Enhanced K-means Clustering

A Project

Submitted in partial-fulfilment of the

requirement for the award of the degree of

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE & ENGINEERING

BY

Sudhanshu Kumar 20050445046

DEPARTMENT of COMPUTER SCIENCE & ENGINEERING

CAMBRIDGE INSTITUTE OF TECHNOLOGY

TATISILWAI, RANCHI – 835103.

(2024)

DECLARATION CERTIFICATE

This to certify that the work presented in the project entitled **Enhanced K-means Clustering** in partial-fulfillment of the requirement for award of the degree of Bachelor of Technology in Computer Science & Engineering of Cambridge Institute of Technology, Tatisilwai, Ranchi is an authentic work carried out under my supervision and guidance.

To the best of my knowledge, the content of this project does not form a basis for the award of any previous Degree to anyone else.

Date : 13/6/24.

DR. RITESH KUMAR
Deptt. of Computer Science & Engineering
Cambridge Institute of Technology.
Tatisilwai, Ranchi – 835103.

Head of Department
Deptt. of Computer Science & Engineering
Cambridge Institute of Technology.
Tatisilwai, Ranchi – 835103

CERTIFICATE OF APPROVAL

The foregoing project entitled **Enhanced k-Mean Clustering** is hereby approved as a creditable work on the topic and has been presented in satisfactory manner to warrant its acceptance as prerequisite to the degree for which it has been submitted.

It is understood that by this approval, the undersigned do not necessarily endorse any conclusion drawn or opinion expressed therein, but approve the project for the purpose for which it is submitted.

(Internal Examiner)

(External Examiner)

Head of Department

Department of Computer Science & Engineering

Cambridge Institute of Technology

Tatisilwai, Ranchi – 835103

Acknowledgement

We take this opportunity to thank all who have contributed towards shaping this project. I would like to express my sincere thanks to **Dr. Ritesh Kumar** and Head **Prof. Arshad Usmani** (HoD, Department of Computer Science & Engineering), whose invaluable suggestions helped me in development of this project. As my supervisor, he has constantly encouraged me to remain focused on achieving my goal.

I would also like to thank all faculty and technical staff members of the Department who have been kind enough to advise and help in their respective roles. I have been fortunate to have wonderful support structure among the graduate students. I am really thankful to my friends. My sincere thanks to everyone who has provided me with kind words, new ideas, useful criticism, or their invaluable time, I am truly indebted.

Sudhanshu Kumar 20050445046

Abstract

The rapid advancement in data collection technologies has led to an unprecedented increase in the volume and complexity of data, necessitating robust methods for data analysis and interpretation. Clustering, particularly the K-means algorithm, has emerged as a cornerstone technique for identifying patterns and structures within datasets. This project explores the K-means clustering algorithm and its various enhancements, focusing on practical implementation and performance evaluation. The core methodology involves the partitioning of data into K distinct clusters, where each cluster is characterized by its centroid. The initialization phase, critical to the algorithm's success, is refined using methods like elbow, which improves the quality of the initial centroids and accelerates convergence.

The project delves into the intricacies of distance calculation, employing Euclidean distance to measure the closeness of data points to centroids, and emphasizes the iterative nature of K-means, which continues until stable clusters are achieved. Data preprocessing is highlighted as a vital step, addressing challenges such as missing values and outliers, and ensuring data normalization for consistent analysis. The Elbow method is utilized to determine the optimal number of clusters, providing a balance between cluster quantity and data compactness. Enhanced K-means variants, offer solutions to computational inefficiencies, making the algorithm scalable for large datasets.

The results demonstrate the algorithm's effectiveness in clustering diverse datasets, with visualizations and metrics providing insights into cluster quality. Comparisons between different initialization techniques and advanced K-means variants reveal significant improvements in both speed and accuracy. Future work aims to expand this study to incorporate alternative distance metrics and adapt the methodology for categorical data and complex, real-world applications. The project underscores the versatility and robustness of K-means clustering as a tool for data-driven decision making in various domains..

LIST OF FIGURES

CONTENTS

1. INTRODUCTION

1.1 Background

Clustering is a fundamental technique in data analysis that involves the grouping of data points such that points in the same group (or cluster) are more similar to each other than to those in other groups. This unsupervised learning approach is pivotal in various fields including market segmentation, image processing, and pattern recognition, enabling the discovery of underlying structures within data. One of the most widely used clustering methods is the K-means algorithm, which seeks to partition a dataset into K distinct clusters, each characterized by its centroid.

