGIS and QGIS [4], which are, respectively, the most popular proprietary and open source GIS. The tools fall into two categories, based on which type of attribute they use to create clusters: spatial and attribute-based. The first category finds groups of data points based solely on their spatial location, while in the second one uses the values of one or more numerical attributes.

2.3.1 ArcGIS tools

In ArcGIS there are different tools for what concerns cluster analysis, in both the spatial and attribute-based category. For the spatial category the choice is Density-based Clustering. It finds clusters of data points within surrounding noise based on their spatial distribution. The Density-based Clustering tool works by detecting areas where points are concentrated and where they are separated by areas that are empty or sparse. Points that are not part of a cluster are labelled as noise. Optionally, the time of the points can be used to find groups of points that cluster together in space and time. This tool uses unsupervised machine learning clustering algorithms which automatically detect patterns based purely on spatial location and the distance to a specified number of neighbours.

Examples of the second type are the Find K-Means tool and the more complete Multivariate Clustering [14], shown in figure 2.3. Multivariate Clustering performs clustering on one or more selected attribute with the algorithm K-Means or K-Medoids. It also includes a support functionality to give information on the best number of clusters. After the analysis is complete, there are multiple graphs available that can be used to better understand the formed clusters and the performances of each attribute used.

Finally, there is the tool Spatially Constrained Multivariate Clustering that combines the two previous categories allowing clustering with both attributes and locations in the analysis.



Geoprocessing

Multivariate Clustering

Parameters Environments

* Input Features

* Output Features

* Analysis Fields

Select All

Clustering Method: K means

Initialization Method: Optimized seed locations

Number of Clusters

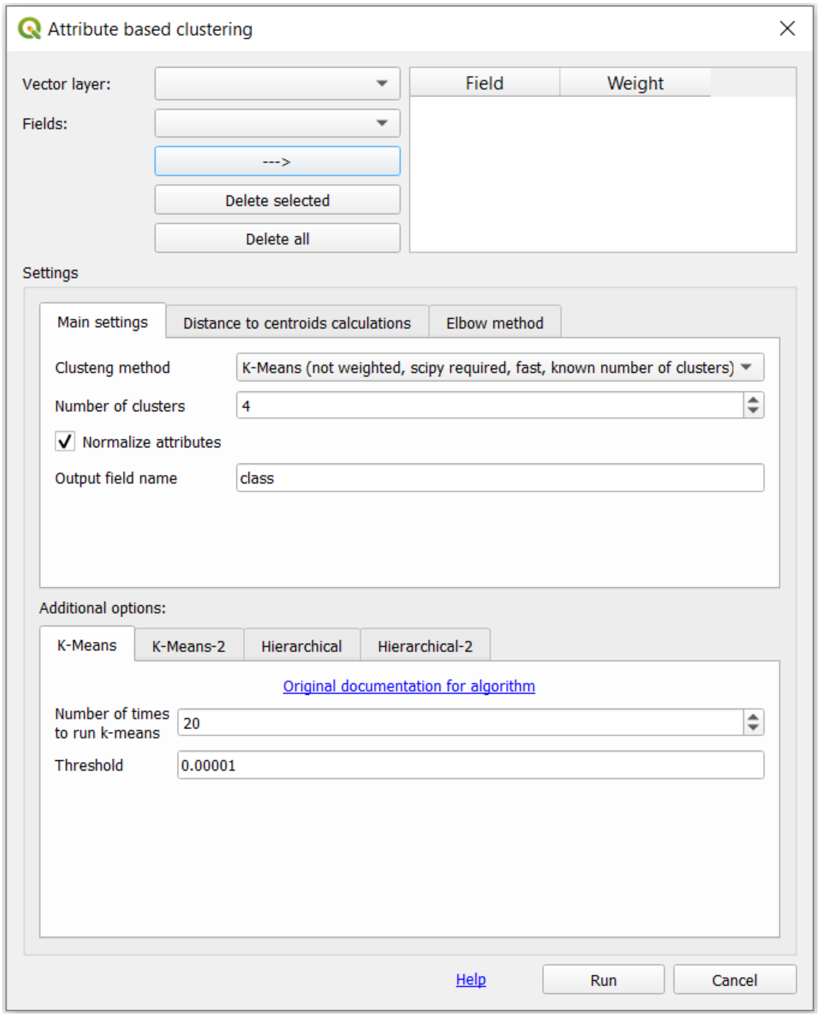
Output Table for Evaluating Number of Clusters

Run

Figure 2.3: ArcGIS Multivariate Clustering

2.3.2 QGIS tools

In QGIS, the tools to perform cluster analysis are more scarce, and provided with plugins developed by users. In the attribute based category, the best solution is Attribute Based Clustering [15], while for the spatial category Cluster Maps. Attribute Based Clustering, shown in figure 2.4, allows the selection between two different algorithms, K-Means and Hierarchical, and provides a good number of parameter settings for both of them. Before the analysis, there is the possibility to plot a graph that helps the user with the selection of the best number of clusters.



Attribute based clustering

Vector layer:

Fields:

Settings

Clusteng method: K-Means (not weighted, scipy required, fast, known number of clusters)

Number of clusters: 4

✓ Normalize attributes

Output field name: class

Additional options:

Original documentation for algorithm

Number of times to run k-means: 20

Threshold: 0.00001

Figure 2.4: Attribute Based Clustering plugin

2.4 Motivation and goals

While in ArcGIS there is a good choice for clustering analysis with tools providing a large number of possibilities beyond the mere application of a clustering algorithm, in QGIS the available solutions present only few additional functionalities from the basic ones. For this reason, the main goal of our work is to develop a one-stop tool for QGIS that is capable to provide support for the whole process of cluster analysis, from the pre-processing of data and feature selection, to the evaluation of the results. Moreover, our research aims to allow the use of the system in a wide range of use cases. This variety refers both to the field of application, for example from social-economic analysis to climate and urban studies, but also to the scale of the data, with efforts to optimize the execution times and implement scalable alternatives for each functionality. One of the main challenges of geospatial analysis is that the data often contains a multitude of different attributes, placing the problem in a high dimensional space that could reduce the performances of the analysis, as explained in section 3.5.2. Due to this, we paid particular attention to the pre-processing and feature selection parts, which are essential to reduce the dimensionality of the dataset. These functionalities are lacking in the tools in ArcGIS, and completely overlooked in the ones available for QGIS. Lastly, our final focus is the usability of the tool. Since the users of QGIS are often not familiar with advanced machine learning techniques, the interface and structure of the developed system should be as intuitive and self explanatory as possible.

Chapter 3

Theoretical background

In this chapter we explain the theoretical concept about machine learning and data mining techniques and GIS software that are necessary to understand our work.

3.1 Geographical Information System

A Geographic Information System (GIS) is a system designed to capture, store, manipulate, analyze, manage, and present all types of geographical data. The key word to this technology is Geography, this means that some portion of the data is spatial. In other words, data that is in some way referenced to locations on the earth [16]. There are different options for GIS software to use, both proprietary, like ArcGIS, and open source, as QGIS. The choice for our work is QGIS, mainly for its open nature and support for plugin developers.

3.1.1 GIS data

GIS data can be separated into two categories: spatially referenced data which is represented by vector and raster forms (including imagery) and attribute tables which are represented in tabular format. Within the spatially referenced data group, the GIS data can be further classified into two different types: vector and raster [17].

A spatial reference is the georeferencing and coordinate system (CRS) assigned to any geographic data. The spatial reference defines how geographic data is mathematically transformed onto a flat map with the least amount of distortion. There will always be some sort of distortion in geographic data since it is the projection of three-dimensional data onto a two-dimensional plane. When choosing a spatial reference, it is necessary to choose which type of distortion you want to minimize [18]. When working on different layers it is essential that they are in the same CRS to avoid any error.

Vectors are composed of geometries created by connecting one or more vertex with paths and can be categorized based on their geometry type:

- points
- lines
- polygons

Points have zero dimensions and are stored as couples of coordinates indicating a physical location. They are often used for objects that are too small to be represented by a polygon, like a city at a global scale. Lines are one-dimensional and connect two or more vertices in a path, so they can only be used to calculate length. They usually represent rivers, trails, and roads on maps. Polygons are two dimensional areas and are created connecting three or more vertices in a closed path. With vector data we can use attribute tables where the values of every line are linked to a single geometry.

Raster data is any pixelated (or gridded) data where each pixel is associated with a specific geographical location. The value of a pixel can be continuous (e.g., elevation) or categorical (e.g., land use). If this sounds familiar, it is because this data structure is very common: it's how we represent any digital image. A geospatial raster is only different from a digital photo in that it is accompanied by spatial information that connects the data to a particular location. This includes the raster's extent and cell size, the number of rows and columns, and its coordinate reference system [19]. In a raster, every pixel corresponds to a square on Earth's surface and the side length of it is called spatial resolution. Raster layers can be converted into vector layers using the pixels as points or polygons.

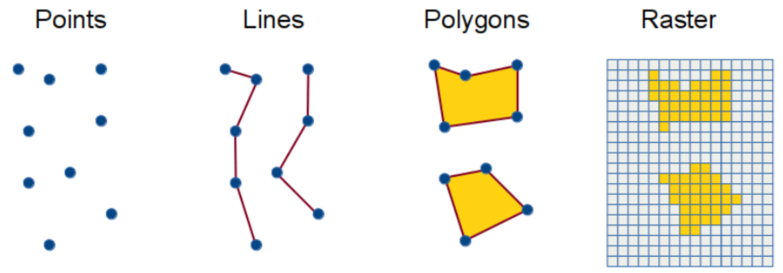


Figure 3.1: Vector and raster data

3.1.2 QGIS plugins

QGIS, as an open-source software, leaves the possibility to the user to implement new functionalities through Python and C++ plugins. To support the development of plugins, QGIS provides a python API [20] to access every functionality and data available in the software through code. Moreover, it makes available a series of guides and tools to help users in the creation, the update, and the publication of plugins. Plugins can be categorized into two types, based on their functionalities:

- regular plugins use a custom user interface and custom logic to process the data, they are usually used when the developer needs more interactions from the users than simple inputs;
- processing plugins are primarily for analysis and limit the user interaction to select only inputs and outputs. They present the standard processing interface from QGIS and have access to other useful functionalities, such as batch processing and the possibility to be used from the python console. Given these advantages, this should be the preferred option when developing a plugin with no particular needs.

The plugin we implemented, presented in chapter 5, falls into the first category. This choice is well justified since we had to combine in a single software a variety of functionalities that would have required multiple processing plugins.

In QGIS a row of a data table is called a feature, while in machine learning that name is used to indicate columns. To avoid confusion in the following sections, the term "feature" will only be used for columns. To refer to a row, we will use other common terms like data point or sample.

3.2 Machine learning

Machine learning (ML) is a subset of artificial intelligence (AI) that provides systems the ability to automatically learn and improve from experience without being explicitly programmed. Machine learning focuses on the development of computer programs that can access data and use it to learn for themselves. The process of learning begins with observations or data, such as examples, direct experience, or instruction, in order to look for patterns in data and make better decisions in the future based on the examples that we provide. The primary aim is to allow the computers to learn automatically without human intervention or assistance and adjust actions accordingly [21].

Machine learning algorithms can be divided into four main categories based on how they learn, and the data used for training:

- supervised learning algorithms start from a known training dataset with labelled data and produce an inferred function used to make predictions on unseen data. In an optimal scenario, the function is able to assign to every new data point a correct output value. The algorithm can evaluate the accuracy of the prediction with a defined loss function, and improve the model in the case that the results are not precise as expected. Widely used algorithms in this category are linear regression and support vector machines.

- unsupervised learning algorithms take as input data that is not labelled or classified. The goal of this models is to find hidden structures or patterns of the data points, without a right or wrong output. This freedom in learning an output can bring obvious disadvantages in relation to supervised learning since it is more difficult to evaluate the performance of the system and its accuracy. On the other hand, the workload to prepare the input dataset is minimal, since it does not require labels. One of the most common unsupervised learning operations is clustering, which aims to separate similar data points in non-specified groups.

- Semi-supervised learning algorithms are a combination of the two previous categories, since they use for training both labelled and unlabelled data. Usually, there is a vast majority of the latter and a small amount of the former that can produce a considerable improvement on the accuracy of the learner. This type of system can be particularly useful when the labelling of data is difficult or time consuming, while unlabelled data is easily accessible.

- reinforcement learning algorithms are methods that interact with their environment by producing actions and discover errors or rewards. Trial and error search and delayed reward are the most relevant characteristics of reinforcement learning. This method allows machines and software agents to automatically determine the ideal behaviour within a specific context in order to maximize its performance. Simple reward feedback is required for the agent to learn which action is best, this is known as the reinforcement signal [21].

After giving an introduction on the different paradigms of machine learning, we will now mostly consider unsupervised learning and clustering in particular, which is the main focus of our work.

3.3 Clustering algorithms

Clustering is an unsupervised machine learning task, where the main goal is to partition objects into groups of similar objects (clusters) and to discover hidden structures or patterns in the data. The objects are typically described as vectors of features (also called attributes) and can be numerical (scalar) or categorical. The assignment can be hard, where each object belongs to one cluster, or fuzzy, where an object can belong to several clusters with a probability. The clusters can be overlapping, though typically they are disjoint. Fundamental in the clustering process is the use of a distance measure, a function that quantifies the similarity of two objects [22].

The term clustering does not correspond to a specific procedure, but to a general problem that can be solved by using various algorithms. The algorithms can be categorized based on their cluster model:

- Connectivity-based clustering: these models are based on the idea that objects are more similar to close objects rather than far ones. Therefore, the clusters are formed by connecting data points based on their distance, defined using a defined distance function. These algorithms can follow either a bottom-up or a top-down approach. The first case, called agglomerative, starts with each data point in its own clusters and iteratively aggregates the closest pair of clusters. The top-down or divisive approach starts from a single cluster containing all objects and splits one cluster every step. At the end of both processes, the output is not a simple portioning of the dataset, but the extensive hierarchy of points joining at different distances, hence the name hierarchical that can be used for these algorithms.

- Centroid-based clustering: the clusters in this case are represented by a central vector, not necessarily part of the dataset, called centroid. These are iterative algorithms in which the notion of similarity is not derived by the distance between data points, but the distance to the centroids. The number of clusters required is specified beforehand, and the process becomes an optimization problem. Since the optimization problem is NP-hard the common approach is to search for approximate solutions. A widespread algorithm of this family is K-Means.

- Distribution Models: These clustering models are based on the notion of how probable is it that all data points in the cluster belong to the same distribution (For example: Normal, Gaussian). These models often suffer from overfitting. A popular example of these models is Expectation-maximization algorithm which uses multivariate normal distributions [23].

- Density models: These models search the data space for areas of varied density of data points in the data space. It isolates various density regions and assigns the data points within these regions in the same cluster. Popular examples of density models are DBSCAN and OPTICS [23].

After the brief overview of the different algorithms to perform cluster analysis, we can now focus on the ones we used during our work: Agglomerative hierarchical and K-Means.

3.3.1 Agglomerative hierarchical clustering

As we introduced before, agglomerative hierarchical clustering starts by separating every data point in its own cluster. Then the two closest clusters get merged together and this step is repeated until only one cluster is left. The result of this procedure is a hierarchy of the clusters showing at which distance they are merged; this graph can be displayed graphically using a dendrogram. In algorithm 1 we show the pseudocode of hierarchical clustering and in figure 3.2 an example of a dendrogram.

On the vertical axis there is the distance while on the horizontal one there are 20 clusters of one data point. Samples 16 and 19 are at a distance of about

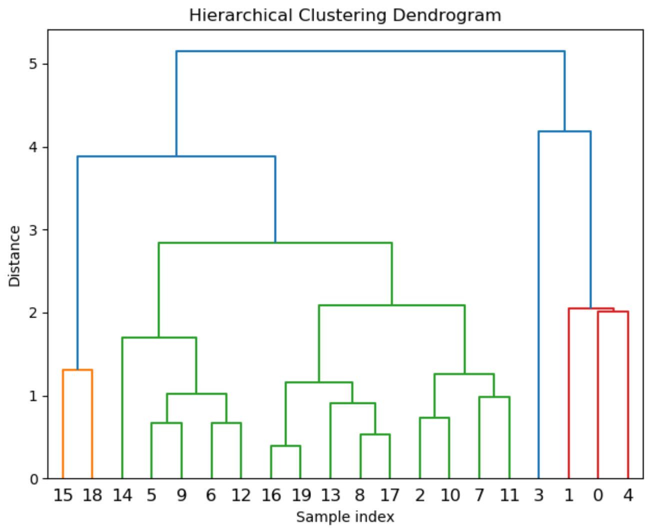


Figure 3.2: Hierarchical clustering dendrogram example

0.4 as we can see on the graph, and are the first two to be merged forming a new cluster. Going from top to bottom we can see the order in which every group is created. The dendrogram can be used to select the best number of clusters, as explained in section 3.4.4.

Algorithm 1: Agglomerative hierarchical clustering

```
Input: Data Points $X = \{x_1, \ldots, x_n\} \subseteq \mathbb{R}^d$, number of clusters $k$,
       distance function $\delta$
Output: Clusters $C$
$C = \{C_i = \{x_i\} | x_i \in X\}$
$D = \{D_{i,j} = \delta(x_i, x_j) | x_i, x_j \in X\}$
while $|C| > k$ do
    Find closest pair of cluster $C_i, C_j \in C$
    $C_{ij} = C_i \cup C_j$
    $C = C / \{C_i, C_j\} \cup C_{ij}$
    Update $D$ to reflect the distance between $C_{ij}$ and existing clusters
end
return $C$
```

3.3.2 K-Means

K-Means is an iterative clustering algorithm that aims to find a local maxima in each iteration. First, we need to specify the parameter k, representing the number of clusters required, and assign every data point randomly to a cluster. Then, the algorithm computes the centre (centroid) for every cluster. The next step is to calculate the distance of data points to the centroids and reassign every object to the closest cluster. Finally, it recomputes the cluster centroids and repeats the last two steps until convergence, meaning that the data points assignments no longer change. The standard version of K-Means uses random initialization for the first partition of the objects, but there are also other methods to choose the initialization that will yield different results. A technique commonly used to address the problems of a poor random initialization, is to repeat the process multiple times and select the partitioning with the best sum of squared error (SSE). The error of each data point is defined as its distance to the closest centroid.

In algorithm 2 we show the pseudocode for the standard version and in figure 3.3 some iterations of a K-Means example [24] with few points.

Algorithm 2: Standard K-Means

```
Input: Data Points $X = \{x_1, \dots, x_n\} \subseteq \mathbb{R}^d$, number of clusters $k$
Output: A set of $k$ Centroids: $C = \{c_1, \dots, c_n\} \subseteq \mathbb{R}^d$
Initialize $C = \{c_1, \dots, c_n\} \subseteq \mathbb{R}^d$ at random
while $C$ has not converged do
    $S_i = 0, \forall i \in [k]$
    foreach $x \in X$ do
        $j^* = argmin_j ||x_i - c_j||$
        $S_{j^*} \leftarrow S_{j^*} \cup \{x_j\}$
    end
    $c_j \leftarrow \frac{1}{S_j} \sum_{x \cup S_j} x, \forall j \in [k]$
end
return $C$
```

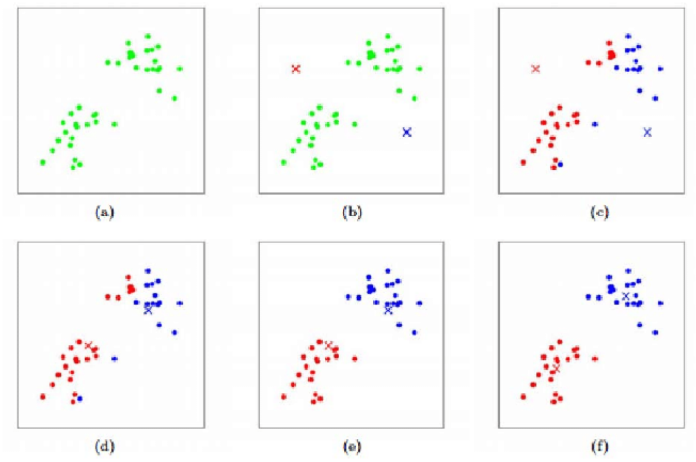


Figure 3.3: K-Means example iterations

3.3.3 K-Means vs Agglomerative hierarchical

Of course, there is not a definite answer on which of the two presented algorithms is the best, as they have different advantages and disadvantages, and the decision of the preferred algorithm is based on the specific application that we are considering.

