

MASTERARBEIT | MASTER'S THESIS

Titel | Title

Introducing Spatial Heterogeneity via Regionalization Methods
in Machine Learning Models for Geographical Prediction: A
Spatially Conscious Paradigm.

verfasst von | submitted by
Lukas Bögl BSc

angestrebter akademischer Grad | in partial fulfilment of the requirements for the degree of
Master of Science (MSc)

Wien | Vienna, 2024

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:

Masterstudium Kartographie und Geoinformation

Betreut von | Supervisor:

Ass.-Prof. Dr. Ourania Kounadi BSc MSc

Contents

List of Figures	v
Acknowledgements	vii
Abstract	ix
Kurzfassung	xi
1. Introduction	1
1.1. Background	1
1.2. Problem Statement	3
1.3. Main Objective and Research Questions	4
1.3.1. Research questions	4
1.3.2. Methodology / Research Design	5
1.3.3. Limitations	5
1.4. Related Work	6
2. Theoretical Framework	7
2.1. Fundamentals in Space	7
2.1.1. Conceptualising Space	7
2.1.2. First Law of Geography	8
2.1.3. On the validity of TFL	9
2.2. Properties of Spatial Data	10
2.2.1. Spatial Autocorrelation	10
2.2.2. Spatial Heterogeneity	12
2.2.3. Scale and the MAUP	13
2.3. Foundations for Modeling Spatial Phenomena	14
2.3.1. Spatial Analysis	14
2.3.2. Global versus Local Statistics	15
2.3.3. Spatial Weight Matrices	15
2.4. Spatial Statistical Methods	17
2.4.1. Spatial Modeling & Geographically Weighted Regression	17
2.4.2. Regionalization	18
2.4.3. Terms & Concepts in Regionalization	19
2.4.4. Regionalization Methods	23
2.5. Machine Learning	25
2.5.1. General Remarks on Machine Learning	25

2.5.2.	Machine Learning Algorithms for Regression	26
2.5.3.	Random Forests	27
2.6.	Spatially Conscious Machine Learning	29
2.6.1.	Spatial Machine Learning	29
2.6.2.	Geographically Weighted Random Forest	30
2.7.	Performance Evaluation	31
2.7.1.	Mean Absolute Error	31
2.7.2.	Mean Squared Error and Root MSE	32
2.7.3.	R-squared	32
2.7.4.	Statistical Testing	32
2.7.5.	Spatial Autocorrelation in Residuals	33
2.7.6.	Metrics for Regionalization	33
3.	Methodology	35
3.1.	Software / Programming Languages	35
3.1.1.	Python	35
3.1.2.	R	37
3.1.3.	QGIS	38
3.1.4.	Notepad++	38
3.1.5.	GitHub	38
3.2.	Area of Interest	38
3.3.	Data	40
3.4.	Data exploration	40
3.5.	Data Preprocessing	45
3.6.	Workflow	46
3.6.1.	Random Forest	46
3.6.2.	Geographically Weighted Regression	46
3.6.3.	Geographically Weighted Random Forest	47
3.6.4.	Regionalized Random Forest (RegRF)	47
3.6.5.	Comparison of the workflows and models	48
4.	Results and Discussion	51
4.1.	Observations while modeling	51
4.2.	Performance evaluation	53
4.2.1.	Random Forest	53
4.2.2.	Geographically Weighted Regression	54
4.2.3.	Geographically Weighted Random Forest	56
4.2.4.	Regionalized Random Forest (RegRF)	57
4.3.	Performance Comparisons	63
4.3.1.	Comparing Variations of RegRF	63
4.3.2.	Comparing RegRF to Benchmark Methods	65
4.4.	Answering the Research Questions	69
5.	Conclusion	73

Bibliography	75
A. Appendix	85
A.1. Code	85
A.1.1. Random Forest	85
A.1.2. Geographically Weighted Regression	89
A.1.3. Geographically Weighted Random Forest	95
A.1.4. RegRF - WARD	100
A.1.5. RegRF - AZP	111
A.1.6. RegRF - KMEANS	122
A.1.7. RegRF - SKATER	132
A.1.8. RegRF - MAXP	144
A.2. Experiment Results tables	156
A.2.1. Complete table of performance metrics	157
A.2.2. "Optimization" of WARD and AZP	159

List of Figures

2.1. Types of spatial autocorrelation (own creation, adapted from Grekousis (2020, 213))	11
2.2. Exemplary study area	16
2.3. Rook's Case (left) and Queen's Case (right) for cell E	16
2.4. Dendrogram (own creation, adapted from Grekousis (2020, 293))	20
2.5. Single and complete linkage; adapted from Raschka and Mirjalili (2017, 364).	22
2.6. Illustration of a simple Decision Tree	28
2.7. ML with spatial data; simplified from Nikparvar and Thill (2021, 8).	29
3.1. Overview Map of California	39
3.2. The visibly gridded arrangement of the points	41
3.3. Distribution plot for the dependent variable	42
3.4. Distribution plots for the independent variables	43
3.5. Map depicting median house value	44
3.6. Point displacement, screenshots from QGIS	45
3.7. RegRF Workflow	49
4.1. LISA - Clusters of Residuals for Random Forest (Split 2 / Small Dataset)	54
4.2. LISA - Clusters of Residuals for GWR (Split 3 / Small Dataset)	55
4.3. LISA - Clusters of Residuals for GWRF (Split 2 / Small Dataset)	56
4.4. LISA - Clusters of Residuals for RegRF (WARD / Split 2 / Big Dataset)	58
4.5. LISA - Clusters of Residuals for RegRF (WARD / Split 1 / Small Dataset)	58
4.6. LISA - Clusters of Residuals for RegRF (AZP / Split 1 (left) + 2 (right) / Small Dataset)	59
4.7. LISA - Clusters of Residuals for RegRF (SKATER / Split 1 / Small Dataset)	60
4.8. LISA - Clusters of Residuals for RegRF (KMEANS / Split 2 / Small Dataset)	61
4.9. LISA - Clusters of Residuals for RegRF (MAXP / Split 3 / Small Dataset)	62
4.10. LISA clusters of residuals (Split 3, Small Dataset): Top left: RF; Top right: GWR; Bottom left: GW-RF; Bottom right: RegRF (WARD_CN)	68

Acknowledgements

I would like to express my deepest gratitude to my supervisor, Dr. Ourania Kounadi, for the invaluable guidance, support, and expertise provided throughout the course of this research. Your mentorship was crucial to the success of my project, and I am deeply appreciative of your unwavering patience and encouragement.

I am also very grateful to my colleagues in the Digital Geography working group for their collaboration and the stimulating discussions that have contributed significantly to my research work. I extend additional thanks to those who contributed during our group meetings—your perspectives were crucial. I also want to express my gratitude to my friends outside the university; your companionship during my studies has created lasting memories. Thank you all for accompanying me through the challenges of this journey and for your steadfast support.

I must extend my heartfelt thanks to my parents, Ruth and Hans, for their unconditional love and support. Your belief in my abilities and constant encouragement have been an important source of strength and motivation from the beginning of my academic journey to this very day.

Special thanks to my girlfriend, Luisa, for her love, understanding, and support. You have been my rock during the ups and downs of not only this thesis but throughout all my studies. Your presence and encouragement have made all the difference.

This thesis would not have been possible without each of you. Thank you all for being part of this journey.

Abstract

Recent years have seen a rapid increase in the capabilities of machine learning models. In this realm, this thesis aims to present a novel approach for predictive modeling which is able to take spatial heterogeneity into account, as well as evaluate various approaches to solving this problem. This is done by introducing regionalization methods in a preprocessing step of the practical workflow. Five different regionalization methods are tested: WARD, AZP, KMEANS, SKATER, and MAXP, which are applied to two datasets of varying size, which is another aspect of interest to this research. The dataset in use is the "California Housing Dataset", and the dependent variable is the median house price in an area. The proposed approach is termed *RegRF*, which can be further divided into three modeling strategies: RegRF_CN, RegRF_SEP, and RegRF_CN-W. The first strategy, RegRF_CN, is a conventional Random Forest with the addition of using the region number as an additional independent variable. RegRF_SEP creates a separate model for each region, that uses only the data from that region. RegRF_CN-W is a mix of both approaches: It creates a separate model for each region, but with the distinction that it uses all of the training data from the other regions. The region-specificity stems from putting more weight on the training data from the region the model is being trained for. The weight depends on the size of the region, meaning that bigger regions will be weighted more heavily. RegRF is compared to three established predictive modeling methods: Random Forest, Geographically Weighted Regression, and Geographically Weighted Random Forest. The implementation of RegRF has shown its ability to significantly increase the performance of the predictive models in comparison to "non-spatial" Random Forest models (between 5% and 18% increase in R^2 values), while only taking a few seconds longer to compute. In some scenarios, RegRF was able to compete with the well established Geographically Weighted Regression (between 1% better and 4% worse judging from the R^2 values, depending on the situation), while only requiring a fraction of the computational effort. The third approach, Geographically Weighted Random Forest, was not able to be outperformed when modeling the smaller dataset (between 1% and 7% decrease in R^2 values), but in this work this method was not able to finish computation for the larger scale dataset. These results clearly show the potential for the performance of machine learning models to increase when taking spatial phenomena into account. Given the fact that RegRF is not explicitly optimized for the data at hand, further improvements regarding its performance are possible.

Kurzfassung

In den letzten Jahren haben sich die Fähigkeiten von Modellen des maschinellen Lernens rasant verbessert. Diese Arbeit zielt darauf ab, einen neuartigen Ansatz für Vorhersagemodelle vorzustellen, wodurch diese in der Lage sind, räumliche Heterogenität zu berücksichtigen, sowie verschiedene Ansätze zur Lösung dieses Problems zu bewerten. Dies geschieht durch die Inklusion von Regionalisierungsmethoden in einem Vorverarbeitungsschritt des praktischen Teils dieser Arbeit. Es werden fünf verschiedene Regionalisierungsmethoden getestet: WARD, AZP, KMEANS, SKATER und MAXP, die auf zwei Datensätze unterschiedlicher Größe angewandt werden, was ein weiterer interessanter Aspekt dieser Untersuchung ist. Der verwendete Datensatz ist das „California Housing Dataset“, die abhängige Variable ist der Median des Immobilienpreises in einem Gebiet. Der vorgeschlagene Ansatz wird als *RegRF* bezeichnet, der sich in drei Modellierungsstrategien unterteilen lässt: *RegRF_CN*, *RegRF_SEP*, und *RegRF_CN-W*. Die erste Strategie, *RegRF_CN*, ist ein herkömmlicher Random Forest mit dem Zusatz, dass die Regionsnummer als zusätzliche unabhängige Variable verwendet wird. *RegRF_SEP* erstellt für jede Region ein separates Modell, das nur die Daten aus dieser Region verwendet. *RegRF_CN-W* ist eine Mischung aus beiden Ansätzen: Es wird für jede Region ein eigenes Modell erstellt, wobei jedoch alle Trainingsdaten der anderen Regionen verwendet werden. Die Regionenspezifität ergibt sich daraus, dass die Trainingsdaten aus der Region, für die das Modell trainiert wird, stärker gewichtet werden. Die Gewichtung hängt von der Größe der Region ab, was bedeutet, dass größere Regionen stärker gewichtet werden. *RegRF* wird mit drei etablierten vorhersagenden Modellierungsmethoden verglichen: Random Forest, Geographically Weighted Regression und Geographically Weighted Random Forest. Die Implementierung von *RegRF* hat gezeigt, dass es die Leistung der Vorhersagemodelle im Vergleich zu „nicht-räumlichen“ Random-Forest-Modellen erheblich steigern kann (zwischen 5% und 18% Steigerung der R^2 -Werte), während die Berechnung nur wenige Sekunden länger dauert. In einigen Szenarien konnte *RegRF* mit der bewährten Geographically Weighted Regression konkurrieren (zwischen 1% besser und 4% schlechter, je nach Situation, gemessen an den R^2 -Werten), während es nur einen Bruchteil des Rechenaufwands erforderte. Der dritte Ansatz, Geographically Weighted Random Forest, konnte bei der Modellierung des kleineren Datensatzes nicht übertroffen werden (zwischen 1% und 7% schlechtere R^2 -Werte), aber in dieser Arbeit war diese Methode nicht in der Lage, die Berechnungen für den größeren Datensatz abzuschließen. Diese Ergebnisse zeigen deutlich, dass die Leistung von Modellen des maschinellen Lernens gesteigert werden kann, wenn räumliche Phänomene berücksichtigt werden. In Anbetracht der Tatsache, dass *RegRF* nicht explizit für die vorliegenden Daten optimiert ist, sind weitere Verbesserungen hinsichtlich seiner Leistung möglich.

1. Introduction

The opening chapter aims to introduce the topic from different points of view. At the beginning of the chapter, the background of the topic is presented, so a broader range of readers are able to follow the concepts and thoughts presented in the further course of the thesis. After illuminating the background, a short overview of the historical evolution of the methods and concepts used in spatial analytics is presented. It should be mentioned that this evolution is closely connected to the theoretical base of this scientific field, which is elaborated in more detail in chapter 2 "[Theoretical Framework](#)". In the section following this, reasons for why further examination of the topic in a scientific context is necessary are given, framing the statement of a problem the thesis aims to solve. Section [1.3](#) describes the main objective to be achieved by answering various research questions throughout the thesis. These questions are also outlined in this section, as well as limitations the research faces. Afterwards, related work is discussed. This mainly includes descriptions of current works, as a historical overview is already given at the beginning of the chapter.

1.1. Background

The centerpiece for the majority of research on spatial statistical analysis is Tobler's First Law of Geography: *"Everything is related to everything else, but near things are more related than distant things."* (Tobler [1970](#), 236) This train of thought has been valid for the past 50 years and is a key concept in the world of spatial analysis. As straightforward as this law sounds, as complicated its quantification and its statistical description is. In non-spatial statistics, a large share of methods for analysis assumes that observations in a dataset are randomly distributed and independent from each other. This is an assumption that is often violated when working with spatial data, which of course has consequences for the analysis of such (Rogerson [2021](#), 1). The reason for this phenomenon is a concept in spatial analytics called *spatial heterogeneity or spatial non-stationarity*, which means a nonuniform distribution of observations and/or their values in space, a varying mean across spatial units, and the dependence of values of a variable on its location (Grekousis [2020](#), 13f; Haining [2010](#), 205f). Because of this, spatial statistical methods that account for these phenomena need to be used (Anselin [1995](#), 93f).

The methods in use today have a long history that has its roots in research that was carried out by statisticians looking for ways to deal with problems that arose from trying to determine the correlation between values that are close in time or space. Back then, there was no difference made between values following each other in space or time, but as space is more complicated due to its second dimension, the ideas were demonstrated

1. Introduction

using correlation caused by time. The approach taken by Student (1914, 179f) was to "correlate the 1st, 2nd, 3rd...nth differences between successive values of our variable with the 1st, 2nd, 3rd...nth differences between successive values of the other variable."

Neprash (1934, 167f) stated that researchers in contact with correlating variables have been leaning towards the assumption that if there is co-existence of two characteristics in a spatial unit, there also is some kind of causal relationship between them. The ability to interpret this correlation between variables distributed in space was very limited and a phenomenon whose existence just had to be accepted. This is due to the fact that data involved in such analyses stand in conflict with two crucial requirements of a correct use of correlation techniques. Specifically, these are the homogeneity of the distribution of the traits within a particular area and the independence of the spatial units themselves.

In the same year, Stephan (1934, 4) pointed out that there is no foundation for assuming that an incident in a census tract is independent from an incident in a nearby census tract, especially if these tracts happen to be neighbours. A powerful quote from this paper is: *"Data of geographic units are tied together, like bunches of grapes, not separate, like balls in an urn"*.

Fisher (1935, 66) mentions that when designing an experiment, for which researchers can choose between several areas of land, the choice should fall on areas that seem homogenous in regard to the soil's structure, for example. *"After choosing the area we usually have no guidance beyond the widely verifiable fact that patches in close proximity are commonly more alike [...] than those which are far apart"*.

The research in this phase was characterized by recognizing the concept of spatial autocorrelation and trying to describe its meaning. After that, researchers shifted their focus on the quantification of spatial autocorrelation, with the publishing of works whose methods are still in use today (Griffith 2012, 3f).

This period saw the publication of various important works, three of which are to be described as especially influential. The first one was published by Moran (1950), in which he introduced the Moran Coefficient (MC), better known as *Moran's I*, a measure of spatial autocorrelation. It measures the degree of similarity between values of a variable for locations close to each other compared to locations that are further away. Another metric that was introduced during that time is called the *Geary Ratio*, which is also a metric for spatial autocorrelation. The difference lies in the fact that the Geary Ratio compares the spatial distribution of a variable with a theoretical distribution of that same variable (Geary 1954). According to Griffith (2012, 3), the most important work stems from Cliff and Ord, as they further developed these two statistics to make hypothesis testing possible. The mentioned concepts will be elaborated in more detail in section 2.2.

In the years after these publications, plenty of researchers made influential contributions to the research field, the most noteworthy one being "Geographically Weighted Regression: A Method for Exploring Spatial Nonstationarity" by Brunsdon et al. (1996). They suggested using different regression models that are altered depending on the place they are deployed in. The resulting local models were a big improvement over previous works that could only measure trends in parameter variation over space (Brunsdon et al. 1996, 282; Casetti 1972, 81f).

As stated by Student (1914, 179f), many of these methods were "tedious in the extreme" to carry out with the technology available at the respective times. Various theoretical and practical works from the last decades show how the use of computers and artificial intelligence/machine learning in the form of decision trees and random forests can solve this problem. Exemplary mentions are the works of Hengl et al. (2018), Georganos et al. (2019) and Talebi et al. (2022), more related work is described in section 1.4.

1.2. Problem Statement

The superficial problem overshadowing all of the research done in spatial analytics is the ubiquitous uncertainty surrounding spatial data. Important technical terms commonly used to describe this are *spatial heterogeneity* and *spatial non-stationarity*, which are synonymous in common use (Goodchild 2004, 302). They can be described as the situation, in which the statistical properties of data (e.g. mean), are dependent on the absolute location of observations (Fotheringham 2009, 400). To still be able to analyse spatial data, the goal of researchers is to build statistical models that are able to take these phenomena into account. Prior to the works of Cliff and Ord, the former one being a geographer and the latter one being a statistician, works from researchers that applied standard statistical techniques have frequently been criticized by other researchers for using inappropriate techniques for quantifying phenomena in their respective fields (Griffith 2012, 13).

The emergence of numerous works in recent years shows the dynamics surrounding this field of research. As elaborated in more detail in the following chapter, a multitude of approaches have been considered, tried and evaluated. The method proposed in this thesis aims to add to this process by implementing a combination that has, at the time of writing this thesis, not been deployed in a scientific context. Adding to this it is important to keep the following quote, originally applied to non-machine learning statistical models, in mind: "all models are wrong but some are useful" (Box 1979, 2). Consequently, the search for "more useful" ones will probably never come to an end. As a result of this, thinking of new approaches and exploring all the possibilities current technology provides is necessary to maybe be able to improve the performance of machine learning models. By executing a novel approach I hope to make a meaningful contribution to this discourse.

Finding spatial phenomena and modeling them also leads to my personal motivation for working on this topic. I have always found it fascinating to think about the reasons why some places are so different from others. This might have obvious reasons in some cases, especially when these places are in different countries or on different continents. But more often than not, I have found big distinctions between places that should be more alike, but somehow they are not. Having the opportunity to explore the phenomena present in space using Python in a scientific context is greatly appreciated and can be seen as another motivation factor. With the proposed approach I hope to contribute to a better understanding of the underlying processes.

1.3. Main Objective and Research Questions

The main objective of the thesis is to introduce a novel method that accounts for spatial heterogeneity in data during the modeling process for geographical prediction. As made clear in the previous sections, as well as in section 1.4, there are various approaches to deal with this phenomenon, the method proposed in this thesis is based on *regionalization*. In brief, regionalization means the procedure of grouping "a large number of spatial objects into a smaller number of subsets of objects, which are internally homogeneous and occupy contiguous regions in space" (Assunção et al. 2006, 797). Regionalization will be elaborated in detail in chapter 2.4.2. When regionalization is combined with machine learning algorithms it implies either the design of multiple models, which are deployed on the different regions in the dataset, or the creation of an additional predictor variable in the form of a region number. The resulting models will be used to predict the values of the respective target variables in a geographical context. With the investigation of the predictive performance of the proposed approach, I hope to make a useful contribution to the scientific field of spatial analysis. More specifically, I would like to find out how good predictions made with a combination of Random Forest and regionalization are compared to benchmark methods like Random Forest, statistical methods like Geographically Weighted Regression (Brunsdon et al. 1996), and novel spatially conscious approaches such as Geographically Weighted Random Forest (Georganos and Kalogirou 2022). I aim to do this by first conducting an extensive theoretical review of the underlying spatial statistical concepts and the actual methods used for modeling the spatial structure of a study area. This theoretical examination is necessary to be able to correctly conceptualize the models on one hand, and to correctly interpret the results on the other hand.

The practical use of my thesis will depend greatly on the comparative predictive performance. If the model is able to perform better than benchmark models, widespread implementation in institutions, companies and academic research institutes that have a connection to spatial data science is possible. This could moreover lead to improved decision making by policymakers which could be beneficial for the public. A practical example for this would be a model of resource use for an area. Having a better model could prevent policymakers from missing the point in time where it is necessary to adopt measures that prevent any form of shortage, thus clearly benefiting the general public. I term the proposed approach *Regionalized Random Forest (RegRF)*.

1.3.1. Research questions

1. How does the predictive performance of "Regionalized Random Forest" vary by different regionalization techniques?
2. How does "Regionalized Random Forest" perform compared to benchmark methods, either these being Spatial Statistical methods or Machine Learning algorithms?
3. How does "Regionalized Random Forest" perform compared to existing spatially conscious ML approaches such as the GW-RF?

1.3. Main Objective and Research Questions

Working questions:

What benchmark models for geographical prediction exist and how do they work?

What regionalization methods are there and how do they work?

What metrics can be used to evaluate the performance of spatial models?

1.3.2. Methodology / Research Design

The aforementioned questions will be elaborated with two kinds of methods. Theoretical questions demand a literature review to be carried out, which will be useful for numerous reasons. First off, it is of utmost importance to understand the underlying theoretical concepts and spatial statistical methods as good as possible to be able to correctly design the modeling workflow and carry out the practical analysis. Furthermore this is also necessary to appropriately interpret the results. The mentioned practical analysis leads to the second method applied in the thesis, which is the creation of the described models with Python. More specifically, I will use various libraries to define regions in a dataset, before applying the Random Forest method to each of these regions to be able to model the dataset as good as possible. In the second modeling approach, the region number will be used as an additional independent variable. The third approach creates a separate model for each region, but with the distinction that it uses all of the training data from the other regions. The region-specificity stems from putting more weight on the training data from the region the model is being trained for. The weight depends on the size of the region, meaning that bigger regions will be weighted more heavily. Subsequently, there will be a vast number of models with a multitude of results and performance indicators, which will be tested for significance and evaluated in great detail. The libraries used will be standard ones like pandas, as well as more specific ones such as scikit-learn for the machine learning tasks and pysal/spopt for the regionalization tasks. More details on the methods are given in [chapter 3](#).

An important aspect of such research is the data used for modeling. The modeling workflow will be carried out using the “California Housing” dataset. It contains longitude and latitude, as well as attributes like the number of rooms, median income of the households in the block, among others. It has become a standard dataset for spatial analysis, shown by its use in a multitude of publications, for example by Klemmer et al. ([2019](#)) and Liu et al. ([2022](#)).

1.3.3. Limitations

As the research objective is specified quite narrowly, there are not a lot of limitations this research faces, apart from general ones that are quite obvious. One of these is "endless" trial and error in hope of creating better models. The proposed method will be implemented and compared to benchmark methods. There will be no extensive search for a different approach, instead the result will be subject to interpretation and discussion as it is. The reason for this lies in the fact that this is a thesis underlying time constraints.

It is noteworthy that reproducibility of this research is a prerequisite, which is why freely available software and data need to be used. Successively, the modeling will be

1. Introduction

carried out with freely available software, for which Python is a very good fit. The limitations regarding the data used are definitely stronger, as there are not as many datasets with spatial features available for free as one might think. Of course, it would be possible to create these on one's own (e.g. merging different datasets from Eurostat), but the preferred approach is to use data that have some kind of history or validation in the scientific field.

1.4. Related Work

In the field of human geography - more specifically population modeling, Georganos et al. (2019) and Georganos and Kalogirou (2022) gave their Machine Learning models spatial consciousness by introducing a method called "Geographical Random Forest". It is based on the idea that spatial phenomena might be modeled better with an ensemble of local models, instead of one global model. They created an algorithm that creates a model for each data point that is based on data close to that point, instead of one model for the entire study area. It is important to mention that this algorithm is not only applicable to the field of population modeling. It can be applied in various fields, as has been the case in recently published works in health science (Quiñones et al. 2021; Lotfata et al. 2022), geomorphology (Quevedo et al. 2021) and ecology (Santos et al. 2019). This technique will be subject to more detailed elaboration in its own subsection later in this thesis (2.6.2).

Of course, there are other approaches to the use of Machine Learning in spatial modeling. A prominent example for this stems from Hengl et al. (2018), where buffer distances from locations are used to incorporate the effects of geographical proximity into the model. The exemplary application of this method was carried out in the field of pedology.

Another technique was introduced by Mueller et al. (2018). They carried out a cluster analysis whose results were incorporated into the creation of the models to estimate health insurance status, the focus was put on localizing the model procedure in order to add performance and increase the insights that can be generated by the process.

Stojanova et al. (2011) proposed "Predictive Clustering Trees", a combination of clustering and prediction. This method identifies clusters of similar observations, before designing a model for prediction for each of the clusters. These two can be seen as the most closely related works, with the important difference being the use of clustering instead of regionalization.

Talebi et al. (2022) proposed a sophisticated approach that produces an algorithm that is able to incorporate heterogeneity, spatial dependencies and complex spatial patterns through the use of vectorised patterns from spectral information. Spectral information instantly hints at the practical application of this method, which was done in remote sensing. More specifically, the researchers tried to automate geology mapping in order to carry out geochemical prediction.

2. Theoretical Framework

Chapter two aims to outline the theoretical foundation of the research. It starts with conceptualizing what *space* actually is and traces the path to how Tobler's law came into existence. After grasping these concepts and thoughts, the chapter moves on to the properties distinguishing spatial from non-spatial data: spatial autocorrelation, spatial heterogeneity and scale. Afterwards the foundations for modeling these phenomena are laid out, before describing how these can be used for spatial statistical analyses. Subsequently, the use of machine learning is introduced, both in non-spatial and spatial analysis. The chapter concludes by giving details on the evaluation of machine learning models by discussing the metrics of their performance, as well as techniques for significance testing.

2.1. Fundamentals in Space

This section is about conceptual foundations that shape our understanding of space. As this is a subject not only discussed in geography, some of the thoughts have a philosophical character, while also being influenced by physical sciences. First, the most important concepts of space for this research are presented, before moving on to Tobler's First Law and its validity.

2.1.1. Conceptualising Space

As proposed by Curry (1998, 24), concepts of space follow a continuum. On one end of it, there are concepts in which "the relationship among objects is strictly contingent." In the concepts on the other end of the continuum "the objects in space have very strong, even necessary and intrinsic relations with one another and the space in which they are located." This leads to the understanding and treatment of space as either an object-filled container, or as the result of the relationships these objects have with each other. Some perceptions of space have their origin in physics, where space and time have been researched with great effort. From this point of view, there are measurable physical laws that underlie space. These perceptions have been used in quantitative geographic research as well, more specifically on the search for laws for space, which would help to shine light on the logics behind human activity in space (Shields 1997). In these conceptions, space is "a kind of absolute grid, within which objects are located and events occur." (Curry 1995, 5) A valuable formulation from Gleeson (1996, 390) on this concept of space is that it is "an absolute container of static, though movable, objects and dynamic flows of behaviour." The relational approach is the one underlying this research.

2. Theoretical Framework

This absolutist point of view has been opposed by the relational approach, whose followers believe that space is organized through the actions of humankind, and that these actions in turn impart significance on space. They oppose the argument of space consisting of objects, as they argue that space is produced from social, economic, political and cultural relationships. In this concept, it is "constituted through social relations and material social practices." (Massey 1994, 254) Analysis carried out within this concept is centered on uncovering the spatial variation of phenomena and the reasons for it. It wants to identify socio-spatial processes at various scales, and break down how they "produce and reproduce space." (Dodge and Kitchin 2001, 29) As pointed out by Curry (1995), a substantial amount of quantitative research claims to be based on the absolutist, measurable concepts, when the majority of it is effectively relational, because models are made from relative elements themselves. This claim is supported by thoughts from the domain of space-time physics: "space and time are dynamic quantities: when a body moves, or a force acts, it affects the curvature of space-time – and in turn the structure of space-time affects the way in which bodies move and forces act. Space and time not only affect but are also affected by everything that happens in the universe." (Hawking 1998, 38)

2.1.2. First Law of Geography

Tobler's first law of geography - often referred to as "TFL" - has been cited in and is a focal point in a multitude of works in the field of spatial analysis. It reads as follows: "Everything is related to everything else, but near things are more related than distant things." (Tobler 1970, 236). One might ask how he came to state this and what his thoughts behind it were, which is a valid scrutinization, given the broad application and the big impact this law had and still has (Goodchild 2004, 301). In the very publication, numerous trains of thought are presented that lead Tobler to stating his law, some of which will be described in the following paragraphs.

First it is important to look at the actual objective of his work, which was the creation of a statistical model for population growth, emphasizing the geographical distribution of this phenomena. It should also be pointed out that his goal was to create a model that is primarily instructional, meaning it is suited for pedagogic purposes, which he states multiple times over the course of the paper. At the very beginning he also mentions an assumption for the research, which is "everything is related to everything else", which would imply an infinitely complex model. He also notes that his superordinate goal is to avoid creating a model with complicated rules. It should have high success while being as simple as possible. It would be obvious to believe that these thoughts contradict each other, which he addresses by stating that just because "a process appears complicated is [...] no reason to assume that it is the result of complicated rules", which he substantiates with examples (Tobler 1970, 234).

Practically, he aims to achieve this by only including the "facts" of death, birth and migration, but not going into any details on the reasons behind these events. In the concrete case of the demographic model, the population is estimated based on previous populations only, disregarding factors like the job market and transportation, among

others. This leads to the fact that the only difference between cities in the model is the density of their population. Consequently, the "model attempts to relate population growth only to population in the immediately preceding time period." (Tobler 1970, 234f)

One might raise the question, if "everything is related to everything else", how can the model only be based on the mentioned factors? To address this, Tobler once again states that simplicity is one of the main objectives and seen as an advantage, especially in a pedagogic context. He stays true to this when outlining the model's evaluation, which is done by comparing maps from two different points in time - 1930 and 1940 - with the Pearson Correlation Coefficient, that also makes the formulation of a linear regression equation which can be used for prediction, possible. Using this equation, the 1940 population can be predicted as a linear function of the 1930 population, meaning it again violates his fundamental assumption of everything being related to everything else, by being based on just the population of that place at a previous point in time. Including everything means, when done consequently, basing the 1940 population on the 1930 population of all other possible places, which would lead to a massive multiple regression equation. To remain true to his goal of simplicity, instead of this, he invokes Tobler's first law (TFL) of geography: "everything is related to everything else, but near things are more related than distant things", acknowledging that this is ignorant of a huge portion of the world and more small-minded than the assumption he made earlier (Tobler 1970, 235f). This leads him to the final model, the estimation of a cell's population that has a length of 1.5 miles based on a linear function of this and neighboring cells in the foregoing time period (Tobler 1970, 237). Apart from this, in the original publication, no more attention is given to the effects TFL might have on the field of spatial analysis.

2.1.3. On the validity of TFL

Before going into further detail on these effects, it is worth to shortly discuss what a *law* is in a scientific context, for which the remarks made by Goodchild (2004, 300) serve as a foundation. In all spheres of science, the goal of research is to demystify phenomena by finding overarching guidelines, which is a process of simplification that leads to the statement of *principles*. Using principles is an established method in pedagogics, because they teach students things that are usually true, which is why they serve as the basis for the majority of scientific education (Goodchild 2004, 300). It has not been generally agreed upon what makes a principle a law, and the ideas about this differ across the domains of science. Historically, the discussion whether a law should be deterministic or not, has been the most intense. But when taking a more practical approach, the discussion about the difference between laws, principles, theories or norms might be irrelevant, at least in common use. On this Goodchild notes: "if a general statement is empirically valid, its value results from its simplicity, its elegance, and its utility in design and prediction, not from its title." (Goodchild 2004, 300)

The connection between pedagogic practice and simplicity regarding scientific laws made by Goodchild is especially interesting when recalling Tobler's initial objective, which was the creation of an *instructional* model, that is suited for use in a pedagogic context. To take this a step further, Tobler repeatedly mentioned the importance of creating a

2. Theoretical Framework

simple model, another parallel to Goodchild's characterization of a scientific law. Still, it is most likely farfetched to suggest that given the criteria Tobler imposed on his work, he already set himself up for "finding" a scientific law in TFL. More likely, and pointed out by both Goodchild (2004, 301) and Miller (2004, 284), it was the growing interest in spatial statistics and quantitative geography over the last decades that promoted TFL to the status it has today.

Of course, as TFL nowadays serves as the fundamental concept in the area of geostatistics (Goodchild 2004, 301), various researchers have made remarks on it, have validated it, and have also questioned it. In 2004, Miller (2004, 284) asked in a slightly provocative way: "How could a sentence justifying heuristic calculations in a crude urban growth simulation generate an icon now known as Tobler's First Law (TFL)? Why has this law resonated so strongly in geography?" It just appeared to be a generally agreeable proposal back in 1970, which has been consistent until today because the terms *near* and *related* are valuable notions at the base of spatial modeling and analysis. Until today TFL is still sensible, in great part because of the increased interest in geographic information science that led to more sophisticated methods for the examination of these principles (Goodchild 2004, 301). To check its validity, it is insightful to think of a real-world situation in which TFL is not true. In this situation, every room could be filled with all the world's climates and the wildest differences in topography all at once, it would be impossible to inhabit or characterize such a world, whereby any simplifications and descriptions of the world in a geographic context rely on TFL being valid. What this also shows is that "TFL is [...] thus obvious in hindsight, and surely no more than common sense", which is not an exclusion criterion for scientific laws. This is under no circumstances to be seen in a negative way, as it is exactly this simplicity - which has been mentioned so often - that has propelled TFL to enjoy the reputation it has today, by being so close to what humans see and experience every day (Goodchild 2004, 301f). Given how casually Tobler mentions the law in the paper, practically as a workaround to avoid an enormous multiple regression equation that the slow computers of the time might have had problems with, this evolution seems almost ironic (Miller 2004, 284).

2.2. Properties of Spatial Data

After familiarizing the abstract concepts and laws applied in spatial analysis, the following paragraphs describe the ubiquitous phenomena surrounding spatial data: *spatial autocorrelation*, *spatial heterogeneity*, and *scale*. The literature has reached a widespread consensus that these three properties are the defining characteristics of spatial data (Nikparvar and Thill 2021, 5).

2.2.1. Spatial Autocorrelation

Grekousis (2020, 208) defines spatial autocorrelation as "the degree of spatial dependency, association or correlation between the value of an observation of a spatial entity and the values of neighboring observations of the same variable", while noting that

spatial association and spatial dependence are also commonly used to represent spatial autocorrelation.

This definition was developed well after Cliff and Ord (1969) published their influential considerations regarding spatial autocorrelation, nonetheless the mentioned notation is a good compromise between simplicity, accuracy, and usefulness. By no means were they the first ones to describe it and propose methods for measuring it, this was done by Moran (1950) and Geary (1954), with mostly conceptual works from other researchers dating back even further (Getis 2010, 256). Still, the classic publication from Cliff and Ord (1969) has had the biggest influence of them all, as it was the starting point of the era of hypothesis tests through the creation of a statistical foundation for it (Griffith 2012, 4; Getis 2010, 256).

The connection between spatial autocorrelation and spatial dependence was - among others - pointed out by Fotheringham (2009, 398). He argues that these two terms are interchangeable, because their main point is the same: They both describe the case, in which "observations drawn from different locations are not independent of each other, with observations at locations nearer to each other being more similar [...] or less similar [...] than observations at locations farther apart." This means that for the measurement of spatial autocorrelation, it is necessary to describe the "relationship between the degree of similarity between observations and the distance separating them." This refers to the spatial dependence present in the dataset, from which a spatial weighting function relating the likeness of observations to the distance between them can be derived. Consequently, spatial autocorrelation measures the intensity and direction of the relationship, in simple words is the degree of spatial dependence (Fotheringham 2009, 399). When looking at Moran's I , values close to $+1$ indicate *positive* spatial autocorrelation, values close to -1 indicate *negative* spatial autocorrelation, with 0 indicating spatial randomness. Figure 2.1 illustrates these types with the values of Moran's I .

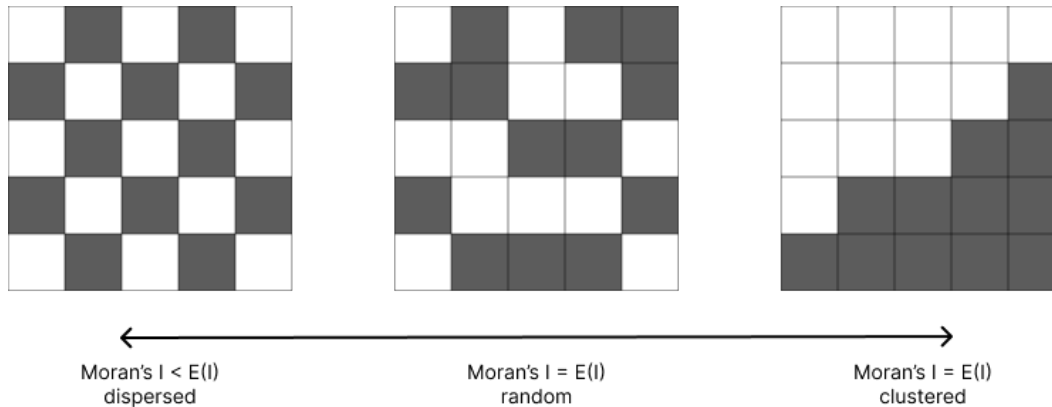


Figure 2.1.: Types of spatial autocorrelation (own creation, adapted from Grekousis (2020, 213))

The remarks made about spatial dependence and figure 2.1 might already make it obvious, but it has also been pointed out by Goodchild (2004, 302) that spatial dependence,

2. Theoretical Framework

the situation in which locations close to each other are more (or less) similar than the ones further away, is the foundation for TFL.

Statistics for measuring spatial autocorrelation are Moran's I (Moran 1950), Geary's C (Geary 1954) and Getis-Ord G (Getis and Ord 1992). What they have in common is that they provide one value for the entire study area, making them *global* measures which indicate the average spatial dependency in the study area. Interesting differences of this statistic in space that might exist are hidden when calculating a global statistic, which can be uncovered by calculation of *local* measures.

2.2.2. Spatial Heterogeneity

Spatial heterogeneity reflects the "variability or difference between the values of observations over space." (Dall'erba 2009, 683) This can be understood as nonuniformity of observations in space, a distinguishing characteristic between spatial and nonspatial data. These nonuniform effects can easily be observed in everyday life, an example for this being a highway or a river disrupting the continuity of a geographic variable, for example the composition of the soil (Grekousis 2020, 13).

In this context it is vital to also shed light on the term of *spatial (non)stationarity*. A classic assumption - that is probably not based on a lot of logical reasons - is that the relationships which are constructing a global statistic do not vary significantly over space. This assumption is termed spatial stationarity. As pointed out by a myriad of researchers, including Fotheringham (2009, 399f) and Rogerson (2021, 1), this is a very questionable assumption to make when analysing space, which is why *spatial nonstationarity* has been established. A meaningful quote from one of the most influential works from the research field reads "spatial nonstationarity is a condition in which a simple 'global' model cannot explain the relationships between some sets of variables." To combat this, we use local statistics that are able to take spatial nonstationarity into account (Brunsdon et al. 1996, 281).

To conclude on the concepts mentioned in the preceding paragraphs, it is helpful to recall some facts about them:

- Spatial autocorrelation measures the degree and intensity of spatial dependence (Fotheringham 2009, 398; Grekousis 2020, 208).
- Spatial heterogeneity and spatial nonstationarity are used practically synonymously (Goodchild 2004, 302).
- Spatial nonstationarity is the reason why local statistics are needed and useful (Fotheringham 2009, 399).
- Spatial dependence is the foundation for modeling spatial nonstationarity/heterogeneity (Fotheringham 2009, 398).

2.2.3. Scale and the MAUP

According to Sankey (2016, 1021), "spatial scale describes both the geographic extent and the resolution of a study".

The geographical extent or scale is typically set before starting the process of spatial analysis, as it has a strong influence on data gathering, the choice of methods, and how results will be interpreted. A common example is the representation of a city at varying scales. At a small scale (e.g., a whole country), a city is typically depicted as a point. When enlarging the scale to a regional level, this point usually turns into a polygon before being split into multiple polygons (in the form of census tracts for example). It is understood as the "classic" scale in geography, for example 1:10,000 for a city. The second notation of scale means the scale of analysis. It is tightly connected to the geographical scale, in fact the two are often used synonymously. Scale of analysis describes the shape and size of the area on which spatial statistical analysis is carried out on. In essence, it describes from which level spatial phenomena are seen, and can be seen "as the geographical extent that spatial [...] variability is analyzed through the construction of an appropriate spatial weights matrix." (Grekousis 2020, 12)

When carrying out analysis with areal data that has been aggregated in some way, for example analysis using census tracts, a problem that arises is the *modifiable areal unit problem* (MAUP). It states that the results of the analysis are very sensitive to how the units which the data refer to are designed (Fotheringham and Wong 1991, 1025). Designing the units differently would lead to different results, a very prominent and established example for this is the 2000 presidential election in the United States. Al Gore received more votes, but because of the design of counties George Bush still won the election. Were the counties designed differently, the outcome could have easily been the opposite. Generally, it is appropriate to say that the larger the areal units get, the more intense the aggregation of the data becomes, making generalization more prominent. When generalization exists, the therethrough aggregated values gravitate more towards the global mean, with deviations becoming milder. Of course, this leads to the loss of valuable information (Grekousis 2020, 13).

2.3. Foundations for Modeling Spatial Phenomena

The following section will be about the fundamental techniques that allow for the quantification of the aforementioned concepts and properties. To start, explanations on what spatial analysis is exactly about are given. Afterwards, local and global statistics are discussed, before moving on to outlining spatial weights matrices. All of these are necessary techniques and thoughts to grasp for understanding the methods described in the following section.

