Enhancing the Performance of PSO Algorithm for Clustering High dimensional data using Autoencoders

Shivalingappa Battur[1], Shrinidhi R H[2], Arya Kinagi[3], Nayana D G[4], Priya M[5], and S G Totad[6]

KLE Technological University, Hubli, Karnataka

Abstract. The emergence of big data has brought new challenges in processing and analyzing large and complex datasets due to their high dimensionality. Unsupervised learning techniques like clustering have become powerful tools for identifying patterns and relationships in data without the need for labelled examples. One popular Unsupervised data clustering technique is K-means and Particle Swarm Optimization(PSO). Combining both techniques' benefits, K-means clustering with PSO optimization can produce better clustering outcomes. The Elbow approach, which offers the K value for implementing K-means and PSO, automates the data clustering. Clustering high-dimensional data can be challenging due to the curse of dimensionality, where the number of dimensions dramatically outnumbers the number of data points. Therefore, a dimensionality reduction technique must be employed to enhance the performance of clustering high-dimensional data. Thus, we used Autoencoder as one of the dimensionality reduction techniques with K-means and PSO clustering and compared the clustering performance on reduced and original data. To implement this method, we used the Nifty 100 (1 min) stock market dataset from the Kaggle website.

Keywords: Big data, High-dimensional data, Unsupervised data, Big data Clustering, Feature Extraction, Autoencoders, Optimization of clustering, Elbow Method, Particle Swarm Optimization(PSO), K-means

1 Introduction

The modern world is driven by data, and the amount of data generated from various sources is increasing exponentially. As a result, the idea of big data has emerged, which refers to large and complex data sets that are becoming increasingly common in many industries. However, the emergence of big data has also brought new challenges, such as complexity and difficulties in processing and analyzing the data due to its size and high dimensionality. High-dimensional data, a type of big data containing numerous variables or features, poses a significant challenge in extracting useful information. To address these challenges, machine learning techniques like unsupervised learning have emerged, which can identify patterns and relationships in data without the need for labelled examples. By leveraging unsupervised learning, it is possible to handle high dimensionality and extract meaningful insights from big data.

In unsupervised learning, clustering is a commonly used technique where data points are grouped based on similarity[4]. In big data, clustering has become a powerful tool for identifying groups or clusters of similar data points, enabling researchers to identify patterns and gain insights into complex data sets. This technique has gained significant attention recently, mainly to analyze high-dimensional data. Numerous businesses, including healthcare, banking, and marketing, could gain from clustering. For instance, clustering can identify patient groups with similar medical conditions in healthcare, leading to more personalized treatments.

Similarly, in finance, clustering can help identify patterns in stock market data, such as clustering companies with similar stock movements. If a particular company's stock shows strong performance, the other companies within the same cluster will likely exhibit favourable results, leading to more accurate predictions. By leveraging unsupervised learning, it is possible to handle high dimensionality and extract meaningful insights from big data[8].

K-means is a well-liked clustering method that divides data points into K clusters based on how similar they are. [5]. This technique updates the closest cluster centroid based on the newly allocated data points by continually assigning data points to the centroid. [10].

Despite its many advantages, clustering has certain drawbacks, such as selecting the proper distance measures and figuring out the correct number of clusters. These challenges require careful consideration and attention to ensure the accuracy and reliability of clustering results[8]. However, while clustering can be an effective way to identify patterns in data, the process of clustering can be computationally intensive and time-consuming. Traditional clustering algorithms require optimization to ensure that they are effective in identifying meaningful clusters.

One such optimization technique is PSO, used successfully in clustering algorithms. PSO is a common optimization technique for data clustering. It draws inspiration from nature. The behavior of a particle swarm, each representing a potential solution, is modeled by PSO[23]. By optimizing the parameters of the clustering method, PSO contributes to the accuracy and effectiveness of the clustering process in Big data clustering. PSO, for instance, can asist in choosing the ideal cluster centroids and the proper distance metric for the clustering procedure. It helps overcome the drawbacks of conventional clustering methods, such as K-means, which may require assistance with huge data sets.

PSO and K-means combine the advantages of both techniques [2] by using PSO to enhance the initial K-means centroids [1]. PSO seeks the ideal set of starting centroids by maximizing a fitness function that considers the total distances between each data point and its closest centroid. Each data point is assigned to a cluster by K-means clustering based on the nearest centroid, and the centroids are updated after the best initial centroids have been found. K-means clustering has been demonstrated to perform better when using a hybrid strategy that combines PSO and K-means, especially when the data is highly dimensional.

However, employing the PSO algorithm in a high-dimensional search space might be difficult since getting bogged down in a local optimum is simple rather than the global one. Furthermore, the PSO iterative process might be slow, so it might take a while to discover a decent solution.

There are several ways to manage high-dimensional data and overcome the above problem, which is advantageous for clustering. One such technique is Feature extraction, which selects or transforms the input data to a lower-dimensional space that preserves the essential information, which is particularly useful for big data sets where the dimensionality of the data can be very high. The need for Feature extraction in Big data arises because a high number of features can lead to a high degree of redundancy and noise in the data, making it challenging to analyze and extract valuable insights.

Principal Component Analysis (PCA), Linear Discriminant Analysis (LDA), and Autoencoders are a few feature extraction techniques. Autoencoders are neural networks that can extract features from input data and detect anomalies by learning a compressed version. Autoencoders are particularly useful for high-dimensional data, where traditional clustering algorithms may need to be more effective. The need for Autoencoders arises because high-dimensional data can be challenging to analyze and extract insights due to its size and complexity[29]. Autoencoders can decrease the dimensionality of data while retaining crucial information, simplifying the task of processing and analyzing the data.

Finding the optimal configuration of data points into clusters that maximize the distance between the clusters while minimizing the distance between points within each cluster is the aim of PSO clustering. The algorithm must determine how many clusters to look for to accomplish this. When determining the number of clusters, methods like the Elbow technique can estimate the ideal number of clusters. The elbow strategy can be used to calculate the appropriate number of clusters when utilizing PSO with K-means, which eventually increases the precision of the clustering results.

This Paper proposes a method to optimize the performance of the PSO Algorithm for Clustering High-dimensional data employing Autoencoders. We have mainly focused on the Stock Market dataset, Nifty-100 1-min Stock Market data, from Kaggle. The aim is initially to design and develop a PSO algorithm for automatic data clustering and then to develop PSO employing Autoencoder for data clustering. Finally, compare the performance of the former and later using different validity indices, which will be discussed in other sections. We will be inferring new insights into clustered data of the Stock Market. Spark, is the platform used for parallel and Distributed computing

This paper is organized as follows, *Section 2* explains the Literature Survey, followed by Background Study of the techniques used in the paper in *Section3*, Proposed Methodology in *Section4*, *Section 5* shows the results, *Section 6* Summarizes the whole Paper, and the last section contains all the References.

