UNIVERISTY OF PIRAEUS - DEPARTMENT OF INFORMATICS

ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΙΡΑΙΩΣ – ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

MSc «Cybersecurity and Data Science»

ΠΜΣ «Κυβερνοασφάλεια και Επιστήμη Δεδομένων»

MSc Thesis

Μεταπτυχιακή Διατριβή

Thesis Title: Τίτλος Διατριβής:	**Recommendations for Interactive data visualization of spatial - categorical data** Προτάσεις για διαδραστική οπτικοποίηση χωρικών - κατηγορικών δεδομένων
Student's name-surname: Ονοματεπώνυμο Φοιτητή:	**Routsi Theodoros** Ρούτση Θεόδωρος
Father's name: Πατρώνυμο:	**Michalaki** Μιχαλάκη
Student's ID No: Αριθμός Μητρώου:	**ΜΠΚΕΔ 21046**
Supervisor: Επιβλέπων:	**Papastefanatos George, MSc Instructor** Παπαστεφανάτος Γεώργιος, Διδάσκων ΠΜΣ

January 2024 / Ιανουάριος 2024

3-Member Examination Committee

Τριμελής Εξεταστική Επιτροπή

Γεώργιος Παπαστεφανάτος Διδάσκων ΠΜΣ	Δημήτριος Αποστόλου Καθηγητής	Σταύρος Μαρούλης Διδάσκων ΠΜΣ

Abstract

Data visualization plays a crucial role in modern businesses and organizations. Nowadays, the data volumes are highly increased and the analysts need to manage and visualize all this data to find out trends or insights that may help their research to advance. In order to help analysts visualization recommendation systems are broadly used. These systems offer a collection of capabilities that aim at the visualization of the data contained in enormous datasets much faster than the analysts should have done by examining every variable by themselves, supporting storytelling and helping them to discover many insights. So, this master thesis delves into the world of visualization recommendation systems, providing an extensive overview of the different types of systems available, implementation best practices, and methods for evaluating their effectiveness.

Notably, our investigation addresses a recognized gap in Visualization Recommendation Systems (VRSs), emphasizing the absence of systems adept at guiding users toward areas of interest for exploration and in-depth study. Despite the prevalence of tools for geospatial data analysis, existing platforms often lack the functionality to provide directed recommendations on regions worthy of examination, a deficiency our research seeks to rectify.

So, in this thesis, we focused on the recommendation of views representing areas through a spatial partitioning scheme, knowing this is a flexible way of representing spatial and geographical data. We used hexagons as a grid, in order to analyze the behavior of the grid itself as representing the behavior of all points underneath instead of every individual point. This thesis covers the innovation of recommending and visualizing hexagons, providing readers with a comprehensive understanding of the potential applications of this new format.

Specifically, our contribution is the creation of a visualization recommendation approach that mostly supports geographical data, using a hexagonal grid for studying the behavior of the points underneath in three different use cases. This approach solves two major problems; (a) there are not many existing VRSs that emphasize on the recommendation of geographical areas and (b) the existing VRSs usually recommend specific neighborhoods, cities regions etc. instead of areas that are not depended on the geopolitical boarders, like for instance an area that a part of it belongs to a city and the other part of it belongs to another city. We call these use cases our approach conducts “methods” and they are respectively named as “Global Reference View and Global Target Views”, “Local Reference View and Local Target Views” and “Local Reference View and Local Target Views with Spatial Distance”. The first one returns candidate interesting hexagons defined through all the geographical areas a dataset contains, while the other two methods are related to the study of smaller geographical areas of the dataset defined by the user. The difference between the second and the third method relates to the interestingness the user shows in the spatial distance among the location he wants to study and the hexagons around it.

Overall, this thesis is a valuable resource for anyone seeking to gain a deeper understanding of visualization recommendation systems and their applications in modern businesses and organizations. The inclusion of hexagonal visualization adds an innovative and powerful tool to the visualization toolbox, opening up new possibilities for exploring and analyzing complex data sets. Furthermore, we conducted a “User Study” evaluation for our approach and since the results of the evaluation were shown as positive, we believe this is a succeeded starting point for creating a VRS in the future that will effectively and efficiently help analysts in their studies.

Περίληψη

Η οπτικοποίηση δεδομένων διαδραματίζει κρίσιμο ρόλο στις σύγχρονες επιχειρήσεις και οργανισμούς. Σήμερα, οι όγκοι δεδομένων είναι πολύ αυξημένοι και οι αναλυτές πρέπει να διαχειριστούν και να οπτικοποιήσουν όλα αυτά τα δεδομένα για να ανακαλύψουν τάσεις ή ιδέες που μπορεί να βοηθήσουν την έρευνά τους να προχωρήσει. Προκειμένου να βοηθηθούν οι αναλυτές, χρησιμοποιούνται ευρέως συστήματα συστάσεων οπτικοποίησης. Αυτά τα συστήματα προσφέρουν μια συλλογή δυνατοτήτων που στοχεύουν στην οπτικοποίηση των δεδομένων που

περιέχονται σε τεράστια σύνολα δεδομένων πολύ πιο γρήγορα από ό,τι θα έπρεπε να είχαν κάνει οι αναλυτές εξετάζοντας κάθε μεταβλητή μόνοι τους, υποστηρίζοντας την αφήγηση και βοηθώντας τους να ανακαλύψουν πολλές ιδέες. Έτσι, αυτή η μεταπτυχιακή διατριβή εμβαθύνει στον κόσμο των συστημάτων συστάσεων οπτικοποίησης, παρέχοντας μια εκτενή επισκόπηση των διαφορετικών τύπων διαθέσιμων συστημάτων, τις βέλτιστες πρακτικές εφαρμογής και τις μεθόδους αξιολόγησης της αποτελεσματικότητάς τους.

Ειδικότερα, η έρευνά μας αντιμετωπίζει ένα αναγνωρισμένο κενό στα Συστήματα Συστάσεων Οπτικοποίησης (VRS), τονίζοντας την απουσία συστημάτων ικανών να καθοδηγούν τους χρήστες προς περιοχές ενδιαφέροντος για εξερεύνηση και σε βάθος μελέτη. Παρά την επικράτηση εργαλείων για την ανάλυση γεωχωρικών δεδομένων, οι υπάρχουσες πλατφόρμες συχνά δεν διαθέτουν τη λειτουργικότητα για την παροχή κατευθυνόμενων συστάσεων σχετικά με περιοχές που αξίζουν εξέτασης, μια έλλειψη που η έρευνά μας επιδιώκει να διορθώσει.

Έτσι, σε αυτήν τη διατριβή, επικεντρωθήκαμε στη σύσταση προβολών που αντιπροσωπεύουν περιοχές μέσω ενός σχήματος χωρικής καταμέτρησης, γνωρίζοντας ότι αυτός είναι ένας ευέλικτος τρόπος αναπαράστασης χωρικών και γεωγραφικών δεδομένων. Χρησιμοποιήσαμε εξαγωνικά ως πλέγμα, προκειμένου να αναλύσουμε τη συμπεριφορά του πλέγματος ως αναπαράσταση της συμπεριφοράς όλων των σημείων κάτω από αυτό, αντί κάθε ξεχωριστού σημείου. Αυτή η διατριβή καλύπτει την καινοτομία της σύστασης και οπτικοποίησης εξαγώνων, παρέχοντας στους αναγνώστες μια ολοκληρωμένη κατανόηση των πιθανών εφαρμογών αυτής της νέας μορφής.

Συγκεκριμένα, η συνεισφορά μας είναι η δημιουργία μιας προσέγγισης συστάσεων οπτικοποίησης που υποστηρίζει κυρίως γεωγραφικά δεδομένα, χρησιμοποιώντας ένα εξαγωνικό πλέγμα για τη μελέτη της συμπεριφοράς των σημείων κάτω από τρία διαφορετικά σενάρια χρήσης. Αυτή η προσέγγιση λύνει δύο βασικά προβλήματα: (α) υπάρχουν ελάχιστα υπάρχοντα συστήματα προτάσεων οπτικοποίησης που επικεντρώνονται στη σύσταση γεωγραφικών περιοχών και (β) τα υπάρχοντα συστήματα προτάσεων συνήθως συνιστούν συγκεκριμένες γειτονιές, πόλεις, περιοχές κ.λπ. αντί για περιοχές που δεν εξαρτώνται από τα γεωπολιτικά σύνορα, όπως, για παράδειγμα, μια περιοχή που ένα μέρος της ανήκει σε μια πόλη και το άλλο μέρος ανήκει σε μια άλλη πόλη. Καλούμε αυτά τα σενάρια που πραγματοποιεί η προσέγγισή μας "μεθόδους" και ονομάζονται αντίστοιχα “Global Reference View and Global Target Views”, “Local Reference View and Local Target Views” και “Local Reference View and Local Target Views with Spatial Distance”. Η πρώτη επιστρέφει υποψήφια ενδιαφέροντα εξάγωνα που καθορίζονται μέσω όλων των γεωγραφικών περιοχών που περιλαμβάνει ένα σύνολο δεδομένων, ενώ οι άλλες δύο μεθόδοι σχετίζονται με τη μελέτη μικρότερων γεωγραφικών περιοχών του συνόλου δεδομένων που καθορίζονται από τον χρήστη. Η διαφορά μεταξύ της δεύτερης και της τρίτης μεθόδου σχετίζεται με το ενδιαφέρον που επιδεικνύει ο χρήστης για τη χωρική απόσταση μεταξύ της τοποθεσίας που θέλει να μελετήσει και των εξαγώνων γύρω της.

Συνολικά, αυτή η διατριβή είναι μια πολύτιμη πηγή για όποιον θέλει να αποκτήσει μια βαθύτερη κατανόηση των συστημάτων προτάσεων οπτικοποίησης και των εφαρμογών τους σε σύγχρονες επιχειρήσεις και οργανισμούς. Η συμπερίληψη της εξαγωνικής οπτικοποίησης προσθέτει ένα καινοτόμο και ισχυρό εργαλείο στην εργαλειοθήκη οπτικοποίησης, ανοίγοντας νέες δυνατότητες για εξερεύνηση και ανάλυση σύνθετων συνόλων δεδομένων. Επιπλέον, διενεργήσαμε μια αξιολόγηση με τη μέθοδο της "Μελέτης Χρήστη" για την προσέγγισή μας, και καθώς τα αποτελέσματα της αξιολόγησης παρουσιάστηκαν θετικά, πιστεύουμε ότι αυτό αποτελεί ένα επιτυχημένο σημείο εκκίνησης για τη δημιουργία ενός Συστήματος Προτάσεων Οπτικοποίησης στο μέλλον που θα βοηθήσει αποτελεσματικά και αποδοτικά τους αναλυτές στις μελέτες τους.

Table of Contents

1 Introduction

Data visualization is a crucial tool for data analysis and understanding. It is actually an important process in data science. It has been used to help people make sense of data and make better decisions. Nowadays many sources, like IoT, have led to an aggressive increase in the data volumes available for data analysis. With the increasing availability of large datasets, it has become increasingly important to have efficient ways to visualize this data [1]. To explore these massive and structurally complicated datasets, data analysts often utilize visual data analysis tools, such as Tableau, Qlik, Lyra, Amazon Quicksight, Microsoft Power BI, Google Fusion Tables etc. [2]. However, choosing the right visualization for particular datasets can be challenging, especially for non-experts. This is where data visualization recommendation systems come into play. These systems provide users with suggestions for visualizations that are best suited for their data, based on the data itself and the user's preferences. In addition, not all the visualization tools support geospatial data.

1.1 Visualization Recommendation Systems

The goal is to create visualizations, which support storytelling [16, 17], using an interactive way, that helps the analyst to find interesting and unique insights, patterns or trends without spending much time on studying the attributes/features of a given datasets. This process should be done as effectively and efficiently as possible. However, it is rather easy said than done, so we should know that this is not a process someone can do with ease and it also acquires skillful analysts too. For this purpose, the data mining and machine learning community has developed a large swath of statistical analysis tools such as Knime, RapidMiner, SAS, and SPSS, and programming libraries [38, 39] for doing complex analytics tasks such as classification, clustering, and dimensionality reduction. Only expert users with detailed knowledge of algorithmic details and parametrizations can use these tools effectively.

Data visualization recommendation systems can be based on different techniques, including statistical methods, machine learning algorithms, and expert knowledge. Regardless of the method used, the goal is to provide users with visualizations that are effective and easy to understand. There are several factors that need to be considered when making a recommendation for a data visualization, including the type of data, the size of the data, the distribution of the data, and the users' goals and preferences.

Existing visualization recommendation systems broadly fall into three categories [9,10,11,12]: (i) content-based; (ii) collaborative-based and (iii) hybrid-based that combines recommendation techniques, to gain better performance with fewer drawbacks. There are also some other categories like [12,15] knowledge-based systems rule-based systems and utility-based systems. All these systems have their strengths and weaknesses. Visualization recommender systems either suggest data queries (selecting *what* data to visualize) or visual encodings (*how* to visualize selected data) [68] and the types of visualizations usually refer to histograms, bar charts, (3D) scatterplots, time series, node-link diagrams etc.

In order to evaluate the effectiveness and the efficiency of a visualization recommendation system there are many ways broadly used. These ways usually refer to (a) user testing; (b) accuracy metrics and (c) user satisfaction surveys. User testing involves gathering feedback from users on the usability and effectiveness of the visualization recommendation system. Accuracy metrics measure the accuracy and relevance of the recommendations generated by the visualization recommendation system and some of the most common metrics used to evaluate these systems include precision, recall, and F1 score. User satisfaction surveys can provide insights into the overall satisfaction and usefulness of the visualization recommendation system and include questions about ease of use, relevance of recommendations, and overall satisfaction with the system.

There are several benefits to using data visualization recommendation systems. One of the main benefits is that they simplify the process of selecting a visualization, making it easier for

non-experts to understand their data. Additionally, data visualization recommendation systems can improve the accuracy and effectiveness of visualizations, helping users to identify important trends and patterns in their data. Another benefit of data visualization recommendation systems is that they can reduce the time and effort required to create visualizations. By providing recommendations for visualizations, users can avoid the time-consuming process of trying out different visualizations to see what works best. This can save time and increase productivity, allowing users to focus on other aspects of their data analysis.

1.2 Problems and Challenges of VRSs

Despite the many benefits of data visualization recommendation systems, there are also some limitations. One of the main limitations is that the recommendations generated by these systems are only as good as the data and algorithms used. If the data is poor quality or the algorithms are not properly designed, the recommendations may not be accurate or effective. Another limitation of data visualization recommendation systems is that they can be complex and difficult to use, especially for non-experts. This can make it challenging for users to get the most out of these systems, especially if they are not familiar with data visualization or the algorithms used.

So, we could say that in today's data-driven world, data visualization has become an essential tool for understanding and communicating complex data. With so many different types of visualizations available, it can be challenging to know which one to choose for particular datasets. That's where data visualization recommendation systems come in. These systems leverage machine learning algorithms and user feedback to generate recommendations on the best visualization type to use for particular datasets. By automating the process of visualization selection, these systems can save time and improve the accuracy of insights gained from data analysis.

Our investigation addresses a recognized gap in Visualization Recommendation Systems (VRSs): the absence of systems adept at guiding users toward areas of interest for exploration and in-depth study. Despite the prevalence of tools for geospatial data analysis, a deficiency persists in their ability to provide directed recommendations on regions deemed worthy of examination. Existing platforms, proficient in geodata-centric analyses, often lack functionality for proactive suggestion of specific areas meriting detailed scrutiny. Our research aims to rectify this gap by introducing a pragmatic approach that facilitates geospatial analysis and furnishes users with recommendations, thereby enhancing analytical efficacy and user experience in geospatial data exploration. Moreover, prevailing systems tend to recommend entities like regions or cities, overlooking the unique capabilities of grids, such as hexagons, which offer more precise and flexible recommendations, covering areas spanning multiple countries. This aligns with the nuanced requirements of spatial analysis and user exploration in diverse geographic contexts

1.3 The contribution of this thesis

In this master thesis, we present a visualization recommendation method that supports geographical data (geodata) and works with a hexagonal-based partitioning of the space. We want to create a system that helps the analysts to find useful insights and interesting patterns between geographical areas. Usually, geodata have two columns, among all the other columns, that contains the geographical information that refers to the latitude and the longitude of every input of the datasets. These two variables help us visualize data points on the map, but there are some limitations using data points for analysis that make it difficult discovering interesting insights. The main difficulties are that there is no clear form of classification for points within a certain area and (b) It may be computationally expensive for a computer system to calculate distances of neighboring points to group them together. So, we used hexagons as a grid, in order to analyze the behavior of the grid itself as representing the behavior of all points underneath. We chose hexagon-based grid instead of something else, because hexagons have only one distance between a hexagon’s center-point and its neighbors’, compared to two distances for squares or

three distances for triangles. This property greatly simplifies performing analysis and smoothing over gradients.

Our approach suggests to the users, hexagons that they may find interesting. The interestingness of a hexagon is computed by a deviation-based utility function that it measures the difference between each target view from the reference view. The reference view is a group-by clause created by a set of attributes, the dimension / categorical attribute(s), the measure / numeric attribute and the aggregation function, the users want to study, over the whole datasets. Similar to the reference view the target view is created via a group-by clause of the same set of attributes and aggregation function as a representation of the behavior of a unique hexagon. Then, a map partitioned by colored hexagons is shown to the users, where the color of each hexagon defines the interestingness of this hexagon.

This approach supports three different methods. The first method refers to the whole datasets or a subset of it defined by the users and it is named “Global Reference View and Global Target Views”. This means that the reference view and the target views are created based on the whole datasets and the approach returns interesting hexagons through all the area covered by the datasets. The second method, which is called “Local Reference View and Local Target Views”, asks the user to input a geographical point (latitude, longitude) so they may find insights locally, in a certain radius around the point of interest, where the reference view and the target views are created in the same way as earlier based on all the data contained in this limited area. Lastly, the third method, the “Local Reference View and Local Target Views with Spatial Distance”, works as the second one having a key difference, where the interestingness of a view is measured not only based on the difference between reference and target views but also based on an additional feature that computes the distance between the point of interest and the center of each hexagon.

In our case, we used the user testing method to evaluate our approach. The main parameters we wanted to evaluate fall in to two major categories; (i) the time needed to show the results (efficiency) and (ii) the simplicity of showing the results, where the analysts have to say if they can detect the most interesting hexagons easily or not. Additionally, one third parameter we would like to examine refers to the utility function we used, so we could compare it to other utility functions and see which one returns more interesting views, that would help the users to find insights or patterns. We need to evaluate all three methods our approach implements, individually, via these parameters to have an extensive overview of how our method works and as future work to make the changes needed in order to improve it.

We have to mention that we used ChatGPT[1] to help us implement some topics in this master thesis.

1.4 Thesis structure outline

In this thesis, we explore the realm of recommendation systems and visualization techniques, focusing on their integration for more effective data analysis. The literature review in Chapter 2 provides a deep understanding of various recommendation systems and visualization types. It discusses best practices for implementation and introduces evaluation metrics such as Recall, Precision, Accuracy, ROC Curve, and F-Measure. A unique contribution lies in the introduction of hexagonal-based visualizations. Chapter 3 details the methodology, outlining the creation of hexagons and the utility

[1] ChatGPT is a powerful AI tool that is broadly used in many different topics. We, used this tool to: (a) generate interesting information and (b) express the text in a more scientific way. Even though ChatGPT helped us immensely, using an AI system for generating information carries risks that refer to three major categories [35]; (1) Risk of Text Plagiarism, (2) Risk of Idea Plagiarism and the (3) Introduction of False and Outdated Information. These risks arise due to the facts that ChatGPT may suggest phrases or sentences that are too similar to existing works, it can suggest unpublished work if shared on it by other researchers and it also adds incorrect details to any text that it edits or checks, including supporting citations and references or it adds outdated information because its dataset is not up-to-date.

The way we used ChatGPT did not let us deal with the risk of the idea plagiarism. So, we tried to avoid the other two risks by taking several precautions. To prevent text plagiarism we carefully paraphrased ChatGPT's output, ensuring a unique presentation of ideas. To validate information accuracy and currency, we cross-referenced with our own knowledge and consulted relevant scientific literature. In any case, we performed our own research for every topic this master thesis covers minimizing reliance on the tool and upholding academic integrity.

function, utilizing metrics like Euclidean Distance, Cosine Similarity, Manhattan Distance, KL Divergence, MAD etc. Chapters 4 and 5 present the implementation and demonstration of three distinct methods, ranging from global to local views, with spatial considerations. These methods are applied to real-world use cases, demonstrating their practicality and effectiveness, being our key contributions to the world of VRSs. Chapter 6 delves into user evaluation, providing valuable insights into user experience and system usability according to their interaction with our implementation, giving us better understanding for further improvements. The thesis concludes in Chapter 7, summarizing the findings and contributions, emphasizing the significance of the research in the recommendation system domain. It also outlines future research directions, suggesting potential enhancements and applications for the proposed methods, thereby paving the way for further advancements in the field of recommendation systems and data visualization.

2 Recommendation systems – Generic Knowledge

Data visualization is an important process in data science. Data scientists have to discover interesting and unique insights and patterns that may be useful to understand and maybe solve real world problems. Then, these insights should be able to be understood both by other scientists and by not so skillful users, too. In other words, analysts have to provide this information with the most suitable way that supports the idea of storytelling [16,17]. So, scientists were in need of visualization recommendation systems that help them not only to select the most appropriate type of visualization for their data, like bar charts, pie charts, line plots, scatter plots etc, but also to help them discover unique insights much faster or even to find insights that they would have probably not noticed in other way. Many of these systems use machine learning algorithms and user feedback to analyze datasets and suggest visualization types that are likely to reveal insights in the most effective way possible. There are, also, systems that do not use ML but they are more interested in users' feedback and work in a more interactive way in order to capture their space of interest and generate useful visualizations.

The process of data visualization can really be a troublemaker. While visualizing data, analysts have to deal with lots of difficulties involving many different factors like the enormous size of a datasets that makes it difficult to select the right features to visualize or even what type of visualization should be created by the analyst. It is a process that should also need plenty of time and exploration. To solve this problem and many other problems visualization recommendation systems, that use a combination of data characteristics and user feedback to generate recommendations and even powerful algorithms, are usually used. So, visualization recommendation systems can be especially useful not only for skillful analysts but also for those who are new to data analysis or lack the expertise to select the best visualization type for their data. They can not only save time but to improve the precision of insights gained from data analysis, too.

In this chapter, we provide an overview of the types of recommendation systems, focusing on the visualization recommendation systems with an emphasis on their types and the underlying methodologies that drive their functionality. We will dissect the various recommendation algorithms, ranging from machine learning-based approaches that leverage intricate data patterns to interactive systems that prioritize user engagement and feedback. Additionally, we will scrutinize the most commonly utilized visualization recommendation systems, examining their strengths and limitations. Furthermore, we will navigate through the maze of visualization types, unraveling the complexities associated with options like bar charts, pie charts, line plots, and scatter plots as well as our main way of representing views, a grid – based visualization relying on the shape of hexagons. We aim to empower both novice and experienced analysts, providing them with the knowledge and tools necessary to select the most suitable visualization techniques for their datasets. Throughout this chapter, we will also explore best practices for implementing visualization recommendation systems. Moreover, we will discuss key evaluation metrics such as Recall, Precision, Accuracy, ROC Curve, and F-Measure, illuminating the path toward assessing the efficiency and effectiveness of visualization recommendation systems.

2.1 Types of recommendation systems

Existing types of recommendation systems, generally, broadly fall into three categories [9, 10, 11, 12], the (i) content-based systems, the (ii) collaborative-based systems and the (iii) hybrid-based systems. We will also discuss about three more types of recommendation systems [12,15]; the (iv) knowledge-based systems; the (v) rule-based systems and (vi) utility-based systems. Collaborative filtering is probably the most popular technique currently used in recommender systems (e.g. at Amazon [8]). However, collaborative filtering (as well as content-based filtering) assumes that there is historical rating data available for a large number of items. As a result, it suffers from the traditional "cold start" [21] problems when historical ratings are sparse. Knowledge-based filtering [14], in contrast, does not depend on history and therefore, does not suffer from cold start problems.

Content-based systems generate recommendations based on the content of the data itself. For instance, if a user has watched and enjoyed several action movies, a content-based recommendation system might suggest other action movies with similar themes, actors, and directors. Measuring the utility of content-based filtering is commonly calculated by using heuristic functions, such as the cosine similarity metric. Content-based systems can be highly effective at generating accurate recommendations, but they can also be complex and computationally intensive.

Collaborative [37] is the process of filtering or evaluating items using the opinions of other people. They generate recommendations based on the preferences and feedback of other users. These systems rely on data from multiple users to generate recommendations, making them effective at generating personalized recommendations There have been many collaborative filtering algorithm measures that calculate the similarities among features as the interestingness measure. The commonly used similarity measures in the literature include mean-squared difference, Pearson correlation, cosine similarity, Spearman correlation, and adjusted cosine similarity [22, 26]. However, collaborative filtering systems can be limited by the quality and quantity of data, and they may not be effective for new or niche datasets.

Hybrid-based [40, 44, 45, 46] recommender systems combine two or more recommendation techniques to gain better performance with fewer of the drawbacks of any individual one. Most commonly, collaborative filtering is combined with some other technique in an attempt to avoid the ramp-up problem, where the converse of this problem is the stability vs. plasticity problem for such learners. All of the learning-based techniques (collaborative, content-based and demographic) suffer from the ramp-up problem in one form or another.

Knowledge-based [36] filtering uses knowledge about users and products to pursue a knowledge-based approach to generating a recommendation, reasoning about what products meet the user's requirements. Unlike other recommender systems, they do not depend on large bodies of statistical data about particular rated visualizations or particular users. Users are an integral part of the knowledge discovery process, elaborating their information needs in the course of interacting with the system.

Utility-based filtering. Instead of attempting to create long-term generalizations about their users, utility-based recommenders base their recommendations on an assessment of how well a user's needs and the range of available options match. Utility-based recommenders base their recommendations on a calculation of each object's utility to the user. The utility function that the system has derived for the user is thus the user profile, and it uses constraint satisfaction techniques to find the best match.

Rule-based filtering systems generate recommendations based on predefined rules and heuristics. These systems typically have a set of predefined rules [46] that are applied to the data, such as "use a pie chart for categorical data" or "use a scatter plot for continuous data". Though effective for many use cases, these systems suffer from three major limitations. First, visualization is a complex process that may require modelling non-linear relationships that are difficult to capture with simple rules. Second, crafting rule sets is a costly process that relies on expert judgment. Lastly, as the dimension of input data increases, the combinatorial nature of rules results in an explosion of possible recommendations. Rule-based systems are often simple and easy to implement, but they can be limited by the quality and comprehensiveness of the rules.

At this point, we have to mention that there are many other visualization recommendation systems that use different filtering methods like demographic recommender systems, and these we mentioned are the ones we thought that are the most commonly used. In fact, all of them have to deal with many different issues [33] like 'cold start' that mentioned earlier, 'data sparsity', 'scalability', 'diversity' and 'habituation effect' and this is the reason we can not say which one of them is the best to use, as all of them have their advantages and disadvantages, because of their different point of view/scope. Most of them are based on different Machine Learning approaches for forecasting the best visualization, via interacting with the user's preferences until a time point and the relation the features of the given data have.

2.2 Visualization Recommendation Systems

Visualization Recommendation Systems (VRS) have emerged as indispensable tools in the field of data analysis, offering valuable guidance to analysts in selecting appropriate visualization techniques for their datasets. These systems leverage a combination of sophisticated algorithms and user feedback to automate the process of choosing the most effective visual representations. One of the key features of VRS is their ability to handle the overwhelming variety of visualization options available, such as bar charts, heat maps, and network diagrams. By analyzing the characteristics of the datasets and understanding the analytical goals, VRSs can suggest suitable visualizations, saving analysts significant time and effort in the exploration phase of their research.

Moreover, VRS often incorporate machine learning techniques, including clustering algorithms and regression models, to identify patterns within data and recommend visualizations that best highlight these patterns. These algorithms consider factors like the nature of the data (categorical, numerical, time-series) and the relationships between variables, ensuring that the recommended visualizations are not only visually appealing but also statistically meaningful. Additionally, VRS continue to evolve, integrating advanced techniques like natural language processing to comprehend user queries and preferences, thereby customizing recommendations further.

Visualization recommender systems either suggest data queries (selecting *what* data to visualize) or visual encodings (*how* to visualize selected data) [68]. Some related works that support more data, encoding, and task types are SAGE [52], BOZ [13], and Show Me [34]. Recently, hybrid systems such as Voyager [46], Explore in Google Sheets [20, 66], VizDeck [43], and DIVE [19] combine visual encoding rules with the recommendation of visualizations that include non-selected columns.

Visual encoding [68] is the process of mapping data variables to visual properties in a chart or graph. In other words, it's the way in which we represent data visually, by using different shapes, colors, sizes, positions, and other visual cues to convey information about the underlying data. The choice of visual encoding depends on the nature of the data and the type of chart or graph being used. For example, we might use color to represent a categorical variable, such as different types of fruit, or size to represent a numerical variable, such as the number of units sold. Here are some common visual encodings and the types of data they are typically used to represent:

a) Position: The location of points or bars on a chart or graph can represent numerical or categorical data.

b) Color: Different hues, saturations, and brightness levels can be used to represent categorical or ordinal data, as well as numerical data through the use of a color scale.

c) Size: The size of points or bars can represent numerical data, such as the frequency or magnitude of a variable.

d) Shape: Different shapes can be used to represent categorical data or to differentiate between multiple groups or series.

e) Texture: Different patterns or textures can be used to represent categorical or ordinal data, or to add texture or detail to a chart or graph.

f) Orientation: The direction or angle of bars or lines can be used to represent numerical or categorical data.