2.3.1. Spatial Analysis

Many studies are based on a count and/or measurement of objects or individuals. The widespread implementation of GIS made it easy to connect these casual data (e.g., socio-economic, gender) to locations (e.g., administrative units, buildings), thus transforming them to spatial data. This connection makes the analysis of these data more interesting, transforming it to *spatial analysis*. Through this, it is possible to create new insights which in turn allow for creation of knowledge (Grekousis 2020, 2). In this realm, there is no absolute definition of spatial analysis, which is why various points of view from highly esteemed researchers are presented:

- Spatial analysis is "the formal quantitative study of phenomena that manifest themselves in space" (Anselin 1989, 2).
- Spatial analysis describes "a set of techniques designed to find patterns, detect anomalies, or test hypotheses and theories based on spatial data" (Goodchild 2008, 200).
- Spatial analysis examines "how the physical environment and human activities vary across space – in other words, how these activities change with distance from reference locations or objects of interest" (Wang 2014, 27).

Application of spatial analysis leads to the possibility of uncovering patterns that would not have been found otherwise. An example for this is the identification of clustered incidence of an illness, which makes it possible to adopt measures that combat further spreading (Bivand et al. 2008). Grekousis (2020, 3f) distinguishes between different types of spatial analysis:

- *Spatial point pattern analysis* means the analysis of a set of data points to reveal which of these three states it can be characterized into: clustered, dispersed or random. In practice, this could mean the analysis of a set of points including all the events of a medical condition (e.g., stroke) patients have. Analysing this pattern might make it possible to find out about reasons behind the occurrence of the condition.
- In *spatial analysis for areal data*, data are compiled into areal units (e.g., administrative units). The subject of research is how areas close to each other interact

2.3. Foundations for Modeling Spatial Phenomena

with and relate to each other. As an example, one might ask if people with similar income are grouped in specific areal units or a combination of multiple units.

- *Geostatistical data analysis* (for continuous data) is a part of statistics that is comprised of the analysis and the modeling of *continuous field* variables. Typically, samples of a continuous phenomenon, like air temperature or pollution, are interpolated and thus calculated for the complete surface under investigation (O’Sullivan and Unwin 2010, 115).

In the following paragraphs, techniques which are necessary to carry out the mentioned types of analysis are elaborated.

2.3.2. Global versus Local Statistics

An important concept that needs to be addressed when carrying out research on spatial analysis is *local* vs. *global* statistics. The easiest way to make this distinction is with an example. Think of the mean air temperature of a country at any given time, where all the measurements from that country are averaged and summarized in one single value. This single value would be termed a *global* statistic. To make this statistic a *local* one, one possible way would be to look at a multitude of weighted means that only take neighboring values into account, or just look at all the individual measurements from which the *global* average was calculated in the first place (Fotheringham 2009, 399f).

This can also be applied to indicators of spatial association, first shown by Anselin (1995, 93). Global indicators, like Moran’s I , are hereby decomposed to local indicators of spatial association (LISA). This makes it possible to find local nonstationarity-pockets, as well as to find out how each location contributes to the global statistic, which in turn allows outliers to be uncovered. Local variations of spatial analysis give better insights to the relationships present in space, they are like a magnifying glass that helps to see more detail. In the absence of local statistics, all one can see is uniformity, with them patterns of relationships in space can be seen (Fotheringham et al. 2002, 25).

2.3.3. Spatial Weight Matrices

The aforementioned spatial relationships - to be precise the researcher’s understanding of them - are denoted in a spatial weights matrix, commonly termed W (Getis 2010, 260). As this matrix describes spatial dependence in a formal, numerical way, it is essential for modeling spatial phenomena (Getis and Aldstadt 2004, 91). Thereby it allows for incorporation of observations that are close in space into the analysis, thus taking TFL into account. The elements constructing W describe the closeness or contiguity of each other. The value is 0 if the observations do not have a meaningful connection, and unequal to 0 if they do (Kelejian and Piras 2017, 3). For this it is important to define the contiguity, a property defining if observations are near to each other. There are two common types of contiguity:

- Rook’s Case: Edges only; Polygons sharing a border (edge) are classified as neighbors.

2. Theoretical Framework

- Queen's Case: Edges and corners; Polygons sharing a border or a corner (node) are classified as neighbors (Grekousis 2020, 23).

To clarify, it is helpful to look at the following visualization in an exemplary study area (Figure 2.1).

A	B	C
D	E	F
G	H	I

Figure 2.2.: Exemplary study area

A	B	C
D	E	F
G	H	I

A	B	C
D	E	F
G	H	I

Figure 2.3.: Rook's Case (left) and Queen's Case (right) for cell E

It is important to mention that the situation pictured describes the simplest case possible, being based on contiguity and only taking *direct* neighbors into account, called *first-order neighbors*. The contiguity is called first-order contiguity when only directly contiguous features are considered. Second-order contiguity means the situation, in which neighbors of neighbors are also included (Grekousis 2020, 23). Of course, the described cases of contiguity only apply to polygons, but W can also be created in other ways. In contrast to contiguity, distance measures can also be used, with Euclidean and Manhattan distances being the most prominent. Another possibility is the use of the nearest neighbor method, where a specified number of neighbors which are closest to an observation are considered (Getis and Aldstadt 2004, 91; Grekousis 2020, 17, 27f).

2.4. Spatial Statistical Methods

This section covers spatial modeling, accompanied by Geographically Weighted Regression (GWR) and Regionalization, which is a group of methods for partitioning spatial data. All of these are techniques of central importance for this thesis, as the proposed approach will be measured against GWR, and regionalization is part of the strategy suggested. Due to their extent, the elaborations on regionalization are split into three sections. The first one describes it from a general perspective, defining it and outlining its use. After that, important terms and concepts which are necessary for carrying out a regionalization procedure are described, before going into detail on the regionalization methods at the end of the chapter.

2.4.1. Spatial Modeling & Geographically Weighted Regression

Spatial modeling investigates "how spatial dependence, spatial autocorrelation and spatial heterogeneity can be modeled in order to produce reliable, predicted spatial outcomes" (Grekousis 2020, 4). As an example, spatial modeling can be used to model the relation a house price has with its location, in order to then be able to predict the house price when new data emerges. From this, the distinction between spatial modeling and spatial analysis becomes obvious: While spatial analysis refers to understanding data and summarizing it in a way it can be useful for various applications, spatial modeling is about finding underlying relationships in data that allow for making predictions when confronted with new data. An important technique for modeling is regression, which is explained in more detail in section 2.5.2.

Geographically weighted regression (commonly termed GWR) was first introduced in 1996, before being published in a more comprehensive book in 2002 (Brunsdon et al. 1996; Fotheringham et al. 2002). The central thought underlying GWR is the goal of uncovering local relations in spatial data, related to the case described in section 2.3.2. The example of average values given there can be termed global observations, which only make sense if there is no significant difference in the various local observations - from which the global observation is derived. An increase in the spatial variation of the local observations is always accompanied by a decrease in the reliability of the global observation. The exemplary scenario of mean air temperature in the aforementioned section is exaggerated and not something researchers would actually publish - in contrast to another methodology frequently used in spatial analysis. To exemplify this, imagine a regression model that predicts house prices across a country based on attributes like age of the house, size or neighborhood attributes, for example. The result of this regression are parameter estimates that show the relation the house price and a respective attribute have with each other. These estimates are in fact *global* statistics, which might not be useful for modeling *local* conditions, as each parameter reflects the *average* relationship of price and an attribute. The age of a house might drive the price up in rural areas, where old houses are seen as desirable objects. In bigger cities, the opposite might be the case. As these two effects cancel each other out in the global statistic, this could easily result in the assumption that the price is independent from the age of the house. Of course, this is

2. Theoretical Framework

wrong in many cases and leads to a huge loss of information on the local variations of the relationship. Regardless, it seems that many researchers - at least at the time when GWR was first published - were happy to accept these *global statements of spatial relationships*, while *global observations*, like the mentioned mean air temperature, were avoided. This is the reason for the proposal of GWR, a method that uses *local statements of spatial relationships* (Fotheringham et al. 2002, 2f).

GWR enhances the usual framework of a regression by using local parameter estimates instead of global ones. This is made possible by using an uninterrupted surface of varying parameter values, from which the parameters for every location in the study area can be extracted. Simplified, the surface is created the following way:

1. Decide for which location parameters shall be estimated.
2. Select a subset of points near this location.
3. Estimate parameters for the location using the subset.
4. Repeat with all locations.

It is also worth to mention that this allows for parameters to be estimated for any given location, it is indifferent whether data have been recorded at that location or not (Fotheringham et al. 2002, 52ff). GWR will be applied to the data in the practical part of this thesis, for which further elaboration in the context of this research will be given in section 3.6.2, in addition to a description of the results in section 4.2.2.

2.4.2. Regionalization

Regionalization describes the procedure of grouping "a large number of spatial objects into a smaller number of subsets of objects, which are internally homogeneous and occupy continuous regions in space." (Assunção et al. 2006, 797) The outcome of this procedure are newfound areas, referred to as *regions*. Influential work on the subject - especially in a socio-economic context - was pioneered by Openshaw (1977), where the process was mainly termed *zone design*.

In practice, the application of regionalization methods is important for policymakers and spatial planners, as the methods allow for grouping of administrative units of various scales (e.g. counties, districts, census tracts) into bigger regions. As these regions are homogeneous in themselves, policies can be applied to the bigger units instead of the many small ones, which is an efficiency enhancement (Grekousis 2020, 304f). Additionally, grouping units into regions can help to protect privacy, equalize population numbers, mitigate the impact of outliers or inaccuracies within the data, as well as alleviate visualization and interpretation (Wise et al. 1997; Wise et al. 2001). There are many regionalization methods such as *SKATER* (Assunção et al. 2006), *spatially constrained k-means* (Mignotte 2011), *AZP* (Openshaw 1977), *spatially constrained hierarchical clustering (SCHC)* (Duque et al. 2007), and *max-p* (Duque et al. 2012). All of the mentioned methods will be evaluated in this thesis and thus are described in more detail

in the following subsection. Also, it is necessary to outline some terms and concepts used in these methods, which is done with the following elaborations.

2.4.3. Terms & Concepts in Regionalization

This subsection will explain important terms and concepts which are necessary for carrying out a regionalization procedure in practice. This includes different approaches to it, varying uses of the term distance, as well as different types of linkages.

Firstly, it makes sense to distinguish between the different general approaches when trying to solve a regionalization problem, which can also be termed zonation or districting. Anselin (2020) distinguishes between two approaches: hierarchical and partitioning methods.

Regionalization can be seen as a specific case of clustering that adds spatial constraints. Several concepts that will be mentioned have been developed "aspatially", but spatial constraints can be added to make regionalization possible. Because of this, the next paragraphs will elaborate clustering approaches from a more general perspective, non-spatial as well as spatial. Two approaches will be described: hierarchical and partitioning methods.

Influential work on hierarchical clustering methods was done by Joe H. Ward (1963), who is also the creator of *Ward's linkage*, which will be described in one of the following paragraphs. He was looking for a way to "*cluster large numbers of objects, symbols, or persons into smaller numbers of mutually exclusive groups, each having members that are as much alike as possible*". As mentioned above with regionalization in general, according to him this can lead to easier understanding of relationships in big data sets, therethrough enabling efficiency increasements. The downside of clustering is that it results in a loss of information, which can be quantified in a metric termed *error sum of squares*. This metric is used to calculate a number that reflects how good the clustering result is, which is commonly termed *objective function*. It can not only refer to minimizing the error sum of squares, but to anything the researcher aims to investigate, most of the times the objective function is to minimize the information lost by grouping observations, reflected by the mentioned error sum of squares. The assumption before starting a hierarchical clustering process is that the maximum information can be extracted from an ungrouped set of n observations. These n observations as clusters, each of them containing one member, are the starting point for the process. The initial phase is carried out by grouping two of these clusters in one, bigger, cluster. The two clusters that impair the objective function the least when merged, are chosen. The result of this are $n - 1$ remaining subsets, for which a decision is made whether to join another observation to this cluster, or to form a new cluster made up of two different observations. This process can be continued until all n observations are part of the same cluster (Ward 1963, 236ff). It is possible to perform this procedure "bottom-up", like described in the previous sentences, or "top-down" by starting with all observations in one cluster and splitting the most unlike observations into two clusters until each observation is in its own cluster. These approaches are called agglomerative and divisive hierarchical clustering. In both approaches, a *dendrogram* is used to show the process and the connections between the

2. Theoretical Framework

clusters (Figure 2.4). Usually, an intermediary result that corresponds to the desired number of clusters is used, which should lead to clusters that are similar in size, have distinct borders and are as homogeneous as possible. Finding this optimal number can be challenging in cases where the desired number of clusters is not known. One way to find a fitting number is by inspecting the dendrogram and the inconsistency coefficient. (Grekousis 2020, 293f, 296f). Methods belonging to this group that will be part of the practical part are *Ward* and *SKATER* (Anselin 2020; Feng et al. 2022).

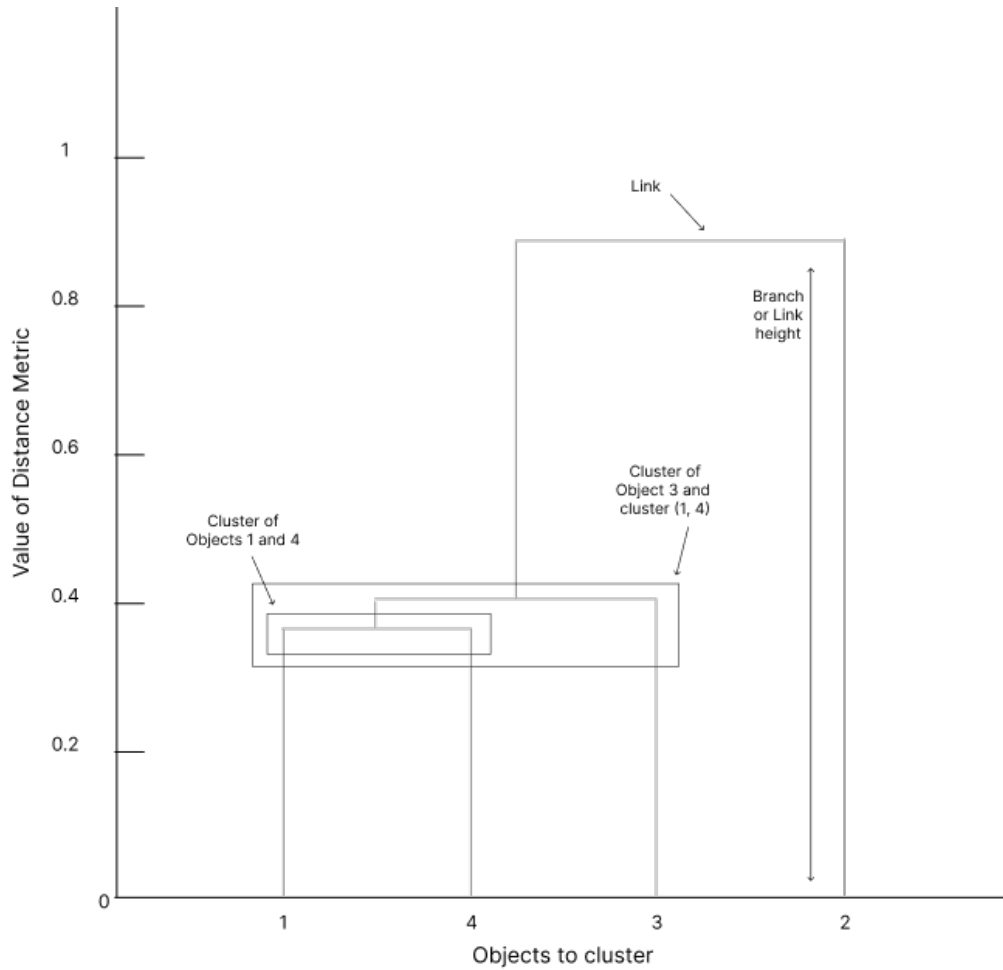


Figure 2.4.: Dendrogram (own creation, adapted from Grekousis (2020, 293))

Partitional clustering, which the prominent *k-Means*-algorithm belongs to, has been pioneered by Stuart P. Lloyd (1982), originally for pulse-code modulation in digital signal processing. Since then, it has found widespread application outside its original field, and has become the most used clustering algorithm. Starting from a number k and n data points, the algorithm tries to find k centers which minimize the sum of squared distances from each data point to the closest center. In the beginning, k centers are

chosen randomly and each point is allocated to the nearest center. After that, the center is calculated again as the center of all points allocated to it. These steps, allocation and center recomputation, are then repeated until the centerpoint does not change anymore (Arthur and Vassilvitskii 2007, 1027f). Because it does not need training, k-means is an unsupervised machine learning algorithm. Before running it, three parameters should be defined: the choice of where to place the initial cluster centers, the number of clusters and which variables to incorporate (Grekousis 2020, 299f). As the result of k-means depends greatly on the initial cluster centers, various approaches on how to place the initial cluster centers have been evaluated. The most popular one is to divide the data into clusters randomly and compute the cluster centers from there, other noteworthy ones are Forgy, Macqueen, Kaufman and k-means++ (Peña et al. 1999, 1029, 1031; Arthur and Vassilvitskii 2007, 1028f). Another parameter is the number of clusters k , similar to the aforementioned problem in hierarchical clustering. If not too computationally expensive, trial and error can be a valid approach to this. In other cases, the elbow method can be used, which denotes the point at which adding additional clusters does not significantly increase the information gained (Grekousis 2020, 300f). Another way is to use the so-called silhouette method, originally introduced by Peter J. Rousseeuw (1986). It can be used to interpret and validate clustering results. By comparing the plots and numbers resulting from using it on various numbers of k it can be used to find the best fitting number of clusters in a dataset (Subbalakshmi et al. 2015, 348). Finally, the variables that will be considered for the clustering process need to be selected. Alternatively, it can be defined how heavily they are weighted. Ideally, the researcher has prior knowledge about the subject and is able to infer the decision from there. If that is not the case, one approach is to start with a few variables and add new ones in the process while checking the aforementioned elbow and silhouette scores before moving on to the next variable. Another possibility is using the R^2 score, which can show how good a variable is at separating the data into clusters. One additional aspect to consider are outliers, whose existence influences results dramatically. In cluster analysis, outliers can be seen as separate very small clusters that are distant to the other clusters. These sometimes uncover noteworthy patterns that should be looked at in more detail (Grekousis 2020, 303f). Assunção et al. (2006, 798) points out that when using these methods, the algorithm itself does not consider the constraints that can be imposed by neighborhood relations, and that these are added in an additional step or added indirectly, for example by using coordinates as attributes. In addition to k-means, other partitioning methods that will be used are AZP and max-p (Anselin 2020; Feng et al. 2022).

Another concept that needs clarification are the different uses of the term *distance*. Intuitively, distance is thought to be the length of a straight line on a plane connecting two points A and B, with A having the coordinates X_1 and Y_1 and B having the coordinates X_2 and Y_2 in a projected coordinate system. The most common way to calculate this is by using the Euclidean distance:

$$D = \sqrt{(X_2 - X_1)^2 + (Y_2 - Y_1)^2} \quad (2.1)$$

In most scenarios, for example when analyzing socioeconomic phenomena, Euclidean

2. Theoretical Framework

distance is the most reasonable choice. For other applications, mainly when studying travel distance as well as when studying urban environments, Manhattan distance is suited better. It differs from Euclidean distance in the fact that it is able to approximate the deviation from a straight line, which is almost never realistic due to building blocks or streets. As the subject of this thesis has a definitive socioeconomic influence by looking at housing prices, the distance measure of choice is the Euclidean distance (Grekousis 2020, 17f; Lankford 1969, 196).

The other use of the term distance refers to *(dis)similarity*, termed *attributive distance* by Wang (2014, 149), where it is also stated that the most common metric for this is also the Euclidean distance. In this case, a distance metric (e.g. Euclidean distance) can be used to denote how similar two samples are. This is done by forming a space in which the values of the attributes are represented on a number of axes - for two variables, there are two axes. Between these plotted points, the measured distance refers to how similar two observations are, with a small distance implying high similarity (Raschka and Mirjalili 2017, 103).

With these distances, another important term in regionalization and its different types can be discussed, namely *linkages*. The most relevant types of linkages are single linkage, average linkage, centroid linkage, complete linkage, and Ward's linkage (Grekousis 2020, 295). The goal of every clustering process is the calculation of clusters that do not vary too much in size, are compact and homogeneous in themselves.

Single linkage is often used synonymously with the nearest neighbor method. It is carried out by finding distances between the attributively closest members for each pair of clusters and combining the two clusters that have the smallest attributive distance between its most similar members. The result of using this method are often ranking clusters that also lack balance, making it a poor decision in the majority of scenarios (Wang 2014, 149ff.). Complete linkage is resembling of this, with the difference lying in the fact that the most unlike members are compared for merging the clusters, instead of the most similar ones. This approach is useful if the goal of the clustering are very compact clusters (Raschka and Mirjalili 2017, 364; Wang 2014, 151). Figure 2.5 illustrates these two types.

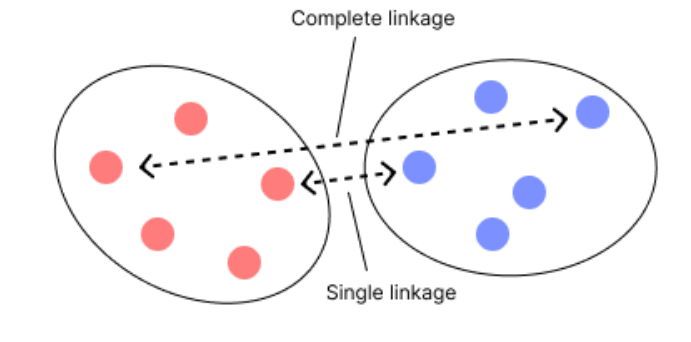


Figure 2.5.: Single and complete linkage; adapted from Raschka and Mirjalili (2017, 364).

Another type of linkage is average linkage. Here, the distance between two clusters is calculated from the average distance between all pairs of observations. This leads to average linkage being similar to measuring the distance between the centers of the clusters (Grekousis 2020, 295). Very closely related to this is another type, namely centroid linkage, the only difference being the use of the squared distance from the observations to their cluster centroids, instead of the standard average, making it especially useful in the presence of outliers (Wang 2014, 151). The final type of linkage is Ward's linkage. Here the objective is to merge the two clusters which increase the intra-cluster variance of the newly formed one as little as possible. This leads to clusters that are even in size and spherical (Raschka and Mirjalili 2017, 364; Wang 2014, 151).

2.4.4. Regionalization Methods

The following paragraphs will shed light on the details of the approaches investigated in this thesis. An evaluation of these will be carried out when discussing the results of the practical part at a later stage in this thesis. Before getting into detail on AZP, SKATER and max-p, two methods that have actually been part of the previous paragraphs, SCHC and k-means, will be subject to some additional elaborations.

The method that will be discussed first is spatially constrained hierarchical clustering (SCHC) using *Ward's* linkage. In the *spspt* Python package, it is termed *WardSpatial*. Many important things about hierarchical clustering and linkages have been said already. In this case, an agglomerative hierarchical clustering approach is applied using Ward's linkage and an additional spatial connectivity constraint. The number of regions, as well as which attribute to use for the task needs to be defined by the user (Feng et al. 2022).

The principle behind the k-means algorithm has already been discussed in detail in the previous section. To use it for solving a regionalization task, spatial contiguity constraints are imposed (Feng et al. 2022).

The automatic zoning procedure (AZP) was originally described by Stan Openshaw (1977). It tries to group spatial units into a number of regions which needs to be defined by the user. The criterion for forming the regions, the objective function, is the within region error sum of squares. The algorithm starts with an initial solution, with n units grouped into k regions. There are different approaches to constructing this initial solution, one example is to randomly select k units and add neighboring features until arriving at an end result. By doing this, each n is grouped exactly into one region k and contiguity is guaranteed. After this, one random k is chosen and all n neighboring this region are considered for a move to the chosen region k . The premise for a possible move is the preservation of contiguity. The decision whether the unit n is moved to the other region is made based on the objective function. This is repeated with all the neighbors until there are no more neighbors left to consider for possible moves. Then, another region is chosen at random and the process is repeated with its neighbors, until all regions have been evaluated in this fashion. After that, the whole process is repeated with the updated regions, until there is no more improvement in the objective function. One thing to consider with this is that when operating like this, the algorithm does not look for the

2. Theoretical Framework

best move globally, but locally, as only one neighbor is considered at a time, which can lead to it getting trapped in local "solutions". To combat this, different approaches like simulated annealing and tabu search have been proposed, which can help mitigate these problems by allowing moves that do not improve the objective function, for example. Another major factor to mention are the starting points for the initial solution, called seeds, which have a big impact on the results (Openshaw 1977; Anselin 2020).

SKATER was developed by Assunção et al. (2006) and stands for "Spatial 'K'luster Analysis by Tree Edge Removal." This algorithm uses a connectivity graph to define the relationships between objects. Each object is represented by a vertex, and these vertices are linked to neighbors with edges. The edges are assigned with a cost, thus integrating the (dis-)similarity of the two objects the edge connects. This graph is then cut at appropriate places, resulting in connected clusters of objects, the *regions*. This transforms the regionalization task into a task of graph partitioning. The goal is to lower the graph's complexity by cutting it at edges that have a high cost, which means cutting it at edges that connect two dissimilar vertices. Removing edges splits the graph into multiple so called subgraphs, which represent the regions (Assunção et al. 2006, 799f). To clarify more on this, it should be mentioned that first, the graph is cut into two parts, forming two clusters. Afterwards, each of these parts gets cut again, and the process is repeated until the desired number of regions are reached. The number of regions to form needs to be specified beforehand. As with the other methods, attribute constraints can be used (Grekousis 2020, 308).

Max-p, originally called "the max-p regions problem", distinguishes itself from the other methods by not needing the number of regions as an input parameter. In fact, it makes this part of the solution process. The algorithm requires another parameter: a minimum size. This can either be a number of features a region is made of, or a minimum population that a region needs to have, for example. The objective function of this algorithm also differs from the other algorithms, as it is twofold: Similar to the other ones, the overall homogeneity, as the error sum of squares within a region, is considered. In addition to that, the number of regions k (the use of p is more common for this method) is part of the objective function. Priority is given to finding a larger p , a bigger number of regions, over the homogeneity. This means that the maximum p solution does not necessarily lead to the best result regarding the within region error sum of squares. As a perfect solution for this is impossible to find in real life scenarios, the algorithm uses a heuristic, which is not guaranteed to find the global optimum, which requires trial and error with the available parameters. The heuristic consists of three steps. In the first step, the largest possible p value (max-p) which can fulfill the minimum size and the contiguity criteria is determined. For this, different solutions are considered, which are found by growing the regions from random starting points and adding neighbors until the minimum size criterion is met. Not all solutions can be considered in this step, but repeating it as often as possible will lead to a higher number of p . During this, some units will not be assigned to a region, due to them breaking at least one of the two constraints, which will lead to them being stored in enclaves. These need to be assigned to one region before

the optimization process can be started. When this has been done, all of the possible solutions for a value p are considered and processed in the same way as for AZP (Wei et al. 2021; Anselin 2020; Duque et al. 2012).

2.5. Machine Learning

This section introduces the topic of Machine Learning. At the beginning, Machine Learning is elaborated in a broad and more general sense, briefly describing its evolution, what types of Machine Learning exist, and how these techniques can be used to our advantage. Afterwards, the use of Machine Learning for solving regression problems, as well as Random Forests, is explained. As the predictive models in the practical part will be Random Forest regression models, this is of great importance. The explicit use of Machine Learning for spatial analysis is outlined in the following section (2.6).

2.5.1. General Remarks on Machine Learning

In the second half of the 20th century, Machine Learning (ML) has emerged as a subdiscipline of artificial intelligence (AI) (Raschka and Mirjalili 2017, 2). AI itself is the "science of finding theories and methodologies that can help machines understand the world and accordingly react to situations in the same way that humans do." (Joshi 2017, 8) Machine learning is one of the most prominent ways to integrate AI into a machine. It involves algorithms that are able to learn by themselves, distilling knowledge and insights out of data. In the time we live in, data is a resource available in affluence. For humans, combing through excessive amounts data manually to find patterns and rules for model-building is a tedious task. Machine learning is a potent way to tackle this problem, as it allows for the extraction of knowledge from data, paving the way for performance-improvements of models for prediction in order to make decisions driven by data. It is not only an important area of research in computer science, but also takes a huge role in everyday life. Some examples of how Machine Learning improves our day-to-day life are spam filters for emails, web search engines, software for recognition of voice and text, and face recognition. There are three types of Machine Learning: supervised learning, unsupervised learning, and reinforcement learning (Raschka and Mirjalili 2017, 2; Joshi 2017, 31). The most common ones are supervised and unsupervised learning, which is why the next paragraphs will only focus on these two types, leaving out reinforcement learning.

In supervised learning, the aim is to design a model from training data with labels, in order to make predictions on new data. Labelled data means that the possible outputs are known beforehand. As an example, it is helpful to think of a dataset containing emails, which have been correctly marked as spam and not-spam. When a new email is received, the model predicts which of these categories it belongs to. The assignation to categories, as in this example, is called a *classification* task. When the outcome is a continuous value, the term used is *regression*. The former is described in the paragraphs below, the latter in the following subsection.

2. Theoretical Framework

Classification predicts, based on previous records, which category an instance belongs to. It can only predict categories present in the data the model was trained with, these categories are commonly termed class labels. The mentioned example of spam detection is a classic example of a binary classification, as there are only two classes an email can belong to: spam or no spam. Of course, non-binary classification is also possible. Any class label that was present in the dataset used for training can be assigned to unseen data. Handwritten character recognition is a common example for multiclass classification (Raschka and Mirjalili 2017, 3f).

In contrast, unsupervised learning deals with unlabeled data, meaning the possible outcomes are unknown. A typical technique that uses unsupervised learning is finding subgroups in a dataset with clustering. It is useful for organizing information into sensible smaller groups. The objects in the clusters resulting from this are similar to each other, and dissimilar to objects in other clusters, thus this technique is also called unsupervised classification. Clustering is valuable for giving data structure and finding meaningful patterns in it. To make this more explicit, the example of a customer database is well-suited. Finding groups of similar customers of a company allows employees to roll out targeted campaigns for marketing (Raschka and Mirjalili 2017, 7).

2.5.2. Machine Learning Algorithms for Regression

Before discussing regression with Machine Learning, it is useful to shortly look at regression analysis in general. It is a group of techniques which allow for exploration of the relationship between two or more variables, these being one dependent variable and one or more independent variables. It helps to understand how the output variable changes with changing one or more input variables (Joshi 2017, 54). In a lot of cases the analysis consists of establishing a linear regression model, in which the dependent variable can be denoted as a linear combination consisting of the independent variables. A simple linear regression model can be written as:

$$y_i = \beta_0 + \beta_1 x_i + \epsilon_i$$

In this notation, y_i is the dependent (or response) variable, β_0 is the intercept, β_1 the slope, x_i the independent (or predictor) variable, and ϵ_i is the error term. For example, a description of the relation between average blood pressure and age could be formulated as:

$$\text{average blood pressure} = 95 + 0.80 * \text{age} + \epsilon_i$$

Two uses of regression which are worth highlighting are:

1. *Description*, where researchers aim to characterize a relationship between variables without making many assumptions about it. This might help to establish a meaningful relationship further in the research process and serves as an overview of relationships that might or might not be present.

2. *Prediction*, where response variables are predicted for various values of the input variables. For this, numerous assumptions and factors must be considered and met, for example not making predictions outside the range of the sampled data used for creating the equation (Young 2017, 3f, 8f).

When doing a regression task with Machine Learning, it belongs to the category of supervised learning (Joshi 2017, 54). The goal is to learn the values of the slope in order to predict the responses of the target variables to new explanatory ones which have not been part of the data the model was trained with. Depending on the task and data to analyse, there are different ways to practically deal with this objective. The simplest of them is basic linear regression, also called univariate linear regression, which can be understood as finding the "line of best fit" or regression line for the data. When using multiple explanatory variables, the technique is called multiple linear regression, where the line of best fit is transformed to the best fitting hyperplane. So far, regression methods for linear relationships have been described. Of course, regression analysis can also be carried out when the assumption of linearity is violated, which is then termed *polynomial* regression. Conceptual differences arise when doing a regression task with *Random Forest* (Raschka and Mirjalili 2017, 310-342), which will be elaborated following some introductory remarks on this algorithm.

2.5.3. Random Forests

To actually carry these techniques out, the most common algorithm for supervised learning is *Random Forest*, the version introduced here is based on the work of Breiman (2001). The random forest is made out of an ensemble of decision trees, with the multitude of trees constructing a "forest". The thought underlying random forest is averaging a multitude of decision trees in scope of creating a robust model with good generalization performance and insusceptibility to overfitting (Raschka and Mirjalili 2017, 98). The algorithm can be described in three steps:

1. Generate n_{tree} samples through bootstrapping from the data (bootstrapping = sampling with replacement).
2. For every sample, create a classification or regression tree that is split at its nodes based on a random sample of the predictor variables, called m_{try} .
3. Aggregate all predictions made by the n_{tree} decision trees into the final prediction (majority vote for classification, average for regression).

Reasons for why the Random Forest algorithm will be used in this work are manifold. They are very user-friendly, as they can be used for both classification and regression tasks. Additionally, they only need two input parameters: m_{try} being the number of variables in each random subset, and n_{tree} describing how many trees the forest should consist of (Liaw and Wiener 2002, 18). Adding to that, it has been shown to be suitable for geographical modeling, for example by Hengl et al. (2018), Nussbaum et al. (2018), and Prasad et al. (2006). Moreover, tree-based methods have shown to be the best suited of

2. Theoretical Framework

all statistical learning methods for spatial data analysis. This is due to their transparency, the aforementioned simplicity, their ability to express nonlinear relations, their strengths in handling big amounts and dimensions of data, as well as missing values (Kuhn and Johnson 2013).

Before describing regression tasks with Random Forest, it is useful to take a look at the multitude of Decision Trees (DT) the Random Forest is built with, because the two tasks of regression and classification do not differ substantially from each other. A Decision Tree can be described as a structure which splits data into different branches by making a decision at each of the splits. This makes it possible to arrive at a decision by following the grown tree based on the data in use. Which and how these decisions are made is decided by training algorithms, depending on the connections between target labels and input data. A very simplified exemplary Decision Tree helping a pupil decide on his or her activity for the rest of the day can be seen in figure 2.6.

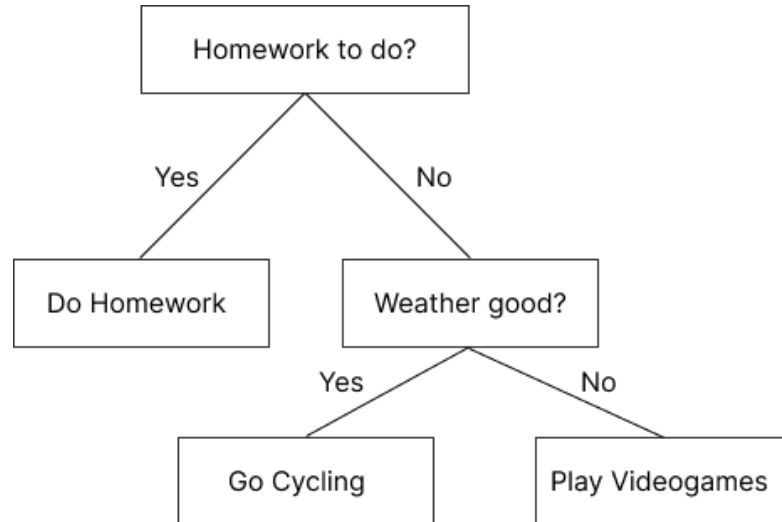


Figure 2.6.: Illustration of a simple Decision Tree

The Decision Tree learns various questions which are then answered to in turn infer the labels of the sampled data. The example above uses categorical data for this, but the same can be done when basing the decision on continuous values, for example instead of "Weather good?" the question could be "Temperature $\geq 20^{\circ}$ C?". In general, Decision Trees are grown based on a goal the algorithm wants to achieve, which in the case of regression is minimising the Mean Squared Error, for example. The final prediction for the target variable is obtained by averaging all predictions made from the Decision Trees (Raschka and Mirjalili 2017, 88f, 342; Joshi 2017, 64f).

2.6. Spatially Conscious Machine Learning

The following section illuminates Machine Learning from a spatial perspective. First, ways of giving Machine Learning models spatial consciousness are elaborated, before moving on to a technique which the approach of this thesis will be compared against - Geographically Weighted Random Forest.

2.6.1. Spatial Machine Learning

Machine Learning can be extended to *spatial* Machine Learning by using data that has a spatial component, without adaptations being necessary in the used algorithms. Of course, it is possible to adapt the algorithm as well, more details on this are given in the following paragraphs.

Machine Learning has been used in a variety of fields, including fields in which spatial effects matter. Of course, the special properties of spatial data also need to be considered in Machine Learning tasks. Some exemplary works where this was done include research in land cover classification (Zhang et al. 2019), urban growth (Guan et al. 2005), disaster management (Resch et al. 2018), crop yield prediction (Masjedi et al. 2019), point cloud classification (Zhao et al. 2018), spatial interaction (Yan and Thill 2009), and spatiotemporal prediction (Wu et al. 2021).

As elaborated in various sections of this thesis, spatial data has special characteristics distinguishing them from other types of data, examples for this being spatial dependence and spatial heterogeneity. These properties also need to be taken into account when applying Machine Learning to spatial data. In fact, explicitly incorporating them has shown to have the ability to improve the performance of Machine Learning models. Conversely, ignoring them or using them in an inappropriate manner can have a negative impact by decreasing the performance of the created machine learning models (Nikparvar and Thill 2021, 1).

To use spatial data in Machine Learning, some type of spatial information needs to be added to the learning process. Figure 2.7 shows how this process can be organized into two steps, these being the spatial observation matrix (observation matrix is a commonly used term in Machine Learning that in essence describes a table filled with data), and the learning algorithm.

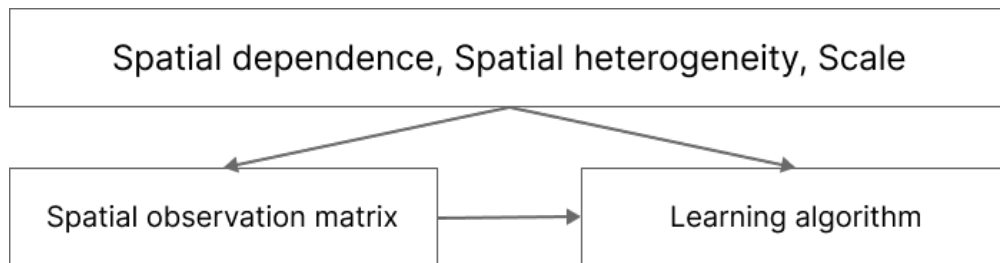


Figure 2.7.: ML with spatial data; simplified from Nikparvar and Thill (2021, 8).

2. Theoretical Framework

When integrating spatial properties into the observation matrix, standard Machine Learning methods and algorithms can be used without adapting them. This of course postulates that the observation matrix is as good as possible, which requires consideration of various important aspects, these being proper sampling techniques, the inclusion of spatial features, dimensionality reduction and the handling of missing data (Nikparvar and Thill 2021, 8).

An alternative to incorporating spatial properties into the observation matrix and using standard Machine Learning techniques is to directly integrate them into the algorithm. Decision trees in the form of random forests, neural and deep neural networks as well as support vector machines have received substantial consideration in spatial sciences. An example for decision trees and random forests that has been implemented recently is called geographically weighted random forest (GW-RF) (Nikparvar and Thill 2021, 13f), which is described in the following subsection.

Academic works always mention future research opportunities, in the context of this thesis gaps and opportunities which have often been pointed out are related to noisy results due to the use of non-spatial learners like Random Forest and non-spatial bootstrapping, as well as looking for ways to explore machine learning models locally (Talebi et al. 2022, 19f; Georganos et al. 2021, 134f; Georganos and Kalogirou 2022, 7). The proposed approach, which gets its spatial consciousness by applying regionalization methods before deploying random forests takes on these challenges and tries to fill the mentioned gaps. Theoretically, RegRF makes local parameter tuning and feature selection (n_{tree} and m_{try}) possible, as well as local feature selection in each of the regions/models, making it a valuable new approach in spatial analytics and spatial Machine Learning.

2.6.2. Geographically Weighted Random Forest

GW-RF is one of the methods the proposed approach is measured against. For this reason, going into detail on this specific method is of great value.

It was introduced by Georganos et al. (2019) with the scope of filling the need for spatially conscious Machine Learning algorithms, that are able to consider spatial variations and spatial relations in their predictions. Their concept stems from the idea of decomposing a global model into many smaller local sub-models, a well established idea that has been described in the chapters on local and global statistics (2.3.2) and GWR (2.4.1). They suggest using an "ensemble of local models rather than a single, global one." This approach allows to put the focus on interesting local variations and to attain more accurate predictions when spatial heterogeneity is present. The method was compared to an analysis carried out with casual random forest.

In standard Machine Learning techniques, the model is designed with training data from the area of interest, not taking the locations into account. This lacks integration of spatial non-stationarity, which is present most of the time when analysing spatial data and is important for being able to uncover underlying phenomena present in the data. In addition to that, the ignorance of spatial factors in a model leads to a deceptive interpretation of the performance metrics of the model. The authors propose to solve

these problems by designing a separate random forest for each location (each data point), that only takes data points which are close in space into account. This is not as efficient from a computational perspective, but allows for the local exploration of the model and the effects scale has on it. Additionally, as multiple random forests are created for the data points, the local performance metrics as well as locally important features help to better understand the underlying processes. The most important specification when designing these models is which data points to incorporate in each model. This is done by the k-nearest neighbor method (Georganos et al. 2019, 1ff).

The application of these techniques results in a better overall performance compared to the non-spatial random forest method. The suitability of this method can also be seen in the widespread application the method has received since its introduction, as it was implemented in works from various researchers and domains, including population modeling, health science, ecology, and geomorphology (Georganos et al. 2019; Georganos et al. 2021; Georganos and Kalogirou 2022; Quiñones et al. 2021; Lotfata et al. 2022; Quevedo et al. 2021; Santos et al. 2019).