2 Literature Survey

The authors of [3] have presented a unique clustering algorithm that combines K-Means and PSO methods . The suggested algorithm, IKPSO, seeks to get around K-Means' drawbacks, including its sensitivity to initial centroid positions and propensity to become stuck in local optima. In the initialization step of IKPSO, the dataset is divided into K clusters using K-Means, and the initial centroids are established. These centroids are then used to initialize the particles in the PSO algorithm. PSO is then utilized to update the particles and increase the clustering accuracy as the optimization phase begins. The optimal initial centroids for K-means clustering are determined by PSO using a fitness function that considers the separation between data points and the cluster centroids. The IKPSO algorithm's performance is evaluated on various benchmark datasets. The results demonstrate its performance compared to clustering algorithms, such as K-Means, and PSO-based clustering algorithms. The algorithm's strengths include

overcoming the initialization sensitivity and local optima problems of K-Means and improving clustering accuracy and efficiency. The results show that IKPSO outperforms the PSO algorithm regarding the results obtained and the number of iterations needed. In addition, compared to PSO, IKPSO averages 52.38% faster problem-solving in normalized datasets and 52.30% faster in original datasets. These results encourage us to use PSO and Kmeans in our clustering.

The authors of [18] have compared their proposed method, which uses convolutional autoencoder-based feature extraction and clustering, to a clustering method based on K-means. They explained that K-means have limitations that make them less effective for clustering, such as requiring manual feature engineering, struggling with high-dimensional data, and being sensitive to initialization. On the other hand, the authors' proposed approach uses convolutional autoencoders to automatically learn high-level features from raw load data, eliminating the need for manual feature engineering and addressing the limitations of K-means. The authors showed that their proposed approach outperformed the K-means methods regarding clustering accuracy and stability. Thus, the authors concluded that their proposed approach based on convolutional autoencoders is superior to K-means for clustering because it can achieve accurate and stable clustering without manual feature engineering or initialization. This paper gave us a solution to overcome the curse of high dimensionality and encouraged us to use a feature extraction technique, in our case Autoencoder to get the desired results.

Yaniv Opochinsky et al. [16] have proposed a novel approach to unsupervised deep clustering using K-autoencoders. The paper discusses the limitations of existing methods, such as the inability to handle complex high-dimensional data and the reliance on manual feature engineering. The authors employ K-autoencoders to learn the feature representation and clustering assignments end-to-end concurrently to get around these restrictions. The suggested technique performs at the cutting edge on several benchmark datasets, including text and image datasets. The design of the K-autoencoder and the optimization technique used to optimize the feature representation and clustering assignments jointly are described in length by the authors as other aspects of the suggested approach. Additionally, they offer a visualization of the assigned clustering and learned feature representation, enabling comprehension of the underlying data structure. The paper concludes that the proposed approach is effective and interpretable and suggests that future work could extend the proposed system to handle other types of data, such as time series data, or incorporate additional constraints or objectives to improve performance further.

Choosing the ideal number of clusters for the k-means clustering method is discussed by the authors of [15]. To do this, they combine the elbow approach with purity evaluation on a dataset of headline news. The elbow method is widely used for estimating the optimal number of clusters by plotting the percentage of variance explained as a function of the number of clusters and searching for an "elbow" point on the curve. The quality of the clusters generated by the k-means approach is evaluated by the authors using a purity evaluation. They use a dataset of news headlines to test their strategy, and the findings show that six clusters are the ideal number. They also demonstrate that their approach beats existing approaches, such as the silhouette method and gap statistic, for figuring out the perfect number of clusters. The authors claim that their method, mainly used to cluster text data, offers a practical and efficient way to establish the ideal number of clusters for the k-means algorithm.

Authors of [28] compare the performance of different validity indices on clustering results obtained from four swarm-intelligence-based clustering algorithms mentioned in their paper. The Silhouette Coefficient and Calinski-Harabasz Index performed well across all algorithms and datasets, while the Davies-Bouldin Index and Dunn Index performed well on some algorithms and datasets but not on others. The Silhouette Coefficient calculates how closely two clusters resemble one another and how far apart they are from one another. The Silhouette Coefficient is high when the clusters are well separated, and the data points inside each cluster are comparable. On some algorithms and datasets, the DB Index—which calculates the average similarity between each cluster and its most similar cluster—performed admirably. The research offers insightful recommendations for choosing suitable validity indices for assessing clustering outcomes from swarm-intelligence-based algorithms.

3 Background Study

3.1 Autoencoders

Autoencoders refer to a specific kind of neural network that can perform unsupervised learning tasks, including data compression, feature extraction, and data generation.
An autoencoder comprises two main components: an encoder and a decoder. The encoder's role is to reduce the dimensionality of the input by compressing it, while the decoder's job is to take this compressed representation and reconstruct the original input. Minimizing the difference between the original input

and the reconstructed output is the goal of an autoencoder. A loss function like mean squared error is typically used to do this. [3]

Mathematically, the encoder function can be represented as:

$$h = f(x) = \sigma(Wx + b) \tag{1}$$

where σ , W, b, and h represent the weight matrix, bias vector, hidden representation, and activation function, respectively.

The decoder function can be represented as:

$$y = g(h) = \sigma(W'h + b') \tag{2}$$

where W' is the transpose of W, b' is the decoder's bias vector, and y is the output of the reconstruction.

An autoencoder's goal is to reduce the difference between input data x and the output data y, which is often measured by a loss function like mean squared error:

$$L(x, y) = ||x - y||^2 \tag{3}$$

where ||.|| denotes the Euclidean distance.
The autoencoder is trained by minimizing the loss function using backpropagation and gradient descent. The loss gradients with respect to the weights and biases are computed and used to update the parameters in the network. Once the autoencoder is trained, it can be used for various tasks like data compression, feature extraction, and data generation.

3.2 K-means and PSO

K-means algorithm converges faster than PSO, but it can have lower accuracy if the initial cluster centres are poorly chosen. One reason for the speed of K-means is that it updates cluster centres by taking the mean point of cluster members. This advantage can be used to improve the PSO algorithm's convergence rate. This algorithm makes a small modification to the PSO algorithm. Once the positions and velocities of each particle are updated, the algorithm checks if the new position is an improvement over the best-observed position. If it is, the cluster centre is replaced with the new position. The algorithm involves the following steps:

If the new position is superior to the best-observed position, the cluster center is replaced after updating each particle's position and speed.[3] The algorithm's steps are listed below.:

1. Generate N particles at random.
2. Formulas (4) and (5), are used to update each particle's position and velocity respectively.
3. If the new position results in a better objective function value than the current position, replace the pi with the mean point of the cluster members.
4. If the new position is superior to the one that has been best observed, replace the cluster centre with it..
5. Till the stopping requirement is fulfilled, repeat steps 3 and 4 as necessary.

$$\begin{aligned} V_i^{New} = w * V_i^{old} + c_1 * r_1 * (pi - x_i^{old} + c_2 \\ *r_2 * (p_g - V_i^{old}) \end{aligned} \tag{4}$$

Where the particle best position is represented as p_i and the global best position is represented as p_g The algorithm uses two positive constants, c_1 and c_2 as well as a randomly generated inertia weight, w The range of the uniformly generated random numbers r_1 and r_2 is between 0 and 1.

$$x_i^{New} = x_i^{old} + V_i^{New} \tag{5}$$

It is important to note that this process is only applied to some new positions to avoid repetitive calculations. Additionally, the K-Means algorithm is not used independently but incorporated into the PSO algorithm. Our clustering algorithm employs an objective function similar to that used in K-Means. This function uses the Euclidean distance to calculate the sum of distances between all samples and cluster centers. The full range of the dimension can likewise be used to normalize feature vectors.

