CLUSTER ANALYSIS: A COMPREHENSIVE AND VERSATILE QGIS PLUGIN FOR PATTERN RECOGNITION IN GEOSPATIAL DATA

A. Folini [1,*], E. Lenzi [2], C. A. Biraghi [3]

[1] Department of Civil and Environmental Engineering, Politecnico di Milano, Italy - andrea.folini@polimi.it
[2] Department of Electronics, Information and Bioengineering, Politecnico di Milano, Italy - emilia.lenzi@polimi.it
[3] Department of Architecture, Built environment and Construction engineering, Politecnico di Milano, Italy
- carloandrea.biraghi@polimi.it

Commission IV, WG IV/4

KEY WORDS: Clustering, Feature selection, Clusters evaluation, Pattern recognition, QGIS Plugin, Python, FOSS4G.

ABSTRACT:

As geospatial data continuously grows in complexity and size, the application of Machine Learning and Data Mining techniques to geospatial analysis is increasingly 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 (GIS) are necessary to support researchers and organizations in data exploration, pattern recognition, and prediction in the various applications of geospatial data. In this work, we present Cluster Analysis, a Python plugin that we developed for the open-source software QGIS and offers functionalities for the entire clustering process. Or tool provides different improvements from the current solutions available in QGIS, but also in other widespread GIS software. 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. To highlight both the potential of the plugin and its limitations in real-world scenarios, the development is integrated with a considerable experimental phase with data of different natures and granularities. 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.

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 that is of great interest to 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 (Kanevski et al., 2009).

An example of this application is SIMBA (Lenzi, 2020), a clustering-based methodology developed to support and systematize the analysis of the built environment in the Integrated Modification Methodology (IMM) (Tadi et al., 2020) . Although SIMBA provides a useful methodology to select a representative and a 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 (QGIS Development Team, 2022), 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 open-source GIS. In particular, the goals of the work are:

1. develop a plugin for QGIS that is capable to support the whole process of cluster analysis, from the pre-processing of data and feature selection to the evaluation of the results;
2. provide flexibility to allow the use on data of different nature and size;
3. guarantee accessibility and ease of use, as GIS users often lack advanced machine learning and computer science knowledge.

The final version of the tool we developed is composed of three main parts: (i) feature cleaning for dimensionality reduction, (ii) feature selection and clustering, and (iii) 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 was essential to highlight both the potential of the plugin and its limitations

* Corresponding author

in real-world scenarios. A great volume of experiments was conducted on data about the city of Milan, describing social-demographics, urban, and climatic characteristics with widely different granularities.

In order to cover all the topics related to the plugin development, the rest of the paper is structured as follows. At first, we present an explanation of the machine learning concepts useful to understand the plugin functionalities, followed by a brief description of the solutions currently available for clustering in GIS. The following sections provide the details about the implementation of all the functionalities of the plugin, and an overview of the experiments performed to assess its quality. The paper closes by summarizing the results obtained by the final version of the plugin and outlining the possibilities for future developments.

2. STATE OF THE ART AND MACHINE LEARNING BACKGROUND

2.1 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 term clustering does not correspond to a specific procedure, but to a general problem that can be solved by using various algorithms. In this work, we use two of the most widespread clustering algorithms: Agglomerative Hierarchical (Nielsen, 2016) and K-Means (MacQueen, 1967).

Agglomerative hierarchical clustering starts by separating every data point in its cluster. Then the two closest clusters get merged 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.

K-Means is an iterative clustering algorithm that aims to find a local maximum 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 center (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.

Obviously, 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. K-Means performs considerably better in terms of space and time complexity; therefore it is the best solution to handle big datasets. However, the clear 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.

2.2 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 (Bellman, 1966) as the "curse of dimensionality". This concept refers to the explosive nature of spatial dimensions and their 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 (Venkat, 2018).

Feature selection for unsupervised learning is usually more challenging than when dealing with supervised learning, since 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 in (Dy and Brodley, 2004) or the ranking algorithm from (Dash and Liu, 2000) that we implemented in our work. 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.