2.7. Performance Evaluation

The following section presents some metrics that allow for the evaluation of statistical and machine learning models. As the created models will be compared to each other and to benchmark models, ways to assess how performant they are are an important element of this research. The mentioned ones represent a selection, as it is not possible to elaborate on all available metrics that could possibly be used. Which performance indicators are used also depends greatly on the type of model to evaluate. Most metrics introduced refer to evaluating regression models, as the predictions in the practical part are done solely with regression models. In the equations in the following sections, X_i is the predicted value at point i , and Y_i is the actual value at point i . The multitude of models also leads to a vast number of performance metrics, making statistical tests for significance necessary, which is why ways to address this are given. As a regionalization procedure will be carried out prior to the regression modelling and prediction, ways to assess the results of regionalization are also needed and thus presented at the end of this section.

2.7.1. Mean Absolute Error

A commonly used evaluation metric for regression models is the mean absolute error (MAE). It is constructed by taking the absolute values of each prediction error and calculating the mean of those. The individual prediction error is found by calculating the difference between the predicted and the actual (ground truth) value (Sammut and Webb 2011a).

$$\text{MAE} = \frac{1}{m} \sum_{i=1}^m |X_i - Y_i| \quad (2.2)$$

2. Theoretical Framework

The MAE is part of a family of measures which evaluate the fit of a model based on the distance between the actual data points and the regressor (Chicco et al. 2021, 3).

2.7.2. Mean Squared Error and Root MSE

Another member of this family of indicators is the mean squared error (MSE). It is constructed the same way as the MAE, with the difference that it refers to the mean of the *squared* prediction errors (Sammut and Webb 2011b). Squaring the prediction error leads to an emphasis on bigger values, which means bigger errors and/or outliers are weighed more heavily, meaning that the MSE is more sensitive to outliers and well suited for detecting them.

$$\text{MSE} = \frac{1}{m} \sum_{i=1}^m (X_i - Y_i)^2 \quad (2.3)$$

A descendant from the MSE is the square root of it, often termed RMSE (root mean squared error), it standardizes the units of the MSE, making interpretation easier. (Chicco et al. 2021, 3ff)

$$\text{RMSE} = \sqrt{\frac{1}{m} \sum_{i=1}^m (X_i - Y_i)^2} \quad (2.4)$$

2.7.3. R-squared

The coefficient of determination, often called R^2 or R -squared, was first introduced by Wright (1921). It measures the direct contribution of a given factor for explaining the variability in the system.

$$R^2 = 1 - \frac{\sum_{i=1}^m (X_i - Y_i)^2}{\sum_{i=1}^m (\bar{Y} - Y_i)^2} \quad (2.5)$$

The original definition states that if all factors are taken into account, the R^2 value equals 1, implying a "perfect" model. It shows the relation between the model for prediction and the ground truth.

2.7.4. Statistical Testing

A vast number of statistical tests exist for various scenarios, which is why a few things need to be considered in order to choose an appropriate testing method. The next few paragraphs will delineate the steps leading to this decision, before shortly describing the statistical tests which will be used to evaluate the results of the practical part of this thesis.

Statistical testing is important when trying to find out whether variations in values of a dataset are random or not. One factor contributing to the decision on which statistical test to use is the distribution of the values. For normally distributed data, parametric

tests are used most of the time. If the assumption of normal distribution is not met, non-parametric tests need to be used (Bluman 2012, 672). So, the first step is finding out whether the data at hand meet this assumption or not. For this, another statistical test can be used, namely the Shapiro-Wilk-Test, which obtains a test statistic by comparing the sample at hand with a completely normal distribution (Shapiro and Wilk 1965, 591).

The practical examination of the results with the Shapiro-Wilk-Test showed that only some experiment configurations produced normally distributed results, which is why the decision was made to use non-parametric tests. In this realm, multiple options for specific use cases are available, so again it is important to choose an appropriate one for the task at hand. As the comparison of two or more experiment configurations is desired, multiple tests are needed for this. When comparing just two, the Wilcoxon test is suited, and for more than two, the Friedman test will be used (Currell and Dowman 2009, 299). The decision on which one of these to use is made based on whether the samples to compare can be classified as dependent or not. As they are results based on the same dataset, just with a different configuration of the experiment, they are dependent which leads to the choice of the statistical tests that are briefly described in the following paragraph (Bluman 2012, 492).

As is standard with statistical tests, the Wilcoxon signed-rank test starts by defining the hypotheses H_0 (No difference between two compared samples) and H_1 (Claim that there is a difference between the two samples). After that, the test value is computed and compared to the critical value. Based on this, a decision is made whether to reject H_0 or not (Bluman 2012, 689f). The equivalent to this for more than two samples is the Friedman test (Currell and Dowman 2009, 311).

2.7.5. Spatial Autocorrelation in Residuals

An important aspect in the evaluation of spatial predictive models is the presence and/or the degree of *residual spatial autocorrelation* (RSA). The goal of analyses should be minimizing residual spatial autocorrelation, which can be assessed using the Moran's I metric or local indicators of spatial association (LISA). LISA can also be used to then explore these residual clusters which can also help understand the underlying processes further (Liu et al. 2022; Anselin 1995).

2.7.6. Metrics for Regionalization

As the ground truth data for real-life regionalization procedures are unknown most of the time, specific measures to evaluate the results need to be used. These unsupervised metrics give information about the homogeneity of the regions and the heterogeneity between regions. One such metric is the Calinski–Harabasz index, which indicates the similarity between objects in the same region and the dissimilarity between different regions (Aydin et al. 2021, 2298ff). The within-region similarity is calculated from the distances between objects in a region to the centroid of that region. The dissimilarity is computed from the distance between a region's centroid and the global centroid. The

2. Theoretical Framework

higher this index, the better the result of the regionalization procedure (Arbelaitz et al. 2013, 245f).

Another unsupervised metric is the Davies-Bouldin Index, introduced by Davies and Bouldin (1979). It is similar to the Calinski–Harabasz index, but arrives at the result differently. Here, the similarity in regions is measured by the distances from features to a region centroid, and the difference between regions measured as the distance between the regional centroids. Lower values indicate better results (Arbelaitz et al. 2013, 245f).

One more metric from this realm, the Silhouette Coefficient has shortly been mentioned before. It was formulated by Rousseeuw (1986). Here, the similarity in a region is calculated from the distance between all the objects of the region, and the difference between regions stems from the distance of the nearest neighbours. The higher the score, the better the regionalization (Arbelaitz et al. 2013, 246).

3. Methodology

Chapter 3 introduces the practices used for carrying out the proposed modeling approach. From a computational perspective, this involves two different programming languages, Python and R, as well as other applications or software tools. After explaining their use, the study area is examined, before having a look at the dataset itself. These two steps are necessary for finding interesting characteristics, patterns, or other factors that need to be considered when designing the modeling workflow. Speaking of which, the modeling workflow is subject to elaboration in the subsequent section. In fact, it is necessary to design multiple parallel workflows which are combined into one big workflow. The reason for this is the nature of this research, which aims to evaluate a "new" workflow, RegRF, against established ones like GW-RF, standard RF, and GWR. To compare the performance of each of these methods, it is of course necessary to apply them to the data available. This means that analysis will be carried out using all the mentioned techniques, before comparing them to the performance of RegRF.

3.1. Software / Programming Languages

This section elaborates the software and programming languages used for the analysis. As Python plays the central role, it is elaborated in the subsection following this. Afterwards, another programming language in the form of *R* is introduced, which will be used for carrying out the GW-RF analysis. Then, two additional applications, QGIS and Notepad++, are shortly described. This is followed by introducing GitHub, which will be used for the distribution of the code.

3.1.1. Python

Python is a free and powerful programming language with readable syntax. It can run basically anywhere, including Windows, Mac OS X, Linux and Unix, and is free in the sense that it is free to download as well as free to modify and redistribute. Its abilities span all the way from being beginner friendly to offering powerful features for advanced users (Python Software Foundation [2022](#)).

It is a very well established and popular language in data science, which is why a lot of useful libraries, which can be described as add-ons, have been developed, mostly by its tremendous developers and the open-source community itself. The performance of Python as an interpreted language, meaning it is for most parts readable to humans, is not on par with other languages, which is the reason why extension libraries trying to tackle this problem as good as possible, have been developed. There are various ways to

3. Methodology

get a distribution of Python, the distribution used for this thesis has been downloaded via Anaconda, which is a well-established alternative in the context of scientific computing. Anaconda is also free to use and makes it even easier to install the essential libraries (also called packages or extensions) for data science (Raschka and Mirjalili 2017, 13f). The widespread establishment, combined with its interpretability and the availability of extensions are the most important reasons for why the decision to use Python for the practical part of this research was made. The following paragraphs introduce the used libraries and shortly state what they are used for.

NumPy is the number one library for handling arrays in Python. An array is a data structure that efficiently stores and accesses multidimensional arrays, and enables a wide variety of scientific computation' (Harris et al. 2020, 357). Each array can contain one type of data, some examples for which are real and complex numbers, strings and timestamps. Its development began in 2005, and as a huge number of other Python libraries including SciPy, Matplotlib, pandas and scikit-learn are based on NumPy, its importance is almost unmeasurable (Harris et al. 2020, 357). In this work, NumPy is explicitly used for carrying out GWR, which requires arrays as inputs. In other scripts, its use is limited to very basic operations on the data in use, like listing unique values in certain columns, for example. Implicitly, NumPy is used non-stop, as most other libraries in use are built on top of NumPy.

Pandas is a Python library used for data analysis. It treats data as a so-called *DataFrame* object, which allows data manipulation in a convenient manner. With it, data in different formats and structures can be written and read. It contains functionalities for handling missing data, reshaping, slicing, indexing, merging and joining data sets while being optimized for performance (McKinney 2010). In the practical part it is used for reading the .csv file which contains the data, preprocessing it in various ways to prepare it for the actual analysis and saving those prepared files. It is also used for various things in the analysis part, for example dropping certain columns, renaming them, or creating subsets of the DataFrames in use. An important extension in the geospatial realm is *GeoPandas*, which will be described next.

GeoPandas was founded in 2013 and adds support for geographic data to pandas. It introduces two new objects called GeoSeries and GeoDataFrame, which are subclasses of the respective pandas.Series and pandas.DataFrame objects. These are able to work with geometry objects and perform operations on them (Jordahl et al. 2020). GeoPandas naturally plays an important role in this research, as it proposes a workflow for modelling and predicting on geographic - or spatial - data. All the scripts for the practical part use spatial data, so GeoPandas is of central importance for all the scripts implemented in Python. Firstly, it is used to create "real" geographic data sets, as the geographic context is originally just given as latitude and longitude, from which point geometry features can be created. The resulting GeoDataFrames can be saved as shapefiles, for example, which are then used as starting points for modelling and prediction. In these scripts, the GeoDataFrames are used to perform basic pandas operations, like the ones mentioned above, but also for explicitly spatial operations like spatial joins.

3.1. Software / Programming Languages

Matplotlib is a data visualization library that allows users to explore the data in use and produce raster and vector images, which can be saved in various formats. Its goal is to make casual plots as easy as possible, while also allowing more advanced users to build complex visualizations (Hunter 2007). This is exactly what it is used for in this work. In the stage of data preprocessing, it is used to get quick overview maps, either in order to get a feel for the dataset, or to check if (spatial) processing steps worked the way they were intended to. In the analysis part, it is used to visualize the regionalization results, which is a very important step, because all of the subsequent steps in the workflow are based on this result. Finally, it is used to map the results. These plots are saved as image files and used in section 4 for evaluation and discussion of the results.

PySAL is a library for geospatial data science that focuses on vector data. It facilitates the implementation of spatial applications in order to detect clusters, hotspots, construct data graphs, do statistical modeling and to carry out spatio-temporal data analysis, for example (Rey and Anselin 2023). PySAL plays an important role in the practical part of this thesis. It is used to construct the spatial weights matrices from the GeoDataFrames, for the regionalization itself, and for evaluation of the results. The functions for regionalization are organised in a submodule called *spopt* (spatial optimization). Spopt is still under active development and contains methods for regionalization and facility location (Feng et al. 2022). Another PySAL submodule in use is *esda*, which stands for exploratory spatial data analysis. It contains functions to compute LISA and Moran's I, among others (Rey and Anselin 2007). Both LISA and Moran's I are used to evaluate the results of the analysis. One additional library in use, that is also part of PySAL, is *mgwr* (Oshan et al. 2019). It contains a Python implementation of GWR, which has been described in detail in section 2.4.1, which is exactly what it will be used for.

Scikit-learn integrates a wide variety of machine learning algorithms to solve supervised and unsupervised problems. Its goal is to make machine learning accessible for non-specialists, focusing on user-friendliness, performance, a thorough documentation, and consistency. It is used for both academic and commercial purposes. Being based on other scientific Python libraries, it can easily be used in applications of all kinds, by using its algorithms as components for specific use cases (Pedregosa et al. 2011, 2826-2829). Naturally, this is what this library will be used for in this thesis. The first use is to split the dataset into training and testing data, which is done three times to generate three different splits. Then, a classic workflow for random forest regression is implemented. This includes hyperparameter optimization, model training, as well as predicting on the test dataset. Another use are some performance metrics included in scikit-learn, which are used to assess both the predictive performance of the regression models, as well as the results of the regionalization procedures.

3.1.2. R

R is a system which combines an interpreted computer language with an environment in which it can run. Its intended use is statistical computation and creation of graphics. The main component is the language itself, which allows users to branch, loop, and define

3. Methodology

functions. Its built-in functions span across a wide variety of statistical techniques, some examples for which are regression models, clustering, and time series analysis (R Core Team 2023). The use of R in this thesis is limited to running one part of the analysis, namely Geographically Weighted Random Forest, which is part of the package *SpatialML*. *SpatialML* is an R package developed by Georganos and Kalogirou (2022) and is available in the CRAN repository, as is usually the case for R extensions. It is the R implementation of the algorithm described in section 2.6.2.

3.1.3. QGIS

QGIS is a free and open source GIS application which supports a wide variety of data formats and functionalities. It allows users to view, handle, modify and analyze data, as well as create printable maps (QGIS Development Team 2023). In the case of this research, it is used for certain steps in the data preprocessing stage, which are unnecessarily complicated with Python, and very convenient and fast with QGIS. Also, it is used for the creation of certain maps, for example the overview map for the area of interest in section 3.2.1.

3.1.4. Notepad++

Notepad++ is a free editor for source code supporting various languages. It is intended to replace the standard Notepad in the MS Windows environment (Ho 2023). It can be used to open, look at, and make changes in files from a vast amount of types. In this work, it was used to extract the California Housing Data from a .txt file which also contained written information like metadata, with an unformatted table below the text lines containing the actual data.

3.1.5. GitHub

GitHub is a platform which allows users to share and distribute code. In the case of this work it is used to allow others to reproduce the practical part of this research, as well as give a starting point for further research efforts. The created repository can be found [here](#).

3.2. Area of Interest

The analysis in this thesis is carried out on the area of the state of California in the United States of America (US). The reason for this has been stated before, in short this is the case because certain requirements had to be met. One is the reproducibility of this study, and the other one is some kind of validation of the data in other scientific works. The "California Housing" dataset is able to fulfill those. The following subsection will shed light on the area of interest in terms of basic geographical information as well as special features regarding the subject area of this research, real estate analysis. At this point, it is senseful to again stress that real estate market analysis itself is not the goal

3.2. Area of Interest

of this research. Instead, the goal is to evaluate a new approach for spatially conscious machine learning, which is why the area does not need to be especially suited for real estate analysis.

California is a state on the west coast of the US and has a surface area of about 423853.4 km², making it the third largest state in the US by area. To put the size in perspective, California is just a bit smaller than Austria and Germany combined (441000 km²). It borders Oregon in the north, Nevada in the east, Arizona in the southeast and Mexico in the south. In the 2020 decennial census, it had a population of more than 39.538 million people, the highest count of all the US states. The median household income in California is \$84 907, which is \$15 000 more than the US average, a massive plus. Regarding housing and real estate, the median gross rent in California is \$1 750, also a lot more than the average in the US at \$1 191. Homeownership rates are lower than the US average at 55,9% compared to 65,4%. The value of houses is more than \$500 000 for over 65% of all units, with 23% costing more than \$1 million. The average house value is \$648 100. In total, in 2020, there were almost 14.4 million housing units in total in California (U.S. Census Bureau [2023](#)).

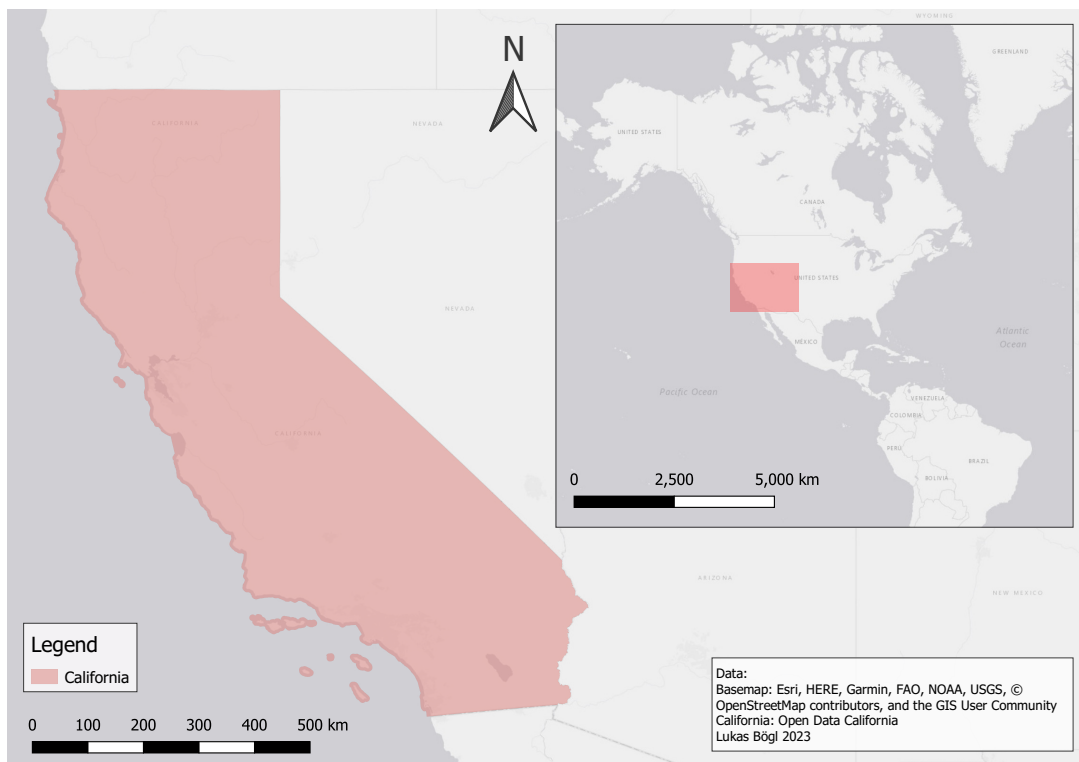


Figure 3.1.: Overview Map of California

3. Methodology

3.3. Data

After introducing the area of interest for this work, this section will delineate the data. As previously mentioned, Machine Learning allows computers to learn from data, making it obvious that the data at use are a core component of every Machine Learning task (Raschka and Mirjalili 2017, 2). Here, the dataset is only introduced in its original form, data exploration and the necessary preprocessing steps are laid out in the sections following this.

The California Housing dataset was introduced by Pace and Barry (1997). Not only did they use it for their own research, the researchers also curated this dataset themselves. According to them, information were collected for every census block group in California in the 1990 census, which implies that every point in the dataset actually refers to one census block group from 1990. The next section will take a closer look at the dataset, uncovering some questionable irregularities. Here, it is subject to description in its original form. Over the years, it has become an exemplary dataset for spatial data in spatial autocorrelation analysis and machine learning tasks (Klemmer et al. 2019; Liu et al. 2022). There are a total of 20640 observations with 9 variables each, meaning the shape of the dataframe is $20640 \text{ rows} \times 9 \text{ columns}$. The variables are listed and described in the table below.

Variable	Description
median house value	Median House Price
median income	Median Income (10 000)
housing median age	Median House Age
total rooms	Total number of rooms
total bedrooms	Total number of bedrooms
population	Total number of inhabitants
households	Total number of households
latitude	Coordinate (WGS84)
longitude	Coordinate (WGS84)

Table 3.1.: Variables of the California Housing dataset and their description.

3.4. Data exploration

In this section, the dataset will be inspected more closely. Looking at descriptive statistics and carrying out Exploratory Spatial Data Analysis (ESDA) is often seen as the first task which needs to be done in a modelling process. It is suited for quickly getting an overview of what the data is about, the range of its values, or detecting errors before they become problematic, to name a few examples (Grekousis 2020, 61). Also, it has proven to

be very important for finding out about the necessary processing steps before starting the analysis.

Before exploring the dataset, it might be of interest to take a look at the notes its creators included in the description of the dataset. Additionally, some remarks on this descriptions are made. For this, it is important to look at Figure 3.2.



Figure 3.2.: The visibly gridded arrangement of the points

The original description of the dataset reads: "We collected information on the variables [...] using all the block groups in California from the 1990 Census. In this sample a block group on average includes 1425.5 individuals living in a geographically compact area. Naturally, the geographical area included varies inversely with the population density. We computed distances among the centroids of each block group as measured in latitude and longitude" (Pace and Barry 1997, 295). Clearly, this indicates the data referring to census block groups, which are divisions of census tracts. Normally, these cover a contiguous area (U.S. Census Bureau 2022). This stands somewhat in contradiction to the information given about the points. The points could of course be centroids of polygons, but extensive searches across a wide variety of platforms did not result in finding polygon geometries which these points could refer to. Adding to this, such administrative borders would be highly unlikely to result in centroids that are arranged as regularly as in this case. The regularity of these points spans all across California. Furthermore, a big number of overlapping point geometries can be found. In extreme cases, there are eight or more points stacked on top of each other, with each one of them having different attribute values, but identical coordinates. This is also not in line with the things mentioned by the creators of the dataset, as in the case of overlapping points there would be multiple points

3. Methodology

for one hypothetical block group. Unfortunately, after extensive research, these questions remain unanswered and the decision was made to move on with the dataset as it is and adding preprocessing steps, which will be elaborated in section 3.5. Reasons for this are its use in other scientific works (Klemmer et al. 2019; Liu et al. 2022), where it has nonetheless proven its suitability. Of course, other big data sets are available online, for example U.S. presidential election data, but these often lack independent variables based on which a dependent variable can be predicted. These would be suited for regionalization tasks, but carrying out predictive analysis is not possible without useful attribute columns. The preprocessing steps which were found to be necessary in order to overcome these problems are explained in more detail in the next section.

First, the exploratory data analysis will focus on non-spatial characteristics like the distribution of values for the respective variables and their correlations. After that, the dataset will be inspected from a spatial perspective. Figure 3.3 shows a diagram for the dependent variable (Median House Value - MedHouseVa), and Figure 3.4 shows the independent variables.

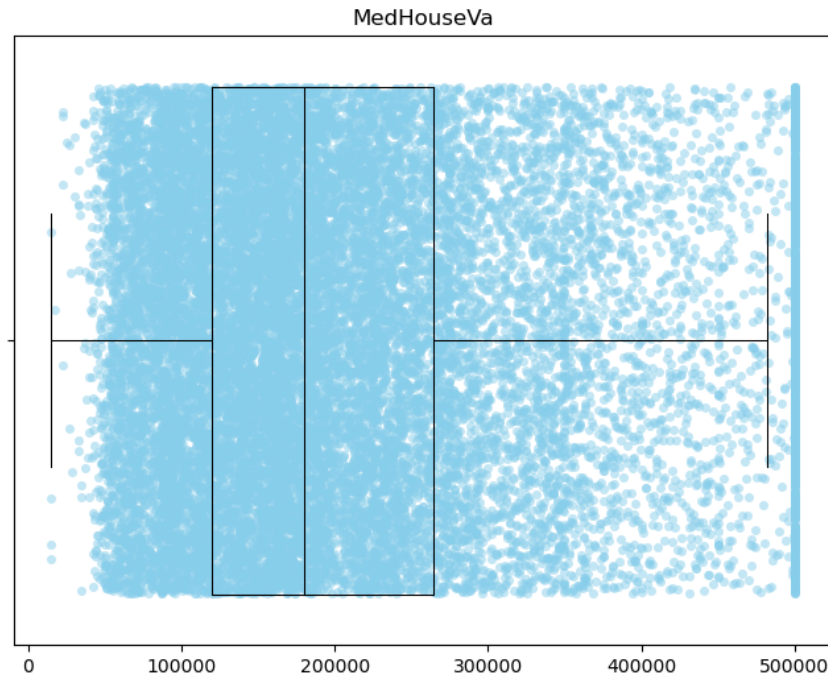


Figure 3.3.: Distribution plot for the dependent variable

The mean of the median house value is 206855.8, with the lowest price being 14999 and the highest one being 500001. There are a few outliers at the top, meaning a few median house values are extremely high. There are no outliers at the bottom. The values are spread out fairly even, with a definitive tendency for the values to be between 120000 and 265000.

3.4. Data exploration

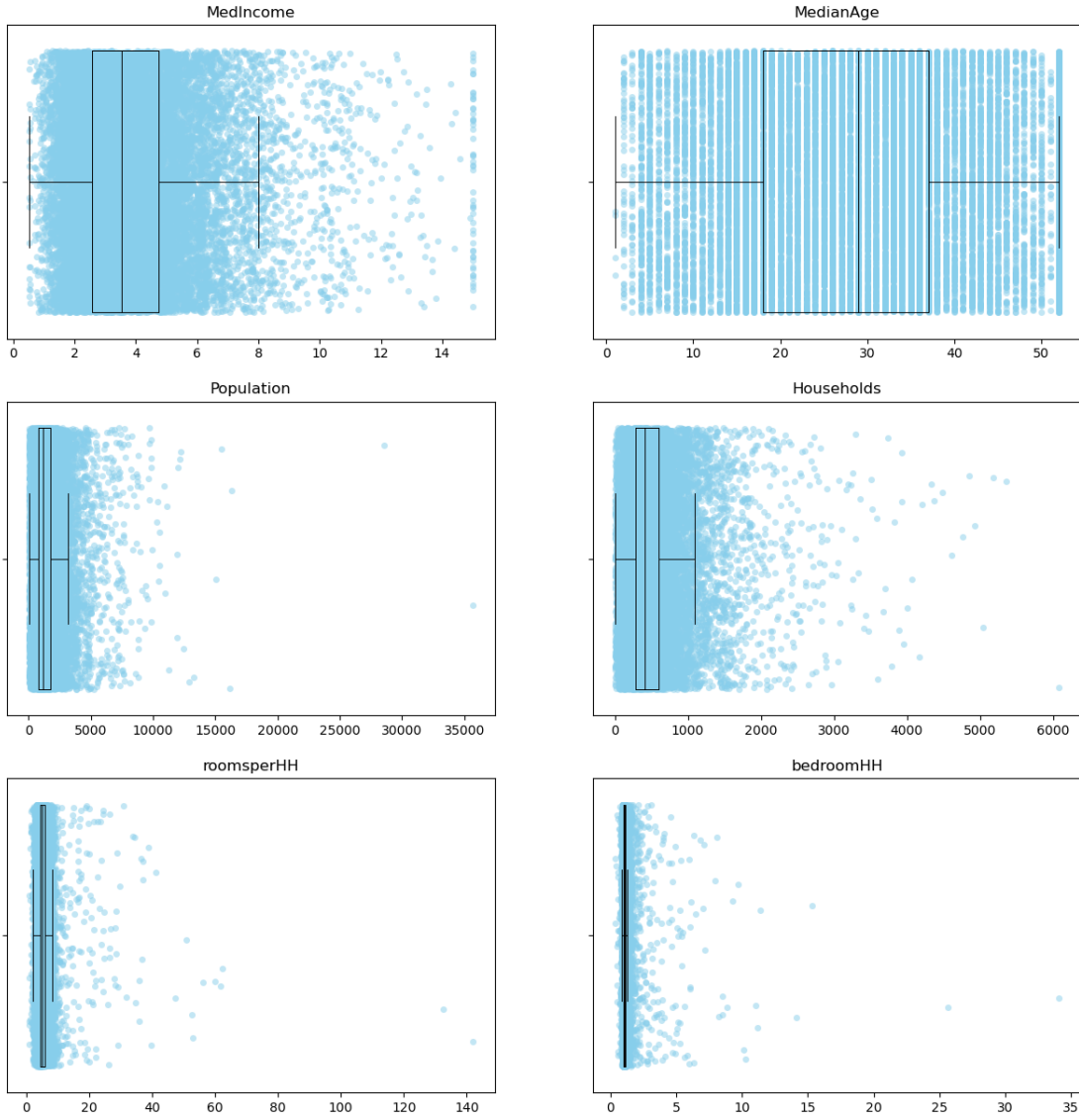


Figure 3.4.: Distribution plots for the independent variables

The independent variables show varying distributions. Only two of them could be described as fairly spread out. The median income (MedIncome) is given in units of 10000, and there are a few outliers at the top of the range. A lot of the values lie between 2.5 and 4.7, the distribution of values is positively skewed. The median age shows an interesting pattern, which is due to the values only being whole numbers. This variable shows a quite even distribution. The same cannot be said for the remaining 4 variables, whose values are all concentrated near the bottom of the range, with only a small number of outliers at the top. This is especially prominent for roomsperHH and bedroomHH.

3. Methodology

Interesting patterns in the dependent variable can also be uncovered by looking at a simple map (Figure 3.5)

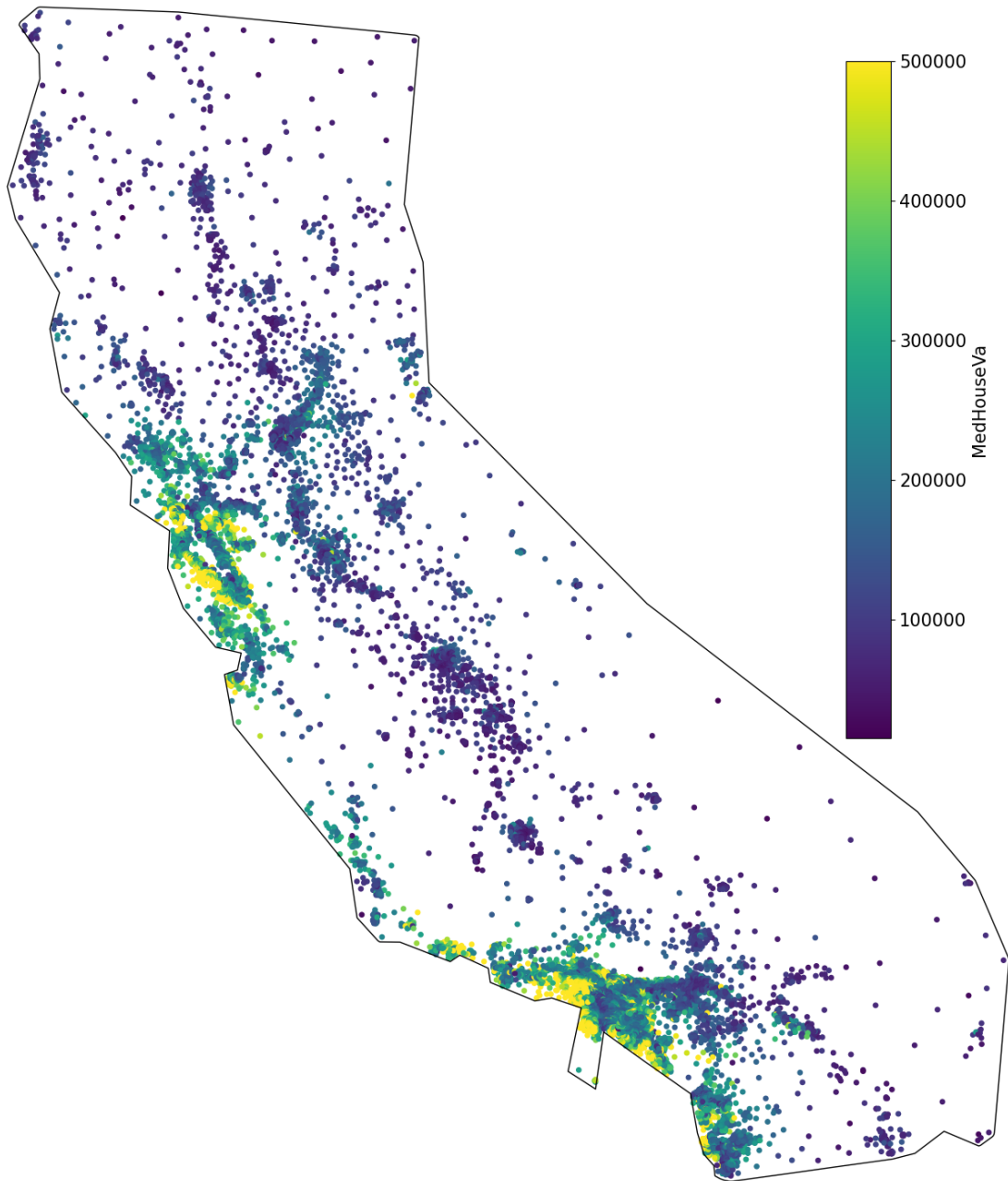


Figure 3.5.: Map depicting median house value

The map illustrates the spatial distribution of the dependent variable, median house value, across the study area. The color bar on the right hand side represents the values,

with colors at the top of the bar indicating higher values. There are a few obvious groups with very high values near the coast, and tendentially low values with increased distance to the coast.

3.5. Data Preprocessing

As is often the case, the data at hand cannot be used for analysis in the state they have been described in in the previous sections. As the data preparation took a respectable time and also showed to have major influence on the analysis and the results, the following section is dedicated solely to it.

The preprocessing can be structured in six steps:

1. The columns "total rooms" and "total bedrooms" are divided by the column "households", which will yield the number of rooms and bedrooms per household, instead of the total number. This step is not stringently required for Random Forest, but nonetheless implemented to give all the predictive methods the same chances (Raschka and Mirjalili 2017, 120f).
2. As the geographical information is given in latitude and longitude columns, these are converted to point geometries using GeoPandas.
3. All the overlapping point features are displaced using the QGIS processing tool "Points displacement", visible in Figure 3.6. In the top left, for example, there were four overlapping points. This was necessary because a Python function that is used later in the workflow creates Voronoi polygons for the point layer, and overlapping points would cause problems in this step.

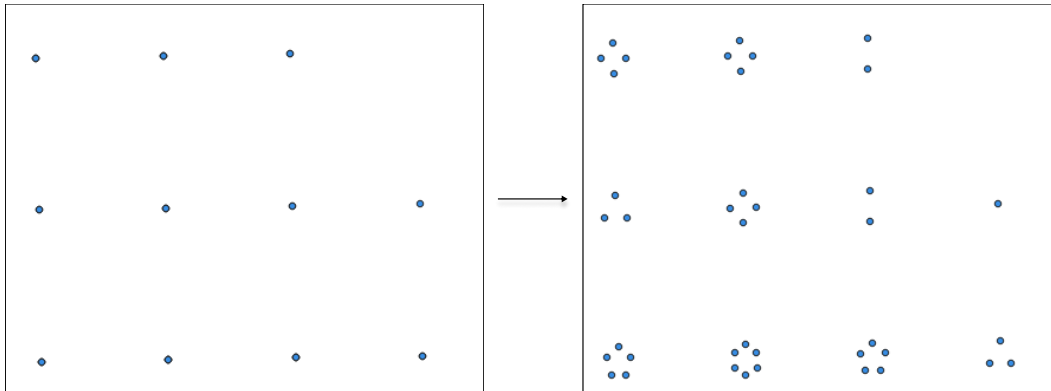


Figure 3.6.: Point displacement, screenshots from QGIS

4. In order to make sure the distance measures used in some of the processing and analysis steps are correct, the dataset is reprojected from degrees to a *projected*

3. Methodology

CRS (Anselin 2020). The one in use from here on is *NAD83 / California Albers* (EPSG: 3310).

5. To allow for comparison between the different methods under evaluation, identical train-test-splits need to be carried out. This would be a non-problem if only Python was used, because identical splits could be guaranteed using the random seed in the function *train-test-split* from scikit-learn. As this is not the case, the split is done as a preprocessing step, and the training and testing data are later read separately for each method.
6. To generate more reliable estimates on the predictive performance, a *nested-cross-validation* approach is imitated. By essentially splitting the data in three different ways and carrying the experiment out three times, this decreases the impact one specific train-test-split has (Raschka and Mirjalili 2017, 203f).

For each fold of the nested cross-validation, this leads to two shapefiles, one for training and one for testing, resulting in a total number of six shapefiles. In addition to that, .csv files are also saved, the reason for this being the simpler implementation of GW-RF when using this filetype as input. Again, one training and one testing file is generated for each train-test-split, resulting in six additional .csv-files and twelve files in total.

3.6. Workflow

This section describes all the steps in the modeling and prediction workflow. As this thesis compares a new approach to existing ones, it consists of multiple scripts, and their results are compared at the end. The elaborations on RegRF will be more extensive than the other ones, which is to be expected as it is the focal point of this work. All of the various methods are implemented with the exact same dataset and train-test-split, as described before. All of the scripts/notebooks with which these steps have been carried out can be found on GitHub and in the [Appendix](#).

3.6.1. Random Forest

General remarks on random forests have also already been made, namely in section 2.5.3. This workflow is not very extensive, as it only consists of reading the data, creating and choosing the model based on the training data using hyperparameter optimization (HPO), and then using the model to predict on unseen data. Again, Moran's I in the residuals and LISA are calculated.

3.6.2. Geographically Weighted Regression

As for the other two benchmark methods, elaborations on GWR have also been given already, in section 2.4.1. The workflow is similar to the other two benchmark methods. One difference is the use of numpy arrays instead of pandas dataframes, the reason for this is the nature of the used library and function. Additionally, an appropriate bandwidth

is selected by using functions included in the mgwr Python library. Residual spatial autocorrelation and LISA are calculated at the end.

3.6.3. Geographically Weighted Random Forest

Geographically Weighted Random Forest has been described in section 2.6.2. The .csv-files for training and testing are fed to the SpatialML R extension. This includes finding an appropriate number for m_{try} as well as the bandwidth. The prediction residuals are saved and imported to Python to allow for the exploration of spatial autocorrelation and LISA statistics.

3.6.4. Regionalized Random Forest (RegRF)

The workflow for RegRF is schematically visualized in figure 3.7. It can be categorized into five steps, which are described below the diagram:

1. Define parameters and carry out the regionalization procedure on the training data polygons. The choice was made to directly use polygons as the creation of the spatial weights matrix would automatically create Voronoi polygons when it is run on point data. Additionally, polygons are necessary for joining the test data points in the following step. As the number of regions to look for needs to be specified for every regionalization method except one, finding the best number is the first challenge that needs to be addressed. As mentioned in section 2.7.6, the plan is to use unsupervised evaluation metrics to assess the result and adapt it if necessary. The result of this is an additional column in the geodataframe which contains the number of the region this observation belongs to.
2. Introduction of the test data (points) to the workflow. To find out which region the observations in the test set are part of, a spatial join is carried out, resulting in the same additional column with the region number, which is of course necessary to be able to create a predictive model and evaluate it.
3. Carry out the first of three prediction approaches, called RegRF_CN (Cluster Number). This is almost identical to a normal RF, the only difference being the use of an additional column - the region number - as an independent variable. After fitting the model, predictions on the test set are carried out and reported.
4. The second prediction approach, which constructs one separate RF for each region, termed RegRF_SEP (Separate), and reporting of prediction performance on the test set.
5. Predict with the final version, called RegRF_W (Weighted). Here, there are multiple RF models, again one for each region, but with an important distinction: For each region, all of the training data is used, but more weight is put on the samples from the region the forest is currently being created for. More precisely, this means that each "region specific" RF will be based on all the other regions as well, but when

3. Methodology

training the forest, more data will be taken from that specific region than from other regions. The weight is decided from the number of samples in the region, meaning that for bigger regions, the weight will be bigger. When training the forest like this, all the test points will receive a prediction for each "region specific" model. This means that there are multiple predictions for each row, which is a problem for performance evaluation. This is solved by averaging all the predictions.

Depending on the regionalization results and the modeling with the training data, this process is repeated until a satisfactory regionalization result is obtained, after that it is used for predicting the test labels from the test features.

3.6.5. Comparison of the workflows and models

Judging from the elaborations given in this section, but also by looking at the jupyter-notebooks themselves, big distinctions can be made between RegRF and the benchmark methods. These established methods are fairly straightforward in their implementation and do not require extensive amounts of lines of code. The same cannot be said for RegRF. While it is essentially "just" carrying out a regionalization task beforehand, extra time and effort needed for preparations, debugging and implementation was necessary. In addition to that, three different approaches for prediction (RegRF_CN, RegRF_SEP, and RegRF_W) are evaluated, which again substantially increases the complexity of the process.

