universität wien

MASTERARBEIT / MASTER'S THESIS

Titel der Masterarbeit / Title of the Master's Thesis

Multidimensional Spatio-Temporal Change - DBSCAN (MDSTC-DBSCAN): A new spatiotemporal clustering technique for the segmentation of transaction prices submarkets. A case study in Vienna.

verfasst von / submitted by

Lorenz Treitler BSc

angestrebter akademischer Grad / in partial fulfilment of the requirements for the degree of

Master of Science (MSc)

Wien, 2023 / Vienna, 2023

Studienkennzahl lt. Studienblatt / degree programme code as it appears on the student record sheet:	UA 066 856
Studienrichtung lt. Studienblatt / degree programme as it appears on the student record sheet:	Master Cartography and Geoinformation
Betreut von / Supervisor:	Ass.-Prof. Dr. Ourania Kounadi, BSc MSc

Acknowledgements

I would like to express my gratitude to all those who have supported and motivated me throughout the preparation of my master's thesis and during my entire course of studies.

First, I would like to thank Dr. Ourania Kounadi for her invaluable supervision of my master's thesis. Her guidance, constructive feedback, and constant motivation have been crucial throughout the process of writing my master's thesis. Additionally, I am grateful for the regular group meetings where we exchanged our experiences, and I would also like to thank all my fellow students who participated in these meetings.

I would like to extend a special thanks to Matthias Grosse and Alexander Bosak for their support, flexibility, and provision of the data. The high-quality data provided played a crucial role in yielding meaningful results and addressing my research questions effectively.

I am profoundly grateful to all friends and my family for their unwavering understanding and constant motivation throughout my studies. I want to thank all those who supported me and made my study time incredibly valuable. I am particularly grateful to my parents, Roland and Gerlinde, for their unwavering support throughout the studies.

Lastly, I would like to thank my girlfriend Sophie for her invaluable contributions throughout the preparation of my master's thesis and my studies. Thank you for your unwavering understanding and support!

Abstract

This master's thesis focuses on the utilization of spatio-temporal clustering methods to delineate submarkets based on real estate transaction prices. Real estate submarkets are spatial areas that exhibit similar characteristics within themselves while differing from other submarket areas. Considering that transaction prices possess spatial autocorrelation and are temporally interconnected, the incorporation of spatio-temporal methods enhances the creation of real estate submarkets. By incorporating the temporal dimension, the resulting submarket areas become more stable.

The aim of this thesis was to compare two spatio-temporal clustering methods, namely Co-clustering with Clustering Geodata Cubes (CGC) and Multidimensional Spatio-Temporal DBSCAN (MDST-DBSCAN), for the creation of transaction price submarkets. However, these two methods failed to yield satisfactory results. Therefore, a novel approach is proposed to delineate these clusters, referred to as Multidimensional Spatio-Temporal Change DBSCAN (MDSTC-DBSCAN). This method incorporates the temporal change in transaction prices along with spatial proximity to identify spatial areas with similar transaction prices. It represents an advancement over MDST-DBSCAN for this use case as it considers the change over time as valuable information rather than a constraint.

Sixteen transaction price submarkets were identified in Vienna. The 19th district exhibited the highest prices, while certain clusters in the outskirts displayed lower prices but with higher rates of price increase. The results, indicating variations in price growth rates, confirm the importance of considering the temporal changes in transaction prices, which exhibit spatial diversity. Additionally, a lower Moran's I value was observed for 2022 compared to previous years, indicating a higher level of homogeneity in transaction prices in 2022. This finding was also supported by the cluster analysis, as less expensive clusters demonstrated higher rates of price increase compared to many of the more expensive clusters.

The results obtained from the proposed method MDSTC-DBSCAN were promising. Future research could focus on further development and improvement of this method. Additionally, applying it to transaction price data from other regions and its potential use for other data could provide valuable insights into the algorithm's usability. Future research can enhance the algorithm and broaden its potential use cases.

Kurzfassung

Die vorliegende Masterarbeit befasst sich mit der Anwendung von räumlich-zeitlichen Clustermethoden zur Abgrenzung von Submärkten auf der Basis von Immobilienkaufpreisen. Immobilienteilmärkte sind räumliche Gebiete, die in sich ähnliche Merkmale aufweisen, sich aber von anderen Submärkten unterscheiden. Da Transaktionspreise eine räumliche Autokorrelation aufweisen und zeitlich miteinander verbunden sind, können durch die Verwendung von raum-zeitlichen Methoden die Bildung von Immobilienteilmärkten verbessert und stabilere Submärkte erstellt werden.

Das Ziel dieser Arbeit war es, zwei raum-zeitliche Clustering-Methoden, nämlich Coclustering with Clustering Geodata Cubes (CGC) und Multidimensional Spatio-Temporal DBSCAN (MDST-DBSCAN), für die Bildung von Kaufpreis-Submärkten zu vergleichen. Da die beiden Methoden keine zufriedenstellenden Ergebnisse lieferten, wird ein neuer Ansatz zur Abgrenzung von Clusters vorgeschlagen, der als Multidimensional Spatio-Temporal Change DBSCAN (MDSTC-DBSCAN) bezeichnet wird. Bei dieser Methode wird die zeitliche Veränderung der Transaktionspreise mit räumlicher Nähe berücksichtigt, um räumliche Bereiche mit ähnlichen Transaktionspreisen zu identifizieren. Die Methode stellt gegenüber MDST-DBSCAN eine Verbesserung für diesen Anwendungsfall dar, da sie die zeitliche Veränderung als Information und nicht als Einschränkung verwendet.

In Wien wurden sechzehn Kaufpreis-Submärkte identifiziert. Der 19. Bezirk wies die höchsten Preise auf, während bestimmte Clusters in den Außenbezirken niedrigere Preise, aber höhere Preissteigerungsraten aufwiesen. Die Ergebnisse, die auf unterschiedliche Preissteigerungsraten hindeuten, bestätigen, wie wichtig es ist, die räumlich unterschiedlichen, zeitlichen Veränderungen der Transaktionspreise zu berücksichtigen. Des Weiteren wurde für das Jahr 2022 ein niedrigerer Moran's I-Wert als in den Vorjahren festgestellt, was auf ein höheres Maß an Homogenität der Transaktionspreise im Jahr 2022 hinweist. Dieses Ergebnis wurde auch durch die Clusteranalyse bestätigt, da die weniger teureren Clusters zum Großteil höhere Preissteigerungsraten aufwiesen als die teureren Clustern.

Die mit der vorgeschlagenen Methode MDSTC-DBSCAN erzielten Ergebnisse sind vielversprechend. Künftige Forschungsarbeiten könnten sich auf die weitere Entwicklung und Verbesserung dieser Methode konzentrieren. Darüber hinaus könnte die Anwendung des Verfahrens auf Transaktionspreisdaten aus anderen Regionen und seine potenzielle Verwendung für andere Daten wertvolle Erkenntnisse über die Verwendbarkeit des Algorithmus liefern. Des Weiteren können künftige Forschungsarbeiten den Algorithmus verbessern und seine potenziellen Anwendungsfälle erweitern.

Contents

Acknowledgements i

Abstract iii

Kurzfassung v

List of Tables xi

List of Figures xiii

List of Algorithms xv

1 Introduction 1
- 1.1 Relevance and Aim of the Thesis . . . 1
  - 1.1.1 Theoretical Background . . . 3
- 1.2 Methodology . . . 5
- 1.3 Research Questions . . . 5

2 Real Estate Submarkets 7
- 2.1 Creating Real Estate Submarkets . . . 7
- 2.2 Usage of data-intensive Techniques . . . 9
  - 2.2.1 Spatio-temporal Methods for Analysis of Submarkets . . . 10
- 2.3 Summary . . . 11

3 Spatio-Temporal Data Mining 13
- 3.1 Overview . . . 13
  - 3.1.1 Spatial and spatio-temporal autocorrelation . . . 14
  - 3.1.2 Spatio-temporal Data . . . 16
  - 3.1.3 Spatio-temporal data mining techniques and applications . . . 18
- 3.2 Spatial Clustering . . . 19
  - 3.2.1 Distance Measurements . . . 19
  - 3.2.2 Clustering Categories . . . 19
- 3.3 Spatio-Temporal Clustering . . . 20
  - 3.3.1 Clustering Spatio-temporal data with an additional attribute . . . 22
- 3.4 Metrics . . . 24
  - 3.4.1 Internal Metrics . . . 26
- 3.5 Clustering Geodata Cubes (CGC) . . . 27
  - 3.5.1 CGC algorithm and input parameters . . . 27

3.5.2 CGC/BCAT in research . . . 28
3.6 MDST-DBSCAN . . . 29
3.6.1 MDST-DBSCAN algorithm . . . 29
3.6.2 MDST-DBSCAN in research . . . 30
3.7 Summary and Findings . . . 31

4 Methodology & ESTDA **33**
4.1 Softwares . . . 34
4.1.1 Python . . . 34
4.1.2 GeoDa . . . 34
4.1.3 GitHub . . . 34
4.2 Study Area - Real Estate Market in Vienna . . . 34
4.3 Data . . . 35
4.3.1 Data Exploration . . . 36
4.3.2 Exploratory Spatio-Temporal Data Analysis . . . 39

5 Clustering Geodata Cubes (CGC) & MDST-DBSCAN **45**
5.1 Clustering Transaction Prices with Clustering Geodata Cubes (CGC) . . . 45
5.1.1 Pre-Processing for CGC . . . 45
5.1.2 CGC clustering . . . 47
5.1.3 Advantages and Disadvantages CGC . . . 49
5.2 MDST-DBSCAN . . . 49
5.2.1 Results of MDST-DBSCAN . . . 51
5.2.2 Problems with MDST-DBSCAN for creating Transaction Price Submarkets . . . 52
5.3 Spatio-Temporal Metric . . . 54

6 Multidimensional Spatio-Temporal Change DBSCAN (MDSTC-DBSCAN) **55**
6.1 Explanation of MDSTC-DBSCAN . . . 55
6.2 Operability of MDSTC-DBSCAN . . . 58
6.3 Results of MDSTC-DBSCAN . . . 59
6.4 Creating spatially continuous Submarket Areas . . . 62

7 Analysis of Clustering Results **63**
7.1 Spatio-Temporal Analysis of Clusters . . . 63
7.1.1 Census Districts based Analysis of Clusters . . . 66
7.2 Comparison of MDST-DBSCAN and MDSTC-DBSCAN . . . 68

8 Discussion **71**
8.1 Answer of Research Questions . . . 71
8.2 Discussion Results . . . 72
8.2.1 Discussion Clusters . . . 73
8.3 Discussion Methods . . . 75

9 Conclusion **77**

Bibliography **79**

10 Appendix **85**
10.1 Code MDSTC-DBSCAN 85
10.2 Code MDST-DBSCAN 89

List of Tables

3.1 Partitioning vs. Density Based Clustering 21
3.2 Selection of clustering methods on ST data with an additional attribute . 24

4.1 First five rows of data 36
4.2 Transaction prices (€/m²) and Number of Purchase Contracts 2018 - 2022 37
4.3 Moran's I values 2018 - 2022 42

5.1 Cluster values MDST-DBSCAN 51

6.1 Cluster values MDSTC-DBSCAN 60

7.1 Linear Regression Coefficients of the Clusters 64
7.2 Comparison of Transaction Price Clusters 65
7.3 Completion Rate and Population Change in Clusters 67
7.4 Comparison of MDSTC-DBSCAN and MDST-DBSCAN 68
7.5 Comparison of initial point chosen for MDSTC-DBSCAN and MDST-DBSCAN 69

List of Figures

3.1 Distribution of Moran's I values . . . 15
3.2 Spatio-temporal Datatypes . . . 17
3.3 BCAT cube: A) data cube, B) regular tri-clusters C) irregular tri-clusters (Wu, Zurita-Milla, Izquierdo Verdiguier, and Kraak, 2018, p.73) . . . 28
3.4 DBSCAN (own representation; based on (Raschka and Mirjalili, 2017, p.373)) 30
3.5 Detecting Neighboring Objects in three Dimensions (own representation; based Choi and Hong (2021, p.4)) . . . 30

4.1 Flowchart of Empirical Part . . . 33
4.2 Purchasing Prices of Dwellings in Vienna 2018 - 2022 (Data: Exploreal) . 35
4.3 Transaction prices of newly built dwellings Vienna 2018 - 2022 . . . 38
4.4 Distribution of Transaction Prices (€/m²) Vienna . . . 39
4.5 Transaction prices of newly built dwellings Vienna 2018 - 2022 . . . 40
4.6 Moran's I Scatterplot . . . 41
4.7 Moran's I 2018 - 2022 . . . 42
4.8 LISA Maps . . . 43
4.9 Differential Local Moran's I (dark red = HH, light red = LH, dark blue = LL, light blue = HL, grey = non-significant) . . . 44

5.1 Grid over investigation area 2022 (Grid Size: 2,671.68 * 2,938.11 meters) . 46
5.2 Spatial and Temporal Co-Clusters by CGC . . . 48
5.3 Spatio-Temporal Clusters created by CGC Co-Clustering after K-Means refinement . . . 49
5.4 Spatio-Temporal Clusters identified by MDST-DBSCAN for Transaction Prices in Vienna 2018 - 2022 . . . 53

6.1 Multidimensional Spatio-Temporal Change Density-Based Clustering of Application with Noise . . . 57
6.2 Spatio-Temporal Clusters identified by MDSTC-DBSCAN for Transaction Prices in Vienna 2018 - 2022 . . . 61

7.1 MDSTC-DBSCAN results . . . 64
7.2 Transaction price submarkets . . . 67

List of Algorithms

1 MDSTC-DBSCAN Algorithm 56

1 Introduction

The aim of the thesis is to build transaction price submarkets of dwellings in Vienna by performing spatio-temporal clustering and to analyse the change of purchasing prices of dwellings in these clusters between 2018 and 2022.

1.1 Relevance and Aim of the Thesis

Location is one of the primary factors influencing real estate prices. Hence, knowing where and how the prices evolve is crucial to understanding the real estate markets of cities. In addition to the spatial autocorrelation of real estate prices, these prices are also temporally linked. (Yao and Stewart Fotheringham, 2016). Data from the real estate database Exploreal was applied to tackle this challenge of creating transaction price submarkets. The data covers transaction prices per m^2 of all newly built apartments in Vienna.

Real estate submarkets are homogeneous areas that share similar attributes within themselves compared to the other real estate submarket areas. When creating these submarkets, the objective therefore is to find clusters that have similar values and to detect areas that differ from the other identified clusters (Kopczewska and Ćwiakowski, 2021). Identifying these patterns in data and distinguishing such areas is the main task of clustering. According to Keskin and Watkins (2017) the use of submarkets has different key advantages. Models that try to predict housing prices show a higher accuracy when using submarkets as input parameters. Furthermore, they can help different stakeholders, such as planners but also participants in the real estate market to understand the patterns and therefore make better decisions.

Submarkets don't have fixed boundaries and may vary over time. Therefore, the temporal in addition to the spatial dimension is also of importance. There is a lack of studies that tried to do spatio-temporal cluster analysis when defining submarkets. Most of these studies didn't use the temporal aspect as a dimension. For instance, Kopczewska and Ćwiakowski (2021) compared the outputs of the different years. In this case the years were separated from each other, and the clusters of the different years were compared to explore the stability of these. Similarly, Costello, Leishman, Rowley, and Watkins (2019) tried to explore the spatio-temporal stability of these submarkets and especially what the drivers are. Hence, in this thesis a different approach to Kopczewska and Ćwiakowski (2021) and Costello, Leishman, Rowley, and Watkins (2019) is applied, as the aim is to not only explore the spatio-temporal stability but rather to actually create transaction price submarkets. The drivers are not discovered in this thesis but the aim is to build more stable submarkets, as the temporal aspect is also part of the clustering. Time is

utilized in a manner that allows for the consideration of temporal changes in transaction prices.

Keskin and Watkins (2017) claimed that the utilization of new and better methods may be of greater importance for creating submarkets, than to simply increase the amount of data utilized. Shi and Pun-Cheng (2019) stated that (spatio-temporal) clustering techniques for data that include more dimensions should be created. Even though, nowadays there already are more algorithms than in 2019 that tackle this problem, these methods have yet to be analysed. Choi and Hong (2021) also were aware of the problem that most spatio-temporal clustering algorithms only used event data, without additional attributes. For this reason, they developed Multidimensional Spatio-Temporal Density-Based Spatial Clustering of Applications with Noise (MDST-BDSCAN) – an algorithm that overcomes this problem. Ku, Nattino, Grootes, Izquierdo-Verdiguier, Girgin, and Zurita-Milla (2022) introduced Clustering Geodata Cubes (CGC) – a python package which allows to cluster data in up to three dimensions.

As the latter of these two methods (CGC) was developed in 2022 and has not been analysed intensely by papers, one objective of this thesis is to test it. Further more, the MDST-DBSCAN clustering algorithm shall be used to create transaction submarkets. Choi and Hong (2021) stated that one problem for this algorithm is finding the best parameters automatically. Therefore, one aim in this thesis regarding the MDST-DBSCAN algorithm is to tackle this problem and to also go further and detect potential improvements for both methods.

However, the implementation of these algorithms showed that neither of them was perfect to delineate transaction prices submarkets. Hence, a third method - Multidimensional Spatio-Temporal Change DBSCAN (MDSTC-DBSCAN) was developed.

In a last step the outputs of the analysed clustering algorithms are compared with internal evaluation metrics. Hence, the aim is to find the one clustering algorithm that is best for the used data respectively datatype. Davies-Bouldin-index (Davies and Bouldin, 1979) was applied and modified for spatio-temporal data to compare the clustering outputs. This is necessary as different input parameters and algorithms can produce different outputs to the same input data. Hence, knowing which algorithm is better suited for this kind of data is crucial to detect the most reliable pattern (Shi and Pun-Cheng, 2019).

This master's thesis seeks to apply the aforementioned methods to delineate submarkets based on transaction price data within Vienna. The real estate market reports of the last years show a strong increase in prices for all of Austria and Vienna (Nationalbank, 2023). There are different reasons for these increasing prices such as a rise of material and therefore building costs, that affect the whole real estate market all over Austria. Market reports (such as BUWOG Group GmbH and EHL Wohnen GmbH (2022)) show an analysis of different districts but cannot overcome the problem of Modified Areal Unit Problem (MAUP). As these districts are not homogeneous areas it is necessary to also analyse the prices without this administrative units to identify where clusters are and where the prices have risen differently than in other regions.

Stadt Wien states that it is necessary to offer as many newly built apartments as needed

to let the prices stay stable (Stadt Wien, 2020). By observing the detected cluster areas, one possible outcome might be to detect areas where prices have risen comparatively much and where intervention might be needed. Therefore, as stated in a previous paragraph, one of Keskin and Watkins (2017) advantages of these clusters could be demonstrated here. Furthermore, this method will analyse how homogeneous the apartment market in Vienna was before Covid and nowadays. Still, the aim is not to explore deeply the reasons why the prices have developed differently in different clusters. Potential causes will be mentioned but a deeper analysis of the reasons would require further, more detailed research and will be left to the experts in this field.

The data of Exploreal only includes transaction prices of new housing construction projects with at least five dwelling units. Hence, the total market of second-hand flats and smaller projects with less than five units are not examined. Therefore, the results will reflect real estate submarkets of the new housing construction projects, that will be a good proxy for the total market. As a consequence of the dataset, not all Viennese areas are members of clusters in the results; unavoidably areas with no newly built houses will be excluded.

1.1.1 Theoretical Background

There are many studies that have tried to create housing submarkets. Keskin and Watkins (2017) mention three different methods to define these – the approach of predefined areas, the use of statistical methods such as cluster analysis, and expert-based methods. Some studies, such as Keskin and Watkins (2017), found that the data-intensive PCA/cluster analysis showed worse results than the expert-based method (Keskin and Watkins, 2017). However, this work focuses on statistical and data-intensive methods, and proposes a new approach with spatio-temporal data. The aim here is to find submarkets that have similar values within the clusters and are as different as possible from the others (Keskin, 2022).

There are not many studies in housing, specifically in creating submarkets, that have used similar spatio-temporal methods and analysis as applied in this thesis. One advantage of spatio-temporal analysis compared to 2D-spatial analysis is the ability to examine how the clusters behave over time and verify their stability. Kopczewska and Ćwiakowski (2021) analysed the stability of submarkets by comparing the outputs of clusters from different years. This study serves as an example where time was used sequentially and not as an additional dimension of the data. While this method already provides new insights into the data analyzed in this thesis, the aim is to leverage the temporal attribute as an additional dimension.

The difference of spatio-temporal analysis compared to spatial analysis is the inclusion of the time dimension. Incorporating an additional attribute to the three dimensions X, Y (Location), and T (Time) is often a challenging task (Shi and Pun-Cheng, 2019). Spatio-temporal analysis and research have gained increasing importance and have been utilized in various domains, including climatology, epidemiology, crime analysis, urban modelling, and many more. The objective of these analyses is to identify patterns and structures in the data (Das and Ghosh, 2020).

As mentioned previously, many papers have already applied spatio-temporal clustering

to gain new insights on different topics. However, often they have only used either the temporal or the spatial component to perform the clustering task. Co-clustering techniques combine these dimensions to cluster both the spatial and temporal dimensions simultaneously and have recently become a trend in spatial research. There have been co-clustering studies that attempted to discover spatio-temporal patterns in temperature, disease, mobility data, and more. Wu (2022a) employed the Bregman co-clustering algorithm to identify areas with similar trends rather than absolute values. Several recent papers have utilized unsupervised techniques in the urban context and further developed existing methods, as demonstrated by Choi and Hong (2021). Additionally, there has been a significant number of publications in the last few years (Wang and Biljecki, 2022).

Agrawal, Garg, Sharma, and Patel (2016) defined criteria that spatial-temporal clustering algorithms should fulfill. These methods should exhibit flexibility in terms of the shape and dimensionality of the data, and the input data should not heavily influence the method. Furthermore, the method and its results should be easily understandable. Based on these criteria, the authors developed the OPTICS algorithm to make it applicable for spatio-temporal data. The OPTICS algorithm orders data to discover clusters. A comparison with ST-DBSCAN revealed that OPTICS outperforms the latter method in terms of runtime and qualitative performance.

One way to categorize clustering are 3 groups – partitioning, hierarchical, and density-based methods, which group the clustering methods based on how the cluster similarities are calculated. Some studies also use more than these three categories. Another way would be to divide these methods into groups according to what dimensions are used – non-spatial, spatial or spatio-temporal. Since the latter group plus an additional attribute – the transaction prices – is applied in this thesis, a method that takes all dimensions into account is necessary. However, the problem is that many of these clustering algorithms only work with event data, without an additional attribute. (Choi and Hong, 2021)

Oliveira, Santos, and Pires (2013) previously proposed a spatio-temporal clustering approach that incorporated spatial, temporal, and an additional attribute. In a similar vein, Choi and Hong (2021) developed the MDST-DBSCAN method, an extension of DBSCAN and ST-DBSCAN, capable of analyzing an additional attribute alongside spatial and temporal dimensions. This algorithm is analyzed in this thesis, as mentioned earlier.

Another clustering method is co- and tri-clustering, which can group data based on spatial, temporal, and another attribute. An example of such a clustering method is BCAT I (Bregman cuboid average tri-clustering algorithm), which can analyze 3D data with high precision but is computationally less efficient compared to other clustering methods (Wu, Wei, and Li, 2020). One of the latest Python packages for performing co- or tri-clustering tasks is the Clustering Geodata Cubes (CGC) package. Ku, Nattino, Grootes, Izquierdo-Verdiguier, Girgin, and Zurita-Milla (2022) developed this package, which is capable of performing co-clustering (spatial, temporal) or tri-clustering tasks. The latter method utilizes the Bregman cube average tri-clustering (BCAT) algorithm to cluster data based on the spatial, temporal, and an additional dimension.

1.2 Methodology

In section 3, various methods are compared, and the findings are valuable for refining the empirical part. This section aids in the exploration of relevant concepts of spatio-temporal data, clustering, as well as in identifying methods to compare the outputs of the clustering algorithms. Furthermore, in 3 the two clustering algorithms – CGC and MDST-DBSCAN - are explored. Chapter 2 assists in understanding the concept submarkets are and consequently exploring the requirements that transaction price submarkets must fulfill.

The practical part of the study primarily employed Python. Following data collection, the first step in the empirical part involved conducting descriptive analysis of the data. This analysis was necessary to gain insights into the data and illustrate the geographical distribution of prices. Spatial methods, such as Morans' I, were utilized during this step. Subsequently, the main part of the study entailed the use of clustering methods, specifically CGC and MDST-DBSCAN, along with the development of an extension to MDST-DBSCAN. When performing the co-clustering, polygons are needed as inputs. In order to perform co-clustering, the input required the definition of polygons. To address the Modified Areal Unit Problem (MAUP), which remains a challenge in clustering tasks (Shi and Pun-Cheng, 2019), a regular raster was adopted instead of administrative areas. The goal with MDST-DBSCAN was to automatically determine the optimal input parameters using Python. However, the outputs of the clustering algorithms indicated that neither approach met the requirements. Hence, a new clustering algorithm - MDSTC-DBSCAN - was developed as an extension of MDST-DBSCAN.

For this analysis, solely Python and open-source geoinformation systems and methods were employed. Consequently, it was necessary to convert the code of the MDST-DBSCAN algorithm, originally implemented in R, to Python. The developed framework, was published in a publicly accessible GitHub repository. This ensures that future work can build upon the existing framework and further improve the algorithms.

The final step involved examining the results of the clustering methods. The outputs of the algorithms were compared, using metrics and other comparative parameters. Among these clustering techniques, the clusters produced by the method that demonstrated higher accuracy were further explored.

1.3 Research Questions

The research questions were grouped in three objectives. The first one deals with the submarkets themselves and how the transaction prices have evolved during the investigation period. The best clustering technique was applied to create the submarkets and spatio-temporal methods were after that used to inspect those results.

Objective 1: Analyse Vienna's real estate prices.

- **RQ1:** What are the transaction price submarkets in Vienna?
- **RQ2:** Does the price rising rate differ between the submarkets?

- **RQ3:** How has the homogeneity of the transaction prices of apartments in Vienna changed during the COVID pandemic?

The second objective is to compare the spatio-temporal clustering methods MDST-DBSCAN and CGC. Further more, potential improvements of the methods should be proposed and developed.

Objective 2: Explore and compare a point versus polygon based spatio-temporal clustering technique for delineating real estate submarkets of transaction prices.

- **RQ1:** Which MDST-DBSCAN parameters derive more stable clusters for the study area of Vienna?
- **RQ2:** Is CGC or MDST-DBSCAN a better method to delineate real estate submarkets?
- **RQ3:** What improvements can be made to enhance the performance of the superior method, that would achieve even better results?

Since only Python is used in this thesis and MDST-DBSCAN is implemented in R, a Python version of this algorithm has been developed.

Technical objective 3: Develop a python implementation of the MDST-DBSCAN algorithm.

2 Real Estate Submarkets

The definition of real estate submarkets has been deeply investigated. The objective of submarket areas is to divide the real estate market, respectively the study area, into smaller areas. Real estate submarkets are homogeneous areas that share similar attributes within themselves compared to the other real estate submarket areas (Kopczewska and Ćwiakowski, 2021). Another definition is that apartments within submarkets are similar replacements for the other apartments but unfavorable replacements for apartments of other submarkets (Bourassa, Hamelink, Hoesli, and MacGregor, 1999). Hence, submarkets have a comparable objective as clustering methods - to group similar objects.

Another feature of submarkets is that these areas must be geographically contiguous and need borders, that can either be real administrative or other borders (Wu, Wei, and Li, 2020). However, no common definition of submarkets exists. Yet, submarkets must meet the three characteristics of **similarity, geographic contiguity and replacement**. As mentioned above in different words, submarkets are considered to be present, if the prices are similar within one submarket and prices vary between submarkets for a "hypothetical, standardized housing unit" (Watkins, 2001). Even though this is hypothetical and the same dwelling does not exist in different areas, new construction apartments, which are used in this thesis, are better comparable than second-hand flats. This is an advantage of the data used in this thesis.

Keskin and Watkins (2017) enumerate three advantages of submarkets, highlighting their necessity. Firstly, predictive data mining models perform better when submarkets are used as input parameters. Secondly, submarkets can be beneficial to different stakeholders, including those with development or strategic tasks. Lastly, other participants of the housing market, such as buyers, investors, and real estate agents, can also benefit from submarkets. For instance, submarkets can help these groups to become more aware of differences between various areas.

