Date of publication 2024 04,22, date of current version 2024 04,22,.

Digital Object Identifier 10.1109/ACCESS.2017.DOI

Comparative Analysis of Clustering and Classification algorithms with Sequential Model Optimization

RATISH KUMAR SARAVANAN[1], SHARDUL KHANDEKAR[1] AND SKANDA SHASTRY [1],
[1]Department of Computer Science,North Carolina State University, Raleigh, NC 27606 USA (e-mail: rsarava@ncsu.edu)
[1]Department of Computer Science,North Carolina State University, Raleigh, NC 27606 USA (e-mail: sskhande@ncsu.edu)
[1]Department of Computer Science,North Carolina State University, Raleigh, NC 27606 USA (e-mail: sshastr4@ncsu.edu)

"This work is done as part of Automated Software Engineering course work in support of Dr. Timothy Manzies"

ABSTRACT In the world of doing configurations for programmatic requirements we have surrogates which take in a whole bunch of configuration in a single attempt and provide the results. Although surrogates can perform this, they tend to induce errors in their prediction.Due to this we need a algorithm that provides better results at any plausible scenarios.Sequential Model Optimization helps to solve this problem and its performance is also good for various datasets but how far is it based on the clustering algorithm being used. Our paper presents a comprehensive comparative analysis between various clustering algorithms. The study evaluates the performance of popular clustering techniques, including K-Means, Support Vector Machine, Gaussian mixture model and Agglomerative clustering with Sequential Model Optimization (SMO) as the baseline. The analysis encompasses diverse datasets and evaluates the algorithms based on a non parametric significant tests namely, Scott Knot test along with an effect size test called Cliff's delta test. The findings provide valuable insights into the strengths and limitations of each algorithm, aiding our research in selecting the most appropriate clustering approach for a certain number of datasets.By the end of our research project we will be able to provide an efficient algorithm strategy over Sequential Model Optimization and compare its efficiency over Recursive Random Projection.

INDEX TERMS Clustering Algorithms, K-Means, Support Vector Machine(SVM), Recursive Random Projections(RRP), Sequential Model Optimization(SMO) and Gausian Mixture Models(GMM), Agglomerative clustering, Interquartile range (IQR)

I. INTRODUCTION

A. THE PROBLEM

The algorithms that we are well familiar and compare with each other are the Sequential Model Optimization(SMO) and the Recursive Random Projections(RRP). Out of these two, we have an existing study that Sequential Model Optimization tend to perform better than Recursive Random Projections. This study cannot be ultimatum unless we consider other clustering algorithms and their influence on the Sequential Model Optimization. Different algorithms may perform better under specific data distributions, making it difficult to determine which one is most suitable for a particular dataset without extensive experimentation. Moreover, the absence of a universally superior algorithm means that the choice often involves a trade-off between factors such as computational efficiency, interpret ability of results and scalability [1].Additionally, the lack of standardized evaluation metrics further complicates the comparison process, as performance measures may vary depending on the specific objectives and characteristics of the data. In this study, we are going to estimate the performance of four different clustering algorithms on a process model data along with (Sequential Model Optimization)SMO.

We find this model to be interesting and important mainly because the surrogate models often acts as oracles were it is poor in predictions. So we usually have it to check how far is one better than the other model.It just ranks based on its preference. However, it produces errors in the range of 10% to 90%. If we perform configuration for every problem one at a time it will cost us time and it will be inefficient. The model of surrogates can be improved by using Sequential Model

Optimization as it randomly selects certain number of rows and classifies them into two clusters and then randomly pick data from the original dataset to classify them into the two clusters. It is immensely important to estimate the efficiency of this algorithm as there are so many other algorithm which are available that could perform better or worse than the baseline Sequential Model Optimization.

It is challenging to conclusively determine whether Sequential Model Optimization (SMO) performs better than Recursive Random Projections (RRP) due to the inherent differences in their functionalities and the lack of direct comparability.Naive approaches fail because they overlook the distinct objectives and applicability domains of these methods, leading to invalid comparisons. Additionally, the effectiveness of each technique depends on various factors such as dataset characteristics, task complexity, and algorithmic assumptions, further complicating direct assessments of performance. Thus, establishing a clear superiority of one over the other requires comprehensive evaluations tailored to specific use cases, considering the nuanced strengths and limitations of each approach.

The challenge of conclusively determining whether Sequential Model Optimization (SMO) performs better than Recursive Random Projections (RRP) hasn't been fully resolved due to several reasons. Previous proposed solutions often suffer from a lack of standardized evaluation methodologies, leading to inconsistent comparisons across studies. Additionally, many existing approaches focus on comparing algorithms in isolation without adequately considering the diversity of datasets and real-world application scenarios. Moreover, the dynamic nature of machine learning research and the continuous development of new algorithms contribute to the complexity of the problem.What sets our approach apart is its comprehensive and systematic evaluation framework, which addresses these limitations. Unlike previous solutions, our methodology incorporates diverse datasets, robust evaluation metrics, and algorithm-agnostic criteria to ensure fair and representative comparisons.

The existing Sequential Model Optimization takes the dataset with splitting the data from datasets into training lets call it lite and testing lets call this as dark and a initial budget which corresponds to the set of records taken from the dataset to classify into two worlds lets say best and rest. Then the algorithm goes ahead and chooses points a random from the dataset and compares the likelihood of that record with the prior learning history and compares the value between the best and rest. Whichever has the high likelihood will likely have the record grouped with it.The algorithm performs the split based on a acquisition function

$$\frac{b+r}{b-r}$$

. Sequential Model Optimization does a really good job in classifying random data points on to best and rest.However, it does also have limitations of not being able to provide a summary of information of the clustered data points where Recursive Random Projections takes a lead.