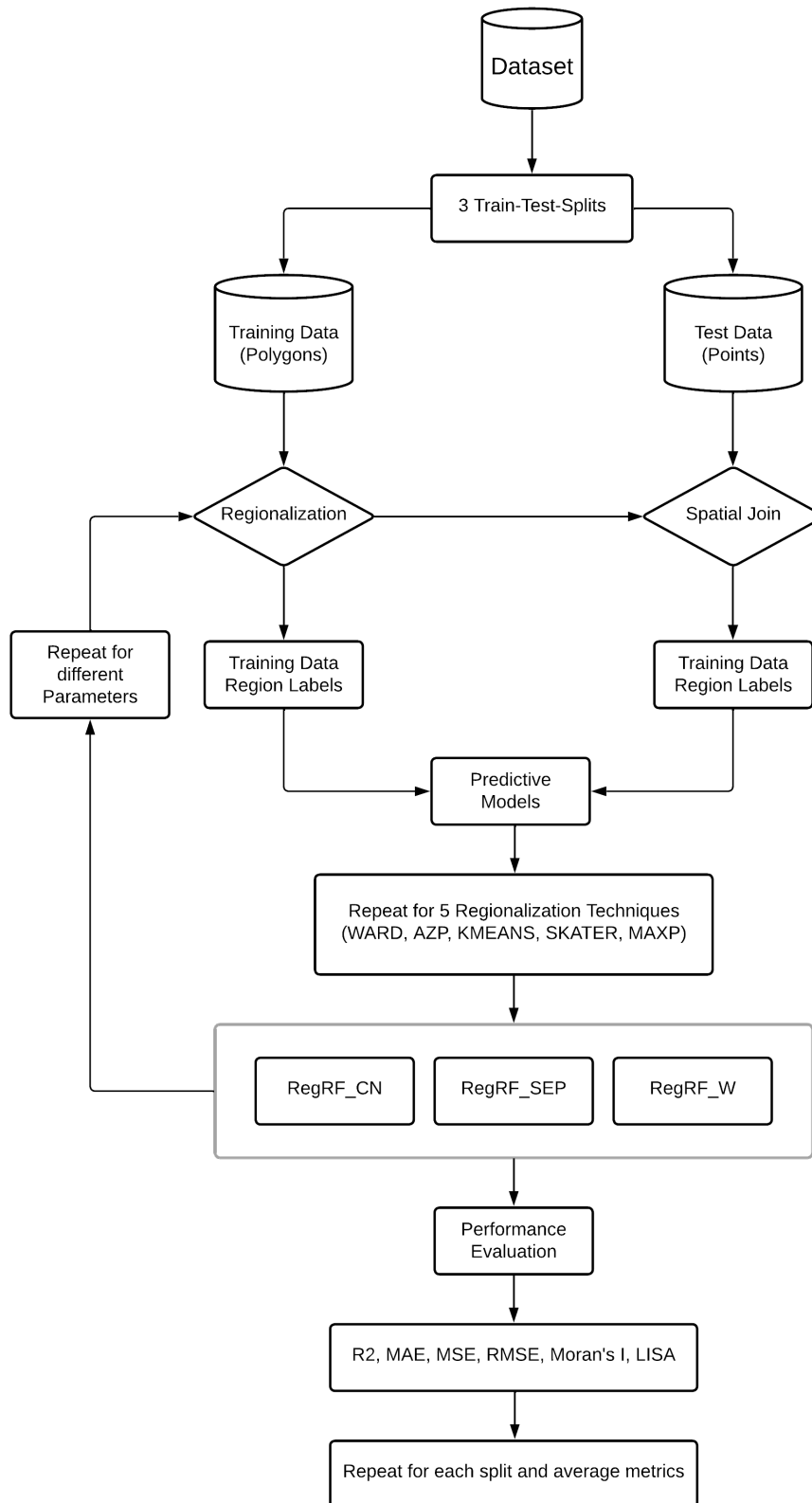


Figure 3.7.: RegRF Workflow

4. Results and Discussion

Chapter four evaluates and discusses the results of the practical part which has previously been described. Before getting into this, problems and limitations that only were realized after starting the workflow are given. This also includes how those were addressed. After that, the results of the different methods are given by listing the relevant performance metrics. Comparisons between the benchmark methods and RegRF are made after that. The chapter concludes by answering the research questions.

4.1. Observations while modeling

The modeling workflow was started by implementing the benchmark methods, as this was expected to work smoothly and deliver the first results quickly. No problems were met for RF and GWR. For GW-RF, the time and resources needed for running the script were more extensive. More exact numbers will be given in the next chapter, but the script took more than 100 hours to finish, despite the use of a quite powerful machine (i9 CPU, 256GB RAM). The RAM needed for saving all the local models exceeds the RAM available in most private devices, meaning it would not be possible to finish on a "casual" computer. It should be mentioned that this does not include finding the optimal value of m_{try} , which also took more than 15 hours. In addition to that, it was tried to use the built-in function for finding the best bandwidth, but this attempt had to be stopped after about 24 hours. Due to this, the decision was made to use the result of the bandwidth estimation in the GWR script. It is important to keep in mind not only for these problems but also upcoming difficulties, that the number of models that need to be created is quite large. This is not necessarily true for the benchmark methods, which were planned to be run for each split, totaling 9 models (three methods across three splits). Where this factor really comes into play is RegRF, as there are five regionalization methods which were scheduled to run for each split. 15 different models would also not be of great concern, but as many different combinations of parameters (e.g. number of regions), methods (e.g. AZP, S'KATER) and modeling approaches (RegRF_CN, RegRF_SEP, RegRF_W) need to be tested, the number of models for just one split can easily surpass 100. Scaling this to three splits again increases the complexity, which is something that was underestimated in the design of the experiment. This led to a myriad of problems, which, as well as how they were solved, will be elaborated in the next paragraphs. After that, the chapter will move on to listing the performance metrics for each of the methods.

From the five regionalization methods, only two were able to finish computation in a satisfying manner and reasonable amount of time, which is a prerequisite due to the trial-and-error nature of this experiment. Ward and AZP were able to deliver results quite

4. Results and Discussion

fast and allowed the workflow to keep going. Max-P, where the amount of regions to look for is not given by the user, is also able to compute, but only when setting the parameters in a way that makes the results impractical. Specifically, the minimum number of objects in a region needs to be set very low, resulting in a very high number of very small regions. The remaining two methods, k-means and SKATER, either did not run long enough to finish (k-means), or produced inexplicable results (SKATER). K-means was stopped after about 24 hours. Because of this, the experiment had to be carried out with just the methods which ran smoothly, these being Ward and AZP.

Of course, adaptations had to be made in order to account for these challenges. The most obvious approach is decreasing the sample size, which is the main reason for the very long runtime of some of the regionalization algorithms. The regionalization methods work with and have been evaluated on various smaller sample sizes in a multitude of publications, including Aydin et al. (2021), and Duque et al. (2007). Decreasing the sample size of course brings other new challenges. Firstly, it decreases the size of the training data, therethrough giving the models less information to learn from. Another problem in the spatial context is that interesting geographic variations or patterns can be lost. The decision was made to overlook these disadvantages in order to be able to evaluate all of the regionalization methods, as well as compare them with the benchmark methods. The sample size was decreased from 20640 to 2051, almost exactly ten percent. This process was carried out in QGIS, first all overlapping points were removed by averaging them into one point. After this, a new column was created and populated with random numbers, before randomly selecting one of these numbers and only keeping points with the corresponding number in the generated column. Of course, this also meant repeating the preprocessing steps of splitting the data, as described in steps 5 and 6 in section 3.5.

Another realization that was made when implementing RegRF and looking at preliminary results was the poor performance of RegRF_SEP compared to the other two variations. As the next paragraph will illustrate, there were a vast amount of models that needed to be run, which is why the decision was made to exclude RegRF_SEP from the workflow.

At this sample size, the regionalization algorithms were able to run in a feasible amount of time. To still be able to compare RegRF to benchmark methods, these were also tested again with the smaller sample size. While inspecting the first preliminary results, the realization was made that the unsupervised evaluation metrics listed in section 2.7.6 do not correspond with good performance of RegRF. More specifically, the best regionalization result - judging from those metrics - does not lead to the best performing RegRF. Once more this led to issues which needed to be solved. In order to optimize RegRF, all combinations of parameters, methods and modeling approaches (variations of RegRF) had to be tested. Even with the small sample size, some of the regionalization methods took a considerable amount of time to compute, so the only two methods running fast enough for this "trial-and-error" approach were chosen for the optimization. These are Ward and AZP. As a result of this, the workflow was left with three train-test-splits, two regionalization methods, and two modeling approaches, totaling twelve combinations. Additionally, the previously mentioned problem of the optimization of RegRF led to

testing all of these twelve combinations with the number of regions to find ranging from 2 to 70. This leads to testing 69 models for each of the twelve combinations, resulting in 828 models and their performance metrics, which will be elaborated in the next section and compared in section 4.3.

Here, it is also noteworthy to mention the metrics for evaluating the regionalization results which were discussed in section 2.7.6. These have shown to give good results when a small number of regions is created, which does not lead to the best performing predictive models. This means that these metrics are not useful for RegRF.

4.2. Performance evaluation

As mentioned in the last section, compromises regarding the tested methods had to be made. Even though the experiment could not be completed for the whole dataset, the obtained performance metrics are still given and evaluated. For the small dataset, there are more results available, so the evaluation and comparison of these performance metrics will be more extensive. In the tables depicting the performance metrics, the big dataset will be on the left side, and the small dataset on the right side. First, the predictive performance of each method will be evaluated separately, comparisons between the benchmark methods and RegRF will take place in the following section.

4.2.1. Random Forest

Split	R^2	MI-res	Insig. LISA (4128)		R^2	MI-res	Insig. LISA (411)
1	0.672	0.338	2873		0.636	0.294	312
2	0.676	0.373	3019		0.677	0.306	263
3	0.674	0.38	2510		0.643	0.243	279

Table 4.1.: Performance for RF across three different train-test-splits (Left: Full Dataset; Right: Small Dataset (10% sample)).

Random Forest is the only non-spatial method in the experiment. For the big dataset, the R^2 values are very similar across the three train-test-splits, hovering around 0.674. The spatial autocorrelation of the residuals is between 0.338 and 0.38, which indicates a trend of similar residual values being close to each other in space. Out of all 4128 data points in the test set, between 2500 and 3000 are insignificant, which is about 75% in the best performing case (Split 2). For the smaller dataset, the values vary a bit more depending on the split. Here, the second split has the best performance with 0.677. Interestingly, the global spatial autocorrelation of the residuals is also the most intense with 0.306. This can also be seen in Figure 4.1, which is a map depicting the local indicators of spatial association (LISA). On the coast, there are multiple hotspots, but

4. Results and Discussion

the most pronounced patterns are the coldspots in the interior of the state. In general, 265 of 411 data points are insignificant, about 64%.

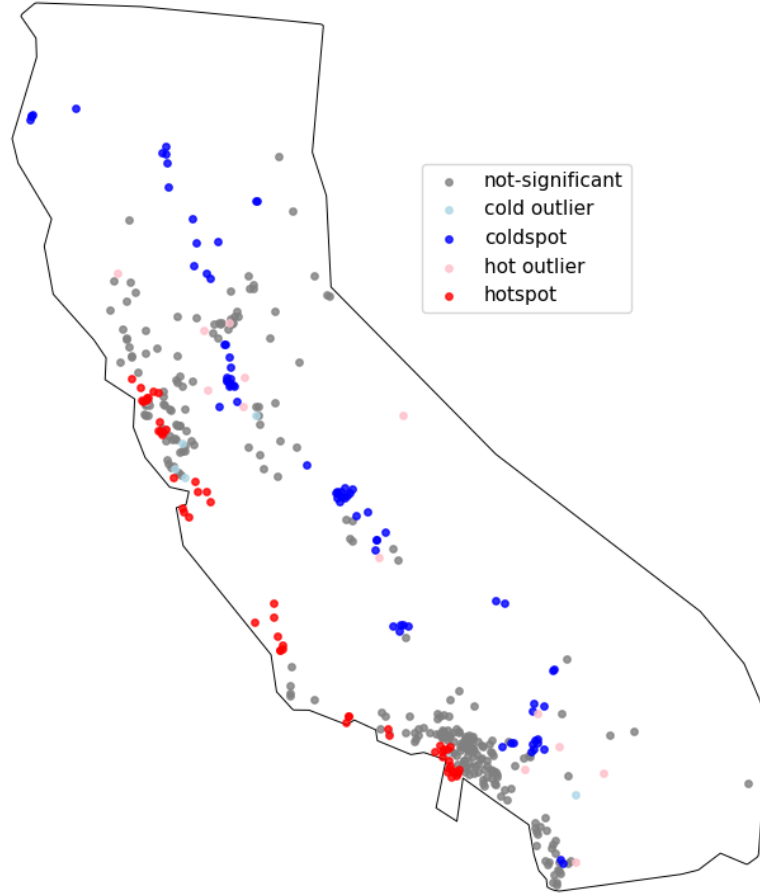


Figure 4.1.: LISA - Clusters of Residuals for Random Forest (Split 2 / Small Dataset)

4.2.2. Geographically Weighted Regression

The performance for GWR varies slightly depending on the train-test-split. For the big dataset, the R^2 values lie between 0.801 and 0.833, and they all show values very close to 0 for the spatial autocorrelation of the residuals, meaning that the residuals are spread nearly randomly across the study area. For all three splits, roughly 90% of LISA clusters are insignificant, which is a high value. This is also true for the small dataset (Figure 4.2), which shows a map of LISA-clusters for the third split of the small dataset. Here, 352 of 411 data points are insignificant, which is a high value at around 85%. As is expected with these values for the spatial performance metrics, no obvious patterns can be found when mapping the LISA clusters. There are a few hot- and coldspots here and there, but no bigger agglomerations of one or the other can be found. The R^2 values are lower for

4.2. Performance evaluation

the small dataset at 0.74 and 0.755. For this dataset, split 3 is the best performing one, which is also underpinned by the residual spatial autocorrelation at almost 0. Especially for the big dataset, the different splits of the data have a bigger impact on the R^2 values than they do for RF.

Split	R^2	MI-res	Insig. LISA (4128)		R^2	MI-res	Insig. LISA (411)
1	0.801	0.026	3685		0.74	0.043	352
2	0.827	0.04	3774		0.74	0.052	353
3	0.833	0.042	3731		0.755	0.006	360

Table 4.2.: Performance for GWR across three different train-test-splits (Left: Full Dataset; Right: Small Dataset (10% sample)).

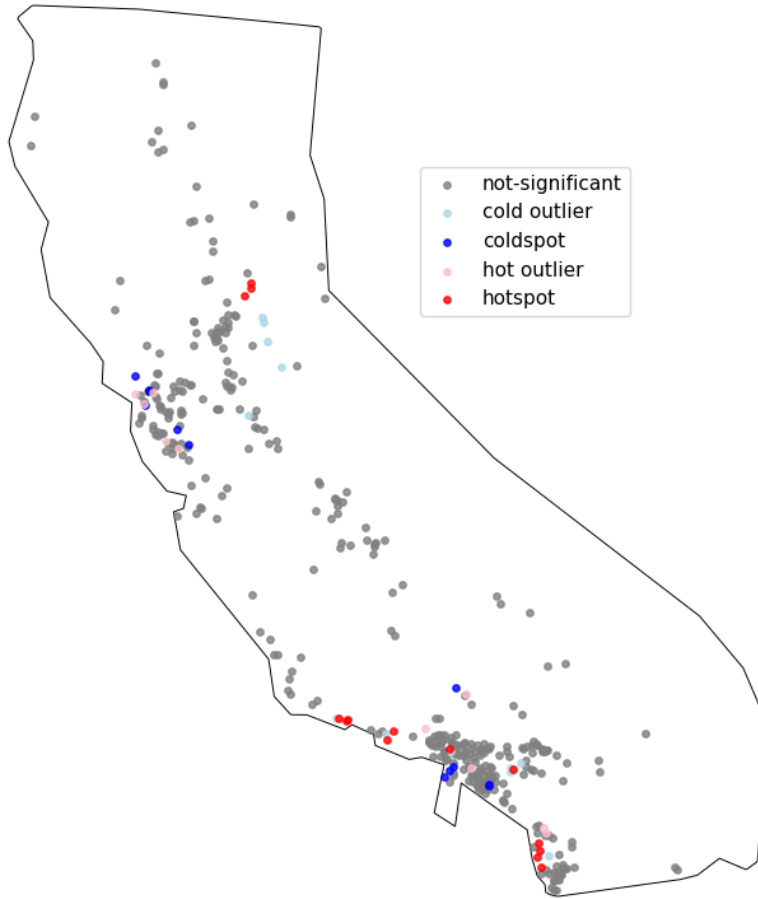


Figure 4.2.: LISA - Clusters of Residuals for GWR (Split 3 / Small Dataset)

4.2.3. Geographically Weighted Random Forest

As mentioned earlier, GWRF was not able to finish computation for the whole dataset, which is why only results for the small dataset can be given here.

Split	R^2	MI-res	Insig. LISA (4128)	R^2	MI-res	Insig. LISA (411)
1	-	-	-	0.758	0.12	327
2	-	-	-	0.774	0.07	355
3	-	-	-	0.765	0.07	356

Table 4.3.: Performance for GW-RF across three different train-test-splits (Left: Full Dataset; Right: Small Dataset (10% sample)).

The R^2 values are fairly similar in each of the three splits, ranging from 0.758 to 0.774. The Moran's I values in the residuals are very close to zero, indicating spatial randomness, which is desired in spatial modeling. The map of the LISA-clusters (Figure 4.3) shows one big hotspot in the southern area of the state. Apart from that, a few small coldspots can be seen in the northern part of the coast. Overall, 355 of 411 data points are insignificant, which is 86%.

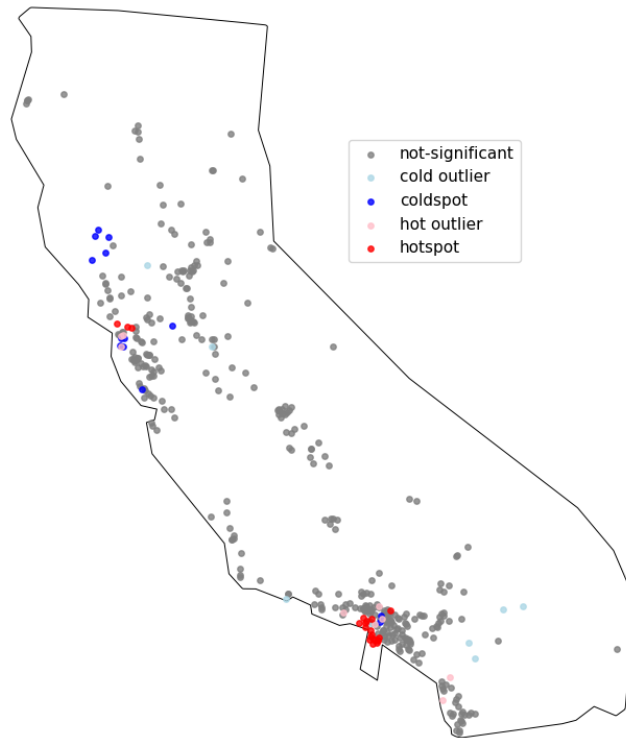


Figure 4.3.: LISA - Clusters of Residuals for GWRF (Split 2 / Small Dataset)

4.2.4. Regionalized Random Forest (RegRF)

As there are five regionalization methods to test, this subsection presents five tables. For all but one regionalization method, the results are only given for the small dataset. Reasons for this have been laid out previously, the main reason being the long time it takes the other methods to compute, which makes the experiment impossible to carry out due to its trial-and-error nature. Here, only a selection of the best performers (regarding parameters and modeling approaches) is given, which is justified as the benchmark methods also include steps to boost their performance by selecting the best models. This selection is the modeling approach "CN", where the region number is used as an additional independent variable in the predictive model, as well as 23 regions, which has shown to produce the best performance metrics. A complete table with all results from all tested combinations is available in the [appendix](#).

The first variation to evaluate is the variation of RegRF which uses the *WARD* regionalization method. As with all the examples given here, the modeling approach is "CN" with 23 regions. This is also the only case where experiments with RegRF are feasible when using the big dataset. The R^2 values of this are 0.78 for split 1, and almost 0.8 for splits 2 and 3. The spatial autocorrelation of the residuals is the lowest in split 2 with 0.216, which still indicates a tendency of similar residual values being close to each other in space. Here, the number of insignificant LISA clusters is also the highest with 79%. A map depicting these can be seen in Figure 4.4. There, a lot of hotspots can be seen along the coast of California. Very close to those, a few coldspots can be seen as well. In the interior of the state, the patterns are less pronounced, just a few coldspots can be found here and there.

For the small dataset, the R^2 values are lower, which is to be expected given the significantly smaller size of the training data. Nonetheless, the Moran's I of the residuals shows a more random distribution of the residuals in space. Interestingly, the model with the best R^2 value at 0.758 in split 3 shows the highest degree of spatial autocorrelation in the residuals as well as the lowest number of insignificant LISA clusters at 73%. Regarding this, split 1 performs a lot better, where 83% of those are insignificant. Because of this, the choice was made to map split 1 (Figure 4.5). Inspecting this map shows one big hotspot in the northern half of the coast, as well as a few hotspots in the south, where some coldspots can be found as well.

Split	R^2	MI-res	Insig. LISA (4128)		R^2	MI-res	Insig. LISA (411)
1	0.78	0.24	3182		0.749	0.129	343
2	0.797	0.216	3274		0.716	0.118	319
3	0.797	0.253	3133		0.758	0.167	302

Table 4.4.: Performance for RegRF (Ward - CN) across three different train-test-splits with 23 regions (Left: Full Dataset; Right: Small Dataset (10% sample))

4. Results and Discussion

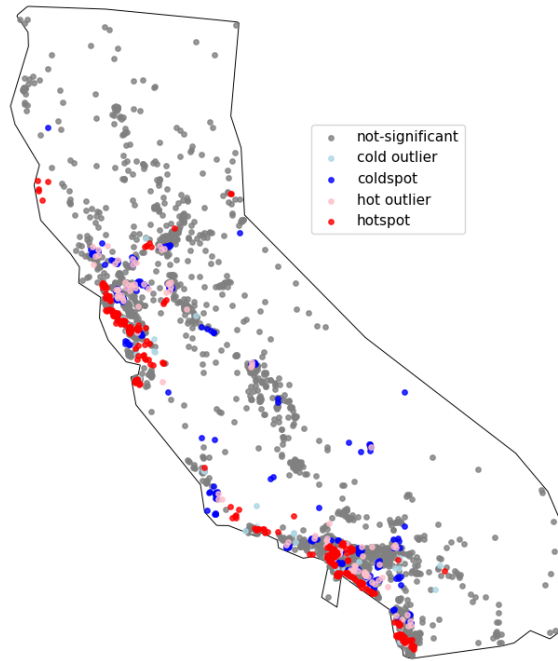


Figure 4.4.: LISA - Clusters of Residuals for RegRF (WARD / Split 2 / Big Dataset)

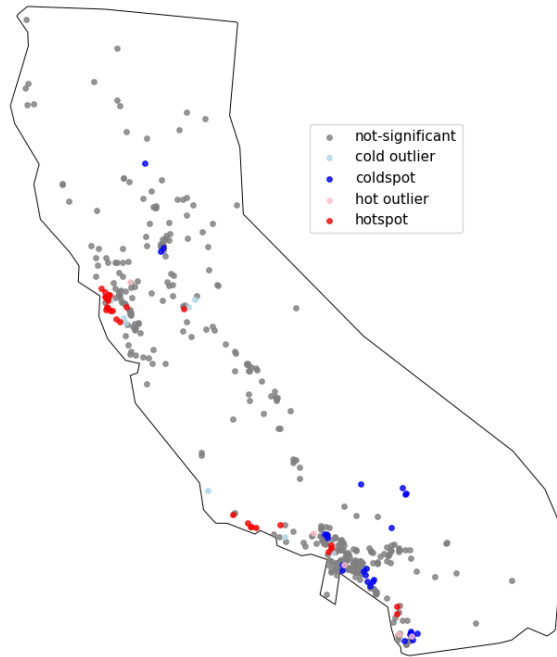


Figure 4.5.: LISA - Clusters of Residuals for RegRF (WARD / Split 1 / Small Dataset)

4.2. Performance evaluation

For all the remaining regionalization methods (AZP, SKATER, KMEANS and MAX-P) computation was only possible or sensible with the small dataset. For AZP, the parameter combination shown here varies largely depending on the split, fluctuating by as much as 10% between split 1 and 2. Again, this is possibly due to the size of the training data. In the best case, the values for the insignificant LISA clusters are quite high with 354 out of 411, which is 86%, a high number which is desired. It is noteworthy that this is also the case with the lowest R^2 value. The model with the highest R^2 performs the worst regarding the spatial performance metrics. To be able to inspect this phenomenon more closely, both split 1 and 2 are shown next to each other in figure 4.6. Upon inspection of these maps, a lot of similarities can be found. Firstly, there are hotspots in the northern half of the coast in both splits, as well as a few coldspots in the southern half and in the interior of the state. The coldspots in the interior are more pronounced in split 1, while the coldspots in the south are more prominent in split 2.

Split	R^2	MI-res	Insig. LISA (4128)	R^2	MI-res	Insig. LISA (411)
1	-	-	-	0.659	0.123	354
2	-	-	-	0.727	0.141	334
3	-	-	-	0.717	0.113	337

Table 4.5.: Performance for RegRF (AZP - CN) across three different train-test-splits with 23 regions (Left: Full Dataset; Right: Small Dataset (10% sample))

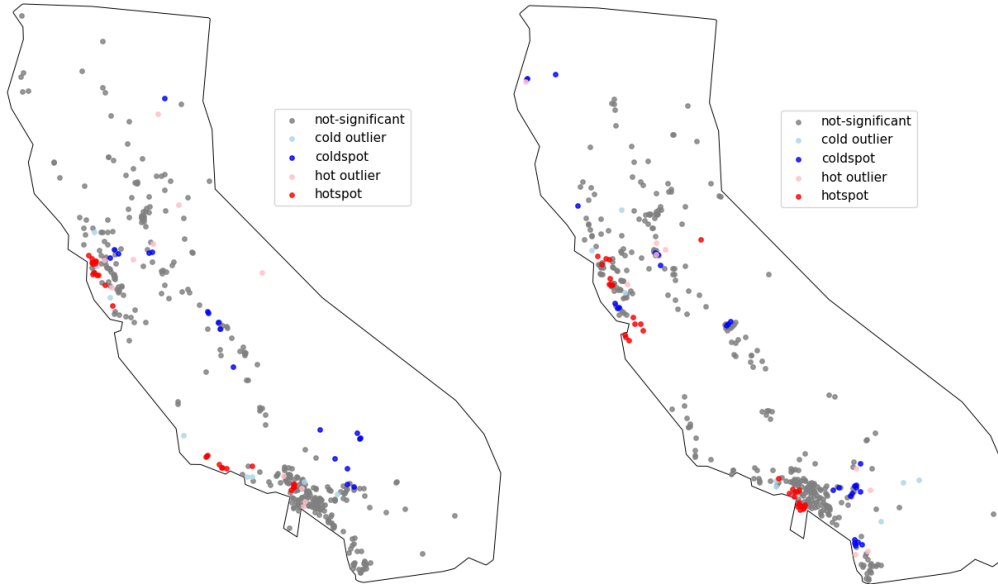


Figure 4.6.: LISA - Clusters of Residuals for RegRF (AZP / Split 1 (left) + 2 (right) / Small Dataset)

4. Results and Discussion

The next regionalization method to use for RegRF is SKATER. Here, some more processing was needed as the algorithm automatically increases the number of regions and produced a few island - regions, which are regions with just 1 object. These were merged with the biggest neighbor. Nonetheless, the results are not all bad, especially with split 2 showing an R^2 of 0.75. The spatial autocorrelation in the residuals is moderate, with the most intensely clustered residual values being in split 3. The best value for the LISA clusters can be found in split 1, where 76% are insignificant. For SKATER, split 1 is shown in Figure 4.7, which shows a pattern similar to the other LISA maps. Hotspots can be discovered along the coast, and coldspots can be found in the interior of the state.

Split	R^2	MI-res	Insig. LISA (4128)	R^2	MI-res	Insig. LISA (411)
1	-	-	-	0.714	0.16	316
2	-	-	-	0.75	0.159	301
3	-	-	-	0.707	0.214	304

Table 4.6.: Performance for RegRF (SKATER - CN) across three different train-test-splits with 23 regions (Left: Full Dataset; Right: Small Dataset (10% sample))

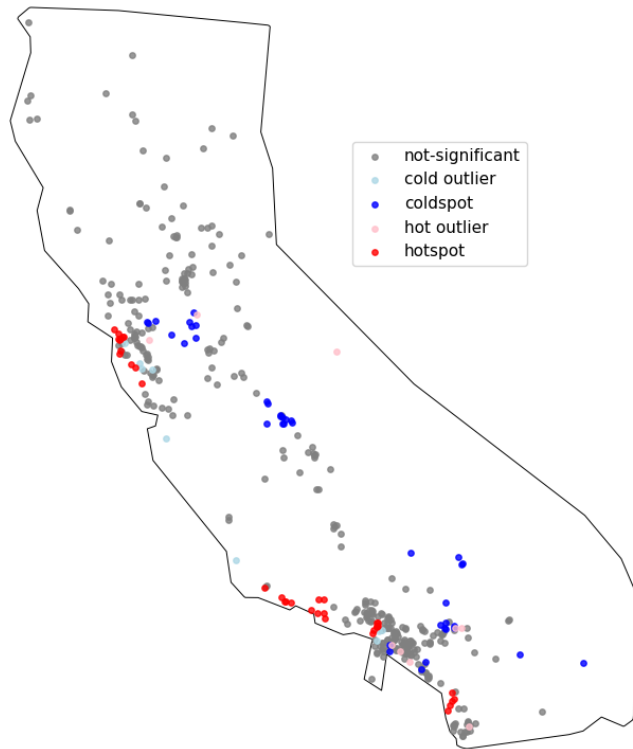


Figure 4.7.: LISA - Clusters of Residuals for RegRF (SKATER / Split 1 / Small Dataset)

The last regionalization method that requires the user to input the number of regions to find is KMEANS. For this, the R^2 values range from 0.702 to 0.734. The residuals show a tendency to be clustered, especially for splits 2 and 3. Despite this, the number of insignificant LISA clusters is quite high around 330 of 411, which is approximately 80% for all three splits. This can be inspected more closely in Figure 4.8, where the LISA clusters of split 2 are shown. There, some interesting variations to the other methods can be seen. The most prominent is that hotspots can be seen in the middle part of the coast as well. Here, the coldspots in the interior of the state are very well defined and they stretch all the way to the southern border. Generally, the hot- and coldspots are well separated when using RegRF with KMEANS.

Split	R^2	MI-res	Insig. LISA (4128)	R^2	MI-res	Insig. LISA (411)
1	-	-	-	0.703	0.216	323
2	-	-	-	0.734	0.238	327
3	-	-	-	0.702	0.232	333

Table 4.7.: Performance for RegRF (KMEANS - CN) across three different train-test-splits with 23 regions (Left: Full Dataset; Right: Small Dataset (10% sample))

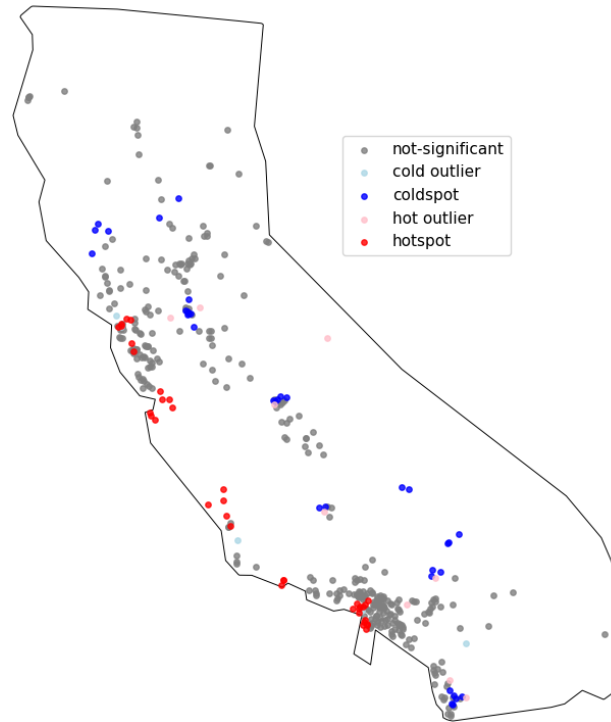


Figure 4.8.: LISA - Clusters of Residuals for RegRF (KMEANS / Split 2 / Small Dataset)

4. Results and Discussion

The final regionalization method is MAX-P, which takes a threshold (maximum number of objects per region). To arrive at a somewhat similar or comparable result, experiments were carried out with various threshold values to arrive at the same number of regions as the other methods. Here, the R^2 values lie between 0.705 and 0.738. The spatial performance metrics also show some fluctuation, with the lowest value of Moran's I at 0.137 and the highest one at almost 0.25, which indicates a tendency for similar residual values to be close to each other in space. In the worst case, 71% of LISA clusters are insignificant, and in the best case 78% of them are insignificant. In this case, split 3 performs best across all performance metrics given here, which is why the choice was made to depict it in Figure 4.9. Upon inspection, a well separated pattern can be seen. There are a few coldspots along the coast and again coldspots in the interior, the pattern of whose looks different than in the other maps, as the coldspots stretch all the way across the state from the southeast to the northwest.

Split	R^2	MI-res	Insig. LISA (4128)	R^2	MI-res	Insig. LISA (411)
1	-	-	-	0.705	0.248	306
2	-	-	-	0.734	0.228	293
3	-	-	-	0.738	0.137	323

Table 4.8.: Performance for RegRF (MAX-P - CN) across three different train-test-splits with 23 regions (Left: Full Dataset; Right: Small Dataset (10% sample))

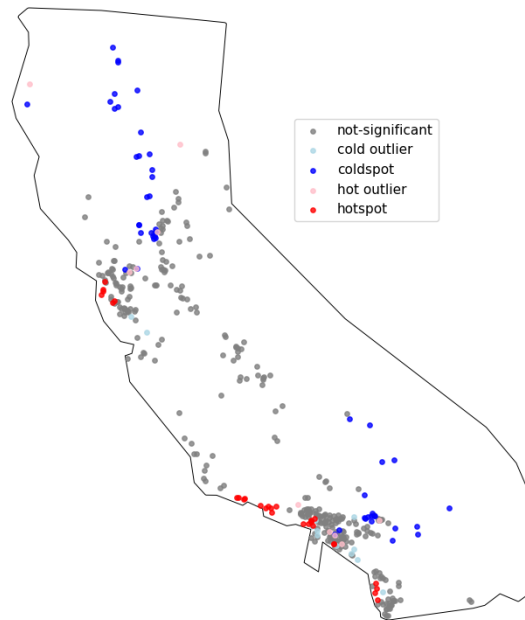


Figure 4.9.: LISA - Clusters of Residuals for RegRF (MAXP / Split 3 / Small Dataset)

4.3. Performance Comparisons

So far, the benchmark methods and variations of RegRF have only been evaluated on their own, without comparisons being made, which is the purpose of the following section. Before diving into this, a few things need to be discussed, one of which is that the comparison will be twofold.

Firstly, the different variations of RegRF will be compared to each other in order to be able to explore the impact the different regionalization techniques, modeling approaches and parameters have on the performance metrics of the predictive models. This will be done in a multitude of tables due to the number of variations that need to be compared, which would be unfeasible to do in one table due to its size. It should also be noted that the comparisons here only affect the small dataset, as for the big dataset only one variation of RegRF was able to finish computation. After this, a selection of those variations will be compared to the benchmark methods. Only comparing a selection of the best RegRF variations is valid for the following reason: All the benchmark methods include procedures to boost their performance, in other words they are optimized. For RF, this happens via hyperparameter optimization, and for GWR and GW-RF this is done by optimizing the bandwidth in the modeling workflow. As these methods are more mature, ways to optimize the results are built into the workflow, which is not the case for RegRF. Here, a vast amount of combinations had to be tested manually, before being able to choose the best models. Nonetheless, this can be seen as a manual optimization procedure, and all the evaluated performance metrics can be characterized as optimized. It is important to note that significance tests have been conducted to assess the differences between the compared models in cases where this procedure is fitting, and these results influence the findings presented in this chapter.

4.3.1. Comparing Variations of RegRF

Table 4.9. lists the R^2 values as well as the number of insignificant LISA clusters for all regionalization methods and the modeling approach "CN". For each regionalization algorithm, 23 regions were produced. The decision to only list these two performance metrics - one nonspatial and one spatial - was made in order to preserve a degree of clarity and simplicity, as mentioned before a more complete list can be found in the [appendix](#). The three splits listed are identical to the tables in the previous sections, a new addition here is the last row which represents the average R^2 and LISA clusters for each method, which is very helpful for comparing them. Looking at those values shows that WARD performs the best regarding the R^2 values, followed by MAXP with SKATER being very close to it. KMEANS, but especially AZP are lacking behind in this category. One possible reason for this has been mentioned in a previous part of this work, which is that the numbers given here were optimized using the WARD regionalization algorithm. Given this fact, both SKATER and MAXP produce very respectable outcomes.

When looking at the number of insignificant LISA clusters, the performance of the respective algorithms is very different. For this performance metric, AZP performs the best, where 342 out of 411 LISA clusters (83%) are insignificant. This is interesting, as

4. Results and Discussion

AZP produced the worst R^2 values. The second and third highest values for the LISA clusters can be found for KMEANS and WARD, and SKATER and MAXP perform the worst in this category with only 307 of 411 (75%) clusters being insignificant. Maps depicting these LISA clusters have been presented in the previous section.

Another aspect that deserves some attention is the runtime of the algorithms. There are big differences to be found, for example WARD is very fast and takes about 13 seconds for this dataset. AZP is able to finish computation in about 170 seconds, SKATER and KMEANS both need about 130 seconds to deliver a result. By far the longest computation time can be found for MAXP, which takes more than 40 minutes. One reason for this could be the fact that MAXP determines the number of regions on its own.

	WARD CN		AZP CN		SKATER CN		KMEANS CN		MAXP CN	
Split	R^2	LISA	R^2	LISA	R^2	LISA	R^2	LISA	R^2	LISA
1	0.749	83%	0.659	86%	0.714	77%	0.703	79%	0.705	74%
2	0.716	78%	0.727	81%	0.75	73%	0.734	80%	0.734	71%
3	0.758	73%	0.717	82%	0.707	74%	0.702	81%	0.738	79%
$\bar{X}$	0.741	78%	0.701	83%	0.724	75%	0.713	80%	0.726	75%

Table 4.9.: Performance for RegRF (all regionalization methods - CN) across three different train-test-splits with 23 regions, small dataset. LISA: Percentage of insignificant clusters of residuals.

The second modeling approach in use is "CN_W". As a reminder, this creates a separate model for each region which still uses all of the data from all regions, with the distinction that more weight is put on the data from the region that is being modeled. The weight is decided by the size of the region, meaning that for large regions the weight will be bigger. This was explained in more detail in section 3.6. The results of this are quite similar to the previous approach, with the difference that MAXP slightly outperforms WARD regarding the R^2 values, the remaining methods perform in the same order. This means that SKATER follows MAXP and WARD as the third best performing combination, followed by KMEANS. Once more, AZP is lacking behind regarding this performance metric. When looking at the LISA values, the results are again quite different from the R^2 values, but also different from the LISA values from RegRF_CN. Here, the difference between the respective method is small, with the best performer in KMEANS producing 339 out 411 (82%) insignificant LISA clusters. This is followed very closely by SKATER, MAXP and WARD. Even the worst performing method in this regard, which is AZP, is close to the best values and produced 330 insignificant LISA clusters (80%). To sum up, all values for the LISA clusters lie between 330 and 339 when using the approach CN_W, which are all pretty high values at more than 80%.

For this approach, the runtime of the regionalization methods itself remains unchanged,

4.3. Performance Comparisons

as the preprocessing in which the algorithms are run is identical. The only thing that changes regarding the computational effort is the construction of multiple models, which adds about 1 second per region. In this case, where 23 regions are formed, this means that approximately 23 seconds need to be added to the runtime for each method. Due to this, the differences between the respective regionalization algorithms remain unchanged regarding this aspect. Comparing the predictive performance of RegRF_CN to RegRF_CN_W, it can be seen that the R^2 values are similar, with a slight advantage for RegRF_CN. For the number of insignificant LISA clusters, RegRF_CN_W performs better.

	WARD CN_W		AZP CN_W		SKATER CN_W		KMEANS CN_W		MAXP CN_W	
Split	R^2	LISA	R^2	LISA	R^2	LISA	R^2	LISA	R^2	LISA
1	0.739	85%	0.674	81%	0.705	82%	0.695	81%	0.707	77%
2	0.698	84%	0.72	77%	0.744	85%	0.732	84%	0.746	84%
3	0.740	75%	0.707	83%	0.701	80%	0.7	83%	0.728	84%
$\bar{X}$	0.726	81%	0.7	80%	0.717	82%	0.709	82%	0.727	82%

Table 4.10.: Performance for RegRF (all regionalization methods - CN_W) across three different train-test-splits with 23 regions, small dataset. LISA: Percentage of insignificant clusters of residuals

4.3.2. Comparing RegRF to Benchmark Methods

Previously, only comparisons between the different approaches of RegRF itself were made. This section focuses on comparing RegRF to the chosen benchmark methods, which are Random Forest (RF), GWR (Geographically Weighted Regression), and GW-RF (Geographically Weighted Random Forest). As discussed before, these methods are able to optimize their outputs, which is why the best combination of parameters and methods obtained by RegRF will be used for this comparison. This is the modeling approach CN with 23 regions and the regionalization algorithm WARD. Also, as this combination was also able to run when using the big dataset, choosing this combination will allow to compare RegRF to the benchmark methods for both the big and the small dataset. First, the models using the small dataset will be compared with the table below:

The first obvious takeaway from this is the underwhelming performance of the traditional RF as the only non-spatial method, regarding both the R^2 values and the number of insignificant LISA clusters. The best performance regarding the R^2 is obtained by GW-RF with a value of 0.766. RegRF is the third best by a close margin (0.004), being slightly worse in this regard than GWR. As these two are very close, looking at the LISA clusters can give additional insights, which shows that GWR performs better regarding

4. Results and Discussion

this performance metric (78% vs 86%). In fact, GWR also outperforms GW-RF in this category.

Taking a closer look at the train-test-splits can also uncover some interesting patterns: This shows that the variance for R^2 across the three splits is the highest for RegRF when looking at the spatial methods. For RegRF, it is $5e-4$, while for GWR and GW-RF the variance is $8e-5$ and $6e-5$, respectively. This means that both GWR and GW-RF do not seem to be that reliant on a "fitting" dataset, which is likely due to their integrated optimization mechanisms. In contrast to that, the optimization for RegRF was done manually which very likely leads to it not being able to account for differences in the data in the same manner. This is also underlined by the following: For GWR and GW-RF, the bandwidth value was estimated separately for each split (82 for splits 1 + 2, and 77 for split 3). For RegRF, all parameters were exactly the same in all three splits. So, GWR and GW-RF are better at adapting to the data at hand which is likely to lead to their results having a smaller variance and higher performance across the splits.

For the comparison to the benchmark methods, the time the methods need to deliver results are an important factor. The variation of RegRF under investigation here takes about 13 seconds to finish, while the nonspatial RF has a runtime of 10 seconds. GWR needs 11 seconds to deliver results, and GW-RF finishes computation in 51 seconds. These numbers show that RegRF can significantly increase model performance while adding very little computational effort in comparison to Random Forest. GWR takes a few seconds longer, but only increases the accuracy by a small amount. GW-RF takes by far the longest, a tradeoff that has to be made if the user wants to obtain the best performing model.

