Geospatial Analysis on Fuel Prices in the Greek Market

Department of Economics

Τμήμα Οικονομικών Επιστημών
Πανεπιστήμιο Ιωαννίνων

Master Thesis

Bakalis Dimitrios
A.M. 13

Supervisor: Stavrakoudis Athanasios

February 13, 2024

Abstract

This thesis undertook an extensive analysis on a national scale to discern trends and correlations in fuel prices across diverse community types in Greece. This investigation proved crucial for a comprehensive understanding of the Greek market dynamics and the nuanced influences on fuel price fluctuations from neighboring communities, municipalities, islands, and other geographical entities. The analysis spanned nearly a decade, covering the period from 2014 to 2023, necessitating the amalgamation of diverse datasets encompassing fuel-related information, census data, and geospatial details of various community types. Subsequent to the assembly of these datasets, a series of meticulous data preprocessing steps were executed to align the data for both general analysis and time-series clustering. A central aspect of the study involved the application of K-means clustering to different levels of communities, shedding light on the similarity of fuel types among them. The calculation of clusters was facilitated through the implementation of dynamic time warping (DTW) to construct the DTW matrix, a crucial component for time-series clustering. The ensuing presentation of clustering results offered valuable insights into the correlations and resemblances among fuel prices across distinct communities, prefectures, island communities, islands and island complexes.

Abstract in Greek

Στην παρούσα διατριβή διεξήχθη μια εκτενής ανάλυση σε εθνική κλίμακα για τον εντοπισμό των τάσεων και των συσχετίσεων στις τιμές των καυσίμων σε διάφορους τύπους κοινοτήτων στην Ελλάδα. Αυτή η έρευνα αποδείχθηκε ζωτικής σημασίας για την κατανόηση των δυναμικών της ελληνικής αγοράς και των πολύπλοκων επιδράσεών της στις διακυμάνσεις των τιμών των καυσίμων από γειτονικές κοινότητες, νομούς, νησιά και άλλες γεωγραφικές οντότητες. Η ανάλυση κάλυψε σχεδόν δέκα χρόνια, από το 2014 έως το 2023, απαιτώντας τον συνδυασμό διάφορων συνόλων δεδομένων που περιλαμβάνουν πληροφορίες σχετικές με τα καύσιμα, δεδομένα από απογραφές καθώς και γεωχωρικές λεπτομέρειες για διάφορους τύπους κοινοτήτων. Μετά τη συγκέντρωση αυτών των συνόλων δεδομένων, εκτελέστηκε μια σειρά από βήματα προεπεξεργασίας των δεδομένων για την ευθυγράμμιση τους τόσο για γενική ανάλυση όσο και για τη συσταδοποίηση με βάση τις χρονοσειρές. Κεντρικό στοιχείο της μελέτης ήταν η εφαρμογή της συσταδοποίηση K-means σε διάφορα επίπεδα κοινοτήτων, αναδεικνύοντας την ομοιότητα των τύπων καυσίμων μεταξύ τους. Η υπολογιστική διαδικασία των ομάδων υποστηρίχθηκε από την εφαρμογή του dynamic time wraping (DTW) για τον υπολογισμό του πίνακα DTW, κρίσιμο στοιχείο για τη συστηματική ομαδοποίηση χρονοσειρών. Τα απο— τελέσματα της συσταδοποίησης παρουσιάστηκαν τελικά, αποκαλύπτοντας ενδιαφέρουσες πτυχές σχετικά με τις συσχετίσεις και τις ομοιότητες των τιμών των καυσίμων μεταξύ διαφορετικών κοινοτήτων, νομών, νησιωτικών κοινοτήτων, νησιών καθώς και συμπλεγμάτων νησιών.

Acknowledgements

I would like to express my deepest gratitude to my thesis supervisor, Professor Athanasios Stavrakoudis, for his unwavering support and guidance throughout the entire thesis process. Professor Stavrakoudis's expertise, insightful suggestions, and commitment to excellence have been invaluable in shaping and refining this research.

I am also immensely thankful to my friends for their steadfast belief in me. Their encouragement, motivation, and camaraderie have played a pivotal role in overcoming challenges and reaching the completion of this thesis. Your support has been a source of inspiration, and I am grateful for the shared moments of both academic and personal growth.

Furthermore, I extend heartfelt thanks to my life partner for being a constant pillar of support. Your understanding, encouragement, and patience during the highs and lows of the thesis journey have been immeasurable. Your unwavering belief in my abilities has been a driving force, and I am grateful for your presence on this transformative academic endeavor.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Overview

The escalating complexity of fuel markets demands a comprehensive understanding of the dynamics influencing price fluctuations, particularly in diverse geographical and community contexts. This thesis addresses this imperative by conducting an extensive national-scale analysis focused on the Greek fuel market. Over nearly a decade, spanning from 2014 to 2023, our investigation delves into the intricate relationships between fuel prices and diverse community types, including communities, islands, and other geographical entities. This research aims to unravel the nuanced influences on fuel prices, emphasizing the role of neighboring communities in shaping market trends.

The significance of this study lies in its commitment to providing a holistic perspective on Greek market dynamics. By amalgamating diverse datasets encompassing fuel-related information, census data, and geospatial details of various community types, we aim to uncover trends that transcend temporal and spatial boundaries. Through meticulous data preprocessing steps, we align these datasets for both general analysis and time-series clustering. The utilization of advanced techniques such as K-means clustering and dynamic time warping (DTW) adds depth to our investigation, allowing for a nuanced exploration of fuel price correlations among different community types.

1.2 Thesis Organization

The thesis follows a structured organization, beginning with Chapter 1's introduction, which expounds on the primary focus, delineates the problem definition, and highlights the study's contributions.

Chapter 2 engages in a comprehensive literature review on methodologies related to time series analysis, statistical testing as well as time series clustering.

Chapter 3 provides an overview of available data concerning gas stations and geospatial information for all Greek communities.

Moving forward, Chapter 4 intricately describes the methodology, encompassing data preprocessing and the application of time series clustering. This chapter elucidates the steps taken to prepare the data for analysis. Simultaneously, it introduces and justifies the application of time series clustering techniques employed in the subsequent analysis.

Following this, Chapter 5 meticulously presents the outcomes of clustering experiments alongside various insightful statistics.

Chapter 6 serves as the conclusion, summarizing findings and offering a conclusive perspective on the overall study.

Lastly, the Appendix contains additional Figures for the reader's better understanding of the clustering results, the statistical testing as well as the Python scripts utilized in all experiments, ensuring complete transparency in presenting the study's outcomes.

1.3 Problem Definition

The fuel market, with its intricate web of influences and fluctuations, demands a thorough investigation to comprehend the dynamics shaping price trends. This thesis addresses this imperative with a focused exploration of the Greek fuel market, spanning the period from 2014 to 2023. The primary challenge at hand is to decipher the multifaceted relationships between fuel prices and diverse community types across Greece, including communities, islands, and various geographical entities. The pressing need for such an analysis arises from the inherent complexities that these different entities introduce into the market dynamics.

The crux of the matter lies in understanding how neighboring communities, island communities, islands, and other geographical entities contribute to the nuanced influences on fuel price fluctuations. This intricacy necessitates an extensive national-scale analysis, considering the diverse nature of these communities and their potential impact on market trends. The overarching goal is to offer a comprehensive view of the Greek fuel market dynamics, unraveling the hidden patterns that may be obscured by temporal and spatial variations.

1.4 Contribution

This thesis makes several key contributions to the understanding of fuel markets, specifically within the Greek context. Firstly, it provides a comprehensive analysis of fuel price trends over a significant period, offering valuable insights into the factors influencing market dynamics. The inclusion of diverse datasets, spanning fuel-related information, census data, and geospatial details, enables a multifaceted examination of the interplay between fuel prices and community characteristics.

The application of K-means clustering to different levels of communities enhances our understanding of the similarities in fuel types among them. This clustering analysis, supported by dynamic time warping (DTW), introduces a novel approach to investigating time-series data, enabling a more accurate representation of temporal relationships. The resulting clusters not only shed light on the diversity among community types but also reveal hidden patterns in fuel price fluctuations, thereby contributing to the broader understanding of market dynamics.

Furthermore, this study contributes to the methodological landscape by showcasing the effectiveness of dynamic time warping in constructing the DTW matrix for time-series clustering. The integration of this technique demonstrates its utility in revealing meaningful correlations and resemblances among fuel prices across distinct communities, island communities, and island complexes.

Overall, this research serves as a valuable resource for policymakers, industry stakeholders, and researchers seeking a deeper comprehension of fuel market dynamics in Greece.

Chapter 2

Literature Review

2.1 Times Series Clustering

This section focuses on the utilization of clustering algorithms for time series data analysis. Clustering serves as a fundamental data mining and machine learning technique, facilitating the grouping of similar data points into coherent clusters to enable exploratory analysis and pattern recognition. With the increasing significance of temporal data analysis, the application of clustering methodologies tailored to time series has garnered considerable attention. Notably, k-means and hierarchical clustering are two widely adopted approaches for partitioning time-dependent sequences into meaningful groups, thereby unveiling underlying patterns and valuable insights. Throughout this section, we shall extensively investigate the intricacies of k-means, hierarchical clustering, and other pertinent techniques, elucidating their potentials and limitations in diverse time series datasets. A comprehensive understanding of their distinctive characteristics and challenges will enable the effective extraction of crucial knowledge from time series data across a wide range of domains, including finance, healthcare, and environmental monitoring, contributing to the advancement of scientific knowledge in this field.

Clustering, as a data mining technique, involves organizing similar data into related or homogeneous groups without prior knowledge of the groups' definitions [1]. This process entails forming clusters by grouping objects that exhibit maximum similarity within the group while minimizing similarity with objects in other groups. Clustering is valuable for exploratory data analysis, objectively revealing structures in unlabelled datasets by organizing them into similar groups. Furthermore, it serves as a pre-processing step for other data mining tasks and plays a crucial role in complex systems.

With the increasing capabilities of data storage and processors, real-world applications have the ability to retain data for extended periods. As a result, many applications store data in the form of time-series data, encompassing sales data, stock prices, exchange rates in finance, weather data, biomedical measurements (e.g., blood pressure and electrocardiogram readings), biometrics data (facial recognition image data), particle tracking in physics, and more. Various domains, including Bioinformatics, Biology, Genetics, Multimedia [2, 3, 4], and Finance, have witnessed the utilization of time-series data. Consequently, an abundance of time-series data has opened new opportunities for researchers in data mining communities over the last decade.

As a consequence, numerous research projects have emerged in diverse areas to analyze time-series data for various purposes. These purposes include subsequence matching, anomaly detection, motif discovery [5], indexing, clustering, classification [6], visualization [7], segmentation [8], pattern identification, trend analysis, summarization [9], and forecasting. Additionally, ongoing research projects are dedicated to enhancing existing techniques [10, 11].

Time-series clustering is a distinct form of clustering that deals with temporal sequences. A sequence containing a series of nominal symbols from a specific alphabet is commonly referred to as a temporal sequence, while a sequence comprising continuous, real-valued elements is termed a time-series [12]. Time-series are considered dynamic data since their feature values change over time, with each point of the series representing one or more chronologically arranged observations. Due to their temporal nature, time-series data are inherently high-dimensional and often large in size [6, 13, 14]. These data are prevalent across various domains, including science, engineering, business, finance, economics, healthcare, and government, making them of significant interest [15]. Despite each time-series comprising numerous data points, it can also be regarded as a single object [16]. Clustering such intricate objects holds particular advantages as it reveals compelling patterns in time-series datasets, including both frequent and rare patterns. This poses various research challenges, such as devising methods to identify dynamic changes in time-series, anomaly and intrusion detection, process control, and character recognition [17, 18, 19].

After reviewing the literature, it becomes evident that the majority

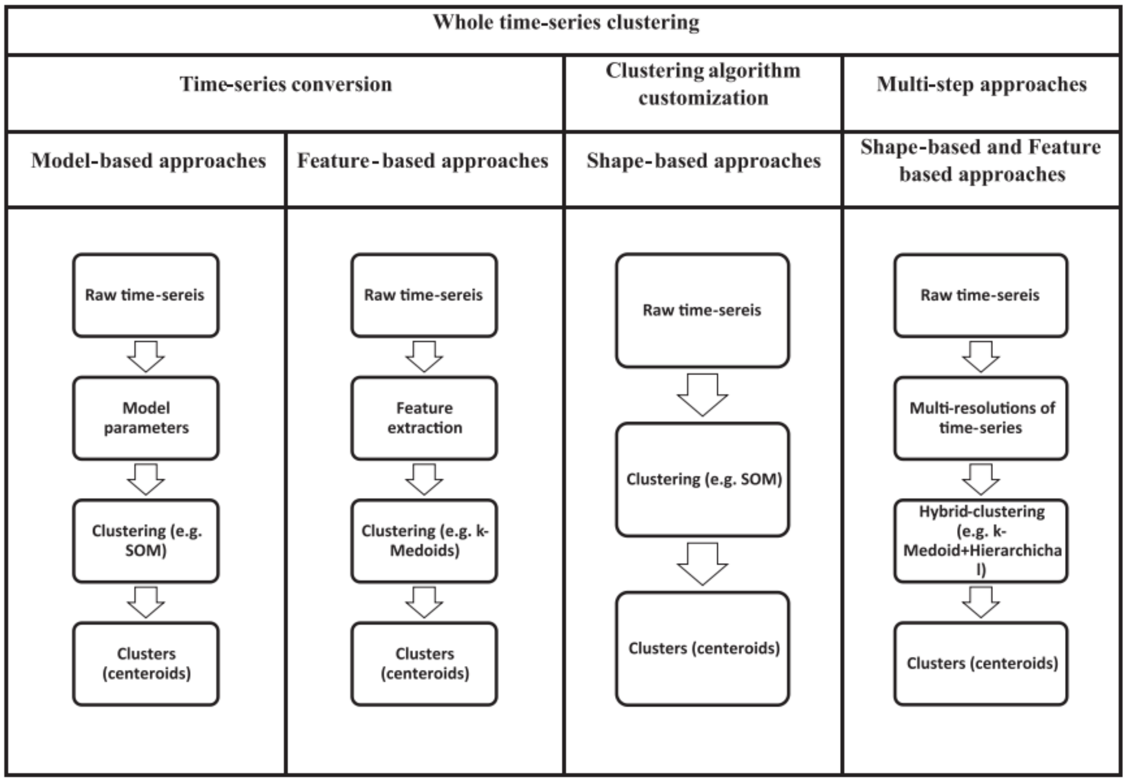


Figure 2.1: The time-series clustering approaches.

of clustering works related to time-series can be categorized into three main groups: *whole time-series clustering*, *subsequence clustering* and *time point clustering*. Whole time-series clustering involves clustering a set of individual time-series based on their similarity. In this context, clustering refers to applying conventional (usually) clustering algorithms to discrete objects, where the objects are time-series, as seen in Figure 2.1. The goal is to group similar time-series together, allowing for the identification of patterns and relationships within the dataset.

2.1.1 K-means

K-means clustering is a vector quantization method, originally used in signal processing, aimed at partitioning n observations into k clusters. Each observation is assigned to the cluster with the nearest mean, serving as a prototype of that cluster. Consequently, the data space is divided into Voronoi cells through this partitioning. K-means clustering minimizes within-cluster variances (squared Euclidean distances) to achieve its objective, while regular Euclidean distances pose a more challenging Weber problem. In this context, the mean optimizes squared errors, while only the geometric median minimizes Euclidean distances. Alternatively, k-medians and k-medoids offer improved Euclidean solutions.

Solving the k-means clustering problem is computationally difficult, falling under NP-hard complexity. However, efficient heuristic algorithms rapidly converge to a local optimum. These algorithms often employ an iterative refinement approach, similar to the expectation-maximization algorithm used in Gaussian mixture modeling for mixtures of Gaussian distributions. Both k-means and Gaussian mixture modeling utilize cluster centers to model the data, though k-means clustering tends to produce clusters with comparable spatial extent, while the Gaussian mixture model allows clusters to have different shapes.

It is essential to distinguish the unsupervised k-means algorithm from the k-nearest neighbor classifier, a popular supervised machine learning technique for classification. They share similar names, leading to confusion. However, applying the 1-nearest neighbor classifier to the cluster centers obtained by k-means enables the classification of new data into existing clusters. This approach is known as the nearest centroid classifier or the Rocchio algorithm.

2.1.2 Hierarchical Clustering

Hierarchical clustering is a widely used method for grouping objects, wherein clusters are formed in a manner that ensures objects within a cluster exhibit similarity while being different from objects in other clusters. These clusters are visually represented in a hierarchical tree known as a dendrogram.

One of the key benefits of hierarchical clustering is that it does not require pre-specification of the number of clusters. Instead, by cutting the dendrogram at an appropriate level, the desired number of clusters can be obtained. This flexibility allows for an adaptive approach to clustering, enabling a deeper understanding of the underlying data structure.

Additionally, hierarchical clustering facilitates data summarization and organization through dendrograms, providing an intuitive means to examine and interpret the resulting clusters. This hierarchical representation allows for a comprehensive view of the data, revealing relationships and hierarchical structures present in the dataset.

The real-life applications of hierarchical clustering are diverse and widespread across various domains. In bioinformatics, hierarchical clustering is employed to group animals based on their biological features, aiding in the reconstruction of phylogeny trees. In the business realm, this method is used to segment customers or create hierarchies of employees based on their salary levels, leading to targeted marketing strategies and improved organizational structures. Image processing benefits from hierarchical clustering in grouping handwritten characters in text recognition, utilizing the similarity of character shapes for accurate classification. Moreover, in information retrieval, hierarchical clustering helps categorize search results based on the relevance to user queries, enhancing the efficiency and effectiveness of information retrieval systems.

Hierarchical clustering encompasses two main types: agglomerative and divisive. The agglomerative approach starts with each object as an individual cluster and progressively merges clusters until a single cluster remains, ultimately forming a cluster that contains all elements. On the other hand, the divisive method begins with all objects in a single cluster and then proceeds to divide or split clusters step by step until each object forms a separate cluster. The division process is governed by principles that maximize the distance between neighboring objects in the cluster.

Between agglomerative and divisive clustering, the agglomerative method is generally preferred due to its efficiency and its ability to produce more balanced clusters. The agglomerative approach naturally complements the hierarchical structure and facilitates easier interpretation of the resulting clusters.

In conclusion, hierarchical clustering is a powerful technique that finds application in a wide range of fields, aiding in the discovery of meaningful patterns and structures in datasets. Its flexibility, adaptability, and comprehensibility make it a valuable tool for exploratory data analysis and pattern recognition in various domains.

2.2 Dynamic Time Warping

Dynamic Time Warping (DTW) has emerged as a prominent technique across diverse fields, such as signal processing, time series analysis, and pattern recognition. One of the foundational publications in the field is the work of Sakoe and Chiba [20], who introduced the concept of DTW for spoken word recognition. This seminal paper laid the groundwork for using DTW to align and compare time-varying sequences efficiently.

A comprehensive overview of speech recognition techniques, including DTW, can be found in the book by Rabiner and Juang [21]. This source provides valuable insights into the application of DTW in the context of speech processing.

Efficiency concerns regarding DTW are addressed by Keogh and Ratanamahatana [22], who proposed an indexing method for faster and more scalable time series matching. This contribution is crucial for enhancing the computational aspects of DTW, making it more applicable to large datasets.

Salvador and Chan [23] focused on optimizing the time and space complexity of DTW, making it suitable for real-time applications. Their work significantly reduces the computational cost while maintaining accuracy, further enhancing the practical utility of DTW.

A thought-provoking perspective on DTW is presented by Ratanamahatana and Keogh [24] in their paper titled "Everything you know about dynamic time warping is wrong." This work highlights common misconceptions and provides insights into the correct usage and interpretation of the algorithm, contributing to a more nuanced understanding of DTW.

Vlachos, Kollios, and Gunopulos [25] extend the application of DTW beyond one-dimensional sequences by introducing an approach to discovering similar trajectories in multidimensional space. This demonstrates the versatility of DTW in various domains beyond traditional signal processing.

Müller's book [26] on "Information Retrieval for Music and Motion" delves into the application of DTW in music similarity analysis, providing an extensive exploration of content-based retrieval techniques for music and motion data.

Berndt and Clifford [27] discuss the application of DTW in finding patterns within time series data, showcasing its utility in identifying meaningful structures in temporal sequences.

The application of DTW extends to bioinformatics, as seen in the work of Aach and Church [28], who apply DTW to align gene expression time series. This exemplifies the adaptability of DTW in handling diverse data types.

Chen, Özsu, and Oria [29] propose an approach for similarity search in moving object trajectories using DTW. This work demonstrates the relevance of DTW in handling dynamic and evolving data, further expanding its applicability.

In summary, the literature on DTW is rich and varied, encompassing foundational works in speech recognition, optimization techniques, multidimensional trajectory analysis, music similarity, and applications in bioinformatics and dynamic data sets.

2.3 Unit Root Testing

Unit root testing, particularly using the Augmented Dickey-Fuller (ADF) test, is a fundamental aspect of time series analysis with significant implications for modeling and forecasting in various fields such as economics, finance, and environmental science. The ADF test, introduced by Dickey and Fuller [30], has become a cornerstone method for assessing the stationarity properties of time series data.

Dickey and Fuller's pioneering work laid the theoretical groundwork for the ADF test, presenting the concept of unit roots in autoregressive time series models and deriving the distributional properties of the estimators under the null hypothesis of a unit root. This seminal paper established the framework for unit root testing and its relevance in analyzing non-stationary data.

Building upon the foundational work of Dickey and Fuller, subsequent studies have extended and refined the ADF test, addressing various practical challenges and theoretical considerations. Said and Dickey [31] proposed modifications to the ADF test to accommodate autoregressive-moving average (ARMA) models of unknown order, enhancing its applicability to real-world datasets with complex dynamics.

Kwiatkowski et al. [32] introduced the KPSS test as an alternative to the ADF test for testing stationarity, providing complementary insights into the behavior of time series data. This comparative analysis between the ADF and KPSS tests has contributed to a deeper understanding of unit root testing methodologies and their relative strengths in different contexts.

Further advancements in unit root testing have focused on improving the efficiency and robustness of the ADF test. Elliott, Rothenberg, and Stock [33] proposed modifications to address serial correlation and heteroscedasticity in the data, introducing the GLS-detrended ADF test as a more efficient alternative.

In addition to methodological developments, researchers have also explored the empirical applications of unit root testing across diverse fields. Studies have examined the stationarity properties of economic indicators, financial time series, and environmental variables, shedding light on the long-term behavior and trends in these datasets.

In conclusion, unit root testing with the Augmented Dickey-Fuller test continues to be a fundamental tool in time series analysis, facilitating the identification of stationary and non-stationary processes in empirical data. Ongoing research efforts aim to refine existing methodologies, expand the empirical applications, and deepen our theoretical understanding of unit root testing in diverse contexts.

2.4 Cointegration Testing

Cointegration, introduced by Clive Granger in 1987, has become a fundamental concept in time series analysis, particularly in the context of testing for long-run relationships among economic variables. Cointegration arises when two or more non-stationary time series share a common stochastic trend, implying a stable long-term relationship despite short-term fluctuations.

One of the seminal contributions to the field is by J.G. MacKinnon, whose work has significantly advanced our understanding of cointegration tests. In the paper [34], MacKinnon presented approximate asymptotic distribution functions for unit-root and cointegration tests. The paper offers valuable insights into the statistical properties of these tests, providing researchers with tools to assess the reliability of their findings.

MacKinnon's continued contributions are evident in his 2010 paper [35]. In this work, MacKinnon focused on providing critical values for cointegration tests, crucial for determining the significance of the test statistics. The availability of accurate critical values is essential for researchers and practitioners alike, as it ensures the robustness and validity of cointegration tests in empirical studies.

These two papers by MacKinnon collectively form a cornerstone in the cointegration literature, offering researchers a solid foundation for conducting cointegration tests and interpreting their results. The rigorous statistical methodologies presented in these papers have become standard references in the field, guiding researchers in assessing the presence of long-term relationships among economic variables.

In conclusion, the works of J.G. MacKinnon have significantly enriched the field of cointegration testing, providing researchers with essential tools and critical values for robust empirical analyses. The methodologies and insights presented in these papers continue to influence and shape the way economists and statisticians approach cointegration testing in various applications.

Chapter 3

Data

3.1 Overview

This study involved the utilization of two distinct data types. The first category comprised geospatial data, encompassing information pertaining to local, municipal, and pseudo-municipal communities within Greece. The second category encompassed time-series data capturing the prices of various fuels retailed by gas stations nationwide.

This chapter aims to elucidate the inherent characteristics of the employed data, elucidate the methodologies employed in merging and aligning datasets, and address several challenges encountered in working with the data, along with the corresponding solutions devised to overcome them.

3.2 Communities

3.2.1 Community Codes

As previously mentioned, our research on a national scale necessitated the utilization of available community information. To initiate our investigation, we acquired publicly accessible census data from the Hellenic Statistical Authority for the years 2001, 2011, and 2021.

Upon commencing the data analysis, a primary challenge arose in the initial stages, requiring the matching of unique codes for each community across the three datasets. Despite minor fluctuations in data across census years, certain communities amalgamated, resulting in modifications to their distinct codes. To address this issue, we conducted a manual matching process for each community across the three datasets, utilizing both the original community name and the associated municipality name.

This methodology enabled us to establish a historical profile for each municipality throughout the three census years, encompassing information such as population size, community name, community code and municipality name. An illustrative example of the matched data is presented in Table 3.1.

Table 3.1: Matching of Data Between 2001, 2011 and 2021 of a Random Community.

Year	Community Name	Community Code	Municipality Name	Population
2001	Δ.Δ. Αγγελοκάστρου	01020100	ΔΗΜΟΣ ΑΓΓΕΛΟΚΑΣΤΡΟΥ	—
2011	Τοπική Κοινότητα Αγγελοκάστρου	38030201	Δήμος Αγρινίου	1297
2021	Δημοτική Κοινότητα Αγγελοκάστρου	38020201	ΔΗΜΟΣ ΑΓΡΙΝΙΟΥ	1099

Furthermore, in an effort to enhance the reader's understanding of the datasets, we present various statistics concerning the data from each census year in Table 3.2.

Table 3.2: Census Data Depicting Years: 2001, 2011 and 2021.

Year	Num. of Communities	Num. of Municipalities	Total Population
2001	6130	1003	—
2011	6155	325	11.166.922
2021	6132	332	10.482.487

3.2.2 Community Neighbors

A pivotal aspect of this study involved the implementation of time-series clustering. To execute this task, it was imperative to establish a grouping of communities, enabling the algorithm to compute the clusters. By "group of communities", we refer to all the physically adjacent neighbors of each community. To achieve this, we utilized a shapefile containing information about all available communities in Greece, once again sourced from the Hellenic Statistical Authority.

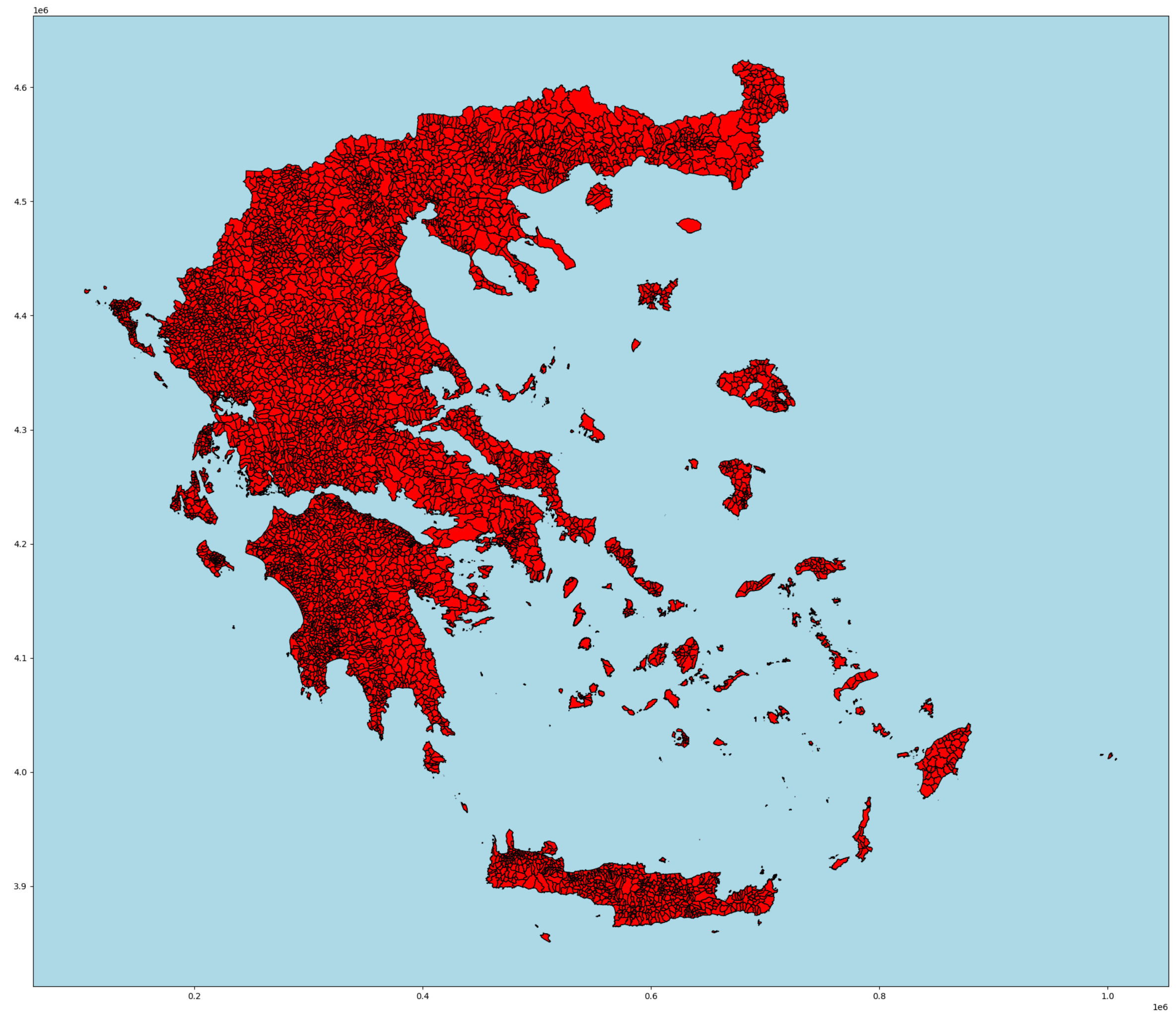


Figure 3.1: 2021 map of Greece's 6132 communities.

In this particular shapefile, the representation of each community's shape is depicted as a polygon on a map, reflecting the latest recorded year of 2021. Consequently, it was necessary to align all information pertaining to each community with the unique code from the processed dataset mentioned in subsection 3.2.1. In Figure 3.1, we can see all 6132 communities of Greece with their borders plotted on a map.

In order to find all neighbors per community we developed an algorithm that identifies neighboring features within the geospatial dataset.

It begins by calculating the bounding box of a given spatial geometry and utilizes a spatial index to identify potential neighbors with intersecting bounding boxes. Subsequently, a subset of potential matches is retrieved from the original dataset, and this set is further refined by selecting features that spatially intersect with the input geometry. The algorithm identifies precise matches among the potential communities based on spatial intersections and extracts all relevant neighbors. The final output is presented as a list of identifiers associated with the neighboring community codes, demonstrating a systematic approach for spatial proximity analysis.