The first factor we consider is the computational complexity and the scaling on larger datasets. The space requirements for K-Means are modest because only the data points and centroids are stored. Specifically, the storage required is $O((m + K)n)$, where m is the number of points and n is the number of features. The time requirements are also modest, basically linear in the number of data points. In particular, the time complexity is $O(I * K * m * n)$, where I is the number of iterations before convergence. Since I is often small and can be safely bounded, as most changes typically occur in the first few iterations, we can state the algorithm is linear in m, as long as the number of clusters K is significantly less than m [25]. Agglomerative Hierarchical clustering presents instead worse complexities for both time and space. It requires the storage of a similarity matrix for all pairs of m data points, for a total space complexity of $O(m^2)$. The time complexity, after performing some optimizations, is $O(m^2 * \log m)$ [25]. Therefore, K-Means is clearly the best solution to handle big datasets.

However, obvious disadvantages of K-Means are the necessity to select the number of clusters a-priori and the random nature of the algorithm, which may produce different solutions in different runs. The hierarchical algorithm is not affected by these problems, since it provides the hierarchy of all the clusters and is deterministic, meaning it always provides the same solution.

The last factor we consider is the shape of the clusters in the two approaches. Hierarchical clustering can work well on data of any distribution, since the concept of cluster boundaries is not defined. While K-Means works better when the shape of the clusters is hyper spherical.

3.4 Additional machine learning concepts

In this section we cover some additional topics of machine learning that are important for our work, such as distance measures, the scaling of data, the problem of the number of clusters, and the evaluation of clustering.

3.4.1 Distance measure

Clustering is based on the concept of similarity of data points, which measures how much two objects are similar. Similarity is usually defined as a distance in the space, where the dimensions represent the features of the dataset. The distance can be computed in different ways, where widely used formulas are:

- Euclidean distance;
- Manhattan distance;

- Cosine similarity;
- Hamming distance, used for binary vectors.

Only the first two are explained, since they are the ones allowed in our plugin.

Euclidean distance is the most common approach and is the length of the direct segment connecting two points. It is defined as:

$$d(x,y) = \sqrt{\sum_{k=1}^{n}(x_k - y_k)^2}$$

The Manhattan distance is the sum of the lengths of the projections of the line segment between the points onto the coordinate axes, and is described by the formula:

$$d(x,y) = \sum_{k=1}^{n}|x_k - y_k|$$

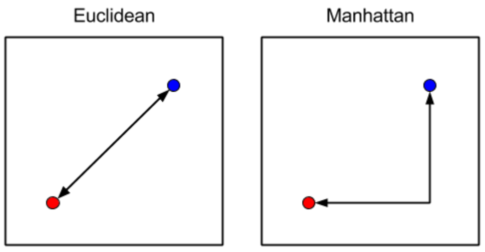


Figure 3.4: Euclidean vs manhattan distance

The choice of the distance measure is important in the cluster analysis, since it affects the final result. In hierarchical clustering it is possible to select the preferred distance, while in the standard version of K-Means it is mandatory to use Euclidean distance. In hierarchical clustering it is necessary to define the distance between clusters as well. Common measures used are complete, single, and average. The first two use respectively the maximum and minimum distances among all data points in the two clusters. Average computes the average of the distances of each object in the two clusters.

3.4.2 Data scaling

In a dataset it is common to have features that represent measures with different units or different scales. When performing an analysis on data that is based on distance, such as clustering, it is essential to take into account these differences. A good approach to handle this problem is to scale the data to bring every dimension to the same equal weight. Two possible transformations to apply are:

- Normalization (or Min-Max normalization): rescales the range of every feature to $[0, 1]$ using the formula:

$$x' = \frac{x - min(x)}{max(x) - min(x)}$$

- Standardization (or Z score normalization): brings the mean of each feature to 0 and the standard deviation to 1, defined as:

$$z = \frac{x - \mu}{\sigma}$$

The decision to scale data and how to do it heavily depends on the features and the algorithms used. For this reason, before applying any transformation, it is necessary to have some information about the data and the process.

3.4.3 Clustering evaluation

Validating the performance of cluster analysis is not as trivial as counting the number of errors or the precision and recall like in the case of supervised learning algorithms. To evaluate a clustering experiment, we usually try to compute a metric describing how well the similar points are grouped together or, when possible, compare its performance to a gold standard. The first approach is called internal evaluation, while the second one is external evaluation.

For internal evaluation, there are different metrics to measure intra-cluster similarity (samples within a cluster are similar) and inter-cluster similarity (samples from different clusters are dissimilar). Two of the most common metrics are:

- Silhoutte coefficient
  The Silhouette Coefficient is defined for each data point and is composed of two scores: the mean distance between a data point and all other points in the same cluster, and the mean distance between a sample and all other points in the next nearest cluster. The Silhouette Coefficient for a set of samples is given as the mean of the Silhouette Coefficient for each sample. The score is bounded between -1 for incorrect clustering and +1 for highly dense clustering. Scores around zero indicate overlapping clusters. The score is higher when clusters are dense and well separated, which relates to a standard concept of a cluster [26]. Silhouette for a single data point and for a set of data are defined as:

$$s(i) = \frac{b(i) - a(i)}{\max(a(i), b(i)} \qquad S = \frac{1}{n}\sum_{i=1}^{n} s(i)$$

  where a is the average distance between a data point i and all the other ones in the cluster to which i belongs, and b is the minimum average distance from i to all clusters to which it does not belong.

- Davies Bouldin index
  The Davies-Bouldin index is another internal evaluation metric, where the validation of how well the clustering has been done is made using quantities and features inherent to the dataset. The index directly evaluates intra-cluster similarity and inter-cluster differences and is defined as a ratio of these two measures. The score can only be positive with a minimum of 0, with lower values indicating better clustering. The formula to calculate the Davis-Bouldin index is:

  $$DB = \frac{1}{n}\sum_{i=1}^{n} \max_{j=1}\left(\frac{\delta_i + \delta_j}{d(c_i, c_j)}\right)$$

  where n is the number of clusters, c_x the centroid of cluster x, δ_x the average distance of all elements in cluster x to centroid c_x, and $d(c_i, c_j)$ is the distance between centroids c_i and c_j.

In external evaluation, the clustering results are compared to a benchmark or gold standard, which is a labelling of the data points produced by an expert of the field. The external metrics evaluate how well the clustering matches the gold standard classes. The comparison matrix introduced in SIMBA that we implemented in our plugin can be seen as a metric in this category. In that case, the benchmark labels are not directly assigned to the data points, but are found with clustering using a set of features selected by a human expert.

3.4.4 Number of clusters

Another challenging task in clustering, related to the evaluation, is the decision of the optimal number of clusters. A simple approach could use an evaluation metric on different cluster numbers and select the one with the best result. Here we will focus on two graphical methods that we implemented in our system and can provide useful insight: the dendrogram and the elbow method.

The dendrogram, as already anticipated before, can be used as a tool for this decision. The best choice for the number of clusters is the number of vertical lines in the dendrogram cut by a horizontal line that can transverse the maximum distance vertically without intersecting a cluster. In figure 3.5 the maximum distance is represented by the segment AB, and the choice is 4 clusters.

The other method, called knee-elbow analysis, is based on a graph with the trends of the Within clusters sum of squares (WSS) and Between clusters Sum of Squares (BSS). These two measures represent, respectively, how closely related are objects in clusters and how well separated are different clusters. The technique consists in looking for a knee or an elbow in the trends, showing a significant modification in the metrics. In the example in figure 3.6, a good choice would be 5 clusters, as shown by the sudden flattening of the two trends.

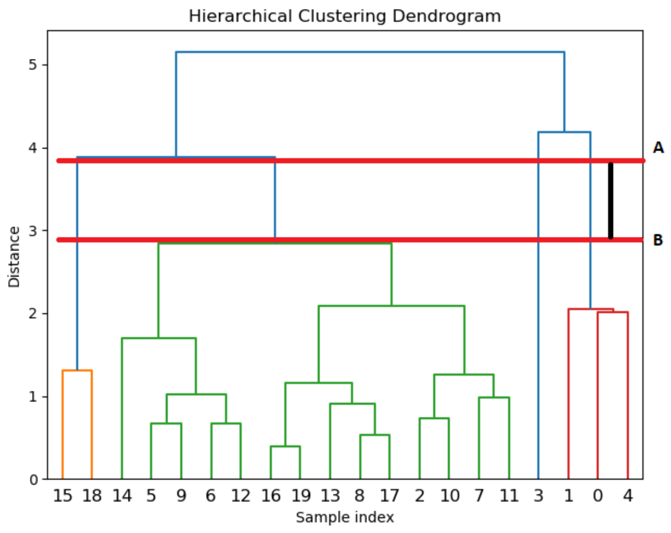


Figure 3.5: Example of number of clusters selection from dendrogram

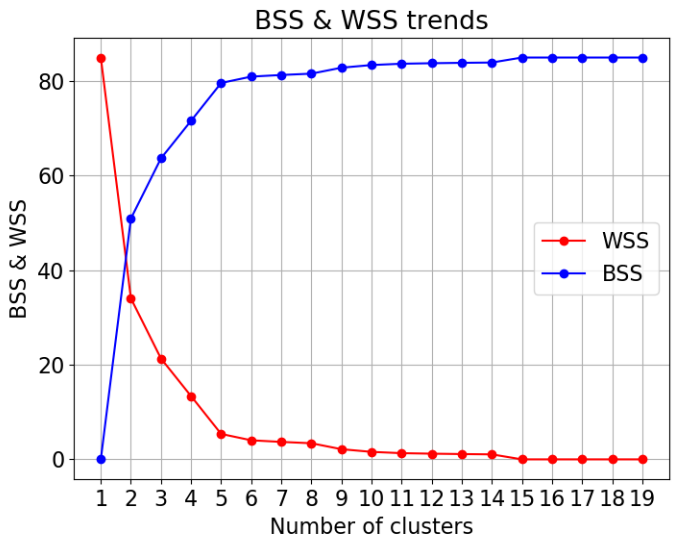


Figure 3.6: Elbow method example

3.5 Feature selection

One of the most important steps of machine learning, especially with high-dimensional problems, is feature selection, which consists in the selection of the optimal subset of features that will be used in the model. This process will determine the quality and the performance of the produced system. Indeed, having fewer features than required will produce a model that is too simple and not capable to predict the right output or to find the best patterns in the data; on the other hand, selecting too many features may lead to overfitting and excessive increase of the model complexity.

Feature selection is closely related to a common problem in machine learning first introduced by Bellman [27] as the "curse of dimensionality". This concept refers to the explosive nature of spatial dimensions and its resulting effects, such as an exponential increase in computational effort, large waste of space, and poor visualization capabilities. A higher number of dimensions theoretically allows more information to be stored, but practically rarely helps due to the higher possibility of noise and redundancy in real world data [28].

To mitigate these problems, there are different feature selection algorithms that can identify and remove irrelevant and redundant dimensions. They can be separated into three main categories:

- wrappers require some method to search the space of all possible subsets of features, assessing their quality by learning and evaluating a classifier with that feature subset. The feature selection process is based on a specific machine learning algorithm that we are trying to fit on a given dataset. It follows a greedy search approach by evaluating all the possible combinations of features against the evaluation criterion. The wrapper methods usually result in better predictive accuracy than filter methods, at the cost of high computational complexity [29].

- filter methods, unlike wrappers, are independent of any machine learning algorithm, and base their selection on a rank of the features. The rank is computed using intrinsic properties of the features and is usually fast to calculate. For these reasons, they are less computationally expensive than wrappers, but with lower prediction performance. After obtaining the rank, the best number of dimensions can be found with cross-validation or other techniques.

- embedded methods perform the selection as part of the model construction process. An example of this category is Lasso Regression, which assigns a penalty to all dimensions, shrinking a lot of them to zero. At the end of the procedure, only the features with a coefficient greater than zero are used for the model. In terms of computational complexity, these algorithms tend to place between the previous two approaches.

3.5.1 Feature selection for clustering

Feature selection for unsupervised learning is usually more challenging than when dealing with supervised learning, since, as already explained, it is difficult to evaluate the performances of the model without a proper label on data; this causes classic algorithms to not work on clustering. Moreover, in the literature there are few attempts to overcome these problems. Some examples are the wrapper framework for unsupervised learning proposed by Dy and Brodley [30] or the ranking algorithm from Dash and Liu [31] explained in section 5.3.1.

A way to reduce the dimensionality of data before clustering, or before using a feature selection algorithm, is to drop the features that we know are most likely irrelevant or redundant, such as features with a really small variability or that are highly correlated with other ones.

3.5.2 Clustering in high dimensionality

For clustering purposes, the most relevant aspect of the curse of dimensionality concerns the effect of increasing dimensionality on distance or similarity. As we saw before, most clustering techniques depend critically on the measure of distance or similarity, and require that the objects within clusters are, in general, closer to each other than to objects in other clusters [32]. Unfortunately, when dealing with spaces in a lot of dimensions, the data points and their distance measure does not behave as intuitively expected. In [33] is shown that, under certain reasonable assumptions on the data distribution, the ratio of the distances of the nearest and farthest neighbours to a given target in high dimensional space is almost 1 for a wide variety of data distributions and distance functions. This is clearly a situation we want to avoid in clustering, since the definition of close points becomes useless. The best distance metric to use when dealing with high dimension is Manhattan, followed by Euclidean, as proven in the work by Aggarwal et al. [34].

Chapter 4

Datasets description

To show the versatility of the developed tool we chose to perform analysis on different types of datasets and with variable sizes. All the datasets refer to the city of Milan and can be grouped in four different categories:

- climate data
- urban data
- demographic and social data
- buildings data

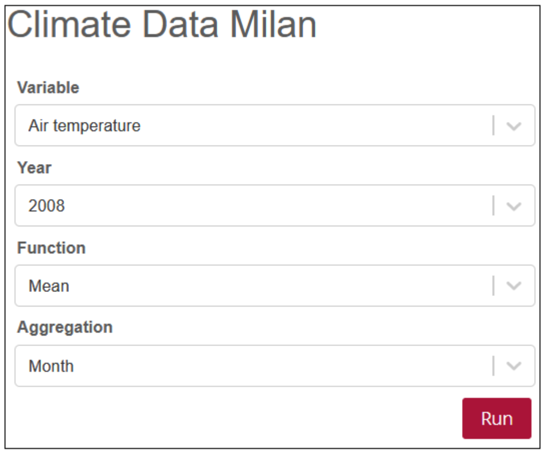
4.1 Climate data

All climate data is from the Climate Data Source from Copernicus [35] and is accessed through the provided Toolbox using an application that we developed for this purpose, shown in figure 4.1. The raw data consist of air temperature, humidity, and wind maps with a spatial resolution of 100 metres and time resolution of an hour, collected from 2008 to 2017. The information about the variables we used is reported in table 4.1.

Variable name	**Unit**	**Comment**
Near surface air temperature	K	Reported at 2m
Near surface relative humidity	1	Reported at 2m
Near surface wind speed	m/s	Reported at 2m, only scalar component

Table 4.1: Climate variables description

The data is aggregated over every year and also over months for 2016 and 2017, using both maximum and mean. For every different variable, time frame, and statistical operation, we produce a NetCDF file with the Toolbox. Since the output file is in a raster format, we need to convert it with QGIS to a vector


Climate Data Milan

Variable

Air temperature

Year

2008

Function

Mean

Aggregation

Month

Figure 4.1: Copernicus Toolbox application for climate data

type to use it in the plugin. After changing the CRS and the format, we can join the different vector layers to obtain the datasets for the experiments:

- all variables aggregated by year with max and mean, data from 2008 to 2017, for a total of 60 features;
- all variables aggregated by month with max and mean, data from 2016, for a total of 72 features;
- all variables aggregated by month with max and mean, data from 2017, for a total of 72 features.

All these datasets are composed of 17877 data points with a squared polygon as geometry, which corresponds to the surface of Milan with a 100 meters resolution. In figure 4.2 we show the process flow of the climate datasets creation.

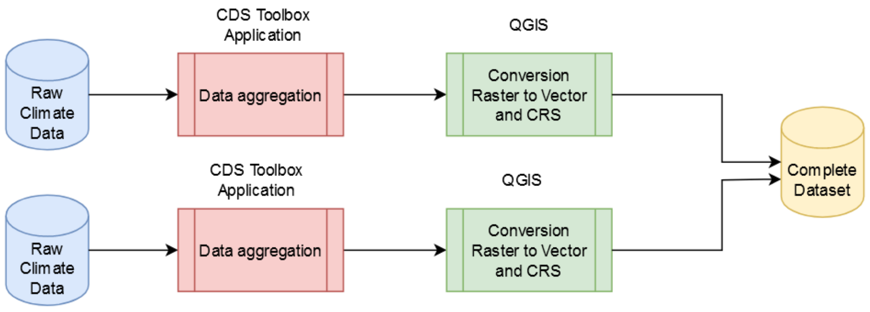


Figure 4.2: Process flow of climate dataset creation

4.2 Urban data

These are the datasets describing the city of Milan in terms of metrics and attributes from IMM, characterized by three different granularities:

- ACE (Area di Censimento) is a partition of the city of Milan into 85 units based on census areas;
- NIL (Nuclei di Identità Locale) are 88 units that can be defined as neighbourhoods of Milan. They are introduced from PGT (Piano di Governo del Territorio) as areas interconnected by infrastructures, services for mobility and green areas. Compared to ACE, NIL have different shapes and also include the big rural areas outside the urban part of the city;
- Block is the largest and heterogenous dataset of the three, composed by 4328 data points. Each of these geometries is a polygon delimited by street areas (Area stradale, 010104 [36]), where the street area can include different types of viability, e.g. pedestrian zones and tram lines. For this reason, the sizes and compositions of the blocks vary greatly from the city center to the rural areas in the outskirts.

For what concerns the features, in all three datasets we find Metrics and Attributes that describe various characteristic of the territorial units. In NIL and ACE there are 109 features while for Blocks only 99, since we removed the ones that are not meaningful at this granularity.

Metrics are quantitative measures that can pinpoint significant features of the spatial organization of the urban elements in order to characterize the concept of KC. The ones available in our dataset are related to Permeability and Porosity and represent the ratio between different built areas (Volume) and different empty spaces (Void). Attributes are used to compute metrics and sometimes correspond to them (as for Area and Perimeter). They are data related to the morphological characteristic of the territory. Both Metrics and Attributes in our dataset are related to:

- Buildings
- Courts
- Blocks
- Districts

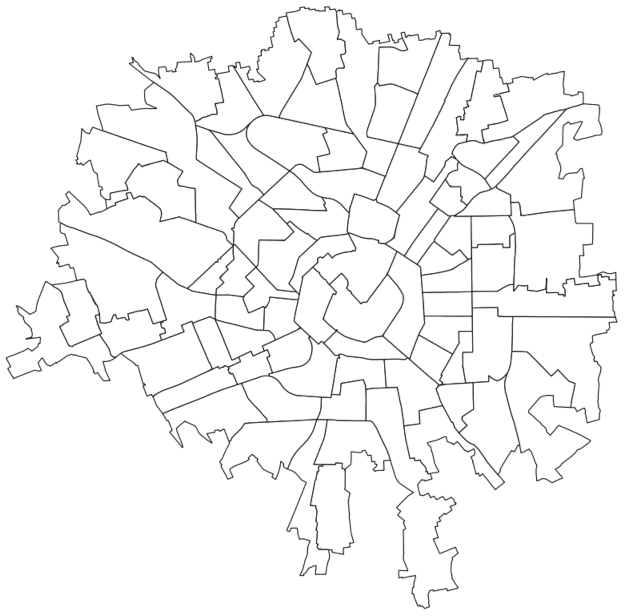


Figure 4.3: ACE map

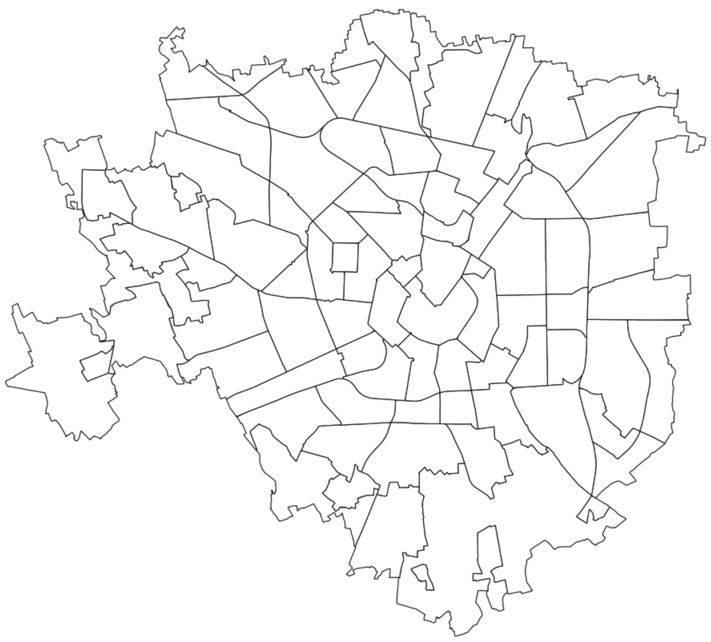


Figure 4.4: NIL map

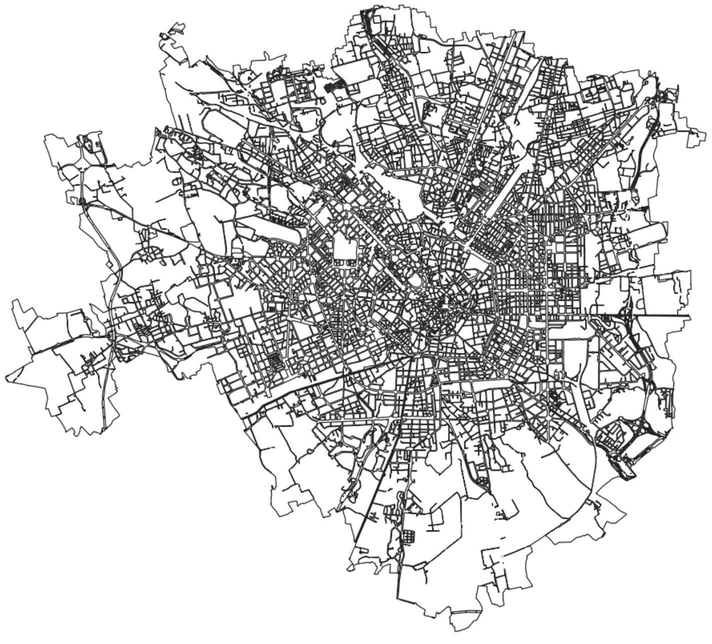


Figure 4.5: Block map

4.3 Demographic and social data

The demographic and social dataset [37] is composed by 32 statistical indicators calculated on the basis of the census in 2011, grouped into the 85 ACE described previously. The indicators include data about different types, such as age distribution, employment, and education, regarding both the entire population or specific classes. Some example are the population density, young (15-29 years old) people neither in employment nor in education, families with 5 or more components.