By choosing the appropriate visual encoding, we can create effective and informative visualizations that help us to explore, understand, and communicate data. However, it's important to use visual encoding appropriately and with care, as inappropriate or misleading visual encodings can distort or obscure the underlying data and lead to incorrect conclusions.

The visualizations used by many ML-based recommender systems are generated by rule-based systems and evaluated under controlled settings. Though effective for many use cases, these systems suffer from three major limitations. First, visualization is a complex process

that may require modelling non-linear relationships that are difficult to capture with simple rules. Second, crafting rule sets is a costly process that relies on expert judgment. Lastly, as the dimension of input data increases, the combinatorial nature of rules result in an explosion of possible recommendations.

On the other hand, ML-based systems propose to train models that learn directly from data and can be embedded into systems *as-is*. ML-based systems directly learn the relationship between data and visualizations by training models on analyst interaction. Some ML-based Visualization Recommender Systems are **DeepEye** [33], **Data2Vis** [4], **Draco-Learn [6], VizML** [69], **ViewSeeker** [41]. Each one of the above systems contributes, in a unique way, in the analysis, visualizing the most interesting attributes, according to each one's criteria.

Voyager

Voyager is a rule-based visualization tool that supports exploration of multivariate, tabular data and privileges data variation over design variation. This means that it performs different variable selections and transformations instead of different visual encodings of the same data. It also, automatically generates a diverse set of visualizations and have the user select among them with the appropriate filtering and recommendation strategies needed in order to promote relevant views. It provides valid visual encodings and attempts to produce relevant and effective views based on a user's current exploration state. Voyager shows univariate summaries of all variables prior to user interaction and analysts can indicate those variables and transformations they wish to include. Voyager organizes suggested charts by clustering encoding variations of the same data and showing a single top-ranked exemplar of each cluster. It also partitions the main gallery into a section that involves only user-selected variables and a section that includes additional, non-selected, variables recommended by the system.

VizDeck

The web-based visual analytics tool VizDeck automatically suggests a selection of suitable visualizations for relational data based on the statistical features. VizDeck employs vizlets to succinctly represent the structure of several visualizations at once. Vizlets are widely defined as any UI element that either displays data or allows user input. The user rearranges these vizlets, eliminates the unnecessary ones, and elevates the ones that demand more investigation. In the form of a collection of key-value pairs that collectively characterize the visualization, it generates concise definitions of vizlets known as protovizlets. Bar charts, histograms, line charts, timeline plots, pie charts, maps, and multi-select boxes are some of the Vizlet types. The protovizlets are arranged according to their score, which is determined by calculating the likelihood that each vizlet will be chosen based on a learning link between statistical aspects of the data used to create the vizlet and the type of visualization. The system's log data is used as the ground truth to train the model. A vizlet is deemed significant enough to be examined in more detail each time a user promotes it. Similar to a vote, a discard action indicates that the user did not find the widget useful. A classification as "good" or "bad" is indicated by a positive net sum of these votes among users. Next, an attempt is made to predict this classification using data features. In order to vary the order in which the vizlets are displayed to the user, the server additionally compares the protovizlets against one another using similarity scores. The server then chooses the top-k protovizlets to suggest to the user.

DIVE

DIVE is a mixed-initiative system that combines recommendations to enable storytelling, visualization, and statistical analysis. It combines the various data exploration pipeline steps with a description of a system that expands the application of mixed-initiative methods to statistical analysis and data model inference. The relevance score R = |S C|/|C|, which indicates the number of user-selected fields included in the visualizations, is first calculated for each visualization by Dive. In addition to the typical descriptive statistics (min, max, mode, average) for 1-D

visualizations, it also calculates the statistical aspects of the visualizations, such as entropy and normality. Dive calculates the correlation and the coefficients of a linear fit for 2-D visuals.

Four typical statistical analysis tasks are supported by DIVE. Users can generate 1-D and 2-D aggregate tables using the aggregate job that show the count, mean, or sum of elements in a group. Users of correlation can build correlation matrices between different quantitative areas. Users can utilize one-way or two-way ANOVA to compare group means. By default, analyses are carried out on either previously selected user fields or randomly chosen valid fields on these three pages. Simple linear or logistic regressions are supported by regression. By taking the log or square, users can additionally add interaction terms or change independent variables. Independent variables are ordinarily picked by the forward selection method.

DeepEye

DeepEye trains binary classifiers to determine whether visualizing the given datasets is meaningful, trains a supervised learning to rank model using, and presents both a graph-based approach and rule-based optimizations finding the visualizations that are not needed using real world datasets. It consists an offline component and an online component. The offline component relies on examples to train a binary classifier to determine whether a visualization is good or not, and a *learning-to-rank* model that ranks visualizations. This process is done periodically when there are plenty of examples available or else experts specify partial order as rules-based on their knowledge. From the other hand, online component identifies all possible visualizations, uses the trained classifier to determine whether a visualization is good or not, employs either the learning-to-rank model or expert provided partial orders to select top-*k* visualizations.

VizML

VizML learns visualization design choices from a large corpus of datasets and associated visualizations. It uses neural networks to predict design choices. Firstly, design choices are made depending on datasets, task, and context. Then, it works by collecting and cleaning the corpus, extracting features from each dataset, and extracting five key design choices from corresponding visualizations. The datasets used to train VizML models are extremely diverse in shape, structure, and distribution. It only recommends the latter and not data queries and visual encodings. In addition, it has not been created an application that employs this visualization model. After the feature extraction, encoding-level design choices take place. Then the neural network, using 5-fold cross-validation, having naive classifiers guessing the base rates with high accuracies was trained with the Adam optimizer and a mini-batch size of 200.

ViewSeeker

ViewSeeker adopts the machine learning model-based approach supporting two forms of adaptation (a) Utility function tuning and (b) utility function integration. The first one, Utility function tuning, interacts with the analyst to determine the most suitable parameters of a utility function in the current analysis context. The second one selects example views for labeling and to predict the contribution of each utility function in a multi-objective “ideal” utility function in the current analysis context. The UFs, used in ViewSeeker, adopt a reference parameter representing common methods for a view, and the view interestingness is measured by the distance between the target view and the reference view. The greater this difference, the higher the utility is. Measuring the difference between two views essentially equals measuring the distance between their two normalized probability distributions.

Draco-Learn

Draco-Learn [6] is a formal model that represents visualizations as logical facts and design guidelines as hard and soft constraints. Constraint weights are learned using a ranking support vector machine trained on ranked pairs of visualizations harvested from graphical perception studies [28, 53]. Draco then recommends visualizations that satisfy these constraints by solving a combinatorial optimization problem. To achieve its goal, it was formulated a visualization

description language based on the Vega-Lite grammar [54] and then this language was extended to express datasets and task characteristics.

VisualFacts

There are also platforms that specify on geolocated data. One of them, that inspired us, is VisualFacts [73]. VisualFacts is a visual analytics platform for big geo-located data that assists users perform analysis of raw data files of varying quality (with duplicates or missing data) in rich visual ways. It visually interacting with the data on a map, combines and integrates technologies from RawVis [74] and QueryER [75]. The main idea of this platform is a visual aware in-memory index, which is constructed on-the-fly and adjusted to user interaction, as well as an engine which offers on-the-fly visual ER and clustering of duplicate data.

2.3 Types of visualization

Visualization recommendation systems can be categorized based on the type of visualization they recommend. Based on previous researches [24] and for each data type appropriate visualization techniques and visualization tasks have been designed [25]. So, we could say that the most common types of visualizations that visualization systems recommend are the following ones and they are described more analytically in the Figure 1 [67]:

- histograms, word clouds and box plots for 1-dimensional data,
- bar charts, scatter plots, matrices, linked histograms etc. for 2-dimensional data,
- 3D scatter plots or metaphoric worlds for 3-dimensional data,
- timeline visualizations such as theme rivers, clustered time series or time matrices,
- stacked displays such as tree-maps, sunbursts, hyperbolic trees, dendrograms, cone and radial trees for hierarchical data,
- node-link diagrams with graph layout algorithms such as Reingold and TIlford, H-trees, network diagrams and Balloon graphs for representing relationships. Venn diagrams, Euler diagrams and cluster maps are used for representing relationships between sets,
- elastic lists, parallel coordinates, data meadows, etc. for multi-dimensional data.

Bar charts are the most common type of visualization that are often used to compare different categories of data. Bar chart recommendation systems use algorithms to recommend the best type of bar chart for a given datasets, based on factors such as the number of categories, the range of values, and the desired level of detail. Scatterplots are another popular type of visualization that are used to show the relationship between two variables. Scatterplot recommendation systems can suggest the best type of scatterplot based on factors such as the size of the datasets, the nature of the relationship between the variables, and the desired level of detail. Network diagrams are used to represent relationships between entities in a datasets. Network diagram recommendation systems can suggest the best type of diagram based on factors such as the size and complexity of the network, the types of entities involved, and the desired level of detail.

It's always good to notice that all types of visualizations are useful and their selection is based on many factors and use cases in order to represent the given data in the best way. So, it would be unprofessional and not right to say that the other visualization techniques like histograms, 3D scatterplots, boxplots etc., are useless, as they provide many insights such as the distribution of the data (histograms), the number of outliers (boxplots) and many more.

Furthermore, we will explore one more grid – based visualization type, that have the shape of hexagons. Hexagons, with their symmetry and uniformity, have fascinated scientists, mathematicians, and artists for centuries. They find practical applications in architecture, engineering, and design, offering efficient use of space and material. Hexagons' tessellation property is valuable in computer science and network topology. In geography, hexagons create uniform maps, while in computer graphics, they simulate natural phenomena. Hexagons' applications extend to science, representing molecular structures, and to art, creating intricate patterns. Overall, hexagons' unique properties make them indispensable in various fields, ensuring their continued significance in science, technology, art, and design.

Chart	Dimensions	Characteristics
Histogram	1D	Numerical (Continuous) on x-axis Numerical/Function on y-axis
Pie chart	1D	Part-whole (Categorical)
Area graph	1D - 2D	Numerical (Continuous)/Temporal on x-axis Numerical/Function on y-axis
Line graph	1D - 2D	Numerical (Continuous)/Temporal on x-axis Numerical/Function on y-axis
Boxplot	2D	Numerical, Categorical & Temporal
Bar graph	2D	Categorical/Temporal on x-axis Numerical/Function on y-axis
Heatmap	2D - 3D	Relational data (Numerical, Categorical & Temporal)
Scatterplot	2D - 4D	Numerical (Discrete) on axis Categorical as color/glyphs
Stacked bar graph	2D or more	Grouped categories (one bar) on x-axis Numerical/Function on y-axis
Multiset bar graph	2D or more	Grouped categories (multiple bars) on x-axis Numerical/Function on y-axis
Stacked area graph	2D or more	Numerical (Continuous)/Temporal on x-axis Numerical/Function on y-axis Category represented by line
Stream graph	2D or more	Temporal Category represented by color
Radar chart	2D or more	Relational data (Both Numerical & Categorical)
Multi-line graph	2D or more	Numerical (Continuous)/Temporal on x-axis Numerical/Function on y-axis Category represented by line
Bubble plot	4D	Numerical (Discrete) on axis and as size Categorical as color
Parallel Coordinates	HD	Relational data (Usually Numerical-Categorical can be used)
ScatterPlot Matrix	HD	Numerical (Discrete) on axis Categorical as color/glyphs
Matrix of Histograms	HD	Numerical (Discrete)
Matrix of Line Graphs	HD	Numerical (Continuous)/ Temporal on x-axis Numerical/Function on y-axis
Matrix of Area Graphs	HD	Numerical (Continuous)/ Temporal on x-axis Numerical/Function on y-axis
Matrix of Multi-line Graphs	HD	Numerical (Continuous)/ Temporal on x-axis Numerical/Function on y-axis Category represented by line
Matrix of Heatmaps	HD	Relational data (Numerical, Categorical & Temporal)

In this master thesis, we tried to create a visualization recommendation method via analyzing geospatial data. There are many ways to analyze data related to maps. One of the most common ways is to plot a list of coordinates and see where the densities and clusters arise. However, this way has two major disadvantages, (a) There is no clear form of classification for points within a certain area and (b) It may be computationally expensive for a computer system to calculate distances of neighboring points to group them together. So, we found a grid framework containing points within a certain fence and we analyzed the behavior of the grid itself as representing the behavior of all points underneath. A common form of using a grid system more locally is to define zones in cities, by drawing out a shape over a neighborhood, city, district or country. However, this way, sometimes has an overlapping issue between two or more zones because one zone may be larger than one other zone. Furthermore, zones or districts may be defined based on natural features, population, economy, etc. and are subject to change over time.

Figure 1. Types of charts and their characteristics

In order to cover the Earth with repeated tiling we need to choose the shape that acts as a building block facilitating the complete tiling [70]. This shape needs to be closed and uniformly repeating. The most competitive candidates are the triangle (3 sides), square (4 sides), and hexagon (6 sides), but there is no right choice because it depends on the different use-cases. So, the grid system we chose refers to hexagons, because hexagons have only one distance between a hexagon's center-point and its neighbors', compared to two distances for squares or three distances for triangles. This property greatly simplifies performing analysis and smoothing over gradients.

Hexagon-based visualization recommendation systems are a promising area of research, as they offer a unique and flexible way to visualize complex data. By using hexagons to represent data points, these systems can provide a more intuitive and informative view of the data, while also allowing for more advanced analysis techniques such as clustering and machine learning. As more research is conducted in this area, it is likely that hexagon-based visualization recommendation systems will become increasingly common and useful for a wide range of applications.

2.4 Best practices for implementing VRS

Implementing an effective visualization recommendation system requires careful consideration of several factors, including data preprocessing, algorithm selection, and user interface design. In this chapter, we'll outline some best practices for each of these areas. Some of them include (a) Data Preprocessing, (b) Algorithm Selection, (c) User interface Design, (d) Testing and Evaluation.

Before implementing a visualization recommendation system, it's essential to (a) preprocess the data to ensure its quality and usability. This includes cleaning and standardizing the data, as well as identifying and addressing any missing values or outliers. Additionally, it's important to select the appropriate features and variables for analysis, as this can affect the accuracy and relevance of the recommendations. However, it is not always needed to apply the data preprocessing just because the analysts may want to find out interesting patterns among the spotted outliers or even they might be interested in analyzing the raw datasets for many reasons.

Then, the choice of algorithm is critical to the effectiveness of a visualization recommendation system. There have been many algorithms that calculate the similarities among features in order to measure the interestingness of a view such as mean-squared difference, Pearson correlation, cosine similarity, Spearman correlation, and adjusted cosine similarity. There are also several machine learning algorithms that can be used, including decision trees, neural networks, and clustering algorithms. The selection of the algorithm will depend on the specific requirements and constraints of the application, such as the size of the datasets, the complexity of the data, and the desired level of accuracy.

User Interface Design: The user interface of a visualization recommendation system plays a crucial role in its usability and effectiveness. The interface should be intuitive and easy to use, with clear instructions and guidance on how to input data and interpret the recommendations. It's also important to provide users with options to customize and refine the recommendations based on their preferences and requirements.

Testing and Evaluation: Finally, it's essential to test and evaluate the effectiveness of the visualization recommendation system. This includes conducting user testing to gather feedback and identify areas for improvement, as well as evaluating the accuracy and relevance of the recommendations using metrics such as precision, recall, and F1 score.

By following these best practices, you can create an effective visualization recommendation system that delivers meaningful insights and value to users. In the next chapter, we'll explore various methods for evaluating the effectiveness of a visualization recommendation system.

2.5 Evaluating the effectiveness

Evaluating the effectiveness of a visualization recommendation system is essential to determine whether it meets the needs and requirements of the users. There are several methods for evaluating the effectiveness of these systems, including user testing, accuracy metrics, and user satisfaction surveys.

1. User testing: User testing involves gathering feedback from users on the usability and effectiveness of the visualization recommendation system. This can be done through surveys, interviews, or observation of user behavior. User testing can provide valuable insights into the strengths and weaknesses of the system, as well as identify areas for improvement. It is an iterative process that allows developers to iteratively refine the system based on real-world user interactions, ultimately leading to a more user-centric and effective solution.
2. Accuracy metrics: Accuracy metrics measure the accuracy and relevance of the recommendations generated by the visualization recommendation system. Common metrics used to evaluate these systems include precision, recall, and F1 score. Precision

measures the proportion of relevant recommendations among all recommended visualizations, while recall measures the proportion of relevant recommendations among all relevant visualizations. F1 score is the harmonic mean of precision and recall and is often used to evaluate the overall effectiveness of the system.

3. User satisfaction surveys: User satisfaction surveys can provide insights into the overall satisfaction and usefulness of the visualization recommendation system. These surveys can include questions about ease of use, relevance of recommendations, and overall satisfaction with the system. User satisfaction surveys can help identify areas for improvement and provide valuable feedback for future development. Furthermore, they foster a direct and constructive dialogue between users and developers, fostering an environment for continuous enhancement and refinement of the system to align with evolving user expectations.

Evaluating a recommending system (RS) requires both identifying the characteristics that make an excellent RS and identifying how they should be quantified. The most commonly used metrics for evaluating a RS's performance, especially when they use ML algorithms, are recall, precision, accuracy, ROC curves, and F-measure, as they have already been analyzed in [33].

2.5.1. Recall and Precision

Precision and recall are two metrics used worldwide in order to characterize a recommender system as successful or unsuccessful based on the recommendations they provide. Recall is the proportion of user-recommended visuals that are appealing to the user out of all the visualizations that should be recommended, and precision is the percentage of visualizations that the user finds appealing overall. Confusion matrix computation is used to determine precision and recall measures. If the user is interested in the proposed visualization, it will be regarded successful; if not, it will be considered unsuccessful. The confusion matrix illustrates the four possible outcomes ('a', 'b', 'c' and 'd') of each recommendation.

Where "a" denotes the number of genuine positive visualizations that the RS has successfully obtained and recommended for recommendations. The value of "b" represents the number of visualizations that the RS recommended but that were ultimately rejected. The amount "c" denotes the number of RS-recommended disqualified visualizations. The number of objects that are marked and retrieved as "not recommended" is represented by the true negative value, "d". The precision and the recall are computed as following:

$$Precision = \frac{a}{a+b}$$

$$Recall = \frac{a}{a+c}$$

It should be mentioned that a good VRS tries to optimize both metrics simultaneously. For instance, it can recommend many visualizations to the user, obtaining maximum coverage. The Precision would still be as low as the ratio of useful visualizations in the pool.

2.5.2. Accuracy

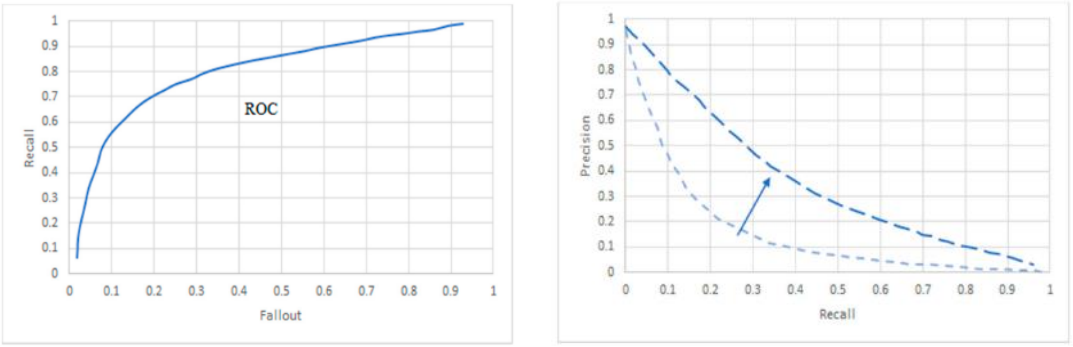
The organization's requirements must be taken into account while choosing a VRS based on an evaluation metric. Predicted ratings are typically employed as a system evaluation metric. Because there is no specific way to judge whether a recommendation is precise or not, determining the correctness of VRS is not simple [42]. By applying split-validation of data for offline comparisons, one must look for minimal prediction errors in order to evaluate a VRS's accuracy. Consider the method where we give VRS the 80% of a user's visualization history datasets and ask it to anticipate the remaining 20%. In that situation, we may determine the correctness of the system using real recommendations.

$$Accuracy = \frac{Number\ of\ successful\ recommendations}{\text{Total number of recommendations}}$$

Accuracy is used for evaluation in many cases; for example, in scoring an algorithm, the root of the mean square error (RMSE) is used [43]. Other alternatives of the RMSE are the mean average error, the normalized mean average error, and mean square error. RMSE is most appropriate since it measures all ratings' inaccuracies, whether they are positive or negative. RMSE is recommended to be used in cases where the errors cannot be differentiated.

2.5.3. ROC Curve

Precision and recall can be replaced by ROC (receiver operating characteristic) analysis. The recall values decrease as the precision increases. The goal of ROC analysis is to recover the visualizations that are the most engaging while avoiding retrieving the uninteresting ones [44]. This is accomplished by maximizing recall—also known as the true positive rate—and minimizing fallout, also known as the false positive rate. When the threshold is modified, the trade-off between recall and precision is visually represented using ROC curves, which allows us to categorize a visualization as "to be recommended" or "not to be recommended" [45] The ROC, accuracy, and recall curves can all be optimized similarly. Pushing the peak of the curve in the direction of precision = 1 and recall = 1 will optimize the recall and precision values (Figure 2). The output of a perfect prediction system is a ROC curve that eventually reaches all of the intriguing visuals before turning to the remaining visualizations. As with precision and recall measurements, ROC curves presume binary relevance. One of two categories, a successful recommendation or an unsuccessful recommendation, is used to classify visualizations. However, taking this into account means that the ROC measure is unaffected by the arrangement of intriguing visuals. An ideal ROC curve is produced when all pertinent visualizations appear before the uninteresting visualizations [16]. We can examine the region under the ROC curve to use it as a performance indicator [46]. The likelihood of the system properly choosing between two visuals, where one visualization is randomly selected from the set of acceptable visualizations and the second visualization is selected from the set of unsatisfactory visualizations, is represented by the area under the ROC curve.



(a) (b)

Figure 2. (a) ROC curve for the recall against fallout, (b) Simultaneous optimization of the precision and recall.

2.5.4. F-Measure

Another statistic generated from accuracy and recall is the F-measure, which behaves similarly to both of those metrics. F-measures may be a more useful statistic than precision and recall since they provide different information that, when combined, can complete each other. The metric that performs better than the others will be reflected in the F-measure. The F-measure is

the number of tests that must be run in order to find the first failure from the perspective of probability [47]. The value of F can be changed to give one metric more importance than another. The consistent mean of precision and recall (F1), where 1 is the most popular F-measure, Because the F-measure has a maximum value of 1, all forecasts and recommendations are accurate. Understanding ranked retrieval is helpful. Recall is the ratio of the top k relevant things in this case, and Precision is the ratio of the top k relevant visualizations. The user is assumed to only look at the top-k results when using ranked retrieval. Knowing this, [48] demonstrates that the upward tendency of the recall value will lead the F-measure to increase as the value of k increases. Equations (2) and (3) display the results of the F and F1 calculations.

$$F_\beta = \frac{precision * \ recall}{(1-b) * precision + \beta * recall}$$

$$F_1 = \frac{2 * precision * recall}{precision + recall}$$

You can get a thorough picture of the success of the visualization recommendation system and pinpoint areas for development by combining various evaluation techniques. In order to maintain the system's performance and value for users, it is crucial to continuously review and improve it.

Previous studies indicate that effectiveness is ongoing. For instance, Cleveland and McGill utilize absolute mistake rates to assess performance on simple perceptual tasks [31], Saket et al. use time and accuracy preference to assess task performance [27], and Borkin et al. employ a normalized memorability score. Multiple visualizations can display the same data equally well, according to discussions by visualization specialists [32, 29].

In conclusion, visualization recommendation algorithms provide a potent means of deciding which form of visualization is most appropriate for a given datasets. You may develop an efficient visualization recommendation system that gives users useful insights and value by putting best practices for data pretreatment, algorithm selection, and user interface design into action and constantly assessing the system's efficacy.

3 Methodology

Nowadays, analysts have to deal with high dimensional datasets, that makes it difficult for them to handle all the information and to find interesting insights and patterns. For this purpose, we created an approach that interacts with the users and helps them to make visualizations much easier, based on their needs. It is designed to support not only input datasets that are spreadsheets of multi-dimensional data, which are the most common type of datasets, but it especially supports tabular data that have columns/variables containing geospatial data (latitude, longitude). So, in this chapter we will present the methodology we used to create this approach. This means that we will discuss topics such as (i) the choice of hexagons for the creation of grid – based visualizations, (ii) the way the reference view and the target views were created, (iii) how the utility function works, (iv) which deviation – based metrics should we choose and (iv) how the utility score helps us understand the level of interestingness of a certain target view.

3.1 A quick overview

Our approach aims to provide efficient and effective recommendations of the most interesting hexagons-areas based on the users' preferences to help them for their analysis. It works, generally, with a four-step process accomplishing some different methods, which we will discuss later in this master thesis. Firstly, (a) it creates hexagons based on the geospatial data existing. It (b) uses a group-by clause over the datasets for the variables and the aggregation function that the users are interested in creating, in this way, the reference view. Based on the same group-by clause it creates the target views for every unique hexagon. Then it (c) computes the utility score between the reference view and the target views. Finally, to visualize the top-k most interesting hexagons (d) it shows different charts e.g. maps, bar-charts etc. These steps are also shown in the next image (Figure 3) that is actually a flow diagram of how our approach works.

These datasets can be modeled as a set of measure attributes that contain numeric values, as well as a set of dimension attributes that contain categorical values. In our implementation, data's quantitative attributes are classified as measures, while nominal and temporal ones are classified as dimensions. Usually, a view (i.e., histogram or bar chart) represents a query, which in our case is expressed and evaluated using pandas (python library), with a group-by clause over a database D. Data can be modeled as a set of measure attributes M = {m1, m2, m3, ...} and a set of dimension attributes A = {a1, a2, a3, ...}, as well as a set F = {f1, f2, f3, ...} of standard Python aggregate functions that are applied to generate an aggregate query/view with a group-by clause. The measure attributes are the set of attributes that contain measurable value and can be aggregated. The dimensional attributes are the set of attributes on which measure attributes are viewed. So, we can represent each view as a triple (a, m, f), such that one aggregate function "f" is applied on dimension attribute "a" over the corresponding measure attribute "m". In our case the set (A, M, F) is defined by the user. Thus, the total number of possible views is: |A|×|M|×|F|.

In order to discover and recommend the set of k most interesting views from a large number of views, utility scores are required to rank the views. To compute such utility scores, existing literature has proposed several utility functions (UFs). Some commonly used UFs include deviation [71], accuracy [5], usability [5] and p-value [3]. Our recommendation system supports breadth and depth exploration with deviation-based utility function. Deviation is commonly used to measure the interestingness of the charts recommended during visualization [18, 23, 41]. Especially, in our first implementation of our method, we decided the calculation of the utility scores to be based on the calculation of the Euclidean distance between the reference view and every individual target view. Then, after some observations, we decided to use an updated deviation-based utility function made of more than one variable. It was actually based on four different distance metrics, including Euclidean distance as one of them, that does not lead to a single clear-cut answer. The complexities of data analysis and visualization demand a multi-faceted approach, and our utility function aims to capture these diverse aspects.

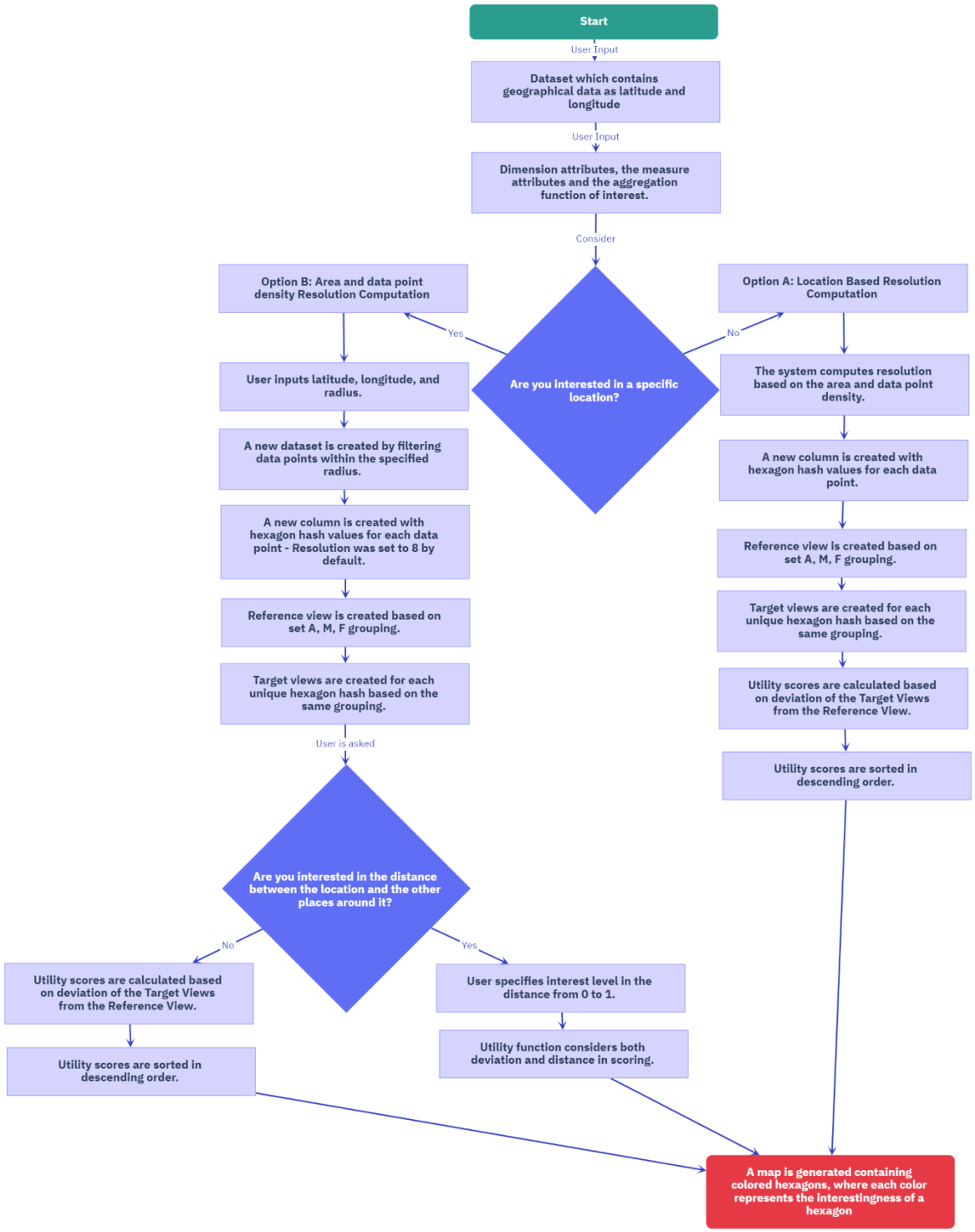


Figure 3. Flow diagram of our approach. It shows how our approach works step by step.