After the definition of submarkets and highlighting the reasons why the building of submarkets is necessary, the next section explains different approaches for the creation of submarkets.

2.1 Creating Real Estate Submarkets

Before creating submarket areas, the first step was to be aware that such submarkets exist. Many different papers, starting from the 20th century, discussed the existence of housing submarkets. In the mid-20th century, filtering methods were introduced that presumed the existence of submarkets. Researchers found out that the area of the dwelling was relevant to its price. Additionally, studies have demonstrated that the geographic location

impacts the selection of a property during the property-buying process. This comes along with imbalances in markets that, according to these studies, lead to submarkets. However, in later research, long-term imbalances were not confirmed, and it was stated that the market tends to be more stable in the long term. The submarkets and dissimilarity of prices are considered as a snapshot in time, meaning that stability exists in these areas. However, in the long term, the prices of submarkets converge. Nowadays, the existence of submarkets is not in doubt (Keskin and Watkins, 2017; Watkins, 2001).

The basic identification of submarkets is as follows. The first step is to trace possible submarket-areas by dividing the data. Next, modelling methods can be employed to determine standard prices of dwellings with similar quality within each submarket. In a last step tests assessing statistical significance of price dissimilarities between submarkets should be performed (Keskin and Watkins, 2017). Especially the first step is of importance for this thesis. Hence, the delineation of data to submarkets is explained in the next paragraphs.

In earlier times, the creation of submarkets heavily relied on existing expertise of the housing market. Submarkets were typically created by grouping administrative units, with clustering being one method to perform this task. Studies found out that two of the submarket's characteristics, namely similarity and replaceability, could not be fully met by these approaches. The emergence of new statistical techniques in addition to a rising amount of data, particularly locational data of individual dwellings, led to the 'microstructural turn' in this research area in the 2000s. One of these methods was spatial clustering, which achieves the submarket's objective of creating spatially contiguous areas. By grouping similar objects, it also meets the similarity characteristic of submarkets. (Keskin and Watkins, 2017; Kopczewska and Ćwiakowski, 2021; Wu, Wei, and Li, 2020)

Keskin and Watkins (2017) have identified and described three different approaches for delineating submarkets. The **hierarchical perspective** involves both existing expertise and empiric outcomes to group neighborhoods into submarkets. This method can rely on different administrative units, such as school districts, census tracts or postal code areas. However, a major disadvantage of this method is the need for existing expertise to select the appropriate administrative units, as there is no single best area or predefined criterion for selecting them. Another approach is to let **experts form the submarkets**. This approach is based on the knowledge of experts with regard to the diverse elements that comprise submarkets, such as neighborhood properties. Submarkets are, according to this method, areas with dwellings that share similar characteristics within and a greater difference to the characteristics of other submarkets. The studies have shown promising results, with the difficulty of discovering the optimal technique to represent the agent's opinions. (Keskin and Watkins, 2017)

The third method to create submarkets are **data-driven methods** or statistical techniques such as principal component analysis (PCA) and cluster analysis. There have been various techniques developed that use data to delineate submarkets. For instance, methods to reduce the number of features, such as PCA have been employed to create submarkets based on the data. However, in the past some studies detected this method to create less appropriate submarkets than the expert-based techniques. As this thesis also

utilizes data-intensive methods, the next section outlines some data-intensive techniques that have been employed. (Keskin and Watkins, 2017)

2.2 Usage of data-intensive Techniques

The creation of submarkets using data-intensive methods can be distinguished based on the types of data utilized. Many studies have focused on dwelling-specific data, such as their prices and the components that influence those. However, non-property-data, such as the neighborhood, were often not considered. Three kinds of data can be differentiated, when creating submarkets: **exogenous, external and endogenous**. Exogenous data refers to characteristics of dwellings, while external data refers to the geographic position and the environment of an apartment. These two factors influence the endogenous factor: the price of a dwelling. These different data types can be utilized and combined to define and delineate submarket areas. (Kopczewska and Ćwiakowski, 2021)

So, most delineations of submarkets utilize more than just the prices of dwellings, using additional factors such as property characteristics, buyers' characteristics, and neighborhood attributes (Watkins, 2001). Transaction Prices can be defined as a function

$$p = f(S, N, L)$$

where S = property characteristics, N = neighborhood attributes and L = locational characteristics. However, this definition neglects the effect of spatial dependency and heterogeneity (Yao and Stewart Fotheringham, 2016). That is why it is important to also consider the effect of spatial autocorrelation on housing prices (Wu, Wei, and Li, 2020). It is well-established that prices near to each other are correlated, for instance due to shared neighborhood attributes and other characteristics (Huang, Wu, and Barry, 2010). It is worth noting that housing prices are also influenced by macroeconomic characteristics, such as inflation rate (Soltani, Pettit, Heydari, and Aghaei, 2021). However, these factors are not considered in this thesis.

The functionality of data-driven approaches was already described above. Various methods have been applied in studies to create submarkets, including ordinary least squared clustering (OLS), clustering the factors of geographically weighted regression (GWR), principal component analysis (PCA) in combination with clustering or regionalisation (Kopczewska and Ćwiakowski, 2021). According to Wu, Wei, and Li (2020) some of the most popular algorithms in submarket delineation include K-means, Ward's clustering and CART decision trees. Clustering the coefficients of GWR has also been a popular method in submarket research (Kopczewska and Ćwiakowski, 2021).

GWR has been used in many studies and, in addition to traditional regression, incorporates the geographic location as input. GWR has also been further developed to include the effect of other factors such as educational features (Wang, Zhang, and Zhao, 2023) or time (Yao and Stewart Fotheringham, 2016). PCA has been utilized to reduce the number of features used as input data to the submarket creation, such as those mentioned above that define the price of dwellings. For example, Soltani, Pettit, Heydari, and Aghaei

(2021) used the PCA output as input to the regionalisation method SKATER to delineate submarkets.

In their study, Bourassa, Hamelink, Hoesli, and MacGregor (1999) suggested that high quality transaction data in combination with spatial methods would be an interesting field, for instance to calculate the nearness and spatial dependence. Moreover, they also state that the robustness could be tested if temporal data were available. As mentioned, there are studies, for instance Costello, Leishman, Rowley, and Watkins (2019), that use spatio-temporal data to investigate submarkets. The reason why in this thesis only transaction prices are used to gain insights is that the focus lies on the spatial-temporal factor. It should be noted that the objective of this thesis is not to create submarkets as defined above, but rather to create submarkets of transaction prices.

2.2.1 Spatio-temporal Methods for Analysis of Submarkets

The temporal behaviour of submarkets is an interesting research field that has not yet been deeply analysed. However, the necessity of analysing time in combination with submarkets has long been evident. As it is nowadays clear that submarkets do exist, the temporal component is an additional feature of relevance to the characteristics mentioned above. Real estate prices are not only spatially but also temporally linked, which can lead to new submarket boundaries over time. Studies have for instance examined this change of submarkets over time or the stability of submarkets. (Kopczewska and Ćwiakowski, 2021; Yao and Stewart Fotheringham, 2016; Costello, Leishman, Rowley, and Watkins, 2019)

Research on temporal effects in submarkets began in the 2000s, with, as mentioned above, for instance studies investigating the stability of submarkets over time. Yao and Stewart Fotheringham (2016) applied GWR to analyse the spatiotemporal change of housing prices and their influential variables. They compared the output of the years and incorporated time by analysing if these variables remain the same over the years. Costello, Leishman, Rowley, and Watkins (2019) investigated how the GWR coefficients vary over time. Geographically and temporally weighted regression models (GWTR) can examine both the spatial and temporal dimension and has been used in research (Huang, Wu, and Barry, 2010).

In a recent study, Soltani, Pettit, Heydari, and Aghaei (2021) compared different regression models and clustered their results to delineate real estate submarkets. The study discovered that models incorporating both the spatial and temporal dimensions achieved better results than those using only the spatial dimension. Additionally, the study explored the existence of spatio-temporal non-stationarity. Huang, Wu, and Barry (2010) similarly discovered the advantage of applying time to GWR and incorporating GWTR. The GWTR-model achieved better results than the GWR model. Kopczewska and Ćwiakowski (2021) analysed the stability of submarkets by comparing the clustering outputs of GWR coefficients from different years, exploring how these boundaries of submarkets change over time and how stable these areas are. Incorporating the temporal component here provided valuable insights into changes in submarkets, their factors, and their stability, which can inform the planners, as mentioned above.

Costello, Leishman, Rowley, and Watkins (2019) argued that while there is an increasing understanding of the factors that affect housing prices, little is known about the temporal changes and reshaping of submarket boundaries. Therefore, further research on temporal stability of price submarkets is necessary. The drivers of the changes are diverse and can for instance be inner-city movement of people, policies or simply changing tastes. However, since this thesis does not explore these, there is no need of analysing them further.

Research on spatio-temporal submarket delineation showed spatio-temporal variation and non-stationarity of house prices and submarkets. The analysed methods mostly analysed the temporal stability or analysed how different characteristics change. However, the objective of the methods applied in this thesis is to actually create more stable submarkets by incorporating the temporal dimension in addition to the spatial dimension.

2.3 Summary

To summarize, it has to be stated that the submarkets that are formed in this thesis differ from the traditional definition of submarkets as only transaction prices are being used to detect spatial areas. As Keskin and Watkins (2017) stated, the use of better methods might be more important than using more data. Hence, this is an objective of this thesis, as two novel methods are used and compared. On the other hand, it is clear that in order to create real submarkets, as defined in literature, these clustering methods in a future step should also be applied to the other data, mentioned in the sections above. The aim is to actually apply methods that can create spatio-temporally stable transaction price submarkets. This is a different approach to for instance Kopczewska and Ćwiakowski (2021), who focused on the stability of submarkets without creating spatio-temporal submarkets.

3 Spatio-Temporal Data Mining

The following sections cover relevant concepts that are necessary to understand the spatio-temporal clustering methods discussed in this thesis. First, an introduction to different topics of spatio-temporal data mining, here on called STDM, is presented. In the subsequent sections, clustering, the main topic of this thesis, is explored in more detail, with a focus on spatio-temporal clustering. Next, internal evaluation metrics are examined, with the aim to detect one or more internal evaluation metrics which can compare the outputs of the two clustering algorithms the best. In a last step these two clustering techniques - Clustering Geodata Cubes (CGC) and Multidimensional Spatio-Temporal Density-based Spatial Clustering of Application with Noise - shall be theoretically analyzed and compared.

3.1 Overview

Spatial analysis is rather difficult to define and there have been different definitions in the past decades. However, it can be summarized as different approaches that

> "describe, analyse, simulate and predict spatial patterns and/or processes of geographic phenomena [...]." (Gao, 2021, p. 1 - 2)

Hence, spatial analysis includes a wide range of techniques that try to find patterns in spatial data and furthermore try to predict how the data may develop in the future.

Data mining can be seen as "knowledge mining from data [...]." (Han, Kamber, and Pei, 2012, p. 6). Data Mining is a process to discover insights from large data sets with the objective to bridge the gap between raw input data and meaningful information. Data mining tools are designed to help to create information from the big amount of data available today (Han, Kamber, and Pei, 2012). There are numerous fields, such as climate science, neuroscience, environmental science, epidemiology, crime and more, that gather ST data and therefore would profit of new or better developed STDM-methods. New challenges and problems come along with ST data due to it's complex structure, that differs from data applicable to traditional data mining. Especially the combination of the spatial and temporal attributes offer new chances but also challenges, obstacles and problems (Atluri, Karpatne, and Kumar, 2019).

Data used in spatial data mining needs some geographic domain, which can either be points, lines or polygons, to be spatial. Following the definition of spatial analysis two dimensions are included – the x- and y-coordinate. According to Gao (2021) time is already included in spatial data, as it captures changes over time. However, whenever

time is added to the two-dimensional space (x and y) at least three dimensions (easting, northing, t) are the result.

While spatial and spatio-temporal data share some similarities, there are also many differences between them. Spatio-temporal data, here on called ST data, according to Atluri, Karpatne, and Kumar (2019), is characterized by two major features - it is **auto-correlated** and may be non-stationary, spatially and temporally. ST data therefore behaves contrarily to the presumption of many data mining tasks, where identically distributed (i.i.d.) data is assumed. As a result, common data mining methods may not be effective for analyzing ST data, as they disregard the spatial and temporal dependence of ST data. To tackle this challenge, ST data mining tasks have to be aware of this fact to get accurate and meaningful results. (Atluri, Karpatne, and Kumar, 2019)

3.1.1 Spatial and spatio-temporal autocorrelation

Spatial autocorrelation describes to what degree a value is correlated with its neighboring values. Therefore, it measures the dependence and association between values, and analyzes how they are distributed across space. Positive autocorrelation shows that higher values are nearby higher, respectively lower values are nearby lower values. Hence, spatial autocorrelation follows the **first law of geography**, that states that values near to each other are more similar than to values farther away. Spatial autocorrelation therefore is, what makes the analysis of spatial patterns interesting and of relevance (Grekousis, 2020). Adding the temporal component to the data makes it not only spatially but also temporally autocorrelated (Atluri, Karpatne, and Kumar, 2019).

There are different metrics that examine to what degree data is spatially autocorrelated. One of them is Equation 3.1 **Moran's I index**, which calculates the global autocorrelation:

$$I = \frac{n}{\sum_{i=1}^{n}\sum_{j=1}^{n} w_{ij}} \cdot \frac{\sum_{i=1}^{n}\sum_{j=1}^{n} w_{ij}(x_i - \bar{x})(x_j - \bar{x})}{\sum_{i=1}^{n}(x_i - \bar{x})^2} \tag{3.1}$$

N refers to the count of all objects, x_i representing the value of object i, respectively this also accounts for x_j and j. $\bar{x}$ represents the mean of the values, w the spatial weights, for instance between object i and j. Moran's I is calculated to detect whether nearby objects are more similar than objects farther away. Spatial weights are used to standardize the Moran's I value. If row standardization is applied, the outcome of the formula varies from -1 to 1. A score above 0 indicates that the values are clustered and spatially autocorrelated. Respectively, values below 0 show that values next to each other have differing values and therefore negative autocorrelation. The closer the outcome is to 0, the less autocorrelation is detected in the data. Figure 3.1 provides a visual representation of this distribution of the values and the related Moran's I values. Clustered data leads to a high Moran's I value, with I being greater than the expected value, and vice versa, dispersed data shows a low Moran's I value (Grekousis, 2020).

There are also other metrics that measure global autocorrelation (e.g. Geary's C index, General G-Statistic). But since these were not applied in this thesis these are not described further. Global autocorrelation only tells how the data is distributed but it doesn't show

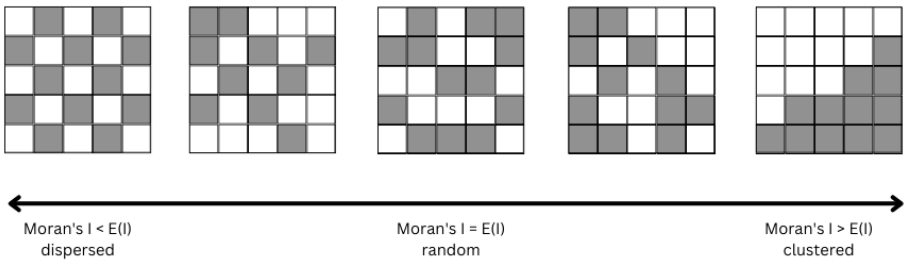


Figure 3.1: Distribution of Moran's I values
(own representation; based on Grekousis (2020, p.232))

where the data is clustered and tells nothing about the values. To examine the local autocorrelation other metrics such as **Local Moran's I** are needed. This measurement detects hot and cold spots and hence, more about the actual values of the data is analysed. (Grekousis, 2020)

However, ST data is the subject of this thesis. So when examining autocorrelation, in addition to spatial also spatio-temporal autocorrelation must be explored. **Bivariate Moran's I** can fulfill this task as it is able to, in addition to it's general purpose of showing the correlation of two variables in the context of neighboring values, compute the spatio-temporal connection of a single attribute. Again, as explained above, when introducing global Moran's I autocorrelation, the output lies between -1 and 1 if row standardization is used. Bivariate Moran's I compares the mean of the neighboring values at an earlier time with the analysed value. Therefore, one is able to see how earlier nearby values influence the value at a current timestamp (Grekousis, 2020). However, there is also some critique to this method, that it is often understood wrongly by users (Anselin,Luc, 2019). Due to it's complexity, Bivariate Moran's I is not utilized for this specific data in this thesis. For the sake of completeness it is still mentioned in this paragraph.

Another concept of Moran's I that includes time is **differential Moran's I**. This concept discovers if the rise of a value of one attribute goes along with a similar rise of the nearby values of that same attribute. Hence, it tells if these changes are spatially clustered. A positive autocorrelation here indicates that the nearest values have similar shifts. Either the value and the nearest neighbors show a high shift leading to a high-high type or they only show a small change showing a low-low clustering. Low autocorrelation here means that one's attributes change is high, while the surrounding neighbors show a low change of values. (Grekousis, 2020)

According to Atluri, Karpatne, and Kumar (2019) besides autocorrelation another characteristic of ST data that has to be considered, when performing classical data-mining

tasks, is **heterogeneity**. Autocorrelation provides insights about spatial clusters and outliers and hence spatial heterogeneity. Spatial heterogeneity means that spatial data is non-stationary. Data in classical data mining methods, as mentioned above, often require complete randomness. In the context of spatial data this would be complete spatial randomness and the data would have to comply with the **two orders of stationarity**. The first order describes, that all locations have the same likelihood of an event to happen there. The second order states that the occasion of one location is not relying on or influenced by happenings at other locations. Hence, spatial and ST data cannot meet this. If data would meet these stationarity-requirements spatio-temporal analysis would not be of interest (Grekousis, 2020). Dealing with temporal data brings another aspect in addition to the spatial component, as data can also show temporal non-stationarity or heterogeneity. For instance, plant measurements and the analysis of these measurements show different outcomes in different seasons of the year (Atluri, Karpatne, and Kumar, 2019).

These concepts play a role in the empirical part of this thesis and shall help when analysing the data and the outcomes. The subsequent chapter deals with this ST data, as it is necessary to understand what kind of ST data is used in this thesis and how it differs from other data.

3.1.2 Spatio-temporal Data

ST data has many different expressions. It can be expressed in different levels of graininess from seconds to years, depending on the data. Time can be cyclic, which can for instance be the nighttime/daytime or seasons of the year. Here, time always returns to a starting point. However, in this master's thesis time is not analysed in it's cyclical state but rather time is only examined linearly. Knowing the time of the day or year when a transaction was executed is not necessary. It is only interesting to be aware when a transaction happened in comparison to other transactions (Gao, 2021).

ST data contains up to five dimensions, for instance when the data contains a Z-value (e.g. elevation) and an attribute (Gao, 2021). However, spatial data mostly has the two locational dimensions (x and y) and a variable. Adding time to these three dimensions, which is the case in this thesis, leads to four-dimensional data (Shi and Pun-Cheng, 2019; Oliveira, Santos, and Pires, 2013). The non-spatial attribute, for instance transaction prices, can change over time. In this case the prices may rise if a flat is sold again. However, in the data of this thesis such resales are not included but prices at a certain location may also change over time, if a similar flat in the same building is sold (Gao, 2021).

Spatio-temporal datatypes have been classified into different groups. Divisions were made according to how the spatial and temporal components are used in the data (Atluri, Karpatne, and Kumar, 2019). However, the number of categories differs between papers and is not permanent. For instance, Atluri, Karpatne, and Kumar (2019) only present four categories, including raster data, while for instance Kisilevich, Mansmann, Nanni, and Rinzivillo (2010) introduced five groups, that only contain point data. These datatypes are shown in Figure 3.2 and include events, geo-referenced variables, geo-referenced

time series, moving points and trajectories. While events, geo-referenced variables and time-series show data of a fixed location, moving points and trajectories show dynamic locations, with the data moving across space. The former datatypes are more relevant for this thesis, that is why the latter two categories are not explored more deeply.

Event Data contains three dimensions - the location (x,y) and the temporal (t) information. An example would be epidemic disease cases. The last temperature or precipitation value at a weather station that is monitored over time would be an example of geo-referenced data items. These are variables at a certain location that are being witnessed over time. This datatype has a non-spatial attribute in addition to x, y and t. Geo-referenced time-series retain all the data of an item developing over time at such a fixed location. An example here would again be temperature, collected daily at a city over years. In this case the information about for example the daily temperature is kept over the period and the whole period can be analysed. (Ansari, Ahmad, Khan, Bhushan, and Mainuddin, 2020; Kisilevich, Mansmann, Nanni, and Rinzivillo, 2010)

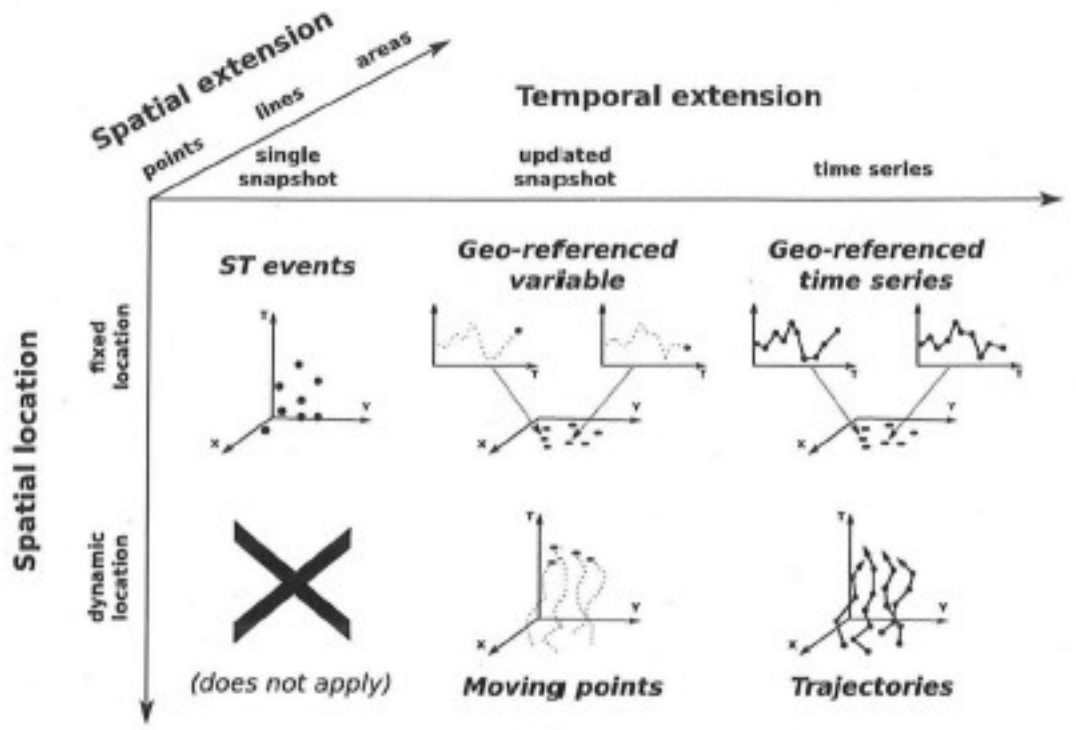


Figure 3.2: Spatio-temporal Datatypes
(Kisilevich, Mansmann, Nanni, and Rinzivillo, 2010)

However, the datatype of an object does not have to be in a constant state and can be converted to other datatypes, for instance if the ST data mining method doesn't allow a certain datatype as input. One example is to transform event data to raster cells through

summarizing the event counts in the raster cells (Atluri, Karpatne, and Kumar, 2019). Another one is the transformation of transaction prices used in this thesis to georeferenced time series. This allows to cluster them into data cubes, which is required as input for the Clustering Geodata Cubes (CGC) method (Ku, Nattino, Grootes, Izquierdo-Verdiguier, Girgin, and Zurita-Milla, 2022).

When observing the mentioned ST data types one question, according to the data used in this thesis arises: Which category contains data with a flexible location and an additional non-spatial attribute? Oliveira, Santos, and Pires (2013) stated this problem that, at least in 2013, no clustering approaches were using ST data with an additional attribute. Atluri, Karpatne, and Kumar (2019) mentioned, that event data can have an additional attribute. Oliveira, Santos, and Pires (2013) was the first clustering algorithm to incorporate an additional attribute in addition to ST data. Even though, nowadays more clustering techniques that can apply 4D-data exist (Choi and Hong, 2021; Ku, Nattino, Grootes, Izquierdo-Verdiguier, Girgin, and Zurita-Milla, 2022; Wang, Cao, and Yu, 2022), a classification of this datatype is still missing. No general term for ST data with an additional attribute exists in literature. However, if a classification is needed the data examined in this thesis would be categorized somewhere between event data and geo-referenced time series. It is also important to note that in the literature, spatio-temporal data with an additional attribute is sometimes defined as 4D and at other times as 3D. This is due to the different ways of handling space. Space can either be seen as a single dimension or treated as two separate dimensions - the x-coordinate and y-coordinate. This further emphasizes the necessity of classifying this type of data.

3.1.3 Spatio-temporal data mining techniques and applications

The aim of STDM is to explore new insights and patterns from spatial-temporal data (Soltani, Pettit, Heydari, and Aghaei, 2021). Different techniques can be used to gain these insights. Das and Ghosh (2020) state six spatio-temporal techniques: prediction, change detection, outlier detection, hotspots detection, partitioning and summarization and coupling/tele-coupling. Other papers have similar groups but name them differently (Atluri, Karpatne, and Kumar, 2019).

The aim of spatio-temporal **prediction** is to let models learn from the independent variables to predict another variable in the future. ST **change detection** tries to find changes in a given dataset, for example temperature change, of a certain region. Finding data points that are different to the trend is the aim of **outlier detection**, while the objective of the **hotspot detection** is to find areas where similar features occur more often than expected. ST **Partitioning** refers to grouping features that are alike. For instance, to explore areas in trajectory data where cars move similarly. ST **Coupling** has the objective to explore two/more features with similar spatial and temporal expression. (Das and Ghosh, 2020)

In the subsequent chapter, clustering is analysed more deeply. As the primary objective is to group similar points, and given that the other techniques are not deployed in this thesis, an analysis is omitted. (Atluri, Karpatne, and Kumar, 2019)

3.2 Spatial Clustering

The objective of (spatial) clustering tasks is to group data into clusters with features that show similar values. Therefore, the values in one cluster should be alike the other values in this certain cluster and as different as possible to values in other clusters. These clusters are formed by different distance measurements. Additionally to classical clustering, spatial clustering uses the location (x, y) of objects as input data (Shi and Pun-Cheng, 2019). Regionalization is a part of spatial clustering, that forms clusters that are adjacent in space. Clusters are therefore adjacent regions that show similar values within and as dissimilar values as possible to other clusters. However, in this thesis regionalization is not applied. (Grekousis, 2020)

3.2.1 Distance Measurements

As mentioned above, clusters are groups of points that are similar to each other and dissimilar to points of other groups. Similarity in spatial clustering is defined by distance measurements, where points within clusters shall have small distances, while they shall have large distance to points in other clusters. There are different distances such as the euclidean - the straight-line - distance 3.2, which is calculated as

$$\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2} \tag{3.2}$$

and manhattan distance 3.3

$$|x_2 - x_1| + |y_2 - y_1| \tag{3.3}$$

which should be applied when measuring distances in real world, for example street networks. A generalization and combination of these two is the Minkowski distance. (Grekousis, 2020)

The distances of one object to the other objects can be kept in a dissimilarity matrix, as seen in 3.2.1. Using these, the distance between two points can be calculated by using different distance measurements and applying them to any additional attributes. As a result distances between objects, in all dimensions are received, if these for example meet any threshold and may be applied to form clusters.

$$D = \begin{pmatrix} \textit{observations} & 1 & 2 & 3 & \cdots \\ 1 & 0 & d(1,2) & d(1,3) & \cdots \\ 2 & d(2,1) & 0 & d(2,3) & \cdots \\ 3 & d(3,1) & d(3,2) & 0 & \cdots \\ \vdots & \vdots & \vdots & \ddots & \end{pmatrix}$$

3.2.2 Clustering Categories

Clustering is an unsupervised technique, meaning that no ground-truth data is provided by the user. Spatial clustering has had many use cases in different topics, such as crime

analysis, mobility or disease pattern detection (Shi and Pun-Cheng, 2019). Clustering can be classified into four groups - hierarchical, partitioning, density-based and grid-based clustering. However, these groups vary and are not deterministic. **Hierarchical** clustering forms clusters by putting the data points in groups in different levels. It either starts from the bottom, with each point belonging to an own cluster or at the top, where one big cluster for all the data is divided into smaller clusters. In the former method - agglomerative hierarchical clustering - the data points are then grouped until a pre-defined point is reached or until all points belong to one group. In the case of **grid-based** clustering the study area is split up into (raster) cells. These raster cells are then used for clustering, which generally has the advantage of faster computation times (Lamb, Downs, and Reader, 2020; Grekousis, 2020; Han, Kamber, and Pei, 2012).