For clustering purposes, the most relevant aspect of the curse of dimensionality concerns the effect of increasing dimensionality on distance and similarity. As we saw before, most clustering techniques depend critically on these two measures and require that the objects within clusters are, in general, closer to each other than to objects in other clusters (Steinbach et al., 2004). Unfortunately, when dealing with spaces in a lot of dimensions, the data points and their distance measure does not behave as intuitively expected. In (Beyer et al., 1999) is shown that, under certain reasonable assumptions on the data distribution, the ratio of the distances of the nearest and farthest neighbors to a given target in high dimensional space is almost 1 for a wide variety of data distributions and distance functions. This means that all the data points are almost equidistant from each other, which is 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 a high dimensional space is Manhattan, followed by Euclidean, as proven in the work (Aggarwal et al., 2001).

2.3 Evaluation for clustering

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

called an internal evaluation, while the second one is an external evaluation (Liu et al., 2010).

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: Silhouette coefficient and Davies-Bouldin index (Liu et al., 2010). This type of evaluation is easy to compute and to interpret, but it is not always meaningful, like in the case of datasets with few data points.

In external evaluation, the clustering results are compared to a benchmark or gold standard, which is a labeling of the data points produced by an expert in the field. The external metrics evaluate how well the clustering matches the gold standard classes (Liu et al., 2010). While these metrics provide a good assessment of the quality of clusters, they are rarely applicable since external information about data is difficult to obtain.

2.4 Optimal number of clusters

Another challenging task in clustering, related to the evaluation, is the decision on 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. In our work, we focus on two graphical methods that can provide useful insights to the users: the dendrogram of hierarchical clustering and the knee-elbow method, based on the trends of Within clusters Sum of Squares (WSS) and Between clusters Sum of Squares (BSS).

Using the dendrogram, the best choice for the number of clusters is the number of vertical lines cut by a horizontal line that can transverse the maximum distance vertically without intersecting a cluster (Kaushik, 2016). In figure 1 the maximum distance is represented by segment AB, and the choice would be 4 clusters.

The second technique, instead, consists in looking for a knee or an elbow in the WSS and BSS trends, showing a significant modification in the metrics (Thorndike, 1953). In the example in figure 2, a good choice would be 5 clusters, as shown by the sudden flattening of the two trends.

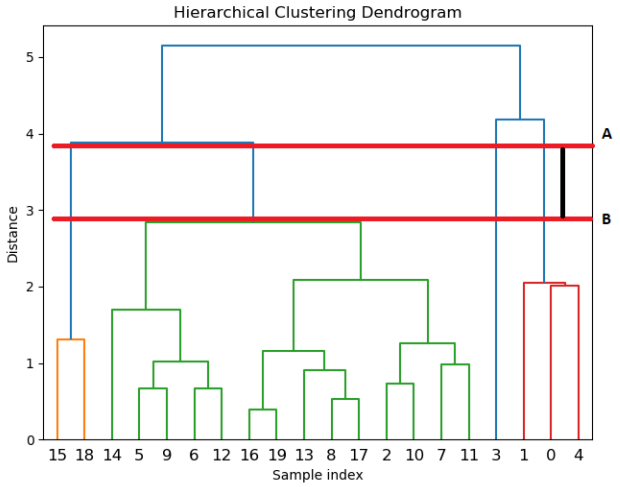

Figure 1. Dendrogram example

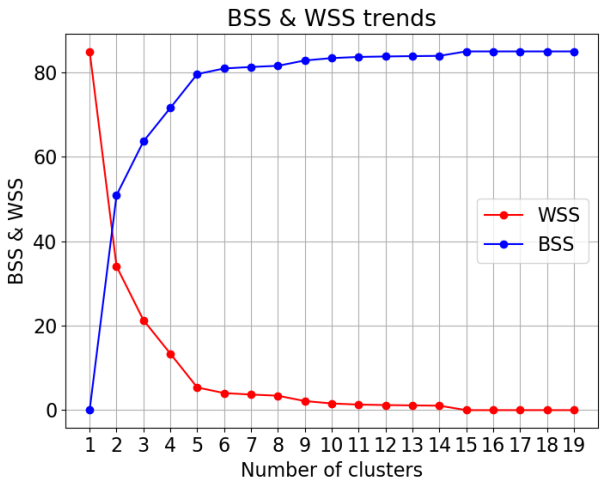

Figure 2. WSS and BSS trends example

2.5 Clustering tools in GIS

Both QGIS and ESRI ArcGIS (Environmental Systems Research Institute, 2021), the most popular open-source and proprietary GIS, provide various alternatives for attribute-based cluster analysis. Despite the good number of available tools, all of them lack functionalities that are essential to face the challenges of clustering geospatial data, such as their high dimensionality, poor quality, and large dataset size.

