ELLIPTICAL MIXTURE MODELS IMPROVE THE ACCURACY OF GAUSSIAN MIXTURE MODELS WITH EXPECTATION-MAXIMIZATION ALGORITHM

Xiaoying Zeng and Eugene Pinsky

Department of Computer Science, Metropolitan College, Boston University, Boston, MA, USA

ABSTRACT

This study addresses the limitations of Gaussian Mixture Models (GMMs) in clustering complex datasets and proposes Elliptical Mixture Models (EMMs) as a robust and flexible alternative. By adapting the Expectation-Maximization (EM) algorithm to handle elliptical distributions, the study introduces a novel computational framework that enhances clustering performance for data with irregular shapes and heavy tails. Leveraging the integration of R's advanced statistical tools into Python workflows, this approach enables practical implementation of EMMs. Empirical evaluations on three datasets Rice, Customer Churn, and Glass Identification demonstrate the superiority of EMMs over GMMs across multiple metrics, including Weighted Average Purity, Dunn Index, Rand Index, and Silhouette Score. The re- search highlights EMMs as a valuable tool for advanced clustering tasks and provides insights into their potential applications in handling real-world datasets with complex covariance structures.

KEYWORDS

Gaussian Mixture Models, Elliptical Distribution Mixture Models, Expectation-Maximization algorithm, Clustering, Multidimensional Data

1. INTRODUCTION

The Gaussian Mixture Model (GMM) is a statistical approach extensively applied in clustering tasks within machine learning. It operates under the assumption that the data is generated from a combination of multiple Gaussian distributions, each representing a cluster, with the parameters of these distributions being estimated iteratively. This probabilistic framework enables GMM to effectively model intricate data structures, making it particularly advantageous for datasets with overlapping clusters and continuous features. [15]. This method simplifies clustering tasks by leveraging the Expectation-Maximization (EM) algorithm [1], making it a powerful tool for identifying underlying patterns in data. GMMs have been extensively studied and applied in various fields, as discussed in the comprehensive work by McLachlan and Peel [2].

Gaussian Mixture Models (GMMs) rely on the assumption that the underlying data conforms to a Gaussian distribution. This inherently imposes constraints, as Gaussian distributions are symmetric and feature lighter tails, limiting their capacity to capture more complex or heavy-tailed data patterns. (e.g. [13, 14]) EMMs allow for elliptical distributions, which can handle asymmetry and "fat" tails, providing a more flexible and robust approach to clustering. The statistical properties and implications of elliptical distributions for modeling multivariate data have been explored by

Bibhu Dash et al: IOTBC, NLPAI, BDML, EDUPAN, CITE - 2025
pp. 87-106, 2025. IJCI – 2025 DOI:10.5121/ijci.2025.140206

Kent and Tyler [5]. Also, Peel and McLachlan [6] have developed robust methods for fitting mixtures of t-distributions, a specific case of elliptical distributions, which are more resilient to outliers and heavy tails. Additionally, the work by Samorodnitsky and Taqqu [7] on stable non-Gaussian processes provides a foundational understanding of the behavior of data that exhibits heavy tails, highlighting the importance of moving beyond Gaussian assumptions in probabilistic modeling.

However, while previous studies have laid a strong theoretical foundation for elliptical distributions and their applications, their practical integration into clustering methods remains underexplored. Specifically, the adaptation of the Expectation-Maximization (EM) algorithm for Elliptical Mixture Models (EMMs) has not been systematically studied, particularly in the context of handling real-world datasets with non-Gaussian patterns and complex covariance structures. Additionally, most existing work focuses on theoretical models without providing a robust framework for implementing EMMs in practical machine learning workflows. This lack of accessible implementations and empirical validations on diverse datasets highlights the need for further exploration in this area.

Therefore, the motivation behind this research stems from the inherent limitations of Gaussian Mixture Models (GMMs), which assume that data follows a Gaussian distribution characterized by symmetry and ”thin” tails. The limitations of GMMs in modeling non-Gaussian data have been well-documented [3, 4]. Real-world data often deviates from these assumptions, exhibiting more complex covariance structures and heavy tails. For instance, datasets in domains such as biomedical analysis, customer behavior modeling, and forensic science frequently display non-Gaussian patterns that GMMs fail to capture effectively. These challenges highlight the need for more flexible models that can accommodate complex data structures and better reflect real-world scenarios. This work is further motivated by the limited practical applications and algorithmic customizations of Elliptical Mixture Models (EMMs) in addressing such challenges, despite their theoretical promise.

This paper addresses the aforementioned gaps by making the following contributions:

1. **Customization of the EM Algorithm**: We adapt the EM algorithm to handle elliptical distributions, extending its applicability beyond Gaussian data. This customization enables EMMs to effectively model datasets with non-Gaussian patterns and complex covariance structures.
2. **Integration of Statistical Tools**: By integrating R’s advanced statistical functions with Python’s machine learning framework, we provide a seamless implementation for generating, fitting, and calculating elliptical probability densities, enhancing computational flexibility and accuracy.
3. **Improved Clustering for Complex Data**: The proposed method demonstrates superior clustering performance compared to GMMs on real-world datasets characterized by non-Gaussian patterns, as validated by experiments.
4. **Practical Applications**: We illustrate the utility of EMMs through comprehensive experiments on three real-world datasets: the Rice dataset, the Customer Churn dataset, and the Glass Identification dataset. The results reveal that EMMs outperform GMMs under various conditions in terms of clustering accuracy and robustness.

2. METHODOLOGICAL BACKGROUND

2.1. Maximum Likelihood Estimation and the EM Algorithm

Probability quantifies the likelihood of an event, while likelihood evaluates how well a model explains observed data. Maximum Likelihood Estimation (MLE) is a statistical method for estimating model parameters by maximizing the likelihood of the observed data. However, MLE faces challenges when dealing with latent variables, which are unobservable aspects of the data, such as cluster assignments in unsupervised learning.

The Expectation-Maximization (EM) algorithm addresses these challenges by iteratively applying MLE in two steps: the Expectation Step (E-step) and the Maximization Step (M-step). In the E-step, the algorithm calculates the posterior probabilities of latent variables given the current parameters. In the M-step, these probabilities are used to update the model parameters, maximizing the log-likelihood. This process repeats until the model converges, enabling the estimation of parameters in the presence of latent variables.