3.2 Creation of hexagons

As said, the first step of our problem formulation refers to the creation of hexagons that cover a certain area. For this purpose, H3 helped us to implement this step. H3 [70] is an open source framework developed by Uber in the C Programming Language. At its core, H3 is a geospatial analysis tool that provides a hexagonal, hierarchical spatial index to gain insights from large geospatial datasets. The building blocks of H3 are different sized regular hexagons. Any location on the planet can be attributed to a H3 Hexagon down to a precision of 0.0000009 km² area. Each H3 hexagon can be thought of as its own object and each object can be accessed in a very short amount of time given its ID. H3 contains a total of 16 resolutions as described in the table below, and each resolution has a certain number of hexagons that span the entire earth as a layer ranging from 122 hexagons in the highest layer and about 569 trillion hexagons at the lowest layer (Figure 4). Each layer consists of a more granular level of hexagons and each hexagon of every layer has its own unique ID and can very quickly perform geospatial calculation.

H3 Resolution	Average Hexagon Area (km^2)	Average Hexagon Edge Length (km)	Number of unique indexes
0	4,250,546.8477000	1,107.712591000	122
1	607,220.9782429	418.676005500	842
2	86,745.8540347	158.244655800	5,882
3	12,392.2648621	59.810857940	41,162
4	1,770.3235517	22.606379400	288,122
5	252.9033645	8.544408276	2,016,842
6	36.1290521	3.229482772	14,117,882
7	5.1612932	1.220629759	98,825,162
8	0.7373276	0.461354684	691,776,122
9	0.1053325	0.174375668	4,842,432,842
10	0.0150475	0.065907807	33,897,029,882
11	0.0021496	0.024910561	237,279,209,162
12	0.0003071	0.009415526	1,660,954,464,122
13	0.0000439	0.003559893	11,626,681,248,842
14	0.0000063	0.001348575	81,386,768,741,882
15	0.0000009	0.000509713	569,707,381,193,162

H3 Resolutions

Figure 4. The area a hexagon covers, the edge length of a hexagon and the number of unique hexagons that are created based on each resolution.

The layered approach of different sized hexagons is what lends H3 its power of "hierarchy". Every low-resolution hexagon contains a set of child hexagons in higher resolutions. Every hexagon of a resolution has sibling hexagons that share the same set of parent hexagons. The layers in essence define a tree of hexagons with the last layer (resolution) containing 569 trillion siblings. The resolution can be adjusted accordingly and the hexagons can be viewed over the entire earth.

Selecting a resolution is not something easy to be done, since the idea of partitioning the space in to smaller areas (hexagons) should be something meaningful that has a reasonable impact in the scientists' studies. For instance, if the given datasets has points from all over the world, the analyst may find useful setting a resolution from 3 to 5, to create hexagons that probably overlap a "big" amount of data points since they cover a bigger area each one. From the other hand, if they set a high resolution such as 9 – 10 or above, the hexagons that are going to be created will cover small areas like neighborhoods, that this will lead unreliable results because of the small amount of data points each hexagon contains and the heterogeneity among the countries, the cities, the cultures and so on. So, selecting the right resolution is something that really matters and the wrong choice will probably lead to ineffective conclusions.

When choosing the resolution for creating hexagons or any other spatial grid, there are several factors the analyst needs to consider. Some of these key factors to study before deciding on the resolution refer to the scale and density of the data, the level of detail the analyst wants to represent as well as the performance requirements.

- **Data scale**: If the datasets covers a large geographical area, the analysts may want to choose a lower resolution to reduce the number of hexagons and improve performance.

On the other hand, if the datasets focus on a smaller region, they can select a higher resolution to capture more localized details.

- **Data density**: If the datasets has a high density of points or features across the geographic extent, a higher resolution can help capture and visualize the finer details. Conversely, if the density is low, a lower resolution may be sufficient.
- **Level of detail**: It refers to the level of detail the analyst wants to represent on the map. This means that higher resolutions allow for more precise representation of features and variations in the data, while lower resolutions provide a more generalized view.
- **Computational Resources**: The analysts have to consider the computational resources available to them. For instance, higher resolutions will result in more hexagons and potentially larger datasets, requiring more processing power and memory. So, they should ensure that their system can handle the computational demands of working with the chosen resolution.
- **Visualization Requirements**: Another factor that should indicate in the decision of selecting the most suitable resolution refers to the way the analyst plans on how to visualize the data. Higher resolutions may result in visually cluttered maps if there are too many hexagons. On the other hand, lower resolutions may oversimplify the data representation. Find a balance that allows for clear and meaningful visualization.
- **Data Accuracy**: The evaluation of the accuracy and precision of the data is usually a handy process. A very high-resolution grid may create an illusion of precision if the underlying data is not accurate or precise enough. The analysts should consider the limitations and uncertainties associated with their data when selecting the resolution.
- **Spatial Analysis Goals**: Determine the specific goals of the spatial analysis. Different resolutions may be suitable for different objectives. For example, if the analysts are interested in identifying broad-scale patterns, a lower resolution may be sufficient. On the other hand, if they need to capture fine-scale spatial relationships, a higher resolution may be necessary.
- **Data Size and Storage**: Furthermore, the analysts should take into account the size of the datasets they are interested in studying and its storage constraints. Higher resolutions will generate more hexagons and potentially larger datasets, requiring more storage space. Ensure that you have enough storage capacity to accommodate the chosen resolution.

It's recommended to experiment with different resolutions and evaluate the visual output and performance to determine the optimal choice for a specific dataset and use case. If the resulting hexagons are too large and not capturing the desired level of detail, you can try increasing the resolution. However, keep in mind that higher resolutions will significantly increase the number of hexagons and could impact performance. So, by considering these factors and understanding their specific analysis requirements, the analysts can make an informed decision about the resolution to use for creating hexagons or any other spatial grid.

3.3 Target – Reference View

In a visualization recommendation system, reference view and target view are two important concepts that are used to recommend suitable visualizations for a given datasets. Reference view refers to an existing visualization that the users have already created or chosen as a starting point. The reference view can be used to guide the recommendation system to suggest similar or related visualizations that can help the users explore their data more effectively. The reference view can be used to convey information about the data types, dimensions, and other properties of the data that are relevant for the recommendation system. Target view, on the other hand, refers to the type of visualization that the analysts want to create or explore. The target view may

be specified by the users' explicitly or implicitly, based on their preferences or goals. The recommendation system can use the target view to suggest suitable visualizations that can meet the users' needs and requirements. The target view can be specified by selecting one or more visual encodings, such as bars, lines, scatterplots, or heatmaps, and by specifying the data dimensions that need to be visualized.

By combining the reference view and target view, the recommendation system can generate a set of candidate visualizations that are similar or dissimilar to the reference view and that satisfy the requirements of the target view. For instance, in the next map (Figure 5) we can see colored hexagons that each one represents a target view and their color corresponds to its interestingness that is a resulting of each one's contrast to the reference view, where the reference view represents what happens as an average behavior at the whole dataset. The recommendation system may also take in to account other factors, such as the size of the dataset, the complexity of the data, and the users' interaction history, to generate personalized and relevant recommendations.

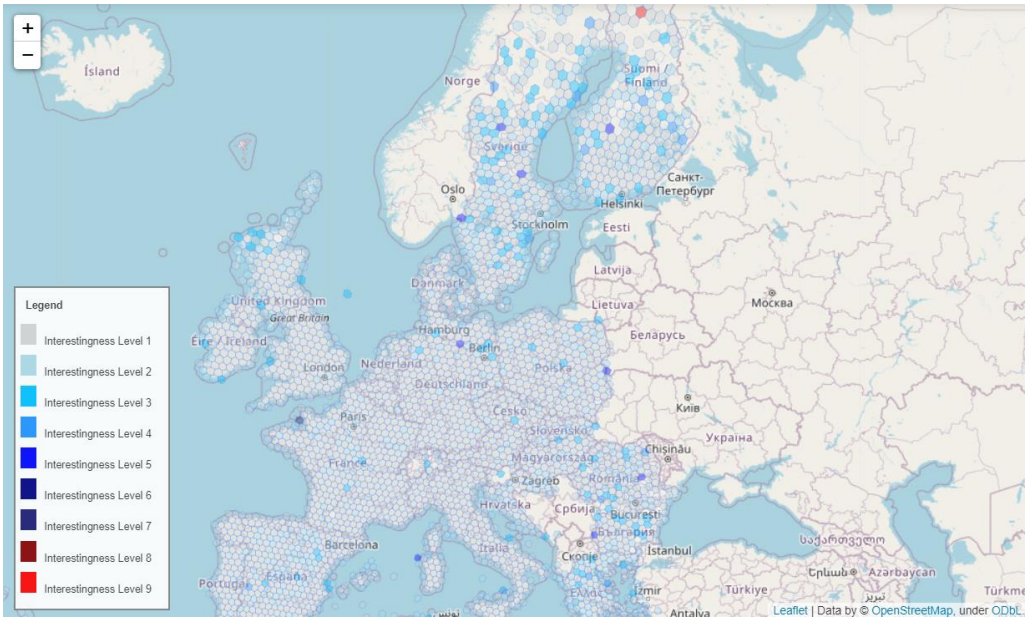


Figure 5. Hexagons that each one corresponds to a target view and that shows the interestingness of each hexagon based on the user's criteria.

The recommendation system may also incorporate feedback from the users, such as ratings or preferences, to improve its recommendations over time. By learning from the users' interactions and preferences, the recommendation system can become more personalized and adaptive to the users' needs.

It's worth noting that reference view and target view are not always necessary for a visualization recommendation system. Some systems may rely on other sources of information, such as the characteristics of the data or the users' interaction history, to generate relevant recommendations. However, reference view and target view can be useful tools for users who have a clear idea of what they want to visualize and who want to explore different options based on their existing knowledge and preferences.

Reference view and target view can be created in a variety of ways, depending on the visualization tool or system being used. Some common ways to create reference view are (a) by importing an existing visualization from a file or online source, such as a CSV file, an Excel sheet, or a chart from a website, (b) by creating a new visualization from scratch, using the built-in tools and features of the visualization software, this can involve selecting a datasets, choosing a visual

encoding [68], and customizing the appearance and layout of the visualization, (c) by using a pre-defined template for example visualization as a starting point, and modify it to suit your needs, (d) by selecting a visualization from a gallery or library of visualizations, and use it as a reference for further exploration. As for the target view, it can be created via many ways such as (a) by using a query interface to specify the data dimensions and filters that need to be visualized, this can involve selecting variables, defining conditions, and specifying aggregation or grouping options, (b) via drag and drop variables or data fields onto a canvas, and use a visual encoding menu to select the type of chart or graph that best represents the data (c) entering a natural language query or command, using tools such as chatbots or voice assistants, to generate a visualization based on your verbal input, (d) browsing a gallery or library of visualizations, and select a pre-defined visualization that matches your needs and preferences.

Some other, more specific - programming ways to create the reference view and the target views are:

1. Random sampling: The reference view and target views can be created by randomly selecting a subset of the data. This can be a simple and fast approach, but it may not provide a representative sample of the data and could lead to biased results.
2. Clustering: The reference view and target views can be created by clustering the data into groups based on similarities between data points. This can be useful for identifying patterns or clusters of similar views, but may be computationally intensive and require careful selection of clustering algorithms and parameters.
3. Principal Component Analysis (PCA): The reference view and target views can be created by performing PCA on the data to identify the most important dimensions or features. This can be useful for reducing the dimensionality of the data and identifying key patterns or relationships, but may require expertise in statistical analysis and data modeling.
4. Domain-specific criteria: The reference view and target views can be created based on domain-specific criteria or guidelines, such as data privacy or regulatory requirements. This can be useful for ensuring compliance and consistency, but may be subjective and require careful consideration of different stakeholder perspectives.

In some cases, the reference view and target view may be created simultaneously, as the users explore different options and refine their visualization goals. The visualization recommendation system can then use this feedback to generate new recommendations or refine existing ones.

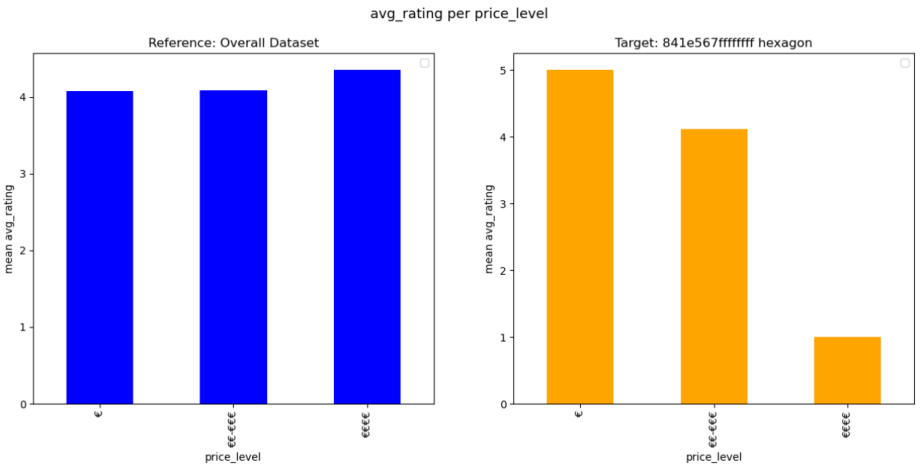


Figure 6. Separated bar charts showing the comparison between target view (hexagon "841e567fffffff") and reference view average rating values per their respective price level.

In our case, we created both the reference view and the target view using group-by clauses. This means that views rely on groupings of values that are present in the data set. We

ask from the analysts to input the dimension attributes they are interested in studying, as long as a measure attribute and an aggregation function as shown in the next figure. Then, we create the reference view by using these inputs into a group-by clause referring the whole datasets. Furthermore, we create the target views based on the same set and clause but for every hexagon created at the previous step. So, we put in contrast every target view (hexagon) with the reference view and return the most similar, with the reference view, target views like in Figure 6.

Creating the reference view and target view based on the group-by clause, given dimension attribute, measure attribute, and aggregation function is a good way to generate informative and meaningful visualizations. By specifying the dimension attribute and aggregation function in the group-by clause, we can group the data by different levels of granularity, such as by year, month, or day. This allows us to explore the data at different levels of detail and uncover patterns and trends that may not be visible at the aggregate level. By selecting a specific measure attribute to visualize, we can focus the visualization on a specific aspect of the data that is of interest or importance. This helps to make the visualization more meaningful and informative for the intended audience. By using a visual encoding such as color or shape to represent different groups or categories within the data, we can easily compare and contrast the values of the measure attribute between different groups. This can help to identify similarities and differences between groups and highlight areas of interest or concern. Additionally, because the reference view and target view are based on the group-by clause, they can be easily customized and iterated upon by changing the dimension attribute, measure attribute, or aggregation function. This makes it easy to explore different aspects of the data and refine the visualization to better meet the needs of the intended audience. So, summarizing the above, it (a) allows the exploration of data at different levels of granularity, (b) it provides a clear focus for the visualization, (c) it facilitates comparison between groups and (d) it enables easy customization and iteration.

Some other advantages that this way of visualization has are that it helps to identify outliers, enables easy filtering and drilling down, it provides a clear structure for the visualization, it facilitates collaboration and communication, supports statistical analysis, enables time series analysis, helps to identify data quality issues and enables dashboarding and reporting and many other benefits.

Overall, creating the reference view and target view based on the group-by clause, given dimension attribute, measure attribute, and aggregation function is a good way to create informative and meaningful visualizations that help to explore and communicate data effectively.

In our study, we decided to create the target views based on each unique hexagon. This means that each target view represents the behavior of each hexagon individually and it is created by the same group-by clause as the reference view.

3.4 Utility Function

Utility functions are mathematical functions that assign a numerical value to a particular action or state. In the context of recommendation systems, a utility function is used to estimate how much a user will like a visualization based on their preferences and historical behavior.

Utility functions are important in visualization recommendation systems too, because they provide a way to measure the usefulness of a particular visualization for a specific user or group of users. By using a utility function, the recommendation system can suggest visualizations that are likely to be most relevant and useful for a particular user, based on their preferences and past interactions with the system.

There are different types of utility functions that can be used in visualization recommendation systems, including:

1. Content-based utility functions: These functions use the characteristics of the data being visualized to estimate how useful a visualization will be for a particular user.
2. Collaborative filtering utility functions: These functions use data from other users to estimate how useful a visualization will be for a particular user.

3. Hybrid utility functions: These functions combine content-based and collaborative filtering approaches to provide a more accurate estimate of how useful a visualization will be for a particular user.
4. Contextual utility functions: These functions take into account contextual information, such as the time of day, location, or device being used, to recommend visualizations that are most relevant and useful for a particular user in a specific context.
5. Social utility functions: These functions use social network data to estimate how useful a visualization will be for a particular user based on the preferences and behavior of their social connections.
6. Implicit feedback utility functions: These functions use implicit feedback, such as a user's clicks or browsing history, to estimate how useful a visualization will be for a particular user.
7. Multidimensional utility functions: These functions take into account multiple factors, such as user preferences, contextual information, and social network data, to recommend visualizations that are most relevant and useful for a particular user in a specific context.

There are also many other types of utility functions, but these are the most used ones. The choice of which type of utility function to use depends on the specific goals and context of the visualization recommendation system, as well as the available data and resources and we analyzed them in depth at the chapter “2.1 Types of recommendation systems”.

3.4.1 Euclidean Distance

In our case, we firstly, used the Euclidean distance in order to measure the utility scores and use them to define the most interesting views. Euclidean distance is a commonly used measure in utility functions, particularly in content-based filtering approaches for visualization recommendation systems. Euclidean distance is a mathematical measure that calculates the distance between two points in a multi-dimensional space [47].

By representing visualizations as points in a multi-dimensional space, the Euclidean distance quantified the spatial differences, distinguishing similar visualizations as closer points and dissimilar ones as more distant. Its intuitive interpretation, resilience against outliers, and capacity for parallel computation on extensive datasets made it an ideal choice. Furthermore, the versatility of Euclidean distance enabled its application to various data types, including numerical, categorical, binary, and even time series or graph data, making it an indispensable tool for recommendation systems dealing with diverse visualizations. One more example that proves the broadly use of Euclidean distance refers to the fact that the k-means clustering algorithm uses the Euclidean distance [48,49] to measure the similarities between objects. However, the choice of similarity metric should be carefully tailored to the specific characteristics of the data and the goals of the recommendation task.

Euclidean Distance between two points, A(x_1, y_1, ..., n_1) and B(x_2, y_2, ..., n_2), in an n-dimensional space is calculated as:

$$d(A,B) = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2 + \ldots + (n_2 - n_1)^2}$$

Where,

- $d(A,B)$: *Euclidean Distance between points A and B*
- $x_i, y_i, \ldots, n_i$: *Coordinates of points A and B in each dimension*

3.4.2 Other Metrics

It's important to note that Euclidean distance is not the only measure used in utility functions. Other measures, such as cosine similarity, Pearson correlation, or Jaccard similarity etc., may

also be used depending on the specific requirements and goals of the visualization recommendation system. Some of the most known and widely used measures are:

i. **Cosine similarity** emerges as a pivotal metric in recommendation systems, bridging gaps in both text-based and visualization-focused applications. Its significance lies in its ability to measure the angle between vectors [47], facilitating comparisons between documents, images, or feature vectors in high-dimensional spaces. In text-based recommendation systems, cosine similarity enables the identification of semantically similar documents, accommodating varying lengths and frequencies. Similarly, in visualization contexts, it gauges the resemblance between visualizations by comparing their feature vectors, proving instrumental in clustering algorithms and classification tasks. However, Cosine similarity falls short when faced with processes involving magnitude comparisons, weighted features, or negative correlations, so it should be avoided in these cases.

Cosine Similarity between two vectors, $\vec{A} = (A1, A2, \dots, An)$ and $\vec{B} = (B1, B2, \dots, Bn)$is computed as:

$$Similarity(\vec{A}, \vec{B}) = \cos(\theta) = \frac{\vec{A} \bullet \vec{B}}{\|\vec{A}\| * \|\vec{B}\|} = \frac{\sum_{i=1}^{n} A_i * B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} * \sqrt{\sum_{i=1}^{n} B_i^2}}$$

Where,

- Similarity (A, B): Cosine Similarity between vectors A and B
- A • B: Dot product of vectors A and B
- ||A||, ||B||: Magnitude (norm) of vectors A and B

ii. **KL divergence** is a vital measure in quantifying the difference between two probability distributions. In the realm of visualization, it compares data distributions or probability density functions among different views, aiding in identifying unique characteristics or variations. It serves as a tool to assess the effectiveness of visualization techniques by evaluating how accurately they represent underlying data distributions and is instrumental in detecting outliers, pinpointing anomalous data points in visualizations.

However, it's crucial to recognize its limitations. KL divergence is asymmetric, meaning [57] results vary based on the order of compared distributions, and requires identical support for accurate comparison. Sensitivity to small sample sizes, lack of direct intuitive interpretation in real-world units, and the need for careful consideration before application are significant aspects to bear in mind. Despite its complexities, KL divergence remains invaluable for in-depth analysis and evaluation of data distributions in visualization contexts. The KL divergence has also been adopted in some CF algorithms [58 – 61]. KL Divergence between two probability distributions P(x) and Q(x) is computed as:

$$DKL(P||Q) = \sum P(x) * \log(\frac{P(x)}{Q(x)})$$

Where,

- DKL (P || Q): KL Divergence between distributions P and Q
- P(x), Q(x): Probability density functions at point x

iii. **Manhattan distance**, also known as taxicab distance or L1 distance, measures the sum of absolute differences between the coordinates of two points. In visualization recommendation systems, Manhattan distance can be used to measure the similarity between two visualizations based on how many changes in direction are required to get from one visualization to another. It is particularly well-suited for grid-based or grid-like visualizations, where the data is organized in a structured manner, such as heatmaps, chessboard-like grids, or pixel-based representations. It can be also useful in path planning algorithms, such as the A* algorithm, where diagonal movements are not

allowed or have different costs. It is commonly used in methods where movements are constrained to grid-based environments. Furthermore, in many cases, Manhattan distance is used to address the fat tail problem by minimizing the error caused by the fat tail like in [51, 55].

However, if diagonal movements are important in the analysis or if the analyst needs a metric that captures distance in all directions Manhattan distance should not be used. It may not be the best choice for continuous data or non-grid-like visualizations. It, also, does not take into account the scale or range of the data [51]. Additionally, in clustering processes the selection of distance metric plays a very important role and should be made carefully. For example, as well explained in [50], the distortion in k-means using Manhattan distance metric was less than that of k-means using Euclidean distance metric. Manhattan Distance between two points, A(x_1, y_1, ..., n_1) and B(x_2, y_2, ..., n_2), in an n-dimensional space is calculated as:

$$d(A,B) = |x_2 - x_1| + |y_2 - y_1| + \ldots + |n_2 - n_1|$$

Where,

- $d(A,B)$: *Manhattan Distance between points A and B*
- $x_i, y_i, \ldots, n_i$: *Coordinates of points A and B in each dimension*

iv. Another well-known deviation-based utility function is the **Mean Absolute Deviation (MAD).** MAD provides a measure of the average absolute difference between each data point and the mean. It is less sensitive to extreme values or outliers compared to other deviation-based measures like standard deviation. By calculating the MAD between corresponding data points between two vectors, you can quantify the average deviation or difference between them. A higher MAD value indicates a greater deviation or interestingness. MAD is particularly useful when the analyst wants to measure the dispersion or spread of data points in a visualization. It provides a robust measure of variability, especially in the presence of outliers or extreme values or non-normal distributions. Additionally, because MAD treats deviations in any part of the scale identically, it is preferred to squared-error-based measures for ordinal data [63]. Furthermore, several metrics, such as MAD, have been used to evaluate RS under different notions of fairness [64, 65]. The formula for calculating MAD is as follows:

$$MAD = \frac{1}{N} * \sum |x_i - \mu|$$

where:

- MAD is the Mean Absolute Deviation
- N is the number of data points
- x_i is each individual data point
- μ is the mean of the data points

However, MAD does not have a direct interpretation in the original units of the data. It is typically expressed in the same units as the data, but its magnitude does not convey the same intuitive meaning as the standard deviation. MAD can also be affected by small sample sizes. As the number of data points decreases, the robustness of MAD may diminish. In such cases, it is advisable to consider the appropriate sample size and assess the reliability of the MAD estimate.

There are also many other metrics that can be used as utility features of a utility function that measure the similarity between two data points. Some of them are **Jaccard distance** that measures the dissimilarity between two sets by calculating the ratio of the size of the intersection of the sets to the size of the union of the sets, the **Mahalanobis distance** that measures the distance between two points in a multi-dimensional space, taking into account the correlation between the dimension and. **Pearson correlation** that measures the linear correlation between two variables and is commonly used to measure similarity between continuous variables.

A summary of some of the most used metrics is shown in the next table (Table 1).

Table 1. Metrics and use cases

Metric	Use Case	Avoid
Euclidean Distance	Comparing similarity/dissimilarity of data points	When dimensions have different scales or magnitudes
KL Divergence	Comparing differences in data distributions	When distributions have different supports or are asymmetric
Manhattan Distance	Grid-based or grid-like visualizations	When diagonal movements are important or data is not grid-based
Cosine Similarity	Assessing similarity based on vector orientation/direction	When magnitude/length of vectors is crucial or negative correlations are important
MAD	Assessing dispersion and dealing with outliers	When widely recognized metrics are required or for small sample sizes

Summing up, some utility features, that are commonly used, are the deviation between target view and reference view with different distance measures: Euclidean distance, Kullback-Leibler divergence (KLdivergence), Earth Mover Distance (EMD), L1 distance, L2 distance and the maximum deviation in any individual bin. Some other utility features represent the usability [5], accuracy [5], and p-value [3]. Usability refers to the quality of the visualization in terms of providing the analyst with an understandable, uncluttered representation, which is quantified via the relative bin width metric. Accuracy refers to the ability of the view to accurately capture the characteristics (i.e., distribution) of the analyzed data, which is measured in terms of Sum Squared Error (SSE). The p-value is a statistical term defined as “the probability of obtaining a result equal to or more extreme than what is actually observed, with the given null hypothesis being true” [7]. In the problem of view recommendation, the null hypothesis refers to the reference view, and the extremeness of the results refers to the interestingness of the target views. So, we could say that in visualization and data analysis, the choice of utility metric depends on the specific context, requirements, and characteristics of the data being analyzed and there is not a single metric that is universally used in most cases.

3.4.3 Utility score

The utility score can show something interesting by identifying visualizations that are similar to the users’ query visualization or even visualizations that are dissimilar to their query in terms of the underlying data points, dimensions, and characteristics. For example, if a user submits a scatter plot showing the relationship between two variables, the recommendation system can calculate the utility score between the data points in the query visualization and the data points in other visualizations in its database. The system can then rank the visualizations based on their similarity to the query visualization, with the most similar visualizations receiving the highest utility scores.

This approach can help users discover new visualizations that are similar or dissimilar to their query visualization but may show different aspects of the data or may be presented in a different format. Additionally, deviation – based utility features, that we decided to use in our approach, like the Euclidean distance, the Manhattan distance etc. can provide a measure of how much the recommended visualization deviates from the query visualization, which can help users

gauge the relevance of the recommendations and make informed decisions about which visualizations to explore further. So, we could say that in a visualization recommendation system, the utility score is a measure of similarity between a query visualization and the recommended visualizations. Based on the utility features used in the utility function either the higher the utility score is, the more dissimilar the recommended visualization is to the query visualization or the lower the utility score is the more similar they are. In other words, the utility score indicates which visualization is more relevant and useful to the users' original query and is therefore more likely to be of interest to the users. The system can use the utility scores to rank and filter the recommended visualizations, so that the most similar and useful visualizations are presented to the user first. Common distance metrics were noticed earlier in chapters 3.4.1 and 3.4.2. This score is typically a numerical value between 0 and 1.

In our case, we decided to compare all the candidate distance metrics we had in mind that captured many different aspects of the data. So, observing all the results, the individual utility features provided, we came up with the idea of creating a utility function mixed of four different distance metrics to measure the interestingness of each view. This utility function combines some of the features we already talked about. Specifically, this updated utility function takes into account four deviation – based utility features; the (a) Euclidean distance that at the beginning it was the only utility feature we chose to use, the (b) Manhattan distance, the (c) KL divergence and the (d) Mean Absolute distance (MAD). The results they provided will be further discussed in chapter 5.