	WARD CN		RF		GWR		GW-RF	
Split	R^2	LISA	R^2	LISA	R^2	LISA	R^2	LISA
1	0.749	83%	0.636	76%	0.74	86%	0.758	80%
2	0.716	78%	0.677	64%	0.74	86%	0.774	86%
3	0.758	73%	0.643	68%	0.755	88%	0.765	86%
$\bar{X}$	0.741	78%	0.652	69%	0.745	86%	0.766	84%

Table 4.11.: Performance for RegRF (WARD - CN - 23 Regions) + Benchmark Methods across three different train-test-splits with 23 regions, small dataset. LISA: Percentage of insignificant clusters of residuals

Additional insights can be gathered by inspecting the following table which shows the results for the big dataset:

	WARD CN		RF		GWR		GWRF	
Split	R ²	LISA	R ²	LISA	R ²	LISA	R ²	LISA
1	0.78	77%	0.672	70%	0.801	89%	-	-
2	0.797	79%	0.676	73%	0.827	91%	-	-
3	0.797	76%	0.674	61%	0.833	90%	-	-
$\bar{X}$	0.791	77%	0.674	68%	0.82	90%	-	-

Table 4.12.: Performance for RegRF (WARD - CN - 23 Regions) + Benchmark Methods across three different train-test-splits with 23 regions, big dataset. LISA: Percentage of insignificant clusters of residuals

In this case, the performance across all methods is higher, which can be attributed to the substantially increased amount of data that is available for training the models. At the same time, the differences between the respective methods are more pronounced. As mentioned before, GW-RF was not able to finish computation for this dataset, so GWR is the best performing method with an average R² of 0.82. This is again followed by RegRF at 0.791, and the non-spatial RF is way behind at 0.674. This is also reflected by the number of insignificant LISA clusters, which is higher than for the small dataset at 90% for GWR, but a bit lower for RegRF at 77%. Random Forest is again outperformed significantly, where just 67% of LISA clusters are insignificant. A collection of maps, one for each method, can be seen in Figure 4.10, where the residual values are mapped onto a basemap.

The differences between the train-test-splits regarding the R² values tend to be less pronounced with the big dataset, which makes sense due to the bigger size of the dataset, whereby the choice of which samples to use for training and testing becomes less impactful. For RegRF, it is lower than for the small dataset at 1e-4. For RF, the performance is very constant with a variance of 4e-6. The biggest differences can be found for GWR with a variance of 3e-4, which is in fact higher than for the small dataset. Not just for GWR, also for RegRF the first split underperforms in comparison to the other two splits.

The runtimes for some of these methods drastically increase when increasing the sample size. For the big dataset, Random Forest needs 41 seconds to finish, the configuration of RegRF under investigation here follows closely behind this at 45 seconds. GWR also sees a substantial increase in computational effort necessary, as it needs more than 8.5 minutes to arrive at a result. As has been discussed before, GW-RF was not able to finish computation at all. These aspects show that RegRF can be advantageous in the scenario

4. Results and Discussion

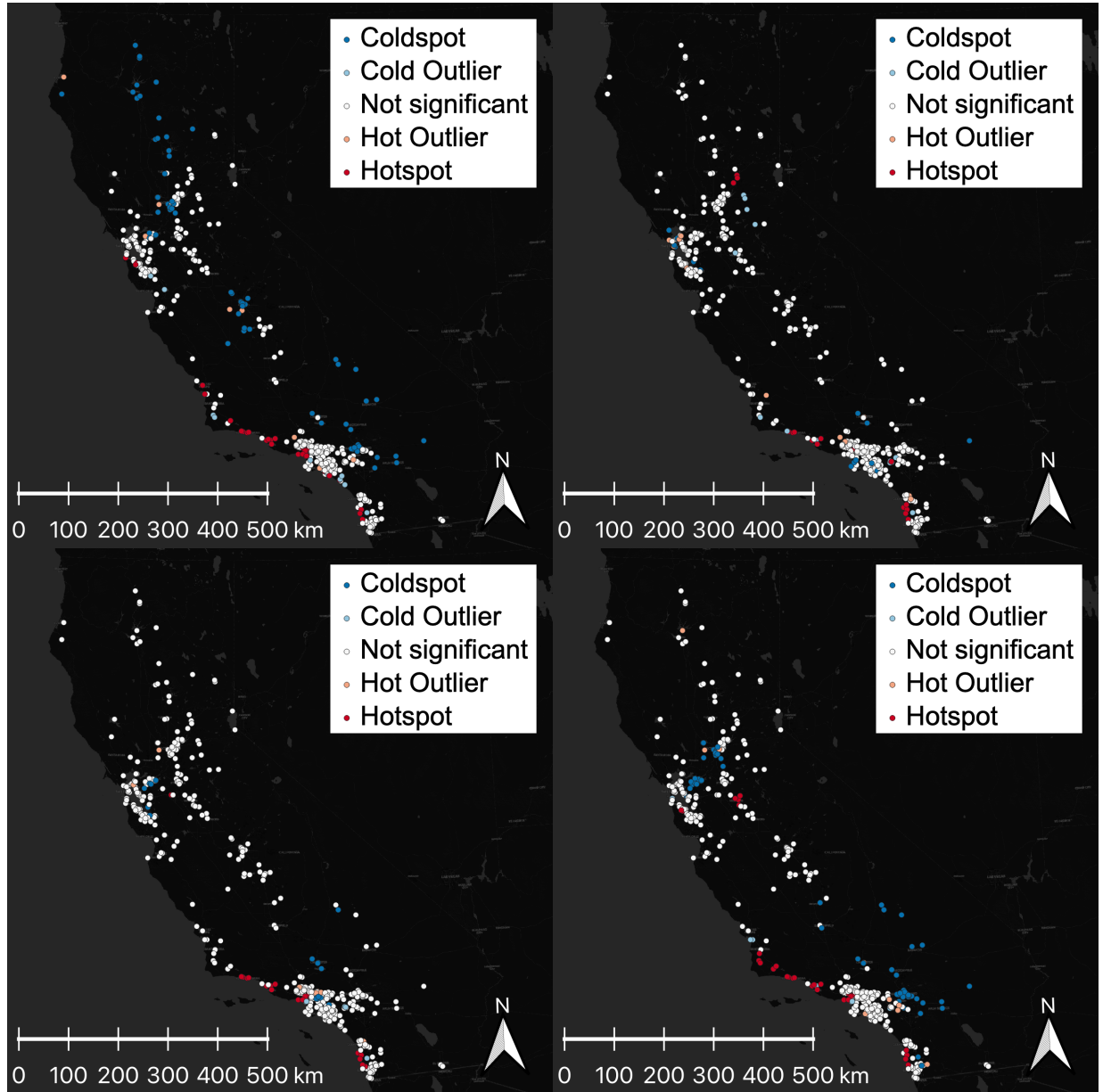


Figure 4.10.: LISA clusters of residuals (Split 3, Small Dataset): Top left: RF; Top right: GWR; Bottom left: GW-RF; Bottom right: RegRF (WARD_CN)

at hand, as its runtime is only slightly higher than RF, while substantially increasing the performance of the predictive model. GWR is more accurate, but also takes more than 11 times longer to deliver a result.

4.4. Answering the Research Questions

This chapter intends to answer the research questions stated at the beginning of this thesis. Implicitly, these have been answered over the course of the last few subsections, but answering them explicitly is desired in order to present the findings in a more organized manner. For this, each of the research questions, which have been stated in chapter 1.3.1, will be mentioned again before answering them individually. These answers will focus on the main findings, more details have been given in the previous section. In the introductory chapter, additional "working questions" have also been stated. These have been implicitly answered in chapter 2 and are not part of the findings this thesis aims to present, which is why they are left out in this section.

1. How does the predictive performance of "Regionalized Random Forest" vary by different regionalization techniques?

This question was investigated in detail in subsection 4.3.1. The condensed findings presented in the tables are that there are significant differences regarding the performance of the predictive models, depending on which regionalization method is used in the workflow. However, the differences in performance depend on more than just the regionalization algorithm. Still, it is useful to take another look at them before going into more detail. WARD performed very good across both modelling approaches (CN and CN_W), which is likely due to the experiment being optimized using this method, which has been justified and explained in previous sections. Nonetheless, there are interesting patterns to be found for the remaining four methods as well. AZP is the worst performing method for both modelling approaches regarding the R^2 values, but it produces an acceptable number of insignificant LISA clusters in both cases. The next method, SKATER, delivers good results regarding both the R^2 values and the LISA clusters. A balance between both of these metrics might be desired in some situations, for which SKATER would be a very good fit. The final method which requires the user to input the number of regions is KMEANS. Its performance is underwhelming regarding the R^2 values, the number of insignificant LISA clusters shows good performance, even showing the best results for CN_W. Nonetheless, as the R^2 lack that far behind, it is suspected that KMEANS will not be the method of choice for most situations. This is also underlined by its relatively long computation time. The final regionalization algorithm that has been tested is MAXP, where the number of regions is determined by the algorithm itself. This has produced good performance metrics, especially for CN_W, and the R^2 values are high for both approaches. For CN_W, it is justifiable to consider MAXP as the best performing algorithm.

Generally speaking, there are a few interesting takeaways that can be obtained by deducting an experiment like this and looking at the table containing the resulting performance metrics from the appendix (link). The main takeaway of this is that there are other factors than just the regionalization algorithms that influence the performance metrics. One of these is of course the modelling approach - CN or

4. Results and Discussion

CN_W. Another big factor is the number of regions the dataset is split into, the influence of which varies depending on the variation of RegRF in use. In this work, this was only possible to be explored for a small dataset with the methods WARD and AZP, but it is certain that more insights can be gained by extending the evaluation to the other three methods.

Another distinguishing factor between the respective regionalization algorithms is the computational effort. There are big differences regarding this aspect, which also greatly influenced the findings of this work. The very long computation time for the full dataset had the biggest influence, as this led to three of the five regionalization being left out of the analysis for the full dataset. Even for the small dataset, there were big differences in this regard. Especially MAXP took significantly longer with more than 40 minutes, the fastest one was WARD with 13 seconds. The other three take between 130 and 170 seconds.

2. How does "Regionalized Random Forest" perform compared to benchmark methods, either these being Spatial Statistical methods or Machine Learning algorithms?

One of the most important aspects of such research is how the proposed approach performs in comparison to benchmark methods. For this work, Geographically Weighted Regression (GWR - section 2.4.1) and Random Forest (RF - section 2.5.3) have been chosen. In comparison to the spatial statistical method of GWR, RegRF performs worse by a close margin when using the modelling approach CN. The R^2 value is only 0.004 lower than for GWR, a difference that will not be noticed in the vast majority of situations. Given this, the significantly lower computation time of RegRF gives the proposed approach relevance and should be of interest to researchers and users in the field of spatial prediction. For the bigger dataset, the difference in performance is bigger, but this is fostered by the following: RegRF in its present configuration is optimized for the small dataset, while GWR has mechanics for performance optimization that are used whenever it is run. So, GWR is always more optimized towards the current situation, which can not be said for RegRF. To combat this, an optimization extension needs to be implemented. Only after this it would be possible to reliably conclude that RegRF performs better or worse than GWR. One definitive advantage of RegRF in comparison to GWR is the computation time, especially for large scale data. For the full dataset, GWR takes more than ten times longer to deliver results. For the smaller dataset, this difference is not that pronounced, but as mentioned before the performance also does not vary by as much as for the bigger dataset.

The second benchmark method, the famous Machine Learning algorithm Random Forest, was outperformed in every scenario. This is visible from the tables given in the previous chapter, and the big table in the appendix underscores this even more. From there, it is evident that RegRF can significantly boost the performance metrics, no matter in which configuration. Regardless of the modelling approach and the number of regions in use, RegRF outperforms RF by a large margin. As this means that better performing models can be created without adding a lot of

4.4. Answering the Research Questions

computation time on one hand, but also without the need of spatial knowledge in order to find a fitting number of regions on the other hand, this can be seen as a very important finding of this work.

3. How does "Regionalized Random Forest" perform compared to existing spatially conscious ML approaches such as the GW-RF?

RegRF was compared to one additional method, which is the recently proposed approach Geographically Weighted Random Forest (Georganos and Kalogirou 2022). The question at hand can only be answered for the small dataset, as GW-RF was not able to finish computation in a reasonable amount of time for the big dataset, which naturally is not a desirable trait for researchers and users of the algorithm. Additionally, it is only available in R, which makes it harder to implement given the rapid rise of Python in both the spatial and non-spatial world of data analysis over the course of the last years. Three attempts were made to run it using the big dataset, and even when deleting the part of code where the bandwidth is estimated, the computation was not able to finish in seven days, despite the computer being a powerful machine learning station. So, for the big dataset, RegRF obviously has a significant advantage.

In contrast to this stand the results of the application of GW-RF with the small dataset. Here, it outperformed all the other methods regarding the R^2 values by quite a big margin. The computational effort needed was also the highest, and keeping the powerful computer in use as well as the problems with the bigger dataset in mind, questions remain regarding the computation of this on standard computers. Nonetheless, in this setting, RegRF was not able to compete with GW-RF.

5. Conclusion

The concluding chapter aims to examine the most important findings of this thesis. After this, future perspectives for research on the topic will be discussed.

Previous sections have delineated the results of the experiment in a very detailed manner. From this, the most important conclusion that can be drawn is the vast improvement achieved by the use of RegRF over traditional Machine Learning approaches in the form of Random Forest. Even without knowledge about the area at hand or expertise on how to correctly specify the parameters in the predictive models, significant improvements in prediction accuracy can be made with practically no increase in computation time, depending on the regionalization algorithm in use. RegRF was even able to compete with the established spatial statistical method Geographically Weighted Regression in certain scenarios. The third approach RegRF was compared against, Geographically Weighted Random Forest, was not able to be matched, but it should be mentioned that the practical application of this algorithm was not possible for the large scale dataset that was used in the practical part of this work, for which RegRF was also able to compute in a very convenient amount of time. When comparing the proposed approach to the three mentioned methods, one very important factor should be kept in mind: There are no automatic optimization mechanisms in the regionalization part of RegRF. In contrast to this, both the other spatial methods do have a mechanism of this sort by determining the best bandwidth for each model run. Additionally, as RegRF is heavily based on a "casual" Random Forest, it is possible to use it for both classification and regression tasks, which can not be said for GWR and GW-RF. Also, GWR can only deal with a linear relationship between the target and the input features, which is not a problem for RegRF and GW-RF. Due to this, for certain scenarios it can be argued that RegRF is the best choice, which ultimately depends on factors like the size of the dataset, the (non-)linearity between dependent and independent variables, whether it is a supervised task or not, as well as the levels of measurement of the input features.

This leads to future perspectives that should be addressed. As mentioned, a mechanism for optimizing the number of regions as well as other parameters could further boost the performance of Regionalized Random Forest. In addition to the number of regions, this could include the most fitting regionalization algorithms as well as the minimum number of objects per region. It is possible that this would significantly increase the computational effort necessary, which so far can be seen as an advantage over GWR and GW-RF, but to arrive at a reliable conclusion regarding this, a practical experiment would need to be carried out. Another aspect that should be addressed by future research ambitions is the implementation of additional regionalization algorithms that are able to deal with large-scale datasets. In this thesis, only two of the five evaluated methods were able to produce results for the big datasets. All of them were able to run for a smaller

5. Conclusion

dataset, but even there some of them took considerable amounts of time. Given the trend in the last years as the processing and analysis of large scale datasets has become more and more important, this is an opportunity researchers in the field of spatial analytics and prediction should not pass up. The proposed approach has shown its potential which could be boosted further by more modern regionalization algorithms. This is also underscored by the fact that the newest algorithm under evaluation here stems from the year 2012. In the light of the current advancements in the fields of artificial intelligence, data processing and applied informatics, this seems to lack behind in actuality. This also leads to the final issue that needs to be discussed, which is the importance of including spatial aspects in Machine Learning wherever possible. "Casual" Machine Learning algorithms have become very popular over the last few years, but the spatial component still seems to be absent from the minds of many users of these methods. The experiment carried out in this work has shown the vast improvements that can be achieved by including spatial aspects, even without extensive knowledge about the area of interest of the spatial methods in use.

Bibliography

- Anselin, Luc (1st Apr. 1989). “What is special about spatial data? Alternative perspectives on spatial data analysis”. In: *Alternative Perspectives on Spatial Data Analysis* 89.4.
- (1995). “Local Indicators of Spatial Association—LISA”. In: *Geographical Analysis* 27.2, pp. 93–115. ISSN: 1538-4632. DOI: [10.1111/j.1538-4632.1995.tb00338.x](https://doi.org/10.1111/j.1538-4632.1995.tb00338.x). URL: <http://onlinelibrary.wiley.com/doi/abs/10.1111/j.1538-4632.1995.tb00338.x> (visited on 04/11/2022).
- (2020). *GeoDa - Documentation*. GeoDa Workbook. URL: <https://geodacenter.github.io/documentation.html> (visited on 11/02/2023).
- Arbelaitz, Olatz, Ibai Gurrutxaga, Javier Muguerza, Jesús M. Pérez and Iñigo Perona (Jan. 2013). “An extensive comparative study of cluster validity indices”. In: *Pattern Recognition* 46.1, pp. 243–256. ISSN: 0031-3203. DOI: [10.1016/j.patcog.2012.07.021](https://doi.org/10.1016/j.patcog.2012.07.021). URL: <http://dx.doi.org/10.1016/j.patcog.2012.07.021>.
- Arthur, David and Sergei Vassilvitskii (2007). “K-Means++: The Advantages of Careful Seeding”. In.
- Assunção, R. M., M. C. Neves, G. Câmara and C. Da Costa Freitas (Aug. 2006). “Efficient Regionalization Techniques for Socio-economic Geographical Units Using Minimum Spanning Trees”. In: *International Journal of Geographical Information Science* 20.7, pp. 797–811. ISSN: 1365-8816, 1362-3087. DOI: [10.1080/13658810600665111](https://doi.org/10.1080/13658810600665111). URL: <http://www.tandfonline.com/doi/abs/10.1080/13658810600665111> (visited on 09/01/2023).
- Aydin, Orhun, Mark. V. Janikas, Renato Martins Assunção and Ting-Hwan Lee (2nd Nov. 2021). “A quantitative comparison of regionalization methods”. In: *International Journal of Geographical Information Science* 35.11, pp. 2287–2315. ISSN: 1365-8816, 1362-3087. DOI: [10.1080/13658816.2021.1905819](https://doi.org/10.1080/13658816.2021.1905819). URL: <https://www.tandfonline.com/doi/full/10.1080/13658816.2021.1905819> (visited on 13/11/2023).
- Bivand, Roger, Edzer J. Pebesma and Virgilio Gómez-Rubio (2008). *Applied Spatial Data Analysis with R*. Use R! New York: Springer. 374 pp. ISBN: 978-0-387-78170-9 978-0-387-78171-6.
- Bluman, Allan G. (2012). *Elementary statistics: a step by step approach*. 8th ed. New York: McGraw-Hill. 816 pp. ISBN: 978-0-07-338610-2 978-0-07-756437-7 978-0-07-743858-6.
- Box, G.E.P. (1979). “Robustness in the Strategy of Scientific Model Building”. In: *Robustness in Statistics*. Elsevier, pp. 201–236. ISBN: 978-0-12-438150-6. DOI: [10.1016/B978-0-12-438150-6.50018-2](https://doi.org/10.1016/B978-0-12-438150-6.50018-2). URL: <https://linkinghub.elsevier.com/retrieve/pii/B9780124381506500182> (visited on 11/01/2023).
- Breiman, Leo (1st Oct. 2001). “Random Forests”. In: *Machine Learning* 45.1, pp. 5–32. ISSN: 1573-0565. DOI: [10.1023/A:1010933404324](https://doi.org/10.1023/A:1010933404324). URL: <https://doi.org/10.1023/A:1010933404324> (visited on 06/11/2022).

Bibliography

- Brunsdon, Chris, A. Stewart Fotheringham and Martin E. Charlton (1996). “Geographically Weighted Regression: A Method for Exploring Spatial Nonstationarity”. In: *Geographical Analysis* 28.4, pp. 281–298. ISSN: 00167363. DOI: [10.1111/j.1538-4632.1996.tb00936.x](https://doi.org/10.1111/j.1538-4632.1996.tb00936.x). URL: <https://onlinelibrary.wiley.com/doi/10.1111/j.1538-4632.1996.tb00936.x> (visited on 08/11/2022).
- Casetti, Emilio (1972). “Generating Models by the Expansion Method: Applications to Geographical Research*”. In: *Geographical Analysis* 4.1, pp. 81–91. ISSN: 00167363. DOI: [10.1111/j.1538-4632.1972.tb00458.x](https://doi.org/10.1111/j.1538-4632.1972.tb00458.x). URL: <https://onlinelibrary.wiley.com/doi/10.1111/j.1538-4632.1972.tb00458.x> (visited on 01/12/2022).
- Chicco, Davide, Matthijs J. Warrens and Giuseppe Jurman (5th July 2021). “The Coefficient of Determination R-squared Is More Informative than SMAPE, MAE, MAPE, MSE and RMSE in Regression Analysis Evaluation”. In: *PeerJ Computer Science* 7, e623. ISSN: 2376-5992. DOI: [10.7717/peerj-cs.623](https://doi.org/10.7717/peerj-cs.623). URL: <https://peerj.com/articles/cs-623> (visited on 06/11/2022).
- Cliff, A. D. and J. K. Ord (1969). “The Problem of Spatial Autocorrelation”. In: *Studies in Regional Science*, pp. 25–55.
- Currell, Graham and Antony Dowman (2009). *Essential mathematics and statistics for science*. 2nd ed. OCLC: ocn278980847. Chichester, West Sussex ; Hoboken, NJ: Wiley-Blackwell. 387 pp. ISBN: 978-0-470-69449-7 978-0-470-69448-0.
- Curry, M (1995). “On Space and Spatial Practice in Contemporary Geography”. In: *Concepts in Human Geography*, pp. 3–32.
- Curry, Michael R. (1998). *Digital Places: Living with Geographic Information Technologies*. London ; New York: Routledge. 191 pp. ISBN: 978-0-415-13014-1 978-0-415-13015-8.
- Dall’erba, S. (2009). “Exploratory Spatial Data Analysis”. In: *International Encyclopedia of Human Geography*. Elsevier, pp. 683–690. ISBN: 978-0-08-044910-4. DOI: [10.1016/B978-008044910-4.00433-8](https://doi.org/10.1016/B978-008044910-4.00433-8). URL: <https://linkinghub.elsevier.com/retrieve/pii/B9780080449104004338> (visited on 25/01/2023).
- Davies, David L. and Donald W. Bouldin (Apr. 1979). “A Cluster Separation Measure”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* PAMI-1.2, pp. 224–227. ISSN: 2160-9292. DOI: [10.1109/tpami.1979.4766909](https://doi.org/10.1109/tpami.1979.4766909). URL: <http://dx.doi.org/10.1109/TPAMI.1979.4766909>.
- Dodge, Martin and Rob Kitchin (2001). *Mapping Cyberspace*. London ; New York: Routledge. 260 pp. ISBN: 978-0-415-19883-7 978-0-415-19884-4.
- Duque, Juan C., Luc Anselin and Sergio J. Rey (Aug. 2012). “THE MAX-P-REGIONS PROBLEM*”. In: *Journal of Regional Science* 52.3, pp. 397–419. ISSN: 00224146. DOI: [10.1111/j.1467-9787.2011.00743.x](https://doi.org/10.1111/j.1467-9787.2011.00743.x). URL: <https://onlinelibrary.wiley.com/doi/10.1111/j.1467-9787.2011.00743.x> (visited on 30/01/2023).
- Duque, Juan Carlos, Raúl Ramos and Jordi Suriñach (July 2007). “Supervised Regionalization Methods: A Survey”. In: *International Regional Science Review* 30.3, pp. 195–220. ISSN: 0160-0176, 1552-6925. DOI: [10.1177/0160017607301605](https://doi.org/10.1177/0160017607301605). URL: <http://journals.sagepub.com/doi/10.1177/0160017607301605> (visited on 21/09/2023).
- Feng, Xin, Germano Barcelos, James D. Gaboardi, Elijah Knaap, Ran Wei, Levi J. Wolf, Qunshan Zhao and Sergio J. Rey (2022). “spopt: a python package for solving spatial

- optimization problems in PySAL". In: *Journal of Open Source Software* 7.74, p. 3330. DOI: [10.21105/joss.03330](https://doi.org/10.21105/joss.03330). URL: <https://doi.org/10.21105/joss.03330>.
- Fisher, Ronald (1935). *The Design of Experiments*. Edinburgh: Oliver and Boyd.
- Fotheringham, A S and D W S Wong (July 1991). "The Modifiable Areal Unit Problem in Multivariate Statistical Analysis". In: *Environment and Planning A: Economy and Space* 23.7, pp. 1025–1044. ISSN: 0308-518X, 1472-3409. DOI: [10.1068/a231025](https://doi.org/10.1068/a231025). URL: <http://journals.sagepub.com/doi/10.1068/a231025> (visited on 02/02/2023).
- Fotheringham, A. Stewart (Oct. 2009). "The Problem of Spatial Autocorrelation" and Local Spatial Statistics: Spatial Autocorrelation and Local Spatial Statistics". In: *Geographical Analysis* 41.4, pp. 398–403. ISSN: 00167363. DOI: [10.1111/j.1538-4632.2009.00767.x](https://doi.org/10.1111/j.1538-4632.2009.00767.x). URL: <https://onlinelibrary.wiley.com/doi/10.1111/j.1538-4632.2009.00767.x> (visited on 04/01/2023).
- Fotheringham, A. Stewart, Chris Brunsdon and Martin Charlton (2002). *Geographically Weighted Regression: The Analysis of Spatially Varying Relationships*. Chichester, England ; Hoboken, NJ, USA: Wiley. 269 pp. ISBN: 978-0-471-49616-8.
- Geary, R. C. (Nov. 1954). "The Contiguity Ratio and Statistical Mapping". In: *The Incorporated Statistician* 5.3, p. 115. ISSN: 14669404. DOI: [10.2307/2986645](https://doi.org/10.2307/2986645). JSTOR: [2986645](https://www.jstor.org/stable/2986645).
- Georganos, Stefanos, Tais Grippa, Assane Gadiaga, Sabine Vanhuysse, Stamatis Kalogirou, Moritz Lennert and Catherine Linard (May 2019). "An Application of Geographical Random Forests for Population Estimation in Dakar, Senegal Using Very-High-Resolution Satellite Imagery". In: *2019 Joint Urban Remote Sensing Event (JURSE)*. 2019 Joint Urban Remote Sensing Event (JURSE), pp. 1–4. DOI: [10.1109/JURSE.2019.8809049](https://doi.org/10.1109/JURSE.2019.8809049).
- Georganos, Stefanos, Tais Grippa, Assane Niang Gadiaga, Catherine Linard, Moritz Lennert, Sabine Vanhuysse, Nicholas Mboga, Eléonore Wolff and Stamatis Kalogirou (20th Jan. 2021). "Geographical Random Forests: A Spatial Extension of the Random Forest Algorithm to Address Spatial Heterogeneity in Remote Sensing and Population Modelling". In: *Geocarto International* 36.2, pp. 121–136. ISSN: 1010-6049, 1752-0762. DOI: [10.1080/10106049.2019.1595177](https://doi.org/10.1080/10106049.2019.1595177). URL: <https://www.tandfonline.com/doi/full/10.1080/10106049.2019.1595177> (visited on 04/11/2022).
- Georganos, Stefanos and Stamatis Kalogirou (Sept. 2022). "A Forest of Forests: A Spatially Weighted and Computationally Efficient Formulation of Geographical Random Forests". In: *ISPRS International Journal of Geo-Information* 11.9 (9), p. 471. ISSN: 2220-9964. DOI: [10.3390/ijgi11090471](https://doi.org/10.3390/ijgi11090471). URL: <https://www.mdpi.com/2220-9964/11/9/471> (visited on 04/11/2022).
- Getis, Arthur (2010). "Spatial Autocorrelation". In: *Handbook of Applied Spatial Analysis*. Ed. by Manfred M. Fischer and Arthur Getis. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 255–278. ISBN: 978-3-642-03646-0 978-3-642-03647-7. DOI: [10.1007/978-3-642-03647-7_14](https://doi.org/10.1007/978-3-642-03647-7_14). URL: http://link.springer.com/10.1007/978-3-642-03647-7_14 (visited on 04/11/2022).
- Getis, Arthur and Jared Aldstadt (July 2004). "Constructing the Spatial Weights Matrix Using a Local Statistic". In: *Geographical Analysis* 36.2, pp. 90–104. ISSN: 00167363.

Bibliography

- DOI: [10.1111/j.1538-4632.2004.tb01127.x](https://doi.org/10.1111/j.1538-4632.2004.tb01127.x). URL: <https://onlinelibrary.wiley.com/doi/10.1111/j.1538-4632.2004.tb01127.x> (visited on 30/12/2022).
- Getis, Arthur and J. K. Ord (1992). “The Analysis of Spatial Association by Use of Distance Statistics”. In: *Geographical Analysis* 24.3, pp. 189–206. ISSN: 00167363. DOI: [10.1111/j.1538-4632.1992.tb00261.x](https://doi.org/10.1111/j.1538-4632.1992.tb00261.x). URL: <https://onlinelibrary.wiley.com/doi/10.1111/j.1538-4632.1992.tb00261.x> (visited on 04/11/2022).
- Gleeson, B. J. (1996). “A Geography for Disabled People?” In: *Transactions of the Institute of British Geographers* 21.2, p. 387. ISSN: 00202754. DOI: [10.2307/622488](https://doi.org/10.2307/622488). JSTOR: 622488.
- Goodchild, Michael F. (June 2004). “The Validity and Usefulness of Laws in Geographic Information Science and Geography”. In: *Annals of the Association of American Geographers* 94.2, pp. 300–303. ISSN: 0004-5608, 1467-8306. DOI: [10.1111/j.1467-8306.2004.09402008.x](https://doi.org/10.1111/j.1467-8306.2004.09402008.x). URL: <http://www.tandfonline.com/doi/abs/10.1111/j.1467-8306.2004.09402008.x> (visited on 16/01/2023).
- (2008). “Data Analysis, Spatial”. In: *Encyclopedia of GIS*. Ed. by Shashi Shekhar and Hui Xiong. Boston, MA: Springer US, pp. 200–203. ISBN: 978-0-387-30858-6 978-0-387-35973-1. DOI: [10.1007/978-0-387-35973-1_236](https://doi.org/10.1007/978-0-387-35973-1_236). URL: http://link.springer.com/10.1007/978-0-387-35973-1_236 (visited on 24/01/2023).
- Grekousis, George (31st May 2020). *Spatial Analysis Methods and Practice: Describe – Explore – Explain through GIS*. 1st ed. Cambridge University Press. ISBN: 978-1-108-61452-8 978-1-108-49898-2 978-1-108-71293-4. DOI: [10.1017/9781108614528](https://doi.org/10.1017/9781108614528). URL: <https://www.cambridge.org/core/product/identifier/9781108614528/type/book> (visited on 24/01/2023).
- Griffith, Daniel A. (May 2012). “Spatial Statistics: A Quantitative Geographer’s Perspective”. In: *Spatial Statistics* 1, pp. 3–15. ISSN: 22116753. DOI: [10.1016/j.spasta.2012.03.005](https://doi.org/10.1016/j.spasta.2012.03.005). URL: <https://linkinghub.elsevier.com/retrieve/pii/S2211675312000127> (visited on 04/01/2023).
- Guan, Qingfeng, Liming Wang and Keith C. Clarke (Jan. 2005). “An Artificial-Neural-Network-based, Constrained CA Model for Simulating Urban Growth”. In: *Cartography and Geographic Information Science* 32.4, pp. 369–380. ISSN: 1523-0406, 1545-0465. DOI: [10.1559/152304005775194746](https://doi.org/10.1559/152304005775194746). URL: <http://www.tandfonline.com/doi/abs/10.1559/152304005775194746> (visited on 10/02/2023).
- Haining, Robert P. (2010). “The Nature of Georeferenced Data”. In: *Handbook of Applied Spatial Analysis*. Ed. by Manfred M. Fischer and Arthur Getis. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 197–217. ISBN: 978-3-642-03646-0 978-3-642-03647-7. DOI: [10.1007/978-3-642-03647-7_12](https://doi.org/10.1007/978-3-642-03647-7_12). URL: http://link.springer.com/10.1007/978-3-642-03647-7_12 (visited on 04/11/2022).
- Harris, Charles R. et al. (Sept. 2020). “Array programming with NumPy”. In: *Nature* 585.7825, pp. 357–362. DOI: [10.1038/s41586-020-2649-2](https://doi.org/10.1038/s41586-020-2649-2). URL: <https://doi.org/10.1038/s41586-020-2649-2>.
- Hawking, Stephen (1998). *A Brief History of Time*. Updated and expanded tenth anniversary ed. New York: Bantam Books. 212 pp. ISBN: 978-0-553-10953-5 978-0-553-38016-3.

- Hengl, Tomislav, Madlene Nussbaum, Marvin N. Wright, Gerard B.M. Heuvelink and Benedikt Gräler (29th Aug. 2018). “Random Forest as a Generic Framework for Predictive Modeling of Spatial and Spatio-Temporal Variables”. In: *PeerJ* 6, e5518. ISSN: 2167-8359. DOI: [10.7717/peerj.5518](https://doi.org/10.7717/peerj.5518). URL: <https://peerj.com/articles/5518> (visited on 06/11/2022).
- Ho, Don (2023). *Notepad++*. URL: <https://notepad-plus-plus.org/> (visited on 16/02/2023).
- Hunter, J. D. (2007). “Matplotlib: A 2D graphics environment”. In: *Computing in Science & Engineering* 9.3, pp. 90–95. DOI: [10.1109/MCSE.2007.55](https://doi.org/10.1109/MCSE.2007.55). URL: <https://matplotlib.org/stable/users/project/index.html>.
- Jordahl, Kelsey et al. (July 2020). *geopandas/geopandas: v0.8.1*. Version v0.8.1. DOI: [10.5281/zenodo.3946761](https://doi.org/10.5281/zenodo.3946761). URL: <https://doi.org/10.5281/zenodo.3946761> (visited on 10/12/2023).
- Joshi, Prateek (2017). *Artificial Intelligence with Python: Build Real-World Artificial Intelligence Applications with Python to Intelligently Interact with the World around You*. Birmingham Mumbai: Packt. 430 pp. ISBN: 978-1-78646-439-2.
- Kelejian, Harry and Gianfranco Piras (2017). “Spatial Models: Basic Issues”. In: *Spatial Econometrics*. Elsevier, pp. 1–10. ISBN: 978-0-12-813387-3. DOI: [10.1016/B978-0-12-813387-3.00001-9](https://doi.org/10.1016/B978-0-12-813387-3.00001-9). URL: <https://linkinghub.elsevier.com/retrieve/pii/B9780128133873000019> (visited on 27/01/2023).
- Klemmer, Konstantin, Adriano Koshiyama and Sebastian Flennerhag (23rd May 2019). *Augmenting Correlation Structures in Spatial Data Using Deep Generative Models*. arXiv: [1905.09796](https://arxiv.org/abs/1905.09796) [cs, stat]. URL: <http://arxiv.org/abs/1905.09796> (visited on 15/02/2023).
- Kuhn, Max and Kjell Johnson (2013). *Applied Predictive Modeling*. New York, NY: Springer New York. ISBN: 978-1-4614-6848-6 978-1-4614-6849-3. DOI: [10.1007/978-1-4614-6849-3](https://doi.org/10.1007/978-1-4614-6849-3). URL: <http://link.springer.com/10.1007/978-1-4614-6849-3> (visited on 14/02/2023).
- Lankford, Philip M. (1969). “Regionalization: Theory and Alternative Algorithms”. In: *Geographical Analysis* 1.2, pp. 196–212. ISSN: 1538-4632. DOI: [10.1111/j.1538-4632.1969.tb00615.x](https://doi.org/10.1111/j.1538-4632.1969.tb00615.x). URL: <https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1538-4632.1969.tb00615.x> (visited on 09/01/2023).
- Liaw, Andy and Matthew Wiener (2002). “Classification and Regression by randomForest”. In: 2, p. 5.
- Liu, Xiaojian, Ourania Kounadi and Raul Zurita-Milla (Apr. 2022). “Incorporating Spatial Autocorrelation in Machine Learning Models Using Spatial Lag and Eigenvector Spatial Filtering Features”. In: *ISPRS International Journal of Geo-Information* 11.4 (4), p. 242. ISSN: 2220-9964. DOI: [10.3390/ijgi11040242](https://doi.org/10.3390/ijgi11040242). URL: <https://www.mdpi.com/2220-9964/11/4/242> (visited on 04/11/2022).
- Lloyd, S. (Mar. 1982). “Least Squares Quantization in PCM”. In: *IEEE Transactions on Information Theory* 28.2, pp. 129–137. ISSN: 0018-9448. DOI: [10.1109/TIT.1982.1056489](https://doi.org/10.1109/TIT.1982.1056489). URL: <http://ieeexplore.ieee.org/document/1056489/> (visited on 19/10/2023).

Bibliography

- Lotfata, Aynaz, Stefanos Georganos, Stamatis Kalogirou and Marco Helbich (8th Nov. 2022). “Ecological Associations between Obesity Prevalence and Neighborhood Determinants Using Spatial Machine Learning in Chicago, Illinois, USA”. In: *ISPRS International Journal of Geo-Information* 11.11, p. 550. ISSN: 2220-9964. DOI: [10.3390/ijgi11110550](https://doi.org/10.3390/ijgi11110550). URL: <https://www.mdpi.com/2220-9964/11/11/550> (visited on 14/11/2022).
- Masjedi, Ali, Neal R. Carpenter, Melba M. Crawford and Mitch R. Tuinstra (June 2019). “Prediction of Sorghum Biomass Using Uav Time Series Data and Recurrent Neural Networks”. In: *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*. 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW). Long Beach, CA, USA: IEEE, pp. 2695–2702. ISBN: 978-1-72812-506-0. DOI: [10.1109/CVPRW.2019.00327](https://doi.org/10.1109/CVPRW.2019.00327). URL: <https://ieeexplore.ieee.org/document/9025597/> (visited on 10/02/2023).
- Massey, Doreen B. (1994). *Space, Place, and Gender*. Minneapolis: University of Minnesota Press. ISBN: 978-0-8166-8653-7.
- McKinney, Wes (2010). “Data Structures for Statistical Computing in Python”. In: *Proceedings of the 9th Python in Science Conference*. Ed. by Stéfan van der Walt and Jarrod Millman, pp. 51–56. URL: <https://pandas.pydata.org/about/>.
- Mignotte, Max (Jan. 2011). “A De-Texturing and Spatially Constrained K-means Approach for Image Segmentation”. In: *Pattern Recognition Letters* 32.2, pp. 359–367. ISSN: 01678655. DOI: [10.1016/j.patrec.2010.09.016](https://doi.org/10.1016/j.patrec.2010.09.016). URL: <https://linkinghub.elsevier.com/retrieve/pii/S016786551000320X> (visited on 31/01/2023).
- Miller, Harvey J. (June 2004). “Tobler’s First Law and Spatial Analysis”. In: *Annals of the Association of American Geographers* 94.2, pp. 284–289. ISSN: 0004-5608, 1467-8306. DOI: [10.1111/j.1467-8306.2004.09402005.x](https://doi.org/10.1111/j.1467-8306.2004.09402005.x). URL: <http://www.tandfonline.com/doi/abs/10.1111/j.1467-8306.2004.09402005.x> (visited on 18/01/2023).
- Moran, P A P (1950). “Notes on Continuous Stochastic Phenomena”. In: p. 8.
- Mueller, Erik, J. S. Onésimo Sandoval, Srikanth Mudigonda and Michael Elliott (28th Dec. 2018). “A Cluster-Based Machine Learning Ensemble Approach for Geospatial Data: Estimation of Health Insurance Status in Missouri”. In: *ISPRS International Journal of Geo-Information* 8.1, p. 13. ISSN: 2220-9964. DOI: [10.3390/ijgi8010013](https://doi.org/10.3390/ijgi8010013). URL: <http://www.mdpi.com/2220-9964/8/1/13> (visited on 13/12/2022).
- Neprash, Jerry A (1934). “Some Problems in the Correlation of Spatially Distributed Variables”. In.
- Nikparvar, Behnam and Jean-Claude Thill (12th Sept. 2021). “Machine Learning of Spatial Data”. In: *ISPRS International Journal of Geo-Information* 10.9, p. 600. ISSN: 2220-9964. DOI: [10.3390/ijgi10090600](https://doi.org/10.3390/ijgi10090600). URL: <https://www.mdpi.com/2220-9964/10/9/600> (visited on 01/02/2023).
- Nussbaum, Madlene, Kay Spiess, Andri Baltensweiler, Urs Grob, Armin Keller, Lucie Greiner, Michael E. Schaepman and Andreas Papritz (10th Jan. 2018). “Evaluation of Digital Soil Mapping Approaches with Large Sets of Environmental Covariates”. In: *SOIL* 4.1, pp. 1–22. ISSN: 2199-398X. DOI: [10.5194/soil-4-1-2018](https://doi.org/10.5194/soil-4-1-2018). URL: <https://soil.copernicus.org/articles/4/1/2018/> (visited on 14/02/2023).