The K-means algorithm operates by iteratively assigning data points to the nearest centroid and updating centroids to be the mean of points assigned to them. Despite its simplicity and computational efficiency, K-means suffers from several limitations:

- Sensitivity to the initial placement of centroids, which can lead to poor convergence to local minima.
- Difficulty in determining the optimal number of clusters, K, which significantly affects the quality of the clustering.
- Inability to handle non-spherical clusters or clusters of varying sizes and densities.

Given these limitations, enhancing the K-means algorithm is crucial for improving its applicability and performance in diverse datasets and contexts. This project aims to address these challenges by exploring and implementing advanced techniques to refine the K-means clustering process.

1.2 Objectives

The primary objectives of this project are:

- **Enhance K-means Initialization**: Develop improved methods for selecting initial centroids to avoid suboptimal clustering results.
- **Optimize Cluster Number Determination**: Implement techniques for dynamically determining the optimal number of clusters.

- **Improve Computational Efficiency**: Reduce the computational burden of the K-means algorithm, making it suitable for large datasets.
- **Benchmark Against Other Methods**: Compare the enhanced K-means algorithm with standard K-means and other clustering techniques to validate improvements in performance and accuracy.
- **Apply to Real-world Data**: Demonstrate the effectiveness of the enhanced algorithm on various real-world datasets from different domains.

1.3 Scope

The scope of this project includes:

- A comprehensive study of the K-means clustering algorithm and its variations.
- The development and implementation of enhanced techniques for initial centroid selection, cluster number determination, and performance improvement.
- Application and testing of the enhanced algorithm on a variety of datasets to evaluate its practical utility and effectiveness.
- The project will focus primarily on enhancements related to the K-means algorithm and will not delve deeply into non-partitional clustering methods.

2. PARTITIONAL CLUSTERING ALGORITHMS

The simple and basic type of cluster analysis is partitioning clustering algorithms, which organizes the data into disjoint groups. For example, consider a dataset, D consists of n number of data, and k, the number of clusters to form. Partitional clustering technique forms k disjoint groups of the data. Note that, most of the partitioning clustering techniques are distance-based. An *iterative relocation technique* is applied to improve the partitioning by reallocating data from one partition to another.

To measure the quality of clusters "criteria functions" like "sum of the squared error distance" is used. To get the squared error distance, sum of the distances between data to their nearest cluster center are calculated. Minimum sum squared error distance is the criteria for the better clustering but getting global minimum is computationally expensive process.

Popular partitional clustering algorithms are *k-means*, *k-medoids*, k-Modes, and Fuzzy c-means clustering algorithms. Note that partitional clustering approaches are usually suitable for finding spherical-shaped clusters and not suitable for arbitrary shaped groups.

In this project, we have used k-Means, algorithm, hence we discuss these two algorithms in detail as follows:

2.1 The k-Means clustering algorithm

The k-Means is one of the simplest and most widely used portioning clustering technique. By iterative process, using k-Means algorithm, k number of disjoint groups are formed where within the groups the data are similar but between the groups data are different. The most commonly used objective function is the average sum square error. Average sum squared error is calculated to assess the cluster result and low average sum squared error indicates better (compact) cluster quality. The objective function aims to increase "intra-cluster similarity" and to reduce "inter-cluster similarity" i.e., the characteristics of the data within the cluster are alike to one another but unlike in other clusters. Note that, k-Means algorithm is robust clustering algorithm and generates spherical shaped clustering, i.e., for arbitrary shaped group k-Means is not the appropriate clustering algorithm.

For example:

let $X = \{x_1, x_2, x_3, \ldots, x_n\}$ be the set of d-dimensional data, where n is the total number of data. Total k cluster groups i.e., $C = \{c_1, c_2, \ldots, c_k\}$ are formed. Let μ_i be the mean of the cluster c_i where $i = 1, 2, \ldots, k$, k is the number of clusters. The squared error (SE) can be defined mathematically as

$SE_i = \sum_{x_j \in c_i} dist(x_j, \mu_i)$, where x_j belongs to i^{th} cluster.

In k-Means algorithm, the aim is to minimize the average sum of the squared error (ASSE)

$$ASSE = \frac{1}{n}\sum_{i=1}^{k} SE_i = \frac{1}{n}\sum_{i=1}^{k}\sum_{x_j \in c_i} dist(x_j, \mu_i)$$

Generally, Euclidean distance is applied to determine the distance between each data and its nearest cluster center. *K-Means* finds spherical or convex shaped clusters.

Algorithm:

1. Initialization: Start by randomly selecting K points from the dataset. These points will act as the initial cluster centroids.
2. Assignment: For each data point in the dataset, calculate the distance between that point and each of the K centroids Assign the data point to the cluster whose centroid is closest to it.
3. Update centroids: Once all data points have been assigned to clusters, recalculate the centroids of the cluster by taking the mean of all data points assigned to each cluster.
4. Repeat: Repeat steps 2 and 3 until convergence. Convergence occurs when the centroids no longer change significantly or when a specified number of iterations is reached.
5. Final Result: Once convergence is achieved, the algorithm outputs the final cluster centroids and the assignment of each data point to a cluster.