In ArcGIS Pro, the main choice is the processing tool Multivariate Clustering, which provides clustering on one or more selected attributes with the algorithms K-Means and K-Medoids. It also includes a support functionality to give information on the best number of clusters and multiple graphs that can be used to better understand the formed clusters and the performances of each attribute used.

In QGIS, clustering is provided by plugins developed by the community, and the most complete solution for non-spatial clustering is Attribute Based Clustering (Kazakov and QGIS Development Team, 2021). Attribute-Based Clustering offers a good amount of settings for clustering with both algorithms Agglomerative Hierarchical and K-Means, but has few other functions for important steps of cluster analysis.

3. CLUSTER ANALYSIS PLUGIN

Cluster Analysis allows to perform attribute-based clustering on numerical fields of vector files with any geometry type and is compatible with the most common vector formats, such as Shapefile, CSV, GeoPackage, and GeoJSON.

The plugin is developed in Python (version 3.7), and all the necessary files and base code are created using Plugin Builder (GeoApt LLC and QGIS Development Team, 2018). The GUI is designed with QT Designer (Qt Project, 2018) and additional QGIS custom widgets, in order to have automatic updates of the interface after the user adds or removes layers to the project.

The latest version of the plugin is available on the official QGIS Plugin Repository (https://plugins.qgis.org/plugins/Cluster-Analysis-plugin-main/), while the entire source code, along with a User Guide, is available on Github (https://github.com/folini96/Cluster-Analysis-plugin).

3.1 Implementation overview

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

- feature cleaning
- feature selection and clustering
- evaluation

All of the functionalities in the different sections are implemented as stand-alone to guarantee that the users can perform only the actions they need.

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 (The pandas development team, 2021) and NumPy (Harris et al., 2020) to guarantee high performance.

Another important focus of the development process is the extensibility of the project. We designed the application to be easily expanded in the future with new functionalities or algorithms, both by the authors and the QGIS developer community

3.2 Feature cleaning

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. To avoid the modification of the original dataset, once the process is complete, the plugin creates a new vector layer containing only the selected fields.

Since highly correlated features usually provide redundant information and can lead to an overweight of some characteristics, the first filter removes the features that are correlated above a user-defined threshold. The user can also select the criterion used to keep a feature among a multicollinear group from the following options:

- 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 the max interval of values and the domain of the feature, where a higher value indicates better coverage of the domain is preferred. To use this criterion all the features must have the same domain specified by the user.

The other two filters identify the attributes with constant values for all the data points or with only few outliers differentiating from them. These types of features don't provide any valuable information and can worsen the performance of clustering. 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 (Kuhn, 2008):

- 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%).

3.3 Feature Selection and Clustering

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.

The entropy feature selection, presented in (Dash and Liu, 2000), provides a complete ranking of the features based on their impact in separating the data into well-formed clusters. From the ranking, we extract only the features that guarantee a positive effect on the data separation. The algorithm can be performed on the entire dataset, with a time complexity quadratic on the number of data points, or exploiting random sampling. The sampling version removes the dependency on the number of data points, providing great scalability for larger datasets. The use of smaller samples instead of the entire dataset could reduce the quality of the selection. Moreover, this version of the algorithm could yield slightly different results on each execution due to its random component. To mitigate these problems, it is possible to select a larger number of algorithm iterations and size of the samples, which are defaulted respectively to 35 and 100.

The two alternative algorithms for clustering are the ones we presented in Section 2:

- Agglomerative hierarchical
- K-Means

For both algorithms we use the functions implemented in scikit-learn (Buitinck et al., 2013). 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 not editable and set to "complete", which means we use the maximum distance between clusters.

With 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 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 the space and time complexity of both algorithms, the use of only hierarchical clustering would limit the analysis to small datasets.

Before performing clustering, the plugin offers the possibility to plot the dendrogram and WSS-BSS trends to facilitate the choice of the number of clusters, and to scale the datasets with standardization or normalization. The data scaling is used to bring every dimension to the same equal weight and is particularly important when the features in the dataset represent measurements with different units or different scales.