2.2. Concept of Elliptical Distributions

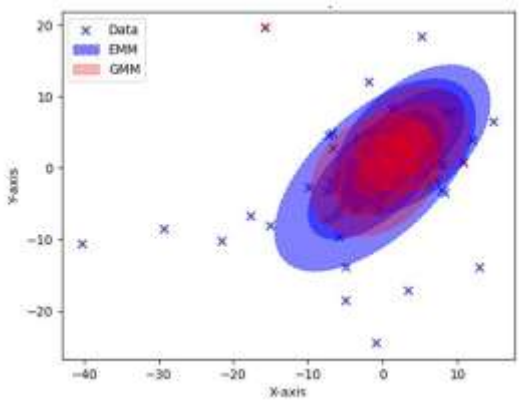

Figure 1. Comparison of clustering shapes between Elliptical Mixture Models (EMM) and Gaussian Mixture Models (GMM). The blue ellipses represent the clusters modeled by EMM, which allow for flexible shapes and can accommodate non-circular or elongated data distributions. In contrast, the red ellipses represent the clusters modeled by GMM, which are constrained to symmetric Gaussian distributions and struggle to capture data with elliptical patterns. The plot highlights the superior flexibility of EMM in capturing data distributions with varying eccentricities and orientations.

Elliptical distributions generalize the multivariate Gaussian distribution by allowing for greater flexibility in modeling data characteristics such as tail behavior and shape. Defined by their elliptical level sets, these distributions include key parameters[11]:

1. **Shape Parameter (α)**: Controls the heaviness of the tails.
2. **Location Parameter (δ)**: Specifies the central tendency, akin to the mean in Gaussian distributions.
3. **Covariance Matrix (Σ)**: Determines the shape and orientation of the ellipsoids.

Gaussian distributions are a special case of elliptical distributions with $\alpha = 2$, but elliptical distributions can accommodate data with heavier tails and skewed structures. This flexibility makes them a powerful tool for modeling complex datasets, as illustrated in Figure 1.

2.3. Elliptical Mixture Model

The Expectation-Maximization (EM) algorithm is widely used for estimating parameters in the presence of latent variables. When applied to elliptical mixture models, the algorithm is adapted to account for the elliptical characteristics of data distributions, making it more flexible than traditional Gaussian Mixture Models.

The algorithm involves iteratively computing the responsibilities for each data point in the Expectation (E) step and updating the model parameters in the Maximization (M) step. For details on the mathematical formulation and implementation, refer to [11] and [12].

2.4. Differences Between Elliptical Mixture Models and Gaussian Mixture Models in Handling Datasets

Both EMM and GMM rely on the EM algorithm for parameter estimation, but their underlying assumptions and applications differ significantly. Table 1 summarizes these differences.

Table 1. Comparison of Gaussian Mixture Models and Elliptical Mixture Models

Aspect	Gaussian Mixture Models	Elliptical Mixture Models
Assumption	Normally Distributed Data	Flexible Shape and Tail Behavior
Probability	Based on Gaussian Density Function	Based on Elliptical Density Function
Parameter	Updates mean (μ_k) and covariance (Σ_k)	Updates location (δ_k), covariance, and shape (α_k)
Process Steps	Multi-Normal fitting → Density estimation for Multi-Normal function	Data → Covariance Matrix → Elliptical fit → NearPD regularization → Elliptical stable probabilities
Computation Time	Faster due to native Python implementation	Slower due to R function calls via rpy2

By incorporating elliptical distributions, EMM provides greater flexibility and robustness, making it partic- ularly suitable for datasets with non-Gaussian characteristics.

3. INTEGRATING R'S STABLE 5.3 PACKAGE WITH PYTHON FOR ELLIPTI- CAL MIXTURE MODELS

3.1. Motivation for Integration

Elliptical Mixture Models (EMM) require robust tools for defining, fitting, and estimating elliptical distri- butions. While R's STABLE 5.3 package offers advanced statistical tools for robust elliptical distribution fitting, Python provides a more convenient environment for data handling, visualization, and machine learn- ing. Combining these strengths allows us to achieve robust modeling while leveraging Python's efficient workflows.

The integration is achieved through the rpy2 library, which acts as a bridge between Python and R. This package allows Python users to call R functions, access R objects, and utilize R's extensive library ecosystem within Python workflows. By combining R's statistical power with Python's flexibility, we can create an efficient pipeline for robust statistical modeling and data analysis.

3.1.1. Overview of rpy2

The rpy2 library serves as a Python interface to R, facilitating seamless communication between the two languages. One of its primary advantages is the ability to call any R function directly from Python through the rpy2.robjects module. This direct functionality significantly enhances the integration of R's statistical capabilities into Python workflows. Additionally, rpy2 supports automatic conversion between Python data structures, such as NumPy arrays and pandas DataFrames, and their R counterparts, including matrix and data.frame. This feature eliminates the need for manual data type adjustments, streamlining the integration process.

The library also allows users to load and utilize R packages within Python scripts. For instance, the STABLE package, which offers robust tools for elliptical distribution modeling, can be leveraged in Python through rpy2. Furthermore, rpy2 is particularly well-suited for interactive applications, as it integrates seamlessly with Jupyter Notebooks. This compatibility makes it an excellent choice for exploratory data analysis and visualization tasks.

By combining Python's versatility and machine learning capabilities with R's statistical expertise, rpy2 enables the creation of efficient and powerful workflows for complex data science projects.

3.1.2. When to Use rpy2?

The rpy2 library proves particularly valuable in several scenarios, especially those requiring the integration of Python and R's unique strengths. First, it is indispensable for accessing specialized R packages that provide advanced statistical tools or visualization capabilities unavailable in Python. For example, tasks such as fitting elliptical distributions using the STABLE package can be efficiently accomplished with rpy2.

Second, the library is highly effective in interdisciplinary projects that combine R's statistical analysis capabilities with Python's machine learning workflows. By facilitating seamless integration, rpy2 enables researchers to leverage the best features of both languages without compromising efficiency.

Third, rpy2 supports the reuse of existing R-based models or analyses within Python environments. This eliminates the need to rewrite models, saving time and effort while ensuring consistency in statistical computations. Finally, it enhances workflows involving complex visualizations, allowing users to utilize R's sophisticated visualization libraries, such as ggplot2, in conjunction with Python's data processing capabilities.

By bridging the gap between Python and R, rpy2 creates an interoperable and efficient pipeline for data science tasks, enabling researchers to address complex challenges by combining the strengths of both languages.