To showcase the capabilities of the proposed methodology, we project a random community in Figure 3.2 with its adjacent neighbors, as outputted by the algorithm.

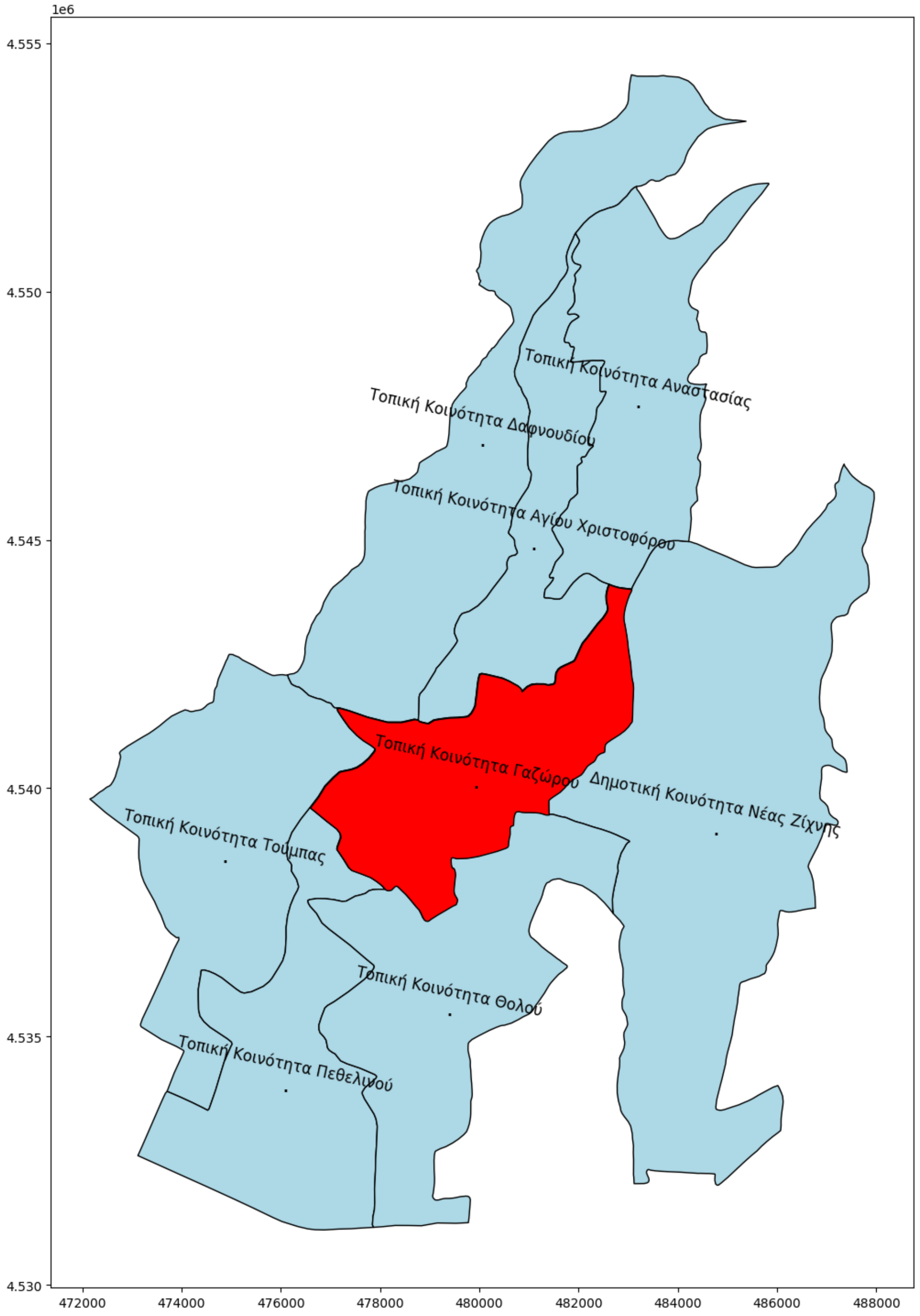


Figure 3.2: Neighbors of ”Τοπική Κοινότητα Γαζώρου” community.

3.2.3 Island Communities

This study included an investigation into the potential correlations in fuel prices among various islands and island complexes. To achieve this, it was imperative to compile comprehensive geospatial data pertaining to extant island communities. Additionally, the identification of all individual islands and island complexes in Greece was a prerequisite for conducting the analysis.

Individual Islands

To ascertain the communities within a singular island, a meticulous process was undertaken involving the manual cross-referencing of each individual community code. This cross-referencing utilized two distinct shapefiles supplied by the Hellenic Statistical Authority. Consequently, a total of 1470 island communities and 65 islands were successfully identified. Part of the outcome of this identification process is visually represented in Figure 3.3.

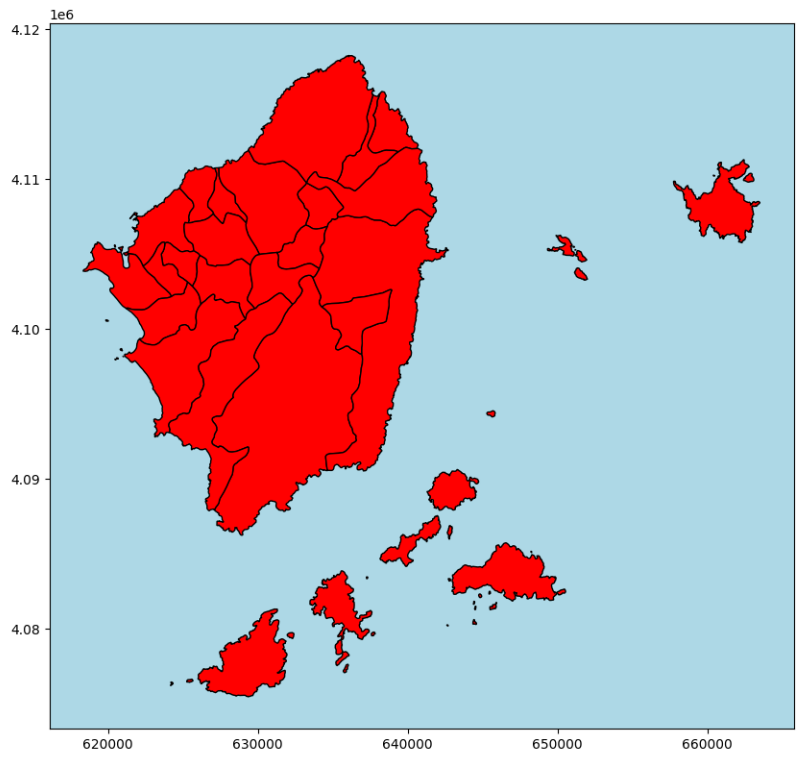


Figure 3.3: Communities of ”Νάξος” island.

Complex of Islands

Upon successfully determining all island communities, the subsequent task involved the manual categorization of each island into its corresponding island complex. Greece comprises six distinct island complexes, namely: "ΚΥΚΛΑΔΕΣ," "ΣΠΟΡΑΔΕΣ," "ΔΩΔΕΚΑΝΗΣΑ," "ΕΠΤΑΝΗΣΑ," "ΑΡΓΟΣΑΡΩΝΙΚΟΣ," and "ΑΝΑΤΟΛΙΚΟ ΑΙΓΑΙΟ."

Through the integration of information from the aforementioned shapefiles, all six island complexes were successfully identified along with their respective constituent islands. The outcome of this complex identification process is visually presented in Figures 3.4 and 3.5.

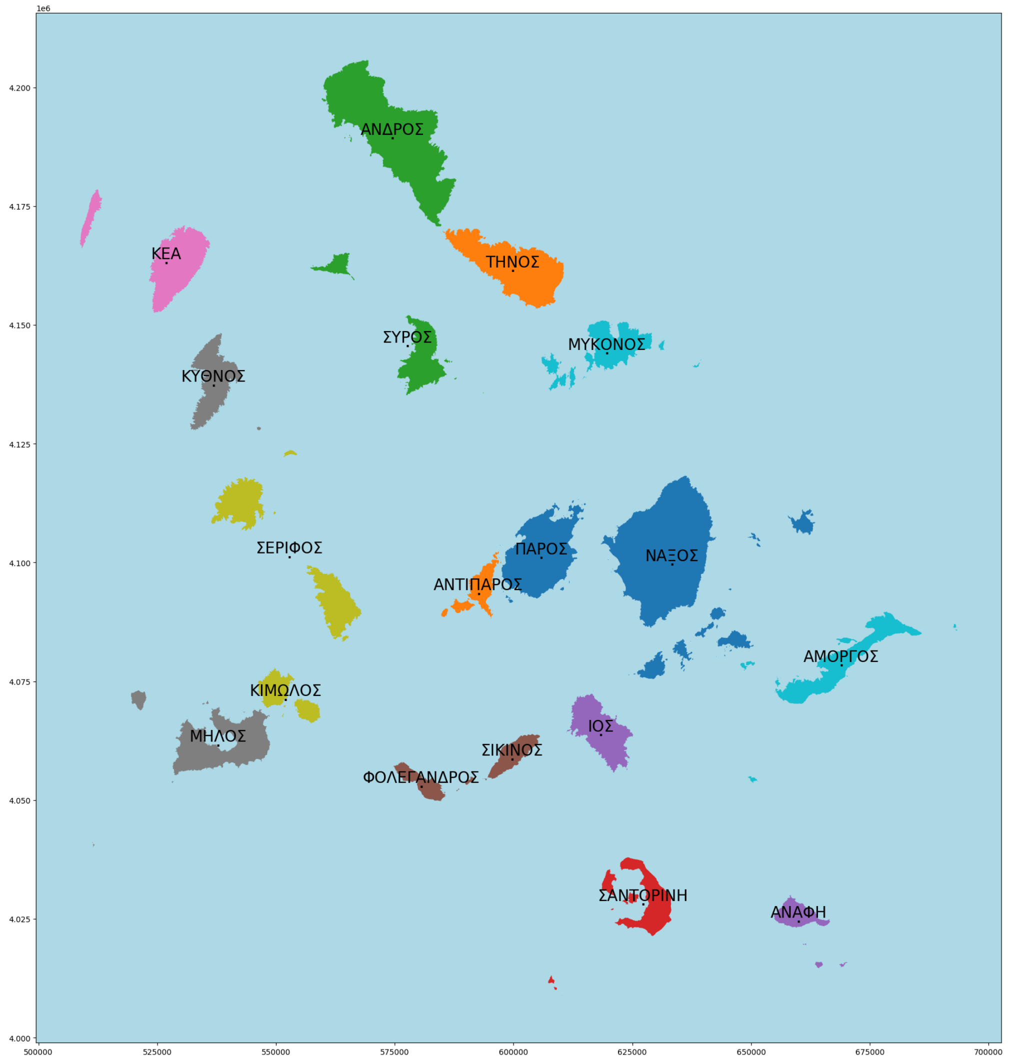


Figure 3.4: Island complex of Greece, named "Κυκλάδες".

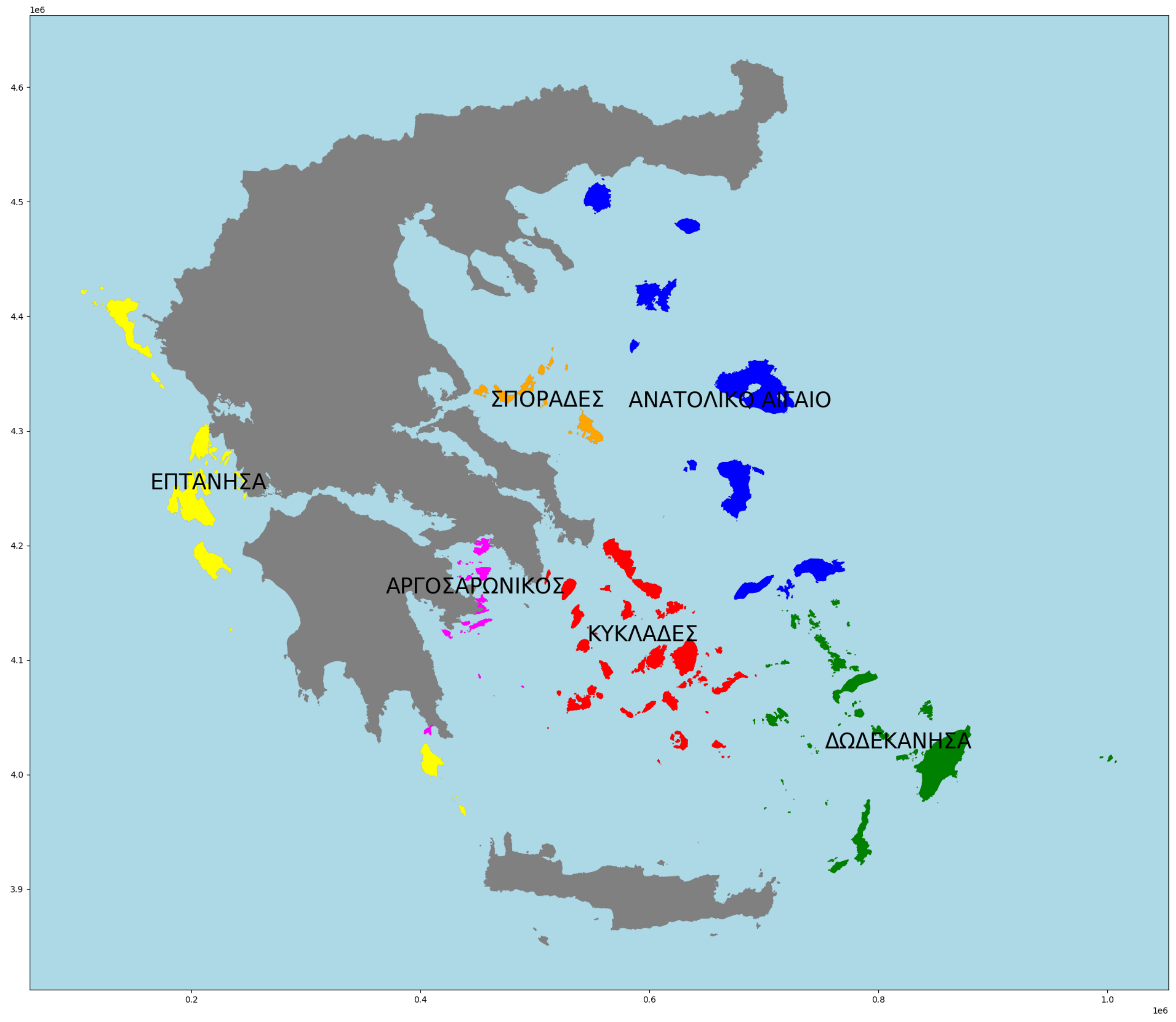


Figure 3.5: Island complexes of Greece.

3.2.4 Prefectures

This study encompassed an examination of potential correlations in fuel prices among various prefectures. To accomplish this, it was crucial to gather extensive geospatial data related to existing prefecture communities. Moreover, the identification of all individual prefectures in Greece was a prerequisite for conducting the analysis.

To identify the communities within a specific prefecture, a rigorous process was initiated, involving the careful cross-referencing of each individual community code. This cross-referencing utilized two distinct shapefiles provided by the Hellenic Statistical Authority. As a result, a total of 1842 communities and 39 prefectures were successfully identified. A visual representation of part of the outcome of this identification process is presented in Figure 3.6.

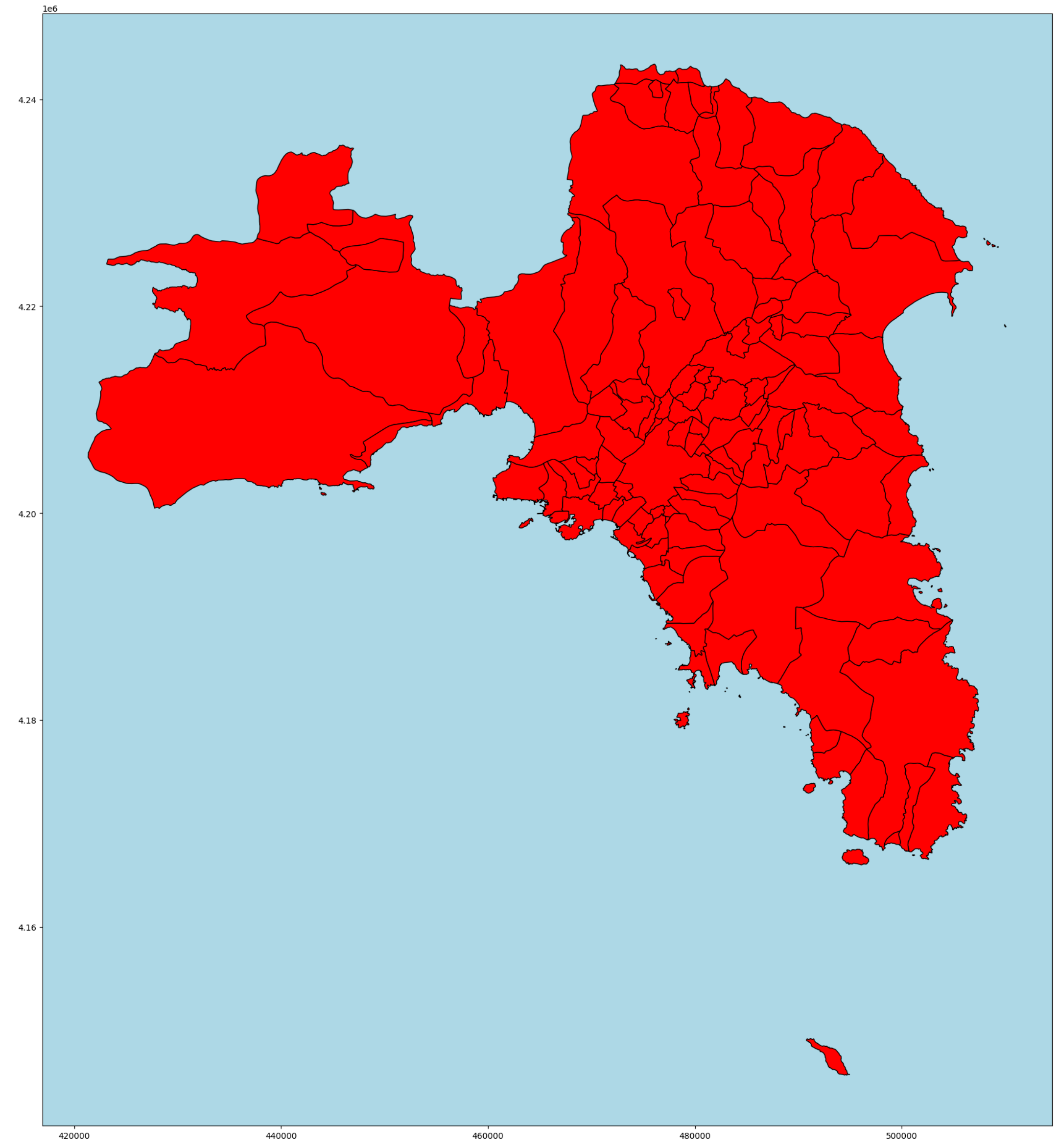


Figure 3.6: Communities of "Νομός Αττικής" prefecture.

3.3 Gas Stations

An integral facet of this study involved the analysis of fuel prices across diverse communities. To facilitate this investigation, we acquired a dataset encompassing information on various fuel types and their measurements spanning from 2014 to late 2023. The focus of the study centered on three specific fuel types: Unleaded 95, Unleaded 98/100, and Diesel. The dataset comprised details from 8051 gas stations, with each station being assigned a unique community code. Notably, these community codes were derived from the 2001 census data, necessitating a meticulous matching process conducted in the preceding Section 3.2 to align the gas station community codes from 2001 to 2011 with those of 2021.

Following the successful matching of each gas station to its corresponding 2021 community code, three distinct datasets were obtained, each specific to the fuel type under consideration in this study. The results of this process are comprehensively presented in Table 3.3.

Furthermore, to enhance comprehension of the matching procedure, Figures 3.7, A.1, and A.2 depict the geographical distribution of all communities containing gas stations that retail the fuel types "Unleaded 95," "Unleaded 98/100," and "Diesel," respectively, across the Greek map.

Lastly, the dataset presented its own set of challenges, primarily stemming from the absence of consistent fuel measurements. Factors such as gas station closures, divergent measurement frequencies among stations, and the pronounced impact of the pandemic (COVID-19) on both prices and measurement frequency contributed to this irregularity. In response to these challenges, the subsequent chapter outlines the devised solutions implemented to address those issues.

Table 3.3: Statistics About Different Fuel Types

Fuel Type	Num. of Gas Stations	Num. of Communities
Unleaded 95	7755	1842
Unleaded 98/100	4934	1090
Diesel	7751	1841

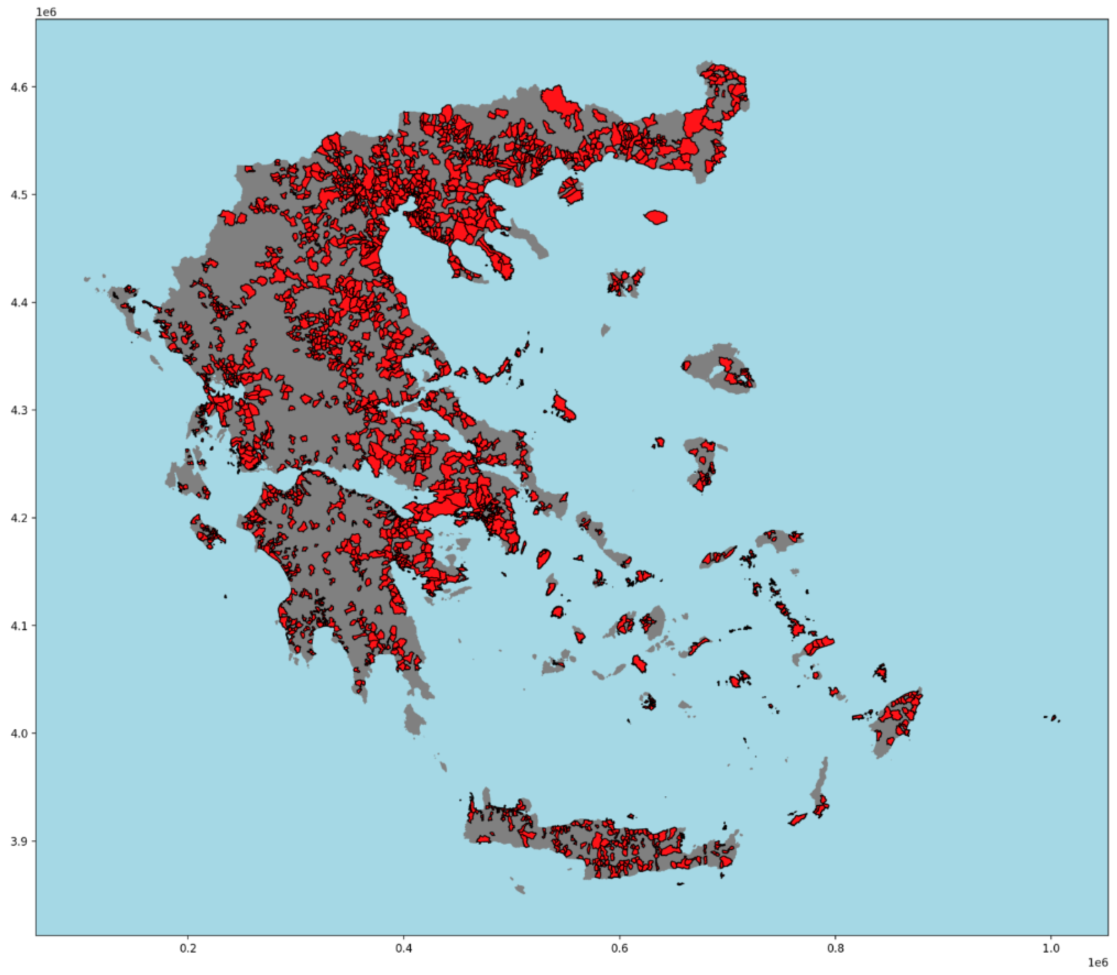


Figure 3.7: Map of Greece, including 1842 "Unleaded 95" communities.

Chapter 4

Methodology

4.1 Overview

The objective of this study was to conduct time series clustering to examine the impact of fuel prices within communities and their neighboring counterparts. To achieve this, the initial task involved generating fuel time series for each available community, categorized by fuel type.

Given the disparate frequencies in the data, the first procedural step entailed creating time series with a daily frequency while imputing missing values. Subsequently, the consolidation of information across multiple time series for each community occurred, considering the presence of multiple gas stations per community and various fuel types.

Upon establishing a singular time series per community, comprising the average daily fuel prices aggregated from all available gas stations, the clustering algorithm was applied to each community and its adjacent neighbors. The time series data underwent normalization through the "Time Series Scaler Mean Variance" Scaler filter to expedite and fortify the clustering process.

The K-means algorithm, employing the DTW filter for time series clustering, was utilized. The outcomes of the clustering process were stored and visually represented on a dynamic map, illustrating communities within their respective clusters, the average prices of each community per fuel type, and the average prices within each cluster. Subsequent sections elaborate on the detailed steps of the applied methodology.

4.2 Data Preprocessing

This section delineates the essential data preprocessing steps undertaken for each community, ensuring the generation of complete time series with a daily frequency of measurements, a prerequisite for subsequent clustering procedures.

The initial phase of data imputation involved calculating the average daily price of the relevant fuel type from all available gas stations within each community. Following this, the second step entailed imputing missing dates and fuel prices with the last observed value. Gas stations abstaining from reporting any fuel prices for a consecutive period of 30 days were classified as "closed" for that month, thereby exempted from both the average pricing calculation and the imputation process.

The steps of the procedure are visually presented, for a random community, in Figures 4.1-4.4.

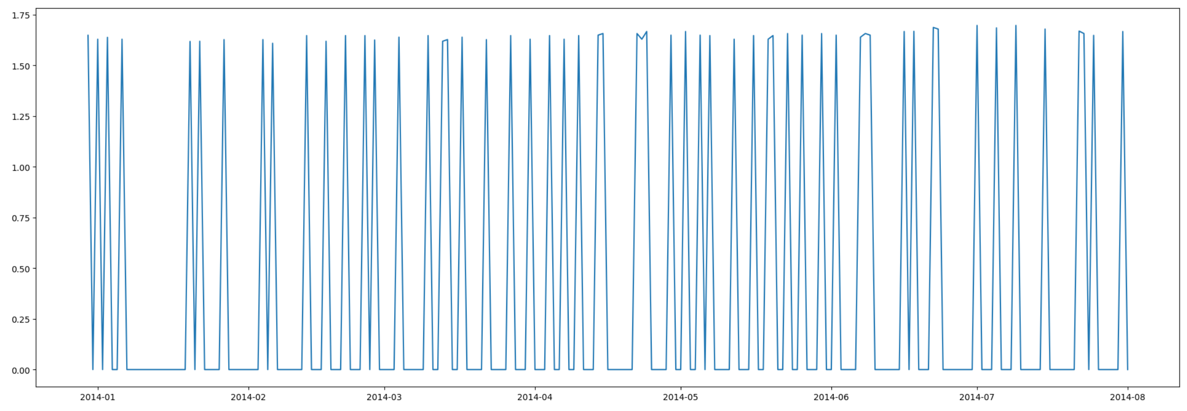


Figure 4.1: One Week of "Unleaded 95" fuel prices, including missing Values (0), for "Τοπική Κοινότητα Γαζώρου' community.

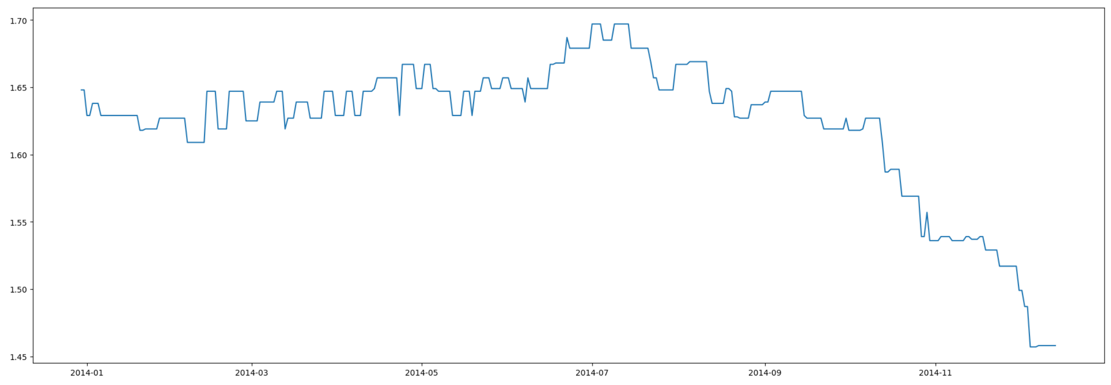


Figure 4.2: One week of "Unleaded 95" fuel prices with time imputation, for "Τοπική Κοινότητα Γαζώρου' community.

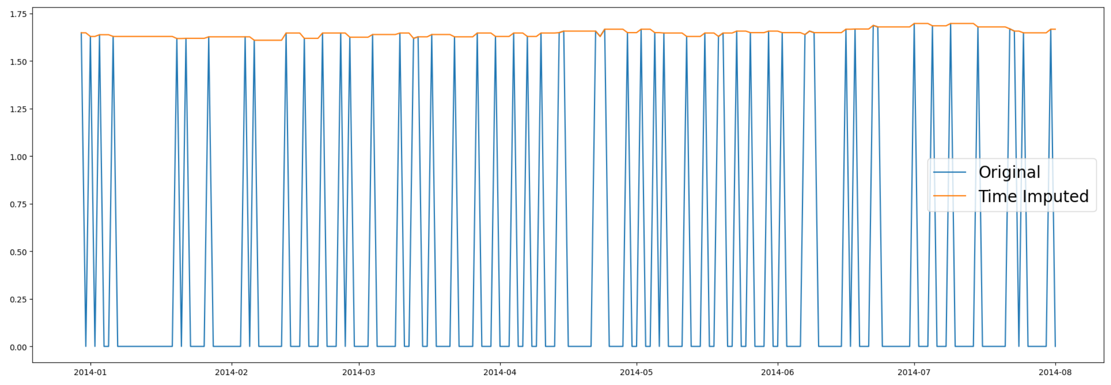


Figure 4.3: One week of "Unleaded 95" fuel prices, for "Τοπική Κοινότητα Γαζώρου' community.

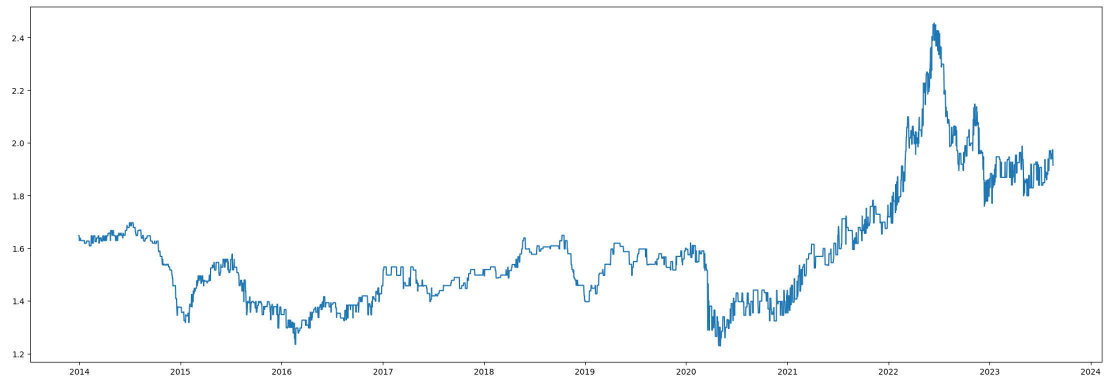


Figure 4.4: Final "Unleaded 95" time series for "Τοπική Κοινότητα Γαζώρου' community.