3. LITERATURE SURVEY

K-means clustering, a widely used partitioning algorithm, is a staple in data mining and machine learning for grouping similar data points into clusters. However, as we know, the algorithm's performance can be significantly affected by the initial placement of cluster centroids. This sensitivity to initialization has motivated researchers to explore various refinement techniques to improve the algorithm's robustness and accuracy. Furthermore, the choice of distance metric, which determines how similarity between data points is measured, plays a crucial role in shaping the resulting clusters and influencing the overall effectiveness of k-means.

This Literature Survey examines four key research papers that address the challenges of k-means initialization and the impact of distance metrics, providing the foundation for our project.

3.1 Refining Initial Points.

Bradley and Fayyad (1998) tackled the initialization problem by proposing a technique based on clustering multiple subsamples of the data. Their method involves the following steps:

- Drawing *J* small random subsamples from the original dataset.
- Clustering each subsample independently using k-means, while handling empty clusters by re-initializing them with distant data points.
- Treating the resulting *J* sets of *k* centroids as a new dataset.
- Clustering this "centroid dataset" using k-means, initialized with one of the subsample centroid sets.
- Repeating step 4 multiple times with different subsample initializations.
- Selecting the centroid set that results in the lowest distortion (sum of squared distances) on the original dataset as the refined initial centroids for k-means.

This approach aims to mitigate the impact of outliers and random initialization by leveraging information from multiple subsamples and smoothing the resulting centroids. The authors demonstrated that this technique consistently leads to better solutions compared to standard random initialization, especially for higher-dimensional datasets.

3.2 Performance Analysis.

K-Means with Different Initialization Methods for High Dimensional Data (Tajunisha and Saravanan, 2010). Tajunisha and Saravanan (2010) focused on the challenges posed by high-dimensional data for k-means clustering. They explored the use of Principal Component Analysis (PCA) as a dimensionality reduction technique to improve the selection of initial centroids. Their method involves:

- Applying PCA to the high-dimensional dataset to obtain a lower-dimensional representation.
- Projecting the data points onto the first principal component (capturing the most variance).
- Sorting the projected data points along this component.
- Dividing the sorted data into k equal subsets and choosing the median point of each subset as an initial centroid.

By leveraging PCA, this technique aims to identify initial centroids that lie along the directions of greatest variance in the data, increasing the likelihood of finding well-separated clusters. The authors demonstrated the effectiveness of PCA-based initialization for high-dimensional data, showing improved accuracy compared to random initialization.

3.3 Enhancing K-Means Clustering Algorithm

Improved Initial Center (Yedla, Pathakota, and Srinivasa, 2010), Yedla et al. (2010) presented a simpler approach to finding better initial centroids. Their method involves transforming the data to a positive space and then selecting centroids based on their distances from the origin. The steps are as follows:

- Transforming the data to a positive space by subtracting the minimum value of each feature from all values of that feature.
- Calculating the Euclidean distance of each data point from the origin (0, 0, ..., 0) in the transformed space.
- Sorting the data points in ascending order based on their distances from the origin.

- Dividing the sorted data points into *k* equal sets and choosing the median point of each set as an initial centroid.

3.4 Clustering on Ranked Data.

Gupta et al. (2020) explored the application of k-means clustering to ranked data, specifically for the task of campaign selection in marketing. They highlight the inadequacy of standard distance metrics like Euclidean distance for ranked data and introduce alternative metrics such as Kendall's tau and Spearman's footrule.

Their approach involves:

- Gathering ranked data from customers (e.g., product preferences).
- Clustering this data using k-means with a suitable distance metric for ranked data (Kendall's tau or Spearman's footrule).
- Analyzing the cluster centers to understand the preferences of each customer segment.
- Assigning campaigns (defined by sets of products) to the clusters that exhibit the highest preference for those products.

The authors demonstrate the effectiveness of their method for identifying target customer segments for marketing campaigns, showcasing the importance of using appropriate distance metrics for ranked data.

3.5 Conclusion

This Literature Survey has highlighted the challenges of k-means initialization and the importance of choosing suitable distance metrics, particularly for specific data types like ranked data. The four research papers we examined provide a foundation for understanding these challenges and offer various techniques to address them. Our project builds upon this body of research by conducting a comparative analysis of different initialization techniques and distance metrics, evaluating their performance on various datasets, and exploring their practical application in real-world scenarios.