- O’Sullivan, David and D. Unwin (2010). *Geographic Information Analysis*. 2nd ed. Hoboken, N.J: John Wiley & Sons. 405 pp. ISBN: 978-0-470-28857-3.
- Openshaw, S. (1977). “A Geographical Solution to Scale and Aggregation Problems in Region-Building, Partitioning and Spatial Modelling”. In: *Transactions of the Institute of British Geographers* 2.4, pp. 459–472. ISSN: 0020-2754. DOI: [10.2307/622300](https://doi.org/10.2307/622300). JSTOR: [622300](https://www.jstor.org/stable/622300).
- Oshan, Taylor, Ziqi Li, Wei Kang, Levi Wolf and A. Fotheringham (8th June 2019). “mgwr: A Python Implementation of Multiscale Geographically Weighted Regression for Investigating Process Spatial Heterogeneity and Scale”. In: *ISPRS International Journal of Geo-Information* 8.6, p. 269. ISSN: 2220-9964. DOI: [10.3390/ijgi8060269](https://doi.org/10.3390/ijgi8060269). URL: <https://www.mdpi.com/2220-9964/8/6/269> (visited on 26/02/2023).
- Pace, R. Kelley and Ronald Barry (May 1997). “Sparse Spatial Autoregressions”. In: *Statistics & Probability Letters* 33.3, pp. 291–297. ISSN: 01677152. DOI: [10.1016/S0167-7152\(96\)00140-X](https://doi.org/10.1016/S0167-7152(96)00140-X). URL: <https://linkinghub.elsevier.com/retrieve/pii/S016771529600140X> (visited on 15/02/2023).
- Pedregosa, F. et al. (2011). “Scikit-learn: Machine Learning in Python”. In: *Journal of Machine Learning Research* 12, pp. 2825–2830.
- Peña, J.M, J.A Lozano and P Larrañaga (Oct. 1999). “An Empirical Comparison of Four Initialization Methods for the K-Means Algorithm”. In: *Pattern Recognition Letters* 20.10, pp. 1027–1040. ISSN: 01678655. DOI: [10.1016/S0167-8655\(99\)00069-0](https://doi.org/10.1016/S0167-8655(99)00069-0). URL: <https://linkinghub.elsevier.com/retrieve/pii/S0167865599000690> (visited on 18/10/2023).
- Prasad, Anantha M., Louis R. Iverson and Andy Liaw (Mar. 2006). “Newer Classification and Regression Tree Techniques: Bagging and Random Forests for Ecological Prediction”. In: *Ecosystems* 9.2, pp. 181–199. ISSN: 1432-9840, 1435-0629. DOI: [10.1007/s10021-005-0054-1](https://doi.org/10.1007/s10021-005-0054-1). URL: <http://link.springer.com/10.1007/s10021-005-0054-1> (visited on 14/02/2023).
- Python Software Foundation, The (22nd Nov. 2022). *BeginnersGuide/Overview - Python Wiki*. URL: <https://wiki.python.org/moin/BeginnersGuide/Overview> (visited on 07/02/2023).
- QGIS Development Team (2023). *QGIS Geographic Information System*. QGIS Association. URL: <https://www.qgis.org> (visited on 10/12/2023).
- Quevedo, Renata Pacheco, Daniel Andrade Maciel, Tatiana Dias Tardelli Uehara, Matej Vojtek, Camilo Daleles Rennó, Biswajeet Pradhan, Jana Vojteková and Quoc Bao Pham (20th Oct. 2021). “Consideration of Spatial Heterogeneity in Landslide Susceptibility Mapping Using Geographical Random Forest Model”. In: *Geocarto International* 0.0, pp. 1–24. ISSN: 1010-6049. DOI: [10.1080/10106049.2021.1996637](https://doi.org/10.1080/10106049.2021.1996637). URL: <https://doi.org/10.1080/10106049.2021.1996637> (visited on 12/12/2022).
- Quiñones, Sarah, Goyal Aditya and Zia U. Ahmed (2021). “Geographically Weighted Machine Learning Model for Untangling Spatial Heterogeneity of Type 2 Diabetes Mellitus (T2D) Prevalence in the USA”. In: *Scientific Reports (Nature Publisher Group)* 11.1. DOI: [10.1038/s41598-021-85381-5](https://doi.org/10.1038/s41598-021-85381-5). URL: <https://www.proquest.com/docview/2505575206/abstract/59B5F609C8184FD1PQ/1> (visited on 12/12/2022).

Bibliography

- R Core Team (2023). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing. Vienna, Austria. URL: <https://www.R-project.org/> (visited on 10/12/2023).
- 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. 595 pp. ISBN: 978-1-78712-593-3.
- Resch, Bernd, Florian Usländer and Clemens Havas (4th July 2018). “Combining Machine-Learning Topic Models and Spatiotemporal Analysis of Social Media Data for Disaster Footprint and Damage Assessment”. In: *Cartography and Geographic Information Science* 45.4, pp. 362–376. ISSN: 1523-0406, 1545-0465. DOI: [10.1080/15230406.2017.1356242](https://doi.org/10.1080/15230406.2017.1356242). URL: <https://www.tandfonline.com/doi/full/10.1080/15230406.2017.1356242> (visited on 10/02/2023).
- Rey, Sergio J. and Luc Anselin (2007). “PySAL: A Python Library of Spatial Analytical Methods”. In: *The Review of Regional Studies* 37.1, pp. 5–27.
- (2023). *Python spatial analysis library*. URL: <https://pysal.org/pysal/> (visited on 09/12/2023).
- Rogerson, Peter (2021). *Spatial Statistical Methods for Geography*. Thousand Oaks, California: SAGE. 234 pp. ISBN: 978-1-5297-0745-8 978-1-5297-0744-1.
- Rousseeuw, Peter J. (1986). “Silhouettes: A Graphical Aid to the Interpretation and Validation of Cluster Analysis”. In: *Journal of Computational and Applied Mathematics* 20, pp. 53–65. ISSN: 03770427. DOI: [10.1016/0377-0427\(87\)90125-7](https://doi.org/10.1016/0377-0427(87)90125-7). URL: <https://linkinghub.elsevier.com/retrieve/pii/0377042787901257> (visited on 20/10/2023).
- Sammur, Claude and Geoffrey I. Webb (2011a). “Mean Absolute Error”. In: *Encyclopedia of Machine Learning*. Ed. by Claude Sammur and Geoffrey I. Webb. Boston, MA: Springer US, pp. 652–652. ISBN: 978-0-387-30768-8 978-0-387-30164-8. DOI: [10.1007/978-0-387-30164-8_525](https://doi.org/10.1007/978-0-387-30164-8_525). URL: http://link.springer.com/10.1007/978-0-387-30164-8_525 (visited on 03/02/2023).
- (2011b). “Mean Squared Error”. In: *Encyclopedia of Machine Learning*. Ed. by Claude Sammur and Geoffrey I. Webb. Boston, MA: Springer US, pp. 653–653. ISBN: 978-0-387-30768-8 978-0-387-30164-8. DOI: [10.1007/978-0-387-30164-8_528](https://doi.org/10.1007/978-0-387-30164-8_528). URL: http://link.springer.com/10.1007/978-0-387-30164-8_528 (visited on 03/02/2023).
- Sankey, Temuulen Tsagaan (2016). “Scale, Effects”. In: *Encyclopedia of GIS*. Ed. by Shashi Shekhar, Hui Xiong and Xun Zhou. Cham: Springer International Publishing, pp. 1–8. ISBN: 978-3-319-23519-6. DOI: [10.1007/978-3-319-23519-6_1161-2](https://doi.org/10.1007/978-3-319-23519-6_1161-2). URL: http://link.springer.com/10.1007/978-3-319-23519-6_1161-2 (visited on 02/02/2023).
- Santos, Fabián, Valerie Graw and Santiago Bonilla (23rd Dec. 2019). “A Geographically Weighted Random Forest Approach for Evaluate Forest Change Drivers in the Northern Ecuadorian Amazon”. In: *PLOS ONE* 14.12. Ed. by Zhenlong Li, e0226224. ISSN: 1932-6203. DOI: [10.1371/journal.pone.0226224](https://doi.org/10.1371/journal.pone.0226224). URL: <https://dx.plos.org/10.1371/journal.pone.0226224> (visited on 06/11/2022).

- Shapiro, S. S. and M. B. Wilk (Dec. 1965). “An Analysis of Variance Test for Normality (Complete Samples)”. In: *Biometrika* 52.3/4, p. 591. ISSN: 0006-3444. DOI: [10.2307/2333709](https://doi.org/10.2307/2333709). URL: <http://dx.doi.org/10.2307/2333709>.
- Shields, Rob (1997). “Spatial Stress and Resistance: Social Meanings of Spatialization”. In: *Space and Social Theory: Interpreting Modernity and Postmodernity*. Ed. by Georges Benko and Ulf Strohmayer. Blackwell, pp. 33–186.
- Stephan, Frederick F (1934). “Sampling Errors and Interpretations of Social Data Ordered in Time and Space”. In.
- Stojanova, Daniela, Michelangelo Ceci, Annalisa Appice, Donato Malerba and Sašo Džeroski (2011). “Global and Local Spatial Autocorrelation in Predictive Clustering Trees”. In: *Discovery Science*. Ed. by Tapio Elomaa, Jaakko Hollmén and Heikki Mannila. Vol. 6926. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 307–322. ISBN: 978-3-642-24476-6 978-3-642-24477-3. DOI: [10.1007/978-3-642-24477-3_25](https://doi.org/10.1007/978-3-642-24477-3_25). URL: http://link.springer.com/10.1007/978-3-642-24477-3_25 (visited on 06/02/2023).
- Student (Apr. 1914). “The Elimination of Spurious Correlation Due to Position in Time or Space”. In: *Biometrika* 10.1, p. 179. ISSN: 00063444. DOI: [10.2307/2331746](https://doi.org/10.2307/2331746). JSTOR: [2331746](https://www.jstor.org/stable/2331746).
- Subbalakshmi, Chatti, G. Rama Krishna, S. Krishna Mohan Rao and P. Venketeswa Rao (2015). “A Method to Find Optimum Number of Clusters Based on Fuzzy Silhouette on Dynamic Data Set”. In: *Procedia Computer Science* 46, pp. 346–353. ISSN: 18770509. DOI: [10.1016/j.procs.2015.02.030](https://doi.org/10.1016/j.procs.2015.02.030). URL: <https://linkinghub.elsevier.com/retrieve/pii/S1877050915000940> (visited on 20/10/2023).
- Talebi, Hassan, Luk J. M. Peeters, Alex Otto and Raimon Tolosana-Delgado (Jan. 2022). “A Truly Spatial Random Forests Algorithm for Geoscience Data Analysis and Modelling”. In: *Mathematical Geosciences* 54.1, pp. 1–22. ISSN: 1874-8961, 1874-8953. DOI: [10.1007/s11004-021-09946-w](https://doi.org/10.1007/s11004-021-09946-w). URL: <https://link.springer.com/10.1007/s11004-021-09946-w> (visited on 06/11/2022).
- Tobler, W. R. (June 1970). “A Computer Movie Simulating Urban Growth in the Detroit Region”. In: *Economic Geography* 46, p. 234. ISSN: 00130095. DOI: [10.2307/143141](https://doi.org/10.2307/143141). JSTOR: [143141](https://www.jstor.org/stable/143141).
- U.S. Census Bureau (Apr. 2022). *Glossary*. URL: https://www.census.gov/programs-surveys/geography/about/glossary.html#par_textimage_4 (visited on 10/12/2023).
- (26th Jan. 2023). *California - Census Bureau Profile*. URL: <https://data.census.gov/profile?g=0400000US06> (visited on 07/02/2023).
- Wang, Fahui (2014). “Quantitative Methods and Socio-Economic Applications in GIS”. In.
- Ward, Joe H. (Mar. 1963). “Hierarchical Grouping to Optimize an Objective Function”. In: *Journal of the American Statistical Association* 58.301, pp. 236–244. ISSN: 0162-1459, 1537-274X. DOI: [10.1080/01621459.1963.10500845](https://doi.org/10.1080/01621459.1963.10500845). URL: <http://www.tandfonline.com/doi/abs/10.1080/01621459.1963.10500845> (visited on 16/10/2023).
- Wei, Ran, Sergio Rey and Elijah Knaap (2nd Jan. 2021). “Efficient regionalization for spatially explicit neighborhood delineation”. In: *International Journal of Geographical*

Bibliography

- Information Science* 35.1, pp. 135–151. ISSN: 1365-8816, 1362-3087. DOI: [10.1080/13658816.2020.1759806](https://doi.org/10.1080/13658816.2020.1759806). URL: <https://www.tandfonline.com/doi/full/10.1080/13658816.2020.1759806> (visited on 07/12/2023).
- Wise, Stephen, Robert Haining and Jingsheng Ma (Apr. 2001). “Providing Spatial Statistical Data Analysis Functionality for the GIS User: The SAGE Project”. In: *International Journal of Geographical Information Science* 15.3, pp. 239–254. ISSN: 1365-8816, 1362-3087. DOI: [10.1080/13658810151072877](https://doi.org/10.1080/13658810151072877). URL: <http://www.tandfonline.com/doi/abs/10.1080/13658810151072877> (visited on 21/09/2023).
- Wise, Steve, Robert Haining and Jingsheng Ma (1997). “Regionalisation Tools for the Exploratory Spatial Analysis of Health Data”. In: *Recent Developments in Spatial Analysis*. Ed. by Manfred M. Fischer and Arthur Getis. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 83–100. ISBN: 978-3-642-08321-1 978-3-662-03499-6. DOI: [10.1007/978-3-662-03499-6_5](https://doi.org/10.1007/978-3-662-03499-6_5). URL: http://link.springer.com/10.1007/978-3-662-03499-6_5 (visited on 21/09/2023).
- Wright, Sewall (1921). “Correlation and Causation”. In: *Journal of agricultural research* 20.7, pp. 557–585.
- Wu, Sensen, Zhongyi Wang, Zhenhong Du, Bo Huang, Feng Zhang and Renyi Liu (4th Mar. 2021). “Geographically and Temporally Neural Network Weighted Regression for Modeling Spatiotemporal Non-Stationary Relationships”. In: *International Journal of Geographical Information Science* 35.3, pp. 582–608. ISSN: 1365-8816, 1362-3087. DOI: [10.1080/13658816.2020.1775836](https://doi.org/10.1080/13658816.2020.1775836). URL: <https://www.tandfonline.com/doi/full/10.1080/13658816.2020.1775836> (visited on 10/02/2023).
- Yan, Jun and Jean-Claude Thill (June 2009). “Visual Data Mining in Spatial Interaction Analysis with Self-Organizing Maps”. In: *Environment and Planning B: Planning and Design* 36.3, pp. 466–486. ISSN: 0265-8135, 1472-3417. DOI: [10.1068/b34019](https://doi.org/10.1068/b34019). URL: <http://journals.sagepub.com/doi/10.1068/b34019> (visited on 10/02/2023).
- Young, Derek S. (2017). *Handbook of Regression Methods*. Boca Raton: CRC Press. ISBN: 978-1-4987-7529-8.
- Zhang, Ce, Isabel Sargent, Xin Pan, Huapeng Li, Andy Gardiner, Jonathon Hare and Peter M. Atkinson (Feb. 2019). “Joint Deep Learning for Land Cover and Land Use Classification”. In: *Remote Sensing of Environment* 221, pp. 173–187. ISSN: 00344257. DOI: [10.1016/j.rse.2018.11.014](https://doi.org/10.1016/j.rse.2018.11.014). URL: <https://linkinghub.elsevier.com/retrieve/pii/S0034425718305236> (visited on 10/02/2023).
- Zhao, Ruibin, Mingyong Pang and Jidong Wang (4th May 2018). “Classifying Airborne LiDAR Point Clouds via Deep Features Learned by a Multi-Scale Convolutional Neural Network”. In: *International Journal of Geographical Information Science* 32.5, pp. 960–979. ISSN: 1365-8816, 1362-3087. DOI: [10.1080/13658816.2018.1431840](https://doi.org/10.1080/13658816.2018.1431840). URL: <https://www.tandfonline.com/doi/full/10.1080/13658816.2018.1431840> (visited on 10/02/2023).

A. Appendix

A.1. Code

The following sections show the code used to implement the practical part of this work. This is also available on GitHub: <https://github.com/lukasboegl/Regionalized-Random-Forest>. For a more complete version, please visit the GitHub page, as some paths have been omitted in the printed versions to allow for better readability and formatting.

A.1.1. Random Forest

```
# Imports
import pandas as pd
import numpy as np
import geopandas as gpd
import libpysal as ps
from sklearn.ensemble import RandomForestRegressor
from sklearn.model_selection import RandomizedSearchCV
from sklearn.metrics import mean_absolute_error, r2_score, mean_squared_error
import math
import matplotlib.pyplot as plt
from time import time
import esda
import contextily as ctx

# Load and prepare data
# BIG DATASET
# Split 1
houstrain = gpd.read_file(r"pth")
houstest = gpd.read_file(r"pth")

# Split 2
houstrain = gpd.read_file(r"pth")
houstest = gpd.read_file(r"pth")

# Split 3
houstrain = gpd.read_file(r"pth")
houstest = gpd.read_file(r"pth")
```

A. Appendix

```
# SMALL DATASET
# Split 1
housestrain = gpd.read_file(r"pth")
housetest = gpd.read_file(r"pth")

# Split 2
#housestrain = gpd.read_file(r"pth")
#housetest = gpd.read_file(r"pth")

# Split 3
#housestrain = gpd.read_file(r"pth")
#housetest = gpd.read_file(r"pth")

print(housestrain.shape)
print(housetest.shape)
housestrain.head(2)

housetest.head(2)

# Define dependent and independent variables
# X = independent; y = dependent
X_train = housestrain.drop(["MedHouseVa", "geometry"], axis=1)
y_train = housestrain['MedHouseVa']

X_test = housetest.drop(["MedHouseVa", "geometry"], axis=1)
y_test = housetest['MedHouseVa']

print('Training Features Shape:', X_train.shape)
print('Training Labels Shape:', y_train.shape)
print('Testing Features Shape:', X_test.shape)
print('Testing Labels Shape:', y_test.shape)

# "Raw" Random Forest Regression - to show influence HPO has
rf = RandomForestRegressor(random_state=42, n_jobs=-1)
rf.fit(X_train, y_train)
predictions = rf.predict(X_test)

errors = abs(predictions - y_test)

print('MAE:', round(np.mean(errors), 2), 'units.')
r2 = round(r2_score(y_test, predictions), 2)
print("R2: ", r2)
mse = mean_squared_error(y_test, predictions)
```

```

print("MSE: ", mse)
print("RMSE: ", math.sqrt(mse))

# HPO - random search
param_dist = {
    'n_estimators': np.arange(50, 210, 10),
    'max_features': ['sqrt', 'log2'],
    'max_depth': [None] + list(np.arange(10, 110, 10)),
    'min_samples_split': np.arange(2, 12),
    'min_samples_leaf': np.arange(1, 5),
    'bootstrap': [True, False]
}

random_search = RandomizedSearchCV(
    rf, param_distributions=param_dist,
    scoring='neg_mean_squared_error',
    cv=5, n_iter=20, n_jobs=-1, random_state=42)
random_search.fit(X_train, y_train)
print("Best Hyperparameters:", random_search.best_params_)

# Build RF with best params and predict on test data
best_rf = RandomForestRegressor(**random_search.best_params_,
                                random_state=42, n_jobs=-1)
best_rf.fit(X_train, y_train)
predictions = best_rf.predict(X_test)

# Calculate Evaluation Metrics
print('MAE:', round(np.mean(abs(predictions - y_test)), 2), 'units.')
print("R2: ", round(r2_score(y_test, predictions), 3))
print("MSE: ", mean_squared_error(y_test, predictions))
print("RMSE: ", np.sqrt(mean_squared_error(y_test, predictions)))
print("Time for RF + pred: {:.2f} seconds".format(time() - start_time))

# LISA
#Calculation of TEST ERROR
housetest['predictions'] = predictions.flatten()
housetest['y_test'] = y_test
housetest['test_res'] = housetest['y_test'] - housetest['predictions']

print(housetest.shape)
housetest.head(1)

```

A. Appendix

```
fig, ax = plt.subplots(figsize=(10, 10), subplot_kw={'aspect': 'equal'})
housestest.plot(column='test_res', scheme='Quantiles', k=5,
                 cmap='coolwarm', legend=True, ax=ax)
ax.set_axis_off()
plt.show()

fig, ax = plt.subplots(figsize=(10, 10), subplot_kw={'aspect': 'equal'})
housestest.plot(column='test_res', scheme='Quantiles', k=5,
                 cmap='coolwarm', legend=True, ax=ax, markersize=5)
ctx.add_basemap(ax, crs=housestest.crs.to_string(),
                 source=ctx.providers.CartoDB.Positron)
ax.set_axis_off()
save_path = r'Y:\Lukas\thesis\regionalization plots/basemap_RF.png'
#plt.savefig(save_path)
plt.show()

# new spatial weights for moran's I
wi = ps.weights.distance.KNN.from_dataframe(housestest, k=6)
wi.transform = 'r'
y = housestest['test_res']

mi = esda.Moran(y, wi, permutations = 1000, two_tailed=False)
print( 'Moran I is', round(mi.I, 3), ', p-value is', mi.p_sim,
      ', expected value is', mi.EI)

# Calculate LISA
lisa = esda.moran.Moran_Local(y, wi)

# Calculate spatial groupings and significance
housestest['Is'] = lisa.Is
housestest['pvalue'] = lisa.p_sim
housestest['group'] = lisa.q
housestest.loc[housestest['pvalue'] > 0.05, 'group'] = 'not-significant'
housestest.loc[housestest['group'] == 1, 'group'] = 'hotspot'
housestest.loc[housestest['group'] == 2, 'group'] = 'cold outlier'
housestest.loc[housestest['group'] == 3, 'group'] = 'coldspot'
housestest.loc[housestest['group'] == 4, 'group'] = 'hot outlier'
print(housestest['group'].value_counts())
housestest.head(2)

state_boundary = gpd.read_file(r"Y:/Lukas/thesis/bus1123/bufferedcliprahmen.shp")
fig, ax = plt.subplots(figsize=(10,10), subplot_kw={'aspect': 'equal'})
```

```

group_colors = {'not-significant': 'grey',
                'hotspot': 'red',
                'coldspot': 'blue',
                'cold outlier': 'lightblue',
                'hot outlier': 'pink'}

state_boundary.plot(ax=ax, edgecolor='black', linewidth=0.7, facecolor='none')

for ctype, data in housestest.groupby('group'):
    if ctype == 'not-significant':
        color = group_colors[ctype]
        data.plot(color=color, ax=ax, label=ctype, markersize=15, alpha=0.8)

for ctype, data in housestest.groupby('group'):
    if ctype != 'not-significant':
        color = group_colors[ctype]
        data.plot(color=color, ax=ax, label=ctype, markersize=15, alpha=0.8)

ax.legend(bbox_to_anchor=(0.8, .8), prop={'size': 11})

ax.set_axis_off()
save_path = r'Y:\Lukas\thesis\regionalization plots\lisa_RF_S2_small.png'
#plt.savefig(save_path)
plt.show()

# Runtime
print("Time for whole script: {:.2f} seconds".format(time() - start_time))

```

A.1.2. Geographically Weighted Regression

```

# Imports
import numpy as np
import pandas as pd
import geopandas as gpd
import libpysal as ps
from mgwr.gwr import GWR, MGWR
from mgwr.sel_bw import Sel_BW
import matplotlib.pyplot as plt
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
import mapclassify
import esda
import contextily as ctx

```

A. Appendix

```
# Load and prepare data
# BIG DATASET
# Split 1
#housestr = pd.read_csv(r"pth", sep = ";")
#houseste = pd.read_csv(r"pth", sep = ";")

# Split 2
#housestr = pd.read_csv(r"pth", sep = ";")
#houseste = pd.read_csv(r"pth", sep = ";")

# Split 3
#housestr = pd.read_csv(r"pth", sep = ";")
#houseste = pd.read_csv(r"pth", sep = ";")

# SMALL DATASET
# Split 1
housestrain = gpd.read_file(r"pth")
housestest = gpd.read_file(r"pth")

# Split 2
#housestrain = gpd.read_file(r"pth")
#housestest = gpd.read_file(r"pth")

# Split 3
#housestrain = gpd.read_file(r"pth")
#housestest = gpd.read_file(r"pth")

housestrain.head(1)

# The following lines are only needed for the big dataset
,,,
housestrain = gpd.GeoDataFrame(
    housestr, geometry=gpd.points_from_xy(housestr.lon, housestr.lat),
    crs="EPSG:3310")

housestest = gpd.GeoDataFrame(
    houseste, geometry=gpd.points_from_xy(houseste.lon, houseste.lat),
    crs="EPSG:3310")
housestest = housestest.drop(["lon", "lat"], axis=1)
housestrain = housestrain.drop(["lon", "lat"], axis=1)

housestrain.head(1)
,,,

```

```

print(housestrain.crs)

housestrain['X'] = housestrain['geometry'].x
housestrain['Y'] = housestrain['geometry'].y
print(housestrain.shape)
housestrain.head(1)

housetest['X'] = housetest['geometry'].x
housetest['Y'] = housetest['geometry'].y
print(housetest.shape)
housetest.head(1)

y_train = housestrain["MedHouseVa"].values.reshape((-1, 1))
X_train = housestrain[['MedIncome', 'MedianAge', 'Population', 'Households',
                        'roomsperHH', 'bedroomHH']].values
u_train = housestrain['X']
v_train = housestrain['Y']
coords_train = list(zip(u_train, v_train))

print(y_train.shape)
print(X_train.shape)
print(u_train.shape)
print(v_train.shape)
print(len(coords_train))

y_test = housetest["MedHouseVa"].values.reshape((-1, 1))
X_test = housetest[['MedIncome', 'MedianAge', 'Population', 'Households',
                    'roomsperHH', 'bedroomHH']].values
u_test = housetest['X']
v_test = housetest['Y']
coords_test = list(zip(u_test, v_test))

print(y_test.shape)
print(X_test.shape)
print(u_test.shape)
print(v_test.shape)
print(len(coords_test))

# Bandwidth Selection
gwr_selector = Sel_BW(coords_train, y_train, X_train, fixed=False)
gwr_bw = gwr_selector.search(criterion='AICc')
print(gwr_bw)

```

A. Appendix

```
# Calibrate model and predict
model = GWR(coords_train, y_train, X_train, gwr_bw)
gwr_results = model.fit()

print(gwr_results.aic)
print(gwr_results.R2)

scale = gwr_results.scale
residuals = gwr_results.resid_response

test_coords_array = np.array(coords_test)
test_X_array = np.array(X_test)
test_y_array = np.array(y_test)

# check if the split is done correctly
print("train_coords shape:", np.array(coords_train).shape)
print("train_X shape:", np.array(X_train).shape)
print("train_y shape:", np.array(y_train).shape)
print("test_coords shape:", np.array(coords_test).shape)
print("test_X shape:", np.array(X_test).shape)
print("test_y shape:", np.array(y_test).shape)

test_results = model.predict(test_coords_array, test_X_array,
                             scale, residuals)

# check if this lines up properly as well
print(test_results.predictions.shape)
print(test_y_array.shape)

# Evaluate Results
mae = mean_absolute_error(test_y_array, test_results.predictions)
mse = mean_squared_error(test_y_array, test_results.predictions)
rmse = np.sqrt(mse)
r2 = r2_score(test_y_array, test_results.predictions)

print(f'MAE: {mae}')
print(f'MSE: {mse}')
print(f'RMSE: {rmse}')
print(f'R2: {r2}')
print(f'BW used in this run: {gwr_bw}')
print("Time for GWR + prediction: {:.2f} seconds".format(time() - start_time))
```

```

# LISA
housestest['predictions'] = test_results.predictions.flatten()
housestest['y_test'] = y_test
housestest['test_res'] = housestest['y_test'] - housestest['predictions']

print(housestest.shape)
housestest.head(1)

# plot residuals
fig, ax = plt.subplots(figsize=(15, 15), subplot_kw={'aspect': 'equal'})
housestest.plot(column='test_res', scheme='Quantiles', k=5,
                 cmap='coolwarm', legend=True, ax=ax)
ax.set_axis_off()
save_path = r'Y:\Lukas\thesis\regionalization plots/res_RF.png'
#plt.savefig(save_path)
plt.show()

fig, ax = plt.subplots(figsize=(10, 10), subplot_kw={'aspect': 'equal'})
housestest.plot(column='test_res', scheme='Quantiles', k=5, cmap='coolwarm',
                 legend=True, ax=ax, markersize=5)
ctx.add_basemap(ax, crs=housestest.crs.to_string(),
                 source=ctx.providers.Cartodb.Positron)
ax.set_axis_off()
save_path = r'Y:\Lukas\thesis\regionalization plots/basemap_GWR.png'
#plt.savefig(save_path)
plt.show()

# new spatial weights for moran's I
wi = ps.weights.distance.KNN.from_dataframe(housestest, k=7)
wi.transform = 'r'
y = housestest['test_res']

mi = esda.Moran(y, wi, permutations = 1000, two_tailed=False)
print( 'Moran I is', round(mi.I, 3), 'and p-value is', mi.p_sim,
       'expected value is', mi.EI)

# Calculate LISA
lisa = esda.moran.Moran_Local(y, wi)

# Calculate spatial groupings and significance
housestest['Is'] = lisa.Is
housestest['pvalue'] = lisa.p_sim

```

A. Appendix

```
housestest['group'] = lisa.q
housestest.loc[housestest['pvalue'] > 0.05, 'group'] = 'not-significant'
housestest.loc[housestest['group'] == 1, 'group'] = 'hotspot'
housestest.loc[housestest['group'] == 2, 'group'] = 'cold outlier'
housestest.loc[housestest['group'] == 3, 'group'] = 'coldspot'
housestest.loc[housestest['group'] == 4, 'group'] = 'hot outlier'
print(housestest['group'].value_counts())
housestest.head(2)

# Vizualize spatial groupings and significance
state_boundary = gpd.read_file(
    r"Y:/Lukas/thesis/bus1123/bufferedcliprahmen.shp")
fig, ax = plt.subplots(figsize=(10,10), subplot_kw={'aspect':'equal'})

group_colors = {'not-significant': 'grey',
                'hotspot': 'red',
                'coldspot': 'blue',
                'cold outlier': 'lightblue',
                'hot outlier': 'pink'}

state_boundary.plot(ax=ax, edgecolor='black', linewidth=0.7, facecolor='none')

for ctype, data in housestest.groupby('group'):
    if ctype == 'not-significant':
        color = group_colors[ctype]
        data.plot(color=color, ax=ax, label=ctype, markersize=15, alpha=0.8)

for ctype, data in housestest.groupby('group'):
    if ctype != 'not-significant':
        color = group_colors[ctype]
        data.plot(color=color, ax=ax, label=ctype, markersize=15, alpha=0.8)

ax.legend(bbox_to_anchor=(0.8, .8), prop={'size': 11})

ax.set_axis_off()
save_path = r'Y:\Lukas\thesis\regionalization plots\lisa_GWR_S3_small.png'
plt.savefig(save_path)
plt.show()

# Runtime
print("Time for whole script: {:.2f} seconds".format(time() - start_time))
```

A.1.3. Geographically Weighted Random Forest

R script for the prediction

```
install.packages("SpatialML")
install.packages("sf")
start_time <- Sys.time()
library(SpatialML)
library(ranger)
library(sf)

# Specify the file path and name
# Big Dataset - does not run
# Split 1
#file_path_train <- "pth"
#file_path_test <- "pth"
# Split 2
#file_path_train <- "pth"
#file_path_test <- "pth"
# Split 3
#file_path_train <- "pth"
#file_path_test <- "pth"

#Small Dataset
# Split 1
file_path_test <- "pth"
file_path_train <- "pth"

# Split 2
#file_path_test <- "pth"
#file_path_train <- "pth"

# Split 3
#file_path_test <- "pth"
#file_path_train <- "pth"

# Read the CSV file with semicolon separator
train.df <- read.csv(file_path_train, header = TRUE, sep = ";", dec = ".")
head(train.df)

test.df <- read.csv(file_path_test, header = TRUE, sep = ";", dec = ".")
head(test.df)

set.seed(123)
```

A. Appendix

```
# Need to separate the coordinates for feeding it into the functions
traincoords <- subset(train.df, select = -c(
  MedHouseVa, MedIncome, MedianAge, Population, Households, roomsperHH, bedroomHH))
head(traincoords)

train.df <- subset(train.df, select = -c(lat, lon))
str(train.df)

results = rf.mtry.optim(MedHouseVa ~ MedIncome + MedianAge + Population
  + Households + roomsperHH + bedroomHH,
  dataset = train.df)

# Bandwidth was taken from the GWR Python script
bwe <- grf.bw(formula = MedHouseVa ~ MedIncome + MedianAge + Population
  + Households + roomsperHH + bedroomHH,
  dataset = train.df,
  kernel = "adaptive",
  coords = traincoords,
  bw.min = 40,
  bw.max = 100,
  step = 10,
  trees = 1,
  mtry = 3)

traincoords <- traincoords[, c("lon", "lat")]

grf1 <- grf(MedHouseVa ~ MedIncome + MedianAge + Population
  + Households + roomsperHH + bedroomHH, dframe=train.df,
  bw=75, kernel="adaptive", coords=traincoords, mtry=2)

predict <- predict.grf(grf1, test.df, x.var.name="lon", y.var.name="lat")
str(test.df)

# Extract the predicted and actual values
predicted <- predict
actual <- test.df$MedHouseVa

print(predicted)
print(actual)

# Calculate residuals
residuals <- actual - predicted
```

```

head(residuals)
# Save all columns from the test data, including residuals, to a CSV file
output_data <- cbind(
  test.df,
  MedHouseVa_Predicted = predicted,
  Residuals = residuals
)

output_file_path <- "pth"
write.csv(output_data, file = output_file_path, row.names = FALSE)

# Calculate R2
r2 <- cor(predicted, actual)^2

# Calculate MAE
mae <- mean(abs(predicted - actual))

# Calculate RMSE
rmse <- sqrt(mean((predicted - actual)^2))

# Print the evaluation metrics
print(paste("R2:", r2))
print(paste("MAE:", mae))
print(paste("RMSE:", rmse))

end_time <- Sys.time()
elapsed_time <- end_time - start_time
print(paste("Script execution time: ", elapsed_time))

```

Python script for the evaluation

```

import numpy as np
import libpysal as ps
import geopandas as gp
import pandas as pd
import geopandas as gpd
import matplotlib.pyplot as plt
import matplotlib as mpl
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
import mapclassify
import esda
import contextily as ctx

```

A. Appendix

```
# Modeling done in the R Script
# This is just reading a table that contains the residuals

# Split 1
houses = pd.read_csv(r"pth")

# Split 2
#houses = pd.read_csv(r"pth")

# Split 3
#houses = pd.read_csv(r"pth")

housesgdf = gp.GeoDataFrame(
    houses, geometry=gp.points_from_xy(houses.lon, houses.lat), crs="EPSG:3310"
)

housesgdf.info()
housesgdf.head(3)

# plot residuals
fig, ax = plt.subplots(figsize=(8,8), subplot_kw={'aspect':'equal'})
housesgdf.plot(column='Residuals', scheme='Quantiles', k=5,
               cmap='bwr', legend=True, ax=ax)

fig, ax = plt.subplots(figsize=(10, 10), subplot_kw={'aspect': 'equal'})
housesgdf.plot(column='Residuals', scheme='Quantiles', k=5, cmap='coolwarm',
               legend=True, ax=ax, markersize=5)
ctx.add_basemap(ax, crs=housesgdf.crs.to_string(),
               source=ctx.providers.CartoDB.Positron)
ax.set_axis_off()
save_path = r'Y:\Lukas\thesis\regionalization plots/basemap_GWRF.png'
#plt.savefig(save_path)
plt.show()

# new spatial weights for moran's I
wi = ps.weights.distance.KNN.from_dataframe(housesgdf, k=6)
wi.transform = 'r'
y = housesgdf['Residuals']

mi = esda.Moran(y, wi, permutations = 1000, two_tailed=False)
print( 'Moran I is', round(mi.I, 3), '; p-value is', mi.p_sim,
      '; expected value is', mi.EI)
```

```

# Calculate LISA
lisa = esda.moran.Moran_Local(y, wi)

# Calculate spatial groupings and significance
housesgdf['Is'] = lisa.Is
housesgdf['pvalue'] = lisa.p_sim
housesgdf['group'] = lisa.q
housesgdf.loc[housesgdf['pvalue'] > 0.05, 'group'] = 'not-significant'
housesgdf.loc[housesgdf['group'] == 1, 'group'] = 'hotspot'
housesgdf.loc[housesgdf['group'] == 2, 'group'] = 'cold outlier'
housesgdf.loc[housesgdf['group'] == 3, 'group'] = 'coldspot'
housesgdf.loc[housesgdf['group'] == 4, 'group'] = 'hot outlier'
print(housesgdf['group'].value_counts())
housesgdf.head(2)

# Vizualize spatial groupings and significance
state_boundary = gpd.read_file(r"Y:/Lukas/thesis/bus1123/bufferedcliprahmen.shp")
fig, ax = plt.subplots(figsize=(10,10), subplot_kw={'aspect':'equal'})

group_colors = {'not-significant': 'grey',
                'hotspot': 'red',
                'coldspot': 'blue',
                'cold outlier': 'lightblue',
                'hot outlier': 'pink'}

state_boundary.plot(ax=ax, edgecolor='black', linewidth=0.7, facecolor='none')

for ctype, data in housesgdf.groupby('group'):
    if ctype == 'not-significant':
        color = group_colors[ctype]
        data.plot(color=color, ax=ax, label=ctype, markersize=15, alpha=0.8)

for ctype, data in housesgdf.groupby('group'):
    if ctype != 'not-significant':
        color = group_colors[ctype]
        data.plot(color=color, ax=ax, label=ctype, markersize=15, alpha=0.8)

ax.legend(bbox_to_anchor=(0.8, .8), prop={'size': 11})

ax.set_axis_off()
save_path = r'Y:\Lukas\thesis\regionalization plots\lisa_GWRF_S2_small.png'
#plt.savefig(save_path)
plt.show()

```

A. Appendix

A.1.4. RegRF - WARD

```
# Imports, Load Data and inspect
import geopandas as gpd
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from libpysal.weights import Queen, Rook, KNN
import spopt
from spopt.region import WardSpatial
import shapely
from sklearn.ensemble import RandomForestRegressor
from sklearn.model_selection import RandomizedSearchCV
from sklearn.metrics import r2_score, mean_absolute_error, mean_squared_error
from sklearn.metrics import davies_bouldin_score, calinski_harabasz_score
from sklearn.metrics import silhouette_score
import libpysal as ps
import esda
import math
import contextily as ctx
pd.set_option('display.max_columns', None)

# BIG DATASET
# Split 1
#pth = r"pth"
#test_p_pth = r"pth"

# Split 2
#pth = r"pth"
#test_p_pth = r"pth"

# Split 3
#pth = r"pth"
#test_p_pth = r"pth"

# SMALL DATASET
# Split 1
pth = r"pth"
test_p_pth = r"pth"

# Split 2
#pth = r"pth"
#test_p_pth = r"pth"
```

```

# Split 3
#pth = r"pth"
#test_p_pth = r"pth"

houses = gpd.read_file(pth)

houses.head(2)

houses.info()

# WARD Regionalization
houses.shape

attribute_name = ['MedHouseVa']

wq = Queen.from_dataframe(houses)
wq.n

n_clusters = 23

#%%time
model = WardSpatial(houses, wq, attribute_name, n_clusters)
model.solve()

column_name = f'ward_{n_clusters}'

houses[column_name] = model.labels_

houses.plot(column=column_name, categorical=True)

# Choose best model, then only keep necessary columns
houses.head(1)

keepcolumns = ['MedHouseVa', 'MedIncome', 'MedianAge', 'Population',
               'Households', 'roomsperHH', 'bedroomHH', 'geometry',
               column_name]
houses = houses.loc[:, keepcolumns]
houses.head()

# Spatial join the test data

test_p = gpd.read_file(test_p_pth)

```

A. Appendix

```
test_p.head(2)

test_p.shape

test_p.plot()

# temporary copy so just the columns wanted are used in the join
temp_train_gdf = houses[[column_name, 'geometry']].copy()

joined_gdf = gpd.sjoin(test_p, temp_train_gdf, how="left", op="within")
joined_gdf = joined_gdf.drop(columns=['index_right'])
# --> provides test data with cluster number

houses.head(1)

joined_gdf.head(1)

joined_gdf.plot(column=column_name, categorical=True, figsize=(5,5), )

# Variable Definitions for RF models
y_train = houses['MedHouseVa']
y_test = joined_gdf['MedHouseVa']

X_train = houses.drop(["MedHouseVa", "geometry"], axis=1)
X_test = joined_gdf.drop(["MedHouseVa", "geometry"], axis=1)

print('Training Features Shape:', X_train.shape)
print('Training Labels Shape:', y_train.shape)
print('Testing Features Shape:', X_test.shape)
print('Testing Labels Shape:', y_test.shape)

# Define labels and weights for loop - change depending on which model get used
region_labels = houses[column_name].values
region_labels_test = joined_gdf[column_name].values
weights = [len(y_train[region_labels == i]) for i in range(len(
    np.unique(region_labels)))]

param_dist = {
    'n_estimators': np.arange(50, 210, 10),
    'max_features': ['sqrt', 'log2'],
    'max_depth': [None] + list(np.arange(10, 110, 10)),
    'min_samples_split': np.arange(2, 12),
    'min_samples_leaf': np.arange(1, 5),
```

```

    'bootstrap': [True, False]
}

rf = RandomForestRegressor(random_state=42, n_jobs=-1)

# RegRF_CN with Region Number as additional independent variable + HPO
#%%time
#start_time = time()
random_search = RandomizedSearchCV(rf, param_distributions=param_dist,
                                   scoring='neg_mean_squared_error',
                                   cv=5, n_iter=20, n_jobs=-1,
                                   random_state=42)

random_search.fit(X_train, y_train)
print("Best Hyperparameters:", random_search.best_params_)

#%%time
hpo_rf = RandomForestRegressor(**random_search.best_params_,
                              random_state=42, n_jobs=-1)

hpo_rf.fit(X_train, y_train)
predictions = hpo_rf.predict(X_test)

# Calculate Evaluation Metrics
errors = abs(predictions - y_test)
print('Mean Absolute Error:', round(np.mean(errors), 2), 'units.')
r2 = round(r2_score(y_test, predictions), 3)
print("R2: ", r2)
mse = mean_squared_error(y_test, predictions)
print("MSE: ", mse)
print("RMSE: ", math.sqrt(mse))
#print("Time for RegRF_CN: {:.2f} seconds".format(time() - start_time))

# LISA
test_p['predictions'] = predictions.flatten()
test_p['y_test'] = y_test
test_p['test_res'] = test_p['y_test'] - test_p['predictions']

print(test_p.shape)
test_p.head(1)

# plot residuals
fig, ax = plt.subplots(figsize=(8,8), subplot_kw={'aspect':'equal'})
test_p.plot(column='test_res', scheme='Quantiles', k=5,
            cmap='bwr', legend=True, ax=ax)