4.3 Clustering

In this study, the objective was to cluster time series data representing fuel prices for ”Unleaded 95”, ”Unleaded 98/100”, and ”Diesel” across communities and their neighbors from 2014-2023. The methodology involved data collection from various gas stations within each community, computation of the average fuel price per day for each community, identification of adjacent neighbors for each community, and calculation of time series data for fuel prices in these neighboring areas.

The clustering approach employs Dynamic Time Warping (DTW) as a similarity measure for time series data, as shown in Figure 4.5. The process involves the calculation of a distance matrix between all pairs of time series using DTW, as showin in Figure 4.6 , followed by the application of K-means clustering, to group time series into clusters based on their DTW distances. The iterative updating of cluster centers minimizes the total within-cluster distance, assigning each time series to the cluster with the nearest center, as shown in Figure 4.7. The clusters are then visualized and analyzed to interpret fuel price trends.

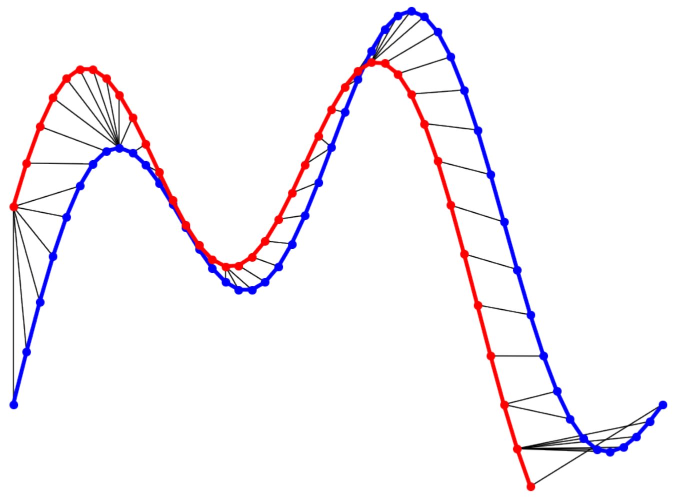
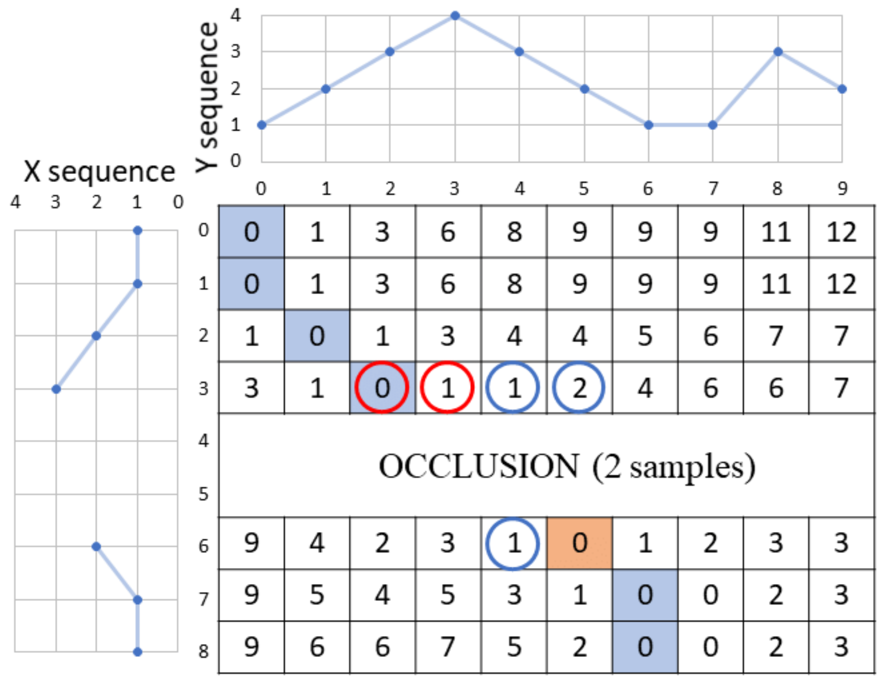


Figure 4.5: Dynamic time warping on times-series data.



	0	1	2	3	4	5	6	7	8	9
0	0	1	3	6	8	9	9	9	11	12
1	0	1	3	6	8	9	9	9	11	12
2	1	0	1	3	4	4	5	6	7	7
3	3	1	0	1	1	2	4	6	6	7
4	OCCLUSION (2 samples)									
5										
6	9	4	2	3	1	0	1	2	3	3
7	9	5	4	5	3	1	0	0	2	3
8	9	6	6	7	5	2	0	0	2	3

Figure 4.6: Dynamic time warping matrix with occlusion.

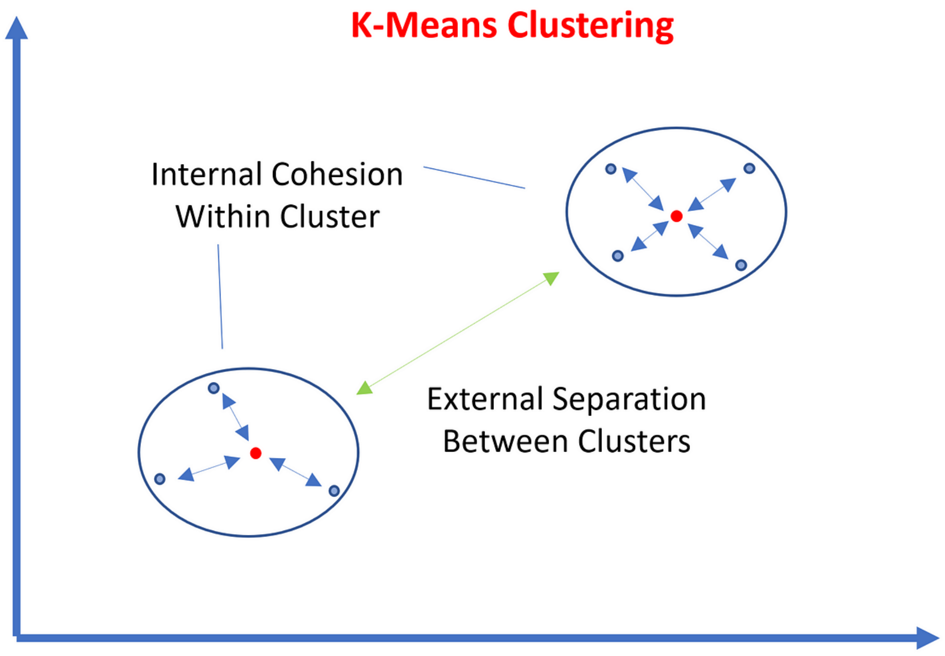

K-Means Clustering
Internal Cohesion Within Cluster
External Separation Between Clusters

Figure 4.7: K-Means clustering.

Coding demonstration of the K-means clustering utilizing the dynamic time warping, with Python code:

```
results = []
for dd_code in unique_dd_codes_per_fuelTypeID:

    temp_dd = geospatial_df[geospatial_df["
      KALCODE"] == dd_code]
    neighbors = temp_dd["neighbors"].values

    time_series_list = []
    dd_names = []

    if neighbors.shape[0] != 0:

        for dd in neighbors[0]:
            dd_names.append(dd)
            temp_df = fuelPrices_per_dd[
              fuelPrices_per_dd["KOD_21"] == dd]
            temp_df_gas_prices = temp_df["
              fuelPrice"].values
            time_series_list.append(
              temp_df_gas_prices)

        max_length = max(len(ts) for ts in
          time_series_list)
        time_series_padded = [np.pad(ts, (0,
          max_length - len(ts)), 'constant',
          constant_values=np.nan) for ts in
          time_series_list]

        X = to_time_series_dataset(
          time_series_padded)
        X = TimeSeriesScalerMeanVariance().
          fit_transform(X)
        model = TimeSeriesKMeans(n_clusters=3,
  metric="dtw", max_iter=5)
y = model.fit_predict(X)
results[dd_code] = y
```

4.4 Unit Root Testing Analysis

The Augmented Dickey-Fuller (ADF) test is a statistical tool used to ascertain the presence of a unit root in a time series dataset, which indicates non-stationarity. Non-stationarity refers to a situation where the statistical properties such as mean, variance, and autocorrelation structure of a time series change over time. The ADF test works by formulating hypotheses about the stationarity of the time series.

To evaluate these hypotheses, the ADF test employs a regression model, typically a first-difference model. In this model, the first difference of the time series (Δy_t) is regressed on lagged values of the time series itself (y_{t-1}) and possibly a linear trend term (βt). The model can be represented as:

$$\Delta y_t = \alpha + \beta_t + \gamma y_t + \delta_1 \Delta y_{t-1} + \delta_2 \Delta y_{t-2} + ... + \delta_p \Delta y_{t-p} + \epsilon_t, \quad (4.1)$$

where Δy_t is the first difference of the time series at time t, α is a constant term, β_t is a linear trend term (if included), γ is the coefficient of the lagged dependent variable, $\delta_1, \delta_2, ..., \delta_p$ are coefficients of the lagged differences of the dependent variable and ϵ_t is the error term.

The null hypothesis (H_0) posits that the time series has a unit root, implying non-stationarity. Mathematically, it can be represented as:

$$H_0 : \gamma = 0 \quad (4.2)$$

The alternative hypothesis (H_1) suggests that the time series is stationary, meaning it lacks a unit root. Mathematically, it can be represented as:

$$H_1 : \gamma < 0 \quad (4.3)$$

Once the model is estimated using ordinary least squares (OLS) regression, the ADF test statistic is computed. This test statistic compares the estimated coefficient of the lagged dependent variable (γ) to its standard error. The test statistic is then compared to critical values from a distribution table, which are determined based on the sample size and chosen significance level (e.g., 1%, 5%, 10%) [32].

If the test statistic is less than the critical value, the null hypothesis of a unit root is rejected, indicating that the time series is stationary. Conversely, if the test statistic exceeds the critical value, the null hypothesis cannot be rejected, suggesting non-stationarity.

In conclusion, the ADF test provides valuable insights into the stationarity properties of time series data, helping researchers and practitioners determine whether a time series is stationary or non-stationary, and informing subsequent analysis and modeling decisions.

Coding demonstration of the ADF unit root testing, with Python code:

```
from arch.unitroot import ADF

adf_results = []
for ts, name in zip(time_series, list_of_names):

    ts_cleaned = ts[~np.isnan(ts)]
    if len(ts_cleaned) > 0:
        adf = ADF(ts_cleaned, trend="ct", lags=30)
        adf_results.append((name, adf))
    else:
        print(f"No valid data for ADF test in {name}")

for name, adf in adf_results:
    print(adf.summary().as_text())
```

4.5 Cointegration Testing Analysis

Cointegration testing is a statistical method used to assess whether multiple time series possess a long-term relationship, despite individual series being non-stationary. The concept of cointegration was introduced by Clive Granger, emphasizing the presence of a stable, shared stochastic trend among variables. The most common cointegration test is the Engle-Granger two-step procedure [36], which involves two main steps: regression and residual analysis.

In the first step, the time series are regressed against each other to identify potential cointegration relationships. Specifically, if there are two time series, say Y_t and X_t, the regression equation takes the form:

$$Y_t = \beta_0 + \beta_1 X_t + \epsilon_t \tag{4.4}$$

If the residuals (ϵ_t) from this regression are stationary, it suggests the existence of a cointegrating relationship between Y_t and X_t.

The second step involves testing the stationarity of the residuals. The Augmented Dickey-Fuller (ADF) test or the Phillips-Perron test is often employed for this purpose. If the residuals are found to be stationary, it implies the presence of cointegration, indicating a long-term relationship between the time series.

The null hypothesis (H_0) in cointegration testing states that there is no cointegrating relationship between the time series, implying that the coefficient of the lagged variable in the regression equation is equal to zero. Mathematically, it can be represented as:

$$H_0 : \beta_1 = 0 \tag{4.5}$$

In contrast, the alternative hypothesis (H_1) asserts the presence of cointegration, indicating a non-zero coefficient of the lagged variable. Mathematically, it can be represented as:

$$H_1 : \beta_1 \neq 0 \tag{4.6}$$

These hypotheses reflect the underlying assumptions being tested in cointegration analysis. The null hypothesis assumes no long-term relationship between the time series, while the alternative hypothesis suggests the presence of a stable, shared stochastic trend among variables.

The statistical significance of cointegration is typically assessed using the p-value and t-statistic derived from the cointegration test. The t-statistic measures the strength of the cointegrating relationship, indicating the number of standard deviations the estimate is from zero. The higher the t-statistic, the more likely it is that the relationship is genuine.

The p-value, on the other hand, represents the probability of obtaining the observed results (or more extreme) under the assumption that there is no cointegration. A small p-value (usually below a chosen significance level, e.g., 0.05) leads to the rejection of the null hypothesis of no cointegration, suggesting a strong likelihood that cointegration exists.

In summary, the cointegration test assesses whether time series possess a stable, long-term relationship by analyzing the stationarity of residuals after regressing the series against each other. The t-statistic and p-value derived from this test provide quantitative measures of the strength and significance of the cointegrating relationship, aiding researchers in making informed decisions about the presence of cointegration in their data.

Coding demonstration of the cointegration testing, with Python code:

```
import statsmodels.tsa.stattools as ts

def run_cointegration_test(time_series_padded):
    num_series = len(time_series_padded)
    t_statistics = np.zeros((num_series, num_series))
    p_values = np.zeros((num_series, num_series))

    for i in range(num_series):
        for j in range(i+1, num_series):
            ts1 = time_series_padded[i]
            ts2 = time_series_padded[j]
            mask = ~np.isnan(ts1) & ~np.isnan(ts2)
            ts1 = ts1[mask]
            ts2 = ts2[mask]

            result = ts.coint(ts1, ts2)
            t_statistics[i, j] = result[0]
            p_values[i, j] = result[1]

    return t_statistics, p_values

t_statistics, p_values = run_cointegration_test(time_series)
```

Chapter 5

Results

In this chapter, we present the outcomes of the clustering procedure, organized into two distinct sections for enhanced comprehension. The initial section details the results for all communities in Greece according to fuel type.

The subsequent section focuses on individual island communities as well as the islands within each island complex, considering all fuel types. Additionally, the section encompasses all island complexes for the three available fuel types: "Unleaded 95," "Unleaded 98/100," and "Diesel."

Lastly, the chapter provides statistical summaries and graphical representations of the clustering process.

5.1 Visualization

To enhance the comprehension of the results, and given the exceptionally high number of communities, we present a single case of clustering for each clustering category. The plots illustrate the outcome of the clustering procedure, with distinct clusters distinguished by different colors. The color scheme is derived from the average fuel price per cluster, with lighter to darker shades of red indicating lower to higher average prices. Moreover, certain communities are highlighted with a yellow border, signifying their specific cluster or the capital of a prefecture.

5.2 Individual Communities

Following the completion of the clustering procedure, the outcomes for individual communities are outlined in this section. Specifically, for the "Unleaded 95" fuel type, data from 1842 communities were available, revealing a significant finding: 1280 communities, constituting 69.48%, did not demonstrate strong resemblance in their time series with neighboring communities.

Table 5.1: Statistics of Clustering Procedure for all Communities of Greece.

Fuel Type	0 Neighbors	1 Neighbor	2 Neighbors	3 Neighbors	$\geq$ 4 Neighbors
Unleaded 95	1280 (69.48%)	238 (12.92%)	173 (9.39%)	86 (4.66%)	65 (3.52%)
Unleaded 98/100	904 (82.93%)	94 (8.62%)	45 (4.12%)	20 (1.83%)	27 (2.47%)
Diesel	1305 (70.88%)	229 (12.43%)	161 (8.74%)	69 (3.74%)	77 (4.18%)

Similarly, for the "Unleaded 98/100" fuel type, 1090 communities were considered. The key observation once again highlighted a substantial proportion, with 904 communities (82.93%) exhibiting no significant resemblance in their time series with neighboring communities.

Concluding the individual community analysis, the "Diesel" fuel type had information available from 1841 communities. Of these, 1305 communities, or 70.88%, displayed no significant resemblance with their neighbors in terms of time series. These findings underscore the diversity and variations in fuel price trends across different community types.

The detailed analysis, as illustrated in Table 5.1, further breaks down the distribution of communities exhibiting resemblance with zero, one, two, three, and at least four neighbors.

Due to the considerable number of 1842 communities, we have strategically chosen to present the clustering results for a randomly selected community. The outcomes of the clustering procedure for this community are depicted in Figures 5.1, B.1, and B.2.

These figures provide a representative snapshot of the clustering process and its impact on fuel types "Unleaded 95," "Unleaded 98/100," and "Diesel" for the community within the broader analysis.

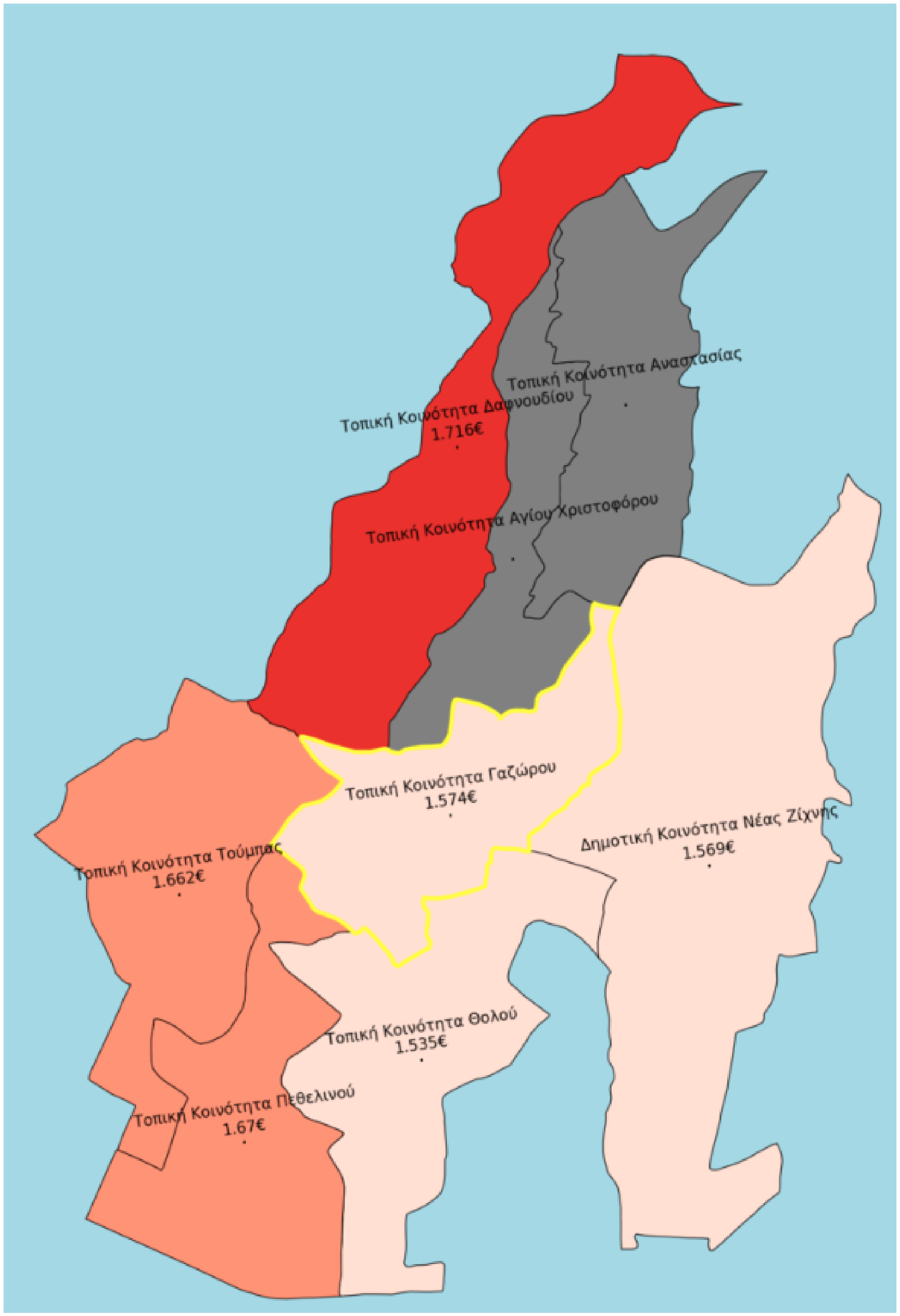

Τοπική Κοινότητα Αναστασίας
Τοπική Κοινότητα Δαφνουδίου
1.716€
Τοπική Κοινότητα Αγίου Χριστοφόρου
Τοπική Κοινότητα Γαζώρου
1.574€
Δημοτική Κοινότητα Νέας Ζίχνης
1.569€
Τοπική Κοινότητα Τούμπας
1.662€
Τοπική Κοινότητα Θολού
1.535€
Τοπική Κοινότητα Πεθελινού
1.67€

Figure 5.1: Clustering of "Τοπική Κοινότητα Γαζώρου' community, for fuel type "Unleaded 95". Average fuel price per community, annotated.

5.3 Island Communities

This section delves into the clustering outcomes of the second level of analysis undertaken in this study, focusing on the island level. It offers insights into the clustering procedure for individual island communities, individual islands within each island complex, and, finally, clustering across all island complexes. The comprehensive exploration of these clustering results enhances our understanding of fuel price trends and correlations at the island level, contributing valuable insights to the broader analysis.

5.3.1 Individual Island Communities Clustering

In this subsection we project the results of the clustering regarding all island communities. After the completion of the clustering procedure, this subsection details the outcomes for individual island communities. Specifically, for the "Unleaded 95" fuel type, data from 442 communities revealed a notable finding: 363 communities, comprising 82.12%, did not exhibit a strong resemblance in their time series with neighboring communities.

Similarly, for the "Unleaded 98/100" fuel type, where 294 island communities were considered, a significant proportion was observed, with 278 communities (94.55%) displaying no substantial resemblance in their time series with neighboring communities.

Concluding the individual community analysis for the "Diesel" fuel type, information from 442 island communities was available. Of these, 371 communities, or 89.93%, exhibited no significant resemblance with their neighbors in terms of time series. These findings highlight the diverse fuel price trends across different community types.

Table 5.2: Statistics of Clustering Procedure for all Island Communities of Greece.

Fuel Type	0 Neighbors	1 Neighbor	2 Neighbors	3 Neighbors	$\geq$ 4 Neighbors
Unleaded 95	363 (82.12%)	48 (10.85%)	20 (4.52%)	10 (2.26%)	1 (0.22%)
Unleaded 98/100	278 (94.55%)	10 (3.40%)	3 (1.02%)	2 (0.68%)	1 (0.34%)
Diesel	371 (83.93%)	42 (9.50%)	16 (3.61%)	10 (2.26%)	3 (0.67%)

Furthermore, the detailed analysis, as depicted in Table 5.2, provides a breakdown of the distribution of island communities exhibiting resemblance with zero, one, two, three, and at least four neighbors.

Given the considerable number of 442 communities, we strategically present the clustering results for a randomly selected island community. The outcomes of the clustering procedure for this community are illustrated in Figures 5.2, B.3, and B.4.

These figures offer a representative glimpse into the clustering process and its implications for fuel types "Unleaded 95," "Unleaded 98/100," and "Diesel" within the context of island communities.

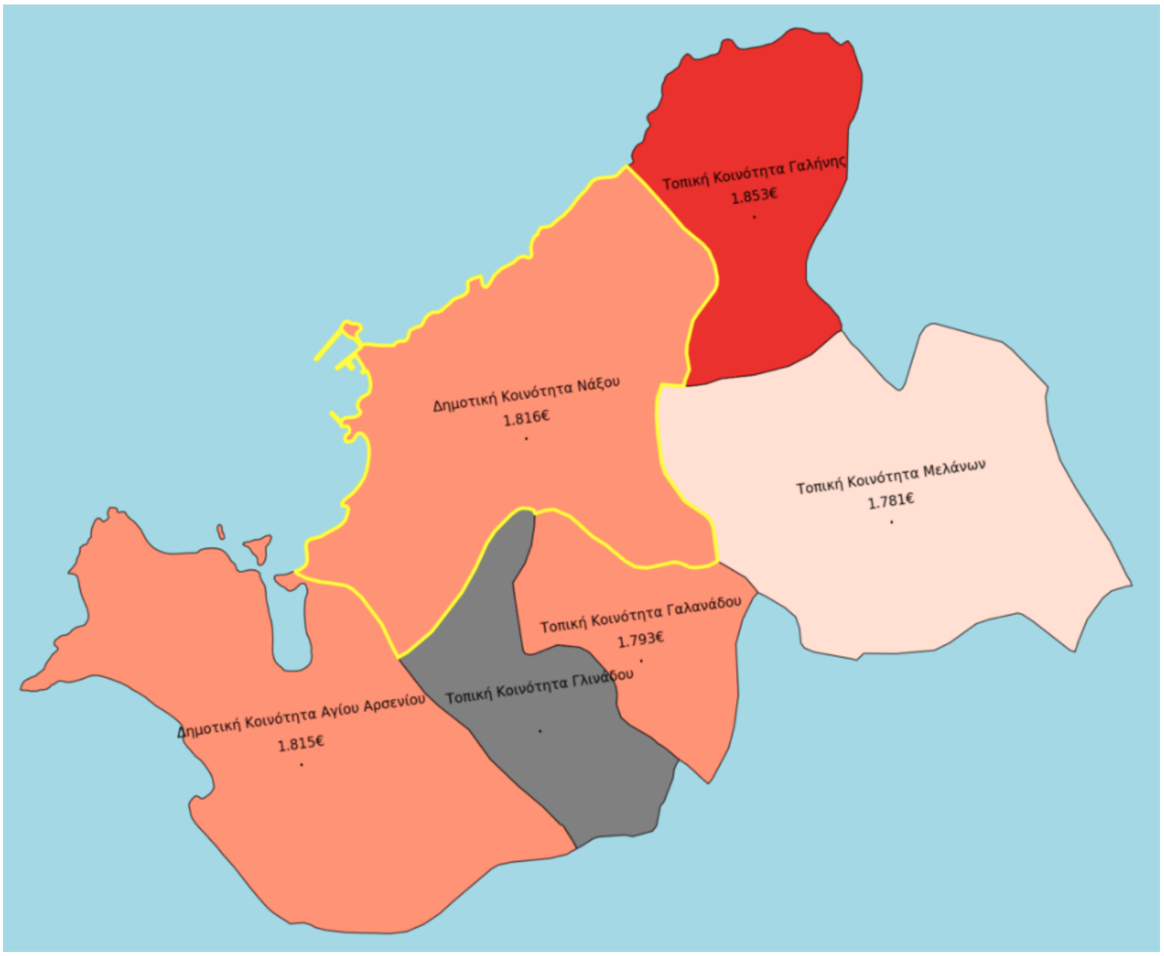


Figure 5.2: Clustering of "Δημοτική Κοινότητα Νάξου" community, belonging to island of "Νάξος", for fuel type "Unleaded 95". Average fuel prices per community, annotated.

5.3.2 Island Clustering

In this subsection, we analyze the clustering results for individual islands within each island complex. As elucidated in section 3.2.3, all six major island complexes in Greece were successfully identified, namely: "ΚΥΚΛΑΔΕΣ", "ΣΠΟΡΑΔΕΣ" , "ΔΩΔΕΚΑΝΗΣΑ", "ΕΠΤΑΝΗΣΑ", "ΑΡΓΟΣΑΡΩΝΙΚΟΣ", and "ΑΝΑΤΟΛΙΚΟ ΑΙΓΑΙΟ". Furthermore, a total of 50 islands were identified across all island complexes. The clustering procedure was conducted six times, with the algorithm set to identify four clusters within each island complex, encompassing the islands associated with each respective complex for the three fuel types: "Unleaded 95," "Unleaded 98/100," and "Diesel."

The results are presented in a similar format as the previous section in Tables 5.3, 5.4, and 5.5 for the respective fuels. The clusters are sorted from lowest average fuel price to highest.

These tables offer a detailed breakdown of the clustering outcomes for individual islands within their respective island complexes, shedding light on the patterns and similarities in fuel price trends across these geographical entities.

Table 5.3: Statistics of Clustering Procedure for Island Communities Having "Unleaded 95" Fuel Information.

Complex	Cluster 1	Cluster 2	Cluster 3	Cluster 4	Total Islands
"ΣΠΟΡΑΔΕΣ"	1	1	1	1	4
"ΑΝΑΤΟΛΙΚΟ ΑΙΓΑΙΟ"	4	1	3	1	9
"ΔΩΔΕΚΑΝΗΣΑ"	1	10	2	2	15
"ΚΥΚΛΑΔΕΣ"	4	3	4	3	14
"ΑΡΓΟΣΑΡΩΝΙΚΟΣ"	1	1	0	0	2
"ΕΠΤΑΝΗΣΑ"	1	3	1	1	6

Considering the number of six island complexes, we strategically showcase the clustering results for a randomly chosen island complex. The outcomes of the clustering procedure for this specific complex are depicted in Figures 5.3, B.5, and B.6.

These figures offer an illustrative overview of the clustering process and its implications for fuel types "Unleaded 95," "Unleaded 98/100," and "Diesel" within the context of island complexes. These visual representations provide valuable insights into the clustering patterns

Table 5.4: Statistics of Clustering Procedure for Island Communities Having "Unleaded 98/100" Fuel Information.

Complex	Cluster 1	Cluster 2	Cluster 3	Cluster 4	Total Islands
"ΣΠΟΡΑΔΕΣ"	1	1	1	1	4
"ΑΝΑΤΟΛΙΚΟ ΑΙΓΑΙΟ"	1	1	4	2	8
"ΔΩΔΕΚΑΝΗΣΑ"	9	1	4	1	15
"ΚΥΚΛΑΔΕΣ"	1	6	5	1	14
"ΑΡΓΟΣΑΡΩΝΙΚΟΣ"	1	0	0	0	1
"ΕΠΤΑΝΗΣΑ"	1	3	1	1	6

Table 5.5: Statistics of Clustering Procedure for Island Communities Having "Diesel" Fuel Information.

Complex	Cluster 1	Cluster 2	Cluster 3	Cluster 4	Total Islands
ΣΠΟΡΑΔΕΣ	1	1	1	1	4
ΑΝΑΤΟΛΙΚΟ ΑΙΓΑΙΟ	1	6	1	1	9
ΔΩΔΕΚΑΝΗΣΑ	1	10	1	3	15
ΚΥΚΛΑΔΕΣ	2	4	6	1	14
ΑΡΓΟΣΑΡΩΝΙΚΟΣ	1	1	0	0	2
ΕΠΤΑΝΗΣΑ	2	1	2	1	6

and variations in fuel prices specific to the selected island complex.

For a more comprehensive understanding of the clustering procedure, the projection of the time series is presented in Figures 5.4 and 5.5, where the time series belonging to the same cluster are plotted on the same grid. These figures offer a visual representation of the clustering outcomes, facilitating the observation of patterns and trends within each cluster.

