AI-driven design exploration: use of Reinforcement learning-based recommender system for parametric design space exploration

Md Shariful Alam

A thesis
submitted in partial fulfillment of the
requirements for the degree of

Master of Science

University of Washington
2023

Committee:
Tomás Méndez Echenagucia
Narjes Abbasabadi
Karthik Mohan

Program Authorized to Offer Degree:

Architecture

University of Washington

Abstract

AI-driven design exploration: use of Reinforcement learning-based recommender system for parametric design space exploration

Md Shariful Alam

Chair of the Supervisory Committee:

Tomás Méndez Echenagucia

Department of Architecture

In the current practice of architectural design, performance analysis is an essential step that involves simulating various design options to identify the most optimal solution. However, the process can be time-consuming, especially when the design space is vast. To address this issue, designers often use optimization algorithms to find the best solution, but simulating each design option is still a significant bottleneck. Surrogate models offer a potential solution by creating a simplified model that approximates the behaviors of the actual system. This model can then be used to simulate multiple design options efficiently. While surrogate models can help speed up the performance analysis process, they still require a significant amount of data to train effectively. Additionally, optimization done with surrogate models cannot account for aesthetic preferences, which are essential for architectural design.

The paper proposes a novel design framework that leverages AI and machine learning models to address the aforementioned challenges. To demonstrate the efficacy of the framework, a parametric model is

developed to generate a large number of design alternatives for a multi-story office building in Seattle. Multiple design spaces of different sizes are investigated to validate the framework. The proposed framework consists of two sections. The first section involves three consecutive layers to enable faster and more accurate prediction of performance for all design alternatives. The annual energy consumption is simulated using EnergyPlus. The first step is to convert the design parameters into weighted parameters to aid the machine learning models in understanding their distinct behaviours. The number of weighted parameters is then reduced to three using different dimensionality reduction algorithms to visualize clusters in the last step. The final step involves clustering the entire design space effectively so that the performance outcome of the centroid of the cluster can be a proper representative of all other data points in that cluster. Multiple combinations of weighting parameters, dimensionality reduction methods, and clustering models are experimented with to identify the set of algorithms that can predict the performance outcome of the entire design space with the least amount of error using a smaller number of clusters.

The second section of the proposed framework involves an online dashboard that enables the exploration of the design space. The dashboard includes a reinforcement learning-based recommender system that seeks to understand user preferences through interaction and recommends similar design alternatives in each iteration. The reward function of the recommender system is customized to prioritize high-performing alternatives and pull the designer's preference in that direction. The proposed framework enables designers to explore a massive design space strategically and effectively within a short amount of time.

Keywords: Parametric design, machine learning, reinforcement learning, recommender system, artificial intelligence, design space exploration, surrogate model, annual energy consumption, data mining, computational design, Energy plus, clustering, dimensionality reduction, design dashboard.

Table of Contents

Acknowledgements

I would like to start expressing my gratitude towards my advisor Professor Tomás Méndez Echenagucia. I remember the day I explained my interest to work on machine learning and how he persuade me in making something meaningful for the architects. While I was lost, he guided me to work on this excellent idea. At the end I am feeling complacent for achieving promising results from this study under his excellent guidance. He introduced me the amazing world of design computation and I found my spark in it. Tomas is more than a Professor, not only to me but also every other student he supervised. He is a friend, a mentor, a well-wisher and full of inspiration. I feel extremely lucky for having him as my chair, it's the greatest award I could have from the University of Washington.

I would also like to extend my thanks to Professor Narjes Abbasabadi for her unwavering support and invaluable guidance. Her insights into machine learning and artificial intelligence were instrumental in helping me achieve my goals. I am grateful to Professor Karthik Mohan for his genuine interest in my study and his diverse perspectives, which inspired me to have confidence in my multidisciplinary research.

I am deeply grateful to Professor Mehlika Inanici, and everyone involved in awarding me the "Top Scholar Award." The scholarships I received from the Department of Architecture made it possible for me to pursue my studies without financial worries. Professor Mehlika Inanici, as the program head, has always supported and encouraged me to push boundaries and go beyond expectations. I would also like to thank Professor Chris Meek for sharing his invaluable knowledge on building performance analysis and energy consumption. And a special thanks to Roark Congdon for making my journey so enjoyable and peaceful. Your dedication has taught me to approach work with enthusiasm and stay motivated.

I extend my sincere thanks to Chi Aoyama for sharing his insights as a practicing professional. His input helped me rethink my study in the context of real-world designers and enabled me to generate more practical implementations of my research. I am grateful to Jason Steiner for showing a genuine interest in my thesis and providing constructive feedback during the review process. And thanks to Ramon Weber for generously giving his time and sharing his expert opinions.

Lastly, I would like to express my deepest appreciation to my parents, who have dedicated their lives to providing me with a good education. Their unwavering support and presence in my life have allowed me to pursue my research without any worries. I would not have come this far or pursued my passions without them. They have always been my main driving force, and I am forever grateful for their love and encouragement.

Chapter 1: Introduction

Our design process has changed over time. In the early stages, our design was more dependent on our intuitive cognition and personal experiences. With the development of technology now we achieved the ability to predict the performance of our design before execution. Various tools have been developed to simulate daylight [1], acoustics [2], and radiance [3] inside or outside our buildings. Within any given context we can now predict the annual energy consumption of our design within a very short time [4]. To provide a more sustainable design solution Architects are now more prone to analyze their design solutions and redesign if the predicted performance is not up to the mark. This transformation from a linear design workflow to a circular one brings out many benefits to the industry. Now designers can be more accurate and confident about their designs.

1.1 Current limitations in parametric design space explorations:

It is obvious that the more experiments we can go through at the initial stage of our design, the more we can avoid expenses at the later stages [5]. The process of re-designing and re-modelling for better performance outcomes still requires a lot of time. This can be avoided to some extent, by making a parametric model that can generate many different alternatives with the changes of different parameters [6]. In the parametric model a 'Design space' is an essential concept [7]. It encompasses the full spectrum of design alternatives that can be explored, within a specific set of design constraints. The design space can include thousands of design options having different materials, shapes, sizes or orientations. Each design option can be referred to as a data point.

1.1.1 Lack of inclusion of User's preference

If the design space is large enough, investigating all of them would be extremely tedious for the design team. To find the best option for them we need to run optimization [8]. The goal of optimization is to find a design solution that satisfies the project's functional specifications and environmental objectives while maximizing efficiency and lowering costs. Different algorithms can perform faster optimization. 'Genetic algorithm' is a type of evolutionary algorithm inspired by natural selection [9], which is being widely used for architectural optimization. It iterates over different solutions strategically and asses the fitness of each solution comparing its performance to a set of criteria like energy efficiency, structural stability, etc. However, the algorithm depends upon the quantitative outcome of any design. As it is hard to quantify qualitative aspects of any design the optimization process can offer a solution which is well performing and less visually appealing. Selection through pure optimization is often referred as oversimplification of the design problem, as it neglects qualitative aspect of design options [10].The process does not incorporate the designer's preference or personal bias, which can be pretty impactful to generate a good design. Designers strive to achieve an optimal balance between their preferences and the performance outcomes, which often proves challenging with traditional optimization methods.

1.1.2 Time Limitations in Performance Simulations

Another limitation of computational optimization is the need to perform performance simulations for every data point. This simulation process can be time-consuming, especially for computationally expensive simulations. Consequently, the optimization of a large design space can take several hours or days to complete, leading to significant time and resource consumption.

Surrogate models have emerged as a solution to expedite the design optimization process [11]. these models can quickly predict a design's performance without requiring a complete simulation to be conducted each time. Different machine learning models are trained by a set of previously evaluated designs and their corresponding performance metrics. After the model is trained, it can approximate performance outcomes pretty fast [12] , which can speed up the optimization process. The accuracy of the prediction depends upon the number of training data and hyperparameters of the model used. However, if the design space is vast, a

large number of simulations must be performed to prepare the training set for the surrogate model to ensure accurate predictions.

These two drawbacks for the currently followed design framework: extensive time consumption to evaluate the design space and lack of user input regarding aesthetic preferences in design optimization have been addressed by many researchers over the past few years, which is discussed in the literature review section. This paper proposes a framework to overcome the mentioned drawbacks with the help of Artificial intelligence (AI) and machine learning (ML) models.

1.2 Thesis outline

There has been extensive use of AI in the AEC (architecture Engineering Construction) industry over the past few years [13]. For the improvement of computational powers, It became easier to implement complicated models for automation and optimization. The proposed framework consists of two connected subdivisions. In the first division, the authors tried reducing the time consumption for evaluating huge design space and in the second division, we explained how we can incorporate users' aesthetic demands in decision-making.

The whole framework was established to search for a suitable design solution for a four-storied office building located in Seattle. The annual energy consumption is considered as its performance outcome done by EnergyPlus [14]. Experimented design spaces encompass design options with different lengths, widths, window-to-wall ratios (WWR) and depths of the shading devices. The first task is to learn about the performance outcomes of all solutions quickly so that they can be used for further decision-making. The first division devoted to doing this comprises multiple consecutive steps. The last two steps of this division use dimensionality reduction models and clustering models respectively. An interactive online dashboard is designed where users can like or dislike any design options. A reinforcement learning-based recommender system is deployed to recommend new solutions to the designer that have lower annual energy consumption. This way the designers can explore the huge design space efficiently and strategically.

The paper details which models can considerably reduce computation time and highlights the enhanced capabilities of the proposed framework in effectively exploring vast design spaces. Additionally, it elucidates how an interactive decision-making process can assist designers in making informed and logical decisions.

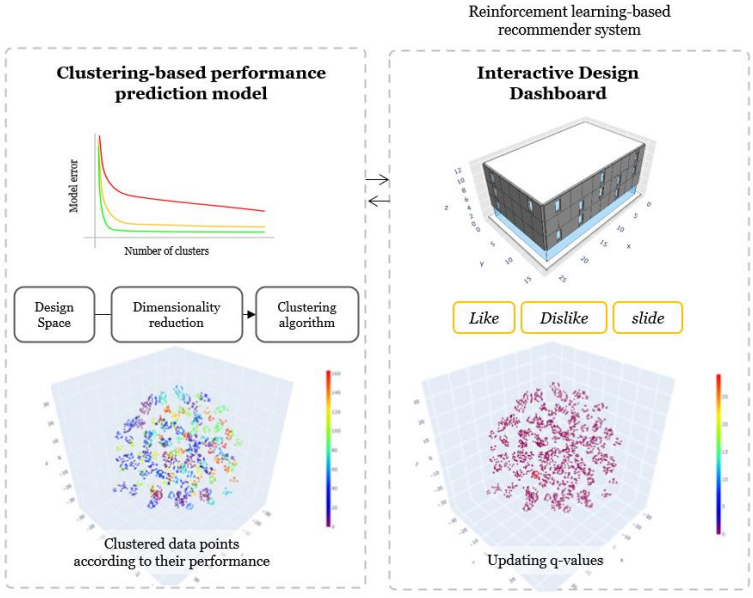


Figure 1-1 Thesis outline

Chapter 2: Literature Review

2.1 Design space exploration study

Navigating large design spaces poses a significant challenge for designers, but researchers have risen to the occasion with innovative ideas incorporating machine learning algorithms. In a notable study [15], researchers introduced gradient-based guidance as a versatile approach for exploring various parametric truss designs, enabling users to flexibly examine both favorable and unfavorable performance outcomes. Another study [16] tackled the complex task of understanding the relationship between design parameters and performance by leveraging canonical correlation analysis (CCA) [17]. This analytical method facilitated a deeper understanding of the optimal directions to pursue specific design criteria. Furthermore, to support the extensive exploration of design spaces with robust visualization and data-driven decision-making capabilities, a dedicated tool called 'Design Space Exploration (DSE)' [18] was developed.

To have an overall idea of the huge design space generated by Combinatorial Equilibrium Modelling (CEM) [19] clustered the design options by self-organizing maps (SOM) and tried getting insights by visualizing clusters with the help of a Uniform Manifold Approximation and Projection algorithm (UMAP) [20]. [10] tried a similar way but also attempted to include the designer's understanding of non-quantifiable aspects. The incorporation of the human designer's preferences in their proposed framework adds qualitative limitations on top of quantitative ones. The paper presents a computational design approach that benefits from human interaction and machine intelligence collaboration. The framework includes five steps, one of which is 'selection' where the designers need to filter out non-prefer design options, resulting in smaller design space with preferred options.

While the five steps design framework was utilized by [10] to design a structurally sound stadium, the same process is used by [21] to revisit an existing structure. Inspired by the existing structure they generated more options with the help of CEM, then clustered by SOM for a low dimensional representation of the whole design space. The arrangement of design options in a 2-dimensional grid was extremely helpful in negotiating between multiple judging criteria like, structural, daylight and radiance analysis.

There can be two ways to incorporate user preferences in the multi-object optimization process: first, the users can alter the weightings between well-known objectives as part of a multi-objective search, and the second user can choose which designs to consider for each generation[22]. [23] Have utilized a genetic algorithm-based optimization process to generate a single-family residence, which allows users to filter design options after each evolution. The process used named non-dominated sorting genetic algorithm (NSGA) [24] allowed the users to make more personalized optimization, that helps incorporate their preferences in decision-making. [25] Has created a grasshopper plug-in named 'Biomorpher' based on a similar approach, which groups similar types of design options and displays them in an interactive dashboard to assist the designers in filtering options that are not preferred. Involved interactive evolutionary computation (IEC) [26] optimization algorithm process performs guided optimization to achieve a design solution that satisfies both performance and aesthetic preferences.

To include the designer's qualitative measure [27] followed a framework that finds out numerical measurements for aesthetical preferences by a survey. The survey displays different options in an interactive dashboard made by grasshopper plug-in Human UI [28], where participants vote for what they like and based on that incorporated algorithm decide the quantitative measurement of aesthetics for that design option. Both the performance and aesthetic measurements are then utilized while running multi-object optimization. The idea of a design dashboard is not new. While it is fascinating to hide all complicated scripts and only get the benefits of design computation it also ensures broader participation in decision making. [29] In the final step of their thesis has produced a guided dashboard to design a Highrise with the help of the Human UI. The developer of Human UI leveraged the idea of the dashboard in a new platform named 'Hypar'[30], which enables designers to design in an interactive way and generate multiple options within a very few amount of time.

2.2 Recommender system study

In the quest for effective techniques to explore design space, researchers are actively seeking collaborations with diverse professionals to harness their expertise in built environment studies. These collaborations hold the potential to unlock the power of design computation and machine learning. While many studies have proposed effective pathways utilizing multi-objective optimization processes, there are still unexplored avenues to be pursued, and one such avenue is the utilization of recommender systems. A recommender system can be described as an information filtering system that suggests relevant items, such as products, movies, or videos, to its users [31]. Incorporating a recommender system in suggesting design options from a design space offers a significant advantage: it relieves users from the burden of comprehending the complexities of the entire design space. Users can explore stress-free while still obtaining options that align with their intentions.

Amidst the multitude of recommender systems currently employed on various websites, reinforcement learning-based recommender systems[32]. introduce an innovative paradigm that brings forth numerous benefits. Unlike commonly used collaborative or content-based filtering approaches [33] which heavily rely on extensive user information to deliver effective recommendations, reinforcement learning-based systems offer a promising solution. These traditional approaches often require users to provide personal details such as age, gender, or location, or endure a prolonged period for the system to grasp their preferences—an issue known as the 'cold start' problem in the realm of recommender systems [34].

In contrast, reinforcement learning-based recommender systems present an alternative approach to address this challenge (see 3.2). By leveraging the principles of reinforcement learning, these systems have the ability to rapidly learn about users' preferences through interactive experiences. They can generate accurate recommendations within a significantly shorter timeframe As we want our AI to be extremely unbiased of the designer’s sex, age, location or profession, this method can help us make unbiased decisions that are only a reflection of its user. It is very simple but effective. This simplicity and effectiveness make reinforcement learning-based recommender systems increasingly popular. It can make ethical choices for its users and focus more on making better recommendations neglecting any commercial intuition. The team firmly believes that implementing such a remarkable tool for exploring parametric design space can usher in a new era of participatory design and playful exploration.

Chapter 3: Methodology

As previously mentioned, the thesis is divided into two sections. The first section examines the design framework to be used in the second section for predicting performance outcomes with speed and accuracy. The investigation takes place across three different design spaces in succession, with the findings from one space informing the next. The design spaces increase in size with each iteration. The first design space is relatively small, consisting of 2500 datasets with 48 parameters, while the second design space has 3226 datasets with 50 parameters, the third design space has 6000 datasets with 98 parameters and the fourth design space has 5470 datasets with 43 parameters.

3.1 Section 01: Clustering-based performance prediction

3.1.1 Design Space 00

The first investigated design space mentioned as 'Design space 00' consists of 2500 design options. All the design options have the same size of 25m long in the north-south direction and 15m wide in the east-west direction. The type of building is a four-storied 'mid-rise apartment' [35] where the ground floor is considered a retail store. The ground is completely transparent with the highest window-to-wall ratio(WWR) while it's different for different options on the upper floors. Floor to floor height is 3m.

The apartment portion of the north and south façade is divided into a grid of 5x3, so each floor is having a maximum of 5 windows on one side. The east and west façade of the apartment portion is divided into 3x3, having a maximum of 3 separate windows on each floor on each side. For the whole building, we can have 48 (5+3+5+3) window segments in total. Whether there will be a window in the window segment or not depends upon the parameters of the design option.

A pandas DataFrame [36] (see Table 3-3) consisting of 2500 rows and 48 columns is created, where each row represents a unique design option. The values in each column are either zero or one. If a value is zero, it indicates that the corresponding window segment does not contain any windows. If it is one, then a window with a height of 2m and a width of 1m is present in that segment. In design space 00, the window size is constant for all design options, but the windows are located in different segments. The first 15 columns of the DataFrame represent the window positions on the north facade, where the first column corresponds to the first window position on the first floor at the north facade, and the 6th column (column index=5) corresponds to the first window position on the second floor at the north facade. Window positions for the west, south, and east facades are denoted by column indexes 15-23, 24-38, and 39-47, respectively (see Figure 3-1)

For the north and south façades only 10 random combinations are picked up for each side. In total 20 random options are picked up which are completely different from each other. As for the east and west façade number of possible combinations would be smaller, 5 random combinations are picked up for each side. Then these four sets of combinations are cross-multiplied to create a whole set of building parameters. They generated a set of 2500 possible design options.

How much energy (kWh) would be consumed by the users annually due to heating, cooling and lighting is simulated by EnergyPlus (version 22-2-0). The whole process was conducted in Python (version 3.9), and the authors utilized an external Python library called 'Compas eplus' [37] to communicate with EnergyPlus written by Tomás Méndez Echenagucia. Python scripts were written to produce geometries that Compas can read, which were then converted into EnergyPlus readable IDL format to generate outcomes. Table 3-1 provides the properties that were fixed for the building during the simulation. To be noted, all simulations for this paper followed the same mentioned simulation properties.

Simulation Properties	**values**
Location	Seattle
Building type	DOE midrise apartment
Wall	Typical Insulated Steel Framed Exterior Wall-R16
Window	Generic Double Pane
Floor	Generic Interior Floor
Roof	Generic Roof
Simulation outcome unit	Kilowatt-hour (kWh) / year

Table 3-1 Simulation properties followed for all studies.

Building properties (Design Space 00)	**values**
Building length	25m (north-south)
Building width	15m (east-west)
Window segments	5m x 5m
Window sizes	2m height x 1m width
Ground floor WWR	0.6
Number of floors	4
Floor-to-floor height	3m
Surrounding context	Not considered
DataFrame properties	
Amount of data points	2500
Number of parameters	48

Table 3-2 Building Properties for Design Space 00.

	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**
1182	0	1	1	1	1	0	1	1	1	0	1	1	0	1	1	1	0	0	1
239	0	0	0	0	0	1	1	1	0	1	0	0	1	1	0	0	1	0	0
999	1	1	0	0	1	0	1	1	0	0	1	1	0	1	0	0	1	0	0
875	1	1	0	0	1	0	1	1	0	0	1	1	0	1	0	1	0	0	1
1601	1	0	0	1	0	0	1	1	1	0	0	0	1	1	1	1	0	0	1

Table 3-3 Data frame sample for Design Space 00.

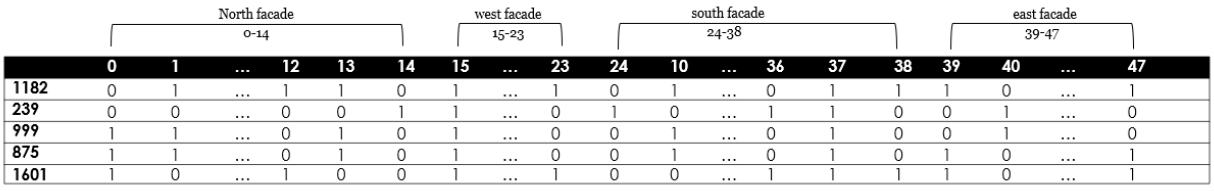


Figure 3-1 Data structure of Design Space 00

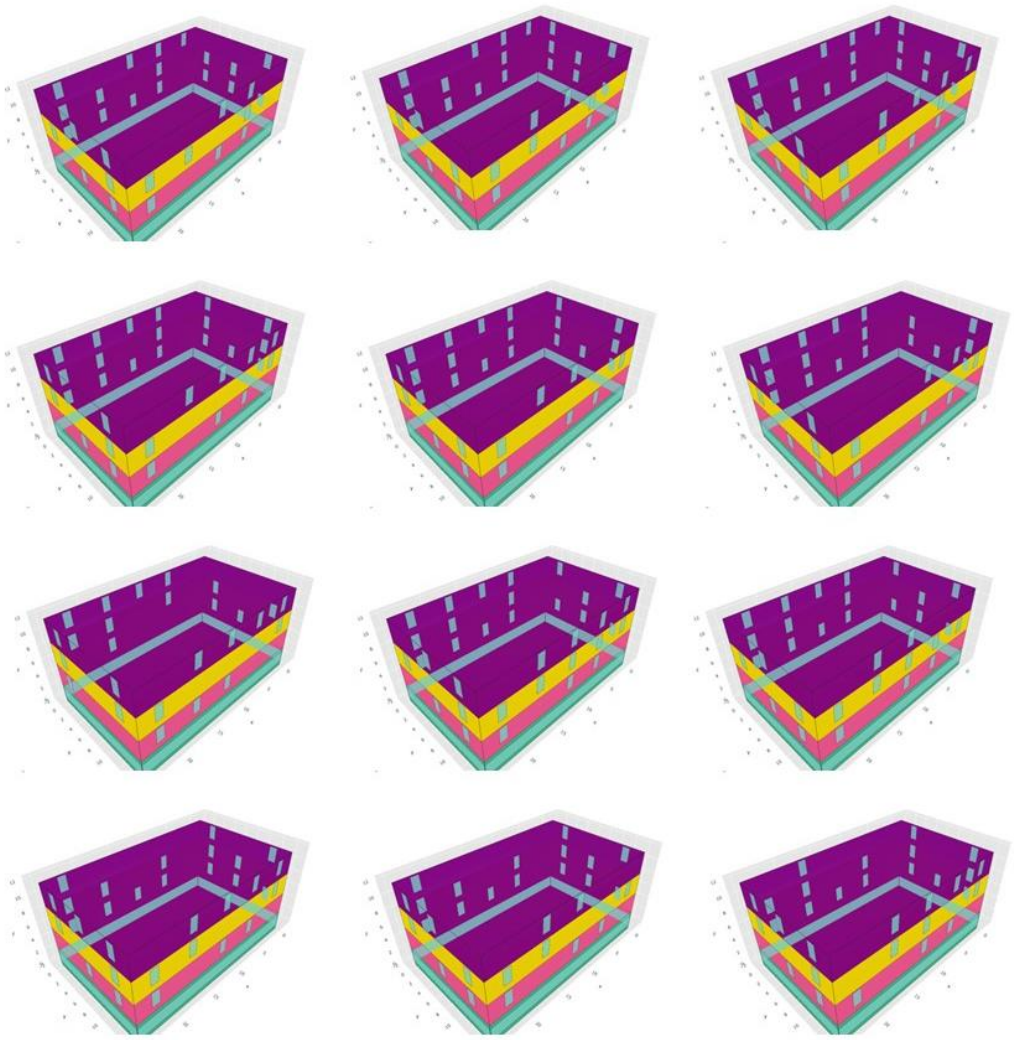


Figure 3-2 Randomly selected 12 design options from Design Space 00

3.1.1.1 Clustering Frameworks for design space 00

If we group the design space into clusters and simulate only the data points closest to the cluster centroids, we can approximate the performance outcomes of the remaining data points within the same cluster. This allows us to reduce the number of simulations needed significantly. However, it's important to note that as the number of clusters decreases, the number of required simulations decreases but the prediction error may increase. In this paper, we investigate the relationship between the number of clusters and the framework error, aiming to determine the optimal balance between the number of simulations and prediction accuracy. Our goal is to generate well-performing clusters that minimize the error in our predictions while considering the trade-off between accuracy and computational efficiency.

While there are many unsupervised clustering algorithms available in Python's Sklearn library [38], clustered output also depends upon the given dataset for clustering. In our current dataset, we have 48 features for one design, which means we have 48-dimensional data points. When the data points have very high dimensions and are comparatively low in number, it can occur the 'curse of dimensionality' [39], which imposes further challenges and issues like increased computational complexity, decreased model performance and data sparsity. To avoid such issues, it is important to reduce the dimensions before using the dataset for clustering. This adds another step in the considered framework. Dimensionality reduction

is a method used in machine learning and data analysis to minimize the number of features or variables in a dataset while preserving as much relevant information as feasible [40].

Python's Sklearn library provides various algorithms for dimensionality reduction. In addition to the dimensionality reduction models available in Sklearn, the authors of this study also proposed other approaches for dimensionality reduction that are specifically applicable to similar types of datasets. Regardless of how the building is designed to conform to the framework, all building information needs to be converted into a uniform data frame format. The manner in which building information is extracted as features will impact the outcomes of subsequent stages, as they are correlated with this data representation.

To understand how well clustering can be done after reducing the dimensions, each dimensionality reduction model converted the 48-dimensional data points into only 3-dimensional data points. The given outcome is then plotted and visualized with the help of Python's Plotly library [41]. Generated 3D visualization helped the team explain in advance why the clustering methods will not perform well or not. Data points with reduced features are used for clustering instead of the actual one. A comparison was also made with the clustering done with the actual dataset to find out the effects of dimensionality reduction.

The team investigated both linear and non-linear types of dimensionality reduction algorithms [42] before clustering. The following image (see Figure 3-3) shows the investigated workflows for design space 00.

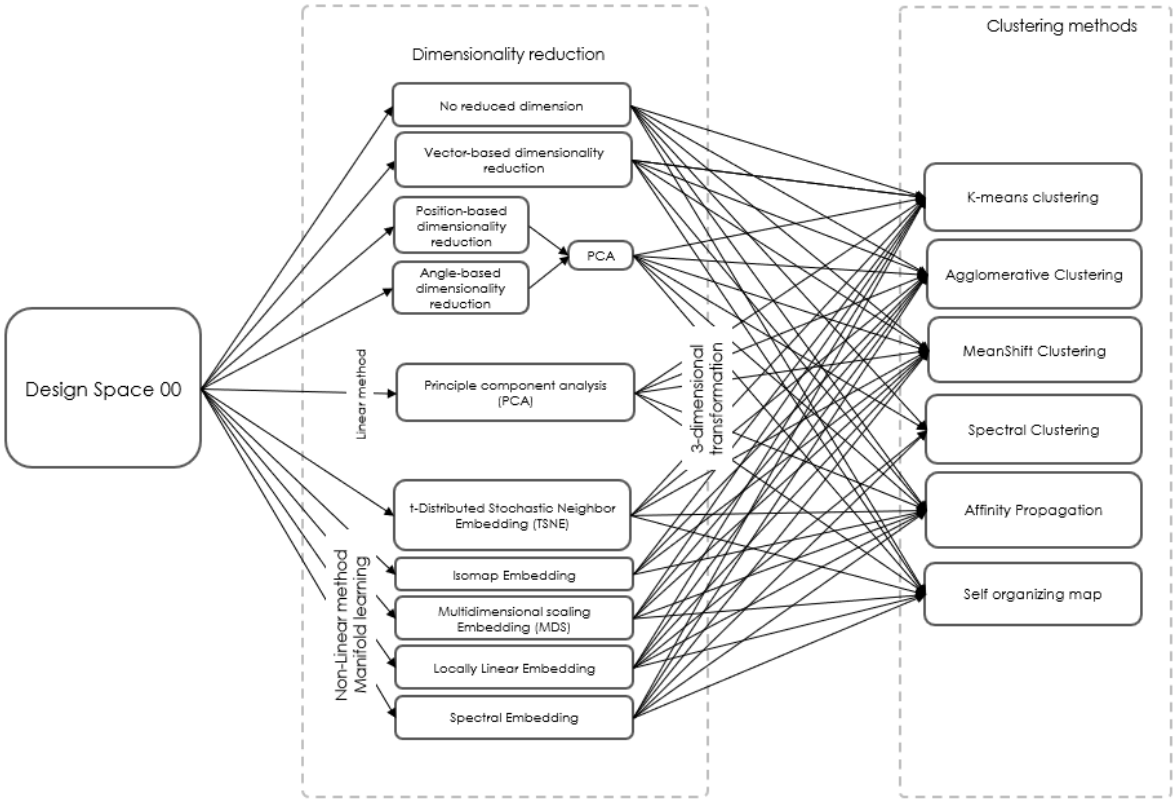


Figure 3-3 Clustering frameworks for Design Space 00.

The authors first assessed the quality of the clustering for the design space before evaluating the performance prediction errors. This was done by creating a pixelated image of multiple clusters using each clustering method. Each pixel in the image corresponds to a window segment, and all window segments from the four different facades were combined to visualize the entire design in a 2D graph. The color of each pixel represents the frequency of windows appearing in that particular window segment, with darker colors

indicating higher frequencies and lighter colors indicating lower frequencies. A lighter color suggests that there is less likelihood of having a window in that segment if a design option is randomly selected from the corresponding cluster.

Pixelated representations of multiple clusters are placed on top of one another to get an overview of the used clustering algorithm. If the generated image is noisy, that refers to a higher variation on a single cluster. If the image is less noisy, we can conclude that data points are organized with fewer differences in them. However, if we only consider fewer clusters to be done by the algorithms, the generated image will be noisier. For why, the number of clusters to be provided by the algorithm is kept constant for most of the frameworks to make an apple-to-apple comparison between different methods.

After figuring out which set of dimensionality reduction methods and clustering algorithms can generate useful clusters, the team proceeded to evaluate their performance in predicting annual energy consumption, referred to as "model error" throughout the paper. To find out model error, only the performance outcome of the data point closest to the centroid of the cluster is extracted. If we consider that is the performance outcome for the rest of the data points in the cluster, how much mean absolute error (MAE) [43] we can have, is calculated for each data point. What percentage of the actual value is the calculated MAE is considered as the final MAE. The MAE of all clusters was summed and divided by the number of clusters to calculate the average MAE for the entire clustering method. This average MAE is considered the performance metric for each framework.

To learn the simulation outcome of all design options, running energy plus simulation for all would be required. However, frameworks can approximate simulation outcomes by strategically selecting a moderate amount of simulations from the design space. Understanding which number of clusters can generate lower model error ' can aid designers in determining which framework to use. The more clusters there are the more performance simulations that must be run, resulting in a more time-consuming process. The framework that produces lower model error ' while generating fewer clusters can be deemed effective in reducing simulation time.

A Pareto distribution [44] chart is created to determine the framework that would work best in reducing simulation time. The x-axis represents the number of clusters generated by the framework, while the y-axis represents the corresponding 'model error'. A framework that achieves a value closer to the centre can be considered more efficient in reducing simulation time. This Pareto distribution can be named as ‘model error vs cluster count Pareto distribution’. In total Pareto distribution of 3 frameworks were compared with each other to find out the best one.

3.1.1.1.1 Dimensionality reduction methods for design space 00

The design space 00 framework consists of two main steps. The first step involves reducing the dimensionality of the 48-dimensional DataFrame. In addition to using conventional dimensionality reduction methods for data mining, the team explored geometry-based dimensionality reduction processes specifically tailored to the context at hand. These novel methods are purpose-oriented and offer potential for various other applications. They provide innovative abstractions of building information while preserving crucial details related to building energy consumption. Experimented dimensionality reduction methods are listed below with details:

3.1.1.1.1.1 Vector-based dimensionality reduction

This Vector-based dimensionality reduction process is utilized multiple times in further explorations in this paper. Here each façade is expressed as a vector. At first single vectors are extracted for each wind position. If we consider these vectors as lines, the starting point of these lines is the centroid of the building volume, and the endpoint is the centroid of each window position. So the direction of all vectors is pointing outward from the centroid. The north and the south façade would have 15 vectors for each and the east and the west façade would have 9 vectors for each. As the facades for the ground floor are the same as for all design options, they are not taken into consideration. Each vector (their coordinates) is then multiplied by the

assigned WWR (window-to-wall ratio). If there is no window in the window position the value of the vector would be zero. All individual vectors of a single façade are combined to get one vector for each façade. For design space 00, we have only four facades, so each design option will be represented by four vectors. This process keeps the information of in which direction we have more aperture and after investigating their simulation outcome we can understand if this information is helpful in grouping similar design options.

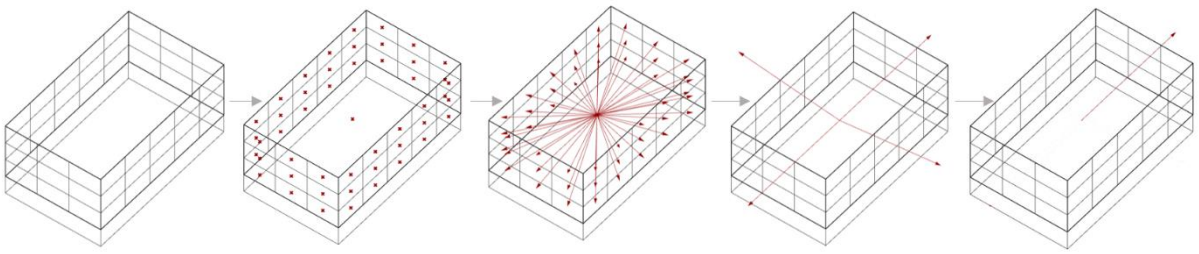


Figure 3-4 Vector based data conversion process.

For the frameworks of design space 00, these four vectors are combined to turn them into one single vector (see Figure 3-4). That final vector would have 3 coordinates. Following the way we are converting a 48-dimensional DataFrame into 3 so that we can plot those data points into three-dimensional space (see Table 3-4).

	0	**1**	**2**
0	48.28286	22.97535	19.78686
1	42.29711	20.88835	15.95505
2	43.48012	20.60049	16.80255
3	43.70315	22.4515	18.87851
4	46.98316	22.59763	20.35446

Table 3-4 Converted data frame by vector-based dimensionality reduction method

3.1.1.1.1.2 Position-based dimensionality reduction

In this process, the facades are subdivided into one or two divisions, indicating the presence and number of windows in each subdivision. The north and south facades are divided into two divisions. One division consists of 2 window positions on each floor, totaling 6 window positions, while the other division consists of 3 window positions on each floor, totaling 9 window positions. On the other hand, the east and west facades are not subdivided, resulting in 9 window positions for each facade. The WWRs assigned to all window positions of a single division are summed up to get a single value for each subdivision. Overall, there are a total of 6 subdivisions (see Figure 3-5), which effectively reduces the dimensionality of the original 48-dimensional DataFrame to a 6-dimensional DataFrame (see Table 3-5).

	0	**1**	**2**	**3**	**4**	**5**
0	6	0	3	6	6	0
1	5	3	3	3	6	9
2	3	3	0	6	4	6
3	0	0	3	6	6	6
4	3	3	6	9	6	0

Table 3-5 Converted data frame by Position-based dimensionality reduction method.

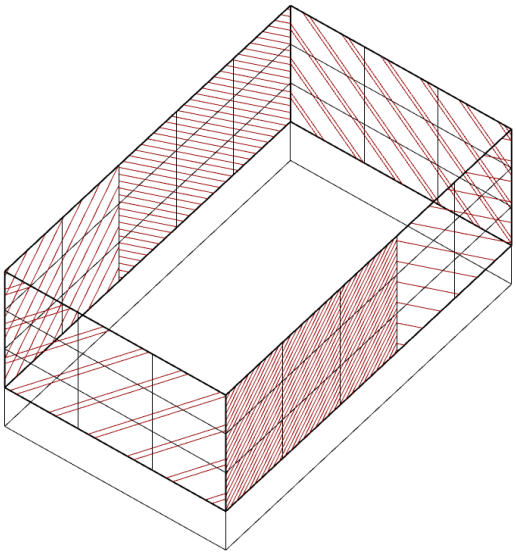


Figure 3-5 Segmented facades for position-based dimensionality reduction.

This subdivision process provides valuable information about the proximity of windows in the north or south facades to those in the east or west facades. To enable the visualization of the data points in a three-dimensional space, Principal Component Analysis (PCA) is performed on the 6-dimensional DataFrame. PCA transforms the data into a new DataFrame with three dimensions.

3.1.1.1.1.3 Angle-based dimensionality reduction

In this process, each individual facade is transformed into a single vector. Instead of considering the coordinates of these vectors, the focus is on measuring the angle of each vector relative to a reference line. The reference line is also represented as a vector, originating from the centroid of the building volume and pointing towards the north direction (see Figure 3-6). For design space 00, there are four vectors corresponding to the four facades, resulting in four angles in total. By adopting this approach, the initial 48-dimensional DataFrame is reduced to a 4-dimensional representation (see Table 3-6). After applying PCA, the data is further condensed into a 3-dimensional format for visualization purposes. This process enables the algorithm to discern the predominant aperture direction for each facade.

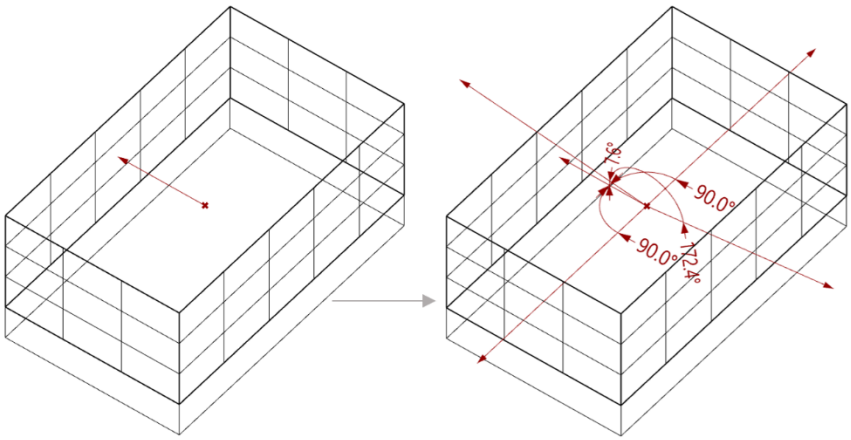


Figure 3-6 Angle-based data conversion process.

	0	1	2	3
0	0.001526	0.003115	0.001176	0.003626
1	0.001517	0.002445	0.001297	0.00319
2	0.001519	0.002681	0.001175	0.003456
3	0.001503	0.002445	0.001195	0.003331

Table 3-6 Converted data frame by Angle-based dimensionality reduction.

The team followed both linear and non-linear dimensionality reduction methods traditionally used in data processing. Linear dimensionality reduction methods aim to find a lower-dimensional representation of the data by applying linear transformations, while non-linear dimensionality reduction methods are designed to capture non-linear relationships and complex structures in the data. As we didn't have much precedent study on which type would work effectively for this research the team investigated both types.

3.1.1.1.1.4 Principle component analysis (PCA)

Principle component analysis is a widely used linear dimensionality reduction method, which identifies the principal components of the data, which are new variables that are linear combinations of the original features [45]. The process captures the most variation in the data, and each succeeding component, which is orthogonal (uncorrelated) to the preceding components, catches the highest remaining variance. In this study, the team utilized the PCA model from the scikit-learn library [46] (Sklearn) in Python to perform the analysis. Hyperparameters of this model are kept default.

3.1.1.1.1.5 t-Distributed Stochastic Neighbor Embedding (TSNE)

t-SNE (t-Distributed Stochastic Neighbor Embedding) is a non-linear dimensionality reduction process widely used in data visualizations. First, it creates a probability distribution, which ensures that similar data points have higher probability and dissimilar ones have lower. When the data points are projected into lower dimensional space the same probability is used to cluster similar data points and create more divergence [47] . TSNE model from the scikit-learn library [48] (Sklearn) in Python is used. Hyperparameters of this model are kept default only for design space 00, while in other design spaces, they are altered.

3.1.1.1.1.6 Isomap Embedding

The procedure starts with creating a neighborhood graph based on the pairwise distances between data points. Following that, it applies graph-based algorithms to calculate the geodesic distances between each pair of data points, capturing the data's inherent geometric structure. Lastly, it uses these geodesic distances to embed the data in a lower dimension [49]. As it is a nonlinear dimensionality reduction it can provide a more accurate representation of data points than any linear dimensionality reduction process. Isomap model from the scikit-learn's manifold library [50] in Python is used. Hyperparameters of this model are kept default.

3.1.1.1.1.7 Multidimensional scaling Embedding (MDS)

For our investigation, we selected Euclidean distance as the distance matric. After the distance matrix is fixed the process computes a dissimilarity matrix, which contains the pairwise distance between all pair of data points. When the data points are projected into a lower dimensional space, the process tries to reproduce the initial dissimilarity matrix through an iterative process [51]. MDS model from the scikit-learn's manifold library [52] in Python is used. Hyperparameters of this model are kept default.

3.1.1.1.1.8 Locally Linear Embedding

This is a non-linear dimensionality reduction method. Based on the distance metric the process at first defines its k-nearest neighbors. After it projects the data points into a lower dimension it tries to reconstruct the local relationship. The resulting embedding preserves the relationship between neighboring data points, which helps to capture the non-linear behavior of data points [53]. 'LocallyLinearEmbedding' model from the scikit-learn's manifold library [54] in Python is used, where hyperparameters are left at their default values.

3.1.1.1.1.9 Spectral Embedding

Spectral Embedding leverages spectral features to expose the underlying structure of the data. At first, it constructs an affinity matrix that computes the pairwise similarities or distances between data points. We have selected 'nearest neighbors' as our similarity measures. The affinity matrix is used to compute graph Laplacian, which is the measure of how different data points relate to each other. 'SpectralEmbedding' model from the scikit-learn's manifold library [55] in Python is used. This model's hyperparameters are set to default.

3.1.1.1.1.10 No dimensionality reduction:

The team also investigated frameworks that used the initial data frame directly for clustering without applying any dimensionality reduction method, to figure out the benefits of reducing dimensions before clustering.

3.1.1.1.2 Clustering methods for design space 00:

In the second step of the framework, after reducing the dimensionality of the DataFrame, clustering techniques are applied to identify meaningful patterns and groupings within the data. This step aims to partition the design space into distinct clusters, where design options within the same cluster exhibit similar characteristics or share common features. To explore the design space in this paper, the team examines various combinations of dimensionality reduction and clustering methods. The team investigated 10 different dimensionality reduction techniques, along with the framework without any dimensionality reduction. Additionally, they explore 6 different clustering methods, which are mentioned below. Considering the different combinations of dimensionality reduction and clustering methods, there are approximately 60 possible frameworks discussed in the paper. The clustering methods used in the experiments are derived from scikit-learn's cluster library [56], with their default parameter values set by the authors. Additionally, the team explored other dimensionality reduction and clustering methods such as DBSCAN (Density-based spatial clustering of applications with noise) [57] and feature agglomeration [58]. However, those methods did not yield meaningful results that could be discussed, and therefore they are not mentioned in the paper.

3.1.1.1.2.1 K-means clustering

K-means clustering is a popular unsupervised clustering approach in architectural research [59]. The procedure begins with the deployment of a certain number of centroids at random. Their location is updated by computing the mean of their allocated data points until convergence [60],[61]. K-means is a straightforward and efficient clustering technique that frequently yields insightful conclusions. However, the number of clusters we desire will affect how successful it is.

3.1.1.1.2.2 Agglomerative Clustering

Agglomerative clustering is a bottom-up method in which each data point is first treated as a separate cluster. The tiny clusters are then blended into a larger cluster based on the distance matrix. The method by which distances are estimated is referred to as linkage criteria [62], [63] . We have used 'ward's linkage' for all our investigations. Clusters are combined in the procedure until the given quantity is reached, and a dendrogram is created. When dealing with high-dimensional data, the procedure can often take a very long time.

3.1.1.1.2.3 MeanShift Clustering

It's a density-based clustering method. The process begins with estimating the density of the data points, using the kernel density estimation technique [64]. It's a function that defines the shape and reach of density estimation. scikit-learn's 'MeanShift' cluster model [61] is used which required the bandwidth value instead of the number of clusters to be achieved. Bandwidth refers to the radius or distance around each data point within which the algorithm searches for more. So, bandwidth value for any feature space highly depends upon how much they are spread out. This clustering method generated promising outcomes, as our data points are already densified by different non-linear dimensionality reduction processes. The task of the clustering algorithm is to identify each cluster.

3.1.1.1.2.4 Spectral Clustering

All the steps of this process are the same as the spectral embedding, except it provides us with the prediction of which data points belong to which cluster. Spectral clustering is good at recognizing clusters even when their forms are complicated, or they are not linearly separable [65], [66].

3.1.1.1.2.5 Affinity Propagation

This method works by iteratively propagating the willingness of one data point to be an exemplar for another. Based on the similarity matrix of the input data, in each iteration, the process revises two matrices: the responsibility matrix and the availability matrix. The responsibility matrix defines how compatible a data point is to be an exemplar for another data point and availability matrix defines well-suited data point to choose another data point as its exemplar. This updates are continued until convergence and after that the algorithm identifies which are the most representative data points. To have clusters, the non-exemplar data points are assigned to their nearest exemplar [67], [68].

3.1.1.1.2.6 Self-organizing map (SOM)

Self-organizing map is a neural network-based unsupervised clustering method. SOM's core tenet is neighborhood preservation. In the grid of neurons, neurons that are near to one another are more likely to have weight vectors that are similar, suggesting that they react to input patterns similarly [69], [70] . The SOM learns to arrange the neurons in such a way that comparable input patterns activate nearby neurons by repeatedly modifying the weights depending on the input data.

3.1.2 Design Space 01:

Design space 01, the second design space investigated, consists of 3226 design options. These options vary in both length and width, ranging from 10m to 50m. Unlike the previous design space, which had binary window segments, each window segment in this design space has a unique value expressed as a decimal. These values determine the window-to-wall ratio (WWR) and size of each window. A value of zero signifies no window in that segment, while non-zero values range between 0.3 and 0.7 (see Table 3-7). All design options are four-storied while the ground floor configuration remains the same as Design Space 00.

Similar to the previous design space, the north and south facades of the building are divided into a grid of 5x3, while the east and west facades are divided into a grid of 3x3. Regardless of the building's size, each floor of the building will have five window positions for the north and south facades and three window positions for the east and west facades. The DataFrame for this design space is also similar to the previous one but has two additional columns that define the length and width of each design option (see Figure 3-8). It contains 50 columns and 3226 rows. To prevent bias, the length and width columns are reduced to two decimal places and multiplied by 100 before being used in the models. EnergyPlus simulation properties remain unchanged from before.

Building properties (Design Space 01)	**values**
Building length	Range (10m-50m) (north-south)
Building width	Range (10m-50m) (east-west)
Window segments (North-South)	5 x3
Window segments (East-west)	3 x 3
Window sizes	Range(0m-5m)
Ground floor WWR	0.6
Number of floors	4
Floor-to-floor height	3m
Surrounding context	Not considered
DataFrame properties	
Amount of data points	3225
Number of parameters	50

Table 3-7 Building properties for Design Space 01.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
82397	0.7	0.7	0.4	0	0.4	0	0	0.7	0.7	0.3	0	0	0.3	0.3	0	0	0.7	0.3
1702	0.6	0.6	0	0	0	0.5	0.5	0	0	0.5	0.7	0	0.6	0	0.7	0.6	0.6	0
52330	0	0	0.7	0.6	0.7	0	0.3	0	0.5	0	0.6	0.5	0	0	0.4	0.5	0	0.3
87482	0.6	0.4	0	0	0.5	0	0	0.6	0	0.7	0	0	0	0.4	0	0	0	0

Table 3-8 Data frame sample for Design Space 01.

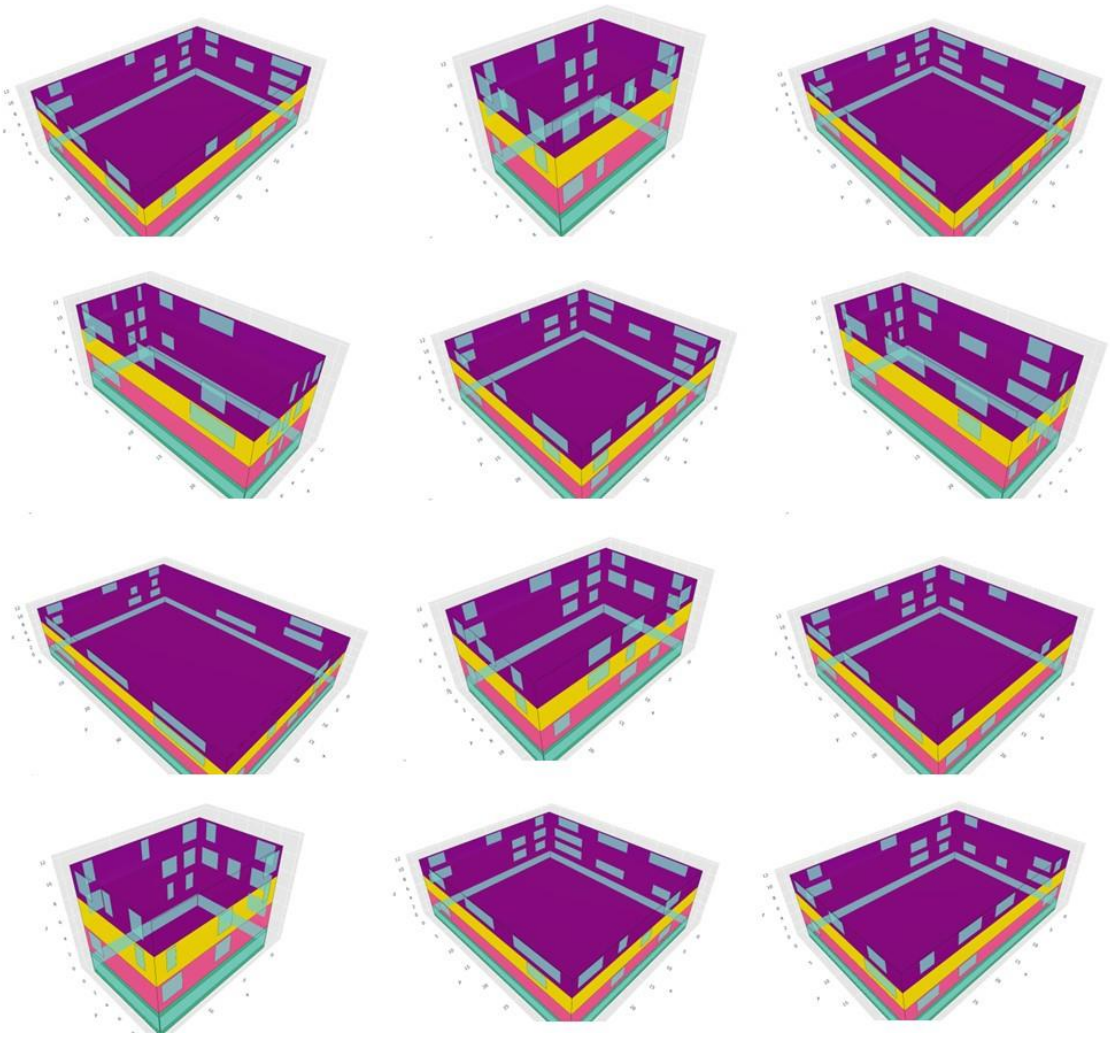


Figure 3-7 Randomly selected 12 Design options from Design Space 01.

	North facade 0-14						west facade 15-23			south facade 24-38						east facade 39-47				length	width
	0	**1**	**...**	**12**	**13**	**14**	**15**	**...**	**23**	**24**	**10**	**...**	**36**	**37**	**38**	**39**	**...**	**47**	**48**	**49**	
1182	0.6	0.7	...	0.4	0	0	0	...	0.7	0.5	0.3	...	0.4	0	0	0	...	0.6	0.35	0.45	
239	0	0	...	0.3	0.3	0.7	0	...	0.3	0.7	0	...	0	0.3	0.4	0.3	...	0.7	0.5	0.35	
999	0	0	...	0.6	0.3	0	0.4	...	0	0.6	0.3	...	0.3	0	0.3	0	...	0	0.15	0.1	
875	0.5	0.6	...	0.6	0.6	0.7	0.3	...	0.7	0.3	0.3	...	0	0.5	0.5	0.7	...	0	0.35	0.45	
1601	0	0	...	0	0.7	0.6	0.3	...	0.4	0.5	0.4	...	0	0	0.4	0.4	...	0	0.35	0.25	

Figure 3-8 Data structure of Design Space 01.

3.1.2.1 Clustering Frameworks for design space 01:

To streamline the investigation of the next design space, the team applied the insights gained from analyzing the previous design space (see 4.1). After evaluating various frameworks for design space 00, the team selected three that generated the most accurate models with distinct clusters. All three frameworks utilized mean-shift clustering, but with different dimensionality reduction methods, namely t-distributed stochastic neighbor embedding (t-SNE), isomap embedding, and spectral embedding.

Surprisingly, none of the finalized frameworks performed effectively for design space 01 (see 4.2.1), as each created only one to three distinguishable clusters of data points. As mean-shift clustering is a density-based clustering method, it can only generate effective clusters when the data points are grouped logically. The reason for the failure of the frameworks was the characteristics of the DataFrame. It's worth noting that if researchers want to apply the most effective framework identified in this paper to a different design space, they must transform all the information into a DataFrame similar to ours to ensure similar outcomes.

In order to create effective clusters, it is crucial to understand how different parameters affect the performance outcome and incorporate this information into the DataFrame. To gain insights into which parameters have the greatest impact on performance, a smaller design space was created and analyzed. This design space, called "Test design space 00," includes 401 two-story design options with varying lengths and widths between 10m and 50m, each featuring only one window on each side. The WWR ratio, which defines the size of the window, ranges from 0.1 to 0.9 across the different options, while the ground floor configuration remains the same as Design Space 00 (see Table 3-9).

Building properties (Test Design Space 00)	**values**
Building length	Range (10m-50m) (north-south)
Building width	Range (10m-50m) (east-west)
WWR (Window sizes)	Range(0.1 – 0.9)
Ground floor WWR	0.6
Number of floors	2
Floor-to-floor height	3m
Surrounding context	Not considered
DataFrame properties	
Amount of data points	401
Number of parameters	6

Table 3-9 Building properties of Test Design Space 00.

The DataFrame for Test Design Space 00 includes 6 columns and 401 rows. The first four columns specify the WWR ratio for the north, west, south, and east facades, respectively, while the last two columns indicate the length and width of each design option (see Table 3-10). Energy plus simulations were performed using similar properties to determine the annual energy consumption per square meter for each design option. To understand the impact of different parameters on performance, a linear regression model was run with the data we generated. The resulting weights from the model identify which columns have the most significant influence on performance.

	0	1	2	3	4	5
22295	0.1	0.9	0.1	0.9	0.1	0.1
27113	0.2	0.1	0.7	0.2	0.25	0.25
153943	0.8	0.2	0.1	0.7	0.35	0.5
6996	0.1	0.3	0.2	0.8	0.45	0.3
3376	0.1	0.2	0.1	0.6	0.5	0.25

Table 3-10 Data frame sample for Test Design Space 00.