3.1.3. Further Reading

For a detailed introduction to rpy2, refer to the official documentation [17]. Additional examples and use cases can be found in Mu¨ller's book [16] and Urbanek's research on R-Python integration [18].

3.2. Overview of the Implementation Process

The integration follows these steps:

1. Setting up the Python environment: Import necessary libraries, configure the rpy2 environment, and activate data conversion between NumPy and R.
2. Defining and fitting elliptical distributions: Use R's mvstable.elliptical() to define the distribu- tions and mvstable.fit.elliptical() to estimate their parameters, as explained in Appendix A.2.
3. Handling positive definite matrices: Employ the nearPD() function to ensure covariance matrices are valid for probability density calculations.
4. E-step in the EM algorithm: Calculate the posterior probabilities (responsibilities) of each data point for all distributions.
5. M-step in the EM algorithm: Update the parameters of each distribution and the mixture weights based on the calculated posterior probabilities.
6. Iterative convergence: Repeat E-step and M-step until the algorithm converges to a stable solution.

Each of these steps utilizes R's robust statistical capabilities, while the workflow is managed within Python for ease of integration and analysis.

3.3. Key Considerations in Integration

Handling Non-Positive Definite Matrices: The output covariance matrix from mvstable.fit.elliptical() may not always be positive definite, which is required for calculating the PDF using dmvstable.elliptical(). To address this, we use the nearPD() function from the R package Matrix, which adjusts the matrix to be positive definite while preserving its original structure.

Data Transformation Between Environments: The numpy2ri module of rpy2 ensures seamless con- version of data between Python's NumPy arrays and R's matrix objects, allowing the integration to work smoothly without manual data type adjustments.

3.4. Code Implementation

The implementation details for integrating R's advanced statistical capabilities into Python workflows are outlined in Appendix A. These include step-by-step instructions for importing libraries, defining and fitting elliptical distributions, and calculating their probability density functions. Specifically, Appendix A.1 describes the process of importing the necessary libraries and managing potential warnings during the integration. Appendix A.2 provides details on defining and fitting elliptical mixture models, ensuring accurate parameter estimation. Finally, Appendix A.3 explains the procedure for calculating probability density functions for elliptical distributions.

By following the methodology presented in these appendices, users can effectively utilize R's advanced statistical tools within Python workflows, enabling robust and seamless modeling of elliptical distributions. This approach not only enhances computational flexibility but also bridges the gap between the statistical strengths of R and the machine learning capabilities of Python.

3.5. E-step in the EM Algorithm

During the Expectation (E-step) of the algorithm, the posterior probabilities, also referred to as responsi- bilities, are calculated for each data point relative to all distributions in the mixture

model. This process involves several critical steps. First, the posterior probabilities for each data point are computed using the probability density function (PDF) and the corresponding mixture weights. Next, these probabilities are normalized across all distributions to ensure they sum to one for each data point. Finally, labels are assigned to each data point based on the distribution with the highest posterior probability.

The computed posterior probabilities are instrumental in determining a weighted log-likelihood for the data, which acts as a key convergence metric for the Expectation-Maximization (EM) algorithm. A detailed Python implementation of the E-step is provided in Appendix A.4.

3.6. M-step in the EM Algorithm

In the Maximization (M-step) of the Expectation-Maximization (EM) algorithm, the parameters of the distributions and the mixture weights are updated based on the responsibilities computed in the Expectation (E-step). This step begins with recalculating the mixture weights as the sum of the responsibilities for each distribution. Subsequently, the mean, covariance, and additional parameters of each distribution are updated using the weighted data points. To ensure robustness, R's `mvstable.fit.elliptical()` function is employed to estimate key parameters, including the shape (α), location (δ), and other relevant properties of the distributions.

These updates allow the model to better capture the underlying structure of the data in subsequent iterations. Further details on the implementation of the M-step can be found in Appendix A.4.

3.7. Iterative Convergence

The Expectation-Maximization (EM) algorithm iteratively alternates between the Expectation (E-step) and Maximization (M-step) to refine model parameters until convergence. This process ensures that the clustering procedure yields optimal and robust parameter estimates. The convergence of the algorithm is monitored using specific criteria to determine when stable values have been reached.

Firstly, **log-likelihood stability** is employed as a primary convergence criterion. The algorithm is considered to have converged when the change in the log-likelihood of the data between consecutive iterations falls below a predefined threshold, such as 10^{-6}. This approach ensures that the algorithm halts when further iterations yield no significant improvement in the model's fit.

Secondly, **parameter stability** provides an additional measure of convergence. This criterion evaluates whether updates to critical model parameters, such as mixture weights, means, and covariance matrices, become negligible across iterations. When these updates fall below a minimal threshold, the parameters are deemed stable, and the algorithm terminates.

Finally, a **maximum number of iterations** is defined to prevent infinite looping. This safeguard is particu- larly important in edge cases where the algorithm fails to converge due to complex data structures or poor initialization. By capping the number of iterations, the process ensures computational efficiency without compromising the reliability of the clustering results.

By employing these criteria collectively, the EM algorithm achieves a balance between convergence accuracy and computational efficiency, ensuring robust performance in clustering and parameter estimation tasks.

For detailed implementation of this iterative process, including the E-step and M-step functions, refer to Appendix A.4.

4. METHODOLOGY

4.1. Dataset Description

This study employs three datasets: the Rice dataset, the Customer Churn dataset, and the Glass Identification dataset. Each dataset presents unique characteristics and challenges, making them suitable for demonstrating the capabilities of Gaussian Mixture Models and Elliptical Mixture Models.

1. **Rice Dataset** [8]: This multivariate dataset includes 3,810 instances of rice grains from two species, with 7 morphological features extracted from images of each grain. These attributes, derived from grain images, include area, perimeter, lengths of the major and minor axes, eccentricity, convex area, and extent. The dataset also contains class labels identifying the species of each grain. The dataset is primarily used for classification tasks in the biology domain.
2. **Customer Churn Dataset** [9]: This multivariate dataset includes 13 features related to customer behavior, such as call failures, subscription length, usage frequency, SMS frequency, distinct called numbers, age, customer value, and churn status, with 3,150 instances. Binary features were removed because the EM algorithm, particularly when used with Gaussian Mixture Models (GMMs) or Ellip- tical Distribution Mixture Models (EMMs), assumes that the data follows a continuous distribution. Binary features, being categorical, do not naturally fit into these distributional assumptions, which can complicate the modeling process.
3. **Glass Identification Dataset** [10]: This multivariate dataset consists of 214 instances of glass samples, categorized into 6 types based on their oxide content.This dataset contains nine continuous features, including the refractive index and the weight percentages of various oxides such as sodium, magnesium, aluminum, silicon, potassium, calcium, barium, and iron. The target label identifies the type of glass, making it a valuable resource for classification tasks. Its primary applications are in physics and chemistry, particularly in forensic science, where it aids in determining glass types based on their chemical composition.