Summing up, utility scores are typically calculated based on a combination of different factors, including similarity, novelty, diversity, and relevance to the users' query. The relative importance of these factors can vary depending on the specific system and context and may be adjusted based on user feedback or other factors. The choice of distance metric used to calculate the utility score can have a significant impact on the quality of the recommendations. We will discuss in the next chapters the way we chose distance metrics to use as features to compute the utility score of the visualizations and suggest the most interesting ones. In addition, other factors such as usability, performance, and accessibility can also play an important role in determining how useful and effective the system is for users.

4 Data Exploration Methods

Data exploration is a critical step in the data analysis process, where we try to identify patterns, trends, and relationships within the data. This step is essential for developing hypotheses and making informed decisions. In this master thesis, we tried to create a visualization recommendation system that explores different exploration methods using reference and target views and highlight their applicability to various data exploration contexts. Specifically, we will consider three methods (a) global reference view and global target views; (b) local reference view and local target views and (c) local reference view and local target views with spatial distance.

4.1 Method 1: Global Reference View and Global Target Views

In this method, both reference and target views are computed using the entire datasets, identifying interesting regions in the whole datasets. This method is suitable for understanding global patterns and trends. For instance, consider a dataset containing information about worldwide temperature patterns. The global reference view would refer to the entire world, and the target views would refer to individual hexagons, where each one represents different regions, such as continents, countries, cities or even a block of them like 3 to 4 nearby cities. By evaluating interestingness scores for each target view, we can identify the most interesting regions and gain insights into global temperature patterns like in Figure 7.

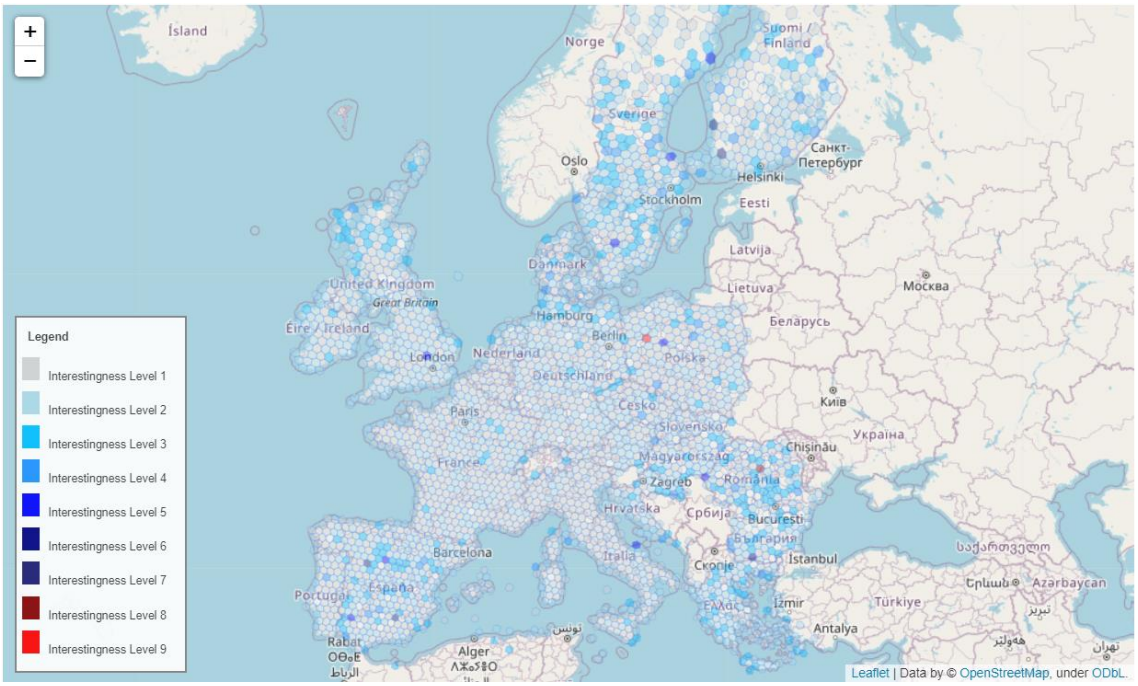


Figure 7. Map showing the interestingness of every hexagon based on the user's criteria and the whole under study data set.

Firstly, analysts input the datasets they are interested in studying. Then, the method, in order to cover the space given from the datasets with the most suitable number of hexagons defines the proper resolution. For this purpose, we created a function that applies two major steps (a) it finds out the area that the datasets points cover and (b) it computes the density of the data points in every 1km^2, where the density is measured as the total number of points contained in the datasets divided by the area we previously computed. Based on these two "key" factors, the resolution is set and then the hexagons are being created. The choice of 1km² as the area for

density calculation was made to establish a standardized unit widely used in geographical analyses. This unit facilitates easy comparison and interpretation of density values across different datasets and geographical regions. The resolution of hexagons is calculated by computed the total area and density by the following formulas and some given thresholds.

$$Total\ area\ = width * height *\ 6371^2$$

Where:

$Total\ area$: $the\ total\ area\ covered\ by\ data\ points\ (in\ km^2\ )$
$width = \max lon - \min lon\ \ (\max lon\ and \min lon\ \ in\ radians)$
$height = \max lat - \min lat\ (\max lat\ \ and \min lat\ \ in\ radians)$
6371: the Earth′s radius

and

$$Density = \frac{Total\ number\ of\ data\ points}{Total\ area\ covered\ by\ data\ points\ (in\ km^2)}$$

We, experimentally, found out that the area and density thresholds that will work the best as boundaries to select the resolution which varies from 4 to 8. We decided that a resolution less than 4 does not interest us since there are being created huge hexagons that do not offer the details needed and generates a very general set of information. From the other hand, a resolution more than 8 usually drives us into a neighborhood level of exploration that makes it difficult to the analyst to study and discover patterns among an enormous set of neighborhoods contained in the datasets causing the issue that it requires plenty of time in order to analyze them all. In other words, higher resolutions will generate more hexagons and potentially larger datasets, requiring more storage space a surely much more time to find something interesting. So, every time an analyst inputs a new dataset to study, the most suitable resolution is set according to these two main factors and the most appropriate hexagons are created.

It is important to notice that by creating the hexagons, we mean that we get the hash value of each hexagon stored in a new column of the datasets named “h3_index”, using the «h3» python library. In this way, every data point (latitude, longitude) belongs to a unique hexagon where each hexagon should contain more than one data point, where this is the meaning of studying hexagons instead of data points.

Then, the users are asked to input the attributes, from the data set, that they want to study. Specifically, the users are asked to insert the measure attribute they are interested in, as many dimension / categorical attributes they want and the aggregation function needed. After that, the reference view is being created as a group-by clause based on the set (dimension attribute, measure attribute, aggregation function) the users inserted previously, and it refers the whole datasets. Right after the reference view, the target views are created based on the same set but for every unique hexagon. To be more specific, after each iteration a target view is created based on the same group-by clause as the reference view representing the view (bar chart etc.) of each unique hexagon and instantly the utility score between reference view and target view is calculated and stored into a list. This happens for every unique hexagon. The utility score is calculated by the deviation – based utility feature «Euclidean distance». Then, the target views with the highest utility score are shown to the analysts through bar charts. Although bar charts are used, the main way the views are shown to the users is via the map where it is all divided in to hexagons colored based on their interestingness and when the users select a hexagon then pop ups another diagram, a bar char till now, of the target view. All these steps are shown right below.

Algorithm of the 1st method

Algorithm: HexagonVisualization(dataSet)
Input: dataSet: a collection of data points

```
Output: A visual map of hexagons with varying colors based on interestingness
1:      resolution ← ComputeResolution(dataSet)
2:      hexagon_hashes ← CreateHexagons(dataSet, resolution)
3:      hexagons ← ListOfUniqueHexagons(datasets, hexagon_hashes)
4:      (A, M, F) ← GetUserInputAttributes()
5:      RV ← GroupBy(dataSet, A, M, F)
6:      Views ← empty list

7:      FOR each hexagon IN hexagons DO
8:          TV ← GroupBy(dataSet, A, M, F, hexagon)
9:          US ← DeviationBasedUtility(TV, RV)
10:         Append (US, TV) to Views
11:     END FOR

12:     Sort Views by US in descending order
13:     VisualizeMap(Views, RV)
```

4.2 Method 2: Local Reference View and Local Target Views

In this method, we do not cover large areas, since it is created to help the user to find interesting insights in a certain area around them. The users insert the point (latitude, longitude) of interest and a radius around this point in kilometers. Then the datasets is minimized in a way that it contains only the data inside the bounding box that was created. So, the resolution is set always at number 8, since the datasets contains points in a small area such as a city. Then, the hexagons are created as in “Method 1”. We should say that the bounding box we mentioned earlier is more like a circle where its center is the point of interest the users inserted earlier and the radius is the radius the users inserted respectively, but we will call it for convenience “bounding box”.

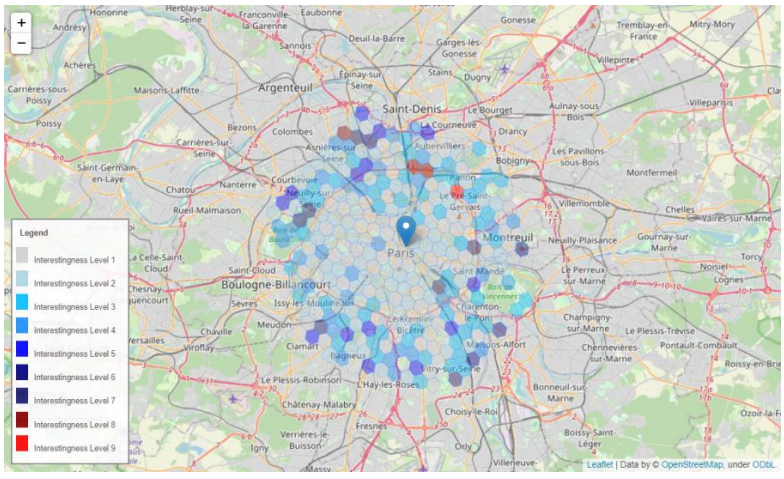


Figure 8. Map showing the interestingness of every hexagon based on the user’s criteria for a specific area of the data set (here the specific area is around the area around Paris).

The next step refers to the creation of the reference view and target views. Both reference and target views focus on the current bounding box viewed by the analysts. So, the reference view is created just like in the first use case, where the system asks the users to insert the dimension and measure attributes they are interested in and the corresponding aggregation function. Then, via a group-by clause based on the set of attributes and aggregation function the

reference view is created for all the existing data in the bounding box. After that, at each iteration we compute both the target view of the corresponding unique hexagon inside the bounding box via the same group-by clause as the reference view and the utility score measured via the Euclidean distance between target and reference view.

Then, the results, are shown using a map (Figure 8) divided in hexagons where the hexagons are colored in a way the users should be able to notice the most interesting hexagons among them since there are labels to explain the interestingness each color represents. At each hexagon, by clicking on it, there is a popup option where a bar chart of target view is shown.

So, the second method, works just like the first method where the major difference is that in the second use case the analysts are interested in a certain area and not the whole datasets. We could say that it is like filtering the initial data to create a smaller dataset that contains only the geographical area the users are interested in, instead of an enormous data collection. This approach, emphasizes identifying interesting patterns and trends within the local context. It is ideal for in-depth local data exploration. For example, suppose we have a dataset containing information about crime rates in a city. In that case, the local reference view would be the current bounding box viewed by the users, and the target views would be smaller regions within the bounding box, such as neighborhoods. By evaluating interestingness scores for each target view, we can identify the most interesting regions and gain insights into local crime patterns.

Algorithm of the 2nd method

```
Algorithm: HexagonVisualization(dataSet)
Input: dataSet: a collection of data points
Output: A visual map of hexagons with varying colors based on interestingness

location ← GetUserInputsCoordinates(latitude, longitude)
hexagon_hashes ← CreateHexagons(dataSet, location, resolution=8)
hexagons ← ListOfUniqueHexagons(datasets, hexagon_hashes)
(A, M, F) ← GetUserInputAttributes()
RV ← GroupBy(dataSet, A, M, F)
Views ← empty list

FOR each hexagon IN hexagons DO
        TV ← GroupBy(dataSet, A, M, F, hexagon)
        US ← DeviationBasedUtility(TV, RV)
        Append (US, TV) to Views
END FOR

Sort Views by US in descending order
VisualizeMap(Views, RV)
```

4.3 Method 3: Local Reference View and Local Target Views with Spatial Distance

In the third use case we implement all the steps made in “Method 2” as well as we introduced one additional factor, the “spatial distance”. This means that in this case, the users input the point where they are interested in studying around it, the radius they want and the set of measure

attribute, dimension attributes and aggregation function they want to study. Then, the reference view and the target views are about to be created via the group-by clause we already said. The reference view is created based on the current bounding box while the target views are representing the group-by clause for each hexagon existing in the bounding box.

In this method, the interestingness metric, defined using the Euclidean function, it also calculates the spatial distance between the center of each hexagon and the location the users inserted. The users are asked by the system to refer how much they are interested in the distance between the point they inserted and the center of each hexagon from 0 to 1, where an option near to 0 means that a user is not concerned that much about the distance. From the other hand, if the users input a real number close to 1 it means that they are mostly interested in the closest hexagons and not concerned about the similarity or dissimilarity of the target view in contrast to the reference view. So, we create a linear equation to compute the utility score having two factors. The first one represents the similarity scores between target and reference views and the second one the distance between each target view and the reference view. Both these factors have their weights that their sum equals to 1 and they are selected by the users in a way that they set the biggest number as a weight to the factor they are interested in the most. It is important to notice that both factors get normalized between 0 and 1 in order to compute the utility score. So, the equation that measures the interestingness in this case seems like the one below:

$$y = (1 - a) * x_1 + a * (1 - x_2)$$

where,

- y: the utility score
- x_1: the similarity score between target view and reference view
- x_2: the distance between the point the users inserted and the center of each hexagon
- α: the weight of how interesting the users think the distance is to them (0,1)

Then, the system shows the interesting hexagons using a map (Figure 9) just like in the previous two methods.

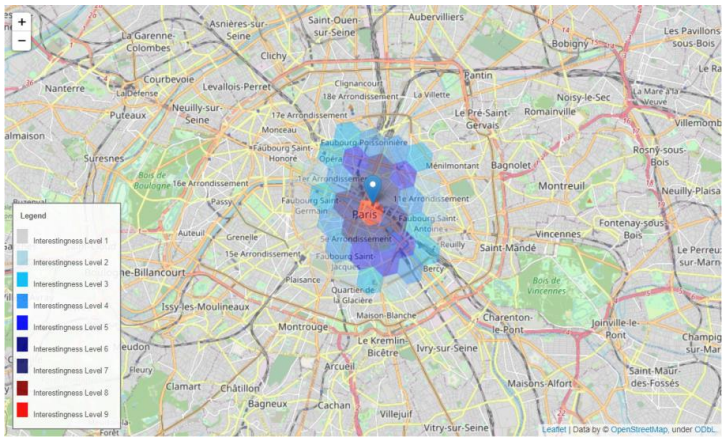


Figure 9. Map showing the interestingness of every hexagon based on the user's criteria for a specific area of the data set (here the specific area is around the area around Paris) where the interestingness is based more in the distance from the point of interest (Paris) and the hexagons around it than the diversity from the reference view.

This use case has its own benefits because the users make more personalized decisions. For example, if the analysts view Athens on the map, the distance from Athens is considered in the interestingness metric, and hexagons closer to Athens will be more interesting. This method allows for comparing local patterns to global ones while incorporating spatial proximity, providing insights into potential similarities or differences. For instance, consider a dataset containing information about population density worldwide. The local reference view would be the current bounding box viewed by the users, and the target views would be individual regions, such as

cities or countries worldwide. By evaluating interestingness scores for each target view, we can identify the most interesting regions in terms of population density and compare them to the local context. It can also be used in many different ways like for finding restaurants to visit that are both close to the users and appear to have some similarities or dissimilarities with the reference view. This means they can find restaurants close to them that have similarities or dissimilarities with the average view of the restaurants existing in the bounding box they created.

In summary, different exploration methods using reference and target views provide insights into data patterns, trends, and relationships. By considering global, local, and spatial distance information, we can gain insights into the most interesting regions within the data. These methods are applicable to various data exploration contexts, including temperature patterns, crime rates, and population density etc. The proposed Spatially-Adjusted Interestingness Measure (SAIM) provides a comprehensive assessment of interesting areas on the map tailored to specific data exploration objectives, balancing the emphasis on bar chart distribution similarity and spatial proximity.

Algorithm of the 3rd method

```
Algorithm: HexagonVisualization(dataSet)
Input: dataSet: a collection of data points
Output: A visual map of hexagons with varying colors based on interestingness

1:      location ← GetUserInputsCoordinates(latitude, longitude)
2:      alpha ← GetUserInputDistanceInteretingness() # from 0 to 1
3:      hexagon_hashes ← CreateHexagons(dataSet, location, resolution=8)
4:      hexagons ← ListOfUniqueHexagons(datasets, hexagon_hashes)
5:      (A, M, F) ← GetUserInputAttributes()
6:      RV ← GroupBy(dataSet, A, M, F)
7:      Views ← empty list

8:      FOR each hexagon IN hexagons DO
9:          TV ← GroupBy(dataSet, A, M, F, hexagon)
10:         DVS ← DeviationBasedUtility(TV, RV)
11:         DistS ← DistanceFromLocation(location, center of hexagon)
12:         US ← UtilityScore(DVS, DistS)
13:         Append (US, TV) to Views
14:     END FOR

15:     Sort Views by US in descending order
16:     VisualizeMap(Views, RV)
```

4.4 Implementation

In order to implement these three methods, we used python as the main programming language. Python is a programming language that has gained popularity for its simplicity and flexibility in data analysis, visualization, and machine learning. In this master thesis it was used python as the main programming language because of some benefits like (a) Rich Data Visualization Libraries such as Matplotlib, Seaborn, Plotly, Folium etc., (b) Easy Data Manipulation, (c) Support for Machine Learning and (d) Cross-Platform Compatibility. Specifically, these libraries offer a variety of charts, graphs, and plots, including line charts, bar charts, scatter plots, heatmaps, and of

course maps, the main way we show interesting insights to the users. They, also, provide flexibility and customization options that allow developers to create personalized visualizations that meet the unique needs of their projects. Data manipulation like data cleaning, transforming, and restructuring raw data into a more meaningful format that can be easily analyzed and visualized libraries such as Pandas and NumPy, becomes much easier, because, these libraries provide a set of functions and tools that allow developers to filter, sort, and aggregate data with ease.

Python stands out in the realm of programming languages due to its robust support for machine learning through libraries like scikit-learn, TensorFlow, and PyTorch. Its cross-platform compatibility allows developers to write code once and run it seamlessly across various devices and operating systems, saving time and effort. Python offers a plethora of data visualization options, such as Datawrapper, Tableau, Plotly Dash, PyViz, Mode, IBM Watson Studio, RapidMiner, KNIME, and QlikView, making it an ideal choice for building recommendation systems. These visualization tools, coupled with Python's easy data manipulation and machine learning capabilities, empower data scientists to create accurate predictive models. Additionally, Python's library h3 facilitates the creation of hexagons, a vital component in various studies. Python's popularity, user-friendly interface, and versatile libraries make it a top choice for developers seeking to craft recommendation systems that deliver valuable insights and seamless user experiences.

So, thanks to python, we present some examples of how an interaction between our approach and the users is (Figure 10). The approach, in this example learns about the attributes the users are interesting in studying, the location they want to study and how much interested they are in the distance between their location and the other places. In addition, our approach shows some more details during the computation of the utility score, if requested, such as the reference view and the target views.

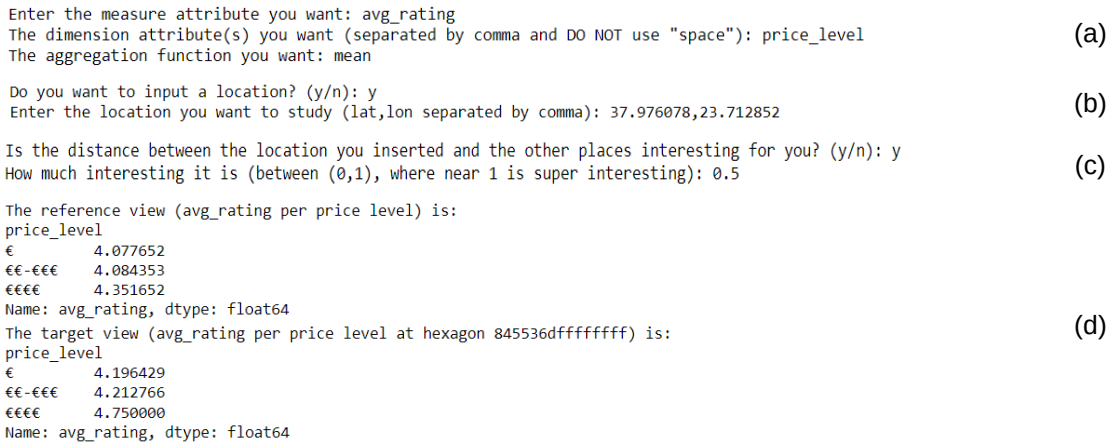


Figure 10. (a) The approach interacts with the users asking them about the attributes and the aggregation function they want. (b) The approach asks the users if they are interested in a certain location, if so, they are asked to insert the location (lat, lon). (c) The approach, asks the users if they are interested in the distance between the location they inserted and the other places around it, so they can input how interesting the distance is to them. (d) In our approach the users can also see the values of the reference view and each target view in a tabular form if they ask for it.

5 Demonstration of the methods

In this section, we had to see how our system works using real world data. We used a data set from Kaggle and made several processes. Data preparation was the primary process, where we firstly examined the datasets about its shape, its variables / columns, some statistics, outlier detection etc. and secondly, we made a suitable data cleaning like removing outliers and empty columns, transforming the data etc. Then, we used our three methods on that datasets and observed the results. It is important to note that we implemented our methods using several

distance metrics, one at a time, as the utility function that measures the interestingness of a visualization and we present the ones that gave us the most interesting results. Lastly, we used a combination of those metrics to create our final utility function that captures the interestingness of the visualizations from different aspects. So, summing up, in the next sections, we analyze how our methods work on real – world data sets.

5.1 Data Preprocessing - Cleaning

We applied our method, for all three methods we discussed earlier, to the datasets "tripadvisor_european_restaurants.csv" from Kaggle. It contains 1083397 rows and 42 columns and among these 42 columns there are two columns that have the geographical data needed, the variables "latitude" and "longitude". It is important to notice that we use a copy of the original datasets so we can do any transformation we want in order to measure the interestingness and then we suggest visualizations with the attributes and values from the original datasets.

Before setting our methods into action, we prepared the data. Initially, we implemented processes like cleaning and transforming the data into a format that is suitable for visualization. We did procedures such as removing outliers, transforming variables, and imputing missing values too. It's important to notice that many recommendation systems do not involve processes like data cleaning because they want to show many different insights including data anomalies like distribution of the outliers etc. In our case, just because we want to focus our study at showing and suggesting the most interesting hexagons, we decided to clean the data, so, we (a) removed the duplicates, (b) the empty columns and (c) the empty rows. We used data that has geospatial variables, which we aim at to study, like latitude and longitude. So, (d) we removed also, all the data that has empty rows in these two columns (latitude, longitude). After these operations, the dimensions of the datasets got reduced to 1067607 rows and still 42 columns.

Furthermore, the last steps of the data cleaning process contained (e) the check if there were any outliers, (f) their removal and (g) the normalization of the remaining data. Outliers can be detected via many ways. The most commonly used methods are the quantile method, the z-score method as well as the box-plots. We used the quantile method to detect the outliers and then we completely removed them. Additionally, we filled all the missing values of the numeric variables, that we decided to keep, with the average value of the corresponding variable. After these operations the datasets had 651902 rows and yet 42 variables/columns. Lastly, we decided to transform all the values, except the values belonging to the variables "latitude" and "longitude", by normalizing them to be between 0 and 1. For this purpose we used the following formula:

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

where:

z_i: $the\ i^{th}\ normalized\ value\ in\ the\ dataset$
x_i: $the\ i^{th}\ value\ in\ the\ dataset$
$\min(x)$: $the\ minimum\ value\ in\ the\ coulmn/variable\ x\ of\ the\ dataset$
$\max(x)$: $the\ maximum\ value\ in\ the\ column/\ variable\ x\ of\ the\ dataset$

5.2 Use Case 1 – Method 1st

In this chapter we implemented all the things we discussed earlier in chapter 4.1. Our system interacts with the users making them questions like "Do you want to input a location?", or even asks them to do submit information they are interested in like "Enter the location you want to study", "Enter the measure attribute you want", etc. Specifically, in the first use case (1st method), the users are interested in all areas contained in the datasets. In our case, it creates hexagons along the Europe's map. So, the datasets seems to cover a big area, actually much bigger than

the corresponding threshold. However, its density is slightly under the corresponding threshold. So, the most suitable resolution for creating hexagons in the whole datasets, that should probably help the analyst discover interesting insights and patterns, seems to be 4, which corresponds to an area of almost 1770 km^2 per hexagon, as presented in Figure 4, which it is similar to the area of a big town, or sometimes 2-3 smaller towns together. This would result in a moderate number of hexagons, providing a general overview of the data distribution across the datasets and as a result it could be a reasonable starting point. So, there were created a sum of 3045 unique hexagons.

So, in this chapter we present the results of the implementation of our first method. This concludes bar charts and maps with colored hexagons that are showing some of the most interesting target views (hexagons) compared to the reference view that were created based on different sets (A, M, F). It's important to notice that all these visualizations were generated using Euclidean distance as the only utility feature while our last implementation with the combined utility features will be presented in the chapter 6.

5.2.1 Visual insights for the most interesting view based on our attributes

In the first example, the user asked to see what happens with the «mean» value (aggregation function) of the measure attribute «avg_rating» to the corresponding dimension attributes «vegetarian_friendly» «price_level». So, the reference view and the target views are created as we discussed previously. Also, after computing the utility score, we asked for the top-5 most interesting views, which means the views with the highest utility score. Here we present some of the most useful to find insights views.

Based on the image, bar chart, bellow (Figure 11), the users can understand that in the hexagon with hash value «841e567fffffff» there are both vegetarian and non-vegetarian friendly restaurants. The vegetarian friendly restaurants are all mid-price restaurants that it probably makes it affordable for many people to visit. They are also rated above the average rating of the same type of restaurants through all Europe. The users can also see that this hexagon mostly contains non-vegetarian friendly restaurants of all different price level types restaurants. Especially, the non-vegetarian cheap restaurants seem to have a really high average rating, almost 5 out of 5, and much higher compared to the other restaurants of this type, that have an average rating around 4 out of 5.

From the other hand, we can see that there are some really expensive non-vegetarian friendly restaurants that are rated really low, like 1 out of 5. Lastly, about this view, we see that the average rating of the mid-price non-vegetarian friendly restaurants is slightly lower than the corresponding value resulting from the reference view.

Overall, the bar chart we examined (Figure 11) was rated as the most interesting view, from the system, based on the «Euclidean Distance» and we already can make some speculations of what kind of insights we can generate. For example, the analyst should be able to think that this hexagon mostly contains restaurants that are interested in meat-based menus or even that this hexagon may be visited mainly from not wealthy people. It is an interesting hexagon for sure, since the analyst can already speculate many things as well as from the fact that it differs a lot from the general view of Europe.

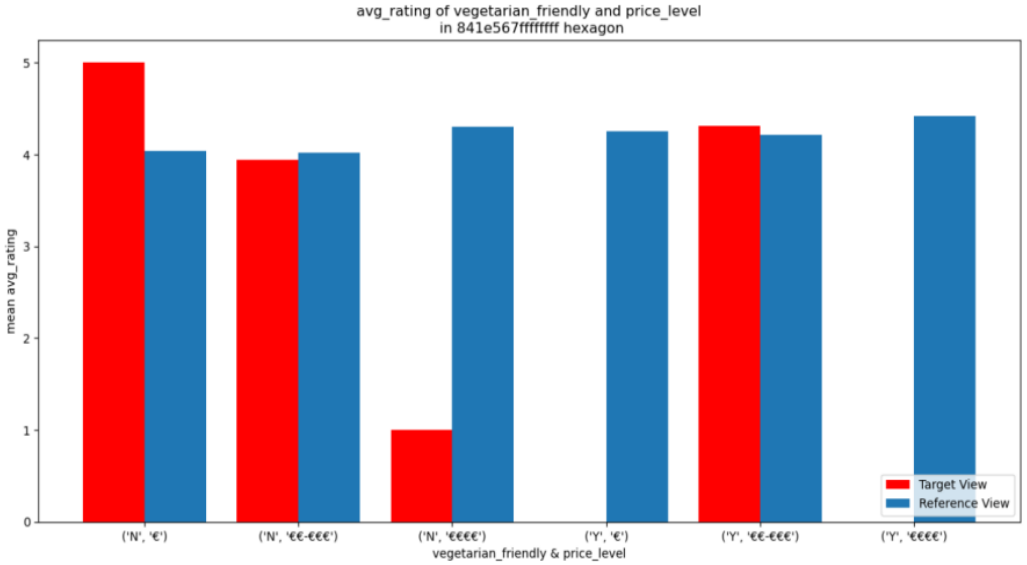


Figure 11. Bar chart showing the comparison between target view (hexagon "841e567fffffffff") and reference view average rating values per vegetarian friendly options and their respective price level.

We represent these results in separated bar charts (Figure 12), too, as it is shown in the following image.