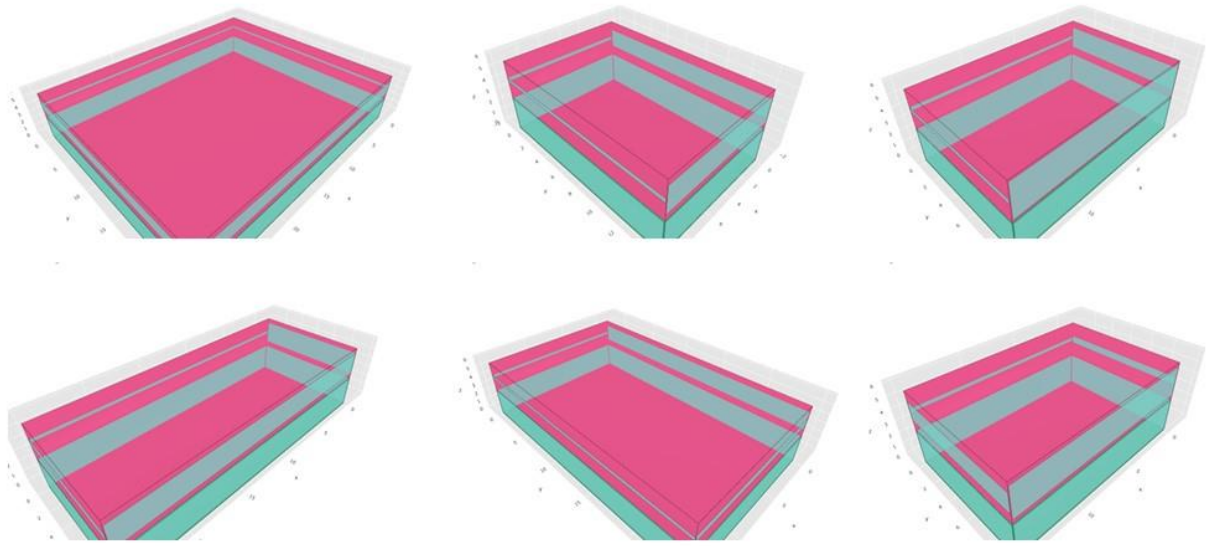


Figure 3-9 Randomly selected 6 design options from Test Design Space 00.

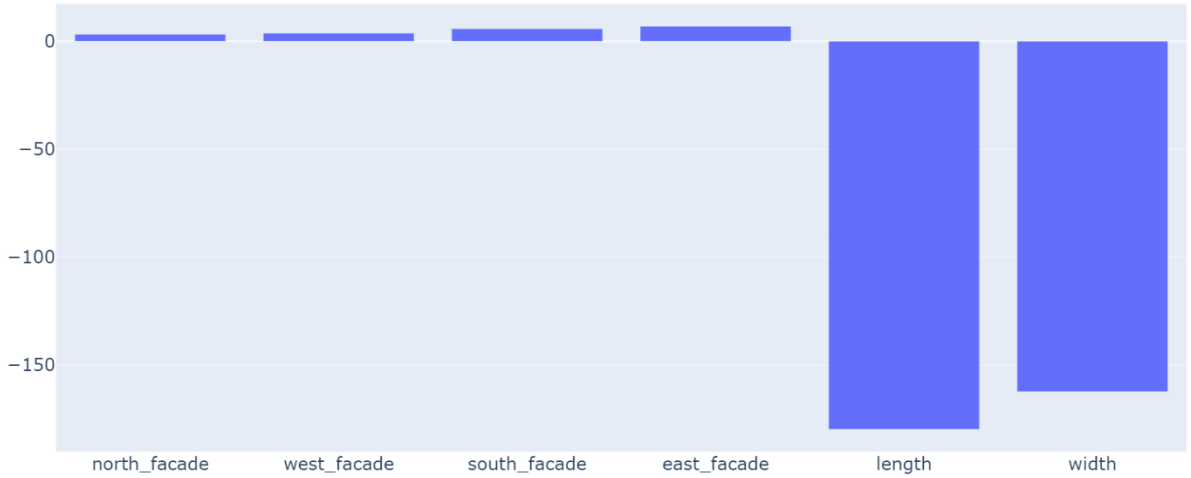


Figure 3-10 Weights for different parameters of Test Design Space 00.

The figure showing the weights (see Figure 3-10) clearly explains different properties of the design have different impacts on performance outcomes. It is proven that each column or parameter should be treated in a way that the dimensionality reduction method can promote density to design options having similar

performance outcomes. To do so each column should be weighted effectively. The team investigated many ways of weighting design parameters before running dimensionality reduction and clustering algorithms. 3D scattered plots explained how the added weights helped the data points grouped together and generated meaningful clusters.

Three dimensionality reduction methods are mentioned before and mean-shift clustering is used after adding weights. 'model error vs cluster count Pareto distribution' is generated for all different frameworks. In total 21 frameworks were compared with each other to find out the best one (see Figure 3-11).

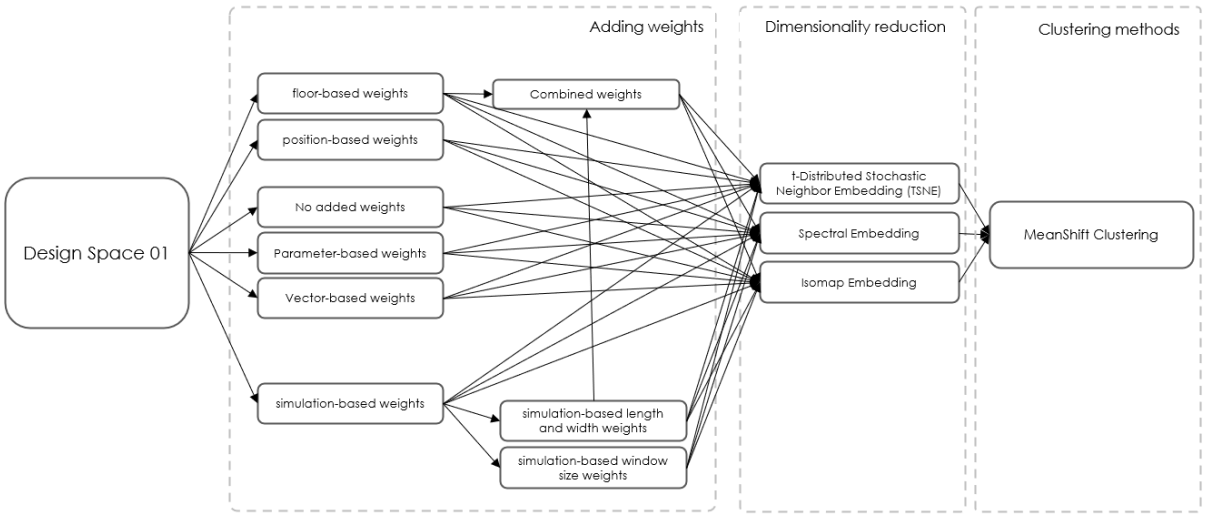


Figure 3-11 Clustering frameworks for Design Space 01.

3.1.2.1.1 Adding weights methods for Design Space 01:

If we delve into the process in greater detail, our initial step involves transforming our 50-dimensional data frame into a lower dimensional representation through various techniques outlined below. Subsequently, the lower dimensional data points, along with their corresponding simulated outcome (annual energy consumption per square meter), are employed as training input for a linear regression model [71] using scikit-learn's default hyperparameters [72]. This regression model operates as a non-linear regression with higher regularization. Exploring different hyperparameters of the model could potentially yield more accurate results, although this aspect has not been investigated in this paper.

Upon running the regression model, we obtain weights for each column or different characteristics of the building, which are then multiplied with the columns of the lower-dimensional data frame or directly to those character defining parameters of the actual data frame. As a result, we obtain a new weighted data frame that incorporates the importance of each characteristic or parameter. This new data frame serves as the basis for further dimensionality reduction, resulting in a final data frame with only three dimensions. It is important to note that the clustered outcome differs significantly across different techniques, as each technique emphasizes distinct characteristics of the building, such as the number of windows on each floor, the quantity of windows on different facades, length, width, and other relevant factors. For Design Space 01, the linear regression model was trained using all 3225 simulated outcomes, while the subsequent explorations involved a smaller sample size.

3.1.2.1.1.1 Floor-based weights

In this process, we transformed the original DataFrame into a new data frame consisting of only three columns. These columns represent the total sum of window-to-wall ratios (WWRs) on each of the three

floors (see Figure 3-12, Table 3-11). Since the ground floor is consistent across all designs, it is not considered in this analysis, as is the case with other techniques. The converted data frame, along with its corresponding simulated outcomes, serves as the training input for the linear regression model. As a result, the regression model provides three distinct weights for the windows on each of the three floors. These weights are then multiplied to the respective columns in the original data frame. For example, the columns representing the windows on the first floor are multiplied by the weight found for the first floor, and the same applies to the columns for the second and third floors. The resulting weighted data frame is subsequently utilized for further dimensionality reduction procedures.

	0	**1**	**2**
85401	1.5	0.6	1.6
46952	0.3	0	1.4
47872	0.3	1.6	1
52120	1.2	0.6	1.5
52141	1	1.4	1.1

Table 3-11 Converted data frame for floor-based adding weights method.

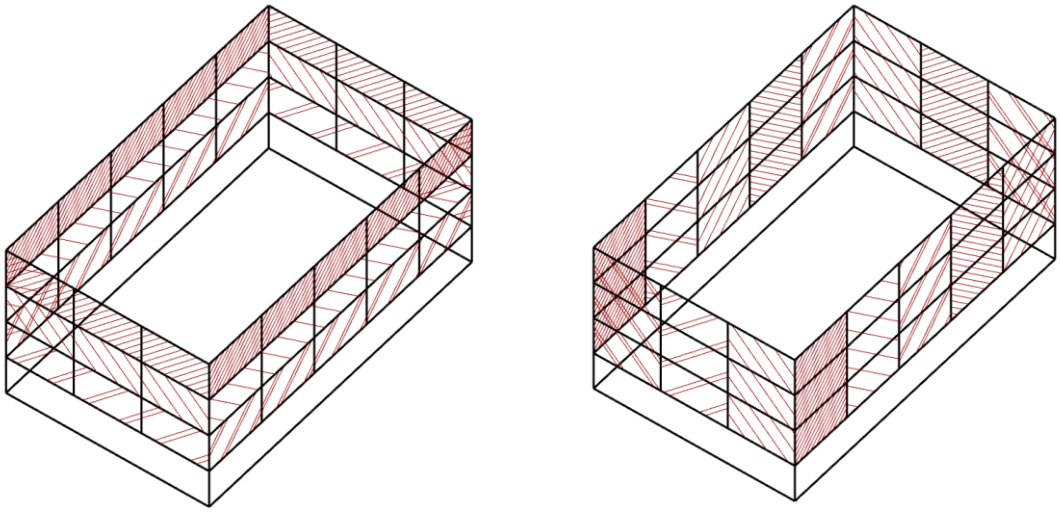


Figure 3-12 Floor-based adding weights method (left), Window position-based adding weights method (right).

3.1.2.1.1.2 Window position-based weights

In this process, the four facades of the building are divided vertically into 16 subdivisions. The north and south facades each have five window positions, totaling 10 subdivisions, while the east and west facades have six subdivisions in a similar manner. Each subdivision represents the sum of window-to-wall ratios (WWRs) for three window positions on three floors of a given design option. The initial data frame is transformed into a 16-dimensional data frame (see Table 3-12), where each row represents the total WWR for one of the 16 subdivisions (see Figure 3-12). This new low-dimensional data frame captures information about the distribution of apertures across different sides of the building.

After running the linear regression model, we obtain weights for the windows placed in each subdivision. These 16 weights are then multiplied with the corresponding columns in the original data frame. This process allows us to determine the influence of windows located in specific subdivisions, such as the south facade or the south-eastern side, on the overall performance outcome. The resulting weighted data frame provides insights into the significance of different window placements and their impact on performance. It is subsequently used for further dimensionality reduction procedures, enabling a more concise representation of the data while preserving the important characteristics related to aperture distribution.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
85401	0.4	0.5	0.7	0.5	1.3	1	1	1.4	0.6	0.3	0.6	1	0.9	0.9	1.2	1.6
46952	0.3	0.7	0.3	0.9	1.1	0.7	1	0.9	0.6	0.9	0.4	0	0.9	0	0.7	1
47872	0.3	0.8	0.7	1.2	0.9	1	0.7	0	0.7	0.8	0.3	0	0.7	1.1	0.7	1.1
52120	0.5	0.8	0.6	1.3	0.8	1.4	1.2	1.1	1.2	0	1.8	0.3	0.4	0.8	0.9	1.6
52141	0.6	1.1	0.4	1	0.9	2.1	0.8	1.2	0.7	0	1.9	0.7	0.6	1.3	1.2	1

Table 3-12 Converted data frame for window position-based adding weights method.

3.1.2.1.1.3 Parameter-based weights

In this approach, the initial data frame is used directly as the input for training the linear regression model, along with the corresponding simulated outcomes. The regression model generates 50 different weights for each of the 50 columns in the data frame. These weights are then multiplied with their respective columns to create a new data frame that retains the same structure but is weighted.

This method offers a flexible way to incorporate weights, eliminating the need to convert the data frame as described before. It allows for the inclusion of unique and diverse parameters in the design space, such as louvers and shading for different designs. The initial data frame can have a completely different structure and can be easily used in the regression model to obtain weights for the parameters it represents. By avoiding additional conversions of the data frame, this process is straightforward to execute and produces robust weights that can be used by subsequent dimensionality reduction algorithms to understand the influence of different parameters on performance outcomes. It is also applicable to different types of performance outcomes, making it versatile for future analyses.

3.1.2.1.1.4 Vector-based weights

In this approach, the initial data frame undergoes a vector-based dimensionality reduction (see 3.1.1.1.1.1) process, as described earlier, resulting in a 12-dimensional data frame. The process is similar to what we have done for the previous design space, but instead of combining the vectors into one at the end, we now have four vectors, each with three coordinates, yielding a total of 12 parameters to define a single design option (see Table 3-13). This 12-dimensional data frame is used as the training input for a linear regression model, along with their corresponding simulated outcomes. The regression model provides 12 different weights for each design option.

	0	1	2	3	4	5	6	7	8	9	10	11
85401	2.87	2.25	106.40	2.37	1.90	74.40	2.68	3.27	104.00	1.35	1.84	76.60
46952	0.95	1.88	111.00	0.71	1.47	70.40	0.90	2.58	106.80	0.45	1.25	68.60
47872	0.98	1.88	112.20	0.62	1.25	59.40	0.89	2.50	104.60	0.45	1.49	74.20
52120	4.86	2.63	112.40	4.10	2.19	78.60	4.81	3.92	112.20	2.25	2.06	75.00
52141	2.88	2.25	114.40	2.58	1.88	79.40	2.81	3.42	110.20	1.35	1.91	75.40

Table 3-13 Converted data frame for Vector-based adding weights method.

These generated weights are then multiplied with their respective columns in the converted 12-dimensional data frame, creating a newly weighted low-dimensional data frame. This weighted data frame is subsequently used for further dimensionality reductions. Similar to the window position-based adding weights method, this process provides insights into the distribution of apertures. However, it also incorporates information about the building's shape and size, making it a more robust and effective approach. The resulting data points with reduced dimensions carry valuable information that is highly significant and influential for performance prediction.

3.1.2.1.1.5 Simulation-based weights

In this process, the weights obtained from the 'Test design space 00' (see 3.1.2.1) are utilized. As mentioned earlier, the test design space consists of fewer data points and parameters with a similar ground floor condition. By running the linear regression model, we obtain six parameters: four weights for the windows located in the north, west, south, and east facades, and two parameters for the length and width of the building (see Figure 3-10). These six weights are then multiplied with the respective columns of the initial data frame.

The purpose of this process is to establish universal weights that can be applied to any design space. The weights generated for a specific site can be utilized for multiple other design spaces. If an analysis is required for a different location with slightly different conditions, simulations can be run for another small-scale design space. However, it is important to note that the effectiveness of universal weights depends on their accuracy. Further research is needed to explore the generation of weights that can be applied to multiple design spaces, an aspect that has not been extensively investigated in this paper.

3.1.2.1.1.6 Simulation-based length and width weights

This process is similar to the simulation-based adding weights method, with the main difference being the exclusion of weight multiplication for window positions in different facades.

3.1.2.1.1.7 Simulation-based window size weights

This process is same as the simulation-based adding weights method, except this excludes multiplying weights for the length and width of the building. By including two different methods of adding simulation-based weights, the aim is to investigate whether the size and shape of the building have a greater influence on the performance outcome or if it is the window positions that play a more significant role.

3.1.2.1.1.8 Combined weights

This process was explored to investigate the possibility of combining weights obtained from two different techniques and reducing simulation time. It involves the utilization of a total of 5 weights: 3 weights for the windows at three different floors obtained from the floor-based adding weights technique, and 2 weights for the length and width of the building obtained from the simulation-based adding weights technique.

However, it should be noted that weights generated from the two different linear regression models may have significantly different ranges. To address this, all the floor-based weights were remapped to a range of 1 to 10. The remapped floor-based weights are then multiplied with their respective columns, while the weights for the length and width are multiplied with the corresponding length and width columns of the initial data frame. As a result, the newly formed weighted data frame maintains the same structure as before, which is then used for further dimensionality reduction processes.

3.1.2.1.1.9 No added weights

To find out the benefits of adding weights by different techniques we have also kept a framework where no weights were added with the initial data frame. At the end we have compared these different frameworks by generating "Model error vs cluster count' Pareto frontiers for each.

3.1.2.1.2 Dimensionality reduction methods for Design Space 01:

Based on our previous understanding, we have chosen to focus on three dimensionality reduction processes for this design space: t-distributed stochastic neighbor embedding (t-SNE), isomap embedding, and spectral embedding. These methods have shown promising outcomes, while other methods did not generate meaningful results. Minor changes in hyperparameters for the t-SNE model was necessary to make clusters more district and separated from each other (perplexity: 30, early exaggeration: 12). Similar values are used for all further investigations. However, we acknowledge that the excluded methods may still have potential in different contexts or situations. Due to our limited timeline, we made the decision to prioritize the selected methods for further investigation.

3.1.2.1.3 Clustering method for Design Space 01:

Given that the dimensionality reduction processes aim to densify data points into smaller groups, we have opted to utilize the meanshift clustering method for further investigations. Meanshift is a density-based clustering algorithm that is well-suited for identifying clusters of varying shapes and sizes. One of the major advantages of meanshift clustering is its ability to adaptively determine the number of clusters by adjusting the bandwidth parameter. By changing the bandwidth value, we can easily control the granularity of the clustering results and obtain different amounts of clusters, allowing for a more flexible and comprehensive analysis of the data. We experimented with varying bandwidths ranging from 0.001 to 0.051 to obtain different numbers of clusters using. We increased the bandwidth by 0.001 in each iteration and observed a total of 50 iterations.

3.1.3 Design Space 02:

The third design space, called 'Design Space 02', includes 6000 four-storied design options, increasing the number of data points due to an increase in design parameters. This space is similar to the previous one, with the addition of shades on each window to create a more realistic representation of buildings and understand their effects on different frameworks. The DataFrame now has 6000 rows and a total of 98 columns. The first 48 columns define the WWR of windows located on different facades and floors (see Table 3-15), similar to Design Space 00. The 49th and 50th columns represent the lengths and widths of all design options, similar to Design Space 01. The configuration of the ground floor is the same as before.

Columns 50 through the end define the depth of shades, represented by values less than one, indicating the portion of the maximum depth of 2.5m. To determine the assigned depth of shade for any window, values extracted from the DataFrame are multiplied with the maximum depth. The chronology of these shades is the same as the windows, with columns 50-64, 65-73, 74-88, and 89-97 representing the depth of shades for the north, west, south, and east facades, respectively (see Figure 3-13). Each shade covers three sides of the window, leaving the bottom side without a shade. The DataFrame allows for a shade depth of 0 for any window, but no window segment with a WWR of zero has a shade depth other than zero to prevent any bias in the DataFrame.

Building properties (Design Space 02)	**values**
Building length	Range (10m-50m) (north-south)
Building width	Range (10m-50m) (east-west)
Window segments (North-South)	5 x3
Window segments (East-west)	3 x 3
Window sizes	Range(0m-5m)
Shade depth	Range(0m-2.5m)
Ground floor WWR	0.6
Number of floors	4
Floor-to-floor height	3m
Surrounding context	Not considered
DataFrame properties	
Amount of data points	6000
Number of parameters	98

Table 3-14 Building properties for Design Space 02

	North facade 0-14			west facade 15-23			south facade 24-38			east facade 39-47		Length	width	North shade 50-64			west shade 65-73			south shade 74-88			east shade 89-97	
	0	...	14	15	...23	24		38	39	47	48	49	50	...	64	65	...73	74	...	88	89	...97
32459	0.4	...	0	0	...0.4	0.3	0	0	0.7	0	0.4	0	0	...	0	0.5	...0.7	0	...	0	0	...0
82397	0.7	...	0.4	0	...0.4	0	0	0.7	0.7	0.3	0	0	0.3	...	0	0	...0.7	0.3	...	0.6	0.5	...0.6
1702	0.6	...	0	0	...0	0.5	0.5	0	0	0.5	0.7	0	0.6	...	0.7	0.6	...0.6	0	...	0	0	...0
52330	0	...	0.7	0.6	...0.7	0	0.3	0	0.5	0	0.6	0.5	0	...	0.4	0.5	...0	0.3	...	0.4	0	...0.5
87482	0.6	...	0	0	...0.5	0	0	0.6	0	0.7	0	0	0	...	0	0	...0	0	...	0	0	...0

Figure 3-13 Data structure of Design Space 02.

	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**
3177	0	0.4	0	0	0.4	0	0	0.6	0.5	0	0	0	0.5	0.3	0.7	0.7	0.7	0.5
3373	0.3	0	0	0.6	0.4	0	0.3	0	0.5	0	0	0	0	0.4	0.3	0	0.5	0
4243	0	0.7	0	0.7	0	0.3	0.7	0.4	0	0.3	0.3	0.3	0.3	0	0	0	0.6	0
3487	0	0	0	0.7	0	0	0	0	0.7	0.3	0	0.3	0.3	0	0	0	0.5	0
4878	0	0	0	0	0	0.5	0	0	0	0.6	0	0.7	0	0	0.4	0.3	0.5	0.4

Table 3-15 Sample data frame of Design Space 02.

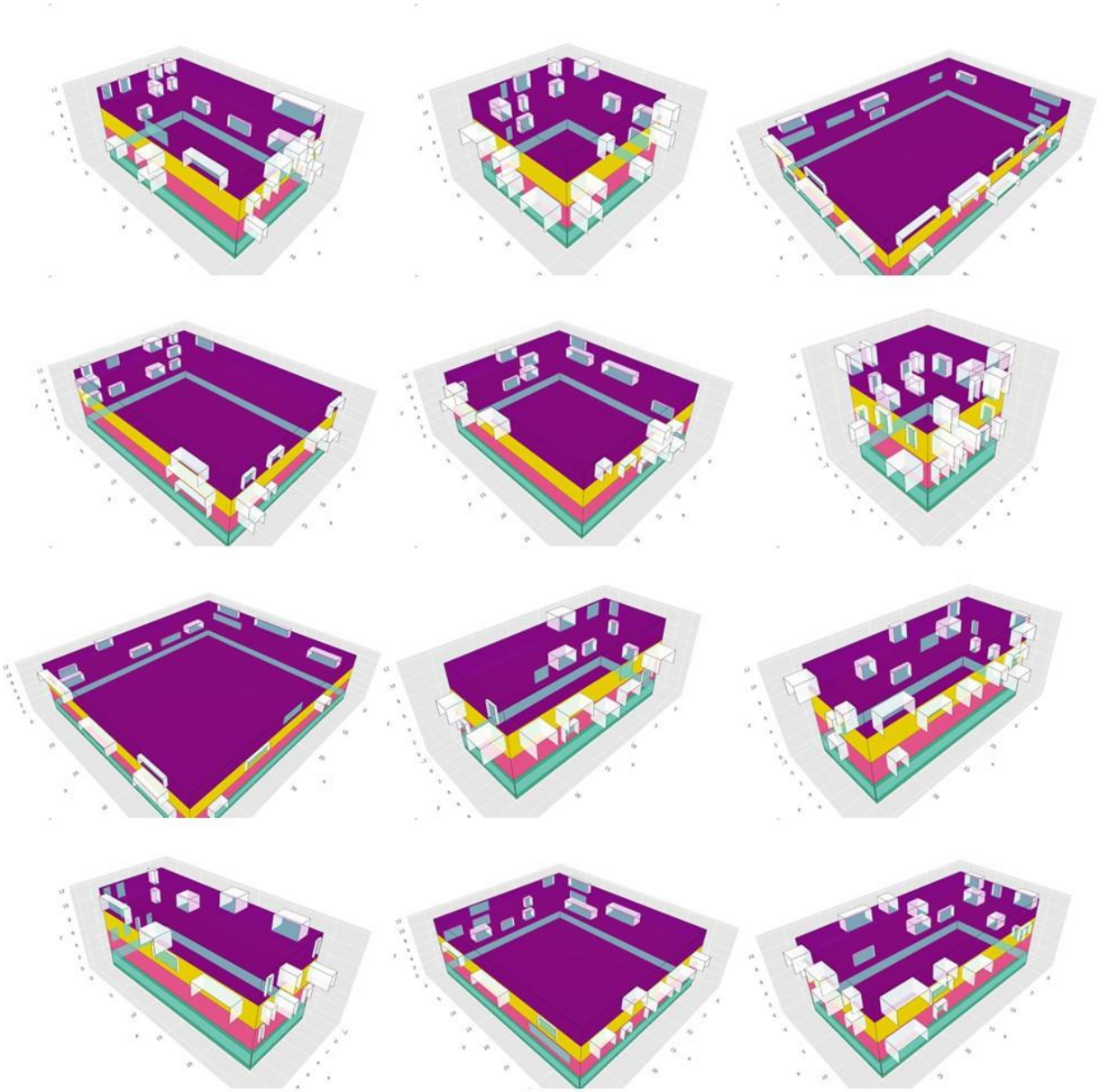


Figure 3-14 Randomly Selected 12 design options from Design Space 02.

3.1.3.1 Clustering Frameworks for design space 02:

Previous studies that involved adding weights to our DataFrame proved to be effective in producing lower model errors while using fewer clusters. However, it is noted in the results section (see 4.2) that most of the weights were derived from data containing the performance outcomes of all design options, which did not provide any insight into how much time could be saved by following a particular framework. To address this issue, we conducted further investigations by trying different percentages of the dataset to obtain the weights. Prior to determining how to add the weights before dimensionality reduction, we incorporated an additional step by defining the percentage of the dataset that would be utilized for linear regression. We experimented with four percentages - 3%, 5%, 10%, and 12% (180, 300, 600, 1200), each of which impacted the weights generated by linear regression and subsequently changed the arrangement of data points in reduced dimensions (see Figure 3-15).

With insights gained from the previous investigation (see 4.2), we proceeded with five adding weights methods, keeping all procedures unchanged except for the vector-based adding weights system. Since the new design space now included shades, we needed to investigate how this new information could be added to the produced vectors. The team developed four different ways of including shading information with the vectors, resulting in eight different ways of adding weights that were compared with a framework that had no weights added. For dimensionality reduction, we used the previously mentioned t-distributed stochastic neighbor embedding (t-SNE), isomap embedding, and spectral embedding (spectral embedding is only when no weights are added), and for clustering, we used only the mean-shift clustering algorithm. A total of 56 frameworks were compared with each other to identify the best one.

The charts depicting the Pareto distribution of 'Model error vs cluster count' for this new design space (see Figure 3-15) provide a clear indication of the potential time savings that can be achieved by following a framework. Initially, simulations were conducted to obtain weights, which were added to the cluster count. The total number of simulations required is the sum of the simulations conducted to obtain weights and the number of clusters provided by the mean-shift algorithm.

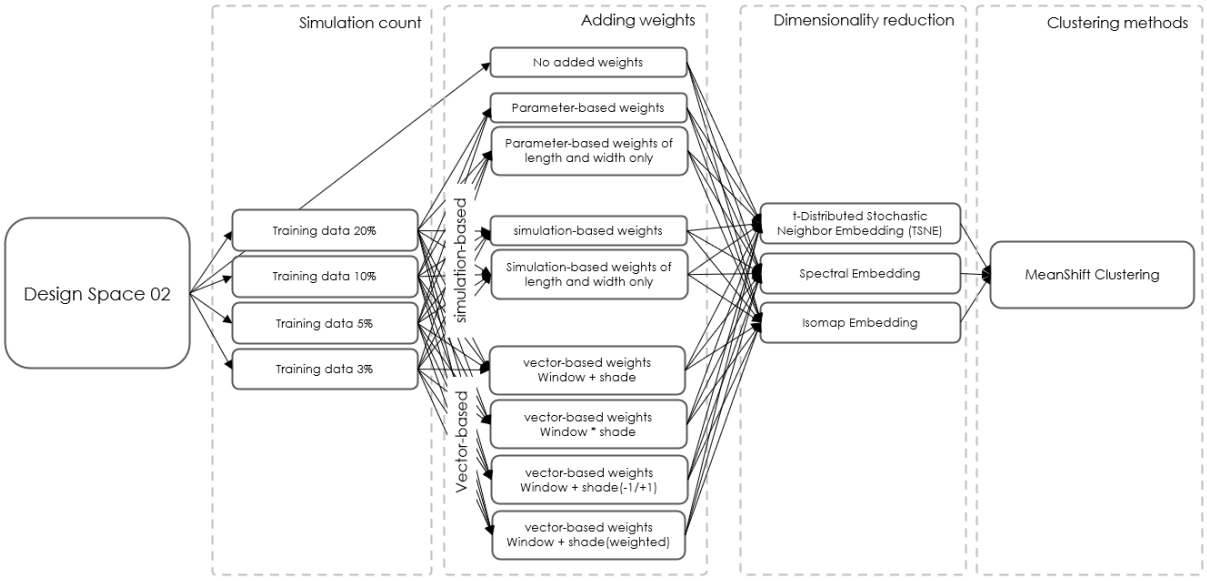


Figure 3-15 Clustering frameworks for Design Space 02.

3.1.3.1.1 Adding weights methods for Design Space 02

For each different framework, after selecting the percentage of the data frame to be used for training the linear regression model, the next step involves adding extracted weights to the data frame. While the dimensionality reduction and clustering methods remain largely the same as in the previous Design Space, the adding weights methods are customized to include shades in this particular design space.

3.1.3.1.1.1 Parameter-based weights

This process is same as the previously used parameter-based adding weights method for design space 00 (see 3.1.2.1.1.3). For this design space we get 98 weights for 98 columns,

3.1.3.1.1.2 Parameter-based length and width weights

This process is similar to the previously used parameter-based adding weights method (see 3.1.2.1.1.3). However, instead of considering all 98 weights provided by the linear regression model, this process focuses solely on the weights corresponding to the building length and width. These weights are multiplied with their respective columns in the data frame. The rationale behind this approach stems from previous insights into the influence of the building's shape or volume on the performance outcome. When we have only used

the weights of length and weights from 'simulation-based adding weights' method, significant decrease of model error is observed (see 4.2.6), for why it is assumed that this process would help in a similar way.

3.1.3.1.1.3 Simulation-based weights

Similar to the previously employed simulation-based adding weights method (see 3.1.2.1.1.5), a new smaller-scale design space called "Test design space 01" has been created to accommodate the inclusion of shades. This new design space consists of 600 design options, which represents approximately 20% of the entire design space.

Test design space 01 is composed of a total of 10 columns. Columns 0 to 3 represent the window-to-wall ratios (WWRs), while columns 4 to 7 correspond to the depth of shades on the north, west, south, and east facades, respectively (see Table 3-17). The remaining two columns define the length and width of each design option. The range of shade depths spans from 0m to 2.5m, mirroring the characteristics of Design Space 02. Furthermore, the ground floor condition remains consistent with Design Space 02, featuring the highest WWR and no shades.

Building properties (Test Design Space 01)	**values**
Building length	Range (10m-50m) (north-south)
Building width	Range (10m-50m) (east-west)
WWR (Window sizes)	Range(0.1 – 0.9)
Shades	Range(0m-2.5m)
Ground floor WWR	0.6
Number of floors	2
Floor-to-floor height	3m
Surrounding context	Not considered
DataFrame properties	
Amount of data points	600
Number of parameters	10

Table 3-16 Building Properties for Test Design Space 01.

	0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
50	0.5	0.3	0.7	0.9	0.25	0.3	0.5	0.7	0.2	0.7
43	0.2	0.7	1	0.2	0.1	0.25	0.7	0.5	0.8	1
51	0.7	0.1	0.1	0.2	0.1	0.1	0.8	1	0.2	0.6
117	0.4	0.1	0.1	0.5	0.1	0.35	0.9	0.7	0.9	0.9
522	0.3	1	0.6	0.5	0.5	0.3	0.7	0.8	1	0.6

Table 3-17 Sample Data frame of Test Design Space 01.

3.1.3.1.1.4 Simulation-based length and width weights

This process takes weights only for the length and width of the building generated from 'Test design space 01'. Extracted weights are multiplied with column 49th and 50th of the design space 02.

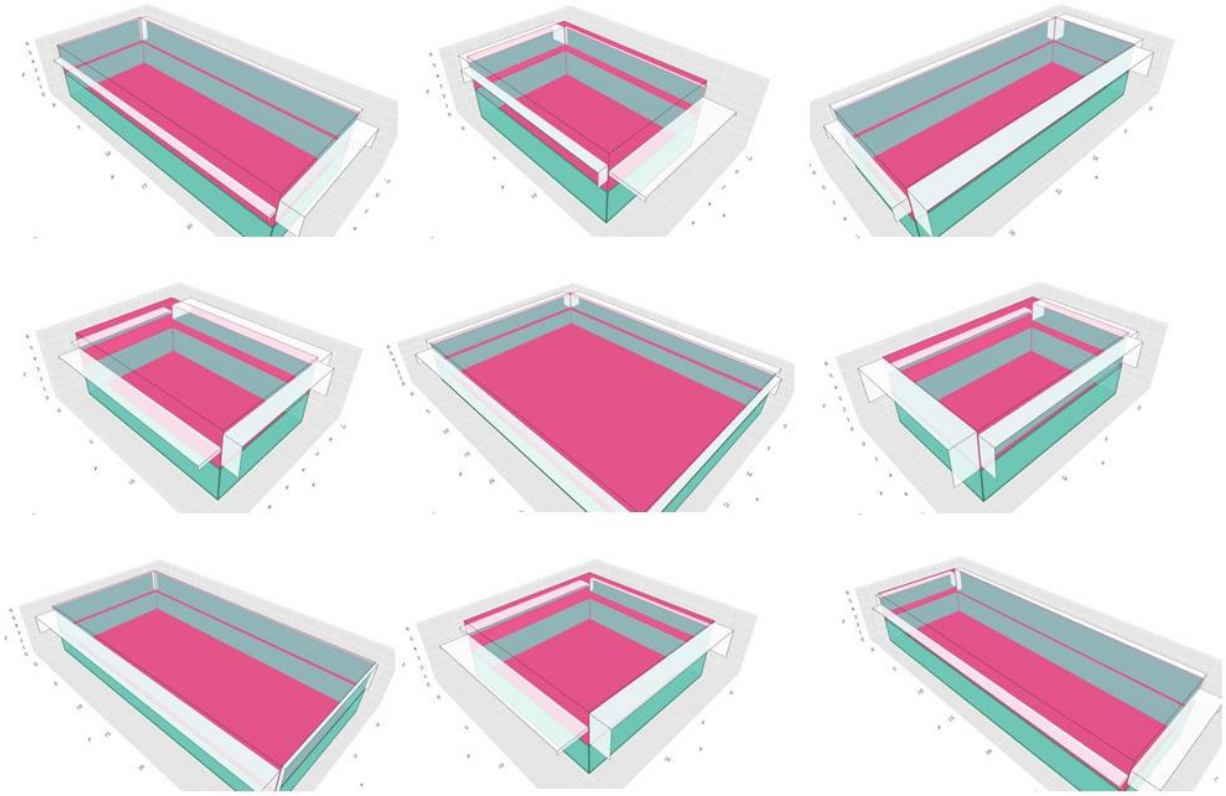


Table 3-18 Randomly selected 9 design options from Test Design Space 01.

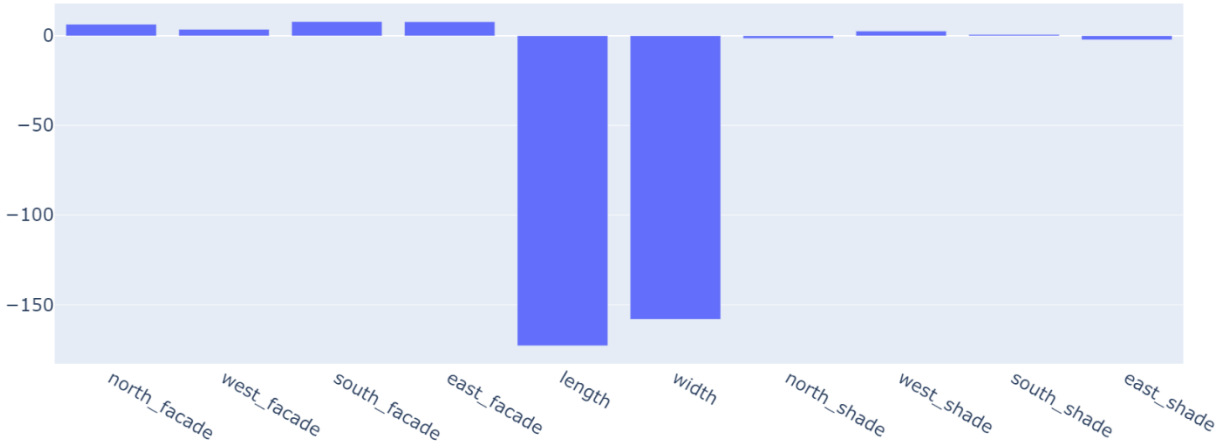


Figure 3-16 Weights for different parameters of Test Design Space 01.

3.1.3.1.1.5 Vector-based (window + shade)

This process is like the vector-based adding weights method employed for design space 01 (see 3.1.2.1.1.4), with the addition of incorporating information about the depth of shades, which presented a significant challenge. In the previous method, the window-to-wall ratio (WWR) was multiplied by the corresponding vector. However, when considering the decimal values representing the shading depth, we explored four different approaches.

In the first method, we computed the sum of the WWR and shading depth value for each window and multiplied the total by the vector associated with that window. Subsequently, all vectors were combined into a single vector for each façade. Given that we have four facades in this design space, each with three coordinates, the resulting data frame became 12-dimensional.

	0	1	2	3	4	5	6	7	8	9	10	11
3177	1.81	2.25	140.20	1.85	2.61	95.20	1.37	3.51	116.80	0.68	1.41	59.60
3373	4.01	1.88	127.20	3.01	1.56	79.40	3.52	3.20	115.80	1.58	1.93	95.00
4243	1.08	3.38	127.00	0.85	2.82	76.40	1.04	6.12	125.00	0.45	3.15	74.40
3487	1.05	1.88	125.80	0.92	1.62	85.80	1.10	3.65	140.60	0.45	1.43	68.40
4878	2.88	3.38	122.20	2.73	2.70	76.40	2.88	5.58	124.20	1.35	2.66	89.00

Table 3-19 Sample Data frame for Vector-based (window + shade) adding weights method.

3.1.3.1.1.6 Vector-based (window * shade)

In this method, instead of multiplying the sum of the window-to-wall ratio (WWR) and shading depth with the corresponding window vector, we multiplied the product of the WWR and shading depth. The more accurate lower dimensional representation would generate lower model error.

	0	1	2	3	4	5	6	7	8	9	10	11
3177	1.31	2.25	104.08	0.99	1.72	65.48	1.18	2.54	96.90	0.68	1.36	54.60
3373	2.94	1.88	98.58	1.87	1.21	59.48	2.87	2.27	97.38	1.58	1.40	68.02
4243	0.84	3.38	96.40	0.53	2.20	57.00	0.84	4.20	99.90	0.45	2.32	59.04
3487	0.82	1.88	97.84	0.59	1.28	64.50	0.83	2.35	103.64	0.45	1.21	58.20
4878	2.40	3.38	98.30	1.58	2.11	58.30	2.41	3.93	99.74	1.35	2.16	64.38

Table 3-20 Sample Data frame for Vector-based (window * shade) adding weights method.

3.1.3.1.1.7 Vector-based (window + shade(weighted))

The influence of shading devices can vary depending on their orientation. For instance, a greater shading depth on the west facade of a building located in Seattle may lead to a significant reduction in annual energy consumption compared to other facades. Recognizing the varying influence of shading devices on performance outcomes, this process aims to incorporate their individual impacts while reducing the dimensionality of the data. To achieve this, weights for the shading depths of the four different facades are obtained from the linear regression model run on "Test design space 01". The decimal shading depth values are multiplied by their corresponding weights. The resulting product is then multiplied by the window-to-wall ratio (WWR) and subsequently multiplied by the respective vector associated with each window. By considering the weights, shading depths, and WWRs in this manner, we aim to capture the distinct influences of shading devices on the overall performance outcome while reducing the dimensionality of the data.

	0	1	2	3	4	5	6	7	8	z	9	10	11
3177	2.30	2.25	179.26	2.69	3.71	126.77	1.46	4.09	133.74		0.68	1.41	57.80
3373	4.62	1.88	144.39	3.44	1.69	88.74	4.20	4.39	136.78		1.58	2.84	142.88
4243	1.30	3.38	130.57	0.94	3.14	73.02	1.33	8.60	152.76		0.45	3.91	85.95
3487	1.22	1.88	139.70	1.34	2.12	120.56	1.28	4.85	176.47		0.45	1.68	80.62
4878	3.23	3.38	142.62	2.81	2.58	81.26	3.26	6.88	151.74		1.35	2.90	119.80

Table 3-21 Sample Data frame for Vector-based (window + shade(weighted)) adding weights method.

3.1.3.1.1.8 Vector-based (window + shade*(-1/1))

The weights obtained for shading depths in different orientations were not uniformly positive. Specifically, the weights for north and south facing shades had negative values, while the weights for west and east facades were positive. In this method, we incorporate these weight values by adding the decimal shading depth value to the window-to-wall ratio (WWR) when the weights are positive, and subtracting the shading depth value from the WWR when the weights are negative. The resulting calculated value is then multiplied by the corresponding vector associated with each window.

	0	1	2	3	4	5	6	7	8	9	10	11
3177	1.10	2.25	86.20	1.85	2.61	95.20	1.37	3.51	116.80	0.68	1.35	55.60
3373	2.50	1.88	81.60	3.01	1.56	79.40	3.52	3.20	115.80	1.58	1.07	45.00
4243	0.76	3.38	97.00	0.85	2.82	76.40	1.04	6.12	125.00	0.45	2.13	56.80
3487	0.75	1.88	81.00	0.92	1.62	85.80	1.10	3.65	140.60	0.45	1.08	51.60
4878	2.32	3.38	93.00	2.73	2.70	76.40	2.88	5.58	124.20	1.35	2.15	53.80

Table 3-22 Sample Data frame for Vector-based (window + shade*(-1/1)) adding weights method.

3.1.3.1.1.9 No added weights:

Similar to the previous design space frameworks, we also explored frameworks without adding any weights. This approach allowed us to examine the benefits and effects of not incorporating weights into the out data frame.

3.1.4 Design Space 03:

The team moved on to more intricate design spaces to assess the performance of the frameworks under diverse conditions. The new design space, named 'Design Space 03', comprises 5470 L-shaped, four-story design options. The length of the two arms of each option ranges from 10m to 50m, while their depth varies from 3m to 35m. Unlike the previous design space, where window segments had different window-to-wall ratios (WWRs) (see Table 3-14, Table 3-15) , in this design space, each face on each floor of any design option has a distinct WWR. There are six faces on each floor, resulting in a total of 18 faces across three floors. The size of the window in each grid position is determined by the WWR, with no windows at all if a face length is 3m or less. The ground floor configuration remains the same as Design Space 00.

The DataFrame for Design Space 03 comprises 43 columns (see Table 3-24). The first 18 columns define the WWR for all faces. The first 6 columns represent the WWR for the 6 faces on the first floor, the next 6 columns represent the second floor, and the last 3 columns represent the third floor. WWRs are randomly assigned values from 0 to 0.98. Any window can have a maximum length of 3m. Columns 18-35 define shades for different faces in a similar manner. The highest depth of any shade is 2.5m. The following 6 columns (36-41) define the length of the 6 faces of all design options, and the last column describes the area of all design options (see Figure 3-18). Similar to Design Space 02, any face with a WWR of zero also has a shade depth of zero, while faces with a WWR greater than zero may have a shade depth of zero.

Building properties (Design Space 02)	**values**
Building north-south arm	Range (10m-50m)
Building east-west arm	Range (10m-50m)
Depth of any arm	Range (3m-35m)
Number of faces on each floor	6
WWR for any face	Range (0-0.98)
Window sizes	Range(0m-3m)
Shade depth	Range(0m-2.5m)
Ground floor WWR	0.6
Number of floors	4

Floor-to-floor height	3m
Surrounding context	Not considered
DataFrame properties	
Amount of data points	5470
Number of parameters	43

Table 3-23 Building Properties for Design Space 02.

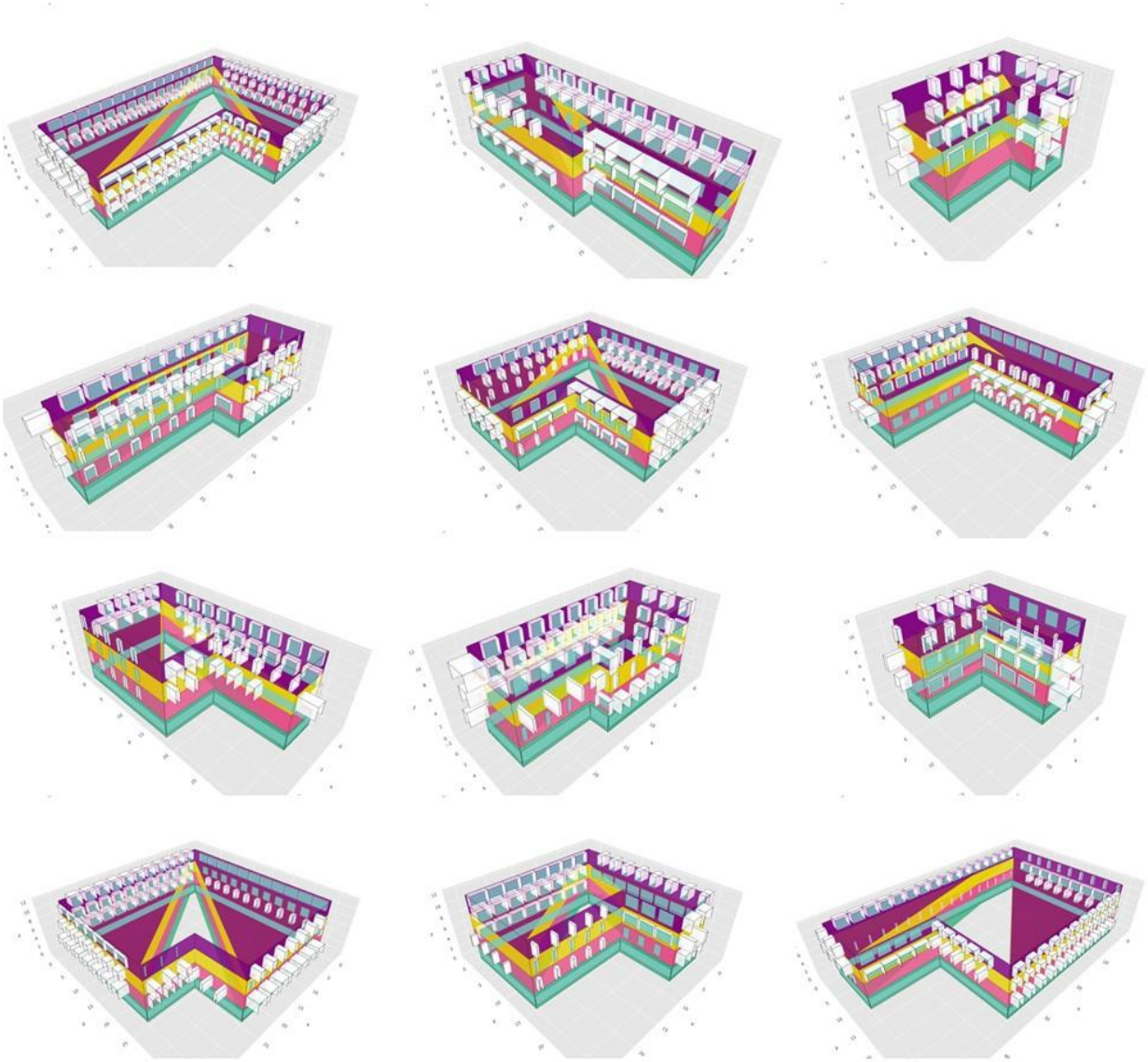


Figure 3-17 Randomly selected 12 Design options from Design Space 03.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15…
3177	0.5	0.3	0.2	0.98	0.7	0.1	0.6	0	0.4	0.1	0.9	0.8	0.1	0.1	0.4	0.2…
3373	0.2	0.7	0	0.9	0.1	0.8	0.98	0.98	0.7	0.8	0.3	0	0.2	0.7	0.8	0.5…
4243	0.98	0.6	0.4	0	0.6	0.4	0	0.8	0.6	0.5	0.5	0.2	0.7	0	0.7	0.1…
3487	0.2	0.98	0.5	0.1	0.98	0	0.9	0.4	0.3	0.2	0.5	0.6	0.6	0.6	0.2	0.1…
4878	0.2	0.7	0.6	0.2	0.3	0.2	0.98	0.1	0.4	0.8	0.1	0.2	0.7	0.3	0.7	0.3…

Table 3-24 Sample data frame of Design Space 03.

	First floor facades			Second floor facades			Third floor facades			First floor Shades			Second floor Shades			Third floor Shades			Length of faces			
	0-5			6-11			12-17			18-23			24-29			30-35			36-41			area
	0	…	5	6	…	11	12	…	17	18	…	23	24	…	29	30	…	35	36	…	41	42
3177	0.5	…	0.1	0.6	…	0.8	0.1	…	0.3	0.8	…	0.1	0.8	…	0.98	0.3	…	0.4	0.35	…	0.5	0.13125
3373	0.2	…	0.8	0.98	…	0	0.2	…	0.9	0.3	…	0.2	0.98	…	0.1	0.1	…	0.98	0.3	…	0.25	0.04875
4243	0.98	…	0.4	0	…	0.2	0.7	…	0.1	0.1	…	0.6	0.2	…	0.7	0.8	…	0	0.1	…	0.45	0.0288
3487	0.2	…	0	0.9	…	0.6	0.6	…	0.6	0.2	…	0.6	0	…	0	0.98	…	0.5	0.45	…	0.35	0.1008
4878	0.2	…	0.2	0.98	…	0.2	0.7	…	0.5	0	…	0	0.5	…	0	0.9	…	0.8	0.5	…	0.1	0.0425

Figure 3-18 Data Structure of Design Space 03.

Clustering Frameworks for design space 03:

The design space generated for the purpose of validating previously investigated frameworks remains largely unchanged, with only a few exceptions. Specifically, it was found that dimensionality reduction through multiplication of WWR and shade, as well as shade weights, was not very effective for the vector-based method (see 4.4.4), and thus was not used in Design Space 03. One area of investigation in this new design space is the determination of the optimal number of vectors for the vector-based dimensionality reduction method. While previous iterations used four facades and three values per vector, the six facades in Design Space 03 will require six vectors and 18 values per design space after dimensionality reduction.

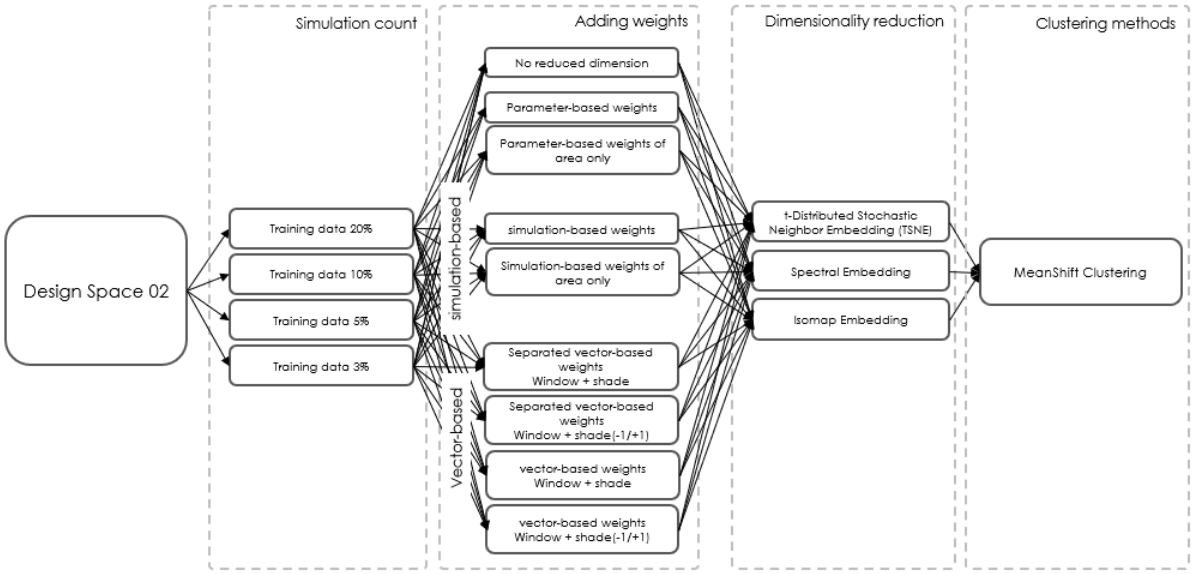


Figure 3-19 Clustering Frameworks for Design Space 03.

Furthermore, we will explore the impact of grouping similarly oriented facades into one vector, resulting in four vectors for each side due to the orthogonal nature of the faces. Additionally, we will consider the weights generated for the previous design space for the simulation-based dimensionality reduction methods. Another change from previous frameworks is the inclusion of total area and its weight, instead of

considering the weights of length and width in methods that previously only incorporated those parameters. Similar to previous design space after selecting simulation count out of four and adding weights we have used three dimensionality reduction methods which are t-distributed stochastic neighbor embedding (t-SNE), isomap embedding, and spectral embedding and one clustering method which is Meanshift clustering (see Figure 3-19).

3.1.4.1.1 Adding weights methods for Design Space 03

The valuable insights gained from the previous experiments conducted on 'Design space 02' (see 4.4.5) have guided us in determining the effectiveness of various frameworks. In this design space, we aim to cross-check the performance of these frameworks. However, as mentioned before, the vector-based adding weights methods, specifically the one considering the product of WWR and shading depth, as well as the method involving multiplication of weights of shading depths for different orientation, did not yield satisfactory results in terms of generating an effective Pareto frontier. As a result, we have chosen to avoid these methods while exploring this particular design space.

3.1.4.1.1.1 Parameter-based weights:

This process is same as explained before for the 'design space 02' (see 3.1.3.1.1.1).

3.1.4.1.1.2 Parameter-based area weight:

As mentioned earlier, the DataFrame used for 'Design space 03' includes an additional column that represents the area of each design option. In this method, we have followed the parameter-based adding weights approach (see 3.1.3.1.1.1), but with a slight modification. Instead of considering all the extracted weights, we have focused on multiplying the weight associated with the last column, which corresponds to the total area of the building. By incorporating the influence of the building's total floor area, we aim to capture its impact on the performance outcome.

3.1.4.1.1.3 Simulation-based weight:

This process is similar to what was done previously for 'design space 02' (see 3.1.3.1.1.3). In this case, we have not generated a new smaller-scale design space, as our objective was to explore the possibility of reusing the extracted weights for multiple frameworks. By applying the previously obtained weights from "Test design space 01" to the current design space, we aim to assess their effectiveness and determine if they can be consistently applied across different scenarios.

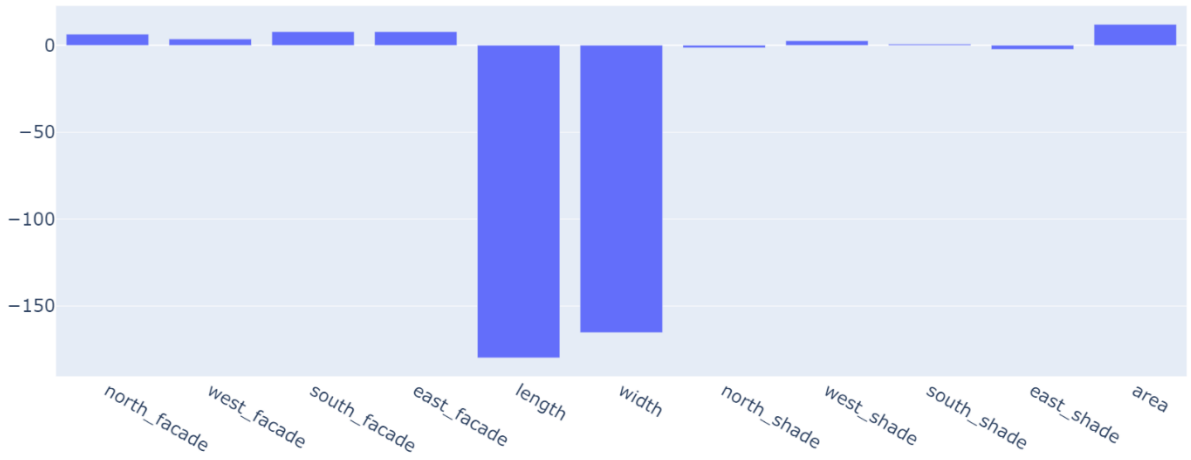


Figure 3-20 Added new area weight from Test Design Space 01.