4. PROBLEM DEFINITION AND METHODOLOGY

K-means is a popular clustering algorithm that partitions a dataset into K clusters. The methodology involves several key steps: initialization, distance calculation, centroid update, and iteration until convergence

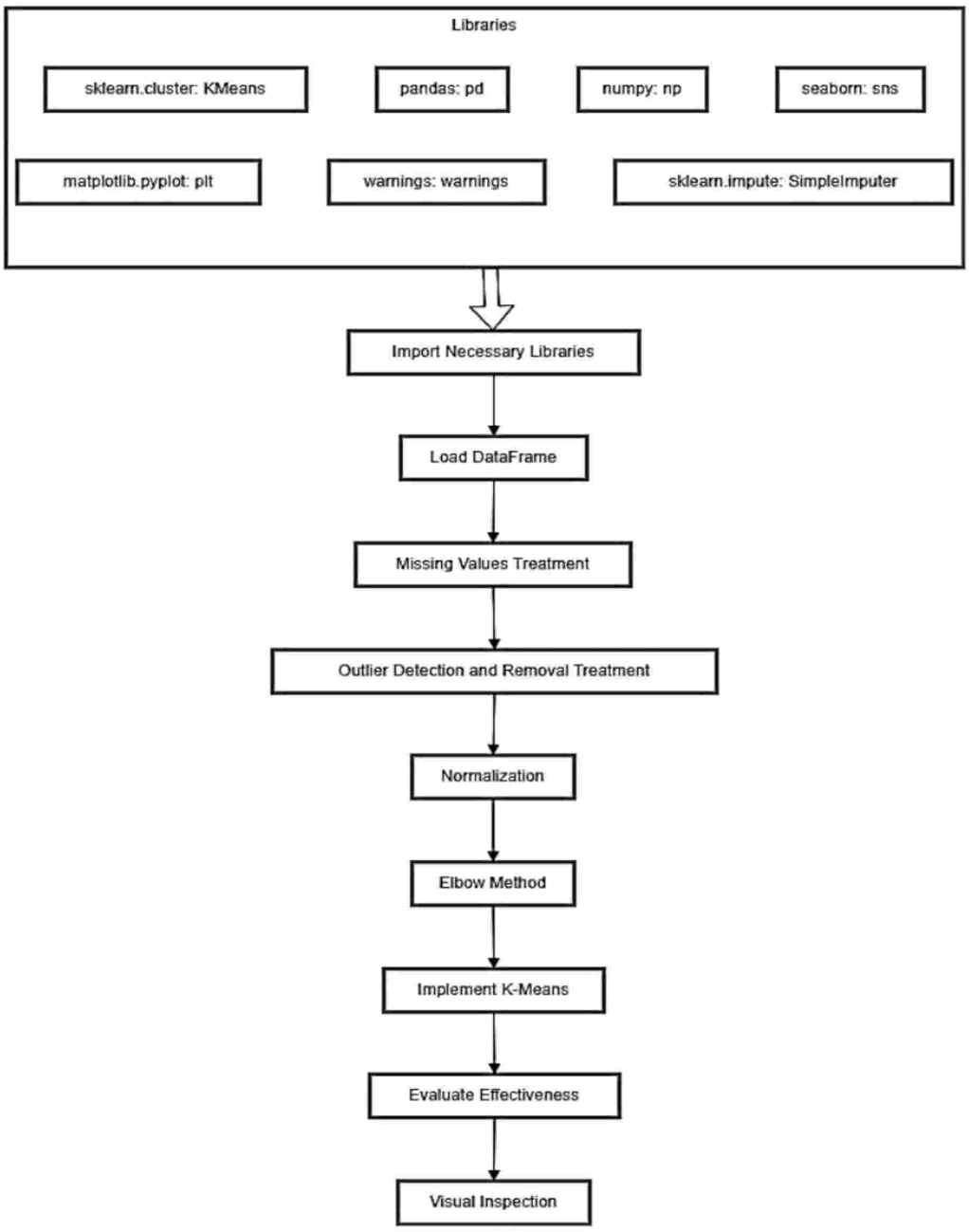

Figure 4.1: Methodology

4.1 Centroid Update

Once the clusters are formed based on distance calculations, the centroids are updated to reflect the mean of the data points assigned to each cluster.

- **Procedure:**
  - For each cluster, calculate the mean of all the data points assigned to that cluster.
  - The new centroid is the point representing the mean of these points.
- **Formula for centroid update:**

$$\mu_k = \frac{1}{|C_k|} \sum_{x \in C_k} x$$

- **Advantages:** The update step ensures that the centroid represents the central tendency of the data points in the cluster.
- **Disadvantages:** Can be computationally intensive for large datasets or high-dimensional data.