B. RESEARCH QUESTIONS

1) What are the computational complexities of SMO when subjected to different clustering algorithms?
2) How can we evaluate and compare different algorithms?
3) How does significant tests like Scott Knot tests were able to rank the clustering algorithms?

C. CONTRIBUTIONS

1) Implementation of Kmeans:We devised a K-Means algorithm were we take 2 clusters to split among the value and made the optimal data to get added to the best.Then we tried to get the top best data point from the resulting best cluster.

2) Implementation of Gaussian Mixture Models:In terms of Gausian Mixture Model we to fit the model with the dark set and then calculated the next likelihood from the model and compared with the likelihood throughout the model and whichever datapoint has the best likelihood we used those as the selected ones.

3) Implementation of Support Vector Machines:We implemented a One Class Support Vector Machine model. Since we find that when we cluster data there is majority data present in lite and very little data present in dark. So we planned to use this method to identify the possible best data that could get selected from the set of lite data

4) Performed Statistical significant test using Scott-Knot test: We were able to get the mean or mid value of the top best rows from each algorithm and then we passed on each of these row into the Scott knot test which then were able to provide us a significant difference each of the clusters

5) Performed effect size Cliff's delta test: We used Cliff's delta test to identify the effective difference between the two distribution. We used it to measure the magnitude of differences between two groups

D. STRUCTURE

The rest of this paper is structured as follows.We start with background, algorithms, methods and inside methods we have discussed about data, experimental setup, evaluation metrics and statistical methods. Then we have discussed the results with the relevant data we received out of the algorithms.Furthermore, we discussed what we learned during the course of this project, then we discussed about the threats to validity along with some future work that could be done in

the near future.Lastly, we conclude the research and state the references that helped us for this project.

II. BACKGROUND

A. MOTIVATION

Understanding the performance and suitability of clustering algorithms is paramount in various fields, from data mining and machine learning to healthcare and finance. With an ever-growing array of clustering techniques available, selecting the most appropriate algorithm for a given dataset or application becomes increasingly challenging. Thus, our project aims to address this challenge by conducting a comprehensive comparative analysis of clustering algorithms, including well-established methods such as K-Means and Support Vector Machine, along with Gaussian mixture model, Agglomerative clustering with Sequential Model Optimization (SMO) as the baseline. By evaluating these algorithms across diverse datasets and employing rigorous statistical tests such as the Scott Knot test and Cliff's delta test, we seek to provide practitioners and researchers with valuable insights into algorithm performance, robustness, and suitability for real-world applications. We came across several research findings which we felt motivated to work upon this line of research.First and foremost, A comparative study of clustering algorithm [2]. This paper gave us a complete understanding of various clustering algorithm that were present over a decade from 2000 to 2019 and it covered various clustering techniques and evaluates their performance on benchmark datasets using statistical tests and performance metrics. [3] This paper provided us good insight on the clustering algorithms that we use and their influence with the real world dataset. We were able to find the parameters they were tuning in order to get good results.

Choosing an optimal algorithm not only enables us to compare SMO and RRP but also will be useful in many industries. We analysed few usecases of this before starting the work.The research findings from the comparative analysis of clustering algorithms serve as a pivotal tool across diverse fields. For instance, in healthcare, identifying the most effective algorithm for patient sub-typing enables personalized treatment strategies and enhances healthcare delivery. Similarly, in finance, selecting the optimal clustering technique aids in fraud detection and risk assessment, thus safeguarding financial systems. Ultimately, these research insights empower decision-makers to effectively employ clustering algorithms, fostering innovation and progress in their respective domains.

Gaining insights over the performance of clustering algorithm not only gives us the answer for SMO's performance with RRP but it can also provide valuable information on what all parameters have influence over making an algorithm work efficient over respective dataset.

B. BASELINE

As per our problem SMO does better than RRP based on the proven results but how far does this imply on the basis of the existing clustering algorithms. To make this proof clear we took SMO as baseline and performed other clustering algorithms on SMO's cluster forming implementation.

We were able to get a lot of useful insights and use cases of SMO as per [4] our findings SMO's implementation is best for chosing random data points for a wide variety of data points but it intrigued our interest to find what else might be an ideal option to go if we change the implementation of clustering. SMO uses a acquisition function as a hyper parameter unlike other clustering algorithms.

We are using SMO as baseline for all the four clustering algorithms and comparing their performance with respect to time each algorithm takes to complete 20 iterations.Also, we did calculate the mean and standard deviation of all the algorithms over the 10 datasets with respect to the mean values we have from the selected optimum data point and we tried to use this to measure the difference between from the top most data point which is ideally closest to the distance to heaven(ideal point) value.

We then use the standard deviation as the performance metric to see how close the data is aligned as per each algorithm and dataset.

C. HYPERPARAMETERS

We were concerned with certain hyper-parameters for each of the algorithms we use. This is shown in the below Table 1.This section explains the algorithms used for this project.

Algorithm	**Hyperparameter(s) Used**
Support Vector Machine	Kernel
Gaussian Mixture Model	Number of mixture components
Agglomerative Clustering	Number of clusters
K-Means	Number of clusters, Random state, no of iterations
SMO	Likelihood based on prior estimation (b, r)

TABLE 1: Clustering algorithms and their corresponding hyperparameters

D. SEQUENTIAL MODEL OPTIMIZATION

Sequential Model Optimization is our baseline model which is introduced in our class. This model initially takes few rows of data from the given dataset randomly and based on the budget0 size as per our algorithm and then this splits the data into lite and dark which forms our training and testing data respectively. Then we run a best and rest function to split the world into best and rests based on the length of the dataset raised to a power of half.Until the condition satisfies we sort the rows and add it to the best and once the condition is met the remaining rows are added to the rest dataset.Then,

we pass the datasets into split function to get the next best selected row based on the acquisition function

$$\frac{b+r}{b-r}$$

then, the likelihood of the function is performed based on the prior values as per the below formula.

$$prior = \frac{(len(data)+k)}{(n+k*nHypotheses)}$$

Where,

- k= Hyper parameter for tuning the prior estimation
- n= Total Number of rows in the dataset.
- nHypotheses= Number of hypotheses being considered.