4.2. Data Preprocessing and Parameter Initialization

To ensure the datasets were suitable for clustering using Gaussian Mixture Models (GMMs) and Elliptical Mixture Models (EMMs), several preprocessing steps were applied. These steps prepared the data for effective clustering while maintaining the validity of distributional assumptions.

First, **standardization** was performed on the features of each dataset using StandardScaler. This process ensured that all features had zero mean and unit variance, mitigating the sensitivity of clustering algorithms to differences in feature scaling.

Second, **encoding labels** was carried out for the datasets. Class labels were transformed into numerical values using LabelEncoder, facilitating compatibility with clustering evaluation metrics, such as the confusion matrix and purity calculations.

Third, **categorical feature handling** was implemented, specifically for the Customer Churn dataset. Binary features were removed to align with the assumptions of GMM and EMM, which operate under the premise of continuous distributions. This step ensured that the models were applied appropriately to the datasets.

Following preprocessing, the clustering parameters were initialized as follows:

Centroids: Initial cluster centroids were randomly sampled within the range of feature values for each dataset. This approach ensured that the starting centroids were diverse and representative of the overall data distribution.

Covariance Matrices: Covariance matrices for each cluster were initialized as identity matrices. This configuration provided equal variance across all dimensions at the start of the clustering process, facilitating balanced parameter estimation.

Mixing Coefficients: Mixing coefficients were initialized with an even distribution across clusters unless explicitly specified. This ensured that no cluster was given undue weight during the early stages of the algorithm.
These preprocessing and initialization steps collectively ensured that the datasets were well-prepared for clustering, allowing GMMs and EMMs to operate effectively and produce meaningful results.

4.3. Evaluation Metrics

The performance of Gaussian Mixture Models (GMMs) and Elliptical Mixture Models (EMMs) was com- pared using several widely recognized evaluation metrics. These metrics provide complementary perspec- tives on clustering quality, allowing for a comprehensive assessment of model performance.

4.3.1. Weighted Average Purity

The Weighted Average Purity evaluates the alignment of clusters with the actual class labels by calculating the proportion of correctly grouped elements within each cluster. This metric is computed as the weighted average of the purity of each cluster, with higher values indicating a stronger correspondence between predicted clusters and true labels. Weighted Average Purity is particularly useful for assessing clustering outcomes when ground truth labels are available [19, 20].

4.3.2. Dunn Index

The Dunn Index assesses the compactness and separation of clusters. Compactness reflects the closeness of points within the same cluster, while separation measures the distance between distinct clusters. A higher Dunn Index indicates better clustering performance, as it reflects well-separated and compact clusters. This metric was originally introduced by J. C. Dunn to evaluate clustering quality in fuzzy partitions [21, 22].

4.3.3. Rand Index

The Rand Index measures the similarity between predicted cluster assignments and actual labels. It evaluates all pairs of samples to determine whether they belong to the same or different clusters in both the predicted and actual groupings. A Rand Index of 1 indicates perfect agreement between predicted and true clustering assignments. This metric has been widely employed in clustering

validation since its introduction by W. M. Rand [23, 24].

4.3.4. Silhouette Score

The Silhouette Score quantifies how similar a data point is to its own cluster compared to other clusters. It ranges from -1 to 1, with higher values indicating better clustering quality. The Silhouette Score combines insights into cluster cohesion (how tightly grouped the points in a cluster are) and separation (how distinct a cluster is from others). This metric, introduced by P. J. Rousseeuw, serves as a robust tool for interpreting and validating clustering results [25, 26].

4.4. Model Convergence

The training of Gaussian Mixture Models (GMMs) and Elliptical Mixture Models (EMMs) was conducted iteratively using the Expectation-Maximization (EM) algorithm. The convergence of the models was evaluated based on predefined criteria to ensure robust and stable solutions while minimizing the risk of overfitting or excessive computation.

The criteria were employed to determine convergence: Log-Likelihood Tolerance and Maximum Iterations.

The iterative process was terminated when the absolute difference in log-likelihood values between consec- utive iterations fell below a predefined tolerance threshold of 10^{-4}. This ensured that the models reached a stable state with minimal fluctuations in their likelihood values.

To safeguard against infinite loops in edge cases, a maximum limit of 1000 iterations was imposed. This constraint provided a balance between computational efficiency and the thoroughness of parameter opti- mization.

By adhering to these convergence criteria, the models achieved stable and reliable solutions, facilitating an accurate comparison of clustering performance across different datasets and methodologies.

5. RESULTS

This section compares the clustering performance of Gaussian Mixture Models (GMM) and Elliptical Mixture Models (EMM) on three datasets. We provide visualizations of clustering distributions and detailed analysis using evaluation metrics to highlight scenarios where Elliptical Mixture Models outperforms Gaussian Mixture Models and vice versa.