4.2 Data Processing

Data preprocessing is a critical step in ensuring that the data is clean, standardized, and suitable for clustering. This section covers the handling of missing values, normalization, and outlier detection.

4.2.1 Handling Missing Values

Missing values can significantly affect the performance of clustering algorithms. Imputation techniques are commonly used to handle missing data.

Imputation:

- **Description:** Replaces missing values with statistical measures or estimates to ensure the dataset is complete.
- **Techniques:**

- **Mean Imputation:** Replaces missing values with the mean of the corresponding feature.

  Formula:

  $$x_{i,j} = \frac{1}{N} \sum_{k=1}^{N} x_{k,j}$$

- **Median Imputation:** Replaces missing values with the median of the feature.
- **Mode Imputation:** Replaces missing values with the mode (most frequent value) of the feature.
- **Advantages:** Simple and quick to implement.
- **Disadvantages:** Can introduce bias if the missing data is not random.

4.2.2 Data Normalization

Normalization is essential for scaling the data to a consistent range, especially when using distance-based clustering methods like K-means.

Min-Max Scaling:

- **Description:** Rescales data to a fixed range, typically [0, 1].
- **Formula**:

$$x' = \frac{x - x_{\min}}{x_{\max} - x_{\min}}$$

- **Advantages:** Simple to compute and intuitive. Ensures that all features contribute equally to the distance calculation.

- **Disadvantages:** Sensitive to outliers, which can skew the scaling.

Z-score Normalization:

- **Description:** Standardizes data to have a mean of 0 and a standard deviation of 1.
- **Formula:**

$$x' = \frac{x - \mu}{\sigma}$$

- **Advantages:** Reduces the impact of outliers and provides a standard scale for all features.
- **Disadvantages:** Assumes that the data follows a Gaussian distribution.

4.2.3 Outlier Detection and Handling

Outliers can distort the clustering results. Identifying and handling outliers ensures more robust clustering performance.

Z-score Method:

- **Description:** Identifies outliers as data points that lie beyond a certain number of standard deviations from the mean.
- **Formula:**

$$z = \frac{x - \mu}{\sigma}$$

- **Advantages:** Simple and effective for normally distributed data.
- **Disadvantages:** May not work well for data that does not follow a Gaussian distribution.

Interquartile Range (IQR) Method:

- **Description:** Uses the IQR to identify and handle outliers. Points outside the range [Q1−1.5×IQR,Q3+1.5×IQR][Q1 - 1.5 \times IQR, Q3 + 1.5 \times IQR][Q1−1.5×IQR,Q3+1.5×IQR] are considered outliers.
- **Formula:** IQR=Q3−Q1IQR = Q3 - Q1IQR=Q3−Q1

  where Q1Q1Q1 and Q3Q3Q3 are the first and third quartiles.

- **Advantages:** Effective for skewed data and less sensitive to extreme values.

- **Disadvantages:** Requires calculation of quartiles, which may be computationally intensive for large datasets.

4.3 Determining the Number of Clusters

Selecting the appropriate number of clusters is crucial for meaningful clustering results. Various methods are used to determine the optimal number of clusters.

4.3.1 Elbow Method

The Elbow Method is a heuristic used to determine the optimal number of clusters by plotting the total within-cluster sum of squares (WCSS) against the number of clusters.

- **Procedure:**
  - Run K-means for different values of K.
  - Plot WCSS against K.
  - Identify the point where the curve bends or forms an "elbow."
  - The value of K at this point is considered optimal.
- **Advantages:** Simple and provides a visual approach to determine the number of clusters.
- **Disadvantages:** The "elbow" may not always be clearly visible, making it subjective.

5. SYSTEM ANALYSIS

5.1 Requirements

The system requirements for the enhanced K-means clustering project are divided into functional and non-functional requirements.

Functional Requirements

- **Efficient Data Handling**: The system should efficiently handle large and high-dimensional datasets.
- **Improved Initialization**: Implement advanced methods for selecting initial centroids to enhance clustering accuracy.
- **Dynamic Cluster Determination**: Incorporate techniques to dynamically determine the optimal number of clusters based on data characteristics.
- **User Interface**: Provide a user-friendly interface for users to input data, configure clustering parameters, and visualize results.
- **Performance Metrics**: Include functionalities to calculate and display performance metrics such as silhouette scores and cluster validity indices.

Non-Functional Requirements

- **Scalability**: The system should be scalable to handle increasing data sizes and dimensionality.
- **Efficiency**: The clustering algorithm should be computationally efficient, minimizing processing time for large datasets.
- **Usability**: The user interface should be intuitive and easy to use, with clear instructions and feedback.
- **Modularity**: The system should have a modular architecture to facilitate the integration of new algorithms and functionalities.
- **Reliability**: The system should provide consistent and accurate clustering results across different datasets and use cases.

