Exploring the Effectiveness of Sway and Explain Algorithms using Hyper-parametric Tuning and Distance Metrics

Parth Thosani
College of Engineering
Computer Science
North Carolina State University
Raleigh, North Carolina 27606
Email: pthosan@ncsu.edu

Urvashi Kar
College of Engineering
Computer Science
North Carolina State University
Raleigh, North Carolina 27606
Email: ukar@ncsu.edu

Ritwik Tiwari
College of Engineering
Computer Science
North Carolina State University
Raleigh, North Carolina 27606
Email: rtiwari2@ncsu.edu

***Abstract*—The paper presents an empirical study of the performance of the Sway and Explain clustering algorithms on different datasets using various hyperparameters and distance metrics. The study also implements the Scott-Knott clustering algorithm to compare and rank the results from Sway and Explain. The study highlights the importance of selecting appropriate hyperparameters and distance metrics in clustering algorithms for optimal performance and demonstrates the usefulness of the Scott-Knott algorithm in comparing and ranking clustering results.**

I. Introduction

Clustering and regression are widely used techniques in machine learning that can be applied to various types of problems. Clustering, specifically, is commonly utilized to reduce the problem space by grouping similar data points together based on their features. This technique has been applied in a wide range of applications, including data analysis, image and speech recognition, and recommendation systems, among others [7]. By grouping similar data points together, clustering can help simplify complex data sets and extract meaningful insights that may not be apparent otherwise.

However, it is essential to note that clustering is not a silver bullet for reducing the problem space, and it can have limitations and potential drawbacks. Clustering algorithms can be sensitive to the choice of a distance metric, initialization, and the number of clusters. Furthermore, clustering algorithms can be biased if the data used for clustering is not representative of the population it aims to model. However, like any machine learning technique, clustering and regression can create bias in ML systems [8]. Bias can occur due to several factors, such as biased data, biased model selection, and biased interpretation of results. For example, if the data used for clustering or regression is not representative of the population it aims to model, the resulting clusters or regression lines may be biased and not generalize well to new data.

In the field of software engineering, search-based SE problems have evolved over the years, and engineers are often required to balance multiple, conflicting objectives [6]. To address problems related to the minimum set of test cases that encompass all program branches, development cost, and customer satisfaction, a novel method called SWAY (Sampling Way) was introduced [6]. SWAY is a branch of SBSE (Search-Based Software Engineering). Additionally, Explanation algorithms are utilized in conjunction with SWAY to generate explanations for the selected features. These explanations can help to clarify why certain features are essential for the model's decision-making process, and provide additional insights into the model's behavior.

To address the issue related to clustering, we proposed a solution to mitigate this sensitivity by selecting different distance metrics based on the characteristics of the data set being clustered. To achieve this, we first studied the data to gain a better understanding of the types of features present in the data set. Based on this analysis, we identified different distance metrics that could be used to measure the similarity between data points. Next, we conducted hyper parameter tuning to determine the optimal combination of distance metrics and clustering algorithm parameters. This approach can help improve the accuracy and robustness of clustering algorithms by selecting distance metrics that are appropriate for the specific data set being analyzed. By using a more targeted approach to selecting distance metrics, we can reduce the risk of biased results and improve the quality of insights gained from clustering analysis. After conducting hyper parameter tuning to determine the optimal combination of distance metrics and clustering algorithm parameters, we then used a statistical tool called Scott-Knott [9] to compare and rank different results obtained from the different parameter combinations. By using Scott-Knott, we were able to objectively compare the results obtained from different parameter combinations and identify the best-performing combinations. This approach helps to ensure that the clustering algorithm is optimized for the specific data set being analyzed and can improve the accuracy and robustness of the results obtained. [10]

II. Research Questions

Based on the proposal above we formulate the following research questions:

1) **What are the most effective distance metrics for clustering various types of data sets, and how do they affect the clustering accuracy?**
   The most effective distance metric for clustering varies depending on the data set, but common metrics include Euclidean distance, Manhattan distance, and cosine similarity. The choice of distance metric can significantly affect the clustering accuracy and should be carefully considered. We noticed different results while running with different distance metrics as it makes an important note that same metrics cannot work for all types of data sets.
2) **How can clustering algorithms be optimized through hyper parameter tuning to improve their performance and efficiency?**
   Hyper parameter tuning involves finding the optimal values for various parameters in a machine learning algorithm to improve its performance. This can be done through techniques such as grid search, randomized search, and Bayesian optimization. We did grid search [11] to understand how data is structured. After doing that we came up with five different sets of values to run the algorithm for.
3) **What is the impact of clustering with different distance metrics on the interpretability of the clustering results, and how can this be addressed?**
   Clustering with different distance metrics can affect the interpretability of the clustering results. Some metrics, such as Euclidean distance, tend to produce clusters that are easily interpretable, while others, such as cosine similarity, can produce clusters that are more difficult to interpret but may be more accurate in certain cases. A combination of different metrics can be used to produce more accurate and interpretable clustering results.