For **partitioning** clustering, the data is split into a certain number of groups. This is mostly performed by distance measurements between the points and a cost function that shall be decreased. One famous example for this group is k-means clustering, and another one is Clustering Geodata Cubes (CGC) - a clustering algorithm used in this thesis. With k-means there are some starting clusters for which the center is defined (mean). The mean of a cluster is calculated by the mean of the spatial features x and y of all the objects in a cluster and can be weighted by other attributes such as population. All points are assigned to a cluster, in the next step the distance of each point to the center is computed. Then all are assigned to the best cluster again and all the steps from above are repeated up to the point where the clusters are not changing between the iterations or a defined number of repetitions is reached. The clusters can be measured by the squared error, which is the distance between all objects in a cluster and its center. (Ku, Nattino, Grootes, Izquierdo-Verdiguier, Girgin, and Zurita-Milla, 2022; Grekousis, 2020; Han, Kamber, and Pei, 2012)

Density-based methods rely on the concept, that clusters enlarge until the density in the cluster reaches some certain point. A high density in general means that a lot of objects are in one area while a low density shows areas where no other objects are nearby. The input parameters of density-based methods are mostly a count of how many features are needed to from a cluster and a radius that specifies the size of neighborhoods. DBSCAN and its extensions to ST data are one example of this clustering group. This category is favourable if not all the data is part of clusters. Table 3.1 shortly summarizes some facts about the two most relevant clustering techniques for this thesis - partitioning and density-based clustering. (Choi and Hong, 2021; Grekousis, 2020; Han, Kamber, and Pei, 2012)

After this short introduction into the topic of (spatial) clustering, the next chapter deals with the time dimension in spatial clustering. Spatio-temporal clustering and especially concepts that are relevant for this thesis are explored.

3.3 Spatio-Temporal Clustering

ST Clustering is often not too different from two-dimensional spatial clustering as the categories and their characteristics from above also apply for ST clustering. Yet, adding

	Partitioning Clustering	**Density based Clustering**
Method	Clustering Geodata Cubes (CGC)	(MDST-)DBSCAN
Forming clusters using	Distances	Density (count of points in neighbourhood)
Input parameter	Often Number of clusters (e.g. at k-means)	Size of neighbourhood (radius), lowest count of objects accepted for a cluster
Advantage	Works well for smaller datasets and used in many tri-clustering (x,y,t + attribute) tasks	Not every point is part of a cluster –> detect noise/outliers and finds clusters of different shapes

Table 3.1: Partitioning vs. Density Based Clustering
(Han, Kamber, and Pei, 2012; Grekousis, 2020)

time to the spatial dimensions makes the data more difficult to handle. An advantage of the temporal component is that new insights in data is offered and hence other patterns and clustering results might be detected. (Shi and Pun-Cheng, 2019)

There are different approaches of how to group ST clustering techniques. However, according to Wu, Cheng, Zurita-Milla, and Song (2020) no common classification of how to group ST clustering methods has been found yet. To state one example, Liu, Deng, Bi, and Yang (2014) for instance classified them into three groups - distance techniques, scan techniques and density techniques. The first group measures the spatio-temporal distance between objects to define clusters. The scan methods, which are not a part of this thesis, use scan statistics respectively scanning windows to discover clusters in the data. Density based methods, such as DBSCAN, are an extension of the spatial clustering technique and try to detect areas according to the count of similar objects in an area. However, the clustering groups from a previous chapter may also be applied to spatio-temporal clustering methods. This thesis leaves out scan techniques and rather focuses on density-based and distance-based clustering techniques.

Another classification of ST clustering methods is grouping them into one-way, co- or tri-clustering methods (Wu, Cheng, Zurita-Milla, and Song, 2020). It has to be noted that co-clustering and tri-clustering algorithms cannot be seen as a synonym for clustering in two respectively three dimensions. None of the analysed papers that applied tri-clustering methods (Wu, Cheng, Zurita-Milla, and Song, 2020; Henriques and Madeira, 2019; Wu, 2022b) compared these to point-based clustering techniques, for instance the density based method DBSCAN. However, as Wu (2022a, p. 2) states, co-clustering techniques "are capable of discovering both spatial and temporal patterns in the data simultaneously", making them a suitable comparison for spatio-temporal clustering techniques, such as ST-DBSCAN or also MDST-DBSCAN. Both these algorithms cannot be considered a co-clustering or tri-clustering method as they work differently to classical tri-clustering algorithms due to different inputs. For instance, MDST-DBSCAN is a point-based clustering method, whereas tri-clustering methods are polygon-based. In the following

lines the clustering algorithms are grouped according to their count of dimensions.

Clustering in one dimension uses either the location or the time as feature and the other one as attribute. Therefore, the output in this case can be either time-clusters or spatial clusters. Traditional clustering techniques are capable to do spatial and temporal clustering separately. Yet, they do not combine these two components (Wu, Cheng, Zurita-Milla, and Song, 2020). Even though, time in general can be added to spatial data either as an attribute or as a dimension, in ST clustering the temporal component has to be implemented as an additional dimension (Shi and Pun-Cheng, 2019). ST clustering techniques have typically relied on existing (spatial) clustering methods and have further developed these to also analyze the temporal characteristic (Birant and Kut, 2007; Ku, Nattino, Grootes, Izquierdo-Verdiguier, Girgin, and Zurita-Milla, 2022; Choi and Hong, 2021; Oliveira, Santos, and Pires, 2013).

Clustering in two dimensions examines both the temporal and the spatial dimension identically. In contrary to one-way clustering both components are used as a dimension and clusters are generated based on homogeneous areas in both components. Birant and Kut (2007) already introduced ST-DBSCAN, which is a density-based ST clustering method, in 2007. Co-clustering of ST data and clustering ST data cannot be seen as a synonym in research. For instance, Wu, Zurita-Milla, and Kraak (2015) didn't consider ST-DBSCAN, as it is not a co-clustering algorithm according to their definition. Furthermore, this difference arises from the distinct underlying data types.

ST-DBSCAN shows an example to see the difference to one-way clustering algorithms. In contrary to the traditional DBSCAN algorithm, that uses only one distance measure for the spatial component, the ST-DBSCAN technique applies two distance measures - one for the spatial and one for the temporal component. In general, the euclidean distance for spatio-temporal clustering 3.4 can be calculated as

$$\sqrt{(x_2 - x_1) + (y_2 - y_1) + (t_2 - t_1)} \tag{3.4}$$

where space and time of two points are given and the distance in these dimensions is calculated (Fanaee-T, 2012). Wu, Cheng, Zurita-Milla, and Song (2020) stated that until the release of this paper there were no hierarchical co-clustering algorithms applied to ST data.

Co-clustering algorithms generally were introduced in the 1970s - not for spatio-temporal tasks but for other domains such as bioinformatics. According to Wu, Cheng, Zurita-Milla, and Song (2020) the first algorithm that used co-clustering for ST data was introduced by Wu, Zurita-Milla, and Kraak (2015), which was later used in other studies. The algorithm employed in this study was the Bregman block average co-clustering algorithm with I-divergence (BBAC I), which is a partitioning clustering technique.

3.3.1 Clustering Spatio-temporal data with an additional attribute

The following lines analyze clustering techniques that, in addition to the two dimensions mentioned above, apply a third dimension and group the data based on all three dimensions. In this context, dimensions are considered as the parameters used by the clustering

methods to group the data, rather than attributes of the data. Therefore, in this case, clustering methods that detect similarities in space, time, and values and cluster the objects accordingly are examined. The datasets used for these clustering algorithms are defined by objects, their attributes and the objects' connections, which can be temporal. (Henriques and Madeira, 2019). The two methods that are analysed in this work belong to this group. The result of tri-clustering, that was first introduced in 2005, are clusters that are homogeneous in all three dimensions. Same as with co-clustering methods, research in tri-clustering algorithms has mainly focused on non-spatial fields (Wu, Cheng, Zurita-Milla, and Song, 2020). One example is clustering documents according to word-document matrices to find similar documents, as was performed by Mallela, Modha, and Dhillon (2003). This approach achieved better results than one-dimensional clustering methods.

Oliveira, Santos, and Pires (2013) stated that, at this time, there were no ST clustering methods that used space, time and an additional attribute. Hence, they developed an extension of the density-based shared nearest neighbor clustering algorithm that tackles this problem. This was one advantage to the, at that time already existing, ST-DBSCAN (Birant and Kut, 2007) and STSNN (Liu, Deng, Bi, and Yang, 2014), that didn't include an additional non-spatial attribute. Another advantage of the distance function of the 4D-SNN algorithm is the unified integration of all the analysed dimensions. Even though there are hierarchical (e.g. Lamb, Downs, and Reader (2020) and density-based tri-clustering algorithms (e.g. Choi and Hong (2021)) for point-based clustering, most of the polygon-based tri-clustering algorithms have been partitioning clustering methods, according to Wu, Cheng, Zurita-Milla, and Song (2020). Wu, Zurita-Milla, Izquierdo Verdiguier, and Kraak (2018) further developed their co-clustering algorithm, that was mentioned above, to create a tri-clustering algorithm. This algorithm, named Bregman cuboid average tri-clustering algorithm with I-divergence, was then further evolved by Ku, Nattino, Grootes, Izquierdo-Verdiguier, Girgin, and Zurita-Milla (2022) and is analysed in this thesis. Ku, Nattino, Grootes, Izquierdo-Verdiguier, Girgin, and Zurita-Milla (2022) proposed a tri-clustering extension of the co-clustering algorithm of Wu, Zurita-Milla, and Kraak (2015). Hierarchical clustering of ST data hasn't played a great role yet. Lamb, Downs, and Reader (2020) proposed a hierarchical clustering algorithm that was applied to ST data with an additional attribute.

In summary, the traditional one-way-clustering methods have been developed to co-clustering algorithms and these again have been further evolved to make them useful for ST data with an additional attribute. Wu, Zurita-Milla, Izquierdo Verdiguier, and Kraak (2018) mentioned that even though tri-clustering algorithms have the disadvantage of longer computation times and need more input parameters, they outperform one-way-clustering and co-clustering methods in regard to detecting new insights in the data. Many clustering techniques have first been proposed in other fields but in the last years several of these co- and tri-clustering methods have been applied to ST data. The classification of ST clustering techniques is not always into the groups one-way-clustering, co-clustering and tri-clustering. For instance, Shi and Pun-Cheng (2019) presented a survey of ST clustering methods but didn't use these classes. However, to identify the necessary/best algorithm for one's data set such a classification is helpful and that is the reason why

this division is used in this thesis. It is also necessary to be aware that there a great differences between polygon-based tri-clustering methods, that cluster the data according to the dimensions simultaneously, and point-based clustering techniques that use distance measures such as euclidean.

Research on spatio-temporal clustering methods with additional attributes showed that there is no term that summarizes all these techniques. Furthermore, polygon-based tri-clustering methods, such as BCAT, and point-based methods such as MDST-DBSCAN have not been compared or even mentioned in the same papers. Hence, further research in this topic and a terminology is needed to detect the best clustering techniques for the respective application. Until now two different research areas have developed but researchers would profit from bringing these two together. Table 3.2 provides an overview on different clustering techniques that have been used for ST data and an additional attribute. Still, this is not a complete overview of these methods.

Method	**Clustering Technique**	**Notes**
4DSNN (Oliveira, Santos, and Pires, 2013)	Point-Density-Based	First algorithm for ST data with an additional attribute (burnt area)
Bregman Cuboid Average Tri-Clustering Algorithm with I-Divergence (Wu, Zurita-Milla, Izquierdo Verdiguier, and Kraak, 2018)	Partitioning Tri-Clustering	Polygon-based algorithm used for Georeferenced Time-Series
MDST-DBSCAN (Choi and Hong, 2021)	Point-Density-Based	Extension of (ST) DBSCAN algorithm; result highly dependent on parameter selection –> finding the best parameters is challenging
Clustering Geodata Cubes CGC (Ku, Nattino, Grootes, Izquierdo-Verdiguier, Girgin, and Zurita-Milla, 2022)	Partitional Tri-Clustering	Advancement of Bregman cuboid average tri-clustering algorithm with I-divergence and newest analysed method (introduced 2022) –> few experiences

Table 3.2: Selection of clustering methods on ST data with an additional attribute (Oliveira, Santos, and Pires, 2013; Wu, Zurita-Milla, Izquierdo Verdiguier, and Kraak, 2018; Choi and Hong, 2021; Ku, Nattino, Grootes, Izquierdo-Verdiguier, Girgin, and Zurita-Milla, 2022)

3.4 Metrics

Due to the comparison of two different clustering methods, comparison of the outputs in a last step is required. The result of different clustering algorithms varies, especially when

comparing clustering algorithms from two different groups - partitioning vs. density-based respectively polygon-based vs. density-based. Agrawal, Garg, Sharma, and Patel (2016) specified criterions ST data clustering should meet. These are explored in Chapter 4 and Chapter 8 more deeply but these requirements shall help, when comparing the outputs.

Henriques and Madeira (2019) stated some questions that shall be considered when addressing tri-clustering tasks. These include the efficiency of the algorithms as they are rather complex and norms about how to measure the coherence of the clusters, have to be defined. This consistency should be met in all dimensions and none of the analyzed dimensions shall be supreme. Furthermore, the output clusters have to be statistically significant and should not rely too much on the input parameters. The latter point was also mentioned by Agrawal, Garg, Sharma, and Patel (2016), as they state that the input parameters are very sensitive and the result of many clustering techniques, such as Choi and Hong (2021) MDST-DBSCAN or K-Means, highly depend on them. Hence, optimizing these were essential in the empirical part. One method to tackle this problem would be the elbow method, that can be applied to K-Means. This method relies on the concept of comparing the Sum of the squared errors (SSE) of the euclidean distances within clusters, where different input parameters (k) are used and k with the lowest SSE is chosen (Raschka and Mirjalili, 2017). However, in the case of spatio-temporal clustering other optimization tools, that is explored later, are needed. The mentioned challenges apply to all tri-clustering algorithms and shall therefore also be considered when doing tri-clustering with ST data and are explored more deeply below, when analyzing the two main algorithms of this thesis.

So according to Henriques and Madeira (2019) in addition to other parameters, coherence and significance show the quality of the (tri-)clustering results. To analyze the output of tri-clustering methods, metrics are recommended. However, there is no mutual metric for all spatio-temporal clustering techniques that tells how coherent the clusters are. In addition to this, no real values exist that show how well the clustering outputs reflect the real world. External indices such as Rand-statistics could be used when ground-truth data would be available (Agrawal, Garg, and Patel, 2015). Hence, in this thesis internal indices are required.

According to Henriques and Madeira (2019) the performance of tri-clustering algorithms can be explored through different measurements.

- **Accuracy-based views**: To measure, how well an algorithm performs in general by putting objects, that form a hidden cluster into fabricated data. The advantage here is that ground-truth data - the hidden cluster - is available. Different metrics can be applied to all dimensions to for instance identifying the hidden triclusters.
- **Homogeneity views**: To measure the homogeneity-level of tri-clustering outputs. It is proposed to use more than one function to get a holistic view.
- **Statistical significance views**: Likelihood that a tricluster is random shall be measured. Different metrics are available to do this.
- **Domain significance views**: Refers to comparing the output to preliminary

knowledge.

- **Complementary performance**: To get an united view combining the mentioned views should be considered.

3.4.1 Internal Metrics

In the field of spatio-temporal clustering many validation indices, especially internal validation indices, have already been analysed. However, many metrics for partitioning and hierarchical clustering techniques and less for dense and arbitrary shaped clusters have been studied. DBSCAN and hence, MDST-DBSCAN are density-based methods that results in randomly formed clusters (Agrawal, Garg, Sharma, and Patel, 2016). Therefore, a validation index that takes this into account is necessary. For instance, Silhouette - a popular, well performing validation index (Hämäläinen, Jauhiainen, and Kärkkäinen, 2017) - cannot be applied as it does not work with randomly formed clusters. Furthermore, the data does not include real cluster values and therefore no ground-truth data is available. This also limits the number of possible metrics. For instance, (Lamb, Downs, and Reader, 2020) used a homogeneity score and a random index score that compared the output of simulations to the ground truth data.

Some validation indices that are useful for this specific study are presented in the next lines (Agrawal, Garg, Sharma, and Patel, 2016). As explained above it is necessary that the indices work on randomly formed clusters, which the following three metrics should fulfill.

- **Ball-Hall**: The average spread of objects in the clusters tells how well the clustering algorithm works. (Agrawal, Garg, Sharma, and Patel, 2016)

- **Dunn**: Is the relation of the smallest distance between the clusters and the highest distance of objects of one cluster. Hence, the distance between the clusters is compared to the size of clusters. The higher the value, the better the quality of the clusters. (You, 2022; Arbelaitz, Gurrutxaga, Muguerza, Pérez, and Perona, 2013)

- **Davies-Bouldin**: Tells how similar the objects of a cluster are and how different they are from others. It measures the distances of objects to the center of the according cluster and the space of the centers of two clusters. It is often used for evaluating clustering results. (You, 2022; Arbelaitz, Gurrutxaga, Muguerza, Pérez, and Perona, 2013)

These three mentioned validation indices are considered when comparing the clustering results. The literature review showed that there is not one validation metric that outperforms all the other. Hence, all of the metrics might be used to compare the outputs. Furthermore, as Henriques and Madeira (2019) proposed, also other validations shall be applied. The domain significance view from above also plays a role in the evaluation of the clustering results. In Chapter 4.2 the real estate market of Vienna is analysed, which leads to a better understanding of current knowledge about real estate markets in Vienna.

3.5 Clustering Geodata Cubes (CGC)

The next section covers Clustering Geodata Cubes (CGC). The concepts from above shall help understanding this clustering method. CGC is a Python package that was created by Ku, Nattino, Grootes, Izquierdo-Verdiguier, Girgin, and Zurita-Milla (2022). It offers co- and tri-clustering functions to cluster data according to space, time and additional attributes. The partitioning co-clustering algorithm applies the Bregman block average co-clustering algorithm with I-divergence (BBAC I), while the tri-clustering algorithm applies the Bregman cuboid average triclustering algorithm with I-Divergence (BCAT). (Wu, Zurita-Milla, Izquierdo Verdiguier, and Kraak, 2018).

3.5.1 CGC algorithm and input parameters

Even though co-clustering was utilized in Chapter 5.1, the following paragraph describes the tri-clustering method to provide additional insights into why co-clustering was preferred over tri-clustering. The BBAC I algorithm is presented in Figure 3.3. The figure shows A) what the input data looks like, with rows showing locations, columns the years and depth the days of a year - in this specific case. Therefore, one cube represents the additional non-spatial attribute, in this case temperature, at a specific location. In this thesis the cube has to be slightly different, as, in contrast to temperature, the transaction prices do not depend on the seasons and hence not on the time of the year. Transaction prices are, as mentioned above, linear data and not periodical data, which means that days of different years shouldn't be neighboring. Hence, co-clustering might be the better method compared to tri-clustering, if this additional dimension cannot show any deliver further insights in the data. B) shows the output of regular tri-clusters and C) the output of irregular tri-clusters. The clusters of B are created by joining the location and time clusters of temperature values. The structure of the clusters is improved by the I-divergence metric (Mallela, Modha, and Dhillon, 2003) which has outperformed other metrics such as the Euclidean distance. The objective of this algorithm is to decrease the loss of information

$$loss = Information(S; Y; D) - Information(S1; Y1; D1)$$

from the initial data cube to the tri-clustering output. The information refers to common information of different attributes. In the case of this thesis information is the transaction prices in a certain location, at a certain time. The algorithm works by first setting starting clusters of the days, years and locations and then combining these. In a next step, the information dissimilarity is calculated and then the output is improved by using a loss or divergence function, as seen above (Wu, Zurita-Milla, Izquierdo Verdiguier, and Kraak, 2018; Wu, Cheng, Zurita-Milla, and Song, 2020).

CGC offers the BBAC and BCAT algorithm for co- and tri-clustering. Hence, the algorithms work similar as described above. CGC uses several iterations to reduce the impact of the allocation of the objects in the starting cluster. Figure 3.3 shows two different clustering results - regular and the irregular triclusters. K-means, that can be used to fine-tune the regular triclusters to irregular tri-clusters, uses different numbers of

clusters, where the number with the best silhouette score is chosen (Ku, Nattino, Grootes, Izquierdo-Verdiguier, Girgin, and Zurita-Milla, 2022). As input to CGC, the number of clusters in the temporal, spatial and additional attribute's dimension have to be provided. The input parameters are analysed and optimized in detail in section 5.1. (Ku, Nattino, Grootes, Izquierdo-Verdiguier, Girgin, and Zurita-Milla, 2022)

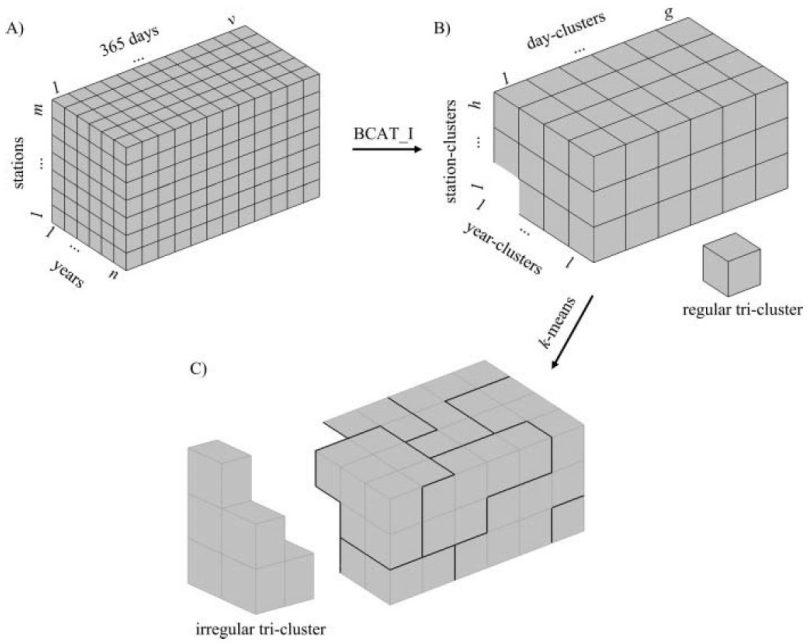


Figure 3.3: BCAT cube: A) data cube, B) regular tri-clusters C) irregular tri-clusters (Wu, Zurita-Milla, Izquierdo Verdiguier, and Kraak, 2018, p.73)

3.5.2 CGC/BCAT in research

CGC has not been analysed by any studies yet, except of Ku, Nattino, Grootes, Izquierdo-Verdiguier, Girgin, and Zurita-Milla (2022), that introduced the package. Wu, Cheng, Zurita-Milla, and Song (2020) compared BCAT to the respective co-clustering algorithm BBAC and to k-means clustering. Even though, BCAT had a longer running time and more input parameters had to be specified, it outperformed the other two in gaining insights in the data. Insights in the data were defined by questions about the topic of air pollution and the authors identified how many were answered by the different algorithms. This is, in addition to the metrics from above, another approach to evaluating clustering results.

The mentioned studies used georeferenced time-series data. Hence, the transaction prices in this thesis need transformation to be suitable for the BCAT algorithm. This showed to be one major challenge in the empirical part, to create polygons that are needed as input parameters. Challenges that could come along with this creation could be the

modified areal unit problem (MAUP) and the loss of information.

3.6 MDST-DBSCAN

Multidimensional spatio-temporal DBSCAN is an extension of DBSCAN respectively ST-DBSCAN that was proposed by Choi and Hong (2021). The density-based clustering algorithm was introduced to overcome the problem of not being able to analyze an additional attribute to the spatio-temporal dimensions. One advantage of DBSCAN to partitioning and hierarchical clustering algorithms is that not every object is part of a cluster, as these objects can also be noise (Raschka and Mirjalili, 2017).

3.6.1 MDST-DBSCAN algorithm

In the following lines, the input parameters to MDST-DBSCAN are explored. In addition to epsilon, which is the radius around a core object, regarded as neighborhood, the lowest number of objects (minpts) to form a cluster is compulsory as input parameter for the DBSCAN algorithm. As one can see in 3.4 there are core points, border points and noise points. The first one applies to objects where minpts is met within the epsilon-radius. Border points are within the epsilon radius but are no core points as their neighboring objects are not enough to meet the minpts-requirement. Noise are objects that are not part of clusters. The algorithm starts with any object in the data and sums up all the neighboring objects that are nearer to itself than epsilon. The distance can be calculated by any distance function such as euclidean, the straight-line distance, or manhattan (Ester, Kriegel, Sander, and Xu, 1996). When the counted objects exceed a higher value than minpts, the object is defined as a core point and a cluster is formed. This is done several times up to the point where all objects were observed and hence, clusters and noise are discovered. ST-DBSCAN, the spatio-temporal extension of DBSCAN, requires another epsilon, which defines the temporal divergence accepted. The algorithm works similar to the steps described above, with the only difference of the additional time-epsilon (Choi and Hong, 2021; Raschka and Mirjalili, 2017).

The MDST-DBSCAN likewise works similarly, with the extension of additional attributes. The user therefore has to provide, in addition to the spatial and temporal epsilon, other threshold values. A dataset containing for instance, besides time and space, two attributes has four epsilons that are defined by the user. These are used to discover if clusters exist. Either all or a certain number of these thresholds must be satisfied for clusters to be created. 3.5 visualizes this; in the first case the neighboring Point B meets all threshold values, while in the second case only one threshold is met. Choi and Hong (2021) apply K-Nearest Neighbor (KNN) to define the epsilon. KNN tells the distances of a point to k nearest neighbors, where k is the number of neighboring points considered (Jacquez, 1996). When plotting the distances usually a curvature point shows the best epsilon value.

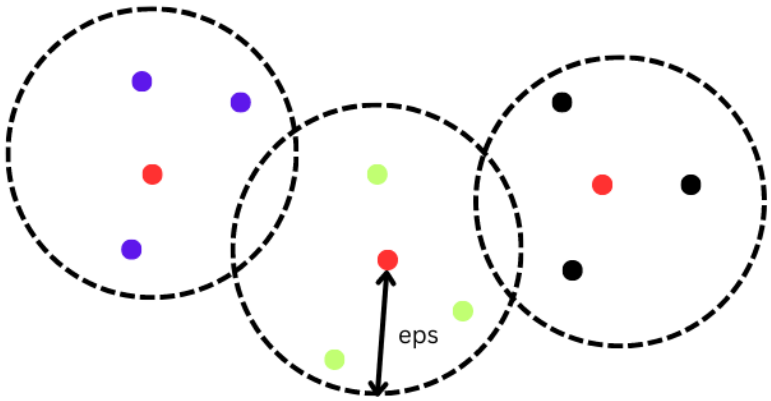


Figure 3.4: DBSCAN (own representation; based on (Raschka and Mirjalili, 2017, p.373))

3.6.2 MDST-DBSCAN in research

In general, for MDST-DBSCAN computation time might be a problem, especially with an increasing number of attributes. Hence, this problem might not be as big in this thesis, as only transaction prices and no other non-spatial/non-temporal attributes are clustered. Workarounds of this challenge include that the distances of the analysed object only and no other ones have to be calculated and if an object exceeds on epsilon value it doesn't have to be compared with the analysed object in the other dimensions. Huang, Qi, Zheng, Zhu, and Shen (2023) introduced a new measure to find the most advantageous compression to reduce the computation costs but to keep the quality.