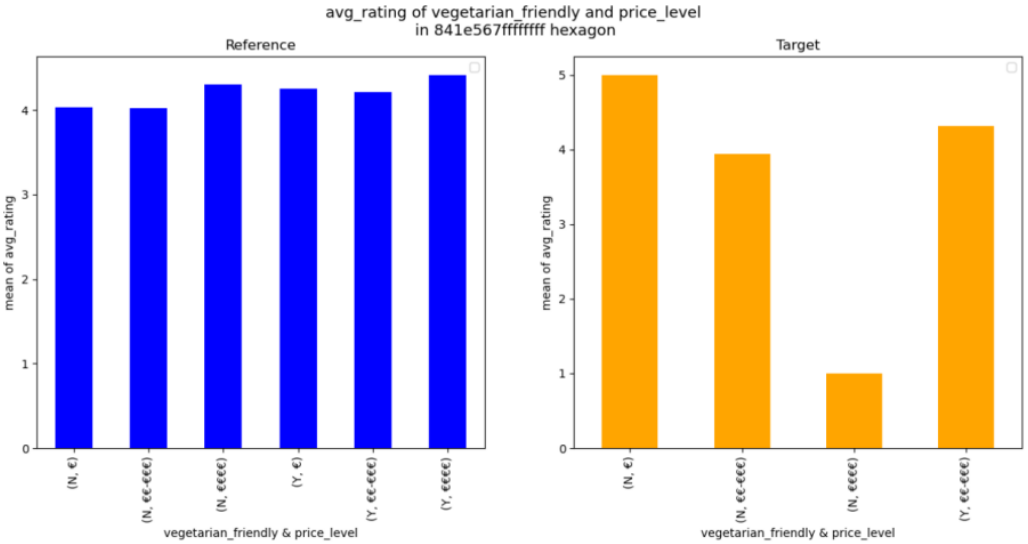


Figure 12. Separated bar-charts showing the comparison between target view (hexagon "841e567fffffffff") and reference view average rating values per vegetarian friendly options and their respective price level.

Finally, the system presents a comprehensive overview of interestingness to the analyst through a map partitioned into hexagonal segments (Figure 13), each uniquely colored to denote the level of interestingness attributed to it. Consequently, when a user selects a specific hexagon, a corresponding bar chart is displayed as a pop-up (Figure 14), providing more detailed insights.

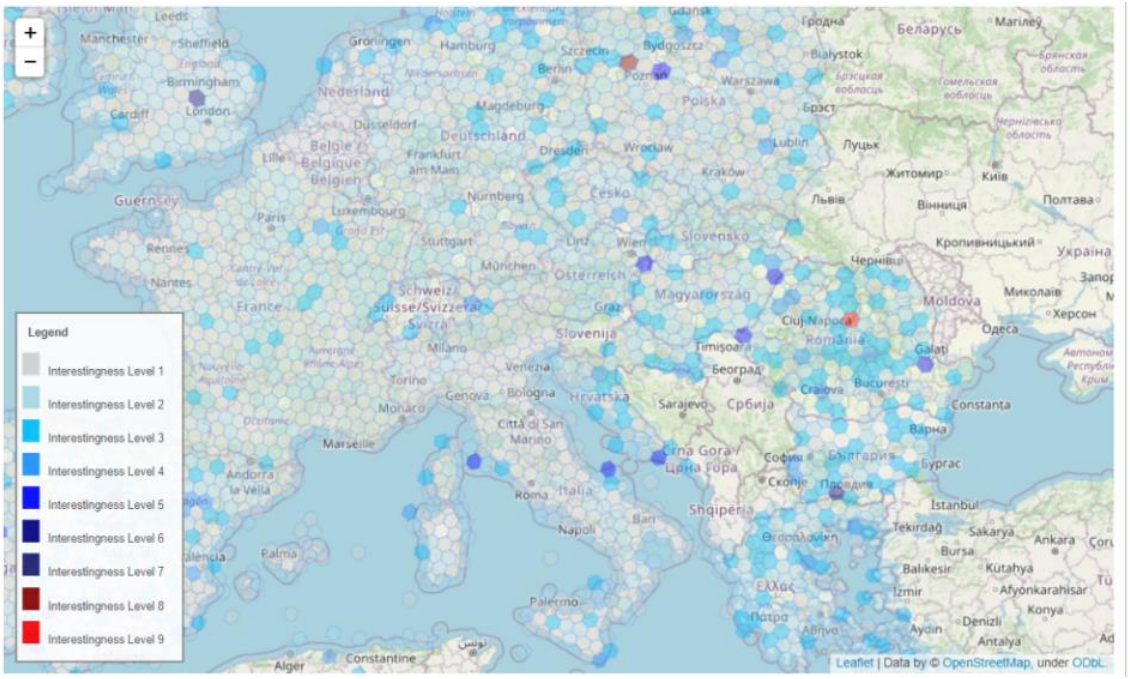


Figure 13. Map showing the interestingness of every hexagon based on average rating values per vegetarian friendly options and their respective price level.

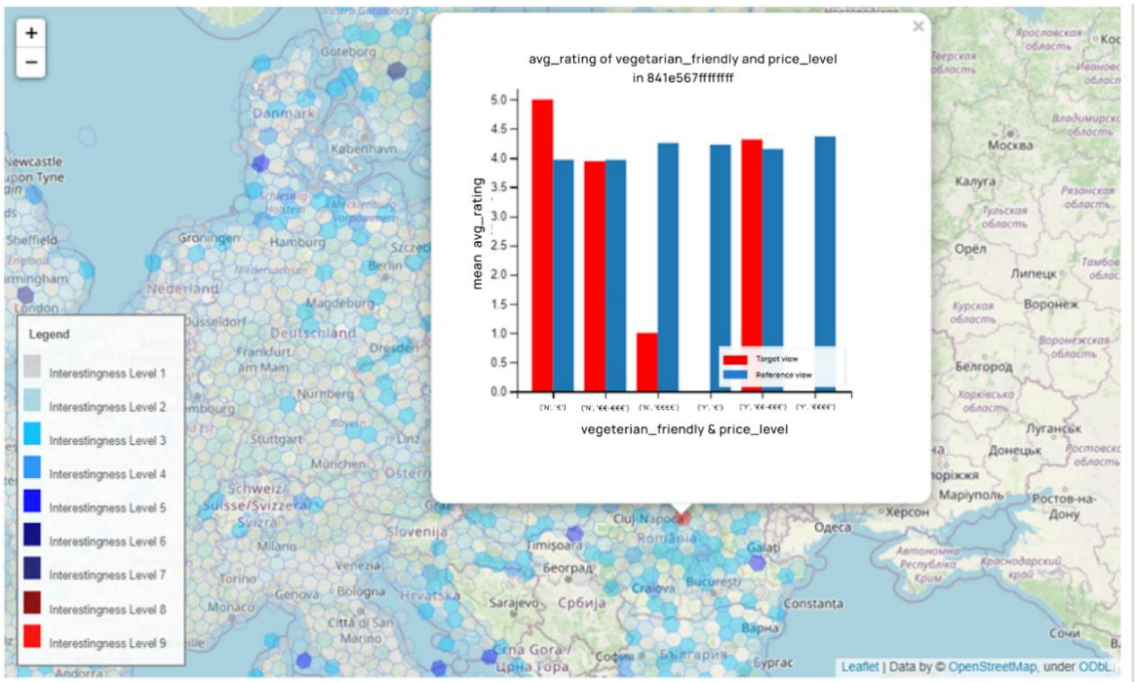


Figure 14. Map showing the interestingness of every hexagon and the popup option (at hexagon "841e567fffffffff") based on average rating values per vegetarian friendly options and their respective price level.

5.2.2 Visual insights for the most interesting view

Next, we represent the second most interesting view that the system provided based on the same attributes and the same utility feature «Euclidean Distance». As we can see it is a much more pluralistic view, since it contains values for all possible dimension attributes combinations.

From these bar charts (Figure 15) the analyst should be able to understand many insights. For example, in hexagon “84390a3fffffffff” the mean average rating of the non – vegetarian friendly restaurants is lower than the mean average rating of the non – vegetarian friendly restaurants in the whole datasets (reference view) having a significant difference at the non – vegetarian friendly super expensive restaurants. It seems that in this hexagon the restaurants that offer vegetarian friendly options are preferred the most since their rating is much higher in any case. From the other hand, the price level is not considered to have such a significant impact at the rating decision maybe except from the case of non – vegetarian most expensive restaurants.

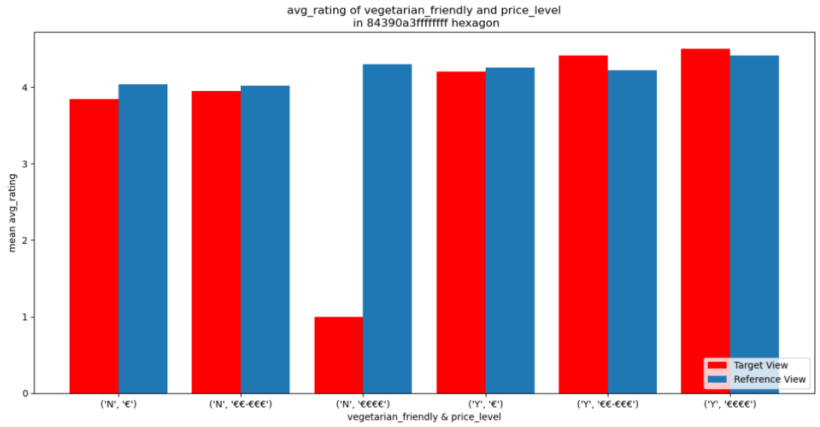


Figure 15. Bar chart showing the comparison between target view (hexagon “84390a3fffffffff”) and reference view average rating values per vegetarian friendly options and their respective price level.

At these separately bar charts (Figure 16) the users should not only be able to obtain the same information as previously but they can also see what happens into this hexagon despite off the general trend. However, these are not the only ways to represent the data, while as said earlier for the most interesting view the system provided, we show the results via hexagons.

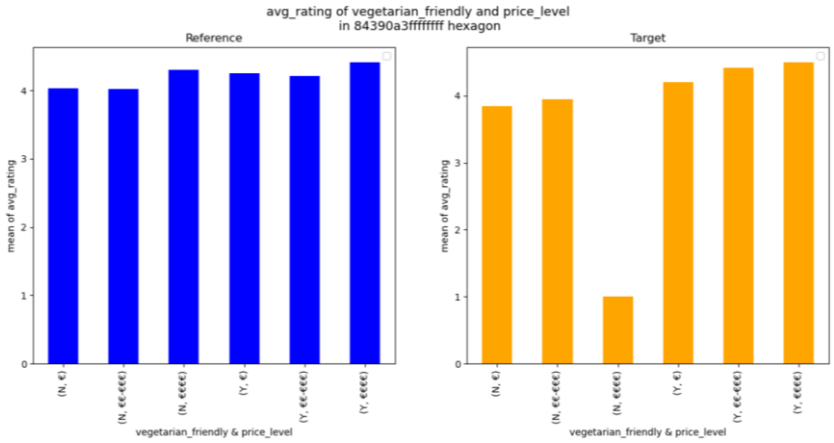


Figure 16. Separated bar charts showing the comparison between target view (hexagon “84390a3fffffffff”) and reference view average rating values per vegetarian friendly options and their respective price level.

Then, as already said, the system shows the results via the map with the popup option like the picture below (Figure 17).

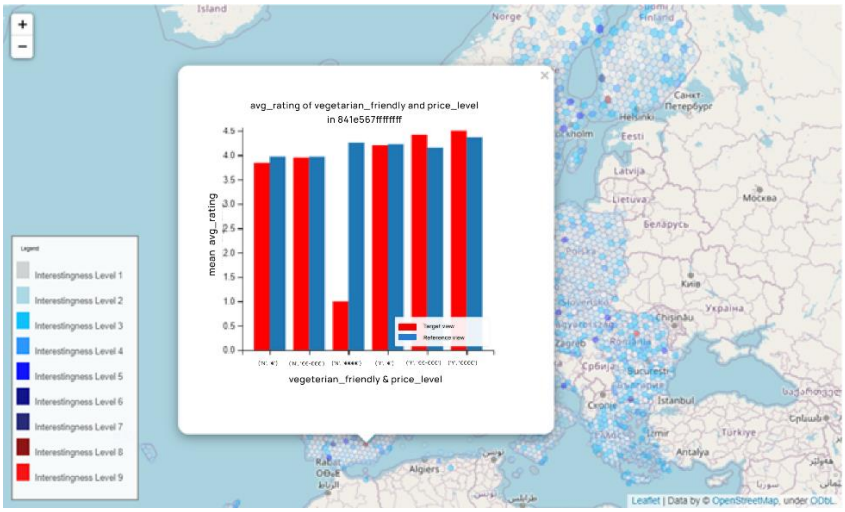


Figure 17. Map showing the interestingness of every hexagon and the popup option (at hexagon "84390a3fffffff") based on average rating values per vegetarian friendly options and their respective price level.

5.3 Use Case 2 – Method 2nd

In the second method, the users are dedicated to study the area around a geographical point of interest. Firstly, the system asks them if they are interested in any specific location. If so, the system requests the entrance of a geographical point and the radius (in kilometers) where the users want to study around the point of interest they inserted. After these inputs, the system limits the data needed and starts a sequence of steps respectively to those made in «Use Case 1 – Method 1st», asking the users about the attributes, measure and dimensions, and the aggregation function they are interested in, to have the group-by clauses in order to create the reference view as well as the target views. Continuing, the system computes the utility score, as mentioned previously, to measure the interestingness and recommend the top-k visualizations.

In our example, the users' point of interest refers to the city of Athens, Greece, (latitude 37.976078 and longitude 23.712852) and the inserted radius around this point was 3 kilometers. In this way, thirty–six (36) hexagons were created. Then, the users select the measure attribute «avg_rating», the dimension attributes «vegan_options» and «gluten_free» and the aggregation function «mean». At the figures below, we show some of the results.

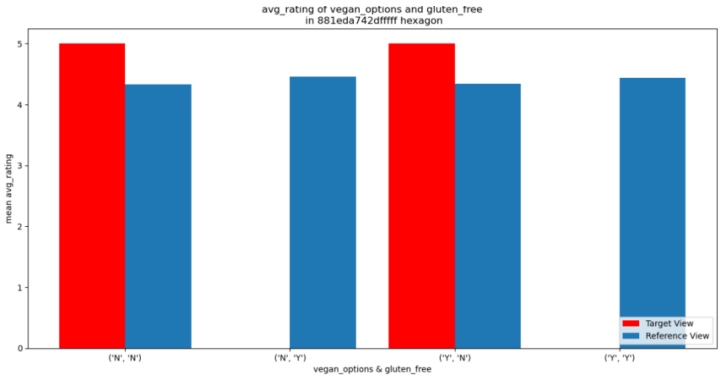


Figure 18. Bar chart showing the comparison between target view (hexagon "881eda742dfffff") and reference view average rating values per vegan and gluten free options.

At the figure right above (Figure 18) we can see that the hexagon «881eda742dfffff» contains restaurants that include both vegan options and non-vegan options in their menu but only non-gluten free options. Those restaurants, that do not include vegan options are rated much higher than those that do and actually above the average rating of all the other combinations. So, the users should be able to understand that in this hexagon the restaurants contained in have to offer plenty of good food choices since they are rated with almost 5 out of 5 and they all contain vegan options. From the other hand, someone only interested in gluten free options should not visit the restaurants in this hexagon.

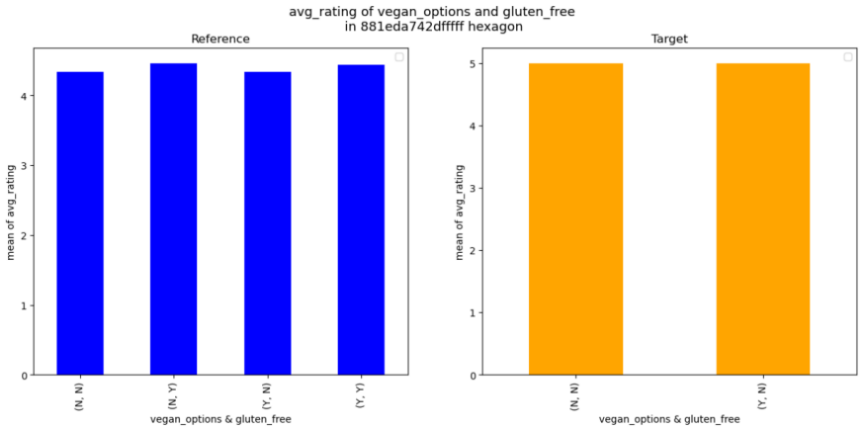


Figure 19. Separated bar charts showing the comparison between target view (hexagon "881eda742dfffff") and reference view average rating values per vegan and gluten free options.

Additionally, we demonstrate alternative methods through which the system presents results to the users. These include displaying the data using separate bar charts (Figure 19) or representing it with hexagons on a map, as previously mentioned (Figure 20). The only distinction lies in the map feature, where the users' inputted point is also depicted, allowing the users to visually gauge the proximity of the desired hexagons to that specific point.

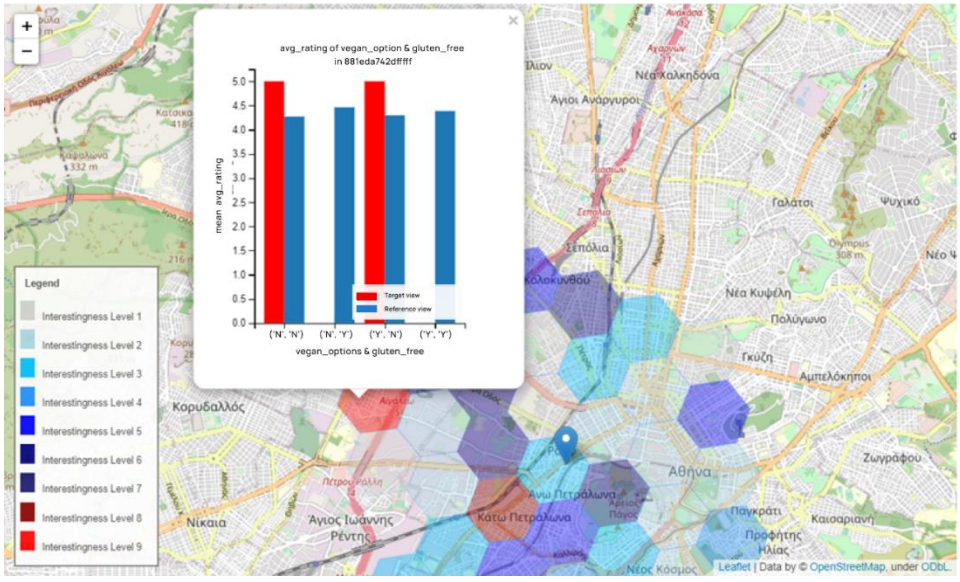


Figure 20. Map showing the interestingness of every hexagon and the popup option (at hexagon "881eda742dfffff") based on average rating values per vegan and gluten free options.

5.4 Use Case 3 – Method 3rd

In use case 3, the users want to study the area around a geographical point of interest as in "Method 2nd" with the difference that in this case the users are interested in the distance between the hexagons and the point of interest too. This means that they want to add the distance as one new feature in our utility function so the most interesting hexagons for them to be defined. Firstly, the system asks the users if they are interested in any specific location. If so, the system requests the entrance of a geographical point and the radius (in kilometers) the users want to study around the point of interest inserted like previously. Then, the next question that is made to the users, by the system, is about the interestingness they have for the distance between the point of interest and the hexagons around it. In this case we suppose they are interested in the distance, so the system asks the users to input a real number from 0 to 1 that represents how much they are interested in distance. So, a new feature and the corresponding weights are added to the utility function. After these inputs, the system limits the data needed and starts a sequence of steps respectively to those made in «Use Case 1 – Method 1st» again, asking the users about the attributes, measure and dimensions, and the aggregation function they are interested in, to have the group-by clauses in order to create the reference view as well as the target views. Later, the system computes the utility score, as mentioned previously, to measure the interestingness and recommend the top-k visualizations.

In our example, the point of interest refers to the city of Paris, France, (latitude 48.856614 and longitude 2.3522219) and the inserted radius around this point was 3 kilometers. In this way, thirty–seven (37) hexagons were created. In addition to these steps, in this case, the weight 0.7 was inserted to show how interested the users are in the factor of distance. The weight of 0.7 means that a user is more interested in finding out patterns in hexagons near to the point of interest than how similar or dissimilar to the reference view the target view is. Then, the users select the measure attribute «food», the dimension attributes «vegan_options» and «price_level» and the aggregation function «mean».

At the figures / bar charts below (Figure 21, Figure 22) we show the results of the hexagon that our system detected as the most interesting. In hexagon «881fb46625fffff» exist restaurants containing both vegan options and non-vegan options in their menu but not expensive restaurants with vegan options. In addition, the food in the restaurants in this hexagon was rated less than the average values of the general (reference) view which is the center of Paris and all the restaurants in a radius of 3km.

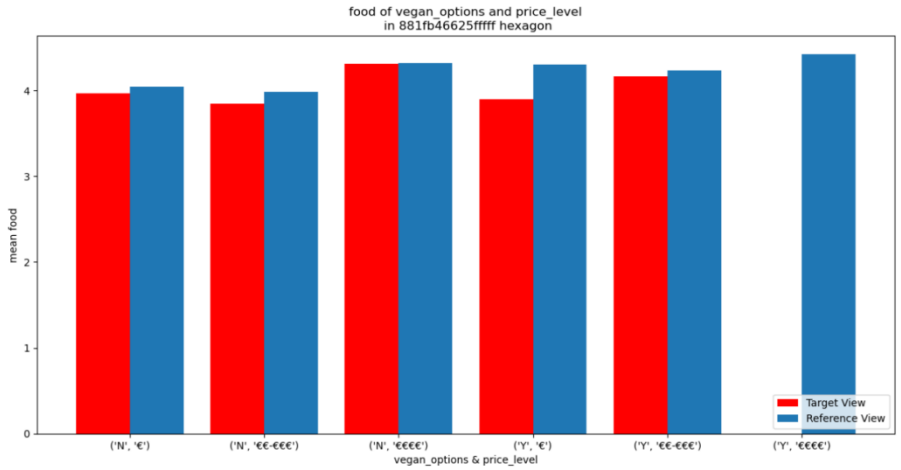


Figure 21. Bar chart showing the comparison between target view (hexagon "881fb46625fffff") and reference view food rating values per vegan options and their respective price level.

Comparing the reference view to the target view or even by comparing the values in the target view individually (Figure 22) seems that the restaurants that do not offer vegan options in their menu, but are the most expensive have their food rated much higher than the other

restaurants do. Their food is also rated really close to the corresponding value of the reference view.

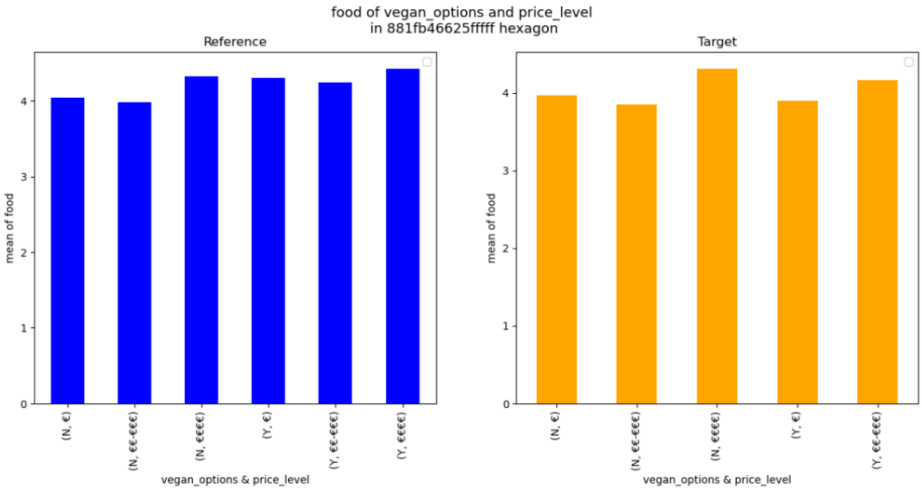


Figure 22. Separated bar charts showing the comparison between target view (hexagon "881fb46625fffff") and reference view food rating values per vegan and gluten free options.

Viewing the map (Figure 23), as expected, most interesting hexagons to the users are those close to the point of interest they inserted, where the most interesting one seems to be the hexagon "881fb46625fffff" that contains the point of interest (Figure 24). Since the dissimilarity factor has a weight of 30% not all the hexagons closest to the point have the same utility score. So, they are not all the same interesting and they are colored a little differently. For example, the hexagon «881fb46753fffff» (Figure 25) even though it is as close to the point as the hexagon «881fb46627fffff» (Figure 26), it is not rated by the system as interesting as the hexagon «881fb46627fffff». The difference is spotted by the color where the hexagon «881fb46753fffff» is colored with a lighter blue than the hexagon «881fb46627fffff» that they represent the interestingness 5 and interestingness level 6 respectively. All these are shown in the images below.

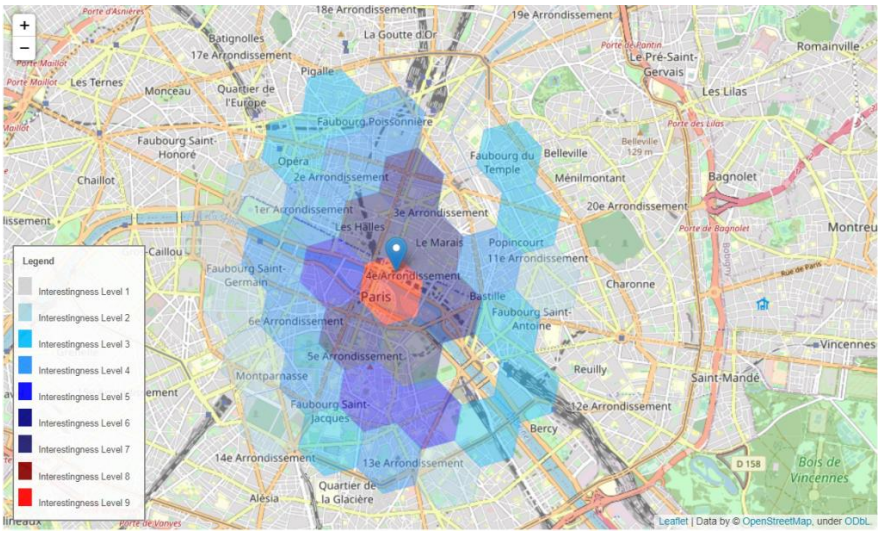


Figure 23. Map showing the interestingness of every hexagon based on food rating values per vegan and gluten free options.

Figure 24. Map showing the interestingness of every hexagon and the popup option (at hexagon "881fb46625fffff") based on food rating per vegan options and respective price level.

Figure 25. Map showing the interestingness of every hexagon and the popup option (at hexagon "881fb46753fffff") based on food rating per vegan options and respective price level

Figure 26. Map showing the interestingness of every hexagon and the popup option (at hexagon "881fb46627fffff") based on food rating per vegan options and respective price level.

5.5 Other metrics & Results

In this section we present the results of the same actions, attributes and aggregation functions for all three methods based on different deviation – based utility features, the RMSE, Manhattan distance, KL divergence and MAD, so we can understand about similarities between them. For this purpose, we observed all three methods using the same attributes and aggregation functions as in the case of Euclidean distance respectively in order to spot some differences.

As shown at the next figures, there are some common characteristics among all these metrics and some dissimilarities, too. As for the similarities, we observe that many hexagons noted as interesting by a utility feature – metric, they are still considered as interesting by other utility features. We can also, see that mostly the non-interesting hexagons provided by the Euclidean distance keep being non-interesting by the other metrics. From the other hand, there is a redistribution of the most interesting hexagons. For example, the hexagon «841e567fffffff», that it is located at Romania, was noted as the most interesting hexagon using the Euclidean distance, but it is not the most interesting one while using the RMSE or the Manhattan distance or the KL divergence as the only utility features where it is still considered one of the most interesting hexagons though. Additionally, while using MAD as the only utility feature the «841e567fffffff» hexagon was not noted as interesting at all. All these results are shown via the next visualizations (from figure 27 to figure 38).

RMSE: Root Mean Square Error

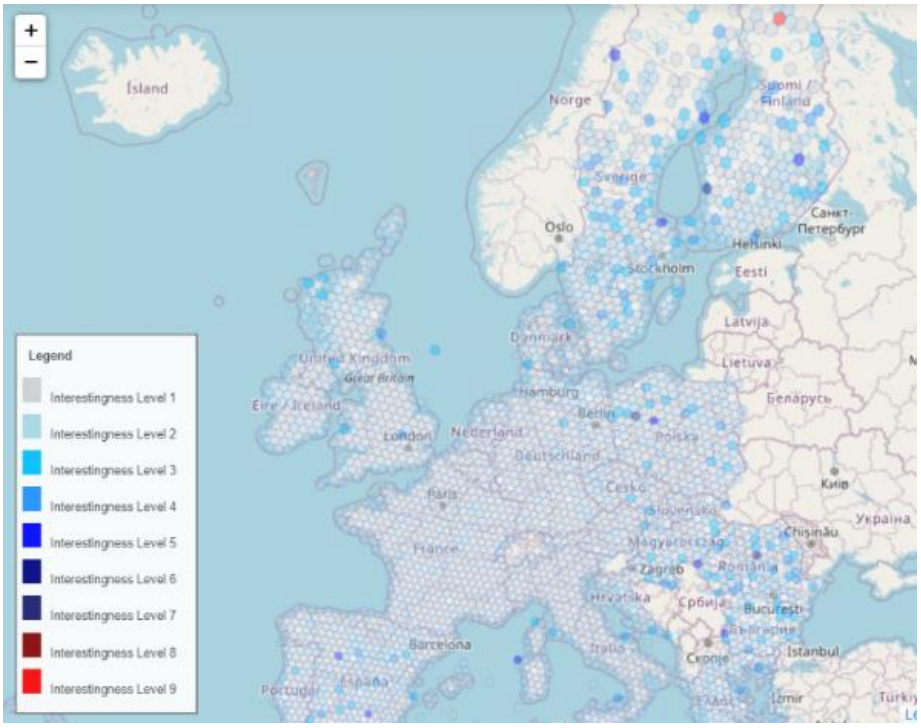


Figure 27. View created by the 1st method, based on the attributes "vegetarian_friendly", "price_level", "avg_rating" and the aggregation function "mean".