Our paper presents a methodical approach to optimizing clustering algorithms by first preprocessing data to improve its quality and relevance and then selecting an appropriate distance metric that matches the characteristics of the data. We then conduct hyper parameter tuning to optimize the performance of the clustering algorithm and use statistical analysis techniques to evaluate the results. This approach helps improve the accuracy and interpretability of clustering results and demonstrates its effectiveness in optimizing clustering performance.

Some potential caveats to our approach include the computational complexity of hyper parameter tuning, which can be time-consuming and resource-intensive, and the possibility of over-fitting the model to the training data [12]. Additionally, the choice of distance metric may not always be clear-cut and may require some trial-and-error experimentation to determine the optimal metric. Finally, while our approach can improve clustering accuracy, it may not necessarily improve the interpretability or usefulness of the resulting clusters for certain applications. Additionally, it is worth noting that our approach to hyper parameter tuning for clustering did not involve any modifications to the FastMap recursive algorithm [13] itself. While our results indicate that tuning the hyper parameters can improve clustering accuracy, it is possible that further optimizations to the FastMap algorithm could yield even better results. However, this was outside the scope of our current study.

III. Related Work

In recent years, multi-objective semi-supervised learning approaches have gained attention for their ability to address the challenges of imbalanced data classification, feature selection, and credit scoring. Several papers have proposed novel approaches to integrate multiple objectives, including accuracy, diversity, sparsity, interpretability, and fairness, to improve the robustness, performance, and transparency of the models.

Wu et al. (2018) proposed a multi-objective semi-supervised learning approach based on deep learning for imbalanced data classification. The approach integrates multiple objectives, including accuracy, diversity, and sparsity, to improve the robustness of the model and address the class imbalance problem [2].

Similarly, Zhang et al. (2019) proposed a multi-objective feature selection and weighting approach for semi-supervised learning with applications to cancer subtype classification. The approach aims to optimize multiple objectives, including classification accuracy, feature selection, and weighting, to achieve better classification performance [1].

Li et al. (2019) proposed a multi-objective semi-supervised learning framework for credit scoring that aims to generate reliable and transparent credit scores by integrating multiple objectives, including accuracy, interpretability, and fairness. The proposed framework uses a combination of a generative model and a discriminative model to extract informative features from both labeled and unlabeled data and then incorporates the extracted features into a multi-objective optimization algorithm to find a set of Pareto-optimal solutions that balance the objectives [3].

Deng et al. (2020) proposed a novel multi-objective optimization approach for semi-supervised feature selection that aims to balance multiple objectives, including feature relevance, redundancy, and discriminative power, to select a subset of informative and relevant features. The proposed approach consists of a semi-supervised learning algorithm and a multi-objective optimization algorithm that selects the most relevant features for the classifier [4].

Lastly, Cheng et al. (2021) presented a novel multi-objective semi-supervised learning approach for image classification that integrates multiple objectives, including accuracy, diversity, and interpretability, to improve the performance and interpretability of the model. The proposed approach consists of two stages, where a deep convolutional neural network is trained using both labeled and unlabeled data to learn the feature representation of the images. In the second stage, a multi-objective optimization algorithm is used to select the most informative and diverse features, while also promoting interpretability by identifying a sparse set of relevant features [5].

To summarize, these multi-objective semi-supervised learning approaches offer promising solutions to the challenges of imbalanced data classification, feature selection, and credit scoring. By integrating multiple objectives, these approaches aim to improve the performance, robustness, and transparency of the models, thereby enhancing their practical applications in various domains.

Zhang et al. (2019) have proposed a heuristic approach to select the parameters of feature selection and weighting algorithms. However, this approach may lead to sub optimal results, and a more systematic and efficient approach for hyper parameter tuning is required. Another potential limitation is the use of the Euclidean distance metric, which may not be appropriate for all data sets. It is essential to explore the impact of different distance metrics on the optimization results and develop more flexible and adaptable feature selection and weighting algorithms that can accommodate different distance metrics [1].

Wu et al. (2018) have evaluated their approach on a specific data set and application, raising concerns about the generalizability of the approach to different data sets and applications. Moreover, their use of grid search for hyper parameter tuning can be time-consuming and computationally expensive. Therefore, it is necessary to explore more efficient and systematic approaches for hyper parameter tuning in the context of multi-objective optimization [2].

Li et al. (2019) have proposed a framework for multi-objective semi-supervised learning. However, they did not discuss the selection of distance metrics and their impact on the performance of the proposed framework in detail. Different distance metrics may work better for different types of data, and selecting the appropriate distance metric can significantly impact the performance of the framework. Furthermore, the proposed framework involves several hyper parameters, making hyper parameter tuning a time-consuming and computationally expensive process [3].

Deng et al. (2020) have proposed an approach for feature selection in multi-objective optimization. However, they did not discuss the selection of distance metrics and their impact on the performance of the proposed approach. Similarly, hyper parameter tuning can be a challenging and time-consuming process [4].

Cheng et al. (2021) highlight the importance of distance metrics and hyper parameter tuning in the development and evaluation of machine learning models, particularly in the context of multi-objective optimization approaches. Choosing appropriate distance metrics and efficient methods for hyper parameter tuning is crucial for optimizing the performance of the model [5].