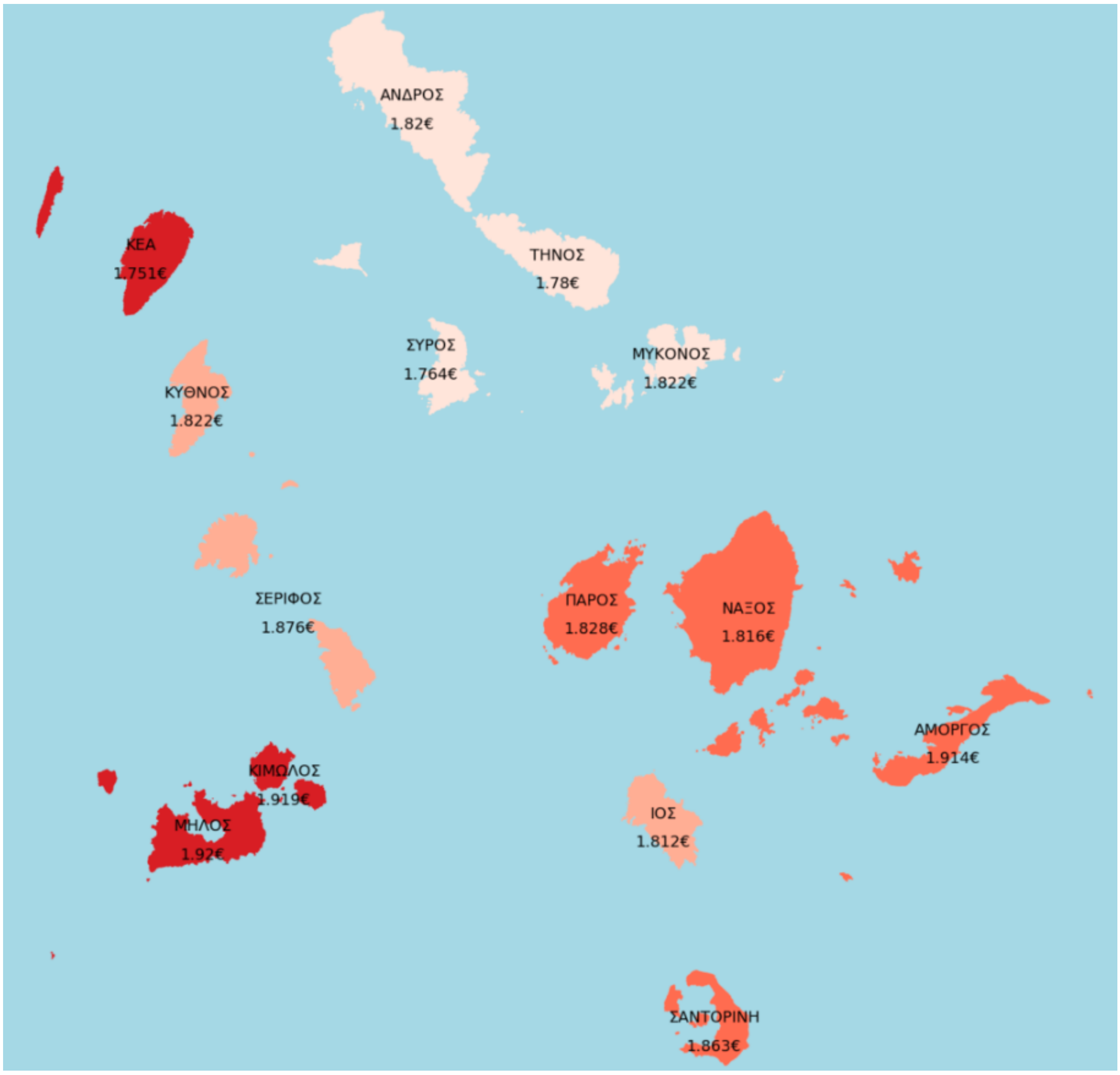


Figure 5.3: Clustering of "Κυκλάδες" island complex, for fuel type "Unleaded 95". Average fuel prices per island, annotated.

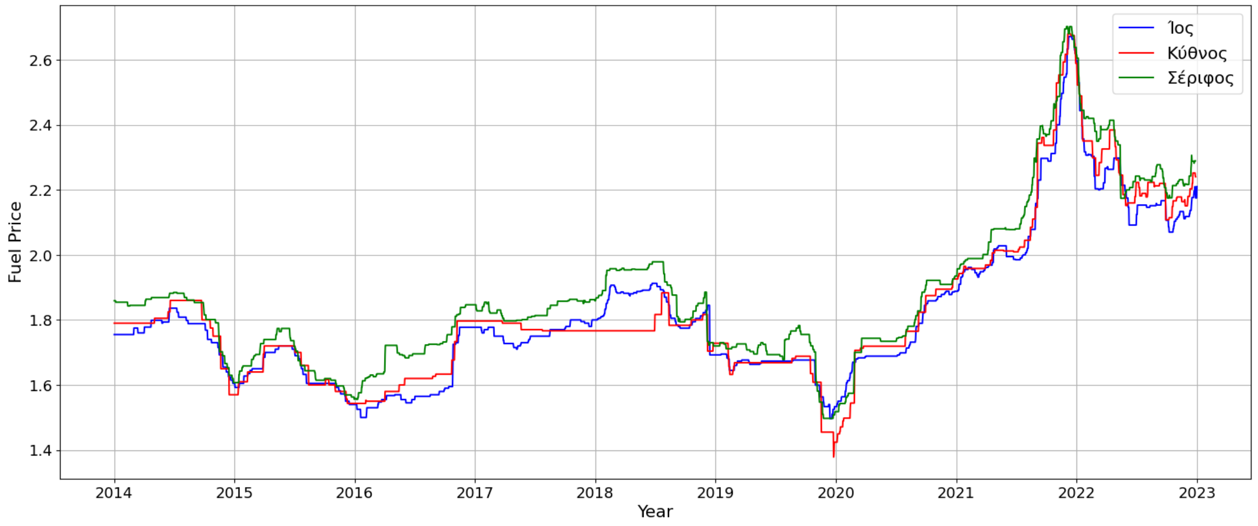


Figure 5.4: Time series comparison between "Ίος", "Κύθνος" and "Σέριφος" islands, belonging to the same cluster "Cluster 2", for fuel type "Unleaded 95".

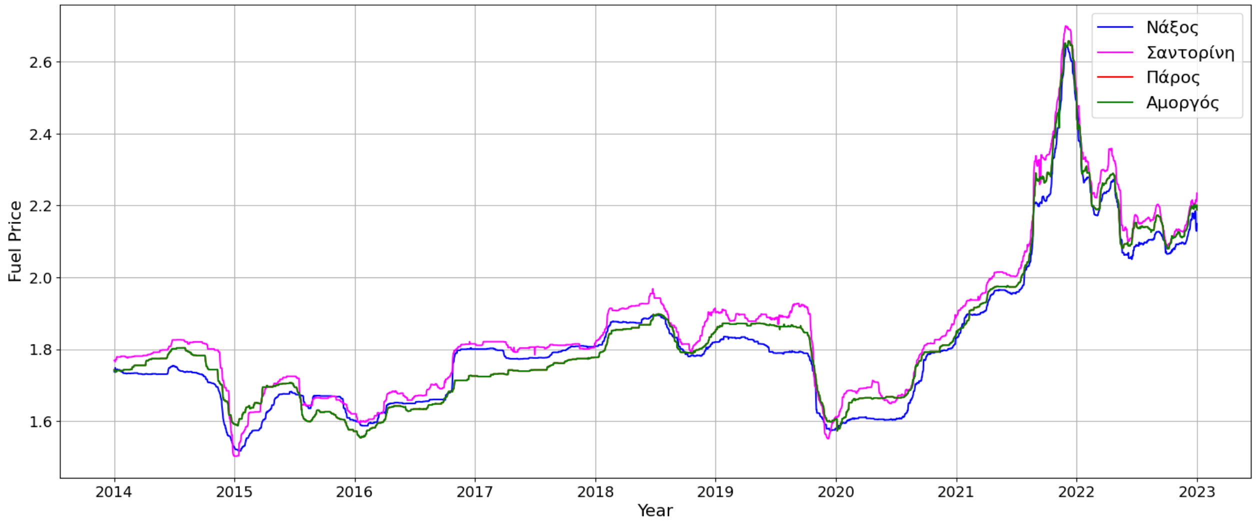


Figure 5.5: Time series comparison between "Νάξος", "Πάρος","Αμοργός" and "Σαντορίνη" islands, belonging to the same cluster "Cluster 3", for fuel type "Unleaded 95".

5.3.3 Complex of Islands Clustering

The final level of analysis in this thesis involved clustering the island complexes. To identify clusters among the six aforementioned island complexes, the algorithm was fine-tuned to identify up to four clusters. The clustering procedure was executed three times, once for each of the three different fuel types.

The outcomes of the clustering process are depicted in Figures 5.6, B.7, and B.8. These figures provide a visual representation of the clustering results, offering insights into the grouping patterns and variations in fuel prices across the different island complexes.

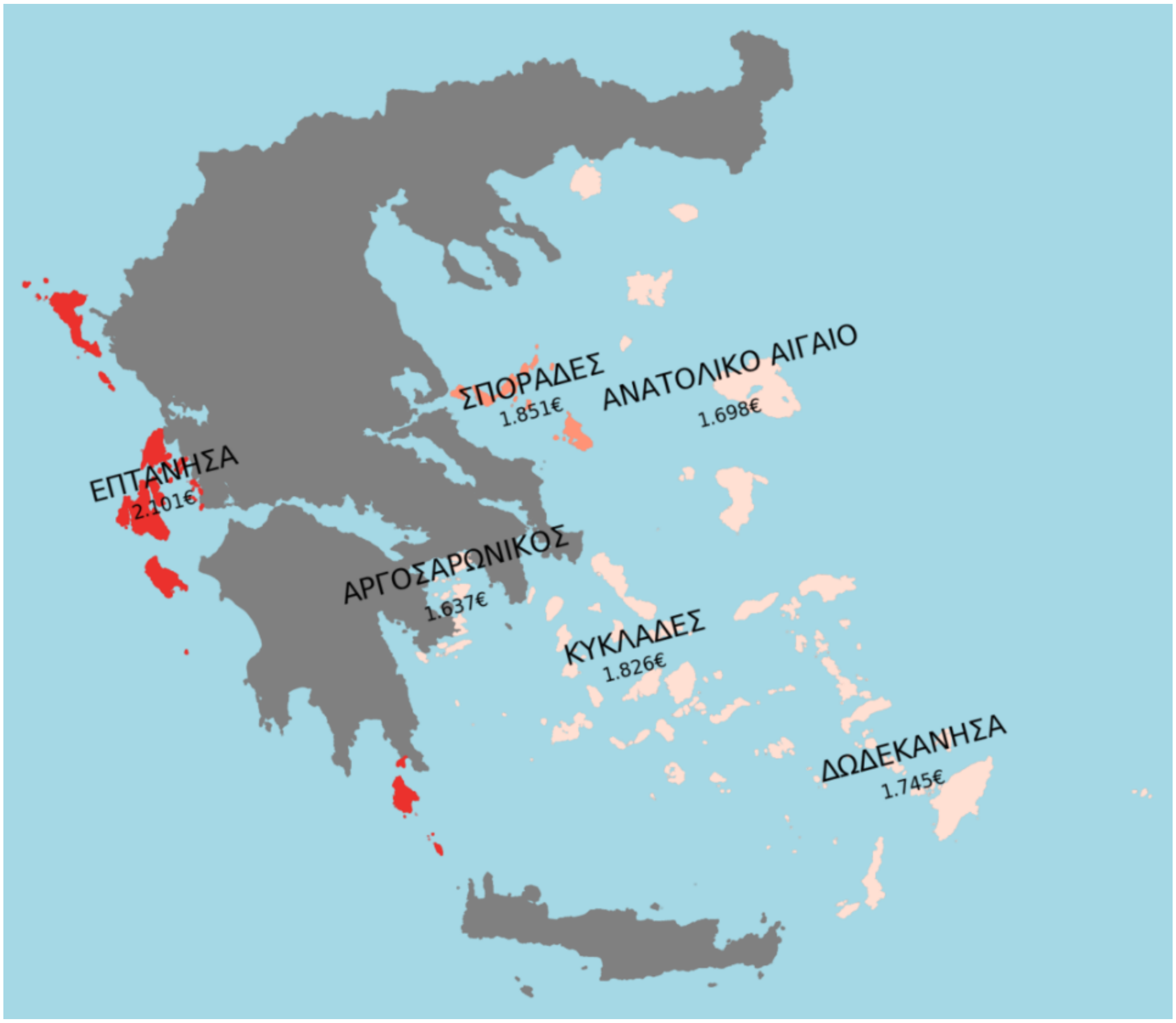


Figure 5.6: Clustering of all island complexes for fuel type "Unleaded 95". Average fuel price per complex, Annotated.

5.4 Prefectures

The final and pivotal segment of the thesis centered on clustering communities at a prefecture level. We set the number of possible clusters to five and calculated the clusters for 40 prefectures. Given the high number of prefectures, we strategically showcase the clustering of the capital of Greece to highlight the pronounced correlation effect. The clustering results are depicted in Figures 5.7, B.9, and B.10.

Analyzing the results for the "Unleaded 95" fuel type, it is evident that a majority of communities exhibit extreme similarities, with cluster 3 encompassing 57 communities. Notably, cluster 5, characterized by the highest average fuel price, includes 15 communities situated in close proximity. These communities are positioned in areas with substantial daily traffic flow, contributing to the higher fuel prices observed.

Furthermore, for the remaining fuel types, "Unleaded 98/100" and "Diesel," the similarities are even more pronounced, with nearly the entire prefecture belonging to a single cluster. In the case of "Unleaded 98/100," all but four out of 75 communities are assigned to cluster 5, and notably, to the cluster with the highest average fuel price. Similarly, for the "Diesel" fuel type, 72 out of 77 communities are assigned to cluster 3, underscoring the substantial correlation and similarities among these communities.

One might speculate that these similarities arise from the urban nature of the area. However, upon examining an instance of a provincial prefecture, we observed that the similarities persist among closely situated communities. This example is illustrated in Figures 5.8 and B.11, where we depict the clustering procedure of a low-residency area situated far from the capital of Greece. For fuel type "Unleaded 95" 25 out of 37 communities are assigned to cluster 2 and for fuel type "Diesel" again 25 out of 37 communities are assigned to cluster 2.

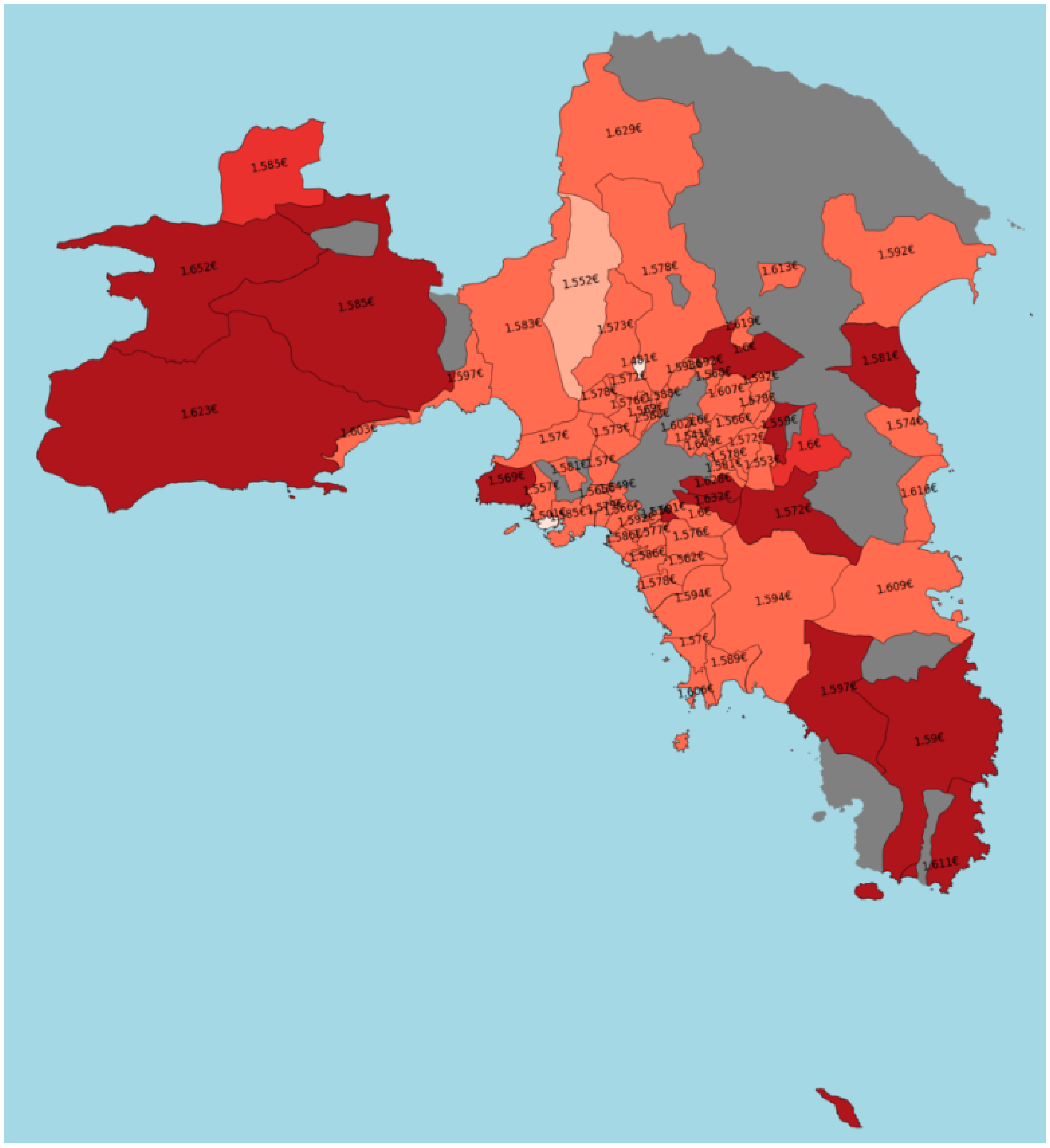


Figure 5.7: Clustering of "Νομός Αττικής' prefecture, for fuel type "Unleaded 95". Average fuel prices per community, annotated.

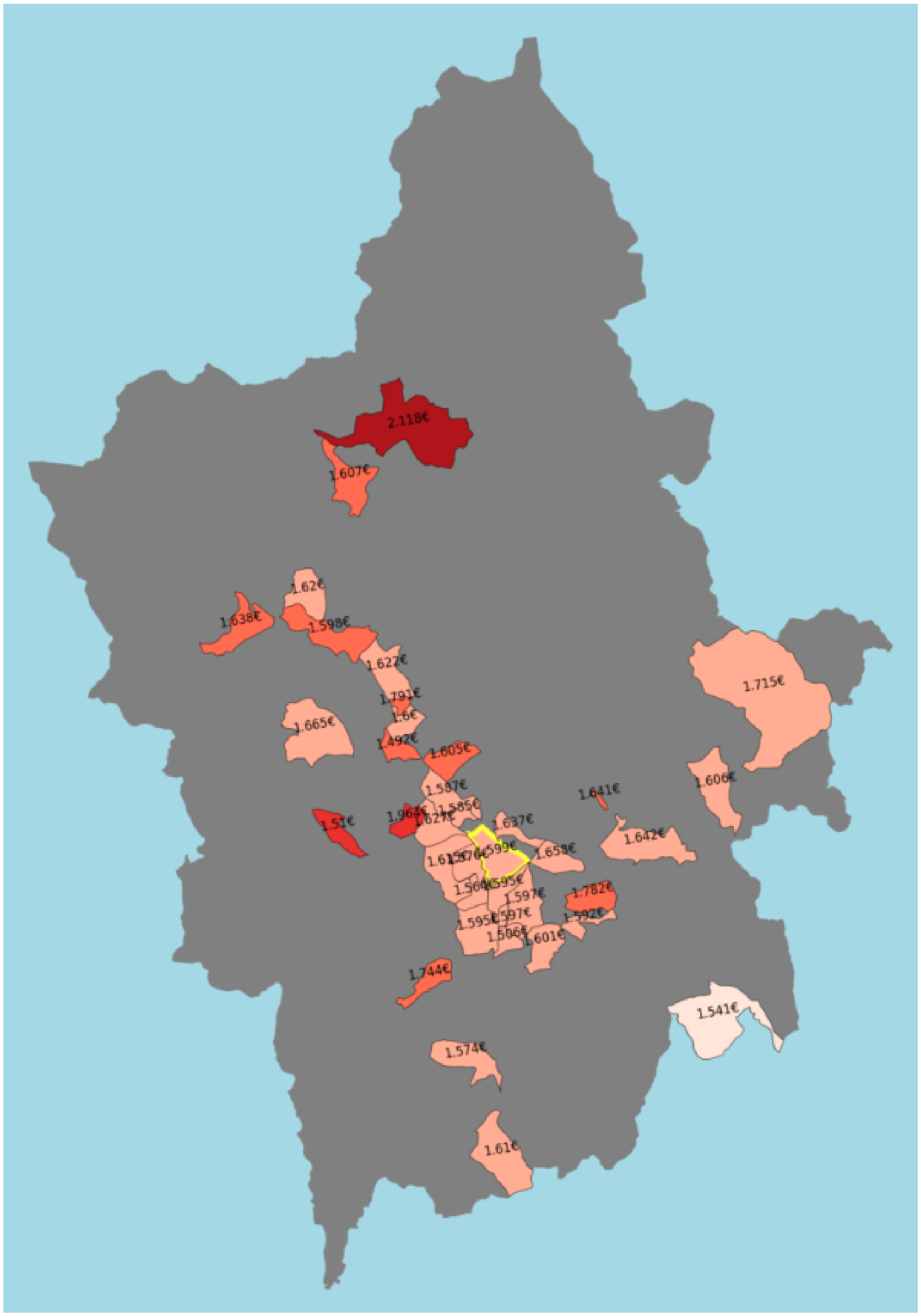


Figure 5.8: Clustering of "Νομός Ιωαννίνων' prefecture, for fuel type "Unleaded 95". Average fuel prices per community, annotated.

5.4.1 Unit Root Test

In order to gain a deeper insight into the time series data, we performed the ADF unit root testing on fuel prices among communities located in Athens, the capital of Greece. This test was conducted for all communities and for the three types of studied fuel: "Unleaded 95," "Unleaded 98/100," and "Diesel." The results of the unit root testing, including the t-statistic and p-value, are depicted in Figures 5.9, 5.10, C.1 and C.2.

Based on the outcomes of the test, we cannot reject the null hypothesis for all the communities tested, indicating that the majority of the time series data have a unit root, implying they are non-stationary. This discovery was crucial to uncover before conducting the subsequent cointegration test, ensuring its meaningful interpretation.

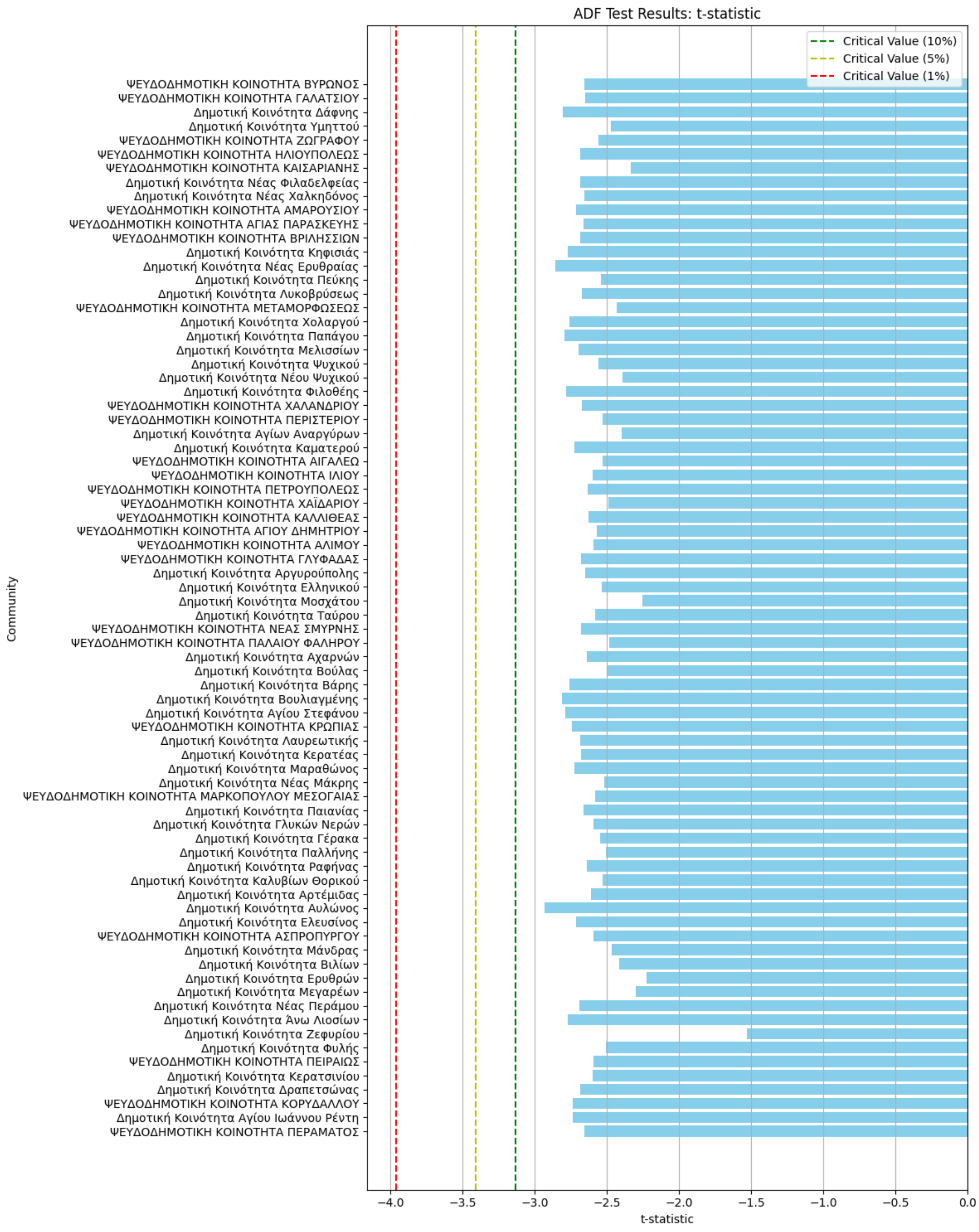


Figure 5.9: t-statistic derived from the unit root test, for fuel type ”Unleaded 95”.

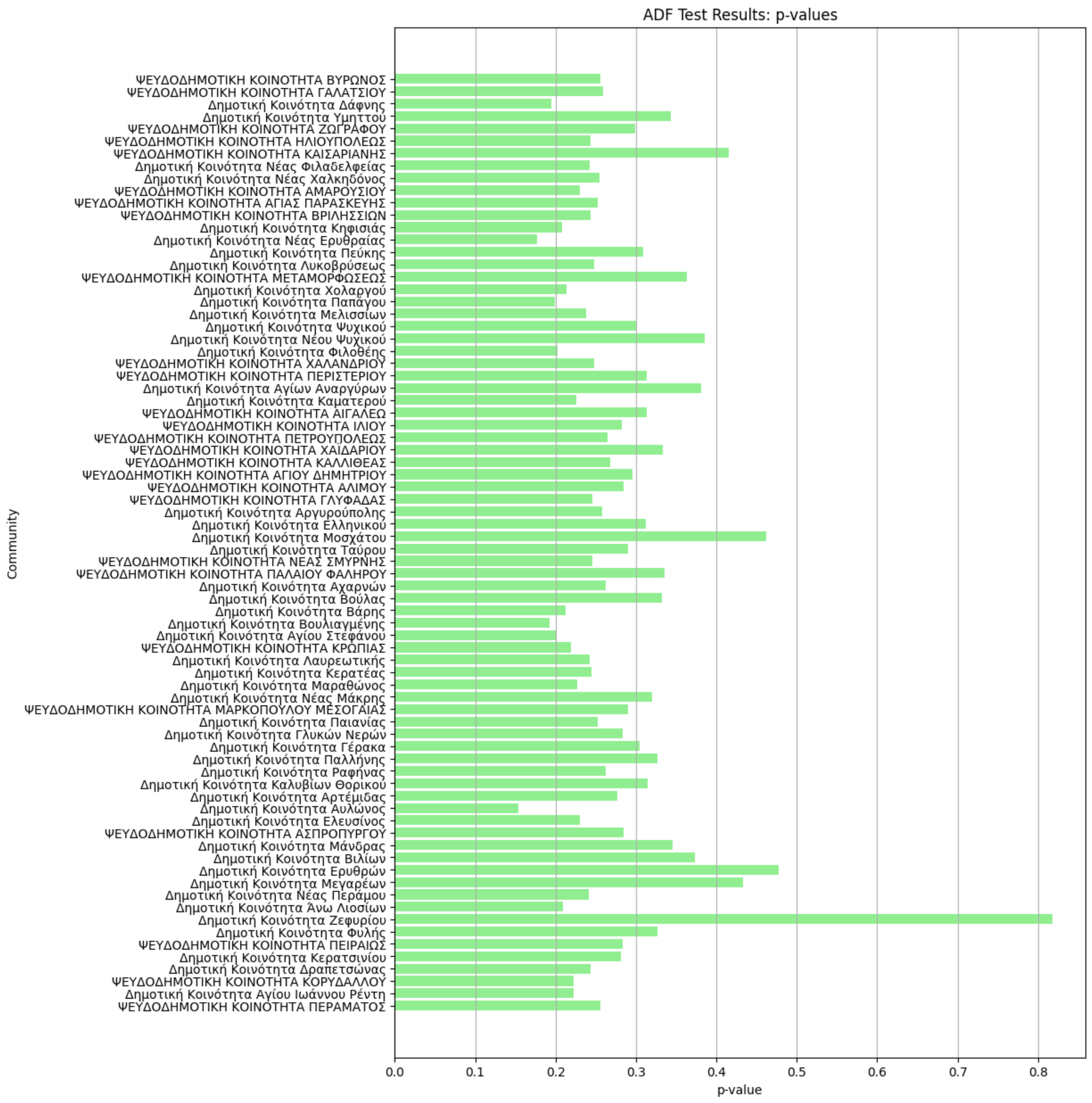


Figure 5.10: p-value derived from the unit root test, for fuel type "Unleaded 95".

5.4.2 Cointegration Test

To further validate the assumptions regarding the significant similarities in fuel prices among the communities belonging to the capital of Greece, we conducted a cointegration test for each fuel type. The results, including the p-value and t-statistical values, are presented in heatmap representations for each fuel type. The heatmaps can be observed in Figures 5.11, 5.12, D.2 and D.1.