3.1.4.1.1.4 Simulation-based area weight:

From "Test design space 01" (see 3.1.3.1.1.3), we have extracted a new weight specifically for the total area of the building (see Figure 3-20). This weight is applied in a same way we have done in "parameter-based area weight" method, where it is multiplied only with the values in the last column of the generated data frame.

3.1.4.1.1.5 Separated vector-based weights (window + shade)

In this process, we apply the vector-based adding weights method, which takes into account the sum of the window and shading depth, similar to previous design spaces. However, unlike the previous design spaces, the design options in this particular design space have six faces: two facing north, one facing west, one facing south, and two facing east (see Figure 3-21). For each of these six faces, we have corresponding vectors with three coordinates. By applying this process, the original 43-dimensional data frame is transformed into an 18-dimensional data frame (see Table 3-25), reducing the dimensionality, and capturing the essential features of the design options in a more concise representation.

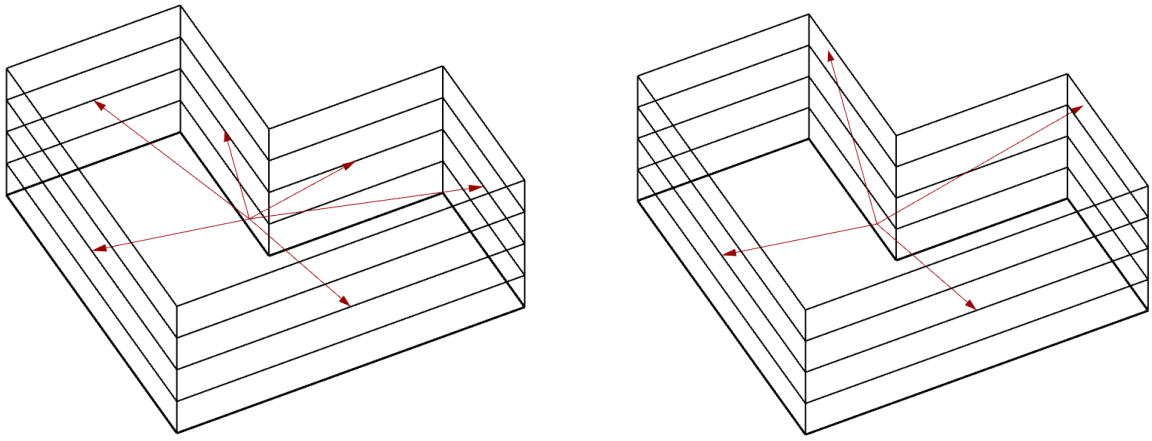


Figure 3-21 Separated vectors (left) Combined vectors (right).

	0	1	2	3	4	5	...	13	14	15	16	17
3177	1035.0	687.5	226.1	770.0	634.4	80.4	...	1132.5	123.0	700.0	2118.1	345.6
3373	634.6	283.0	180.9	280.6	86.2	53.7	...	373.8	71.0	263.1	555.4	199.9
4243	73.6	156.9	66.3	218.1	379.7	114.0	...	200.8	25.7	162.8	1420.0	172.2
3487	1577.4	569.6	352.0	889.7	254.1	106.4	...	805.4	114.3	575.4	924.1	206.3
4878	2321.5	218.8	460.8	205.7	16.8	21.0	...	348.4	135.3	197.2	70.9	48.2

Table 3-25 Sample data frame for Separated vector-based adding weight method.

3.1.4.1.1.6 Separated vector-based weights (window + shade*(-1/1))

In this process, we follow the same approach as the 'Separated vector-based weights (window + shade)' method, which considers six different vectors for the six faces. However, there is a slight modification in how the shading depth values are handled. If the corresponding weight for a shading depth is negative, it is subtracted from the window value. On the other hand, if the weight is positive, the shading depth is added to the window value. This approach, similar to the previously explained vector-based weights (window + shade*(-1/1)) method (see 3.1.3.1.1.8), accounts for the different influences of shading depths on the performance outcome.

	0	1	2	3	4	5	...	13	14	15	16	17
3177	1035.0	687.5	226.1	154.0	438.4	-42.0	...	342.5	-13.5	700.0	2118.1	345.6
3373	634.6	283.0	180.9	136.6	70.2	17.7	...	37.8	-15.8	263.1	555.4	199.9

4243	73.6	156.9	66.3	38.1	249.5	-27.0	...	47.8	-2.3	162.8	1420.0	172.2
3487	1577.4	569.6	352.0	241.7	168.3	3.2	...	462.4	70.2	575.4	924.1	206.3
4878	2321.5	218.8	460.8	35.7	13.4	-6.9	...	62.4	-66.0	197.2	70.9	48.2

Table 3-26Sample data frame for Separated vector-based (-1/1) adding weight method.

3.1.4.1.1.7 Vector-based weights (window + shade)

In this process, the extracted vectors for all faces within each facade orientation are combined into a single vector. This results in a total of four vectors representing the four different facade orientations. The remaining steps of the process are the same as those followed for 'design space 02' (see 3.1.3.1.1.5).

3.1.4.1.1.8 Vector-based weights (window + shade*(-1/1))

This process is same as explained before for the 'design space 02' (see 3.1.3.1.1.8) that considers if the corresponding weights are positive or negative. Similar to the immediately explained method four vectors for four orientation is considered instead of six.

3.1.4.1.1.9 No added weights:

To demonstrate the impact of adding weights, we conducted experiments using frameworks without incorporating any weights.

3.2 Reinforcement learning-based recommender system:

After figuring out the perfect framework that would produce a minimum model error with a fewer number of clusters, it became very easy to predict the performance outcome of any huge design space within a very short amount of time. In this section how important this fast prediction was will be explained. A design dashboard is prepared with Compas and Plotly[41] (see Figure 3-22) which can be accessed online without installing any software. It was possible to present a three-dimensional model of different design options from the design space. The designer can observe the design from different angles and provide feedback. For the design dashboard, we have used design space 00 for demonstration. The dimension of the DataFrame is reduced to three with the help of t-distributed stochastic neighbor embedding (t-SNE) and 163 clusters was made with the help of mean-shift clustering.

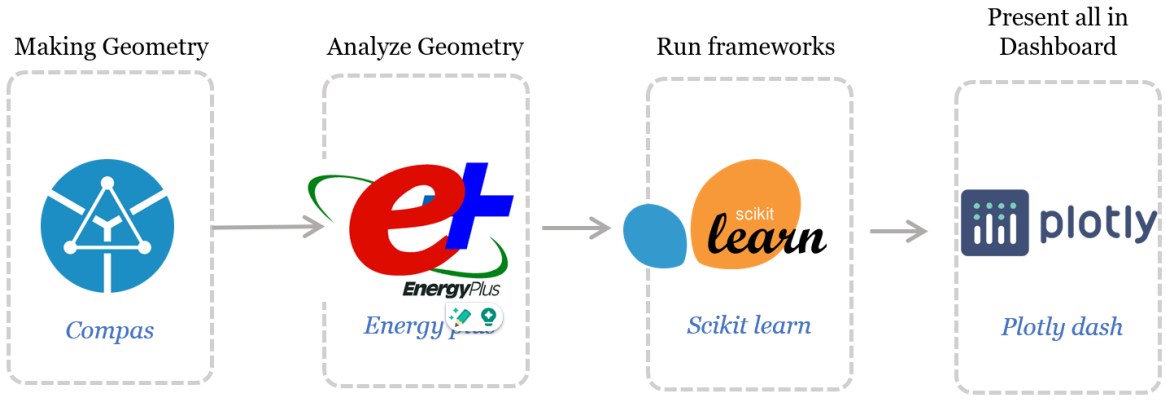


Figure 3-22 workflow for the design dashboard

3.2.1 Process explanation:

We incorporated reinforcement learning [73] to extract the designer's preference strategically. It's a type of machine learning where an agent keeps track of the user's interactions and makes decisions that reflect the user's preference. It's a robust system that constantly learns from its environment through trial and error. Each interaction provides a specified reward, which the agent tries to maximize.

At first, the agent will display a random design option to the designer. The designers can click any of the three buttons placed on top of the dashboard: like, unlike or slide (see Figure 3-25). If the designer hit 'like', the cluster the design option belongs to will be rewarded with a certain positive value. If the designer dislikes the option the reward will be the same but negative. If the designer slides the option the reward will be zero. This process is a type of reinforcement learning-based recommender system [32] which is getting widely popular in recent days.

The rewarded value will be used to update the q-value(q) of any cluster. Each cluster as well as all designs in that cluster will have a 'q-value' (also known as action-value function in reinforcement learning). This value determines which cluster will be displayed next. Before any interaction begins all the clusters would have pre-assigned q values. These values are inversely remapped performance outcomes which denotes that higher the annual energy consumption lower the q-value. All performance outcomes are mapped from 100 to 50 for this system. So, the cluster with the highest annual energy consumption per square meter will have a q-value of 50 and the cluster with the lowest annual energy consumption per square meter will have a value of hundred. Following each interaction, the q-values undergo updates based on the feedback received from the user.

The quantitative value of the user's feedback is named reward(R). A function is established to determine the appropriate reward (positive or negative) for the system. The function's output is influenced by two constants, namely alpha (α) and gamma (γ), as well as the reward(R) and the previous q-value. The old q-value is substituted with the new result derived from the reward function. This equation is commonly referred to as the Bellman equation [74]. Since only one design option is presented in each iteration, the equation becomes simpler as there are no multiple states to consider. Instead, there is only a single state. If multiple options are displayed, the algorithm would need to determine the probabilities of liking each of those options. In that case, each option represents a different state. The algorithm would calculate the probability of liking each option based on the q-values assigned to them. By considering the q-values and applying a probabilistic approach, the algorithm can estimate the likelihood of users liking different options and make decisions accordingly. However, that level of complexity is not addressed in this paper.

$$f(r) = (1-\alpha) * Q + \alpha * (R + \gamma * Q)$$

α = learning rate
γ = discount factor
R = reward defined by β
Q = Previous q-value

Equation 1 Bellman equation for incorporated reinforcement learning

3.2.2 Constants/hyperparameters:

The following four constants need to be predefined before the exploration process begins. These would allow designers to customize their exploration process strategically:

3.2.2.1 Learning rate (alpha (α)):

The alpha (α) value in this equation represents the learning rate, which determines the extent to which recent interactions have a greater impact on the new q-value compared to previous interactions. A higher alpha value gives more weight to recent preferences, which means if the designers regret their past choices when presented with more preferable options, they can change their preferences over time. The alpha value is a parameter that users need to define before they start exploring the design space, and it ranges between 0 and 1. As the number of iterations increases, the influence of alpha becomes greater than that of gamma (γ), which represents the discount factor.

3.2.2.2 Discount factor (gamma (γ)):

Although gamma (γ) may not have a significant impact on single-state reinforcement learning, it is included to facilitate multi-state reinforcement learning in the future. Gamma determines how fast the algorithm aims to reach options with higher q-values. It ensures that designers do not get stuck in a particular set of design options where their q-values may decrease over time. In multi-state reinforcement learning, a higher discount factor encourages the algorithm to prioritize reaching states with the maximum reward in the shortest number of iterations. Like the other constants, gamma also takes a value between 0 and 1, which designers need to assign before the exploration process begins.

3.2.2.3 Exploration exploitation trade-off epsilon (ε):

After the user gives feedback, the agent would have two steps. It can show design options from a randomly selected cluster, or it can show designs from the cluster that have the highest q-value. To avoid jumping all over the design space the system only allows to consider nearby clusters while searching for the highest. The constant epsilon (ε) is responsible for defining the trade-off between exploration and exploitation in the reinforcement learning process. It does not directly impact the reward function but influences the exploration process. Epsilon takes a value between 0 and 1. A higher value of epsilon promotes more random exploration, while a lower value encourages exploitation of the design options that designers have preferred. Within the Python script, an if function is used to compare a randomly generated number between 0 and 1 with epsilon. If the random number is less than epsilon, the system selects a random cluster from the nearby options. It is evident that higher epsilon values increase the possibility of generating numbers below it. The trade-off value can be adaptive over time, as users may desire less exploration as the number of iterations increases.

3.2.2.4 Priority Trade-off (beta (β)):

How much the agent would prioritize the user's preference over pre-assigned q-values is defined by constant beta (β). It is a value between 0 and 1. A higher value of Beta means the user would need to hit fewer 'like' to make the agent prioritize the design option they liked. This constant makes the difference between traditional reinforcement learning-based recommender systems and the proposed one. The commonly used algorithm doesn't need to negotiate between two features of different options. The main goal of those systems is only to increase like, but here our proposed system tries to recommend not only preferred design options but also guides through high-performing clusters to simulate designers consider well-performing designs in a spontaneous way. The process demonstrates a passive way where designers explore huge design spaces strategically and end up selecting options that can be both aesthetically pleasant and well-performing.

3.2.2.4.1 Changes of reward with alpha (α) and gamma (γ)

If the reward is high after any positive interaction it would take fewer hits to suppress the most high-performing design option. Though what should be the reward highly depends upon beta(β), other constants also have impacts on it, for why the function requires all four of them to calculate what should be the perfect value to explore the design space considering trade-offs (see Figure 3-23, Figure 3-24). To understand how we derived the reward function we need to know the inter-relationship between different constants and how they influence the design exploration.

The team conducted an investigation to determine the relationship between assigned rewards and the number of iterations required to surpass the highest q-value with those rewards, considering different values of alpha (α) and gamma (γ). It was found that when the initial reward value is lower, it takes more iterations for a cluster with the lowest q-value to exceed the highest q-value. In this system, all performance outcomes were remapped, with the lowest q-value set to 0 and the highest q-value set to 100. It was observed that the reward must be greater than 100; otherwise, it would take an almost infinite number of iterations for a cluster with a q-value of 0 to surpass a q-value of 100. We are assuming that there is the possibility, the cluster with the lowest q-vale can be the cluster the designers like most. To make the algorithm understand this fact with a positive reward of less than 100, the designers would need to hit 'like' a lot. To avoid such situations, Beta (β) is introduced, which defines an appropriate reward(R) value for the system.

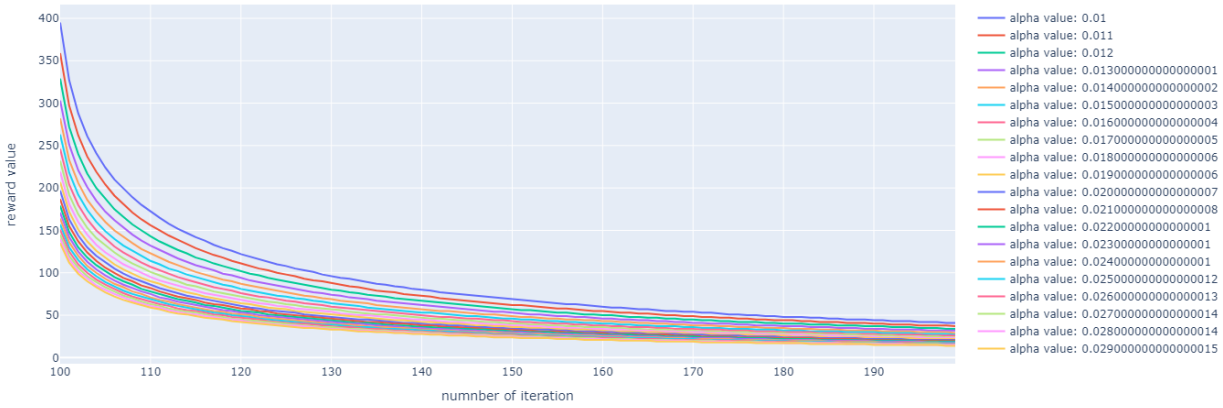


Figure 3-23 Changes of reward value with number of interactions for different alpha values.

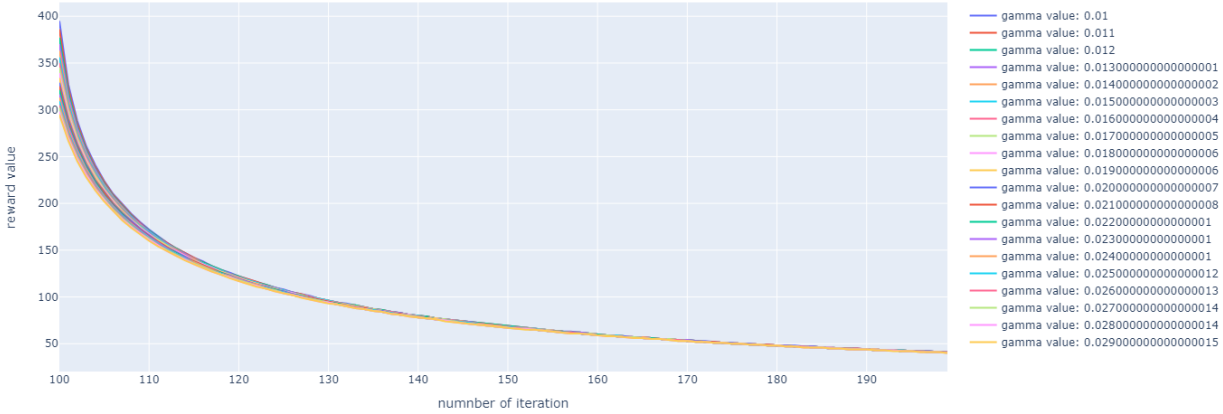


Figure 3-24 Changes of reward value with number of interactions for different gamma values.

3.2.2.4.2 Function that constructs beta(β)

A function has been developed to generate a Pareto frontier that relates reward values (ranging from 100 to 200) on the x-axis to the number of 'likes' required to achieve a q-value of 100 for the cluster with the lowest q-value on the y-axis. The same Bellman equation used in the algorithm is employed in this function to derive the number of 'likes' needed. The epsilon (ε) value is set to 0, indicating no exploration, as added exploration would only lengthen the frontier.

The function can construct the Pareto frontier with any given values of alpha (α) and gamma (γ). Before the exploration process begins, the system runs the function using pre-assigned values of alpha, gamma, and beta. For a gamma value of 0.01 and a reward value of 200, the number of 'likes' required to exceed 100 can range from 14 to 41 for different alpha values. For a reward value of 100, it can range from 135 to 395. Beta (β) defines the number of 'likes' needed to prioritize clusters despite having medium or lower q-values. It provides an appropriate reward function by considering this factor.

To illustrate this, let's consider a system with an alpha value of 0.01 and a gamma value of 0.01. Using the function, the number of 'likes' is determined for reward values ranging from 100 to 200, resulting in a range of 41 to 395. By multiplying the range (354) by beta (0.5), we obtain the desired number of 'likes' (177). The corresponding reward value can be determined from the Pareto frontier. This approach allows the system to find the appropriate reward to proceed.

However, there is an alternative approach that simplifies the process. Instead of generating the Pareto frontier, a straightforward function can be used to calculate the reward value for any given values of alpha, gamma, and beta, as their relationship is direct and well-defined. The function used in our work explores different rewards to find a cluster with q-values of 0 and 100. It's worth discussing whether working with a cluster that has an average q-value of 50, rather than 0, would be more practical, considering that a poorly performing design may also be visually unappealing. This can be an open discussion, and we can easily modify this value in the script if it appears to be more reasonable.

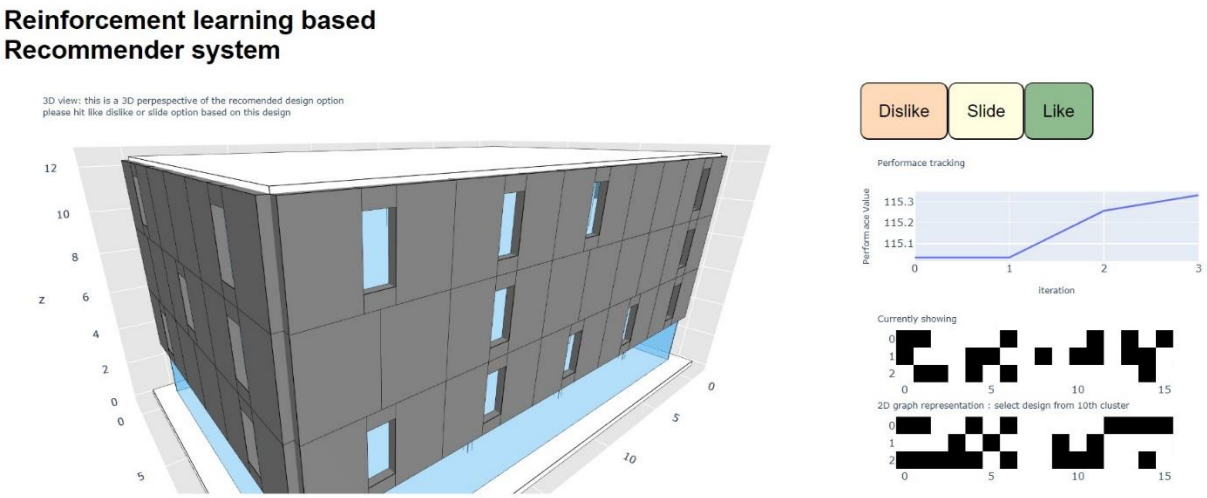


Figure 3-25 Design Dashboard

Chapter 4: Results:

This results section provides detailed insights into the outcomes generated from various frameworks applied to different design spaces. The results are presented both quantitatively and narratively, offering a comprehensive understanding of the findings. Due to space limitations, it was not feasible to include all three-dimensional scattered plots generated for different numbers of clusters. A complete framework encompasses the entire process, starting from weight addition and culminating in the creation of Pareto frontiers. These frameworks were primarily applied to design spaces 01, 02, and 03, resulting in approximately 40-50 scatter plots due to the extraction of model error for multiple cluster values. To provide a concise representation, we have carefully selected the three most comprehensive and informative scatter plots from this extensive set of images.

4.1 Framework outcomes of Design Space 00:

The outcomes of the frameworks are presented in separate sections for each dimensionality reduction method. Within each section, the relevant images associated with that particular method are arranged consecutively. Additionally, a table is provided for each dimensionality reduction method, which references the relevant figures that the reader should refer to for a more detailed analysis.

4.1.1 Vector-based dimensionality reduction results

After performing vector-based dimensionality reduction on our initial data frame, we visualized the data points in a 3-dimensional space and observed a large oval-shaped cluster. It became evident that if we were to apply clustering algorithms directly on this single cluster, the centroids of resulting groups would not effectively approximate the performance outcome. To address this, we introduced bias into the data frame to subdivide the large cluster into smaller groups. In this biasing process, we kept all binary window-to-wall ratios (WWRs) for the north facade unchanged. However, we multiplied all WWRs for the west facade by 5,000,000, all WWRs for the south facade by 50,000, and all WWRs for the east facade by -5,000,000. While different values were tested, it was observed that using larger values for the west facade WWRs stretched the large cluster and created two distinct clusters within it. The resulting biased data frame exhibited two main clusters with roughly identifying small clusters in it (see Figure 4-6).

It is assumed that the formation of these smaller clusters was likely due to limited variations in the data frame. As we explored different combinations and variations, only a subset of the possibilities was selected for further investigation. However, when generating pixelated images for different clustering methods, the results were found to be blurry, indicating that this process was ineffective in creating diverse clusters with similar performance outcomes.

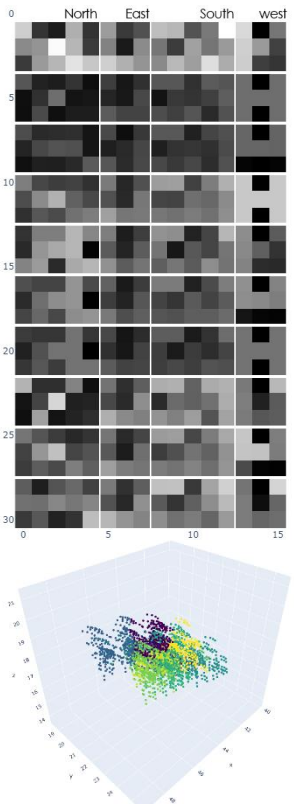


Figure 4-5 K-means Clustering (vector).

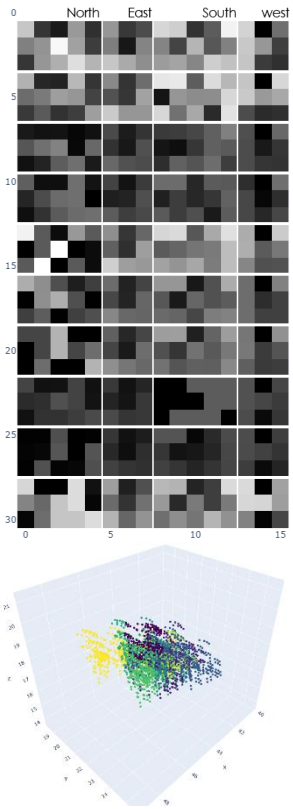


Figure 4-4 Agglomerative clustering (vector).

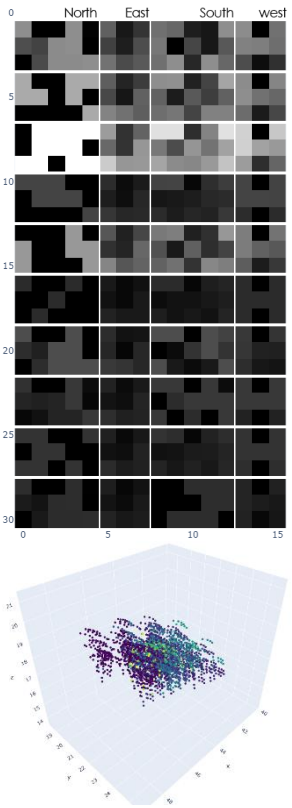


Figure 4-3 Meanshift clustering (vector)

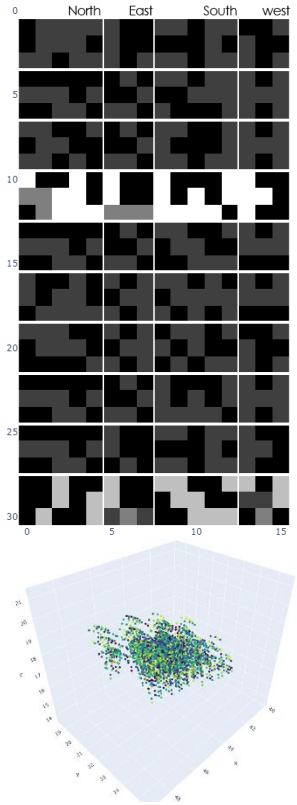


Figure 4-1 Affinity Propagation (vector).

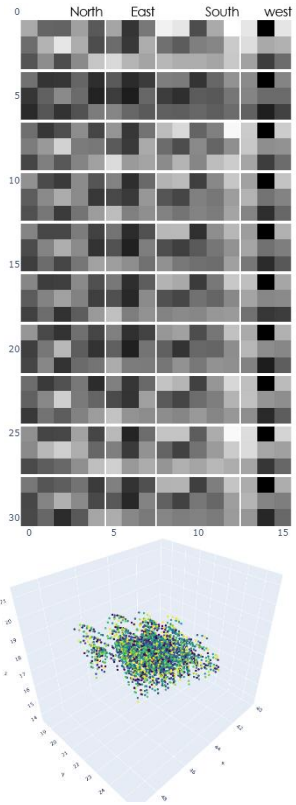


Figure 4-2 Self-organizing Map (vector).

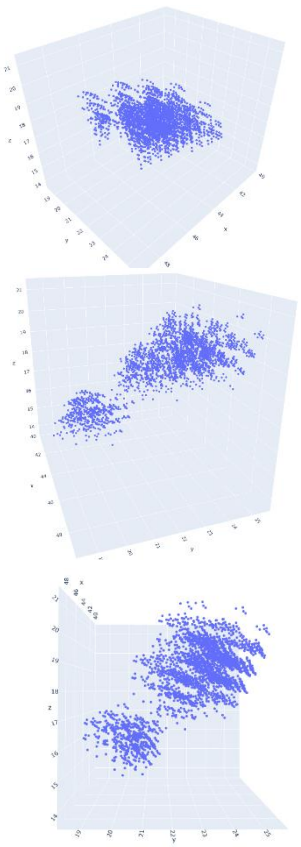


Figure 4-6 Reduced dimension by vector-based Dimensionality reduction

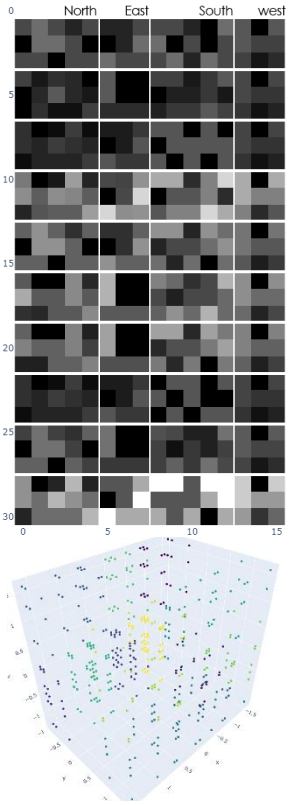


Figure 4-11 K-means Clustering (PCA).

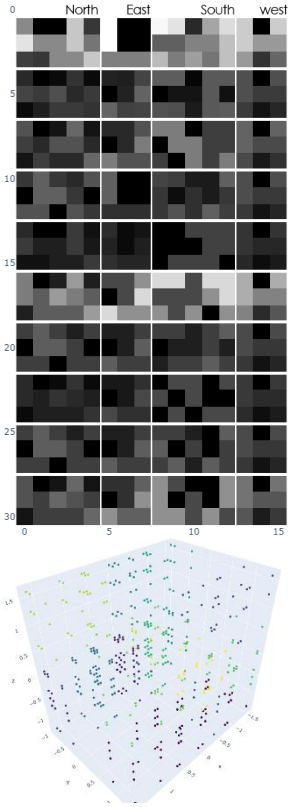


Figure 4-10 Agglomerative clustering (PCA).

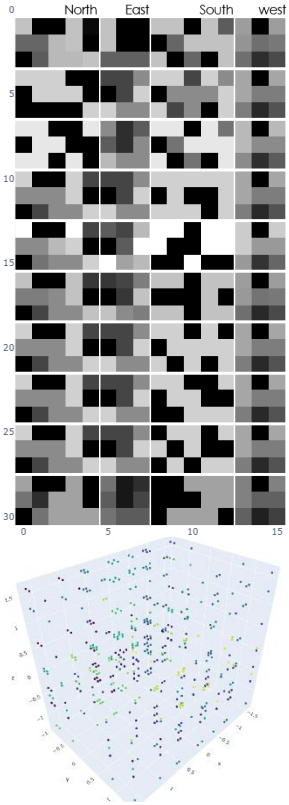


Figure 4-9 Meanshift clustering (PCA).

Figure 4-7 Affinity Propagation (PCA).

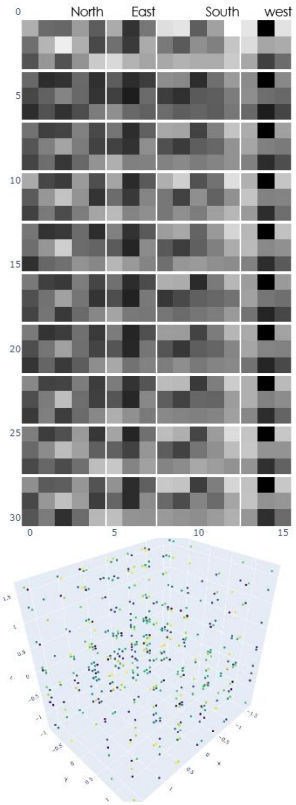


Figure 4-8 Self-organizing Map (PCA).

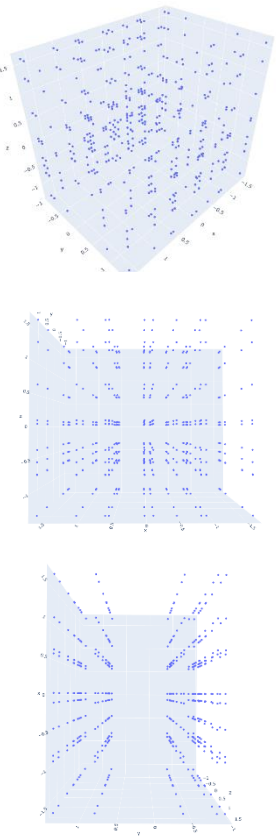


Figure 4-12 Reduced dimension by PCA

Dimensionality reduction method	Clustering method	Cluster count	See figure	Comments
Vector-based dimensionality reduction method	K-means clustering	15	Figure 4-5	Showed slight variations in generated pixelated image for different clusters. However, it should be noted that the output is sensitive to the centroid initialization and the number of clusters chosen.
	Agglomerative clustering	15	Figure 4-4	With this clustering method, the data points were clustered substantially better than with others. This method's effectiveness would be more dependent on the dimensionality reduction technique used before it.
	Meanshift clustering	120	Figure 4-3	With a bandwidth value of 0.3 Meanshift clustering method generated 120 clusters, the pixelated image shows only first 10 images. Some of the clusters are extremely dark which refers higher variations in one single cluster.
	Affinity propagation	2000	Figure 4-1	As data points were not densified properly, Affinity propagation clustering failed to make effective clusters, resulting 2000 clusters in total, which refers that most of the clusters have one single data points.
	Self-organizing maps	15	Figure 4-2	The pixelated images of different clusters look almost the same. The colored images of clusters further demonstrate the inappropriateness of this approach in this case.

4.1.2 Principle component analysis (PCA) results

The application of Principal Component Analysis (PCA) resulted in the positioning of data points in a three-dimensional grid with a rectangular shape. However, it was observed that there were uneven gaps between different grids, indicating a lack of uniformity. These gaps are believed to be a result of insufficient variations in the data frame, as it represents only a small subset of all possible variations. All the data points are sparsely distributed. Due to the inability of the data point arrangement to accurately reflect performance proximity, the subsequent clustering methods proved to be ineffective in grouping data points with similar simulated outcomes.

Dimensionality reduction method	Clustering method	Cluster count	See figure	Comments
Principal Component Analysis (PCA)	K-means clustering	15	Figure 4-11	The generated pixelated image of the clusters exhibited blurriness and darkness, indicating a large number of variations within the clusters. It is hypothesized that by increasing the number of clusters in the K-means algorithm, the resulting pixelated image would become less blurry and more distinct.
	Agglomerative clustering	15	Figure 4-10	Similar to K-means clustering, pixelated image is blurry and dark refers to clusters with large variations.
	Meanshift clustering	90	Figure 4-9	In spite of having sparsely distributed data points, the Meanshift clustering method somehow generated effective clusters with a bandwidth value of 0.3. Generated pixelated image of clusters is less blurry.

	Affinity propagation	2500	Figure 4-7	This method considered each datapoints as one single cluster which resulted in a very clear pixelated image of first ten clusters. The team suggests not to misinterpret pixelated images. They should not be super clear or extremely blurry, but something in between.
	Self-organizing maps	15	Figure 4-8	Different clusters' pixelated images resemble one another almost exactly.

4.1.3 Position-based dimensionality reduction results

As principal component analysis (PCA) is run after running this method, the resulting outcome from this method is similar to that of PCA. The data points are arranged linearly into 10 lines, and these lines are clustered into two main groups. The distances between these virtual lines are approximately equal within each group. Furthermore, the data points within each line appear to be evenly distributed. Different data points tend to have the position in the reduced dimension.

Dimensionality reduction method	Clustering method	Cluster count	See figure	Comments
Position-based dimensionality reduction method	K-means clustering	15	Figure 4-18	Pixelated images of different clusters refer to moderate variations in clusters, while the east and west façade remains almost the same in all clusters.
	Agglomerative clustering	15	Figure 4-17	This method provided moderate number of variations in clusters; few clusters are comparatively darker which refers to large variations.
	Meanshift clustering	80	Figure 4-16	As the number of the clusters increased the generated pixelated image is more distinct than others.
	Spectral clustering	15	Figure 4-15	Displayed good number of variations in all clusters, Pixelated image for north façade is clearer than other three facades.
	Affinity propagation	150	Figure 4-14	due to some data points occupying the same position in the reduced dimensions, Affinity Propagation tends to cluster all of these data points together.
	Self-organizing maps	15	Figure 4-13	Generated pixelated image is extremely blurry, which refers the inappropriateness of this method in this case

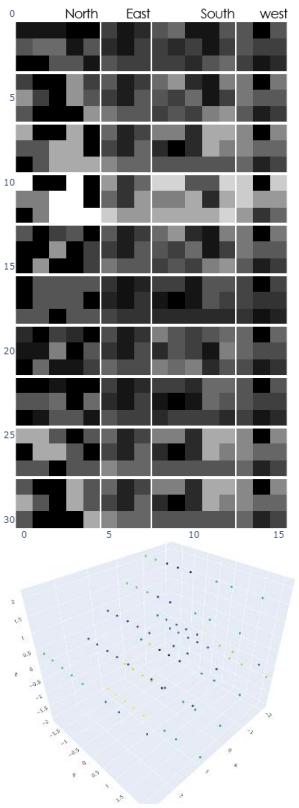


Figure 4-18 K-means Clustering (Position).

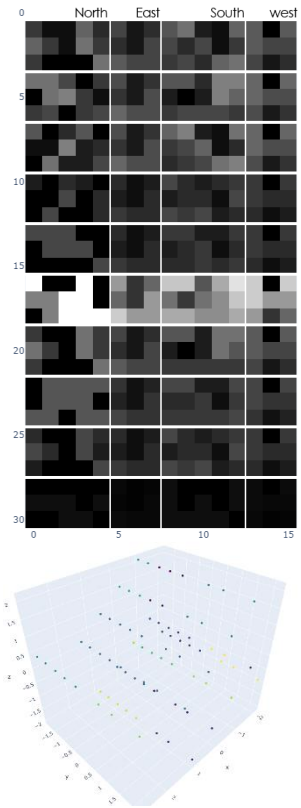


Figure 4-17 Agglomerative clustering (Position).

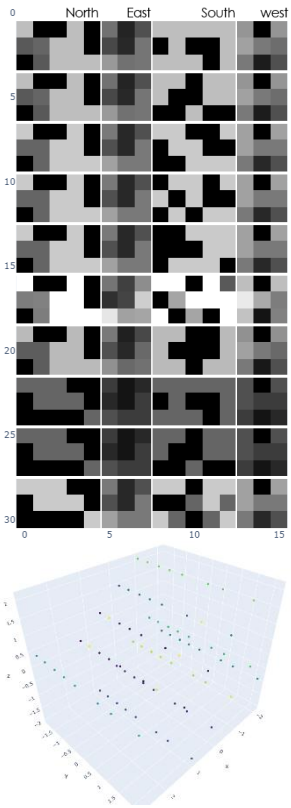


Figure 4-16 Meanshift clustering(Position).

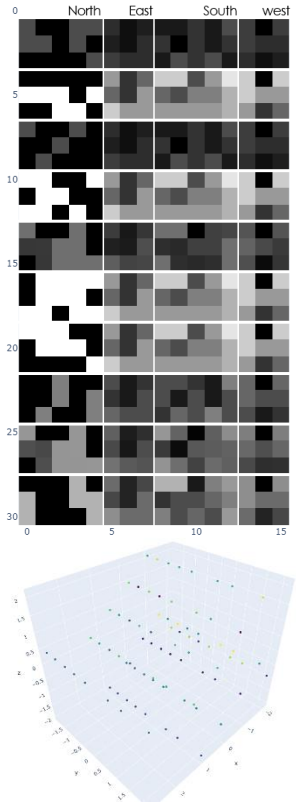


Figure 4-15 Spectral clustering(Position).

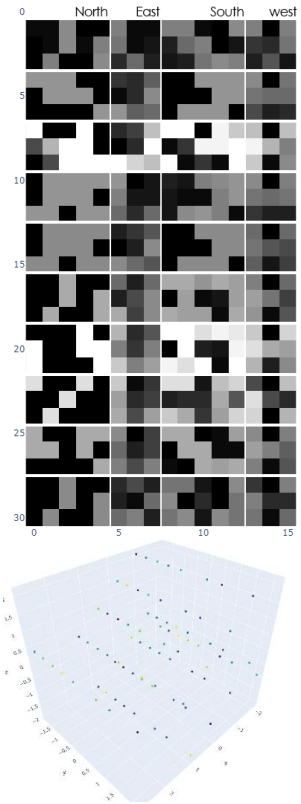


Figure 4-14 Affinity Propagation(Position).

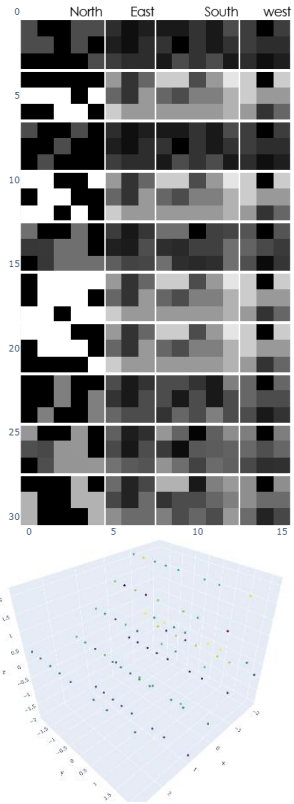


Figure 4-13 Self-organizing Map(Position).

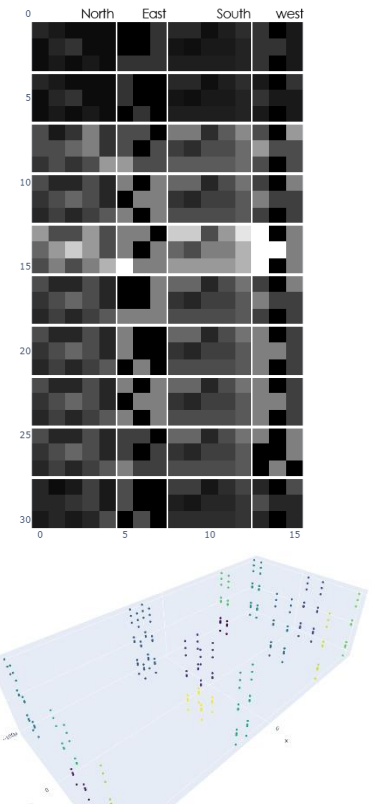


Figure 4-20 K-means Clustering(Angle).

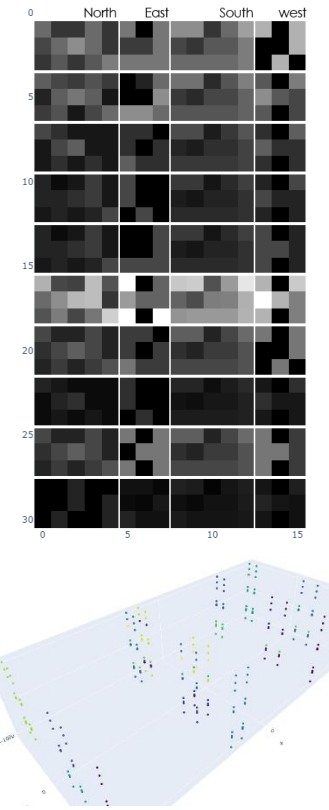


Figure 4-21 Agglomerative clustering(Angle).

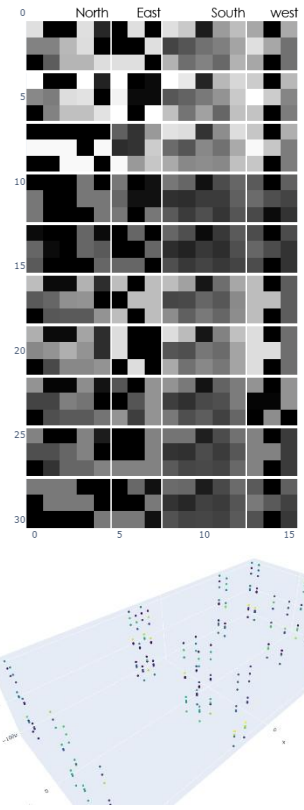


Figure 4-19 Meanshift clustering(Angle)..

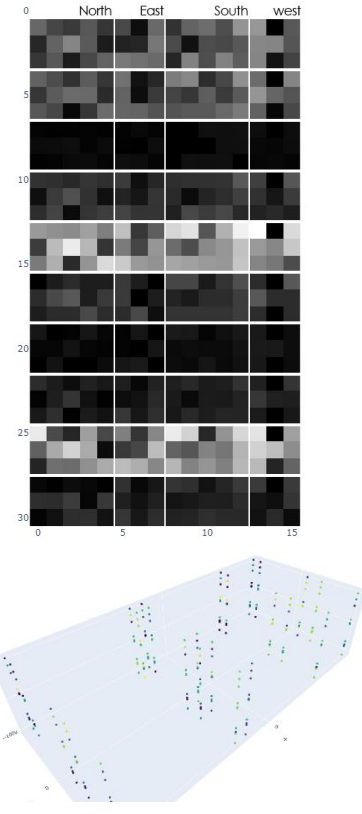


Figure 4-24 Spectral clustering(Angle).

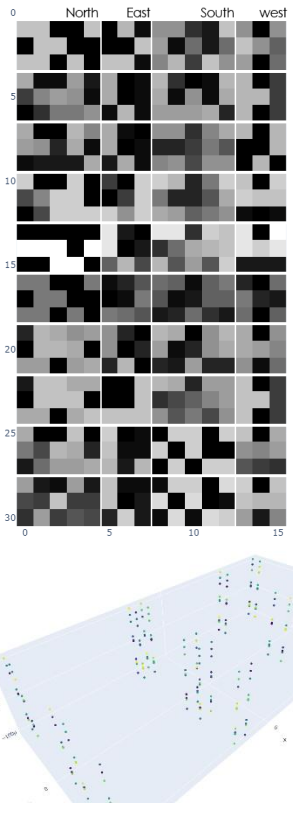


Figure 4-23 Affinity Propagation(Angle).

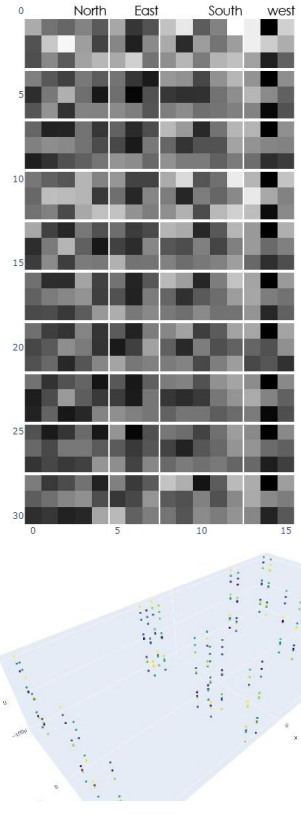


Figure 4-22 Self-organizing Map(Angle).

4.1.4 Angle-based dimensionality reduction results

In the reduced dimension, the data points are sparsely distributed in a 3-dimensional grid. They appear to be arranged in horizontal lines, with approximately 25 lines forming a 5x5 grid. The uneven gaps between different rows are likely a result of limited variation in the initial data frame. Additionally, some data points appear to occupy the same position in the reduced dimension.

Dimensionality reduction method	Clustering method	Cluster count	See figure	Comments
Angle-based dimensionality reduction method	K-means clustering	15	Figure 4-20	The pixelated images of different clusters exhibit extreme darkness and blurriness. When visualizing all ten clusters, it appears that they have the same generated image, albeit with slight variations in brightness.
	Agglomerative clustering	15	Figure 4-21	Slight variations are noticeable in the pixelated images of the north, east, and west facades across different clusters. However, the pixelated image of the south facade for all clusters appears to be the same, with only minor changes in brightness.
	Meanshift clustering	90	Figure 4-19	The produced pixelated image shows variations in the north, east, and west facades, indicating moderate diversity among the clusters. However, the south facade appears to be the same for all clusters, with no noticeable variations.
	Spectral clustering	15	Figure 4-24	The images for different clusters exhibit high contrast, with some clusters appearing extremely dark and others being bright. The clusters with darker images indicate a larger variation within them.
	Affinity propagation	170	Figure 4-23	As the number of clusters increased the generated image refers to moderate amount of distribution in each cluster.
	Self-organizing maps	15	Figure 4-22	Generated pixelated image is extremely blurry, which refers the inappropriateness of this method in this case

4.1.5 t-Distributed Stochastic Neighbor Embedding (TSNE) results

As non-linear dimensionality reduction can capture complex relationship in between different data points TSNE generated one of the most satisfying outcomes among all frameworks experimented for design space 00. The data points are arranged in a large spherical distribution, with smaller clusters formed within this larger group (see Figure 4-30). The data points within each cluster are closely positioned, while maintaining a significant distance from other clusters, making it easy for clustering algorithms to identify and distinguish each cluster. This outcome serves as a strong foundation for density-based clustering algorithms. Based on these results, the team has decided to use TSNE as the preferred dimensionality reduction method for further investigations.

Dimensionality reduction method	Clustering method	Cluster count	See figure	Comments
t-Distributed Stochastic Neighbor Embedding (TSNE)	K-means clustering	15	Figure 4-29	The pixelated image generated by K-means is blurry, which is expected as K-means is not a density-based clustering method. Its effectiveness relies on assigning the appropriate number of clusters.

	Agglomerative clustering	15	Figure 4-28	The pixelated image for some of the clusters appears extremely dark, indicating a large number of variations within those clusters.
	Meanshift clustering	90	Figure 4-27	Meanshift, being a density-based clustering method, is effective in identifying clustered data points accurately. The pixelated image generated by Meanshift shows a moderate number of variations within the clusters. The sensitivity of Meanshift to its assigned bandwidth parameter can impact the clustering results.
	Affinity propagation	225	Figure 4-26	Although Affinity propagation produced a large number of clusters, the pixelated image generated from this method is clear, indicating moderate variations within the clusters. This suggests that Affinity propagation was able to effectively group the data points based on their similarities.
	Self-organizing maps	15	Figure 4-25	Different clusters' pixelated images resemble one another almost exactly.

4.1.6 Isomap Embedding results

Isomap positioned the data points in a roughly identifiable 3-dimensional grid. The grid is divided into large groups, with each group containing 8 horizontally distributed layers of data points (see Figure 4-31). The effectiveness of isomap embedding is further investigated due to its ability to produce a structured distribution. Based on the results, the team decided to use this method for further analysis and exploration.

Dimensionality reduction method	Clustering method	Cluster count	See figure	Comments
Isomap embedding	K-means clustering	15	Figure 4-32	The pixelated image generated by K-means clustering appears blurry, but individual images for different clusters are distinct. It is believed that selecting an appropriate number of clusters can lead to a better understanding of the data and improve the clarity of the pixelated image.
	Agglomerative clustering	15	Figure 4-33	The pixelated images for different clusters are distinct, but they appear darker compared to the images generated by K-means. This indicates the presence of a few more variations within the clusters.
	Meanshift clustering	90	Figure 4-34	Moderate variation is observed in the north and south facades for different clusters, while the pixelated image for the east and west facades appears to be the same across the clusters.
	Affinity propagation	2500	Figure 4-35	The consideration of individual data points as single clusters resulted in an ineffective outcome, particularly in reducing the number of simulations.
	Self-organizing maps	15	Figure 4-36	Different clusters' pixelated images resemble one another almost exactly.

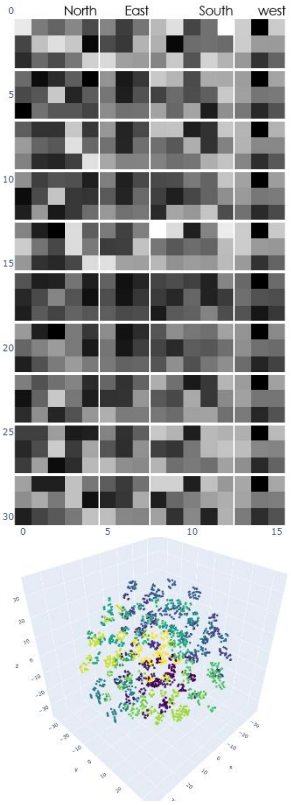


Figure 4-29 K-means Clustering(TSNE).

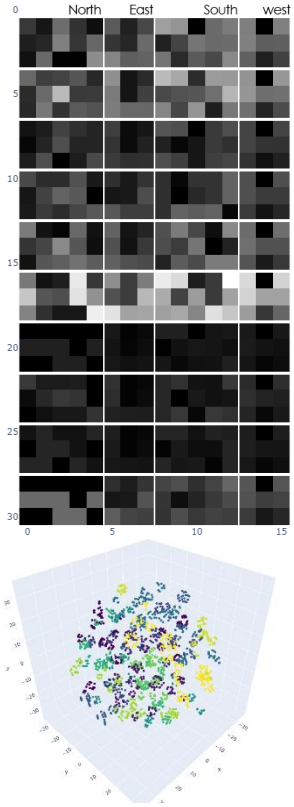


Figure 4-28 Agglomerative clustering(TSNE).

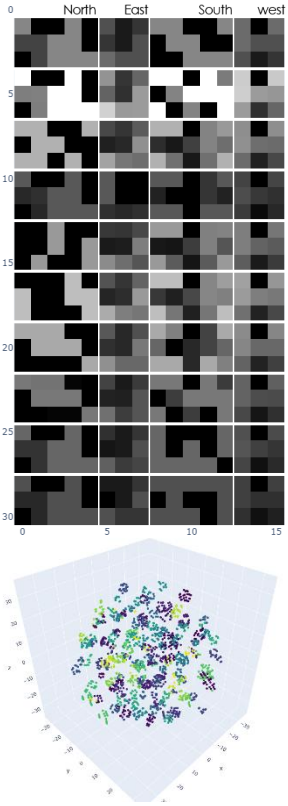


Figure 4-27 Meanshift clustering(TSNE).

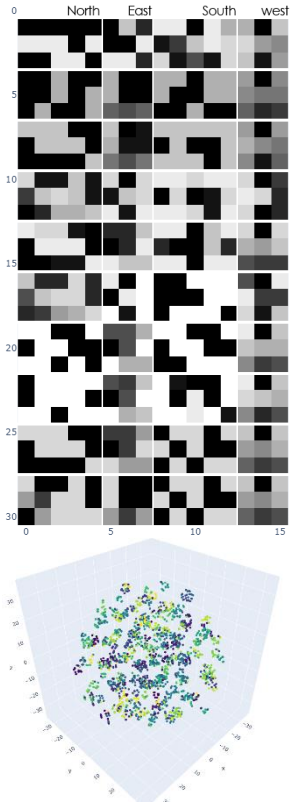


Figure 4-26 Affinity Propagation(TSNE).

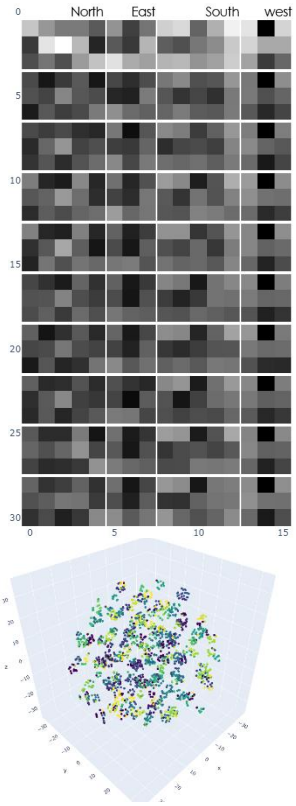


Figure 4-25 Self-organizing Map(TSNE).

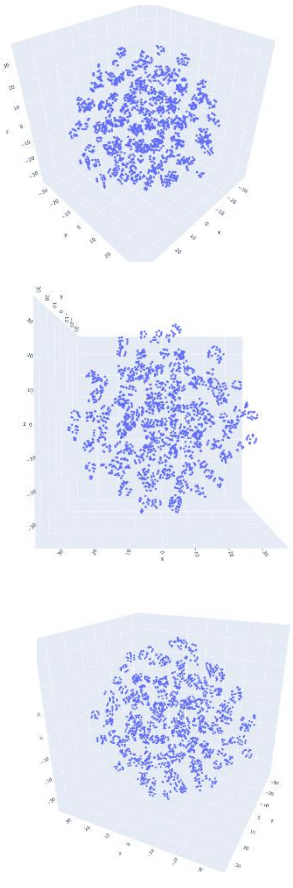


Figure 4-30 Reduced Dimension by TSNE

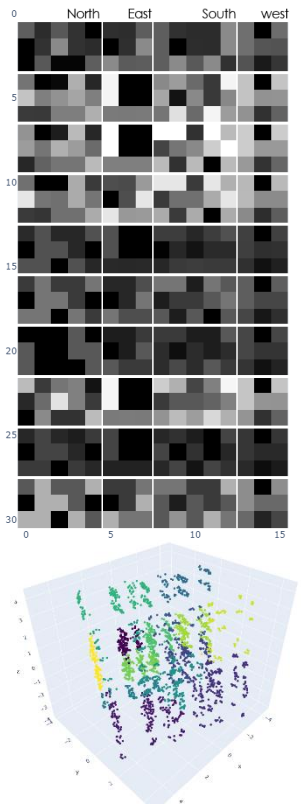


Figure 4-32 K-means Clustering(Isomap).