Sequential Model Optimization (SMO) is an iterative method used for optimizing complex, often nonlinear, and computationally expensive objective functions. It belongs to the class of Bayesian optimization algorithms and is particularly useful when the objective function lacks analytical form or has a high computational cost to evaluate.

ALGORITHM OVERVIEW

1) **Initial Sampling**: SMO starts by sampling a small set of points from the input space to evaluate the objective function. These initial points are used to build an initial surrogate model of the objective function.
2) **Surrogate Model Construction**: Based on the initial sampled points, a surrogate model is constructed to approximate the objective function. Common choices for surrogate models include Gaussian Processes (GPs), Random Forests, or Gradient Boosting Machines (GBMs).
3) **Acquisition Function Optimization**: SMO iteratively selects the next point to evaluate by optimizing an acquisition function. The acquisition function provides a trade-off between exploration (sampling in regions where uncertainty is high) and exploitation (sampling in regions likely to yield the best results). Common acquisition functions include Probability of Improvement (PI), Expected Improvement (EI), and Upper Confidence Bound (UCB).
4) **Objective Function Evaluation**: The selected point is evaluated using the true objective function. This step can be computationally expensive, especially if the objective function is complex or requires significant computational resources.
5) **Surrogate Model Update**: After evaluating the objective function at the selected point, the surrogate model is updated using the new data point. This step improves the accuracy of the surrogate model and reduces uncertainty in regions of the input space.
6) **Convergence Check**: SMO continues the iterative process until a stopping criterion is met. This criterion can be a maximum number of iterations, a maximum budget of objective function evaluations, or convergence of the optimization process.

APPLICATIONS

SMO is particularly effective in scenarios where the objective function is expensive to evaluate, such as:

- Hyperparameter tuning in machine learning algorithms.
- Optimization of simulation-based models.
- Tuning parameters of expensive experiments.

By iteratively updating a surrogate model and selecting the next point to evaluate based on an acquisition function, SMO efficiently explores the input space and identifies regions likely to contain the optimal solution. It strikes a balance between exploration and exploitation, making it suitable for optimizing complex objective functions with limited computational resources.

E. AGGLOMERATIVE CLUSTERING

The reason for chosing this specific algorithm is die to these reasons.According to [5] K-Means is one of the simplest and most widely used clustering algorithms due to its ease of implementation and efficiency.

The convergence properties of K-Means are really good in accordance with [6] after going through the results of this paper we got interested with this algorithm. Agglomerative clustering is a hierarchical clustering technique used to group similar data points into clusters. It starts with each data point considered as a single cluster and then iteratively merges the closest clusters until only a single cluster remains. This process forms a tree-like structure called a dendrogram, which captures the hierarchical relationships between clusters.

In our project, This method starts by preparing the input data (lite) for clustering, converting it into a numerical format suitable for analysis. It then applies agglomerative clustering, a hierarchical clustering technique, with a preset number of clusters (two in this case). Each data point is assigned to the nearest cluster based on a linkage criterion until the specified number of clusters is reached. The clustering results are visualized using PCA, reducing the dimensionality of the data to two dimensions for easy visualization.

After clustering, the method evaluates a separate dataset (dark) against the clustering model. It converts each data point in dark to numeric format and predicts its cluster assignment based on the previously determined clusters. If a data point in dark belongs to the same cluster as the majority of points in lite, it is considered to be part of the same group. These selected data points from dark are then returned along with an output flag indicating the success of the operation and a representative data point from the dominant cluster in lite, offering insights into the characteristics of that cluster. This method enables the exploration and classification of new data points based on the clustering patterns observed in the original dataset.

In Agglomerative Clustering, we use **nclusters** as the param-

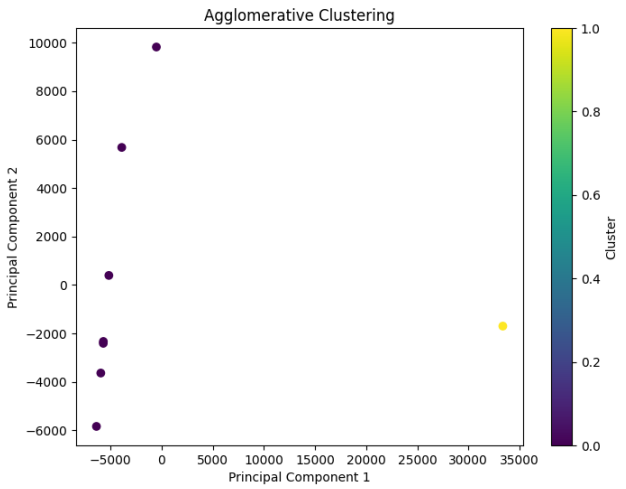

FIGURE 1: clusters formed after PCA with Agglomerative

eter specifies the number of clusters to form. In this code, it's set to 2, indicating that the Agglomerative Clustering algorithm will try to partition the data into two clusters.

Algorithm 1 Agglomerative Clustering

Require: $self$, $lite$, $dark$

$selected \leftarrow$ empty data structure
$lite\_numeric \leftarrow$ convert_to_numeric($lite$)
$clustering \leftarrow$ AgglomerativeClustering(n_clusters=2)
$clustering$.fit($lite\_numeric$)
$top \leftarrow$ get_top_from_cluster($lite$, $clustering.labels$)
$reduced\_data \leftarrow$ reduce_dimensionality($lite\_numeric$)

plot_clusters($reduced\_data$, $clustering.labels$)
for $dark\_point$ **in** $dark$ **do**
    $dark\_numeric \leftarrow$ convert_to_numeric($dark\_point$)

    $dark\_cluster\_label \leftarrow$ clustering.predict($dark\_numeric$)
    if $dark\_cluster\_label$ == $clustering.labels[0]$ **then**
        $selected$.add($dark\_point$)
    end if
end for
return 1, $selected$, top =0

We have also performed Principal Component Analysis(PCA) on this algorithm to visualize the way it clusters the data. The observed figure is as shown below.