3.4 Evaluation

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. To evaluate the quality of the experiments we calculate two indexes and we allow a comparison among experiments on the same dataset.

The indexes used are the internal metrics Silhouette coefficient and Davis-Bouldin index. In addition, to directly compare the clusters formed by two or more experiments, we compute the score, introduced in (Lenzi, 2020), which evaluates how many couples of data points are grouped together in all of the experiments or in none of them.

Every experiment completed in the current session can be stored in a text file, and the experiments saved in previous sessions can be loaded in the plugin and are shown in the evaluation section along with the other ones.

3.5 User Interface

The main objective of the interface is to ensure ease of use for every user, regardless of their experience level with machine learning and GIS. For this reason, most of the design choices have the goal to simplify the 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 in the inputs or the completion of the selected operations. To maintain the simplicity of the user interface, the most technical parameters of the algorithms are moved in an external configuration file in JSON format.

3.6 Configuration file

The JSON file called "Configuration" is in the plugin directory and can be easily modified by any text editor. The settings are set once when the plugin is loaded, so it should be restarted after an update. The modifiable 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;
- 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'.

4. EXPERIMENTAL PHASE

During the entire development process, we conducted a large number of experiments that covered all the features offered by the plugin. The main objective of the experimental phase was not to evaluate the performance of a particular methodology or algorithm, but rather to show the potential of the developed tool to analyze data of different nature and, most important, of different sizes, up to several tens of thousands of data points. Furthermore, the experiments were 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. The datasets used in the experiments refer to the city of Milan and can be grouped into four different categories:

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

For what concerns the size of the datasets, the granularities used range from less than 100 to almost 70000 data points, and up to 109 numerical attributes.

At the end of the development phase, we were able to apply almost all the functionalities for pre-processing, feature selection, clustering, and evaluation to each dataset with few exceptions. In particular, the most critical functions are:

- the score computation, which requires the storage of a matrix with dimension NxN where N is the number of data points and could exceed the memory limits;
- the entropy calculation for datasets with more than a few hundred of data points, which could require several minutes for the execution.

The latter has a direct solution with the use of the random sampling version of the algorithm; while the score would require some modifications to the plugin, such as splitting the matrix into smaller blocks and computing the scores separately.

4.1 Experiments examples

In this section, we briefly report three examples of experiments performed with the plugin, while a more detailed description of the entire experimental phase can be found in (Folini, 2021).

The first example (figure 3) is a partition of Milan, with a 100m spatial resolution, in 5 different climate zones using data about temperature, humidity, and wind speed from Copernicus CDS (Copernicus Climate Change Service (C3S), 2019). The algorithm used is K-Means, given the size of the dataset, and the features are selected with an automatic procedure.

The second experiment (figure 4) tries to identify the neighborhoods in Milan with potential social criticalities starting from