3.3 Elbow Method

The elbow approach is a well-liked way of figuring out how many clusters a clustering algorithm should use. The method's name comes from the elbow-shaped graph that plots the number of clusters against the related variance. Before using the elbow approach, one must first determine the sum of squared distances between each data point and the designated cluster centroid. This value is then plotted against the number of clusters, and the graph is examined to identify a distinct point at which the reduction in the sum of squared distances begins to level off, forming an elbow-like bend in the graph. [15] There are two metrics Distortion and inertia.
Distortion(D): Distortion in K-means clustering is calculated by taking Each cluster's average squared distance from its cluster center is used. Typically, this measurement is done using the Euclidean distance.

$$D = \sum_{k=1}^{K} \sum_{i=1}^{N_k} ||x_i - c||^2 \tag{6}$$

Where K is number of clusters, N_k is number of data points in cluster K.
Inertia(I): The total of the squared distances from each sample to the nearest cluster centre is the inertia.

$$I = \sum d(x, c)^2 \tag{7}$$

Where d(x, c) represents the distance between a data point x and its closest cluster center c. The sum is taken across all data points in the dataset.

4 Proposed Methodology

This section describes the dataset and flow of our proposed work as shown in *Fig1*

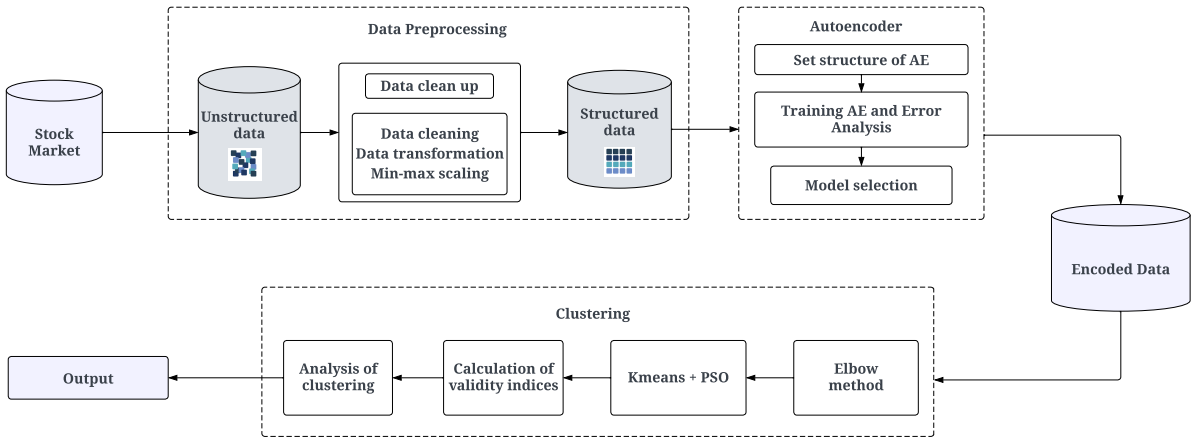


Fig. 1. Proposed Work Flow

4.1 Data Description and Preprocessing

We obtained data from Kaggle, which contains historical daily prices for Nifty 100 stocks and indices traded on the Indian Stock Market. The dataset includes 1-minute interval data from February 2015 to October 2022, including Open, High, Low, Close, and Volume(OHLCV) data for each stock and other 55 technical indicators. "High" refers to the traded stock's highest daily price. The price at which the stock began trading when the market opened is called the "open." "low" refers to the stock's daily low trading price. The price at which the stock ceased trading when the market ended is called "close." The total number of shares of a stock that have been exchanged within a given period, typically a day, is referred to as "volume" in the stock market. Initially, the dataset is large, weighing around 33 GB, and covers 100 Nifty stocks from different sectors. Each CSV file has 60 columns(dimensions) and row entries from

Table 1. Values of DB and Silhouette Indices for the selection of Number of Columns

Datasets/ No. of Columns	High		Low		Open		Close		Volume	
	DB Index	Silhouette Index	DB Index	Silhouette index	DB Index	Silhouette Index	DB Index	Silhouette Index	DB Index	Silhouette Index
45	0.49993	0.57783	0.499887	0.5646	0.49989	0.55962	0.49978	0.55937	0.49956	0.572968
50	0.4999	0.5488	0.50059	0.50061	0.49974	0.53757	0.499988	0.56460	0.499964	0.50888
55	0.4997	0.52807	0.499967	0.57815	0.49958	0.56578	0.49978	0.57685	0.499566	0.572968
65	0.49995	0.55356	0.499887	0.5646	0.49989	0.55962	0.49978	0.55937	0.49991	0.57437

Table 2. Values of DB and Silhouette Indices for the selection of Number of particles

Datasets/ No. of particles	High		Low		Open		Close		Volume	
	DB Index	Silhouette Index	DB Index	Silhouette Index	DB Index	Silhouette Index	DB Index	Silhouette Index	DB Index	Silhouette Index
3	0.49977	0.57686	0.49995	0.54722	0.49975	0.48253	0.49991	0.56626	0.49989	0.54304
4	0.49978	0.57888	0.49990	0.6193	0.49958	0.56576	0.49977	0.56796	0.49980	0.52239
5	0.49990	0.619313	0.49952	0.54921	0.49955	0.57326	0.4995	0.56872	0.49956	0.42683
6	0.4993	0.57782	0.49952	0.54921	0.49956	0.57591	0.49979	0.55936	0.49999	0.52200

February 2015 to October 2022. These 60 indicators are the columns and timestamps from 2015 Feb to 2022 Oct, with data collected daily from 9:15 am to 7:14 pm.

We made some modifications to the dataset to fit our requirements. Although we had data from February 2015 to September 2022, we decided to only use the data for September and October 2022 for our work. We deleted all the other entries. Furthermore, we separated the columns into separate CSV files for Open, High, Low, Close, and Volume. Each row contains the name of the stock and the columns contain time stamps for each minute between 9:15 am and 4:14 pm for each day between September and October 2022. Finally, our newly modified Dataset contains 101 rows and 13,561 columns, weighing 6.1KB.

After cleaning the data, it is common practice to normalize the data for optimal training in deep learning. Normalization is achieved using various techniques such as Z-normalization or min-max normalization. In our proposed work, we have opted for min-max normalization, which scales the data between 0 and 1.

4.2 Implementation

Once the cleaned data is obtained, it is fed into the autoencoder, where we set the structure of the autoencoder to reduce the dimensions from 13,561 to the optimal value obtained through the trial and error method. We emerged to these dimensions such that they should have maximum accuracy, indicating the similarity of the compressed data with the original one and lesser dimensions; as shown in *Table 1*, we need to select one particular value for a number of columns from the table where the DB index value is lesser and the silhouette index value is greater (the Range of DB and Siloutte is formed from -1 to 1). Similarly, as shown in *Table 2*, the selection of number of particles is also made by trial and error, where the DB index value is lesser and the Silhouette index value is greater, and finally, an optimal value for a number of particles is obtained.