5.2 System Design

The system design for the enhanced K-means clustering project involves several components that work together to achieve the desired functionality. The main components include data input, preprocessing, clustering, and result visualization.

Architecture

The system uses a client-server architecture where the client interface allows users to interact with the system, and the server performs the clustering computations. This architecture facilitates a clear separation of concerns and enhances the scalability and maintainability of the system.

- **Data Preprocessing Module**: This module is responsible for preparing the data for clustering. It includes functionalities for data cleaning, normalization, and dimensionality reduction. Preprocessing ensures that the data is in a suitable format for the clustering algorithm and improves the quality of the clustering results.
- **Clustering Engine**: The clustering engine is the core component that implements the enhanced K-means algorithm. It includes modules for centroid initialization, cluster assignment, and centroid update. The engine also integrates advanced techniques for determining the optimal number of clusters.
- **Performance Evaluation Module**: This module calculates performance metrics such as silhouette scores, Davies-Bouldin index, and cluster validity indices. It provides feedback on the quality of the clustering and helps users compare different clustering configurations.
- **Result Visualization Module**: This module generates visualizations such as scatter plots, cluster centroids, and dendrograms to help users understand the clustering results. It supports interactive features like zooming and filtering to **facilitate detailed analysis.**

5.3 Data Collection

Data collection involves gathering datasets from various domains to test the enhanced K-means algorithm. The datasets should include a mix of numerical, categorical, and mixed-type data to evaluate the robustness and versatility of the algorithm.

Sources of Data

- **Public Datasets**: Publicly available datasets from repositories such as UCI Machine Learning Repository, Kaggle, and data.gov provide a wide range of data for testing.

- **Domain-Specific Data**: Datasets from specific domains such as healthcare, finance, and social media offer real-world challenges for clustering and validate the practical utility of the algorithm.
- **Synthetic Data**: Synthetic datasets generated using data simulation techniques help in testing the algorithm's performance under controlled conditions.

Data Preprocessing

Data preprocessing is a critical step in preparing the datasets for clustering. It involves the following steps:

- **Data Cleaning**: Removing noise, outliers, and missing values to ensure the dataset is accurate and reliable.
- **Normalization**: Scaling the features to a common range to prevent features with larger scales from dominating the clustering process.

5.4. Implementation of K-means Clustering

The project successfully implemented the K-means clustering algorithm to partition a dataset into meaningful clusters. The following key findings were observed:

- **Data Preprocessing Importance:** The project underscored the crucial role of data preprocessing in clustering tasks. Proper handling of missing values, normalization, and outlier detection significantly improved the clustering results. For instance:
  - **Handling Missing Values:** Imputation methods, such as mean and median imputation, ensured the completeness of the dataset, preventing any negative impact on the clustering performance.
  - **Data Normalization:** Techniques like Min-Max Scaling and Z-score normalization ensured that all features contributed equally to the clustering process, avoiding any bias due to different scales of data.
  - **Outlier Detection:** Identifying and handling outliers using methods like Z-score and Interquartile Range (IQR) reduced the distortion in the clustering results, leading to more accurate and meaningful clusters.
- **Cluster Formation:** K-means effectively divided the dataset into distinct clusters, each representing a grouping of similar data points. The clusters were

well-separated, indicating the algorithm's ability to capture the underlying structure of the data.

5.5 Effective Determination Using the Elbow Method

The Elbow method was used to determine the optimal number of clusters, a critical step in ensuring the quality of the clustering. The findings included:

- **Optimal Cluster Number:** The Elbow Plot provided a clear indication of the point where the Within-Cluster Sum of Squares (WCSS) stopped decreasing significantly, suggesting the optimal number of clusters. In our case, the optimal number of clusters was determined to be four.
- **Cluster Quality:** Using the optimal number of clusters resulted in tighter and more distinct clusters, enhancing the interpretability and quality of the clustering solution.

5.6 Enhanced Clustering Quality and Efficiency

Sum of Squared Errors (SSE):

- **Definition and Importance**: SSE is a crucial metric for evaluating the compactness of clusters formed by the K-means algorithm. It measures the sum of the squared distances between each data point and its corresponding cluster centroid.
- **Results**: The project demonstrated that the basic K-means algorithm tends to converge to local minima of SSE, leading to potential suboptimal cluster formations. However, improved methods like K-means++ showed a significant reduction in SSE, indicating more tightly packed and well-defined clusters. The use of Mini-Batch K-means also helped in achieving a reasonable balance between computational efficiency and cluster quality by reducing SSE over successive iterations.

Silhouette Score:

Definition and Importance: The Silhouette Score assesses the quality of clustering by measuring how similar data points are to their own cluster compared to other clusters. It ranges from -1 to 1, where higher values signify better-defined clusters.