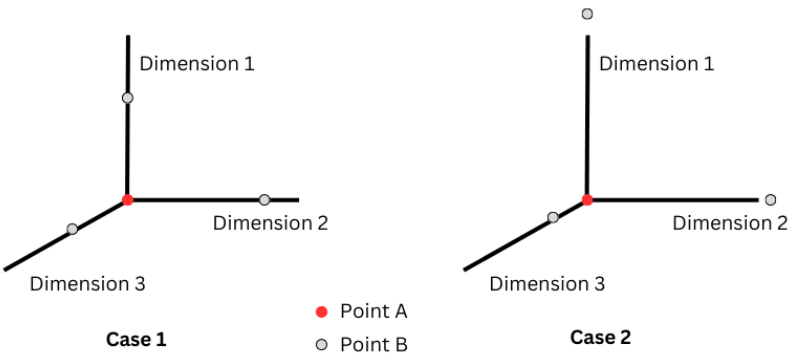


Figure 3.5: Detecting Neighboring Objects in three Dimensions (own representation; based Choi and Hong (2021, p.4))

According to Choi and Hong (2021) there are two other major challenges when applying MDST-DBSCAN. One is that the results are highly dependent on the first object that is chosen when performing the clustering. The other challenge concerns input parameters, as the outcome highly depends on them. Even though the visualization of KNN delivers good results, it is proposed to use different non-visual metrics to decide on the input parameters. There have already been approaches, that automated the parameter search (Zhang, Zhou, Guo, Qi, and Abusorrah, 2022). This fine-tuning of the input parameters was performed in the empirical part. Choi and Hong (2021) likewise to this thesis used transaction prices to test the parameter.

3.7 Summary and Findings

The research on ST data has shown that there is no specific term and no group for ST data with an additional numerical attribute and flexible location. Introducing a term or group for this datatype could lead to a better understanding and more appropriate ST data mining methods. Hence, **Multidimensional Spatio-Temporal Event Data** is proposed as a novel term to summarize this data type.

A similar problem arises when analysing clustering techniques that use this datatype. There is no group that combines these ST clustering methods. Moreover, ST polgyon-based tri-clustering methods, such as CGC, and point-based methods, such as MDST-DBSCAN, have never been compared. A reason for this is the lack of classification of these methods into one group, even though they can both be applied to Multidimensional Spatio-Temporal Event Data. That is why the term **Multidimensional Spatio-Temporal Event Clustering** is introduced to assign them to the same classification of clustering techniques. This thesis shall be a first try to compare these two clustering groups. However, further research is required, to improve the field of STDM of the mentioned datatype and more specifically to facilitate the selection of the best clustering technique. The lack of comparisons between these two groups is a reason, why different evaluation methods were used to compare the outputs of the clustering methods.

Both - CGC and MDST-DBSCAN - can cluster ST data with an additional attribute. Still, there are many differences between these two tri-clustering methods. Above others, CGC is a partitioning polgyon-based clustering technique, while MDST-DBSCAN is a point-density-based technique. This means that both methods have different starting conditions and have all the advantages of their associated clustering group. Nonetheless, these techniques are analysed and compared in more detail in the empirical part of this thesis.

4 Methodology & ESTDA

After analysing the concepts of spatio-temporal data mining and real estate submarkets, the following chapters will cover the empirical part of this thesis. This part begins with collecting the data of the Real Estate Database Exploreal and performing necessary data cleaning. This data is then explored in more detail by spatio-temporal methods. Prior to clustering, the data for CGC required pre-processing, which is explained in 5.1. Following this the methods CGC, MDST-DBSCAN were executed, as displayed in Figure 4.1. However, MDST-DBSCAN required adaption and a third method was developed, to obtain better submarket areas. The reasons for this are explained in the according Chapter 5.2.

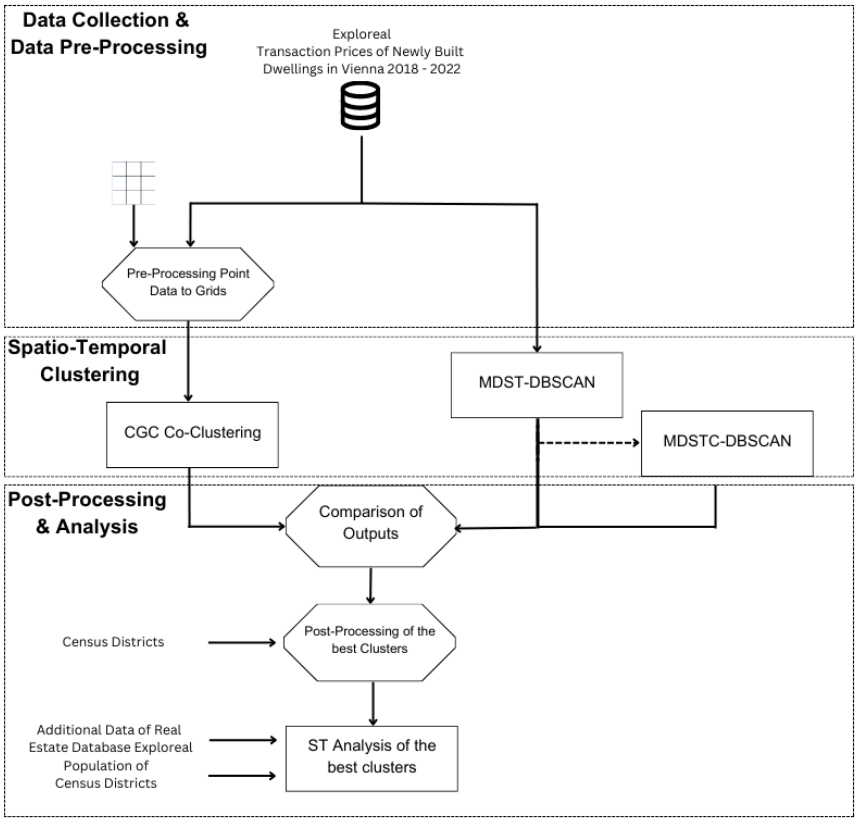


Figure 4.1: Flowchart of Empirical Part

4.1 Softwares

Python was used to perform most of the computations in this thesis. However, GeoDa was also applied to create differentiated Moran's I and to cross-check and compare the results generated with Python in the data exploration part.

4.1.1 Python

Python was the primary tool for data exploration, clustering methods and the comparison of the outputs. It is a main open source programming language that is specifically also useful for data science tasks (Raschka and Mirjalili, 2017). Many packages, such as **NumPy** and **Pandas**, allow to perform data manipulation and analysis. For geo-related tasks packages such as **geopandas**, **pysal** and **shapely** were utilized.

4.1.2 GeoDa

GeoDa is an open-source software designed for spatial data analysis (Anselin, Syabri, and Kho, 2006), which was used in this thesis to control the Moran's I values produced with Python and explore the data. Additionally, the software was used to create the graph of differentiated Moran's I. Even though GeoDa offers many functions, including creating spatial weights, measuring spatial autocorrelation, plotting, and clustering, Python was the preferred software for data exploration in this thesis due to its ability to automate tasks.

4.1.3 GitHub

The code produced in this thesis, including the Python implementations of MDST-DBSCAN and MDSTC-DBSCAN, is available on the open-source version control platform **GitHub**. This platform allows users to collaborate and share code with others. The code is accessible to everyone and can be used for further studies or be improved, if needed.

4.2 Study Area - Real Estate Market in Vienna

Vienna has experienced a great increase of population in recent years leading to pressure on the housing market. To counteract this challenge, Vienna has seen a high construction rate in the past years, with 8,76 dwelling units/1.000 inhabitants, being much higher than 4,84 in total Austria for the years 2020 - 2022 (Exploreal and WKO, 2022). However, according to Nationalbank (2023), in 2022 in Vienna there was still a shortage of 30 000 new buildings to meet demand.

In their analysis of housing construction in Vienna between 2018 and 2021 Plank, Schneider, and Kadi (2022) found out that most of the new construction projects were concentrated in districts in areas outside of the "Gürtel", such as the 21st, 22nd and 23rd district. However, between 2020 and 2022, high construction rates were observed in the 2nd (10.23), 3rd (13.34) and 14th (10.96) districts, additionally to the mentioned

districts. The construction rate is defined as the number of built dwelling units per 1.000 inhabitants per year (Exploreal and WKO, 2022).

Despite the comparatively high construction rates in Vienna in recent years, a significant increase of housing prices could be observed, as displayed in Figure 4.2. This trend is due to various reasons, including regionally larger phenomena, such as the COVID-19 pandemic or the war in Ukraine, that are relevant for many European countries. Other factors are rising material costs, leading to higher construction prices, rising land costs and interest rates. However, these factors are interconnected and can influence each other, leading to further price increases (Nationalbank, 2023).

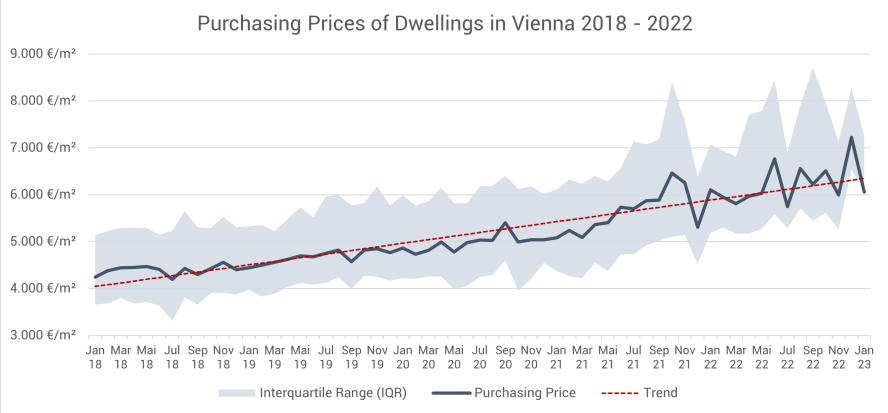


Figure 4.2: Purchasing Prices of Dwellings in Vienna 2018 - 2022 (Data: Exploreal)

Spatial autocorrelation of real estate prices was detected by Herath and Maier (2013). This indicates that submarkets of transaction prices do exist. However, this will be analysed in more depth for this specific dataset in chapter 4.3.1. There is no further research on real estate submarkets or transaction price submarkets in Vienna that is relevant for this thesis.

4.3 Data

As mentioned in chapter 1, the data for this research was extracted from the real estate database Exploreal. Only data of newly built dwellings was collected, meaning no prices of other forms of housing, such as houses, was included. Furthermore, dwellings in loft extensions or redevelopments, data on second occupancy of apartments and unusual prices for the market were excluded, to maintain data homogeneity. Unusual prices can occur for instance if dwellings are sold within company constructs and are highlighted as such by Exploreal. These were excluded, as these could distort the clustering outputs. Moreover, the data indicated that for redevelopment-projects it was difficult to distinguish between those flats that were redeveloped and those that were not. Therefore, these redevelopment-projects were excluded in total. Hence, only new construction project were included in the data.

The transaction prices used in the analysis are transaction prices per square meter. The data on prices and living spaces was obtained from purchase contracts and stored in a relational database. The usable living space does include loggia areas, but no other open spaces such as terraces, balconies or garden areas (§ 2 Abs. 7 WEG 2002). It should be noted, that no further post-processing of the data was carried out, as the methods are expected to be robust enough to handle outliers.

4.3.1 Data Exploration

The data set used in this thesis covers all transaction prices of newly built dwellings in Vienna from 2018 – 2022. Each row in the data contains five attributes, which are listed in Table 4.1. The table displays the first 5 rows of the dataset. The "id" attribute is the unique identifier of each dwelling, the column "date" the transaction date, "Geo Lat" and "Geo Long" indicate the coordinates in WGS84 format and "m2price" indicates the transaction price per m^2. It is important to note that dwellings with the same coordinates, such as those with ID 14656 and ID 14626, are from the same project. In total **19,476** dwellings are included in the data set. The average project during the investigation period contained 25.66 apartments. The smallest projects contain five dwellings, while in the largest one 402 dwellings were built. Five dwellings is the minimum number of dwellings, that Exploreal captures.

ID	**Date**	**Geo Lat**	**Geo Long**	**m2price**
14630	2019-04-11	48.214194	16.330795	3955.34
14682	2019-04-17	48.211914	16.221002	5533.12
14656	2018-11-26	48.225875	16.504206	3493.01
14626	2018-10-05	48.225875	16.504206	3694.99
14319	2019-05-15	48.196005	16.291935	4060.01

Table 4.1: First five rows of data

Table 4.2 presents transaction price values (€/m^2) and the number of purchase contracts for each district, both in total and per 1,000 inhabitants. This information is relevant to get insights into how the values are distributed, as this can impact the clustering results. It must be noted, that the number per 1,000 inhabitants is the number of contracts for the investigation period of five years. This shall just be seen as an indicator for the distribution of the data points but these numbers cannot be compared to other publications or studies. As shown in the table there is a significant difference of purchasing prices between the districts, ranging from a median of 3,989.9 €/m^2 in the 23rd district to 12,762.56 €/m^2 in the first district. The number of purchase contracts shows where most of the data is available. The 22nd district has the highest number of contracts per 1,000 inhabitants, followed by the 14th, 23rd, and 3rd districts. However, there is very limited data in the 8th, 9th and 5th districts with 22, 14 and 10 contracts, respectively. As explored in Chapter 4.2, there are very few new construction projects in the districts located in the city center. This table therefore on the one hand highlights that the median prices of

the districts vary. On the other hand, the downsides of the data are also visible, as the inner districts such as the mentioned districts lack of data. Some districts do not show any data for some years, which must be taken into consideration in the cluster analysis. However, see chapter 5.2 and chapter 6.3 for further analysis on this topic.

District	**Median Transaction Price (€/m²)**	**Count of Purchase Contracts**	**Number of Purchase Contracts/1,000 inhabitants**
01. Bezirk, Innere Stadt	12762.56	137	8.24
02. Bezirk, Leopoldstadt	5192.18	293	2.71
03. Bezirk, Landstraße	5604.22	1572	16.24
04. Bezirk, Wieden	6215.06	117	3.48
05. Bezirk, Margareten	5968.31	10	0.18
06. Bezirk, Mariahilf	6430.93	55	1.75
07. Bezirk, Neubau	7165.54	142	4.49
08. Bezirk, Josefstadt	8478.64	22	0.89
09. Bezirk, Alsergrund	6415.49	14	0.33
10. Bezirk, Favoriten	4177.55	2022	9.26
11. Bezirk, Simmering	4475.34	803	7.37
12. Bezirk, Meidling	4883.88	1055	10.52
13. Bezirk, Hietzing	5909.65	426	7.67
14. Bezirk, Penzing	5281.03	1640	16.94
15. Bezirk, Rudolfsheim-Fünfhaus	4402.67	338	4.44
16. Bezirk, Ottakring	5033.19	823	8.03
17. Bezirk, Hernals	5710.33	297	5.3
18. Bezirk, Währing	6694.86	269	5.22
19. Bezirk, Döbling	7542.06	633	8.38
20. Bezirk, Brigittenau	4920.79	240	2.8
21. Bezirk, Floridsdorf	4302.16	2082	11.32
22. Bezirk, Donaustadt	4658.84	4542	21.35
23. Bezirk, Liesing	3989.9	1945	16.49

Table 4.2: Transaction prices (€/m²) and Number of Purchase Contracts 2018 - 2022

Figure 4.3 provides a visual presentation of the distribution of transaction prices per square meter of the projects. Only one price per project is visible. The map shows similar characteristics to Table 4.2, as it shows higher prices in the center area and in the 18th and 19th district. Lower prices are detected in the outskirts, especially in the east and the south of Vienna. The range of prices is wide, ranging from a minimum of 1,145.34 €/m² to a maximum of 71,553.18 €/m². However, it is important to note that the maximum mean price of a project is 15,983.82 €/m², while the minimum is 1,864.11 €/m². These

values indicate the great range of m²-prices. Additionally, this analysis provides more detailed information than the district-level data in Table 4.2. As the MDST-DBSCAN is a point-based method that incorporates all data points, this analysis of maximum and minimum values is necessary. The following analysis below provides further information on the data set.

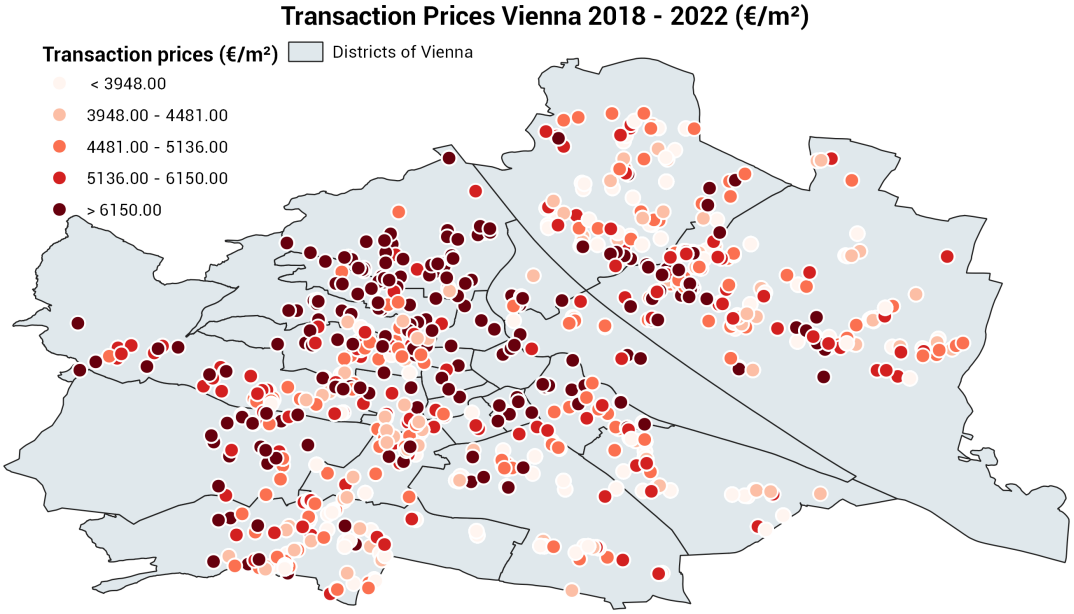
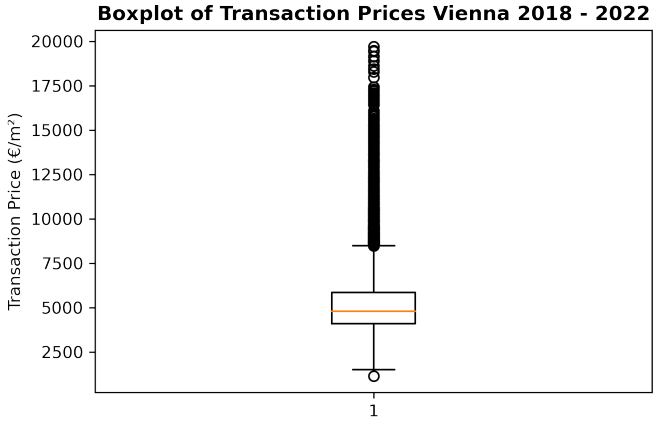


Figure 4.3: Transaction prices of newly built dwellings Vienna 2018 - 2022

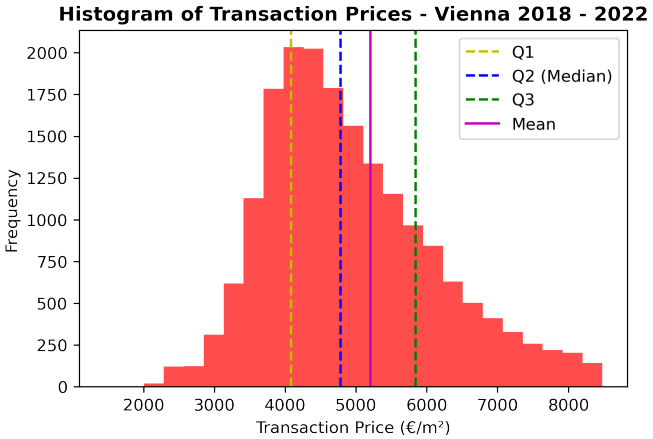
As depicted in Subfigure 4.4a there is a notable amount of outliers exceeding the maximum (Q3 - 1,5 * IQR), with a count of 967. It is important to note, that the boxplot only displays values up to 20,000 €/m². However, it has been previously mentioned, that transaction prices up to 71,553 €/m² exist. There is only one outlier value below the lower limit.

Subfigure4.4b presents the count of values within the range of the lower (1,443.06 €/m²) and upper limit (8,481.68 €/m²) of the boxplot, excluding the outliers. The histogram reveals a right-skewed distribution of values, with most values falling in the bin of the first quartil Q1. Due to the significant number of outliers above the upper limit, the mean (5,204.44 €/m²) is higher than the median (4,784.06 €/m²). This right-skewed distribution indicates that there may be a small number of areas with substantially higher prices. On the other hand, many areas with values around Q1 range (4,082.54 €/m²) might exist. Hence, even though the interquartile range (IQR) is below 2,000 €/m², many values that are outside the IQR indicate variations in prices between the areas, as also observed in Figure 4.3. The high count of values below the mean highlights on the one hand relatively cheaper areas, whereas on the other hand a few dwellings or areas with

very high prices exist. One potential reason for these high prices could be the inclusion of dwellings with big open space, such as terraces that are not included in the living space. However, this aspect will be further explored in the analysis of spatial statistics.



(a) Boxplot of Square Meter Prices (Upper Limit: €20,000/m²)



(b) Histogram of square meter prices

Figure 4.4: Distribution of Transaction Prices (€/m²) Vienna

The outliers were not eliminated for the clustering, as these might represent areas with significantly high prices. The next section will cover spatial and spatio-temporal analysis of the data, to gain insights into the spatial and spatio-temporal trends of the dataset.

4.3.2 Exploratory Spatio-Temporal Data Analysis

Figure 4.5 displays the spatio-temporal distribution of the transaction prices for the investigation period 2018 - 2022, with darker values indicating higher prices and lighter colors indicating lower prices. Spatially, a similar distribution as in Figure 4.3 can be

detected. As was already mentioned in chapter 4.2 prices have risen sharply since 2018 and this trend is also visible in Figure 4.5. However, this graph only shows very limited insight in the data. Subsequent analysis and graphs inspect these patterns more deeply.

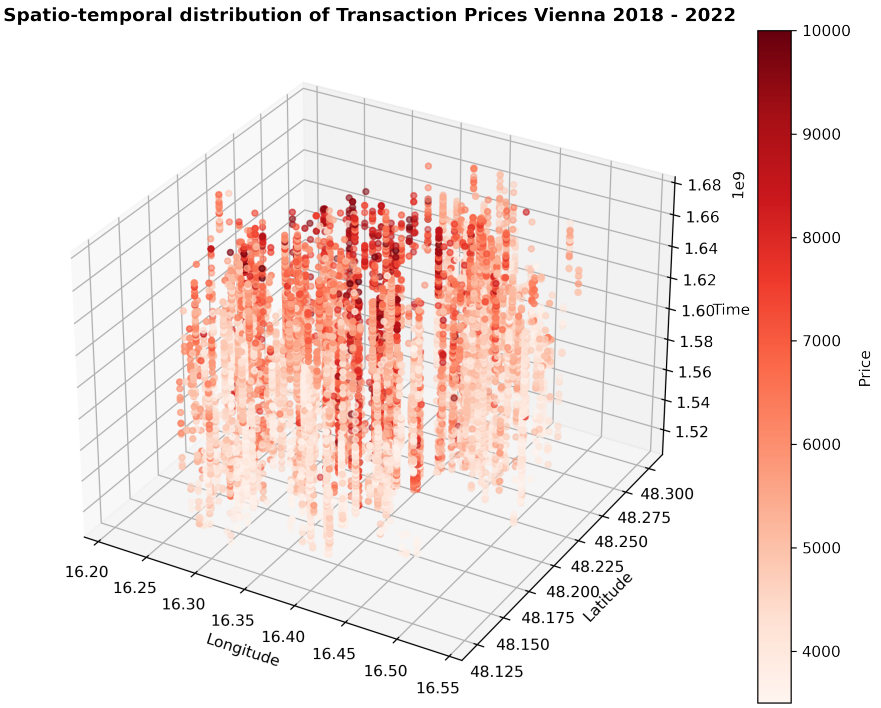


Figure 4.5: Transaction prices of newly built dwellings Vienna 2018 - 2022

Comparing the change in median prices between 2018 and 2022 reveals differences between districts. For instance, the 14th district is only showing a rise of 21 % of median values between 2018 and 2022, while the transaction prices in the 22nd district increased by 56 %. This illustrates that in addition to differences in transaction prices, there are also significant differences between the rates of price change across districts. These two districts were chosen for comparison because they both have a large amount of data (see table 4.2).

Moran's I measures the degree to what extent the values of data points are clustered (positive value) or dispersed (negative value), indicating the degree of autocorrelation, as was discussed in section 3.1.1. KNN was used to create weights, as this method is suitable for point data. Additionally, since KNN takes the k nearest neighbor of a point, it has the advantage to distance-based-methods of ensuring that all objects have neighbors and there are no isolated values (Anselin,Luc, 2020). For this specific dataset, pre-processing was necessary, to be able to calculate Moran's I. Moran's I could not be calculated using KNN for datasets with coordinate duplicates. Several dwellings of one project can have the same coordinates in the data. That is why noise of 0.0001 was added to the coordinates. The analysis showed that when using KNN (k = 40) to define the

spatial weights, the autocorrelation of the whole data is 0.6398. The value shows great spatial autocorrelation, indicating spatially clustered data. Hence, the assumption that the prices are spatially clustered is confirmed. Furthermore, an analysis of the change in Moran's I value with an increasing k value, meaning that more neighbors are considered, was computed. Results showed a decreasing Moran's I value of 0.6398 (k=40) to 0.6100 for k = 70, also indicating that the transaction prices are clustered (Wang, Chang, and Wang, 2019). Figure 4.6 shows the scatterplot of the Moran's I values, displaying the relationship between data values and their neighboring values. This graph provides further information on the Moran's I values. The crossing of the blue lines represents the mean value of transaction prices. Thus, values in the upper right and lower left quadrants indicate positive autocorrelation, with high values neighboring high values in the upper right quadrant and low values neighboring low values in the lower left quadrant. Hence, the other two quadrants indicate low spatial autocorrelation, where neighboring values show dissimilar values. The positive trend of the red regression line indicates positive autocorrelation.

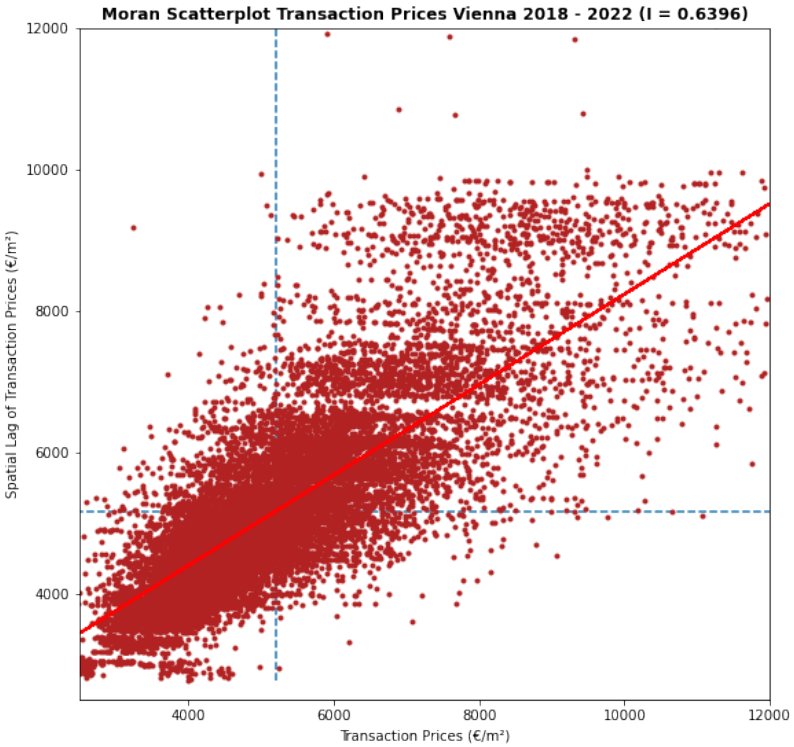


Figure 4.6: Moran's I Scatterplot

As displayed in Figure 4.7 and Table 4.3, the values of Moran's I have been rather stable between 2018 and 2021 but have sharply fallen in 2022. However, one indicator for this might be the lower count of purchase contracts captured by Exploreal in 2022 with 3078, compared to 4448 in 2018. This trend of less purchase contracts might on the one hand be due to the query date of the data (21.04.2023), with not all of the data from 2022 being available. On the other hand, the reason is also the declining number of real estate transactions in 2022 Grosse and Lammer (2022). This topic of decreasing spatial autocorrelation of real estate prices over time could be of relevance for further research. The global Moran's I does not tell anything about the location of hotspots and coldspots. That is why local indicators of spatial analysis (LISA) will be computed, as these might deliver further insights. Furthermore, differential Moran's I can tell about if the changes of values are clustered.