4.4 Building data

Building dataset contains data about all the buildings inside the city. It is composed by 66811 data points, each representing a building (Corpo edificato, 020181 [36]) composed by one or more volumetric units (Unità volumetrica, 020101 [36]), and 34 features that describe the characteristics and the structure of the buildings in both 2D and 3D. In figure 4.6 there is a small portion of the dataset.

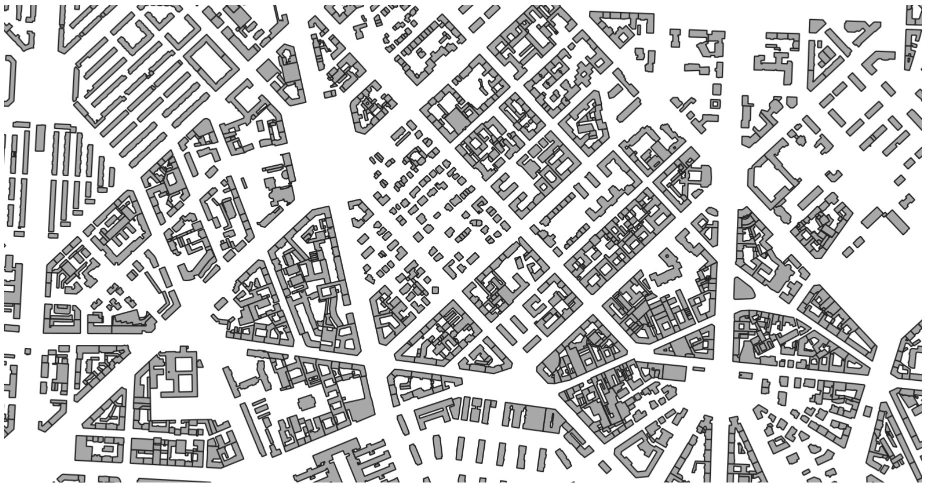


Figure 4.6: Part of Building map

Chapter 5

Plugin implementation

In this chapter we discuss the implementation of the plugin developed for QGIS3 that allows attribute based cluster analysis on a vector file with any geometry type. First, we show an overview of the implementation and then a more detailed description of every part. The details about installation are available in the User Guide in Appendix A.

5.1 Implementation overview

We developed the plugin in Python (version 3.7), and all the necessary files and base code are created using Plugin Builder [38]. The GUI is designed with QT Designer and additional QGIS custom widgets [39], in order to have automatic updates of the interface after the user adds or removes layers to the project.

The plugin is composed of three main parts that will be discussed in the following sections:

- Feature cleaning
- Clustering
- Evaluation

One of the major challenges during development has been allowing most of the functionalities on large datasets as well, both from the point of view of the number of samples and the number of dimensions. To achieve this, we also implemented algorithm options with good time complexities, as in the case of entropy with sampling and K-Means. Moreover, for all the data storage and manipulation done in the system, we use the data structures and functions provided by the libraries pandas [40, 41] and NumPy [42] to guarantee high performance.

In figures 5.1, 5.2, and 5.3 we show high level sequence diagrams to illustrate the interactions between the user, the plugin, and QGIS.

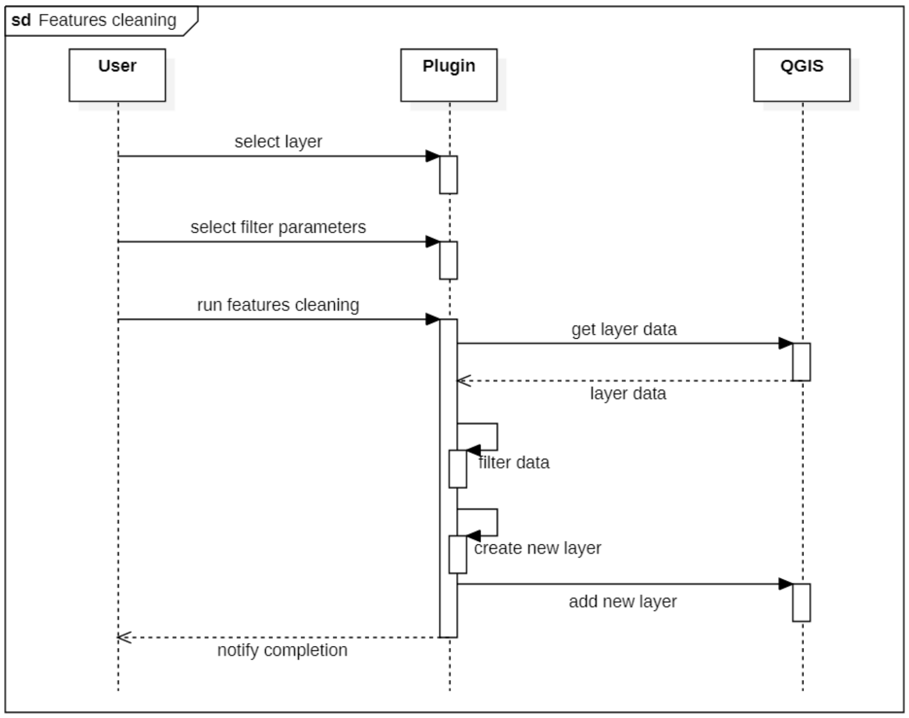


Figure 5.1: Features cleaning sequence diagram

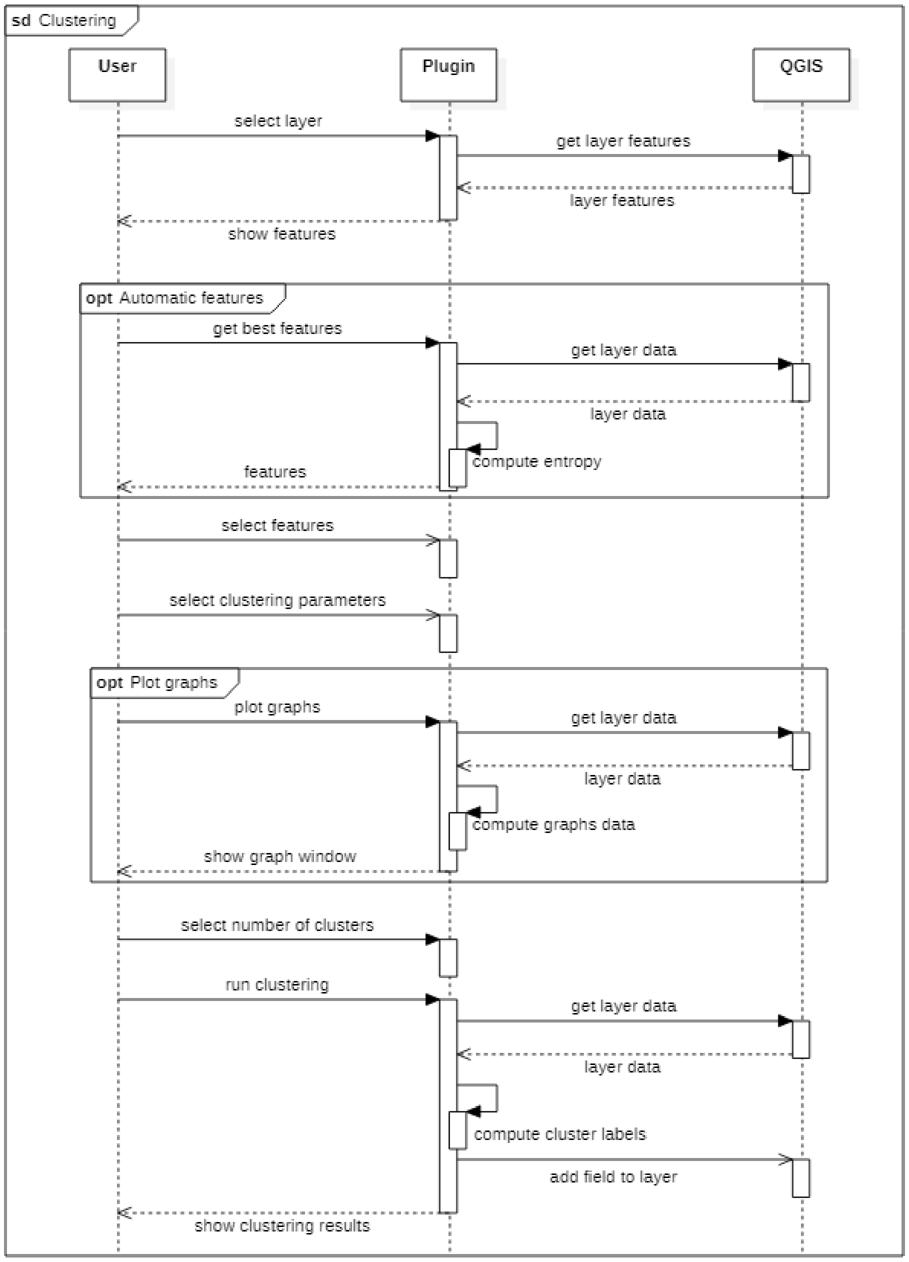


Figure 5.2: Clustering sequence diagram

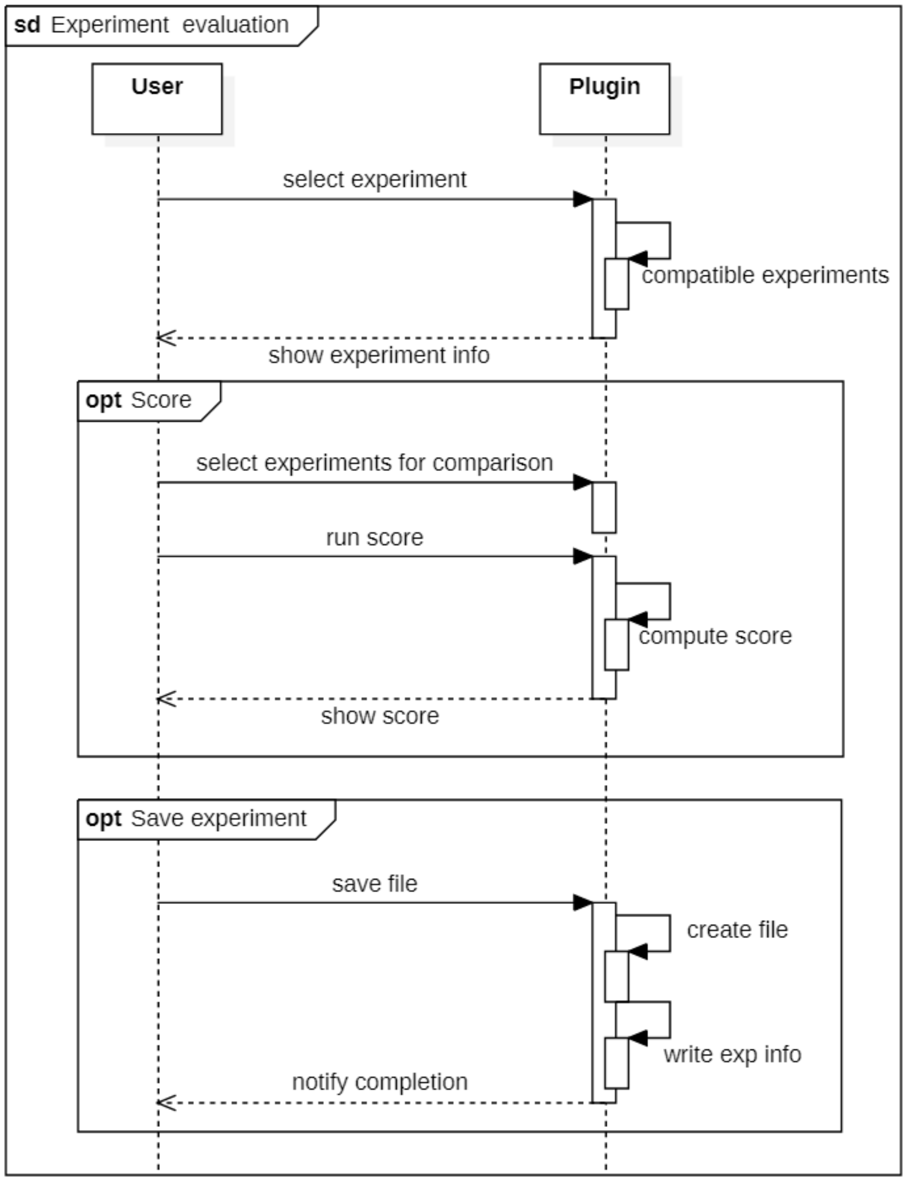


Figure 5.3: Experiments evaluation sequence diagram

5.2 Feature cleaning implementation

In the first section, we want to give options to reduce the dimensionality of the dataset by dropping the features that are most likely bad for clustering. This is important to achieve better results and faster execution time, avoiding the problems of clustering in high dimensionality explained in section 3.5.2.

5.2.1 Highly correlated features

As already discussed before, it can be useful to remove features in the dataset that present a high correlation, both positive and negative, since they provide redundant information and can lead to overweight certain attributes. We give the possibility to choose the threshold of correlation and the criterion used to keep a feature among a multicollinear group:

- order of the attributes in the dataset;
- lower average correlation with all the other features;
- similarity of the feature distribution to the Normal distribution. To check which feature is preferred we calculate the Shapiro-Wilk statistics and select the one with a higher value;
- ratio between max interval of values and domain of the feature, where a higher value indicating better coverage of the domain is preferred. To use this criterion all the features must have the same domain specified by the user.

In algorithm 3 we present the pseudocode of the procedure used to filter the features that we want to keep.

First, we get the symmetric matrix containing the pairwise correlation between every dimension, using the absolute values to account for both positive and negative correlation. Then we store in an array a value for every feature based on the selected criterion. Finally, we scan the upper section of the matrix and for every group with correlation above the threshold we only keep the feature with the best value, flagging the other ones for elimination. To prevent dropping more features than required, it is important to skip iterations regarding already flagged ones.

Algorithm 3: Correlated features filter

```
Input: Dataset matrix $D$, threshold $t$
Output: Filtered data matrix $D$
$C = correlation(D)$ //get correlation matrix
Compute $feature\_values$
$flagged\_features = []$
for $(i = 0)$ to $(num\_of\_features - 1)$ do
    if $i \in flagged\_features$ then
        Skip iteration
    end
    $best\_feature = i$
    for $(j = i + 1)$ to $(num\_of\_features - 1)$ do
        if $j \in flagged\_features$ then
            Skip iteration
        end
        if $C_{ij} > t$ then
            Write in $best\_feature$ the one with higher $feature\_value$
            Add the other to $flagged\_features$
        end
    end
end
$D = D - \{columns \in flagged\_features\}$
return $D$
```

5.2.2 Constant features

Constant features are the attributes in a dataset with the same value for all the points. This type of feature does not provide any information in cluster analysis and only increases the dimensionality causing worse time and clustering performance. For these reasons, they should be removed from the data. Since the values are constant, the variance of the feature is zero and we can use the function VarianceThreshold() from Scikit-Learn [43] with a threshold of 0 to select the fields we need to remove.

5.2.3 Quasi constant features

With quasi constant features, we mean features with a vast majority of constant values and only a few outliers differentiating from them. As in the case of zero variance feature, these ones usually carry no valuable information and, in the worst case, can be detrimental for our analyses. In order to select which features fall into this category, we prefer to avoid using a threshold on variance, which is difficult to define in the general case. Instead, we use two different parameters, introduced in the function NearZeroVar() from the Caret package developed for R [44]:

- ratio between the two most frequent values
- number of unique values relative to the number of samples

To flag a feature, first the frequency of the most prevalent value over the second most frequent value must be above the frequency threshold. Secondly, the number of unique values divided by the total number of samples must also be below the unique values threshold. The thresholds are set for default respectively to 19 and 0.05 (5%).

5.2.4 Creation of new layer

After the filtering of the features is complete, we create a new layer and add it to the QGIS layer section with the name "original-layer-name_mod". In QGIS, it is not possible to duplicate a layer without sharing the same data source, and for this reason, we need to go through more steps. First, we create an empty scratch vector layer with the same geometry type and CRS of the base one and we copy the full dataset from the old to the new vector. Then, for every point in the new vector layer, we assign the geometry of the corresponding point in the old one. Now we can remove the features we previously flagged for elimination.

5.3 Clustering implementation

This section is used to perform clustering on the chosen vector layer. First of all, the user needs to select the features to use in the process. It is possible to select the features both manually and automatically. The automatic feature selection is done using an entropy-based algorithm presented in two versions with different computational complexities. All the experiments carried out are stored as instances of the Experiment() class and can be seen in the Evaluation section. The cluster labels are added to the vector layer as a new field with the name selected by the user.

5.3.1 Feature selection

To perform the analysis the user must select the dimensions he wants to use. It is possible to add them manually from a list, extracted from the layer's dataset considering only the numeric fields, or use the automatic procedure. The latter is a methodology based on entropy feature selection for clustering presented by Dash and Liu [31]. Before showing the implementation of the algorithm, we need to introduce some theory on Entropy and its relation to feature ranking.

Consider each feature F_i as a random variable while f_i as its value, from entropy theory we know that entropy is:

$$E(F_1, \ldots, F_M) = -\sum_{f_1} \cdots \sum_{f_M} p(f_1, \ldots, f_M) \log p(f_1, \ldots, f_M)$$

where $p(f_1, \ldots, f_M)$ is the probability or density at the point $(f_1, \ldots, f_M)$. If the probability is uniformly distributed, we are most uncertain about the outcome, and entropy is maximum. This will happen when the data points are uniformly distributed in the features space. On the other hand, when the data has well formed clusters, the uncertainty is low and so also the entropy. As we do not have a priori information about clusters, calculation of $p(f_1, \ldots, f_M)$ is not direct, but we can use the following way to calculate entropy without any cluster information [31].

Two points belonging to the same cluster or different clusters will contribute to the total entropy less than if they were uniformly separated. Similarity S_{i_1,i_2} between two instances X_{i_1} and X_{i_2} is high if the instances are very close and low if they are far away. Based on these two assumptions, from the work of Dash and Liu, entropy E_{i_1,i_2} will be low if S_{i_1,i_2} is either low or high and E_{i_1,i_2} will be high otherwise [31].