5.1. Rice Dataset

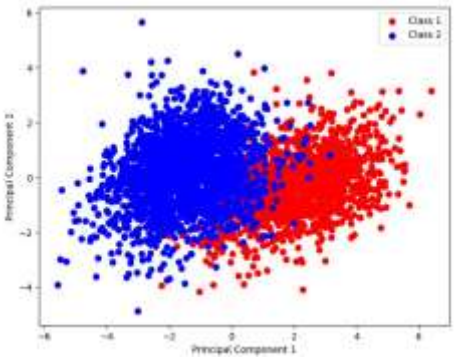

Figure 2. True Label for Rice Dataset

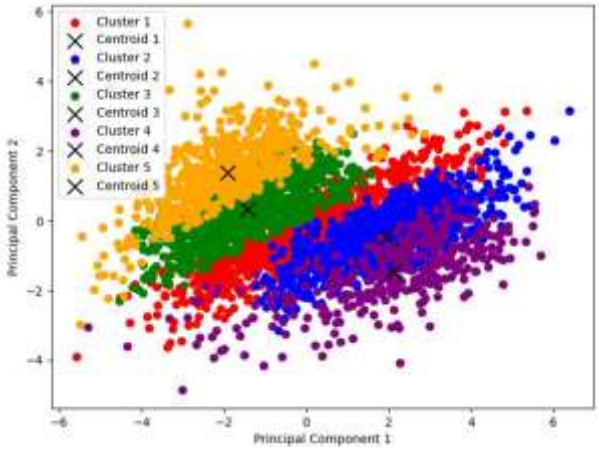

Figure 3. Gaussian Mixture Models for Rice Dataset

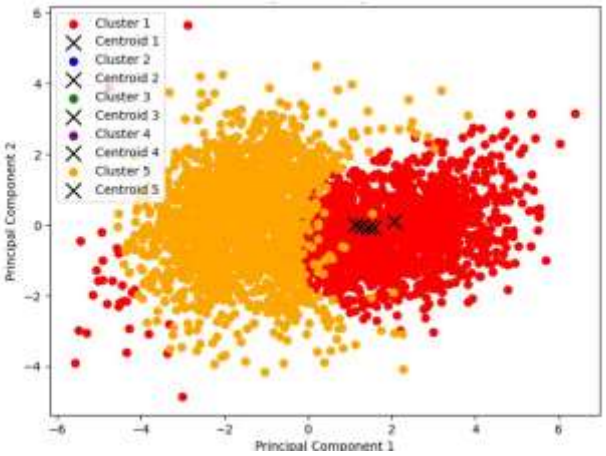

Figure 4. Elliptical Mixture Models for Rice Dataset

Using PCA for dimensionality reduction, these plots present the data in two dimensions. In each plot, different colors represent the identified clusters or classes, and the black 'X' marks indicate the centroids. Figure 2 illustrates the distribution of true class labels in the dataset, serving as a reference point for evaluating clustering outcomes. Figure 3 presents the optimized clustering results obtained using a Gaussian Mixture Model (GMM), which models the data as a combination of multiple Gaussian distributions. The Final Optimized Elliptical Clusters in the Figure 4, which resulted in 2 clusters instead of 5 as in GMM, suggests that the data may be better represented by fewer, more cohesive groups when considering elliptical rather than Gaussian distributions.

Table 2. Evaluation Metrics for Rice Dataset: This table compares the performance of Elliptical Mixture Models (EMM) and Gaussian Mixture Models (GMM). EMM outperforms GMM across all metrics. For example, EMM achieves a higher Weighted Average Purity (0.8837 vs. 0.8472), Dunn Index (0.0081 vs. 0.0079), Rand Index (0.7944 vs. 0.6175), and Silhouette Score (0.3734 vs. 0.0564), demonstrating better clustering performance and alignment with ground truth labels.

Table 2. Evaluation Metrics for Rice Dataset:

Metric	Elliptical MM	Gaussian MM
Weighted Average Purity	**0.8837**	0.8472
Dunn Index	**0.0081**	0.0079
Rand Index	**0.7944**	0.6175
Silhouette Score	**0.3734**	0.0564