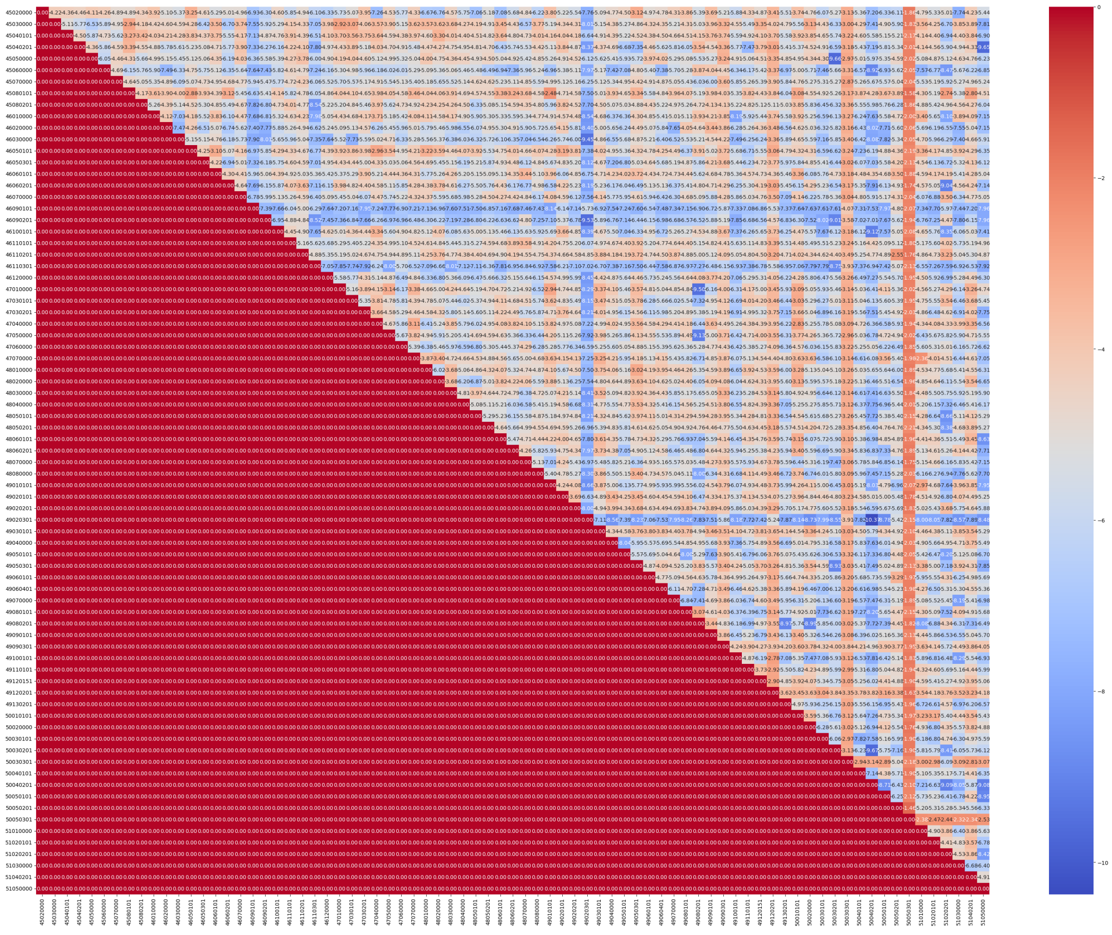


Figure 5.11: t-statistic derived from the cointegration test, for fuel type "Unleaded 95".

Based on the reported metrics, we can confidently reject the null hypothesis, indicating that the time series among the communities of Athens exhibit a significant resemblance and correlation. This not only confirms our initial hypothesis but also validates the results of the clustering analysis, underscoring the effectiveness of the methodology

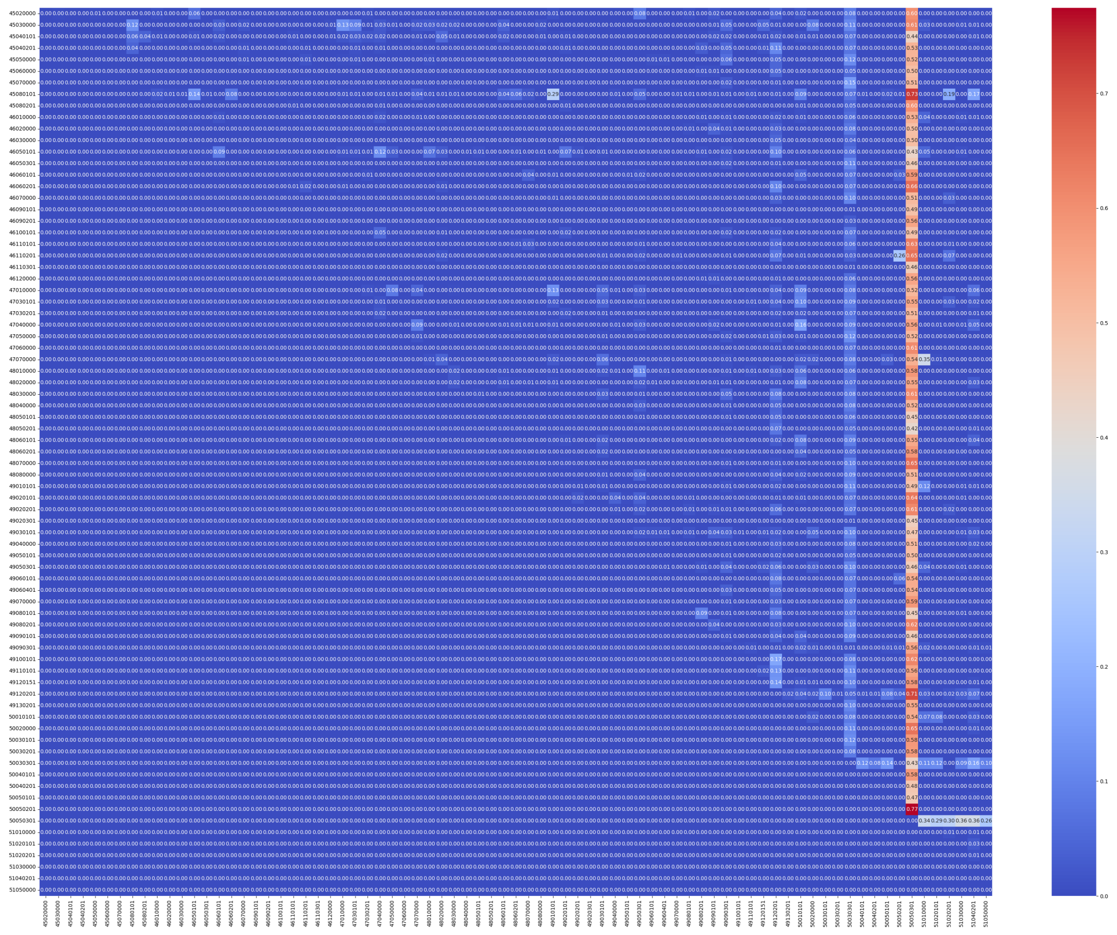


Figure 5.12: p-value derived from the cointegration test, for fuel type "Unleaded 95".

in categorizing communities based on their fuel prices.

Chapter 6

Conclusions

The analysis of fuel prices using advanced clustering techniques revealed compelling findings regarding the interconnectedness of various community types in the Greek market. The influence of neighboring communities on each other was clearly demonstrated, particularly in the context of fuel prices for the three studied fuels. Additionally, the study highlighted a pronounced correlation between islands within an island complex, emphasizing the intricate dynamics that govern fuel pricing in these geographically linked entities. The application of the K-means algorithm further elucidated the influence between complexes of islands, providing a comprehensive understanding of the broader geographical factors influencing fuel prices in Greece. These results underscore the presence of fluctuations and correlations in the Greek fuel market, emphasizing the importance of considering both regional and community-level dynamics when assessing pricing trends. The utilization of sophisticated analytical tools has not only contributed to unraveling these complexities but has also paved the way for more informed discussions and strategic decision-making in the realm of fuel pricing and market dynamics.

Bibliography

[1] Pradeep Rai and Shubha Singh. A survey of clustering techniques. *International Journal of Computer Applications*, 7(12):1–5, 2010.

[2] Vit Niennattrakul and Chotirat Ann Ratanamahatana. On clustering multimedia time series data using k-means and dynamic time warping. In *2007 International Conference on Multimedia and Ubiquitous Engineering (MUE'07)*, pages 733–738. IEEE, 2007.

[3] Chotirat Ann Ratanamahatana and Eamonn Keogh. Multimedia retrieval using time series representation and relevance feedback. In *International Conference on Asian Digital Libraries*, pages 400–405. Springer, 2005.

[4] Vit Niennattrakul and Chotirat Ann Ratanamahatana. Clustering multimedia data using time series. In *2006 International Conference on Hybrid Information Technology*, volume 1, pages 372–379. IEEE, 2006.

[5] Jessica Lin, Eamonn Keogh, Stefano Lonardi, Jeffrey P Lankford, and Donna M Nystrom. Visually mining and monitoring massive time series. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 460–469, 2004.

[6] Eamonn Keogh and Shruti Kasetty. On the need for time series data mining benchmarks: a survey and empirical demonstration. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 102–111, 2002.

[7] Karen Zita Haigh, Wendy Foslien, and Valerie Guralnik. Visual query language: Finding patterns in and relationships among time series data. In *Seventh Workshop on Mining Scientific and Engineering Datasets*, volume 24. Citeseer, 2004.

[8] Eamonn Keogh, Selina Chu, David Hart, and Michael Pazzani. Segmenting time series: A survey and novel approach. In *Data*

mining in time series databases, pages 1–21. World Scientific, 2004.

[9] Jessica Lin, Eamonn Keogh, Stefano Lonardi, and Bill Chiu. A symbolic representation of time series, with implications for streaming algorithms. In *Proceedings of the 8th ACM SIGMOD workshop on Research issues in data mining and knowledge discovery*, pages 2–11, 2003.

[10] Jesin Zakaria, Sarah Rotschafer, Abdullah Mueen, Khaleel Razak, and Eamonn Keogh. Mining massive archives of mice sounds with symbolized representations. In *Proceedings of the 2012 SIAM International Conference on Data Mining*, pages 588–599. SIAM, 2012.

[11] Thanawin Rakthanmanon, Bilson Campana, Abdullah Mueen, Gustavo Batista, Brandon Westover, Qiang Zhu, Jesin Zakaria, and Eamonn Keogh. Searching and mining trillions of time series subsequences under dynamic time warping. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 262–270, 2012.

[12] Mohd Shahnawaz, Ashish Ranjan, and Mohd Danish. Temporal data mining: an overview. *International Journal of Engineering and Advanced Technology*, 1(1):2249–8958, 2011.

[13] Sangeeta Rani and Geeta Sikka. Recent techniques of clustering of time series data: a survey. *International Journal of Computer Applications*, 52(15), 2012.

[14] Jessica Lin, Michail Vlachos, Eamonn Keogh, and Dimitrios Gunopulos. Iterative incremental clustering of time series. In *Advances in Database Technology-EDBT 2004: 9th International Conference on Extending Database Technology, Heraklion, Crete, Greece, March 14-18, 2004 9*, pages 106–122. Springer, 2004.

[15] T Warren Liao. Clustering of time series data—a survey. *Pattern recognition*, 38(11):1857–1874, 2005.

[16] R Pradeep Kumar and P Nagabhushan. Time series as a point- a novel approach for time series cluster visualization. In *DMIN*, pages 24–29. Citeseer, 2006.

[17] Christos Faloutsos, Mudumbai Ranganathan, and Yannis Manolopoulos. Fast subsequence matching in time-series databases. *ACM Sigmod Record*, 23(2):419–429, 1994.

[18] Xiaozhe Wang, Kate Smith, and Rob Hyndman. Characteristic-based clustering for time series data. *Data mining and knowledge Discovery*, 13:335–364, 2006.

[19] Monica Chiş, Soumya Banerjee, and Aboul Ella Hassanien. Clustering time series data: an evolutionary approach. *Foundations of Computational, IntelligenceVolume 6: Data Mining*, pages 193–207, 2009.

[20] Hiroaki Sakoe and Seibi Chiba. Dynamic programming algorithm optimization for spoken word recognition. *IEEE transactions on acoustics, speech, and signal processing*, 26(1):43–49, 1978.

[21] Lawrence R Rabiner and Biing-Hwang Juang. *Fundamentals of speech recognition*. Tsinghua University Press, 1999.

[22] Eamonn Keogh and Chotirat Ann Ratanamahatana. Exact indexing of dynamic time warping. *Knowledge and information systems*, 7:358–386, 2005.

[23] Stan Salvador and Philip Chan. Toward accurate dynamic time warping in linear time and space. *Intelligent Data Analysis*, 11(5):561–580, 2007.

[24] Chotirat Ann Ratanamahatana and Eamonn Keogh. Everything you know about dynamic time warping is wrong. In *Third workshop on mining temporal and sequential data*, volume 32. Citeseer, 2004.

[25] Michail Vlachos, George Kollios, and Dimitrios Gunopulos. Discovering similar multidimensional trajectories. In *Proceedings*

18th international conference on data engineering, pages 673–684. IEEE, 2002.

[26] Meinard Müller. *Information retrieval for music and motion*, volume 2. Springer, 2007.

[27] Donald J Berndt and James Clifford. Using dynamic time warping to find patterns in time series. In *Proceedings of the 3rd international conference on knowledge discovery and data mining*, pages 359–370, 1994.

[28] John Aach and George M Church. Aligning gene expression time series with time warping algorithms. *Bioinformatics*, 17(6):495–508, 2001.

[29] Lei Chen, M Tamer Özsu, and Vincent Oria. Robust and fast similarity search for moving object trajectories. In *Proceedings of the 2005 ACM SIGMOD international conference on Management of data*, pages 491–502, 2005.

[30] David A Dickey and Wayne A Fuller. Distribution of the estimators for autoregressive time series with a unit root. *Journal of the American statistical association*, 74(366a):427–431, 1979.

[31] Said E Said and David A Dickey. Testing for unit roots in autoregressive-moving average models of unknown order. *Biometrika*, 71(3):599–607, 1984.

[32] Denis Kwiatkowski, Peter CB Phillips, Peter Schmidt, and Yongcheol Shin. Testing the null hypothesis of stationarity against the alternative of a unit root: How sure are we that economic time series have a unit root? *Journal of econometrics*, 54(1-3):159–178, 1992.

[33] Graham Elliott, Thomas J Rothenberg, and James H Stock. Efficient tests for an autoregressive unit root, 1992.

[34] James G MacKinnon. Approximate asymptotic distribution functions for unit-root and cointegration tests. *Journal of Business & Economic Statistics*, 12(2):167–176, 1994.

[35] James G MacKinnon. Critical values for cointegration tests. Technical report, Queen's Economics Department Working Paper, 2010.

[36] Robert F Engle and Clive WJ Granger. Co-integration and error correction: representation, estimation, and testing. *Econometrica: journal of the Econometric Society*, pages 251–276, 1987.

Appendix A

Different Maps of Greece

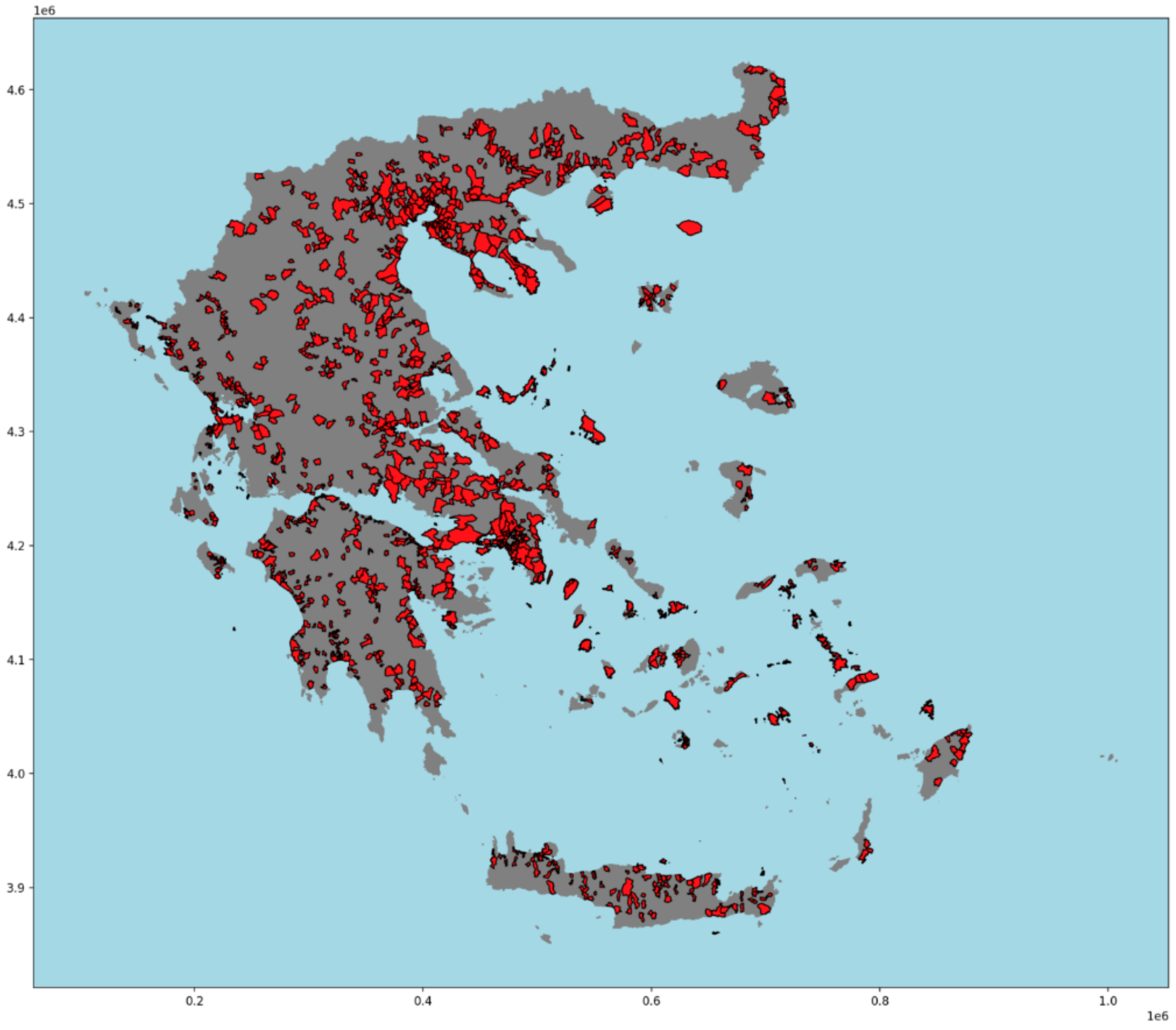


Figure A.1: Map of Greece Including 1090 ”Unleaded 98/100” Communities.

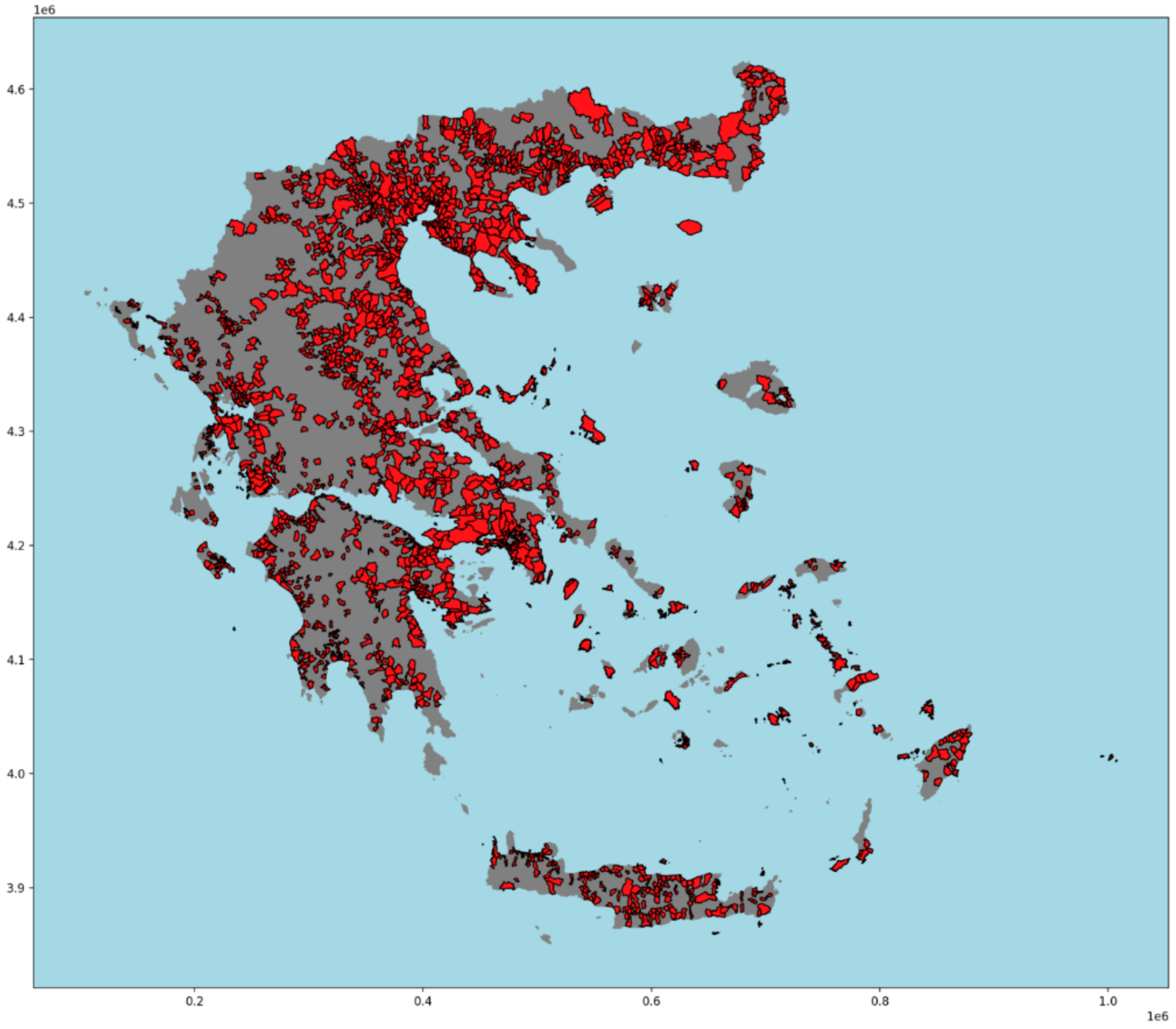


Figure A.2: Map of Greece Including 1841 ”Diesel” Communities.

Appendix B

Additional Clustering Figures

B.1 Clustering of Individual Communities

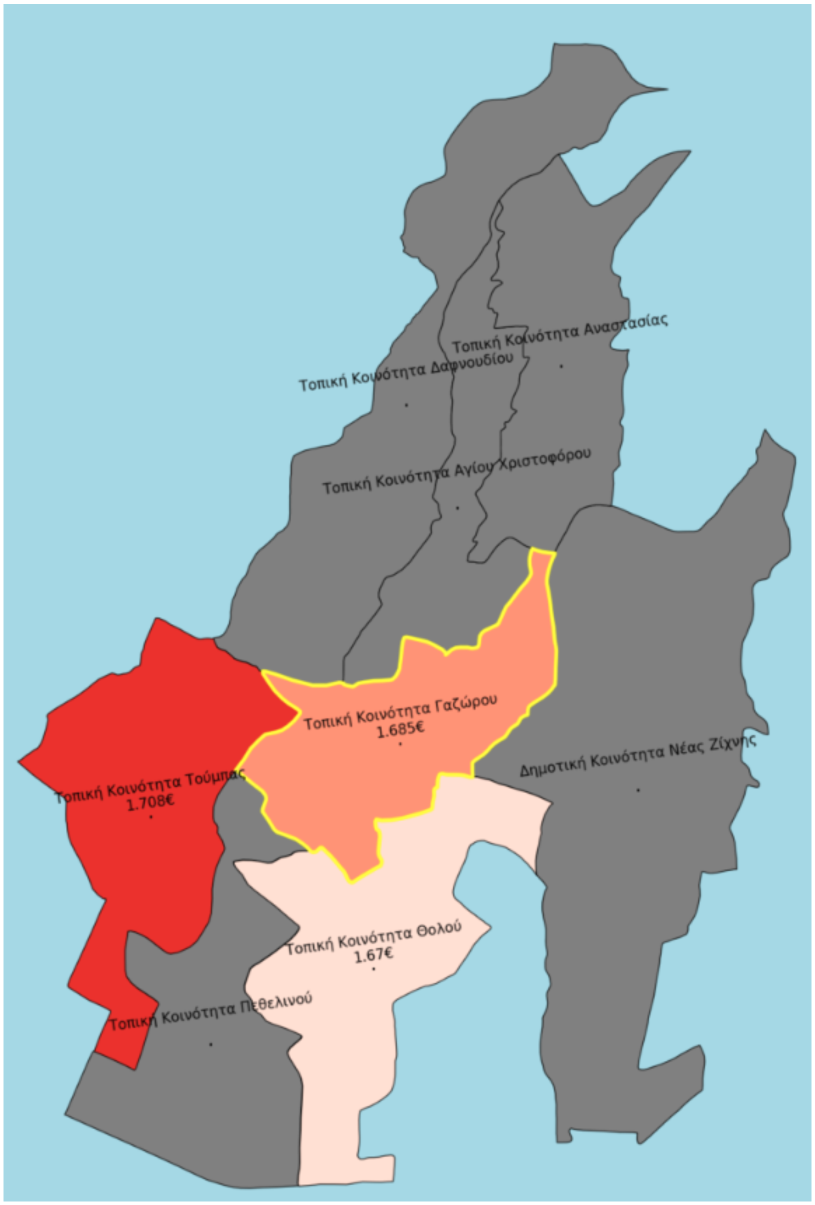

Τοπική Κοινότητα Αναστασίας
Τοπική Κοινότητα Δαφνουδίου
Τοπική Κοινότητα Αγίου Χριστοφόρου
Τοπική Κοινότητα Γαζώρου
1.685€
Δημοτική Κοινότητα Νέας Ζίχνης
Τοπική Κοινότητα Τούμπας
1.708€
Τοπική Κοινότητα Θολού
1.67€
Τοπική Κοινότητα Πεθελινού

Figure B.1: Clustering ”Τοπική Κοινότητα Γαζώρου’ community, for fuel type ”Unleaded 98/100”. Average fuel prices per community, annotated.

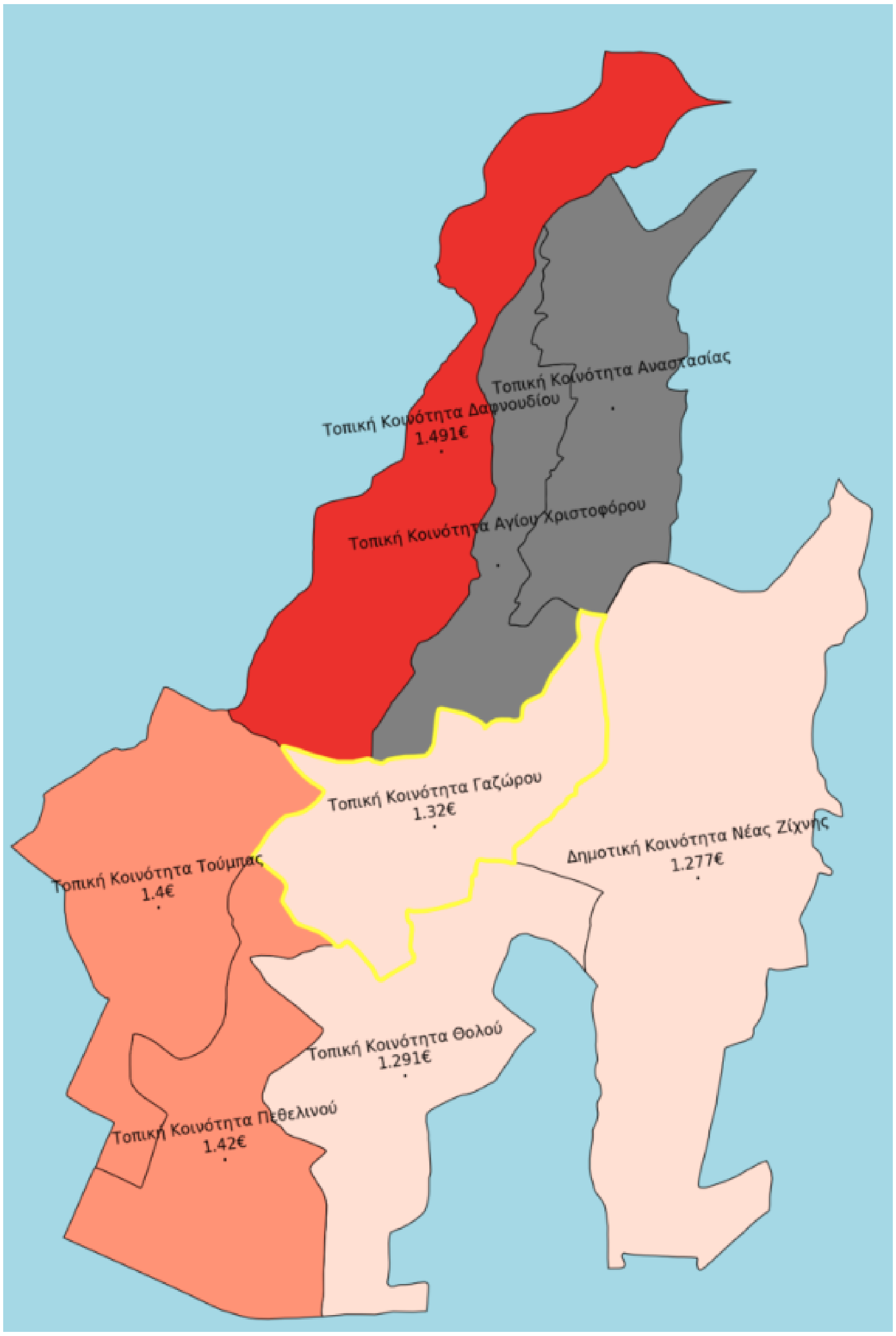

Τοπική Κοινότητα Αναστασίας
Τοπική Κοινότητα Δαφνουδίου
1.491€
Τοπική Κοινότητα Αγίου Χριστοφόρου
Τοπική Κοινότητα Γαζώρου
1.32€
Δημοτική Κοινότητα Νέας Ζίχνης
1.277€
Τοπική Κοινότητα Τούμπας
1.4€
Τοπική Κοινότητα Θολού
1.291€
Τοπική Κοινότητα Πεθελινού
1.42€

Figure B.2: Clustering "Τοπική Κοινότητα Γαζώρου' community, for fuel type "Diesel". Average fuel prices per community, annotated.