Since we need to work on numerical data, we only consider the formulas defined using Euclidean distance. The definition of Similarity is given by $S_{i_1,i_2} = e^{\alpha \times D_{i_1,i_2}}$, where α is the parameter $\alpha = \frac{-\ln 0.5}{\bar{D}}$,with $\bar{D}$ average distance among data points, and distance D is calculated as $D_{i_1,i_2} = [\sum\limits_{k=1}^{M} (\frac{x_{i_1k} - x_{i_2k}}{\max_k - \min_k})^2]^{1/2}$. The interval in the k^{th} dimension is normalized by dividing it by the maximum interval $(\max_k - \min_k)$ before calculating the distance. For every couple of points X_{i_1} and X_{i_2}, Entropy is calculated as $E = -S_{i_1,i_2} \times \log S_{i_1,i_2} - (1 - S_{i_1,i_2}) \times \log(1 - S_{i_1,i_2})$, which assumes the maximum value of 1.0 for $S_{i_1,i_2} = 0.5$; and the minimum value of 0.0 for $S_{i_1,i_2} = 0$ and $S_{i_1,i_2} = 1$. With the previous definitions, Dash and Liu provide the formula for a dataset of N data point as:

$$E = -\sum_{i_1=1}^{N} \sum_{i_2=1}^{N} (S_{i_1,i_2} \times \log S_{i_1,i_2} + (1 - S_{i_1,i_2}) \times \log(1 - S_{i_1,i_2}))$$

Using the algorithm introduced in 4, we can now rank every feature based on their effect on Entropy. We remove each feature and compute E; if the removal of a feature results in minimum E the feature is the least important, and vice versa. To select the best subset of feature, we use the same approach used in

Algorithm 4: Entropy feature selection

```
Input: Features $M$
Output: Best features $M$
for $(i = 0)$ to $|M|$ do
    $P_i = CalcEnt(M_{/i})$
end
$base\_entropy = CalcEnt(M)$
$M = M/\{M_k \mid P_k \leq base\_entropy\}$
return $M$
```

the SIMBA methodology [2], which consists in keeping only the features that once dropped produce an increment in Entropy, compared to the base Entropy computed with all the features.

Despite the optimization done by exploiting the symmetry of the similarity and distance matrixes, the algorithm still performed badly on large datasets; in particular in the cases with thousands of data points, given the time complexity of $O(M * N^2)$. For this reason, we implemented the scalable version of the algorithm presented in the same work.

The new method is based on random sampling to perform the original algorithm on smaller datasets, and removes the complexity dependency on the number of data points. This version leverages the fact that a reasonably small random sample retains the original cluster structure in most cases [31, 45], which is a necessary condition for good entropy performances. We perform the new algorithm 35 times with samples of 100 points as default values. These parameters allow us to keep the execution time in the range of a few minutes regardless of the size of the datasets. After this procedure, we have a ranking of the features and again we select the ones that produce an increment of Entropy.

Algorithm 5: Random sampling Entropy feature selection

```
Input: Features $M$, number of iterations $iter$
Output: Best features $M$
for $(i = 1)$ to $iter$ do
    $S = GetSample(M)$
    for $(i = 0)$ to $|M|$ do
        $P_i = P_i + CalcEnt(S_{/i})$
    end
    $base\_entropy = base\_entropy + CalcEnt(S)$
end
$M = M/\{M_k \mid P_k \leq base\_entropy\}$
return $M$
```

5.3.2 Clustering algorithms

The two alternative algorithms for clustering are the ones we presented in section 3.3:

- Agglomerative hierarchical;
- K-Means.

The possibility to use different clustering algorithms allows the users to select the one that best suits their needs. Especially, as we said before when talking about space and time complexity of both algorithms, the use of only hierarchical clustering would limit the analysis to small datasets. For both algorithm we use the functions implemented in scikit-learn. The parameters we need to define for hierarchical are:

- n_clusters: represents the number of clusters we want to obtain and needs to be specified by the user;
- affinity: distance measure between the sample. We use the Euclidean distance as default;
- linkage: distance measure to use between clusters. The algorithm merges the pairs of clusters that minimize this criterion. The parameter is set to "complete", which means we use the maximum distance between clusters.

For K-Means we only need to specify the number of clusters selected by the users and the other parameters are set to default values.

The choice to scale the dataset is left to the user and it can be selected between standardization and normalization, using respectively the classes StandardScaler() and MinMaxScaler() from scikit-learn.

5.3.3 Graphs

To simplify the choice of the number of clusters, it is possible to plot in a separate window two different graphs using matplotlib library [46]. The first one is the dendrogram showing the tree of hierarchical clustering. The other one is a WSS and BSS graph showing the trends of both values for a number of clusters defaulted to 1 to 19.

5.4 Evaluation implementation

In this section, we show all the experiments carried out in the current session, with a recap of the settings and performances of the experiments and the possibility to save and load them.

5.4.1 Indexes and score

To evaluate the quality of the experiments we calculate two indexes and the comparisons among experiments on the same dataset. The indexes are the internal metrics Silhouette coefficient and Davis-Bouldin index, both computed using the functions Silhouette_Score() and davies_bouldin_score() from scikit-learn.

To make a direct comparison among two or more experiments to see how many samples have been grouped in the same way in all the selected experiments, we compute the comparison matrix with the algorithm shown below. The comparison can only be used on experiments performed on datasets with the same number of data points. To avoid any mistake by the user, the plugin already filters the experiments compatible with the selected one and reports them in the score section.

Comparison_matrix is a symmetric matrix of dimension M where M is the number of points for the dataset. Taking as input a list of cluster results,

Algorithm 6: Comparison matrix computation

```
Input: $cluster\_labels$
Output: $comparison\_matrix$
$num\_points = len(cluster\_labels.columns[0])$
Initialize $comparison\_matrix$ with 0
for $(i = 0)$ to $(num\_points - 1)$ do
    for $(j = i)$ to $(num\_points - 1)$ do
        foreach $exp \in cluster\_labels$ do
            if $exp_i == exp_j$ then
                $comparison\_matrix_{ij} = comparison\_matrix_{ij} + 1$
                if $i \neq j$ then
                    $comparison\_matrix_{ji} = comparison\_matrix_{ji} + 1$
                end
            end
        end
    end
end
return $comparison\_matrix$
```

containing for every point the cluster_id for each experiment, what the matrix evaluate is how many times two points are grouped in the same cluster. For each experiment, it checks if the two points have the same cluster_id and, if they do, it increments the cell corresponding to that couple of points. Using the symmetric property of this matrix, we update two cells at a time except for the diagonal, reducing by almost half the iterations needed. For each comparison we have a matrix having values between zero and N, where N is the number of experiments selected:

- 0 means two points are never grouped in the same cluster;
- 1 to $N - 1$ means two points are in the same cluster only in some experiments;
- N means two points are always grouped together.

This means that positive cases are values 0 and N since they mean clustering in the experiments has produced the same results for that couple. We then compare different experiments using the variable score whose computation is shown below.

Score counts how many times values 0 or N occur in the comparison_matrix and it normalizes this number with the number of points squared. This means that the Score value has a range between 0 and 1, with higher values indicating a higher similarity among the experiment clusters.

Algorithm 7: Score computation

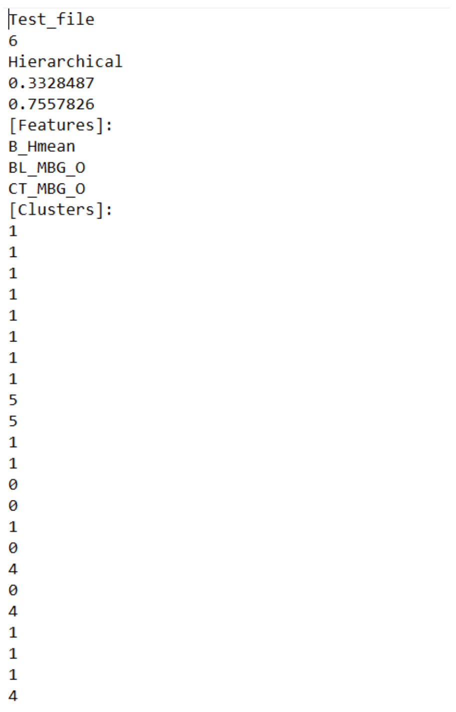
```
Input: $comparison\_matrix$,max_val
Output: Score $s$
$num\_points = len(comparison\_matrix.columns[0])$
foreach $value \in comparison\_matrix$ do
    if $value == 0 \quad OR \; value == max\_val$ then
        $s = s + 1$
    end
    $s = \frac{s}{num\_points^2}$
end
return $s$
```

5.4.2 Load and save experiments

Every experiment completed in the current session can be stored in a text file saved in the folder "Experiments" inside the plugin directory. The structure of the file is simple and contains all the information about the experiment:

- experiment name;
- number of clusters;
- clustering algorithm;
- Silhouette score;
- Davies-Bouldin index;
- list of features
- cluster labels

The experiments saved in previous sessions can be loaded in the plugin and are shown in the evaluation section along with the other experiments. An example of the file structure can be seen in image 5.4.



```
Test_file
6
Hierarchical
0.3328487
0.7557826
[Features]:
B_Hmean
BL_MBG_O
CT_MBG_O
[Clusters]:
1
1
1
1
1
1
1
1
5
5
1
1
0
0
1
0
4
0
4
1
1
1
4
```

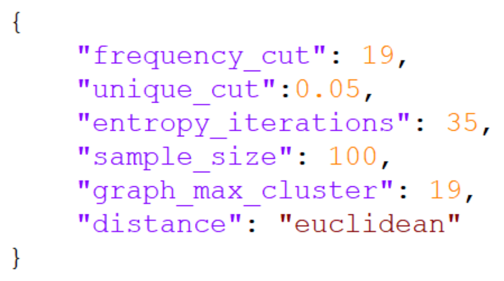
Figure 5.4: Experiment file example

5.5 Configuration file

The plugin is developed to be usable by a general user without an advanced knowledge of machine learning and software programming. For this reason, we decided to move some parameter settings in an external json file. The json file called "Configuration" is in the plugin directory and can be easily modified by any text editor. When the plugin is started we load the file and use it to initialize an instance of Configuration() class. The structure of the configuration file is shown in image 5.5 and the meaning of every variable is described below:

- frequency_cut: the threshold for the ratio of the most common value to the second most common value, used in the quasi-constant feature elimination;
- unique_cut: the threshold for the ratio of distinct values to the number of total samples, used in the quasi-constant feature elimination;
- entropy_iterations: the number of random samples used for the sampling entropy algorithm;
- sample_size: the number of points in every random sample for the sampling entropy algorithm;

- graph_max_cluster: the max number of clusters used when plotting WSS and BSS trends;
- distance: distance measure used in hierarchical clustering, the accepted values are 'euclidean' or 'manhattan'.

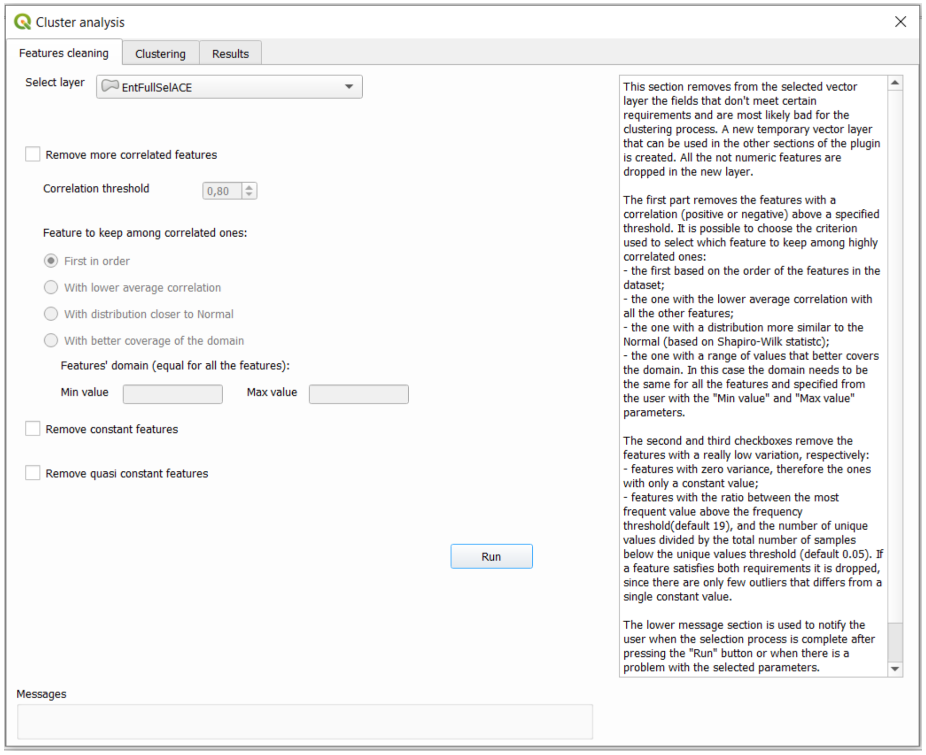


```
{
    "frequency_cut": 19,
    "unique_cut":0.05,
    "entropy_iterations": 35,
    "sample_size": 100,
    "graph_max_cluster": 19,
    "distance": "euclidean"
}
```

Figure 5.5: Configuration json file

5.6 User Interface

The user interface is enclosed in a single window containing a QTabWidget split into three tabs, one for each main functionality of the plugin. The layout of the three tabs is similar and is composed of the widgets for user inputs, a message section, and a brief user guide. The message box is used to notify the user about any error or the completion of the selected operations. In figures 5.6, 5.7, 5.8, and 5.9 we show the interface windows and some examples of notifications from the plugin.



Select layer: EntFullSelACE

Remove more correlated features

Correlation threshold: 0,80

Feature to keep among correlated ones:

- [✓ First in order]
- With lower average correlation
- With distribution closer to Normal
- With better coverage of the domain

Features' domain (equal for all the features):

Min value | Max value

Remove constant features

Remove quasi constant features

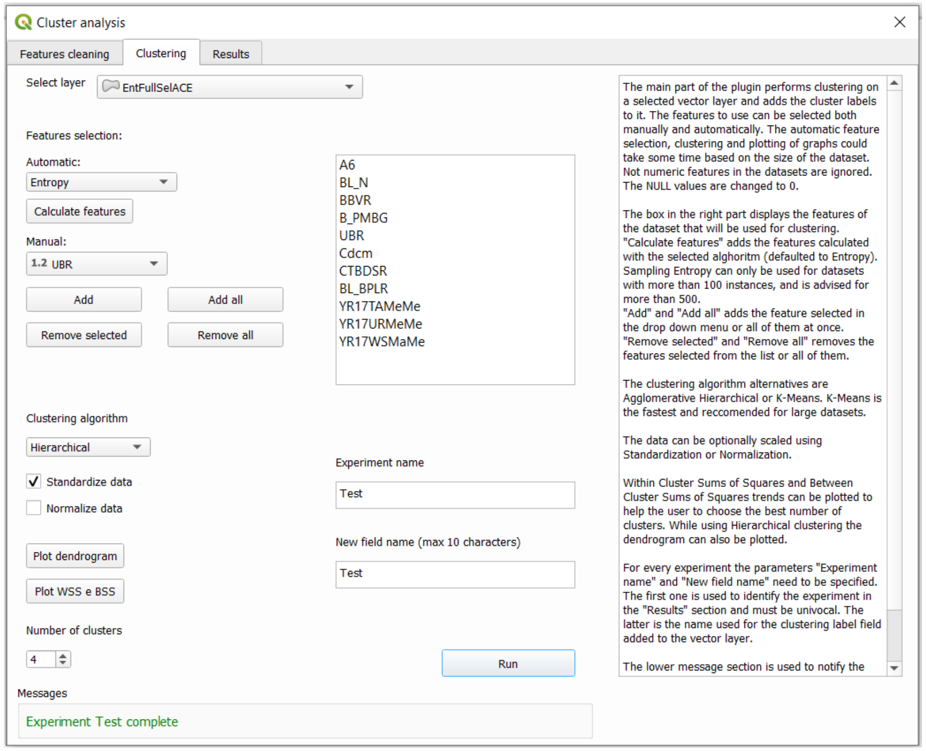
This section removes from the selected vector layer the fields that don't meet certain requirements and are most likely bad for the clustering process. A new temporary vector layer that can be used in the other sections of the plugin is created. All the not numeric features are dropped in the new layer.

The first part removes the features with a correlation (positive or negative) above a specified threshold. It is possible to choose the criterion used to select which feature to keep among highly correlated ones:
- the first based on the order of the features in the dataset;
- the one with the lower average correlation with all the other features;
- the one with a distribution more similar to the Normal (based on Shapiro-Wilk statistc);
- the one with a range of values that better covers the domain. In this case the domain needs to be the same for all the features and specified from the user with the "Min value" and "Max value" parameters.

The second and third checkboxes remove the features with a really low variation, respectively:
- features with zero variance, therefore the ones with only a constant value;
- features with the ratio between the most frequent value above the frequency threshold(default 19), and the number of unique values divided by the total number of samples below the unique values threshold (default 0.05). If a feature satisfies both requirements it is dropped, since there are only few outliers that differs from a single constant value.

The lower message section is used to notify the user when the selection process is complete after pressing the "Run" button or when there is a problem with the selected parameters.

Figure 5.6: Features cleaning section UI



Select layer: EntFullSelACE

Automatic: Entropy

Manual: 1.2 UBR

A6
BL_N
BBVR
B_PMBG
UBR
Cdcm
CTBDSR
BL_BPLR
YR17TAMeMe
YR17URMeMe
YR17WSMaMe

Clustering algorithm: Hierarchical

[✓] Standardize data

[ ] Normalize data

Number of clusters: 4

Experiment name: Test

New field name (max 10 characters): Test

Messages

Experiment Test complete

The main part of the plugin performs clustering on a selected vector layer and adds the cluster labels to it. The features to use can be selected both manually and automatically. The automatic feature selection, clustering and plotting of graphs could take some time based on the size of the dataset. Not numeric features in the datasets are ignored. The NULL values are changed to 0.

The box in the right part displays the features of the dataset that will be used for clustering. "Calculate features" adds the features calculated with the selected alghoritm (defaulted to Entropy). Sampling Entropy can only be used for datasets with more than 100 instances, and is advised for more than 500.
"Add" and "Add all" adds the feature selected in the drop down menu or all of them at once. "Remove selected" and "Remove all" removes the features selected from the list or all of them.

The clustering algorithm alternatives are Agglomerative Hierarchical or K-Means. K-Means is the fastest and reccomended for large datasets.

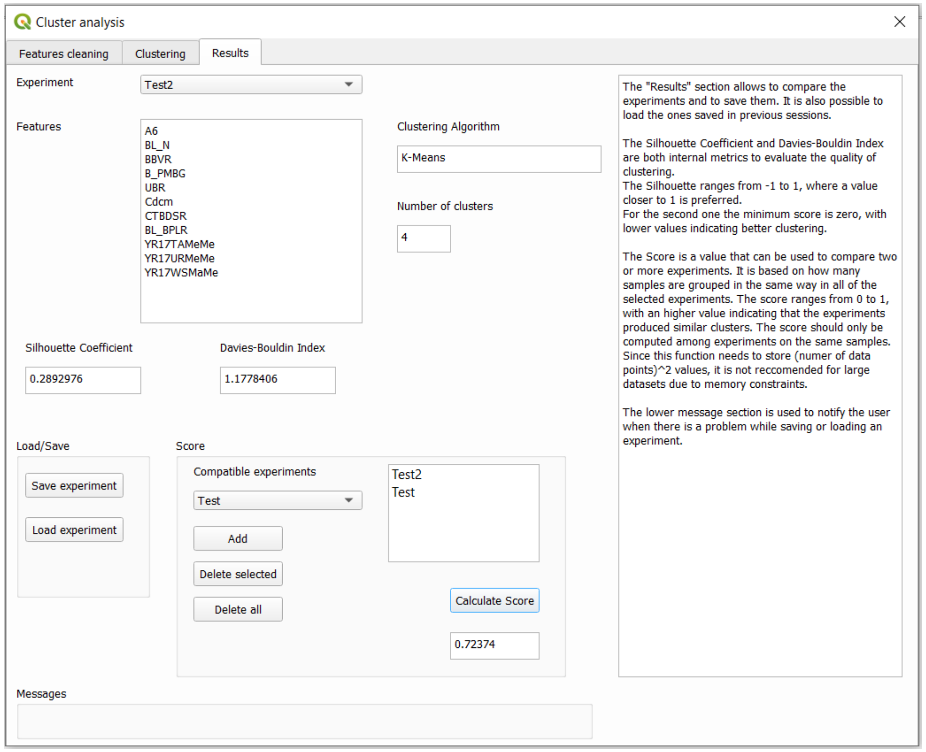
The data can be optionally scaled using Standardization or Normalization.

Within Cluster Sums of Squares and Between Cluster Sums of Squares trends can be plotted to help the user to choose the best number of clusters. While using Hierarchical clustering the dendrogram can also be plotted.

For every experiment the parameters "Experiment name" and "New field name" need to be specified. The first one is used to identify the experiment in the "Results" section and must be univocal. The latter is the name used for the clustering label field added to the vector layer.