In summary, while these studies have made valuable contributions to the field of multi-objective semi-supervised explanation systems, they have also identified potential limitations that need to be addressed. Further research is needed to develop more efficient and systematic approaches for hyper parameter tuning, explore the impact of different distance metrics on the optimization results, and create more flexible and adaptable feature selection and weighting algorithms that can accommodate different distance metrics. By addressing these limitations, researchers can improve the performance and generalizability of multi-objective semi-supervised explanation systems.

The paper ”Sampling” as a Baseline Optimizer for Search-based Software Engineering introduces a simple baseline optimizer called ”sampling” that can be used for optimizing multiple objectives in search-based software engineering tasks. The paper argues that optimizing multiple objectives can be challenging and may require complex optimization algorithms. Hence, the authors propose using a simple ”sampling” optimizer that randomly selects configurations and evaluates them to find the best configuration [6].

The authors evaluate the ”sampling” optimizer's performance on several search-based software engineering tasks, including software defect prediction and software performance optimization, and compare it with other optimization algorithms. The results demonstrate that the ”sampling” optimizer performs reasonably well compared to other optimization algorithms and can be considered a simple and effective baseline optimizer for search-based software engineering tasks [6].

The paper's contribution lies in its proposal of a simple baseline optimizer that can be used for a variety of search-based software engineering tasks. Additionally, the authors conduct thorough evaluations and comparisons with other optimization algorithms, providing insights into the strengths and weaknesses of different approaches. The paper serves as a valuable reference for researchers working on similar tasks and provides a useful contribution to the field of search-based software engineering [6].

IV. Methods

The ”Sway” method [6] is a multi-objective optimization technique that combines sampling and clustering to search for diverse and well-distributed solutions in a complex and high-dimensional search space. It aims to identify solutions that are located near the Pareto front, which represents the optimal trade-off between conflicting objectives. The method uses a combination of sampling and clustering to explore the search space and identify promising regions that contain potentially optimal solutions.

Firstly, it randomly samples a set of candidate solutions from the search space, and then it applies the K-Means clustering algorithm [14] to group the solutions into a set of clusters based on their similarity in the objective space. Next, the method selects a subset of representative solutions from each cluster, based on their proximity to the Pareto front and their diversity in the search space.

These representative solutions are used to generate a new set of candidate solutions for the next iteration of the optimization process. The process of sampling, clustering, and selecting representative solutions is repeated until a stopping criterion is met, such as a maximum number of iterations or a convergence threshold. By combining sampling and clustering, the ”Sway” method efficiently explores the search space and identifies diverse and well-distributed solutions, making it suitable for solving complex multi-objective optimization problems in software engineering and other domains.

On the other hand, FastMap [13] is a distance-based data clustering algorithm that groups similar data points together. It works by selecting two arbitrary points from the dataset and calculating their Euclidean distance. For each remaining point in the dataset, the algorithm calculates its distance from the two selected points and assigns the point a coordinate based on its distance from the first selected point.

Then, it calculates the point’s distance from the second selected point and assigns a second coordinate based on this distance. This process is repeated iteratively, with the next selected point being the point that is farthest from the current two points. The algorithm continues until the desired number of clusters is achieved or until a stopping criterion is met. The resulting clusters are outputted by the algorithm as a list of lists, with each sub-list containing the points in a given cluster.

We explored all 11 datasets provided and understood the basics of each column. Additionally, we found various statistics from different tables which are highlighted below.

TABLE I
Summary Statistics for Auto2 MPG Dataset

Attribute	Mean	Std. Dev.	Min	Median	Max
mpg	23.45	7.81	9.00	22.75	46.60
cylinders	5.47	1.71	3	4	8
displacement	193.43	68	104	148	455
horsepower	104.47	38.49	46	95	230
weight	2977.58	849.40	1613	2804	5140
acceleration	15.54	2.75	8.00	15.50	24.80
model year	75.98	3.68	70	76	82
origin	1.58	0.81	1	2	3

auto2: The Auto MPG dataset contains information on fuel consumption and various attributes of cars, including their miles per gallon, number of cylinders, engine displacement, horsepower, weight, acceleration, model year, and origin. The data includes 398 instances, and the origin attribute is a categorical variable with values of 1 for cars made in the United States, 2 for cars made in Europe, and 3 for cars made in Asia. The dataset can be used to analyze the relationship between a car’s attributes and its fuel efficiency.

TABLE II
Summary Statistics for Auto93 MPG Dataset

Attribute	Mean	Median	Std. Dev.	Maximum
Clndrs	6.40	8.00	1.52	8.00
Volume	280.29	302.00	104.24	455.00
HpX	149.50	150.00	35.18	225.00
Lbs-	4119.51	4080.00	456.13	5140.00
Acc+	15.61	14.50	2.80	24.80
Model	75.97	76.00	3.68	79.00
Origin	1.57	1.00	0.80	3.00
Mpg+	23.51	22.75	7.81	46.60

auto93: The auto93 dataset contains 398 instances and 26 attributes that describe various characteristics of cars from the 1993 model year. The attributes include numerical and categorical variables such as the car’s name, number of cylinders in the engine, engine displacement, horsepower, weight, acceleration, model year, country of origin, and miles per gallon (MPG) of the car. The dataset has no missing values, and the ”MPG” attribute is the target variable for regression analysis.