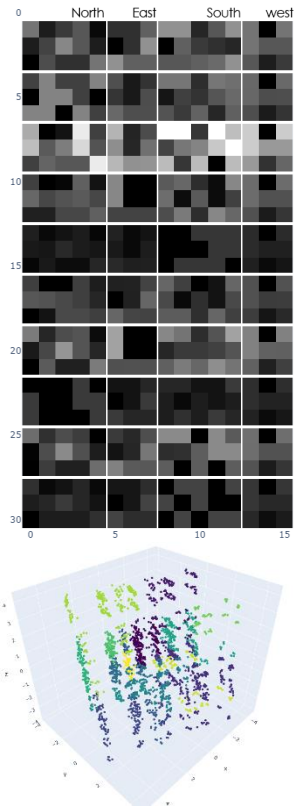


Figure 4-33 Agglomerative clustering(Isomap).

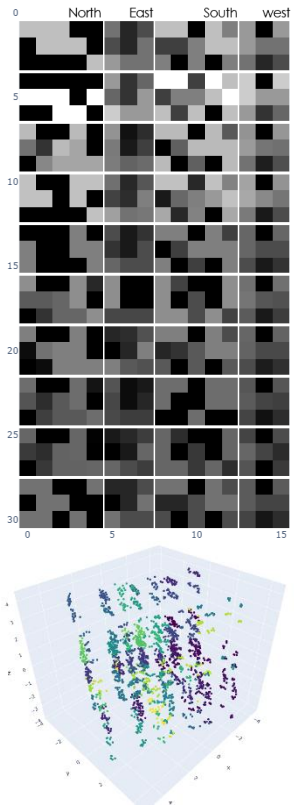


Figure 4-34 Meanshift clustering(Isomap).

Figure 4-35 Affinity Propagation(Isomap).

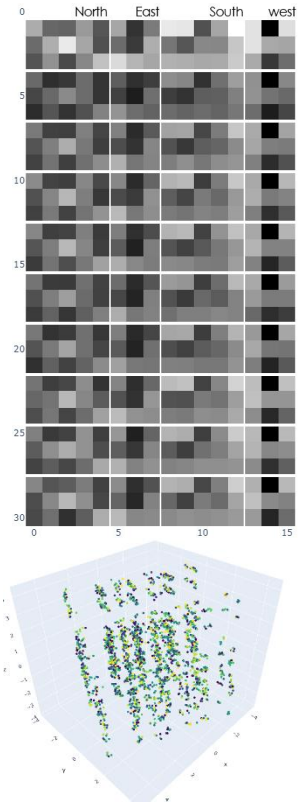


Figure 4-36 Self-organizing Map(Isomap).

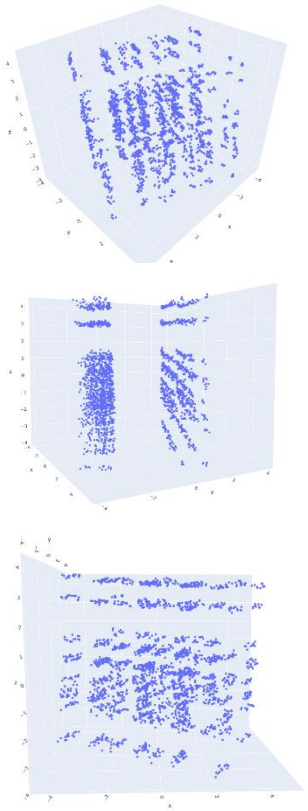


Figure 4-31 Reduced dimension by Isomap.

4.1.7 Multidimensional scaling Embedding (MDS) results

The data points in the reduced dimension are observed to be irregularly distributed within a large sphere (see Figure 4-37). However, the presence of voids between layered data points suggests a lack of variation in the initial data frame, resulting in incomplete filling of these gaps. Despite positioning the data points in a lower dimension, the outcomes of this method did not meet our expectations in terms of facilitating further clustering methods. The uniform distribution within the sphere did not provide meaningful insights or effective clustering patterns.

Dimensionality reduction method	Clustering method	Cluster count	See figure	Comments
Multidimensional scaling Embedding (MDS)	K-means clustering	15	Figure 4-42	The pixelated image generated by K-means clustering appears blurry.
	Agglomerative clustering	15	Figure 4-41	The pixelated image for this method is blurry and darker than k-means.
	Meanshift clustering	105	Figure 4-40	Despite providing with many clusters the generated pixelated image is blurry.
	Affinity propagation	2100	Figure 4-39	The majority of the clusters consist of only one data point, indicating the failure to effectively group similar data points.
	Self-organizing maps	15	Figure 4-38	Different clusters' pixelated images resemble one another almost exactly.

4.1.8 Locally Linear Embedding results

In this process, the data points are positioned in a scattered manner, with a concentration near the center and some points located far apart (see Figure 4-43). The visualized scatter plot reveals a limited number of data points, indicating that multiple points share the same position in the reduced dimension. However, the reasons behind the occurrence of data points that are widely separated are not clear.

Dimensionality reduction method	Clustering method	Cluster count	See figure	Comments
Locally Linear Embedding	K-means clustering	15	Figure 4-48	The pixelated image produced by K-means clustering exhibits blurriness, and certain clusters generate extremely dark images, indicating a significant number of variations within those clusters.
	Agglomerative clustering	15	Figure 4-47	Generated pixelated image is extremely dark for most of the clusters indicating many variations in them.
	Meanshift clustering	100	Figure 4-46	The generated pixelated image appears to be relatively less dark, which can be attributed to the utilization of a larger number of clusters.
	Affinity propagation	1850	Figure 4-45	The vast majority of the clusters only include one data point, which shows that related data points were not properly grouped together.
	Self-organizing maps	15	Figure 4-44	Different clusters' pixelated images resemble one another almost exactly.

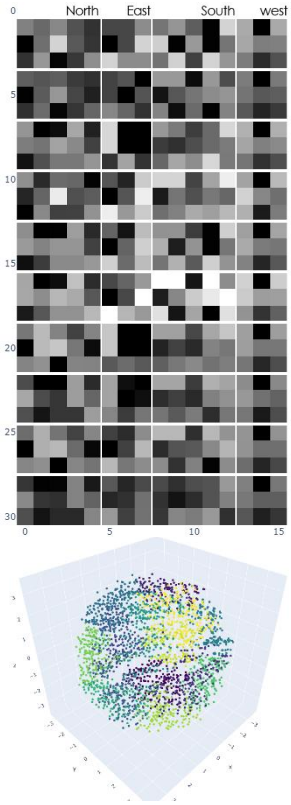


Figure 4-42 K-means Clustering(MDS).

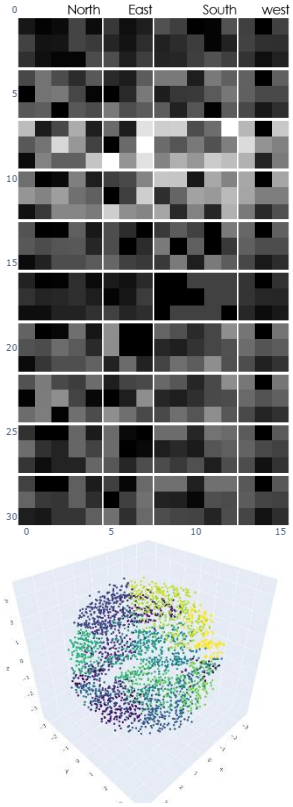


Figure 4-41 Agglomerative clustering(MDS).

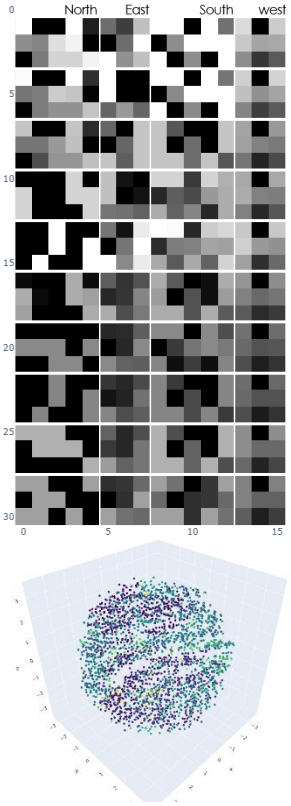


Figure 4-40 Meanshift clustering(MDS).

Figure 4-39 Affinity Propagation(MDS).

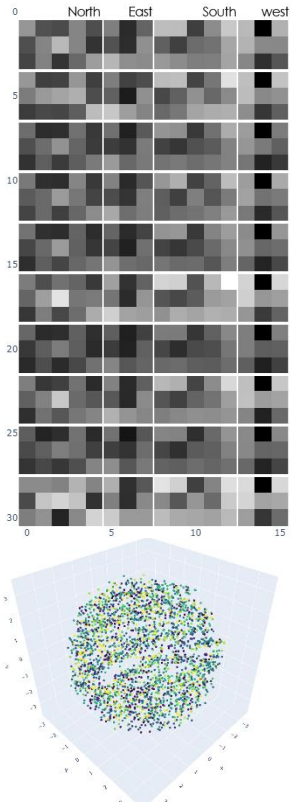


Figure 4-38 Self-organizing Map(MDS).

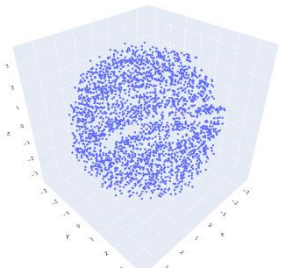


Figure 4-37 Reduced dimension by Multidimensional scaling Embedding (MDS)

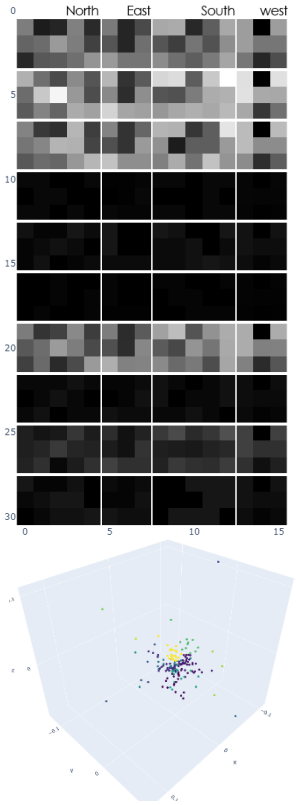


Figure 4-48 K-means Clustering(LLE).

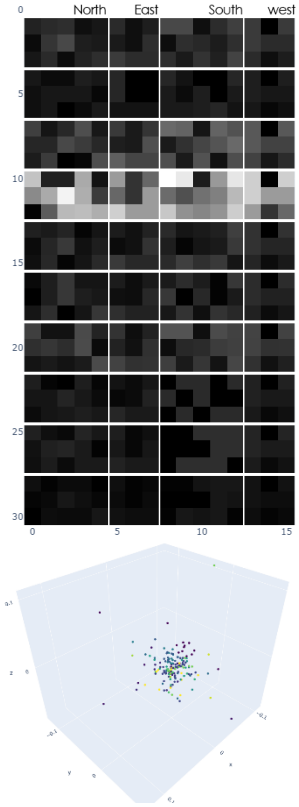


Figure 4-47 Agglomerative clustering(LLE).

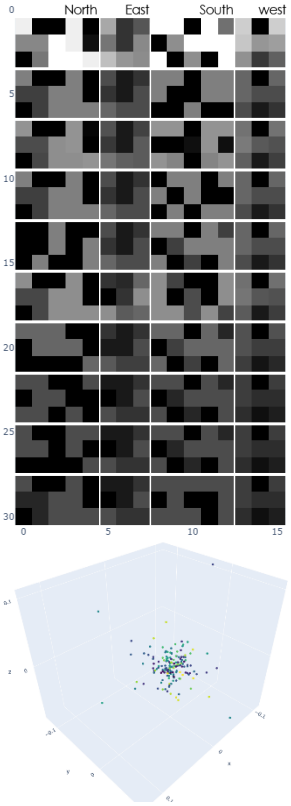


Figure 4-46 Meanshift clustering(LLE).

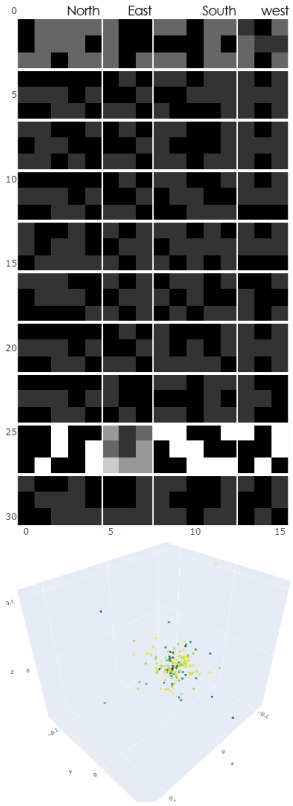


Figure 4-45 Affinity Propagation(LLE).

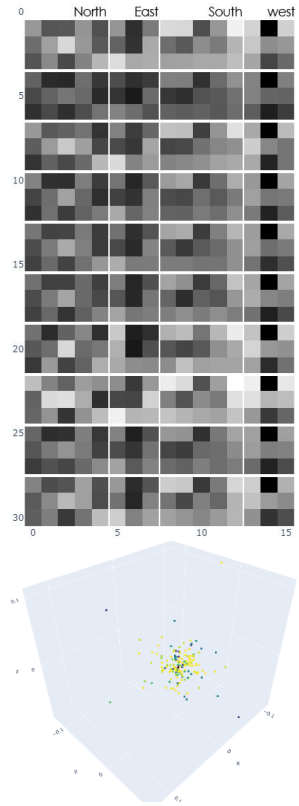


Figure 4-44 Self-organizing Map(LLE).

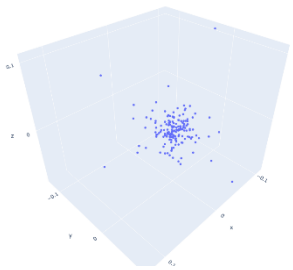


Figure 4-43 Reduced dimension by Locally Linear Embedding (LLE)

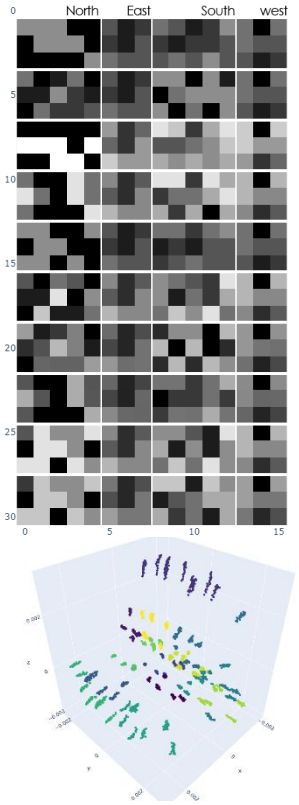


Figure 4-54 K-means Clustering(SE).

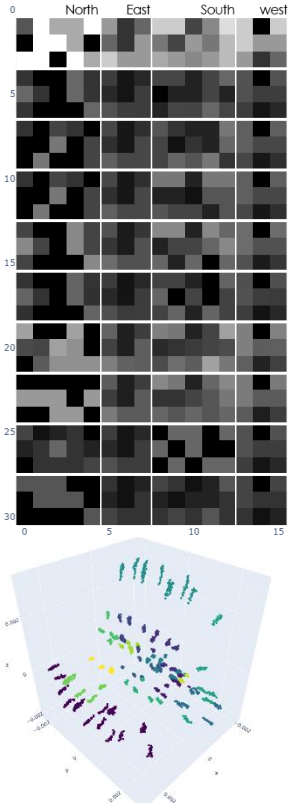


Figure 4-52 Agglomerative clustering(SE).

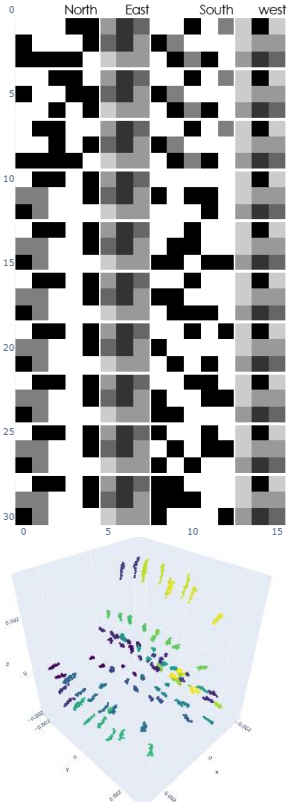


Figure 4-53 Meanshift clustering(SE).

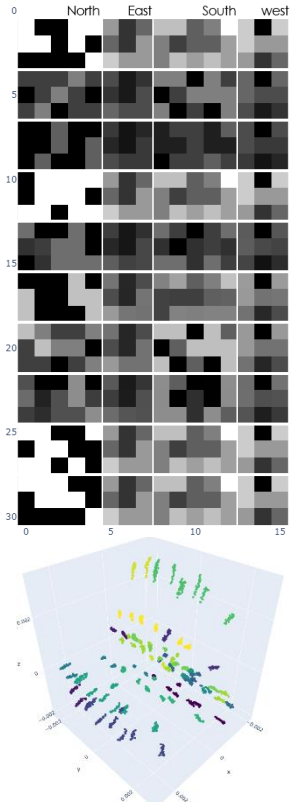


Figure 4-51 Spectral clustering(SE).

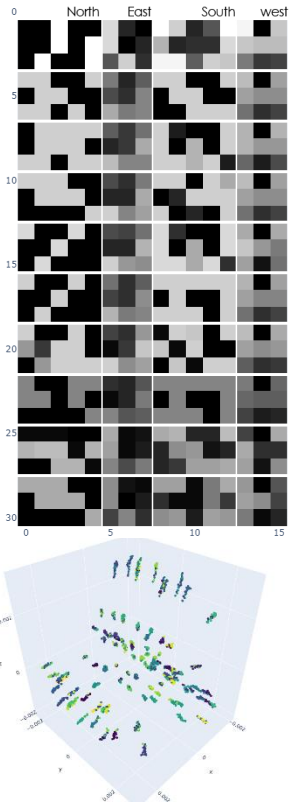


Figure 4-50 Affinity Propagation(SE).

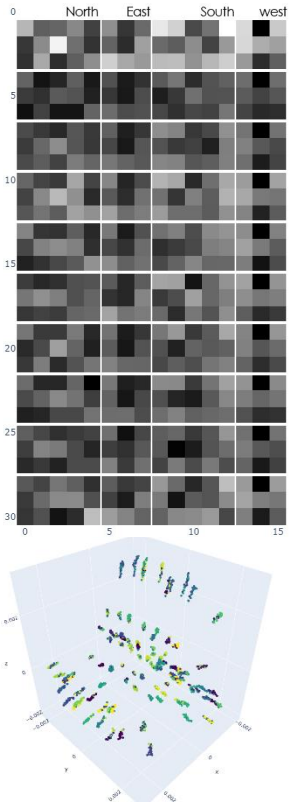


Figure 4-49 Self-organizing Map(SE).

4.1.9 Spectral Embedding results

Spectral embedding proved to be highly effective in organizing the data points in the lower dimension. The data points formed identifiable clusters, with each cluster maintaining a distinct distance from others. This arrangement is suitable for density-based clustering methods. However, it is worth noting that while spectral embedding performed well for the current data structure, its effectiveness may vary for datasets with different structures. The team decided to further explore the potential of spectral embedding in subsequent investigations.

Dimensionality reduction method	Clustering method	Cluster count	See figure	Comments
Spectral embedding (SE)	K-means clustering	15	Figure 4-54	The pixelated image produced by K-means clustering appear blurry. Selecting an optimal number of clusters is believed to enhance the understanding of the data and result in a clearer pixelated image.
	Agglomerative clustering	15	Figure 4-52	The pixelated image is blurry and darker than K-means. However, the effectiveness of this method can be justified by selecting proper number of clusters.
	Meanshift clustering	75	Figure 4-53	Meanshift clustering demonstrates the ability to capture an appropriate number of clusters using a bandwidth of 0.3. Generated pixelated image explains very few variations in different clusters which is not clear.
	Spectral clustering	15	Figure 4-51	Spectral clustering serves as a subsequent step to spectral embedding, and it effectively categorizes the clustered data points.
	Affinity propagation	145	Figure 4-50	The generated pixelated image appear to be clear indicating moderate number of variations in each clusters.
	Self-organizing maps	15	Figure 4-49	Different clusters' pixelated images resemble one another almost exactly.

4.1.10 No dimensionality reduction

The investigated clustering method without any prior dimensionality reduction method. However, relying solely on the pixelated images to assess the effectiveness of the clustering methods poses challenges. Without visualizing the data in the lower-dimensional space, it becomes difficult to gain a comprehensive understanding of the clustering outcomes. The explanations provided so far are based on visual interpretations of the pixelated images.

Dimensionality reduction method	Clustering method	Cluster count	See figure	Comments
No dimensionality reduction	K-means clustering	15	Figure 4-55	The pixelated image generated by K-means clustering appears blurry, but images for different clusters are distinct.
	Agglomerative clustering	15	Figure 4-58	The pixelated image is blurry while different clusters have distinct image outcome.
	Meanshift clustering	80	Figure 4-59	The generated pixelated image refers to vary few variations in different clusters.

	Spectral clustering	15	Figure 4-60	Some clusters in the generated images appear extremely dark, while others have clear and distinct outcomes.
	Affinity propagation	140	Figure 4-57	Despite of taking huge amount of time to run this method, the generated image outcome shows moderate amount of variation in each clusters.
	Self-organizing maps	15	Figure 4-56	The generated pixelated image is extremely blurry, but different clusters exhibit distinct outcomes.

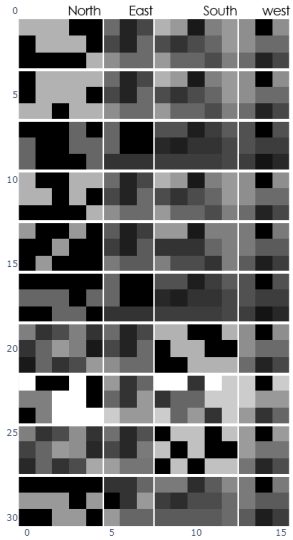


Figure 4-55 K-means Clustering (No dimensionality Reduction).

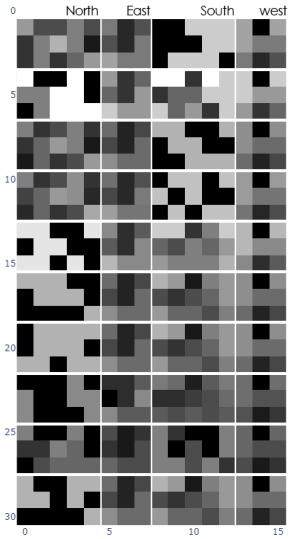


Figure 4-58 Agglomerative clustering (No Dimensionality Reduction).

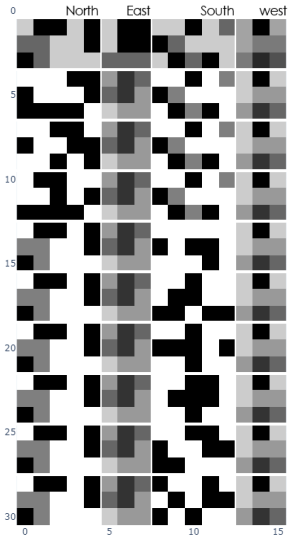


Figure 4-59 Meanshift clustering (No Dimensionality Reduction).

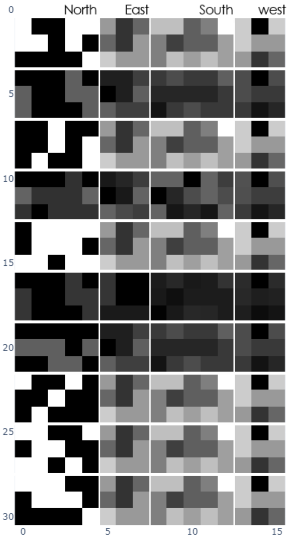


Figure 4-60 Spectral clustering (No Dimensionality Reduction).

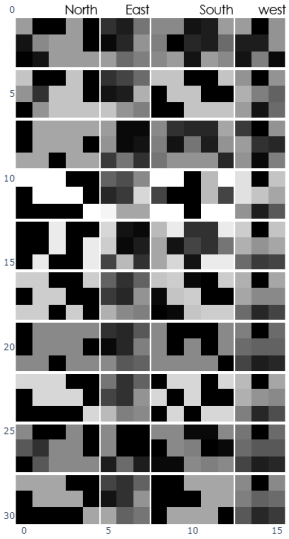


Figure 4-57 Affinity Propagation (No Dimensionality Reduction).

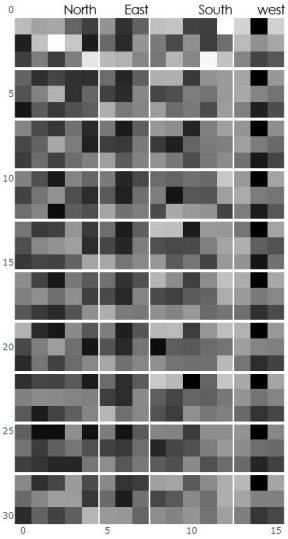


Figure 4-56 Self-organizing Map (No Dimensionality Reduction).

4.1.11 Comparison between different dimensionality reductions:

The team utilized 3-dimensional scatter plots and pixelated images of clusters to gain insights into how effectively different frameworks group similar design options. However, the true measure of their effectiveness lies in comparing their "model error vs cluster count" Pareto frontiers. After analyzing the scatter plots, the team decided to focus on three dimensionality reduction methods: t-distributed stochastic neighbor embedding (t-SNE), isomap embedding, and spectral embedding, followed by Meanshift clustering. These selected methods helped to consolidate similar data points, with Meanshift clustering being particularly advantageous as a density-based clustering approach. The number of clusters can be controlled by adjusting the bandwidth values.

To construct the Pareto frontiers, the team varied the number of extracted clusters to evaluate the tradeoff between cluster count and model error. Among the three extracted Pareto distributions, spectral embedding followed by Meanshift clustering achieved the lowest model error with the fewest clusters (see Figure 4-61). For example, a model error of 0.106 can be achieved with only 197 clusters, demonstrating that the performance outcome of the entire design space can be approximated by simulating just 197 data points.

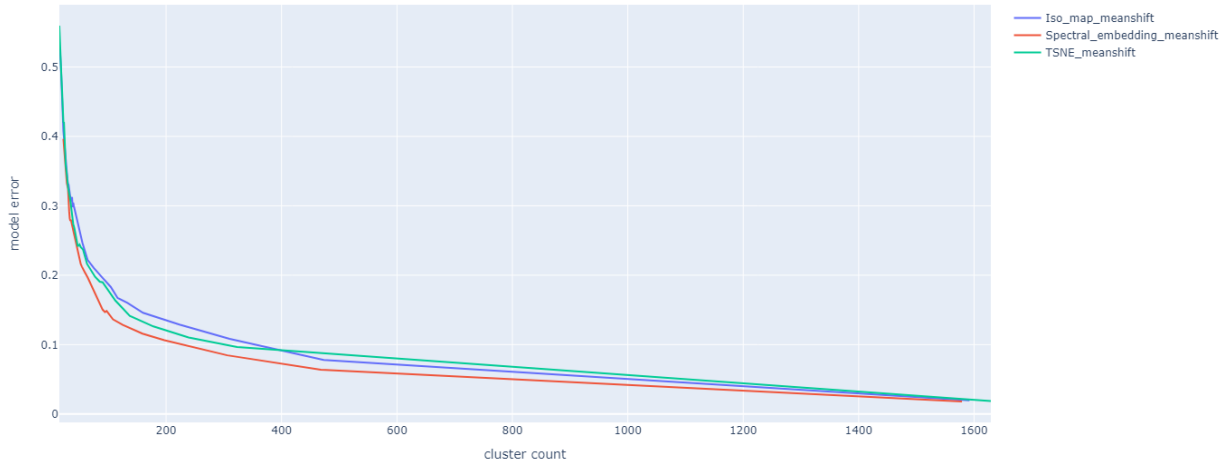


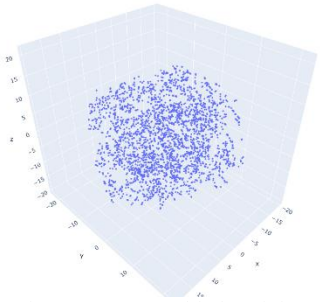
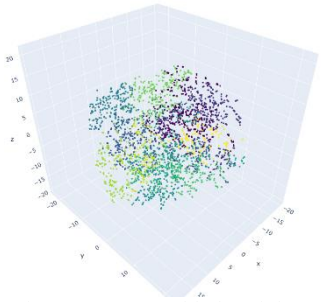
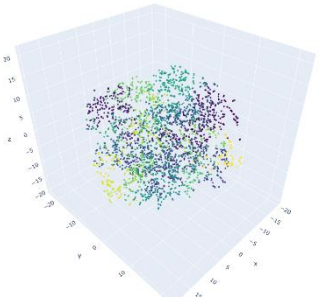
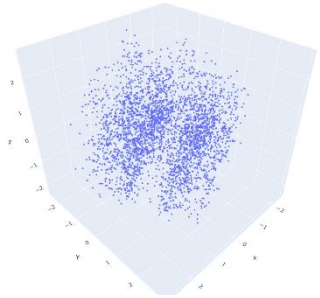
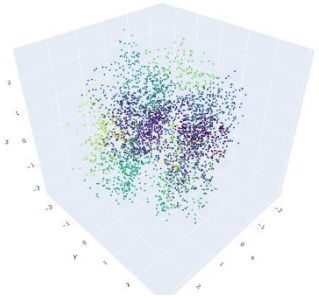
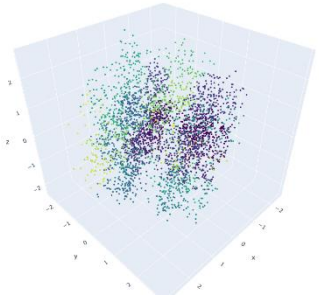
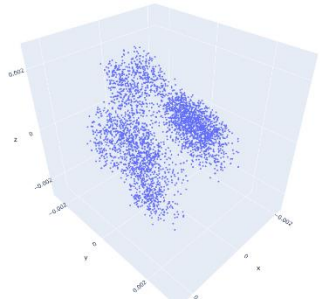
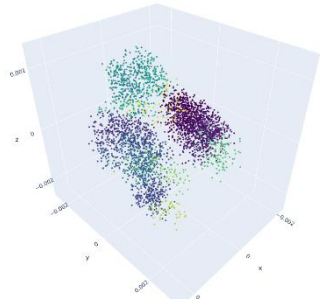
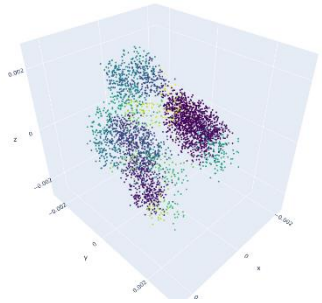
Figure 4-61 Model error vs cluster count Pareto distribution for Design Space 00

4.2 Framework outcomes of Design Space 01:

The results of the frameworks are organized into distinct sections for each adding weights method. Within each section, the corresponding images are arranged in sequential order to facilitate easy reference and analysis. Furthermore, a table is included for each adding weights method, serving as a guide to the relevant figures that readers should consult for a more comprehensive understanding of the outcomes. If spectral embedding dimensionality reduction method is absent in any of these tables, it indicates that it was unsuccessful in producing meaningful outcomes within the respective framework.

4.2.1 No added weights results

When the data frame is not weighted, it poses challenges for the dimensionality reduction method and clustering algorithms to effectively group data points with similar performance outcomes. As a result, the model error tends to be significantly higher in frameworks that utilize unweighted data frames. However, among the three frameworks that were experimented, spectral embedding followed by meanshift clustering yielded a comparatively lower Pareto frontier. It achieved a model error of 24.01 with a cluster count of 59, indicating that the performance prediction for any data point could deviate by approximately 75.99% to 124.01% from its actual outcome.

		Cluster Count 15	Cluster Count 31
TSNE			
	Figure 4-62 No Added Weights, TSNE, Meanshift	Figure 4-63 No Added Weights, TSNE, Meanshift, cluster count 15	Figure 4-64 No Added Weights, TSNE, Meanshift, cluster count 31
Isomap			
	Figure 4-65 No Added Weights, Isomap, Meanshift	Figure 4-66 No Added Weights, Isomap, Meanshift, Cluster count 15	Figure 4-67 No Added Weights, Isomap, Meanshift, Cluster count 31
Spectral Embedding			
	Figure 4-68 No Added Weights, Spectral embedding, Meanshift	Figure 4-69 No Added Weights, Spectral embedding, Meanshift, cluster count 15	Figure 4-70 No Added Weights, Spectral embedding, Meanshift, cluster count 31

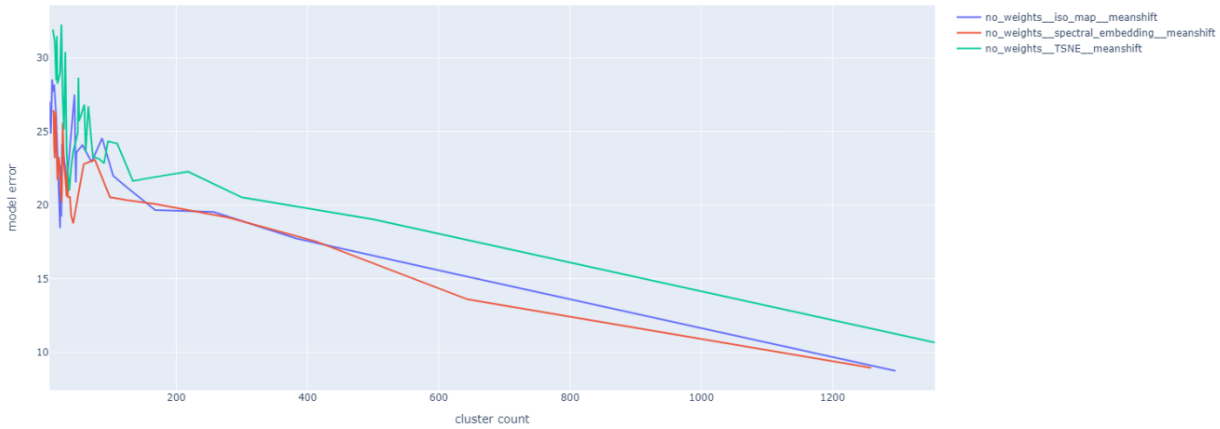


Figure 4-71 Model Error vs Cluster count Pareto Distribution, No added weights

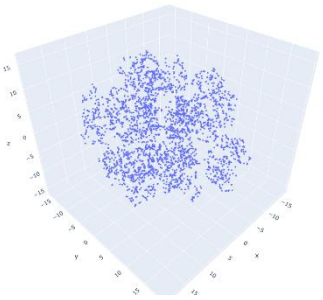
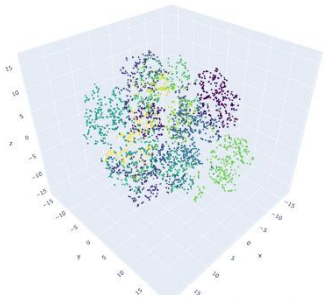
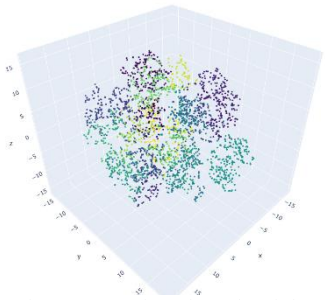
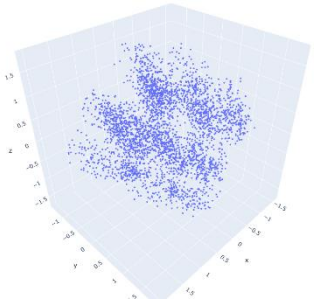
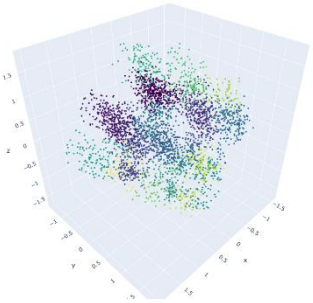
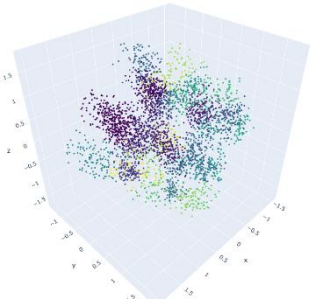
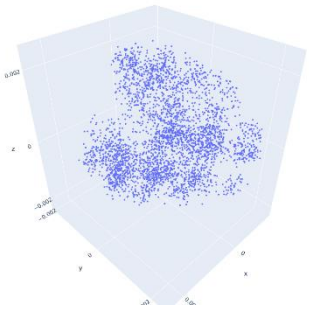
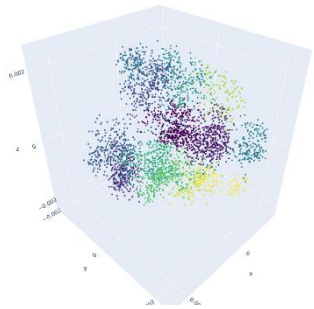
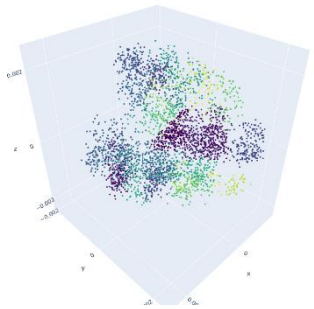
Adding weights method	Dimensionality reduction	Clustering method	See figure	comments
No added weights	TSNE	Meanshift	Figure 4-62	Data points are positioned ununiformly in a roughly spherical shape, which no distinct clusters. Model error out of this framework is the highest among other experimented frameworks for design space 01
	Isomap embedding		Figure 4-65	Data points are ununiformly distributed in roughly identifying two separate groups.
	Spectral Embedding		Figure 4-68	The data points exhibit a scattered distribution, with approximately 3-4 identifiable groups of varying shapes. Increasing the number of subdivisions leads to a lower model error.

4.2.2 Floor-based weights results

The floor-based adding weights method did not produce satisfactory results, as the model error was similar to the frameworks that used an unweighted data frame (see Figure 4-81). While it is evident that the topmost floor may require more energy for cooling compared to other floors, and the middle floor would have relatively lower energy consumption, these deviations did not significantly contribute to the grouping of data points with similar energy consumption patterns. In further investigation we would learn more about which features of the building zones are more influential in terms of performance outcomes and to enhance the accuracy of predictions.

Adding weights method	Dimensionality reduction	Clustering method	See figure	comments
Floor-based weights	TSNE	Meanshift	Figure 4-72	The data points are scattered nonuniformly in an amorphous shape, exhibiting roughly identifiable clusters that are distributed without consistency.
	Isomap embedding		Figure 4-75	Little but more organized than TSNE for why model error is also less for this framework.

	Spectral Embedding		Figure 4-78	The framework is almost similar to the isomap embedding involved approach, but with an increase in the number of clusters, the model error does not rise as significantly as in the isomap framework.

		Cluster Count 15	Cluster Count 31
TSNE			
	Figure 4-72 Floor-based weights, TSNE, Meanshift	Figure 4-73 Floor-based weights, TSNE, Meanshift, Cluster Count 15	Figure 4-74 Floor-based weights, TSNE, Meanshift, Cluster Count 29
Isomap			
	Figure 4-75 Floor-based weights, Isomap, Meanshift	Figure 4-76 Floor-based weights, Isomap, Meanshift, Cluster Count 15	Figure 4-77 Floor-based weights, Isomap, Meanshift, Cluster Count 29
Spectral Embedding			
	Figure 4-78 Floor-based weights, Spectral Embedding, Meanshift	Figure 4-79 Floor-based weights, Spectral Embedding, Meanshift, Cluster Count 15	Figure 4-80 Floor-based weights, Spectral Embedding, Meanshift, Cluster Count 29

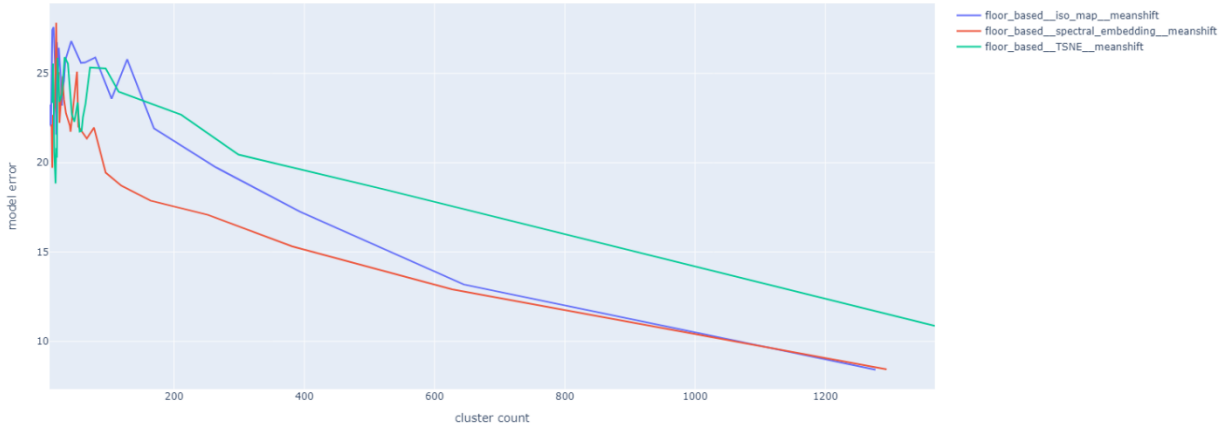
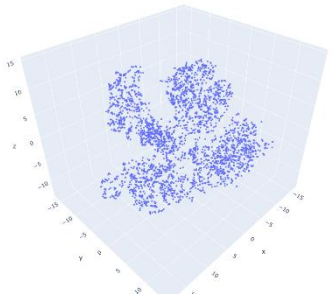
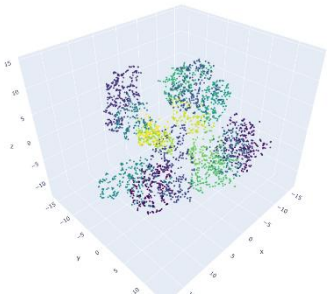
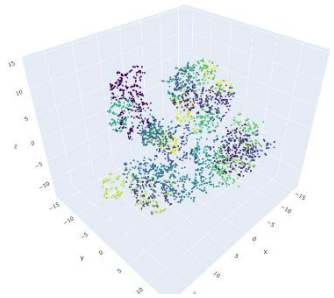
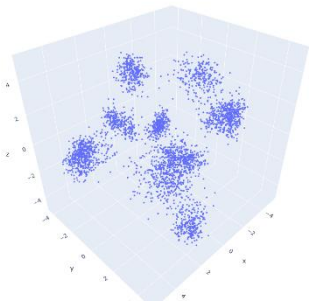
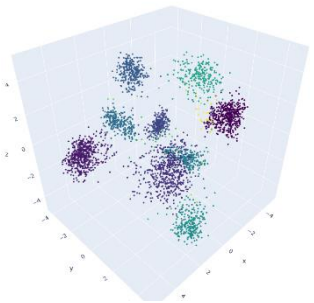
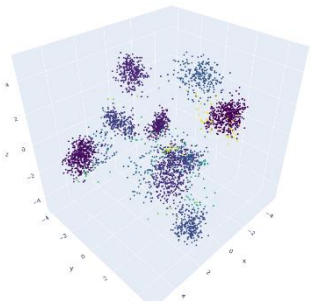
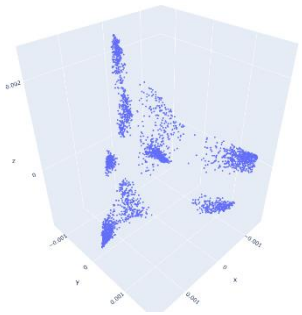
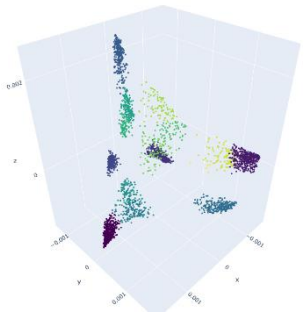
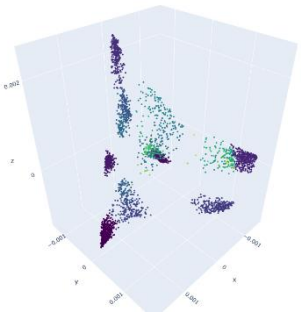


Figure 4-81 Model Error vs Cluster count Pareto Distribution, Floor-based Weights

4.2.3 Window position-based weights results

The position-based weighting method achieved partial success in helping the algorithms comprehend the impact of different apertures in various orientations on performance outcomes. Although the Pareto frontier for TSNE dimensionality reduction remains largely unchanged compared to the unweighted data framework (see Figure 4-91) , the Pareto frontiers for Isomap and spectral embedding dimensionality reduction methods show a significant decrease.

Adding weights method	Dimensionality reduction	Clustering method	See figure	comments
Position-based weights	TSNE	Meanshift	Figure 4-82	The data points exhibit an amorphous distribution, with approximately 7-8 discernible clusters that are separated by distinct gaps. The framework could not achieve significant decrease in model error.
	Isomap embedding		Figure 4-85	The data points are grouped into distinct clusters, leading to a significant reduction in model error.
	Spectral Embedding		Figure 4-88	The data points exhibit a well-clustered pattern with a few scattered distributions. The model error is slightly lower compared to the isomap embedding involved framework.

		Cluster Count 15	Cluster Count (29-30)
TSNE			
	Figure 4-82 Window position-based weights, TSNE, Meanshift	Figure 4-83 Window position-based weights, TSNE, Meanshift, Cluster Count 15	Figure 4-84 Window position-based weights, TSNE, Meanshift, Cluster Count 29
Isomap			
	Figure 4-85 Window position-based weights, Isomap, Meanshift	Figure 4-86 Window position-based weights, Isomap, Meanshift, Cluster Count 15	Figure 4-87 Window position-based weights, Isomap, Meanshift, Cluster Count 30
Spectral Embedding			
	Figure 4-88 Window position-based weights, Spectral Embedding, Meanshift	Figure 4-89 Window position-based weights, Spectral Embedding, Meanshift, Cluster Count 15	Figure 4-90Window position-based weights, Spectral Embedding, Meanshift, Cluster Count 30

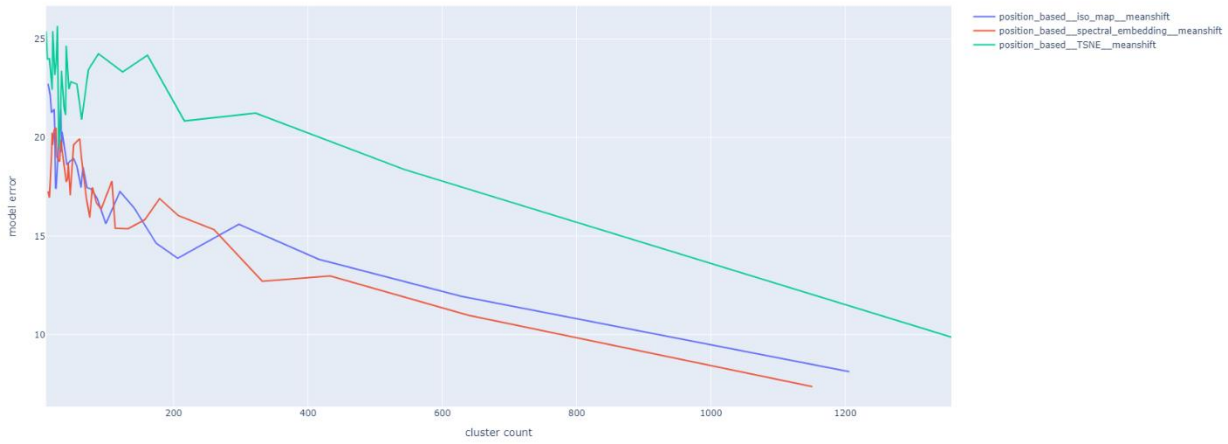


Figure 4-91 Model Error vs Cluster count Pareto Distribution, Window Position-based Weights

4.2.4 Parameter-based weights results

The "parameter-based adding weights" process proves to be effective in providing precise information on the importance of each individual parameter for the performance outcome. As a result, the generated Pareto frontier for this process closely aligns with the bottom line, resulting in a model error as low as 1.2 by clustering the data points into only 26 groups (see Figure 4-98, '*Original*' refers to *parameter-based*) . This framework has demonstrated significant success in effectively clustering the data points into cohesive groups, where data points within the same group exhibit very similar outcomes, aligning with the initial objective.

By focusing on the centroids of each cluster, we can confidently state that all other data points within the same cluster will have simulation outcomes similar to the centroid. The data points are organized into well-defined and identifiable groups, showcasing the effective utilization of density-based clustering algorithms such as Meanshift.

Adding weights method	Dimensionality reduction	Clustering method	See figure	comments
Position-based weights	TSNE	Meanshift	Figure 4-92	The data points are arranged into large, distinct clusters, with each cluster exhibiting a wide dispersion of data points.
	Isomap embedding		Figure 4-95	The data points exhibit higher density within individual groups while maintaining significant separation between the groups.

Cluster Count 14-15 Cluster Count 29

TSNE

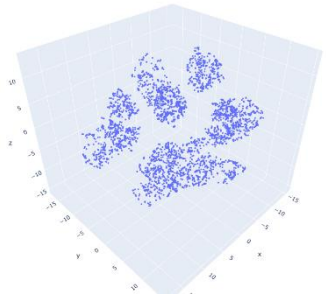


Figure 4-92 Parameter-based weights, TSNE, Meanshift

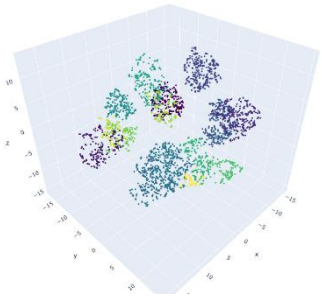


Figure 4-93 Parameter-based weights, TSNE, Meanshift, Cluster Count 15

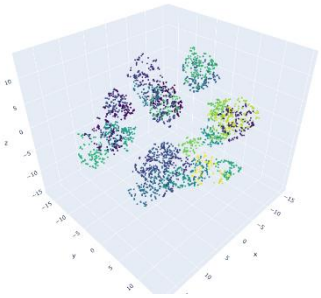


Figure 4-94 Parameter-based weights, TSNE, Meanshift, Cluster Count 29

Isomap

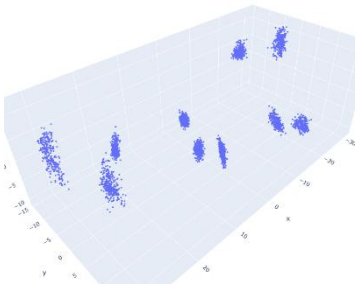


Figure 4-95 Parameter-based weights, Isomap, Meanshift

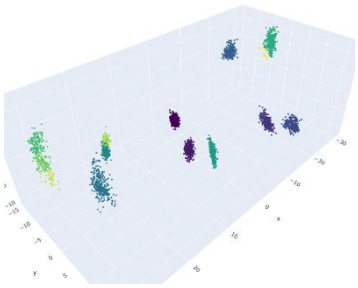


Figure 4-96 Parameter-based weights, Isomap, Meanshift, Cluster Count 14

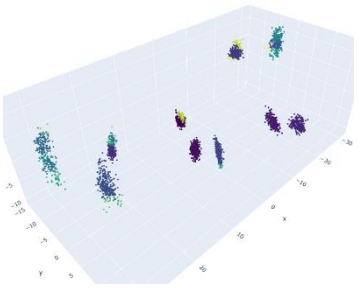


Figure 4-97 Parameter-based weights, Isomap, Meanshift, Cluster Count 29

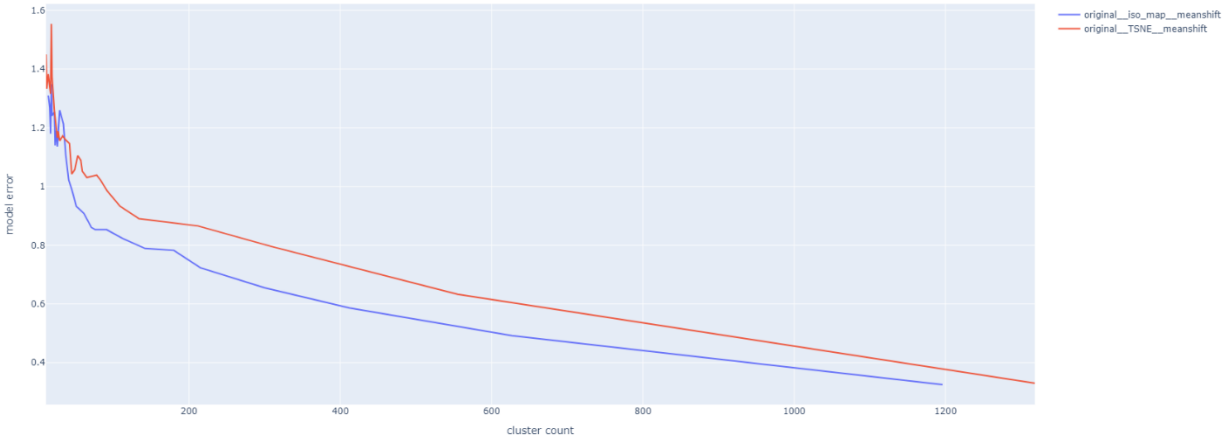


Figure 4-98 Model Error vs Cluster count Pareto Distribution, Parameter-based Weights

4.2.5 Vector-based weights results

The vector-based adding weight method outperformed the parameter-based adding weight method, primarily due to its ability to retain important information that strongly influences performance outcomes.

This method effectively created distinct clusters of data points with similar performance outcomes. Additionally, the vector-based method incorporated information about the shape and volume of the buildings, which have a significant impact on performance outcomes. The process yielded promising results not only for the current design space but also for other tested scenarios. The generated Pareto frontier is positioned below that of the parameter-based adding weight method, resulting in lower model error with a reduced number of clusters (see Figure 4-105).

Adding weights method	Dimensionality reduction	Clustering method	See figure	comments
Vector-based weights	TSNE	Meanshift	Figure 4-99	The data points exhibit a similar clustering pattern to the parameter-based adding weight method with TSNE. They form large, distinct groups with data points widely spread within each group.
	Isomap embedding		Figure 4-102	The data points demonstrate a similar clustering pattern to the parameter-based adding weight method with Isomap embedding. They form densified small groups that are separated from each other by larger gaps.