5.2. Customer Churn Dataset

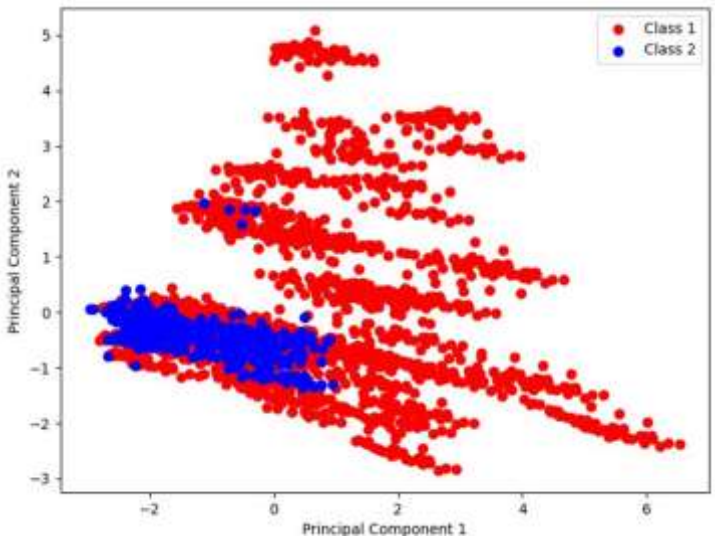

Figure 5. True Labels for Customer Churn Dataset

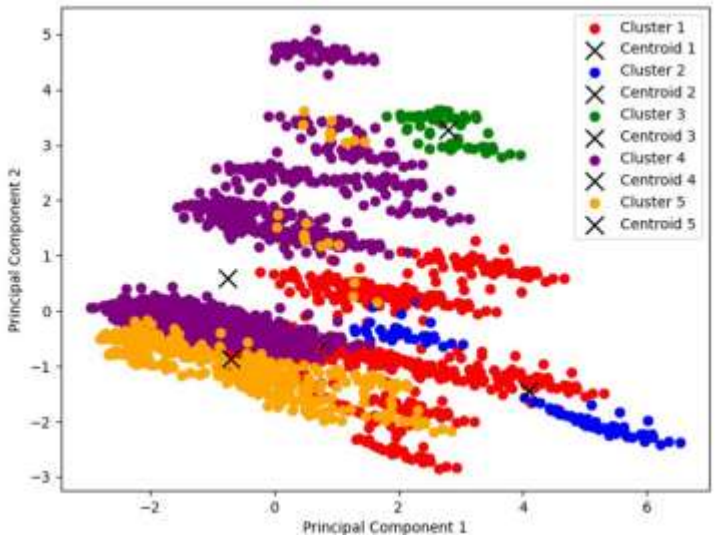

Figure 6. Gaussian Mixture Models for Customer Churn Dataset

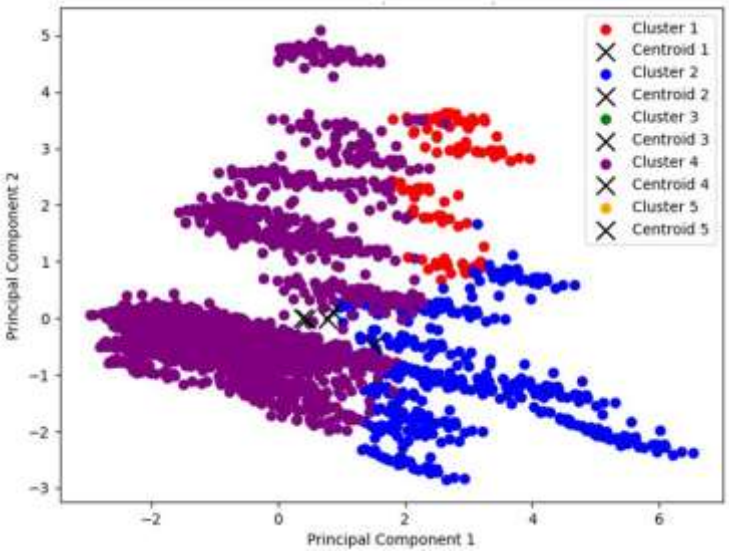

Figure 7. Elliptical Mixture Models for Customer Churn Dataset

Figure 5 shows the distribution of the original customer churn labels, with overlapping classes that suggest challenges for clustering. Figure 6, using a Gaussian Mixture Model (GMM), divided the data into 5 clusters but retained some overlap, reflecting GMM's assumption of spherical distributions. Figure 7, using an Elliptical Mixture Model (EMM), identified 3 more cohesive clusters, suggesting that the data is better represented by fewer elliptical clusters.

Table 3. Evaluation Metrics for Customer Churn Dataset: This table compares the performance of Elliptical Mixture Models (EMM) and Gaussian Mixture Models (GMM) on the Customer Churn dataset. EMM demonstrates superior performance across most metrics, with significantly higher Dunn Index (0.0182 vs. 0.0059), Rand Index (0.5385 vs. 0.4312), and Silhouette Score (0.2912 vs. 0.1119). Both models achieve the same Weighted Average Purity (0.8429), indicating similar alignment with true class labels. These results suggest that EMM provides better clustering quality, particularly in terms of cluster separation and compactness, compared to GMM.

Table 3. Evaluation Metrics for Customer Churn Dataset

Metric	Elliptical MM	Gaussian MM
Weighted Average Purity	**0.8429**	**0.8429**
Dunn Index	**0.0182**	0.0059
Rand Index	**0.5385**	0.4312
Silhouette Score	**0.2912**	0.1119

5.3. Glass Identification Number Dataset

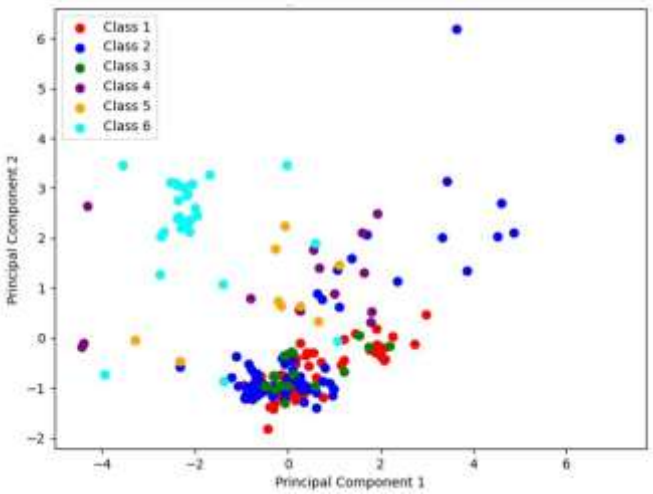

Figure 8. True Labels for Glass Identification Number Dataset

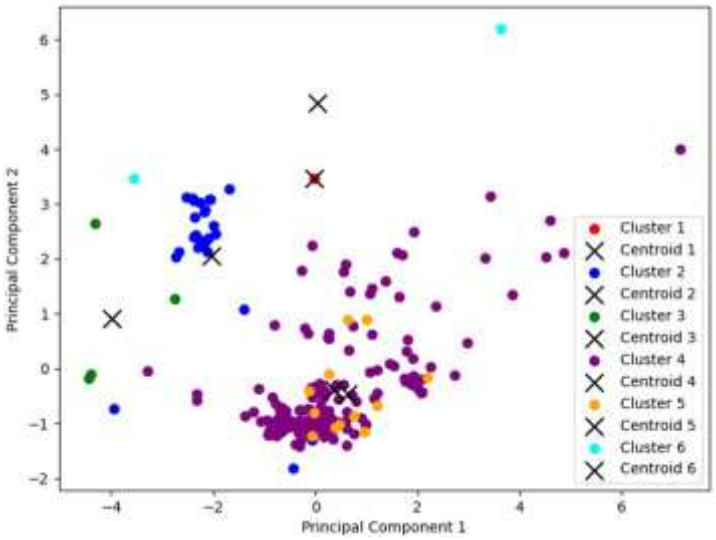

Figure 9. Gaussian Mixture Models for Glass Identification Number Dataset

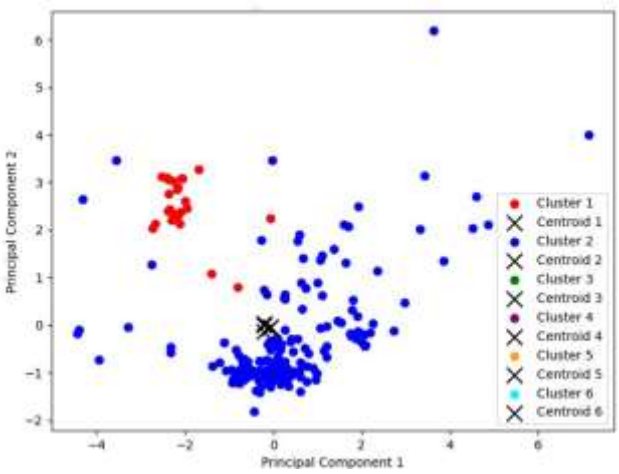

Figure 10. Elliptical Mixture Models for Glass Identification Number Dataset

The three images compare clustering results for the Glass Identification dataset using different approaches. Figure 8 shows the true distribution of the glass types, where some classes overlap, particularly in the central region, while others are more distinct. The GMM clustering in the Figure 9 results demonstrate the model's attempt to divide the data into six clusters, corresponding to the number of true labels. However, there is significant overlap between clusters, indicating that the Gaussian assumption may not perfectly capture the data's structure. In contrast, the EMM clustering in the Figure 10 simplifies the data into two broader clusters, suggesting that the elliptical model found the data structure to be more consistent with fewer, larger clusters. This approach reduces overlap and highlights a potentially underlying bimodal structure in the dataset, although it may lose some granularity compared to GMM.