The lower message section is used to notify the

Figure 5.7: Clustering section UI


Cluster analysis

Features cleaning | Clustering | Results

Experiment: Test2

Features:
A6
BL_N
BBVR
B_PMBG
UBR
Cdcm
CTBDSR
BL_BPLR
YR17TAMeMe
YR17URMeMe
YR17WSMaMe

Clustering Algorithm: K-Means

Number of clusters: 4

Silhouette Coefficient: 0.2892976

Davies-Bouldin Index: 1.1778406

Load/Save
Save experiment
Load experiment

Score
Compatible experiments: Test
Add
Delete selected
Delete all

Test2
Test

Calculate Score
0.72374

Messages

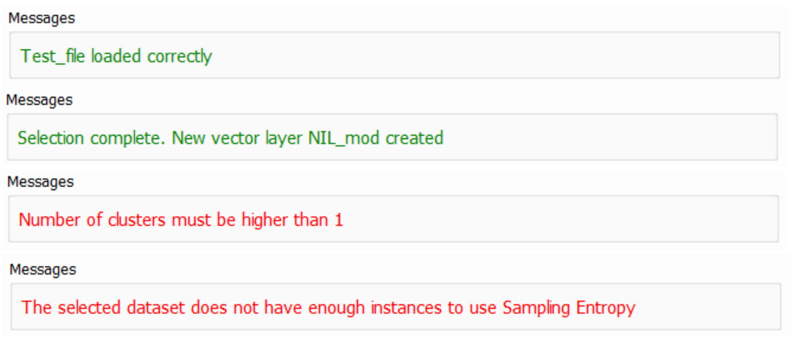
The "Results" section allows to compare the experiments and to save them. It is also possible to load the ones saved in previous sessions.

The Silhouette Coefficient and Davies-Bouldin Index are both internal metrics to evaluate the quality of clustering.
The Silhouette ranges from -1 to 1, where a value closer to 1 is preferred.
For the second one the minimum score is zero, with lower values indicating better clustering.

The Score is a value that can be used to compare two or more experiments. It is based on how many samples are grouped in the same way in all of the selected experiments. The score ranges from 0 to 1, with an higher value indicating that the experiments produced similar clusters. The score should only be computed among experiments on the same samples. Since this function needs to store (numer of data points)^2 values, it is not reccomended for large datasets due to memory constraints.

The lower message section is used to notify the user when there is a problem while saving or loading an experiment.

Figure 5.8: Experiments evaluation section UI


Messages
Test_file loaded correctly

Messages
Selection complete. New vector layer NIL_mod created

Messages
Number of clusters must be higher than 1

Messages
The selected dataset does not have enough instances to use Sampling Entropy

Figure 5.9: Example of notification messages

Chapter 6

Experiments

In this chapter, we will describe some of the experiments we made on the datasets from Chapter 4 and the settings of the plugin used. The main objective of these experiments is not to evaluate the performance of a particular methodology or algorithm, but rather to show the potential of the developed tool to analyse data of different nature and, most important, of different sizes, up to several tens of thousands of data points. Furthermore, the experiments are also essential to identify and understand the shortcomings of the plugin. For these reasons, we preferred to perform analysis on real and complete use cases. All the analyses are performed with the help of experts in the field of geospatial data, and in particular with a great knowledge of the city of Milan.

Since the datasets used are of different types and sizes, the parameters and purposes vary a lot between the experiments. Some settings of the plugin are common for all the experiments, in detail:

- feature correlation: the correlation threshold is set to 0.80 and the criterion to keep features is average correlation;
- quasi constant feature: the thresholds are 19 for frequency and 0.05 for unique values;
- sampling entropy: 35 iterations of the algorithm with samples of 100 points;
- distance measure: the measure for hierarchical clustering is Euclidean. We repeated the experiments with Manhattan distance and the results were similar or equal and they will not be reported here;
- data scaling: most of the features in the datasets have a different scale, since we want to have all the features with the same weight we decided to scale the data in every experiment. The choice is standardization instead of normalization to not suppress the effect of outliers [47];
- K-Means graphs: given the random nature of K-Means, the WSS-BSS graph produces different trends on each run. After trying multiple runs on

a variety of datasets and settings, we concluded that these differences are negligible, and the graph shown for each experiment is selected without a specific criterion.

6.1 Climate experiments

The first type of experiments carried out is on climate datasets. The goal for this analysis was to create a division of the city based on climate parameters and understand which variables work better for the task. Initially, we performed the analysis on the dataset with the highest spatial resolution, and after selecting the best features we used them on the three different granularities of the city. The value of the attributes assigned to each geometry corresponds to the mean of the data points inside its area. The partitions of ACE, NIL, and Block will be compared later with the ones created with urban attributes.

6.1.1 100m resolution

As previously said these datasets are composed of 17877 data points and a lot of features; these dimensions limit the possibilities of the plugin. For this reason, the clustering algorithm in the following experiments is K-Means and for feature selection the sampling version of entropy. Moreover, the elbow method is not applicable, and the number of clusters is selected based on what we considered appropriate for the type of analysis, which is between 2 and 10.

The first experiments are on the annually aggregated dataset, both using all the features and the ones selected automatically. Entropy algorithm selected from every year only the attributes relative to:

- mean temperature
- mean relative humidity
- max wind speed

and performed significantly better, as we will show in section 7.1.

After this, we repeated both experiments on single years, specifically the two most recent, 2016 and 2017. The results confirmed the performance increase with only the automatically selected feature and the selection of the same three variables in 2017 and only the first two in 2016.

The second and third datasets analysed are the monthly aggregated ones for 2016 and 2017. The features selected are reported in table 6.1. The trends of the internal metrics with feature selection are consistent with the yearly aggregated data.

Finally, we analysed monthly data for both 2016 and 2017 using single variables. The results confirmed our expectations since there was a wide gap between the indices of mean temperature and humidity and their mean counterparts. The wind speed values had more moderate differences between the two, but the max attribute turned out to be preferable.

Variable	2016 months selected	2017 months selected
Max temperature	0	0
Mean temperature	12	12
Max relative humidity	0	0
Mean relative humidity	12	12
Max wind speed	11	12
Mean wind speed	5	9

Table 6.1: Monthly Variables from automatic selection for 2016 and 2017

For the next experiments on different granularities, we decided to use data from the three best features, aggregated monthly to avoid suppressing differences that could be caused by averaging over the entire year. Overall, the 2016 and 2017 data did not present clear differences from the point of view of the performance and the groups created, so we selected the most recent year.

6.1.2 ACE climate

This dataset is considerably smaller than the previous ones, this means that we can also use hierarchical clustering as well as the dendrogram and the elbow method. The dendrogram and the WSS-BSS trends for hierarchical clustering are shown in figure 6.1 and 6.2. With the information from the two graphs, we decide to analyse 2, 3, and 5 clusters.

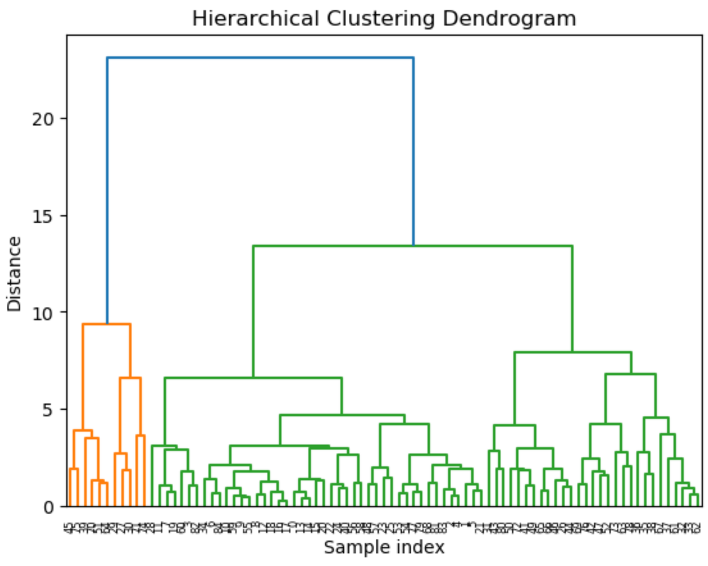


Figure 6.1: ACE climate data dendrogram

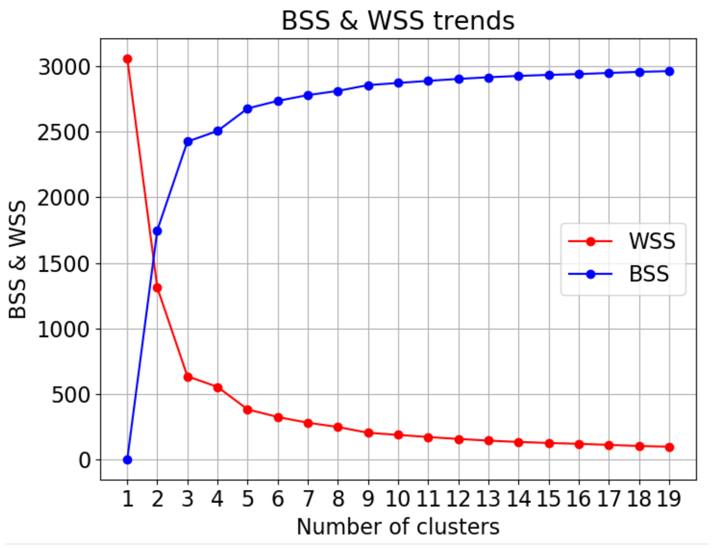


Figure 6.2: ACE climate data BSS and WSS trends for hierarchical clustering

The graph from K-Means, shown in figure 6.3, does not provide a clear choice, so we run the algorithm with the number of clusters set from 2 to 7.

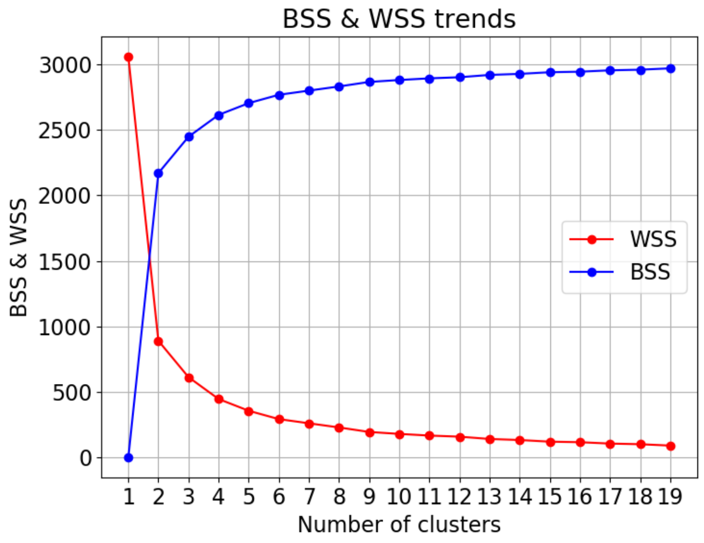


Figure 6.3: ACE climate data BSS and WSS trends for K-Means clustering

6.1.3 NIL climate

From the dendrogram in figure 6.4 and the steps in the trend from figure 6.5, we set the number of clusters for hierarchical clustering to 2, 3, and 6.

As before, in figure 6.6 K-Means presents a graph without steep changes, and we decide to cluster with 3 to 6 groups.

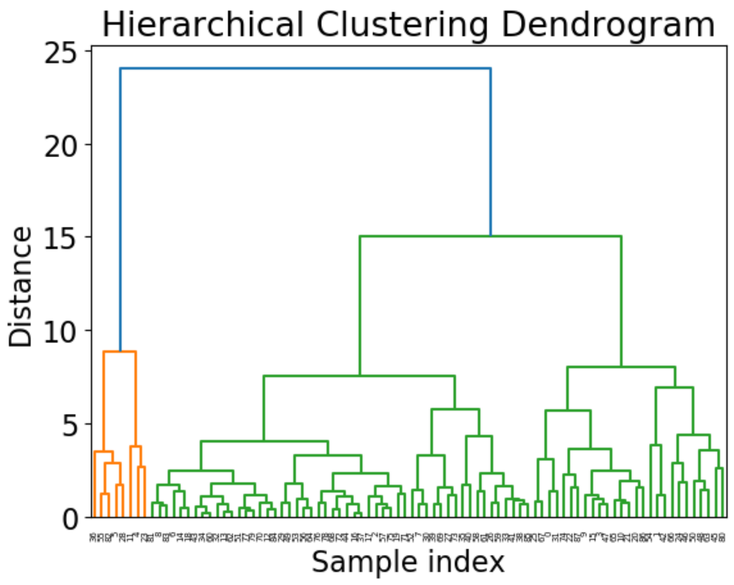


Figure 6.4: NIL climate data dendrogram

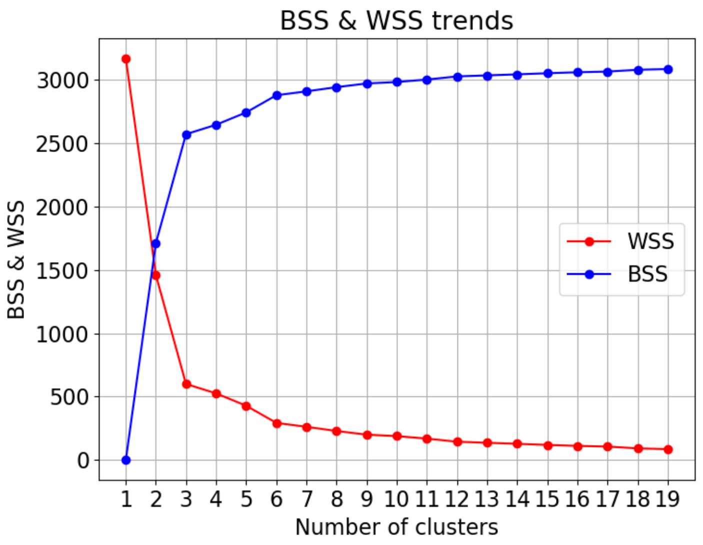


Figure 6.5: NIL climate data BSS and WSS trends for hierarchical clustering

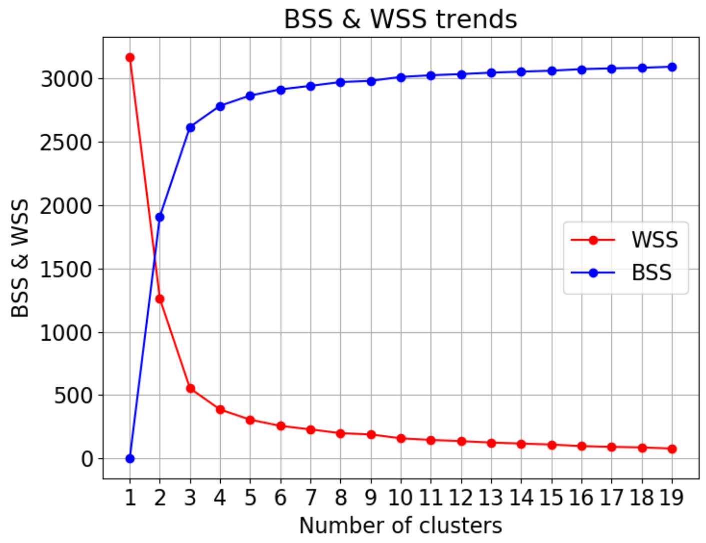


Figure 6.6: NIL climate data BSS and WSS trends for K-Means clustering

6.1.4 Block climate

As said before, one of the focuses on the analysis of the climate partitions on NIL, ACE and Block geometries is to make a comparison with the urban analysis. At the Block granularity, hierarchical clustering had the tendency to find few outliers and put the rest of the points in a big cluster, as shown in the dendrogram in figure 6.7. Since these outliers are mostly meaningless for our purpose, we only used K-Means.

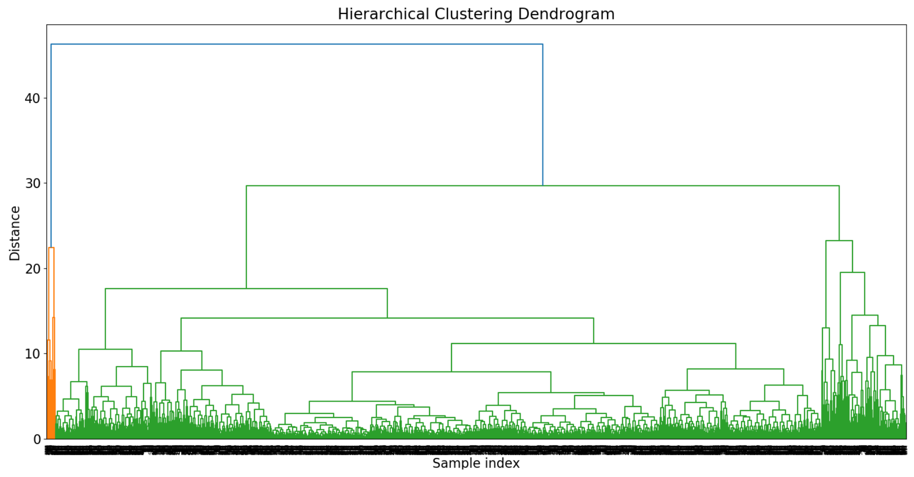


Figure 6.7: Block climate data dendrogram

Once again, the K-Means graph in figure 6.8 doesn't show a clear indication of the best number of clusters, and we select 3 to 7.

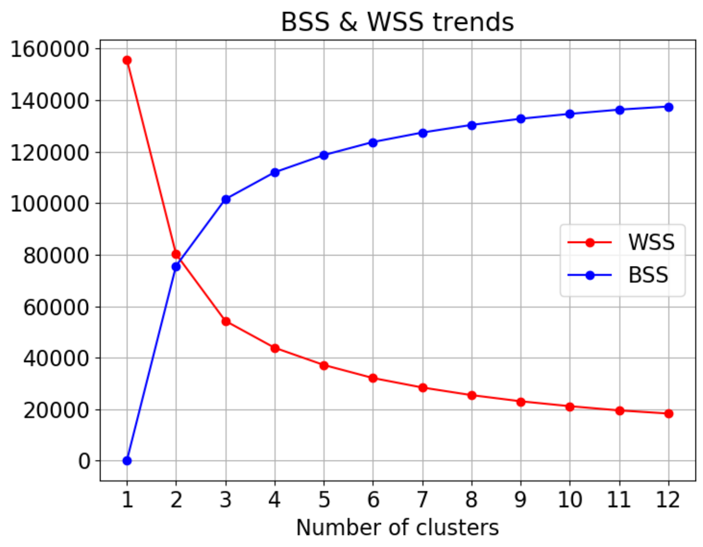


Figure 6.8: Block climate data BSS and WSS trends for K-Means clustering

6.2 Urban experiments

The experiments on the Urban dataset are done on a set of automatically selected features and the obtained partitions will be compared to the ones from the experiments on climate data. Other than understanding the similarities between the zones of the city based on their urban characteristic, the purpose of this analysis is to understand the influence of the parameters on climatic variables. The set of features is selected by removing the highly correlated features and the quasi constant ones, and then using entropy on the remaining features. Before dropping any attributes, we removed from the datasets the information about the size of the geometries (Area and Perimeter), as they are not of any interest in this type of analysis. For ACE and NIL, the entropy algorithm is performed on the entire dataset, while for Block the random sampling version is used due to its size. As for the previous experiments, we used both hierarchical and K-Means for ACE and NIL and only K-Means for Block.

6.2.1 ACE urban

The process of cleaning and feature selection for this dataset selected 8 features. The dendrogram from figure 6.9 and the graph in figure 6.10 suggest 2, 4 and 6 clusters. With 2 clusters there is a group of only 2 points corresponding to the city centre, and in this data we are not interested in small clusters of outliers. For this reason, we take 3 clusters instead, which also simplifies the comparison with the climate partition.