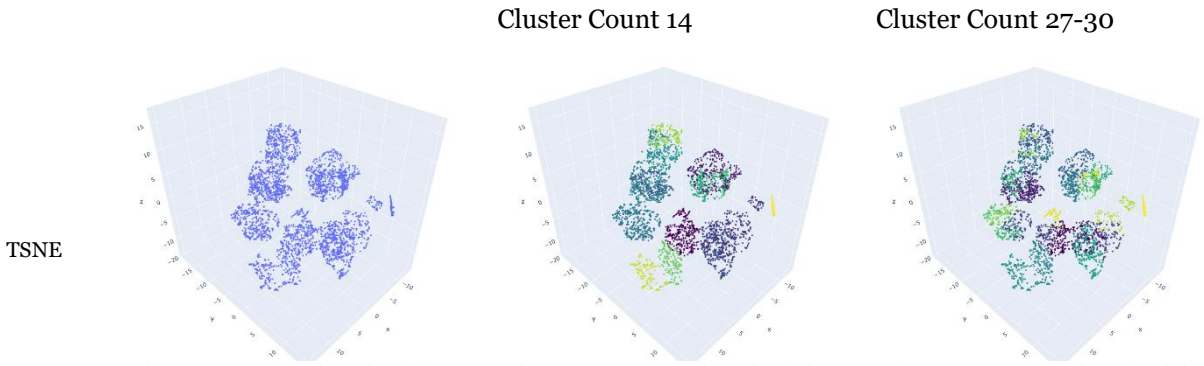


Figure 4-99 Vector-based weights, TSNE, Meanshift

Figure 4-100 Vector-based weights, TSNE, Meanshift, Cluster Count 14

Figure 4-101 Vector-based weights, TSNE, Meanshift, Cluster Count 27

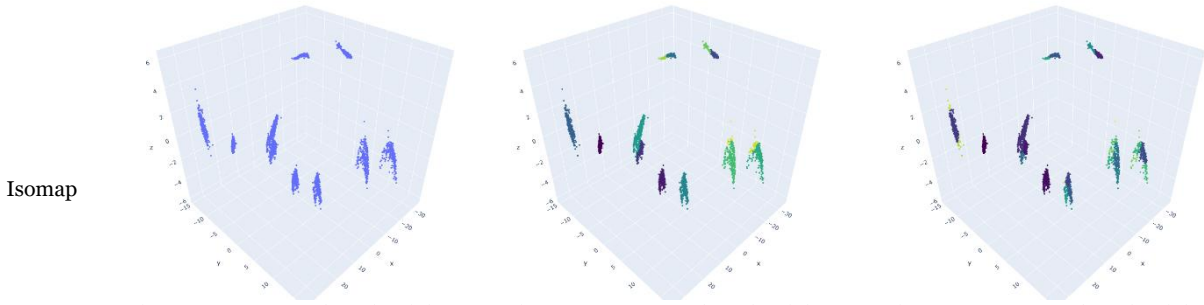


Figure 4-102 Vector-based weights, Isomap, Meanshift

Figure 4-103 Vector-based weights, Isomap, Meanshift, Cluster count 14

Figure 4-104 Vector-based weights, Isomap, Meanshift, Cluster count 30

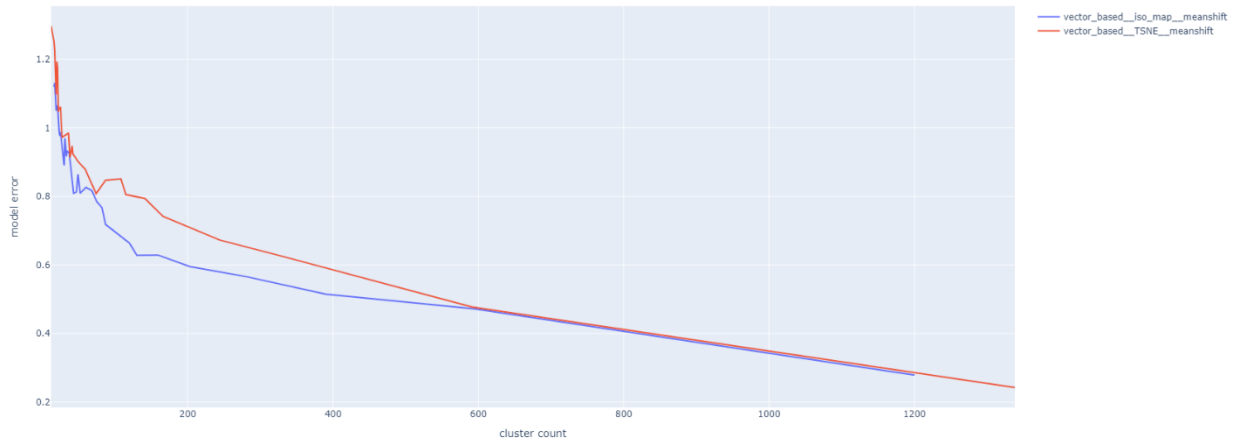


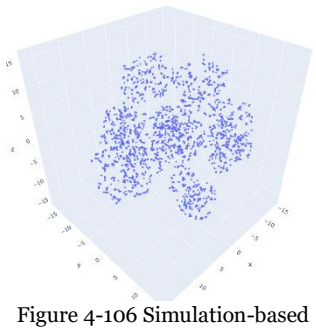
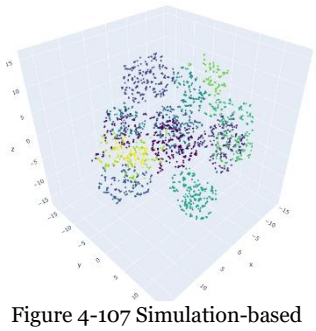
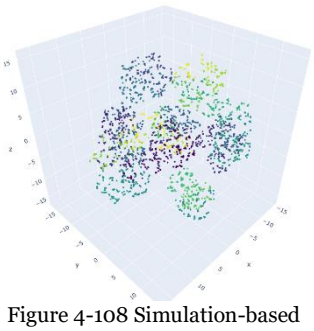
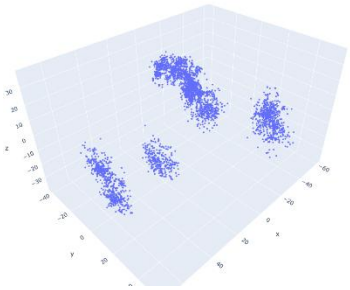
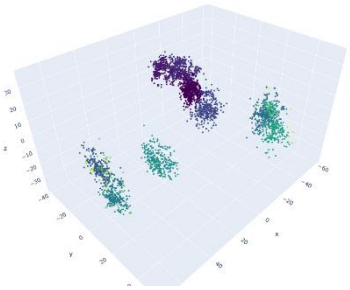
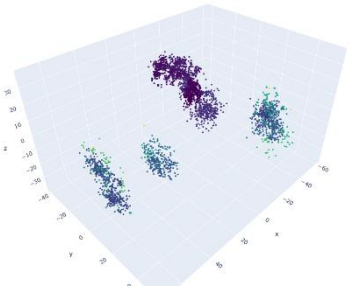
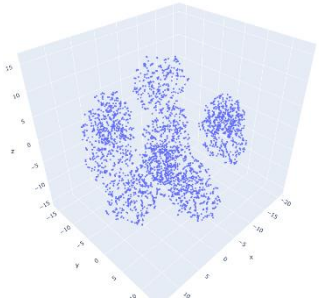
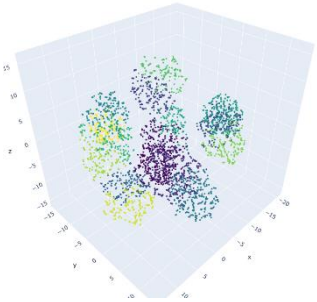
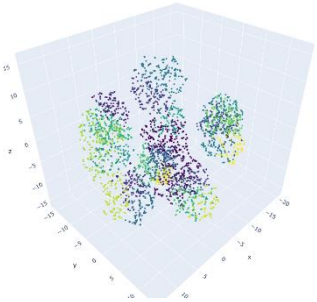
Figure 4-105 Model Error vs Cluster count Pareto Distribution, Vector-based Weights

4.2.6 Simulation-based weights results

The simulation-based adding weight method demonstrated success in reducing model error and performed similarly to the parameter-based adding weights method. The efficacy of this method relies on using it for multiple design spaces effectively, which is discussed later. Subdivisions of the weights generated from 'test design space 00' also came out with valuable insights. Notably, when weights were applied only to different window orientations, the framework performed poorly (see Figure 4-128), resembling the frameworks that used unweighted data. However, when weights were applied solely to length and width, the model error significantly decreased, and the Pareto frontier approached the bottom line (see Figure 4-127). This highlights the greater influence of building size and shape on annual energy consumption compared to the amount of aperture in different facades.

Adding weights method	Dimensionality reduction	Clustering method	See figure	comments
Simulation-based weights	TSNE	Meanshift	Figure 4-106	Data points are distributed in larger groups, tiny clusters with 3-5 data points are visible in those big clusters.
	Isomap embedding		Figure 4-109	The clustering pattern is reminiscent of the parameter-based adding weight method with Isomap embedding, with densified clusters dispersed with larger gaps.
Simulation-based length and width weights	TSNE		Figure 4-112	Data points are positioned in a similar manner to the framework that incorporates all simulation-based weights with TSNE. However, the Pareto frontier is slightly higher in comparison.
	Isomap embedding		Figure 4-115	The clustering pattern observed is similar to the framework that includes all simulation-based weights with isomap embedding. Surprisingly, the Pareto frontier is positioned lower than that framework, indicating better performance.
Simulation-based window size weights	TSNE		Figure 4-118	The data points exhibit a clear organization into small, well-defined groups. However, it is puzzling why the Pareto frontier is positioned

				at an unexpectedly high level despite the presence of distinct clusters.
	Isomap embedding		Figure 4-121	No distinct group is visible. Performance of this framework is almost same as the framework that used unweighted data frame followed by isomap embedding.
	Spectral embedding		Figure 4-124	Data points are scattered in amorphous, indistinct groups, leading to a high model error.

		Cluster Count 15-16	Cluster Count 29-30
TSNE			
	Figure 4-106 Simulation-based weights, TSNE, Meanshift	Figure 4-107 Simulation-based weights, TSNE, Meanshift, Cluster Count 15	Figure 4-108 Simulation-based weights, TSNE, Meanshift, Cluster Count 29
Isomap			
	Figure 4-109 Simulation-based weights, Isomap, Meanshift	Figure 4-110 Simulation-based weights, Isomap, Meanshift, Cluster Count 15	Figure 4-111Simulation-based weights, Isomap, Meanshift, Cluster Count 30
TSNE			
	Figure 4-112 Simulation-based (length, width) weights, TSNE, Meanshift	Figure 4-113 Simulation-based (length, width) weights, TSNE, Meanshift, Cluster Count 16	Figure 4-114 Simulation-based (length, width) weights, TSNE, Meanshift, Cluster Count 30

Isomap

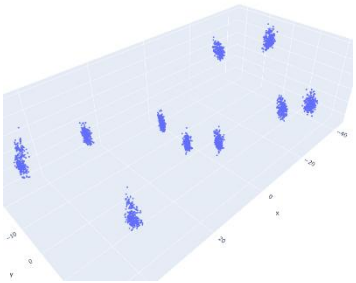


Figure 4-115 Simulation-based (length, width) weights, Isomap, Meanshift

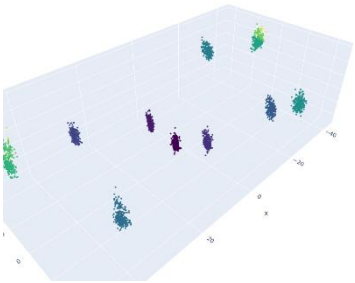


Figure 4-116 Simulation-based (length, width) weights, Isomap, Meanshift, Cluster Count 15

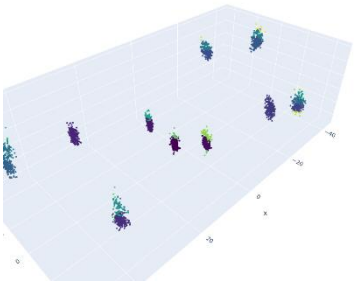


Figure 4-117 Simulation-based (length, width) weights, Isomap, Meanshift, Cluster Count 30

TSNE

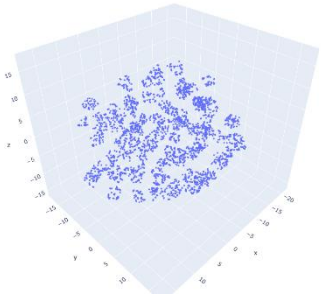


Figure 4-118 Simulation-based (Window Size) weights, TSNE, Meanshift

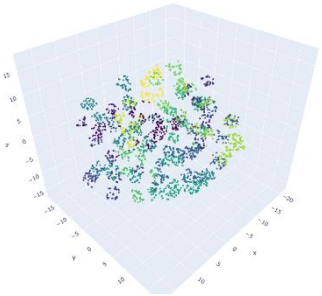


Figure 4-119 Simulation-based (Window Size) weights, TSNE, Meanshift, Cluster Count 15

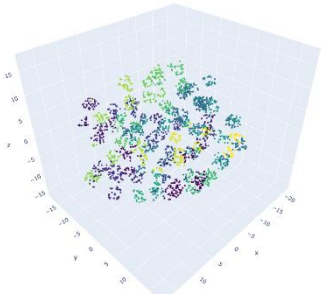


Figure 4-120 Simulation-based (Window Size) weights, TSNE, Meanshift, Cluster Count 29

Isomap

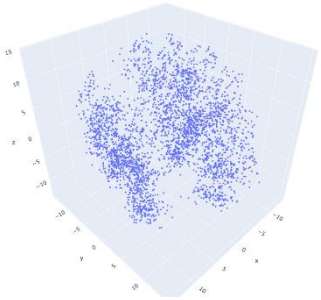


Figure 4-121 Simulation-based (Window Size) weights, Isomap, Meanshift

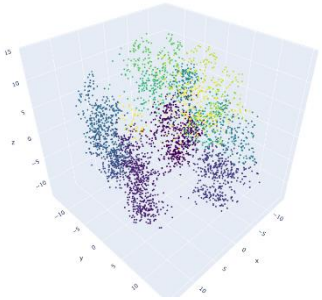


Figure 4-122 Simulation-based (Window Size) weights, Isomap, Meanshift, Cluster Count 13

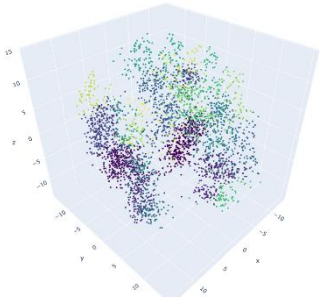


Figure 4-123 Simulation-based (Window Size) weights, Isomap, Meanshift, Cluster Count 30

Spectral Embedding

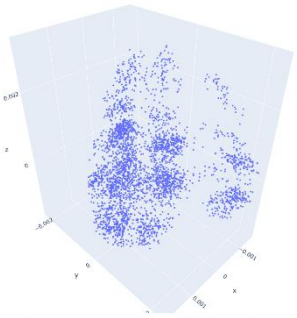


Figure 4-124 Simulation-based (Window Size) weights, Spectral Embedding, Meanshift

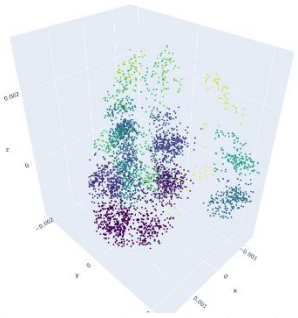


Figure 4-125 Simulation-based (Window Size) weights, Spectral Embedding, Meanshift, Cluster Count 15

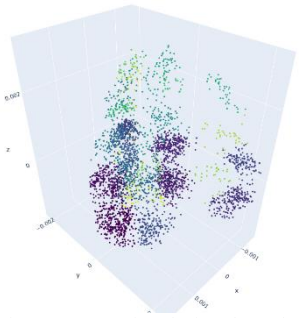


Figure 4-126 Simulation-based (Window Size) weights, Spectral Embedding, Meanshift, Cluster Count 30

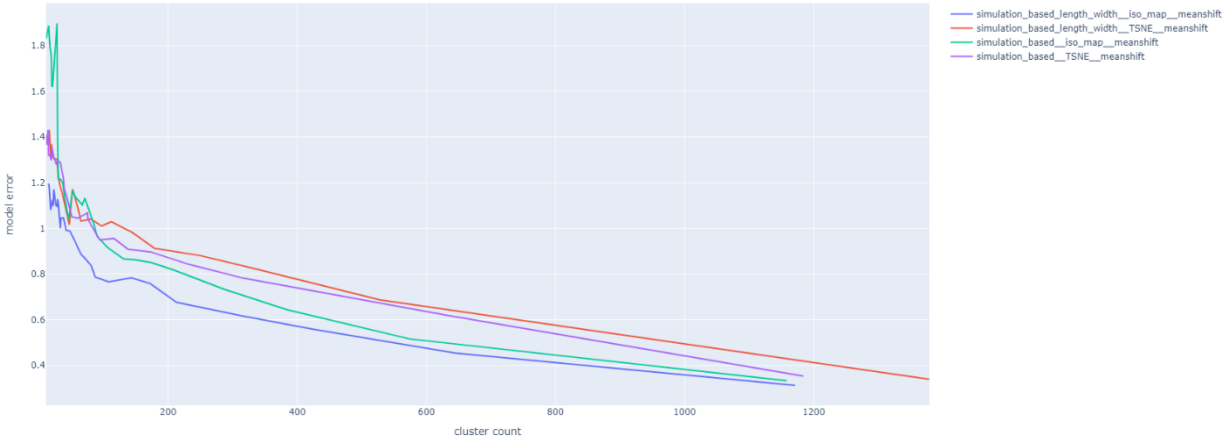


Figure 4-127 Model Error vs Cluster count Pareto Distribution, Simulation-based Weights, Simulation-based Length width Weights

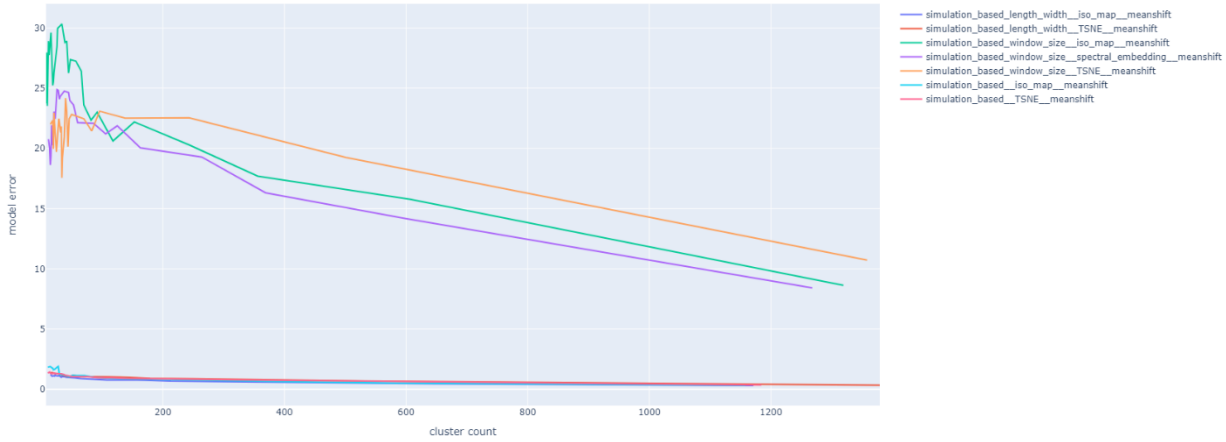


Figure 4-128 Model Error vs Cluster count Pareto Distribution, Simulation-based Weights, Simulation-based Length width Weights, Simulation-based Window Size Weights

4.2.7 Combined weights results

The simulation-based adding weights method provided valuable insights into the importance of building size and volume in predicting annual energy consumption. While this method performed better than the framework with no weights, it was not as effective as the other high-performing methods. However, the process proved to be less effective when combining weights due to the challenge of determining the appropriate altered scale for different weights. This uncertainty regarding the appropriate scale prevented the application of this process in further investigations.

Adding weights method	Dimensionality reduction	Clustering method	See figure	comments
Combined weights	TSNE	Meanshift	Figure 4-129	Data points are well organized in large groups primarily due to the increased emphasis on the weights for the length and width of design options.

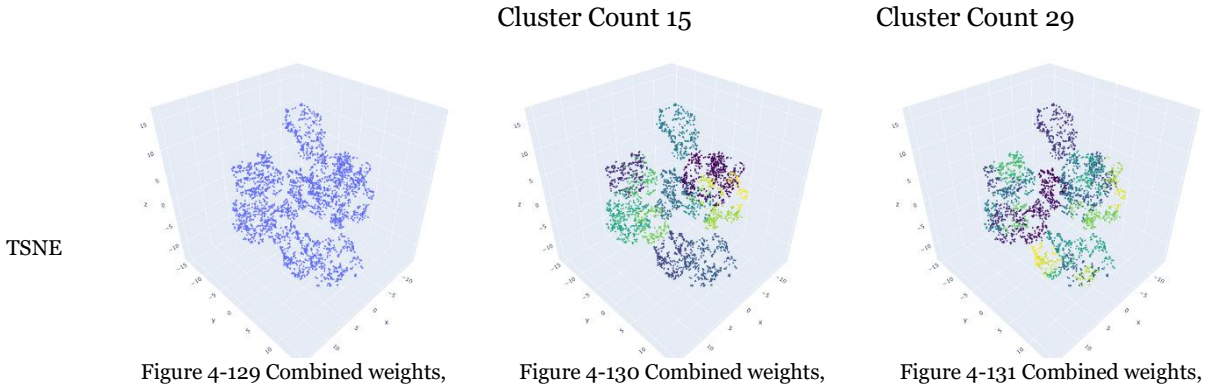


Figure 4-129 Combined weights, TSNE, Meanshift

Figure 4-130 Combined weights, TSNE, Meanshift, Cluster Count 15

Figure 4-131 Combined weights, TSNE, Meanshift, Cluster Count 29

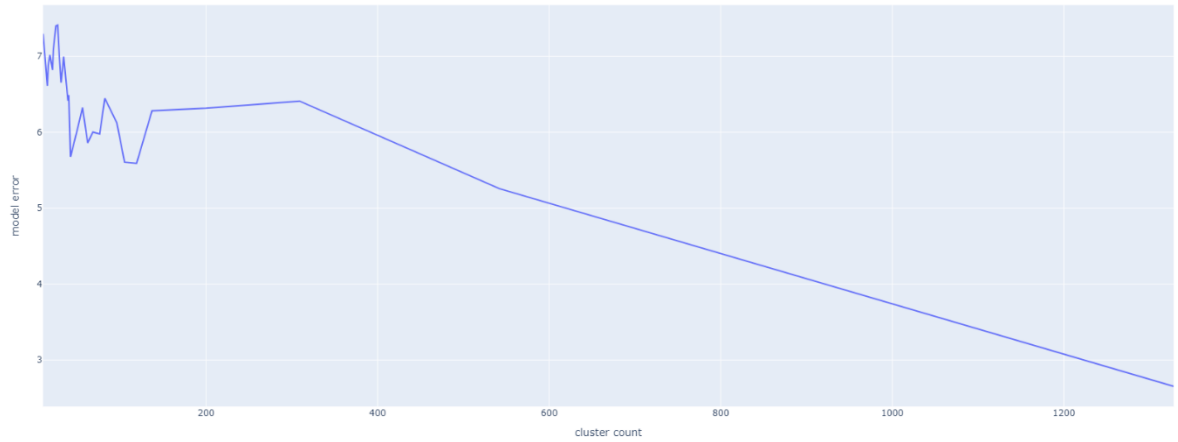


Figure 4-132 Model Error vs Cluster count Pareto Distribution, Combined Weights

4.2.8 Comparison between different frameworks tested for Design Space 01

Upon observing the chart displaying the Pareto frontiers of all the tested frameworks, a distinct subdivision becomes evident. The frontiers for the poorly performing frameworks are clustered towards the top, while the frontiers for the well-performing frameworks are positioned closer to the bottom line(see Figure 4-133) . Through a thorough investigation, we have gained insights that frameworks incorporating information about building size and volume exhibit lower model error with a reduced number of clusters. This finding emphasizes the importance of including information about building shapes in the analysis. For subsequent design spaces, we have exclusively focused on frameworks that lie closer to the bottom line, ensuring their effectiveness is validated through further investigations.

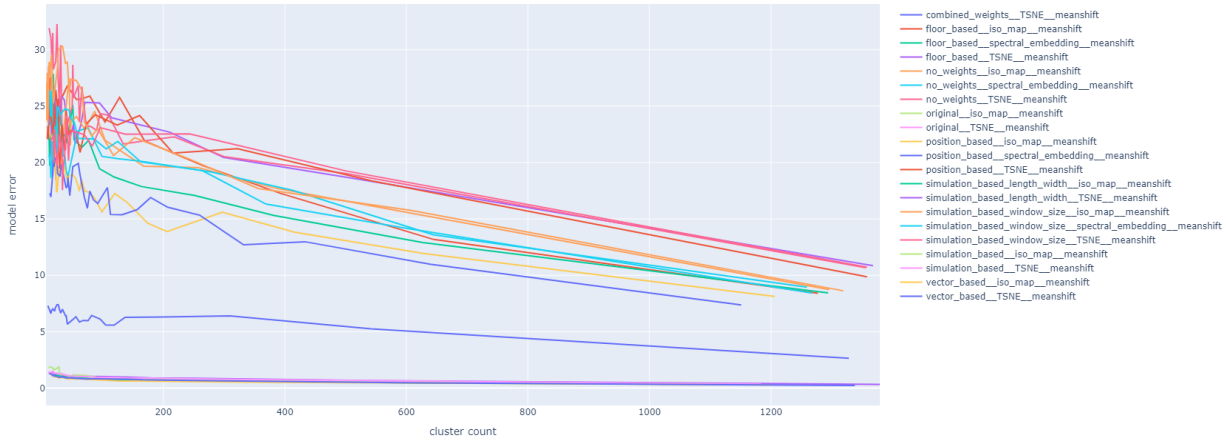


Figure 4-133 Pareto Distribution for all tested frameworks for Design Space 01

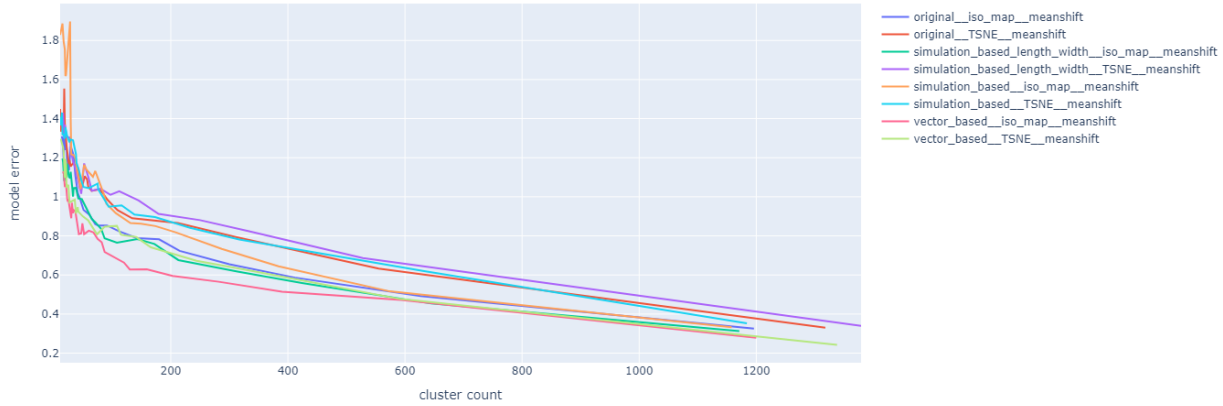


Figure 4-134 Pareto Distribution for only the good performing frameworks for Design Space 01

4.3 Framework Outcomes of Design Space 02:

The results of the frameworks for this design space are categorized into distinct sections based on the adding weights methods, similar to the previous design space. Frameworks used different portions of the initial data frame are explained together, allowing for a comprehensive understanding of their behavior. These tested frameworks offer valuable insights into the amount of simulation required to effectively approximate the annual energy consumption of the entire design space. Furthermore, the well-performing frameworks identified in the previous investigation are thoroughly examined in the context of the new design space, providing a deeper understanding of their efficacy.

4.3.1 No added weights results

Similar to the previous design space, when the data frame is not weighted, the model error is extremely high. The Pareto frontier generated from the framework utilizing TSNE is positioned at the top. On the other hand, the spectral embedding method produces a lower-positioned Pareto frontier among these three frameworks, suggesting better performance in terms of model error (see Figure 4-144).

Adding weights method	Dimensionality reduction	Clustering method	% of data used	See figure	comments
No added weights	TSNE	Meanshift	N/A	Figure 4-135	Data points are distributed in a single large spherical cluster, and within that cluster, no distinct subclusters can be identified.
	Isomap embedding		N/A	Figure 4-138	Data points are scattered within a large amorphous cluster, with no discernible smaller clusters within it.
	Spectral Embedding		N/A	Figure 4-141	The new positions of the data points in the lower dimensional space appear identical to the framework that utilizes Isomap embedding.

Cluster Count 14-15 Cluster Count 27-30

TSNE

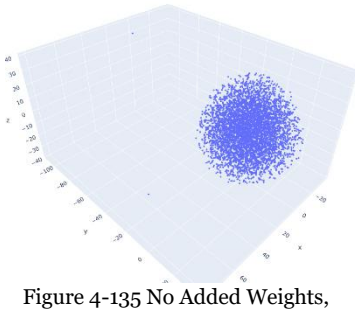


Figure 4-135 No Added Weights, TSNE, Meanshift, DS 02

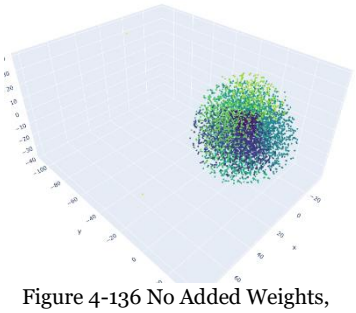


Figure 4-136 No Added Weights, TSNE, Meanshift, DS 02, Cluster Count 14

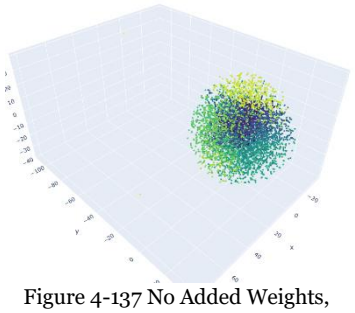


Figure 4-137 No Added Weights, TSNE, Meanshift, DS 02, Cluster Count 27

Isomap

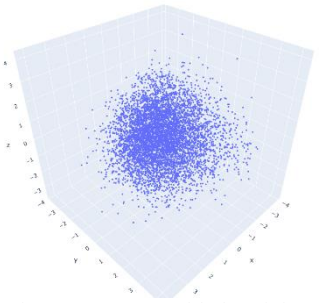


Figure 4-138 No Added Weights, Isomap, Meanshift, DS 02

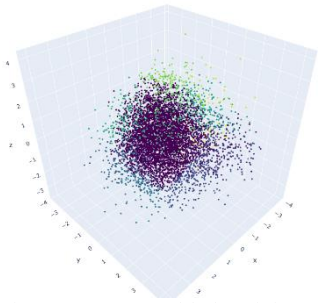


Figure 4-139 No Added Weights, Isomap, Meanshift, DS 02, Cluster Count 14

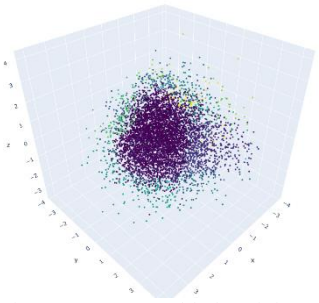


Figure 4-140 No Added Weights, Isomap, Meanshift, DS 02, Cluster Count 28

Spectral Embedding

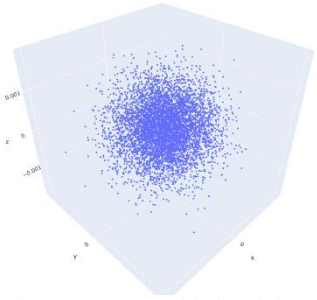


Figure 4-141 No Added Weights, Spectral Embedding, Meanshift, DS 02

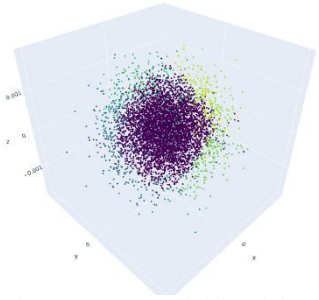


Figure 4-142 No Added Weights, Spectral Embedding, Meanshift, DS 02, Cluster Count 15

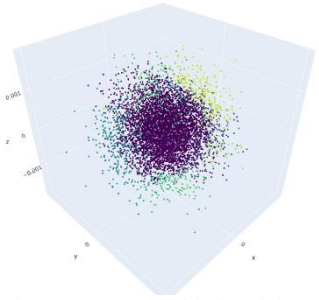


Figure 4-143 No Added Weights, Spectral Embedding, Meanshift, DS 02, Cluster Count 30

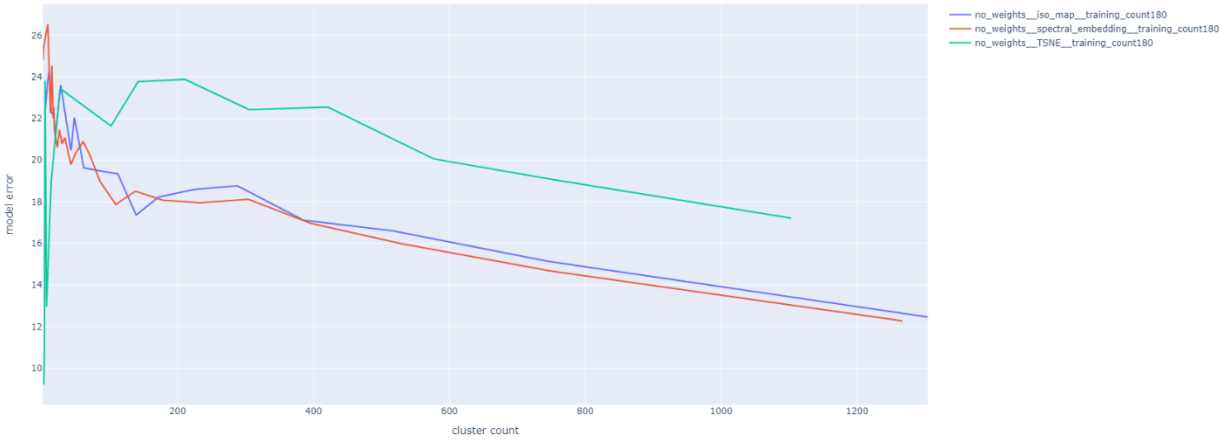


Figure 4-144 Model Error vs Cluster count Pareto Distribution, No Added Weights, Design Space 02

4.3.2 Parameter-based weights results

The parameter-based adding weights method performed well, similar to its performance in previous cases. The frameworks that included all the weights generated by the linear regression model exhibited higher model errors with a smaller amount of data points used for training. When using 20% or 10% of the data points, the generated Pareto frontier appeared almost the same, but for 5% and 3%, the frontier moved upward (see Figure 4-145 *(Training count 180 refers to 3%, 300 refers to 5%, 599 refers to 10% 1199 refers to 20%)*)

Interestingly, the frameworks that used only the weights for length and width resulted in similar Pareto frontiers regardless of the portion of the data frame used for training the linear regression model. The framework that generated a Pareto frontier positioned the closest to the bottom line, resulting in a reduced model error of 1.13 with a cluster count of only 49. By simulating 180 data points for this framework, we were able to approximate the entire design space containing 6000 data points.

Adding weights method	Dimen-sionality reduction	Clustering method	See figure	% of data used	comments
Parameter-based weights	TSNE	Meanshift	Figure 4-147	3%	The data points in this scenario are sparsely distributed within large clusters. The clusters are clearly distinguishable, with a less dense distribution of data points within them. However, the model error is higher compared to other frameworks, indicating a lower level of accuracy.
				5%	The data points in this scenario appear to be in the early stages of forming densified clusters. There are clusters of different sizes, but none of them are distinct or well-defined. The model error is slightly higher compared to other frameworks (see Figure 4-145).
				10%	At this stage, the data points are well-organized into small, densified clusters. The clusters show a clear structure and are relatively distinct from each other. The generated Pareto frontier is positioned lower than the framework that only utilized 5% of the data.
				20%	Data points are highly organized into distinct clusters due to the utilization of more accurate weights. The generated Pareto frontier exhibits a similar shape to the framework that used 10% of the data. However, it is positioned farther from the center.
	Isomap embedding		Figure 4-148	3%	Data points are scattered in an amorphous grid with loosely identifiable groups. Generated Pareto frontier located lower than the framework that utilized the same amount of data points and used TSNE for dimensionality reduction.
				5%	Data points are organized into distinct groups, with smaller groups tending to cluster together, leading to a few large clusters. However, this framework performed worse compared to the 'framework that utilized 5% of the data and used TSNE for dimensionality

					reduction. As a result, the generated Pareto frontier is positioned higher.
				10%	Data points are densely clustered in small, amorphous groups. However, the performance of this framework is inferior to the one that utilized 10% of the data with TSNE for dimensionality reduction. This is reflected in the higher position of the Pareto frontier.
				20%	Data points are well organized in small clusters within this framework. The performance of this framework is comparable to the one that utilized 20% of the data with TSNE for dimensionality reduction. Both frameworks yield similar outcomes in terms of model error and the positioning of the Pareto frontier (see Figure 4-145).
Parameter-based length and width weights	TSNE		Figure 4-149	3%	Data points are remarkably well organized within this framework, even with only 3% of the data points being utilized. Each cluster is clearly identifiable and exhibits a distinct shape. This framework achieves the lowest model error among all the tested frameworks, demonstrating its effectiveness in reducing the simulation count while still providing accurate results.
				5%	Both the clustering pattern and generated Pareto frontier are almost similar to the framework utilized 3% of the data with TSNE.
				10%	The clustering pattern of this framework closely resembles the outcome from the framework that utilized 3% of the data with TSNE. Additionally, the Pareto Frontier generated by this framework aligns with the frontier produced by the framework that used all extracted weights, while both frameworks utilized the same amount of data.
				20%	The outcome of this framework is comparable to the framework that utilized 3% of the data with TSNE. The generated Pareto frontier aligns with the frontier produced by the framework that used all extracted weights from the linear regression model.
	Isomap embedding		Figure 4-150	3%	The data points in this framework are distributed in well-organized, densely populated clusters, forming an amorphous grid-like pattern. This organized clustering leads to the generation of a Pareto frontier that closely aligns with the frontier produced by the framework that used TSNE with the same amount of data utilization.
				5%	The clustering pattern in this framework resembles the one observed in the framework that utilized 3% of the data. The generated Pareto frontier closely aligns with the frontier obtained from the framework that utilized TSNE (see Figure 4-146).

				10%	Similar outcomes as the framework utilized only 3% of the data with isomap embedding.
				20%	More accurate weights did not bring many changes in the data point distribution in lower dimensional space. The distribution is same as the framework utilized only 10% of the data with isomap embedding.

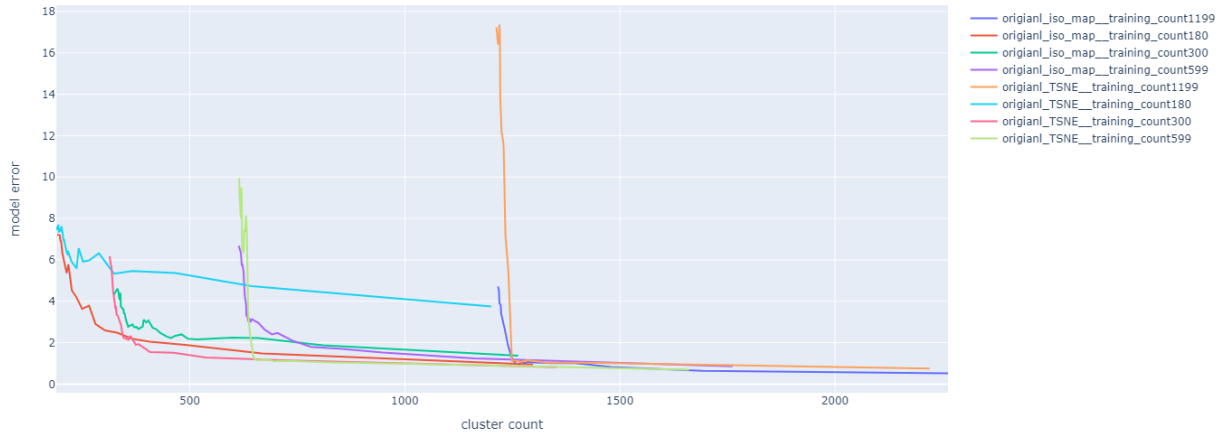


Figure 4-145 Model Error vs Cluster count Pareto Distribution, Parameter-based Weights, Design Space 02

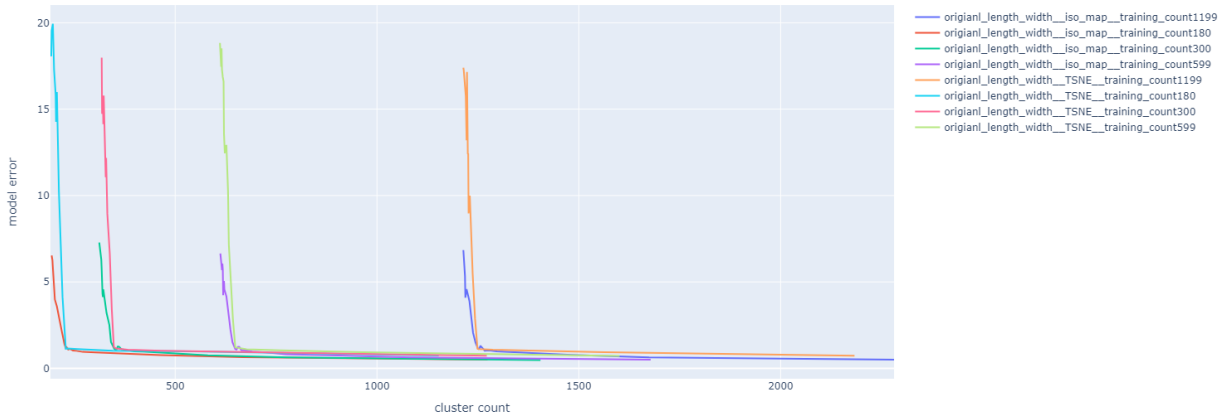


Figure 4-146 Model Error vs Cluster count Pareto Distribution, Parameter-based Length Width Weights, Design Space 02

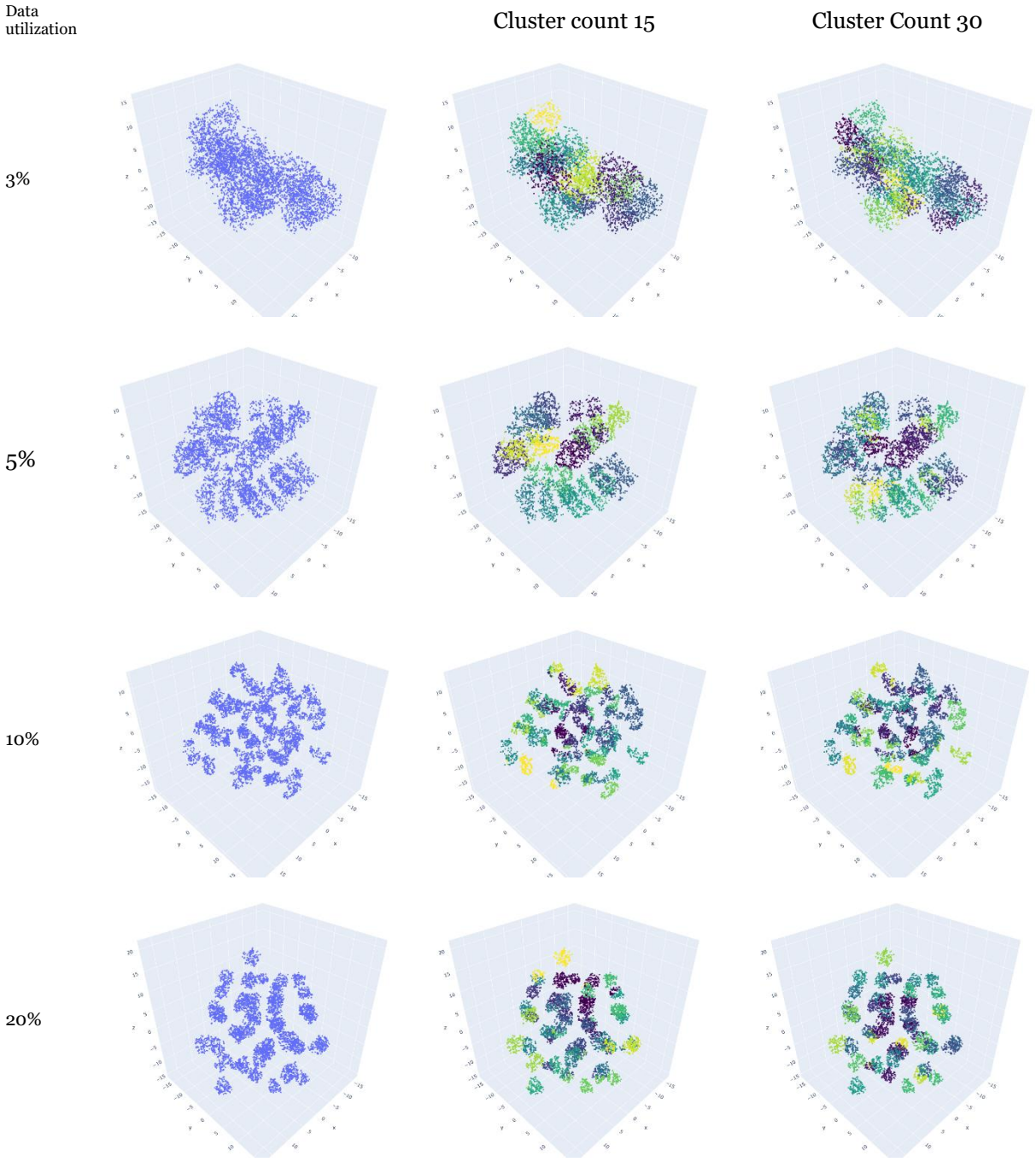


Figure 4-147 Parameter-based Weights, TSNE, Meanshift, DS 02

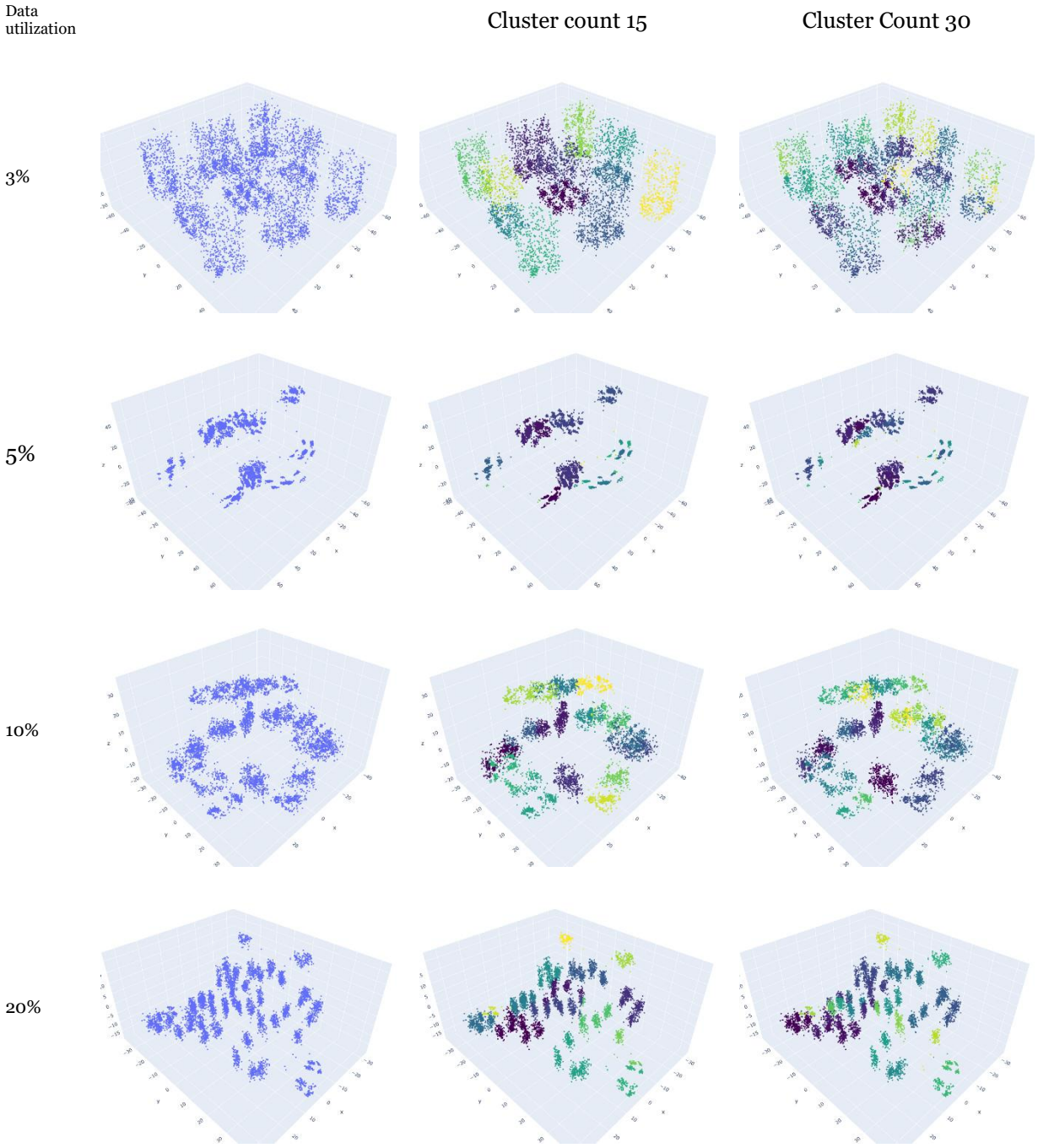


Figure 4-148 Parameter-based Weights, Isomap, Meanshift, DS 02

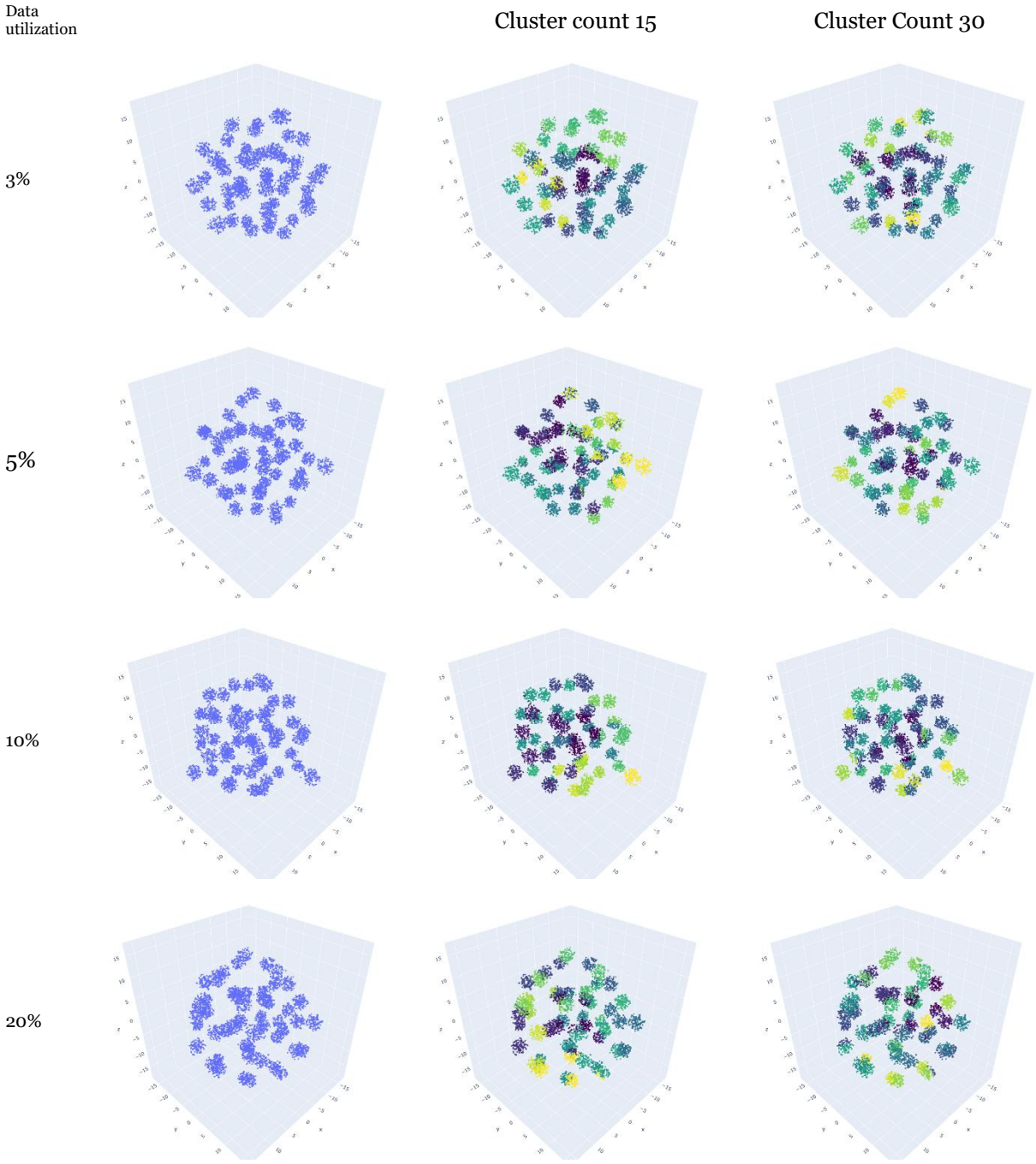


Figure 4-149 Parameter-based Length Width Weights, TSNE, Meanshift, DS 02

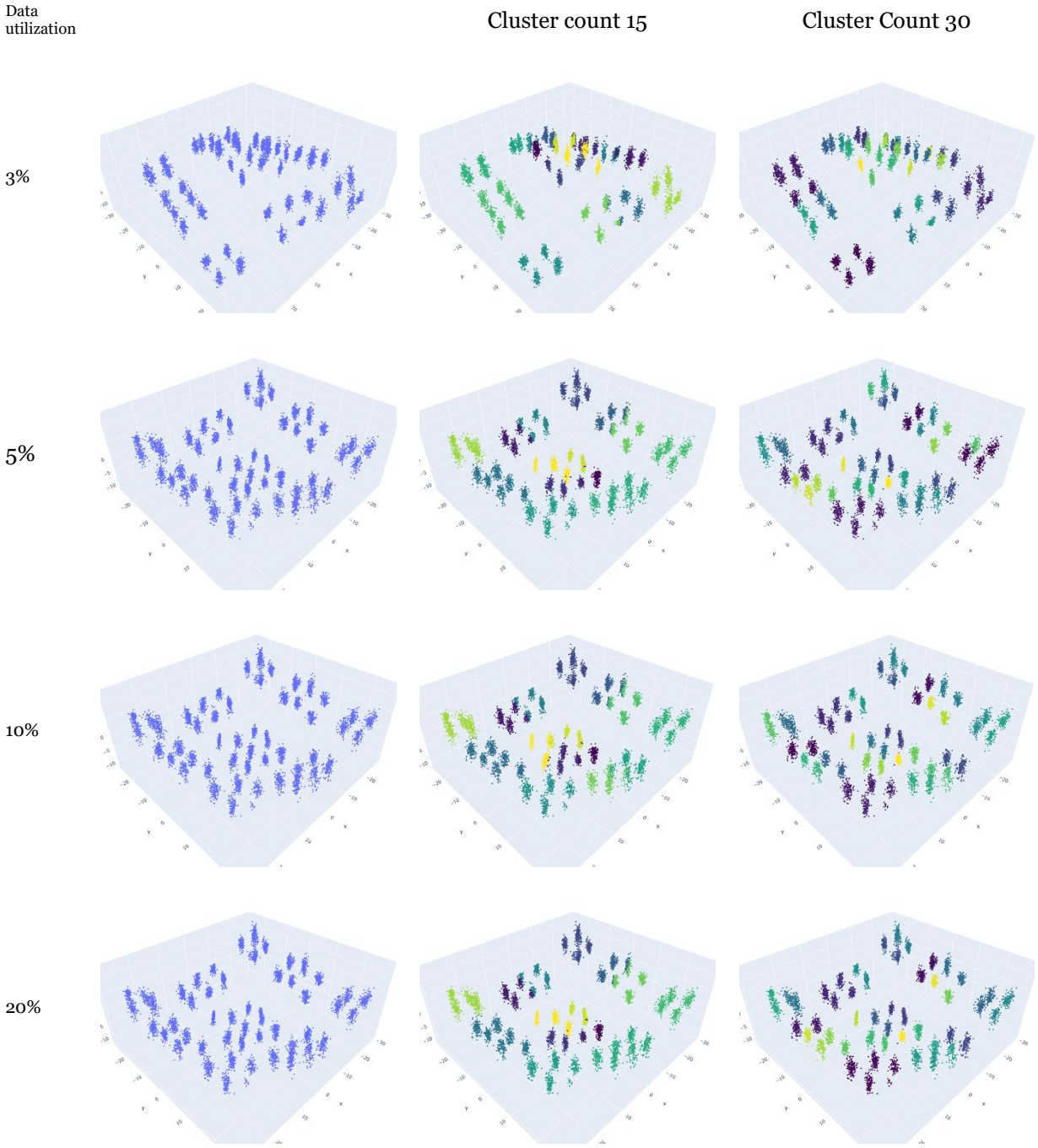


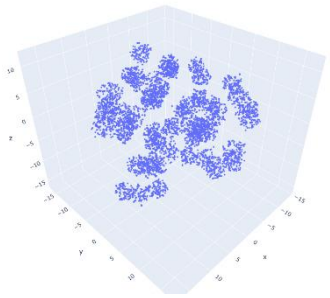
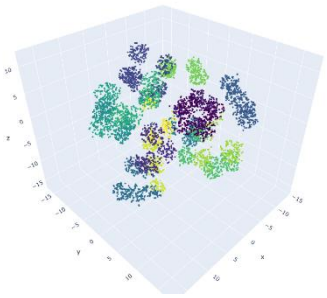
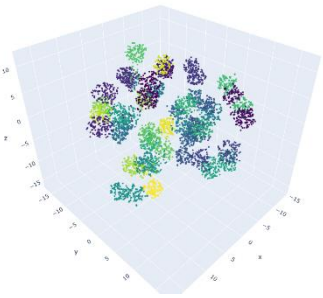
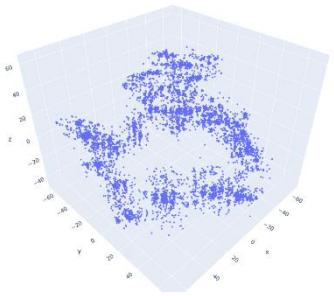
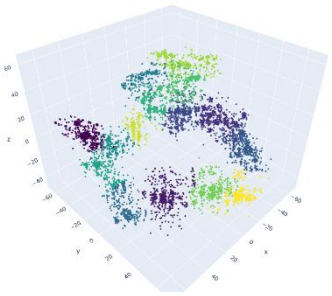
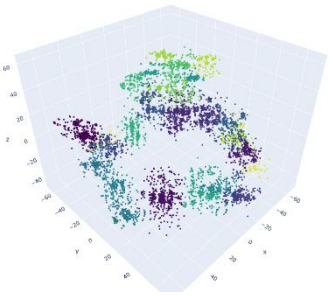
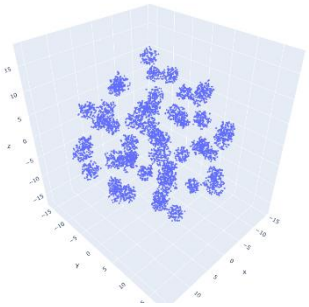
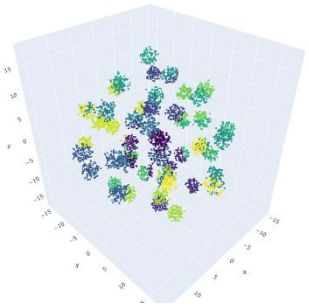
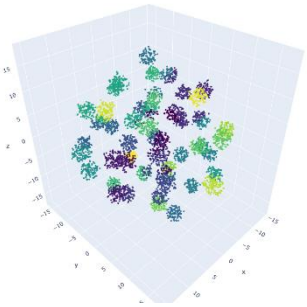
Figure 4-150 Parameter-based Length Width Weights, Isomap, Meanshift, DS 02

4.3.3 Simulation-based weights results

The simulation-based adding weights method proved to be successful in reducing the model error and generating a Pareto frontier positioned close to the bottom line. However, it requires running 600 simulations, which corresponds to 20% of the entire dataset. If other frameworks can achieve similar or better results with a lower number of simulations, it would diminish the effectiveness of the simulation-based adding weights method.