B.2 Clustering of Island Communities

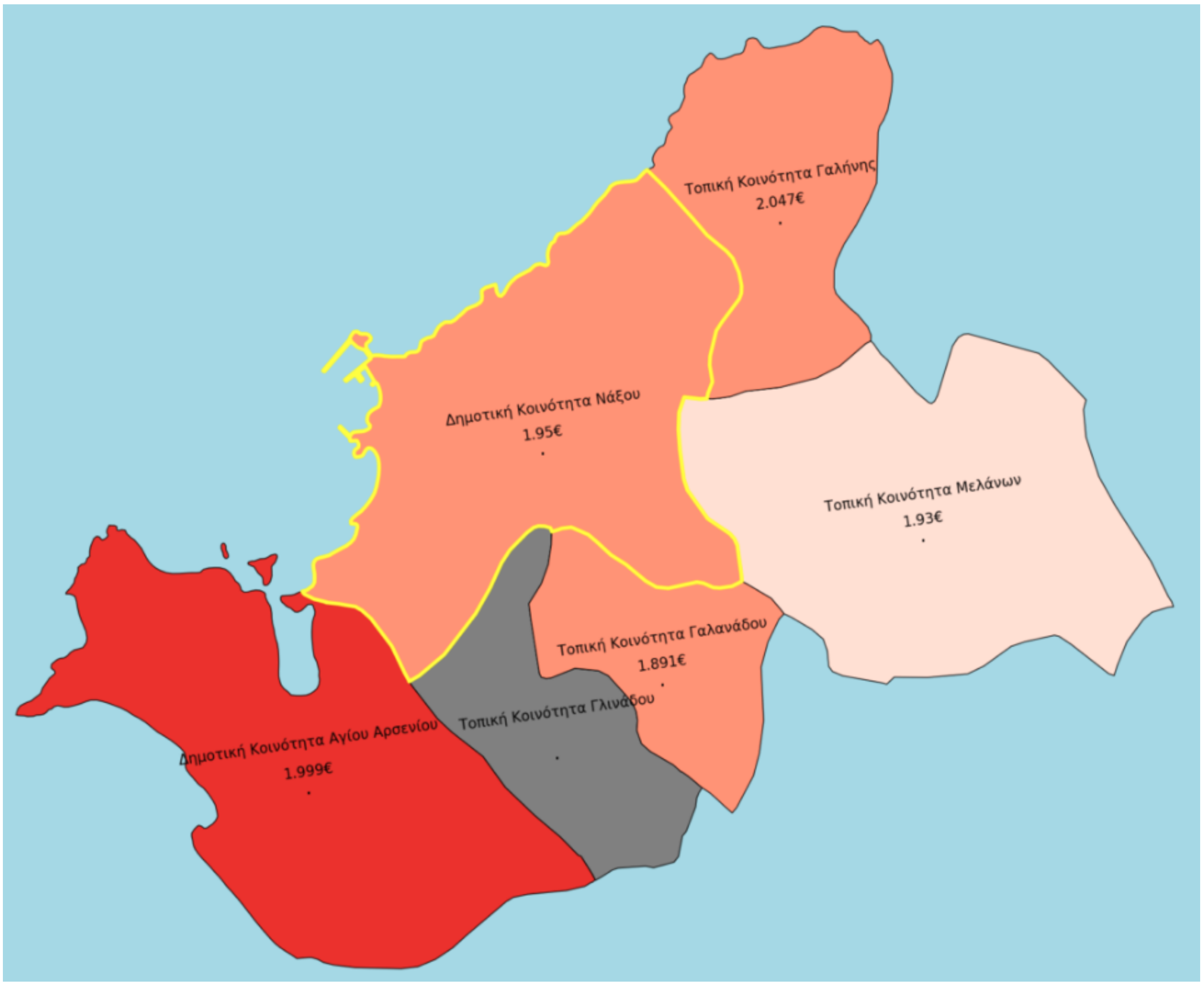


Figure B.3: Clustering of "Δημοτική Κοινότητα Νάξου" island community, that belongs to island of "Νάξος", for fuel type "Unleaded 98/100". Average fuel prices per community, annotated.

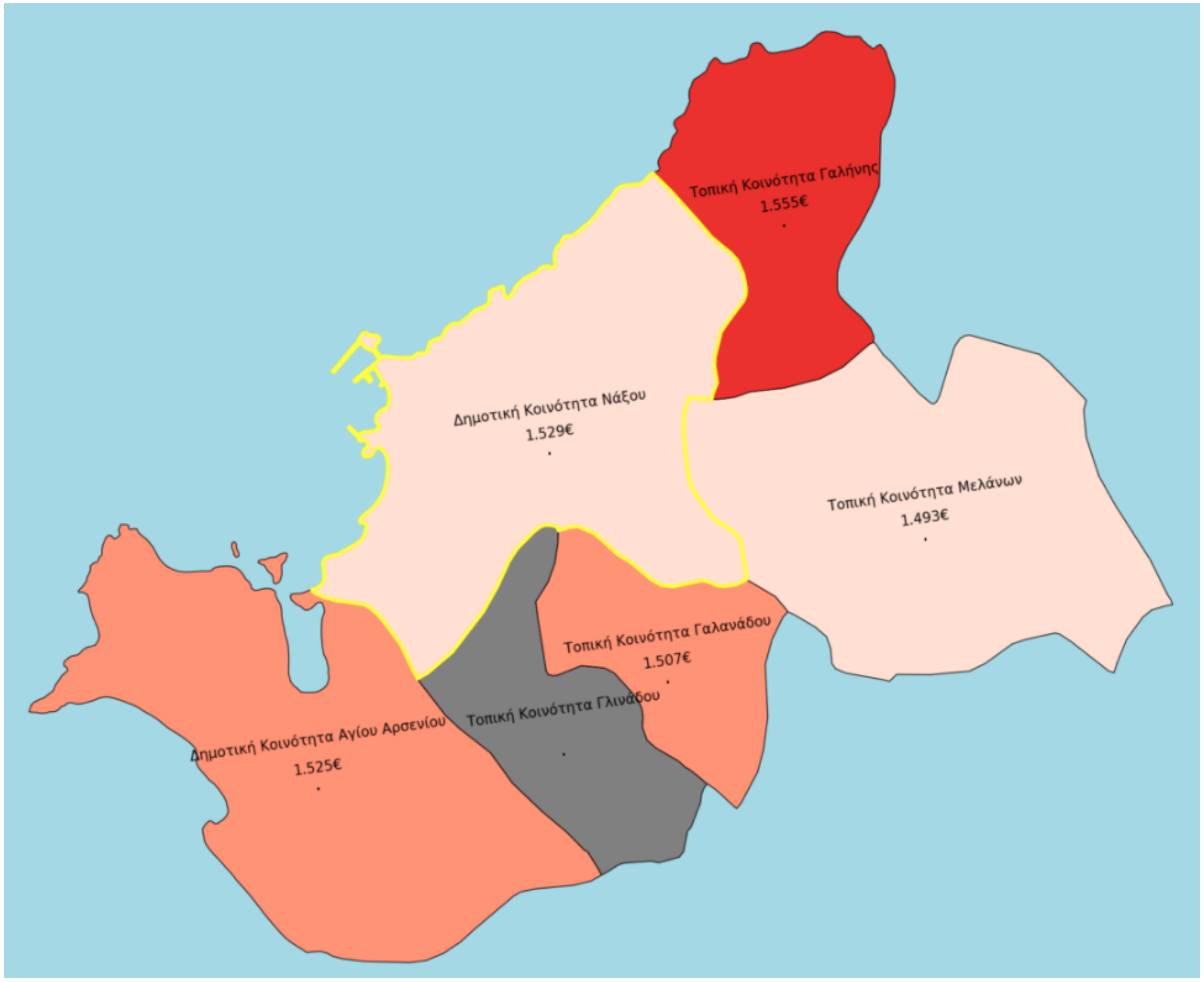


Figure B.4: Clustering of "Δημοτική Κοινότητα Νάξου" island community, that belongs to island of "Νάξος", for fuel type "Diesel". Average fuel prices per community, annotated.

B.3 Clustering of Islands per Island Complex

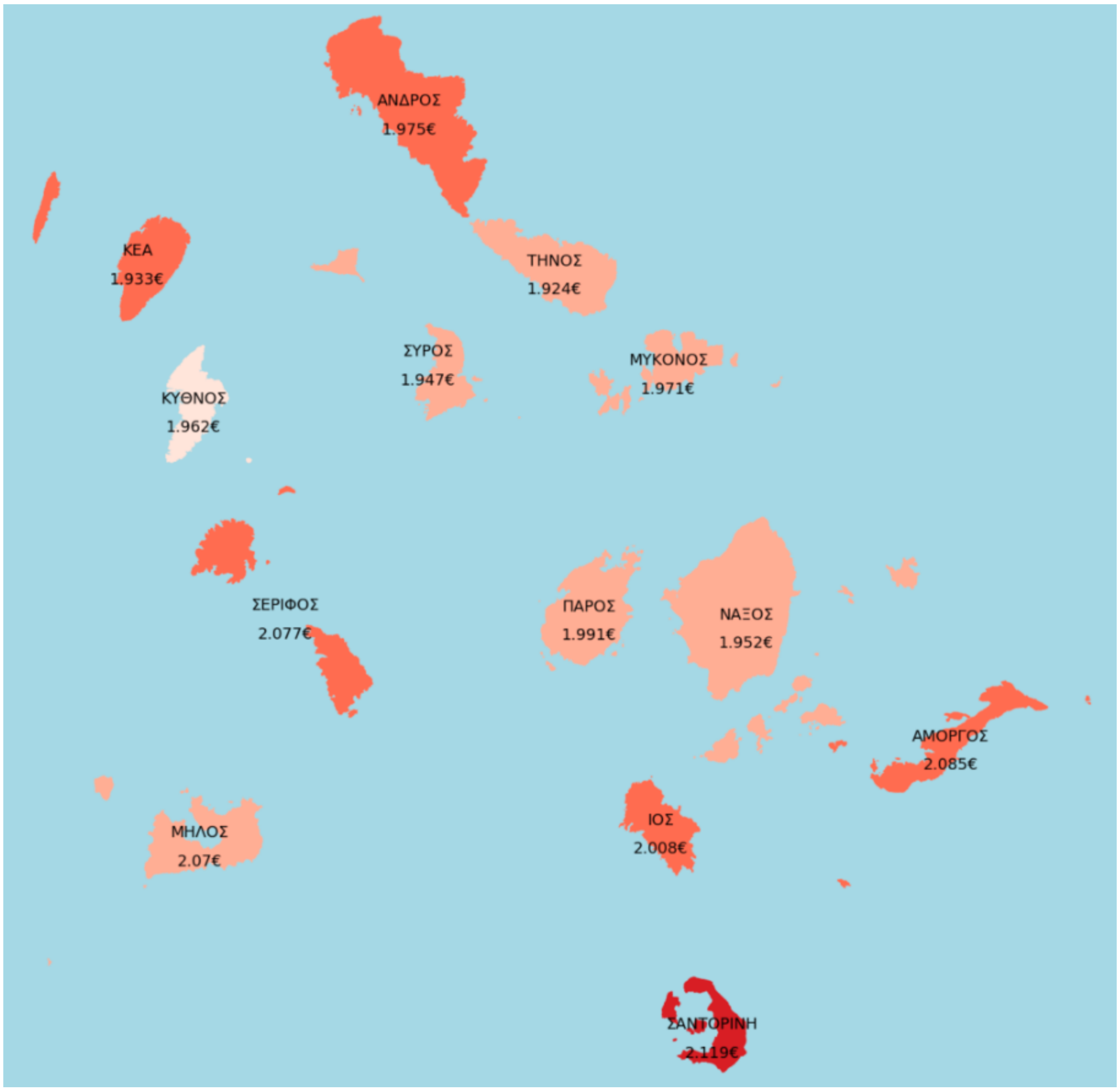


Figure B.5: Clustering of "Κυκλάδες" island complex, for fuel type "Unleaded 98/100". Average fuel prices per island, annotated.

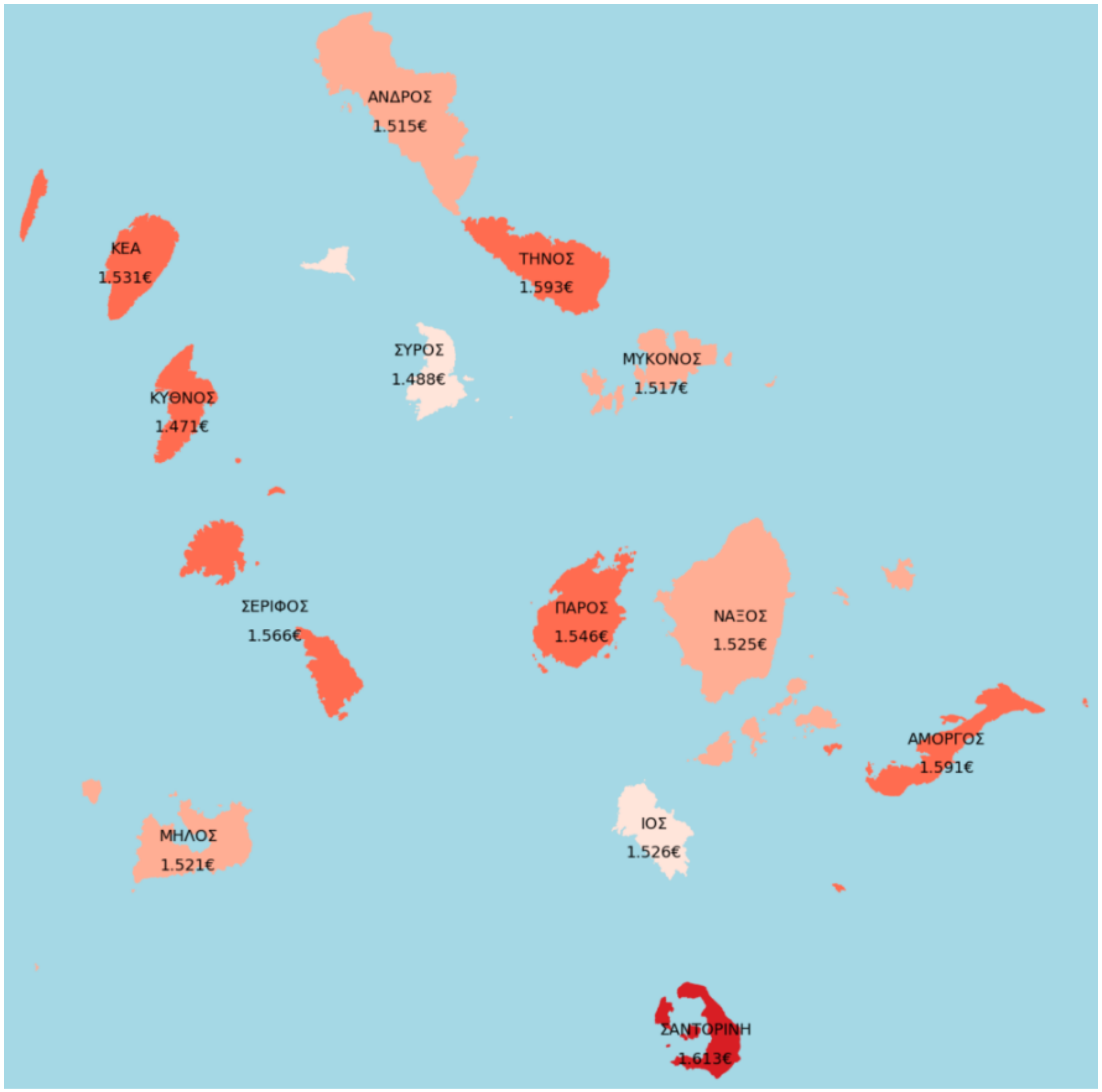


Figure B.6: Clustering of "Κυκλάδες" island complex, for fuel type "diesel". Average fuel prices per island, annotated.

B.4 Clustering of Complex of Islands

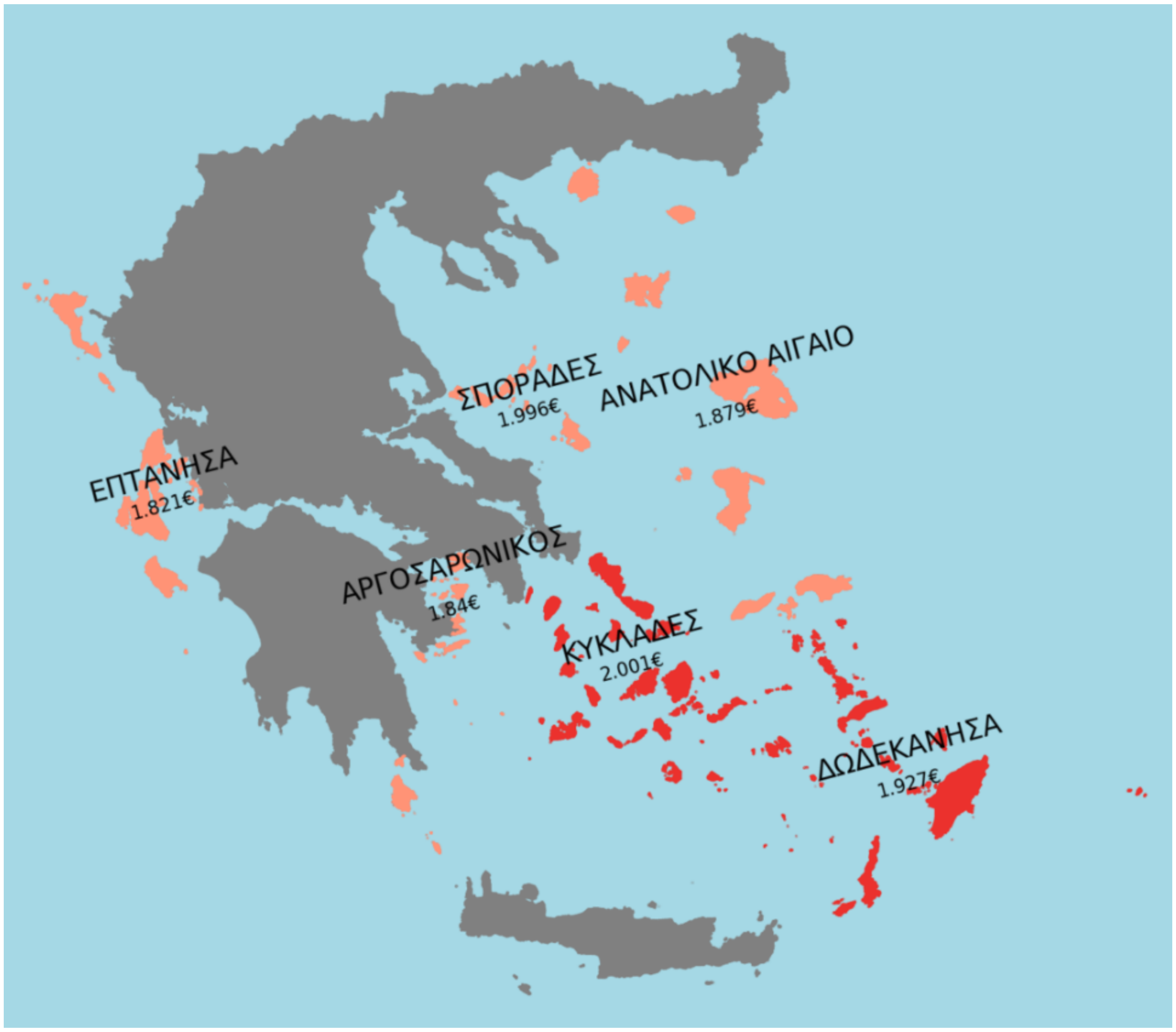


Figure B.7: Clustering of all island complexes, for fuel type "Unleaded 98/100". Average fuel prices per island complex, annotated.

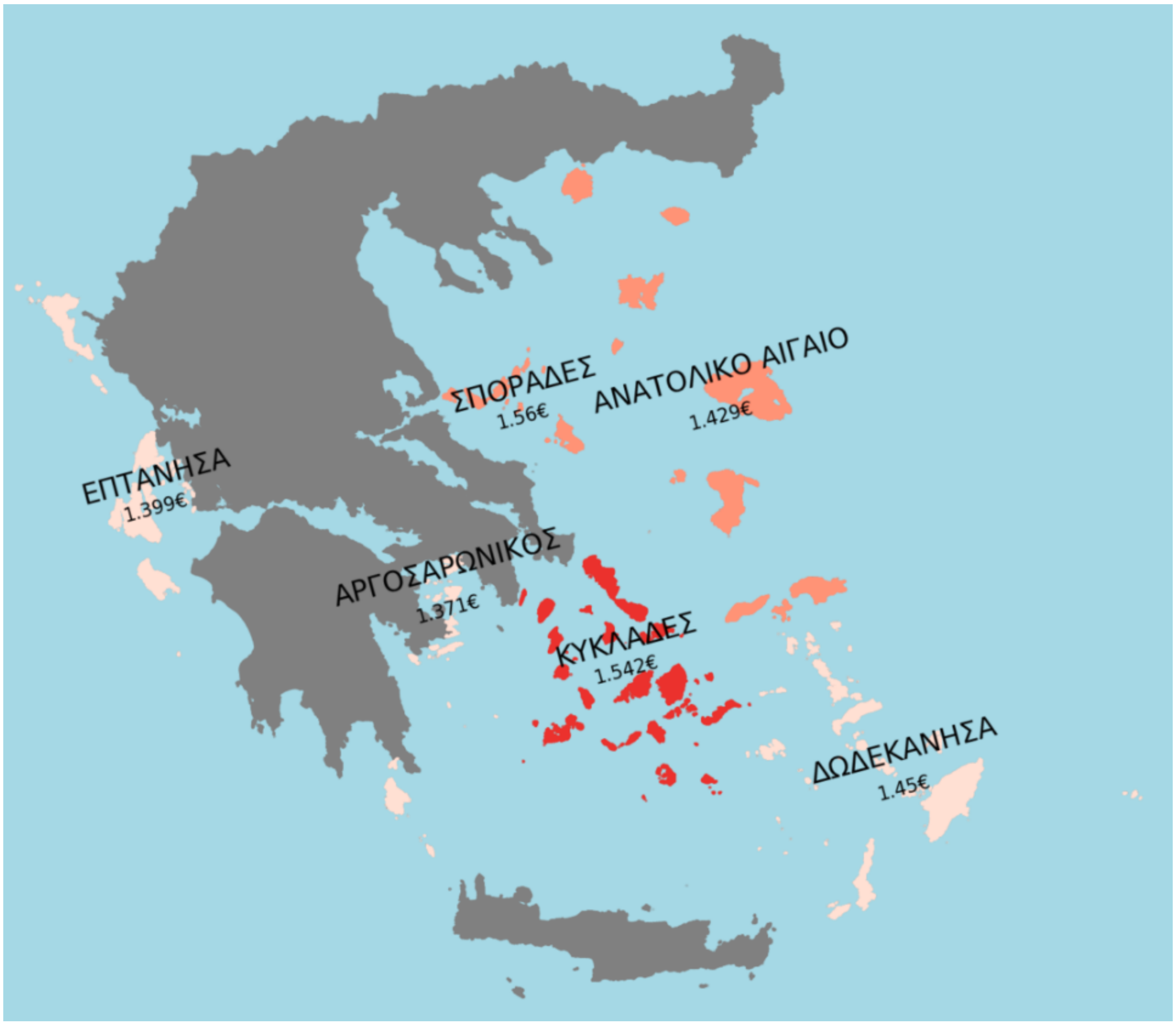


Figure B.8: Clustering of all island complexes, for fuel type "Diesel". Average fuel prices per island complex, annotated.

B.5 Clustering of Communities per Prefecture

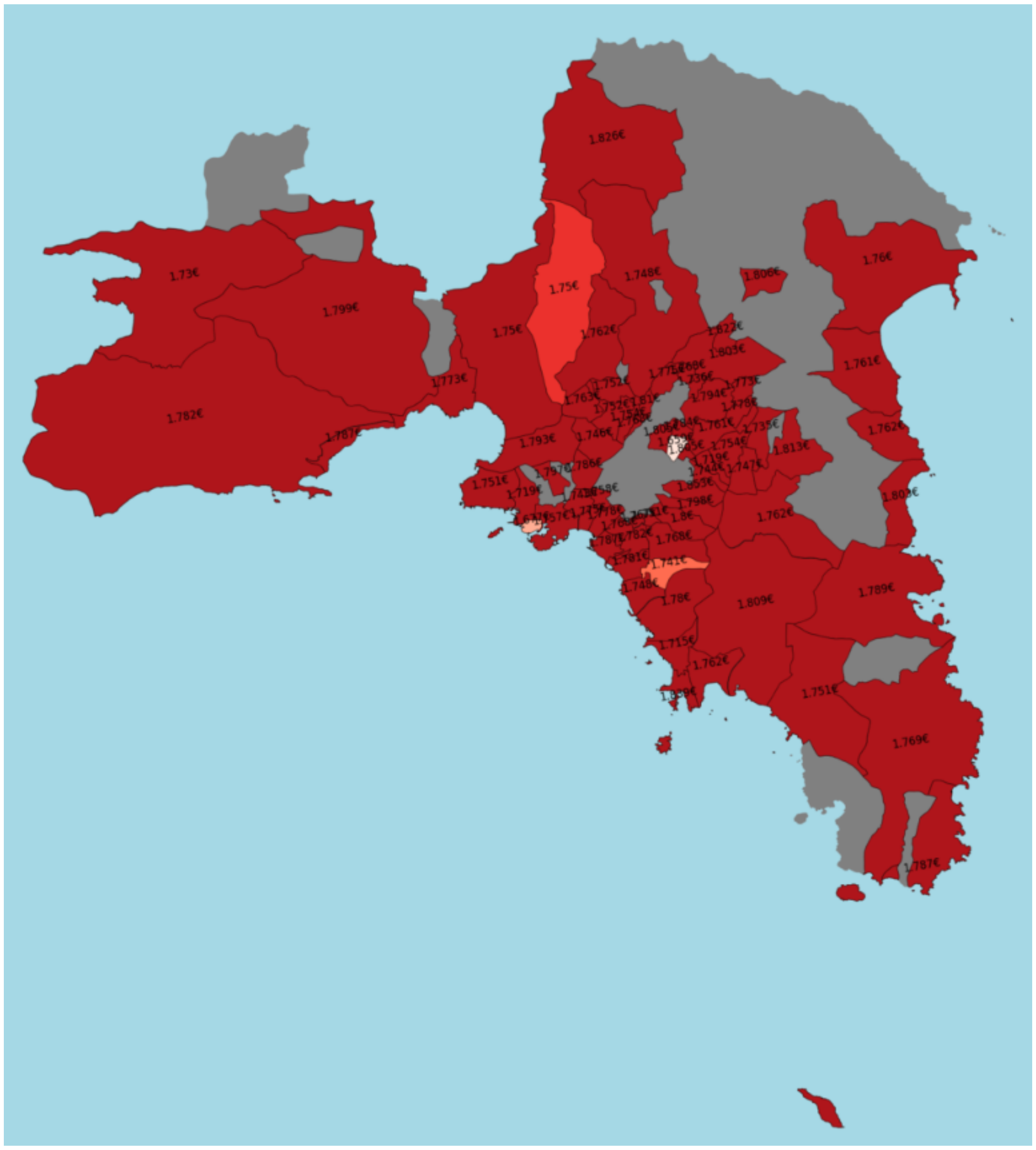


Figure B.9: Clustering of "Νομός Αττικής' prefecture, for fuel type "Unleaded 98/100". Average fuel prices per community, annotated

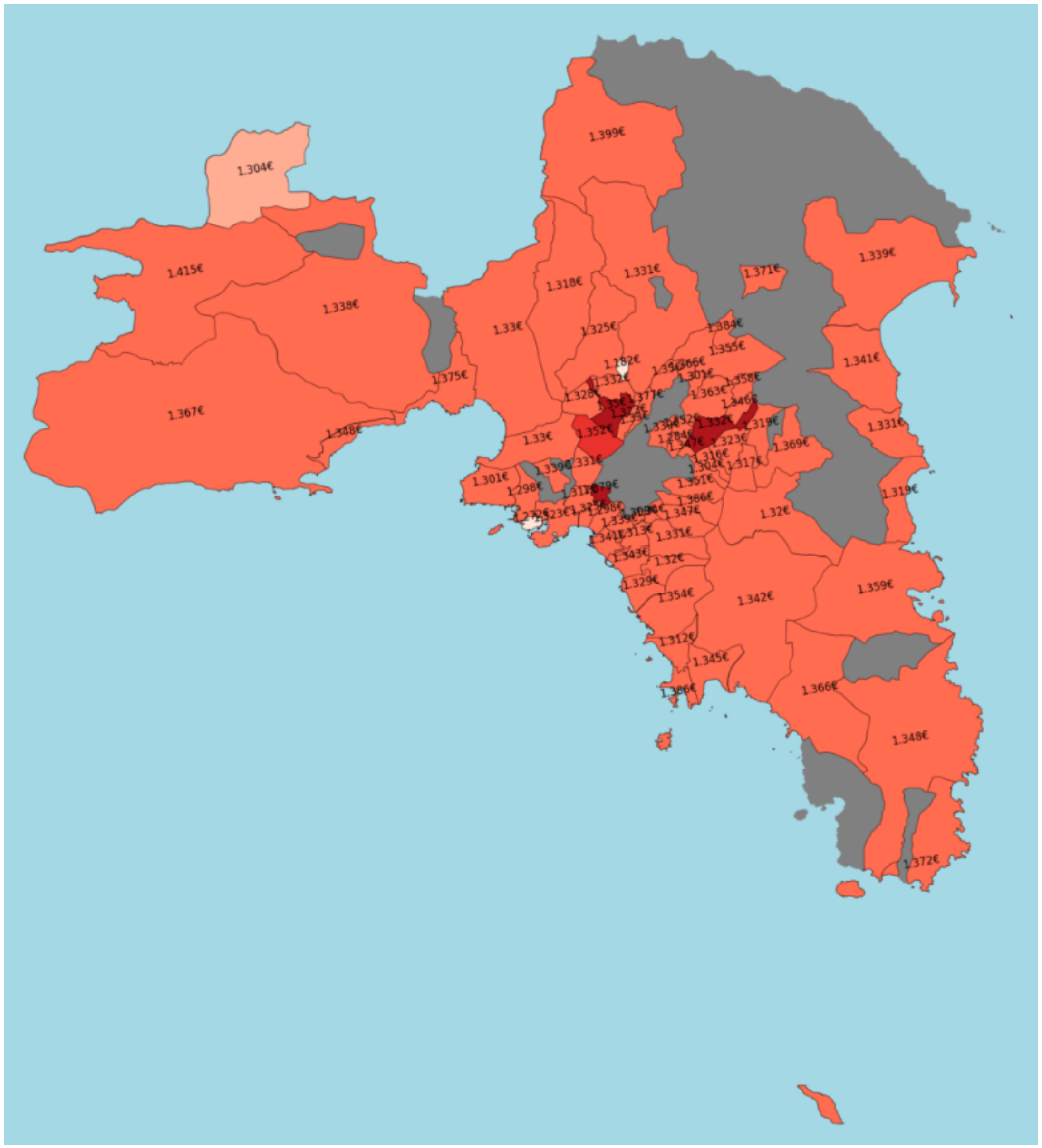


Figure B.10: Clustering of "Νομός Αττικής' prefecture, for fuel type "Diesel". Average fuel prices per community, annotated