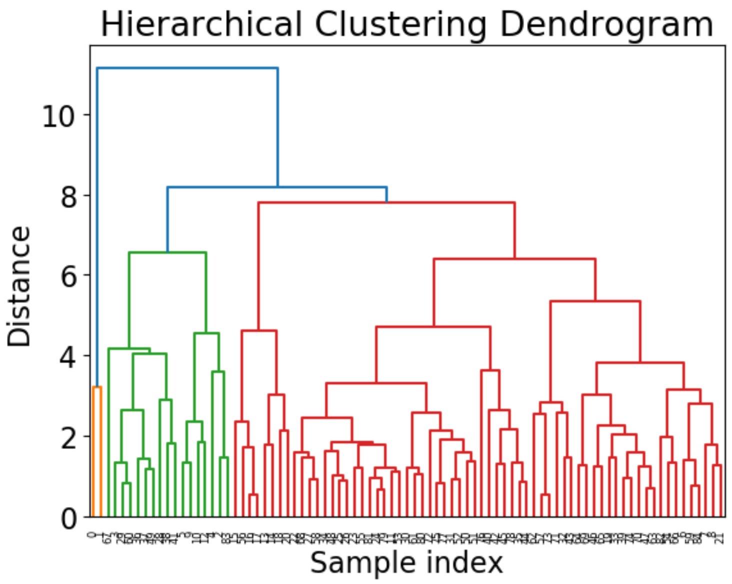


Figure 6.9: ACE urban data dendrogram

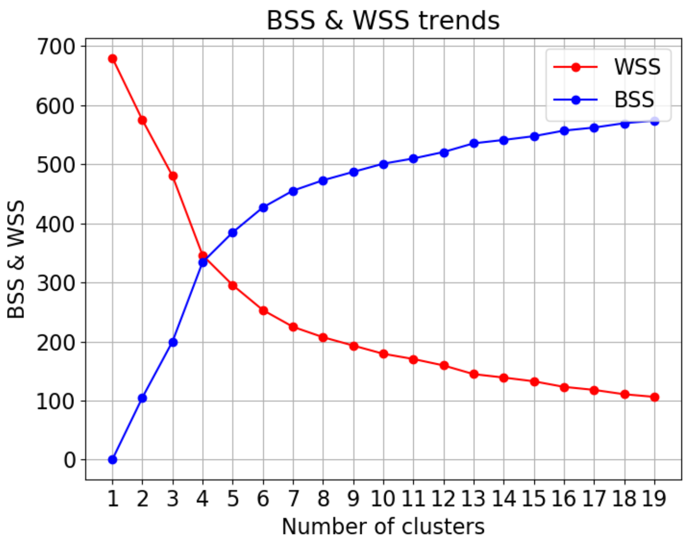


Figure 6.10: ACE urban data BSS and WSS trends for hierarchical clustering

K-Means, as usual, provided a graph with no clear indication. The decision is to take 7 clusters, since the curve in figure 6.11 starts to flatten more at that point.

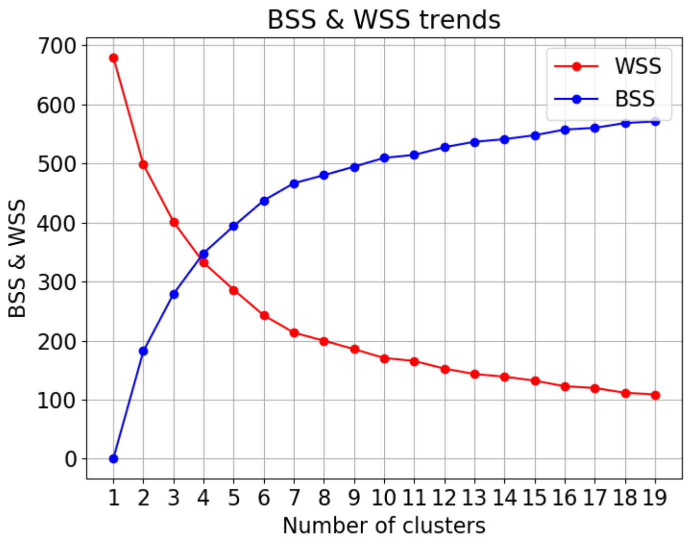


Figure 6.11: ACE urban data BSS and WSS trends for K-Means clustering

6.2.2 NIL urban

In this dataset, after the full selection, the features remained are 12 out of the initial 107. From figure 6.12 we can see that hierarchical clustering formed mostly small clusters. To find a larger division we have to select 6 clusters, which is also a coherent choice with the trends from figure 6.13.

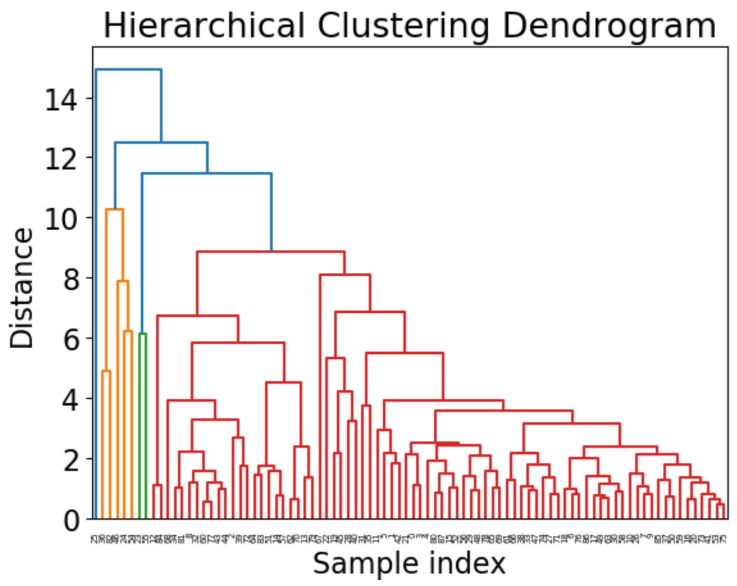


Figure 6.12: NIL urban data dendrogram

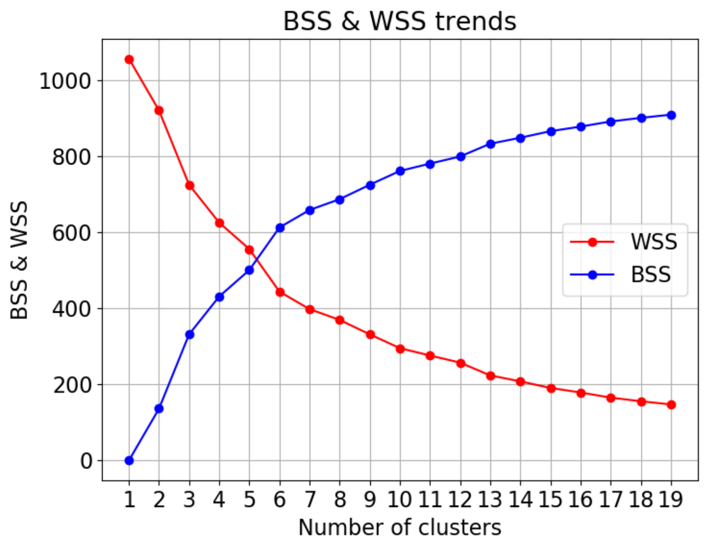


Figure 6.13: NIL urban data BSS and WSS trends for hierarchical clustering

K-Means is used again with a number of clusters set to 7, the WSS-BSS is shown in figure 6.14.

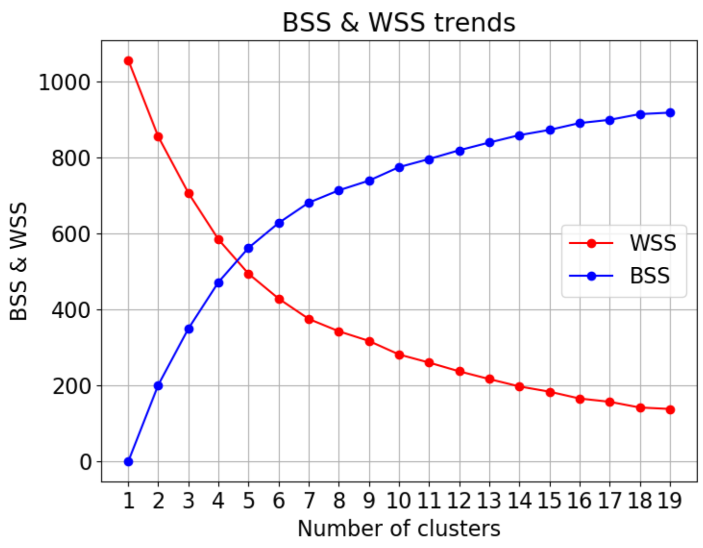


Figure 6.14: NIL urban data BSS and WSS trends for K-Means clustering

6.2.3 Block urban

For this dataset, the automatic procedure kept 19 features, more than in the other cases. In figure 6.15 the WSS-BSS trends show steps at 4 and 6-7 cluster, and these are the numbers that we used.

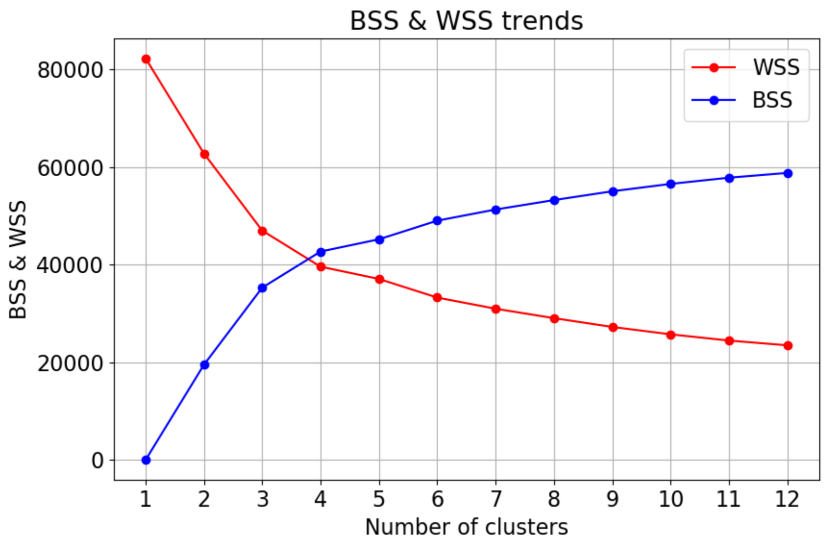


Figure 6.15: Block urban data BSS and WSS trends for K-Means clustering

6.3 Building experiments

For the analysis on the Building dataset, we decided to take a different approach and try experiments on both automatically selected features and on a set of manually selected ones. For the automatic selection we removed at first the correlated and quasi constant features, and after performed entropy with sampling on the remaining ones. At the end of the process, the system selected 6 features. The experts, instead, selected a set of 10 features. Unlike in the previous types of data, for the buildings it is possible to have some information on the classes that can be created. For this reason, the number of clusters created is not based on data but on human knowledge. The numbers selected are 4, 6, 10, and 12. The comparison between the two methods, as shown in section 7.3, is done with the use of internal indexes, score and by manually looking at the classes to understand how they are created and if they have real meaning.

6.4 Demographic and social experiments

In section 4.3 we said that some of the demographics and social indicators are calculated on specific classes. For this reason, we decided to perform clustering on sets of features manually selected to analyse some of the classes individually. Since in this analysis we are particularly interested in finding zones of the city that can be considered outliers, we used hierarchical as it is more suited for the task than K-Means.

6.4.1 Young people education and occupation

The first set of features is relative to young people and their education and unemployment, i.e.:

- mean age;
- young people (15-19) without middle school diploma;
- young people (15-24) without high school graduation;
- young people (15-29) neither in employment nor education.

The numbers of clusters selected, supported by the graphs in figure 6.16 and 6.17, are 2, 4, 5, and 7.

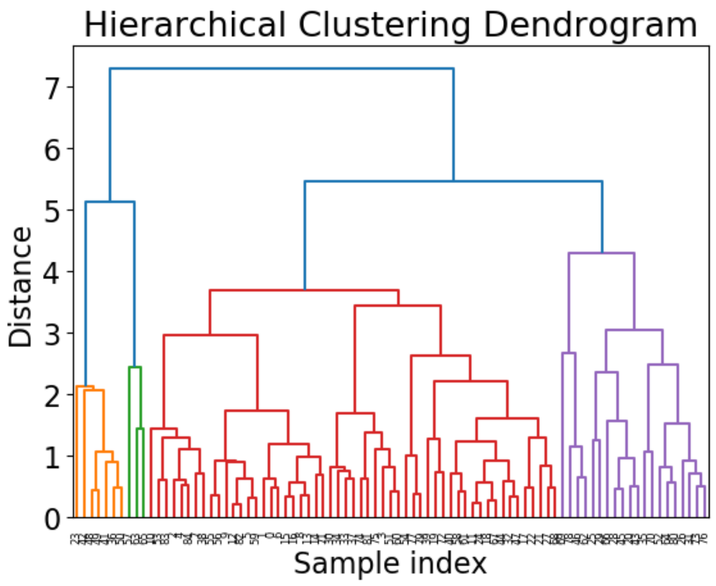


Figure 6.16: Young people data dendrogram

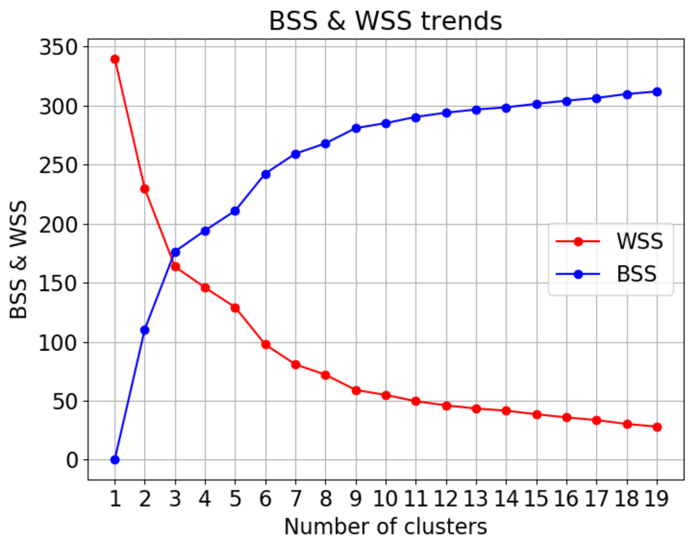


Figure 6.17: Young people data BSS and WSS trends for hierarchical clustering

6.4.2 Housing overcrowding

In this group we included data about population distribution and the composition of the families, i.e.:

- average house size;
- population density;
- average family components;
- families with 5 or more components.

The numbers of clusters selected are 2, 3, and 4 by looking at the dendrogram in figure 6.18 and 7, where the trends of WSS-BSS in figure 6.19 is flattening. Another possible value suggested by the graph, would be 9 clusters, but in this case we were not interested in dividing the city into so many areas.

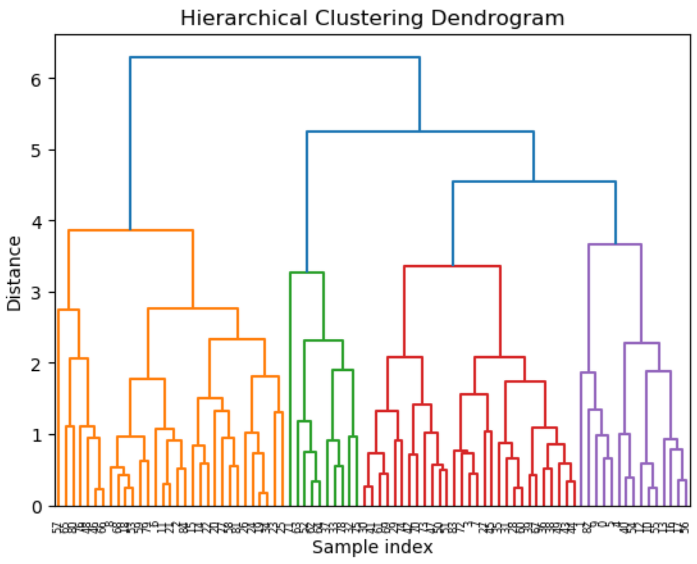


Figure 6.18: Families data dendrogram

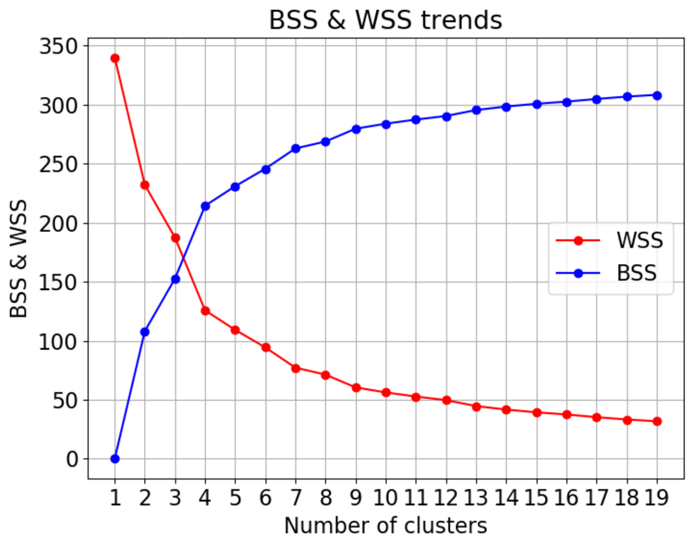


Figure 6.19: Families data BSS and WSS trends for hierarchical clustering

6.4.3 Population isolation

In the last set of features we included mainly attributes about people living alone of various ages, i.e.:

- people over 65 years old living alone;
- young people (15-34) living alone;
- families composed by only one person;
- average family components.

In this experiment, the clusters selected are 2, 3, and 7; in figure 6.20 and 6.21 we can see the dendrogram and the graph.

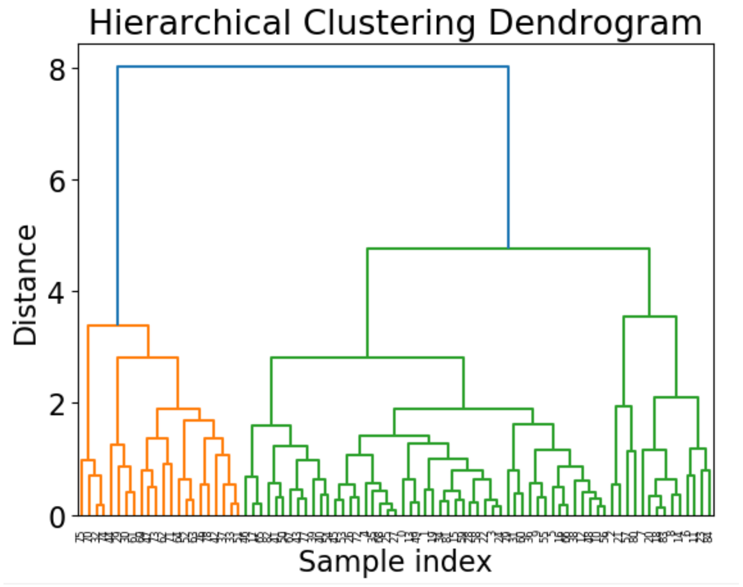


Figure 6.20: Alone people data dendrogram

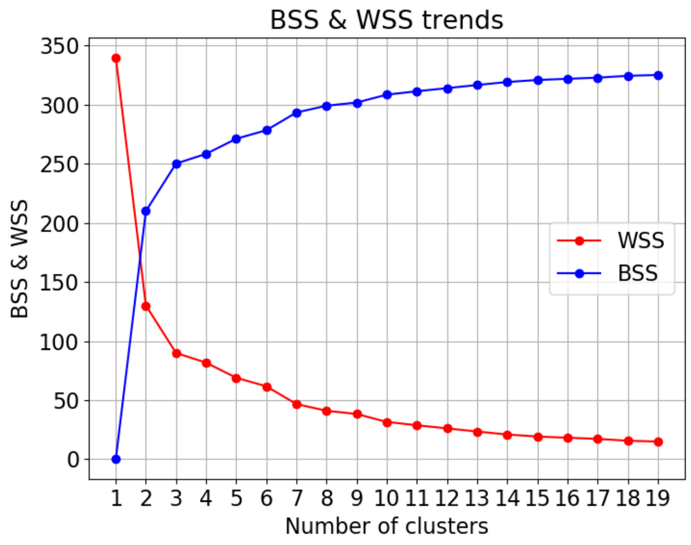


Figure 6.21: Alone people data BSS and WSS trends for hierarchical clustering

Chapter 7

Experiments evaluation

In this chapter, we will analyse the results of the experiments described in chapter 6. As we said previously the method of evaluation will vary based on the analysis characteristic. Since we performed a great number of different experiments, we will try to summarize the results and highlight some important points. With the help of an expert, we also try to understand some of the maps that were created from the experiments. Since the plugin does not provide information about the cluster distance, the colours used for the maps don't follow any precise criterion but are selected for better visualisation.

7.1 Climate results

In this section, we show the results of the climate experiments on the four granularities. First, we talk about the yearly and monthly data and the performance comparisons of experiments without and with entropy feature selection. After this, we will see the behaviour of the selected climate variables on the other divisions of the city.

7.1.1 100m resolution

The experiments and results reported are only for numbers of clusters between 2 and 5, since between 6 and 10 the performance decreased rapidly, and the maps were similar with few small differences. In table 7.1, 7.2, and 7.3 we can see the internal metrics and how the entropy feature selection improved the performance of clustering for the three cases: 2008-2017, 2016, and 2017 yearly aggregated data. In table 7.4, instead, we can see the trends of the metrics for the data from 2017 monthly aggregated.