The simulation-based adding weights method produced similar outcomes as it did for the previous design space. The TSNE dimensionality reduction method performed well when considering all 10 weights from 'test design space 01', while the isomap embedding method generated meaningful clusters when only considering the weights for the length and width parameters.

Adding weights method	Dimensionality reduction	Clustering method	% of data used	See figure	comments
Simulation-based weights	TSNE	Meanshift	N/A	Figure 4-151	Data points are effectively organized into small, well-defined clusters. This framework has successfully generated a Pareto frontier that is positioned very close to the bottom line.
	Isomap embedding		N/A	Figure 4-154	Data points exhibit an amorphous distribution, with some areas of densification and scattered points around them. The generated Pareto frontier for this framework is positioned at the top among the four frameworks that were experimented with.
Simulation-based length width weights	TSNE		N/A	Figure 4-157	The positions of data points in the lower-dimensional space are identical to the framework that utilized all simulation weights with TSNE. This observation further reinforces the strong influence of the length and width parameters on annual energy consumption.
	Isomap embedding		N/A	Figure 4-160	The clustering pattern observed in the current framework is identical to the one obtained from the framework that used weights only for length and width parameters, implemented the parameter-based adding weights method with isomap embedding, and utilized only 3% of the data for training.

		Cluster count 15	Cluster Count 28-30
TSNE			
	Figure 4-151 Simulation-based Weights, TSNE, Meanshift, DS 02	Figure 4-152 Simulation-based Weights, TSNE, Meanshift, DS 02, Cluster Count 15	Figure 4-153 Simulation-based Weights, TSNE, Meanshift, DS 02, Cluster Count 30
Isomap			
	Figure 4-154 Simulation-based Weights, Isomap, Meanshift, DS 02	Figure 4-155 Simulation-based Weights, Isomap, Meanshift, DS 02, Cluster Count 15	Figure 4-156 Simulation-based Weights, Isomap, Meanshift, DS 02, Cluster Count 28
TSNE			
	Figure 4-157 Simulation-based Length Width Weights, TSNE, Meanshift, DS 02	Figure 4-158 Simulation-based Length Width Weights, TSNE, Meanshift, DS 02, CLuster Count 15	Figure 4-159 Simulation-based Length Width Weights, TSNE, Meanshift, DS 02, CLuster Count 28

Isomap

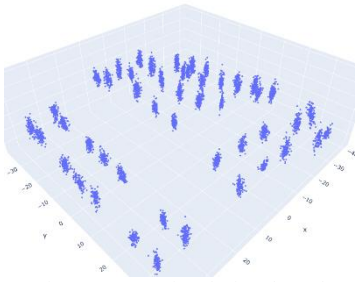


Figure 4-160 Simulation-based Length Width Weights, Isomap, Meanshift, DS 02

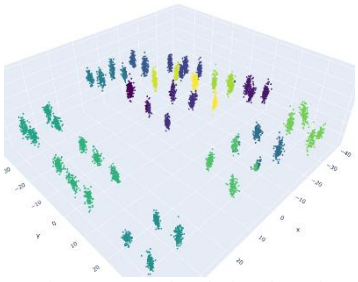


Figure 4-161 Simulation-based Length Width Weights, Isomap, Meanshift, DS 02, Cluster Count 15

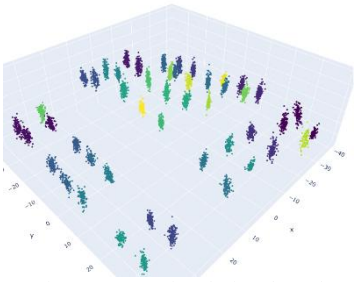


Figure 4-162 Simulation-based Length Width Weights, Isomap, Meanshift, DS 02, Cluster Count 29

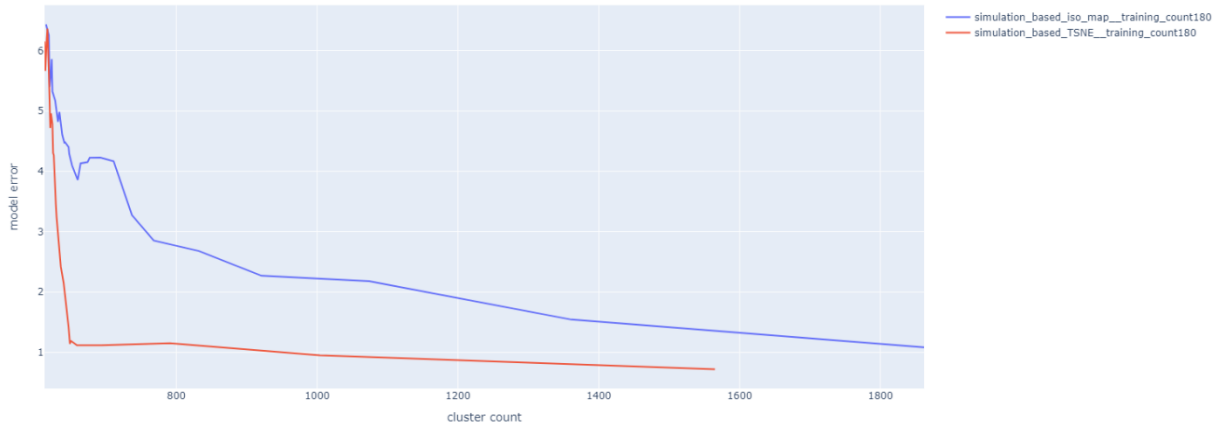


Figure 4-163 Model Error vs Cluster count Pareto Distribution, Simulation-based Weights, Design Space 02

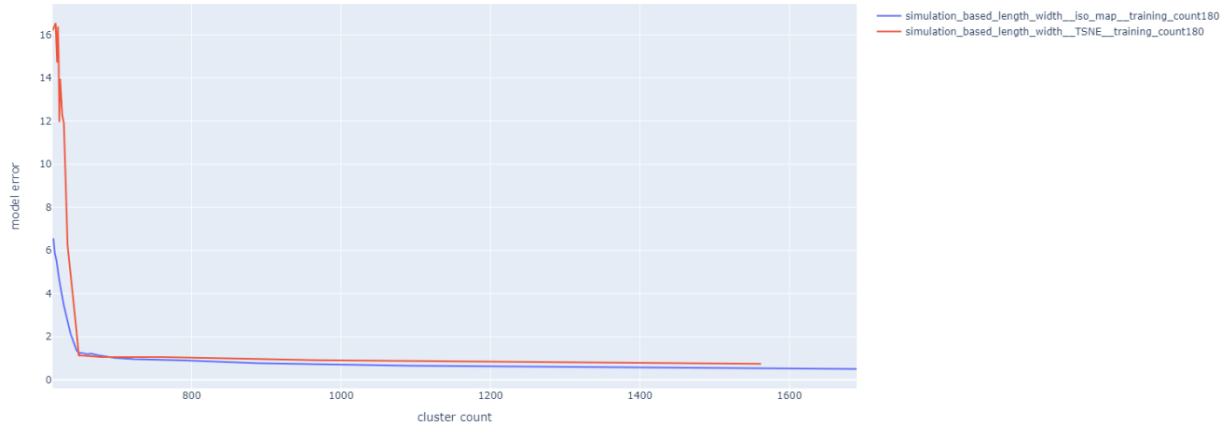


Figure 4-164 Model Error vs Cluster count Pareto Distribution, Simulation-based Length Width Weights, Design Space 02

4.3.4 Vector-based weights results

The vector-based adding weights method emerged as the best-performing framework among all the tested approaches, while the reason is previously explained. While incorporating information about the depth of shade in the vectors posed some challenges, the current investigation has yielded valuable insights into the effectiveness of different processes. The frameworks that considered the sum of window-to-wall ratios (WWRs) and shading depths, as well as the framework that used the sum with positive or negative weighted values of the shading depth, performed well (see Figure 4-173, Figure 4-174). These frameworks generated Pareto frontiers that provide us with values very close to the center, demonstrating their ability to achieve low model errors by utilizing only 3% of the data. This success validates the efficacy of the vector-based approach in this domain-specific scenario. Only these two well-performed vector-based adding weights methods are investigated for the next design space.

However, the remaining two frameworks, one involving the product of WWRs and shading depths and the other using weighted shading depths combined with WWRs, performed relatively poorly (see Figure 4-175, Figure 4-176). This could be attributed to either placing unnecessary emphasis on shading depths or ineffective integration of these two important parameters.

Adding weights method	Dimen-sionality reduction	Clustering method	See figure	% of data used	comments
Vector-based (window + shade) weights	TSNE	Meanshift	Figure 4-165	3%	Data points are efficiently organized in well-defined clusters by utilizing only 3% of the data frame, showcasing the success of the framework in generating a Pareto frontier with lower model error.
				5%	The utilization of more accurate weights has resulted in the clustering of data points into more distinct and well-defined clusters. The generated Pareto frontier is positioned further from the center, similar to the framework that utilized only 3% of the data.
				10%	The clustering pattern and Pareto frontier are identical to the framework that utilized only 3% of the data.
				20%	The clustering pattern and shape of the Pareto frontier obtained from the framework utilizing more accurate weights are identical to the framework that utilized only 3% of the data.
	Isomap embedding		Figure 4-166	3%	The data points in this framework are positioned in linear-shaped clusters. The performance of this framework is relatively poor compared to the TSNE-based framework with the same amount of data utilization.
				5%	The data points are positioned in large amorphous clusters. In comparison to the TSNE-based architecture with the same level of data usage, performance is comparatively poor, for why the generated Pareto frontier is positioned high, indicating a bit higher model error.
				10%	Data points are well-organized in an ununiform grid. the generated Pareto frontier closely aligns with the one

					generated by the framework that utilized TSNE with a similar amount of data utilization.
				20%	Data points are distributed in clusters of different shapes. The created Pareto frontier substantially resembles that produced by the framework that used TSNE with a similar amount of data utilization.
Vector-based (window + shade * (1/-1)) weights	TSNE		Figure 4-167	3%	Utilizing only 3% of the data frame, data points are effectively grouped into well-defined clusters, demonstrating the framework's efficacy in producing a Pareto frontier with decreased model error.
				5%	The clustering pattern and the shape of the generated Pareto frontier are identical to the framework that utilized only 3% of the data.
				10%	Generated Pareto frontier is identical to one generated by the framework utilising the sum of WWRs and shading depths while implementing vector-based adding weights methods with TSNE and the same amount of data utilization.
				20%	Generated Pareto frontier is almost the same as the one generated by the framework that utilized sum of WWRs and shading depths while implementing vector-based adding weights methods with TSNE and the same amount of data utilization
	Isomap embedding		Figure 4-168	3%	In this framework, the data points are clustered in clusters of different shapes and the clusters are distributed unevenly in the lower-dimensional space. The framework demonstrates poor performance compared to the other frameworks that utilized 3% of the data.
				5%	The clustering pattern is more well organized than the framework used isomap with 3% data utilization. The generated Pareto Frontier resembles the ones generated by other good-performing frameworks that utilized 5% of data.
				10%	With more accurate weights data points are now more well-organized. Pareto frontier is closely identical to the ones generated by other good-performing frameworks utilizing the similar portion of the data.
				20%	Data points are well organized in a roughly rectangular grid of clusters, generated Pareto frontier is almost the same as the ones generated by other frameworks utilized 20% of the data.
Vector-based (window * shade) weights	TSNE		Figure 4-169	3%	Data points are successfully categorized into well-defined clusters using just 3% of the data frame, proving the framework's ability to generate a Pareto frontier with less model error.

				5%	Data points are well organized in small clusters, generating Pareto frontier with a bit high model error compared to other frameworks utilizing a similar amount of data.
				10%	The clustering pattern and the shape of the generated Pareto frontier are identical to the framework that utilized only 5% of the data.
				20%	Shapes of the clusters are more distinct, data points are well organized, Pareto frontier is closely identical to the ones generated by other good-performing frameworks utilizing a similar portion of the data.
	Isomap embedding		Figure 4-170	3%	Data points are organized in clusters of amorphous shapes, the generated Pareto frontier produces values bit offset from the center.
				5%	In this framework, the data points are distributed in amorphous clusters without clear patterns or distinct groupings, resulting in generating Pareto frontier with a bit higher model errors.
				10%	Data points are densified in smaller clusters while the clusters are not uniformly distributed in the lower-dimensional space. Generated Pareto frontier is positioned a bit higher than the ones generated by frameworks that utilized a similar amount of data.
				20%	Data points are clustered in densified groups, and unevenly distributed in the lower dimension. Generated Pareto frontier is located a bit higher than others.
Vector-based (window * shade (weighted)) weights	TSNE		Figure 4-171	3%	Data points are well organized in small clusters. The shape of the Pareto frontier is well aligned with other ones generated by good-performing frameworks that utilized similar amount of data.
				5%	Clustering pattern is the same as the framework used the product of WWRs and shading depths while implementing vector based adding weights method and TSNE with similar amount of data utilization.
				10%	Shapes of the clusters are more distinct. Generated pareto frontier is same as other good performing ones.
				20%	Outcomes are the same as the framework utilized 10% of the data with TSNE with similar vector based adding weights method.
	Isomap embedding		Figure 4-172	3%	Data points are distributed in amorphous shape. Generated pareto frontier is closely identical to the one generated by the framework utilized the product of WWRs and shading depths while implementing vector-based adding weights method and isomap embedding with 3% data utilization.

				5%	Data points are scattered in roughly identifiable clusters, surprisingly generated Pareto frontier is identical to ones generated by other good-performing frameworks.
				10%	Variously shaped clusters are used to organize data points. Generated a Pareto frontier with a reduced model error.
				20%	Data points are clustered in groups of different shapes. Generated Pareto frontier is similar to the ones generated by other good-performing frameworks utilized similar amount of data.

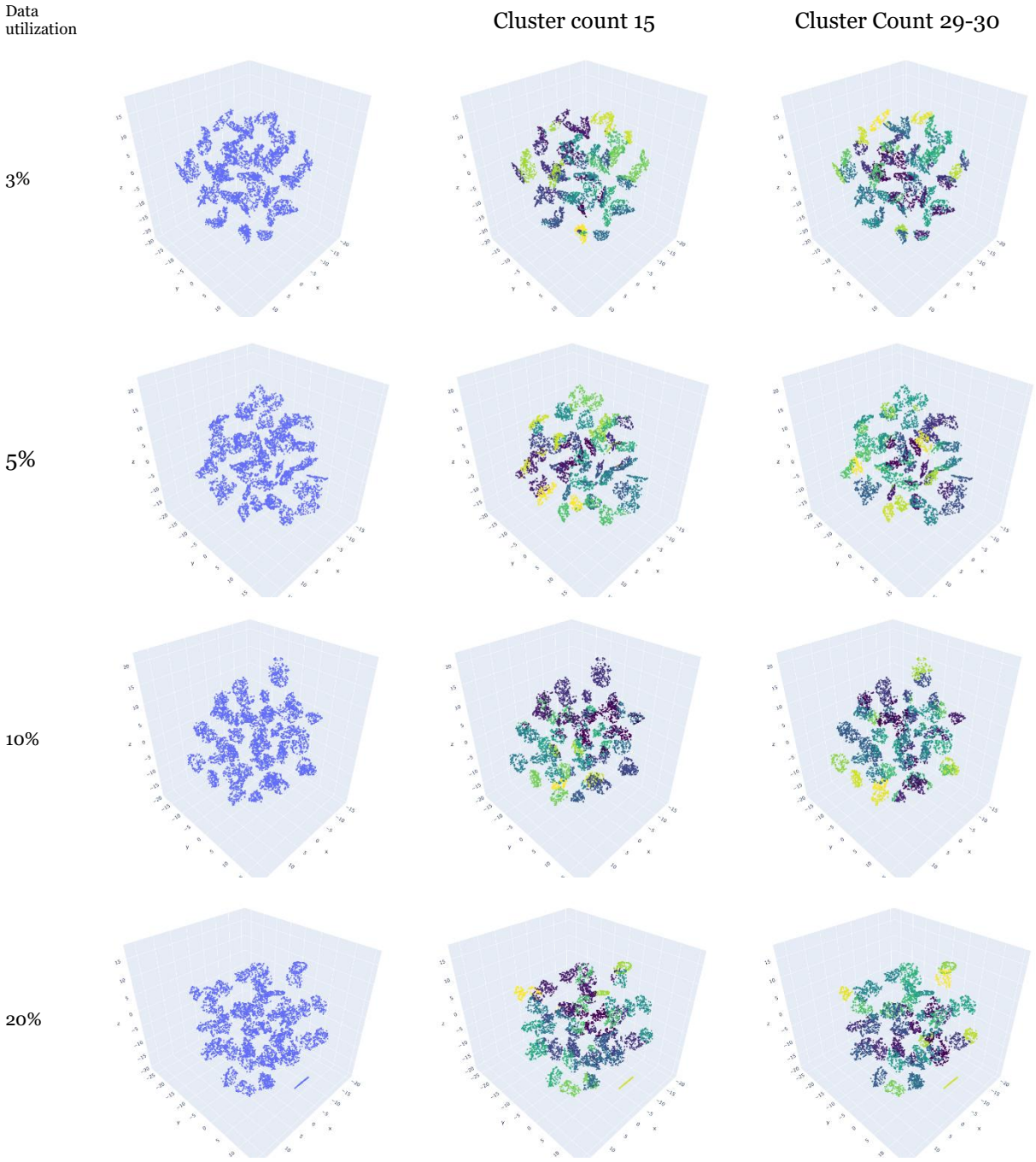


Figure 4-165 Vector-based (Window + Shade) Weights, TSNE, Meanshift, DS 02

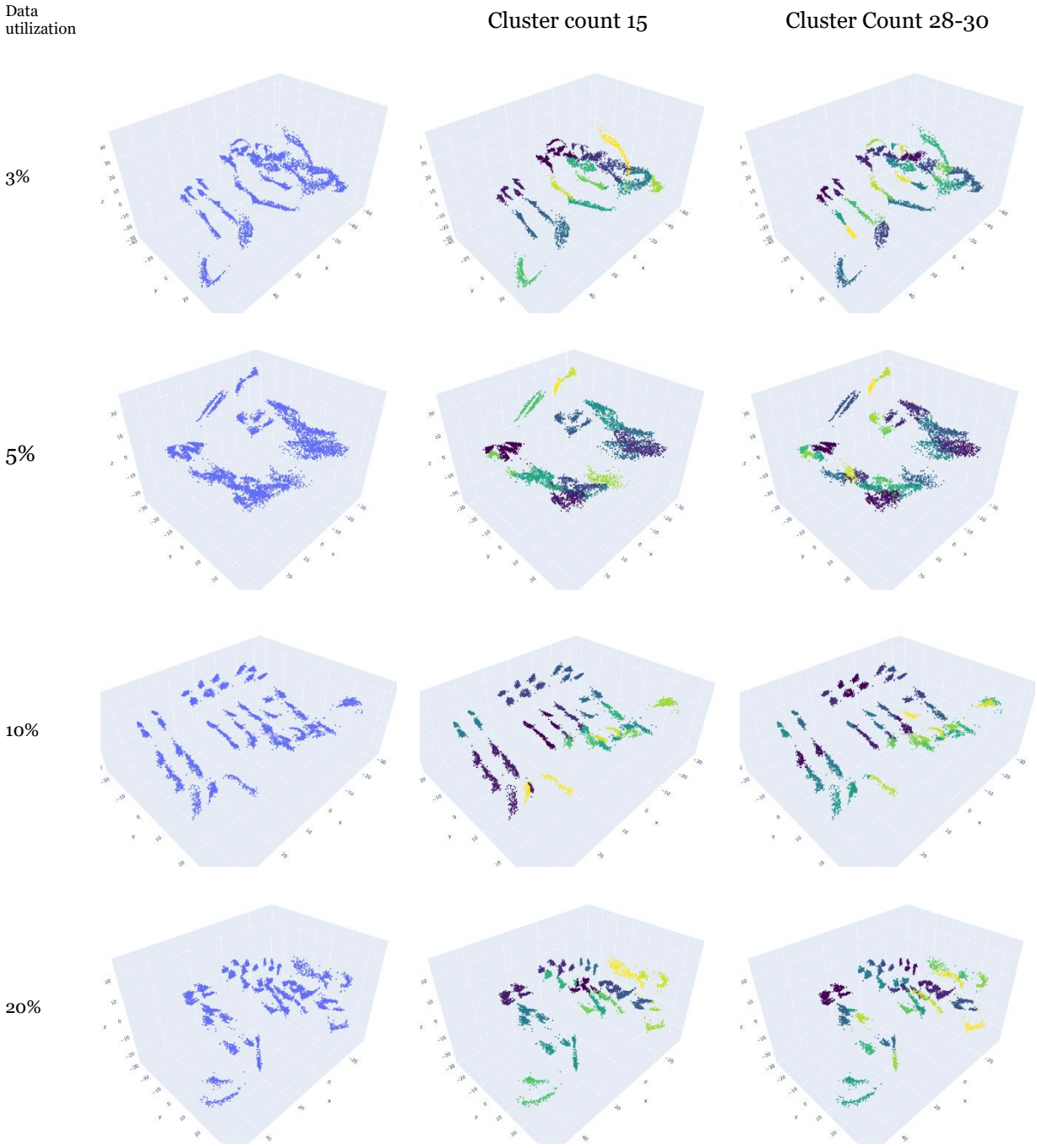


Figure 4-166 Vector-based (Window + Shade) Weights, Isomap, Meanshift, DS 02

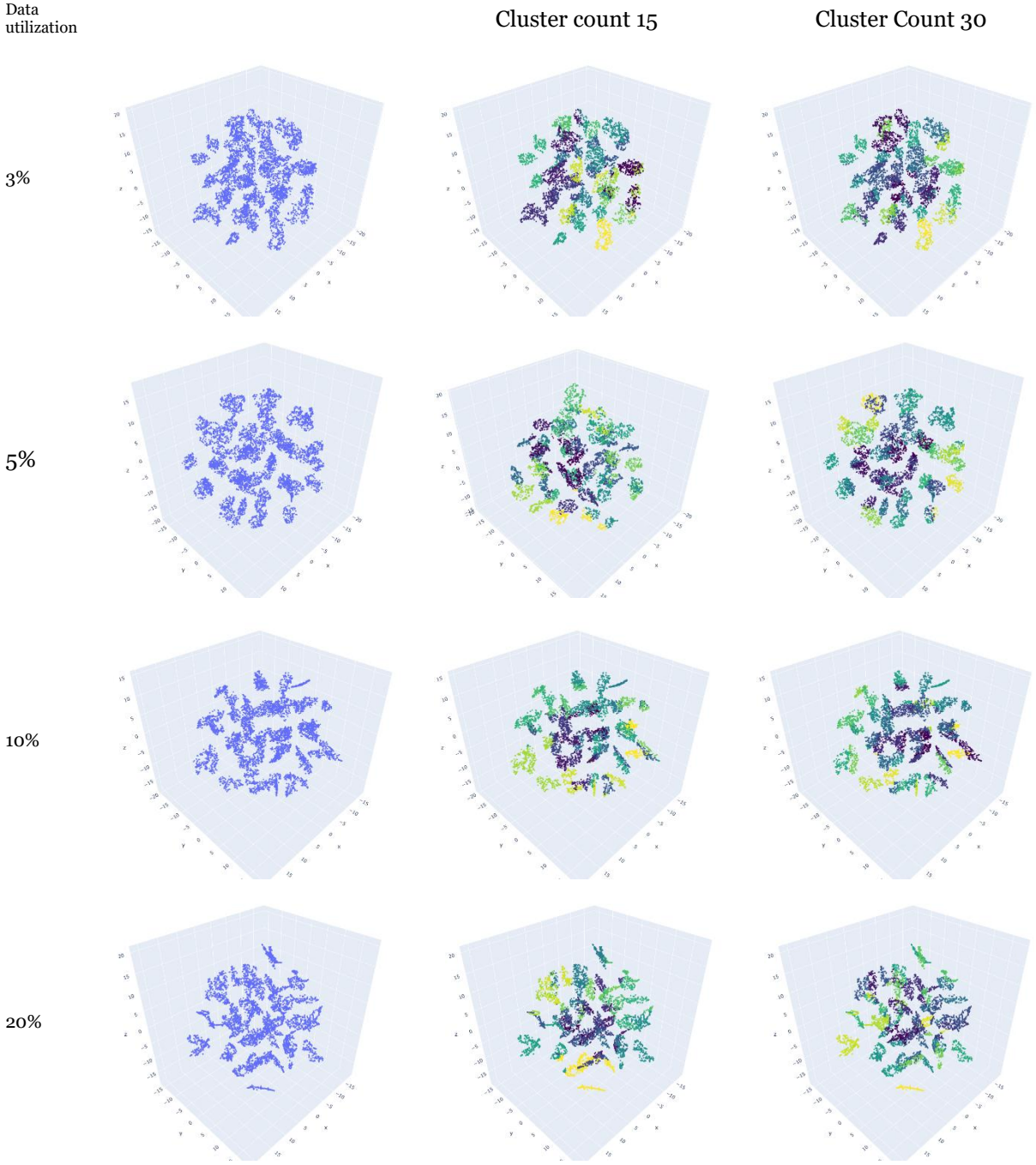


Figure 4-167 Vector-based (Window + Shade*(-1/1)) Weights, TSNE, Meanshift, DS 02

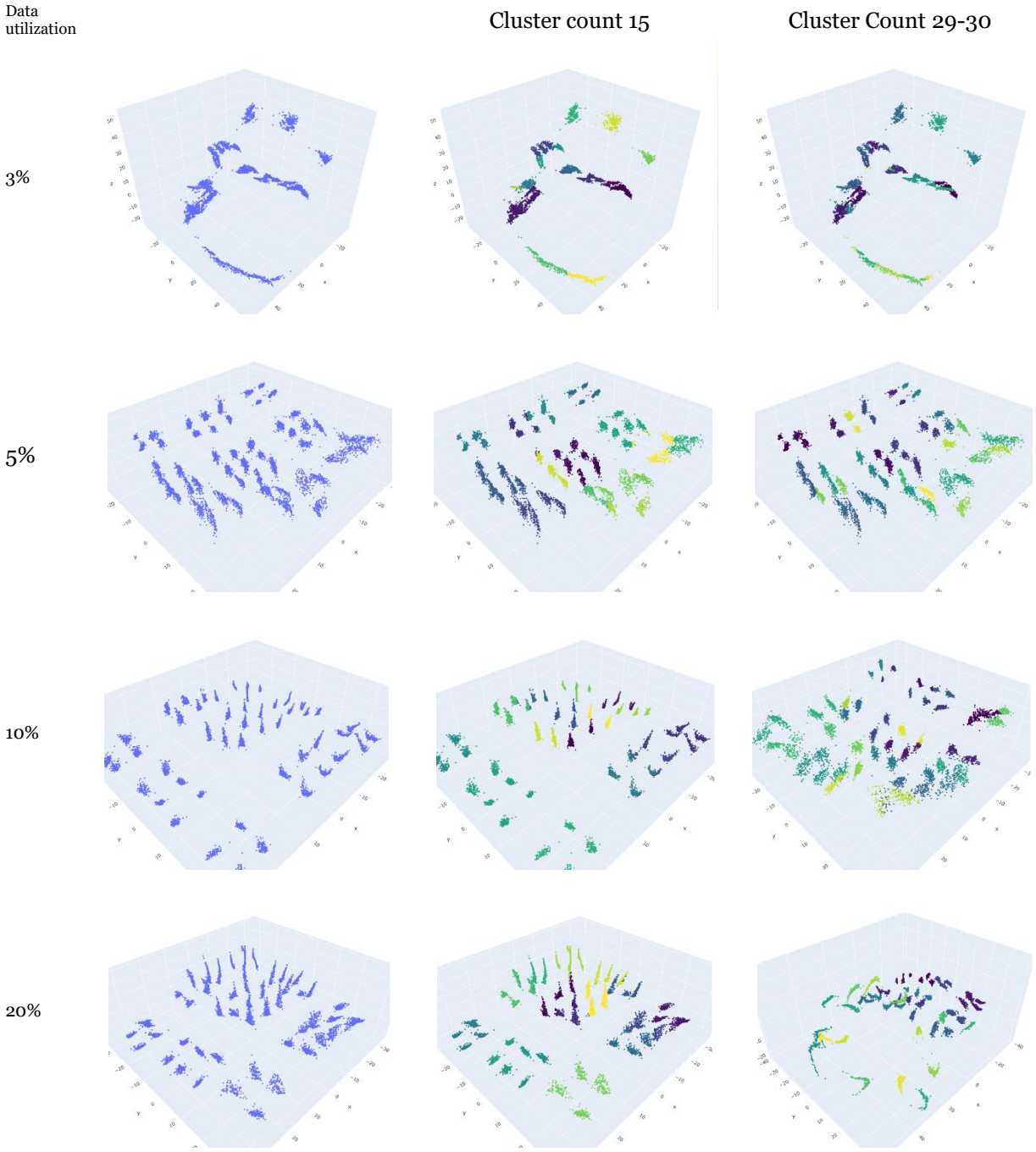


Figure 4-168 Vector-based (Window + Shade*(-1/1)) Weights, Isomap, Meanshift, DS 02

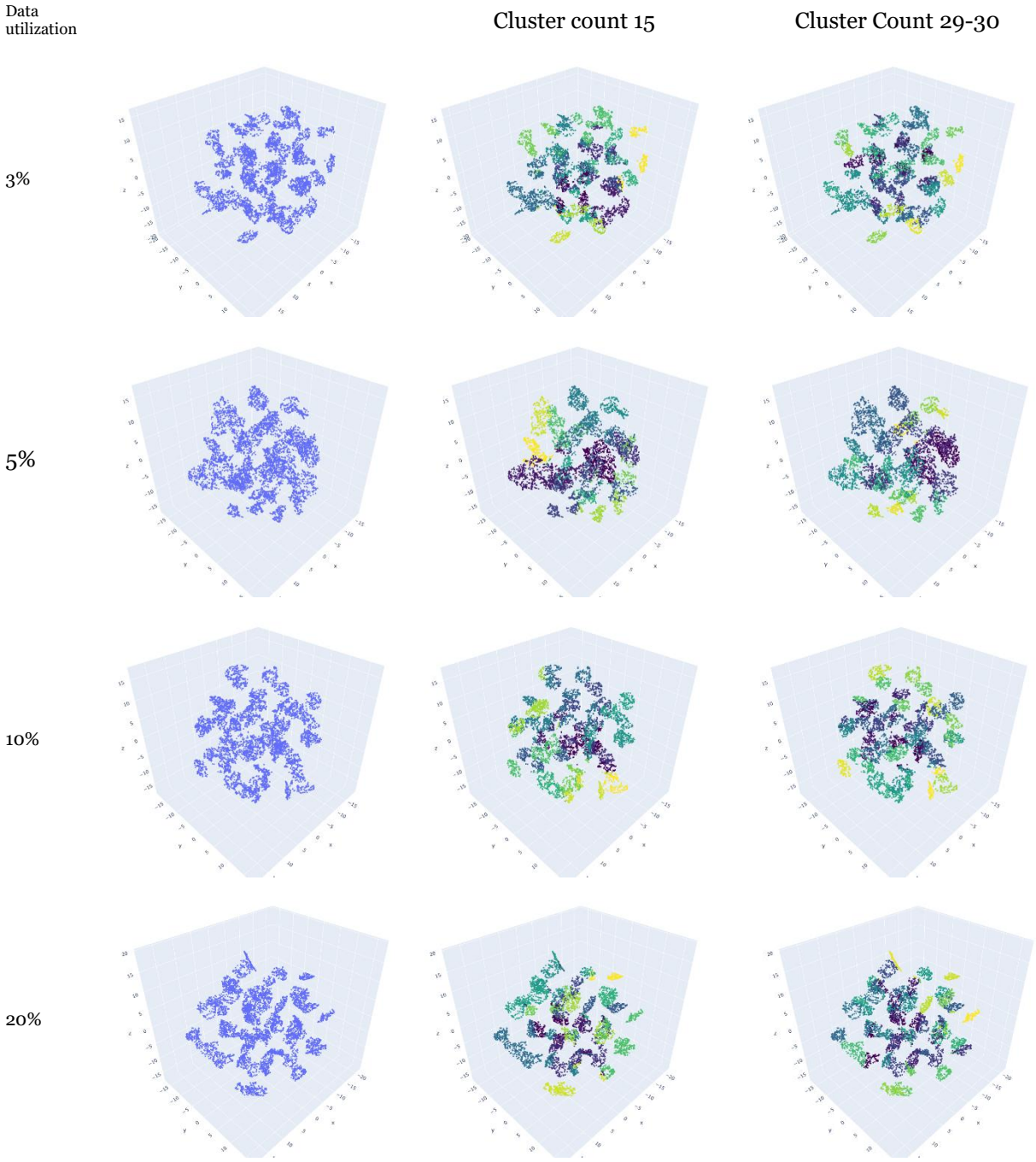


Figure 4-169 Vector-based (Window * Shade) Weights, TSNE, Meanshift, DS 02

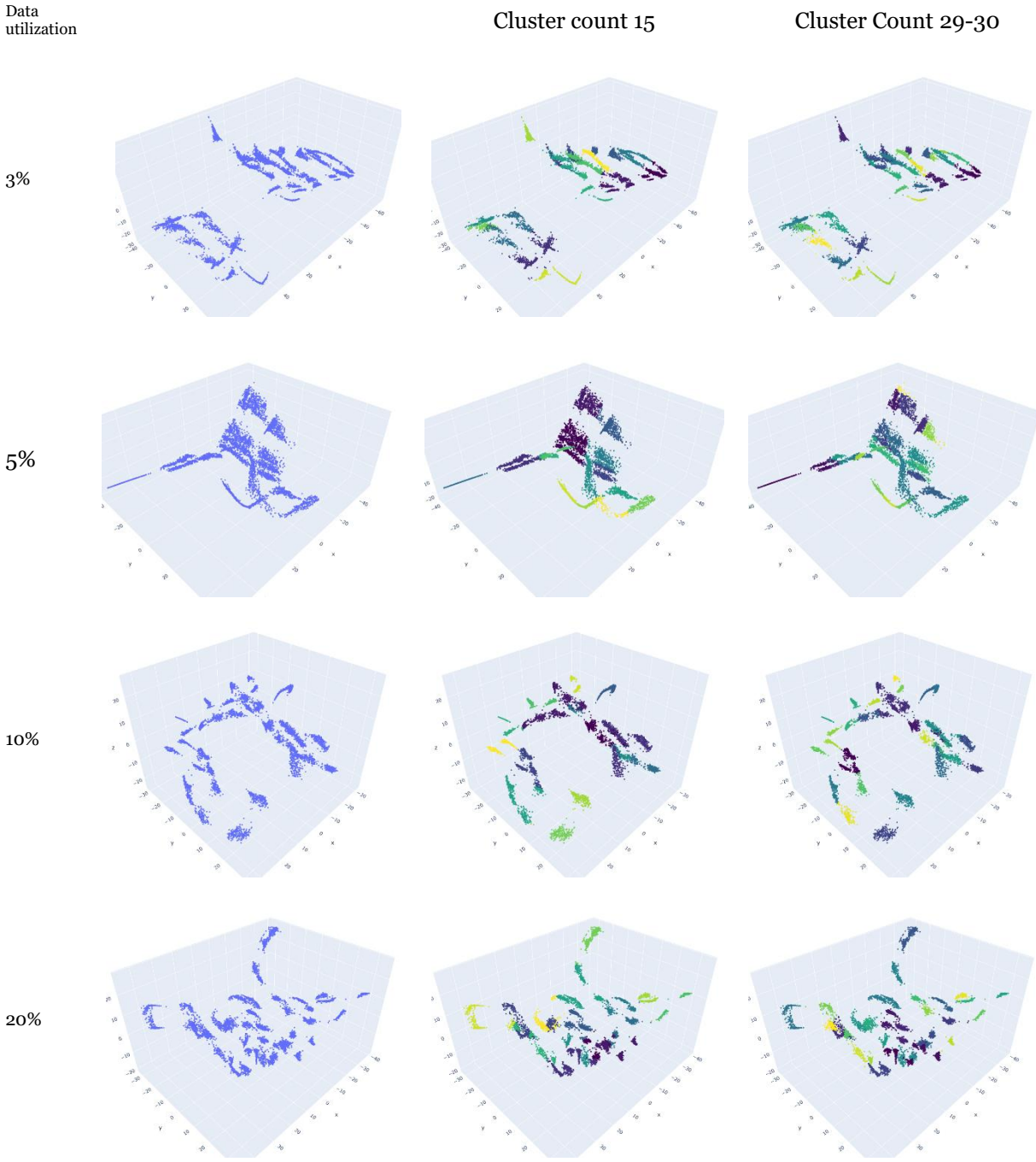


Figure 4-170 Vector-based (Window * Shade) Weights, Isomap, Meanshift, DS 02

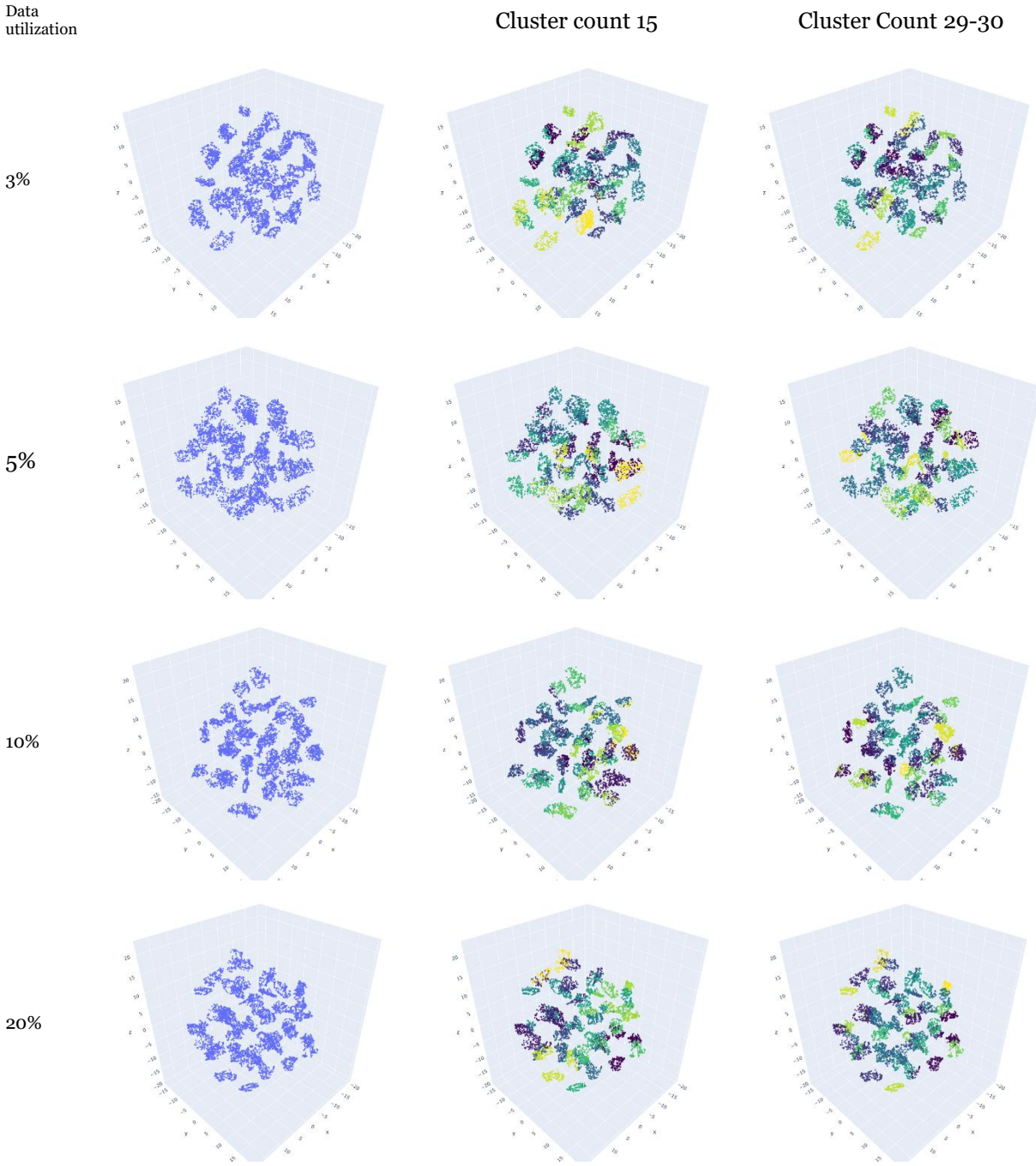


Figure 4-171 Vector-based (Window + Shade(weighted)) Weights, TSNE, Meanshift, DS 02

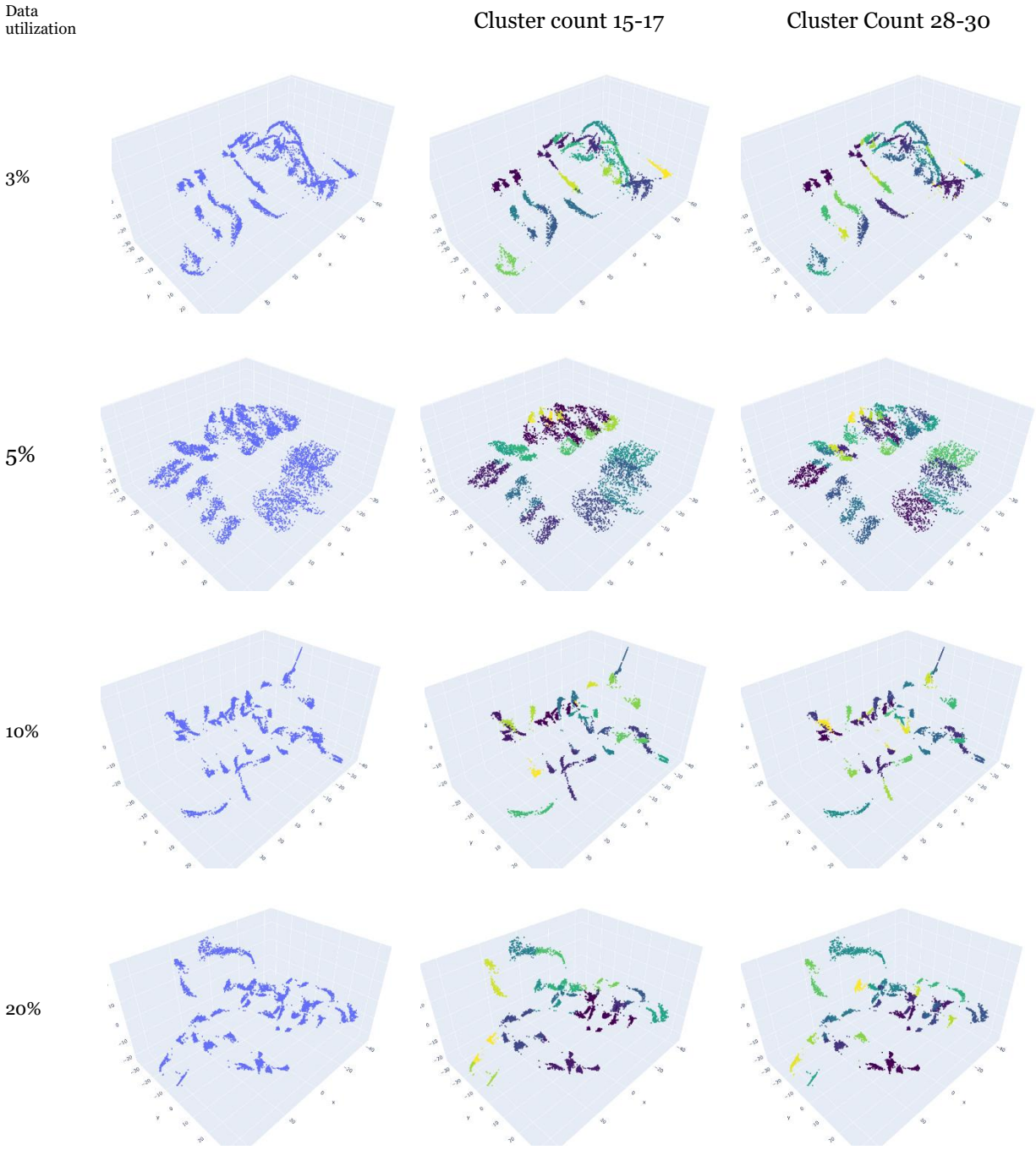


Figure 4-172 Vector-based (Window + Shade(weighted)) Weights, Isomap, Meanshift, DS 02

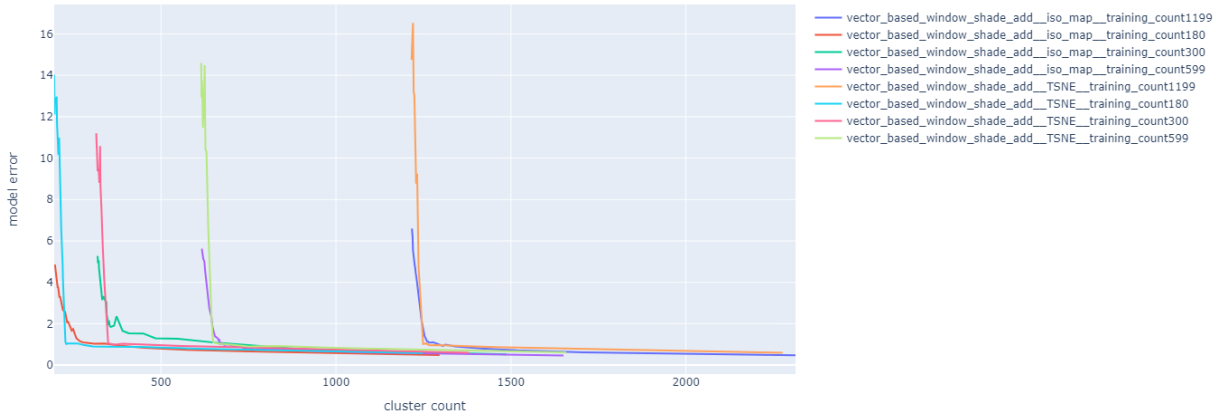


Figure 4-173 Model Error vs Cluster count Pareto Distribution, Vector-based (Window + Shade) Weights, Design Space 02

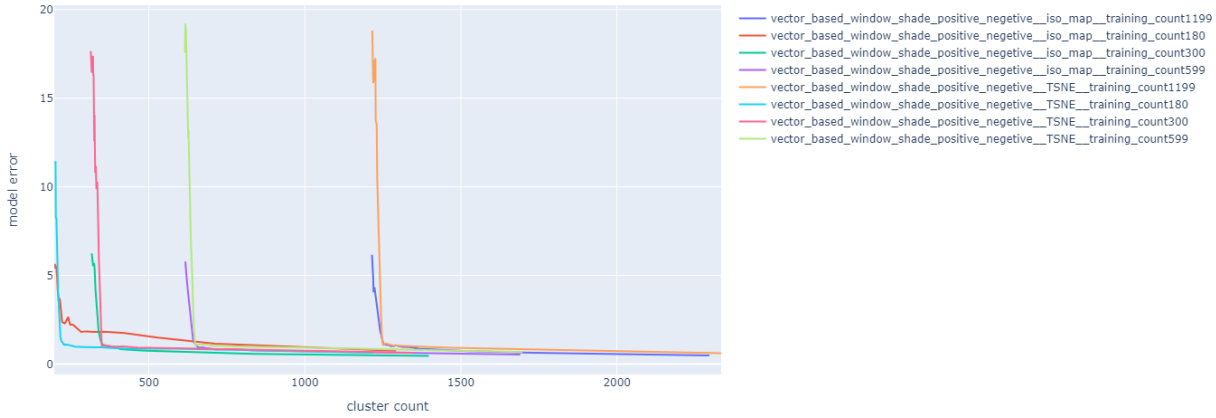


Figure 4-174 Model Error vs Cluster count Pareto Distribution, Vector-based (Window + Shade* (-1/1)) Weights, Design Space 02

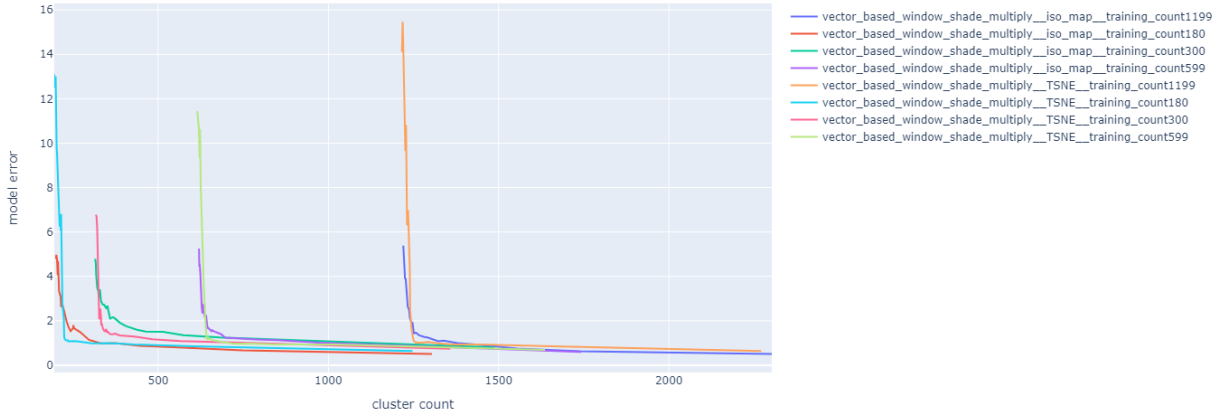


Figure 4-175 Model Error vs Cluster count Pareto Distribution, Vector-based (Window * Shade) Weights, Design Space 02

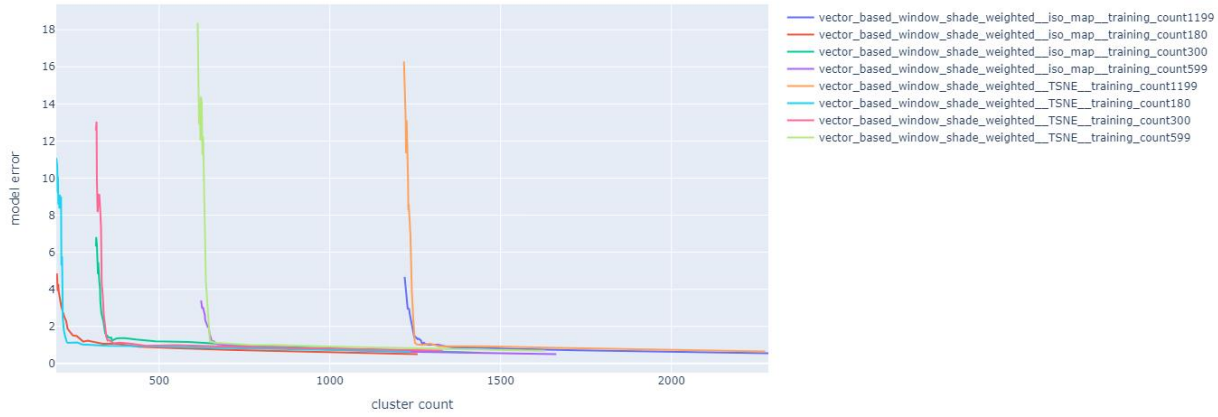


Figure 4-176 Model Error vs Cluster count Pareto Distribution, Vector-based (Window + Shade (weighted)) Weights, Design Space 02

4.3.5 Comparison between different frameworks tested for Design Space 02

The validation of the frameworks for the new design space confirms their performance and effectiveness in approximating the behavior of the design space. Among the tested frameworks, the vector-based adding weights methods, particularly the framework utilizing the sum of window-to-wall ratios (WWRs) and positively or negatively weighted shading depths, performed exceptionally well (see Figure 4-177). The specific impact of combining the two parameters, WWRs and shading depths, on the performance outcome is not yet clear and requires further investigation.

It is interesting to note that the Pareto frontier generated by the frameworks utilizing a similar amount of data is mostly identical. However, there is a noticeable spike or sudden change in the frontier after a certain cluster count, causing the lines to bend at a 90-degree angle. The reason for this behavior is not yet clear and may require additional analysis and exploration.

To further validate the proposed framework and understand its performance in more complex design spaces, Design Space 03 has been created. This new design space, which likely includes more intricate design options, will provide an opportunity to test and evaluate the framework's effectiveness in a broader range of scenarios. By conducting additional experiments and analyses in Design Space 03, a deeper understanding of the framework's capabilities and limitations can be obtained, leading to more robust and reliable results for complex design spaces.

Figure 4-177 Pareto Distribution for all tested frameworks for Design Space 02

4.3.6 Comparison between best performing framework and linear regression-based surrogate model

All of our proposed frameworks consist of four consecutive steps, which may appear more complex compared to widely used surrogate models for performance predictions. However, these frameworks offer significant benefits that can be realized when comparing prediction errors between the two systems. To evaluate the success of reducing errors, we have employed a linear regression-based surrogate model.