```

A. Appendix

```
fig, ax = plt.subplots(figsize=(10, 10), subplot_kw={'aspect': 'equal'})
test_p.plot(column='test_res', scheme='Quantiles', k=5,
            cmap='coolwarm', legend=True, ax=ax, markersize=5)
ctx.add_basemap(ax, crs=test_p.crs.to_string(),
               source=ctx.providers.CartoDB.Positron)
ax.set_axis_off()
save_path = r'Y:\Lukas\thesis\regionalization plots/basemap_WARD.png'
plt.savefig(save_path)
plt.show()

# new spatial weights for moran's I
wi = ps.weights.distance.KNN.from_dataframe(test_p, k=7)
wi.transform = 'r'
y = test_p['test_res']

mi = esda.Moran(y, wi, permutations = 1000, two_tailed=False)
print( 'Moran I is', round(mi.I, 3), 'and p-value is', mi.p_sim,
      'expected value is', mi.EI)

# Calculate LISA
lisa = esda.moran.Moran_Local(y, wi)

# Calculate spatial groupings and significance
test_p['Is'] = lisa.Is
test_p['pvalue'] = lisa.p_sim
test_p['group'] = lisa.q
test_p.loc[test_p['pvalue'] > 0.05, 'group'] = 'not-significant'
test_p.loc[test_p['group'] == 1, 'group'] = 'hotspot'
test_p.loc[test_p['group'] == 2, 'group'] = 'cold outlier'
test_p.loc[test_p['group'] == 3, 'group'] = 'coldspot'
test_p.loc[test_p['group'] == 4, 'group'] = 'hot outlier'
print(test_p ['group'].value_counts())
test_p.head(2)

state_boundary = gpd.read_file(r"pth")
fig, ax = plt.subplots(figsize=(10,10), subplot_kw={'aspect': 'equal'})

group_colors = {'not-significant': 'grey',
               'hotspot': 'red',
               'coldspot': 'blue',
               'cold outlier': 'lightblue',
               'hot outlier': 'pink'}
```

```

state_boundary.plot(ax=ax, edgecolor='black', linewidth=0.7,
                    facecolor='none')

for ctype, data in test_p.groupby('group'):
    if ctype == 'not-significant':
        color = group_colors[ctype]
        data.plot(color=color, ax=ax, label=ctype,
                  markersize=15, alpha=0.8)

for ctype, data in test_p.groupby('group'):
    if ctype != 'not-significant':
        color = group_colors[ctype]
        data.plot(color=color, ax=ax, label=ctype,
                  markersize=15, alpha=0.8)

ax.legend(bbox_to_anchor=(0.8, .8), prop={'size': 11})

ax.set_axis_off()
save_path = r'pth'
#plt.savefig(save_path)
plt.show()

# RegRF_SEP - Separate RF for each Region -
# was omitted in the process (might not run, depending on regionalization)
'''
predictions_list = []
y_test_list = []
residuals_list = []
all_r2 = []
all_mae = []
all_mse = []
all_rmse = []

for i in range(len(np.unique(region_labels))):
    X_region_train = X_train[region_labels == i]
    y_region_train = y_train[region_labels == i]
    X_region_test = X_test[region_labels_test == i]
    y_region_test = y_test[region_labels_test == i]

    rf.fit(X_region_train, y_region_train)

    y_pred = rf.predict(X_region_test)

```

A. Appendix

```
# Save results so they can all be analysed together
predictions_list.extend(y_pred)
y_test_list.extend(y_region_test)
residuals_list.extend(y_region_test - y_pred)

# Same for performance
r2 = r2_score(y_region_test, y_pred)
mae = mean_absolute_error(y_region_test, y_pred)
mse = mean_squared_error(y_region_test, y_pred)
rmse = np.sqrt(mse)

all_r2.append(r2)
all_mae.append(mae)
all_mse.append(mse)
all_rmse.append(rmse)

print(f'Region {i} - R2: {r2:.3f}, MAE: {mae:.3f}, MSE: {mse:.3f},
      RMSE: {rmse:.3f}')
```

```
# Calculate average errors
avg_r2 = np.mean(all_r2)
avg_mae = np.mean(all_mae)
avg_mse = np.mean(all_mse)
avg_rmse = np.mean(all_rmse)

print(f'\nAverage R2: {avg_r2:.3f}, Average MAE: {avg_mae:.3f},
      Average MSE: {avg_mse:.3f}, Average RMSE: {avg_rmse:.3f}')
```

```
,,,

# LISA
test_p['predictions'] = predictions_list
test_p['y_test'] = y_test_list
test_p['test_res'] = residuals_list

print(test_p.shape)
test_p.head(1)

# plot residuals
fig, ax = plt.subplots(figsize=(8,8), subplot_kw={'aspect':'equal'})
test_p.plot(column='test_res', scheme='Quantiles', k=5,
            cmap='bwr', legend=True, ax=ax)
```

```

# new spatial weights for moran's I
wi = ps.weights.distance.KNN.from_dataframe(test_p, k=5)
wi.transform = 'r'
y = test_p['test_res']

mi = esda.Moran(y, wi, permutations = 1000, two_tailed=False)
print( 'Moran I is', round(mi.I, 3), 'and p-value is', mi.p_sim,
      'expected value is', mi.EI)

# Calculate LISA
lisa = esda.moran.Moran_Local(y, wi)

# Calculate spatial groupings and significance
test_p['Is'] = lisa.Is
test_p['pvalue'] = lisa.p_sim
test_p['group'] = lisa.q
test_p.loc[test_p['pvalue'] > 0.05, 'group'] = 'not-significant'
test_p.loc[test_p['group'] == 1, 'group'] = 'hotspot'
test_p.loc[test_p['group'] == 2, 'group'] = 'cold outlier'
test_p.loc[test_p['group'] == 3, 'group'] = 'coldspot'
test_p.loc[test_p['group'] == 4, 'group'] = 'hot outlier'
print(test_p['group'].value_counts())
test_p.head(2)

fig, ax = plt.subplots(figsize=(8,8), subplot_kw={'aspect':'equal'})

group_colors = {'not-significant': 'grey',
                'hotspot': 'red',
                'coldspot': 'blue',
                'cold outlier': 'lightblue',
                'hot outlier': 'pink'}
group_colors

for ctype, data in test_p.groupby('group'):

    color = group_colors[ctype]

    data.plot(color=color, ax=ax, label=ctype)

ax.legend(bbox_to_anchor=(1.0, .5), prop={'size': 12})
ax.set(title='Spatial Groupings')

ax.set_axis_off()

```

A. Appendix

```
plt.show()

#RegRF_W - Multitple Global RFs with more weight on the specific regions
#%%time
models = []
all_r2 = []
all_mae = []
all_mse = []

param_dist = {
    'n_estimators': np.arange(10, 200, 10),
    'max_depth': [None] + list(np.arange(10, 110, 10)),
    'min_samples_split': np.arange(2, 11),
    'min_samples_leaf': np.arange(1, 5),
}

for i in range(len(np.unique(region_labels))):
    region_rf_model = RandomForestRegressor(random_state=42)
    random_search = RandomizedSearchCV(region_rf_model,
                                       param_distributions=param_dist,
                                       scoring='neg_mean_squared_error',
                                       cv=5, n_iter=20, n_jobs=-1,
                                       random_state=42)
    random_search.fit(X_train, y_train, sample_weight=np.where(region_labels == i,
                                                                weights[i], 1))

    best_params = random_search.best_params_

    rf_randhpo = RandomForestRegressor(random_state=42, **best_params)
    rf_randhpo.fit(X_train, y_train, sample_weight=np.where(region_labels == i,
                                                            weights[i], 1))

    models.append(rf_randhpo)

    y_pred = rf_randhpo.predict(X_test)
    r2 = r2_score(y_test, y_pred)
    mae = mean_absolute_error(y_test, y_pred)
    mse = mean_squared_error(y_test, y_pred)

    all_r2.append(r2)
    all_mae.append(mae)
    all_mse.append(mse)

    print(f'Region {i} - R2: {r2:.3f}, MAE: {mae:.3f}, MSE: {mse:.3f},
          Best Hyperparameters: {best_params}')
```

```

avg_r2 = np.mean(all_r2)
avg_mae = np.mean(all_mae)
avg_mse = np.mean(all_mse)

print(f'\nAverage R2: {avg_r2:.3f}, Average MAE: {avg_mae:.3f},
      Average MSE: {avg_mse:.3f}')

#%%time
test_p['residuals'] = 0
all_residuals = []
all_r2 = []
all_mae = []
all_mse = []
all_rmse = []

# Loop over regions
for i in range(len(np.unique(region_labels))):
    rf_randhpo = RandomForestRegressor(random_state=42, n_jobs=-1)
    rf_randhpo.fit(X_train, y_train, sample_weight=np.where(region_labels == i,
                                                             weights[i], 1))

    y_pred = rf_randhpo.predict(X_test)

    residuals = y_test - y_pred

    all_residuals.append(residuals)

    r2 = r2_score(y_test, y_pred)
    mae = mean_absolute_error(y_test, y_pred)
    mse = mean_squared_error(y_test, y_pred)
    rmse = np.sqrt(mse)

    all_r2.append(r2)
    all_mae.append(mae)
    all_mse.append(mse)
    all_rmse.append(rmse)

    print(f'Region {i} - R2: {r2:.3f}, MAE: {mae:.3f}, MSE: {mse:.3f},
          RMSE: {rmse:.3f}')

all_residuals_array = np.array(all_residuals)

```

A. Appendix

```
average_residuals = np.mean(all_residuals_array, axis=0)

test_p['residuals'] = average_residuals

avg_r2 = np.mean(all_r2)
avg_mae = np.mean(all_mae)
avg_mse = np.mean(all_mse)
avg_rmse = np.mean(all_rmse)

print(f'\nAverage R2: {avg_r2:.3f}, Average MAE: {avg_mae:.3f},
      Average MSE: {avg_mse:.3f}, Average RMSE: {avg_rmse:.3f}')

# LISA
test_p['test_res'] = test_p['residuals']

print(test_p.shape)
test_p.head(1)

# Global Spatial Autocorrelation
fig, ax = plt.subplots(figsize=(8,8), subplot_kw={'aspect':'equal'})
test_p.plot(column='test_res', scheme='Quantiles', k=5, cmap='bwr',
            legend=True, ax=ax)

# new spatial weights for moran's I
wi = ps.weights.distance.KNN.from_dataframe(test_p, k=7)
wi.transform = 'r'
y = test_p['test_res']

mi = esda.Moran(y, wi, permutations = 1000, two_tailed=False)
print( 'Moran I is', round(mi.I, 3), 'and p-value is', mi.p_sim,
      'expected value is', mi.EI)

# Calculate LISA
lisa = esda.moran.Moran_Local(y, wi)

# Calculate spatial groupings and significance
test_p['Is'] = lisa.Is
test_p['pvalue'] = lisa.p_sim
test_p['group'] = lisa.q
test_p.loc[test_p['pvalue'] > 0.05, 'group'] = 'not-significant'
test_p.loc[test_p['group'] == 1, 'group'] = 'hotspot'
test_p.loc[test_p['group'] == 2, 'group'] = 'cold outlier'
test_p.loc[test_p['group'] == 3, 'group'] = 'coldspot'
```

```

test_p.loc[test_p['group'] == 4, 'group'] = 'hot outlier'
print(test_p ['group'].value_counts())

# Vizualize spatial groupings and significance
fig, ax = plt.subplots(figsize=(8,8), subplot_kw={'aspect':'equal'})

group_colors = {'not-significant': 'grey',
                'hotspot': 'red',
                'coldspot': 'blue',
                'cold outlier': 'lightblue',
                'hot outlier': 'pink'}
group_colors

for ctype, data in test_p.groupby('group'):

    color = group_colors[ctype]

    data.plot(color=color, ax=ax, label=ctype)

ax.legend(bbox_to_anchor=(1.0, .5), prop={'size': 12})
ax.set(title='Spatial Groupings')

ax.set_axis_off()
plt.show()

# Runtime
print("Time for whole script: {:.2f} seconds".format(time() - start_time))

```

A.1.5. RegRF - AZP

```

# Imports, Load Data and inspect
import geopandas as gpd
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from libpysal.weights import Queen, Rook, KNN
import spopt
from spopt.region import AZP
import shapely
from sklearn.ensemble import RandomForestRegressor
from sklearn.model_selection import RandomizedSearchCV
from sklearn.metrics import r2_score, mean_absolute_error, mean_squared_error
from sklearn.metrics import davies_bouldin_score, calinski_harabasz_score

```

A. Appendix

```
from sklearn.metrics import silhouette_score
import libpysal as ps
import esda
import math
import contextily as ctx
pd.set_option('display.max_columns', None)

# BIG DATASET
# Split 1
#pth = r"pth"
#test_p_pth = r"pth"

# Split 2
#pth = r"pth"
#test_p_pth = r"pth"

# Split 3
#pth = r"pth"
#test_p_pth = r"pth"

# SMALL DATASET
# Split 1
pth = r"pth"
test_p_pth = r"pth"

# Split 2
#pth = r"pth"
#test_p_pth = r"pth"

# Split 3
#pth = r"pth"
#test_p_pth = r"pth"

houses = gpd.read_file(pth)

houses.head(2)

houses.shape

houses.info()

# AZP Regionalization
```

```

houses.shape

attribute_name = ['MedHouseVa']

wq = Queen.from_dataframe(houses)
wq.n

allow_move_strategy = None

n_clusters = 23

model = AZP(houses, wq, attribute_name, n_clusters,
            allow_move_strategy, random_state = 42)

#%%time
model.solve()

column_name = f'azp{n_clusters}'

houses[column_name] = model.labels_

houses.plot(column=column_name, categorical=True)

# Choose best model, then only keep necessary columns
houses.head(1)

keepcolumns = ['MedHouseVa', 'MedIncome', 'MedianAge',
               'Population', 'Households', 'roomsperHH',
               'bedroomHH', 'geometry',
               column_name]
houses = houses.loc[:, keepcolumns]
houses.head()

# Spatial join the test data
test_p = gpd.read_file(test_p_pth)

test_p.head(2)

test_p.shape

test_p.plot()

# temporary copy so just the columns wanted are used in the join

```

A. Appendix

```
temp_train_gdf = houses[[column_name, 'geometry']].copy()

joined_gdf = gpd.sjoin(test_p, temp_train_gdf, how="left", op="within")
joined_gdf = joined_gdf.drop(columns=['index_right'])
# --> gives us test data with cluster number

houses.head(1)

joined_gdf.head(1)

joined_gdf.plot(column=column_name, categorical=True, figsize=(5,5), )

# Variable Definitions for RF models
y_train = houses['MedHouseVa']
y_test = joined_gdf['MedHouseVa']

X_train = houses.drop(["MedHouseVa", "geometry"], axis=1)
X_test = joined_gdf.drop(["MedHouseVa", "geometry"], axis=1)

print('Training Features Shape:', X_train.shape)
print('Training Labels Shape:', y_train.shape)
print('Testing Features Shape:', X_test.shape)
print('Testing Labels Shape:', y_test.shape)

# Define labels and weights for loop - change depending on which model get used
region_labels = houses[column_name].values
region_labels_test = joined_gdf[column_name].values
weights = [len(y_train[region_labels == i]) for i in range(len(
    np.unique(region_labels)))]

param_dist = {
    'n_estimators': np.arange(50, 200, 10),
    'max_depth': [None] + list(np.arange(10, 110, 10)),
    'min_samples_split': np.arange(2, 11),
    'min_samples_leaf': np.arange(1, 5)
}

rf = RandomForestRegressor(random_state=42, n_jobs=-1)

# RegRF_CN with Region Number as additional independent variable + HPO
%%time
random_search = RandomizedSearchCV(rf, param_distributions=param_dist,
                                   scoring='neg_mean_squared_error',
```

```

                                cv=5, n_iter=20, n_jobs=-1,
                                random_state=42)
random_search.fit(X_train, y_train)
print("Best Hyperparameters:", random_search.best_params_)

#%%time
hpo_rf = RandomForestRegressor(**random_search.best_params_,
                                random_state=42, n_jobs=-1)
hpo_rf.fit(X_train, y_train)
predictions = hpo_rf.predict(X_test)

# Calculate Evaluation Metrics
errors = abs(predictions - y_test)
print('Mean Absolute Error:', round(np.mean(errors), 2), 'units.')
r2 = round(r2_score(y_test, predictions), 3)
print("R2: ", r2)
mse = mean_squared_error(y_test, predictions)
print("MSE: ", mse)
print("RMSE: ", math.sqrt(mse))

# LISA
test_p['predictions'] = predictions.flatten()
test_p['y_test'] = y_test
test_p['test_res'] = test_p['y_test'] - test_p['predictions']

print(test_p.shape)
test_p.head(1)

# plot residuals
fig, ax = plt.subplots(figsize=(8,8), subplot_kw={'aspect':'equal'})
test_p.plot(column='test_res', scheme='Quantiles', k=5,
             cmap='bwr', legend=True, ax=ax)

# new spatial weights for moran's I
wi = ps.weights.distance.KNN.from_dataframe(test_p, k=5)
wi.transform = 'r'
y = test_p['test_res']

mi = esda.Moran(y, wi, permutations = 1000, two_tailed=False)
print( 'Moran I is', round(mi.I, 3), 'and p-value is', mi.p_sim,
      'expected value is', mi.EI)

# Calculate LISA

```

A. Appendix

```
lisa = esda.moran.Moran_Local(y, wi)

# Calculate spatial groupings and significance
test_p['Is'] = lisa.Is
test_p['pvalue'] = lisa.p_sim
test_p['group'] = lisa.q
test_p.loc[test_p['pvalue'] > 0.05, 'group'] = 'not-significant'
test_p.loc[test_p['group'] == 1, 'group'] = 'hotspot'
test_p.loc[test_p['group'] == 2, 'group'] = 'cold outlier'
test_p.loc[test_p['group'] == 3, 'group'] = 'coldspot'
test_p.loc[test_p['group'] == 4, 'group'] = 'hot outlier'
print(test_p['group'].value_counts())
test_p.head(2)

state_boundary = gpd.read_file(r"pth")
fig, ax = plt.subplots(figsize=(10,10), subplot_kw={'aspect':'equal'})

group_colors = {'not-significant': 'grey',
                'hotspot': 'red',
                'coldspot': 'blue',
                'cold outlier': 'lightblue',
                'hot outlier': 'pink'}

state_boundary.plot(ax=ax, edgecolor='black', linewidth=0.7, facecolor='none')

for ctype, data in test_p.groupby('group'):
    if ctype == 'not-significant':
        color = group_colors[ctype]
        data.plot(color=color, ax=ax, label=ctype, markersize=15, alpha=0.8)

for ctype, data in test_p.groupby('group'):
    if ctype != 'not-significant':
        color = group_colors[ctype]
        data.plot(color=color, ax=ax, label=ctype, markersize=15, alpha=0.8)

ax.legend(bbox_to_anchor=(0.8, 0.8), prop={'size': 11})

ax.set_axis_off()
save_path = r'Y:\Lukas\thesis\regionalization plots/lisa_azp_S2_small.png'
plt.savefig(save_path)
plt.show()

# RegRF_SEP - Separate RF for each Region -
```

```

# was omitted in the process (might not run, depening on regionalization)
,,,
predictions_list = []
y_test_list = []
residuals_list = []
all_r2 = []
all_mae = []
all_mse = []
all_rmse = []

for i in range(len(np.unique(region_labels))):
    X_region_train = X_train[region_labels == i]
    y_region_train = y_train[region_labels == i]
    X_region_test = X_test[region_labels_test == i]
    y_region_test = y_test[region_labels_test == i]

    rf.fit(X_region_train, y_region_train)

    y_pred = rf.predict(X_region_test)

    # Save results so they can all be analysed together
    predictions_list.extend(y_pred)
    y_test_list.extend(y_region_test)
    residuals_list.extend(y_region_test - y_pred)

    # Same for performance
    r2 = r2_score(y_region_test, y_pred)
    mae = mean_absolute_error(y_region_test, y_pred)
    mse = mean_squared_error(y_region_test, y_pred)
    rmse = np.sqrt(mse)

    all_r2.append(r2)
    all_mae.append(mae)
    all_mse.append(mse)
    all_rmse.append(rmse)

    print(f'Region {i} - R2: {r2:.3f}, MAE: {mae:.3f}, MSE: {mse:.3f},
          RMSE: {rmse:.3f}')

# Calculate average errors
avg_r2 = np.mean(all_r2)
avg_mae = np.mean(all_mae)
avg_mse = np.mean(all_mse)

```

A. Appendix

```
avg_rmse = np.mean(all_rmse)

print(f'\nAverage R2: {avg_r2:.3f}, Average MAE: {avg_mae:.3f},
      Average MSE: {avg_mse:.3f}, Average RMSE: {avg_rmse:.3f}')
'''

# LISA
test_p['predictions'] = predictions_list
test_p['y_test'] = y_test_list
test_p['test_res'] = residuals_list

print(test_p.shape)
test_p.head(1)

# plot residuals
fig, ax = plt.subplots(figsize=(8,8), subplot_kw={'aspect':'equal'})
test_p.plot(column='test_res', scheme='Quantiles', k=5,
            cmap='bwr', legend=True, ax=ax)

# new spatial weights for moran's I
wi = ps.weights.distance.KNN.from_dataframe(test_p, k=5)
wi.transform = 'r'
y = test_p['test_res']

mi = esda.Moran(y, wi, permutations = 1000, two_tailed=False)
print( 'Moran I is', round(mi.I, 3), 'and p-value is', mi.p_sim,
      'expected value is', mi.EI)

# Calculate LISA
lisa = esda.moran.Moran_Local(y, wi)

# Calculate spatial groupings and significance
test_p['Is'] = lisa.Is
test_p['pvalue'] = lisa.p_sim
test_p['group'] = lisa.q
test_p.loc[test_p['pvalue'] > 0.05, 'group'] = 'not-significant'
test_p.loc[test_p['group'] == 1, 'group'] = 'hotspot'
test_p.loc[test_p['group'] == 2, 'group'] = 'cold outlier'
test_p.loc[test_p['group'] == 3, 'group'] = 'coldspot'
test_p.loc[test_p['group'] == 4, 'group'] = 'hot outlier'
print(test_p ['group'].value_counts())
test_p.head(2)
```

```

fig, ax = plt.subplots(figsize=(8,8), subplot_kw={'aspect':'equal'})

group_colors = {'not-significant': 'grey',
                'hotspot': 'red',
                'coldspot': 'blue',
                'cold outlier': 'lightblue',
                'hot outlier': 'pink'}
group_colors

for ctype, data in test_p.groupby('group'):

    color = group_colors[ctype]

    data.plot(color=color, ax=ax, label=ctype)

ax.legend(bbox_to_anchor=(1.0, .5), prop={'size': 12})
ax.set(title='Spatial Groupings')

ax.set_axis_off()
plt.show()

# RegRF_W - Multitple Global RFs with more weight on the specific regions

#%%time
test_p['residuals'] = 0
all_residuals = []
all_r2 = []
all_mae = []
all_mse = []
all_rmse = []

# Loop over regions
for i in range(len(np.unique(region_labels))):
    rf_randhpo = RandomForestRegressor(random_state=42, n_jobs=-1)
    rf_randhpo.fit(X_train, y_train, sample_weight=np.where(region_labels == i,
                                                            weights[i], 1))

    y_pred = rf_randhpo.predict(X_test)

    residuals = y_test - y_pred

    all_residuals.append(residuals)

```

A. Appendix

```
r2 = r2_score(y_test, y_pred)
mae = mean_absolute_error(y_test, y_pred)
mse = mean_squared_error(y_test, y_pred)
rmse = np.sqrt(mse)

all_r2.append(r2)
all_mae.append(mae)
all_mse.append(mse)
all_rmse.append(rmse)

print(f'Region {i} - R2: {r2:.3f}, MAE: {mae:.3f}, MSE: {mse:.3f},
      RMSE: {rmse:.3f}')

all_residuals_array = np.array(all_residuals)

average_residuals = np.mean(all_residuals_array, axis=0)

test_p['residuals'] = average_residuals

avg_r2 = np.mean(all_r2)
avg_mae = np.mean(all_mae)
avg_mse = np.mean(all_mse)
avg_rmse = np.mean(all_rmse)

print(f'\nAverage R2: {avg_r2:.3f}, Average MAE: {avg_mae:.3f},
      Average MSE: {avg_mse:.3f}, Average RMSE: {avg_rmse:.3f}')

# LISA

test_p['test_res'] = test_p['residuals']

print(test_p.shape)
test_p.head(1)

# Global Spatial Autocorrelation
fig, ax = plt.subplots(figsize=(8,8), subplot_kw={'aspect':'equal'})
test_p.plot(column='test_res', scheme='Quantiles', k=5, cmap='bwr',
            legend=True, ax=ax)

# new spatial weights for moran's I
wi = ps.weights.distance.KNN.from_dataframe(test_p, k=5)
wi.transform = 'r'
y = test_p['test_res']
```

```

mi = esda.Moran(y, wi, permutations = 1000, two_tailed=False)
print( 'Moran I is', round(mi.I, 3), 'and p-value is', mi.p_sim,
      'expected value is', mi.EI)

# Calculate LISA
lisa = esda.moran.Moran_Local(y, wi)

# Calculate spatial groupings and significance
test_p['Is'] = lisa.Is
test_p['pvalue'] = lisa.p_sim
test_p['group'] = lisa.q
test_p.loc[test_p['pvalue'] > 0.05, 'group'] = 'not-significant'
test_p.loc[test_p['group'] == 1, 'group'] = 'hotspot'
test_p.loc[test_p['group'] == 2, 'group'] = 'cold outlier'
test_p.loc[test_p['group'] == 3, 'group'] = 'coldspot'
test_p.loc[test_p['group'] == 4, 'group'] = 'hot outlier'
print(test_p['group'].value_counts())
test_p.head(2)

# Vizualize spatial groupings and significance
fig, ax = plt.subplots(figsize=(8,8),
                      subplot_kw={'aspect': 'equal'})

group_colors = {'not-significant': 'grey',
                'hotspot': 'red',
                'coldspot': 'blue',
                'cold outlier': 'lightblue',
                'hot outlier': 'pink'}

group_colors

for ctype, data in test_p.groupby('group'):

    color = group_colors[ctype]

    data.plot(color=color, ax=ax, label=ctype)

ax.legend(bbox_to_anchor=(1.0, .5), prop={'size': 12})
ax.set(title='Spatial Groupings')

ax.set_axis_off()
plt.show()

```

A. Appendix

```
# Runtime
print("Time for whole script: {:.2f} seconds".format(time() - start_time))
```

A.1.6. RegRF - KMEANS

```
# Imports, Load Data and inspect
import geopandas as gpd
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from libpysal.weights import Queen, Rook, KNN
from spopt.region import RegionKMeansHeuristic
import shapely
from sklearn.ensemble import RandomForestRegressor
from sklearn.model_selection import RandomizedSearchCV
from sklearn.metrics import r2_score, mean_absolute_error, mean_squared_error
from sklearn.metrics import davies_bouldin_score, calinski_harabasz_score
from sklearn.metrics import silhouette_score
import libpysal as ps
import esda
import math
pd.set_option('display.max_columns', None)

# BIG DATASET
# Split 1
#pth = r"pth"
#test_p_pth = r"pth"

# Split 2
#pth = r"pth"
#test_p_pth = r"pth"

# Split 3
#pth = r"pth"
#test_p_pth = r"pth"

# SMALL DATASET
# Split 1
pth = r"pth"
test_p_pth = r"pth"

# Split 2
#pth = r"pth"
#test_p_pth = r"pth"
```

```

# Split 3
#pth = r"pth"
#test_p_pth = r"pth"

houses = gpd.read_file(pth)

houses.head(2)

houses.info()

# KMEANS Regionalization
houses.shape

attribute_name = ['MedHouseVa']

wq = Queen.from_dataframe(houses)
wq.n

n_clusters = 23

model = RegionKMeansHeuristic(houses[attribute_name].values, n_clusters, wq)

#%%time
model.solve()

column_name = f'kmeans_{n_clusters}'

houses[column_name] = model.labels_

houses.plot(column=column_name, categorical=True)

# Choose best model, then only keep necessary columns
houses.head(1)

keepcolumns = ['MedHouseVa', 'MedIncome', 'MedianAge',
               'Population', 'Households', 'roomsperHH',
               'bedroomHH', 'geometry',
               'kmeans_23']
houses = houses.loc[:, keepcolumns]
houses.head(1)

# Spatial join the test data

```

A. Appendix

```
test_p = gpd.read_file(test_p_pth)

test_p.head(2)

test_p.shape

test_p.plot()

# temporary copy so just the columns wanted are used in the join
temp_train_gdf = houses[['kmeans_23', 'geometry']].copy()

joined_gdf = gpd.sjoin(test_p, temp_train_gdf, how="left", op="within")
joined_gdf = joined_gdf.drop(columns=['index_right'])
# --> gives us test data with cluster number

houses.head(1)

joined_gdf.head(1)

joined_gdf.plot(column='kmeans_23', categorical=True, figsize=(5,5), )

# Variable Definitions for RF models
y_train = houses['MedHouseVa']
y_test = joined_gdf['MedHouseVa']

X_train = houses.drop(["MedHouseVa", "geometry"], axis=1)
X_test = joined_gdf.drop(["MedHouseVa", "geometry"], axis=1)

print('Training Features Shape:', X_train.shape)
print('Training Labels Shape:', y_train.shape)
print('Testing Features Shape:', X_test.shape)
print('Testing Labels Shape:', y_test.shape)

# Define labels and weights for loop - change depending on which model get used
region_labels = houses['kmeans_23'].values
region_labels_test = joined_gdf['kmeans_23'].values
weights = [len(y_train[region_labels == i]) for i in range(len(
    np.unique(region_labels)))]

param_dist = {
    'n_estimators': np.arange(50, 210, 10),
    'max_features': ['sqrt', 'log2'],
    'max_depth': [None] + list(np.arange(10, 110, 10)),
```

```

    'min_samples_split': np.arange(2, 12),
    'min_samples_leaf': np.arange(1, 5),
    'bootstrap': [True, False]
}

rf = RandomForestRegressor(random_state=42, n_jobs=-1)

# RegRF_CN with Region Number as additional independent variable + HPO
%%time
random_search = RandomizedSearchCV(rf, param_distributions=param_dist,
                                   scoring='neg_mean_squared_error',
                                   cv=5, n_iter=20, n_jobs=-1,
                                   random_state=42)

random_search.fit(X_train, y_train)
print("Best Hyperparameters:", random_search.best_params_)

%%time
hpo_rf = RandomForestRegressor(**random_search.best_params_,
                              random_state=42, n_jobs=-1)

hpo_rf.fit(X_train, y_train)
predictions = hpo_rf.predict(X_test)

# Calculate Evaluation Metrics
errors = abs(predictions - y_test)
print('Mean Absolute Error:', round(np.mean(errors), 2), 'units.')
r2 = round(r2_score(y_test, predictions), 3)
print("R2: ", r2)
mse = mean_squared_error(y_test, predictions)
print("MSE: ", mse)
print("RMSE: ", math.sqrt(mse))

# LISA
test_p['predictions'] = predictions.flatten()
test_p['y_test'] = y_test
test_p['test_res'] = test_p['y_test'] - test_p['predictions']

print(test_p.shape)
test_p.head(1)

# plot residuals
fig, ax = plt.subplots(figsize=(8,8), subplot_kw={'aspect':'equal'})
test_p.plot(column='test_res', scheme='Quantiles', k=5,
            cmap='bwr', legend=True, ax=ax)

```

A. Appendix

```
# new spatial weights for moran's I
wi = ps.weights.distance.KNN.from_dataframe(test_p, k=5)
wi.transform = 'r'
y = test_p['test_res']

mi = esda.Moran(y, wi, permutations = 1000, two_tailed=False)
print( 'Moran I is', round(mi.I, 3), 'and p-value is', mi.p_sim,
      'expected value is', mi.EI)

# Calculate LISA
lisa = esda.moran.Moran_Local(y, wi)

# Calculate spatial groupings and significance
test_p['Is'] = lisa.Is
test_p['pvalue'] = lisa.p_sim
test_p['group'] = lisa.q
test_p.loc[test_p['pvalue'] > 0.05, 'group'] = 'not-significant'
test_p.loc[test_p['group'] == 1, 'group'] = 'hotspot'
test_p.loc[test_p['group'] == 2, 'group'] = 'cold outlier'
test_p.loc[test_p['group'] == 3, 'group'] = 'coldspot'
test_p.loc[test_p['group'] == 4, 'group'] = 'hot outlier'
print(test_p['group'].value_counts())
test_p.head(2)

# Vizualize spatial groupings and significance
state_boundary = gpd.read_file(r"pth")
fig, ax = plt.subplots(figsize=(10,10), subplot_kw={'aspect':'equal'})

group_colors = {'not-significant': 'grey',
                'hotspot': 'red',
                'coldspot': 'blue',
                'cold outlier': 'lightblue',
                'hot outlier': 'pink'}

state_boundary.plot(ax=ax, edgecolor='black', linewidth=0.7, facecolor='none')

for ctype, data in test_p.groupby('group'):
    if ctype == 'not-significant':
        color = group_colors[ctype]
        data.plot(color=color, ax=ax, label=ctype, markersize=15, alpha=0.8)

for ctype, data in test_p.groupby('group'):
```

```

    if ctype != 'not-significant':
        color = group_colors[ctype]
        data.plot(color=color, ax=ax, label=ctype, markersize=15, alpha=0.8)

ax.legend(bbox_to_anchor=(0.8, 0.8), prop={'size': 11})

ax.set_axis_off()
save_path = r'pth'
#plt.savefig(save_path)
plt.show()

# RegRF_SEP - Separate RF for each Region -
# was omitted in the process (might not run, depending on regionalization)
'''
predictions_list = []
y_test_list = []
residuals_list = []
all_r2 = []
all_mae = []
all_mse = []
all_rmse = []

for i in range(len(np.unique(region_labels))):
    X_region_train = X_train[region_labels == i]
    y_region_train = y_train[region_labels == i]
    X_region_test = X_test[region_labels_test == i]
    y_region_test = y_test[region_labels_test == i]

    rf.fit(X_region_train, y_region_train)

    y_pred = rf.predict(X_region_test)

    # Save results so they can all be analysed together
    predictions_list.extend(y_pred)
    y_test_list.extend(y_region_test)
    residuals_list.extend(y_region_test - y_pred)

    # Same for performance
    r2 = r2_score(y_region_test, y_pred)
    mae = mean_absolute_error(y_region_test, y_pred)
    mse = mean_squared_error(y_region_test, y_pred)
    rmse = np.sqrt(mse)

```

A. Appendix

```
all_r2.append(r2)
all_mae.append(mae)
all_mse.append(mse)
all_rmse.append(rmse)

print(f'Region {i} - R2: {r2:.3f}, MAE: {mae:.3f}, MSE: {mse:.3f},
      RMSE: {rmse:.3f}')

# Calculate average errors
avg_r2 = np.mean(all_r2)
avg_mae = np.mean(all_mae)
avg_mse = np.mean(all_mse)
avg_rmse = np.mean(all_rmse)

print(f'\nAverage R2: {avg_r2:.3f}, Average MAE: {avg_mae:.3f},
      Average MSE: {avg_mse:.3f}, Average RMSE: {avg_rmse:.3f}')
'''

# LISA
test_p['predictions'] = predictions_list
test_p['y_test'] = y_test_list
test_p['test_res'] = residuals_list

print(test_p.shape)
test_p.head(1)

# plot residuals
fig, ax = plt.subplots(figsize=(8,8), subplot_kw={'aspect':'equal'})
test_p.plot(column='test_res', scheme='Quantiles', k=5,
            cmap='bwr', legend=True, ax=ax)

# new spatial weights for moran's I
wi = ps.weights.distance.KNN.from_dataframe(test_p, k=5)
wi.transform = 'r'
y = test_p['test_res']

mi = esda.Moran(y, wi, permutations = 1000, two_tailed=False)
print( 'Moran I is', round(mi.I, 3), 'and p-value is', mi.p_sim,
      'expected value is', mi.EI)

# Calculate LISA
lisa = esda.moran.Moran_Local(y, wi)
```

```

# Calculate spatial groupings and significance
test_p['Is'] = lisa.Is
test_p['pvalue'] = lisa.p_sim
test_p['group'] = lisa.q
test_p.loc[test_p['pvalue'] > 0.05, 'group'] = 'not-significant'
test_p.loc[test_p['group'] == 1, 'group'] = 'hotspot'
test_p.loc[test_p['group'] == 2, 'group'] = 'cold outlier'
test_p.loc[test_p['group'] == 3, 'group'] = 'coldspot'
test_p.loc[test_p['group'] == 4, 'group'] = 'hot outlier'
print(test_p['group'].value_counts())
test_p.head(2)

# Vizualize spatial groupings and significance
fig, ax = plt.subplots(figsize=(8,8), subplot_kw={'aspect':'equal'})

group_colors = {'not-significant': 'grey',
                'hotspot': 'red',
                'coldspot': 'blue',
                'cold outlier': 'lightblue',
                'hot outlier': 'pink'}
group_colors

for ctype, data in test_p.groupby('group'):

    color = group_colors[ctype]

    data.plot(color=color, ax=ax, label=ctype)

ax.legend(bbox_to_anchor=(1.0, .5), prop={'size': 12})
ax.set(title='Spatial Groupings')

ax.set_axis_off()
plt.show()

# RegRF_W - Multitple Global RFs with more weight on the specific regions
#%%time
test_p['residuals'] = 0
all_residuals = []
all_r2 = []
all_mae = []
all_mse = []
all_rmse = []

```

A. Appendix

```
# Loop over regions
for i in range(len(np.unique(region_labels))):
    rf_randhpo = RandomForestRegressor(random_state=42, n_jobs=-1)
    rf_randhpo.fit(X_train, y_train, sample_weight=np.where(region_labels == i,
                                                            weights[i], 1))

    y_pred = rf_randhpo.predict(X_test)

    residuals = y_test - y_pred

    all_residuals.append(residuals)

    r2 = r2_score(y_test, y_pred)
    mae = mean_absolute_error(y_test, y_pred)
    mse = mean_squared_error(y_test, y_pred)
    rmse = np.sqrt(mse)

    all_r2.append(r2)
    all_mae.append(mae)
    all_mse.append(mse)
    all_rmse.append(rmse)

    print(f'Region {i} - R2: {r2:.3f}, MAE: {mae:.3f}, MSE: {mse:.3f},
          RMSE: {rmse:.3f}')

all_residuals_array = np.array(all_residuals)

average_residuals = np.mean(all_residuals_array, axis=0)

test_p['residuals'] = average_residuals

avg_r2 = np.mean(all_r2)
avg_mae = np.mean(all_mae)
avg_mse = np.mean(all_mse)
avg_rmse = np.mean(all_rmse)

print(f'\nAverage R2: {avg_r2:.3f}, Average MAE: {avg_mae:.3f},
      Average MSE: {avg_mse:.3f}, Average RMSE: {avg_rmse:.3f}')

# LISA
test_p['test_res'] = test_p['residuals']

print(test_p.shape)
```

```

test_p.head(1)

# Global Spatial Autocorrelation
fig, ax = plt.subplots(figsize=(8,8), subplot_kw={'aspect':'equal'})
test_p.plot(column='test_res', scheme='Quantiles', k=5,
             cmap='bwr', legend=True, ax=ax)

# new spatial weights for moran's I
wi = ps.weights.distance.KNN.from_dataframe(test_p, k=5)
wi.transform = 'r'
y = test_p['test_res']

mi = esda.Moran(y, wi, permutations = 1000, two_tailed=False)
print( 'Moran I is', round(mi.I, 3), 'and p-value is', mi.p_sim,
       'expected value is', mi.EI)

# Calculate LISA
lisa = esda.moran.Moran_Local(y, wi)

# Calculate spatial groupings and significance
test_p['Is'] = lisa.Is
test_p['pvalue'] = lisa.p_sim
test_p['group'] = lisa.q
test_p.loc[test_p['pvalue'] > 0.05, 'group'] = 'not-significant'
test_p.loc[test_p['group'] == 1, 'group'] = 'hotspot'
test_p.loc[test_p['group'] == 2, 'group'] = 'cold outlier'
test_p.loc[test_p['group'] == 3, 'group'] = 'coldspot'
test_p.loc[test_p['group'] == 4, 'group'] = 'hot outlier'
print(test_p ['group'].value_counts())
test_p.head(2)

# Vizualize spatial groupings and significance
fig, ax = plt.subplots(figsize=(8,8), subplot_kw={'aspect':'equal'})

group_colors = {'not-significant': 'grey',
                'hotspot': 'red',
                'coldspot': 'blue',
                'cold outlier': 'lightblue',
                'hot outlier': 'pink'}
group_colors

for ctype, data in test_p.groupby('group'):

```

A. Appendix

```
color = group_colors[ctype]

data.plot(color=color, ax=ax, label=ctype)

ax.legend(bbox_to_anchor=(1.0, .5), prop={'size': 12})
ax.set(title='Spatial Groupings')

ax.set_axis_off()
plt.show()

# Runtime
print("Time for whole script: {:.2f} seconds".format(time() - start_time))
```

A.1.7. RegRF - SKATER

```
# Imports, Load Data and inspect
import geopandas as gpd
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from libpysal.weights import Queen, Rook, KNN
from spopt.region.skater import Skater
import shapely
from sklearn.ensemble import RandomForestRegressor
from sklearn.model_selection import RandomizedSearchCV
from sklearn.metrics import r2_score, mean_absolute_error, mean_squared_error
from sklearn.metrics import davies_bouldin_score, calinski_harabasz_score,
from sklearn.metrics import silhouette_score
import libpysal as ps
import esda
import math
from sklearn.metrics import pairwise as skm
pd.set_option('display.max_columns', None)

# BIG DATASET
# Split 1
#pth = r"pth"
#test_p_pth = r"pth"

# Split 2
#pth = r"pth"
#test_p_pth = r"pth"

# Split 3
```

```

#pth = r"pth"
#test_p_pth = r"pth"

# SMALL DATASET
# Split 1
pth = r"pth"
test_p_pth = r"pth"

# Split 2
#pth = r"pth"
#test_p_pth = r"pth"

# Split 3
#pth = r"pth"
#test_p_pth = r"pth"

houses = gpd.read_file(pth)

houses.head(2)

houses.shape

houses.info()

# SKATER Regionalization
houses.shape

attribute_name = ['MedHouseVa']

wq = Queen.from_dataframe(houses)
wq.n

n_clusters = 22
floor = 1

spanning_forest_kwds = dict(
    dissimilarity=skm.manhattan_distances,
    affinity=None,
    reduction=np.sum,
    center=np.mean,
    verbose=2)

trace = False

```

A. Appendix

```
islands = "increase"

model = Skater(
    houses,
    wq,
    attribute_name,
    n_clusters = n_clusters,
    floor = floor,
    trace = trace,
    islands = islands,
    spanning_forest_kwds = spanning_forest_kwds)

#%%time
model.solve()

column_name = f'skater_{n_clusters}'
print(column_name)

houses[column_name] = model.labels_

houses.head(1)

houses.plot(column=column_name, categorical=True)

pd.set_option('display.max_rows', 28)
houses['number'] = 1
houses[[column_name, 'number']].groupby(by=column_name).count()

single_object_clusters = houses[houses[column_name].apply(
    lambda x: houses[column_name].value_counts()[x] == 1)]

single_object_clusters

# Iterate over single-object clusters
for index, cluster in single_object_clusters.iterrows():
    cluster_geometry = cluster.geometry

    neighboring_clusters = houses[houses.geometry.touches(cluster_geometry)]

    biggest_cluster = neighboring_clusters[
        neighboring_clusters[column_name].apply(lambda x: houses[column_name]
                                                    .value_counts()[x] > 1)]
                                                    .sort_values(
```

```

        by=column_name,
        ascending=False).iloc[0]

    houses.at[index, column_name] = biggest_cluster[column_name]

single_object_clusters = houses[houses[column_name].apply(lambda x: houses[column_name]
                                                            .value_counts()[x] == 1)]

# no more left
single_object_clusters

pd.set_option('display.max_rows', 40)
houses['number'] = 1
houses[[column_name, 'number']].groupby(by=column_name).count()

# Choose best model, then only keep necessary columns

houses.head(1)

keepcolumns = ['MedHouseVa', 'MedIncome', 'MedianAge', 'Population',
               'Households', 'roomsperHH', 'bedroomHH', 'geometry',
               'skater_22']
houses = houses.loc[:, keepcolumns]
houses.head(1)

# Spatial join the test data

test_p = gpd.read_file(test_p_pth)

test_p.head(2)

test_p.shape

test_p.plot()

# temporary copy so just the columns wanted are in the join
temp_train_gdf = houses[['skater_22', 'geometry']].copy()

joined_gdf = gpd.sjoin(test_p, temp_train_gdf, how="left", op="within")
joined_gdf = joined_gdf.drop(columns=['index_right'])
# --> provides test data with cluster number

houses.head(1)