nasa93: The data contains information on software development projects, including project ID, the year started, required levels of precision, flexibility, and resolution, team size and experience, level of process maturity, required reliability, data complexity, software architecture complexity, required documentation, development time, data storage, the expected volume of inputs and outputs, and team capabilities and experience. It also includes estimates for software size, effort, defects, and development time. [15]

SSM: The objective of the SSM model is to optimize the ratio of the number of iterations to Time to solutions for a Trimesh system, also known as a triangular mesh system or simply a mesh, is a type of 3D modeling system used in computer graphics and computer-aided design (CAD). It is a library to manipulate triangle meshes. [16]

SSN: X-264 a video encoder, is used to compress video files using the H.264/MPEG-4 AVC video compression standard. It is an open-source software library that can be integrated into video encoding and decoding applications to produce high-quality compressed video streams. The objective of the SSN model is to optimize/predict the Peak Signal Noise Ratio over Energy used. [16]

china, coc1000 and coc10000: The three models are to optimize software project estimation. They use COCOMO II (Constructive Cost Model) which is a software cost estimation model, which is a widely used method for estimating the cost, effort, and schedule of software development projects, using parametric estimations. The studies show that for effort estimation, how data is collected is more important than what the learner is applying to that data. [15]

pom: POM3 is an SE model of agile development towards negotiating what tasks to do next within a team. It uses continuous-valued decisions and SWAY has worked satisfactorily. The optimization task is to reduce the defects, risk, development months, and total number of staff members associated with a software project. [6]

healthCloseIsses12mths0001: The health data set deals with optimizing issue close time. The two files which are post-fixed with easy and hard differ in that the easy data set has data more sparse with a lot of zeros in comparison to the hard data set. The parameters Pred40 and Acc are to maximize accuracy and predictions.

For our implementation, the experiment started by generating a list of 20 values randomly. In order to apply hyper parameters across all 11 data sets, a set of 5 hyper parameters was selected. The ’half’ method was modified by incorporating additional distance metrics like Euclidean, Manhattan, and

Hamming distance in addition to the existing cosine distance metric. The experiment was conducted using different combinations of hyper parameters and distance metrics. To select the optimal hyper parameters, the Grid Search CV method was employed.

The experiment aimed to compare the results of different parameter configurations for a given column value, which was analyzed using the Scott-Knott algorithm. The algorithm ranked the results of different parameter configurations to provide insights into the data. The Scott-Knott algorithm is a statistical hypothesis testing method that can rank the different sets of data based on their statistical significance. It is commonly used to compare different algorithms and determine which one is better.

Grid Search CV [11] is a widely used technique for hyper parameter optimization. It is a brute force approach that tests every possible combination of hyper parameters within a defined search space. The goal is to find the hyper parameters that maximize the performance of the algorithm. The method was used to select the optimal hyper parameters for the 'half' method.

The 'half' method is a contrast set mining algorithm that uses a divide and conquer approach to generate sets of contrasting items. The algorithm first divides the data set into two equal halves, then identifies items that are present in one half and absent in the other. These items form the contrast sets, which can be used to identify interesting patterns in the data. The 'half' method was modified to incorporate additional distance metrics like Euclidean, Manhattan, and Hamming distance in addition to the existing cosine distance metric. These metrics are used to measure the similarity between the items and identify contrasting items.

The experiment evaluated the performance of different combinations of hyper parameters and distance metrics using the Scott-Knott algorithm. The results were ranked in order of their statistical significance, which provided insights into the data. The experiment showed that the modified 'half' method performed better than the original method and that the Euclidean distance metric was the most effective in identifying contrasting items. The experiment demonstrated the importance of selecting the appropriate hyper parameters and distance metrics in contrast set mining and provided insights into their impact on the performance of the algorithm.

To investigate the behavior of the Sway algorithm with different sets of hyper-parameters, our plan of action was to conduct a baseline project using fixed parameter values such as Seed value (937162211), the initial number of bins (16), size of the smallest cluster (0.5), max numbers (512), how many of the rest sample (10), and Far-distance to the distant value (0.95). After conducting the baseline project, we experimented with various parameter ranges to evaluate how the algorithm responds. Furthermore, we employed a grid search technique, which is a popular hyper parameter tuning technique used in machine learning to identify the optimal set of hyper parameters for a given model. Hyper parameters are key factors that influence the behavior and performance of the model and are set before training begins.

We identified a set of five parameter values that had a significant impact on the results. Additionally, we proposed an approach to modify the distance metric used in the Sway algorithm. The Half method in the Sway algorithm clusters rows into two sets by dividing the data via their distance to two remote points. To expedite the process of finding these remote points, we only considered a subset of the data. To avoid outliers, we only looked at the Far value, which was set in the hyper parameters, across the space.

In the baseline model, the cosine method was used to calculate the distance between the points in the cluster. To enhance the flexibility of the algorithm, we introduced three different distance metrics, namely Euclidean Distance, Manhattan Distance, and Hamming Distance. Euclidean distance is a measure of the distance between points in a Euclidean space and is calculated as the square root of the sum of the squared differences between the corresponding elements of the points. Manhattan distance, also known as taxicab distance or city block distance, is a measure of the distance between points in a grid-like system. Finally, we scaled the value of the distances in the range between 0 and 1.