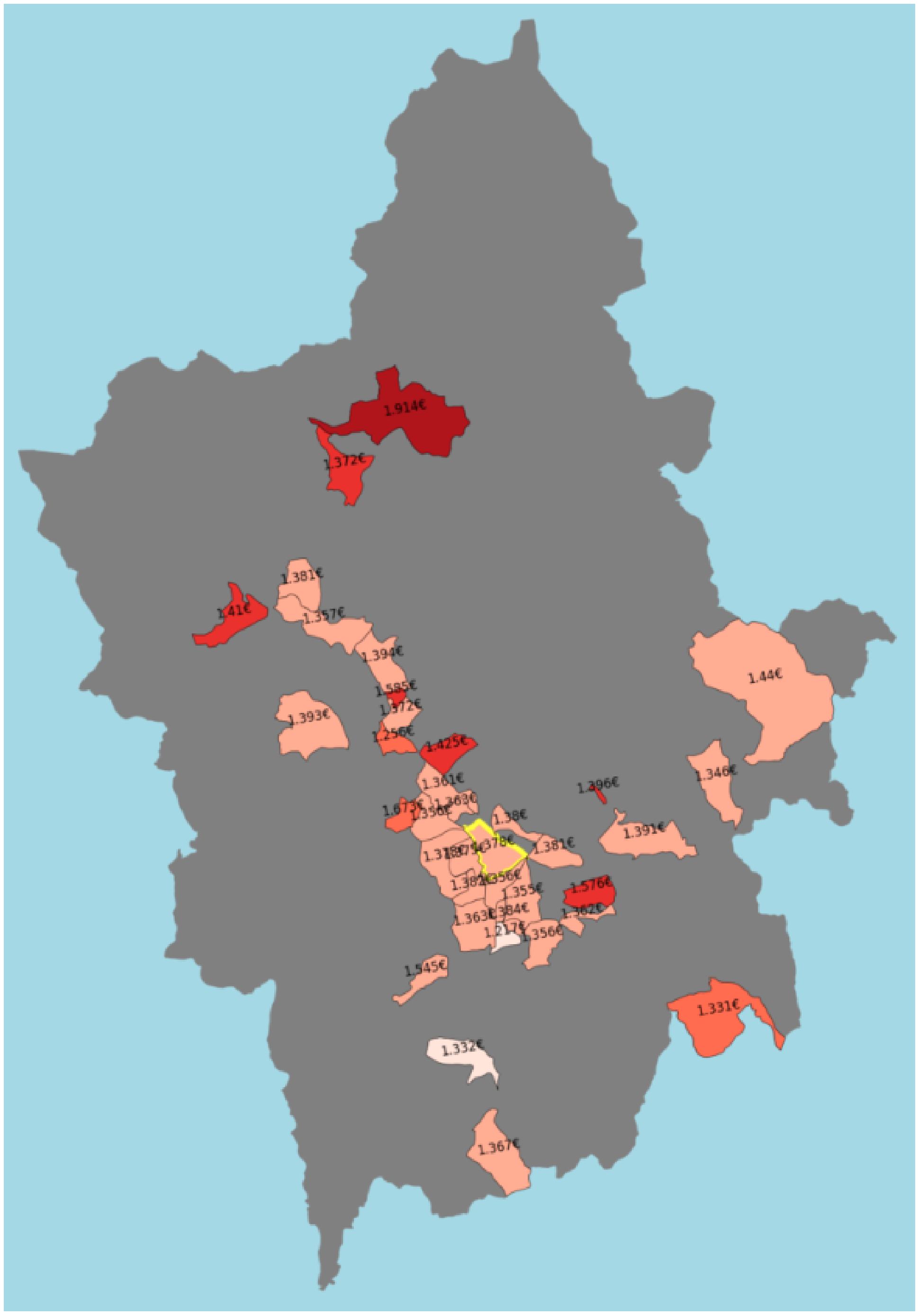


Figure B.11: Clustering of ”Νομός Ιωαννίνων’ prefecture, for fuel type ”Diesel”. Average fuel prices per community, annotated

Appendix C

Additional Unit Root Test Results

C.1 Diesel

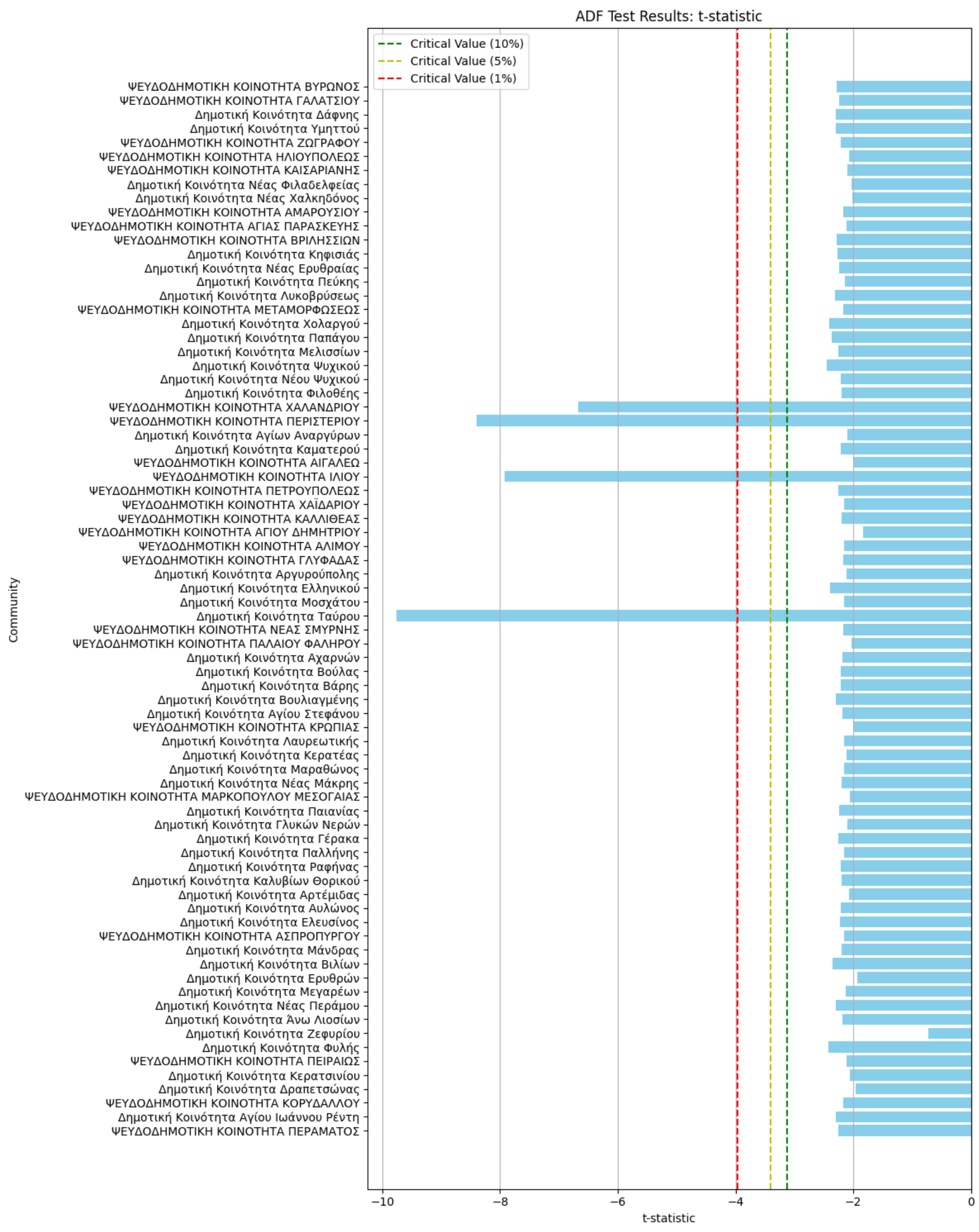


Figure C.1: t-statistic derived from the unit root test, for fuel type ”Diesel”.

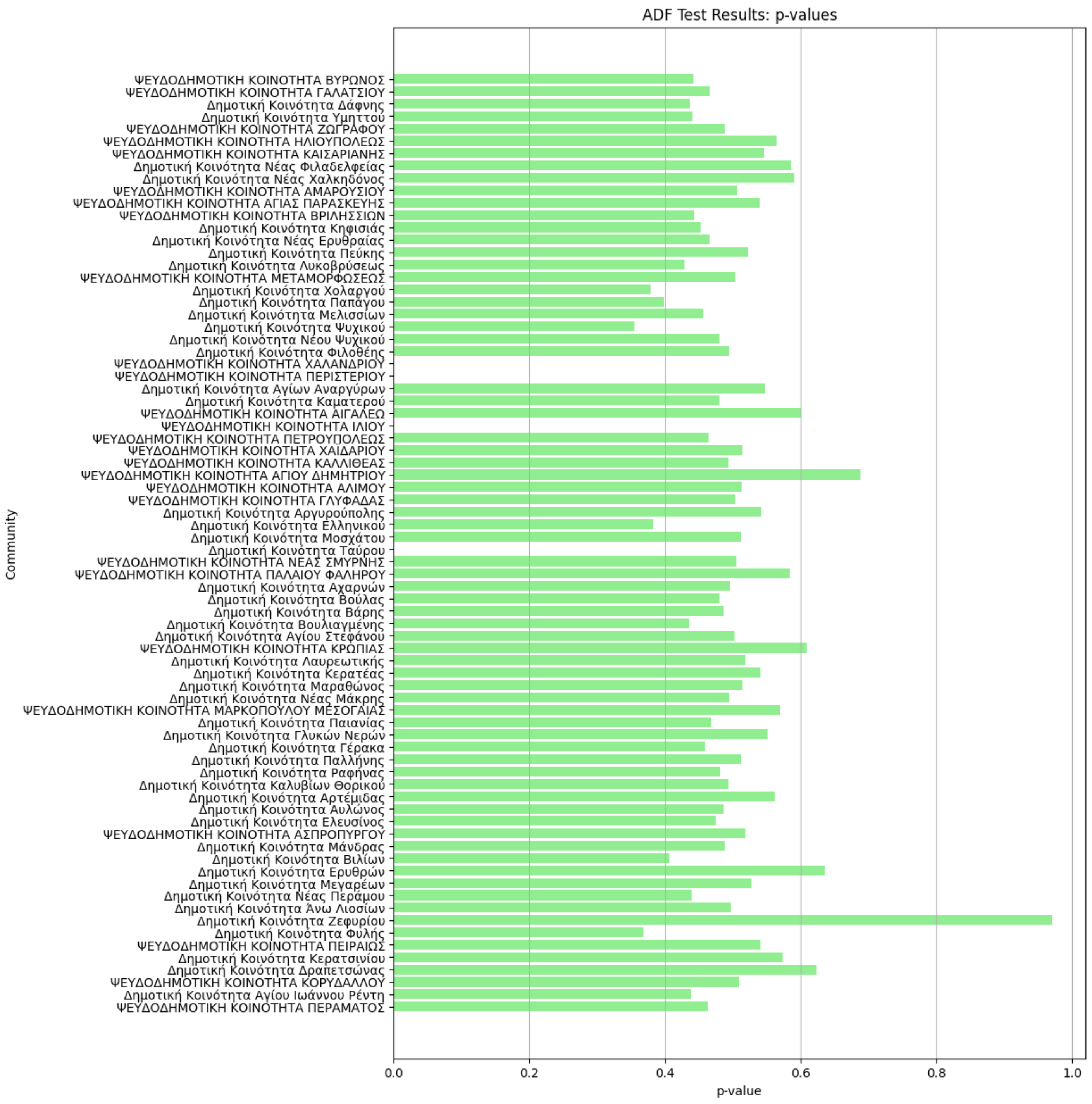


Figure C.2: p-value derived from the unit root test, for fuel type ”Diesel”.

Appendix D

Additional Cointegration Test Results

D.1 Diesel

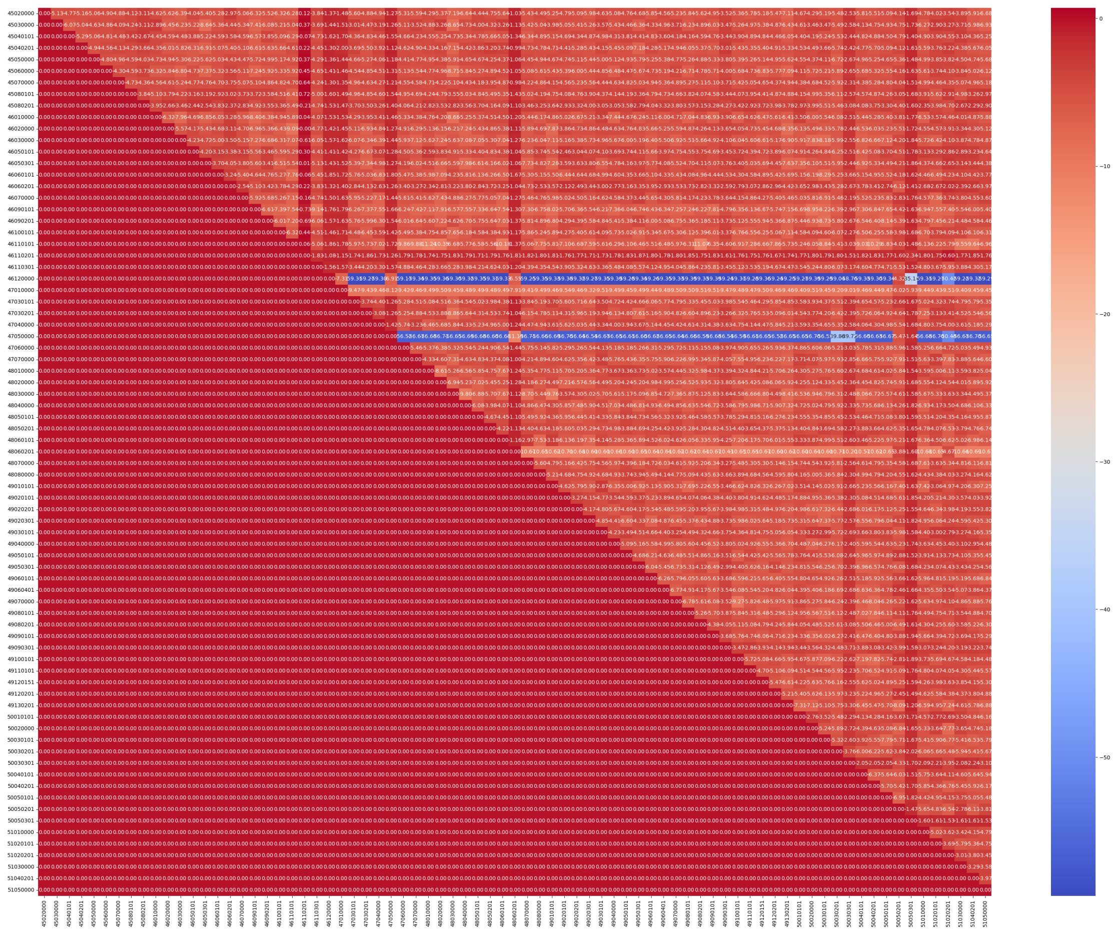


Figure D.1: t-statistic from cointegration test, for fuel type "Diesel"

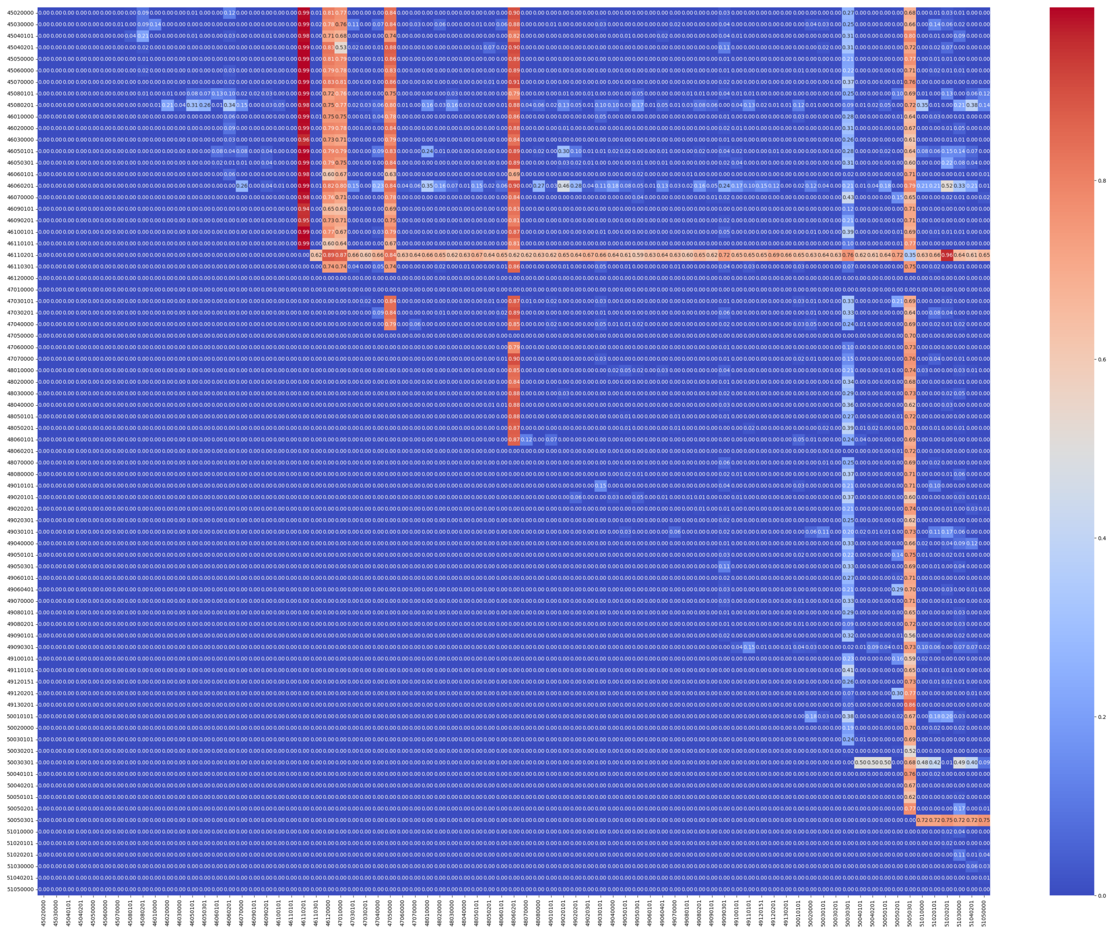


Figure D.2: P-value from cointegration test, for fuel type "Diesel"

Appendix E

Python Scripts

E.1 Script Used for Matching Community Codes with Gas Stations

```
import os
import csv
import pandas as pd
import numpy as np

dd_01_codes = pd.read_excel("mit_2001.xls",
  sheet_name='Mun-DD. (LAU 2)', dtype=str)
dd_01_codes['ddNormalName'] = dd_01_codes['
  ddNormalName'].str.replace(r'(D\.D\.)', r'\1
  ', regex=True)
dd_01_codes = dd_01_codes.loc[:,["ddCode","
  ddNormalName","municipalityName"]]

gas_stations = pd.read_excel("/
  gasStations_fuelGR.xlsx", dtype=str)
gas_stations = gas_stations.loc[:,["gasStationID
  ","ddNormalName","municipalityName"]]

gas_stations_x_ddCode01 = gas_stations.merge(
  dd_01_codes, on=['municipalityName', '
  ddNormalName'], how='left')
gas_stations_x_ddCode01.rename(columns={'
  ddNormalName': 'ddName_01','municipalityName
  ':'municipalityName_01', 'ddCode': 'KOD_01' },
   inplace=True)

kod01_x_kod11 = pd.read_excel("KOD_01_11.xlsx",
  dtype=str)
```

```
kod01_x_kod11 = kod01_x_kod11.loc[:,["KOD01","
  KOD11"]]
kod01_x_kod11.rename(columns={'KOD01': 'KOD_01
  ','KOD11':'KOD_11'}, inplace=True)

gas_stations_x_ddCode01_x_ddCode11 =
  gas_stations_x_ddCode01.merge(kod01_x_kod11,
  on=['KOD_01'], how='left')

dd_11_codes = pd.read_excel("mit_2011.xlsx",
  dtype=str)

eight_digit_pattern = r'^\d{8}$'  # Regex
  pattern for 8-digit numbers
filtered_df = dd_11_codes.copy()
filtered_df = filtered_df[filtered_df['KOD11'].
  str.match(eight_digit_pattern, na=False)]
filtered_df['municipalityCode_11'] = filtered_df
  ['KOD11'].str[:4]

temp_dd11_df = dd_11_codes.copy()
temp_dd11_df = temp_dd11_df.loc[:,["KOD11","
  LEKTM11"]]
temp_dd11_df.rename(columns={'KOD11': '
  municipalityCode_11'}, inplace=True)

filtered_df = filtered_df.merge(temp_dd11_df, on
  =['municipalityCode_11'], how='left')
filtered_df.rename(columns={'LEKTM11_x': '
  ddName_11','LEKTM11_y':'municipalityName_11',
  'totPop':'dd_population_11'}, inplace=True)
filtered_df['municipalityCode+01'] = filtered_df
  ['KOD11'] + '01'
filtered_df.drop(columns=['rowID','LEV11','mPop
  ','fPop','OtaSerNo','lat','lon'], inplace=True
  )

temp_dd11_df = dd_11_codes.copy()
temp_dd11_df = temp_dd11_df.loc[:,["KOD11","lat
  ",'lon']]
temp_dd11_df.rename(columns={'KOD11':'
  municipalityCode+01'}, inplace=True)

filtered_df = filtered_df.merge(temp_dd11_df, on
  =['municipalityCode+01'], how='left')
filtered_df.rename(columns={'KOD11':'KOD_11'},
  inplace=True)
filtered_df.drop(columns=['municipalityCode
  +01'], inplace=True)

temp_dd11_df = dd_11_codes.copy()
temp_dd11_df = temp_dd11_df.loc[:,["LEKTM11",'
  totPop']]
temp_dd11_df.rename(columns={'LEKTM11':'
  municipalityName_11'}, inplace=True)

filtered_df = filtered_df.merge(temp_dd11_df, on
  =['municipalityName_11'], how='left')
filtered_df.rename(columns={'totPop':'
  municipality_population_11','lat':'lat_11','
  lon':'lon_11'}, inplace=True)

dd_11_codes = filtered_df.copy()

gas_stations_x_ddCode01_x_ddCode11 =
  gas_stations_x_ddCode01_x_ddCode11.merge(
  dd_11_codes, on=['KOD_11'], how='left')

dd_21_codes = pd.read_excel("mit_2021.xlsx",
  dtype=str)
```

```
dd_21_codes.rename(columns={'G.K. 2021':'KOD_21
   ', 'PERIGRAFH': 'ddName_21', 'MONIMOS
   PLHTHISMOS ':'population_21'}, inplace=True)
dd_21_codes.drop(columns=['A/A'], inplace=True)

eight_digit_pattern = r'^\d{8}$'  # Regex
   pattern for 8-digit numbers
filtered_df = dd_21_codes.copy()
filtered_df = filtered_df[filtered_df['KOD_21'].
   str.match(eight_digit_pattern, na=False)]
filtered_df.rename(columns={'population_21':'
   dd_population_21'}, inplace=True)
filtered_df['municipalityCode'] = filtered_df['
   KOD_21'].str[:4]

temp_dd21_df = dd_21_codes.copy()
temp_dd21_df = temp_dd21_df.loc[:,["KOD_21","
   ddName_21"]]
temp_dd21_df.rename(columns={'KOD_21': '
   municipalityCode', 'ddName_21':'
   municipalityName_21'}, inplace=True)

filtered_df = filtered_df.merge(temp_dd21_df, on
   =['municipalityCode'], how='left')

temp_dd21_df = pd.read_excel("mit_2021.xlsx",
   dtype=str)
temp_dd21_df.rename(columns={'G.K. 2021':'KOD_21
   ', 'PERIGRAFH': 'ddName_21', 'MONIMOS
   PLHTHISMOS ':'population_21'}, inplace=True)
temp_dd21_df.rename(columns={'KOD_21': '
   municipalityCode','population_21':'
   municipality_population_21'}, inplace=True)
temp_dd21_df = temp_dd21_df.loc[:,["
   municipalityCode","municipality_population_21
```

```
  "]]

filtered_df = filtered_df.merge(temp_dd21_df, on
  =['municipalityCode'], how='left')

dd_21_codes = filtered_df.copy()
dd_21_codes.rename(columns={'municipalityCode':
  'municipalityCode_21'}, inplace=True)
dd_21_codes

kod11_x_kod21 = pd.read_excel("koinotites
  -2011-2021.xlsx", dtype=str)
kod11_x_kod21 = kod11_x_kod21.loc[:,["KOD11_7","
  KOD21_7"]]
kod11_x_kod21.rename(columns={'KOD11_7': 'KOD_11
  ', 'KOD21_7':'KOD_21'}, inplace=True)

gas_stations_x_ddCode01_x_ddCode11_x_ddCode21 =
  gas_stations_x_ddCode01_x_ddCode11.merge(
  kod11_x_kod21, on=['KOD_11'], how='left')

gas_stations_x_ddCode01_x_ddCode11_x_ddCode21

final_df =
  gas_stations_x_ddCode01_x_ddCode11_x_ddCode21.
  merge(dd_21_codes, on=['KOD_21'], how='left')

csv_file_path = 'gas_stations_x_codes.xlsx'
final_df.to_excel(csv_file_path, index=False,
  encoding='utf-8')
```

E.2 Script Used for Times Series Clustering Per Community

```
import matplotlib.pyplot as plt
import geopandas as gpd
import pandas as pd
import numpy as np

from tslearn.preprocessing import
    TimeSeriesScalerMeanVariance
from tslearn.utils import to_time_series_dataset
from tslearn.clustering import TimeSeriesKMeans

df = pd.read_csv('../data/gas_stations_x_codes.
    csv')
df.dropna(subset=['KOD_01'], inplace=True)

gas_stations = df.loc[:,["gasStationID","KOD_21
    "]]

fuels = pd.read_csv("../data/times_prathriwn.csv
    ")

gas_price_ddcode = fuels.merge(gas_stations, on
    =['gasStationID'], how='left')
gas_price_ddcode = gas_price_ddcode.dropna()
gas_price_ddcode['dateDefined'] = pd.to_datetime
    (gas_price_ddcode['dateDefined'], errors='
    coerce')
gas_price_ddcode = gas_price_ddcode.dropna(
    subset=['dateDefined'])
gas_price_ddcode['dateDefined'] =
    gas_price_ddcode['dateDefined'].dt.date

def clustering_per_gas_type(gas_prices,gdf,
fuelTypeID):
  df_x_fuelType = gas_prices[gas_prices["
    fuelTypeID"] == fuelTypeID]
  df_x_fuelType = df_x_fuelType.reset_index()
  df_x_fuelType.drop(columns=['jointFuelTypeID
    ','isPremium','fuelTypeID','index'],
    inplace=True)

  df_x_fuelType['dateDefined'] = pd.
    to_datetime(df_x_fuelType['dateDefined'])
     # Ensure 'dateDefined' is in datetime
    format
  df_x_fuelType.sort_values(['gasStationID', '
    dateDefined'], inplace=True)  # Sort the
    DataFrame
  df_x_fuelType = df_x_fuelType.groupby(['
    gasStationID', 'dateDefined']).mean().
    reset_index()
  def fill_missing_dates(group):
      return group.set_index('dateDefined').
        resample('D').ffill().reset_index()
  df_x_fuelType_imputed = df_x_fuelType.
    groupby('gasStationID').apply(
    fill_missing_dates)
  df_x_fuelType_imputed.reset_index(drop=True,
     inplace=True)
  df_x_fuelType_imputed['KOD_21'] =
    df_x_fuelType_imputed['KOD_21'].astype(int
    ).astype(str).str.zfill(8)

  fuelPrices_per_dd = df_x_fuelType_imputed.
    groupby(['KOD_21', 'dateDefined'])['
    fuelPrice'].mean().reset_index()

  unique_dd_codes_per_fuelTypeID =
  fuelPrices_per_dd["KOD_21"].unique()
geospatial_df = gdf[gdf["KALCODE"].isin(
  unique_dd_codes_per_fuelTypeID)]

geospatial_df = geospatial_df.set_geometry('
  geometry')
gdf_sindex = geospatial_df.sindex
def find_neighbors(row, gdf_sindex):
    possible_matches_index = list(gdf_sindex
      .intersection(row['geometry'].bounds))
    possible_matches = geospatial_df.iloc[
      possible_matches_index]
    precise_matches = possible_matches[
      possible_matches.intersects(row['
      geometry'])]
    return precise_matches['KALCODE'].tolist
      ()
geospatial_df['neighbors'] = geospatial_df.
  apply(lambda row: find_neighbors(row,
  gdf_sindex), axis=1)

geospatial_df = geospatial_df.set_geometry('
  geometry')
gdf_sindex = geospatial_df.sindex
def find_neighbors(row, gdf_sindex):
    possible_matches_index = list(gdf_sindex
      .intersection(row['geometry'].bounds))
    possible_matches = geospatial_df.iloc[
      possible_matches_index]
    precise_matches = possible_matches[
      possible_matches.intersects(row['
      geometry'])]
    return precise_matches['KALCODE'].tolist
      ()
geospatial_df['neighbors'] = geospatial_df.
```

```
    apply(lambda row: find_neighbors(row,
    gdf_sindex), axis=1)

print("Starting Clustering Process:")

results = {}

for dd_code in
    unique_dd_codes_per_fuelTypeID:

    try:
        temp_dd = geospatial_df[
            geospatial_df["KALCODE"] ==
            dd_code]
        neighbors = temp_dd["neighbors"].
            values

        time_series_list = []
        dd_names = []

        if neighbors.shape[0] != 0:  #
            municipalities without neighbors
            are excluded from the procedure

            print("*****************")
            print("Target :", dd_code)

            for dd in neighbors[0]:
                dd_names.append(dd)
                temp_df = fuelPrices_per_dd[
                    fuelPrices_per_dd["KOD_21
                    "] == dd]
                temp_df_gas_prices = temp_df
                    ["fuelPrice"].values
                time_series_list.append(
                        temp_df_gas_prices)

                max_length = max(len(ts) for ts
                   in time_series_list)
                time_series_padded = [np.pad(ts,
                    (0, max_length - len(ts)), '
                   constant', constant_values=np.
                   nan) for ts in
                   time_series_list]

                X = to_time_series_dataset(
                   time_series_padded)
                X = TimeSeriesScalerMeanVariance
                   ().fit_transform(X)
                model = TimeSeriesKMeans(
                   n_clusters=3, metric="dtw",
                   max_iter=5)

                y = model.fit_predict(X)

                results[dd_code] = y

        except ValueError as e:
            print(f"Error for {dd_code}: {e}")
            continue  # Continue with the next
               iteration

    return(geospatial_df,results)

shapefile_path = 'TOP_DHM_KOIN.shp'
gdf = gpd.read_file(shapefile_path)

geospatial_df, results = clustering_per_gas_type
   (gas_price_ddcode,gdf,2) ## 1 --> Unleaded 95
   ## 2 --> Unleaded 98/100 ### Diesel --> 4
```

```
df = pd.DataFrame.from_dict(results, orient='
    index')
df['Result'] = df.apply(lambda row: list(map(int
    , filter(lambda x: not np.isnan(x), np.hstack(
    row)))), axis=1)
df = df.drop(df.columns[:-1], axis=1)

df.to_csv("results_unleaded_95.csv")
```