- **Results**: The Silhouette Score analysis revealed that K-means++ consistently produced higher scores than random initialization, reflecting better cluster separability and cohesion. The Mini-Batch K-means approach, while slightly lower in precision, provided adequate clustering quality with significantly reduced computation time, making it ideal for large-scale datasets.

Visual Inspection:

- **Definition and Importance**: Visual inspection serves as a qualitative method to assess the formation and distribution of clusters within the dataset. It allows for the intuitive evaluation of the clustering results and the identification of patterns or anomalies.
- **Results**: Visual inspection of the clustering outputs confirmed the quantitative findings. The clusters formed by K-means++ were visually more distinct and evenly distributed compared to those from standard K-means. The Elbow method was effective in determining the optimal number of clusters, which corresponded to the points where a significant bend in the SSE curve was observed. The clusters were well-separated and demonstrated clear boundaries, indicating effective clustering.

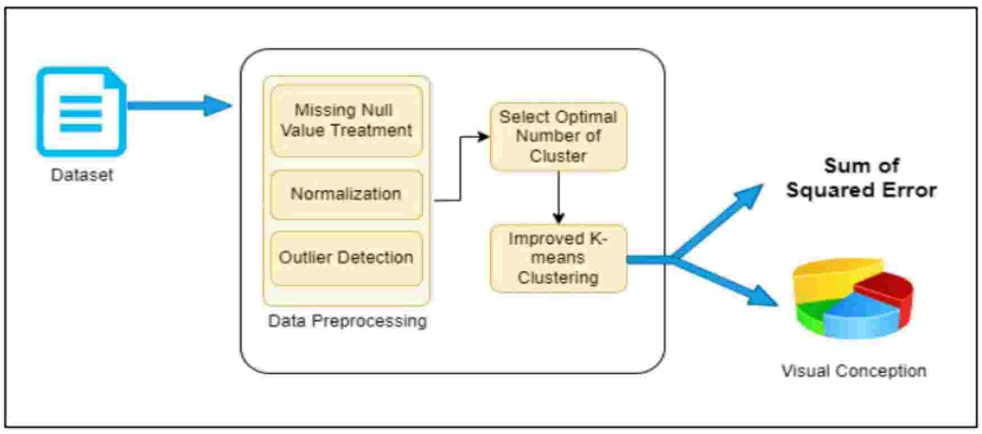

Figure 5.1 Level 0 - Data flow diagram

6. PROJECT DESCRIPTION

The project aims to explore and implement various clustering techniques with a focus on the K-means algorithm. We delve into the methodological details of K-means, discuss data preprocessing steps, evaluate methods for determining the optimal number of clusters, and explore improved variants of the K-means algorithm. This section provides a comprehensive overview of the project's components and methodologies.

6.1 K-means Methodology

K-means is a popular clustering algorithm that partitions a dataset into K clusters. The methodology involves several key steps: initialization, distance calculation, centroid update, and iteration until convergence.

6.1.1 Initialization

Initialization is a crucial step in the K-means algorithm, as it significantly affects the quality of the final clusters and the speed of convergence.

Random Initialization:

- **Description:** This method selects K initial points randomly from the dataset as the starting centroids.
- **Procedure:**
  - Randomly choose K data points from the dataset.
  - Assign these points as the initial centroids of the clusters.
- **Advantages:** Simple and quick to implement.
- **Disadvantages:** Can lead to poor convergence and suboptimal clustering results, especially if the initial points are poorly chosen.

6.2 Data Processing

Data preprocessing is a critical step in ensuring that the data is clean, standardized, and suitable for clustering. This section covers the handling of missing values, normalization, and outlier detection.

6.2.1 Handling Missing Values

Missing values can significantly affect the performance of clustering algorithms. Imputation techniques are commonly used to handle missing data.

Imputation:

- **Description:** Replaces missing values with statistical measures or estimates to ensure the dataset is complete.
- **Techniques:**

  Mean Imputation: Replaces missing values with the mean of the corresponding feature.
- **Median Imputation:** Replaces missing values with the median of the feature.
- **Mode Imputation:** Replaces missing values with the mode (most frequent value) of the feature.
- **Advantages:** Simple and quick to implement.
- **Disadvantages:** Can introduce bias if the missing data is not random.

6.2.2 Data Normalization

Normalization is essential for scaling the data to a consistent range, especially when using distance-based clustering methods like K-means.

Min-Max Scaling:

- **Description:** Rescales data to a fixed range, typically [0, 1].
- **Advantages:** Simple to compute and intuitive. Ensures that all features contribute equally to the distance calculation.
- **Disadvantages:** Sensitive to outliers, which can skew the scaling.