Table 4. Evaluation Metrics for Glass Identification Dataset: This table compares the performance of Elliptical Mixture Models (EMM) and Gaussian Mixture Models (GMM) on the Glass Identification dataset. GMM performs slightly better in Weighted Average Purity (0.4813 vs. 0.4579) and Rand Index (0.5271 vs. 0.4446), indicating a closer match with true class labels in these metrics. However, EMM shows a clear advantage in terms of Dunn Index (0.0983 vs. 0.0346) and Silhouette Score (0.3525 vs. 0.1490), suggesting superior cluster separation and compactness. These results highlight the trade-offs between the two models, with EMM excelling in structural clustering quality and GMM aligning more closely with the ground truth.

Table 4. Evaluation Metrics for Glass Identification Dataset

Metric	Elliptical MM	Gaussian MM
Weighted Average Purity	0.4579	**0.4813**
Dunn Index	**0.0983**	0.0346
Rand Index	0.4446	**0.5271**
Silhouette Score	**0.3525**	0.1490

6. CONCLUSION

This study investigated the performance of Elliptical Mixture Models (EMMs) in comparison to Gaussian Mixture Models (GMMs) across three datasets: Rice, Customer Churn, and Glass Identification. The results demonstrated that EMMs exhibited superior performance in specific cases, particularly when han- dling datasets with complex and non-Gaussian data distributions.

However, it is important to note that these findings are based on a limited number of datasets. Consequently, EMMs should be regarded as a complementary alternative to GMMs, particularly in scenarios where the underlying data exhibits elliptical or non-Gaussian structures.

Beyond clustering, EMMs hold potential in various domains. Their ability to model elliptical or heavy-tailed distributions makes them suitable for **anomaly detection** in areas such as financial transactions, network security, and healthcare records. EMMs also provide robust solutions for **density estimation**, crucial for probabilistic modeling, risk assessment, and generative tasks. Additionally, their capacity to capture non- Gaussian distributions enhances **classification tasks**, improving probabilistic classifiers like Bayesian and semi-supervised learning models.

EMMs can further aid in **dimensionality reduction**, such as Principal Component Analysis, by better representing elliptical data distributions, and they show promise in **multimodal data integration**, enabling the analysis of diverse structured and unstructured datasets. While these applications highlight the versatility of EMMs, further research is needed to validate their efficacy across broader datasets and real-world scenarios. Future work should focus on integrating EMMs into machine learning pipelines and optimizing their computational performance.

DECLARATIONS

Conflict of Interest: We declare that there are no conflicts of interest regarding the publication of this paper.

Author Contributions: All the authors contributed equally to the effort.

Funding: This research was conducted without any external funding. All aspects of the study, including design, data collection, analysis, and interpretation, were carried out using the resources available within the authors' institution.

REFERENCES

[1] A. P. Dempster, N. M. Laird, and D. B. Rubin, "Maximum likelihood from incomplete data via the EM algorithm," J. R. Stat. Soc. B (Methodological), vol. 39, no. 1, pp. 1–22, 1977.

[2] G. McLachlan and D. Peel, Finite Mixture Models. Wiley, New York, 2000.

[3] D. M. Titterington, A. F. Smith, and U. E. Makov, Statistical Analysis of Finite Mixture Distributions. Wiley, New York, 1985.

[4] C. Fraley and A. E. Raftery, "Model-based clustering, discriminant analysis, and density estimation," J. Am. Stat. Assoc., vol. 97, no. 458, pp. 611–631, 2002.

[5] J. T. Kent and D. E. Tyler, "Redescending M-estimates of multivariate location and scatter," Ann. Stat., vol. 19, no. 4, pp. 2102–2119, 1991.

[6] D. Peel and G. J. McLachlan, "Robust mixture modelling using the t distribution," Stat. Comput., vol. 10, no. 4, pp. 339–348, 2000.

[7] G. Samorodnitsky and M. S. Taqqu, Stable Non-Gaussian Random Processes: Stochastic Models with Infinite Variance. Chapman and Hall/CRC, New York, 1994.

[8] D. Dua and C. Graff, "UCI Machine Learning Repository: Rice (Cammeo and Osmancik)," University of California, Irvine, School of Information and Computer Sciences, 2019. [Online]. Available: https://archive.ics.uci.edu/dataset/545/rice+cammeo+and+osmancik. Accessed: 2024-08-14.

[9] M. Hajian, "UCI Machine Learning Repository: Iranian Churn Dataset," University of California, Irvine, School of Information and Computer Sciences, 2021. [Online]. Available: https://archive.ics.uci.edu/dataset/563/iranian+churn+dataset. Accessed: 2024-08-14.

[10] D. Dua and C. Graff, "UCI Machine Learning Repository: Glass Identification Dataset," University of California, Irvine, School of Information and Computer Sciences, 2019. [Online]. Available: https://archive.ics.uci.edu/dataset/42/glass+identification. Accessed: 2024-08-15.