The accuracy of the linear regression-based surrogate model depends on the amount of data used for training. In order to approximate the simulation outcome of the entire design space, simulations need to be run for all training data points. Thus, the number of training data points can be seen as the number of simulations required to capture the behavior of the entire design space. To assess the performance, we created a Pareto frontier where the x-axis represents the number of randomly selected training data points, and the y-axis represents the mean absolute error (MAE) that can be achieved for the corresponding training inputs. It is evident that as the number of training inputs increases, the MAE decreases. However, we observed that after reaching a certain point (reducing the MAE to 9.6), further increasing the number of training inputs does not result in significant changes.

It is important to note that the MAE obtained from the linear regression model and the "model error" for any framework are not directly comparable. The linear regression model randomly selects a subset of training inputs, while the rest of the data points are used to investigate the MAE. On the other hand, the model error represents the avergae percentage of actual outcomes that are predicted incorrectly for all data points in the design space. As the average annual energy consumption for the design space 02 is around 100 Kilowatts hours (kWh) per square meter we can say model error closely resembles MAE. But as the model error is the averaging MAE for the whole design space. There is a possibility that averaging the errors of fewer data points within a framework can further reduce the model error.

Despite the differences in their characteristics, we plot the Pareto frontier of our best-performing framework and the "MAE vs. training count" Pareto frontier of the linear regression-based surrogate model. By doing so, we observe a significant reduction in prediction errors achieved by our proposed framework (see Figure 4-178). This comparison validates the efficacy of our frameworks. While the surrogate model could potentially employ different machine learning models and explore hyperparameters for further reduction in MAE, due to time constraints, we have only utilized linear regression in this study.

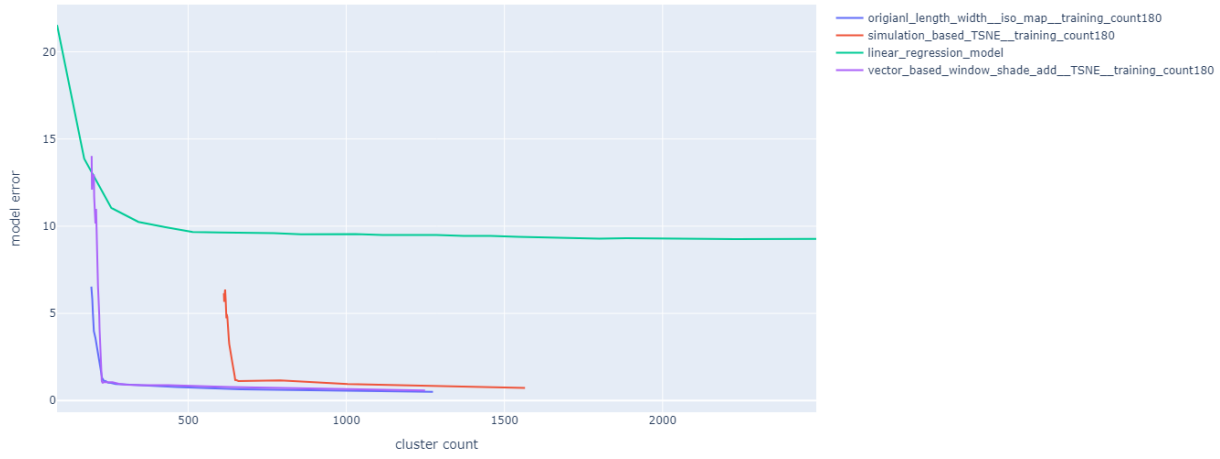


Figure 4-178 Comparison Between best performing framework and surrogate model

4.4 Framework Outcomes of Design Space 03:

The utilization of a distinct data structure in generating design space 03 presents a unique set of challenges for our proposed frameworks. Validating these frameworks by applying them to a different data structure highlights the advantages of data discretization in accurately predicting annual energy consumption. In real-world scenarios, designs are seldom as simplistic as depicted in this paper. Consequently, conducting experiments on this final framework uncovers the potential of these frameworks in more complex design spaces.

4.4.1 No added weights results

While having no weights added with the data frame, the model exhibits high errors, and the Pareto frontier generated by these frameworks consistently appears at the top, similar to the previous experiments conducted without any added weights. However, due to the increased complexity of this new design space, the model errors for different cluster counts are higher compared to the previous design space. In the previous design space, the highest model error ranged between 20 to 25, whereas in this new design space, it ranges from 25 to 30. This increase in overall model error is also evident in all the other tested frameworks (see Figure 4-185 (*Training count 164 refers to 3%, 273 refers to 5%, 547 refers to 10% 1094 refers to 20%)*)

Adding weights method	Dimensionality reduction	Clustering method	% of data used	See figure	comments
No added weights	TSNE	Meanshift	N/A	Figure 4-179	Similar to the previous experiments, the data points in this framework are clustered within a large spherical cluster. The Pareto frontier generated by this framework is positioned higher than the framework that did not involve weight using isomap.
	Isomap embedding		N/A	Figure 4-182	The data points in this scenario are scattered within a large, amorphous cluster, with no identifiable smaller clusters within it.

Cluster Count 13

Cluster Count 29-31

TSNE

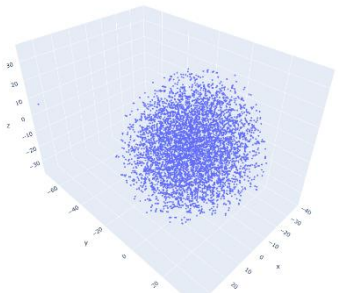


Figure 4-179 No Added Weights, TSNE, Meanshift, DS 03

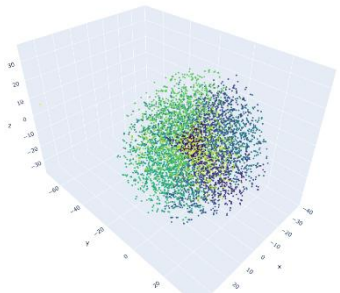


Figure 4-180 No Added Weights, TSNE, Meanshift, DS 03, Cluster count 13

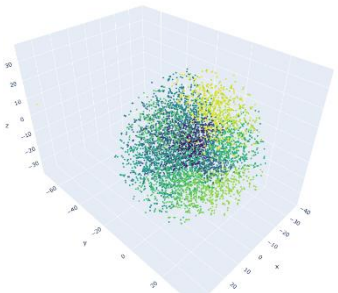


Figure 4-181 No Added Weights, TSNE, Meanshift, DS 03, Cluster count 31

Isomap

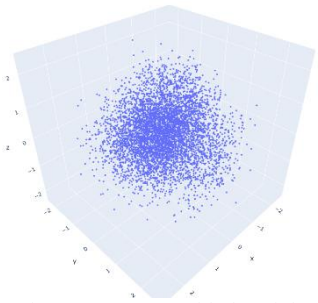


Figure 4-182 No Added Weights, Isomap, Meanshift, DS 03

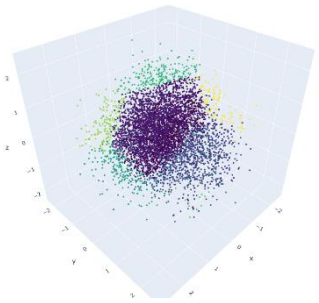


Figure 4-183 No Added Weights, Isomap, Meanshift, DS 03, Cluster Count 13

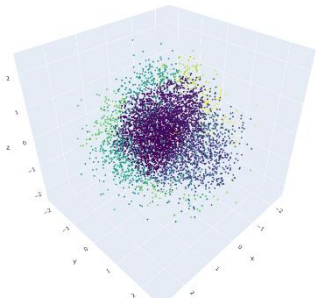


Figure 4-184 No Added Weights, Isomap, Meanshift, DS 03, Cluster Count 29

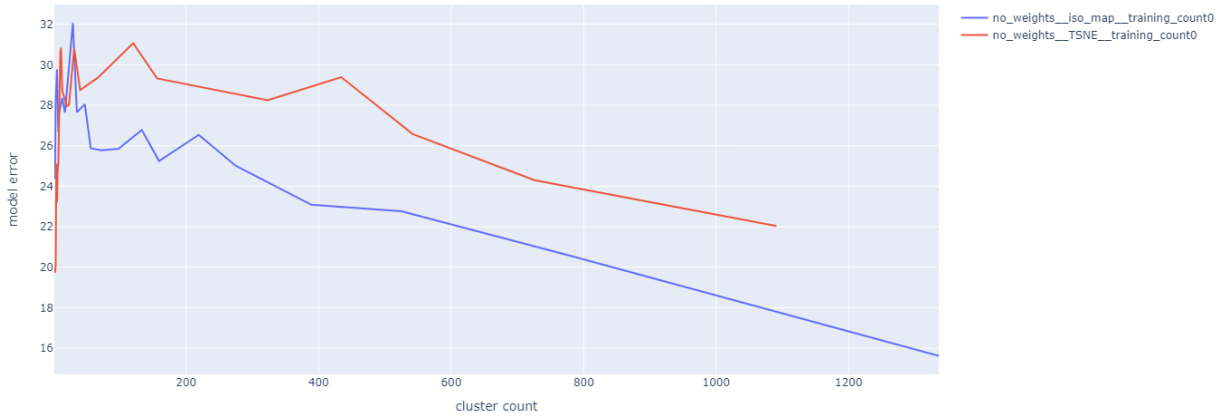


Figure 4-185 Model Error vs Cluster count Pareto Distribution, No Added Weights, Design Space 03

4.4.2 Parameter-based weights results

The parameter-based adding weights methods also performed well for this new design space. The shape of the Pareto frontier differed when using all extracted weights compared to using only weights for the total floor area of the building. relying solely on the weights for the area did not yield significant benefits for this particular design space (see Figure 4-191). In fact, when combining area weights with TSNE, the performance of that specific framework was relatively poor compared to the other approaches.

In contrast to the previous design spaces, where using a larger dataset for training the linear regression model did not result in a significant reduction in model error, for this design space, more accurate weights proved to be extremely beneficial in reducing model error (see Figure 4-186). The framework that utilized only 3% of the data achieved the lowest model error of 2.39, while the framework that utilized 20% of the data achieved the lowest model error of 1.41.

Adding weights method	Dimen-sionality reduction	Clustering method	See figure	% of data used	comments
	TSNE	Meanshift	Figure 4-187	3%	Data points are distributed within 5-6 distinct clusters. Generated Pareto frontier closely

Parameter-based weights					resembles the framework used isomap after utilizing the same amount of dataset.
				5%	Distinct clusters visualized in the previous framework tend to group within smaller clusters. The generated Pareto frontier by the framework is positioned slightly higher compared to the framework that utilized isomap after utilizing the same amount of dataset.
				10%	data points are distributed in small linear-shaped clusters. These smaller clusters are further grouped together within a larger spherical cluster. Surprisingly this framework performed better in reducing model error than the framework that utilized isomap with the same amount of data utilization.
				20%	The arrangement of data points in lower dimensional space looks the same as the framework utilized only 10% of the dataset with TSNE. Model error is lower for this framework, Generated Pareto frontier is closely identical to the framework used isomap with the same amount of data utilization.
	Isomap embedding		Figure 4-188	3%	Data points are distributed in clusters of different shapes. Generated Pareto frontier closely resembles the framework used TSNE after utilizing the same amount of dataset.
				5%	Data points tend to densify in more small clusters, indicating the beginning stage of making a proper arrangement.
				10%	Data points are well organized in linear-shaped clusters. Small clusters are distributed in 3-4 larger clusters.
				20%	The arrangement of data points in lower dimensional space looks the same as the framework utilized only 10% of the dataset with isomap, however, distinct groups of large clusters are subdivided into more smaller groups, making the clusters more identifiable.
Parameter-based area weight	TSNE		Figure 4-189	3%	Data points are positioned in roughly identifiable 4-5 large clusters. Model error for this framework is extremely higher than the framework used isomap with a similar amount of data utilization.
				5%	This framework arranges data points within roughly identifiable large clusters. Like the previous framework, the generated Pareto Frontier is positioned significantly higher compared to the framework that utilized the isomap with a similar amount of data utilization.
				10%	Data points are distributed in an amorphous-shaped large cluster. Surprisingly the utilization of a larger subset of the data contributed to a rise in model error.

				20%	Data points are interestingly distributed in large almost spherical clusters. Model error is surprisingly high with a much higher amount of data utilization. The framework performed the worst among all frameworks that utilized parameter-based weights.
	Isomap embedding		Figure 4-190	3%	The data points appear to disperse from a densified region. Despite the dispersal, the generated Pareto frontier is positioned very close to the bottom line, indicating that the framework achieves low model errors and performs well in predicting performance.
				5%	The arrangement of data points and the shape of the Pareto frontier is almost identical to the previous framework that utilized only 3% of the data with isomap.
				10%	In this design space, the data points are distributed within an amorphous-shaped cluster, lacking any distinct patterns or smaller clusters. Despite the absence of clear clusters, this framework exhibits lower model errors.
				20%	Data points are linearly distributed with a dignified potion at one side. Surprisingly the utilization of a larger dataset resulted in an increased model error. The position of the Pareto frontier is comparatively higher than other frameworks utilized area weights with isomap.

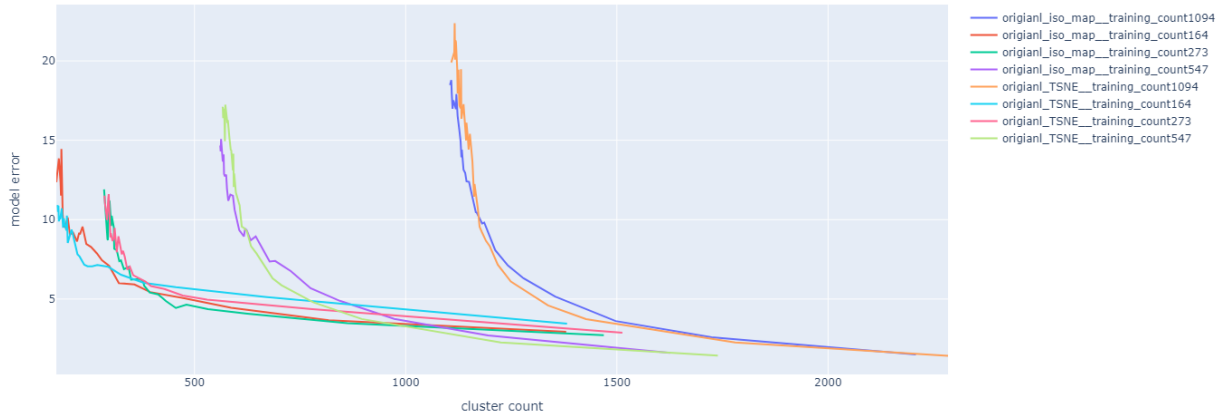


Figure 4-186 Model Error vs Cluster count Pareto Distribution, Parameter-based Weights, Design Space 03

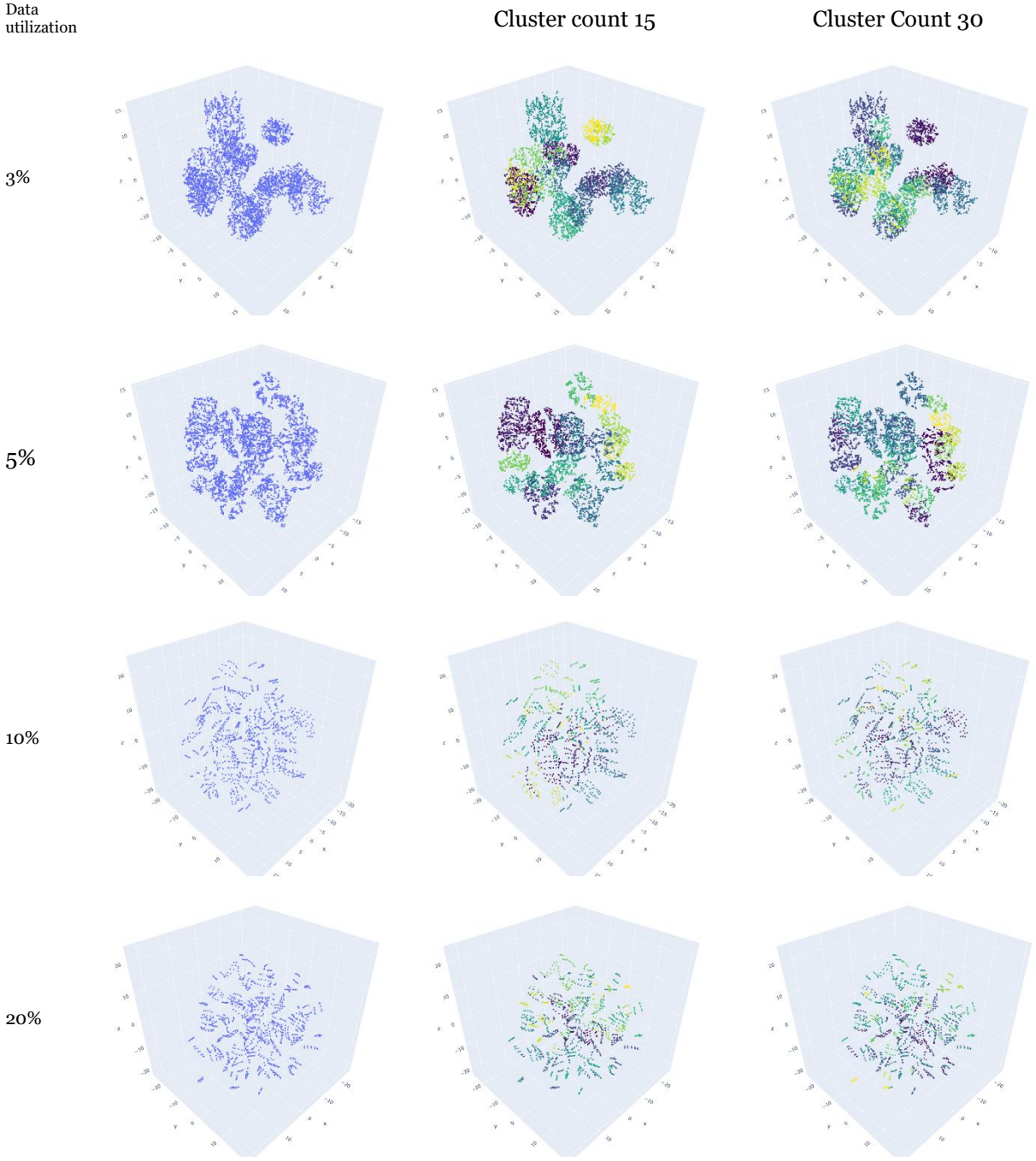


Figure 4-187 Parameter-based Weights, TSNE, Meanshift, DS 03

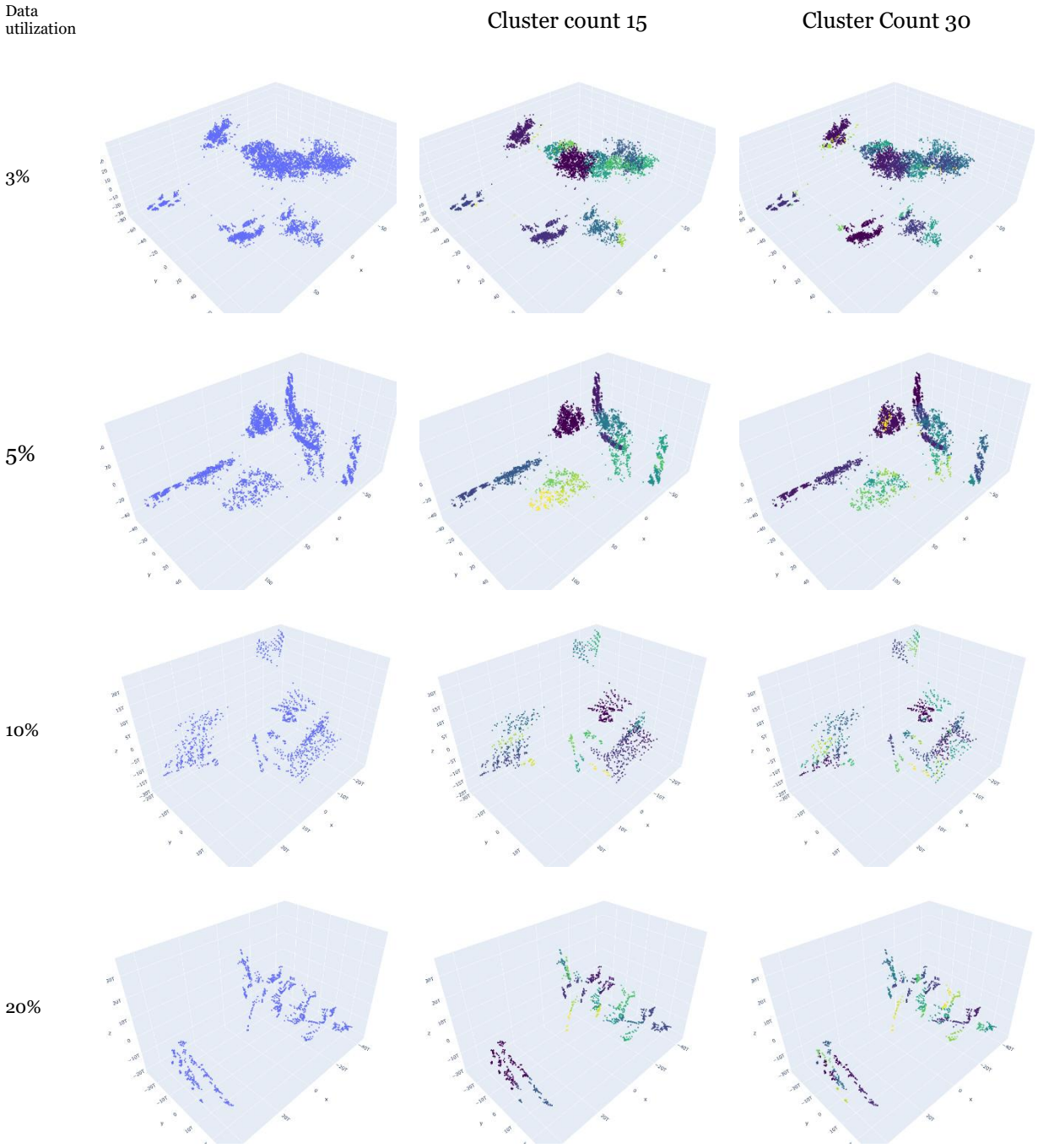


Figure 4-188 Parameter-based Weights, Isomap, Meanshift, DS 03

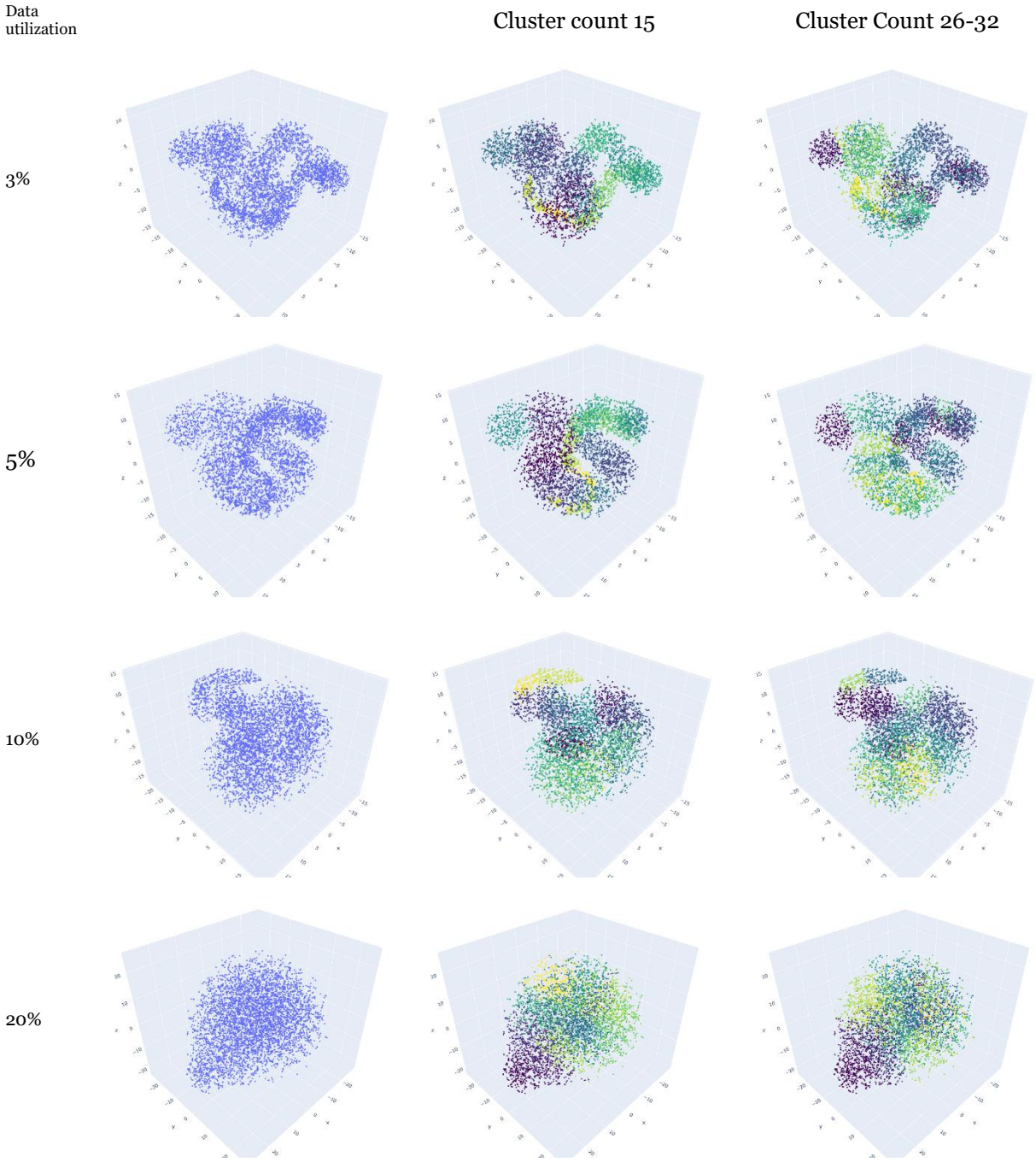


Figure 4-189 Parameter-based area Weight, TSNE, Meanshift, DS 03

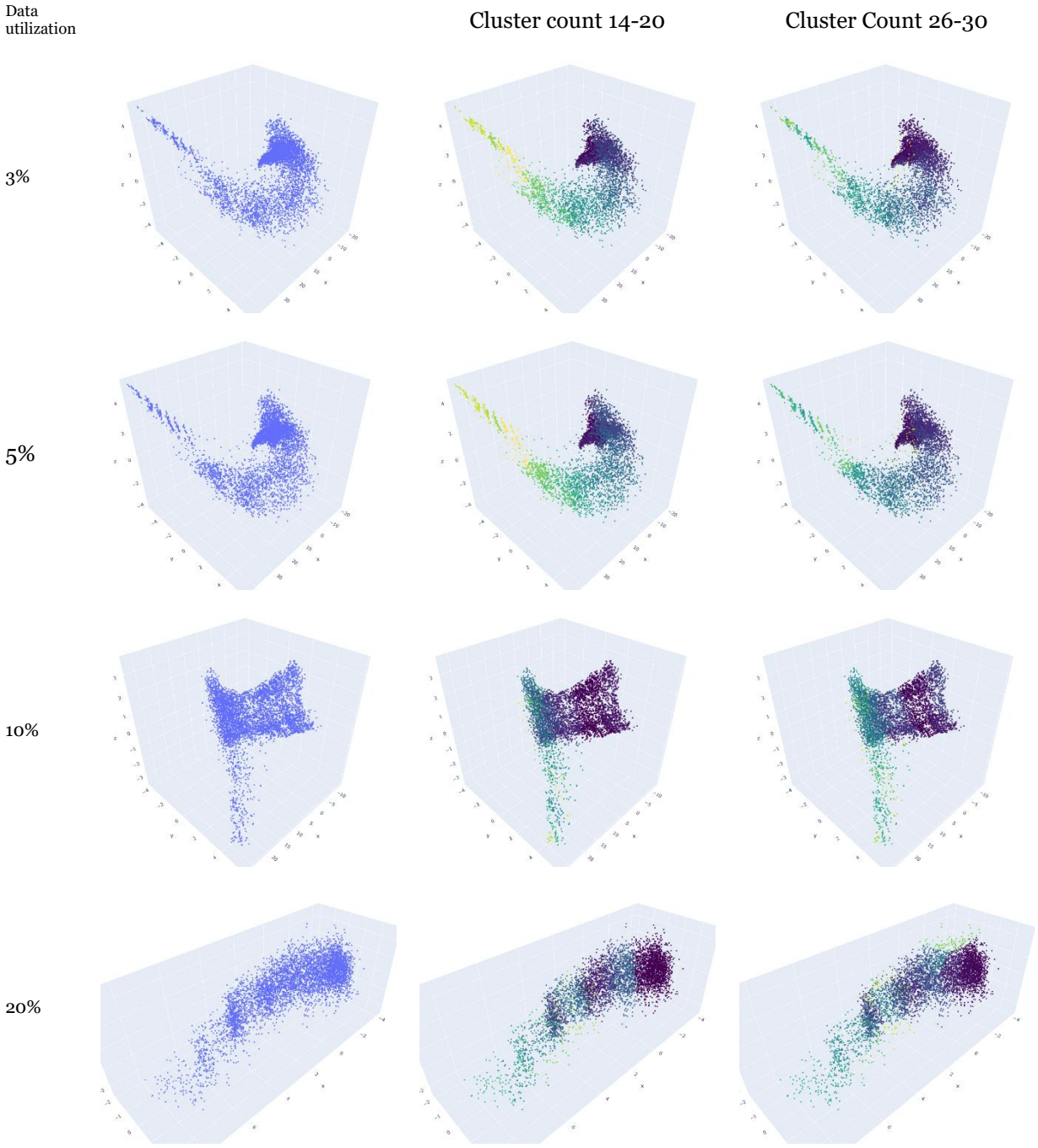


Figure 4-190 Parameter-based area Weight, Isomap, Meanshift, DS 03

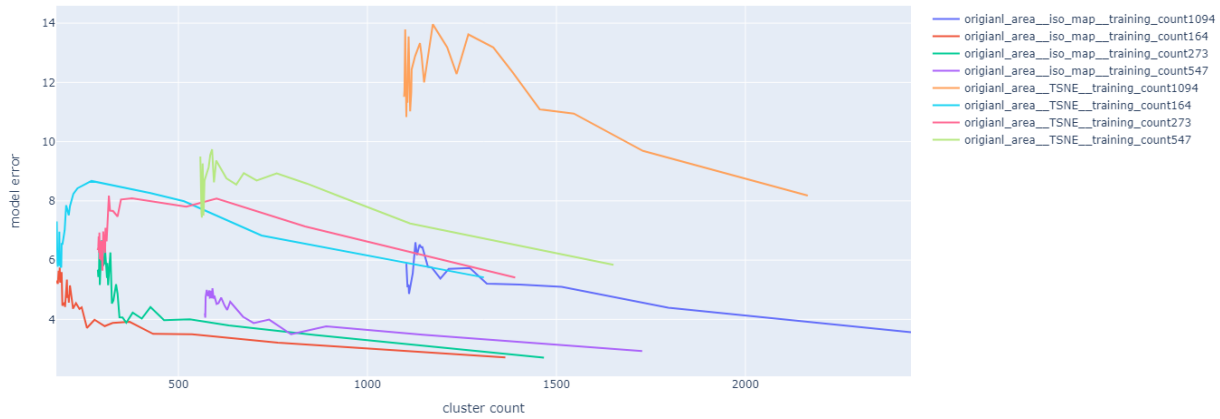


Figure 4-191 Model Error vs Cluster count Pareto Distribution, Parameter-based area Weight, Design Space 03

4.4.3 Simulation-based weights results

The simulation-based adding weights method resulted in ambiguous outcomes, making it challenging to derive meaningful insights. When all extracted weights from 'Test design space 01' were used, the performance of the framework was extremely poor, similar to frameworks that did not incorporate weights in the data frame. This highlights the limitations of using simulation-based extracted weights for multiple design spaces with different data structures. However, when only the weight for the total floor area of the building was utilized, the generated Pareto frontier showed a downward trend, and the frameworks produced lower model errors (see Figure 4-204). Further investigation is required in such scenarios. A more careful extraction of weights from any test design space may reveal their potential benefits for multiple applications.

Adding weights method	Dimensionality reduction	Clustering method	% of data used	See figure	comments
Simulation-based weights	TSNE	Meanshift	N/A	Figure 4-192	Data points are distributed in large spherical clusters same as the frameworks that did not include any weights. Generated Pareto frontier is also the same as the frameworks utilized with no weights.
	Isomap embedding		N/A	Figure 4-195	Data points are distributed in large clusters where no other distinct small clusters are visible.
Simulation-based area weight	TSNE		N/A	Figure 4-198	The data points in this framework are arranged in a similar manner as the frameworks that utilized all simulation-based weights. However, the generated Pareto frontier is positioned slightly lower than those frameworks.
	Isomap embedding		N/A	Figure 4-201	Data points are scattered in an amorphous shape. Surprisingly the generated Pareto frontier is located very close to the bottom line, inducing reduced model error.

TSNE

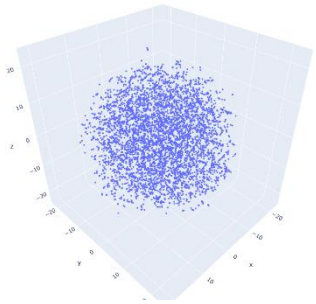


Figure 4-192 Simulation-based Weights, TSNE, Meanshift, DS 03

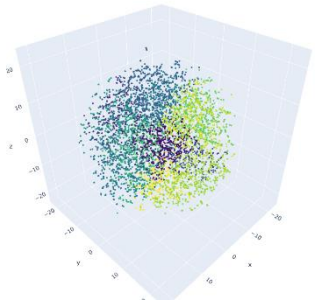


Figure 4-193 Simulation-based Weights, TSNE, Meanshift, DS 03, Cluster count 14

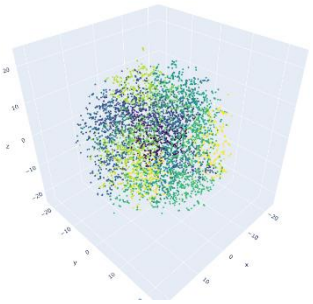


Figure 4-194 Simulation-based Weights, TSNE, Meanshift, DS 03, Cluster count 29

Isomap

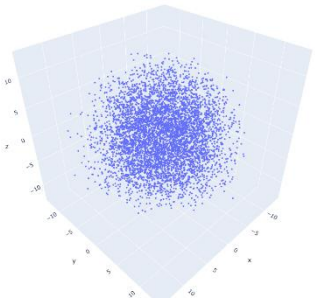


Figure 4-195 Simulation-based Weights, Isomap, Meanshift, DS 03

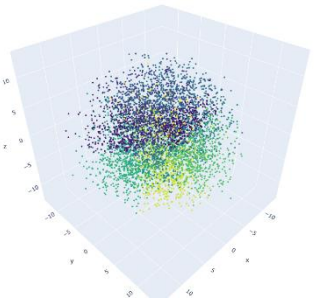


Figure 4-196 Simulation-based Weights, Isomap, Meanshift, DS 03, Cluster count 15

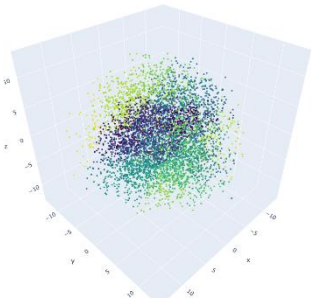


Figure 4-197 Simulation-based Weights, Isomap, Meanshift, DS 03, Cluster count 30

TSNE

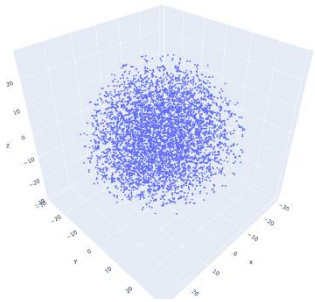


Figure 4-198 Simulation-based area Weight, TSNE, Meanshift, DS 03

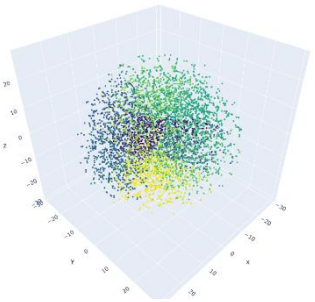


Figure 4-199 Simulation-based area Weight, TSNE, Meanshift, DS 03, Cluster count 15

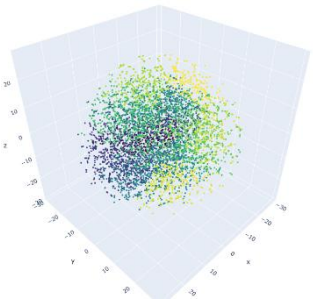


Figure 4-200 Simulation-based area Weight, TSNE, Meanshift, DS 03, Cluster count 25

Isomap

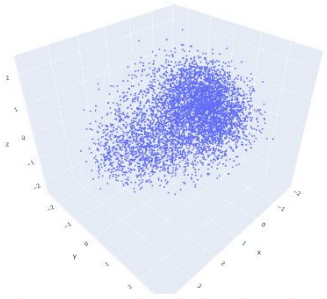


Figure 4-201 Simulation-based area Weight, Isomap, Meanshift, DS 03

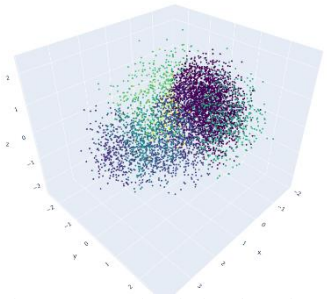


Figure 4-202 Simulation-based area Weight, Isomap, Meanshift, DS 03, Cluster count 14

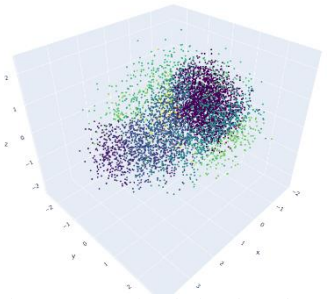


Figure 4-203 Simulation-based area Weight, Isomap, Meanshift, DS 03, Cluster count 27

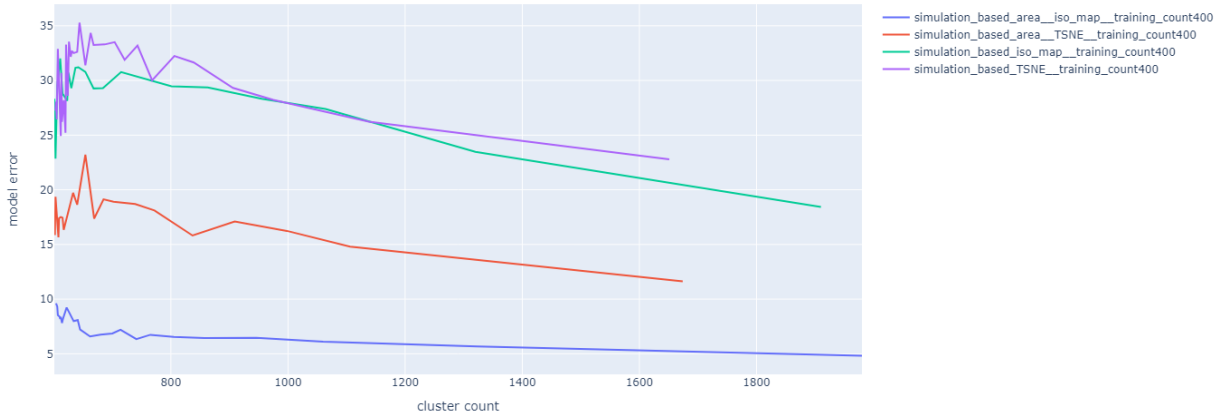


Figure 4-204 Model Error vs Cluster count Pareto Distribution, Simulation-based Weights, Simulation-based area Weight, Design Space 03

4.4.4 Vector-based weights results

The vector-based adding weights method outperformed all other tested frameworks in this design space. Unlike the previous design space, where most of the Pareto frontiers were closely similar, in this case, the Pareto frontiers are positioned with a larger gap between them. The arrangement of data points in lower dimensions is completely different from the frameworks used in design space 02. Although distinct clusters were not identifiable in this scenario, incorporating weights successfully reduced the model error significantly. The frameworks that utilized separate vectors for different façades showed better performance compared to the frameworks that combined all vectors into four. By using separate vectors, more information was included while reducing the dimensionality of the data frame, resulting in higher accuracy in predicting performance outcomes.

Increasing the amount of training data points for the linear regression model and incorporating more accurate weights did not significantly improve the performance of the frameworks in most of the cases. This suggests that even with a smaller amount of data, the frameworks were able to make accurate predictions. When the depth of shades is positively or negatively weighted, the performance of those frameworks dropped.

Adding weights method	Dimen-sionality reduction	Clustering method	See figure	% of data used	comments
Separated Vector-based (window + shade) weights	TSNE	Meanshift	Figure 4-205	3%	Data points are positioned in roughly identifiable large clusters. The generated Pareto frontier is located close to the bottom line.
				5%	Data points started grouping in more distinct clusters. The shape of the generated Pareto frontier remained the same as the framework utilized 3% of the dataset.
				10%	There are fewer discernible clusters in the data, and the shape of the generated Pareto frontier remained unchanged.
				20%	The data points are arranged neatly in small clusters. Interestingly, the generated Pareto frontier is positioned slightly higher than anticipated.
	Isomap embedding		Figure 4-206	3%	Data points are distributed in two amorphous-shaped clusters. All the frameworks utilized isomap outperformed the frameworks used TSNE. The position of the generated Pareto frontier is low, offering reduced model error.
				5%	The arrangement of data points has undergone slight changes, with previously distinct clusters appearing to merge with one another. The position of the generated Pareto frontier is lower compared to the framework that utilized the same amount of dataset with TSNE.
				10%	The data points exhibit an uneven distribution within two distinct clusters. The generated Pareto frontier remains unchanged, positioned slightly higher than anticipated.
				20%	The arrangement of the data points is identical to the framework utilized 10% of the data with isomap. The shape of the Pareto frontier remains the same.
Separated Vector-based (window + shade * (1/-1)) weights	TSNE		Figure 4-207	3%	Data points are distributed in small distinct clusters. The position of the generated Pareto frontier is comparatively high.
				5%	Similar to previous framework data points are organized in large clusters. The generated Pareto frontier closely resembles the framework utilized in 3% of the data with TSNE.
				10%	The data points in this framework also exhibit an organized distribution within large clusters. The shape of the generated Pareto frontier remains consistent.
				20%	The data points exhibit a well-organized distribution within distinct clusters. This framework has achieved a reduced model

					error, resulting in a Pareto frontier that is positioned relatively lower.
	Isomap embedding		Figure 4-208	3%	The arrangement of the data points in this framework is comparable to the other frameworks using isomap embedding. The data points are observed to be distributed among three distinct clusters. Notably, the generated Pareto frontier closely resembles the one produced by the framework that combined vectors, employed positive/negative weights, and utilized isomap embedding with the same amount of data utilization.
				5%	The arrangement of the data points is almost the same with minor changes. This framework performed better than the one that used a similar technique with TSNE.
				10%	The arrangement of the data points in this framework exhibits a consistent pattern, the generated Pareto frontier demonstrates a similar shape to the previous one.
				20%	Though the arrangement of the data points looks unchanged this framework offered more reduced model error with a Pareto frontier that is positioned relatively lower.
Vector-based (window * shade) weights	TSNE		Figure 4-209	3%	Data points are distributed within amorphous-shaped clusters, generated Pareto frontier is located slightly higher than the framework utilized separated vectors with TSNE and a similar amount of data utilization.
				5%	Amorphous clusters started organizing themselves with larger gaps. The shape of the Pareto frontier closely resembles to the previous one.
				10%	The arrangement of the data points remains almost the same. More accurate weights contributed to reduced model error.
				20%	Data points are organized in roughly identifiable big clusters. This framework offered reduced model error due to utilizing a larger amount of dataset.
	Isomap embedding		Figure 4-210	3%	The arrangement is the same as other frameworks used isomap embedding. Compare to the framework used TSNE, the generated Pareto frontier position is much lower.
				5%	Large clusters started merging with each other. Generated Pareto frontier closely resembles the previous one.
				10%	The arrangement of the data points remains unchanged, but the framework offered lower model error than the previous one.
				20%	The arrangement of the data points looks consistent with generated Pareto frontier close to the bottom line.

Vector-based (window + shade * (1/-1)) weights	TSNE		Figure 4-211	3%	In this framework, the data points exhibit a clear organization into large clusters, similar to the framework that utilized separated vectors with the same steps in the process. The generated Pareto frontier closely resembles the one obtained from the aforementioned framework.
				5%	Data points are distributed in roughly identifiable clusters. The Pareto frontier closely resembles the previous one.
				10%	The shape of the clusters became more distinct. The position of the Pareto frontier is slightly higher than anticipated.
				20%	Distinct clusters are separated from each other with larger gaps. This framework offered reduced model error due to utilizing more accurate weights.
	Isomap embedding		Figure 4-212	3%	The arrangement of data points is the same as the other frameworks used isomap embedding.
				5%	In comparison to the previous framework, there are slight changes in the distribution of data points in this framework. However, the shape of the Pareto frontier remains unchanged.
				10%	The arrangement of the data points is consistent with a Pareto frontier of similar shape.
				20%	Utilizing more accurate weights did not produce lower model error.

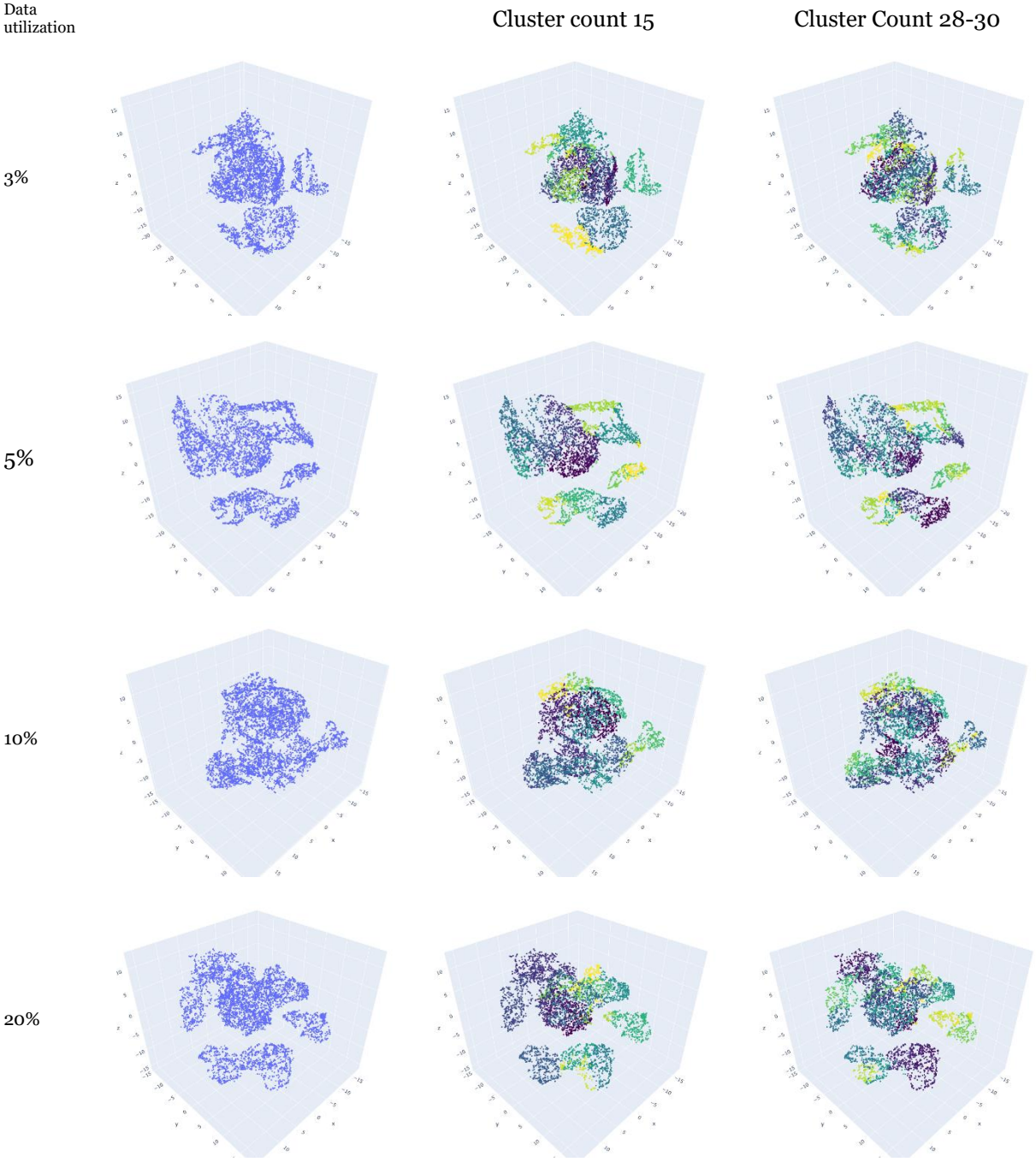


Figure 4-205 Separated Vector-based Weights (window + shade), TSNE, Meanshift, DS 03

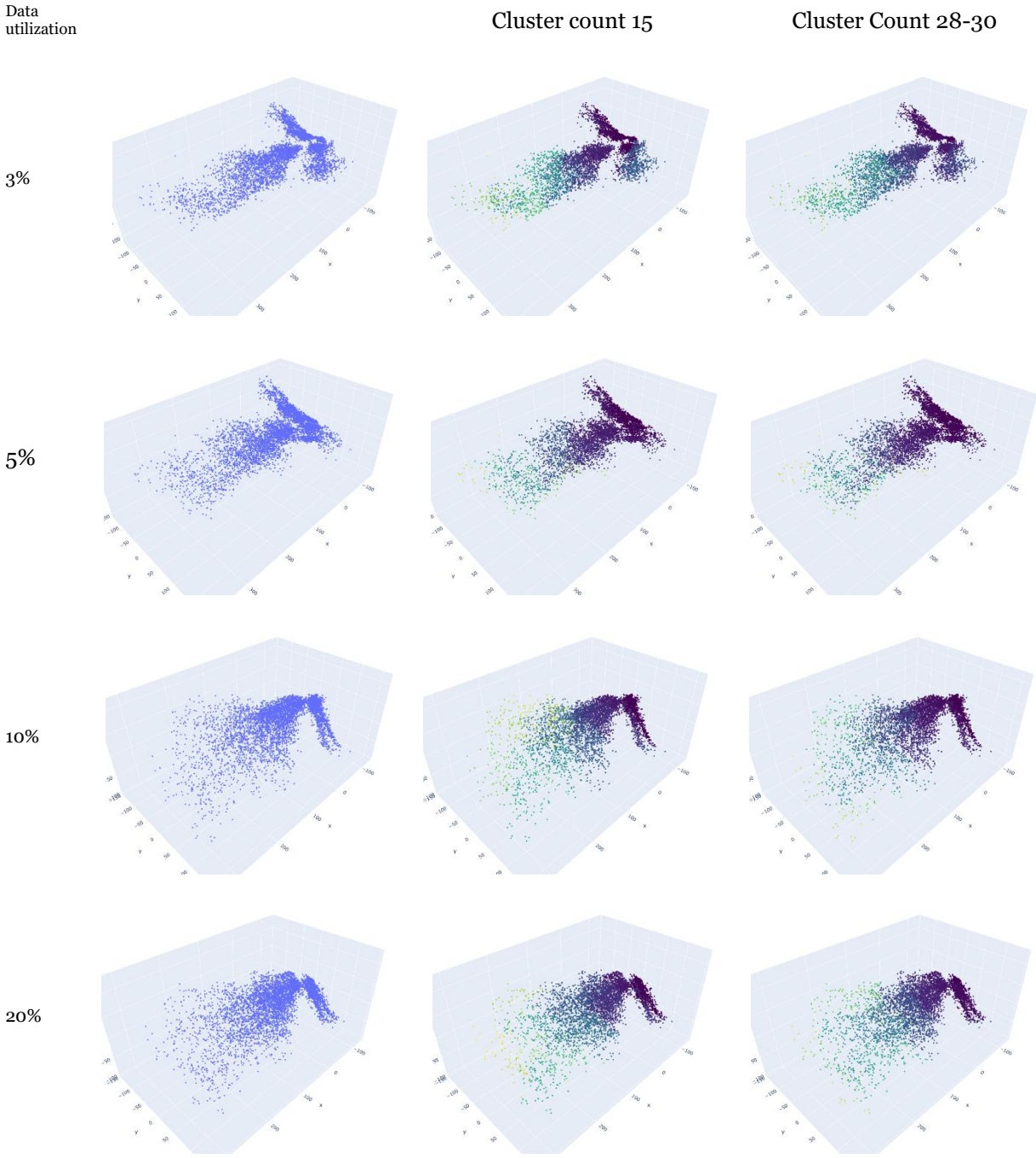


Figure 4-206 Separated Vector-based Weights (window + shade), Isomap, Meanshift, DS 03

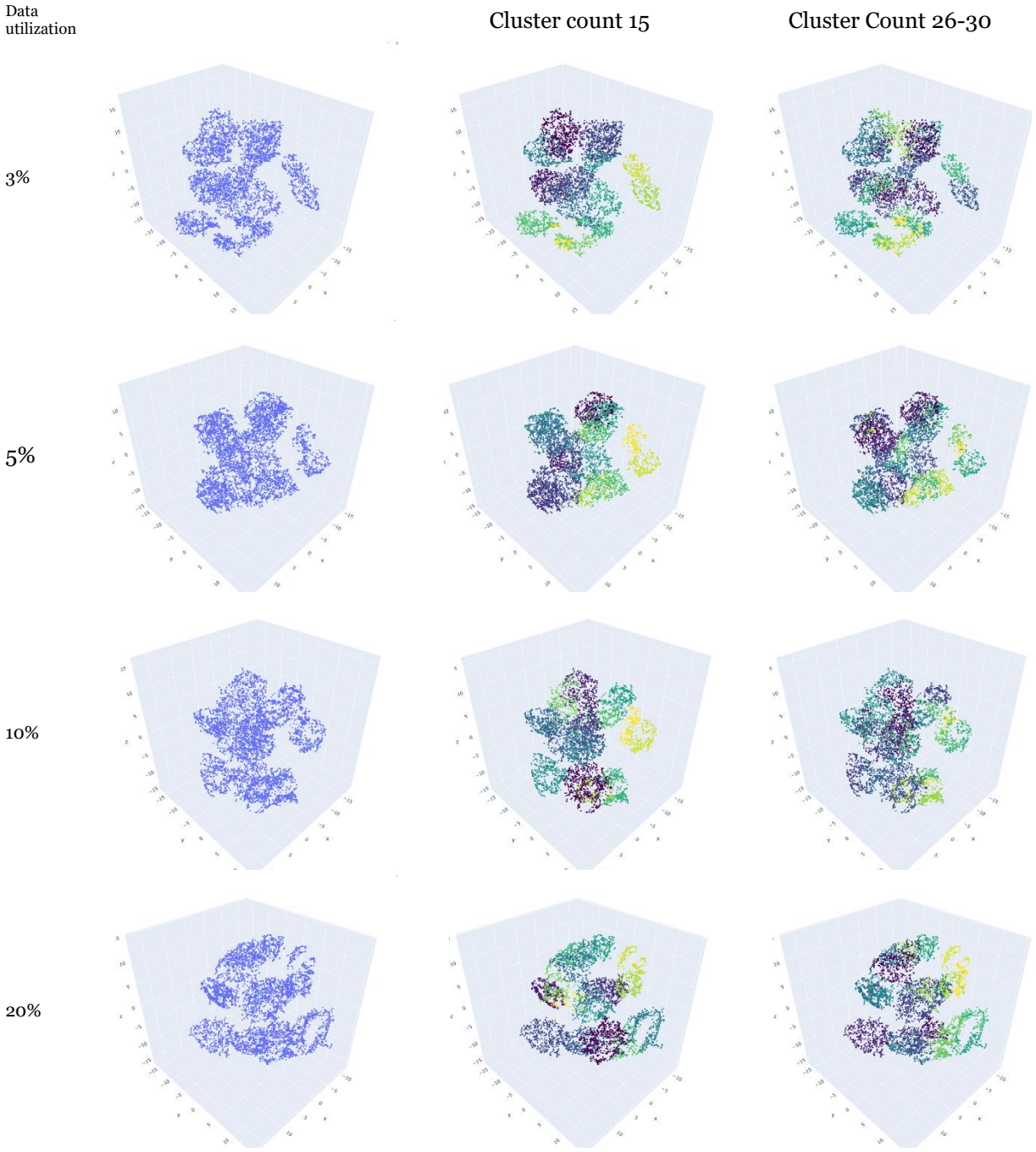


Figure 4-207 Separated Vector-based Weights (window + shade*(-1/1)), TSNE, Meanshift, DS 03