The encoded data from the autoencoder is fed to the elbow method. Here, a list of K values is created, and for each k value, the k-mean is run to find the deals of inertia and distortion, graphs in *Fig2* and *Fig3*, give the optimum K value. These graphs are plotted against distortion and inertia. We need to select one optimal elbow point value from both graphs. As we can see from our data points, we are arriving at the optimal k value of 3. This K value is used in K-means and PSO, where a number of particles and iterations are set to obtain the best clustering result, measured by silhouette and DB validity indices. The lower the DB index value, the higher the silhouette value within the range better the clustering results. This clustering result can help make further analysis. As shown in *Table 1*, the DB and silhouette indices values of K-means and PSO with Autoencoders are less and high, respectively compared to others, resulting in better performance.

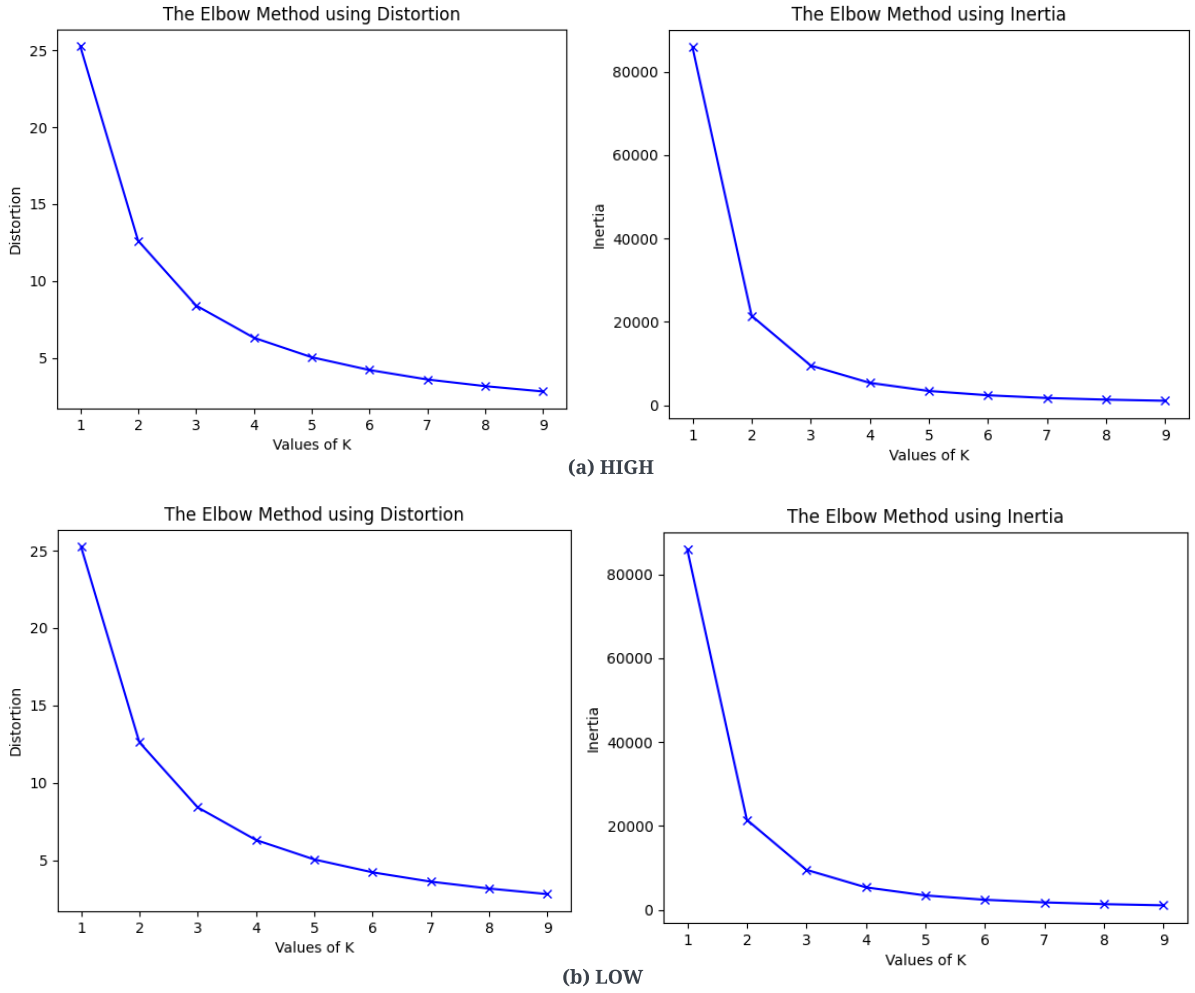


Fig. 2. Graph of Values of Inertia and Distortion for High and Low Features

5 Results

This paper proposes work to enhance the performance of PSO on high-dimensional data using autoencoders. Firstly, we fed our high-dimensional stock market data into autoencoders, reducing our data's size. As shown above, *Table 1* and *Table 2* obtain values through the trial and error method. From those tables, we emerged at the optimal value. This is accomplished by contrasting the values of the DB and Silhouette indices. For instance, consider the high feature, where we obtain lower DB and higher silhouette indices for 45 columns, respectively, and where the best DB and silhouette indices value is obtained for 5 particles. Similarly, the number of particles for OHCLV turns out to be 6,5,5,4,3 respectively and the optimal number of clusters for OHCLV comes out to be 55,45,55,55,65 respectively. Once the number of columns is reduced from 13561 to the optimal value, this reduced data is fed into the elbow method to find K-value for automatic data clustering. Using this K value, K-means and PSO are implemented. From the *Table 3*, we can infer that performance of K-means and PSO with autoencoder is better than K-means and PSO. We can arrive to this conclusion by witnessing the DB and Silhouette Indices comparision as dicussed in the *Section4.2* From *Table 4*, illustrates the execution time of PSO and K-means with and without autoencoders, which compares the performance and proves that our proposed methodology gives appropriate and desired results effectively.

Fig5 and *Fig6* are the graphical representation of high, low, close, and open and volume through autoencoder, dimensions are reduced to 3. Here we can visualize that each color indicates a different cluster, and each cluster contains various stocks with similarities.