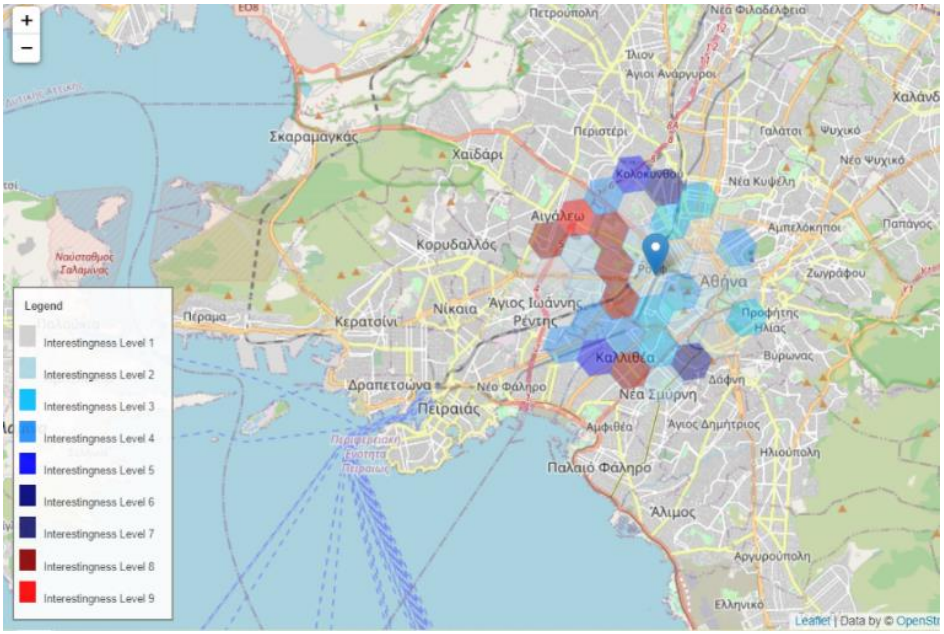


Figure 28. Image created by the 2nd method, based on the attributes "vegan_options", "gluten_free", "avg_rating" and the aggregation function "mean".

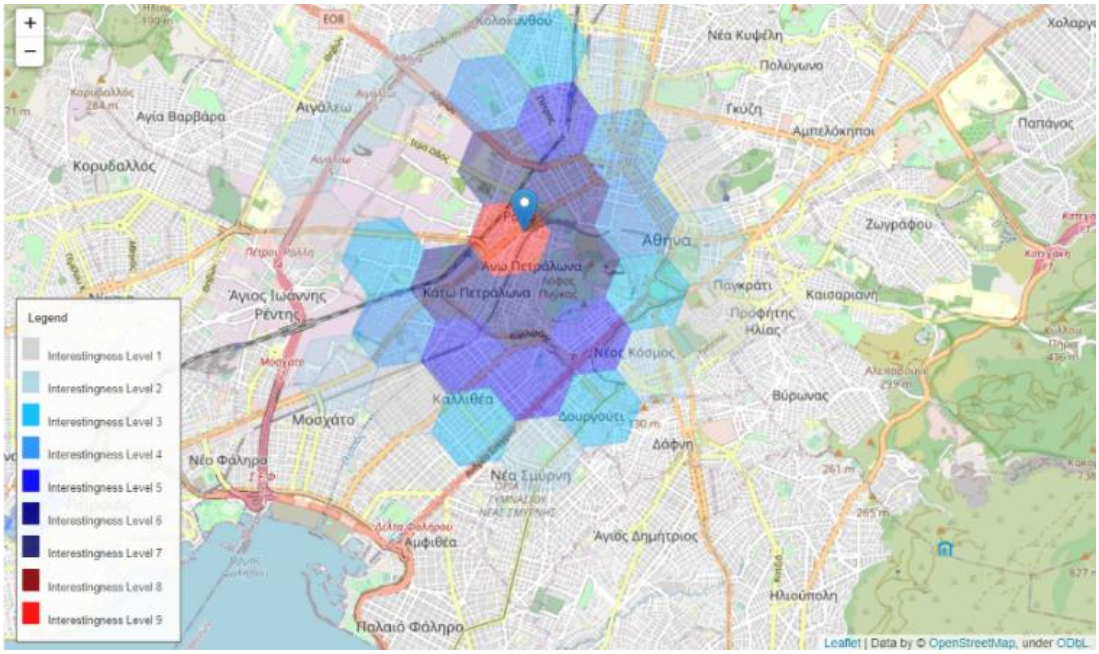


Figure 29. Image created by the 3rd method, based on the attributes "vegan_options", "price_level", "avg_rating" and the aggregation function "mean".

Manhattan distance / L1 distance / Taxicab geometry:

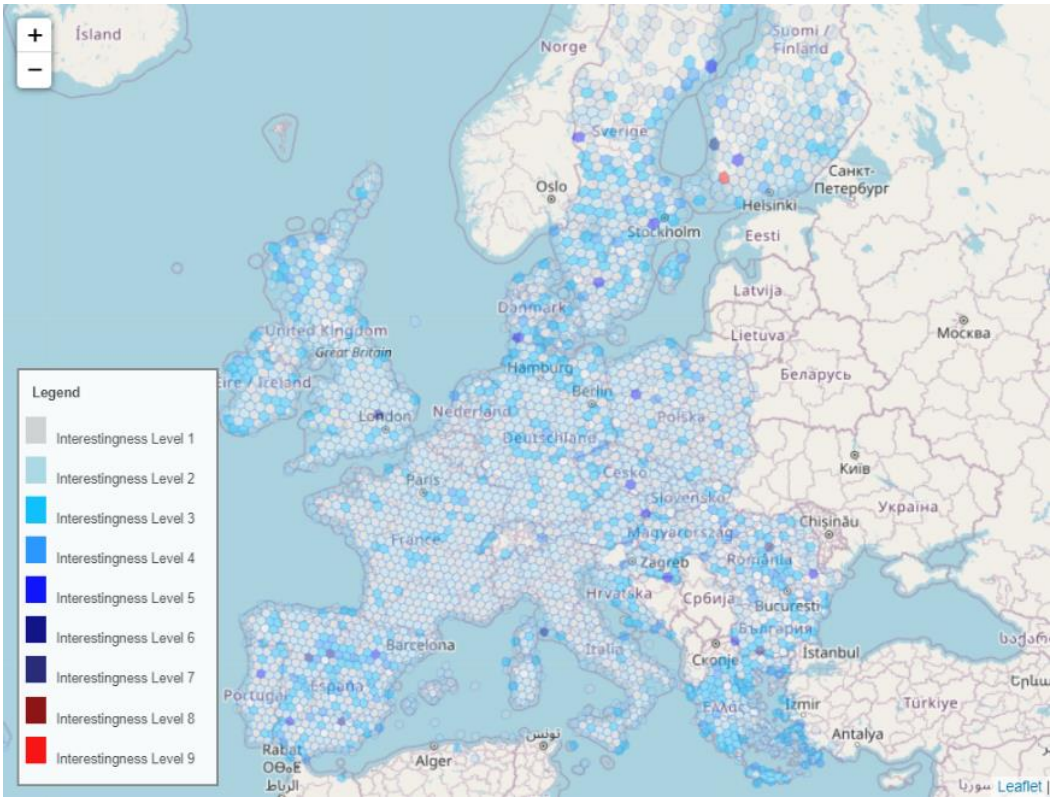


Figure 30. View created by the 1st method, based on the "vegetarian_friendly", "price_level", "avg_rating" attributes and the aggregation function "mean".

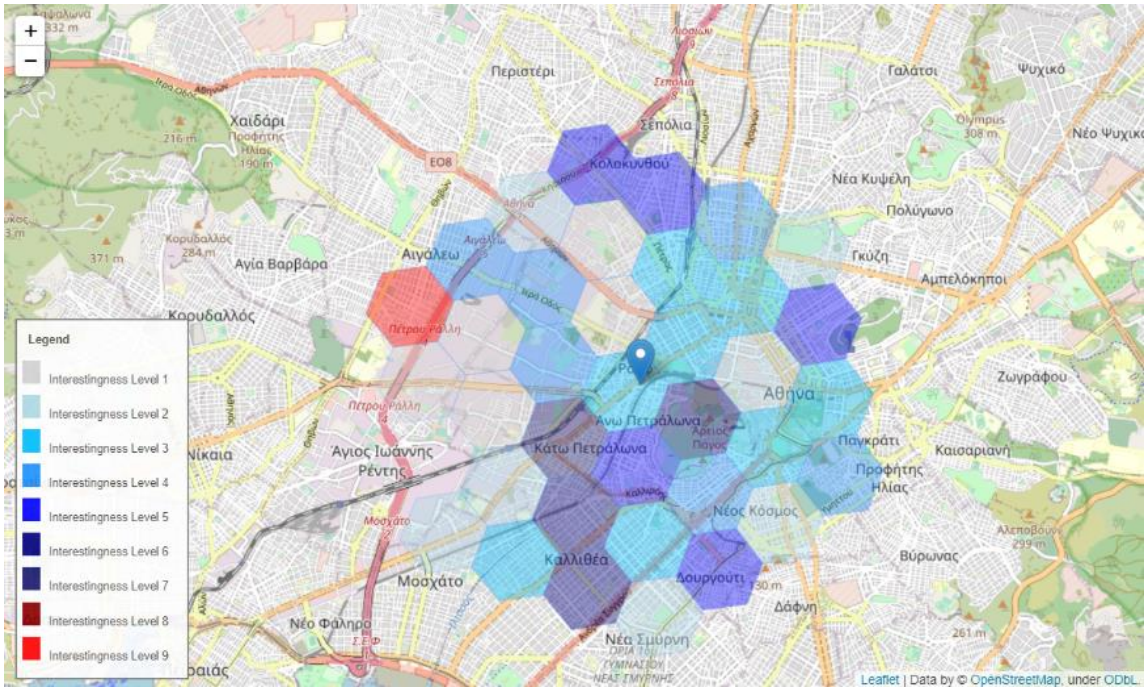


Figure 31. Image created by the 2nd method, based on the attributes "vegan_options", "gluten_free", "avg_rating" and the aggregation function "mean".

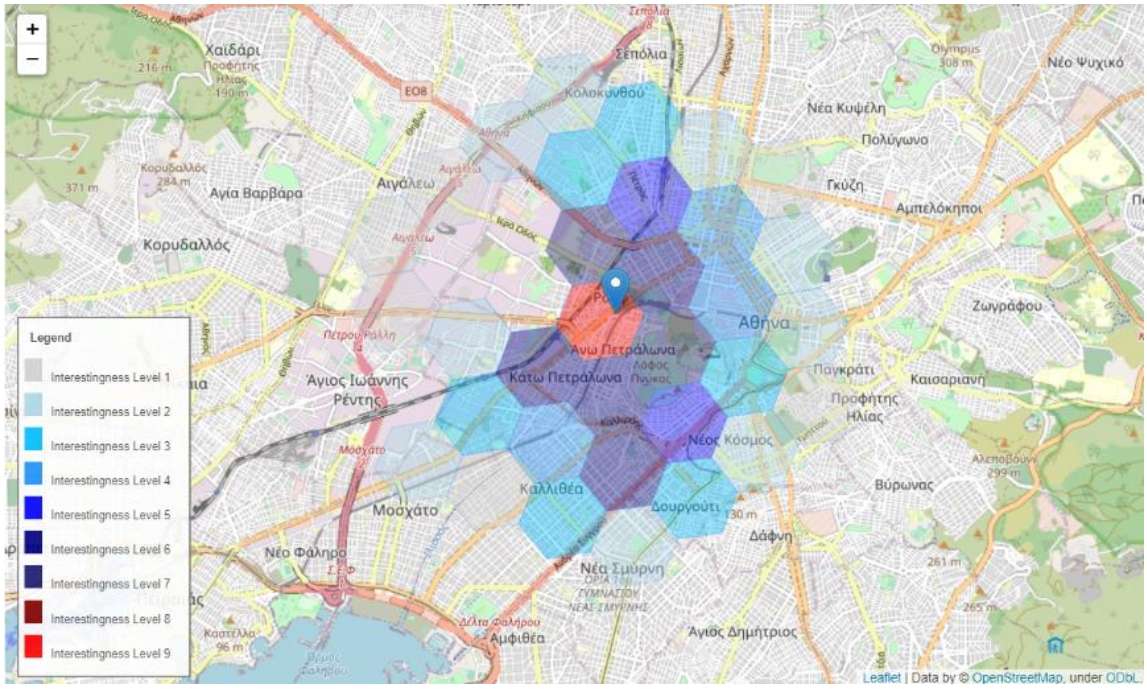


Figure 32. Image created by the 3rd method, based on the attributes "vegan_options", "price_level" "avg_rating" and the aggregation function "mean".

KL divergence:

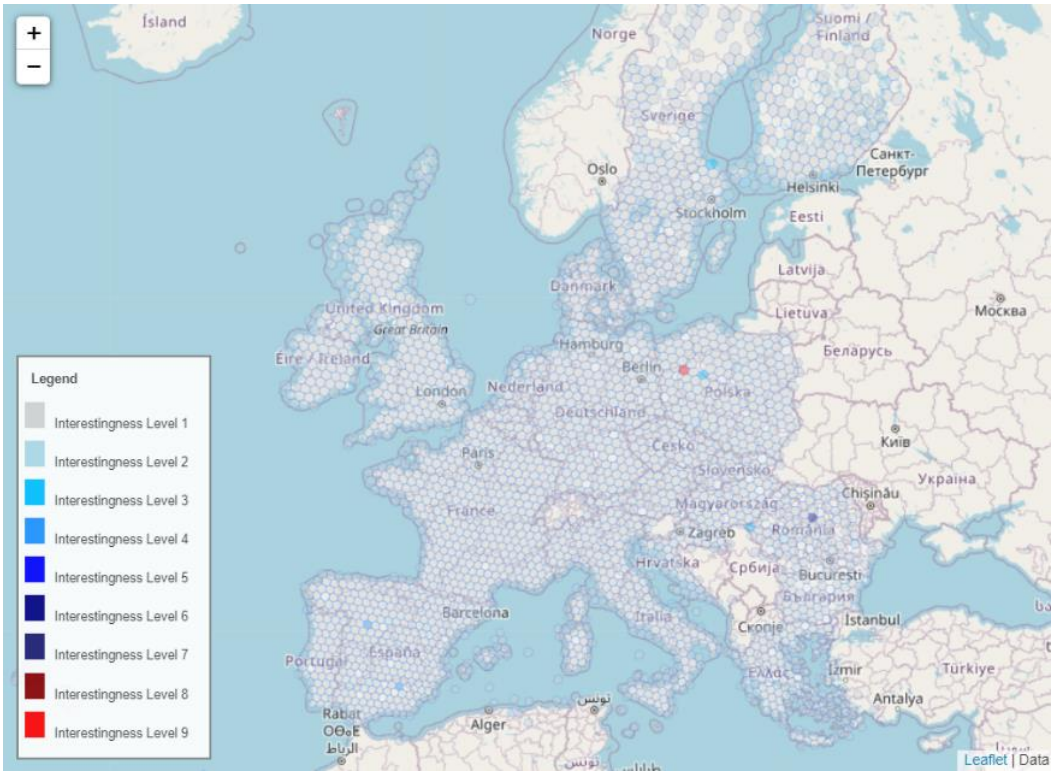


Figure 33. View created by the 1st method, based on the attributes "vegetarian_friendly", "price_level", "avg_rating" and the aggregation function "mean".

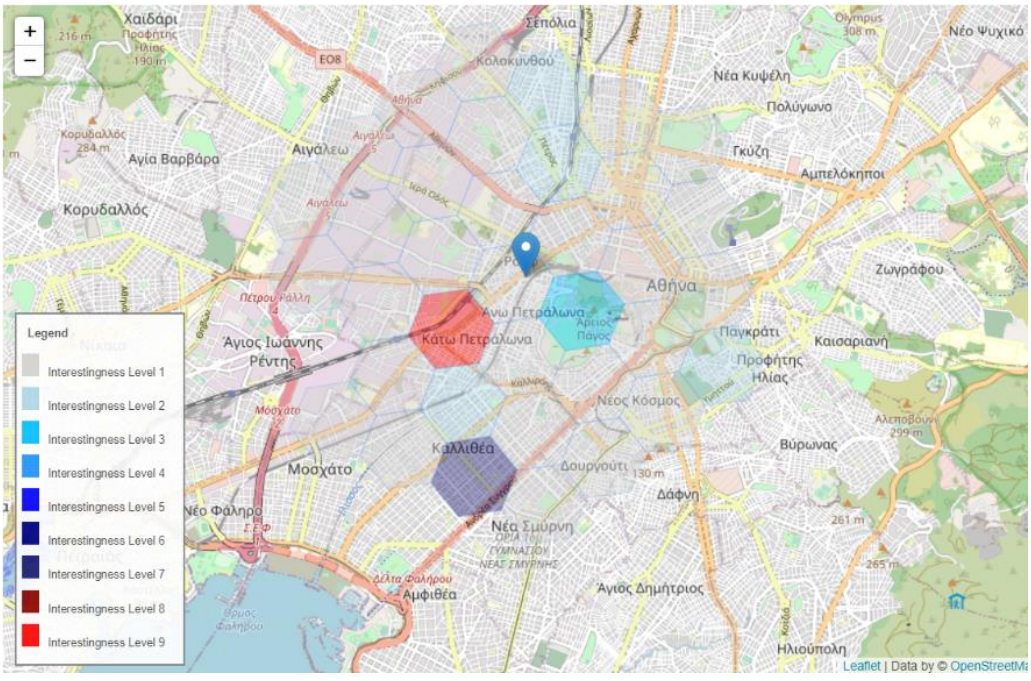


Figure 34. Image created by the 2nd method, based on the attributes "vegan_options", "gluten_free", "avg_rating" and the aggregation function "mean".

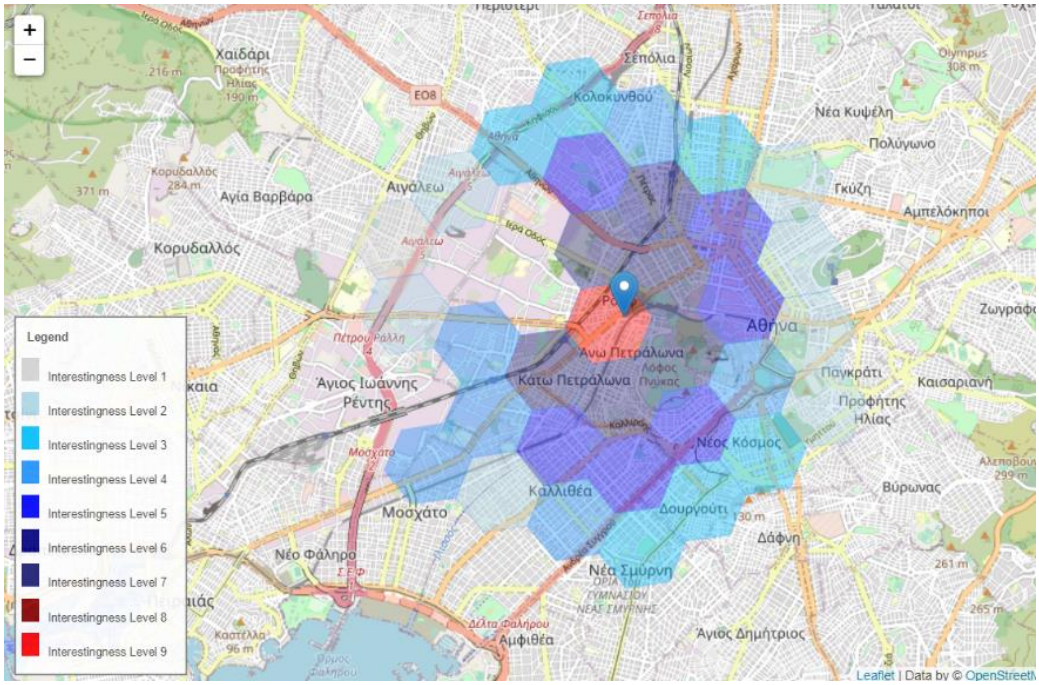


Figure 35. Image created by the 3rd method, based on the attributes "vegan_options", "price_level", "avg_rating" and the aggregation function "mean".

Mean Absolute Distance (MAD):

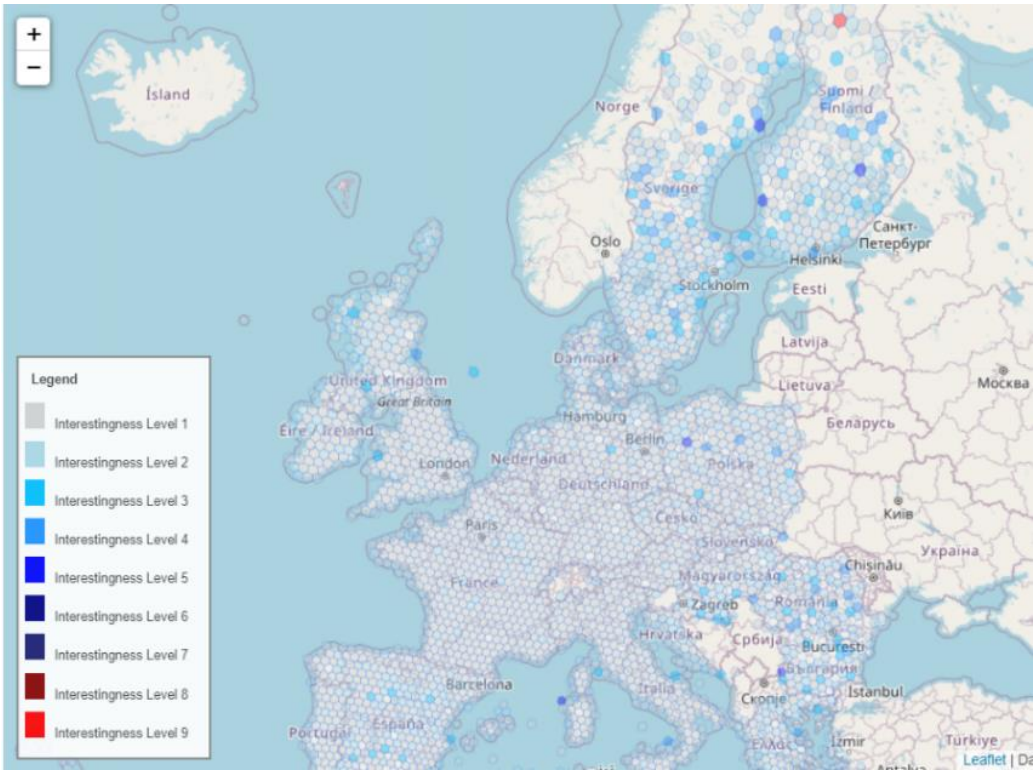


Figure 36. View created by the 1st method, based on the attributes "vegetarian_friendly", "price_level", "avg_rating" and the aggregation function "mean".

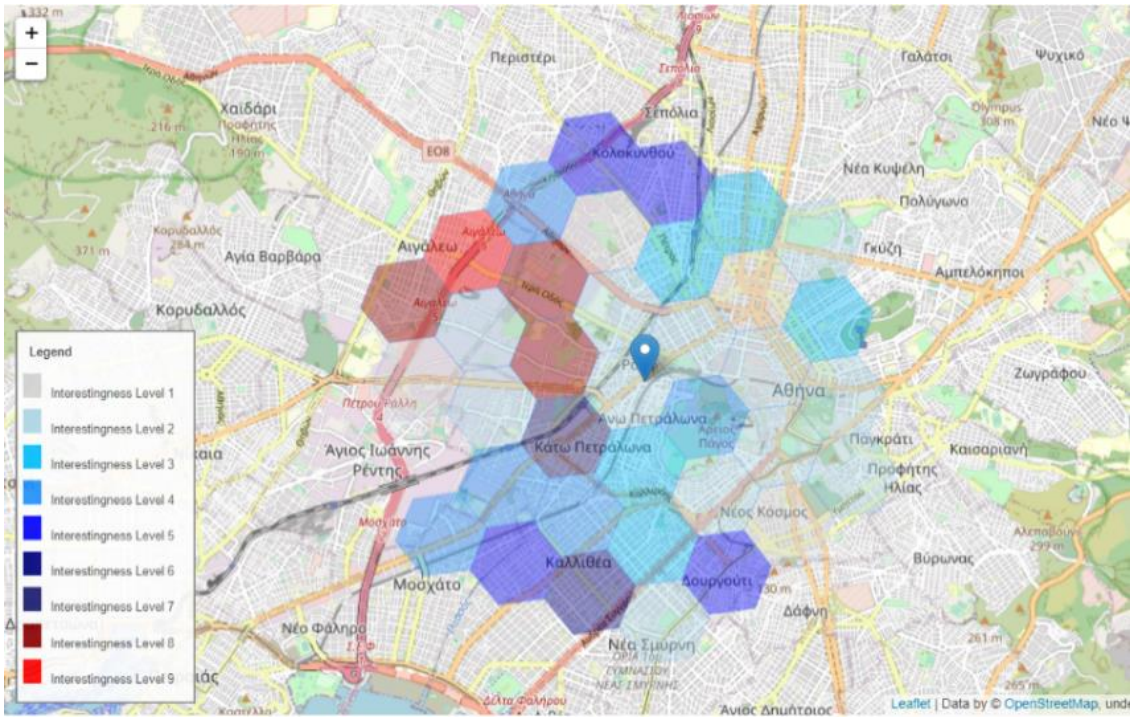


Figure 37. Image created by the 2nd method, based on the attributes "vegan_options", "gluten_free", "avg_rating" and the aggregation function "mean".

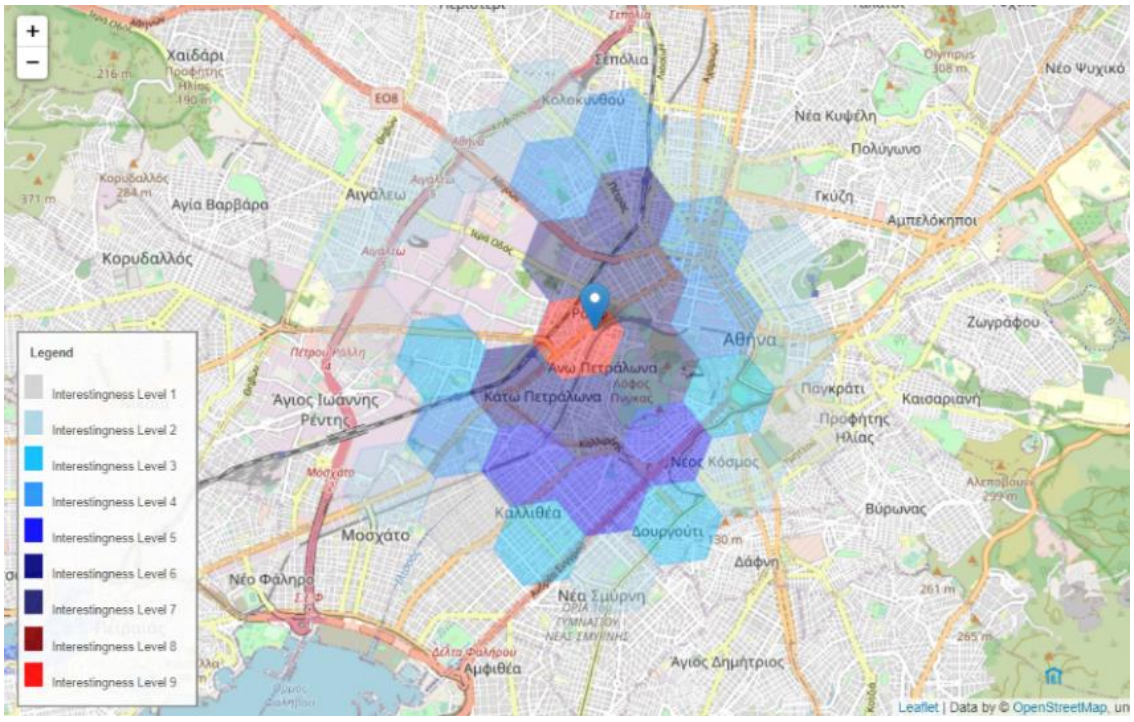


Figure 38. Image created by the 3rd method, based on the attributes "vegan_options", "price_level", "avg_rating" and the aggregation function "mean".

Comparing all these images, we get one major conclusion. This conclusion refers to the differences and the similarities of the results provided by each feature. It seems that all these features, individually, suggest the same hexagons as the most interesting ones, having them sorted in a bit different order. For example, at the first method, the Euclidean distance returns as the most interesting hexagon the one with hash value «841e567fffffffff», at Romania, while the same hexagon is suggested as the second most interesting in any other case instead of when using MAD as the main utility feature, where this hexagon is still one of the most interesting ones but it is not included in the top-5 most interesting ones.

We can also see that using the «KL divergence» as the only utility feature in our method, it creates a map much clearer than the other features, while the hexagons that are considered as interesting are much fewer than in any other case. This means that during the computation of the interestingness of a target view it either orders it as very interesting or not interesting at all, or it is affected by null values or outliers and it orders these target views containing them as not interesting at all.

Furthermore, we can observe that in the third method we got a variety of results. Specifically, in the third method we set the additional variable that measures the distance between the point of interest and the center of each hexagon to have a weight of 0.7. So, we expected the method to return as the most interesting hexagon the one that the point of interest falls into that one, as it happened in every case. However, in many cases the interestingness of the hexagons is not always equally between the ones that have the same distance from the point of interest. This, additionally, let us know which hexagon of the same “ring” is rated as the most interesting the most of the times.