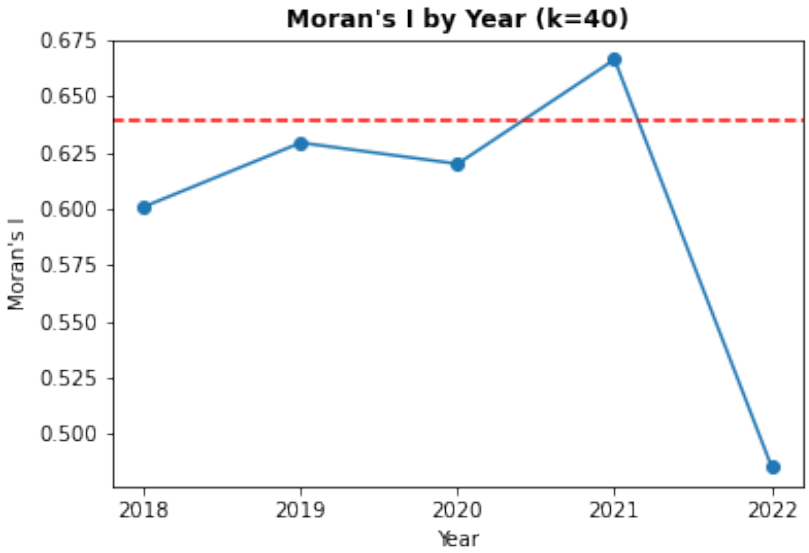


Figure 4.7: Moran's I 2018 - 2022

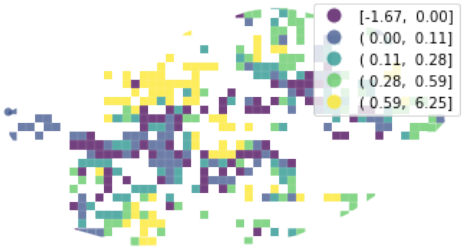
k	2018	2019	2020	2021	2022
$k = 20$	0.6426	0.7167	0.6620	0.6854	0.5079
$k = 40$	0.6010	0.6295	0.6201	0.6665	0.4857
$k = 60$	0.5645	0.5762	0.5827	0.6235	0.4549
$k = 80$	0.5216	0.5224	0.5489	0.5935	0.4310
$k = 100$	0.4754	0.4849	0.5219	0.5687	0.4062

Table 4.3: Moran's I values 2018 - 2022

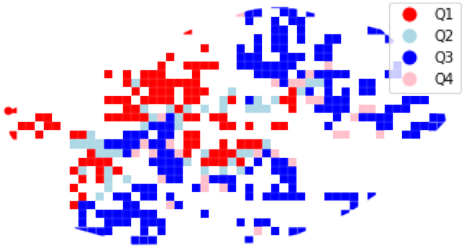
Global Moran's I delivered insights into the autocorrelation of the dataset and the change of spatial autocorrelation over the years. LISA provides more information about

the actual hot/cold spots of real estate prices and their geographic location. To calculate LISA it was necessary to group the data into cells. 50 * 50 cells were created for the investigation area of Vienna, with one cell having the dimensions of 397.52 * 482.98 m. For each cell, the average transaction price per square meter was calculated. The results of the LISA computation are displayed in 4.8.

The spatial analysis of LISA (Rey, Arribas-Bel, and Wolf, 2023) showed significant hot spots in the center and north-west of Vienna, with some cold outliers. Non-significant values were observed in areas around the center, while areas at the outskirts showed cold spots. Especially, for those areas with non-significant hot/cold spots it is of interest, how the clustering algorithms delineate those. To also capture the temporal component, further analysis on the spatial change of LISA values over the years was performed. However, these analyses showed similar patterns as in Figure 4.8. The hot spots remain in the center, the same areas stay non-significant, while the outskirts show cold spots. The number of significant cells in 2022 with 113 is lower than in 2020 (138) and 2021 (136), while the number of non-significant cells stays about the same (between 90 - 100). Reasons for this could be that there is more data in the non-significant areas in 2022. Respectively, more data of significant areas is available in 2020 and 2021. Another reason could be that the prices in the different areas have become more homogeneous in 2022. Hence, fewer statistically significant hot/cold spots are detected. This trend was also observed above, when applying global Moran's I on the data.

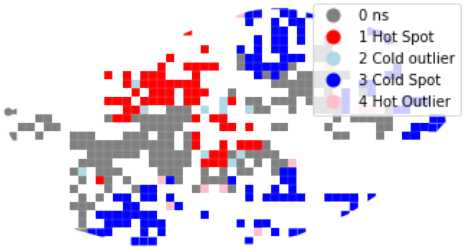


(a) LISA I values



(b) Type of spatial grouping, with Q1 = HH, Q2 = HL, Q3 = LL, Q4 = LH

(c) Statistic Significance of the Hot/Cold Spots ($p < 0.05$)



(d) Significant Hot/Cold Spots

Figure 4.8: LISA Maps

The differential Moran's I was calculated in a last step to deliver information about the change of the transaction prices and especially if these changes are clustered. Due to the mentioned problems of bivariate Moran's I, it was waived. As stated in section 3.1.1, differential Moran's I provides insights about the autocorrelation of changes of a variable. GeoDa was applied to compute this measure. A Moran's I value of 0.714 (k=40) was calculated. As visible in Figure 4.9, the north-east and south of Vienna indicate areas with high-high clusters, meaning that there are high changes neighbored by high changes. The west and center, on the other hand, show low changes of values neighboring low changes. In particular, significant low-low areas were detected in districts with higher absolute values, such as the districts in the center and the 19th district. This trend was also observed when analysing the data on district-level, as evidenced by two examples. The data for the 22nd district shows a rise from 3,637 €/m² to 6,357 €/m², which indicates an increase of 74.8 %. On the other hand, the 19th district only shows a rise of 36.7 % for the same period, starting from a higher level of 6,376 €/m². These results confirm and underline the outputs of the differential Moran's I.

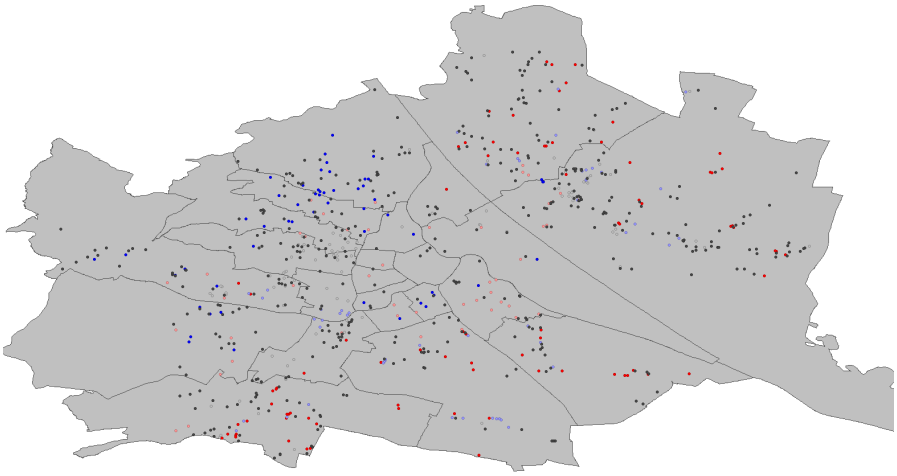


Figure 4.9: Differential Local Moran's I (dark red = HH, light red = LH, dark blue = LL, light blue = HL, grey = non-significant)

This section showed that the transaction prices are spatially clustered. However, especially in cheaper districts on the outskirts a strong increase of prices could be detected, while in more expensive districts such as the 19th, a lower increase of values was observed. The calculation of Moran's I for the different years also confirms this trend, as for 2022 a lower Moran's I value was calculated. This could indicate that the transaction prices are becoming more homogeneous. However, this will be analysed more deeply with the clustering results but it highlights the necessity of analysing the temporal component as well.

5 Clustering Geodata Cubes (CGC) & MDST-DBSCAN

The subsequent chapter deals with the implementation of the two clustering methods CGC and MDST-DBSCAN.

5.1 Clustering Transaction Prices with Clustering Geodata Cubes (CGC)

CGC offers both a co-clustering and a tri-clustering method. Analysis of both has shown that in this case co-clustering is the supreme method. Tri-clustering would need additional information for the bands-dimension, which could for instance be the specific day of the transaction date. However, CGC does not allow null-values. Therefore, grouping this specific data by the days was not meaningful, as almost all locations do not have transactions for each day or month. That is why co-clustering was chosen over tri-clustering.

5.1.1 Pre-Processing for CGC

As already stated previously, CGC is a polygon-based clustering technique. Since a point data set is used, the aggregation of these points to polygons is necessary. One possibility would be to use administrative boundaries to group the data, but one of the principles of this thesis is to not be dependent on such areas. Therefore, a regular grid was applied instead, which can be seen in Figure 5.1. The average transaction price for each location at a certain time point was calculated.

Creating these grid-cells come along with some challenges. One of these is the choice of the grid size, that can have a great influence on the clustering results. Too small grids lead to more cells with null values, which is a problem for CGC. Larger grids however go hand in hand with a greater information loss. That's why different sizes were compared, to find the best size. It is necessary to keep the highest granularity possible but still to not have too many null values.

In addition to spatial aggregation, the temporal component also had to be considered. The goal was not to calculate the mean of all values in the grid for all timestamps, but rather to calculate the mean at a specific location for a specific timestamp. This required defining a suitable period. Creating a grid for each day would be computationally too expensive and would result in many cells containing null values. Therefore, a compromise had to be found between reducing information loss and having enough cells with values.

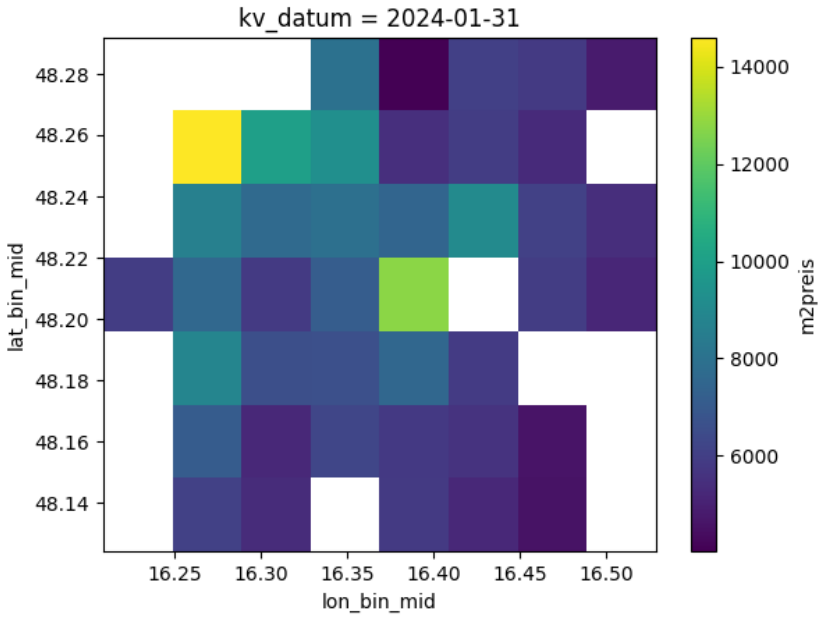


Figure 5.1: Grid over investigation area 2022 (Grid Size: 2,671.68 * 2,938.11 meters)

Different periods were investigated, ranging from days to years. After conducting multiple runs with different combinations of cell sizes and temporal granularity, a temporal resolution of 24 months and grid cells of 2,671.68 * 2,938.11 meters were determined.

Analysis of grouping the data into grids showed that CGC might not be a suitable method for this specific dataset. Even though there are 19,476 data points, they are not evenly distributed throughout the city, and certain coordinates have many transactions, while others have none. Hence, some areas - especially when additionally to space also considering time - do not have any projects, while others have a lot of data points. Calculation of the nearest neighbor index (NNI) of the projects in the data showed a value of 0.55. NNI values below one indicate clustered data, while values above one show dispersed data (Grekousis, 2020). This confirms what could visually already be explored in 4.3. The new construction projects are clustered, and this, combined with the fact that some of these projects have up to 402 transaction prices, suggests that data is not available for all parts of the investigation area.

After grouping the data into the cells, the pandas dataframe was transformed into an xarray dataset, and then into an xarray array. After that the coordinates were stacked and stored in a space object, which is needed for the co-clustering algorithm.

As stated, CGC does not work with null values. As a result, further preprocessing of the data was necessary, to avoid null clusters. The first option would be to delete those cells that contain null values at any timestamp. However, this approach leads to significant information loss, as all spaces that contain null values in one of the months/quarters/years would be deleted. The other option would be to only delete those location-cells that do

not contain data for all of the timestamps. However, in this case, null cells would still be left in the data for those timestamps where a location has a null value. As a result, the output of CGC would still be 0 clusters. Another option would be to interpolate values spatially and temporally. However, the objective was to not manipulate the data in this step. Therefore, additionally to the combination of spatial and temporal grouping that was improved above, the remaining null values had to be deleted. The drawback of this approach is information loss in those areas that do not have any data for at least one timestamp.

5.1.2 CGC clustering

The input parameters to the co-clustering algorithm were the following:

- **Number of Time Clusters**: 2
- **Number of Space Clusters**: 6
- **Maximum Number of Iterations**: 500
- **Convergence Threshold**: 0.01
- **Number of differently initialized Runs**: 15

The analysis was performed following the tutorial of CGC, and, therefore only steps that were not performed in a similar manner are analyzed in more depth.

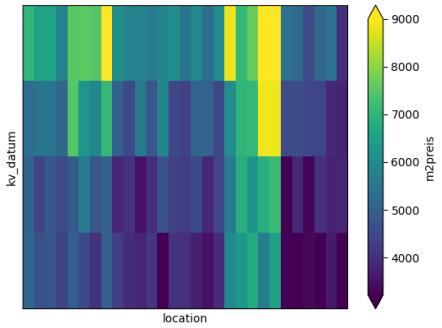
Figure 5.2a presents the mean square meter price for each location at the four time stamps. All locations experienced an increase of prices during the investigation period.

Figure 5.2b illustrates the average square meter price for each co-cluster, at two time-clusters. Once again, it highlights the presence of price increases within the clusters.

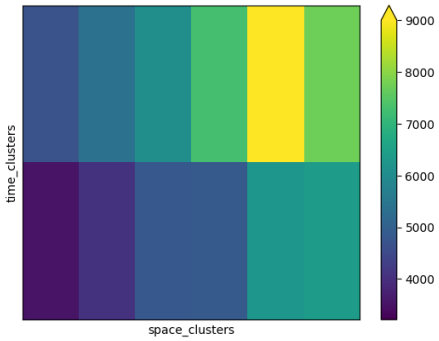
In Figure 5.2c the spatial distribution of the clusters for the two time clusters is depicted. Notably, there are missing values in the center, surrounded by areas with comparatively higher prices.

K-Means was employed to enhance the clustering outcomes, and the results are visualized in Figure 5.3. The optimal value of k, which represents the number of clusters, was determined to be 5. As mentioned earlier, this figure also indicates an upward trend in prices between the time stamps. The southern region exhibits the lowest price values, while the central area and one cluster in the northeast demonstrate higher average transaction prices. However, the application of K-Means refinement in this case only led to minor changes, as depicted in the figure.

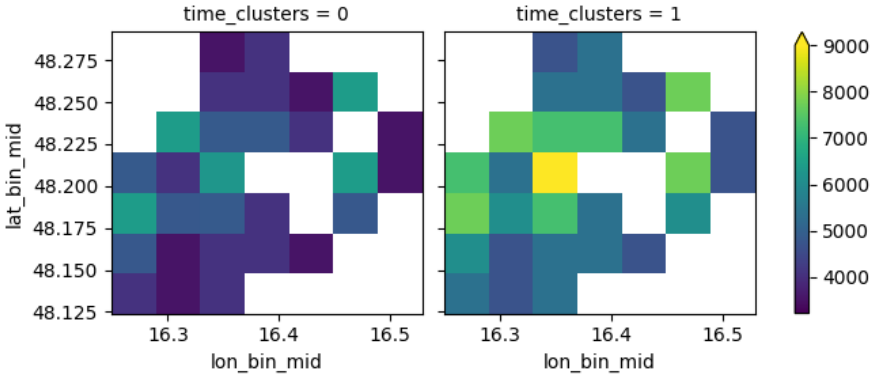
Further-in-depth analysis of CGC was not conducted, as the domain knowledge of the real estate market and submarkets in general indicate that this method does not yield satisfactory results. Due to this realization, the focus was placed on MDST-DBSCAN.



(a) Average square meter price for each location and time



(b) Average square meter price for each co-cluster



(c) Distribution of spatial clusters and their median transaction prices

Figure 5.2: Spatial and Temporal Co-Clusters by CGC

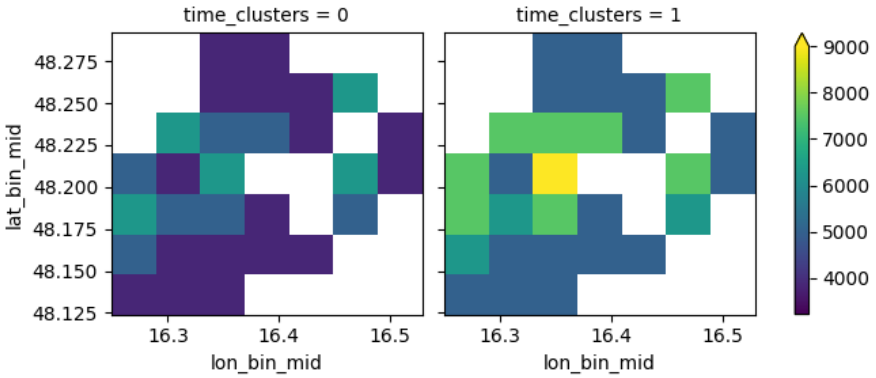


Figure 5.3: Spatio-Temporal Clusters created by CGC Co-Clustering after K-Means refinement

5.1.3 Advantages and Disadvantages CGC

In summary, while CGC is a useful Python package for conducting polygon-based clustering on georeferenced time-series data, it may not be the most suitable method for analyzing multidimensional spatio-temporal event data, such as transaction prices of new construction projects. Although a transformation of the data into spatio-temporal cells was achieved, the resolution of the data proved to be too low to generate meaningful results using CGC. Alternative methods such as MDST-DBSCAN were explored in this thesis to overcome these issues and to provide more insightful clustering outcomes.

5.2 MDST-DBSCAN

Refer to chapter 3.6 for further information on the MDST-DBSCAN algorithm and how it has been used in research. This section covers how MDST-DBSCAN was applied to the data in this thesis, in order to create transaction price submarkets.

As described in 1 the technical objective was to create a python implementation of the MDST-DBSCAN, as this algorithm was only implemented in R. The implementation can be found on GitHub. All the functions could be implemented with Numpy and Pandas. A python class object with functions was created.

When using this implementation, firstly, one has to initiate the MDSTDBSCAN object with the threshold-parameters eps, eps2, eps3 and the number of minimum points minpts. After this you can run the algorithm with .run, where one has to input the data, with x and y coordinates, the time column in datetime-format and the value column as a pandas series. Another function that was implemented is to plot the clusters.

Choi and Hong (2021) stated that the output of MDST-DBSCAN highly depends on the parameters used. Hence, it is necessary to try different parameters. That is why grid search was performed to find the best hyperparameters eps, eps2, eps3 and minpts. Therefore, different values for the hyperparameters and combinations of these were applied

(Raschka and Mirjalili, 2017). For each combination the MDST-DBSCAN algorithm was computed and the best parameters were saved. These are defined by a metric, which in this case was the Dunn-Index and Davies-Bouldin. All hyperparameter combinations were logged to a csv-file, in order to save the results for later times. The grid search was carried out for different ranges of hyperparameter values and the aim was to start with relatively big ranges. Later these were narrowed down to smaller steps and ranges to find the optimum values. As grid search is rather computationally expensive, this step could not be performed on one day. That is why logging of all values was necessary.

As mentioned in 5.3, another aspect was considered when performing the grid search. The number of minpts and the Epsilon-values were defined using visual interpretation or in other words domain knowledge. Additionally, a maximum number of clusters and maximum number of noise points were defined. The first component was necessary, as smaller clusters show spatially, pricewise more homogeneous areas. However, the aim is to form clusters that are not too narrow and show similar characteristics within. The maximum number of noises accepted had to be defined, as a small number of clusters and a good index score often showed a high number of noise - which is not purposeful. Therefore, the number of maximum clusters was defined with 30 and the number of noise accepted with 2700 data points.

After performing multiple runs and fine-tuning the values with grid-search, the following hyperparameters were used for clustering.

- eps: 0.0135
- eps2: 850
- eps3: 1450
- minpts: 55

Even though it is not implemented in the algorithm itself, multiple runs were performed to improve the sorting of the dataframe. For every run, the data was shuffled and the metric 5.3 was computed for each dataframe. 50 runs, therefore 50 different sortings, were computed and the best clusters were saved.

In a next step the output of the clustering algorithm was post-processed. Construction projects should only be assigned to one cluster. However, as every transaction price in the algorithm is treated as an own object, this requirement could not be met by the algorithm itself. Therefore, for all projects with more than one cluster-mark, the cluster with the highest number of transaction prices in this project was assigned to all the other dwellings in this project. Hence, every project only was assigned to one cluster. Another benefit that comes along with this is that it strengthens the robustness of the algorithm. This step makes the algorithm less sensible to the first point that is used in the method, as the impact of single data points is reduced. However, this disadvantage cannot be eliminated completely by this step. Furthermore, the count of noise is also reduced through this step. Overlapping clusters are not completely eliminated, without changing the algorithm itself. However, this step reduced the overlapping cluster areas. This is the biggest achievement of this step.

5.2.1 Results of MDST-DBSCAN

As stated above, the new construction projects are spatially clustered. Especially the area around the "Gürtel" shows many transaction prices. This has to be noted when observing the results.

Table 5.1 shows the clusters and the number of data points per cluster, before and after post-processing (grouping by project-id). As you can see the number of noise reduced dramatically through this step from 1958 to 1299. There is a trend visible, that the clusters with lower numbers have higher counts, while those with higher numbers, for instance Cluster 17, show a lower number of data points. One possible reason for this could be the known dependency of this clustering algorithm on the initial point. All clustering runs showed similar trends of many data points being assigned in the clusters with low numbers. The comparison of the counts before and after post-processing indicate, that these small clusters were already there after the clustering algorithm and not produced by processing. These small clusters could be deleted by increasing the minpts-parameter. However, this would increase the number of noise respectively.

Cluster	Count	Count 2
Noise	1958	1299
Cluster 1	2839	3017
Cluster 2	5094	5304
Cluster 3	4252	4479
Cluster 4	995	1001
Cluster 5	2114	2164
Cluster 6	628	632
Cluster 7	179	192
Cluster 8	221	232
Cluster 9	360	362
Cluster 10	150	150
Cluster 11	123	123
Cluster 12	106	112
Cluster 13	121	127
Cluster 14	76	78
Cluster 16	60	80
Cluster 17	56	35
Cluster 18	84	90

Table 5.1: Cluster values MDST-DBSCAN

Figure 5.4 represents the best output of the MDST-DBSCAN algorithm for the epsilon values mentioned above, after executing 50 runs. Figure 5.4a shows all the projects and their corresponding cluster mark, while Figure 5.4b shows the created polygons. Especially the latter one shows that the size of the polygons differs a lot. There are some very big clusters, such as Cluster 2 and Cluster 3, and on the other hand numerous very

small clusters, such as Cluster 9 and Cluster 8.

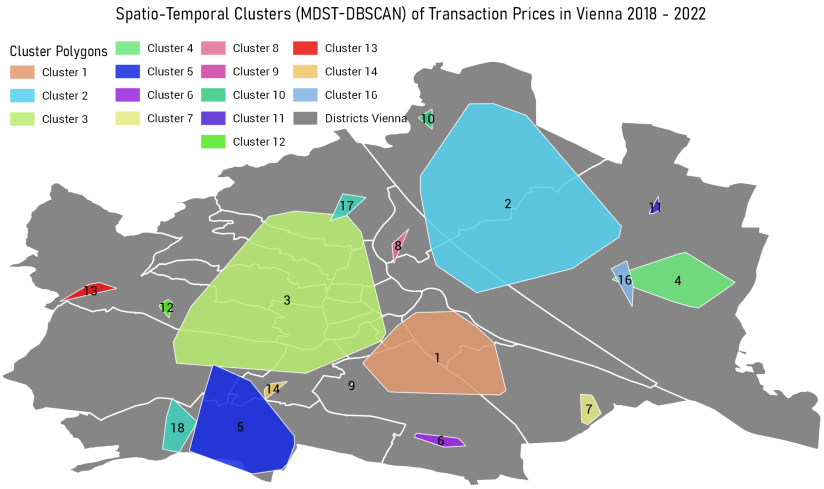
5.2.2 Problems with MDST-DBSCAN for creating Transaction Price Submarkets

As mentioned by Choi and Hong (2021), MDST-DBSCAN is highly affected by the parameter and initial point chosen for the algorithm. This was also detected in this thesis as the outputs changed when slightly changing the parameters. However, this effect can be reduced by performing more runs and by putting a lot of effort to the choice of parameters.

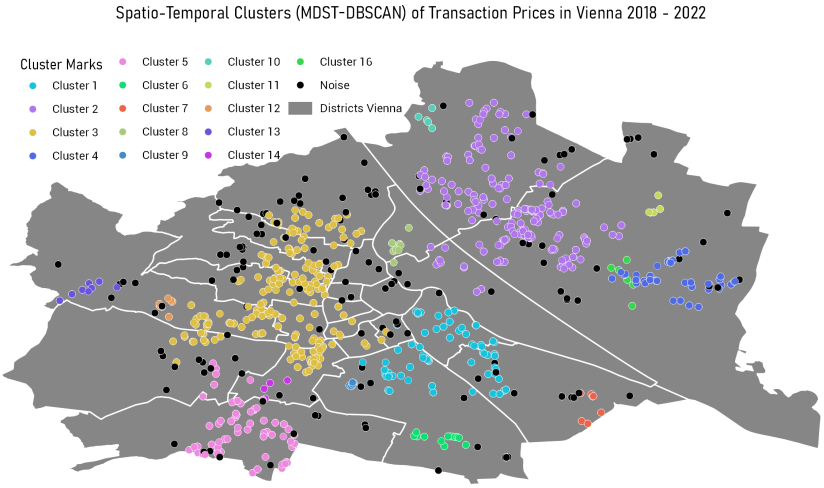
As stated above, MDST-DBSCAN method takes thresholds for the spatial, temporal and value dimension. If a point meets the thresholds and minimum number of points (minpts), this point and its neighboring points are assigned to the same cluster. However, if the time span of the whole period is greater than eps3 (time), which should be the case, more than one cluster could exist for the same spatial area. For instance, if the temporal distance between point i and its nearest neighbor at a later time stamp is further away than eps3, these two points are not connected. Especially with the data used in this thesis, such clusters can occur as the data highly depends on the building year of the houses respectively the transaction date. For instance, there could be areas where only two projects exist – one with a completion year at the beginning of the time span and one at the end of the time span. However, when forming submarkets, the aim is to form spatially contiguous clusters that show polygons in a 2D area with similar prices/attributes. Hence, there should only be one cluster per spatial location. There are different ways to achieve this objective of creating only one cluster per location.

The objective of submarket analysis is to incorporate both spatial and temporal dimensions as valuable components of the data. Thus, regarding the temporal dimension a restriction would limit the clustering analysis and potentially lead to loss of important information, as seen above. Neglecting the temporal component could result in apartments with different timestamps being treated as unrelated, despite being in the same location. Therefore, the aim is to build as stable and informative clusters as possible but only one cluster per location. One approach to achieve this with MDST-DBSCAN could be to merge, spatially overlapping clusters from different timestamps to one cluster. However, this would require post-processing and manipulating the outputs of the clustering algorithm.