Stocks are clustered based on their performance in 2 monthly periods, which helps investors identify patterns or similarities. Investors can know which stocks to invest in or avoid based on the clusters to which the stocks belong. If a particular stock performs well, the other stocks within the cluster will exhibit favourable results. As we can see in *Table 5*, which is a sample data points table, let's take the

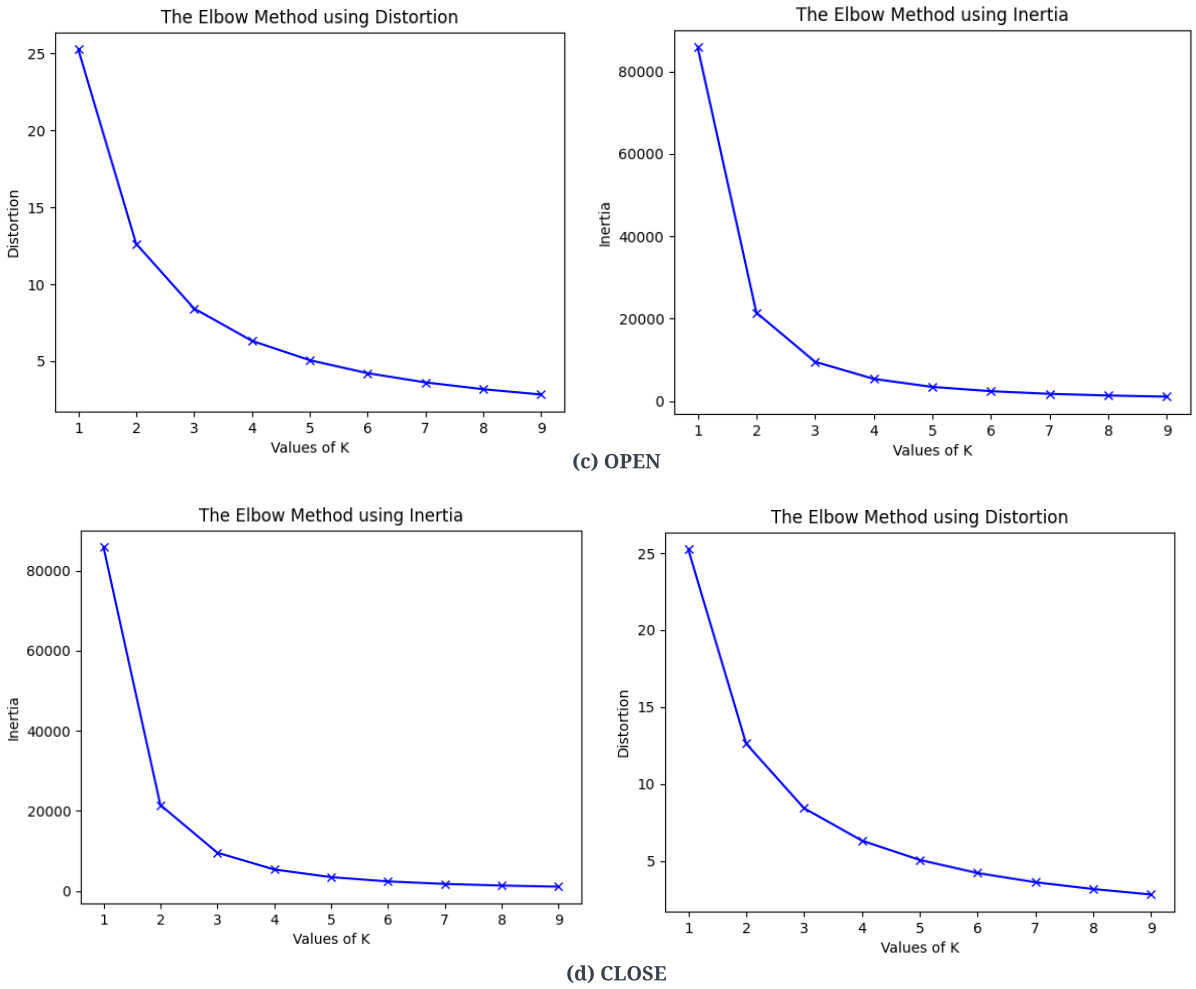


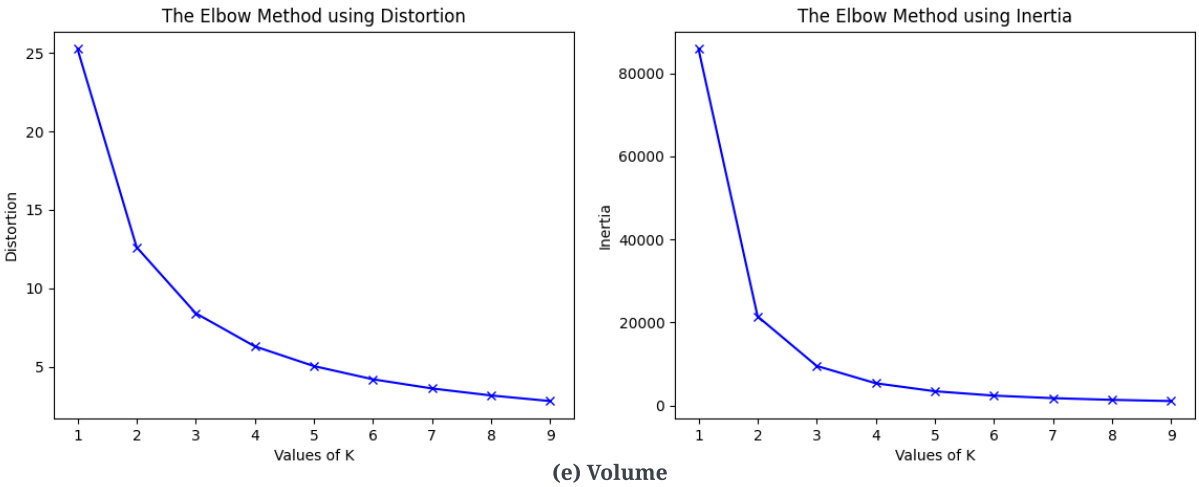
Fig. 3. Graph of Values of Inertia and Distortion for Open, Close and Volume Features

example of Adaniports, we can witness here that along with Adaniports, ACC Cement, Bajaj, Auto, AxisBank etc., belong to the same cluster for all the OHCLV. Similarly in the case of Cipla stocks, as shown in *Table 6*, falling in health sector we can witness that CoalIndia, COLPAL, DABUR, DIVISLAB, DLF etc., belong to the same cluster for all the OHCLV so can infer that this stock performs similarly under all the circumstances. Similarly, if a specific stock performs poorly, the different supplies within the same cluster will probably display unfavourable results. As explained in the previous points, Stocks such as Adaniports, Adaninet, Adanigreen, and AxisBank perform similarly. If Adaninet stock is performing poorly, it's wise not to invest in other stocks, such as Adaniports, Adanigreen etc., which belong to the same cluster.

It can also be used to research patterns across many industries. Some possible sectors include health care, consumer products, Technology, finance, industry, energy, utilities, and materials.