Lastly, the most interesting case seems to refer at the second method. In the second method, the hexagons that are suggested as the most interesting ones differ at any case, except when using the RMSE feature and the MAD feature where the hexagons suggested as the most interesting ones are very similar between these two cases.

5.6 Our Final Approach & Results

Observing all these results, we decided that the utility function should take into account the distance metrics (a) Euclidean distance, (b) Manhattan distance, (c) KL divergence and (d) Mean Absolute distance (MAD). All these utility features equally contribute to the computation of the utility score, which means that they firstly get normalized from 0 and 1 and then, since they are four features, each one is weighted by 25% (each weight is equal to 0.25). The utility score measures the interestingness of a target view (U^T) compared to the reference view (U^R). The target view and the reference view are created, as discussed previously, based on a group–by clause by a set of measure and dimension attributes as well as an aggregation function that the user is interested in, where each target view represents the behavior of all points underneath a certain unique hexagon based on the group-by clause. From the other hand, the reference view represents the behavior of all the points of a specific area just like already analyzed at the three use cases (methods) respectively. So, the utility function comes to be following,

$$y = 0.25 * EuD + 0.25 * KL + 0.25 * Manh + 0.25 * MAD$$

where,

y : $the\ utility\ score\ of\ a\ certain\ U^T\ compared\ to\ the\ U^R$
EuD : $the\ value\ of\ Euclidean\ distance\ between\ a\ certain\ U^T\ and\ the\ U^R$
KL : $the\ value\ of\ KL\ divergence\ between\ a\ certain\ U^T\ and\ the\ U^R$
$Manh$: $the\ value\ of\ Manhattan\ distance\ between\ a\ certain\ U^T\ and\ the\ U^R$
MAD : $the\ value\ of\ Mean\ Absolute\ distance\ between\ a\ certain\ U^T\ and\ the\ U^R$

The utility function we presented right above computes the interestingness of a view as it is in the 1st and 2nd method. In the 3rd method, from the other hand, just because the users are interested both in the similarity between U^T and U^R and the distance of the point of interest they inserted and the center of each hexagon in a predefined, by the users, area. The next equation refers to the way the utility score is measured in the third method.

$$y = (1 - a) * x_1 + a * (1 - x_2)$$

where,

y : $the\ utility\ score\ that\ shows\ how\ much\ interesting\ is\ a\ certain\ U^T$
x_1 : $the\ similarity\ score\ between\ U^T and\ U^R as\ explained\ earlier$
x_2 : $the\ distance\ between\ the\ point\ of\ interest\ and\ the\ center\ of\ each\ hexagon$
a : $the\ weight\ of\ how\ interesting\ the\ user\ thinks\ the\ distance\ is\ to\ him\ (0,1)$

We decided to make these changes in order to take more accurate results. The interestingness of a view is computed via many factors that does not lead to a single clear-cut answer. The complexities of data analysis and visualization demand a multi-faceted approach, and our utility function aims to capture these diverse aspects. By incorporating four deviation-based utility features, we create a comprehensive assessment that helps us quantify the interestingness of each view.

The inclusion of the Euclidean distance, which was our initial choice, serves as a foundational measure of dissimilarity between the target view (UT) and the reference view (UR). This alone was a significant step towards evaluating interestingness, but we understood that a holistic view required more dimensions. The Manhattan distance, KL divergence, and Mean Absolute distance (MAD) further enrich our utility function. Each of these features contributes equally to the overall utility score, creating a balanced evaluation process. To ensure fairness, we normalize these features to a range of 0 to 1, making them directly comparable. We, also, allocate equal weights of 25% to these four features, recognizing the importance of each in gauging interestingness. This equitable distribution ensures that no single feature dominates the assessment and allows for a harmonious synthesis of different perspectives.

So, we present the results while using our method with this combination of utility features. We used the same group-by clauses as in the previous chapters. This means that we used the “avg_rating” as measure attribute, both “vegetarian_friendly” and “price_level” as dimension attributes and we used “mean” as the aggregation function when we used our method as already described for the 1st method. In the next figure, which refers to the 1st use case, we can easily detect the interesting hexagons based on the criteria the users set. It, also, suggests as interesting hexagons those that some utility features, individually, did not notice. We can see a redistribution of the interesting hexagons, too. This means that hexagons noted either as interesting ones by an individual feature previously, they are not that interesting anymore or even some not that interesting hexagons noted by a feature, now, seems to be more interesting.

1st method

Observing the 1st method’s results (Figure 39) we can see that there are differences compared to the results provided by each individual metric. To be more specific, we can see that the hexagons that are noted as the most interesting ones are a combination of those noted as the most interesting by every individual utility feature. Furthermore, we see the same happens among the hexagons that are not noted as the most interesting ones. So, there is a redistribution of the interestingness among the target views (hexagons) where the top interesting hexagons are those that were noted as the most interesting ones by each individual utility feature, but in different order. Similar results we obtain from the 2nd and 3rd methods as we present via the following images (from Figure 40 to Figure 44).

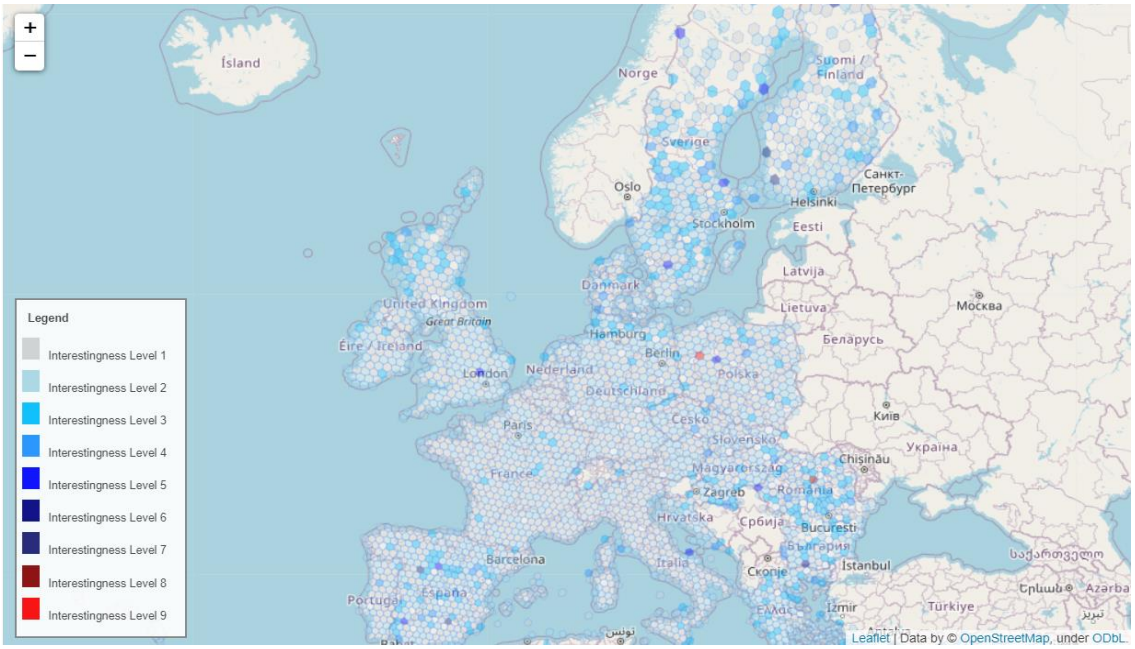


Figure 39. Map showing the interestingness of every hexagon based on average rating values per vegetarian friendly options and their respective price level based on the whole datasets (1st method).

2nd method

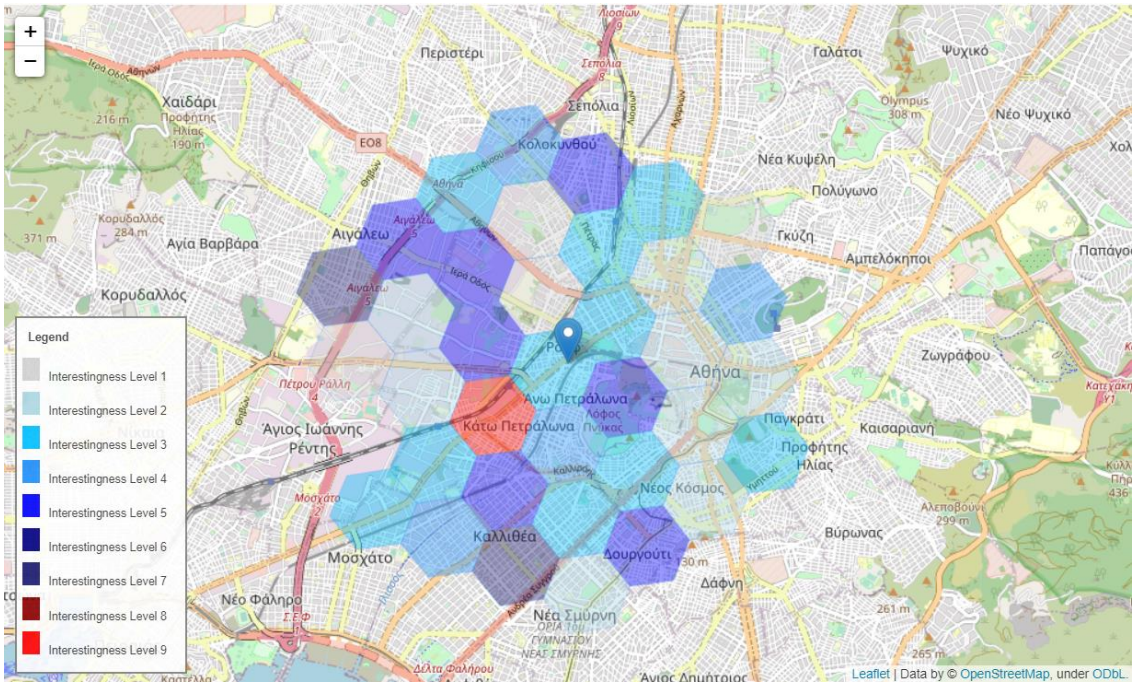


Figure 40. Map created by the 2nd method, showing the interestingness of every hexagon based on average rating values per vegan and gluten free options for the restaurants in the centre of Athens.

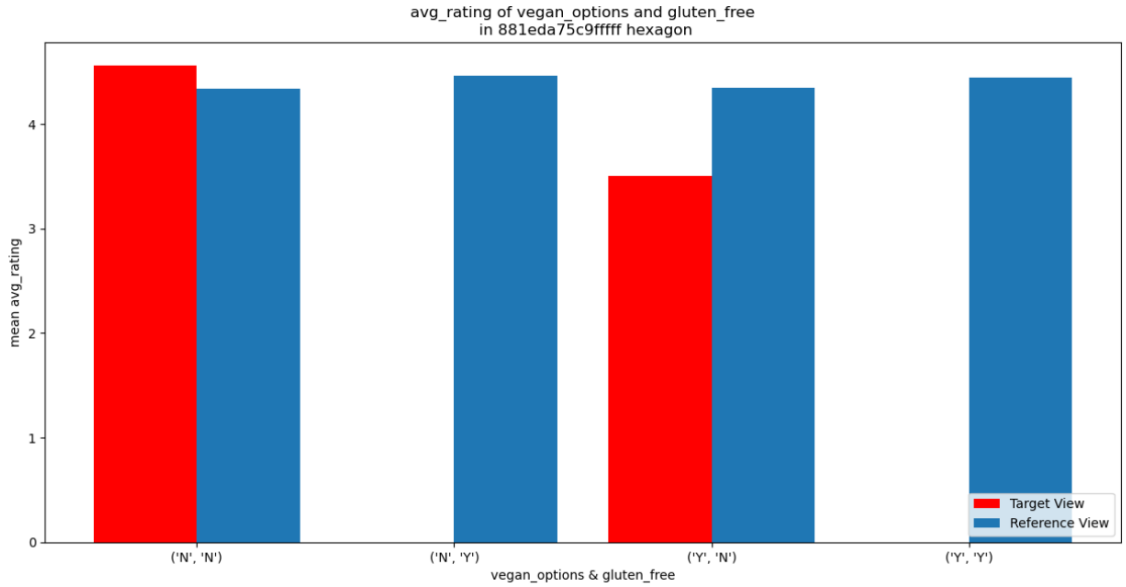


Figure 41. Bar chart showing the comparison between target view (hexagon "881eda75c9fffff") and reference view average rating values per vegan and gluten free options.

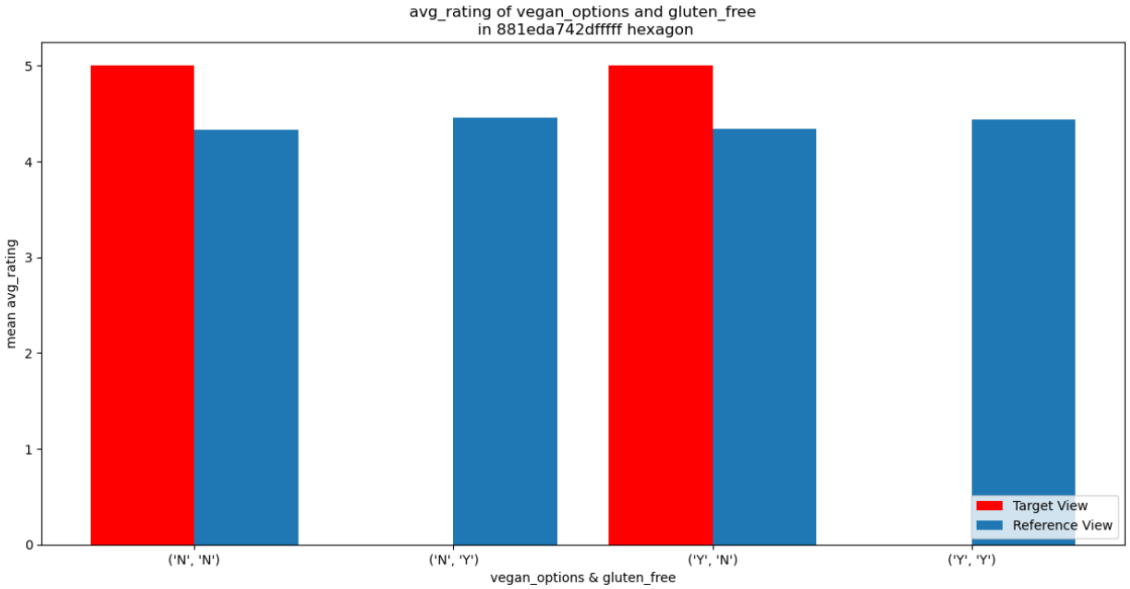


Figure 42. Bar chart showing the comparison between target view (hexagon "881eda742dfffff") and reference view average rating values per vegan and gluten free options.

3rd method

Overall, including a broader range of utility features within the utility function does not yield a definitive and singular answer. Instead, it allows us to quantify the level of interest associated with each view, offering a more multifaceted perspective on the data.

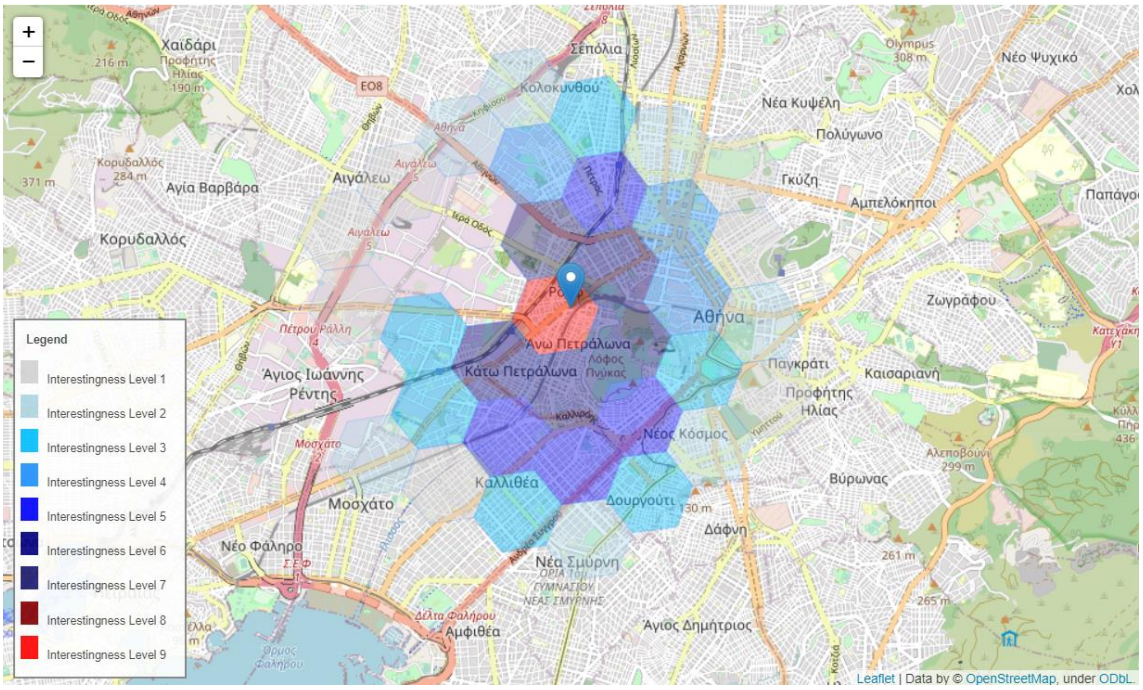


Figure 43. Map created by the 3rd method, showing the interestingness of every hexagon based on average rating values per vegan options and their respective price level for the restaurants in the centre of Athens.

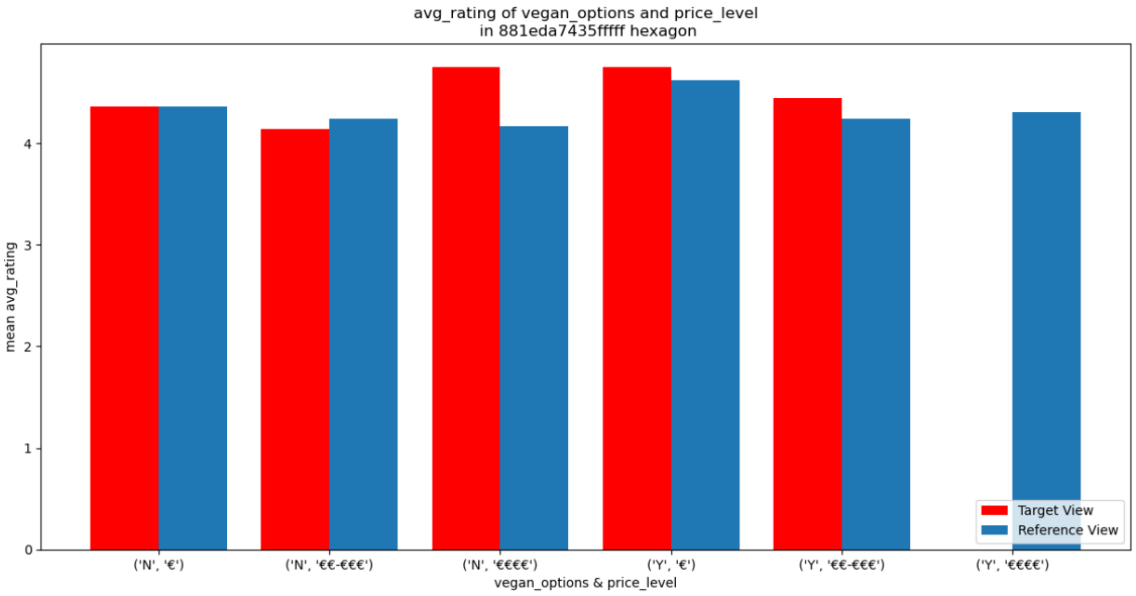


Figure 44. Bar chart showing the comparison between target view (hexagon "881eda7435fffff") and reference view average rating values per vegan options and their respective price level.

6 User Evaluation

In order to evaluate our method, we created a questionnaire and asked different users to complete it. To complete this questionnaire, the users have to run a jupyter notebook and to evaluate the results provided. This type of evaluation is called «user testing» and the main parameters we wanted to evaluate fall in to two major categories; (i) the time needed to show the results (efficiency) and (ii) the simplicity of showing the results, where the users have to say if they can detect the interesting hexagons easily or not. Additionally, one third parameter we would like to examine refers to the utility function we used, so we could compare it to other utility functions and see which one returns more interesting views, that would help the users to find insights or patterns. We need to evaluate all three methods our approach implements, individually, via these parameters to have an extensive overview of how our method works and as future work to make the changes needed in order to improve it.

For this purpose, we ask the users to implement 3 tasks, that correspond to the 3 methods our approach conducts, and state their opinion about our method, so we can evaluate it and make it even better. Guidelines for implementing the tasks are written both in this page and the jupyter notebook you will be given to run. The datasets we gave to the users in order to implement these tasks refers to the restaurants in EU that have been recorded in trip advisor. It contains the geographical data needed that is nothing else than columns of the latitude and longitude each restaurant is placed. It also contains other columns either categorical that contain information such as the names of the restaurants, the type of cuisines each restaurant supports and serves, key words that describe the restaurants individually, 'Y' (yes) - 'N' (no) if a restaurant is vegetarian friendly or not etc. or numeric columns like the average rating of each restaurant the number of people that rated a restaurant as excellent/good/terrible, the number of hours each restaurant remains open daily etc.

The users have to select one measure (numeric) attribute, one or two dimension (categorical) attributes of the datasets and one aggregation function like “mean". All the attributes of the datasets and the aggregation functions are also shown to the users in the jupyter notebook, so they can choose the attributes needed quite easily. To complete the 1st task, firstly, the users have to write the key word "all" which completely corresponds to our approach and wait for the results. Then, the users should pay attention to the time needed to show the results, needs to check if it is possible to detect the most interesting hexagon and to tap on it to see even more detailed results (bar chart).

The second and the third tasks are identical since in both cases the users are given a specific area such as the area around the center of Athens. Then, just like in the first case, the users have to check if the results are provided fast enough and to try to detect the most interesting hexagon and to tap on it. The difference between the 2nd and 3rd tasks is that the users should check if they can notice any changes observing the results the 2nd and 3rd methods provide.

Furthermore, we added one more task that it is optional. We ask the users to repeat the same steps entering the key words "Euclidean", "KL", "Manhattan", "Mad", one of them each time they run the notebook and we ask them to try to observe which one derives the most interesting results according to their opinion. This step was added in order to define which utility feature is the most suitable in generating interesting views, according to the users, so we could change their weights in the equation that measures the interestingness of each view (utility function). This is an area we will investigate even more in the future, so we can improve it even more.

In the next images we present some of the results provided by the questionnaire. The moment we conduct this thesis, a sum of 20 users answered the questionnaire and the evaluation was based on their answers. It is also really important to notice that in all cases where bar charts are created the index “0” resembles something that is really difficult or very slow to be done or to provide results. From the other hand, the index “5” resembles something very easy or fast to be done. So, as much closer a rating is to index “5” the best opinion a user has of our approach.

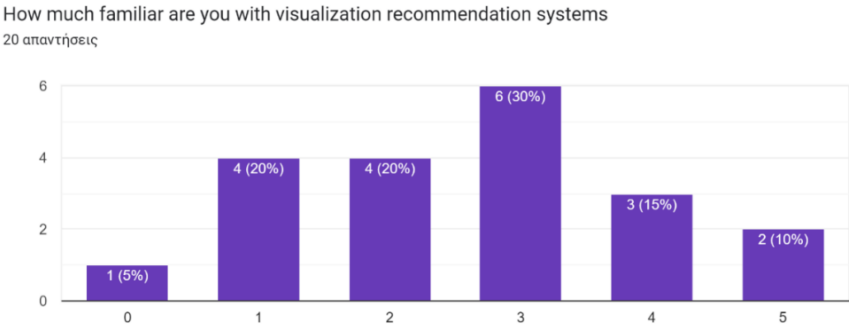


Image 1. Users' familiarity with VRSs.

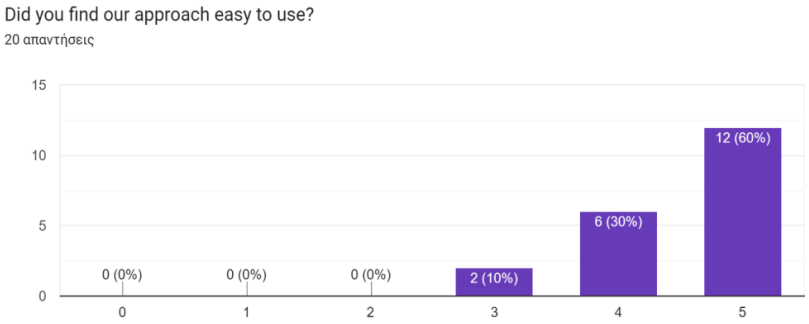


Image 2. How easy it was for the users to make the tasks.

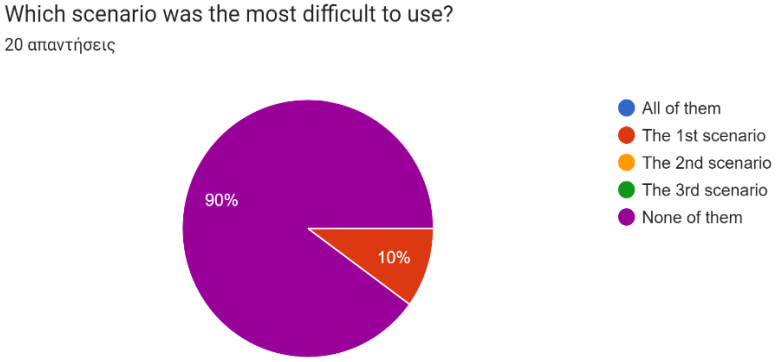


Image 3. The scenario that was the most difficult to do.

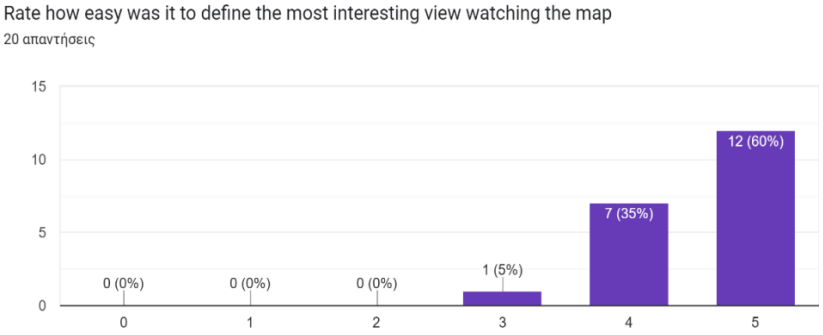


Image 4. Difficulty of defining the most interesting view / hexagon on the map at all tasks.

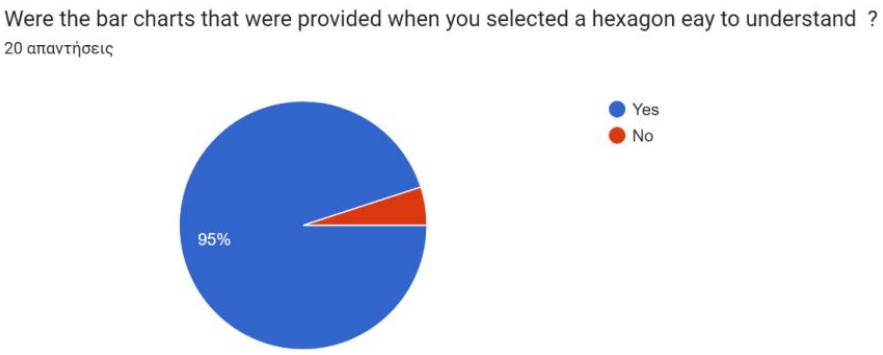


Image 5. Difficulty of understanding the bar charts that pop up on the map after selecting a certain hexagon (general opinion for all tasks).

As shown in these images, the questionnaire was completed by users that have any type of relationship with visualization recommendation systems (*Image 1*). This is important to us because our approach was made not only to help analysts that are already capable of using any sort of systems and that are able to find out any insights or patterns easily by themselves, but also for scientists that have lack of experience and that they want a helping hand in understanding their data and discovering interesting insights out of them. Despite their knowledge about VRSs, the majority of the users characterized our approach easy to use (*Image 2*), so we could say that we managed one of our goals, to make it user friendly, and everyone should be able to find it easy to use. From the other hand, only a small amount of the users that completed the questionnaire said that they found it a bit difficult to use (*Image 2*), adding that the method they found difficult to use was the 1st one (*Image 3*). Additionally, all of the users managed to understand what they were watching either by detecting the most interesting hexagons on the map (*Image* 4) or by watching the content (bar charts) that popups when selecting a certain hexagon (*Image 5*). All this information proves the simplicity we wanted our method to have.

Then, we present the results of one of the most crucial topics that this questionnaire was conducted for, the efficiency. The efficiency of our approach is similar to the time needed to return the results to the users after they input the attributes of the datasets they are interested in. This process contains the creation of hexagons, the computation of the utility score between target views and the reference view, the creation of the map with the hexagons on it and the popup option where it generates a bar chart to better understand the interestingness of each target view etc. and even the filtering of the data based on the location the users inserted for the purposes of the 2nd and 3rd method. As shown in the next figures, the overall time need to complete all tasks was mostly around 5 to 10 minutes (*Image 6*).

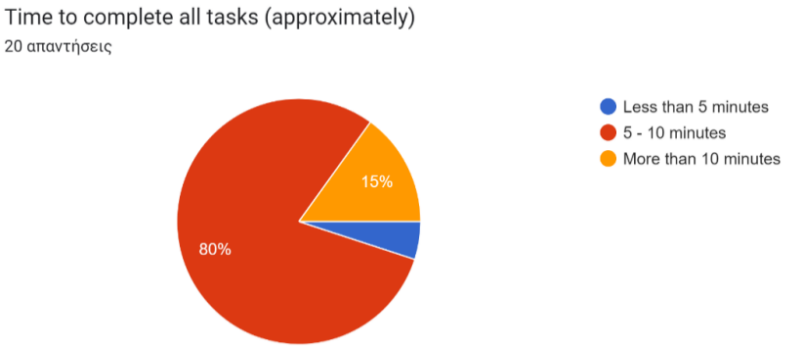


Image 6. Time needed to complete all the tasks.