Another possible solution could be to increase the value of eps3. However, this would mean decreasing the impact of the temporal dimension. Therefore, a compromise between information loss and connecting data points had to be found. Hence, although increasing eps3 can assist in reducing the number of clusters in the same location, it does not guarantee the prevention of such clusters from being formed. In the case of this data, it proved effective in avoiding the creation of spatially overlapping clusters. However, the temporal threshold of 850 days and value threshold of 1,450 €/m² is rather high. These high thresholds led to some very big clusters with a lot of data points. On the other hand, there are spatially very small clusters for the rest of the data. Hence, the aim of not creating too many overlapping clusters could be achieved by accepting less accurate and meaningful results.



(a) Cluster Marks



(b) Cluster Polygons

Figure 5.4: Spatio-Temporal Clusters identified by MDST-DBSCAN for Transaction Prices in Vienna 2018 - 2022

A third possibility was to implement an adaptation of MDST-DBSCAN that doesn't have the disadvantage of multiple clusters for different timestamps of the same location but treats the time dimension differently. Furthermore, the change of values is directly implemented in this case, using linear regression. This adaption is described in the next chapter 6. This algorithm could achieve better results for this specific data set.

5.3 Spatio-Temporal Metric

Metrics that were considered for evaluating the clustering outputs were introduced in 3.4. Davies-Bouldin Index (Davies and Bouldin, 1979) and an implementation of Dunn-Index were tried out in the beginning. Both metrics showed scores indicating very well formed clusters. However, these results were not accurate.

The standard implementations of these indices only consider the spatial component. However, the aim is to find clusters, that are spatially contiguous and coherent but also share similar values, respectively change of values. That is why Davies-Bouldin-Score was combined with the residuals of a linear regression model, as presented in equation 5.3. The score of Davies-Bouldin-Index was added to the normalized mean of the residuals. A linear regression model (x=time, y=transaction prices) was created for all clusters and the mean of the residuals of all transaction prices in the cluster was calculated. This combination allows to consider both the spatial contiguity but also the coherence of values. As none of the analysed papers have used similar metrics, it was necessary to implement this.

$$\text{spatio-temporal metric} = \text{db\_score} + \frac{1}{N}\sum_{i=1}^{N}\left(\frac{|y_i - \hat{y}_i|}{1000}\right) \times \text{value\_weight} \tag{5.1}$$

The output of the metric is interpreted such that the lower the score or reward, the higher the quality of the clustering result. However, this metric could not be used alone, but in combination with the number of clusters and noise points. Therefore, this metric was only applied for fine-tuning the parameters at the end. For the beginning, the number of clusters and noise were, additionally to visual interpretation, also used.

This metric was applied to MDST-DBSCAN and MDSTC-DBSCAN. See the respective chapters below to see how it was applied. In chapter 8 the outputs of MDST-DBSCAN and MDSTC-DBSCAN, when changing the parameters, are compared.

6 Multidimensional Spatio-Temporal Change DBSCAN (MDSTC-DBSCAN)

Multidimensional Spatio-Temporal Change Density-Based Clustering of Application with Noise was developed to overcome the mentioned problems of MDST-DBSCAN of temporally overlapping clusters and not incorporating the change of values. This method is described and executed in the subsequent chapter. The outputs of MDSTC-DBSCAN and MDST-DBSCAN are compared, and the clusters produced by this proposed method are analysed in greater depth using spatio-temporal methods.

6.1 Explanation of MDSTC-DBSCAN

The method is visualized in Figure 6.1 and in the pseudocode 6.

The user must specify two threshold values, namely a spatial threshold eps and a value-threshold eps2. It should be noted that the value threshold is specific to this algorithm and differs from the threshold used in MDST-DBSCAN, as described further below. In addition, the user must specify the minimum number of points (minpts) required to form a dense region, as well as the number of runs (n) for the algorithm. The algorithm is executed n runs, as was specified by the user. At the start of every run, the data set is being shuffled and a random point is chosen. This step is performed, to improve the sorting of the data, as the algorithm's performance can be affected by the way the data is sorted.

In a first step, when looping over all the values, all neighboring points of point i within the spatial eps-threshold are identified and stored in a variable. This is accomplished by calculating the Euclidean distance between each point and its neighboring points. The result is a variable j, containing all the neighboring points of point i. After this, 6.1 **linear regression**

$$y = ax + b \tag{6.1}$$

is computed for point i and all neighboring points j, that were identified in the previous step, where:

- y is the dependent variable of transaction prices.
- x is the independent variable time.

.

This step is performed, to also consider the change of values over time. Then, for each object in the neighboring objects j the value of that certain day is predicted, using

Algorithm 1: MDSTC-DBSCAN Algorithm

```
Input: eps, eps2, minpts, n runs, data
Output: Cluster Marks for every
Function mdstcdbscan (eps, eps2, minpts, n runs, data):
    initialize variables;
    for r in range(n runs) do
        shuffle the data;
        for i in range(n) do
            if point i is not assigned to any cluster then
                find neighboring points within distance threshold eps;
                compute Linear Regression and predict the values for all the
                 neighbors;
                keep all neighbors, where predicted value - actual value <= eps2;
                if i + number of neighboring points < minpts then
                    assign point i as noise;
                end
                else
                    assign point i to the new cluster;
                    add neighboring points to the cluster;
                    for each neighboring point j do
                        find neighboring points of point j within distance threshold
                         eps;
                        compute Linear Regression and predict the values for all the
                         neighbors;
                        keep all neighbors, where predicted value - actual value <=
                         eps2;
                        if i + number of neighboring points > minpts then
                            add neighbors to the cluster;
                        end
                    end
                end
            end
        end
        calculate metric for clusters;
        if the metric is better than best metric then
            save current clustering as the best;
        end
    end
    return the best clustering;
end
```

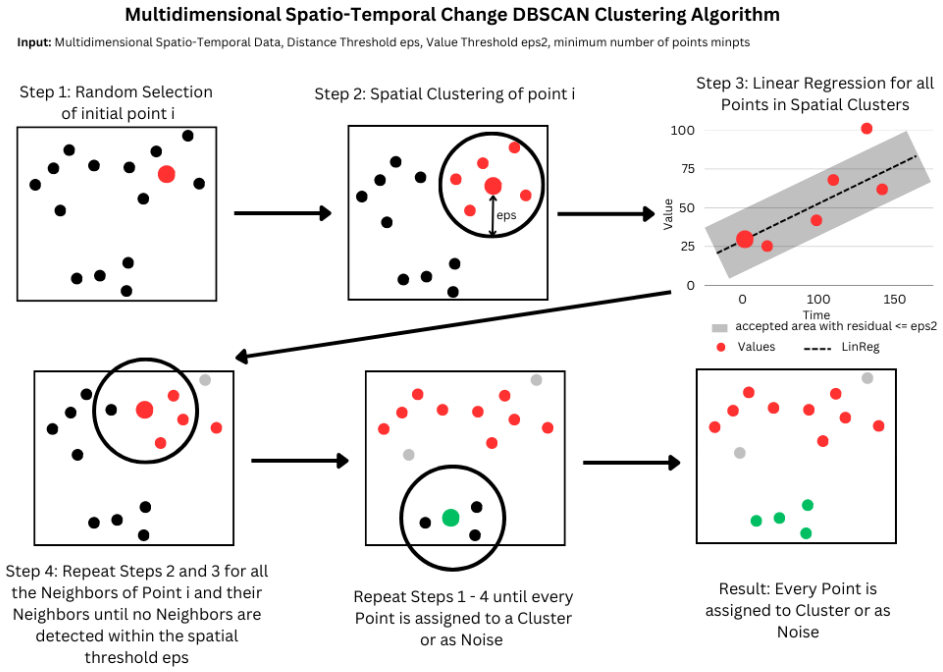


Figure 6.1: Multidimensional Spatio-Temporal Change Density-Based Clustering of Application with Noise

the linear regression model. The actual value of each point is then compared with the predicted value. If the residual is less than the value threshold eps2, the point is kept in the cluster. However, if the residual exceeds eps2, the point is removed from the neighboring points j. If the remaining number of point i plus all neighboring points within the distance and value thresholds eps and eps2 is greater or equal to minpts, a new cluster is created and the cluster mark is assigned to all these values. This procedure is then repeated for all the neighboring points j and their respective neighboring values, until all these neighboring points j and their neighboring points are assigned to a cluster or are marked as noise.

These steps are then repeated for all points that have not been assigned to a cluster or as noise yet. If all points are assigned, this run ends and the result is compared with the best cluster. For each run of the clustering algorithm, a metric (see 5.3) is calculated to evaluate the quality of the ordering. The reward is compared with the best reward until this run. If the reward is lower than the currently best reward, these current clusters are assigned as the new best clusters. At the end of all runs the best clusters, the best sorted data frame and the best reward are returned by the algorithm.

To summarize, the MDSTC-DBSCAN algorithm does not cluster the data according to a temporal and value-threshold but rather clusters it by a value-threshold that considers

the temporal component. For each point i and its neighboring points, a linear regression model is calculated, to capture changes in values over time. Points are kept in the cluster if the residual between the predicted value and actual value is less than the value threshold. However, if the residual exceeds the value threshold, the point is removed from the cluster. It is important to note, that the linear regression is not calculated for all the points of a cluster but rather for all the neighboring points of the analysed point. As a result, one cluster is created by multiple different linear regression models.

6.2 Operability of MDSTC-DBSCAN

As mentioned above, Agrawal, Garg, Sharma, and Patel (2016) proposed certain requirements that can be utilized to evaluate spatio-temporal clustering algorithms. MDSTC-DBSCAN exhibits similar characteristics to MDST-DBSCAN, as it can handle comparable data and generates clusters with similar shape. Nonetheless, there are also some differences between them. These distinctions are elaborated upon in the following sections, which will discuss the operability of MDSTC-DBSCAN and the results of the algorithm.

MDSTC-DBSCAN overcomes the limitations of MDST-DBSCAN, of forming multiple clusters for different timestamps of the same location. This innovative adaptation of MDST-DBSCAN considers changes in the values of the data, enhancing the algorithm's accuracy and applicability. Choi and Hong (2021) mentioned a drawback of MDST-DBSCAN, wherein the results can heavily rely on the initial point chosen and the parameter values. MDSTC-DBSCAN partially addresses these limitations but does not eliminate these issues completely. However, the results of the MDSTC-DBSCAN demonstrated greater robustness than those of the MDST-DBSCAN algorithm. Still, to achieve better results, the algorithm was executed multiple times. It is important to note that the clustering results may not be identical across different runs, as there are numerous possible ways to sort the data. However, using multiple runs and taking those clusters that show the best value according to a metric, can aid finding the best clusters.

It is worth noting that clusters produced by MDSTC-DBSCAN can still overlap, since neighboring points (even from the same project) can be assigned to different clusters. For example, a transaction price of one apartment might suit better to cluster 1, while another transaction price, with for instance price-increasing factors, may be better suited for another cluster.

To address the issue of overlapping clusters the output of the clustering algorithm was post-processed and, again, the major cluster mark for all the data points of every construction project was calculated. The best data frame of MDSTC-DBSCAN was grouped by the project-id and the number of cluster marks per project was counted. The most common cluster mark in this project was assigned to all other data points of this project as well.

The next step involved computing the polygons around the cluster points that could be further analyzed. This was achieved using the Shapely and Geopandas libraries in Python. Furthermore, census districts were utilized, to create spatially contiguous clusters. This is analysed in more depth below in 6.3.

Implementing DBSCAN for the spatial threshold was not suitable, as this method would deliver the same clusters every time. Thus, the linear regression step would only be able to eliminate values from these clusters. However, in the next step, it would not be meaningful to calculate DBSCAN again for these values. The spatial and temporal value clustering have to be performed in one step, rather than being separated by using for instance DBSCAN.

6.3 Results of MDSTC-DBSCAN

It is important to acknowledge that the results of this algorithm are highly sensitive to the chosen hyperparameters. Hence, if the objective is to generate more clusters, this can be easily achieved by changing these parameters accordingly. However, in this particular case the aim was to create a smaller number of clusters, to facilitate further analysis in subsequent steps.

After conducting a grid search, comparing metrics and interpreting visual results, the following hyperparameters were determined for the clustering process:

- eps: 0.022
- eps2: 1700
- minpts: 270
- n runs: 50

The best clusters obtained from the algorithm after 50 runs were used for further analysis. Table 6.1 presents the number of points per cluster, where "Count" refers to the number before post-processing and "Count 2" refers to the number after post-processing. As shown in the table, there are initially 1,218 data points classified as noise, which is reduced to 827 data points after post-processing. This was anticipated, as certain projects may have dwellings with for instance large open spaces, leading to high transaction prices per square meter and consequently being identified as noise.

Cluster 2 is the largest cluster with 3,361 data points, followed by Cluster 4 with 3,124 data points, before post-processing. On the other hand, the smallest cluster is cluster 16 with 113 points, followed by cluster 11 with 131 points. Hence, similar characteristics as with MDST-DBSCAN can be observed, as the first clusters show the highest count. However, in the case of MDSTC-DBSCAN, this phenomenon is less pronounced, compared to MDST-DBSCAN. The number of data points of the specific clusters does not change dramatically after post-processing, and the overall trends remain consistent with the pre-processing phase. However, there is a reduction of overlapping clusters during the post-processing step.

Table 6.1 also includes the median transaction price per square meter for each cluster. Cluster 6, as depicted in Figure 6.2, holds the highest median price with 7,362.37, while cluster 15 exhibits the lowest median price with 3,694.86. Cluster 6 is located in parts of

Cluster	Count	Count 2	Median Transaction Price (€/m²)
Noise	1218	827	7753.66
Cluster 1	3124	3180	4682.71
Cluster 2	3361	3414	5056.56
Cluster 3	1928	1952	4193.02
Cluster 4	3735	3787	5138.98
Cluster 5	962	997	4510.11
Cluster 6	538	565	7362.37
Cluster 7	2183	2255	4197.25
Cluster 8	606	627	4315.91
Cluster 9	501	604	6679.81
Cluster 10	402	312	4782.56
Cluster 11	131	127	6452.11
Cluster 12	228	216	4365.37
Cluster 13	161	185	5938.44
Cluster 14	136	106	3812.66
Cluster 15	150	201	3694.86
Cluster 16	113	122	4093.63

Table 6.1: Cluster values MDSTC-DBSCAN

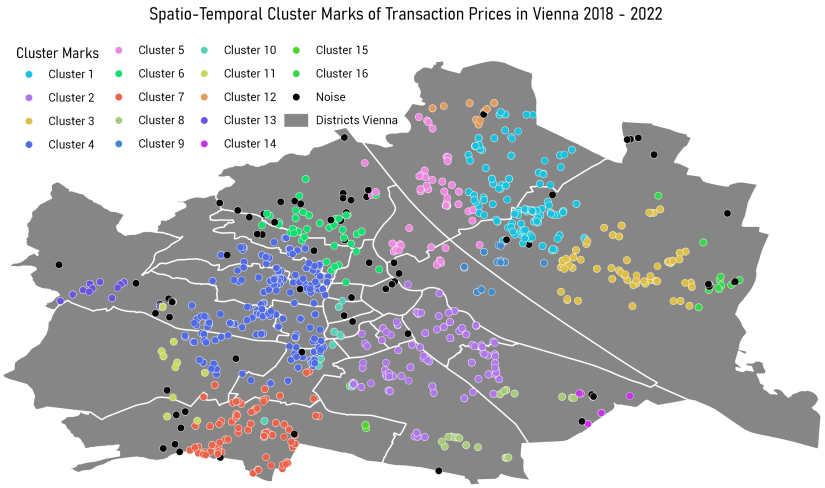
the 19th, 18th and 9th district, which were previously identified as districts with high prices (refer to section 4.3.1).

Figure 6.2 provides maps illustrating the results of MDST-DBSCAN. Subfigure 6.2a displays the cluster marks for each project after post-processing, while Subfigure 6.2b shows the cluster polygons.

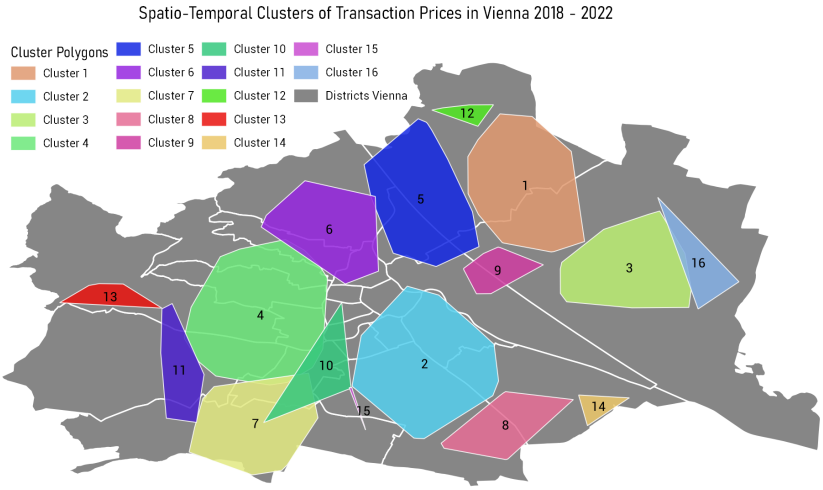
As observed previously, a total of 16 clusters were identified. In terms of spatial extent, cluster 4, 2, 1 and 5 are the largest, while clusters 15, 12, 14 and 13 are the smallest. Notably, the 15th cluster exhibits a very small area but still meets the minimum point requirement (110 points) with 201 transaction prices. Despite cluster 14 not meeting the minimum point requirement after post-processing, it is not deleted. "Discovery of clusters with arbitrary shape" is one of Agrawal, Garg, Sharma, and Patel (2016) requirements for spatio-temporal clustering algorithms. The analysis of clusters of various sizes confirms that this requirement is fulfilled by the MDSTC-DBSCAN algorithm.

As depicted in Figure 6.2 most clusters, (with a few exceptions, such as cluster 13, 12, 3, 16 and 14) exhibit boundaries that span across multiple districts. This observation confirms the anticipated advantage of point-based clustering techniques, as they provide new insights into the data that are not readily available through other clustering methods.

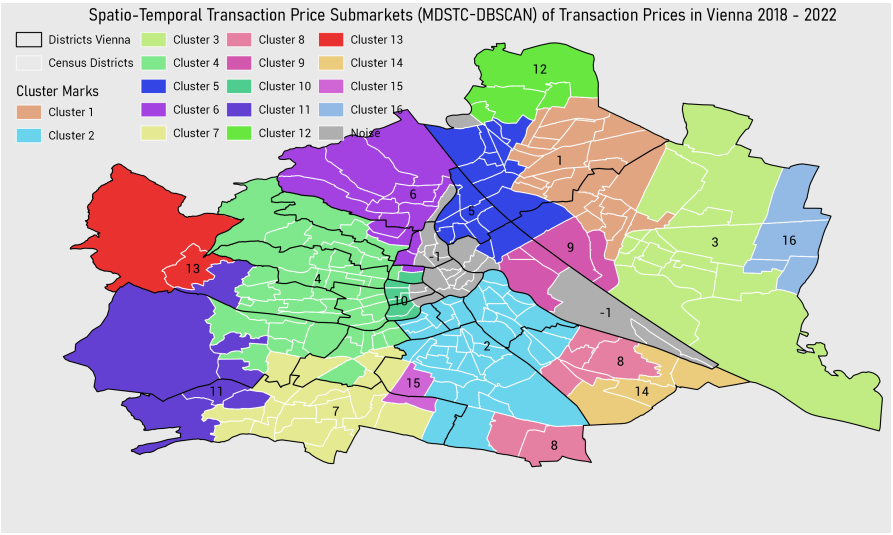
However, it is important to note that overlapping clusters exist, as seen in the case of cluster 10 overlapping with clusters 4 and 7. Agrawal, Garg, Sharma, and Patel (2016)'s requirement regarding the ability to treat neighboring clusters, therefore can not be fully met by this algorithm.



(a) Cluster Marks



(b) Cluster Polygons



(c) Housing Price Submarkets Grouped by Vienna Census Districts (Data: Stadt Wien)

Figure 6.2: Spatio-Temporal Clusters identified by MDSTC-DBSCAN for Transaction Prices in Vienna 2018 - 2022

6.4 Creating spatially continuous Submarket Areas

To address the issue of overlapping clusters and achieve spatially continuous areas for submarkets, the cluster marks were grouped by census districts of Vienna. Figure 6.2c illustrates this map. In a first step for creating these areas, the cluster marks were aggregated by census districts and the most frequent cluster mark was assigned to each census district. However, if noise was the predominant cluster mark, it was disregarded. Only those census districts that did not have any transaction prices belonging to a cluster were assigned the noise label in this initial step.

After this step, certain areas remained that were surrounded by census districts with different cluster marks. For example, within cluster 7 there was one census district with cluster mark 10. In that census district, there was only one project which was assigned to cluster 10. This project can be observed in Figure 6.2a, where one different cluster mark is visible within cluster 7. In such cases, the cluster mark of the surroundings was assigned to the analysed census district to ensure consistency.

In the subsequent step, the census districts that were initially assigned the noise label were further processed. If neighboring census districts with cluster marks existed, the length of the longest border to each cluster mark of neighboring census districts was calculated. The cluster mark of the neighboring census district with the longest border was then assigned to the analysed census district. It is important to note that this step was performed only once, as indicated by the presence of census districts labeled as noise in Figure 6.2c.

An additional advantage of aggregating the cluster marks to census districts is that it allows for further analysis using census districts data. This enables the exploration of additional variables associated with each submarket, such as population (refer to section 7.1.1).

7 Analysis of Clustering Results

The forthcoming analysis will focus on conducting a spatial and spatio-temporal analysis of the clusters generated by MDSTC-DBSCAN. These clusters have been identified as the most significant submarkets. It is important to note, that only those projects that have been assigned a cluster mark were analysed. An alternative approach would involve analysing all the projects within the cluster polygons, including those classified as noise. However, for the purpose of this analysis, the focus will be on projects with assigned cluster marks.

7.1 Spatio-Temporal Analysis of Clusters

Table 7.1 presents the coefficients of the linear regression models for each cluster, as well as the percentage change of the values between 01/01/2018 and 31/12/2022, as predicted by the models. The clustering results map, presented in Section 6.3, can also be seen in Figure 7.1. The highest rise of values has been in cluster 9, followed by Cluster 1, which indicates an increase of 52.71 %. Both of these clusters are located at least partially within the 22nd district. Hence, this corresponds with the trends analysed in the data analysis (refer to Section 4.3.1), where the 22nd district showed a significantly high rate of price growth. The variations between these clusters within the 22nd district highlight the necessity of analysing transaction prices at a sub-district level. Clusters 3 and 16, that are also within the 22nd district, have experienced substantial price increases, with 44.00 % and 49.43 % respectively. However, the starting values (intercept) differ between these clusters within the 22nd district. For example, Cluster 9 exhibits a much higher starting value with 4,918.19 €/m² than cluster 16 with 3,332.91 €/m². Therefore, these clusters present a much more diverse picture than the analysis conducted at the district-level.

When comparing neighboring clusters, both the algorithm's strengths and weaknesses, such as the reliance on the initial data point chosen for clustering, can be observed. For instance, the price growth rate of clusters 1, 5 and 12 in the North of Vienna exhibit 52.71 % 32.86 % and 22.74 % respectively. Even though these clusters show similar starting values, being between 3,904.11 €/m² and 4,043.60 €/m², the algorithm could differ well between neighboring areas with dissimilar prise rising rates.

Other neighboring clusters such as cluster 4 and cluster 6 show similar trends as the mentioned neighboring clusters above. While cluster 4 experienced a steep rise of values with 52.12 %, cluster 6 only indicates an increase of 26.03 %, starting however from a much higher initial value (=intercept) of 6,564.33 €/m² compared to 4,233.14 €/m² in cluster 4.

In the comparison above, it is evident that the cluster with the higher initial value

shows a much lower increase of prices compared to the other cluster. This trend holds true for the entire dataset, with a negative correlation (-0.2813) between the starting value and the percentage change. This finding confirms the trend explored in Section 4.3.1, where a decrease in Moran's I value in 2022 indicated a convergence of dwelling prices in Vienna towards a higher level of homogeneity.

Table 7.2 presents additional attributes that were analysed to compare the different clusters. Neighboring clusters, in some cases, exhibit significant differences in the analysed attributes. For instance, clusters 4 and 5 indicate much lower median prices and usable living areas compared to their common neighbor cluster 6.

However, as aforementioned, the analysis of some neighboring clusters also reveals a weakness. For instance, cluster 3 and cluster 16 do not exhibit these price differences observed between clusters 4 and 6. Specifically cluster 16 has a median price of 4,126.98 €/m², while cluster 3 has a price of 4,192.87 €/m². Hence, further analysis on why these two were not jointed to one cluster should be applied. This highlights a vulnerability of the algorithm, as these two clusters might be merged to one cluster in other runs, depending on the initial selection. The algorithm's reliance on the initial selection of transaction points is a weakness that should be tackled in the future.

Cluster	Slope	Intercept	Change (%)
1	1.13	3904.11	52.71
2	1.57	4342.55	66.02
3	0.87	3620.13	44.00
4	1.21	4233.14	52.12
5	0.73	4043.60	32.86
6	0.94	6564.33	26.03
7	1.15	3352.20	62.84
8	0.99	3711.87	48.46
9	2.63	4918.19	97.60
10	0.94	4516.29	38.00
11	0.82	5915.87	25.38
12	0.49	3959.54	22.74
13	0.25	5732.45	7.89
14	0.91	3260.15	50.67
15	0.19	3674.90	9.66
16	0.90	3332.91	49.43

Table 7.1: Linear Regression Coefficients of the Clusters

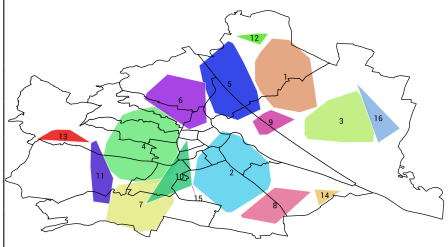


Figure 7.1: MDSTC-DBSCAN results

Table 7.2 provides insights into various attributes of the clusters, including the median transaction date, which varies across the clusters. Since a linear rise of values is expected, the median prices in those clusters with a lower date may be lower, respectively higher for later median dates. Due to this reason, the initial price of the linear regression models is shown above in Table 7.1.

Cluster Number	Median Usable Living Area	Median Transaction Price/m²	Median Open Space in m²	Median Date	Dwellings with Open Space (%)	Number of Projects
1	55.4	4708.81	8.93	2020-08-18	97.30	117
2	55.9	5086.00	8.18	2019-07-09	90.98	90
3	60.2	4192.87	10.00	2020-03-13	98.72	62
4	55.8	5130.42	7.39	2020-07-14	92.13	163
5	56.1	4450.05	7.66	2020-03-02	93.38	47
6	70.3	7382.77	10.14	2020-11-19	90.97	40
7	72.1	4250.00	9.55	2020-01-27	96.27	73
8	59.2	4310.03	9.56	2019-06-05	99.52	19
9	52.3	7144.14	9.19	2021-10-08	97.85	13
10	55.8	4678.57	10.10	2019-10-24	96.15	10
11	78.2	6488.55	13.50	2020-09-25	98.43	11
12	77.7	4441.14	10.15	2021-05-18	99.54	9
13	73.9	6026.24	13.50	2021-02-16	98.92	13
14	73.5	3947.64	10.13	2020-08-19	100.00	6
15	76.5	3765.20	12.78	2019-03-18	98.51	4
16	62.0	4126.98	10.17	2020-09-08	100.00	12
Noise	75.2	6663.24	10.10	2021-03-11	94.07	70

Table 7.2: Comparison of Transaction Price Clusters

The median square meter of open spaces varies across the clusters, ranging from 7.39 m² for cluster 4 to 13.5 m² for cluster 11. It is worth noting, that in each of the clusters more than 90 % of the dwellings have open space areas. The variables price/m² and size of open spaces (balconies, terraces and loggias) demonstrate a significant positive correlation of 0.2655. This indicates that as the open space areas increase, the prices also tend to increase. It is important to note that in the data, the prices of terraces and balconies were not directly included in the dwelling prices, which supports this observed correlation.