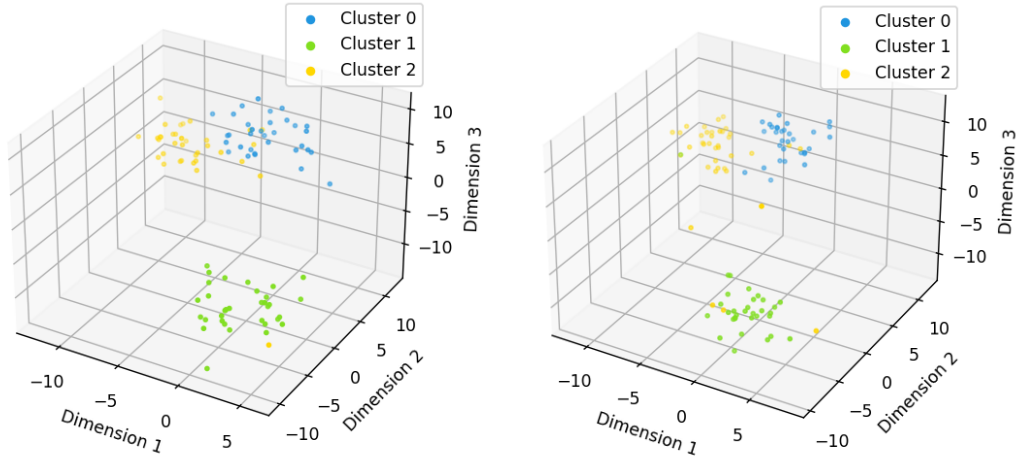
Table 3. DB and Silhouette indices comparision

Datasets/ Methods	K-Means and PSO		K-means and PSO with Autoencoders	
	DB Index	Silhouette Index	DB Index	Silhouette Index
High	**0.993166**	**0.044056**	**0.499879**	**0.598376**
Low	**0.98635**	**0.079333**	**0.492837**	**0.694484**
Close	**0.98474**	**0.046373**	**0.474543**	**0.634368**
Open	**0.93643**	**0.056383**	**0.547732**	**0.745483**
Volume	**0.99736**	**0.043367**	**0.498746**	**0.648464**



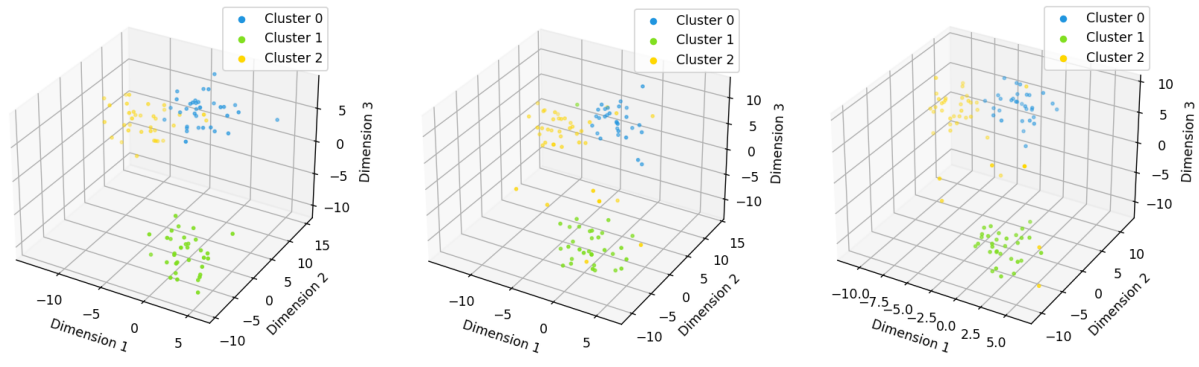
(e) Volume

Fig. 4. Graph of Values of Inertia and Distortion for Open, Close and Volume Features



(d) High (e) Close

Fig. 5. Cluster Visualization for High and Close Datasets



(a) Volume (b) Open (c) Low

Fig. 6. Cluster Visualization for Open, Low and Volume Datasets

If a group of businesses within a sector is doing well, it may indicate a growing trend in that sub-sector. On the other hand, if a group of companies within a sector exhibit underperformance, it can suggest that that sub-sector is experiencing a downward trend.

The number of stocks in each sector belonging to the three clusters is shown in the *Fig7* and *Fig8*. If a cluster indicates successful performance, then sectors with a higher proportion of stocks in that

Table 4. Execution time(in secs) of each dataset run on each algorithm

Datasets/ Algorithm	Kmeans and PSO with AE	Kmeans and PSO without AE	AE
High	**1.0491**	**118.0741**	**88.7339**
Low	**0.9157**	**110.1707**	**89.0774**
Open	**1.1358**	**116.7213**	**88.6308**
Close	**1.1427**	**108.7567**	**81.3436**
Volume	**1.1183**	**109.8830**	**88.9479**

cluster represent successful sectors. For instance, with volume statistics, a sector has more stock purchase and sell movements if a significant number of its stocks are clustered with better-performing equities. It can also be used to examine how various sectors are related and how they depend on one another. The two industries have similar trends if they have the same distribution of stocks within each cluster For example, consider to volume statistics, the distribution of stocks within each cluster in the industrial and technological sectors is nearly the same, as evidenced by the *Fig8*. Thus, there are trends in these two industries.