The users rated the efficiency of the whole approach and tasks above average (*Image 7*), but they waited much longer than expected while using the 1st method of our approach, where in most cases it took 4 to 5 minutes, at a percentage of 40%, or even more than 5 minutes (25%) to show its results (*Image 8*). The majority of users rated the 1st method – task as average or even slow while trying to generate visualizations and only a small amount of them said that it was fast enough (*Image 9*) and it may happen because of the computing power their devices have. From the other hand, the 2nd and 3rd methods were completed much faster by all users (*Image 11, Image 13*) and in most cases it took them less than two minutes (*Image 10, Image 12*). This probably happens because of the volume of the data, especially because the bar charts that are shown from popup choices when the users select a hexagon on the map are computed in advance. In the 1st method the data set contains more than 1 million rows while in the next two methods, by filtering the data needed within a certain bounding box, the volume of the data becomes much less, so our computations are made much faster.

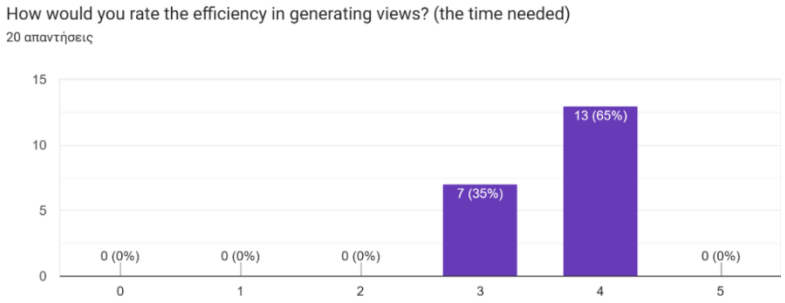


Image 7. Rating the time needed (efficiency) to complete all the tasks.

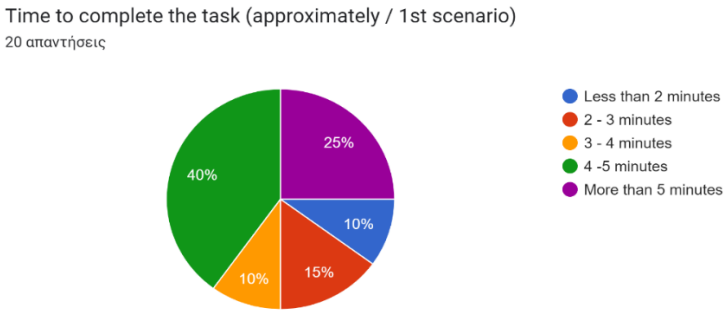


Image 8. Actual time to complete the 1st task.

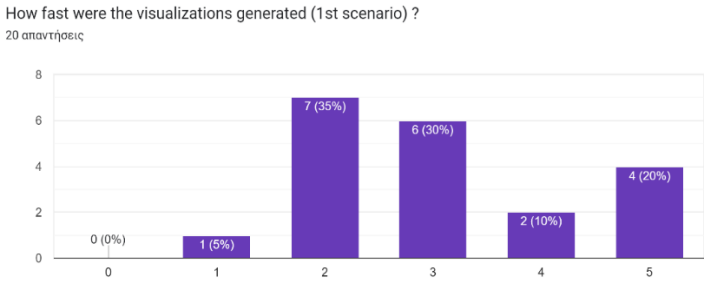


Image 9. Rating the velocity of generating visualizations using the 1st method.

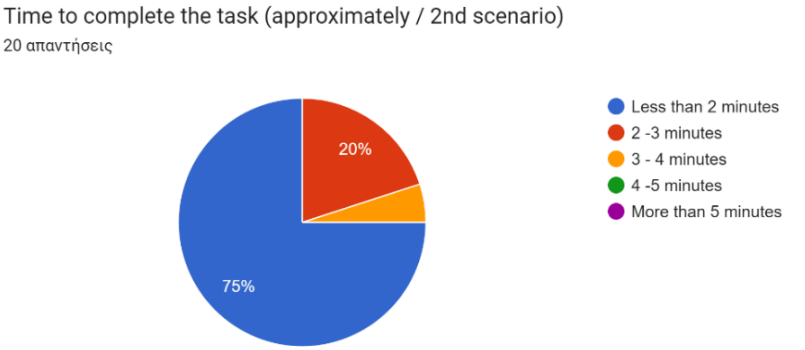


Image 10. Actual time to complete the 2nd task.

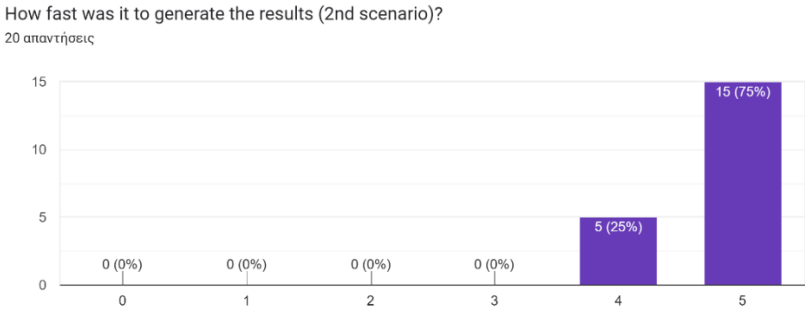


Image 11. Rating the velocity of generating visualizations using the 2nd method.

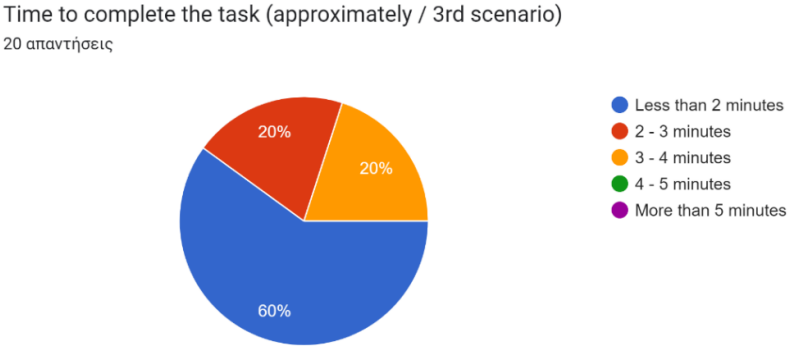


Image 12. Actual time to complete the 3rd task.

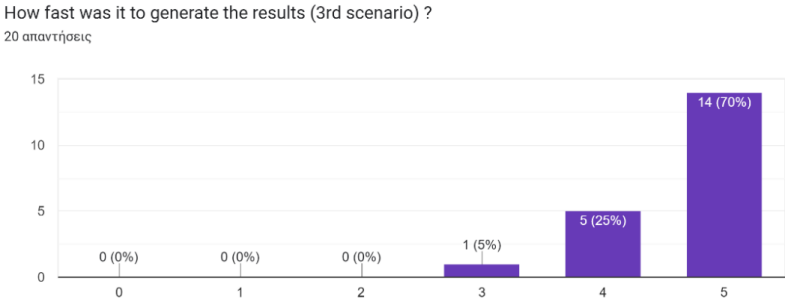


Image 13. Rating the velocity of generating visualizations using the 3rd method.

In addition, we asked the users to complete an optional task in order to see if the contribution of the utility features should be equally derived. Luckily, all of them completed this task (*Image 14*). It actually asks the users to make all these steps again, four times, using as utility function every individual utility feature, one of them each time, and see which one returns the most interesting views based on their preferences.

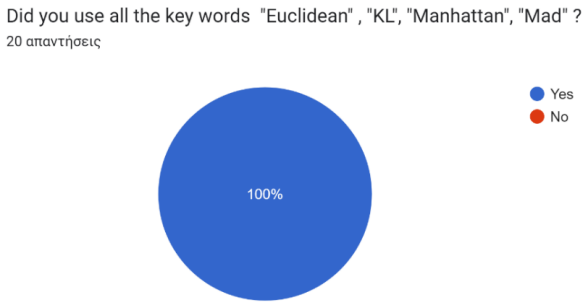


Image 14. If all the key words (deviation based metrics) were used or not.

It seems that Euclidean Distance was the feature that gathered the majority of votes with 45% (*Image 15*). The other three comes to be Manhattan Distance, KL Divergence, MAD from the second to the fourth place respectively (*Image 15*). The outcome lets us understand that the utility features should affect the utility function in a way that is more suitable to the preferences of each user. So, the process of finding the most suitable weights of the utility function's variables is going to concern us in our future studies.

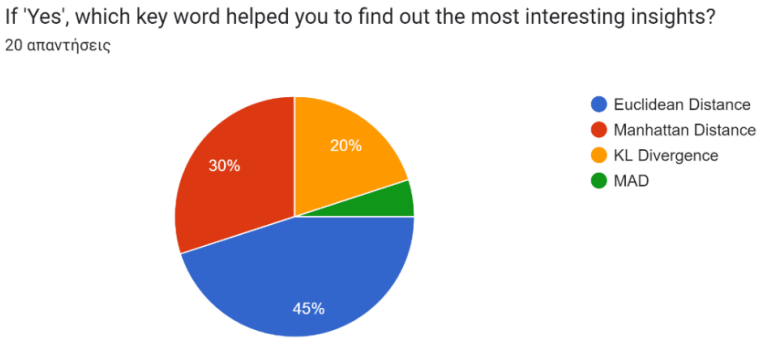


Image 15. Distance metrics the users found as the most useful to use.

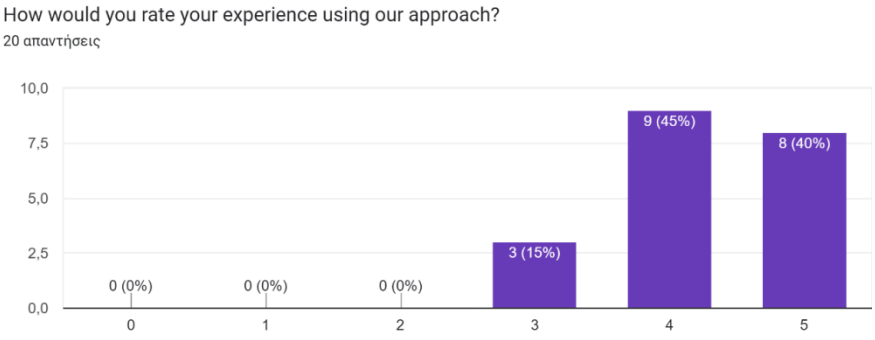


Image 16. Users' rating on our approach.

In conclusion, it appears that users generally had a positive experience with our approach, as indicated by their response to Image 16. Additionally, a majority of them expressed an interest in incorporating our approach into their own research, as evidenced by the findings in Image 17. However, it is worth noting that most users were not entirely satisfied with our approach. This feedback serves as a motivation for us to continue working on this project diligently, with the aim of making substantial improvements.

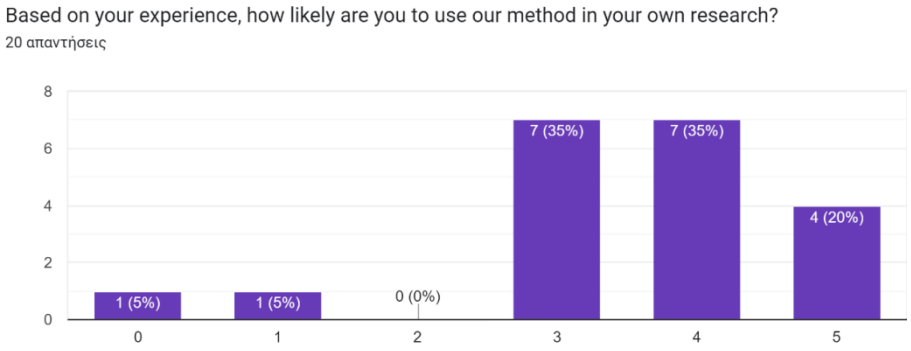


Image 17. Users' preference on our method.

7 Conclusion & Future work

In conclusion, visualization recommendation systems offer a valuable solution for data analysts, scientists, and other users who want to select the most appropriate visualization for their data. These systems can help reduce the time and effort required to explore and analyze data, while also improving the accuracy and relevance of the insights generated. By implementing best practices for data preprocessing, algorithm selection, and user interface design, and regularly evaluating the effectiveness of the system, you can create an effective visualization recommendation system that delivers meaningful insights and value to users. The key to success lies in understanding the specific needs and requirements of your users and tailoring the system to meet those needs. Despite some limitations, data visualization recommendation systems have the potential to greatly enhance the data analysis experience for users of all skill levels.

Overall, our approach was made to help analysts to better understand their data, efficiently and effectively. It, mainly, supports data sets that contain geographical data (latitude, longitude) in order to suggest interesting areas. So, it interacts with the users to understand the variables and locations they want to study and then it shows the most interesting areas on a map via a hexagon – based grid framework, where each hexagon is colored based on its interestingness score. It, also, offers the ability to the users to see the difference between the general view (reference view) of the data set they are studying and a selected hexagon (target view), based on their attribute preferences by a popup option that shows the results using bar charts. However, it needs some improvements like (a) making much faster the first method that was really slow while using large data sets, (b) making it to have more interactions with the analysts to capture better their preferences and (c) creating an engine that allows the tuning of the parameters and their weights that the utility function has. In addition, we have to check some more topics, to have a complete approach, such as fairness like it is analyzed in [72] both for sensitive attributes like ‘gender’, ‘race’ and for spatial fairness.

Finally, we could say that visualization recommendation systems represent a promising area of development in the field of data visualization, and we can expect to see continued innovation and improvement in this area in the coming years.

References

1. Linked Data Visualization: Techniques, Tools and Big Data
Laura Po, Nikos Bikakis, Federico Desimoni, and George Papastefanatos | Morgan & Claypool, 2020
2. X. Qin, Y. Luo, N. Tang, G. Li, Making data visualization more efficient and effective: a survey, VLDB J. 29(1) (2020) 93–117.
3. B. Tang, S. Han, M.L. Yiu, R. Ding, D. Zhang, Extracting top-k insights from multi-dimensional data, in: ACM SIGMOD, 2017
4. V. Dibia and Ç. Demiralp. Data2Vis: Automatic Generation of Data Visualizations Using Sequence to Sequence Recurrent Neural Networks. CoRR, abs/1804.03126, 2018.
5. H. Ehsan, M.A. Sharaf, P.K. Chrysanthis, Muve: efficient multi-objective view recommendation for visual data exploration, in: IEEE ICDE, 2016.
6. D. Moritz, C. Wang, G. L. Nelson, H. Lin, A. M. Smith, B. Howe, and J. Heer. Formalizing Visualization Design Knowledge as Constraints: Actionable and Extensible Models in Draco. IEEE Trans. Visualization & Comp. Graphics (Proc. InfoVis), 2018.
7. Martin Krzywinski and Naomi Altman. 2013. Significance, P values and t-tests. In Nature methods.
8. G. Linden, B. Smith, and J. York. Amazon. com recommendations: Item to-item collaborative filtering. Internet Computing, IEEE, 7(1):76–80, 2003
9. G. Adomavicius and A. Tuzhilin. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. Knowledge and Data Engineering, IEEE Transactions on, 17(6):734– 749, 2005.
10. J. Bobadilla, F. Ortega, A. Hernando, and A. Gutierrez. Recommender ´ systems survey. Knowledge-Based Systems, 46:109–132, 2013.
11. Brusilovsky, Peter; Kobsa, Alfred; Nejdl, Wolfgang (2007). *[Lecture Notes in Computer Science] The Adaptive Web Volume 4321 || Collaborative Filtering Recommender Systems., 10.1007/978-3-540-72079-9(Chapter 9), 291–324.* doi:10.1007/978-3-540-72079-9_9
12. Robin Burke (2002). *Hybrid Recommender Systems: Survey and Experiments., 12(4), 331–370.* doi:10.1023/a:1021240730564
13. S. M. Casner. Task-analytic Approach to the Automated Design of Graphic Presentations. *ACM Trans. Graph.*, 10(2):111–151, Apr. 1991.
14. S. Trewin. Knowledge-based recommender systems. Encyclopedia of Library and Information Science: Volume 69-Supplement 32, page 180, 2000.
15. Knowledge-based recommender systems Robin Burke Department of Information and Computer Science University of California, Irvine.
https://www.cs.odu.edu/~mukka/cs795sum09dm/Lecturenotes/Day6/burke-elis00.pdf
16. Robert Kosara and Jock Mackinlay. 2013. Storytelling: The next step for visualization. *Computer* 46, 5 (2013), 44–50.
17. Bongshin Lee, Nathalie Henry Riche, Petra Isenberg, and Sheelagh Carpendale. 2015. More than telling a story: Transforming data into visually shared stories. *IEEE computer graphics and applications* 35, 5 (2015), 84–90.
18. Doris Jung-Lin Lee, Himel Dev, Huizi Hu, Hazem Elmeleegy, and Aditya Parameswaran. 2019. Avoiding Drill-down Fallacies with VisPilot: Assisted Exploration of Data Subsets. In *Proceedings of the 24th International Conference on Intelligent User Interfaces*. Association for Computing Machinery, New York, NY, USA, 186–196.
19. K. Hu, D. Orghian, and C. Hidalgo. DIVE: A Mixed-Initiative System Supporting Integrated Data Exploration Workflows. In *ACM SIGMOD Workshop on Human-in-the-Loop Data Analytics (HILDA)*. ACM, 2018.
20. Google. Explore in Google Sheets. https://www.youtube.com/watch? v=9TiXR5wwqPs, 2015.
21. Methods and metrics for cold-start recommendations. A. I. Schein, A. Popescul, L. H. Ungar and D. M. Pennock. SIGIR 2002.

22. Gong, S.; Cheng, G. Mining user interest change for improving collaborative filtering. In Proceedings of the 2008 Second International Symposium on Intelligent Information Technology Application, Shanghai, China, 21–22 December 2008; Volume 3, pp. 24–27.
23. Rischan Mafrur, Mohamed A. Sharaf, and Hina A. Khan. 2018. DiVE: Diversifying View Recommendation for Visual Data Exploration. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*. Association for Computing Machinery, New York, NY, USA, 1123–1132.
24. He, C., Parra, D., & Verbert, K. (2016). *Interactive recommender systems: A survey of the state of the art and future research challenges and opportunities. Expert Systems with Applications, 56, 9–27.* doi:10.1016/j.eswa.2016.02.013
25. Shneiderman, B. (n.d.). The eyes have it: a task by data type taxonomy for information visualizations. Proceedings 1996 IEEE Symposium on Visual Languages. doi:10.1109/vl.1996.545307
26. Schafer, J.B.; Frankowski, D.; Herlocker, J.; Sen, S. Collaborative filtering recommender systems. In *The Adaptive Web*; Springer: Berlin/Heidelberg, Germany, 2007; pp. 291–324.
27. B. Saket, A. Endert, and C. Demiralp. Task-Based Efectiveness of Basic Visualizations. IEEE Transactions on Visualization and Computer Graphics, 2018.
28. Y. Kim and J. Heer. Assessing Efects of Task and Data Distribution on the Efectiveness of Visual Encodings. *Computer Graphics Forum (Proc. EuroVis)*, 2018.
29. C. N. Knafic. Is there a single right answer? http://www.storytellingwithdata.com/blog/2016/1/12/ is-there-a-single-right-answer, 2016
30. M. A. Borkin, A. A. Vo, Z. Bylinskii, P. Isola, S. Sunkavalli, A. Oliva, and H. Pfster. What Makes a Visualization Memorable? IEEE Transactions on Visualization and Computer Graphics, 19(12):2306–2315, Dec 2013.
31. W. S. Cleveland and R. McGill. Graphical Perception: Theory, Experimentation, and Application to the Development of Graphical Methods. Journal of the American Statistical Association, 79(387):531–554, 1984
32. B. Jones. Data Dialogues: To Optimize or to Satisfce When Visualizing Data? https://www.tableau.com/about/blog/2016/1/ data-dialogues-optimize-or-satisfce-data-visualization-48685, 2016.
33. Zeshan Fayyaz, Mahsa Ebrahimian, Dina Nawara, Ahmed Ibrahim and Rasha Kashef; Recommendation Systems: Algorithms, Challenges, Metrics, and Business Opportunities, 2020
34. J. Mackinlay, P. Hanrahan, and C. Stolte. Show Me: Automatic Presentation for Visual Analysis. *IEEE Transactions on Visualization and Computer Graphics*, 13(6):1137–1144, Nov. 2007.
35. Uttkarsha Bhosale. The Limitations of ChatGPT: How human editors remain indispensable in academic writing (May 18, 2023)
36. S. Trewin. Knowledge-based recommender systems. Encyclopedia of Library and Information Science: Volume 69-Supplement 32, page 180, 2000.
37. G. Linden, B. Smith, and J. York. Amazon. com recommendations: Item-to-item collaborative filtering. Internet Computing, IEEE, 7(1):76–80, 2003.
38. G. Holmes, A. Donkin, and I. H. Witten. Weka: A machine learning workbench. In Conf. on Intelligent Information Systems '94, pages 357–361. IEEE, 1994
39. Pedregosa et al. Scikit-learn: Machine learning in python. The Journal of Machine Learning Research, 12:2825–2830, 2011.
40. D. B. Perry, B. Howe, A. M. Key, and C. Aragon. Vizdeck: Streamlining exploratory visual analytics of scientific data. 2013.
41. Xiaozhong Zhang, Xiaoyu Ge, Panos K. Chrysanthis, and Mohamed A. Sharaf. 2021. ViewSeeker: An Interactive View Recommendation Framework. *Big Data Res.* 25, C (jul 2021), 15 pages.
42. A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer. Automatic differentiation in PyTorch. 2017.

43. D. B. Perry, B. Howe, A. M. Key, and C. Aragon. VizDeck: Streamlining exploratory visual analytics of scientifc data. In *iConference*, 2013.
44. M. Vartak, S. Huang, T. Siddiqui, S. Madden, and A. Parameswaran. Towards visualization recommendation systems. Workshop on Data Systems for Interactive Analytics (DSIA), 2015.
45. S. van den Elzen and J. J. van Wijk. Small multiples, large singles: A new approach for visual data exploration. Computer Graphics Forum, 32(3pt2):191–200, 2013.
46. K. Wongsuphasawat, D. Moritz, A. Anand, J. Mackinlay, B. Howe, and J. Heer. Voyager: Exploratory analysis via faceted browsing of visualization recommendations. IEEE Transactions on Visualization and Comp. Graphics, 22(1):649–658, 2016.
47. Sulthana, A.R., Gupta, M., Subramanian, S. *et al.* Improvising the performance of image-based recommendation system using convolution neural networks and deep learning. *Soft Comput* **24**, 14531–14544 (2020). https://doi.org/10.1007/s00500-020-04803-0
48. Agrawal R., Faloutsos C., Swami A. Efficient similarity search in sequence databases. Proc. 4 Th Int. Conf. On Foundations of Data Organizations and Algorithms, 1993. – Chicago. pp. 69-84.
49. Fast Distance Metric Based Data Mining Techniques Using P-trees: k-Nearest-Neighbor classification and kClustering : A Thesis Submitted to the Graduate Faculty Of the North Dakota State University.
50. A. Singh, A. Rana, and A. Yadav, “K-means with Three different Distance Metrics,” Int. J. Comput. Appl., vol. 67, no. 10, pp. 13–17, 2013.
51. Wang, Wei; Zhang, Guangquan; Lu, Jie (2015). *Collaborative Filtering with Entropy-Driven User Similarity in Recommender Systems. International Journal of Intelligent Systems, 30(8), 854–870.* doi:10.1002/int.21735
52. S. F. Roth, J. Kolojejchick, J. Mattis, and J. Goldstein. Interactive Graphic Design Using Automatic Presentation Knowledge. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI ’94, pages 112–117, New York, NY, USA, 1994. ACM.
53. B. Saket, A. Endert, and C. Demiralp. Task-Based Efectiveness of Basic Visualizations. *IEEE Transactions on Visualization and Computer Graphics*, 2018.
54. A. Satyanarayan, D. Moritz, K. Wongsuphasawat, and J. Heer. Vega-Lite: A grammar of interactive graphics. IEEE Transactions on Visualization and Computer Graphics, 23(1):341–350, 2017. doi: 10.1109/TVCG.2016.2599030
55. Wang, W., Zhang, G., & Lu, J. (2016). *Member contribution-based group recommender system. Decision Support Systems, 87, 80–93.* doi:10.1016/j.dss.2016.05.002
56. Lesota, O., Melchiorre, A., Rekabsaz, N., Brandl, S., Kowald, D., Lex, E., et al. (2021). “Analyzing item popularity bias of music recommender systems: are different genders equally affected?,” in *Proceedings of the Fifteenth ACM Conference on Recommender Systems, RecSys '21* (New York, NY: Association for Computing Machinery), 601–606. doi: 10.1145/3460231.3478843
57. Musto, Cataldo; de Gemmis, Marco; Semeraro, Giovanni; Lops, Pasquale (2017). *[ACM Press the Eleventh ACM Conference - Como, Italy (2017.08.27-2017.08.31)] Proceedings of the Eleventh ACM Conference on Recommender Systems - RecSys '17 - A Multi-criteria Recommender System Exploiting Aspect-based Sentiment Analysis of Users' Reviews. , (), 321–325.* doi:10.1145/3109859.3109905
58. Kuwata S and Ueda N. One-shot collaborative filtering. In: 2007 IEEE symposium on computational intelligence and data mining, Honolulu, HI, 1 March–5 April 2007, pp. 300–307. New York: IEEE.
59. Yu K, Schwaighofer A, Tresp V et al. Probabilistic memory-based collaborative filtering. IEEE Trans Knowl Data Eng 2004; 16(1): 56–69.
60. Hofmann T and Puzicha J. Latent class models for collaborative filtering. IJCAI 1999; 99: 688–693.
61. Marlin B, Zemel RS, Roweis S et al. Collaborative filtering and the missing at random assumption, 2012, https://arxiv.org/abs/ 1206.5267

62. Kullback RA and Leibler S. On information and sufficiency. Ann Math Stat 1951; 22: 79–86.
63. Ying, Yuanping; Feinberg, Fred; Wedel, Michel (2006). *Leveraging Missing Ratings to Improve Online Recommendation Systems. Journal of Marketing Research, 43(3), 355–365.* doi:10.1509/jmkr.43.3.355
64. Zhu, Z., Hu, X., Caverlee, J.: Fairness-aware tensor-based recommendation. In: Cuzzocrea, A., Allan, J., Paton, N.W., Srivastava, D., Agrawal, R., Broder, A.Z., Zaki, M.J., Candan, K.S., Labrinidis, A., Schuster, A., Wang, H. (eds.) Proceedings of the 27th ACM International Conference on Information and Knowledge Management, CIKM 2018, Torino, Italy, October 22–26, 2018, pp. 1153–1162. ACM (2018). https://doi.org/10.1145/3269206.3271795
65. Yashar Deldjoo;Vito Walter Anelli;Hamed Zamani;Alejandro Bellogín;Tommaso Di Noia; (2021). *A flexible framework for evaluating user and item fairness in recommender systems. User Modeling and User-Adapted Interaction, (), –.* doi:10.1007/s11257-020-09285-1
66. F. Viégas, M. Wattenberg, D. Smilkov, J. Wexler, and D. Gundrum. Generating charts from data in a data table. US 20180088753 A1., 2018.
67. A. Chakrabarti, F. Ahmad, and C. Quix, ‘‘Towards a rule-based visu1261 alization recommendation system,’’ in Proc. 13th Int. Joint Conf. 1262 Knowl. Discovery, Knowl. Eng. Knowl. Manage., 2021, pp. 57–68, doi: 1263 10.5220/0010677100003064
68. K. Wongsuphasawat, D. Moritz, A. Anand, J. Mackinlay, B. Howe, and J. Heer. Towards A General-Purpose Query Language for Visualization Recommendation. In *ACM SIGMOD Workshop on Human-in-the-Loop Data Analytics (HILDA)*, 2016.
69. Hu, Kevin; Bakker, Michiel A.; Li, Stephen; Kraska, Tim; Hidalgo, César (2019). [ACM Press the 2019 CHI Conference - Glasgow, Scotland Uk (2019.05.04-2019.05.09)] Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems - CHI '19 - VizML, 1–12.
70. https://towardsdatascience.com/uber-h3-for-data-analysis-with-python-1e54acdcc908
71. M. Vartak, S. Rahman, S. Madden, A.G. Parameswaran, N. Polyzotis, SEEDB: efficient data-driven visualization recommendations to support visual analytics, Proc. VLDB Endow. 8 (13) (2015) 2182–2193.
72. Giorgos Giannopoulos;George Papastefanatos;Dimitris Sacharidis;Kostas Stefanidis; (2021). Interactivity, Fairness and Explanations in Recommendations . Adjunct Proceedings of the 29th ACM Conference on User Modeling, Adaptation and Personalization.
73. Papastefanatos, G., Alexiou, G., Bikakis, N., Maroulis, S., Stamatopoulos, V.: Visualfacts: A platform for in-situ visual exploration and real-time entity resolution. In: Workshop on Big Data Visual Exploration & Analytics (BigVis) (2022)
74. S. Maroulis, N. Bikakis, G. Papastefanatos, P. Vassiliadis, RawVis: A System for Efficient In-situ Visual Analytics, in: ACM SIGMOD, 2021.
75. G. Alexiou, G. Papastefanatos, V. Stamatopoulos, G. Koutrika, N. Koziris, Queryer: A framework for fast analysis-aware deduplication over dirty data (2022). arXiv:2202.01546.