E.3 Script Used for Times Series Clustering Per Island Complex

```
import matplotlib.pyplot as plt
import geopandas as gpd
import pandas as pd
import numpy as np

from tslearn.preprocessing import
  TimeSeriesScalerMeanVariance
from tslearn.utils import to_time_series_dataset
from tslearn.clustering import TimeSeriesKMeans

from sklearn.preprocessing import LabelEncoder

df = pd.read_csv('../data/gas_stations_x_codes.
  csv')
df = df.loc[:,["gasStationID","KOD_01","KOD_11
  ","KOD_21"]]
df.dropna(subset=['KOD_01'], inplace=True)

gas_stations = df.loc[:,["gasStationID","KOD_21
  "]]

fuels = pd.read_csv("times_prathrion.csv")

gas_price_ddcode = fuels.merge(gas_stations, on
  =['gasStationID'], how='left')
gas_price_ddcode = gas_price_ddcode.dropna()
gas_price_ddcode['dateDefined'] = pd.to_datetime
  (gas_price_ddcode['dateDefined'], errors='
  coerce')
gas_price_ddcode = gas_price_ddcode.dropna(
  subset=['dateDefined'])
gas_price_ddcode['dateDefined'] =
  gas_price_ddcode['dateDefined'].dt.date

islands_df = pd.read_excel("../data/islands.xlsx
  ",dtype=str)
islands_df = islands_df.loc[:,["KALCODE","
  ISLAND_NAME","ISLAND_COMPLEX"]]
islands_df = islands_df.rename(columns = {"
  KALCODE" : "KOD_21"})
islands_df["KOD_21"] = islands_df["KOD_21"].
  astype(float)

complex_to_number = {
    "EUBOIA":0,
    "KRHTH":1,
    "SPORADES":2,
    "ANATOLIKO AIGAIO":3,
    "DWDEKANHSA":4,
    "KYKLADES":5,
    "ARGOSARONIKOS":6,
    "EPTANHSA":7
}

final_df = pd.merge(gas_price_ddcode,islands_df,
  on="KOD_21",how="left")
final_df = final_df.dropna()
final_df['ISLAND_COMPLEX'] = final_df['
  ISLAND_COMPLEX'].map(complex_to_number)

label_encoder = LabelEncoder()
final_df['ISLAND_NAME'] = label_encoder.
  fit_transform(final_df['ISLAND_NAME'])

def clustering_per_gas_type(gas_prices,gdf,
  fuelTypeID,islands_df):
    df_x_fuelType = gas_prices[gas_prices["
    fuelTypeID"] == fuelTypeID]
df_x_fuelType = df_x_fuelType.reset_index()
df_x_fuelType.drop(columns=['jointFuelTypeID
    ','isPremium','fuelTypeID','index'],
    inplace=True)

df_x_fuelType['dateDefined'] = pd.
    to_datetime(df_x_fuelType['dateDefined'])
     # Ensure 'dateDefined' is in datetime
    format
df_x_fuelType.sort_values(['gasStationID', '
    dateDefined'], inplace=True)  # Sort the
    DataFrame

df_x_fuelType = df_x_fuelType.groupby(['
    gasStationID', 'dateDefined']).mean().
    reset_index()

def fill_missing_dates(group):
    return group.set_index('dateDefined').
        resample('D').ffill().reset_index()
df_x_fuelType_imputed = df_x_fuelType.
    groupby('gasStationID').apply(
    fill_missing_dates)
df_x_fuelType_imputed.reset_index(drop=True,
     inplace=True)
df_x_fuelType_imputed['KOD_21'] =
    df_x_fuelType_imputed['KOD_21'].astype(int
    ).astype(str).str.zfill(8)

fuelPrices_per_dd = df_x_fuelType_imputed.
    groupby(['ISLAND_COMPLEX','dateDefined', "
    ISLAND_NAME"])['fuelPrice'].mean().
    reset_index()
```

```
unique_island_complex = fuelPrices_per_dd["
  ISLAND_COMPLEX"].unique()

results= {}  # K-means results
average_prices_per_island = {} # Islands
  with their respective average prices
names_per_complex = {} # Complexes with all
  available islands

for complex in unique_island_complex:

    try:
        print(complex)
        if complex == 0 or complex == 1:
            continue

        time_series_list = []
        temp_names = []

        temp_df = fuelPrices_per_dd[
          fuelPrices_per_dd["ISLAND_COMPLEX
          "] == complex]

        temp_unique_islands = temp_df["
          ISLAND_NAME"].unique()
        for island in temp_unique_islands:
            temp_island_df =
              fuelPrices_per_dd[
              fuelPrices_per_dd["ISLAND_NAME
              "] == island]
            temp_df_gas_prices =
              temp_island_df["fuelPrice"].
              values

            print(island,temp_df_gas_prices.
                mean())

            time_series_list.append(
                temp_df_gas_prices)

            average_prices_per_island[island
                ] = temp_df_gas_prices.mean()

            temp_names.append(island)

        names_per_complex[complex] =
            temp_names

        max_length = max(len(ts) for ts in
            time_series_list)
        time_series_padded = [np.pad(ts, (0,
             max_length - len(ts)), 'constant
            ', constant_values=np.nan) for ts
            in time_series_list]

        X = to_time_series_dataset(
            time_series_padded)
        X = TimeSeriesScalerMeanVariance().
            fit_transform(X)
        model = TimeSeriesKMeans(n_clusters
            =4, metric="dtw", max_iter=5)

        y = model.fit_predict(X)

        results[complex] = y

    except ValueError as e:
        print(f"Error for : {e}")
        continue
    return(results,average_prices_per_island,
      names_per_complex)

shapefile_path = 'TOP_DHM_KOIN.shp'
gdf = gpd.read_file(shapefile_path)

results, average_prices_per_island,
  names_per_complex = clustering_per_gas_type(
  final_df,gdf,1,islands_df) ## 1 --> Unleaded
  95 ## 2 --> Unleaded 98/100 ### Diesel --> 4

df = pd.DataFrame(list(results.items()), columns
  =['ISLAND_COMPLEX', 'RESULT'])
df['ISLAND_COMPLEX'] = df['ISLAND_COMPLEX'].map
  ({v: k for k, v in complex_to_number.items()})

df_names = pd.DataFrame(list(names_per_complex.
  items()), columns=['ISLAND_COMPLEX', 'Names'])
df_names['ISLAND_COMPLEX'] = df_names['
  ISLAND_COMPLEX'].map({v: k for k, v in
  complex_to_number.items()})
df_results_and_names = pd.merge(df,df_names,on="
  ISLAND_COMPLEX",how="left")

average_prices_per_island
df_prices = pd.DataFrame(list(
  average_prices_per_island.items()), columns=['
  ISLAND_NAME', 'AVERAGE_PRICE'])

temp_final_df = final_df
temp_final_df['ISLAND_NAME_2'] = label_encoder.
  inverse_transform(temp_final_df['ISLAND_NAME
  '])
temp_final_df = temp_final_df.loc[:,["
```

```
    ISLAND_NAME","ISLAND_NAME_2","ISLAND_COMPLEX
    "]]

df_prices_final = pd.merge(df_prices,
    temp_final_df,on="ISLAND_NAME",how='left')
df_prices_final['ISLAND_COMPLEX'] =
    df_prices_final['ISLAND_COMPLEX'].map({v: k
    for k, v in complex_to_number.items()})

df_prices_final = df_prices_final.
    drop_duplicates()
df_prices_final = df_prices_final.reset_index()

final_dictionary_names = {}
final_dictionary_prices = {}

complexes = df_prices_final["ISLAND_COMPLEX"].
    unique()

for complex in complexes:
    temp_names = df_prices_final[df_prices_final
        ["ISLAND_COMPLEX"] == complex]["
        ISLAND_NAME_2"].values
    temp_average_values = df_prices_final[
        df_prices_final["ISLAND_COMPLEX"] ==
        complex]["AVERAGE_PRICE"].values

    final_dictionary_names[complex] = temp_names
    final_dictionary_prices[complex] =
        temp_average_values

df_1 = pd.DataFrame(list(final_dictionary_names.
    items()), columns=['ISLAND_COMPLEX', 'NAMES'])
df_2 = pd.DataFrame(list(final_dictionary_prices
    .items()), columns=['ISLAND_COMPLEX', 'PRICES
    '])
df_final_1 = pd.merge(df_1,df_2,on="
    ISLAND_COMPLEX",how='left')

df_final = pd.merge(df_results_and_names,
    df_final_1,on="ISLAND_COMPLEX",how='left')
df_final = df_final.drop(columns= ["Names"])

df_final.to_excel("complex_island_clustering.
    xlsx")
```

E.4 Script Used for Times Series Clustering of All Island Complexes

```
import matplotlib.pyplot as plt
import geopandas as gpd
import pandas as pd
import numpy as np

from tslearn.preprocessing import
  TimeSeriesScalerMeanVariance
from tslearn.utils import to_time_series_dataset
from tslearn.clustering import TimeSeriesKMeans

df = pd.read_csv('../data/gas_stations_x_codes.
  csv')
df.dropna(subset=['KOD_01'], inplace=True)

df = df.loc[:,["gasStationID","KOD_01","KOD_11
  ","KOD_21"]]

fuels = pd.read_csv("times_prathriwn.csv")

gas_price_ddcode = fuels.merge(gas_stations, on
  =['gasStationID'], how='left')
gas_price_ddcode = gas_price_ddcode.dropna()
gas_price_ddcode['dateDefined'] = pd.to_datetime
  (gas_price_ddcode['dateDefined'], errors='
  coerce')
gas_price_ddcode = gas_price_ddcode.dropna(
  subset=['dateDefined'])
gas_price_ddcode['dateDefined'] =
  gas_price_ddcode['dateDefined'].dt.date

islands_df = pd.read_excel("../data/islands.xlsx
  ",dtype=str)
```

```
islands_df = islands_df.loc[:,["KALCODE","
  ISLAND_COMPLEX"]]
islands_df = islands_df.rename(columns = {"
  KALCODE" : "KOD_21"})
islands_df["KOD_21"] = islands_df["KOD_21"].
  astype(float)

complex_to_number = {
    "EUBOIA":0,
    "KRHTH":1,
    "SPORADES":2,
    "ANATOLIKO AIGAIO":3,
    "DWDEKANHSA":4,
    "KYKLADES":5,
    "ARGOSARONIKOS":6,
    "EPTANHSA":7
}

final_df = pd.merge(gas_price_ddcode,islands_df,
  on="KOD_21",how="left")
final_df = final_df.dropna()
final_df['ISLAND_COMPLEX'] = final_df['
  ISLAND_COMPLEX'].map(complex_to_number)


def clustering_per_gas_type(gas_prices,gdf,
  fuelTypeID):
    df_x_fuelType = gas_prices[gas_prices["
      fuelTypeID"] == fuelTypeID]
    df_x_fuelType = df_x_fuelType.reset_index()
    df_x_fuelType.drop(columns=['jointFuelTypeID
      ','isPremium','fuelTypeID','index'],
      inplace=True)

    df_x_fuelType['dateDefined'] = pd.
      to_datetime(df_x_fuelType['dateDefined'])
       # Ensure 'dateDefined' is in datetime
      format
    df_x_fuelType.sort_values(['gasStationID', '
      dateDefined'], inplace=True)  # Sort the
      DataFrame
    df_x_fuelType = df_x_fuelType.groupby(['
      gasStationID', 'dateDefined']).mean().
      reset_index()

    def fill_missing_dates(group):
        return group.set_index('dateDefined').
            resample('D').ffill().reset_index()
    df_x_fuelType_imputed = df_x_fuelType.
        groupby('gasStationID').apply(
      fill_missing_dates)
    df_x_fuelType_imputed.reset_index(drop=True,
       inplace=True)
    df_x_fuelType_imputed['KOD_21'] =
      df_x_fuelType_imputed['KOD_21'].astype(int
      ).astype(str).str.zfill(8)

    fuelPrices_per_dd = df_x_fuelType_imputed.
      groupby(['ISLAND_COMPLEX', 'dateDefined'])
      ['fuelPrice'].mean().reset_index()

    unique_dd_codes_per_fuelTypeID =
      fuelPrices_per_dd["ISLAND_COMPLEX"].unique
      ()
    print(unique_dd_codes_per_fuelTypeID)

    results=[]

    try:
        time_series_list = []
        for complex in
          unique_dd_codes_per_fuelTypeID:
            temp_df = fuelPrices_per_dd[
              fuelPrices_per_dd["ISLAND_COMPLEX
              "] == complex]
            temp_df_gas_prices = temp_df["
              fuelPrice"].values

            print(complex,temp_df_gas_prices.
              mean())

            time_series_list.append(
              temp_df_gas_prices)

        max_length = max(len(ts) for ts in
          time_series_list)
        time_series_padded = [np.pad(ts, (0,
          max_length - len(ts)), 'constant',
          constant_values=np.nan) for ts in
          time_series_list]

        X = to_time_series_dataset(
          time_series_padded)
        X = TimeSeriesScalerMeanVariance().
          fit_transform(X)
        model = TimeSeriesKMeans(n_clusters=4,
          metric="dtw", max_iter=5)

        y = model.fit_predict(X)

        results.append(y)

    except ValueError as e:
        print(f"Error for : {e}")
    return(results)

shapefile_path = 'TOP_DHM_KOIN.shp'
gdf = gpd.read_file(shapefile_path)

results = clustering_per_gas_type(final_df,gdf
   ,1) ## 1 --> Unleaded 95 ## 2 --> Unleaded
   98/100 ### Diesel --> 4
```

E.5 Script Used for Times Series Clustering Per Prefecture

```
import matplotlib.pyplot as plt
import geopandas as gpd
import pandas as pd
import numpy as np

from tslearn.preprocessing import
  TimeSeriesScalerMeanVariance
from tslearn.utils import to_time_series_dataset
from tslearn.clustering import TimeSeriesKMeans

from sklearn.preprocessing import LabelEncoder

df = pd.read_csv('../data/gas_stations_x_codes.
  csv')
df = df.loc[:,["gasStationID","KOD_01","KOD_11
  ","KOD_21"]]
df.dropna(subset=['KOD_01'], inplace=True)

gas_stations = df.loc[:,["gasStationID","KOD_21
  "]]

fuels = pd.read_csv("../data/PRATHRIWN.csv")

gas_price_ddcode = fuels.merge(gas_stations, on
  =['gasStationID'], how='left')
gas_price_ddcode = gas_price_ddcode.dropna()
gas_price_ddcode['dateDefined'] = pd.to_datetime
  (gas_price_ddcode['dateDefined'], errors='
  coerce')
gas_price_ddcode = gas_price_ddcode.dropna(
  subset=['dateDefined'])
```

```
gas_price_ddcode['dateDefined'] =
  gas_price_ddcode['dateDefined'].dt.date

prefecture_df = pd.read_excel("../data/mit_2021.
  xlsx",dtype=str)
prefecture_df = prefecture_df[prefecture_df['G.K
  . 2021'].str.match(r'^\d{8}$')]
prefecture_df = prefecture_df.rename(columns={'G
  .K. 2021' : 'KOD_21'})
prefecture_df["KOD_21"] = prefecture_df["KOD_21
  "].astype(float)

final_df = pd.merge(gas_price_ddcode,
  prefecture_df,on="KOD_21",how="left")
final_df = final_df.dropna()
global label_encoder
label_encoder = LabelEncoder()
final_df['NOMOS_enc'] = label_encoder.
  fit_transform(final_df['NOMOS'])
copy_final_df = final_df
final_df = final_df.loc[:,["gasStationID","
  fuelTypeID","dateDefined","fuelPrice","KOD_21
  ","NOMOS_enc"]]
final_df = final_df.rename(columns={"NOMOS_enc":
   "NOMOS"})

def clustering_per_gas_type(gas_prices,
  fuelTypeID):
    df_x_fuelType = gas_prices[gas_prices["
      fuelTypeID"] == fuelTypeID]
    df_x_fuelType = df_x_fuelType.reset_index()

    df_x_fuelType['dateDefined'] = pd.
      to_datetime(df_x_fuelType['dateDefined'])
       # Ensure 'dateDefined' is in datetime
      format
    df_x_fuelType.sort_values(['gasStationID', '
      dateDefined'], inplace=True)  # Sort the
      DataFrame

    df_x_fuelType = df_x_fuelType.groupby(['
      gasStationID', 'dateDefined']).mean().
      reset_index()

    def fill_missing_dates(group):
        return group.set_index('dateDefined').
          resample('D').ffill().reset_index()
    df_x_fuelType_imputed = df_x_fuelType.
      groupby('gasStationID').apply(
      fill_missing_dates)
    df_x_fuelType_imputed.reset_index(drop=True,
       inplace=True)
    df_x_fuelType_imputed['KOD_21'] =
      df_x_fuelType_imputed['KOD_21'].astype(int
      ).astype(str).str.zfill(8)

    fuelPrices_per_dd = df_x_fuelType_imputed.
      groupby(['NOMOS','KOD_21', "dateDefined"])
      ['fuelPrice'].mean().reset_index()

    unique_island_complex = fuelPrices_per_dd["
      NOMOS"].unique()

    results= {}  # K-means results
    average_prices_per_island = {} # Islands
      with their respective average prices
    names_per_complex = {} # Complexes with all
      available islands

    for complex in unique_island_complex:
```

```
try:
    print(complex)

    time_series_list = []
    temp_names = []

    temp_df = fuelPrices_per_dd[
      fuelPrices_per_dd["NOMOS"] ==
      complex]

    temp_unique_islands = temp_df["
      KOD_21"].unique()
    for island in temp_unique_islands:
        temp_island_df =
          fuelPrices_per_dd[
          fuelPrices_per_dd["KOD_21"] ==
           island]
        temp_df_gas_prices =
          temp_island_df["fuelPrice"].
          values

        #print(island,temp_df_gas_prices
          .mean())

        time_series_list.append(
          temp_df_gas_prices)

        average_prices_per_island[island
          ] = temp_df_gas_prices.mean()

        temp_names.append(island)

    names_per_complex[complex] =
      temp_names
```

```
            max_length = max(len(ts) for ts in
              time_series_list)
            time_series_padded = [np.pad(ts, (0,
               max_length - len(ts)), 'constant
              ', constant_values=np.nan) for ts
              in time_series_list]

            X = to_time_series_dataset(
              time_series_padded)
            X = TimeSeriesScalerMeanVariance().
              fit_transform(X)

            model = TimeSeriesKMeans(n_clusters
              =5, metric="dtw", max_iter=5)

            y = model.fit_predict(X)

            results[complex] = y

        except ValueError as e:
            print(f"Error for : {e}")
            continue

    return(results,average_prices_per_island,
      names_per_complex)

results, average_prices_per_island,
  names_per_complex = clustering_per_gas_type(
  final_df,1) ## 1 --> Unleaded 95 ## 2 -->
  Unleaded 98/100 ### Diesel --> 4

copy_final_df = copy_final_df.loc[:,["NOMOS","
  NOMOS_enc"]]
copy_final_df = copy_final_df.rename(columns={"
  NOMOS":"NOMOS_dec", "NOMOS_enc":"NOMOS"})
copy_final_df = copy_final_df.drop_duplicates()

df = pd.DataFrame(list(results.items()), columns
  =['NOMOS', 'RESULT'])
df_filtered = df[~df['NOMOS'].isin
  ([17.50,20.75,26.50])]
df_filtered['NOMOS'] = pd.to_numeric(df_filtered
  ['NOMOS']).astype('Int64')
df_1 = pd.merge(df_filtered,copy_final_df,on="
  NOMOS",how="left")

df_names = pd.DataFrame(list(names_per_complex.
  items()), columns=['NOMOS', 'Names'])
df_names = df_names[~df_names['NOMOS'].isin
  ([17.50,20.75,26.50])]
df_names['NOMOS'] = pd.to_numeric(df_names['
  NOMOS']).astype('Int64')
df = pd.merge(df_names,copy_final_df,on="NOMOS",
  how="left")
df = df.drop(columns=["NOMOS"])
df_2 = pd.merge(df_1,df,on="NOMOS_dec",how="left
  ")
df_2 = df_2.drop(columns=["NOMOS"])
df_2 = df_2.rename(columns={"NOMOS_dec":"NOMOS
  "})

df_2.to_excel("../data/fuels/Diesel/
  prefecture_clustering.xlsx")

average_prices_per_island
df_prices = pd.DataFrame(list(
  average_prices_per_island.items()), columns=['
  NOMOS', 'AVERAGE_PRICE'])
```

```
df_prices.to_excel("../data/fuels/Diesel/
  average_diesel_prices.xlsx")
```

E.6 Script Used for Unit Root Testing

```
import matplotlib.pyplot as plt
import geopandas as gpd
import pandas as pd
import numpy as np

from sklearn.preprocessing import LabelEncoder

df = pd.read_csv('../data/gas_stations_x_codes.
   csv')
df = df.loc[:,["gasStationID","KOD_01","KOD_11
   ","KOD_21"]]

df.dropna(subset=['KOD_01'], inplace=True)

gas_stations = df.loc[:,["gasStationID","KOD_21
   "]]

fuels = pd.read_csv("../data/TIMES_PRATHRIWVN.
   csv")

gas_price_ddcode = fuels.merge(gas_stations, on
   =['gasStationID'], how='left')
gas_price_ddcode = gas_price_ddcode.dropna()
gas_price_ddcode['dateDefined'] = pd.to_datetime
   (gas_price_ddcode['dateDefined'], errors='
   coerce')
gas_price_ddcode = gas_price_ddcode.dropna(
   subset=['dateDefined'])
gas_price_ddcode['dateDefined'] =
   gas_price_ddcode['dateDefined'].dt.date

prefecture_df = pd.read_excel("../data/mit_2021.
   xlsx",dtype=str)
```

```
prefecture_df = prefecture_df[prefecture_df['G.H
   . 2021'].str.match(r'^\d{8}$')]
prefecture_df = prefecture_df.rename(columns={'G
   .K. 2021' : 'KOD_21'})
prefecture_df = prefecture_df.drop(columns=["
   PRWTEUOUSA"])
prefecture_df["KOD_21"] = prefecture_df["KOD_21
   "].astype(float)

final_df = pd.merge(gas_price_ddcode,
   prefecture_df,on="KOD_21",how="left")
final_df = final_df.dropna()
global label_encoder
label_encoder = LabelEncoder()
final_df['NOMOS_enc'] = label_encoder.
   fit_transform(final_df['NOMOS'])
copy_final_df = final_df
final_df = final_df.loc[:,["gasStationID","
   fuelTypeID","dateDefined","fuelPrice","KOD_21
   ","NOMOS_enc"]]
final_df = final_df.rename(columns={"NOMOS_enc":
    "NOMOS"})

def clustering_per_gas_type(gas_prices,
   fuelTypeID):
     df_x_fuelType = gas_prices[gas_prices["
        fuelTypeID"] == fuelTypeID]
     df_x_fuelType = df_x_fuelType.reset_index()

     df_x_fuelType['dateDefined'] = pd.
        to_datetime(df_x_fuelType['dateDefined'])
         # Ensure 'dateDefined' is in datetime
        format
     df_x_fuelType.sort_values(['gasStationID', '
        dateDefined'], inplace=True)  # Sort the
```

```
  DataFrame

df_x_fuelType = df_x_fuelType.groupby(['
  gasStationID', 'dateDefined']).mean().
  reset_index()

def fill_missing_dates(group):
    return group.set_index('dateDefined').
      resample('D').ffill().reset_index()
df_x_fuelType_imputed = df_x_fuelType.
  groupby('gasStationID').apply(
  fill_missing_dates)
df_x_fuelType_imputed.reset_index(drop=True,
   inplace=True)
df_x_fuelType_imputed['KOD_21'] =
  df_x_fuelType_imputed['KOD_21'].astype(int
  ).astype(str).str.zfill(8)

fuelPrices_per_dd = df_x_fuelType_imputed.
  groupby(['NOMOS','KOD_21', "dateDefined"])
  ['fuelPrice'].mean().reset_index()

time_series_list = []
list_of_names = []

temp_df = fuelPrices_per_dd[
  fuelPrices_per_dd["NOMOS"] == 5]

temp_unique_communities = temp_df["KOD_21"].
  unique()
for community in temp_unique_communities:
    print(community)
    list_of_names.append(community)
    temp_island_df = fuelPrices_per_dd[
      fuelPrices_per_dd["KOD_21"] ==
            community]
        temp_df_gas_prices = temp_island_df["
            fuelPrice"].values

        time_series_list.append(
            temp_df_gas_prices)

    max_length = max(len(ts) for ts in
        time_series_list)
    time_series_padded = [np.pad(ts, (0,
        max_length - len(ts)), 'constant',
        constant_values=np.nan) for ts in
        time_series_list]

    return(time_series_padded,list_of_names)

time_series, list_of_names =
    clustering_per_gas_type(final_df,1) ## 1 -->
    Unleaded 95 ## 2 --> Unleaded 98/100 ###
    Diesel --> 4

from arch.unitroot import ADF  # Assuming you're
     using arch library for ADF test

adf_results = []
for ts, name in zip(time_series, list_of_names):
    ts_cleaned = ts[~np.isnan(ts)]
    if len(ts_cleaned) > 0:  # Ensure there are
        values for the ADF test
        adf = ADF(ts_cleaned, trend="ct", lags
            =30)
        adf_results.append((name, adf))
    else:
        print(f"No valid data for ADF test in {
            name}")
```

```
for name, adf in adf_results:
    print(f"ADF test results for {name}:")
    print(adf.summary().as_text())

adf_df = pd.DataFrame(adf_results, columns=["
  Community", "ADF Test"])
adf_df["p-value"] = adf_df["ADF Test"].apply(
  lambda x: x.pvalue)
adf_df["t-statistic"] = adf_df["ADF Test"].apply
  (lambda x: x.stat)
adf_df = adf_df.sort_values(by="Community",
  ascending=False)

# Plot the p-values and t-statistics
plt.figure(figsize=(12, 15))
plt.barh(adf_df["Community"], adf_df["t-
  statistic"], color="skyblue")
plt.xlabel("t-statistic")
plt.ylabel("Community")
plt.title("ADF Test Results: t-statistic")
plt.grid(axis="x")

plt.axvline(x=-3.13, color='g', linestyle='--',
  label='Critical Value (10%)')
plt.axvline(x=-3.41, color='y', linestyle='--',
  label='Critical Value (5%)')
plt.axvline(x=-3.96, color='r', linestyle='--',
  label='Critical Value (1%)')
plt.tight_layout()
plt.legend()
plt.show()

adf_df["p-value"] = adf_df["p-value"].astype(
  float)
```

```
plt.figure(figsize=(10, 15))
plt.barh(adf_df["Community"], adf_df["p-value"],
    color="lightgreen")
plt.xlabel("p-value")
plt.ylabel("Community")
plt.title("ADF Test Results: p-values")
plt.grid(axis="x")
plt.show()
```

E.7 Script Used for Cointegration Testing

```
import matplotlib.pyplot as plt
import geopandas as gpd
import seaborn as sns
import pandas as pd
import numpy as np

from sklearn.preprocessing import LabelEncoder

df = pd.read_csv('../data/gas_stations_x_codes.
  csv')
df = df.loc[:,["gasStationID","KOD_01","KOD_11
  ","KOD_21"]]

df.dropna(subset=['KOD_01'], inplace=True)

gas_stations = df.loc[:,["gasStationID","KOD_21
  "]]

fuels = pd.read_csv("../data/TIMES_PRATHRIWVN.
  csv")

gas_price_ddcode = fuels.merge(gas_stations, on
  =['gasStationID'], how='left')
gas_price_ddcode = gas_price_ddcode.dropna()
gas_price_ddcode['dateDefined'] = pd.to_datetime
  (gas_price_ddcode['dateDefined'], errors='
  coerce')
gas_price_ddcode = gas_price_ddcode.dropna(
  subset=['dateDefined'])
gas_price_ddcode['dateDefined'] =
  gas_price_ddcode['dateDefined'].dt.date

prefecture_df = pd.read_excel("../data/mit_2021.
    xlsx",dtype=str)
prefecture_df = prefecture_df[prefecture_df['G.H
    . 2021'].str.match(r'^\d{8}$')]
prefecture_df = prefecture_df.rename(columns={'G
    .K. 2021' : 'KOD_21'})
prefecture_df = prefecture_df.drop(columns=["
    PRWTEUOUSA"])
prefecture_df["KOD_21"] = prefecture_df["KOD_21
    "].astype(float)

final_df = pd.merge(gas_price_ddcode,
    prefecture_df,on="KOD_21",how="left")
final_df = final_df.dropna()
global label_encoder
label_encoder = LabelEncoder()
final_df['NOMOS_enc'] = label_encoder.
    fit_transform(final_df['NOMOS'])
copy_final_df = final_df
final_df = final_df.loc[:,["gasStationID","
    fuelTypeID","dateDefined","fuelPrice","KOD_21
    ","NOMOS_enc"]]
final_df = final_df.rename(columns={"NOMOS_enc":
     "NOMOS"})

def clustering_per_gas_type(gas_prices,
    fuelTypeID):
        df_x_fuelType = gas_prices[gas_prices["
            fuelTypeID"] == fuelTypeID]
        df_x_fuelType = df_x_fuelType.reset_index()

        df_x_fuelType['dateDefined'] = pd.
            to_datetime(df_x_fuelType['dateDefined'])
             # Ensure 'dateDefined' is in datetime
            format
        df_x_fuelType.sort_values(['gasStationID', '
  dateDefined'], inplace=True)  # Sort the
  DataFrame

df_x_fuelType = df_x_fuelType.groupby(['
  gasStationID', 'dateDefined']).mean().
  reset_index()

def fill_missing_dates(group):
    return group.set_index('dateDefined').
      resample('D').ffill().reset_index()
df_x_fuelType_imputed = df_x_fuelType.
  groupby('gasStationID').apply(
  fill_missing_dates)
df_x_fuelType_imputed.reset_index(drop=True,
   inplace=True)
df_x_fuelType_imputed['KOD_21'] =
  df_x_fuelType_imputed['KOD_21'].astype(int
  ).astype(str).str.zfill(8)

fuelPrices_per_dd = df_x_fuelType_imputed.
  groupby(['NOMOS','KOD_21', "dateDefined"])
  ['fuelPrice'].mean().reset_index()

time_series_list = []
list_of_names = []

temp_df = fuelPrices_per_dd[
  fuelPrices_per_dd["NOMOS"] == 5]

temp_unique_communities = temp_df["KOD_21"].
  unique()
for community in temp_unique_communities:
    print(community)
    list_of_names.append(community)
    temp_island_df = fuelPrices_per_dd[
            fuelPrices_per_dd["KOD_21"] ==
            community]
        temp_df_gas_prices = temp_island_df["
            fuelPrice"].values

        time_series_list.append(
            temp_df_gas_prices)

    max_length = max(len(ts) for ts in
        time_series_list)
    time_series_padded = [np.pad(ts, (0,
        max_length - len(ts)), 'constant',
        constant_values=np.nan) for ts in
        time_series_list]

    return(time_series_padded,list_of_names)

time_series, list_of_names =
    clustering_per_gas_type(final_df,1) ## 1 -->
    Unleaded 95 ## 2 --> Unleaded 98/100 ###
    Diesel --> 4

import statsmodels.tsa.stattools as ts

def run_cointegration_test(time_series_padded):
    num_series = len(time_series_padded)
    t_statistics = np.zeros((num_series,
        num_series))
    p_values = np.zeros((num_series, num_series)
        )

    for i in range(num_series):
        #k = 0
        for j in range(i+1, num_series):
            ts1 = time_series_padded[i]
            ts2 = time_series_padded[j]

            # Handle NaN values
            mask = ~np.isnan(ts1) & ~np.isnan(
              ts2)
            ts1 = ts1[mask]
            ts2 = ts2[mask]

            result = ts.coint(ts1, ts2)
            t_statistics[i, j] = result[0]
            p_values[i, j] = result[1]

            print(f"Cointegration test result
              for series {i} and {j}: {result}")

            #k+=1

    return t_statistics, p_values

t_statistics, p_values = run_cointegration_test(
  time_series)

def plot_heatmap(matrix, labels, title):
    fig = plt.figure(figsize=(40,30))
    df = pd.DataFrame(matrix, columns=labels,
      index=labels)
    sns.heatmap(df, annot=True, cmap="coolwarm",
       fmt=".2f", xticklabels=True, yticklabels=
      True)
    plt.title(title)
    plt.show()

plot_heatmap(t_statistics, list_of_names, "T-
  Statistic Heatmap")
plot_heatmap(p_values, list_of_names, "P-Value
Heatmap")
```