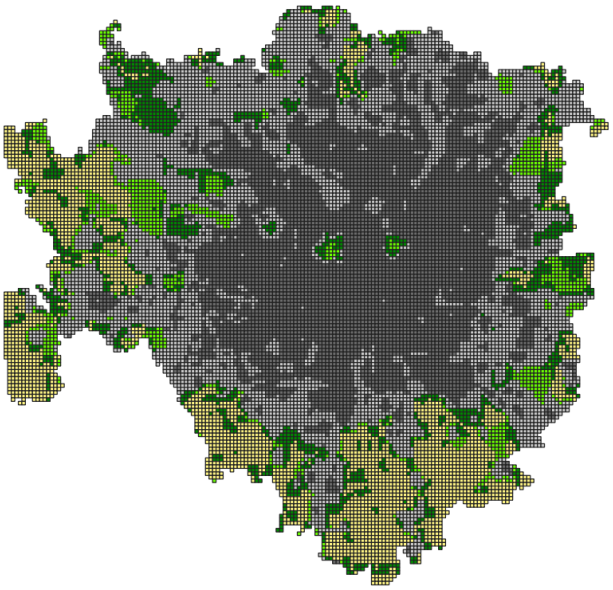

Figure 3. Map of monthly climate data from 2017 with 5 clusters

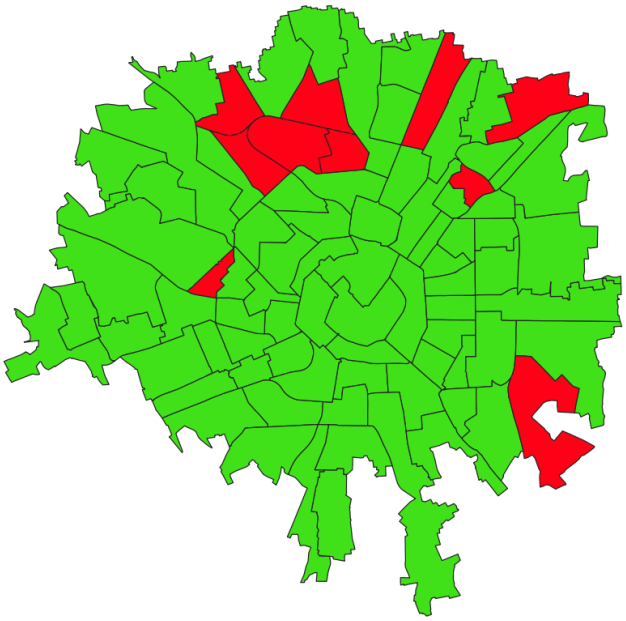

Figure 4. Hierarchical clustering on young people education and occupation data with 2 clusters

features about young people's education and occupation. In this case, the choice for the algorithm is hierarchical, which is more suitable to find outliers in a dataset.

The last one (figure 5) is an attempt to automatically categorize approximately 70 thousand buildings into meaningful typological classes starting from a set of formal properties. Both the features and the number of clusters are selected manually by an expert and the algorithm used is K-Means.

5. CONCLUSIONS AND FUTURE WORKS

The objective of this research was to expand the possibilities of using machine-learning techniques – in particular clustering – on geospatial data within a GIS. To achieve this, we implemented

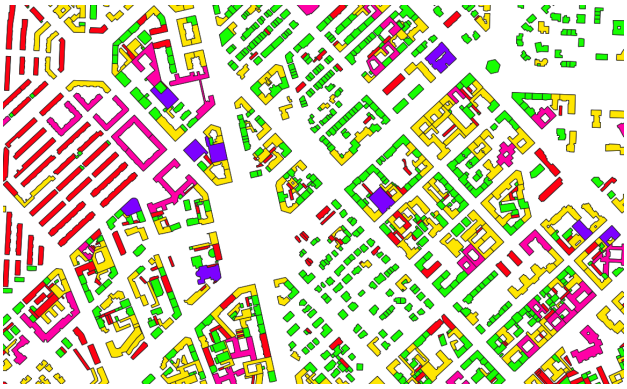

Figure 5. K-Means clustering on buildings data and manually selected features with 6 clusters

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.

As highlighted by the experimental phase, the plugin presents good versatility and can be of great use in various contexts. However, the experiments also identified some limitations that can be improved in future works, especially in 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 first performance improvement should be on the score computation, as it is currently impossible for datasets with tens of thousands of data points, due to memory constraints. For what concerns the addition of new functionalities, the main focus should be on the implementation of a new section dedicated to data visualization and exploration, which would provide important information about the data before the execution of the analysis. Another possibility that would not require much effort is the extension of existing functionalities such as new clustering or feature selection algorithms.

Currently, the plugin has been downloaded by more than 1500 users in a few months, which proves the interest in the application. In the future, we hope it will be able to support the work of researchers and professionals in different fields.

REFERENCES

Aggarwal, C. C., Hinneburg, A., Keim, D. A., 2001. On the surprising behavior of distance metrics in high dimensional space. J. Van den Bussche, V. Vianu (eds), *Database Theory — ICDT 2001*, Springer Berlin Heidelberg, 420–434.

Bellman, R., 1966. Dynamic programming. *Science*, 153(3731), 34–37. doi.org/10.1126/science.153.3731.34.

Beyer, K., Goldstein, J., Ramakrishnan, R., Shaft, U., 1999. When is "nearest neighbor" meaningful? C. Beeri, P. Buneman (eds), *Database Theory — ICDT'99*, Springer Berlin Heidelberg, 217–235.