However, when considering the correlation between the median transaction price/m² and open space areas for the clusters themselves, no significant correlation is observed (p-value 0.5357). For example, the most expensive cluster (cluster 6) has the lowest percentage of apartments with open spaces (90.97%). This suggests that other factors than the open space areas might play a more significant role in determining the prices within the clusters.

Clusters 11, 12 and 15 are associated with the largest flats in terms of square meters, while clusters 9, 1 and 10 indicate smaller flats. Interestingly, the most expensive cluster, cluster 6, consists of relatively large flats with a median size of 70.3 m², while the second most expensive cluster, cluster 9, comprises smaller flats with a median size of 52.3 m². This indicates that the size of flats alone cannot fully explain the price differences per square meter observed across the clusters.

Clusters 15, 14 and 12 show the smallest numbers of projects with 4, 6 and 9 respectively. These clusters also have small spatial extent. Consequently, their results must be considered with caution due to the limited sample size. On the other hand, clusters 4, 1 and 2 consist of the largest numbers of projects, with 163, 117 and 90 projects respectively. It is important to note that there is a high number of noise projects of 70. This means that 10.84 % of the projects were assigned the noise label.

Cluster 7, located in the southern part of Vienna, exhibits a high price rising rate of

62.84 %, starting from a low value of 3,352.20 €/m². A similar trend can be observed for cluster 14, with a rising rate of 50.67 % and a starting value of 3,260.15 €/m². Both clusters also share the common feature of comparatively larger dwelling sizes, with cluster 7 at 72.1 m² and cluster 14 at 73.5 m². Both of these clusters are on the outskirts in the south of Vienna. Thus, in this case bigger dwellings with a comparably lower price have seen a significant price rising rate.

Cluster 12, located in the northern outskirts of Vienna, exhibits similar characteristics to clusters 7 and 14, with a starting value of 3,959.54 €/m² and a median dwelling size of 77.7 m². However, in this area a lower price rising rate of 22.74 % was observed. This discrepancy highlights the need for further analysis to understand the factors contributing to the different price rising rate in these two areas. However, this would go beyond the scope of this work.

Additionally, there are certain areas, such as the noise points in the east of Vienna and the north of the 22nd district, that do not form distinct clusters due to the minpts argument. Transaction prices in the first district were also assigned as noise, indicating outliers in this district.

7.1.1 Census Districts based Analysis of Clusters

Figure 7.2 displays the map of census districts with their assigned cluster mark. Table 7.3 represents the analysed variables for the clusters, that rely on the aggregation of cluster marks to administrative units, such as census districts.

The first column of the table shows the absolute predicted population change between 2018 and 2028 for each cluster, while the second column presents the percentage change. The last column indicates the population in 2022. The "Completion Rate" column represents the number of completed residential units per cluster per year per 1,000 inhabitants, using data of the years 2020 - 2022. This allows for comparisons of the number of residential units constructed across the clusters. It is important to note, that this is different to the data used for clustering, as the clustering-data only includes dwellings and no other residential units such as semi-detached houses and others. The completion rate however uses all residential units, including semi-detached houses, detached houses and town-houses. Considering a three-year period enhances the robustness of the results and facilitates comparison with other publications, for instance Exploreal and WKO (2022).

For cluster 10 a decline in population is predicted, with -0.54 %. The most expensive cluster, cluster 6, shows a modest percentage increase in population increase of 4.69 % and a comparatively low completion rate of 2.64 % for the analysed years. Cluster 9, that showed the highest prices rising rate, indicates a higher completion rate of 7.15 and a population percentage change of 5.61 %.

Cluster 5, despite having a comparatively lower price rising rate of 32.86 %, shows a significant predicted population rise of 8.94 %. Cluster 14, indicates the highest predicted population change of 9.59 %. However, this cluster has only seen a construction rate of 6.01, which is not among the highest. Cluster 14 experienced a substantial increase in prices (50.67%), starting from a relatively low value of €3,260.15/m².

Clusters 15 exhibits the highest completion rate among the analysed years. However, this is partly due to the very low population of 11,343 in this cluster. Clusters 1 and 3 also indicate high completion rates with 15.60 and 19.72 respectively, and their population are also predicted to grow until 2028 by a comparatively high rate of 7.87 % and 9.40 % respectively.

Cluster 12, with a comparatively lower price rising rate and a starting value of 3,959.54 €/m², demonstrates a high completion rate of 14.08 and population growth of 8.39 % until 2028. Hence, this cluster is an attractive destination for both construction projects and people moving to the area, possibly due to relatively lower prices compared to the other clusters.

Cluster Mark	Predicted Population Change (2018 - 2028)	Predicted Population Change (%) (2018 - 2028)	Completion Rate	Population 2022
1	12710	7.87	15.60	175,080
2	24264	5.91	9.67	420,739
3	7970	9.40	19.72	95,018
4	30823	6.65	5.76	452,800
5	15371	8.94	9.31	174,514
6	5119	4.69	2.64	108,276
7	8669	6.81	10.45	140,096
8	2655	8.30	5.16	34,996
9	2864	5.61	7.15	51,055
10	-321	-0.54	5.99	57,421
11	1814	6.40	2.56	28,635
12	1623	8.39	14.08	21,577
13	429	3.26	3.98	13,478
14	3053	9.59	6.01	32,604
15	701	7.33	31.10	11,343
16	855	6.46	15.01	14,772
Noise	2638	2.58	6.35	99,189

Table 7.3: Completion Rate and Population Change in Clusters

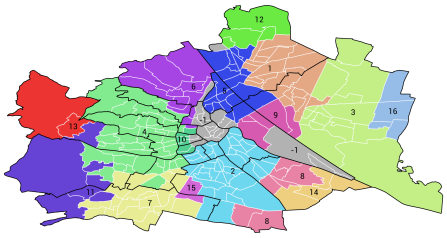


Figure 7.2: Transaction price submarkets

7.2 Comparison of MDST-DBSCAN and MDSTC-DBSCAN

Choi and Hong (2021) identified two key disadvantages of MDST-DBSCAN. Firstly, the results of the algorithm are highly dependent on the chosen hyperparameters, which can significantly impact the clustering outcomes. Secondly, the selection of the initial point can have a strong influence on the generated clusters. This section covers a comparison of these two methods, in relation to these two aspects.

	MDSTC-DBSCAN		**MDST-DBSCAN**	
Percentage Change	**Number of Noise**	**Number of Clusters**	**Number of Noise**	**Number of Clusters**
-10%	1252	19	2293	28
-6%	1185	21	1966	24
-2%	1158	18	2017	21
0%	1068	16	1836	20
2%	841	20	1927	18
6%	967	16	2036	12
10%	1138	14	1780	13

Table 7.4: Comparison of MDSTC-DBSCAN and MDST-DBSCAN

Table 7.4 provides a comparison of the stability of MDST-DBSCAN and MDSTC-DBSCAN, regarding their sensitivity to changes in hyperparameters. The best hyperparameters for both algorithms, as determined in sections 5.2 and 6.3), were used as a baseline. These hyperparameters were then changed by 10 %, 6 %, 2 % in both negative and positive directions to assess their impact on the results.

The number of clusters shows a greater variation for MDST-DBSCAN, ranging from 12 to 28 clusters depending on the specific hyperparameter adjustments. On the other hand, MDSTC-DBSCAN exhibits a narrower range of variation in the number of clusters created, with values ranging between 14 and 21 clusters. The table reveals that MDST-DBSCAN tends to receive a higher number of noise in all the runs, even though the number of clusters is in a similar range as with MDSTC-DBSCAN. However, the stability of noise points is similar for both methods.

Table 7.5 presents a comparison of the score of the metric (as referenced in 5.3), the number of noise points, and the number of clusters for both algorithms when varying the initial point selection.

This comparison once again emphasizes the necessity of running the algorithm multiple times. In terms of the number of clusters generated by MDSTC-DBSCAN, one outlier can be observed in run 9, where only 13 clusters were created. In all other runs, the number of clusters detected ranges between 15 and 17. The third run received the best score, the lowest number of noise points and resulted in 16 clusters. Hence, the metric serves as a reliable indicator for the best clustering algorithm. However, it should be noted that there are outliers in the metrics, as runs 4 and 5 exhibit scores above 4, while the others have significantly lower values.

	MDSTC-DBSCAN			MDST-DBSCAN		
Run	**Score of Metric**	**Number of Noise**	**Number of Clusters**	**Score of Metric**	**Number of Noise**	**Number of Clusters**
0	2.6893	1267	17	2.2378	2083	14
1	2.2502	1248	15	1.7486	2217	13
2	2.3171	1268	17	2.0055	1985	16
3	1.7210	960	16	2.0700	1935	13
4	4.1607	1194	17	1.8843	2175	16
5	4.1524	1256	16	1.8091	2102	12
6	1.9480	1157	16	1.9149	2048	21
7	2.2286	1238	16	1.9600	2165	13
8	2.9807	1209	16	2.1457	1796	20
9	2.0512	1159	13	1.7019	2188	13

Table 7.5: Comparison of initial point chosen for MDSTC-DBSCAN and MDST-DBSCAN

MDST-DBSCAN algorithm produces a greater range of the number of clusters generated, from 12 to 21, when the data is sorted differently. However, it yields similar values regarding the number of noise. Furthermore, when evaluating the algorithm's performance using a spatio-temporal metric, it demonstrates more robust results compared MDSTC-DBSCAN. To explore the reasons of the robustness of the reward for MDST-DBSCAN, more analysis of the spatio-temporal metric would be necessary. However, as the metric is primarily applied to compare the outputs of the same technique and not to compare the outputs of different methods, this difference in results was accepted.

The computational time of the two algorithms was another aspect, that was analysed. In this category, MDST-DBSCAN outperformed the new algorithm with an average time of 170.06 seconds, while MDSTC-DBSCAN exhibited a higher average computation time of 597.90 seconds per run. Consequently, this serves as an advantage for MDST-DBSCAN.

To summarize, the findings suggest that MDSTC-DBSCAN exhibits greater stability regarding the number of clusters generated when compared to MDST-DBSCAN. This is supported by both, the comparison of hyperparameters and of initial point chosen. However, it must be noted that both methods demonstrate similar robustness regarding the number of data points assigned as noise, although MDST-DBSCAN showing a higher number of noise points. Regarding the score of the metric MDST-DBSCAN overall indicated better results, both in robustness and in terms of lower absolute values. However, it remains uncertain if this metric is suitable for comparing different methods, because it was developed for this task and was fine-tuned for comparing different parameters of one clustering algorithm, rather than for comparing different clustering methods. Regarding computation time, there is room for improvement in MDSTC-DBSCAN. The algorithm would greatly benefit from further development in this aspect.

8 Discussion

8.1 Answer of Research Questions

The following paragraphs provide answers to the research questions introduced in Chapter 1.

Objective 1: Analyse Vienna's real estate prices. RQ1: What are the transaction prices submarkets in Vienna? The submarkets for transaction prices, delineated by the newly proposed method MDSTC-DBSCAN, can be observed in Figure 6.2. Sixteen clusters were detected with the input parameters mentioned in chapter 6.3. The most expensive submarkets are Cluster 6, located mainly in the 19th district; Cluster 11, situated on the outskirts of the 13th and 14th districts; and Cluster 9, found in the 2nd and 22nd districts. The lowest prices were observed in clusters 14, 16, and 7. All of them are located in the outskirts, with clusters 14 and 7 situated in the southern 11th and 23rd districts, and cluster 16 in the east of Donaustadt.

RQ2: Does the price rising rate differ between the submarkets? Table 7.1 shows the rates of price increase for each cluster. The rate of price increase differs between the submarkets, with cluster 13 indicating the smallest change of 7.89 % and cluster 9 indicating the highest increase at 97.6 %. Hence, differences in the rates of price increase between spatial areas could be observed. In general, many of the more expensive submarkets, located in the 19th district and on the edges of the 14th and 13th districts, indicated lower price rising rates. Conversely, the less expensive submarkets, for instance in the 23rd district, exhibited higher price rising rates. However, there are exceptions, such as cluster 9 on the border between Leopoldstadt and Donaustadt, which, despite having a high starting value, also demonstrates the highest price rising rate.

RQ3: How has the homogeneity of the transaction prices of apartments in Vienna changed during the COVID pandemic? The analysis indicates that the transaction prices tend to become more homogeneous. After a rise in Moran's I value in 2021, a strong decline from 0.6665 to 0.4857 in 2022 could be observed. This was confirmed by the analysis of the clusters, as cheaper clusters tended to show higher rates of price increase, as mentioned above in the answer to RQ2. However, further analysis on this topic is necessary as the decline of Moran's I could only be observed in 2022.

Objective 2: Explore and compare a point versus polygon based spatio-temporal clustering technique for delineating real estate submarkets of transaction prices.

RQ1: Which MDST-DBSCAN parameters derive better clusters for the study area of Vienna? The best parameters for MDST-DBSCAN are mentioned in chapter 5.2. However, the new method MDSTC-DBSCAN showed more consistent results

as presented in chapter 7.2.

RQ2: Is CGC or MDST-DBSCAN a better method to delineate real estate submarkets? MDST-DBSCAN yielded better results than CGC, as the results of the latter method were not meaningful. CGC was developed for georeferenced time-series data and was not suitable for the data used in this thesis. In the case of MDST-DBSCAN, the initial clusters were quite large, encompassing many data points, while the subsequent ones were very small. Furthermore, this method occasionally generates spatially overlapping clusters for different timestamps, which is not purposeful when creating transaction price submarkets. The newly developed clustering method, MDSTC-DBSCAN, could produce better results than MDST-DBSCAN, as it displays more consistency in the size of the clusters and also incorporates time in a distinct manner, resulting in fewer spatially overlapping clusters. Therefore, MDSTC-DBSCAN is the superior method.

RQ3: What improvements can be made to enhance the performance of the superior method, that would achieve even better results? MDSTC-DBSCAN is an adaption and further development of MDST-DBSCAN, that was proposed in this thesis. The outputs of MDSTC-DBSCAN were superior because the method was better suited to the data and produced more meaningful results. Additionally, it improved the consistency of the clusters compared to MDST-DBSCAN. MDSTC-DBSCAN would benefit from further development regarding the implementation of other regression types than solely linear regression.

Technical objective 3: Develop a python implementation of the MDST-DBSCAN algorithm. The python implementation of the MDST-DBSCAN and MDSTC-DBSCAN algorithms can be found on a Github repository.

8.2 Discussion Results

In general, clusters in the outskirts showed lower prices with bigger dwelling sizes. Furthermore, the findings of Chapter 4 were confirmed through the analysis of the clusters. The 19th district has the highest median transaction price in the dataset, and this is also reflected in cluster 6, which is predominantly located in this district. On the other hand, cluster 7 in the 23rd district exhibits some of the lowest transaction prices but shows a high rate of price increase. This general trend is observed across multiple clusters, indicating that cheaper areas tend to experience a more significant increase in prices compared to already expensive areas.

The outskirts of Vienna present dissimilar results. For instance, the 23rd district displays low prices and high rates of price rise, while the outskirts of the 13th and 14th districts have high prices but low rates of price increase.

The 22nd district presents a differentiated picture. Areas near the 2nd district show high transaction prices and a high rate of price increase (cluster 9). On the other hand, areas on the outskirts of this district (cluster 3 and 16) demonstrate lower prices and lower rates of price increase. A similar differentiation can be observed in the 21st district, where cluster 12 showcases larger dwelling sizes, lower prices, and lower rates of price increase. In contrast, clusters 1 and 5 exhibit higher rates of price increase, higher prices

in general, and smaller apartment sizes.

A drawback of the approach applied in this thesis to delineate real estate submarkets is that only transaction prices and no other data was utilized. Hence, applying the clustering methods to more data is necessary to delineate submarkets as defined by literature (refer to section 2). Other data would be more property characteristics, such as the quality of the building or neighborhood attributes, such as the crime rate in this area.

8.2.1 Discussion Clusters

Cluster 1 shows a total of 3,180 transaction prices (after post-processing), being the third biggest cluster after clusters 4 and 2. Clusters 1 and 4 show a similar price rising rate with 52.71 % and 52.12 %, with starting values of 3,904.10 €/m² and 4,233.14 €/m² respectively, also in a close range. These two clusters also show similar dwelling sizes of 55.4 m² and 55.9 m², respectively. However, spatially these clusters are in different areas, as cluster 1 is in the 21st and 22nd district while cluster 4 is in the 'Gürtel'-area. Furthermore, they do not show similar characteristics regarding the completion rate (years 2020 - 2022), with 15.6 in cluster 1 and 5.76 in cluster 4.

Cluster 2 and 4 have the biggest population with 420,739 and 452,800, respectively. Both clusters show a similar population growth, similar starting price, and a similar median usable living area. Cluster 2 furthermore showed a great price rising rate with 66.02 %, being the second highest.

Hence, the three biggest clusters 1, 2, and 4 show similar characteristics in many categories. However, the price rising rate and completion rate differ between these clusters.

Cluster 3 indicates a considerable size with 1,952 transaction prices analyzed. It is neighboring cluster 16. Cluster 3 only shows a slightly higher starting value than cluster 16 with €3,620.13/m² compared to €3,332.91/m², and a lower price rising rate by 5.43 percentage points. Hence, this result raises questions as these two clusters could be merged into one cluster.

As mentioned above, cluster 4 is the biggest cluster in terms of both the number of transaction prices and population. It neighbors clusters 6, 7, 10, and 11. The starting value in cluster 4, €4,233.14/m², is lower than, for instance, clusters 6, 7, and 10, while cluster 7 shows a lower starting price but a higher price rising rate. With a price rising rate of 52.12 %, cluster 4 shows a rather higher price rising rate. As mentioned above, the cluster exhibits some similar characteristics with clusters 1 and 2 and is well demarcated from its neighbors.

Cluster 5 shows a starting value of 4,043.6 €/m² and a price rising rate of 32.86 %, , which is on the lower end but not among the lowest rising rates. Hence, it has a lower starting price than its neighboring clusters 1 and 6. With a median usable living area of 56.1 m², a similar value as clusters 1, 2, 4, and 10 can be observed. However, this value is much lower than in its neighboring cluster 6, which shows a size of 70.3 m².

Cluster 6 is the most expensive cluster with a starting value of 6,564.33 €/m² and a low price rising rate of 26.03 %. Therefore, similar characteristics as cluster 11 can be observed, which shows a starting price of 5,915.87 €/m² and a price rising rate of 25.38 %.

The median usable living area is also high with 70.3 m² in cluster 6 and 78.2 m² in cluster 11. Both these clusters show a rather low completion rate of 2.64 and 2.56, respectively.

Cluster 7 indicates completely different characteristics with a low starting price of 3,352.20 €/m² and a high price rising rate of 62.84 %. The median usable living area is rather big at 72.1 m². This cluster shows similar characteristics to cluster 14, where a low starting value, a comparably higher price rising rate, and a median dwelling size of 73.5 m² could be observed. Furthermore, both are located at the outskirts of Vienna.

Cluster 8 shows similar characteristics to cluster 7 and is also located at the outskirts. However, its starting value is higher at 3,711.87 €/m² and the price rising rate is 48.46 %. The usable living area amounts to 59.2 m², which is lower than in clusters 7 and 14. It also has to be noted that this cluster is not spatially jointed when analysed by the census districts.

Cluster 9 has a high starting price of 4,918.19 €/m² and the highest price rising rate at 97.60 %. Furthermore, this cluster shows the lowest median usable living area at 52.3 m². It has 13 projects and 604 transaction prices assigned to it. The completion rate of this cluster is 7.15. This cluster is an outlier regarding the combination of price and price rising rate.

Cluster 10 shows a price rising rate of 38.00 % and a price starting value of 4,516.19 €/m². With a median usable living area of 55.8 m², it shows a rather small value, similar to other clusters. However, this cluster is the only one where a population decline (-0.54 %) is predicted.

Cluster 11 was already analyzed in some parts above, as it shows similar characteristics to cluster 6. This cluster indicates the highest median usable living area at 78.2 m², the second highest starting price at 5,915.87 €/m², and a comparatively low price rising rate of 25.38 %. Furthermore, in this cluster and in cluster 13, the highest median open space could be observed at 13.5 m². The completion rate was rather low at 2.56, while a population increase of 6.40 % is predicted (2018 - 2028).

Cluster 12 is located at the northern outskirts and has a lower starting value of 3,959.54 €/m² and a low price rising rate of 22.74 %, along with a rather large median usable living area of 77.7 m². The predicted population change amounts to 8.39 %, and the completion rate was high at 14.08.

Cluster 13 is a small cluster with only 185 assigned transaction prices and 13 projects. It shows the third highest starting price at 5,732.45 €/m² but the lowest price rising rate at 7.89 %. Hence, this cluster shows lower values in both categories compared to its neighboring cluster 11. Like most clusters at the outskirts, it also indicates a higher median usable living area at 73.9 m² and, together with cluster 11, the highest open space at 13.5 m².

Cluster 14 was already analyzed together with cluster 7, as it shows similar characteristics with a low starting price, a higher price rising rate, and a high median usable living area of 73.5 m². Additionally, cluster 14 has the highest predicted population increase between 2018 and 2028 at 9.59 %.

Cluster 15 is a cluster that is not well delineated. It has 201 transaction prices and 4 projects, the lowest number in this category, assigned to it. This cluster shows a starting

value of 3,674.90 €/m² and a low price rising rate of 9.66 %. Furthermore, the dwellings in this cluster have a high median usable living area of 76.5 m². This cluster shows the highest construction rate, which is also due to the low population of 11,343, being the lowest among all clusters.

Cluster 16 was already mentioned in the analysis of cluster 3, as these two clusters are neighboring but show similar characteristics. The cluster is, in addition to the unsatisfactory delimitation from cluster 3, the smallest in terms of assigned transaction prices, with only 122 but with 12 projects. The median size of the dwellings is rather low for a cluster in the outskirts, at 62 m².

8.3 Discussion Methods

The two methods that were initially analyzed - CGC and MDST-DBSCAN - didn't show satisfactory results. The results of CGC were not meaningful, as this method is primarily designed for georeferenced time-series data. The transaction prices used in this thesis did not have the spatio-temporal resolution required to create cells small enough to produce meaningful results.

On the other hand, MDST-DBSCAN produced better results than CGC, but it generated large clusters and some very small ones. It heavily depends on the first point analyzed, and the main issue is that spatially overlapping clusters are possible for different timestamps. However, real estate submarkets are not supposed to spatially overlap. That's why a third method is proposed in this thesis, which considers the temporal component as a valuable component rather than a restriction.

MDSTC-DBSCAN is proposed in this thesis to create transaction price submarkets. It utilizes the temporal component of transaction prices to group them into clusters with similar prices. The method yielded better and more consistent results than MDST-DBSCAN, which was analyzed in Chapter 7.2. However, the analysis of the results revealed that there is room for improvement regarding the dependence on parameters and data sorting. These are two requirements defined by (Agrawal, Garg, Sharma, and Patel, 2016) that should be met by spatio-temporal clustering methods.

Furthermore, this method was only tested with transaction prices. Therefore, in a next step, the method could be applied to other spatio-temporal data with an additional attribute. The results of the method are promising, but to make it useful for other input data, it would be necessary to incorporate other regression types in addition to linear regression. Regression types such as polynomial regression, which better represent the new data, could be used. However, this would increase the impact of outliers, as these regression types fit the data better due to increased polynomial degrees. Additionally, the drawbacks observed in the delineation of clusters 3 and 16, as well as cluster 15, need to be addressed. Therefore, the code should be improved in a future step to make the results less dependent on the order of the input data.

9 Conclusion

To conclude, the proposed method **Multidimensional Spatio-Temporal Change - DBSCAN (MDSTC-DBSCAN)** shows promising results in creating transaction price submarkets. It successfully addresses the problem of spatially overlapping submarkets encountered in MDST-DBSCAN. Additionally, the new method demonstrates improved consistency in its algorithms compared to MDST-DBSCAN, which generated clusters of varying sizes. It is worth noting that while MDST-DBSCAN may be more suitable for other research purposes, MDSTC-DBSCAN is better suited for delineating transaction price submarkets. The literature review and empirical analysis emphasized the need for developing alternative spatio-temporal clustering methods.

One major observation of the literature review was the absence of a specific term for spatio-temporal data with an additional attribute, such as transaction prices. Consequently, there is also a lack of terminology for clustering techniques designed for this data. To address this gap, the terms **Multidimensional Spatio-Temporal Event Data** and **Multidimensional Spatio-Temporal Event Clustering** are proposed as novel terms to encompass this data type and clustering methods applied to it, respectively.

The metric used to compare the clustering results could benefit from improvement. While it was initially designed for this specific dataset, it is necessary to create metrics that have a more general applicability and are not limited to a single dataset. Another possibility is to incorporate noise and the number of clusters into the metric, hence providing a more comprehensive evaluation.

Overall, the results of MDSTC-DBSCAN demonstrated clear and well-defined clusters, with a few exceptions. One issue that should be addressed, is the dependence on the sorting of the data, that can affect the stability of the results. In future research, it is necessary to tackle this dependency and ensure more consistent outcomes. Furthermore, future research should also consider testing the method with transaction price data from other regions, different types of data, and various regression types. This would provide additional insights into the algorithm's usability and enhance its generalizability.

Still, most clusters were delineated well, and domain knowledge confirms the usefulness of the results. For instance, one cluster in most parts of the 19th district of Vienna showed the highest median transaction price, while certain clusters in the southern outskirts exhibited significantly lower prices but a greater rate of price increase. This observation further emphasizes the need for the proposed method, as it takes into account the temporal changes in transaction prices, which can vary across different spatial areas. The analysis of the clusters further supports the findings from comparing the Moran's I value of different years, **indicating a trend towards increased homogeneity in dwelling prices across Vienna, at a high level.**

Bibliography

Agrawal, K P, Sanjay Garg, and Pinkal Patel (2015). "Performance Measures for Densed and Arbitrary Shaped Clusters". In: *International Journal of Computer Science & Communication (ISSN: 0973-7391)* 6.2. DOI: 10.090592/IJCSC.2015.637.

Agrawal, K.P., Sanjay Garg, Shashikant Sharma, and Pinkal Patel (2016). "Development and Validation of OPTICS Based Spatio-Temporal Clustering Technique". In: *Information Sciences* 369, pp. 388–401. ISSN: 00200255. DOI: 10.1016/j.ins.2016.06.048.

Ansari, Mohd Yousuf, Amir Ahmad, Shehroz S. Khan, Gopal Bhushan, and Mainuddin (2020). "Spatiotemporal Clustering: A Review". In: *Artificial Intelligence Review* 53.4, pp. 2381–2423. ISSN: 0269-2821, 1573-7462. DOI: 10.1007/s10462-019-09736-1.

Anselin, Luc, Ibnu Syabri, and Youngihn Kho (2006). "GeoDa: An Introduction to Spatial Data Analysis". In: *Geographical Analysis* 38.1, pp. 5–22. ISSN: 0016-7363, 1538-4632. DOI: 10.1111/j.0016-7363.2005.00671.x.

Anselin,Luc (2019). *Bivariate Spatial Correlation - A Word of Caution*. 2019.

– (2020). *Distance-Band Spatial Weights*. 2020. https://geodacenter.github.io/workbook/4b_dist_weights/lab4b.html#k-nearest-neighbor-weights.

Arbelaitz, Olatz, Ibai Gurrutxaga, Javier Muguerza, Jesús M. Pérez, and Iñigo Perona (2013). "An Extensive Comparative Study of Cluster Validity Indices". In: *Pattern Recognition* 46.1, pp. 243–256. ISSN: 00313203. DOI: 10.1016/j.patcog.2012.07.021.

Atluri, Gowtham, Anuj Karpatne, and Vipin Kumar (2019). "Spatio-Temporal Data Mining: A Survey of Problems and Methods". In: *ACM Computing Surveys* 51.4, pp. 1–41. ISSN: 0360-0300, 1557-7341. DOI: 10.1145/3161602.

Birant, Derya and Alp Kut (2007). "ST-DBSCAN: An Algorithm for Clustering Spatial–Temporal Data". In: *Data & Knowledge Engineering* 60.1, pp. 208–221. ISSN: 0169023X. DOI: 10.1016/j.datak.2006.01.013.

Bourassa, Steven C., Foort Hamelink, Martin Hoesli, and Bryan D. MacGregor (1999). "Defining Housing Submarkets". In: *Journal of Housing Economics* 8.2, pp. 160–183. ISSN: 10511377. DOI: 10.1006/jhec.1999.0246.