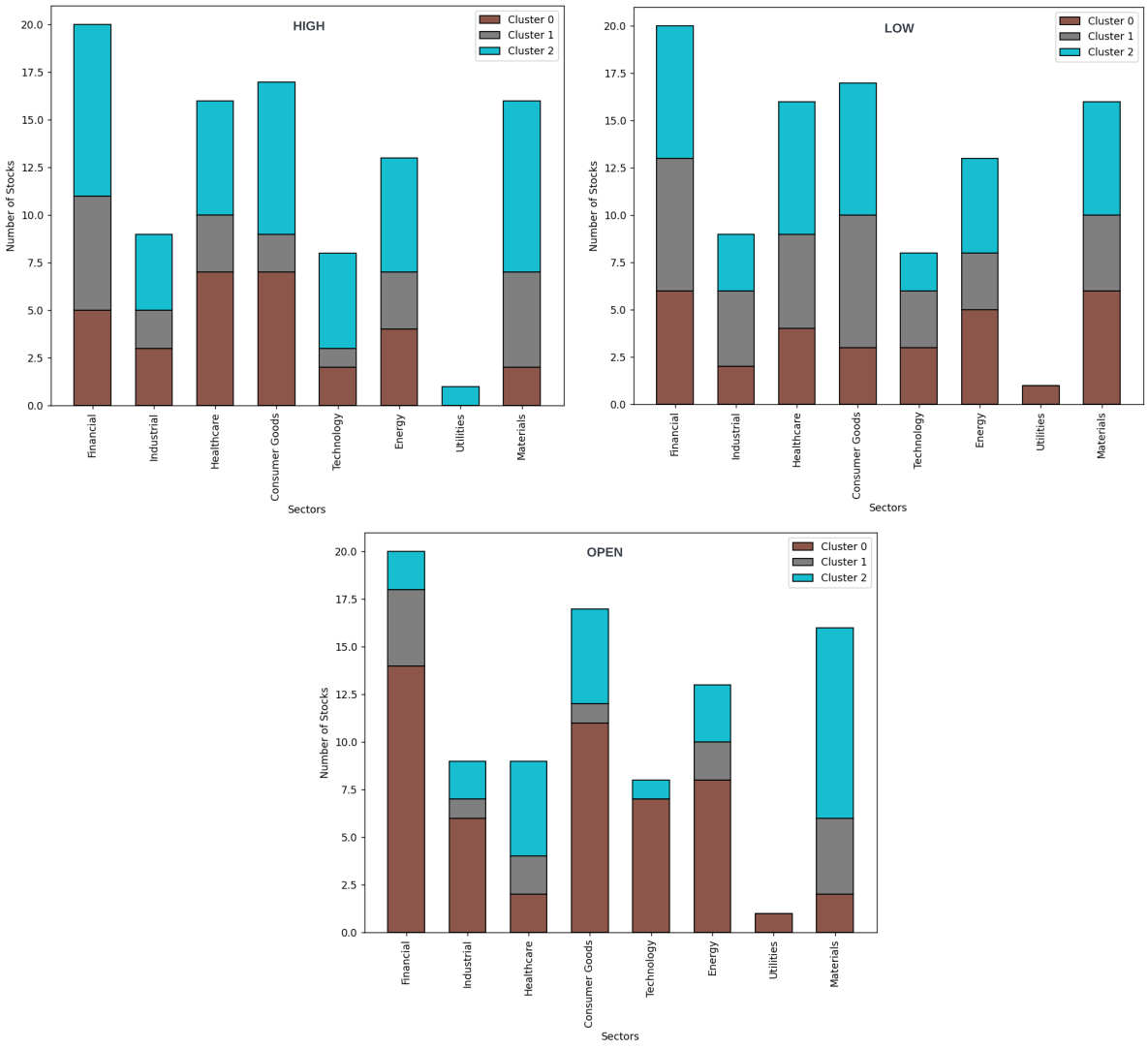


Fig. 7. Sector wise clusters of Stocks in High, low and Open Datasets

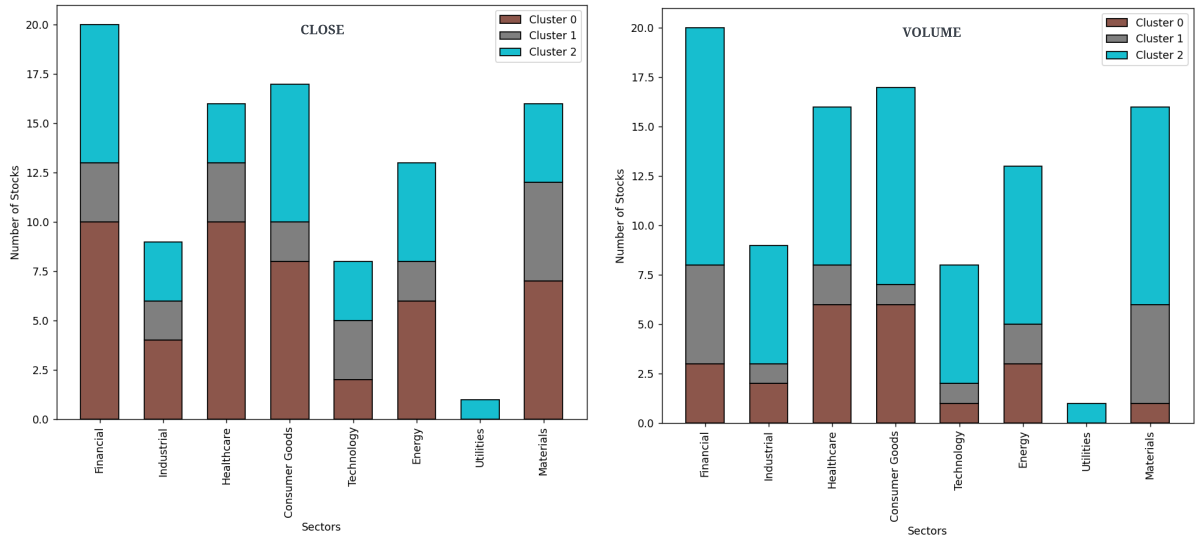


Fig. 8. Sector wise cluster of Stocks in Close and Volume Datasets

Another illustration would be the scenario of open data, where the distribution of stocks within each cluster in the industrial and energy sectors is nearly identical. So, the patterns in these two industries are comparable.

It is essential to keep in mind that the above inferences are not the only factors to decide if a stock can be invested or not. Other factors including news about individual companies, market conditions, and economic data, may also influence the stock behavior. As a result, it's crucial to conduct thorough research and consider various data points before making any investment decisions that are merely based on the stock distribution across clusters.

Table 5. Stocks in the same cluster as ADANIPORTS

Stocks	HIGH	LOW	OPEN	CLOSE	VOLUME
ACC	**1**	**2**	**1**	**0**	**1**
ADANIENT	**1**	**2**	**1**	**0**	**1**
ADANIGREEN	**1**	**2**	**1**	**0**	**1**
ADANIPORTS	**1**	**2**	**1**	**0**	**1**
APOLLOHOSP	**1**	**2**	**1**	**0**	**1**
AUROPHARMA	**1**	**2**	**1**	**0**	**1**
AXISBANK	**1**	**2**	**1**	**0**	**1**
AMBUJACEM	**1**	**2**	**1**	**0**	**1**
ASIANPAINT	**1**	**2**	**1**	**0**	**1**
BAJAJ-AUTO	**1**	**2**	**1**	**0**	**1**
BAJAJFINSV	**1**	**2**	**1**	**0**	**1**
BAJAJHLDNG	**1**	**2**	**1**	**0**	**1**

6 Conclusion and Future Scope

With the proliferation of big data in various domains, clustering has become a popular method for uncovering patterns and structures in large datasets. One such clustering technique is K-means. To overcome several optimization issues of clustering, PSO algorithm is one among the widely used techniques. PSO and K-means algorithm has a major drawback, that is the algorithm's performance tends to deteriorate as the number of dimensions increases. High-dimensional data clustering is a challenging problem that calls for advanced methods and algorithms. To improve clustering outcomes on high dimensional data, we have implemented a novel method that combines PSO and K-means clustering with autoencoders. To employ this, we used the Nifty 100 (1 min) stock market dataset from the Kaggle website. It has

Table 6. Stocks in the same cluster as CIPLA

Stocks	High	Low	Open	Close	Volume
CIPLA	**0**	**2**	**2**	**0**	**0**
COALINDIA	**0**	**2**	**2**	**0**	**0**
COLPAL	**0**	**2**	**2**	**0**	**0**
DABUR	**0**	**2**	**2**	**0**	**0**
DIVISLAB	**0**	**2**	**2**	**0**	**0**
DLF	**0**	**2**	**2**	**0**	**0**
DMART	**0**	**2**	**2**	**0**	**0**
DRREADY	**0**	**2**	**2**	**0**	**0**
EICHERMOT	**0**	**2**	**2**	**0**	**0**
GAIL	**0**	**2**	**2**	**0**	**0**
GLAND	**0**	**2**	**2**	**0**	**0**
GODREJCP	**0**	**2**	**2**	**0**	**0**
GRASIM	**0**	**2**	**2**	**0**	**0**

been demonstrated that our method successfully lessens the effects of the curse of dimensionality. We were able to minimize the dimensionality of the data while maintaining the crucial features by applying autoencoders and PSO to optimize the clustering parameters. Our study demonstrates that, when applied to benchmark datasets, our method outperforms other ones already in use.