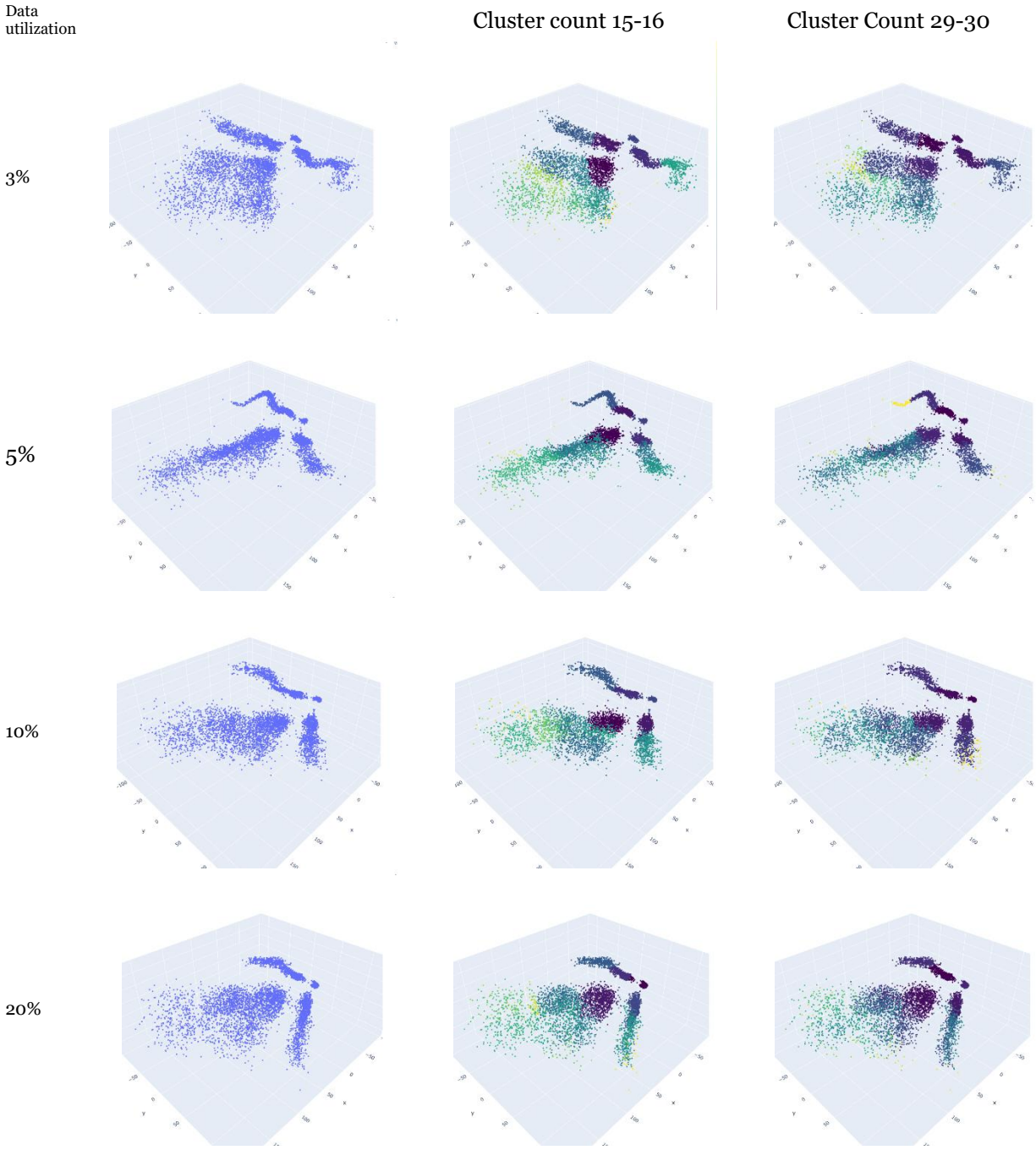


Figure 4-208 Separated Vector-based Weights (window + shade*(-1/1)), Isomap, Meanshift, DS 03

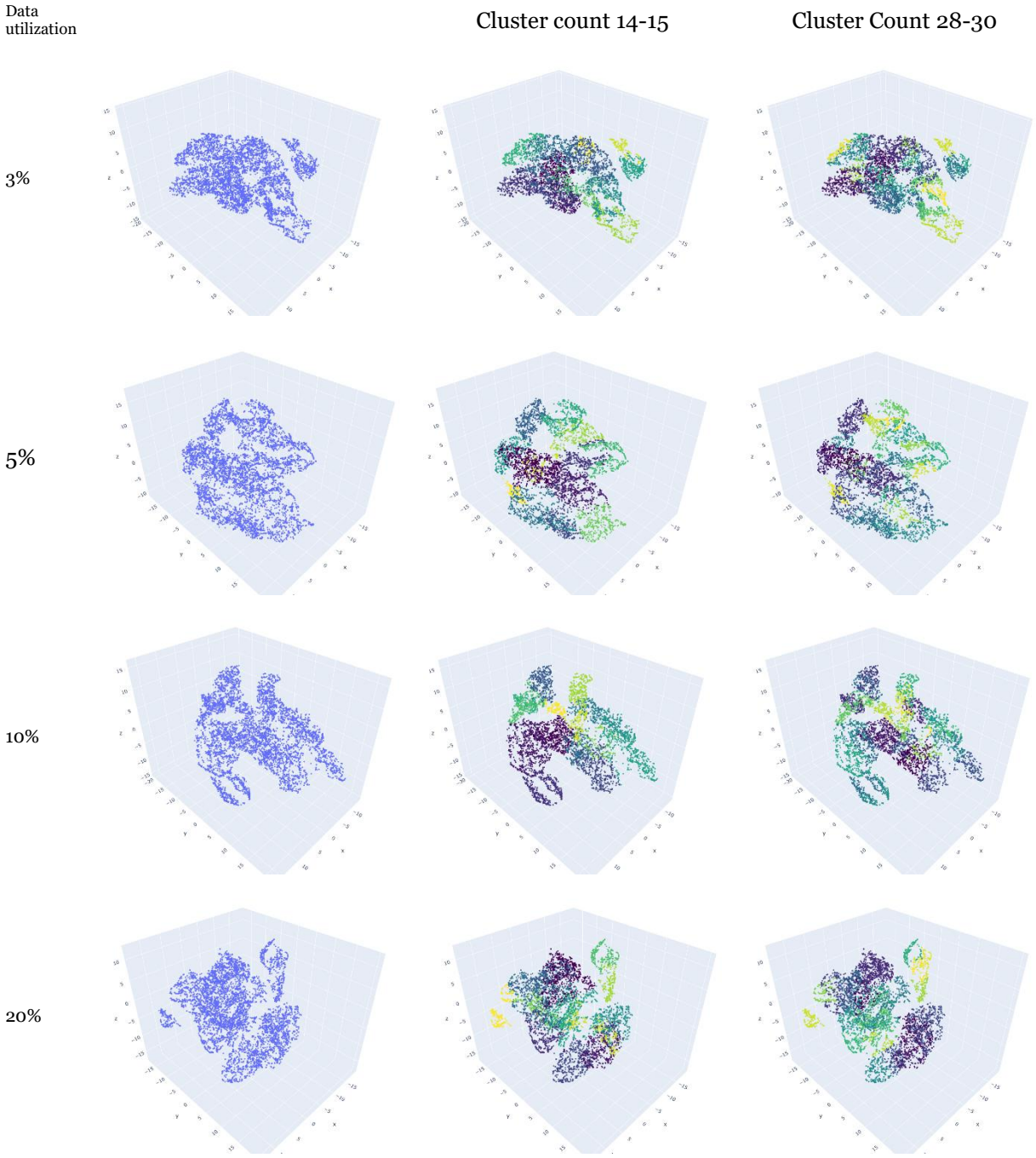


Figure 4-209 Vector-based Weights (window + shade), TSNE, Meanshift, DS 03

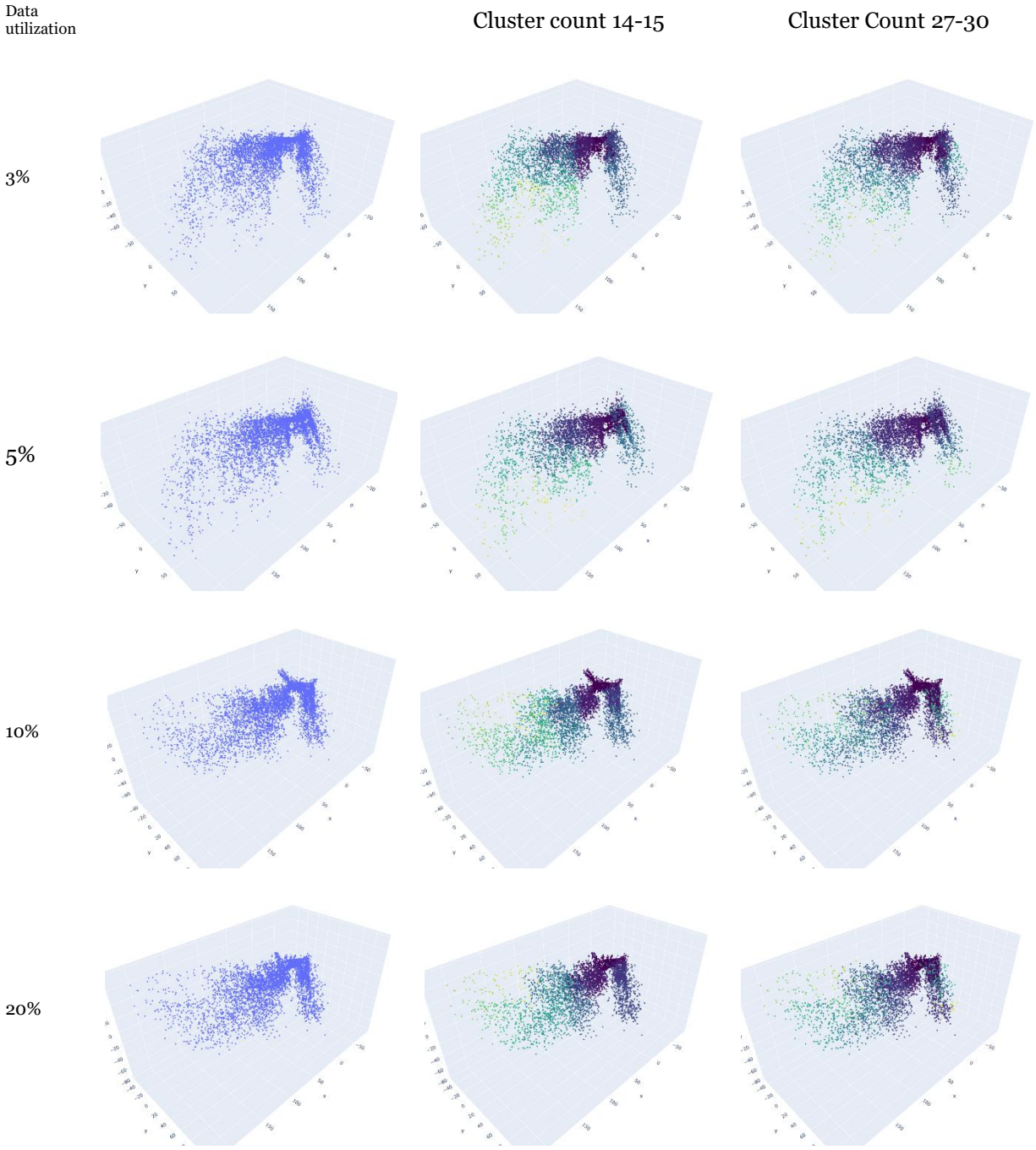


Figure 4-210 Vector-based Weights (window + shade), Isomap, Meanshift, DS 03

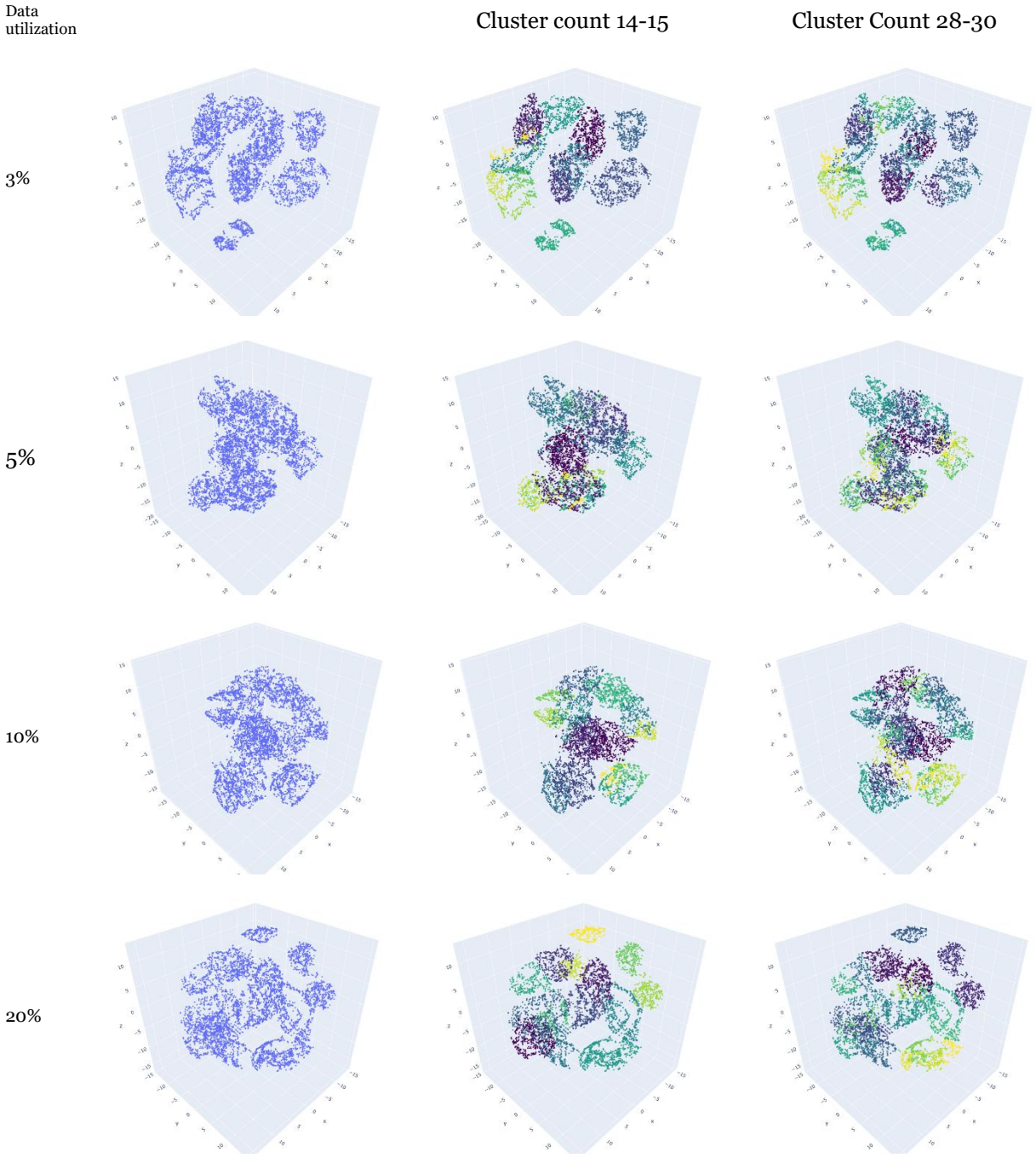


Figure 4-211 Vector-based Weights (window + shade*(-1/1)), TSNE, Meanshift, DS 03

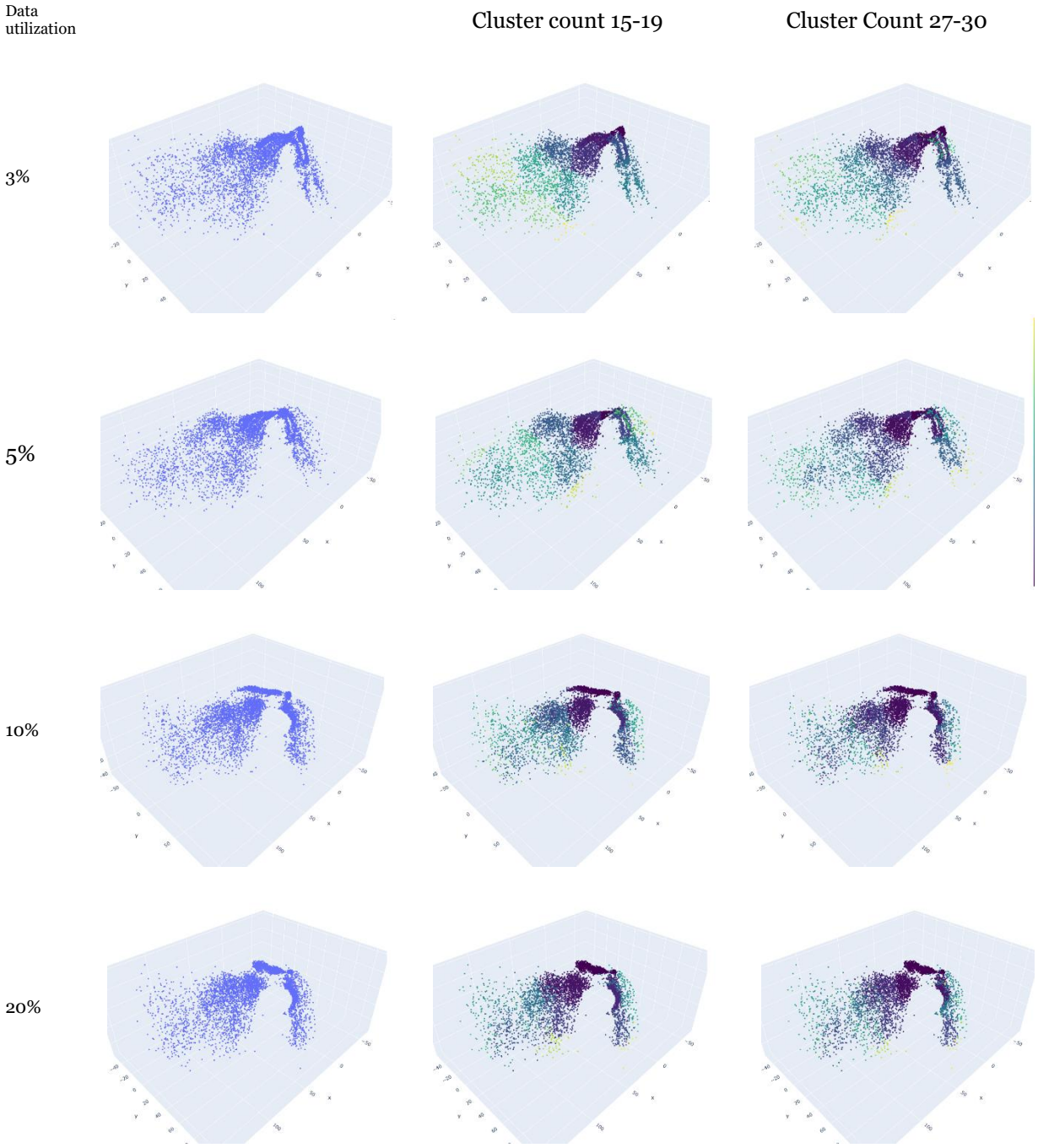


Figure 4-212 Vector-based Weights (window + shade*(-1/1)), Isomap, Meanshift, DS 03

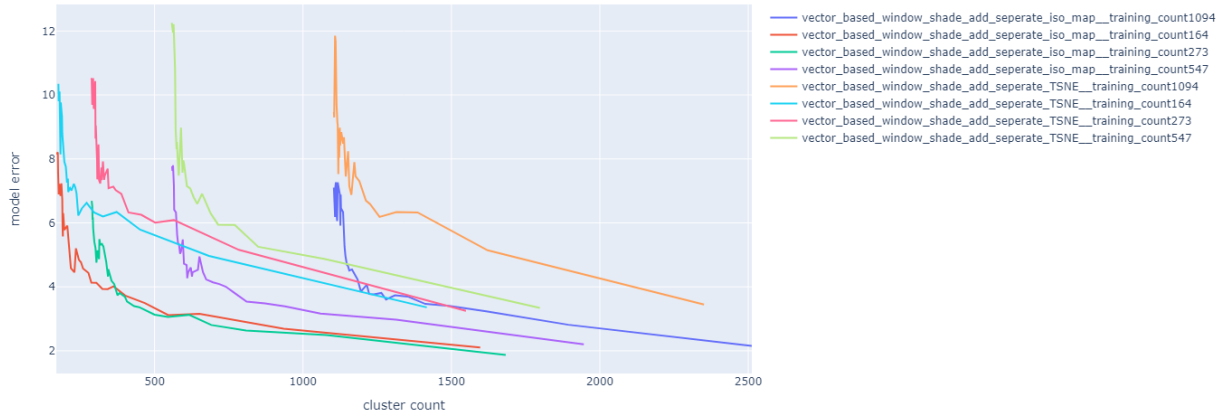


Figure 4-213 Model Error vs Cluster count Pareto Distribution, Separated Vector-based Weights (window + shade), Design Space 03

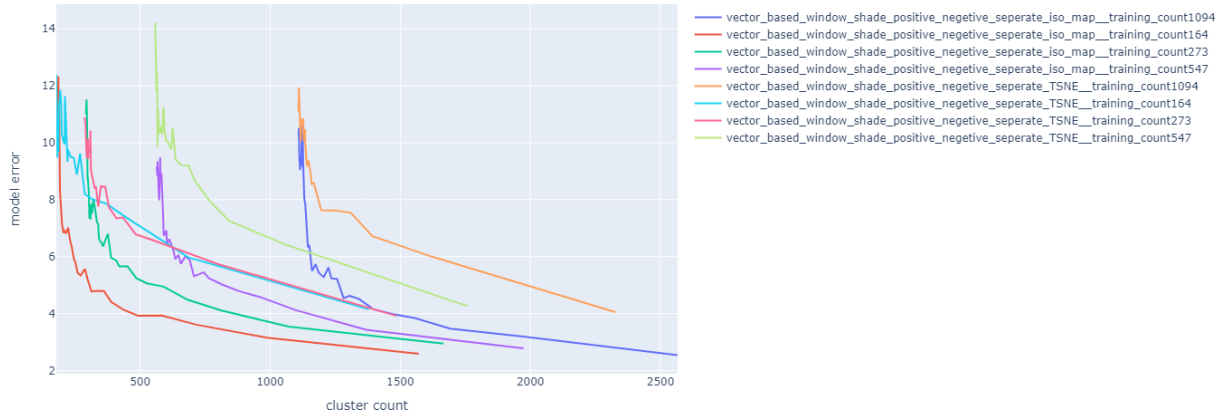


Figure 4-214 Model Error vs Cluster count Pareto Distribution, Separated Vector-based Weights (window + shade*(-1/1)), Design Space 03

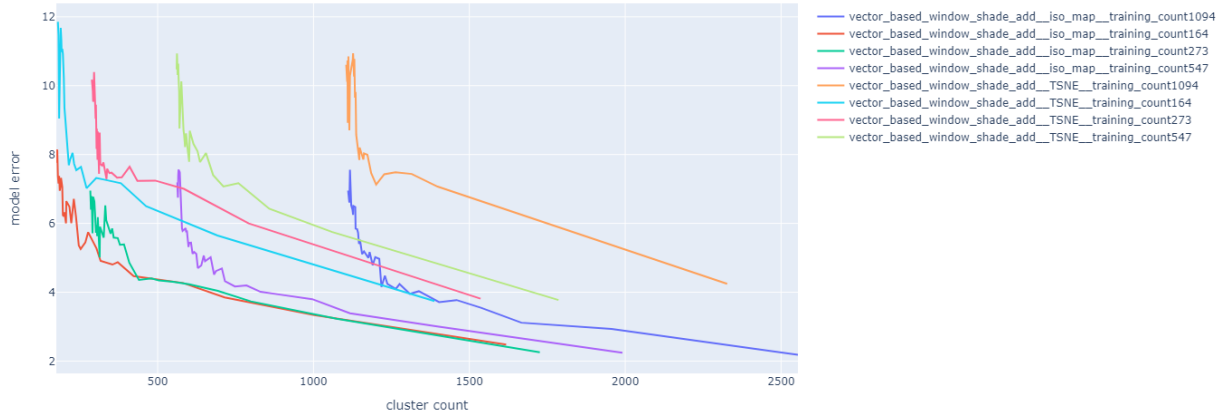


Figure 4-215 Model Error vs Cluster count Pareto Distribution, Vector-based Weights (window + shade), Design Space 03

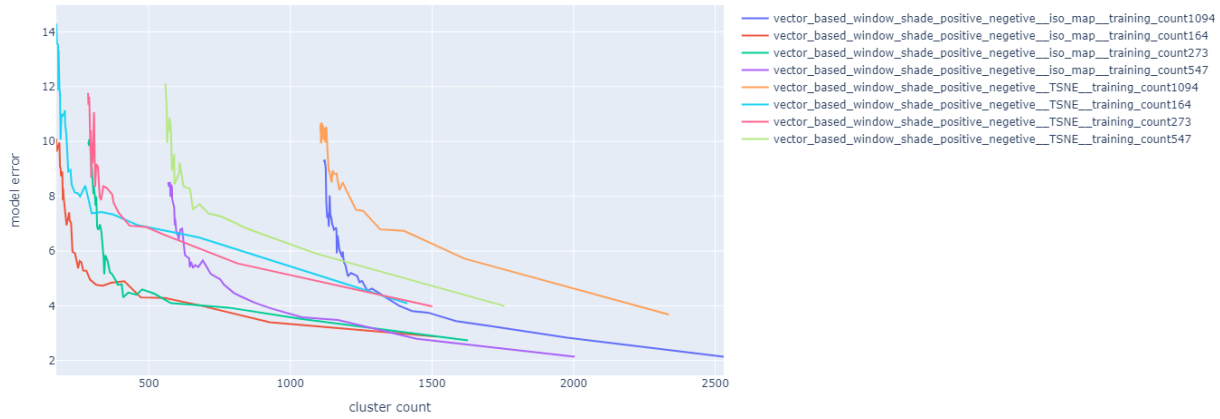


Figure 4-216 Model Error vs Cluster count Pareto Distribution, Vector-based Weights (window + shade*(-1/1)), Design Space 03

4.4.5 Comparison between different frameworks tested for Design Space 03

After evaluating all the frameworks for design space 03, it is evident that vector-based adding weights methods outperformed other frameworks (see Figure 4-217). Specifically, the framework utilizing separated vectors and considering the sum of window-to-wall ratios (WWRs) and shading depths exhibited superior performance. Among the vector-based adding weights methods, isomap embedding yielded promising results with reduced model error. This success can be attributed to the fact that the vector-based approach not only incorporates window information but also captures the building's shape while reducing dimensionality.

On the other hand, parameter-based adding weights methods ranked second in terms of achieving lower model error. Researchers can benefit from this method when applying it to more diverse design spaces. It was observed that weights extracted for the area did not contribute significantly to reducing model error. The overall shape of the building has a greater impact on performance outcomes compared to its floor area. For instance, a linear-shaped building may have ample daylight, resulting in reduced energy consumption for artificial lighting. In contrast, a building with the same square meter area can be square or rectangular, which may require more energy to illuminate its core. The simulation-based adding weights method did not yield clear results, and further investigation was not possible within the given timeline. The shape of all the Pareto frontiers extracted for this design space differed slightly from those obtained in the investigation of design space 02, as the lines did not bend at nearly a 90-degree angle.

4.4.6 Comparison between best-performing frameworks of Design Space 02 and Design Space 03

When comparing similar frameworks between design space 02 and design space 03, we can observe consistent performance across most frameworks, with one exception being the simulation-based adding weights method (see Figure 4-218). As mentioned earlier, the increased complexity of design options in design space 03 resulted in higher model error. The complexity of the interrelationship between design parameters and performance outcomes also affects the accuracy of predictions made by surrogate models. While we have not specifically tested a surrogate model for design space 03, it can be assumed that its mean absolute error (MAE) would be higher compared to our proposed frameworks, as is typically observed.

Figure 4-217 Pareto Distribution for all tested frameworks for Design Space 03

Figure 4-218 Comparison between best-performing frameworks of Design Space 02 and Design Space 03

4.5 Design Dashboard:

Instead of utilizing the widely used Grasshopper plugin Human UI [28], the team opted to use Plotly [41], a Python library, to generate the dashboard. Plotly has undergone significant development in recent years, and one of its major advantages is its integration with Dash [75], a package that enables user interaction. To visualize our 3D model created with Compas [76] mesh, we needed to convert it into Plotly's mesh definition. The conversion function was developed by Tomás Méndez Echenagucia.

The dashboard features a 3D perspective of the design option in the top right corner, based on that the designer have to hit 'like' or 'dislike'. In the top left corner, buttons for "like," "dislike," and "slide" are placed. Below these buttons, a line chart illustrates the change in performance outcome (kWh per year per square meter) with each iteration. Two pixelated images are also displayed, one showing the currently viewed design option and the other displaying a selected data point from a 3D scatter plot upon clicking.

Below the 3D perspective, eight scatter plots are arranged in two rows and four columns. Although these plots may not be crucial for the designer, they aid the team in visualizing changes after each interaction. The first row of plots displays individual data points, while the second row represents each cluster as a single data point (see Figure 4-219).

1. **First row, first column chart:** This chart colors data points based on their cluster number, helping to confirm if closely aligned data points are grouped together.
2. **First row, second column chart:** This chart visualizes the updates in q-values after each iteration. It only displays the changes in q-values, without including the initially assigned values. Each data point within a cluster has the same q-value.
3. **First row, third column chart:** This chart showcases the performance outcome of individual data points (annual energy consumption per square meter) without any remapping.
4. **First row, fourth column chart:** This chart highlights the data point currently displayed at the top, while keeping all other data points white. By clicking any data point on this chart, the pixelated image of the selected design option appears on top. This chart aids in exploring the distribution of different design options in the lower dimensional space.

5. **Second row, first column chart:** This chart displays the standard deviation of the average performance outcome across all clusters. It indicates which portion of the data has a significantly deviated performance outcome from the average.
6. **Second row, second column chart:** This chart illustrates the pre-assigned q-values for each cluster. However, as mentioned earlier, there is a difference between the assigned q-values and the average performance outcome of each cluster. The average performance outcomes are remapped inversely to obtain the q-values, as lower energy consumption is considered better in terms of performance.
7. **Second row, third column chart:** shows the average annual energy consumption per square meter for each cluster. It also includes a line that depicts the movement in the design space during interaction. If the line consistently tends to move towards colder-colored clusters, it indicates that the reinforcement learning algorithm is functioning properly.
8. **Second row, fourth column chart:** colors the cluster from which the currently displayed design option originates. The currently viewed cluster is colored black, while previously visited clusters are colored grey. Nearby clusters to the currently displayed one are also colored grey to provide insights into potential next moves. This chart helps track the exploration of the design space and aids in understanding the algorithm's functioning.

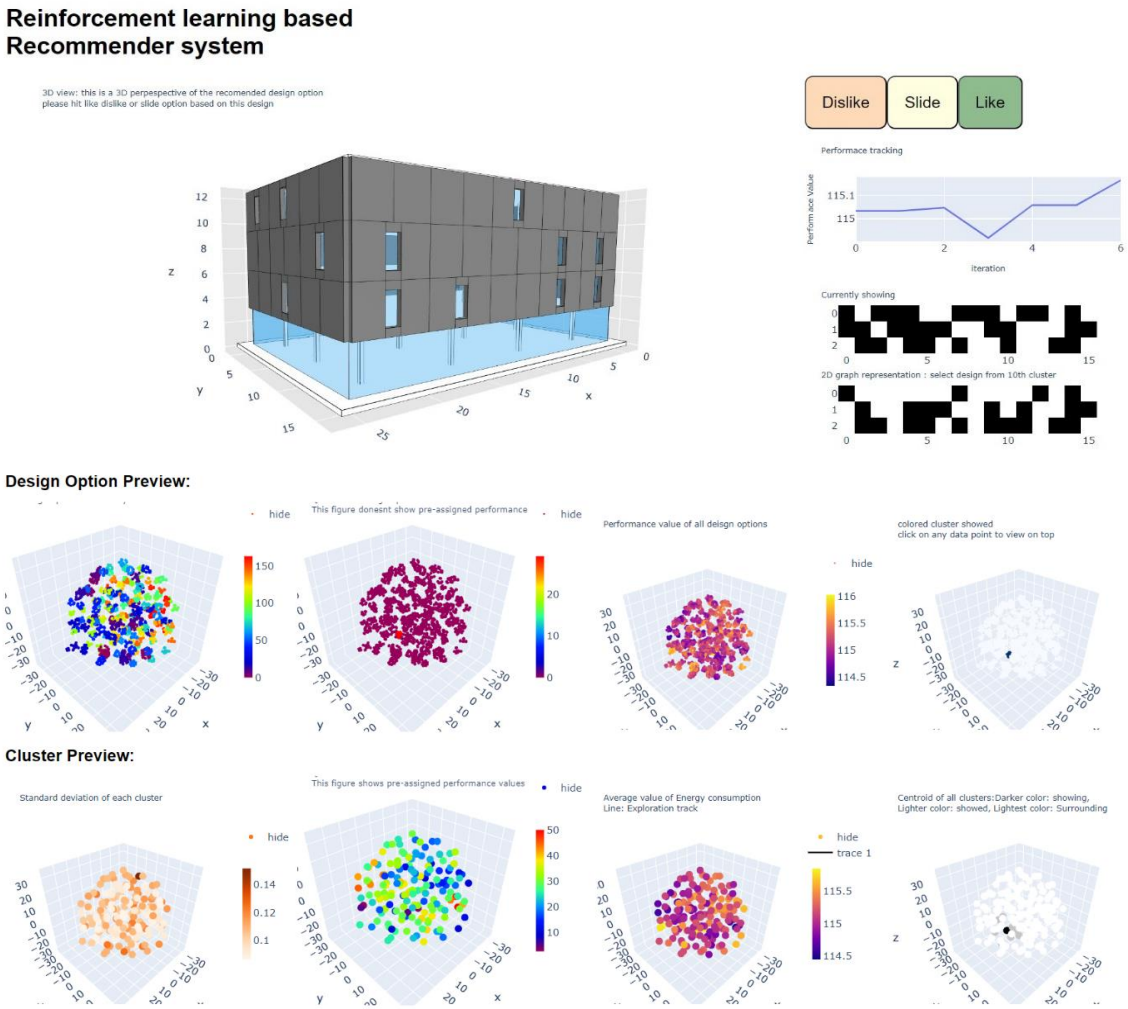


Figure 4-219 Design Dashboard

The dashboard was meticulously developed and made accessible online through Google Cloud service, allowing users from all corners of the world to engage with it. While the dashboard offers valuable functionality, it acknowledges that there is still room for improvement in terms of usability and comprehensiveness. The team prioritized the successful execution of incorporating reinforcement learning-based recommender systems into the exploration of design spaces. The focus was on implementing the core idea and demonstrating its potential, with the intention of refining and fine-tuning the dashboard in the future based on a deeper understanding of user-friendliness and user feedback. Continuous efforts will be made to enhance the dashboard's features and optimize its usability to better serve designers and researchers in their design exploration endeavors.

Chapter 5: Discussions:

The study aimed to address the challenges of exploring huge design spaces by utilizing a reinforcement learning-based recommender system in design space exploration. Given the complex and varied outcomes observed across different frameworks and design spaces, it is challenging to declare a single framework as the most effective. However, the vector-based adding weights method, along with TSNE/Isomap embedding and meanshift clustering, consistently demonstrated strong performance across Design Space 01, 02, and 03. Moreover, when dealing with design options that include more than four faces, using separate vectors for each face resulted in more accurate predictions of performance outcomes.

One of the significant contributions of this research is the demonstration that simulating only 3% of the entire dataset can yield highly accurate approximations of the performance outcomes for the remaining design options. This breakthrough significantly reduces simulation time from months to weeks or even days, removing the previous constraints associated with exploring large design spaces. The potential applications of this approach extend beyond the current study, as it provides valuable insights into the overall understanding of design spaces within a significantly shorter timeframe. This efficiency translates into savings in time, energy, and effort, making the proposed framework a valuable tool in various design exploration scenarios.

5.1 Limitations

While the proposed framework utilizing machine learning and reinforcement learning-based recommender system offers numerous benefits, there are several limitations that need to be addressed in future research.

- Firstly, the current investigations focused solely on predicting the EnergyPlus outcome, which is the annual energy consumption per square meter for heating, cooling, and lighting. It remains unclear how well the proposed framework would perform in predicting other types of simulation outcomes. Discretizing the information into numbers may result in minor or major changes in the outcomes, and the framework's applicability to other simulation outcomes should be explored.
- Secondly, the study executed the idea using a "mid-rise apartment" building as a case study. However, randomly varying window sizes in different orientations may not be applicable to all types of apartment buildings. The investigation of how the proposed process works for different building types and climatic conditions is crucial.
- Thirdly, the mental condition of designers can influence the learning of the reinforcement learning algorithm during the interaction process. If the AI is learning design preferences from multiple individuals, addressing their personal biases and transient mental states becomes a challenge that should be explored.
- In this framework, clustering assumes that design options with different performance outcomes will also exhibit visual differences. To enhance the efficiency of the proposed framework, it is

beneficial to cluster design options based on both visual and performance outcome differences. If designers feel the need for more diversity during exploration, it is recommended to interact with the dashboard using a larger number of clusters.

5.2 Future works

- Furthermore, the proposed frameworks should be further investigated in more complex design spaces that include additional building characteristics such as louvers, podiums, and structural elements. This expansion would provide valuable insights into a broader range of design considerations.
- The study primarily focused on dealing with large design spaces, but how to effectively generate large design spaces with fewer attempts remains unexplored. Developing a thoughtful and comprehensive approach to creating large design spaces with the aid of machine learning tools warrants further exploration.
- Determining appropriate values for the constants used in the reinforcement learning-based recommender system could be investigated through user feedback and experiences with the dashboard. Constructive feedback from users can be utilized to enhance the user-friendliness and comprehensiveness of the dashboard, making it more suitable for professionals from diverse backgrounds.

Chapter 6: Conclusions:

This study presented a pioneering approach to exploring vast design spaces by integrating human intelligence and machine intelligence to generate design solutions that are both high-performing and visually appealing. While previous research predominantly relied on guided optimizations, this study introduced a reinforcement learning-based recommender system that learns from designer interactions to suggest design options that align with their preferences while minimizing annual energy consumption. To enable effective recommendations, the algorithm also needed to learn the performance outcomes of various design options. Recognizing the time-intensive nature of simulating all possible design options, the study explored multiple frameworks that approximated performance outcomes by learning from a subset of the dataset. The team demonstrated that by simulating only 3% of the dataset, accurate predictions of the remaining design options' performance could be achieved.

The study focused on annual energy consumption as a critical metric due to its complex relationship with different design parameters. By harnessing sophisticated machine learning algorithms, the study showcased how large design spaces can be efficiently and effectively explored. This approach capitalizes on the capabilities of design computation and artificial intelligence to address an important problem in the field. The process demonstrates fast and accurate predictions, indicating its sophistication and intelligence. With a shared commitment to creating a sustainable future and reducing carbon emissions, this process contributes by significantly reducing computation time. Additionally, the interactive design dashboard promotes widespread participation in design decision-making.

In conclusion, this study comprehensively addressed various challenges in efficient design space exploration and proposed innovative ideas. Through meticulous research and detailed investigations, these ideas were successfully executed and validated. By tackling multiple aspects of the problem, the study has made significant contributions to advancing the field of design space exploration, paving the way for future advancements in this area.

List of Tables

List of equations

List of Figures

Bibliography

[1] T. Dogan and Y. Park, "A critical review of daylighting metrics for residential architecture and a new metric for cold and temperate climates," *Lighting Research & Technology*, vol. 51, no. 2, pp. 206–230, Apr. 2019, doi: 10.1177/1477153518755561.

[2] H. Baumgartl, J. Tomas, R. Buettner, and M. Merkel, "A deep learning-based model for defect detection in laser-powder bed fusion using in-situ thermographic monitoring," *Prog Addit Manuf*, vol. 5, no. 3, pp. 277–285, Sep. 2020, doi: 10.1007/s40964-019-00108-3.

[3] M. S. Roudsari and M. Pak, "LADYBUG: A parametric environmental plugin for grasshopper to help designers create an environmentally-conscious design," Dec. 2013, Accessed: Jun. 19, 2023. [Online]. Available: https://www.aivc.org/resource/ladybug-parametric-environmental-plugin-grasshopper-help-designers-create-environmentally

[4] N. Fumo, P. Mago, and R. Luck, "Methodology to estimate building energy consumption using EnergyPlus Benchmark Models," *Energy and Buildings*, vol. 42, no. 12, pp. 2331–2337, Dec. 2010, doi: 10.1016/j.enbuild.2010.07.027.

[5] Y. Song, S.-K. Lau, S. S. Y. Lau, and D. Song, "A Comparative Study on Architectural Design-Related Requirements of Green Building Rating Systems for New Buildings," *Buildings*, vol. 13, no. 1, Art. no. 1, Jan. 2023, doi: 10.3390/buildings13010124.

[6] N. Gu, R. Yu, and P. A. Behbahani, "Parametric Design: Theoretical Development and Algorithmic Foundation for Design Generation in Architecture," in *Handbook of the Mathematics of the Arts and Sciences*, B. Sriraman, Ed., Cham: Springer International Publishing, 2021, pp. 1361–1383. doi: 10.1007/978-3-319-57072-3_8.

[7] F. Ritter, P. Geyer, and A. Borrmann, "THE DESIGN SPACE EXPLORATION ASSISTANCE METHOD: CONSTRAINTS AND OBJECTIVES".

[8] M. Rahmani Asl, M. Bergin, A. Menter, and W. Yan, *BIM-based Parametric Building Energy Performance Multi-Objective Optimization*, vol. 2. 2014. doi: 10.52842/conf.ecaade.2014.2.455.

[9] T. Echenagucia, "Computational Search in Architectural Design," 2014. doi: 10.6092/polito/porto/2543137.

[10] K. Saldana Ochoa, P. O. Ohlbrock, P. D'Acunto, and V. Moosavi, "Beyond typologies, beyond optimization: Exploring novel structural forms at the interface of human and machine intelligence," *International Journal of Architectural Computing*, vol. 19, no. 3, pp. 466–490, Sep. 2021, doi: 10.1177/1478077120943062.

[11] C. Mueller and J. Ochsendorf, "Combining structural performance and designer preferences in evolutionary design space exploration," *Automation in Construction*, vol. 52, Apr. 2015, doi: 10.1016/j.autcon.2015.02.011.

[12] T. Wortmann, A. Costa, G. Nannicini, and T. Schroepfer, "Advantages of surrogate models for architectural design optimization," *AI EDAM*, vol. 29, no. 4, pp. 471–481, Nov. 2015, doi: 10.1017/S0890060415000451.

[13] I. As and P. Basu, *The Routledge Companion to Artificial Intelligence in Architecture*. Routledge, 2021.

[14] "EnergyPlus." https://energyplus.net/ (accessed Jun. 19, 2023).

[15] N. C. Brown and C. T. Mueller, "Gradient-based guidance for controlling performance in early design exploration," 2018.

[16] N. C. Brown and C. T. Mueller, "Design variable analysis and generation for performance-based parametric modeling in architecture," *International Journal of Architectural Computing*, vol. 17, no. 1, pp. 36–52, Mar. 2019, doi: 10.1177/1478077118799491.

[17] D. R. Hardoon, S. Szedmak, and J. Shawe-Taylor, "Canonical Correlation Analysis: An Overview with Application to Learning Methods," *Neural Computation*, vol. 16, no. 12, pp. 2639–2664, Dec. 2004, doi: 10.1162/0899766042321814.

[18] N. C. Brown, V. Jusiega, and C. Mueller, "Implementing data-driven parametric building design with a flexible toolbox approach".

[19] "(1) (PDF) Combinatorial equilibrium modeling." https://www.researchgate.net/publication/305887040_Combinatorial_equilibrium_modeling?enrichId=rgreq-9861a58ea2333c6f85e3505195732ae9-XXX&enrichSource=Y292ZXJQYWdlOzMwNTg4NzA0MDtBUzo1NjQ1MjMwNTQ0OTM2OTZAMTUxMTYwNDE5MjAyOA%3D%3D&el=1_x_3&_esc=publicationCoverPdf (accessed Mar. 27, 2023).
[20] L. Fuhrimann, V. Moosavi, P. O. Ohlbrock, and P. D'Acunto, *Data-Driven Design: Exploring new Structural Forms using Machine Learning and Graphic Statics*. 2018.
[21] F. Bertagna, P. D'Acunto, P. O. Ohlbrock, and V. Moosavi, "Holistic Design Explorations of Building Envelopes Supported by Machine Learning," *Journal of Facade Design and Engineering*, vol. 9, pp. 31–46, Apr. 2021, doi: 10.7480/jfde.2021.1.5423.
[22] C. L. Simons, I. C. Parmee, and R. Gwynllyw, "Interactive, Evolutionary Search in Upstream Object-Oriented Class Design," *IEEE Transactions on Software Engineering*, vol. 36, no. 6, pp. 798–816, Nov. 2010, doi: 10.1109/TSE.2010.34.
[23] C. Reeh, "Generative Suburban Frameworks: Emerging Architect-Guided Optimization Workflows Within Suburban Mass Production," *Theses from the M. Arch. Program*, May 2019, [Online]. Available: https://digitalcommons.unl.edu/marchthesis/2
[24] P. Mohapatra, A. Nayak, S. K. Kumar, and M. K. Tiwari, "Multi-objective process planning and scheduling using controlled elitist non-dominated sorting genetic algorithm," *International Journal of Production Research*, vol. 53, no. 6, pp. 1712–1735, Mar. 2015, doi: 10.1080/00207543.2014.957872.
[25] J. Harding and C. Brandt-Olsen, "Biomorpher: Interactive evolution for parametric design," *International Journal of Architectural Computing*, vol. 16, pp. 144–163, Jun. 2018, doi: 10.1177/1478077118778579.
[26] H. Takagi, "Interactive evolutionary computation: fusion of the capabilities of EC optimization and human evaluation," *Proceedings of the IEEE*, vol. 89, no. 9, pp. 1275–1296, Sep. 2001, doi: 10.1109/5.949485.
[27] Y. K. Yi, "Building facade multi-objective optimization for daylight and aesthetical perception," *Building and Environment*, vol. 156, pp. 178–190, Jun. 2019, doi: 10.1016/j.buildenv.2019.04.002.
[28] "Human UI," *Food4Rhino*, Jan. 31, 2016. https://www.food4rhino.com/en/app/human-ui (accessed Jun. 22, 2023).
[29] J. Jongenotter, "Parametric comparison of stability systems: Development of a parametric tool for the comparison and optimisation of four concrete stability systems for high-rise buildings between 100 m and 250 m in an early design phase," 2021, Accessed: Jun. 21, 2023. [Online]. Available: https://repository.tudelft.nl/islandora/object/uuid%3Ab55c53f9-df45-4629-afd6-073bae268f2b
[30] "Hypar." https://hypar.io/ (accessed Jun. 22, 2023).
[31] F. Ricci, L. Rokach, and B. Shapira, "Recommender Systems: Techniques, Applications, and Challenges," in *Recommender Systems Handbook*, F. Ricci, L. Rokach, and B. Shapira, Eds., New York, NY: Springer US, 2022, pp. 1–35. doi: 10.1007/978-1-0716-2197-4_1.
[32] S. Choi, H. Ha, U. Hwang, C. Kim, J.-W. Ha, and S. Yoon, "Reinforcement Learning based Recommender System using Biclustering Technique." arXiv, Jan. 16, 2018. doi: 10.48550/arXiv.1801.05532.
[33] M. Techlabs, "What are the Types of Recommendation Systems?," *MLearning.ai*, Aug. 18, 2021. https://medium.com/mlearning-ai/what-are-the-types-of-recommendation-systems-3487cbafa7c9 (accessed Jun. 22, 2023).
[34] B. Lika, K. Kolomvatsos, and S. Hadjiefthymiades, "Facing the cold start problem in recommender systems," *Expert Systems with Applications*, vol. 41, no. 4, Part 2, pp. 2065–2073, Mar. 2014, doi: 10.1016/j.eswa.2013.09.005.

[35] "Commercial Reference Buildings," *Energy.gov*. https://www.energy.gov/eere/buildings/commercial-reference-buildings (accessed Jun. 19, 2023).
[36] "pandas.DataFrame — pandas 2.0.2 documentation." https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.html (accessed Jun. 19, 2023).
[37] T. M. Echenagucia, "compas_eplus." Mar. 03, 2023. Accessed: Jun. 19, 2023. [Online]. Available: https://github.com/tmsmendez/compas_eplus
[38] F. Pedregosa *et al.*, "Scikit-learn: Machine Learning in Python," *MACHINE LEARNING IN PYTHON*.
[39] M. Verleysen and D. François, "The Curse of Dimensionality in Data Mining and Time Series Prediction," in *Computational Intelligence and Bioinspired Systems*, J. Cabestany, A. Prieto, and F. Sandoval, Eds., in Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, 2005, pp. 758–770. doi: 10.1007/11494669_93.
[40] T. Tr, "Dimensionality Reduction: A Comparative Review".
[41] E. Dabbas, *Interactive Dashboards and Data Apps with Plotly and Dash: Harness the power of a fully fledged frontend web framework in Python – no JavaScript required*. Packt Publishing Ltd, 2021.
[42] R. Pramoditha, "11 Dimensionality reduction techniques you should know in 2021," *Medium*, Sep. 28, 2021. https://towardsdatascience.com/11-dimensionality-reduction-techniques-you-should-know-in-2021-dcb9500d388b (accessed Jun. 19, 2023).
[43] C. J. Willmott and K. Matsuura, "Advantages of the mean absolute error (MAE) over the root mean square error (RMSE) in assessing average model performance," *Climate Research*, vol. 30, no. 1, pp. 79–82, Dec. 2005, doi: 10.3354/cr030079.
[44] A. Lotov and K. Miettinen, *Visualizing the Pareto Frontier*. 2008, p. 243. doi: 10.1007/978-3-540-88908-3_9.
[45] H. Abdi and L. J. Williams, "Principal component analysis," *WIREs Computational Statistics*, vol. 2, no. 4, pp. 433–459, 2010, doi: 10.1002/wics.101.
[46] "sklearn.decomposition.PCA," *scikit-learn*. https://scikit-learn/stable/modules/generated/sklearn.decomposition.PCA.html (accessed Jun. 20, 2023).
[47] L. van der Maaten and G. Hinton, "Viualizing data using t-SNE," *Journal of Machine Learning Research*, vol. 9, pp. 2579–2605, Nov. 2008.
[48] "sklearn.manifold.TSNE," *scikit-learn*. https://scikit-learn/stable/modules/generated/sklearn.manifold.TSNE.html (accessed Jun. 20, 2023).
[49] J. B. Tenenbaum, V. D. Silva, and J. C. Langford, "A Global Geometric Framework for Nonlinear Dimensionality Reduction," *Science*, vol. 290, no. 5500, pp. 2319–2323, Dec. 2000, doi: 10.1126/science.290.5500.2319.
[50] "sklearn.manifold.Isomap," *scikit-learn*. https://scikit-learn/stable/modules/generated/sklearn.manifold.Isomap.html (accessed Jun. 20, 2023).
[51] I. Borg and P. Groenen, *Modern Multidimensional Scaling: Theory and Applications (Springer Series in Statistics)*. 2005. doi: 10.1007/978-1-4757-2711-1.
[52] "sklearn.manifold.MDS," *scikit-learn*. https://scikit-learn/stable/modules/generated/sklearn.manifold.MDS.html (accessed Jun. 20, 2023).
[53] D. Donoho and C. Grimes, "Hessian eigenmaps: Locally linear embedding techniques for high-dimensional data. Proc. National Academy of Science (PNAS), 100, 5591-5596," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 100, pp. 5591–6, Jun. 2003, doi: 10.1073/pnas.1031596100.
[54] "sklearn.manifold.locally_linear_embedding," *scikit-learn*. https://scikit-learn/stable/modules/generated/sklearn.manifold.locally_linear_embedding.html (accessed Jun. 20, 2023).

[55] “sklearn.manifold.SpectralEmbedding,” *scikit-learn*. https://scikit-learn/stable/modules/generated/sklearn.manifold.SpectralEmbedding.html (accessed Jun. 20, 2023).
[56] “2.3. Clustering,” *scikit-learn*. https://scikit-learn/stable/modules/clustering.html (accessed Jun. 20, 2023).
[57] A. Ram, *A Density based Algorithm for Discovering Density Varied Clusters in Large Spatial Databases*. 2015. doi: 10.13140/RG.2.1.4420.1448.
[58] “sklearn.cluster.FeatureAgglomeration,” *scikit-learn*. https://scikit-learn/stable/modules/generated/sklearn.cluster.FeatureAgglomeration.html (accessed Jun. 20, 2023).
[59] A. Alymani, W. Jabi, and P. Corcoran, “Machine Learning Methods for Clustering Architectural Precedents - Classifying the relationship between building and ground,” presented at the eCAADe 2020: Anthropologic : Architecture and Fabrication in the cognitive age, Berlin, Germany, 2020, pp. 643–652. doi: 10.52842/conf.ecaade.2020.1.643.
[60] K. P. Sinaga and M.-S. Yang, “Unsupervised K-Means Clustering Algorithm,” *IEEE Access*, vol. 8, pp. 80716–80727, 2020, doi: 10.1109/ACCESS.2020.2988796.
[61] “sklearn.cluster.KMeans,” *scikit-learn*. https://scikit-learn/stable/modules/generated/sklearn.cluster.KMeans.html (accessed Jun. 20, 2023).
[62] D. Müllner, “Modern hierarchical, agglomerative clustering algorithms.” arXiv, Sep. 12, 2011. Accessed: Jun. 20, 2023. [Online]. Available: http://arxiv.org/abs/1109.2378
[63] “sklearn.cluster.AgglomerativeClustering,” *scikit-learn*. https://scikit-learn/stable/modules/generated/sklearn.cluster.AgglomerativeClustering.html (accessed Jun. 20, 2023).
[64] D. Comaniciu and P. Meer, “Mean shift: a robust approach toward feature space analysis,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 5, pp. 603–619, May 2002, doi: 10.1109/34.1000236.
[65] S. X. Yu and J. Shi, “Multiclass Spectral Clustering”.
[66] “sklearn.cluster.SpectralClustering,” *scikit-learn*. https://scikit-learn/stable/modules/generated/sklearn.cluster.SpectralClustering.html (accessed Jun. 20, 2023).
[67] B. J. Frey and D. Dueck, “Clustering by Passing Messages Between Data Points,” *Science*, vol. 315, no. 5814, pp. 972–976, Feb. 2007, doi: 10.1126/science.1136800.
[68] “sklearn.cluster.AffinityPropagation,” *scikit-learn*. https://scikit-learn/stable/modules/generated/sklearn.cluster.AffinityPropagation.html (accessed Jun. 20, 2023).
[69] “Self-Organizing Map (SOM).” http://users.ics.aalto.fi/jhollmen/dippa/node9.html (accessed Jun. 20, 2023).
[70] “sklearn-som: A simple planar self organizing map.” Accessed: Jun. 20, 2023. [Online]. Available: https://github.com/rileypsmith/sklearn-som
[71] X. Su, X. Yan, and C.-L. Tsai, “Linear regression,” *WIREs Computational Statistics*, vol. 4, no. 3, pp. 275–294, 2012, doi: 10.1002/wics.1198.
[72] “sklearn.linear_model.LinearRegression,” *scikit-learn*. https://scikit-learn/stable/modules/generated/sklearn.linear_model.LinearRegression.html (accessed Jun. 20, 2023).
[73] L. P. Kaelbling, M. L. Littman, and A. W. Moore, “Reinforcement Learning: A Survey,” *Journal of Artificial Intelligence Research*, vol. 4, pp. 237–285, May 1996, doi: 10.1613/jair.301.
[74] E. N. Barron and H. Ishii, “The Bellman equation for minimizing the maximum cost,” *Nonlinear Analysis: Theory, Methods & Applications*, vol. 13, no. 9, pp. 1067–1090, Sep. 1989, doi: 10.1016/0362-546X(89)90096-5.

[75] “Dash Overview.” https://plotly.com/dash (accessed Jun. 23, 2023).
[76] “COMPAS.” https://compas.dev/ (accessed Jun. 23, 2023).