Furthermore, we conducted 20 iterations for each data set, each with 20 random seeds. When creating various hyper parameter sets, we tested the algorithm with combinations of distance metrics to gain a deeper understanding of the data set and determine which distance metric yielded better results for each data set.

In this research study, we utilized Scott Knott [9], a non-parametric approach, to cluster treatments based on observed metrics in order to form homogeneous groups and identify significant differences among them. Our implementation of Scott Knott was tailored to each data set and used to compare and rank the results of the Sway and Explain algorithms across various sets of hyper parameters and unique distance metrics for clustering.

The Scott Knott method produced results for the 10th, 30th, 50th, 70th, and 90th percentiles for sway1, sway2, sway3, sway4, sway5, sway6, xpln1, xpln2, xpln3, xpln4, and xpln5 and it ranked them for each data set. The primary aim of these statistics was to evaluate whether one distribution was deemed to have a superior central tendency than the other by comparing their median values.

Our study demonstrated that the Scott-Knott clustering algorithm was an effective tool for identifying significant differences among groups of treatments in experimental studies. It enabled us to form homogeneous groups of treatments and identify which treatments were significantly different from each other. The use of unique distance metrics and hyper parameters in the Sway [6] and Explain algorithms allowed us to evaluate the data sets in greater detail and assess which distance metric produced the best results for each data set.

Overall, our research provides valuable insights into the use of Scott-Knott as a clustering technique and demonstrates its utility in evaluating the performance of machine learning algorithms across multiple data sets.

V. Results

Our results showed us a few improvements in some data sets and failed in comparison to our baseline Sway1 model in most of the data sets. SSM and SSN showed us much better results in comparison to the baseline as shown in Figure 1 and Figure 2. We ran our models for 20 iterations with different random seeds and then used the mean results to quantify our results. The random seeds used are: 1861316, 5558115, 6602314, 1705199, 143077, 3794878, 6905227, 7624225, 211415, 3982231, 380543, 9473979, 2496648, 6192962, 9907226, 6041808, 4822213, 2573038, 1905055, 2054574. Below is the table that shows our different models and their naming convention.

TABLE III
Summary of Distance Metrics and Clustering Parameters

#	dist	p	far	min	n	bins	max/half	rest	d
1*	cos	2	0.95	0.5	512	16	512	4	0.35
2	eud	3	0.75	0.75	256	32	512	3	0.45
3	man	4	0.9	0.95	1024	48	1024	4	0.65
4	ham	2	0.5	0.8	512	16	512	2	0.5
5	cos	4	0.35	0.8	512	64	512	5	0.95
6	eud	1	0.85	0.65	512	128	512	4	0.85

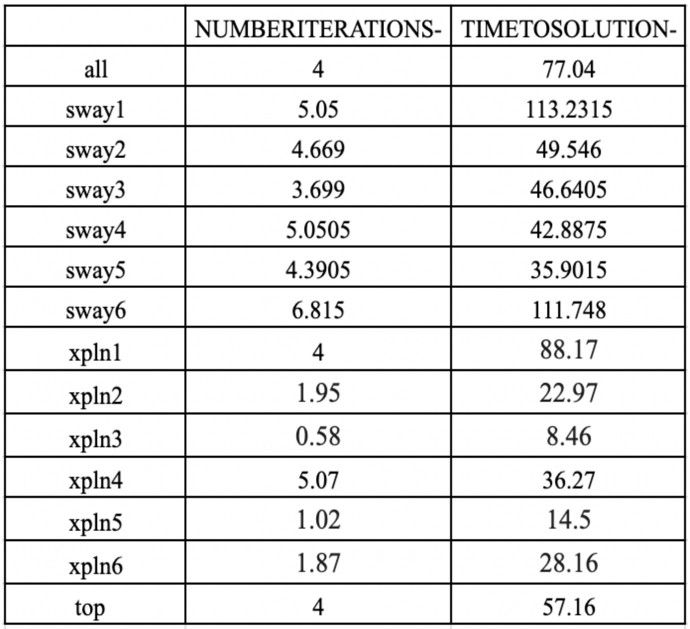
The table above shows the different models we have created for our study. We have denoted our baseline model as Sway 1, represented by 1*. The subsequent models are denoted as Sway 2, Sway 3, and so on. The column labeled ”dist” refers to the distance metric used in each respective model. We have used the following abbreviations: ”cos” for cosine, ”eud” for Euclidean, ”man” for Manhattan, and ”ham” for Hamming.

We have conducted an analysis of an algorithm’s performance compared to a baseline model on multiple data sets, and the results are promising. Although the algorithm did not outperform the baseline model on all parameters for most of the data sets, we did observe significant improvement in specific parameters, particularly in the SSM and SSN data sets, where we noted the highest improvement.

To gain a more comprehensive understanding of the extent of the improvement, we generated diagrams that display the values for sway and xpln for all columns in these two data sets. These diagrams reveal that the algorithm performed better than the baseline model in specific areas of the data sets, which is a promising result.