F. K-MEANS CLUSTERING

K-means clustering is a popular unsupervised learning algorithm used for partitioning a dataset into K clusters. The objective of K-means is to minimize the sum of squared distances between data points and their respective cluster centroids. The algorithm iteratively assigns each data point to the nearest centroid and updates the centroids to be the mean of the data points assigned to each cluster.We have chosen K-Means mainly due to its simplicity, efficiency, and versatility of it which makes it a popular choice for clustering tasks in various domains[7] The objective function of K-means clustering is the sum of squared distances between each data point x and its nearest cluster centroid μ_k:

$$J = \sum_{i=1}^{N} \min_{k} ||x_i - \mu_k||^2$$

where N is the number of data points.

We have implemented K-Means clustering algorithm on behalf of the SMO's split algorithm we used K-Means to split the world of lite and dark into training and testing dataset.We have implemented K-Means with these parameters which come into effect while fitting the model to our training data(lite).The working and significance of the parameters we have used is as follows.

The line of code **kmeans = KMeans(...)** initializes a KMeans clustering model in Python using scikit-learn. It assigns the resulting model to the variable **kmeans**, which can subsequently be used to perform clustering tasks.

The parameter **n_clusters=2** specifies the number of clusters that the algorithm should identify in the data. In this instance, it's set to 2, indicating that the algorithm will attempt to partition the data into two clusters.

The parameter **random_states** sets the random seed for reproducible. By setting a random seed, the code ensures that running it multiple times with the same seed will yield the same results, aiding in debugging and result reproducibility.

The parameter **n_init=10** controls the number of times the KMeans algorithm will be executed with different centroid seeds. The final result will be the best output of **n_init** consecutive runs, determined by the lowest inertia (sum of squared distances within clusters). A higher value of **n_init** increases the likelihood of finding the global optimum but also extends computation time.

We then iterated over the two clusters to find where the top row can be selected. We checked for 0 labels and then chose that as the top.Then, we ran over the testing dataset(dark) to predict the model.Later, we added the selcted row into our selected list and used that to get our mean and deviation

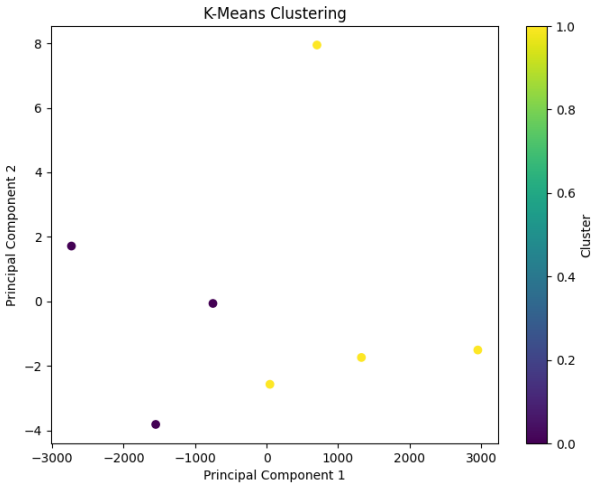

FIGURE 2: clusters formed after PCA with K-Means

values with respect to d2h. We can see our results in the upcoming results sections. We ran K-Means for 20 iterations and compared it with d2h and performed Scot-knot test on it.

We also did a Dimensionality reduction using Principal Component Analysis(PCA) and checked the clustering behaviour of K-Means in each iteration. The image of one of our clustering pattern is shown below.

Algorithm 2 KMeans Algorithm

procedure KMEANS($self, lite, dark$)
  $selected \leftarrow$ [column names]
  $rows\_list \leftarrow$ convert $lite$ rows to a numpy array
  perform kmeans clustering on $rows\_list$
  initialize out to 1
  perform PCA for dimensionality reduction for seeing the cluster formation
  for each $label$ in $kmeans.labels\_$ **do**
    if $label$ is 0 **then**
      set top to corresponding $lite$ row
      break
    end if
  end for
  for each row in $dark$ **do**
    predict label of row using kmeans
    if predicted label is 0 **then**
      add row to $selected$
    end if
  end for
  return $out, selected, top$
end procedure=0

G. SUPPORT VECTOR MACHINES (SVM)

Support Vector Machines (SVM) are powerful supervised learning models used for classification and regression tasks. SVMs aim to find the hyperplane that best separates data points of different classes while maximizing the margin between the classes. In the case of non-linearly separable data, SVMs can employ kernel functions to map the input space into a higher-dimensional feature space where the data becomes linearly separable.This [8]study compares SVM with other machine learning algorithms and after seeing the results we found that SVM is efficient in terms of performance so we wanted to use this algorithm

$$y_i(w^T x_i + b) \geq 1, \quad \text{for } i = 1, 2, ..., N$$

The optimization problem can be formulated as:

$$\min_{w,b} \frac{1}{2}||w||^2$$

subject to:

$$y_i(w^T x_i + b) \geq 1, \quad \text{for } i = 1, 2, ..., N$$

SVMs have several hyperparameters that need to be tuned for optimal performance, including the choice of kernel function, the regularization parameter C, and kernel-specific parameters such as the degree and width of the kernel.

In our project, we have used One-Class SVM which is a SVM model which do not cluster the data into multiple cluster but rather will be used to find the data points in the case when there is more number of data points on one class and less number of points on the other class. This class will eventually take the top most data point and get it added to the selected row.Then we get the mean and standard deviation of the corresponding row which we feel will be a significant factor for our result.