[11] Wikipedia contributors, ”Elliptical distribution,” Wikipedia, The Free Encyclopedia, 2024. [Online]. Available: https://en.wikipedia.org/wiki/Elliptical_distribution. Accessed: 2024- 08-14.
[12] Wikipedia contributors, ”Mixture model,” Wikipedia, The Free Encyclopedia, 2024. [Online]. Available: https://en.wikipedia.org/wiki/Mixture_model#Gaussian_mixture_model. Ac- cessed: 2024-08-14.
[13] J. P. Nolan, Stable Distributions - Models for Heavy Tailed Data. Birkha¨user, Basel, Switzerland, 2013.
[14] J. P. Nolan, ”Multivariate elliptically contoured stable distributions: theory and estimation,” Computational Statistics, vol. 28, no. 5, pp. 2067–2089, 2013. doi:10.1007/s00180-013-0396-7.
[15] T. Hastie, R. Tibshirani, and J. Friedman, The Elements of Statistical Learning: Data Mining, Inference, and Prediction. 2nd ed., Springer, New York, NY, USA, 2009.
[16] Müller, K. (2017). *Introduction to rpy2 and Python-R Integration*. Cambridge University Press.
[17] rpy2 Documentation. (2023). Retrieved from https://rpy2.github.io/doc/
[18] Urbanek, S. (2019). R-Python integration: Bridging the gap between statistical computing and general-purpose programming. *Journal of Statistical Software*, 90(2), 1–15.
[19] Manning, C. D., Raghavan, P., & Schütze, H. (2008). *Introduction to Information Retrieval*. Cambridge University Press.
[20] Domingos, P., & Pazzani, M. (1997). On the optimality of the simple Bayesian classifier under zero-one loss. *Machine Learning*, 29(2-3), 103-130.
[21] Dunn, J. C. (1974). Well-separated clusters and optimal fuzzy partitions. *Journal of Cybernetics*, 4(1), 95-104.
[22] Halkidi, M., Batistakis, Y., & Vazirgiannis, M. (2001). On clustering validation techniques. *Journal of Intelligent Information Systems*, 17(2), 107-145.
[23] Rand, W. M. (1971). Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical Association*, 66(336), 846-850.
[24] Hubert, L., & Arabie, P. (1985). Comparing partitions. *Journal of Classification*, 2(1), 193-218.
[25] Rousseeuw, P. J. (1987). Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. *Journal of Computational and Applied Mathematics*, 20, 53-65.
[26] Kaufman, L., & Rousseeuw, P. J. (2009). *Finding Groups in Data: An Introduction to Cluster Analysis*. Wiley.

APPENDICES

A.1. Code for Calling R Functions in Python

This appendix provides the Python code for defining, fitting, and calculating elliptical distributions using R functions via the rpy2 library.

```
import rpy2.robjects as robjects
from rpy2.robjects import numpy2ri
import numpy as np
import warnings
import os

def custom_rpy2_warn_handler(message, category, filename, lineno, file=None, line
    ↪ =None):
    pass

warnings.showwarning = custom_rpy2_warn_handler
os.environ["LANG"] = "en_US.UTF-8"
numpy2ri.activate()
```

A.2. Defining and Fitting Elliptical Mixture Models

The following functions define elliptical distributions and fit them to data using R's STABLE 5.3 package.

```
def mvstable_define_calling_R(alpha, delta, R):
    robjects.globalenv["alpha"] = alpha
    robjects.globalenv["delta"] = delta
    robjects.globalenv["R"] = R
    r_code = '''
    library(stable)
    ret <- mvstable.elliptical(alpha = alpha, delta = delta, R = R)
    '''
    robjects.r(r_code)
    return True

def mvstable_fit_calling_R(cov_matrix, method_num):
    robjects.globalenv["x"] = cov_matrix
    r_code = '''
    library(stable)
    ret <- mvstable.fit.elliptical(x, method1d = 1)
    '''
    robjects.r(r_code)
    alpha = robjects.r('ret["alpha"]')
    delta = robjects.r('ret["delta"]')
    R = robjects.r('ret["R"]')
    R = nearpd(R[0], corr=True)
    eigenvalues = robjects.r('ret["eigenvalues"]')
    normF = robjects.r('ret["normF"]')
    method1d = robjects.r('ret["method1d"]')
    return alpha[0], delta[0], R, eigenvalues[0], normF[0], method1d[0]

def nearpd(x, corr=True):
    robjects.globalenv["cov_matrix"] = x
    r_code = '''
    library("Matrix")
    ret <- nearPD(cov_matrix, corr = TRUE)
    '''
    robjects.r(r_code)
    mat = robjects.r('ret["mat"]')
```

A.3. Calculating the PDF

The dmvstable.elliptical() function calculates the PDF for elliptical distributions.

```
def dmvstable_elliptical_calling_R(val, alpha, delta, R):
    if val.ndim == 1:
        val = val[:, np.newaxis]
    robjects.globalenv["alpha"] = alpha
    robjects.globalenv["delta"] = delta
    robjects.globalenv["R"] = R
    robjects.globalenv["data_matrix"] = np.array(val)
    r_code = '''
    library(stable)
    probabilities <- apply(data_matrix, 1, function(x) {
        x_matrix <- matrix(x, nrow = 1, ncol = ncol(data_matrix))
        x_matrix_transposed <- t(matrix(x, nrow = 1, ncol = ncol(data_matrix)))
        dmvstable.elliptical(x = x_matrix_transposed, alpha = alpha, R = R, delta
            ↪ = delta)
    })
    '''
    robjects.r(r_code)
    probabilities = robjects.r('probabilities')
    probabilities = np.array(probabilities)
    return probabilities
```

A.4 E-step and M-step in the EM Algorithm

E-step: Calculating Posterior Probabilities In the E-step, we calculate the posterior probability of each data point belonging to each distribution.

```
class EllipticalMixture_self:
    def Estep(self):
        N = len(self.data)
        K = self.n_components
        weights = np.zeros((N, K))
        posterior_probs = np.zeros((N, K))

        for k, distribution in enumerate(self.distributions):
            for i, datum in enumerate(self.data):
                posterior_probs[i, k] = distribution.pdf(datum) * self.mix[k]

        total_probs = np.sum(posterior_probs, axis=1)
        total_probs[total_probs == 0] = 1e-9
        self.loglike = np.sum(np.log(total_probs))
        weights = posterior_probs / total_probs[:, None]

        self.labels = np.argmax(weights, axis=1)
        return weights
```

M-step: Updating Parameters The parameters for each distribution, along with the mixture weights, are recalculated and refined.

```
class EllipticalMixture_self:
    def Mstep(self, weights):
        N, D = self.data.shape
        K = self.n_components

        for k in range(K):
            weight_k = weights[:, k]
            total_weight = weight_k.sum()
            self.mix[k] = total_weight / N
            self.distributions[k].mu = np.sum(self.data * weight_k[:, np.newaxis],
                ↪ axis=0) / total_weight
            diff = self.data - self.distributions[k].mu
            self.distributions[k].weighted_cov = np.dot(weight_k * diff.T, diff) /
                ↪ total_weight + np.eye(D) * 1e-6

            fit_result = mvstable_fit_calling_R(self.distributions[k].weighted_cov,
                ↪  1)
            self.distributions[k].alpha = fit_result[0]
            self.distributions[k].delta = fit_result[1]
            self.distributions[k].R = fit_result[2]
```

The complete code and implementation details can be found at the link provided here: https://drive.google.com/file/d/12mNFhNoShCfVje9sAZXaVcqY0SwHRRuO/view?usp=sharing