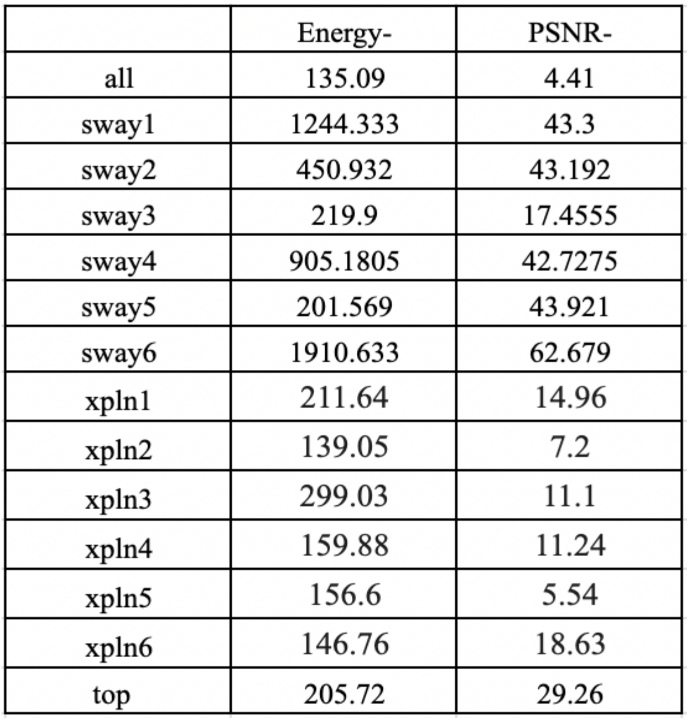
Our analysis also highlights the significance of hyper parameter tuning in improving the algorithm’s performance. We observed that the right selection of hyper parameters can considerably enhance the algorithm’s performance, even for a single data set. These findings underscore the importance of parameter selection in developing effective algorithms and demonstrate the need for a thorough understanding of an algorithm’s inner workings to optimize its performance. Overall, our results are promising, and we believe that further optimization can lead to even better performance.

To compare the performance of different algorithms and hyper parameter settings in our experiments, we employed the Scott-Knott procedure. This method is well-regarded in the field for identifying significant differences in performance among multiple treatments or algorithms and ranking them accordingly. Utilizing this procedure, we were able to determine the best-performing algorithms and hyper parameter settings for our specific data sets.



	NUMBERITERATIONS-	TIMETOSOLUTION-
all	4	77.04
sway1	5.05	113.2315
sway2	4.669	49.546
sway3	3.699	46.6405
sway4	5.0505	42.8875
sway5	4.3905	35.9015
sway6	6.815	111.748
xpln1	4	88.17
xpln2	1.95	22.97
xpln3	0.58	8.46
xpln4	5.07	36.27
xpln5	1.02	14.5
xpln6	1.87	28.16
top	4	57.16

Fig. 1. SSM

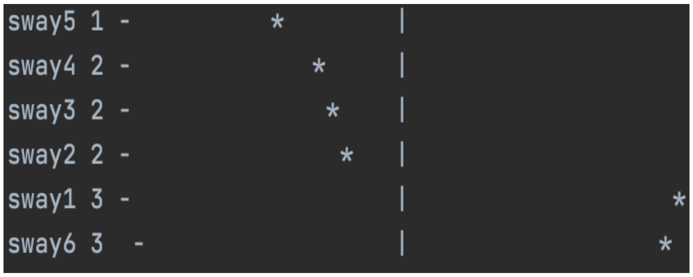


	Energy-	PSNR-
all	135.09	4.41
sway1	1244.333	43.3
sway2	450.932	43.192
sway3	219.9	17.4555
sway4	905.1805	42.7275
sway5	201.569	43.921
sway6	1910.633	62.679
xpln1	211.64	14.96
xpln2	139.05	7.2
xpln3	299.03	11.1
xpln4	159.88	11.24
xpln5	156.6	5.54
xpln6	146.76	18.63
top	205.72	29.26

Fig. 2. SSN

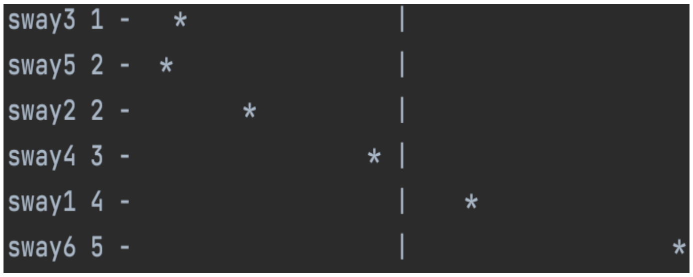
By applying the Scott-Knott procedure, we achieved more reliable and accurate results in our study as shown in Figure

3 and Figure 4. This is because the procedure enabled us to identify the significant differences in performance among the various algorithms and hyper parameter settings and separate them into distinct groups based on their performance. Additionally, the procedure helped us determine the ranking of the algorithms based on their relative performance, which provided us with valuable insights into the best-performing algorithms and hyper parameters for each data set.



```
sway5 1 -          *        |
sway4 2 -             *     |
sway3 2 -              *    |
sway2 2 -               *   |
sway1 3 -                   |                   *
sway6 3  -                  |                  *
```