BUWOG Group GmbH and EHL Wohnen GmbH (2022). *Erster Wiener Wohnungsmarktbericht 2022*.

Choi, Changlock and Seong-Yun Hong (2021). "MDST-DBSCAN: A Density-Based Clustering Method for Multidimensional Spatiotemporal Data". In: *ISPRS International Journal of Geo-Information* 10.6, p. 391. ISSN: 2220-9964. DOI: 10.3390/ijgi10060391.

Costello, Greg, Chris Leishman, Steven Rowley, and Craig Watkins (2019). "Drivers of Spatial Change in Urban Housing Submarkets". In: *The Geographical Journal* 185.4, pp. 432–446. ISSN: 0016-7398, 1475-4959. DOI: 10.1111/geoj.12303.

Das, Monidipa and Soumya K. Ghosh (2020). "Data-Driven Approaches for Spatio-Temporal Analysis: A Survey of the State-of-the-Arts". In: *Journal of Computer Science*

and Technology 35.3, pp. 665–696. ISSN: 1000-9000, 1860-4749. DOI: 10.1007/s11390-020-9349-0.

Davies, David L. and Donald W. Bouldin (1979). "A Cluster Separation Measure". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* PAMI-1.2, pp. 224–227. ISSN: 0162-8828, 2160-9292. DOI: 10.1109/TPAMI.1979.4766909.

Ester, Martin, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu (1996). "A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise". In: p. 6.

Exploreal and WKO, eds. (2022). *Erster Österreichischer Neubaubericht 2022*. 2022.

Fanaee-T, Hadi (2012). "Spatio-Temporal Clustering Methods Classification". In: DOI: 10.13140/RG.2.1.3812.7204.

Gao, Jay (2021). *Fundamentals of Spatial Analysis and Modelling*. First. Boca Raton: CRC Press. ISBN: 978-1-00-322052-7. DOI: 10.1201/9781003220527.

Grekousis, George (2020). *Spatial Analysis Methods and Practice: Describe – Explore – Explain through GIS*. First. Cambridge University Press. ISBN: 978-1-108-61452-8 978-1-108-49898-2 978-1-108-71293-4. DOI: 10.1017/9781108614528.

Grosse, Matthias and Sarah Lammer (2022). "Nach prosperierenden Jahren überschlagen sich seit Monaten die Negativ- schlagzeilen. Bedeuten diese Entwicklungen das Ende der Fahnenstange für den ständigen Aufwärtstrend in der Immobilienbranche? Welches Bild zeichnen die Daten von EXPLOREAL?" In: *ÖVI News Ausgabe 04/22*.

Hämäläinen, Joonas, Susanne Jauhiainen, and Tommi Kärkkäinen (2017). "Comparison of Internal Clustering Validation Indices for Prototype-Based Clustering". In: *Algorithms* 10.3, p. 105. ISSN: 1999-4893. DOI: 10.3390/a10030105.

Han, Jiawei, Micheline Kamber, and Jian Pei (2012). *The-Morgan-Kaufmann-Series-in-Data-Management-Systems-Jiawei-Han-Micheline-Kamber-Jian-Pei-Data-Mining.-Concepts-and-Techniques-3rd-Edition-Morgan-Kaufmann-2011.Pdf*. Elsevier.

Henriques, Rui and Sara C. Madeira (2019). "Triclustering Algorithms for Three-Dimensional Data Analysis: A Comprehensive Survey". In: *ACM Computing Surveys* 51.5, pp. 1–43. ISSN: 0360-0300, 1557-7341. DOI: 10.1145/3195833.

Herath, Shanaka and Gunther Maier (2013). "Local Particularities or Distance Gradient: What Matters Most in the Case of the Viennese Apartment Market?" In: *Journal of European Real Estate Research* 6.2, pp. 163–185. ISSN: 1753-9269. DOI: 10.1108/JERER-10-2011-0022.

Huang, Bo, Bo Wu, and Michael Barry (2010). "Geographically and Temporally Weighted Regression for Modeling Spatio-Temporal Variation in House Prices". In: *International Journal of Geographical Information Science* 24.3, pp. 383–401. ISSN: 1365-8816, 1362-3087. DOI: 10.1080/13658810802672469.

Huang, Changhai, Xucun Qi, Jian Zheng, Ranchao Zhu, and Jia Shen (2023). "A Maritime Traffic Route Extraction Method Based on Density-Based Spatial Clustering of Applications with Noise for Multi-Dimensional Data". In: *Ocean Engineering* 268, p. 113036. ISSN: 00298018. DOI: 10.1016/j.oceaneng.2022.113036.

Jacquez, Geoffrey M. (1996). "Ak NEAREST NEIGHBOUR TEST FOR SPACE-TIME INTERACTION". In: *Statistics in Medicine* 15.18, pp. 1935–1949. ISSN: 0277-6715,

1097-0258. DOI: 10.1002/(SICI)1097-0258(19960930)15:18<1935::AID-SIM406>3.0.CO;2-I.

Keskin, Berna (2022). "Multilevel Approach to the Analysis of Housing Submarkets". In: *Regional Studies, Regional Science* 9.1, pp. 264–279. ISSN: 2168-1376. DOI: 10.1080/21681376.2022.2067005.

Keskin, Berna and Craig Watkins (2017). "Defining Spatial Housing Submarkets: Exploring the Case for Expert Delineated Boundaries". In: *Urban Studies* 54.6, pp. 1446–1462. ISSN: 0042-0980, 1360-063X. DOI: 10.1177/0042098015620351.

Kisilevich, Slava, Florian Mansmann, Mirco Nanni, and Salvatore Rinzivillo (2010). "Spatio-Temporal Clustering". In.

Kopczewska, Katarzyna and Piotr Ćwiakowski (2021). "Spatio-Temporal Stability of Housing Submarkets. Tracking Spatial Location of Clusters of Geographically Weighted Regression Estimates of Price Determinants". In: *Land Use Policy* 103, p. 105292. ISSN: 02648377. DOI: 10.1016/j.landusepol.2021.105292.

Ku, Ou, Francesco Nattino, Meiert Grootes, Emma Izquierdo-Verdiguier, Serkan Girgin, and Raul Zurita-Milla (2022). *CGC: An Open-Source Python Module for Geospatial Data Clustering*. Other. display. DOI: 10.5194/egusphere-egu22-3855.

Lamb, David, Joni Downs, and Steven Reader (2020). "Space-Time Hierarchical Clustering for Identifying Clusters in Spatiotemporal Point Data". In: *ISPRS International Journal of Geo-Information* 9.2, p. 85. ISSN: 2220-9964. DOI: 10.3390/ijgi9020085.

Liu, Qiliang, Min Deng, Jiantao Bi, and Wentao Yang (2014). "A Novel Method for Discovering Spatio-Temporal Clusters of Different Sizes, Shapes, and Densities in the Presence of Noise". In: *International Journal of Digital Earth* 7.2, pp. 138–157. ISSN: 1753-8947, 1753-8955. DOI: 10.1080/17538947.2012.655256.

Mallela, Subramanyam, Darmendra Modha, and Inderjit Dhillon (2003). "Information-Theoretic Co-clustering". In.

Nationalbank, Oesterreichische, ed. (2023). *Property Market Review - Housing markets in Austria Q4 2022*. 2023.

Oliveira, Ricardo, Maribel Yasmina Santos, and Joao Moura Pires (2013). "4D+SNN: A Spatio-Temporal Density-Based Clustering Approach with 4D Similarity". In: *2013 IEEE 13th International Conference on Data Mining Workshops*. TX, USA: IEEE, pp. 1045–1052. ISBN: 978-1-4799-3142-2 978-1-4799-3143-9. DOI: 10.1109/ICDMW.2013.119.

Plank, Leonhard, Antonia Schneider, and Justin Kadi (2022). *Wohnbauboom in Wien 2018-2021. Band 1: Preise, Käufer:innen und Leerstände in der Wohnbauproduktion*. Stand Juni 2022. Stadtpunkte Nr 40. Wien: Kammer für Arbeiter und Angestellte für Wien. ISBN: 978-3-7063-0923-3.

Raschka, Sebastian and Vahid Mirjalili (2017). *Python Machine Learning: Machine Learning and Deep Learning with Python, Scikit-Learn, and TensorFlow*. Second edition, fourth release,[fully revised and updated]. Expert Insight. Birmingham Mumbai: Packt Publishing. ISBN: 978-1-78712-593-3.

Rey, Sergio J., Dani Arribas-Bel, and Levi John Wolf (2023). *Geographic Data Science with Python*. Chapman & Hall/CRC Texts in Statistical Science. Boca Raton: CRC Press. ISBN: 978-0-367-26311-9 978-1-03-244595-3.

Shi, Zhicheng and Lilian Pun-Cheng (2019). "Spatiotemporal Data Clustering: A Survey of Methods". In: *ISPRS International Journal of Geo-Information* 8.3, p. 112. ISSN: 2220-9964. DOI: 10.3390/ijgi8030112.

Soltani, Ali, Christopher James Pettit, Mohammad Heydari, and Fatemeh Aghaei (2021). "Housing Price Variations Using Spatio-Temporal Data Mining Techniques". In: *Journal of Housing and the Built Environment* 36.3, pp. 1199–1227. ISSN: 1566-4910, 1573-7772. DOI: 10.1007/s10901-020-09811-y.

Stadt Wien (2020). *Positionsbestimmung: der STEP 2025 aus heutiger Sicht aktuelle Einblicke und Ausblicke*. Wien: Stadt Wien, Stadtentwicklung und Stadtplanung (Magistratsabteilung 18). ISBN: 978-3-903003-61-3.

Wang, Jing and Filip Biljecki (2022). "Unsupervised Machine Learning in Urban Studies: A Systematic Review of Applications". In: *Cities* 129, p. 103925. ISSN: 02642751. DOI: 10.1016/j.cities.2022.103925.

Wang, Senzhang, Jiannong Cao, and Philip S. Yu (2022). "Deep Learning for Spatio-Temporal Data Mining: A Survey". In: *IEEE Transactions on Knowledge and Data Engineering* 34.8, pp. 3681–3700. ISSN: 1041-4347, 1558-2191, 2326-3865. DOI: 10.1109/TKDE.2020.3025580.

Wang, Wen-Ching, Yu-Ju Chang, and Hsueh-Ching Wang (2019). "An Application of the Spatial Autocorrelation Method on the Change of Real Estate Prices in Taitung City". In: *ISPRS International Journal of Geo-Information* 8.6, p. 249. ISSN: 2220-9964. DOI: 10.3390/ijgi8060249.

Wang, Zengzheng, Fuhao Zhang, and Yangyang Zhao (2023). "Exploring the Spatial Discrete Heterogeneity of Housing Prices in Beijing, China, Based on Regionally Geographically Weighted Regression Affected by Education". In: *Land* 12.1, p. 167. ISSN: 2073-445X. DOI: 10.3390/land12010167.

Watkins, Craig A (2001). "The Definition and Identification of Housing Submarkets". In: *Environment and Planning A: Economy and Space* 33.12, pp. 2235–2253. ISSN: 0308-518X, 1472-3409. DOI: 10.1068/a34162.

Wu, Xiaojing (2022). "Identification of Co-Clusters with Coherent Trends in Geo-Referenced Time Series". In: *ISPRS International Journal of Geo-Information* 11.2, p. 134. ISSN: 2220-9964. DOI: 10.3390/ijgi11020134.

– (2022). "Tri-Clustering–Based Exploration of Spatio-Temporal Heterogeneity of Six Criteria Air Pollutants and Their Relationships in China". In: *Frontiers in Earth Science* 10, p. 951510. ISSN: 2296-6463. DOI: 10.3389/feart.2022.951510.

Wu, Xiaojing, Changxiu Cheng, Raul Zurita-Milla, and Changqing Song (2020). "An Overview of Clustering Methods for Geo-Referenced Time Series: From One-Way Clustering to Co- and Tri-Clustering". In: *International Journal of Geographical Information Science* 34.9, pp. 1822–1848. ISSN: 1365-8816, 1362-3087. DOI: 10.1080/13658816.2020.1726922.

Wu, Xiaojing, Raul Zurita-Milla, Emma Izquierdo Verdiguier, and Menno-Jan Kraak (2018). "Triclustering Georeferenced Time Series for Analyzing Patterns of Intra-Annual Variability in Temperature". In: *Annals of the American Association of Geographers* 108.1, pp. 71–87. ISSN: 2469-4452, 2469-4460. DOI: 10.1080/24694452.2017.1325725.

Wu, Xiaojing, Raul Zurita-Milla, and Menno-Jan Kraak (2015). "Co-Clustering Geo-Referenced Time Series: Exploring Spatio-Temporal Patterns in Dutch Temperature Data". In: *International Journal of Geographical Information Science* 29.4, pp. 624–642. ISSN: 1365-8816, 1362-3087. DOI: 10.1080/13658816.2014.994520.

Wu, Yangyi, Yehua Dennis Wei, and Han Li (2020). "Analyzing Spatial Heterogeneity of Housing Prices Using Large Datasets". In: *Applied Spatial Analysis and Policy* 13.1, pp. 223–256. ISSN: 1874-463X, 1874-4621. DOI: 10.1007/s12061-019-09301-x.

Yao, Jing and A. Stewart Fotheringham (2016). "Local Spatiotemporal Modeling of House Prices: A Mixed Model Approach". In: *The Professional Geographer* 68.2, pp. 189–201. ISSN: 0033-0124, 1467-9272. DOI: 10.1080/00330124.2015.1033671.

You, Geonhwa (2022). "Spatiotemporal Data-Adaptive Clustering Algorithm: An Intelligent Computational Technique for City Big Data". In: *Annals of the American Association of Geographers* 112.2, pp. 602–619. ISSN: 2469-4452, 2469-4460. DOI: 10.1080/24694452.2021.1935207.

Zhang, Tao, MengChu Zhou, Xiwang Guo, Liang Qi, and Abdullah Abusorrah (2022). "A Density-Center-Based Automatic Clustering Algorithm for IoT Data Analysis". In: *IEEE Internet of Things Journal* 9.24, pp. 24682–24694. ISSN: 2327-4662, 2372-2541. DOI: 10.1109/JIOT.2022.3194886.

10 Appendix

10.1 Code MDSTC-DBSCAN

```
import numpy as np
import pandas as pd
from sklearn.linear_model import LinearRegression
from sklearn.metrics import davies_bouldin_score
from collections import defaultdict
from collections import deque

class mdstcdbscan():
    def __init__(self, eps, eps2, minpts, n_runs):
        self.eps = eps
        self.eps2 = eps2
        self.minpts = minpts
        self.n_runs = n_runs

    def run(self, x, y, time, value, id):
        self.x = x
        self.y = y
        self.time = time
        self.value = value
        self.id = id
        min_date = pd.to_datetime(time.min())
        self.time = ((self.time - min_date).dt.days)
        mdstc_data = defaultdict(list)
        best_cluster_mark = None
        best_reward = np.inf
        best_df = None
        for r in range(self.n_runs):
            n = self.x.shape[0]  # length
            classn = np.zeros(n, dtype=int)  # number of points in each cluster
            cluster_mark = np.zeros(n, dtype=int)  # which cluster
            cluster_b = np.zeros(n, dtype=bool)  # which points have already been added to a cluster
            cn = 0  # cluster number

            shuffled_indices = np.random.permutation(n)

            self.x = self.x.iloc[shuffled_indices].reset_index(drop=True)
            self.y = self.y.iloc[shuffled_indices].reset_index(drop=True)
            loc = np.column_stack((self.x, self.y))
```

```
            self.value = self.value.iloc[shuffled_indices].reset_index(drop=True)
            self.time = self.time.iloc[shuffled_indices].reset_index(drop=True)
            self.id = self.id.iloc[shuffled_indices].reset_index(drop=True)
            for i in range(n):
                reachable_points = []

                unclass = np.where(cluster_mark < 1)[0]
                unassigned = unclass
                a = np.array([loc[i, 0], loc[i, 1]])
                a = np.tile(a, (n, 1))
                fordist = np.column_stack((a, loc))
                idist = np.abs(np.sqrt((fordist[:, 0] - fordist[:, 2]) ** 2 + (fordist[:, 1] - fordist[:, 3]) ** 2))

                if cluster_mark[i] == 0:
                    reachables = unassigned[idist[unassigned] <= self.eps]
                    r_values = self.value[reachables]
                    r_time = self.time[reachables]
                    if len(r_time) >= 1:
                        model = LinearRegression().fit(r_time.to_numpy().reshape(-1, 1), r_values.to_numpy().reshape(-1, 1))

                        # predict the value for all reachable points
                        predicted_values = model.predict(r_time.to_numpy().reshape(-1, 1))

                        # calculate the residuals for all reachable points
                        ivaluedist = np.abs(r_values - predicted_values.flatten())
                        reachables = reachables[ivaluedist[reachables] <= self.eps2]

                        #predict the value for point i
                        predicted_value_i = model.predict([[self.time[i]]])

                        # calculate the absolute difference for point i
                        abs_difference_i = np.abs(self.value[i] - predicted_value_i)

                        if abs_difference_i > self.eps2:
                            cluster_mark[i] = -1
                        else:
                            reachable_points.append(i)
                            reachables = np.setdiff1d(reachables, i)
                            while len(reachables) > 0:
                                reachables = reachables.astype(int)
```

```
                                neighbors = reachables
                                reachables = np.array([])

                                unique_elements = list(set(neighbors.
    astype(int)) - set(reachable_points))
                                queue = deque(unique_elements)

                                while queue:
                                    j = queue.popleft()

                                    if j not in reachable_points:

                                        b = np.array([loc[j, 0], loc[j
    , 1]])
                                        jfordist = np.column_stack((np
    .tile(b, (loc.shape[0], 1)), loc))
                                        jdist = np.sqrt((jfordist[:,
    0] - jfordist[:, 2]) ** 2 + (jfordist[:, 1] - jfordist[:, 3]) ** 2)

                                        jreachables = unassigned[jdist
    [unassigned] <= self.eps]
                                        r_values = self.value[
    jreachables]
                                        r_time = self.time[jreachables
    ]

                                        if len(r_time) >= 1:
                                            model = LinearRegression()
    .fit(r_time.to_numpy().reshape(-1, 1), r_values.to_numpy().reshape
    (-1, 1))

                                            # predict the value for
    all reachable points
                                            predicted_values = model.
    predict(r_time.to_numpy().reshape(-1, 1))

                                            # calculate the residuals
    for all reachable points
                                            jvaluedist = np.abs(
    r_values - predicted_values.flatten())
                                            jreachables = jreachables[
    jvaluedist[jreachables] <= self.eps2]
                                            # predict the value for
    point j
                                            predicted_value_j = model.
    predict([[self.time[j]]])

                                            # calculate the absolute
    difference for point j
                                            abs_difference_j = np.abs(
    self.value[j] - predicted_value_j)

                                            if abs_difference_j <=
```

```
    self.eps2:
114
115                                                 reachable_points.
    append(j)
116                                                 for point in
    jreachables:
117                                                     if point not in
    reachable_points:
118
    reachable_points.append(point)
119                                                 unique_elements = list
    (set(neighbors.astype(int)) - set(reachable_points))
120                                                 reachables = np.
    union1d(unique_elements, jreachables)
121                                                 # Add new unique
    elements to the queue
122                                                 for elem in
    unique_elements:
123                                                     if elem not in
    queue:
124                                                         queue.append(
    elem)
125
126                             if len(reachable_points) >= self.minpts:
127                                 unclass = np.setdiff1d(unclass,
    reachable_points)
128                                 cn += 1
129                                 cluster_mark[reachable_points] = cn
130                                 cluster_b[reachable_points] = True
131                                 classn[reachable_points] += 1
132                             else:
133                                 cluster_mark[i] = -1
134             try:
135                 reward = self.custom_metric(self.x, self.y, self.time,
     self.value, cluster_mark)
136             except ValueError:
137                 # Skip invalid clusterings
138                 continue
139             if reward < best_reward:
140                 best_cluster_mark = cluster_mark
141                 best_reward = reward
142                 best_df = pd.DataFrame({'id': self.id, 'x': self.x, 'y
    ': self.y, 'time': self.time, 'value': self.value, 'cluster_mark':
    cluster_mark})
143             mdstc_data['Run'].append(r)
144             mdstc_data['Reward'].append(reward)
145             mdstc_data['Noise'].append(cluster_mark[cluster_mark ==
    -1].shape[0])
146             mdstc_data['Number of Clusters'].append(np.unique(
    cluster_mark).shape[0])
147
148         return best_cluster_mark, best_df, best_reward, mdstc_data
149
```

```
    def custom_metric(self, x, y, time, value, cluster_mark, value_weight=0.5):
        # Compute Davies-Bouldin index
        combined_data = pd.DataFrame({'x': x, 'y': y, 'time': time, 'value': value})
        numerical_data = combined_data[['x', 'y']].copy()
        db_score = davies_bouldin_score(numerical_data, cluster_mark)

        # Fit a linear regression model to the value over time within each cluster and compute the residuals
        combined_data['cluster'] = cluster_mark
        clusters = combined_data['cluster'].unique()
        residuals = []

        for cluster in clusters:
            cluster_data = combined_data[combined_data['cluster'] == cluster]
            X = cluster_data['time'].values.reshape(-1, 1)
            y = cluster_data['value'].values
            model = LinearRegression()
            model.fit(X, y)
            y_pred = model.predict(X)
            residual = np.abs(y - y_pred).mean()
            residuals.append(residual)

        # Combine the Davies-Bouldin index and average residual
        normalized_residuals = np.mean(residuals) / 1000

        combined_score = db_score + np.mean(normalized_residuals) * value_weight
        return combined_score
```

Listing 10.1: Python code for MDSTC-DBSCAN algorithm

10.2 Code MDST-DBSCAN

```
import numpy as np
import pandas as pd
from mpl_toolkits.mplot3d import Axes3D
import matplotlib.pyplot as plt

class MDSTDBSCAN:
    def __init__(self, eps, eps2, eps3, minpts):
        self.eps = eps
        self.eps2 = eps2
        self.eps3 = eps3
        self.minpts = minpts

    def run(self,x, y, time, value):
        self.x = x
        self.y = y
        loc = np.column_stack((self.x, self.y))
```

```
17          self.value = value
18          min_date = pd.to_datetime(time.min())
19          self.time = (time - min_date).dt.days
20
21          n = loc.shape[0] #length
22
23          classn = np.zeros(n, dtype=int) #number of points in each
    cluster
24          cluster_mark = np.zeros(n, dtype=int) #which cluster
25          cluster_b = np.zeros(n, dtype=bool) #which points have already
     been added to a cluster
26          cn = 0 #cluster number
27
28          for i in range(n):
29              unclass = np.where(cluster_mark < 1)[0] #all points that
    are not within a cluster yet
30
31              # Creating distance
32              a = np.array([loc[i, 0], loc[i, 1]])
33              a = np.tile(a, (n, 1))  # Repeat the values for each row
    in loc
34              fordist = np.column_stack((a, loc)) #stacking current
    point with all other points that are not assigned to cluster
35              idist = np.abs(np.sqrt((fordist[:, 0] - fordist[:, 2])**2
    + (fordist[:, 1] - fordist[:, 3])**2)) #euclidean distance
36              forvaluedist = np.column_stack((np.repeat(self.value[i], n
    ), self.value))
37              ivaluedist = np.abs(forvaluedist[:, 0] - forvaluedist[:,
    1])
38              fortime = np.column_stack((np.repeat(self.time[i], n),
    self.time))
39              itimedist = np.abs(fortime[:, 0] - fortime[:, 1])
40              if cluster_mark[i] == 0: #if point not in a cluster yet
41                      #is number of points within distance threshold for
     all attributes greater than minpts?
42                      reachables = np.intersect1d(unclass[idist[unclass]
     <= self.eps],
43                                                  unclass[itimedist[
    unclass] <= self.eps2])
44                      reachables = np.intersect1d(reachables, unclass[
    ivaluedist[unclass] <= self.eps3])
45                      if len(reachables) + classn[i] < self.minpts:
46                          cluster_mark[i] = -1 #if the number is too
    small, point is assigned as noise
47                      else:
48                          cn += 1 #point assigned to cluster
49                          cluster_mark[i] = cn
50                          cluster_b[i] = True #part of cluster
51                          reachables = np.setdiff1d(reachables, i)
52                          unclass = np.setdiff1d(unclass, i) #
    neighboring points = reachables, removed from unclass
53                          classn[reachables] += 1 #belong to the newly
    formed cluster
```

```
54
55                      while len(reachables) > 0: #repeated for all
    neighboring points
56                          reachables = reachables.astype(int) # cast
     reachables to int
57                          cluster_mark[reachables] = cn
58                          neighbors = reachables
59                          reachables = np.array([])
60
61                          for i2 in range(len(neighbors)):
62                              j = neighbors[i2]
63
64                              # Create again when cluster is
    expanding
65                              b = np.array([loc[j, 0], loc[j, 1]])
66                              jfordist = np.column_stack((np.tile(b,
     (loc.shape[0], 1)), loc))
67                              jdist = np.sqrt((jfordist[:, 0] -
    jfordist[:, 2])**2 +
68                                              (jfordist[:, 1] -
    jfordist[:, 3])**2)
69                              jforvaluedist = np.column_stack((np.
    repeat(value[j], n), value))
70                              jvaluedist = np.abs(jforvaluedist[:,
    0] - jforvaluedist[:, 1])
71                              jfortime = np.column_stack((np.repeat(
    self.time[j], n), self.time))
72                              jtimedist = np.abs(jfortime[:, 0] -
    jfortime[:, 1])
73
74                              jreachables = np.intersect1d(unclass[
    jdist[unclass] <= self.eps],
75                                                           unclass[
    jtimedist[unclass] <= self.eps2])
76
77                              jreachables = np.intersect1d(
    jreachables, unclass[jvaluedist[unclass] <= self.eps3])
78
79                              if len(jreachables) + classn[j] >=
    self.minpts:
80                                  cluster_b[j] = True
81                                  cluster_mark[jreachables[
    cluster_mark[jreachables] < 0]] = cn
82                                  reachables = np.setdiff1d(
    reachables, jreachables)
83                                  unclass = np.setdiff1d(unclass,
    jreachables)
84                                  classn[jreachables] += 1
85                                  reachables = np.union1d(reachables
    , jreachables)
86                                  neighbors = np.union1d(neighbors,
    jreachables)
87          return cluster_mark, cluster_b
```

```
88      def plot(self, cluster_mark, cluster_b):
89          # Define colors for each cluster
90          colors = ['r', 'b', 'g', 'c', 'm', 'y', 'k', 'w']
91
92          # Get unique cluster labels
93          labels = np.unique(cluster_mark)
94          n_clusters = len(labels)
95
96          # Create figure and 3D axes
97          fig = plt.figure(figsize=(10, 8))
98          ax = fig.add_subplot(111, projection='3d')
99
100         # Iterate over each cluster
101         for i in range(n_clusters):
102             # Get indices of points in the current cluster
103             idx = np.where(cluster_mark == labels[i])[0]
104
105             # Get x, y, z, and size values for the points in the
    current cluster
106             x = self.x[idx]
107             y = self.y[idx]
108             z = self.time[idx]
109             s = self.value[idx]
110
111             # Plot the points with the appropriate color and size
112             ax.scatter(x, y, z, c=colors[i % len(colors)], alpha=0.5)
113
114         # Set labels for the axes
115         ax.set_xlabel('Longitude')
116         ax.set_ylabel('Latitude')
117         ax.set_zlabel('Days from Start Date')
118
119         # Show the plot
120         plt.show()
```

Listing 10.2: Python code for MDST-DBSCAN algorithm

Eidesstattliche Erklärung

Ich versichere:

- dass ich die Masterarbeit selbstständig verfasst, andere als die angegebenen Quellen und Hilfsmittel nicht benutzt und mich auch sonst keiner unerlaubten Hilfe bedient habe.
- dass alle Stellen, die wörtlich oder sinngemäß aus veröffentlichten und nicht veröffentlichten Publikationen entnommen sind, als solche kenntlich gemacht sind.
- dass ich dieses Masterarbeitsthema bisher weder im In- noch im Ausland (einer Beurteilerin/ einem Beurteiler zur Begutachtung) in irgendeiner Form als Prüfungsarbeit vorgelegt habe.
- dass diese Arbeit mit der vom Begutachter beurteilten Arbeit übereinstimmt.

Datum Unterschrift