```

A. Appendix

```
joined_gdf.head(1)

print(joined_gdf.shape)
print(houses.shape)

joined_gdf.plot(column='skater_22', categorical=True, figsize=(5,5), )

# Variable Definitions for RF models

X_train = houses.drop(["MedHouseVa", "geometry"], axis=1)
y_train = houses['MedHouseVa']

X_test = joined_gdf.drop(["MedHouseVa", "geometry"], axis=1)
y_test = joined_gdf['MedHouseVa']

print('Training Features Shape:', X_train.shape)
print('Training Labels Shape:', y_train.shape)
print('Testing Features Shape:', X_test.shape)
print('Testing Labels Shape:', y_test.shape)

# Define labels and weights for loop - change depending on which model gets used
region_labels = houses['skater_22'].values
region_labels_test = joined_gdf['skater_22'].values
weights = [len(y_train[region_labels == i]) for i in range(len(
    np.unique(region_labels)))]

np.unique(region_labels)

np.unique(region_labels_test)

param_dist = {
    'n_estimators': np.arange(50, 210, 10),
    'max_features': ['sqrt', 'log2'],
    'max_depth': [None] + list(np.arange(10, 110, 10)),
    'min_samples_split': np.arange(2, 12),
    'min_samples_leaf': np.arange(1, 5),
    'bootstrap': [True, False]
}

rf = RandomForestRegressor(random_state=42, n_jobs=-1)

# RegRF_CN with Region Number as additional independent variable + HPO
```

```

#%%time
random_search = RandomizedSearchCV(rf, param_distributions=param_dist,
                                   scoring='neg_mean_squared_error',
                                   cv=5, n_iter=20, n_jobs=-1,
                                   random_state=42)

random_search.fit(X_train, y_train)
print("Best Hyperparameters:", random_search.best_params_)

#%%time
hpo_rf = RandomForestRegressor(**random_search.best_params_,
                              random_state=42, n_jobs=-1)

hpo_rf.fit(X_train, y_train)
predictions = hpo_rf.predict(X_test)

# Calculate Evaluation Metrics
errors = abs(predictions - y_test)
print('MAE:', round(np.mean(errors), 2), 'units.')
r2 = round(r2_score(y_test, predictions), 3)
print("R2: ", r2)
mse = mean_squared_error(y_test, predictions)
print("MSE: ", mse)
print("RMSE: ", math.sqrt(mse))

# LISA
test_p['predictions'] = predictions.flatten()
test_p['y_test'] = y_test
test_p['test_res'] = test_p['y_test'] - test_p['predictions']

print(test_p.shape)
test_p.head(1)

# plot residuals
fig, ax = plt.subplots(figsize=(8,8), subplot_kw={'aspect':'equal'})
test_p.plot(column='test_res', scheme='Quantiles', k=5,
            cmap='bwr', legend=True, ax=ax)

# new spatial weights for moran's I
wi = ps.weights.distance.KNN.from_dataframe(test_p, k=7)
wi.transform = 'r'
y = test_p['test_res']

mi = esda.Moran(y, wi, permutations = 1000, two_tailed=False)
print( 'Moran I is', round(mi.I, 3), 'and p-value is', mi.p_sim,

```

A. Appendix

```
'expected value is', mi.EI)

# Calculate LISA
lisa = esda.moran.Moran_Local(y, wi)

# Calculate spatial groupings and significance
test_p['Is'] = lisa.Is
test_p['pvalue'] = lisa.p_sim
test_p['group'] = lisa.q
test_p.loc[test_p['pvalue'] > 0.05, 'group'] = 'not-significant'
test_p.loc[test_p['group'] == 1, 'group'] = 'hotspot'
test_p.loc[test_p['group'] == 2, 'group'] = 'cold outlier'
test_p.loc[test_p['group'] == 3, 'group'] = 'coldspot'
test_p.loc[test_p['group'] == 4, 'group'] = 'hot outlier'
print(test_p['group'].value_counts())
test_p.head(2)

# Vizualize spatial groupings and significance
state_boundary = gpd.read_file(r"pth")
fig, ax = plt.subplots(figsize=(10,10), subplot_kw={'aspect':'equal'})

group_colors = {'not-significant': 'grey',
                'hotspot': 'red',
                'coldspot': 'blue',
                'cold outlier': 'lightblue',
                'hot outlier': 'pink'}

state_boundary.plot(ax=ax, edgecolor='black',
                   linewidth=0.7, facecolor='none')

for ctype, data in test_p.groupby('group'):
    if ctype == 'not-significant':
        color = group_colors[ctype]
        data.plot(color=color, ax=ax, label=ctype,
                  markersize=15, alpha=0.8)

for ctype, data in test_p.groupby('group'):
    if ctype != 'not-significant':
        color = group_colors[ctype]
        data.plot(color=color, ax=ax, label=ctype,
                  markersize=15, alpha=0.8)

ax.legend(bbox_to_anchor=(0.8, 0.8), prop={'size': 11})
```

```

ax.set_axis_off()
save_path = r'pth'
plt.savefig(save_path)
plt.show()

# ## RegRF_SEP - Separate RF for each Regionn -
# was omitted in the process (might not run, depening on regionalization)
'''
predictions_list = []
y_test_list = []
residuals_list = []
all_r2 = []
all_mae = []
all_mse = []
all_rmse = []

for i in range(len(np.unique(region_labels))):
    X_region_train = X_train[region_labels == i]
    y_region_train = y_train[region_labels == i]
    X_region_test = X_test[region_labels_test == i]
    y_region_test = y_test[region_labels_test == i]

    rf.fit(X_region_train, y_region_train)

    y_pred = rf.predict(X_region_test)

    # Save results so they can all be analysed together
    predictions_list.extend(y_pred)
    y_test_list.extend(y_region_test)
    residuals_list.extend(y_region_test - y_pred)

    # Same for performance
    r2 = r2_score(y_region_test, y_pred)
    mae = mean_absolute_error(y_region_test, y_pred)
    mse = mean_squared_error(y_region_test, y_pred)
    rmse = np.sqrt(mse)

    all_r2.append(r2)
    all_mae.append(mae)
    all_mse.append(mse)
    all_rmse.append(rmse)

```

A. Appendix

```
print(f'Region {i} - R2: {r2:.3f}, MAE: {mae:.3f}, MSE: {mse:.3f},
      RMSE: {rmse:.3f}')
```

```
# Calculate average errors
avg_r2 = np.mean(all_r2)
avg_mae = np.mean(all_mae)
avg_mse = np.mean(all_mse)
avg_rmse = np.mean(all_rmse)

print(f'\nAverage R2: {avg_r2:.3f}, Average MAE: {avg_mae:.3f},
      Average MSE: {avg_mse:.3f}, Average RMSE: {avg_rmse:.3f}')
```

```
,,,

# LISA
test_p['predictions'] = predictions_list
test_p['y_test'] = y_test_list
test_p['test_res'] = residuals_list

print(test_p.shape)
test_p.head(1)

# plot residuals
fig, ax = plt.subplots(figsize=(8,8), subplot_kw={'aspect':'equal'})
test_p.plot(column='test_res', scheme='Quantiles', k=5,
            cmap='bwr', legend=True, ax=ax)

# new spatial weights for moran's I
wi = ps.weights.distance.KNN.from_dataframe(test_p, k=5)
wi.transform = 'r'
y = test_p['test_res']

mi = esda.Moran(y, wi, permutations = 1000, two_tailed=False)
print( 'Moran I is', round(mi.I, 3), 'and p-value is', mi.p_sim,
      'expected value is', mi.EI)

# Calculate LISA
lisa = esda.moran.Moran_Local(y, wi)

# Calculate spatial groupings and significance
test_p['Is'] = lisa.Is
test_p['pvalue'] = lisa.p_sim
test_p['group'] = lisa.q
```

```

test_p.loc[test_p['pvalue'] > 0.05, 'group'] = 'not-significant'
test_p.loc[test_p['group'] == 1, 'group'] = 'hotspot'
test_p.loc[test_p['group'] == 2, 'group'] = 'cold outlier'
test_p.loc[test_p['group'] == 3, 'group'] = 'coldspot'
test_p.loc[test_p['group'] == 4, 'group'] = 'hot outlier'
print(test_p['group'].value_counts())
test_p.head(2)

# Vizualize spatial groupings and significance
fig, ax = plt.subplots(figsize=(8,8), subplot_kw={'aspect':'equal'})

group_colors = {'not-significant': 'grey',
                'hotspot': 'red',
                'coldspot': 'blue',
                'cold outlier': 'lightblue',
                'hot outlier': 'pink'}
group_colors

for ctype, data in test_p.groupby('group'):

    color = group_colors[ctype]

    data.plot(color=color, ax=ax, label=ctype)

ax.legend(bbox_to_anchor=(1.0, .5), prop={'size': 12})
ax.set(title='Spatial Groupings')

ax.set_axis_off()
plt.show()

# RegRF_W - Multitple Global RFs with more weight on the specific regions
#%%time
test_p['residuals'] = 0
all_residuals = []
all_r2 = []
all_mae = []
all_mse = []
all_rmse = []

# Loop over regions
for i in range(len(np.unique(region_labels))):
    rf_randhpo = RandomForestRegressor(random_state=42, n_jobs=-1)
    rf_randhpo.fit(X_train, y_train, sample_weight=np.where(

```

A. Appendix

```
        region_labels == i, weights[i], 1))

y_pred = rf_randhpo.predict(X_test)

residuals = y_test - y_pred

all_residuals.append(residuals)

r2 = r2_score(y_test, y_pred)
mae = mean_absolute_error(y_test, y_pred)
mse = mean_squared_error(y_test, y_pred)
rmse = np.sqrt(mse)

all_r2.append(r2)
all_mae.append(mae)
all_mse.append(mse)
all_rmse.append(rmse)

print(f'Region {i} - R2: {r2:.3f}, MAE: {mae:.3f}, MSE: {mse:.3f},
      RMSE: {rmse:.3f}')

all_residuals_array = np.array(all_residuals)

average_residuals = np.mean(all_residuals_array, axis=0)

test_p['residuals'] = average_residuals

avg_r2 = np.mean(all_r2)
avg_mae = np.mean(all_mae)
avg_mse = np.mean(all_mse)
avg_rmse = np.mean(all_rmse)

print(f'\nAverage R2: {avg_r2:.3f}, Average MAE: {avg_mae:.3f},
      Average MSE: {avg_mse:.3f}, Average RMSE: {avg_rmse:.3f}')

# LISA
test_p.head(3)

test_p['test_res'] = test_p['residuals']

print(test_p.shape)
test_p.head(1)
```

```

# Global Spatial Autocorrelation
fig, ax = plt.subplots(figsize=(8,8), subplot_kw={'aspect':'equal'})
test_p.plot(column='test_res', scheme='Quantiles', k=5,
             cmap='bwr', legend=True, ax=ax)

# new spatial weights for moran's I
wi = ps.weights.distance.KNN.from_dataframe(test_p, k=5)
wi.transform = 'r'
y = test_p['test_res']

# In[60]:

mi = esda.Moran(y, wi, permutations = 1000, two_tailed=False)
print( 'Moran I is', round(mi.I, 3), 'and p-value is', mi.p_sim,
      'expected value is', mi.EI)

# Calculate LISA
lisa = esda.moran.Moran_Local(y, wi)

# Calculate spatial groupings and significance
test_p['Is'] = lisa.Is
test_p['pvalue'] = lisa.p_sim
test_p['group'] = lisa.q
test_p.loc[test_p['pvalue'] > 0.05, 'group'] = 'not-significant'
test_p.loc[test_p['group'] == 1, 'group'] = 'hotspot'
test_p.loc[test_p['group'] == 2, 'group'] = 'cold outlier'
test_p.loc[test_p['group'] == 3, 'group'] = 'coldspot'
test_p.loc[test_p['group'] == 4, 'group'] = 'hot outlier'
print(test_p ['group'].value_counts())
test_p.head(2)

# Vizualize spatial groupings and significance
fig, ax = plt.subplots(figsize=(8,8), subplot_kw={'aspect':'equal'})

group_colors = {'not-significant': 'grey',
                'hotspot': 'red',
                'coldspot': 'blue',
                'cold outlier': 'lightblue',
                'hot outlier': 'pink'}

group_colors

```

A. Appendix

```
for ctype, data in test_p.groupby('group'):

    color = group_colors[ctype]

    data.plot(color=color, ax=ax, label=ctype)

ax.legend(bbox_to_anchor=(1.0, .5), prop={'size': 12})
ax.set(title='Spatial Groupings')

ax.set_axis_off()
plt.show()

# Runtime
print("Time for whole script: {:.2f} seconds".format(time() - start_time))
```

A.1.8. RegRF - MAXP

```
# Imports, Load Data and inspect
import geopandas as gpd
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from libpysal.weights import Queen, Rook, KNN
from spopt.region import MaxPHeuristic as MaxP
import shapely
from sklearn.ensemble import RandomForestRegressor
from sklearn.model_selection import RandomizedSearchCV
from sklearn.metrics import r2_score, mean_absolute_error, mean_squared_error
from sklearn.metrics import davies_bouldin_score, calinski_harabasz_score
from sklearn.metrics import silhouette_score
import libpysal as ps
import esda
import math
pd.set_option('display.max_columns', None)

# BIG DATASET
# Split 1
#pth = r"pth"
#test_p_pth = r"pth"

# Split 2
#pth = r"pth"
#test_p_pth = r"pth"
```

```

# Split 3
#pth = r"pth"
#test_p_pth = r"pth"

# SMALL DATASET
# Split 1
pth = r"pth"
test_p_pth = r"pth"

# Split 2
#pth = r"pth"
#test_p_pth = r"pth"

# Split 3
#pth = r"pth"
#test_p_pth = r"pth"

houses = gpd.read_file(pth)

houses.head(2)

houses.shape

houses.info()

# MAXP Regionalization
houses.shape

attribute_name = ['MedHouseVa']

wq = Queen.from_dataframe(houses)
wq.n

threshold_value = 65
houses["count"] = 1
threshold_name = "count"

# %%time
model = MaxP(houses, wq, attribute_name, threshold_name, threshold_value)
model.solve()

column_name = f'maxp_{threshold_value}'
houses[column_name] = model.labels_

```

A. Appendix

```
pd.set_option('display.max_rows', 300)
houses['number'] = 1
houses[[column_name, 'number']].groupby(by=column_name).count()

houses.plot(column=column_name, categorical=True)

# Choose best model, then only keep necessary columns
houses.head(1)

keepcolumns = ['MedHouseVa', 'MedIncome', 'MedianAge', 'Population',
               'Households', 'roomsperHH', 'bedroomHH', 'geometry',
               'maxp_65']
houses = houses.loc[:, keepcolumns]
houses.head()

# Spatial join the test data
test_p = gpd.read_file(test_p_pth)

test_p.head(2)

test_p.shape

test_p.plot()

# temporary copy so just the columns wanted are in the join
temp_train_gdf = houses[['maxp_65', 'geometry']].copy()

joined_gdf = gpd.sjoin(test_p, temp_train_gdf, how="left", op="within")
joined_gdf = joined_gdf.drop(columns=['index_right'])
# --> provides test data with cluster number

houses.head(3)

joined_gdf.head(3)

joined_gdf.plot(column='maxp_65', categorical=True, figsize=(5,5))

houses.head(1)

# Variable Definitions for RF models
# Need to replace maximum value in "maxp_650" column with 0
maxp_65_max_value = houses["maxp_65"].max()
```

```

houses["maxp_65"].replace(maxp_65_max_value, 0, inplace=True)
joined_gdf["maxp_65"].replace(maxp_65_max_value, 0, inplace=True)

# Define labels and weights for loop - change depending on which model gets used
region_labels = houses['maxp_65'].values
region_labels_test = joined_gdf['maxp_65'].values

y_train = houses['MedHouseVa']
y_test = joined_gdf['MedHouseVa']

X_train = houses.drop(["MedHouseVa", "geometry"], axis=1)
X_test = joined_gdf.drop(["MedHouseVa", "geometry"], axis=1)

weights = [len(y_train[region_labels == i]) for i in range(len(
    np.unique(region_labels)))]

y_train.info()
y_test.head(1)
y_train.head(80)
X_test.head(1)

print('Training Features Shape:', X_train.shape)
print('Training Labels Shape:', y_train.shape)
print('Testing Features Shape:', X_test.shape)
print('Testing Labels Shape:', y_test.shape)

region_labels.shape
region_labels_test.shape
np.unique(region_labels)
np.unique(region_labels_test)

param_dist = {
    'n_estimators': np.arange(50, 210, 10),
    'max_features': ['sqrt', 'log2'],
    'max_depth': [None] + list(np.arange(10, 110, 10)),
    'min_samples_split': np.arange(2, 12),
    'min_samples_leaf': np.arange(1, 5),
    'bootstrap': [True, False]
}

rf = RandomForestRegressor(random_state=42)

# RegRF_CN with Region Number as additional independent variable + HPO

```

A. Appendix

```
%%time
random_search = RandomizedSearchCV(rf, param_distributions=param_dist,
                                   scoring='neg_mean_squared_error',
                                   cv=5, n_iter=20, n_jobs=-1,
                                   random_state=42)

random_search.fit(X_train, y_train)
print("Best Hyperparameters:", random_search.best_params_)

%%time
hpo_rf = RandomForestRegressor(**random_search.best_params_,
                              random_state=42)

hpo_rf.fit(X_train, y_train)
predictions = hpo_rf.predict(X_test)

# Calculate Evaluation Metrics
errors = abs(predictions - y_test)
print('Mean Absolute Error:', round(np.mean(errors), 2), 'units.')
r2 = round(r2_score(y_test, predictions), 3)
print("R2: ", r2)
mse = mean_squared_error(y_test, predictions)
print("MSE: ", mse)
print("RMSE: ", math.sqrt(mse))

# LISA
test_p['predictions'] = predictions.flatten()
test_p['y_test'] = y_test
test_p['test_res'] = test_p['y_test'] - test_p['predictions']

print(test_p.shape)
test_p.head(1)

# plot residuals
fig, ax = plt.subplots(figsize=(8,8), subplot_kw={'aspect':'equal'})
test_p.plot(column='test_res', scheme='Quantiles', k=5,
            cmap='bwr', legend=True, ax=ax)

# new spatial weights for moran's I
wi = ps.weights.distance.KNN.from_dataframe(test_p, k=6)
wi.transform = 'r'
y = test_p['test_res']

mi = esda.Moran(y, wi, permutations = 1000, two_tailed=False)
print('Moran I is', round(mi.I, 3), 'and p-value is', mi.p_sim,
```

```

    'expected value is', mi.EI)

# Calculate LISA
lisa = esda.moran.Moran_Local(y, wi)

# Calculate spatial groupings and significance
test_p['Is'] = lisa.Is
test_p['pvalue'] = lisa.p_sim
test_p['group'] = lisa.q
test_p.loc[test_p['pvalue'] > 0.05, 'group'] = 'not-significant'
test_p.loc[test_p['group'] == 1, 'group'] = 'hotspot'
test_p.loc[test_p['group'] == 2, 'group'] = 'cold outlier'
test_p.loc[test_p['group'] == 3, 'group'] = 'coldspot'
test_p.loc[test_p['group'] == 4, 'group'] = 'hot outlier'
print(test_p['group'].value_counts())
test_p.head(2)

# Vizualize spatial groupings and significance
state_boundary = gpd.read_file(r"pth")
fig, ax = plt.subplots(figsize=(10,10), subplot_kw={'aspect':'equal'})

group_colors = {'not-significant': 'grey',
                'hotspot': 'red',
                'coldspot': 'blue',
                'cold outlier': 'lightblue',
                'hot outlier': 'pink'}

#group_colors
state_boundary.plot(ax=ax, edgecolor='black', linewidth=0.7, facecolor='none')

for ctype, data in test_p.groupby('group'):
    if ctype == 'not-significant':
        color = group_colors[ctype]
        data.plot(color=color, ax=ax, label=ctype, markersize=15, alpha=0.8)

for ctype, data in test_p.groupby('group'):
    if ctype != 'not-significant':
        color = group_colors[ctype]
        data.plot(color=color, ax=ax, label=ctype, markersize=15, alpha=0.8)

ax.legend(bbox_to_anchor=(0.8, 0.8), prop={'size': 11})
#ax.set(title='Spatial Groupings')

ax.set_axis_off()

```

A. Appendix

```
save_path = r'pth'
#plt.savefig(save_path)
plt.show()

# RegRF_SEP - Separate RF for each Region -
# was omitted in the process (might not run, depening on regionalization)
,,,

predictions_list = []
y_test_list = []
residuals_list = []
all_r2 = []
all_mae = []
all_mse = []
all_rmse = []

for i in range(len(np.unique(region_labels))):
    X_region_train = X_train[region_labels == i]
    y_region_train = y_train[region_labels == i]
    X_region_test = X_test[region_labels_test == i]
    y_region_test = y_test[region_labels_test == i]

    rf.fit(X_region_train, y_region_train)

    y_pred = rf.predict(X_region_test)

    # Save results so they can all be analysed together
    predictions_list.extend(y_pred)
    y_test_list.extend(y_region_test)
    residuals_list.extend(y_region_test - y_pred)

    # Same for performance
    r2 = r2_score(y_region_test, y_pred)
    mae = mean_absolute_error(y_region_test, y_pred)
    mse = mean_squared_error(y_region_test, y_pred)
    rmse = np.sqrt(mse)

    all_r2.append(r2)
    all_mae.append(mae)
    all_mse.append(mse)
    all_rmse.append(rmse)

    print(f'Region {i} - R2: {r2:.3f}, MAE: {mae:.3f}, MSE: {mse:.3f},
          RMSE: {rmse:.3f}')
```

```

# Calculate average errors
avg_r2 = np.mean(all_r2)
avg_mae = np.mean(all_mae)
avg_mse = np.mean(all_mse)
avg_rmse = np.mean(all_rmse)

print(f'\nAverage R2: {avg_r2:.3f}, Average MAE: {avg_mae:.3f},
      Average MSE: {avg_mse:.3f}, Average RMSE: {avg_rmse:.3f}')
'''

# LISA
test_p['predictions'] = predictions_list
test_p['y_test'] = y_test_list
test_p['test_res'] = residuals_list

print(test_p.shape)
test_p.head(1)

# plot residuals
fig, ax = plt.subplots(figsize=(8,8), subplot_kw={'aspect':'equal'})
test_p.plot(column='test_res', scheme='Quantiles', k=5,
            cmap='bwr', legend=True, ax=ax)

# new spatial weights for moran's I
wi = ps.weights.distance.KNN.from_dataframe(test_p, k=5)
wi.transform = 'r'
y = test_p['test_res']

mi = esda.Moran(y, wi, permutations = 1000, two_tailed=False)
print( 'Moran I is', round(mi.I, 3), 'and p-value is', mi.p_sim,
      'expected value is', mi.EI)

# Calculate LISA
lisa = esda.moran.Moran_Local(y, wi)

# Calculate spatial groupings and significance
test_p['Is'] = lisa.Is
test_p['pvalue'] = lisa.p_sim
test_p['group'] = lisa.q
test_p.loc[test_p['pvalue'] > 0.05, 'group'] = 'not-significant'
test_p.loc[test_p['group'] == 1, 'group'] = 'hotspot'
test_p.loc[test_p['group'] == 2, 'group'] = 'cold outlier'

```

A. Appendix

```
test_p.loc[test_p['group'] == 3, 'group'] = 'coldspot'
test_p.loc[test_p['group'] == 4, 'group'] = 'hot outlier'
print(test_p['group'].value_counts())
test_p.head(2)

# Vizualize spatial groupings and significance
fig, ax = plt.subplots(figsize=(8,8), subplot_kw={'aspect':'equal'})

group_colors = {'not-significant': 'grey',
                'hotspot': 'red',
                'coldspot': 'blue',
                'cold outlier': 'lightblue',
                'hot outlier': 'pink'}

group_colors

for ctype, data in test_p.groupby('group'):

    color = group_colors[ctype]

    data.plot(color=color, ax=ax, label=ctype)

ax.legend(bbox_to_anchor=(1.0, .5), prop={'size': 12})
ax.set(title='Spatial Groupings')

ax.set_axis_off()
plt.show()

# RegRF_W - Multitple Global RFs with more weight on the specific regions (+ HPO)
#%%time
test_p['residuals'] = 0
all_residuals = []
all_r2 = []
all_mae = []
all_mse = []
all_rmse = []

# Loop over regions
for i in range(len(np.unique(region_labels))):
    rf_randhpo = RandomForestRegressor(random_state=42, n_jobs=-1)
    rf_randhpo.fit(X_train, y_train, sample_weight=np.where(
        region_labels == i, weights[i], 1))

    y_pred = rf_randhpo.predict(X_test)
```

```

residuals = y_test - y_pred

all_residuals.append(residuals)

r2 = r2_score(y_test, y_pred)
mae = mean_absolute_error(y_test, y_pred)
mse = mean_squared_error(y_test, y_pred)
rmse = np.sqrt(mse)

all_r2.append(r2)
all_mae.append(mae)
all_mse.append(mse)
all_rmse.append(rmse)

print(f'Region {i} - R2: {r2:.3f}, MAE: {mae:.3f}, MSE: {mse:.3f},
      RMSE: {rmse:.3f}')

all_residuals_array = np.array(all_residuals)

average_residuals = np.mean(all_residuals_array, axis=0)

test_p['residuals'] = average_residuals

avg_r2 = np.mean(all_r2)
avg_mae = np.mean(all_mae)
avg_mse = np.mean(all_mse)
avg_rmse = np.mean(all_rmse)

print(f'\nAverage R2: {avg_r2:.3f}, Average MAE: {avg_mae:.3f},
      Average MSE: {avg_mse:.3f}, Average RMSE: {avg_rmse:.3f}')

# LISA
test_p.head(1)
test_p['test_res'] = test_p['residuals']

print(test_p.shape)
test_p.head(1)

# Global Spatial Autocorrelation
fig, ax = plt.subplots(figsize=(8,8), subplot_kw={'aspect':'equal'})
test_p.plot(column='test_res', scheme='Quantiles', k=5,
            cmap='bwr', legend=True, ax=ax)

```

A. Appendix

```
# new spatial weights for moran's I
wi = ps.weights.distance.KNN.from_dataframe(test_p, k=5)
wi.transform = 'r'
y = test_p['test_res']

mi = esda.Moran(y, wi, permutations = 1000, two_tailed=False)
print( 'Moran I is', round(mi.I, 3), 'and p-value is', mi.p_sim,
      'expected value is', mi.EI)

# Calculate LISA
lisa = esda.moran.Moran_Local(y, wi)

# Calculate spatial groupings and significance
test_p['Is'] = lisa.Is
test_p['pvalue'] = lisa.p_sim
test_p['group'] = lisa.q
test_p.loc[test_p['pvalue'] > 0.05, 'group'] = 'not-significant'
test_p.loc[test_p['group'] == 1, 'group'] = 'hotspot'
test_p.loc[test_p['group'] == 2, 'group'] = 'cold outlier'
test_p.loc[test_p['group'] == 3, 'group'] = 'coldspot'
test_p.loc[test_p['group'] == 4, 'group'] = 'hot outlier'
print(test_p ['group'].value_counts())
test_p.head(2)

# Vizualize spatial groupings and significance
fig, ax = plt.subplots(figsize=(8,8), subplot_kw={'aspect':'equal'})

group_colors = {'not-significant': 'grey',
                'hotspot': 'red',
                'coldspot': 'blue',
                'cold outlier': 'lightblue',
                'hot outlier': 'pink'}

group_colors

for ctype, data in test_p.groupby('group'):

    color = group_colors[ctype]

    data.plot(color=color, ax=ax, label=ctype)

ax.legend(bbox_to_anchor=(1.0, .5), prop={'size': 12})
ax.set(title='Spatial Groupings')
```

```
ax.set_axis_off()
plt.show()

# Runtime
print("Time for whole script: {:.2f} seconds".format(time() - start_time))
```

A. Appendix

A.2. Experiment Results tables

This section of the appendix contains tables which show additional performance metrics for the results tables from sections [4.2](#) and [4.3](#), as well as one big table with the results of the "manual" optimization procedure for RegRF.

A.2. Experiment Results tables

A.2.1. Complete table of performance metrics

Combination	MAE	R2	MSE	RMSE	MI-res	Insig. LISA
ward_S1_CN_big	37065.47	0.78	2868652298.1477137	53559.80113991942	0.24	3182
ward_S1_W_big	37456.663	0.772	2970506863.918	54447.699	0.192	3287
ward_S2_CN_big	36042.54	0.797	2696154271.0920324	51924.5054968464	0.216	3274
ward_S2_W_big	37184.386	0.783	2890987411.666	53680.651	0.186	3224
ward_S3_CN_big	35899.6	0.797	2634237374.2599974	51324.82220388101	0.253	3133
ward_S3_W_big	37394.911	0.775	2924545782.469	53975.838	0.23	3206
ward_S1_CN_small	42677.5	0.749	3352245114.678477	57898.57610234017	0.129	343
ward_S1_W_small	42750.646	0.739	3474078550.114	58930.139	0.071	348
ward_S2_CN_small	46180.43	0.716	4085462287.951224	63917.62110679045	0.118	319
ward_S2_W_small	46192.111	0.698	4334776469.332	65811.617	0.071	347
ward_S3_CN_small	42348.43	0.758	3340867209.7871246	57800.23537830209	0.167	302
ward_S3_W_small	43256.906	0.74	3579163903.125	59780.059	0.13	310
azp_S1_CN_small	48804.69	0.659	4550296681.047717	67455.88692655161	0.123	354
azp_S1_W_small	48104.238	0.674	4347416951.065	65921.567	0.149	332
azp_S2_CN_small	46696.69	0.727	3917980139.6663055	62593.77077366649	0.141	334
azp_S2_W_small	47230.964	0.72	4025685346.803	63442.094	0.158	317
azp_S3_CN_small	43644.57	0.717	3906379362.387638	62501.03489053312	0.113	337
azp_S3_W_small	44533.543	0.707	4043015088.247	63571.96	0.132	340
skater_S1_CN_small	45631.07	0.714	3810824105.169126	61731.87268477384	0.16	316
skater_S1_W_small	45720.708	0.705	3939003830.067	62745.191	0.165	339
skater_S2_CN_small	43120.22	0.75	3595903175.217925	59965.85007500456	0.159	301
skater_S2_W_small	42603.338	0.744	3675309141.119	60598.344	0.118	348
skater_S3_CN_small	45189.81	0.707	4036772366.883018	63535.59920928595	0.214	304
skater_S3_W_small	45202.196	0.701	4123181147.821	64192.217	0.175	327
kmeans_S1_CN_small	47296.8	0.703	3956109278.344257	62897.609480363055	0.216	323
kmeans_S1_W_small	47526.418	0.695	4065323297.348	63742.795	0.172	332
kmeans_S2_CN_small	46476.32	0.734	3827809181.890953	61869.29110545031	0.238	327
kmeans_S2_W_small	45550.972	0.732	3850793019.649	62038.211	0.176	344
kmeans_S3_CN_small	46230.03	0.702	4101735603.930645	64044.79373009678	0.232	333
kmeans_S3_W_small	45005.859	0.7	4129710478.299	64246.039	0.156	341
maxp_S1_CN_small	47361.61	0.705	3936119198.244523	62738.49853355213	0.248	306
maxp_S1_W_small	46454.825	0.707	3912347009.274	62532.192	0.197	317
maxp_S2_CN_small	46682.62	0.734	3818943079.058248	61797.59768031641	0.228	293
maxp_S2_W_small	44229.501	0.746	3642989449.776	60333.666	0.158	346
maxp_S3_CN_small	42123.68	0.738	3605795166.267442	60048.273632698576	0.137	323
maxp_S3_W_small	42345.467	0.728	3754441131.259	61268.822	0.11	347
rf_S1_small	52382.95	0.636	4854454883.75	69673.92	0.294	312
gwr_S1_small	41615.41	0.7396	3740257801.37	61157.65	0.043	352
gwr_S1_small	40168.7	0.758	3240885992.29	56928.78	0.12	327
rf_S2_small	52881.86	0.677	4640036587.01	68117.81	0.306	263
gwr_S2_small	41615.41	0.74	3740257801.37	61157.65	0.052	353
gwr_S2_small	40035.62	0.774	3247135911.0	56983.646	0.07	355
rf_S3_small	50615.78	0.643	4925101435.25	70179.07	0.243	279
gwr_S3_small	39308.41	0.75493	3377898495.13	58119.69	0.006	360
gwr_S3_small	41176.84	0.765	3241197613.49	56931.52	0.07	356
rf_S1_big	47209.17	0.672	4273350208.98	65370.87	0.338	2873
gwr_S1_big	30436.46	0.801	2598912902.03	50979.53	0.026	3685
gwr_S1_big	-	-	-	-	-	-
rf_S2_big	48330.86	0.676	4313457403.4	65676.92	0.373	3019
gwr_S2_big	29852.45	0.8266	2308218185.94	48043.92	0.04	3774
gwr_S2_big	-	-	-	-	-	-
rf_S3_big	47489.77	0.674	4233114227.18	65062.39	0.38	2510
gwr_S3_big	29013.26	0.8329	2169506875.71	46577.96	0.042	3731
gwr_S3_big	-	-	-	-	-	-

Table A.1.: Table with all results

A.2.2. "Optimization" of WARD and AZP

nCLu	1wardCN	1wardW	1azpCN	1azpW	2wardCN	2wardW	2azpCN	2azpW	3wardCN	3wardW	3azpCN	3azpW
2	0.688	0.689	0.712	0.665	0.734	0.707	0.731	0.709	0.654	0.649	0.644	0.666
3	0.689	0.690	0.688	0.665	0.760	0.721	0.718	0.708	0.674	0.666	0.672	0.663
4	0.690	0.691	0.715	0.684	0.754	0.724	0.697	0.698	0.694	0.681	0.673	0.678
5	0.691	0.692	0.698	0.673	0.751	0.722	0.696	0.710	0.710	0.680	0.690	0.678
6	0.692	0.693	0.705	0.669	0.735	0.722	0.704	0.706	0.713	0.684	0.627	0.643
7	0.693	0.694	0.689	0.660	0.727	0.710	0.717	0.706	0.735	0.714	0.710	0.694
8	0.694	0.695	0.721	0.698	0.751	0.737	0.704	0.703	0.725	0.698	0.682	0.683
9	0.746	0.717	0.701	0.682	0.755	0.741	0.710	0.705	0.732	0.705	0.720	0.718
10	0.741	0.721	0.660	0.649	0.748	0.732	0.725	0.718	0.751	0.725	0.685	0.685
11	0.742	0.737	0.740	0.715	0.750	0.734	0.759	0.741	0.745	0.730	0.690	0.690
12	0.738	0.734	0.731	0.713	0.735	0.722	0.725	0.731	0.749	0.734	0.682	0.688
13	0.729	0.732	0.721	0.698	0.742	0.725	0.694	0.700	0.742	0.728	0.693	0.695
14	0.743	0.731	0.690	0.683	0.734	0.717	0.727	0.724	0.733	0.734	0.703	0.699
15	0.750	0.735	0.699	0.686	0.725	0.721	0.718	0.717	0.750	0.732	0.678	0.677
16	0.734	0.726	0.685	0.676	0.742	0.722	0.745	0.735	0.732	0.719	0.668	0.677
17	0.708	0.706	0.712	0.686	0.732	0.707	0.695	0.712	0.733	0.718	0.706	0.707
18	0.712	0.721	0.710	0.692	0.730	0.703	0.727	0.713	0.734	0.721	0.693	0.694
19	0.745	0.736	0.708	0.691	0.726	0.705	0.699	0.705	0.732	0.717	0.704	0.699
20	0.723	0.721	0.694	0.688	0.729	0.702	0.736	0.728	0.746	0.738	0.667	0.676
21	0.728	0.726	0.725	0.698	0.734	0.710	0.711	0.711	0.751	0.738	0.707	0.702
22	0.745	0.735	0.719	0.698	0.724	0.703	0.719	0.729	0.749	0.741	0.701	0.702
23	0.749	0.739	0.659	0.674	0.716	0.698	0.727	0.720	0.758	0.740	0.717	0.707
24	0.744	0.734	0.728	0.710	0.734	0.713	0.742	0.736	0.743	0.740	0.692	0.687
25	0.745	0.736	0.703	0.685	0.755	0.738	0.698	0.707	0.746	0.732	0.709	0.703
26	0.742	0.738	0.691	0.678	0.745	0.729	0.713	0.709	0.739	0.731	0.701	0.699
27	0.746	0.739	0.663	0.651	0.746	0.730	0.711	0.711	0.734	0.726	0.703	0.694
28	0.736	0.734	0.696	0.694	0.736	0.718	0.742	0.738	0.728	0.716	0.707	0.708
29	0.739	0.738	0.676	0.680	0.747	0.730	0.704	0.700	0.727	0.715	0.714	0.705
30	0.737	0.741	0.687	0.680	0.740	0.743	0.720	0.712	0.722	0.715	0.682	0.681
31	0.745	0.735	0.715	0.690	0.741	0.743	0.720	0.707	0.720	0.724	0.698	0.694
32	0.738	0.736	0.683	0.675	0.742	0.744	0.715	0.718	0.727	0.728	0.735	0.728
33	0.735	0.736	0.676	0.669	0.737	0.744	0.702	0.711	0.736	0.728	0.681	0.678
34	0.741	0.746	0.680	0.688	0.720	0.727	0.720	0.722	0.732	0.726	0.654	0.666
35	0.745	0.747	0.692	0.682	0.718	0.723	0.714	0.720	0.731	0.733	0.664	0.676
36	0.736	0.751	0.687	0.691	0.719	0.728	0.736	0.723	0.728	0.734	0.711	0.704
37	0.739	0.750	0.656	0.654	0.724	0.728	0.698	0.696	0.730	0.731	0.685	0.687
38	0.743	0.752	0.672	0.678	0.725	0.728	0.696	0.708	0.733	0.726	0.722	0.711
39	0.737	0.747	0.681	0.673	0.719	0.727	0.721	0.725	0.728	0.729	0.662	0.669
40	0.739	0.748	0.687	0.680	0.716	0.727	0.744	0.736	0.734	0.728	0.665	0.671
41	0.743	0.749	0.699	0.692	0.726	0.729	0.696	0.705	0.720	0.730	0.695	0.701
42	0.731	0.739	0.667	0.653	0.727	0.729	0.703	0.710	0.714	0.723	0.663	0.669
43	0.721	0.735	0.684	0.688	0.722	0.724	0.753	0.745	0.720	0.729	0.684	0.686
44	0.731	0.729	0.648	0.650	0.716	0.733	0.757	0.746	0.729	0.727	0.689	0.688
45	0.719	0.733	0.686	0.689	0.713	0.729	0.747	0.742	0.714	0.728	0.668	0.671
46	0.738	0.744	0.678	0.667	0.714	0.719	0.736	0.729	0.728	0.728	0.676	0.680
47	0.722	0.728	0.668	0.671	0.723	0.721	0.732	0.724	0.722	0.722	0.667	0.673
48	0.726	0.727	0.681	0.676	0.711	0.726	0.706	0.710	0.735	0.723	0.693	0.692
49	0.730	0.739	0.658	0.659	0.714	0.718	0.698	0.707	0.732	0.727	0.698	0.696
50	0.695	0.708	0.690	0.680	0.714	0.733	0.743	0.733	0.738	0.725	0.697	0.696
51	0.695	0.700	0.725	0.722	0.717	0.730	0.725	0.722	0.733	0.725	0.675	0.679
52	0.689	0.698	0.736	0.733	0.720	0.729	0.736	0.733	0.737	0.726	0.701	0.694
53	0.726	0.723	0.719	0.723	0.715	0.716	0.719	0.723	0.738	0.715	0.699	0.694
54	0.730	0.726	0.726	0.733	0.723	0.722	0.726	0.733	0.724	0.722	0.673	0.672
55	0.718	0.733	0.720	0.717	0.722	0.733	0.720	0.717	0.725	0.718	0.667	0.670
56	0.715	0.721	0.734	0.733	0.720	0.732	0.734	0.733	0.732	0.717	0.665	0.666
57	0.732	0.727	0.731	0.733	0.714	0.720	0.731	0.733	0.725	0.729	0.692	0.688
58	0.746	0.742	0.728	0.725	0.731	0.736	0.728	0.725	0.733	0.731	0.689	0.686
59	0.745	0.741	0.711	0.719	0.725	0.740	0.711	0.719	0.734	0.728	0.674	0.677
60	0.744	0.736	0.682	0.668	0.735	0.740	0.698	0.704	0.737	0.726	0.692	0.685
61	0.746	0.732	0.714	0.716	0.728	0.747	0.714	0.716	0.735	0.712	0.696	0.683
62	0.751	0.729	0.697	0.706	0.737	0.749	0.697	0.706	0.719	0.716	0.683	0.682
63	0.723	0.723	0.701	0.713	0.731	0.741	0.701	0.713	0.733	0.731	0.681	0.683
64	0.722	0.716	0.712	0.712	0.731	0.743	0.712	0.712	0.736	0.748	0.689	0.687
65	0.712	0.719	0.726	0.727	0.719	0.747	0.726	0.727	0.735	0.730	0.686	0.689
66	0.713	0.735	0.715	0.717	0.722	0.750	0.715	0.717	0.711	0.720	0.689	0.683
67	0.739	0.748	0.702	0.709	0.732	0.754	0.702	0.709	0.713	0.722	0.676	0.682
68	0.742	0.747	0.688	0.697	0.722	0.753	0.688	0.697	0.723	0.724	0.708	0.696
69	0.741	0.748	0.716	0.718	0.738	0.753	0.716	0.718	0.718	0.719	0.676	0.683
70	0.741	0.749	0.718	0.720	0.739	0.750	0.718	0.720	0.713	0.719	0.672	0.675

Table A.2.: R^2 values for 2 to 70 regions for WARD and AZP with modeling approaches CN and CN_W for 3 splits

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