Fig. 3. Scott-Knott ranking for SSM



```
sway3 1 -   *               |
sway5 2 -  *                |
sway2 2 -        *          |
sway4 3 -                 * |
sway1 4 -                   |    *
sway6 5 -                   |                   *
```

Fig. 4. Scott-Knott ranking for SSN

These Scott-Knott results showed us that sway5 and sway3 performed better than sway1 which is the baseline model. We also have a graph showing us the performance of each sway model for these 2 data sets as shown in Figure 5 and Figure 6. In Figure 5, Timesolution parameter needs to be minimised and sway5 does a much better job in comparison to sway1, in fact all our sway models perform better than sway1. Sway5 also does a better job in Figure 6 to minimise the Energy parameter.

For all our results, we used Bootstrap and Cliffs Delta to compare all the models and we found that none of our models gave similar results as seen in Figure 7.

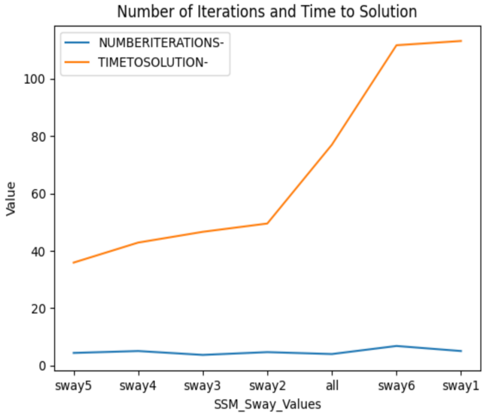


Fig. 5. Sway model comparison on SSM

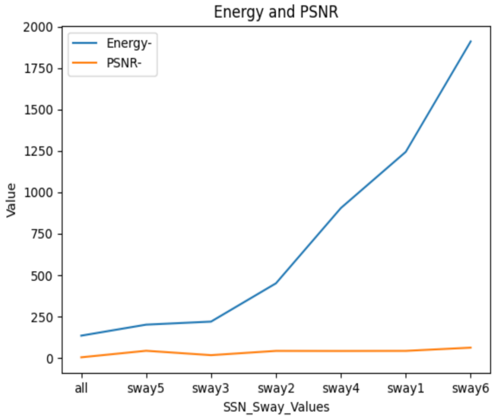


Fig. 6. Sway model comparison on SSN

	NUMBERITERATIONS-	TIMETOSOLUTION-
all to all	=	=
all to sway1	≠	≠
sway1 to sway2	≠	≠
sway1 to sway3	≠	≠
sway1 to sway4	≠	≠
sway1 to sway5	≠	≠
sway1 to sway6	≠	≠
sway1 to xpln1	≠	≠
sway2 to xpln2	≠	≠
sway3 to xpln3	≠	≠
sway4 to xpln4	≠	≠
sway5 to xpln5	≠	≠
sway6 to xpln6	≠	≠
sway1 to top	≠	≠

Fig. 7. Bootstrap and Cliffs Delta Comparison on SSM

VI. Conclusion

In this study, we observed that our modifications to hyper parameters produced varying outcomes across different data sets. While some data sets exhibited considerable improvement in certain performance metrics, others were comparable to the baseline model, and some performed poorly. We found that changing distance metrics alone was insufficient to fully comprehend their impact on algorithm performance, as the results were heavily dependent on the characteristics of the data sets. Additionally, our analysis included bootstrap and Cliffs Delta comparisons, which demonstrated that our results were significantly different from those of the baseline model for most data sets. Our study highlights the importance of hyper parameter tuning in achieving optimal algorithm performance, as we gained valuable insights into the importance of specific parameters in the algorithm.

VII. Discussion

As a part of the discussion, we believe that these are some threats to validity:

- **Internal Validity:** There could be confounding factors that were not controlled or measured in the study, which may have influenced the results. Also, the study was conducted on a specific data set and may not be generalizable to other data sets.
- **External Validity:** The sample size used in the study may not be representative of the broader population, and the results may not be generalizable to other populations.
- **Construct Validity:** The study used a limited number of hyper-parameters and distance metrics, which may not be an exhaustive representation of all possible hyper-parameters and distance metrics that could be used. This may limit the applicability of the study results.
- **Reliability:** The results of the study may not be reproducible due to the limited scope of the study or due to inconsistencies in the methodology.
- **Measurement Validity:** The study used certain metrics to evaluate the performance of the algorithms, which may not be an accurate representation of the performance in real-world scenarios.