N clusters	All features		Feature selection	
	Silhouette	**Davies-Bouldin**	**Silhouette**	**Davies-Bouldin**
2	0.5423	0.7678	0.6632	0.4974
3	0.3646	1.1382	0.5117	0.7752
4	0.3622	1.0661	0.4328	0.8564
5	0.3639	0.9297	0.4339	0.8367

Table 7.1: Internal metrics for experiments on yearly data from 2008-2017

N clusters	All features		Feature selection	
	Silhouette	**Davies-Bouldin**	**Silhouette**	**Davies-Bouldin**
2	0.5268	0.8691	0.71856	0.4109
3	0.5102	0.9150	0.6000	0.5283
4	0.3891	1.0761	0.5706	0.5514
5	0.3340	1.1200	0.5334	0.6042

Table 7.2: Internal metrics for experiments on yearly data from 2016

N clusters	All features		Feature selection	
	Silhouette	**Davies-Bouldin**	**Silhouette**	**Davies-Bouldin**
2	0.5786	0.6505	0.6761	0.4775
3	0.4256	0.9895	0.5231	0.7529
4	0.3764	1.0212	0.4382	0.8372
5	0.3354	1.0857	0.4378	0.8172

Table 7.3: Internal metrics for experiments on yearly data from 2017

N clusters	All features		Feature selection	
	Silhouette	**Davies-Bouldin**	**Silhouette**	**Davies-Bouldin**
2	0.5537	0.6952	0.6415	0.5559
3	0.3708	1.1011	0.5289	0.7700
4	0.3290	1.1602	0.4072	0.8896
5	0.2895	1.2507	0.4072	0.8908

Table 7.4: Internal metrics for experiments on monthly data from 2017

For the single variables, we only performed experiments with 3 clusters, and in table 7.5 we can see the internal metrics of each one.

Variable	**Silhouette**	**Davies-Bouldin**
Max temperature	0.2878	1.2166
Mean temperature	0.5699	0.5733
Max relative humidity	0.4059	0.9070
Mean relative humidity	0.5782	0.5843
Max wind speed	0.5851	0.5299
Mean wind speed	0.5644	0.5624

Table 7.5: Internal metrics for experiments on monthly single climate variables from 2017

The indexes convinced us to use for the other climate experiments only the features that performed better and were selected by the automatic processes. While the choice was clear between max and mean for temperature and humidity, for wind speed we had to also look at the maps. The final choice was max wind speed as it better captured the morphology of the city and identified better solids and voids. Moreover, the mean wind speed is less meaningful since Milan is not a windy city.

In figure 7.1 we show one of the maps created with 2017 monthly aggregated data, using the three selected features and 5 clusters. Compared with the map of Milan, the analysis correctly captured the morphology of the city, separating the big rural areas in the outskirts, the urban part of the centre and the less built areas. Furthermore, the tool also identified more subtle differences in the central area, such as the voids of the railway yards and the green areas inside the city, both large and small. In image 7.2 there is a raster map [48] of Milan that shows the Local Climate Zones [49] of the city, which is a classification of the areas based mainly of surface structure, surface cover, and human activities.

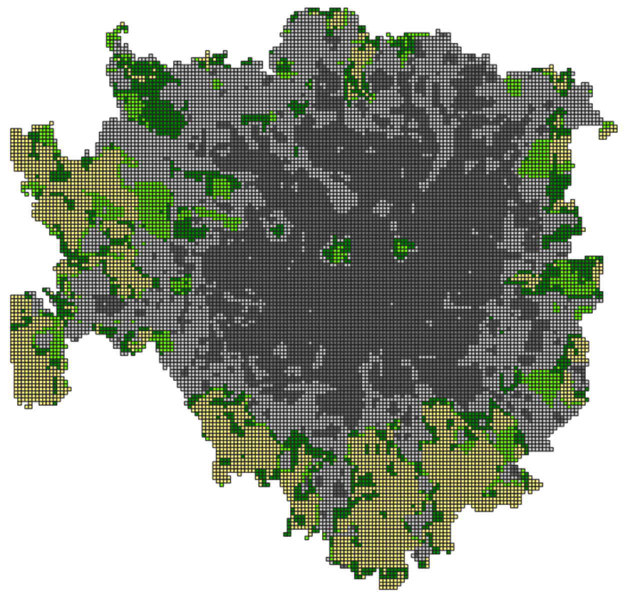


Figure 7.1: Map of monthly climate data from 2017 with 5 clusters

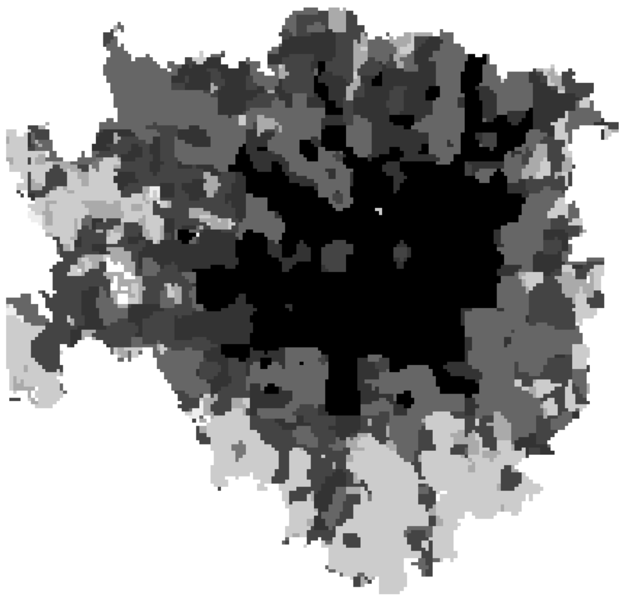


Figure 7.2: Local Climate Zones Milan

7.1.2 ACE climate

On the ACE dataset, hierarchical clustering separated the urban area of the city centre and created some clusters in the suburban areas with 3 and 5 clusters. With 2 clusters, most of the areas are in the same group and the few mostly green areas in a separate one. The same also applies to K-Means up to 5 clusters, while for 6 and 7 clusters it also found a partition in the centre. From the experiments, we can state that this granularity is not suited to create climate partitions. One of the reasons is probably that this dataset is missing the rural and peripheral areas where we can find a different climate. The other is that the urban parks are merged with urban blocks in the same polygon, and when the mean is computed for the entire area, the differences are suppressed. In images 7.3 and 7.4 we show some of the maps for hierarchical and K-Means.

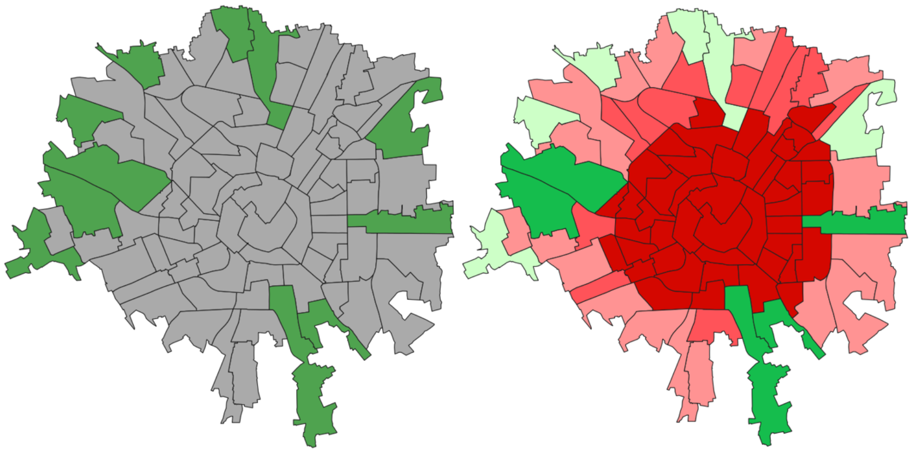


Figure 7.3: Hierarchical clustering on ACE climate data with 2 and 5 clusters

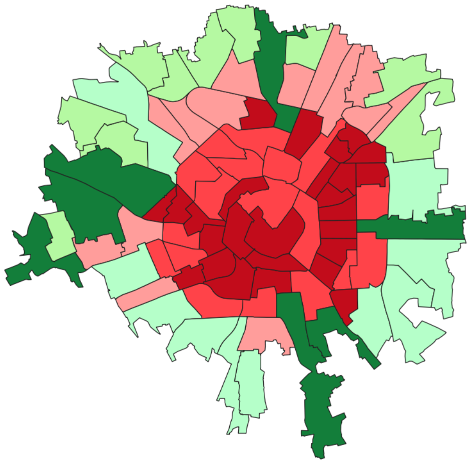


Figure 7.4: K-Means clustering on ACE climate data with 6 clusters

7.1.3 NIL climate

As before, most of the city centre is in the same cluster in both hierarchical and K-Means at different number of clusters, but the urban parks have been recognized and grouped with other green zones. We can also see that the big agricultural parks in the south and the west are always together in their own cluster. In image 7.5 we show a map for the two different clustering algorithms, one with hierarchical at 3 clusters and the other with K-Means at 5.

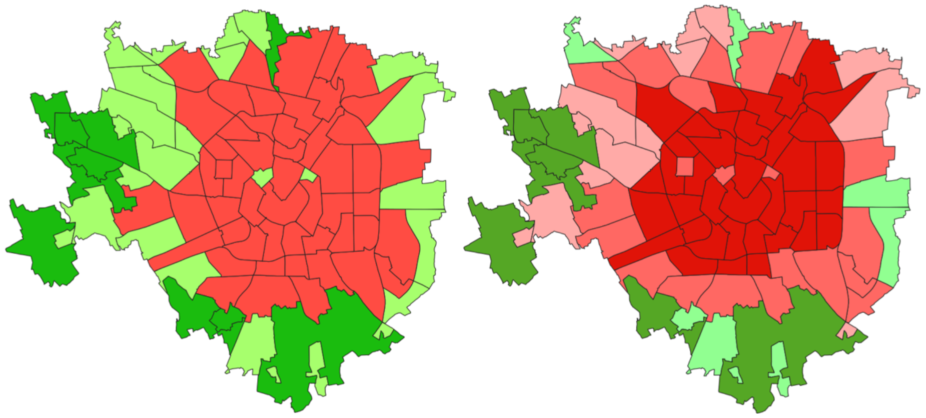


Figure 7.5: Hierarchical and K-Means clustering on NIL climate data with 3 and 5 clusters

7.1.4 Block climate

For the blocks, the clustering behaved similarly to the 100m resolution; this is expected given the finer granularity of the geometries. In image 7.6, the 3 clusters show that the analysis identified three zones with different climatic attributes composed as: mostly green areas inside and outside the city, areas with sparse buildings and some small green sections, areas covered almost exclusively by buildings.

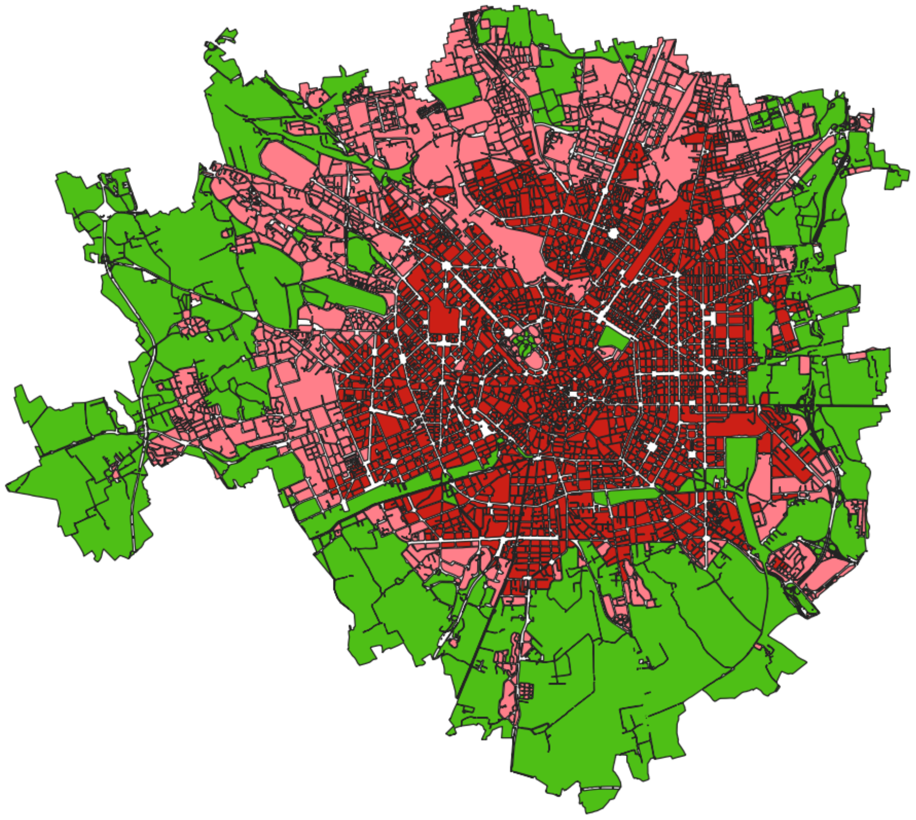


Figure 7.6: K-Means clustering on Block climate data with 3 clusters

7.2 Urban results

For these experiments, we will report the internal metrics and the score compared with the correspondent climate experiments for each dataset.

7.2.1 ACE urban

In table 7.6, we can see how the internal metrics are low for both hierarchical and K-Means clustering. For what concern the comparison with the climate experiments, in table 7.7 we see that the clustering is not similar, except for K-Means with 7 clusters.

N clusters	Algorithm	Silhouette	Davies-Bouldin
3	Hierarchical	0.2237	1.4566
4	Hierarchical	0.2988	1.2396
6	Hierarchical	0.2009	1.3576
7	K-Means	0.2176	1.1589

Table 7.6: Internal metrics for experiments on ACE urban data

N clusters	Algorithm	Score
3	Hierarchical	0.4577
4	Hierarchical	0.4705
6	Hierarchical	0.5707
7	K-Means	0.7580

Table 7.7: Score comparison between ACE climate and urban experiments

7.2.2 NIL urban

From table 7.8 we can see how K-Means provided clusters considerably more separated. As in the previous case, the urban data created different partitions compared to climate data, as displayed in the score values of table 7.9. Analysing the maps of these experiments, we noted some behaviours that are not desirable, such as divisions between the similar agricultural parks and the grouping of urban parks with heavily built areas.

N clusters	Algorithm	Silhouette	Davies-Bouldin
6	Hierarchical	0.2361	1.0520
7	K-Means	0.3880	1.0003

Table 7.8: Internal metrics for experiments on NIL urban data

N clusters	Algorithm	Score
6	Hierarchical	0.5749
7	K-Means	0.4822

Table 7.9: Score comparison between NIL climate and urban experiments

7.2.3 Block urban

Despite having better performance in terms of metrics, which are shown in table 7.10, the clusters provided by these experiments did not provide any good information from an urbanistic point of view. One big cluster contained most of the data points, even ones that have completely different conformation, while almost identical blocks were separated in different small clusters. The score comparison with the climate experiments is not reported, since it presented similar values to the other two datasets.

N clusters	Algorithm	Silhouette	Davies-Bouldin
4	K-Means	0.4185	1.2144
7	K-Means	0.3649	1.1854

Table 7.10: Internal metrics for experiments on NIL urban data

7.2.4 Comments on urban results

Overall, these urban features seemed less suited to form useful clusters in the three datasets compared with the climatic data, at least with an automatic feature selection. The reason for this could be that the datasets contained a large number of features of different types, from the urbanistic perspective. This problem can be mitigated by splitting manually the features in different groups and repeating the experiments on each of them. In the current version, the plugin does not provide functionalities to support the users in the manual selection. In the next chapter, we talk about the implementations that could be helpful for these situations.

7.3 Buildings results

As we said in the previous chapter, the buildings' data have clearer classifications compared to the other datasets. For this reason, the evaluation of the results is also done manually by exploring the map and understanding the formed clusters. A good way to objectively evaluate these experiments would be to define a gold standard for the entire dataset or a portion of it and compare it with the obtained labels.

7.3.1 Automatic features

In the automatic set, the experiment at 4 clusters did not provide a good classification. This is caused by some outliers, which are really big buildings, that are grouped together in a single cluster. Due to this, the other three clusters are not enough to identify all the different types of buildings. For the 6, 10 and 12 experiments, instead, we were satisfied with the results. In table 7.11 we can see the internal metrics of all 4 groupings.

N clusters	**Algorithm**	**Silhouette**	**Davies-Bouldin**
4	K-Means	0.5705	0.9722
6	K-Means	0.4575	1.0477
10	K-Means	0.3813	1.0291
12	K-Means	0.3788	0.9341

Table 7.11: Internal metrics for experiments on buildings' data and automatically selected features

7.3.2 Manual features

The manual features provided a good classification with 4 clusters as well, since it didn't use an entire cluster for the outliers but grouped them in a class of large buildings. The other three experiments performed similarly to the automatic set, and most of the buildings grouped differently don't belong to a clear class but can fit in different ones. The internal metrics, shown in table 7.12, have overall worse performance, but this is expected since the features are not selected using objective parameters. In image 7.7 there is a small part of the dataset grouped in 6 clusters.

N clusters	**Algorithm**	**Silhouette**	**Davies-Bouldin**
4	K-Means	0.3221	1.2236
6	K-Means	0.3317	1.0516
10	K-Means	0.2577	1.1003
12	K-Means	0.2653	1.1049

Table 7.12: Internal metrics for experiments on buildings' data and manually selected features

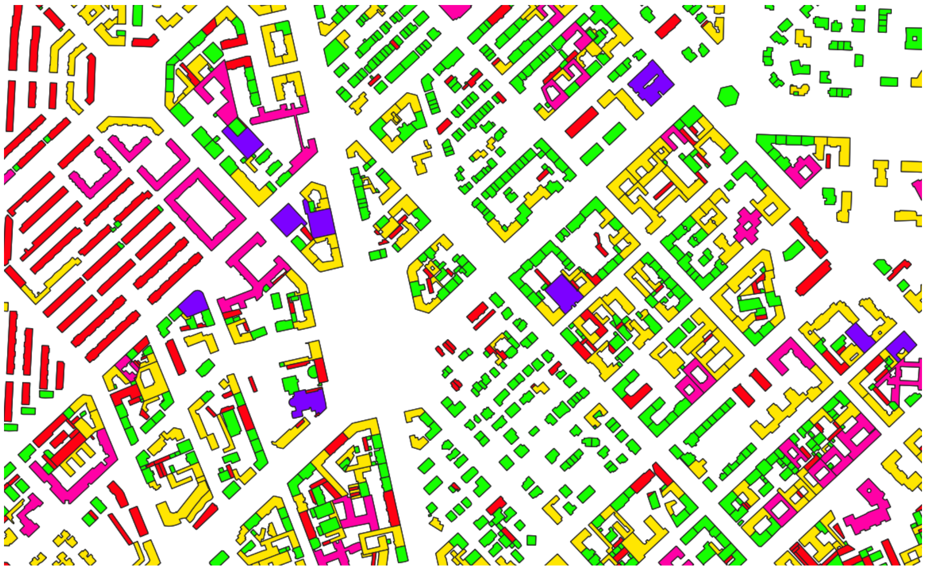


Figure 7.7: K-Means clustering on Building data and manually selected features with 6 clusters

7.3.3 Comparison between manual and automatic

For a dataset with so many data points, the computation of the correlation matrix exceeded the memory limits, requesting more than 35 GB of memory. To compare the two set of features we had to split the labels into 9 parts and compute the score on each of them. This operation resulted too time consuming, and the plugin was inadequate for the task. In table 7.13 we see the score for each part of the dataset and the average of all 9. As expected, the lowest score is for 4 clusters, while the other three numbers provided similar values. In some zones of the city, in particular the city centre (section 9), the score exceeded (or is very close to) 70 percent.

N	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**Average**
4	0.55	0.54	0.52	0.56	0.57	0.56	0.56	0.56	0.57	0.55
6	0.64	0.63	0.61	0.65	0.66	0.64	0.65	0.65	0.68	0.64
12	0.66	0.66	0.62	0.67	0.67	0.65	0.67	0.68	0.71	0.66
4	0.67	0.68	0.63	0.68	0.68	0.66	0.68	0.70	0.73	0.68

Table 7.13: Score comparison on building experiments between manual and automatic feature selection

7.4 Demographics and social results

For these experiments, we don't report internal metrics, but we focus on the analysis of the maps, in particular the ones with a small number of clusters as they are easier to read.

7.4.1 Young people education and occupation

In image 7.8 we can see the maps with 2 and 4 clusters. There is a clear separation between the areas in the centre and in the outskirts. The areas identified are among the frailest neighborhoods of the city and present social criticalities, which is coherent with the attributes.