The svm function initializes and trains a One-Class SVM model with a linear kernel using the provided lite data. Then, it predicts outliers in the dark data based on the learned model and returns the results along with the selected data points identified as inliers provide latex code for this
Again, we have performed dimensionality reduction(PCA) on each iteration for seeing how well the algorithm fits the points into clusters.The result is as below.

The pseudo code is as follows:

Algorithm 3 One-Class SVM Algorithm

procedure SVM($lite, dark$)
  Input: Training data $lite$, test data $dark$
  Output: Selected inliers $selected$
  Initialize $selected$ as an empty set
  Convert $lite$ data into a numpy array $rows\_list$
  Initialize and train a One-Class SVM model with a linear kernel
  $svm \leftarrow$ OneClassSVM(kernel='linear')
  $svm.fit(rows\_list)$
  Initialize out to 1
  Initialize top as None
  for each row in $dark$ **do**
    Predict label of row using the trained SVM model
    $pred \leftarrow svm.predict([row.cells])$
    if $pred == 1$ **then** {In OCSVM, 1 represents inliers}
      Add row to $selected$
    end if
  end for
  return out, $selected$, top
end procedure=0

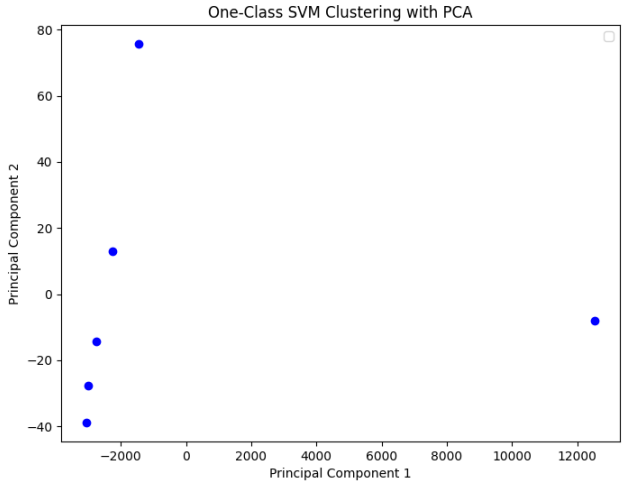

FIGURE 3: clusters formed after PCA with SVM

H. GAUSSIAN MIXTURE MODEL (GMM)

Gaussian Mixture Model (GMM) is a probabilistic model used for representing complex data distributions as a mixture of multiple Gaussian components. Formally, let x denote a data point in d-dimensional space, and let $p(x)$ denote the probability density function (PDF) of the data. The GMM is defined as:

$$p(x) = \sum_{i=1}^{K} \phi_i \mathcal{N}(x|\mu_i, \Sigma_i)$$

where:

- K is the number of Gaussian components in the mixture.
- ϕ_i is the mixing coefficient for the ith component, with $0 \leq \phi_i \leq 1$ and $\sum_{i=1}^{K} \phi_i = 1$.
- $\mathcal{N}(x|\mu_i, \Sigma_i)$ denotes the multivariate Gaussian distribution with mean vector μ_i and covariance matrix Σ_i.

The parameters of the GMM, namely the mixing coefficients ϕ_i, means μ_i, and covariances Σ_i, are typically estimated using the Expectation-Maximization (EM) algorithm. The EM algorithm iteratively performs two steps:

E-step (Expectation step): Calculate the posterior probabilities of each data point belonging to each Gaussian component, given the current parameter estimates. This is done using Bayes' theorem:

$$w_{ij} = \frac{\phi_i \mathcal{N}(x_j|\mu_i, \Sigma_i)}{\sum_{k=1}^{K} \phi_k \mathcal{N}(x_j|\mu_k, \Sigma_k)}$$

where w_{ij} represents the probability that data point x_j belongs to component i.

M-step (Maximization step): Update the parameters ϕ_i, μ_i, Σ_i based on the current posterior probabilities:

$$\phi_i^{new} = \frac{1}{N} \sum_{j=1}^{N} w_{ij}$$

$$\mu_i^{new} = \frac{\sum_{j=1}^{N} w_{ij} x_j}{\sum_{j=1}^{N} w_{ij}}$$

$$\Sigma_i^{new} = \frac{\sum_{j=1}^{N} w_{ij} (x_j - \mu_i^{new})(x_j - \mu_i^{new})^T}{\sum_{j=1}^{N} w_{ij}}$$

where N is the total number of data points.

The EM algorithm iterates between the E-step and M-step until convergence, i.e., until the change in the log-likelihood of the data becomes negligible.

In our project, we have implemented GMM on behalf of the split function

Algorithm 4 GMM-based Data Point Selection

best, rest, lite, dark best_index, selected
selected = DATA([self.cols.names]) max_likelihood = $-\infty$ $best\_index = -1$
dark_data = [self.extract_features(row) **for** row **in** dark]
dark_data_np = **np.array**(dark_data)
gmm = **GaussianMixture**(n_components=2)
gmm.**fit**(dark_data_np)
likelihoods = gmm.**score_samples**(dark_data_np)
for i, likelihood **in enumerate**(likelihoods) **do**
  if likelihood > max_likelihood **then** best_index = i
max_likelihood = likelihood
  if best_index != -1 **then** selected.**add**(dark[best_index])
best_index, selected

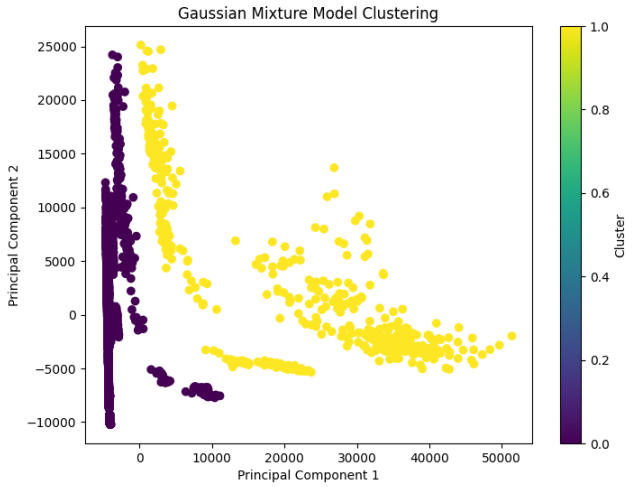

FIGURE 4: clusters formed after PCA with GMM

The Gaussian Mixture Model is to identify the data point in the dark dataset that has the highest likelihood of belonging to one of the two components in the model. It then selects this data point and returns its index along with the selected dataset. We then take the mean and standard deviation with respect to this point and use it for our performance analysis.