Buitinck, L., Louppe, G., Blondel, M., Pedregosa, F., Mueller, A., Grisel, O., Niculae, V., Prettenhofer, P., Gramfort, A., Grobler, J., Layton, R., VanderPlas, J., Joly, A., Holt, B., Varoquaux,

G., 2013. API design for machine learning software: experiences from the scikit-learn project. *ECML PKDD Workshop: Languages for Data Mining and Machine Learning*, 108–122.

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

Dash, M., Liu, H., 2000. Feature selection for clustering. *Knowledge Discovery and Data Mining. Current Issues and New Applications*, Springer Berlin Heidelberg, 110–121.

Dy, J. G., Brodley, C. E., 2004. Feature selection for unsupervised learning. *Journal of machine learning research*, 5(Aug), 845–889.

Environmental Systems Research Institute, 2021. Arcgis desktop, version 10.8.1. https://www.esri.com/it-it/arcgis/products/arcgis-desktop/overview.

Folini, A., 2021. Integrating machine learning techniques into gis software: Development of a comprehensive and versatile qgis plugin for cluster analysis on geospatial data. Master's thesis, Politecnico di Milano, Milan, MI, Italy.

GeoApt LLC and QGIS Development Team, 2018. Plugin Builder. QGIS Software, Version 2.18.0. https://plugins.qgis.org/plugins/pluginbuilder/.

Harris, C. R., Millman, K. J., van der Walt, S. J., Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., Smith, N. J., Kern, R., Picus, M., Hoyer, S., van Kerkwijk, M. H., Brett, M., Haldane, A., del R'ıo, J. F., Wiebe, M., Peterson, P., G'erard-Marchant, P., Sheppard, K., Reddy, T., Weckesser, W., Abbasi, H., Gohlke, C., Oliphant, T. E., 2020. Array programming with NumPy. *Nature*, 585(7825), 357–362. doi.org/10.1038/s41586-020-2649-2.

Kanevski, M., Foresti, L., Kaiser, C., Pozdnoukhov, A., Timonin, V., Tuia, D., 2009. *Machine learning models for geospatial data*. Faculty of Geosciences and Environment, University of Lausanne, Switzerland, 175–227.

Kaushik, S., 2016. An introduction to clustering and different methods of clustering. https://www.analyticsvidhya.com/blog/2016/11/an-introduction-to-clustering-and-different-methods-of-clustering/. Accessed: 2022-05-25.

Kazakov, E., QGIS Development Team, 2021. Attribute based clustering. QGIS Software, Version 2.2. https://plugins.qgis.org/plugins/attributeBasedClustering/.

Kuhn, M., 2008. Building Predictive Models in R Using the caret Package. *Journal of Statistical Software, Articles*, 28(5), 1–26. doi.org/10.18637/jss.v028.i05.

Lenzi, E., 2020. Simba: systematic clustering-based methodology to support built environment analysis. Master's thesis, Politecnico di Milano, Milan, MI, Italy.

Liu, Y., Li, Z., Xiong, H., Gao, X., Wu, J., 2010. Understanding of internal clustering validation measures. *2010 IEEE International Conference on Data Mining*, 911–916.

MacQueen, J., 1967. Some methods for classification and analysis of multivariate observations. *In 5-th Berkeley Symposium on Mathematical Statistics and Probability*, 281–297.

Nielsen, F., 2016. *Hierarchical Clustering*. Springer International Publishing, 195–211.

QGIS Development Team, 2022. Qgis geographic information system. QGIS Association. http://www.qgis.org.

Qt Project, 2018. Qt designer. https://www.qt.io/design, Version 5.11.2.

Steinbach, M., Ertöz, L., Kumar, V., 2004. *The Challenges of Clustering High Dimensional Data*. Springer Berlin Heidelberg, 273–309.

Tadi, M., Zadeh, M. H., Biraghi, C. A., 2020. *The Integrated Modification Methodology*. Springer International Publishing, 15–37.

The pandas development team, 2021. pandas-dev/pandas: Pandas, version 1.1.15. Zenodo. https://doi.org/10.5281/zenodo.3509134.

Thorndike, R. L., 1953. Who belongs in the family? *Psychometrika*, 18, 267–276. doi.org/10.1007/BF02289263.

Venkat, N., 2018. The Curse of Dimensionality: Inside Out. doi.org/10.13140/RG.2.2.29631.36006.