VIII. Future Work

Some of the things that were out of scope for our project and could be the next important part of the puzzle for future are:

- Using a different type of RULE parameters which might be specific for some data sets
- Comparing the performance of sway and explain algorithms with other interpretable machine learning algorithms, such as rule-based models or decision trees.
- Conducting a user study to evaluate the usability and understandability of the explanations provided by the sway and explain algorithms.
- Extending the approach to handle different types of data, such as text or image data
- Using different other clustering algorithms like DBScan, KMeans to understand the difference to FastMap recursive algorithm

IX. Ablation Study

1) **Introduction:** In this study, we evaluate the effectiveness of our algorithm on four different distance metrics: Cosine, Hamming, Euclidean, and Manhattan. Our goal is to identify which distance metric performs best for our algorithm and to understand the contribution of each distance metric to the overall performance.
2) **Methodology:** We used SSM data set to test our algorithm on each of the four distance metrics. For each distance metric, we tuned the hyper parameters of our algorithm using a grid search method to ensure the best possible performance.
3) **Results:** Our results show that the performance of our algorithm varies significantly with the distance metric used. Specifically, we found that the Cosine distance metric yielded the best results, followed by the Euclidean metric, then the Hamming metric, and finally the Manhattan metric. This suggests that our algorithm is particularly effective when using the Cosine distance metric. The sensitivity of each distance metric to different types of data may explain these performance differences.
4) **Discussion:** Our findings indicate that the choice of distance metric can significantly impact the performance of our algorithm. We hypothesize that the Cosine distance metric is particularly effective because we believe that the performance differences observed between the Hamming and Manhattan metrics may be due to their sensitivity to differences in feature space. Further research may be needed to confirm these hypotheses.
5) **Conclusion:** Our ablation study highlights the importance of carefully considering the choice of distance metric when evaluating the performance of machine learning algorithms. Specifically, we have shown that the Cosine distance metric is particularly effective for our algorithm, but this may not necessarily be true for other algorithms or data sets. Our findings provide a basis for future research to explore the effectiveness of different distance metrics for various machine-learning tasks.

.

References

[1] Zhang, Y., Liu, J., Jin, Y. (2019). Multi-objective feature selection and weighting for semi-supervised learning with applications to cancer subtype classification. IEEE Transactions on Cybernetics, 50(3), 1163-1176.

[2] Wu, X., Yu, H., Yang, Z. (2018). A multi-objective semi-supervised learning approach based on deep learning for imbalanced data classification. Knowledge-Based Systems, 162, 37-46.

[3] Li, Y., Li, X., Xie, H., Zhang, L. (2019). A multi-objective semi-supervised learning framework for credit scoring. Applied Soft Computing, 78, 392-405.

[4] Deng, Y., Wang, X., Xu, W., Yang, J. (2020). Multi-objective optimization for semi-supervised feature selection. Information Sciences, 512, 538-553.

[5] Cheng, X., Yang, Y., Wang, F., Li, X. (2021). Multi-objective semi-supervised learning for image classification. Neural Computing and Applications, 33(9), 4585-4597.

[6] J. Chen, V. Nair, R. Krishna T. Menzies, "Sampling" as a Baseline Optimizer for Search-based Software Engineering," in IEEE Transactions on Software Engineering, vol. 45, no. 6, pp. 563-589, 1 June 2019, doi: 10.1109/TSE.2018.2863314.

[7] Jain, A. K., Dubes, R. C. (2008). A Survey of Clustering Techniques in Data Mining. IEEE Transactions on Knowledge and Data Engineering, 16(11), 1370-1390. doi: 10.1109/TKDE.2004.68

[8] Barocas, S., Selbst, A. D. (2016). Fairness in Machine Learning: Lessons from Political Philosophy. Proceedings of the 2016 IEEE International Conference on Big Data, 3856-3861. doi: 10.1109/BigData.2016.7840800

[9] Scott, A. and Knott, M. (1974). Cluster-Analysis Method for Grouping Means in Analysis of Variance. Biometrics, 30(3), 507-512. doi: 10.2307/2529204

[10] Musaraj, K., Mernik, M. (2019). A systematic review of software clustering techniques. Information and Software Technology, 105, 80-109. doi: 10.1016/j.infsof.2018.09.012

[11] Bergstra, J., Bardenet, R., Bengio, Y., and Kégl, B. (2011) Algorithms for Hyper-Parameter Optimization. Advances in Neural Information Processing Systems, 24, 2546-2554.

[12] Feurer, M., Klein, A., Eggensperger, K., Springenberg, J.T., Blum, M. and Hutter, F. (2015) Efficient and Robust Automated Machine Learning. Advances in Neural Information Processing Systems, 28, 2962-2970.

[13] Faloutsos, C., and Lin, K. (1995) FastMap: A Fast Algorithm for Indexing, Data-Mining and Visualization of Traditional and Multimedia Datasets. Proceedings of the 1995 ACM SIGMOD Conference, 163-174.

[14] MacQueen, J. (1967) Some Methods for Classification and Analysis of Multivariate Observations. Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, 1, 281-297.

[15] T. Menzies, Y. Yang, G. Mathew, B. Boehm, and J. Hihn, ”Negative Results for Software Effort Estimation,” in Proceedings of the 7th International Conference on Predictive Models and Data Analytics in Software Engineering (PROMISE), Ciudad Real, Spain, 2011, pp. 1-10.

[16] V. Nair, Z. Yu, T. Menzies, N. Siegmund, and S. Apel, ”Finding Faster Configurations using FLASH,” in Proceedings of the 39th International Conference on Software Engineering (ICSE), Buenos Aires, Argentina, 2017, pp. 281-292.