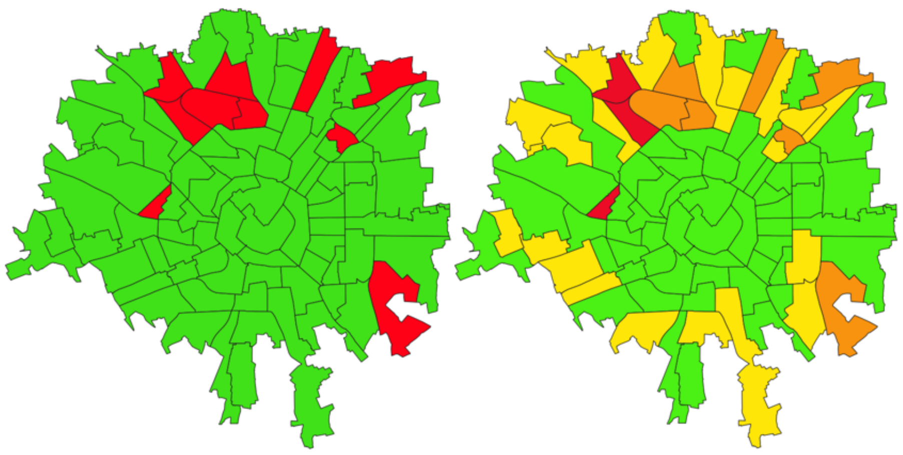


Figure 7.8: Hierarchical clustering on young people education on occupation data with 2 and 4 clusters

7.4.2 Housing overcrowding results

In image 7.9, the first division in 2 clusters creates a ring between the centre of the city and the outermost areas, that follows approximately the outer Ring Road of the city. It is interesting to note how the centre is grouped with the outskirts rather than the adjacent zones. At 4 clusters, the ring persists, and 2 smaller clusters split from the big one. The yellow one is mostly composed of the city centre and other close areas, while the other is in the suburbs and contains the neighbourhoods with very large families.

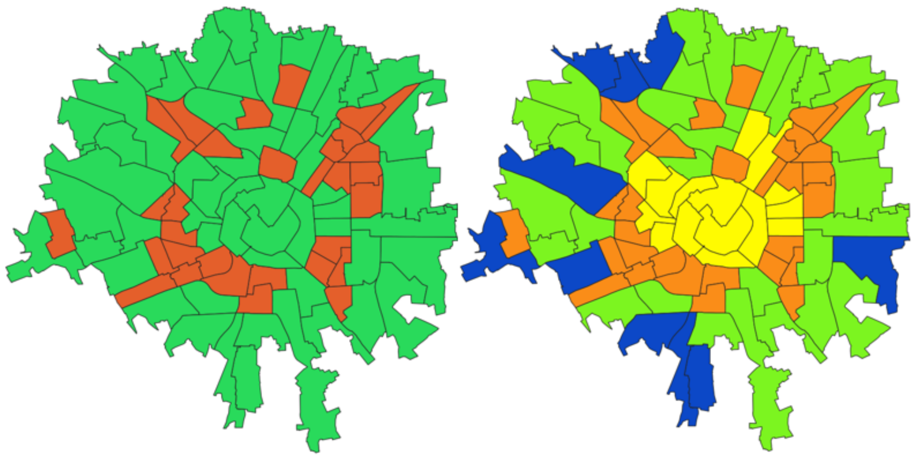


Figure 7.9: Hierarchical clustering on overcrowding with 2 and 4 clusters

7.4.3 Population isolation results

At 2 clusters, we find a group with almost all the outermost areas that is strongly separated from the other cluster, as highlighted by the dendrogram in section 6.4.3. The areas in this group have the lowest percentages of alone people. In image 7.10, instead, we identify in yellow the neighbourhoods with a higher degree of isolation.

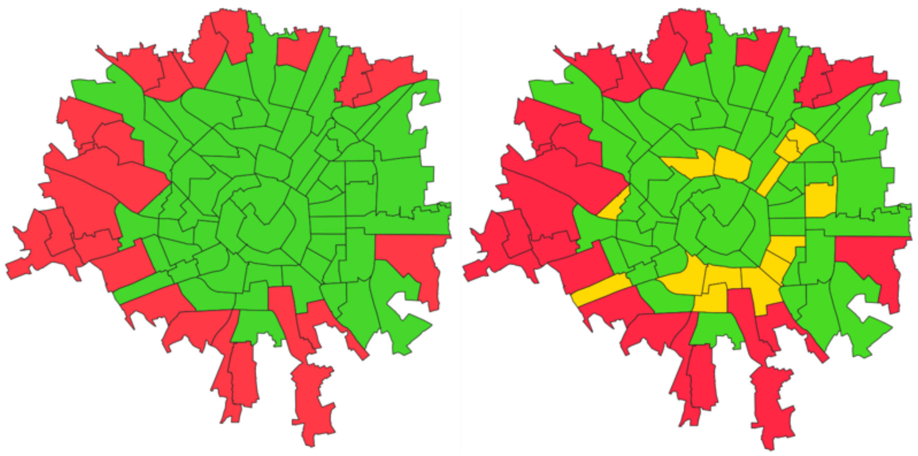


Figure 7.10: Hierarchical clustering on isolation with 2 and 3 clusters

Chapter 8

Conclusions and future work

The objective of this research was to expand the possibilities of using machine learning – in particular clustering – on geospatial data within a GIS. To achieve this, we implemented from scratch a new plugin that introduces functionalities that were not available in the existing QGIS plugins, nor in other widespread proprietary software such as ArcGIS. The new functionalities can be separated into three different categories: pre-processing for dimensionality reduction, feature selection, and clustering evaluation. The first two categories are essential to improve the performance of cluster analysis in high dimensional data, which is often the case when dealing with geospatial data. The evaluation functionalities are helpful for the user to obtain insight and information to understand the quality and utility of the performed analysis. The large number of experiments carried out in this work displayed the versatility of the plugin, which can be used in multiple situations with data of different types and dimensions.

The analyses were not only useful to highlight the strengths of the developed tool, but allowed us to identify its limitations. There is still work to do to achieve a complete and flexible plugin for cluster analysis in QGIS. The future developments can focus mainly on two categories:

- optimization of the software performances to enable the use of each functionality on even bigger datasets;
- expansion and improvement of the analysis functionalities.

The main weakness of performance, at the moment, is the score computation, as we saw in section 7.3. To allow comparison on bigger datasets, a viable solution would be to split the comparison matrix and compute the score on smaller sections, avoiding memory limitations. Other improvements would be to perform the heavy computations in background and to implement multithreading for independent but long computations, such as the matrixes used in entropy feature selection.

The easiest, but very helpful, expansion of functionalities is the addition of the other popular clustering algorithms and more options for the algorithms' settings.

In the same section of the plugin, new algorithms for automatic feature selection or feature reduction can be implemented; a good solution would be Principal Component Analysis. For what concerns the existing feature selection, random sampling is not always the best solution, since geospatial data is often spatially autocorrelated [50]. For this reason, new methods of sampling for the entropy algorithm are a good addition. As felt evident while reporting the experiments' results, a missing functionality is the possibility to see the distances between the cluster, which is essential to understand the clusters displayed on the maps. Finally, future works can implement an entire new section dedicated to data exploration, where it would be possible to display important information about the data to the users. Some features to include in this section are: identification and elimination of outliers in the data that can reduce the clustering quality, computation of statistics about single or multiple features, visualization of the high dimensional space in low dimension with the help of Self Organizing Maps [51].

To allow the use and the implementation of new functionalities to the QGIS community, the plugin will be published in the official repository in the near future.

Bibliography

[1] M. Kanevski et al. "Machine learning models for geospatial data". In: Jan. 2009, pp. 175–227. ISBN: 978 -2 -9403-6808-2.

[2] E. Lenzi. "SIMBA: systematic clustering-based methodology to support built environment analysis". MA thesis. Politecnico di Milano, 2020.

[3] Politecnico di Milano. *IMM Design Lab*. `http://www.immdesignlab.com/`.

[4] QGIS Development Team. *QGIS Geographic Information System*. Open Source Geospatial Foundation. 2021. URL: `http://qgis.org`.

[5] Earth Observing System. *Spatial Analysis: Data Processing And Use Cases*. 2021. URL: `https://eos.com/blog/spatial-analysis/`.

[6] N. Tohidi and R. B. Rustamov. "A Review of the Machine Learning in GIS for Megacities Application". In: *Geographic Information Systems in Geospatial Intelligence*. Ed. by IntechOpen. Oct. 2020. DOI: `10.5772/intechopen.94033`. URL: `https://www.intechopen.com/chapters/73592`.

[7] G. Rousset et al. "Assessment of Deep Learning Techniques for Land Use Land Cover Classification in Southern New Caledonia". In: *Remote Sensing* 13.12 (2021). ISSN: 2072-4292. DOI: `10.3390/rs13122257`. URL: `https://www.mdpi.com/2072-4292/13/12/2257`.

[8] H. A. Kaul and S. Ingle. "Land Use Land Cover Classification and Change Detection Using High Resolution Temporal Satellite Data". In: *The Journal of Environment* 1 (Nov. 2012), pp. 146–152.

[9] E. Aksoy. "Clustering With GIS: An Attempt to Classify Turkish District Data". In: (Jan. 2006).

[10] R. Singh. *Where Deep Learning Meets GIS*. `https://www.esri.com/about/newsroom/arcwatch/where-deep-learning-meets-gis/`.

[11] Politecnico di Milano. *IMM Design Lab*. `http://www.immdesignlab.com/informazioni/`.

[12] C. A. Biraghi. "Multi-Scale Modelling Approach for Urban Optimization: Urban Compactness Environmental Implications". PhD thesis. Politecnico di Milano, 2019.

[13] M. Tadi and S. V. Manesh. "Integrated modification methodology (imm): A phasing process for sustainable urban design". In: (2013).

[14] Environmental Systems Research Institute (ESRI). *ArcGIS Pro, Multivariate Clustering*. Version 2.9. 2021. URL: `https://pro.arcgis.com/en/pro-app/latest/tool-reference/spatial-statistics/multivariate-clustering.htm#L_`.

[15] E. Kazakov. *Attribute based clustering*. Version 2.2. 2021. URL: `https://plugins.qgis.org/plugins/attributeBasedClustering/`.

[16] University of Wisconin-Madison. *Mapping and Geographic Information Systems (GIS) : What is GIS?* URL: `https://researchguides.library.wisc.edu/GIS`.

[17] C. Dempsey. *Types of GIS Data Explored: Vector and Raster*. 2021. URL: `https://www.gislounge.com/geodatabases-explored-vector-and-raster-data/`.

[18] Environmental Systems Research Institute (ESRI). *Defining a spatial reference*. 2009. URL: `http://webhelp.esri.com/arcgiSDEsktop/9.3/index.cfm?TopicName=Defining_a_spatial_reference`.

[19] L. Wasser et al. *datacarpentry/organization-geospatial: Data Carpentry Introduction to Geospatial Concepts August 2018 Release*. Aug. 2018. DOI: `10.5281/zenodo.1404414`.

[20] QGIS Development Team. *Geographic Information System API Documentation*. QGIS Association. 2021. URL: `https://qgis.org/pyqgis/3.16/index.html`.

[21] Expert.ai Team. *What is Machine Learning? A Definition.* 2020. URL: `https://www.expert.ai/blog/machine-learning-definition/`.

[22] D. Gunopulos. "Cluster and Distance Measure". In: *Encyclopedia of Database Systems*. Ed. by L. LIU and M. T. ÖZSU. Boston, MA: Springer US, 2009, pp. 374–375. ISBN: 978-0-387-39940-9. DOI: `10.1007/978-0-387-39940-9_618`. URL: `https://doi.org/10.1007/978-0-387-39940-9_618`.

[23] S. Thilagamani, A. Jayanthiladevi, and N. Arunkumar. "Data mining algorithms, fog computing". In: May 2018, pp. 231–264. DOI: `10.4018/978-1-5225-5972-6.ch012`.

[24] S. Yang, D. Towey, and Z. Q. Zhou. "Metamorphic Exploration of an Unsupervised Clustering Program". In: May 2019, pp. 48–54. DOI: `10.1109/MET.2019.00015`.

[25] P.-N. Tan, M. Steinbach, and V. Kumar. *Introduction to Data Mining, (First Edition)*. USA: Addison-Wesley Longman Publishing Co., Inc., 2005. Chap. Chapter 8: "Cluster Analysis: Basic Concepts and Algorithms". ISBN: 0321321367.

[26] M. Pathak. *Quick Guide to Evaluation Metrics for Supervised and Unsupervised Machine Learning.* 2020. URL: https://www.analyticsvidhya.com/blog/2020/10/quick-guide-to-evaluation-metrics-for-supervised-and-unsupervised-machine-learning/.

[27] R. Bellman. "Dynamic programming". In: *Science* 153.3731 (1966), pp. 34–37.

[28] N. Venkat. "The Curse of Dimensionality: Inside Out". In: (Sept. 2018). DOI: 10.13140/RG.2.2.29631.36006.

[29] A. Gupta. *Feature Selection Techniques in Machine Learning.* 2020. URL: https://www.analyticsvidhya.com/blog/2020/10/feature-selection-techniques-in-machine-learning/.

[30] J. G. Dy and C. E. Brodley. "Feature Selection for Unsupervised Learning". In: *Journal of Machine Learning Research* (2000).

[31] M. Dash and H. Liu. "Feature Selection for Clustering". In: (2000).

[32] M. Steinbach, L. Ertöz, and V. Kumar. "The Challenges of Clustering High Dimensional Data". In: *New Directions in Statistical Physics: Econophysics, Bioinformatics, and Pattern Recognition.* Ed. by L. T. Wille. Springer Berlin Heidelberg, 2004, pp. 273–309. ISBN: 978-3-662-08968-2. DOI: 10.1007/978-3-662-08968-2_16. URL: https://doi.org/10.1007/978-3-662-08968-2_16.

[33] K. Beyer et al. "When Is "Nearest Neighbor" Meaningful?" In: *Database Theory — ICDT'99.* Ed. by C. Beeri and P. Buneman. Springer Berlin Heidelberg, 1999, pp. 217–235.

[34] C. C. Aggarwal, A. Hinneburg, and D. A. Keim. "On the Surprising Behavior of Distance Metrics in High Dimensional Space". In: *Database Theory — ICDT 2001.* Ed. by J. Van den Bussche and V. Vianu. Springer Berlin Heidelberg, 2001, pp. 420–434. ISBN: 978-3-540-44503-6.

[35] Copernicus Climate Change Service (C3S). 2019. URL: https://cds.climate.copernicus.eu/cdsapp#!/dataset/sis-urban-climate-cities?tab=overview.

[36] S. Gelmi and M. Magnani. *Database Topografico - Specifiche di contenuto semplificate.* 2021. URL: https://www.geoportale.regione.lombardia.it/documents/10180/0/Allegato+2_III_Specifiche/19458997-44c0-4d54-b07d-d26ae5d4c6d8.

[37] Comune di Milano. 2011. URL: https://dati.comune.milano.it/dataset/ds95_infogeo_aree_censimento_localizzazione_2011c.

[38] GeoApt LLC. *Plugin Builder.* Version 2.18.0. 2018. URL: https://plugins.qgis.org/plugins/pluginbuilder/.

[39] Qt Project. *Qt Designer.* Version 5.11.2. 2018. URL: https://doc.qt.io/qt-5/qtdesigner-manual.html.

[40] The pandas development team. *pandas-dev/pandas: Pandas.* Version 1.1.15. 2021. DOI: 10.5281/zenodo.3509134. URL: https://doi.org/10.5281/zenodo.3509134.

[41] W. McKinney. "Data Structures for Statistical Computing in Python". In: *Proceedings of the 9th Python in Science Conference.* Ed. by S. van der Walt and J. Millman. 2010, pp. 56–61. DOI: 10.25080/Majora-92bf1922-00a.

[42] C. R. Harris et al. "Array programming with NumPy". In: *Nature* 585.7825 (Sept. 2020), pp. 357–362. DOI: 10.1038/s41586-020-2649-2. URL: https://doi.org/10.1038/s41586-020-2649-2.

[43] L. Buitinck et al. "API design for machine learning software: experiences from the scikit-learn project". In: *ECML PKDD Workshop: Languages for Data Mining and Machine Learning.* 2013, pp. 108–122.

[44] M. Kuhn. "Building Predictive Models in R Using the caret Package". In: *Journal of Statistical Software, Articles* 28.5 (2008), pp. 1–26. ISSN: 1548-7660. DOI: 10.18637/jss.v028.i05. URL: https://www.jstatsoft.org/v028/i05.

[45] U. Fayyad, C. Reina, and P. S. Bradley. "Initialization of Iterative Refinement Clustering Algorithms". In: *9th International Conference on Knowledge Discovery & Data Mining (KDD '98).* 1998, pp. 194–198.

[46] J. D. Hunter. "Matplotlib: A 2D graphics environment". In: *Computing in Science & Engineering* 9.3 (2007), pp. 90–95. DOI: 10.1109/MCSE.2007.55.

[47] P. Bhatia. *Data Mining and Data Warehousing: Principles and Practical Techniques.* Cambridge University Press, 2019. DOI: 10.1017/9781108635592.

[48] M. Demuzere et al. "Mapping Europe into local climate zones". In: *PLOS ONE* 14.4 (Apr. 2019), pp. 1–27. DOI: 10.1371/journal.pone.0214474. URL: https://doi.org/10.1371/journal.pone.0214474.

[49] I. Stewart and T. Oke. "Local Climate Zones for Urban Temperature Studies". In: *Bulletin of the American Meteorological Society* 93 (Dec. 2012), pp. 1879–1900. DOI: 10.1175/BAMS-D-11-00019.1.

[50] A. D. Cliff and K. Ord. "Spatial Autocorrelation: A Review of Existing and New Measures with Applications". In: *Economic Geography* 46 (1970), pp. 269–292. ISSN: 00130095, 19448287. URL: http://www.jstor.org/stable/143144.

[51] T. Kohonen and M. Schroeder. *Self-Organizing Maps.* Jan. 2001. ISBN: 3540679219.

Appendices

Appendix A

Cluster Analysis User Guide

Cluster Analysis is a QGIS 3 plugin to perform clustering based on numerical values on vector layers with any geometry type. The algorithms currently available are Agglomerative Hierarchical and K-Means from the library scikit-learn. Besides, the plugin provides functionalities for dimensionality reduction (highly correlated features, constant and quasi constant features), feature selection, data scaling, and clustering evaluation. For additional information, refer to the brief guide in each section of the UI, and to the file "Plugin implementation" (in particular Chapter 5) in the plugin repository (`https://github.com/folini96/Cluster-Analysis-plugin/tree/main`) for a detailed description of each functionality.

Installation

The plugin is not officially published in the QGIS folder yet, for this reason the installation must be performed manually.

Install dependencies

Install the librarires not available in QGIS or OSGeo4w Python (scipy, pandas, scikit-learn). To install the libraries on Windows follow this steps in the same order:

Open OSGeo4w Shell installed with QGIS as Administrator and type:

```
$ py3_env
$ python -m pip install scipy
$ python -m pip install pandas
$ python -m pip install scikit-learn
```

In Linux or Mac OS X, QGIS uses the standard Python installation (independent of QGIS), so you can install the missing libraries using the Terminal (`https://packaging.python.org/tutorials/installing-packages/`).

Add the plugin to QGIS

To manually install the plugin in QGIS you can perform the following steps:

- Download the entire plugin repository from:
  `https://github.com/folini96/Cluster-Analysis-plugin/tree/main`
- Open QGIS 3 and go to Settings → User Profiles → Open Active User Folder
- In the User Folder go to python → plugins
- Add the plugin to this folder
- Reopen QGIS 3, and go to Plugins → Manage and Install Plugins, and make sure that Cluster Analysis is checked in the Installed tab

A new icon for Cluster Analysis will be added to the Plugins menu and on the QGIS main panel.

Save and load experiments

In the Results section of the plugin, it is possible to save the experiments of the current session in a text file, and to load older ones. The saved files are stored in the Experiments folder inside the plugin folder. To access the plugin folder from QGIS go to Settings → User Profiles → Open Active User Folder → python → plugins. The load function opens by default the Experiments folder, but it is possible to open files stored in other locations. It is not recommended to modify a file and load it in the plugin.

Configuration file

In the plugin folder you can find a json file called Configuration. From this file it is possible to modify some settings of the algorithms in the plugin. To effectively change the settings, after updating the file you have to reload the plugin either with Plugin Reloader or by restarting QGIS. The parameters are:

- *frequency_cut*: the threshold for the ratio of the most common value to the second most common value, used in the quasi-constant feature elimination. Requires a numerical value greater than 0;
- *unique_cut*: the threshold for the ratio of distinct values to the number of total samples, used in the quasi-constant feature elimination. Requires a percentage value expressed as a number between 0 and 1;

- *entropy_iterations*: the number of random samples used for the sampling entropy algorithm. Requires a positive integer, it is suggested to use a number greater than 35. The execution of sampling entropy takes longer as this number grows;

- *sample_size*: the number of points in every random sample for the sampling entropy algorithm; Requires a positive integer, the quality of the selection gets lower the smaller the sample size is, while the execution takes longer as this number grows;

- *graph_max_cluster*: the max number of clusters used when plotting WSS and BSS trends. Requires a positive integer;

- *distance*: distance measure used in hierarchical clustering, the accepted values are 'euclidean' or 'manhattan'.