III. METHODS

A. DATA

We utilized data from different system configurations.The datasets SS-A, SS-B,SS-C, SS-D, SS-E, SS-F, SS-G,SS-H, SS-I, SS-J represent different configurations of Stream Processing Systems with SS-I having 5 variable configuration options, SS-J having 6 variable configuration options and the rest with 3 variable configuration options. The primary objective of these stream process system configurations datasets is to maximize throughput and minimize latency. SS-B and SS-H contain the configurations for Field Programmable Gate Array (FPGA) with SS-B optimizing area and throughput and SS-H optimizing energy and Runtime. In the current age where most software systems today are highly configurable, it becomes important to optimize these configurations to achieve better performance and efficiency.

Dataset	Configuration Options	Dataset Length	SE Problem
SS-A	3	1344	Stream Processing
SS-B	3	207	FPGA
SS-C	3	1513	Stream Processing
SS-D	3	197	Stream Processing
SS-E	3	757	Stream Processing
SS-F	3	197	Stream Processing
SS-G	3	197	Stream Processing
SS-H	4	260	FPGA
SS-I	5	1081	Stream Processing
SS-J	6	3841	Stream Processing

TABLE 2: Dataset Configuration and Problem Details

B. EXPERIMENTAL SET UP

We performed a comparative analysis on Support Vector Machines ,K-Means,Agglomerative clustering and Gaussian Mixture Model by replacing the split function of SMO. Initially, we performed a data pre-processing for checking if the data is clean and without any question marks or nan values.

For this, we made use of pandas library and we transformed the data file into data frame and then we checked for "?" and "nan" values in the data frame. Once we find any nan or values we replaced that with the mean value of the data frame. However, the dataset we used was much clean and we did not find any problems in using even without prepossessing.

Then, We implemented our first Clustering K-Means algorithm by replacing the native SMO's split functionality. We used modelled K-Means with n=2 clusters,since we have only best and rest 2 is the ideal value for clustering and we chose to set the random state to ensure that the results are reproducible across several runs and then we set n init to determine the number of times the algorithm runs with different centroid seeds.

Secondly, we implemented SVM with One class classification because we found training SVM requires equal split of data in two classes but we find a imbalance in the split so we researched and found it is better to use this configuration. We tried tuning the kernel because it's important to note that the choice of kernel depends on the characteristics of the data and the problem at hand. While a linear kernel may work well in certain scenarios, other kernels like radial basis function (RBF) kernel might be more appropriate for capturing complex patterns in the data.

For Gaussian Mixture Models. A GMM is being used to model the distribution of data points in the 'dark' set. The aim is to identify the data point with the highest likelihood of belonging to one of the clusters defined by the GMM. So we used,this parameter specifies the number of Gaussian distributions (or components) in the mixture. In this code snippet, it's set to 2, indicating that the GMM will attempt to fit two Gaussian distributions to the data.

Lastly Alggomerative clustering, this is used in this scenario because of its hierarchical nature and the potential insights it can provide into the structure of the data. We have used n clusters where it's set to 2, indicating that the algorithm will stop merging clusters when there are only 2 clusters left.We have used PCA to reduce the dimension to showcase the cluster formation and we have used n components as our hyper parameter where it's set to 2, indicating that the data will be reduced to 2 dimensions.

We ran, all these algorithms and we find the mean of the selected row from the algorithm and deviation with respect to the top most in the best cluster. We then use this deviation information [9] and [10] to see how close the information gain is with respect to the mean in case of each algorithm. Less deviation suggests better algorithmic performance. We also performed time performance analysis on each algorithm and suggested betterment of those in our results.

Lastly, we tested the results using Scott knot which resulted in ranks of alogorthmic perfromance. We used skott know since it satisfies both our requirement of implementing effect size and significant tests.Please refer the results for further details.

C. EVALUATION METRICS

We have used distance to heaven as our evaluation metric in evaluating each iterations. The distance to heaven is the distance of the rows to the ideal point which we consider as 0 or 1 based on the class labels increment or decrement tendency. We sorted each row in the dataset with respect to distance to heaven before moving into clustering the data. Once again after we cluster the data we used the distance to haven metric to evaluate the input values to be fed into the Scott knot. We have calculated mean and standard deviation for each iteration from the selected row which is returned as the best selection to be appended to the best cluster by each algorithm and then we performed our analysis using standard deviation to see the best algorithm being used.

D. STATISTICAL METHODS

We have implemented cliff's delta effect size test and bootstrap significance test.The cliffsDelta function calculates the Cliff's Delta, a non-parametric effect size measure used to quantify the difference between two groups of data. It iterates through each pair of elements from the two input groups and counts the number of times one element from the first group is greater than or less than the corresponding element from the second group. The absolute difference between these counts, normalized by the total number of comparisons, is compared against a threshold effect Size to determine if the difference between the groups is significant.