Investigating different Autoencoder variants for dimensionality reduction is one possible future direction. Applying this PSO and K-means algorithm to other, larger datasets is a different future approach that may be taken. It can be fascinating to apply the suggested method to practical applications in many industries, such as healthcare, banking, and image and video processing.

References

1. Alireza Ahmadyfard and Hamidreza Modares. Combining pso and k-means to enhance data clustering. In *2008 International Symposium on Telecommunications*, pages 688–691. IEEE, 2008.
2. Habibollah Agh Atabay, Mohammad Javad Sheikhzadeh, and Mehdi Torshizi. A clustering algorithm based on integration of k-means and pso. In *2016 1st Conference on Swarm Intelligence and Evolutionary Computation (CSIEC)*, pages 59–63. IEEE, 2016.
3. Habibollah Agh Atabay, Mohammad Javad Sheikhzadeh, and Mehdi Torshizi. A clustering algorithm based on integration of k-means and pso. In *2016 1st Conference on Swarm Intelligence and Evolutionary Computation (CSIEC)*, pages 59–63. IEEE, 2016.
4. Kamalpreet Bindra and Anuranjan Mishra. A detailed study of clustering algorithms. In *2017 6th international conference on reliability, infocom technologies and optimization (trends and future directions)(ICRITO)*, pages 371–376. IEEE, 2017.
5. Renato Cordeiro de Amorim. A survey on feature weighting based k-means algorithms. *Journal of Classification*, 33:210–242, 2016.
6. Quentin Fournier and Daniel Aloise. Empirical comparison between autoencoders and traditional dimensionality reduction methods. In *2019 IEEE Second International Conference on Artificial Intelligence and Knowledge Engineering (AIKE)*, pages 211–214. IEEE, 2019.
7. Ahmed G Gad. Particle swarm optimization algorithm and its applications: a systematic review. *Archives of computational methods in engineering*, 29(5):2531–2561, 2022.
8. Er Arpit Gupta, Er Ankit Gupta, and Er Amit Mishra. Research paper on cluster techniques of data variations. *International Journal of Advance Technology & Engineering Research*, 1(1):39–47, 2011.
9. Hestry Humaira and R Rasyidah. Determining the appropiate cluster number using elbow method for k-means algorithm. In *Proceedings of the 2nd Workshop on Multidisciplinary and Applications (WMA) 2018, 24-25 January 2018, Padang, Indonesia*, 2020.
10. Abdellah Idrissi, Hajar Rehioui, Abdelquoddouss Laghrissi, and Sara Retal. An improvement of denclue algorithm for the data clustering. In *2015 5th International Conference on Information & Communication Technology and Accessibility (ICTA)*, pages 1–6. IEEE, 2015.
11. Muhammad Imran, Rathiah Hashim, and Noor Elaiza Abd Khalid. An overview of particle swarm optimization variants. *Procedia Engineering*, 53:491–496, 2013.
12. Iain M Johnstone and D Michael Titterington. Statistical challenges of high-dimensional data, 2009.
13. J. Kennedy and R. Eberhart. Particle swarm optimization. In *Proceedings of ICNN'95 - International Conference on Neural Networks*, volume 4, pages 1942–1948 vol.4, 1995.
14. Jiamu Li, Ji Zhang, Mohamed Jaward Bah, Jian Wang, Youwen Zhu, Gaoming Yang, Lingling Li, and Kexin Zhang. An auto-encoder with genetic algorithm for high dimensional data: Towards accurate and interpretable outlier detection. *Algorithms*, 15(11):429, 2022.

15. Dhendra Marutho, Sunarna Hendra Handaka, Ekaprana Wijaya, et al. The determination of cluster number at k-mean using elbow method and purity evaluation on headline news. In *2018 international seminar on application for technology of information and communication*, pages 533–538. IEEE, 2018.
16. Yaniv Opochinsky, Shlomo E Chazan, Sharon Gannot, and Jacob Goldberger. K-autoencoders deep clustering. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4037–4041. IEEE, 2020.
17. S Prasanna and D Ezhilmaran. Stock market prediction using clustering with meta-heuristic approaches. *Gazi University Journal of Science*, 28(3):395–403, 2015.
18. Seunghyoung Ryu, Hyungeun Choi, Hyoseop Lee, and Hongseok Kim. Convolutional autoencoder based feature extraction and clustering for customer load analysis. *IEEE Transactions on Power Systems*, 35(2):1048–1060, 2019.
19. Saad Sadiq, Nicolas Wagner, Mei-Ling Shyu, and Daniel Feaster. High dimensional latent space variational autoencoders for fake news detection. In *2019 IEEE Conference on Multimedia Information Processing and Retrieval (MIPR)*, pages 437–442. IEEE, 2019.
20. Saptarshi Sengupta, Sanchita Basak, and Richard Alan Peters. Particle swarm optimization: A survey of historical and recent developments with hybridization perspectives. *Machine Learning and Knowledge Extraction*, 1(1):157–191, 2018.
21. Congming Shi, Bingtao Wei, Shoulin Wei, Wen Wang, Hai Liu, and Jialei Liu. A quantitative discriminant method of elbow point for the optimal number of clusters in clustering algorithm. *EURASIP Journal on Wireless Communications and Networking*, 2021(1):1–16, 2021.
22. Yuhui Shi et al. Particle swarm optimization: developments, applications and resources. In *Proceedings of the 2001 congress on evolutionary computation (IEEE Cat. No. 01TH8546)*, volume 1, pages 81–86. IEEE, 2001.
23. Mei-Ping Song and Guo-Chang Gu. Research on particle swarm optimization: a review. In *Proceedings of 2004 international conference on machine learning and cybernetics (IEEE Cat. No. 04EX826)*, volume 4, pages 2236–2241. IEEE, 2004.
24. Alexander Strehl and Joydeep Ghosh. Relationship-based clustering and visualization for high-dimensional data mining. *INFORMS Journal on Computing*, 15(2):208–230, 2003.
25. MA Syakur, BK Khotimah, EMS Rochman, and Budi Dwi Satoto. Integration k-means clustering method and elbow method for identification of the best customer profile cluster. In *IOP conference series: materials science and engineering*, volume 336, page 012017. IOP Publishing, 2018.
26. Edy Umargono, Jatmiko Endro Suseno, and SK Vincensius Gunawan. K-means clustering optimization using the elbow method and early centroid determination based on mean and median formula. In *The 2nd International Seminar on Science and Technology (ISSTEC 2019)*, pages 121–129. Atlantis Press, 2020.
27. Chathurika S Wickramasinghe, Daniel L Marino, and Milos Manic. Resnet autoencoders for unsupervised feature learning from high-dimensional data: Deep models resistant to performance degradation. *IEEE Access*, 9:40511–40520, 2021.
28. Rui Xu, Jie Xu, and Donald C. Wunsch. A comparison study of validity indices on swarm-intelligence-based clustering. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 42(4):1243–1256, 2012.
29. Jiang Zhu, Lingda Wu, Hongxing Hao, Xiaorui Song, and Yi Lu. Auto-encoder based for high spectral dimensional data classification and visualization. In *2017 IEEE Second International Conference on Data Science in Cyberspace (DSC)*, pages 350–354. IEEE, 2017.