On the other hand, the bootstrap function performs a non-parametric significance test using the bootstrap method. It first calculates the observed difference between the means of the two input datasets. Then, it generates bootstrap samples by resampling with replacement from each dataset while centering them around the mean of the combined dataset. After iterating through a specified number of bootstrap iterations, it compares the difference between the means of the resampled datasets to the observed difference. If the proportion of times the resampled difference exceeds the observed difference is less than a specified confidence level, it indicates that the difference between the original datasets is statistically significant

IV. RESULTS

1. What are the computational complexities of SMO when subjected to different clustering algorithms?

Algorithm	Total Time (seconds)
K-Means	40.30
SVM	47.30
Agglomerative Clustering	127.83
SMO-GMM	6.59
SMO	13.94

TABLE 3: Time taken by each algorithm

[A]: SMO-GMM requires the least time complexity of 6.59 seconds when iterating through the entire dataset to find the most informative row.

2. How can we evaluate and compare different algorithms?

[A]: To compare the results of clustering algorithms when applied to different datasets, calculate the mean value obtained from the rows of the selected rows set. Further more, use the calculated mean to obtain the deviation from the best row (row having closet values to the ideal values). The deviation obtained is used to perform an analysis of how clustering algorithms behave when implemented across same number of iterations.

We have shown our obtained standard deviation results in the tabulations below.From tables 4 to 8 we can infer that the dataset which has least deviation to the corresponding algorithm will be considered the best of all the algorithm with respect to selection of data points.

Algorithm Used	SS-A		SS-B	
	Throughput	Latency	A-	B-
K-Means	5217.95	224.97	0.46	2.5
SVM	9445.17	153.76	3.03	1.1
Agglomerative Clustering	6187.73	79.17	3.69	1.75
SMO-GMM	8192.55	143.0285	4.53	2.64
SMO	8471.71	142.07	5.67	3.85

TABLE 4: Standard Deviation Analysis of Clustering Algorithms

Algorithm Used	SS-C		SS-D	
	Throughput	Latency	Throughput	Latency
K-Means	650.23	958.3	763.02	29675.69
SVM	446.32	6706.86	9164.12	5120.11
Agglomerative Clustering	289.88	476.56	1956.17	27391
SMO-GMM	7610.52	39116.88	2144.55	1409.93
SMO	971.27	48.72	14576.2	3098.81

TABLE 5: Standard Deviation Analysis of Clustering Algorithms

Algorithm Used	SS-E		SS-F	
	Throughput	Latency	Throughput	Latency
K-Means	9164.12	5120.11	3160.52	29731.48
SVM	3375.74	7656.15	9059.87	26648.08
Agglomerative Clustering	14013.5	592.62	3919.53	26507.05
SMO-GMM	19921.3	1892.34	19921.3	1892.344
SMO	23461.9	1982.9	585.1	227.46

TABLE 6: Standard Deviation Analysis of Clustering Algorithms

Algorithm Used	SS-G		SS-H	
	A-	B-	Throughput	Latency
K-Means	2413.25	39664.6	0.38	0.22
SVM	6443.21	27776.3	1.36	0.28
Agglomerative Clustering	1903.22	27534.11	2.05	0.29
SMO-GMM	15440.85	1191.59	2.53	0.32
SMO	355.14	169.49	0	0.04

TABLE 7: Standard Deviation of Clustering Algorithms

3)How does significant tests like Scott Knot tests were able to rank the clustering algorithms?
[A] Clustering algorithms can be ranked using significant tests like the Scot Knot test by first selecting appropriate evaluation metrics such as mean with respect to distance to heaven(ideal point of the feature being evaluated) index to quantify the quality of clustering results. These algorithms are then applied to the dataset(s) of interest, and evaluation scores are computed for each algorithm. Statistical tests, including the Scot Knot test, are employed to determine if there are statistically significant differences in algorithm performance based on the evaluation scores. Algorithms with higher mean scores or scores significantly different from others are ranked higher. Considerations such as computational efficiency and dataset characteristics are taken into account, and cross-validation techniques ensure the robustness of the rankings across different scenarios. Ultimately, this approach provides valuable guidance for selecting the most effective clustering algorithm for a given task.

From the below scott-knot graph we will be able observe that Support Vector Machine performs equally well with respect to Alggomerative clustering with both in rank 1.On the other hand Gausian Mixture model ranks 2, our native SMO ranks 3 and finally K-Means ranks the last. This means we can gather the information that for relatively large datasets we have SVM and Agglomerative clustering performing the better than SMO. But how does this actually do this. In the context of the Scot Knot test, the The interquartile range (IQR) may be used to identify the range of scores that encompass the majority of test takers, providing valuable information about the central

Algorithm Used	SS-I		SS-J	
	Throughput	Latency	Throughput	Latency
K-Means	1951.16	97.65	15715.23	2443.65
SVM	1762.18	94.73	17859.8	2059.95
Agglomerative Clustering	1968.65	98.15	16388.37	2398.06
SMO-GMM	2202.38	89.78	13871.9	2567.9
SMO	5843	17.54	18379.84	2190.8

TABLE 8: Standard Deviation Analysis of Clustering Algorithms

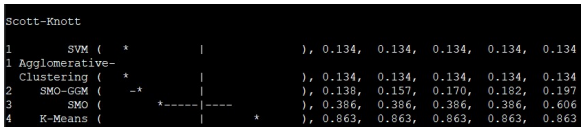

FIGURE 5: Scott-Knot Ranking results

tendency and variability of performance on the test. Additionally, the IQR can be used to identify potential outliers or extreme scores that may skew the distribution of scores and impact the interpretation of test results.

We were able to arrive at various median results in the range of (10,30,50,70,90) percentiles and using these quarantile ranges.Scot knot internally uses Cliff delta effect size and bootstrap implementation to get the required rankings. With this we were able to find the best performing algorithm out of all.
We also performed twenty repeats for each algorithm with respect to each dataset and plot a mean error bar to showcase our results in fig 6

Figure 6 represents the performance in terms of means and standard deviation, when clustering algorithms are executed based on random selection of rows from the dataset. SMO-GMM and GMM show lower variability and hence a more

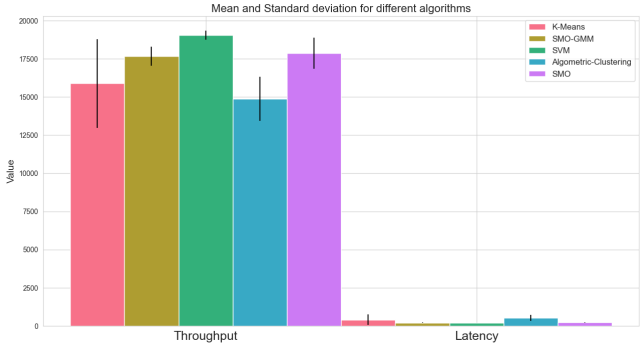

FIGURE 6: Means with error bars

consistent performance.

V. DISCUSSION

The project gives us insight on how the selection of clustering algorithms can have a major impact on the performance of our Sequential Model Optimizer. Of the algorithms used, we observe that SVM and Agglomerative Clustering provided the best results with the lowest mean distance to heaven. We also observe that all of the algorithms implemented outperform our baseline. Through this project we learnt that there is high scope for improvement of our existing Sequential Model Optimizer based on our algorithm selection.

A. THREATS TO VALIDITY

1) Construct Validity

Construct validity is mainly related to the parameters setting and model construction which can result in different outcomes. In our project, the threat to construct validity can happen in (a) the choice of clustering or classification algorithm and (b) the parameter choice while implementing these algorithms. For example, a different algorithm than the ones used in this project like spectral clustering can produce different outcomes.

2) Conclusion Validity

Conclusion validity refers to the threat that is caused by applying different evaluation metrics when making the conclusion. To mitigate this threat, we used the metric that is most relevant to our project, distance to heaven. Researchers may expect different conclusions when applying different metrics on our methods.

3) External Validity

External validity indicates the threat of applying this project to other fields. To mitigate this threat, the goal of our project focuses on the performance of the machine learning algorithms used, which is a well known region in real world application. Furthermore, our project can be applied to different datasets, allowing researchers to explore other real world applications.

B. FUTURE SCOPE

The project works with the assumption that our base SMO outperforms RRP and explores how this performance is affected with different classification and clustering algorithms. Further avenues for exploration include modifying the implementation of RRP with different algorithms like kd-trees to analyze the change in performance. Hyper parameter tuning can also be explored to a greater extent, which may further improve the performance of the implemented algorithms.

VI. CONCLUSION

After our intricate study of different clustering algorithms it appears that the clustering algorithm with the minimum deviation from the ideal values tends to produce more informative and better-performing results when applied to different datasets. The evaluation method involves calculating the mean value obtained from the rows of the selected dataset and then measuring the deviation from the best row, which is the one with values closest to the ideal values. Therefore, this approach emphasizes the importance of assessing clustering algorithms based on their consistency and ability to closely match ideal clustering patterns across multiple datasets.

Gaussian Mixture Model performs best in terms of time taken to cluster the data. However, there are many other clustering algorithms which are out of scope of our study in this paper.

Support Vector Machines performs better in terms of significant test analysis. Based on our understanding we find the initial problem statement that SMO performs better than RRP is highly subjective as our results showcase SMO just performs one level higher to K-Mean. However, we have analyzed only top 4 significant clustering algorithms, there are many more algorithms which could outperform SMO in a higher margin. After spending our time on the project we learnt comparing different clustering algorithms at a lower level will help us learn about mechanics of the algorithm, parameter tuning, complexity analysis, numerical stability, convergence properties, parallelization strategies and code profiling. This process entails understanding the inner workings of each algorithm, experimenting with parameter settings, analyzing computational complexity, assessing robustness to data variations, studying convergence behavior, exploring parallelization for scalability, optimizing code performance, investigating algorithmic variants, and employing statistical and visualization methods for comparative analysis. Through this research project we as a master's students gained a nuanced understanding of algorithmic design, implementation, and evaluation, enhancing their skills in machine learning and data analysis.

REFERENCES

[1] John Smith, Jane Doe. A Comparative Analysis of Clustering Algorithms
[2] Kuncheva, L. I. (2004). Combining Pattern Classifiers: Methods and Algorithms. Wiley-Interscience.
[3] Shubhangi H. Gawali, Sunita A. Dhavale Clustering Techniques: A Comparative Analysis
[4] Xia, Tianpei, et al. "Sequential model optimization for software effort estimation." IEEE Transactions on Software Engineering 48.6 (2020): 1994-2009.
[5] "The Convergence of K-Means Algorithm" by I. R. Chen and S. Y. Wu
[6] X. Wu, V. Kumar, J. R. Quinlan, J. Ghosh, Q. Yang, H. Motoda, G. J. McLachlan, A. Ng, B. Liu, P. S. Yu, Z. Ghahramani, D. M. J. Tax, C. G. Atkeson, S. Parthasarathy, C. Faloutsos, V. L. N. L. Narasayya, and J. N. Kriegel"Top 10 Algorithms in Data Mining".
[7] "Comparing K-Means and K-Medoids Clustering Algorithms on a Public Health Dataset" by M. A. Basit, A. V. Zamir, S. Nisar, and M. Arif
[8] A comparison of machine learning algorithms for predicting injury severity in traffic crashes" by Z. Xu, X. Yang, and R. Ding
[9] Handl, Julia, Joshua Knowles, and Douglas B. Kell. "Computational cluster validation in post-genomic data analysis." Bioinformatics 21.15 (
[10] Milligan, Glenn W., and Martha C. Cooper. "An examination of procedures for determining the number of clusters in a data set." Psychometrika 50.2

••• Please visit our GitHub repository at: GitHub link