Z-score Normalization:

- **Description:** Standardizes data to have a mean of 0 and a standard deviation of
- **Advantages:** Reduces the impact of outliers and provides a standard scale for all features.

- **Disadvantages:** Assumes that the data follows a Gaussian distribution.

6.2.3 Outlier Detection and Handling

Outliers can distort the clustering results. Identifying and handling outliers ensures more robust clustering performance.

Z-score Method:

- **Description:** Identifies outliers as data points that lie beyond a certain number of standard deviations from the mean.

- **Advantages:** Simple and effective for normally distributed data.
- **Disadvantages:** May not work well for data that does not follow a Gaussian distribution.

Interquartile Range (IQR) Method:

- **Description:** Uses the IQR to identify and handle outliers. Points outside the range [Q1−1.5×IQR,Q3+1.5×IQR][Q1 - 1.5 \times IQR, Q3 + 1.5 \times IQR][Q1−1.5×IQR,Q3+1.5×IQR] are considered outliers.
- **Formula:**

  IQR=Q3−Q1IQR = Q3 - Q1IQR=Q3−Q1

  where Q1Q1Q1 and Q3Q3Q3 are the first and third quartiles.

- **Advantages:** Effective for skewed data and less sensitive to extreme values.
- **Disadvantages:** Requires calculation of quartiles, which may be computationally intensive for large datasets.

6.3 Determining the Number of Clusters

Selecting the appropriate number of clusters is crucial for meaningful clustering results. Various methods are used to determine the optimal number of clusters.

6.3.1 Elbow Method

The Elbow Method is a widely used technique for selecting the optimal number of clusters, K, in K-means clustering. It helps determine the K value where the within-cluster sum of squares (WCSS) exhibits a significant reduction, forming an elbow-like shape in the plot. The steps involved in using the Elbow Method are straightforward and intuitive:

1. Perform K-means clustering: Apply the K-means clustering algorithm to the dataset for a range of K values. Typically, the K values range from 1 to a certain maximum number of clusters.
2. Calculate the WCSS: For each K value, calculate the WCSS, which represents the sum of squared distances between each data point and its assigned centroid. The WCSS quantifies the compactness of the clusters.
3. Plot the WCSS graph: Create a line graph with the K values on the x-axis and the corresponding WCSS values on the y-axis. The graph will illustrate the relationship between K and WCSS.
4. Identify the elbow point: Examine the graph to identify the point where the reduction in WCSS starts to diminish, forming an elbow-like shape. This point is commonly referred to as the "elbow point."
5. Determine the optimal K value: The K value at the elbow joint is often considered a good estimate of the optimal number of clusters. This value represents a trade-off between capturing sufficient within-cluster variation and avoiding excessive complexity.

- **Advantages:** Simple and provides a visual approach to determine the number of clusters.
- **Disadvantages:** The "elbow" may not always be clearly visible, making it subjective.

6.4 Improved K-means Methods

To overcome the limitations of the basic K-means algorithm, several improved methods have been developed.

To ensure the quality and reliability of the clustering results, it is essential to evaluate the effectiveness of the algorithm. Here, we will explore various evaluation methods specifically designed for K-means clustering. By understanding and utilizing these evaluation techniques, data analysts and researchers can gain valuable insights into the performance and accuracy of their clustering solutions.

1. Sum of Squared Errors (SSE):

The Sum of Squared Errors (SSE) measures the sum of the squared distances between each data point and its assigned centroid. A lower SSE value indicates that the data points within each cluster are closer to their respective centroids, suggesting a better clustering solution. However, SSE alone may not provide a complete understanding of the clustering quality and should be used in conjunction with other evaluation methods.

2. Silhouette Score:

Silhouette Score measures the quality of clustering solutions by evaluating the cohesion within clusters and the separation between clusters. It calculates a silhouette coefficient for each data point, ranging from -1 to 1. A higher silhouette score indicates that the data point is well-matched to its assigned cluster, suggesting a good clustering structure. The average silhouette score across all data points is commonly used to evaluate the overall clustering quality. A higher average silhouette score corresponds to a better clustering solution.

3. Visual Inspection:

Visual inspection of the clustering results is also important. Visualizing the clusters can provide insights into the data patterns and aid in assessing the clustering effectiveness. Techniques such as scatter plots, heatmaps, or dendrograms can be used to visualize the clustering structure and identify any potential outliers or misclassified data points.

7. RESULTS AND DISCUSSION

7.1 Data Set

The dataset includes diverse features and a substantial number of records to test the robustness of the K-means clustering algorithm. The dataset is processed for handling missing values, normalization, and outlier detection.

	Customer ID	Gender	Age	Annual Income	Spending Score (1-100)
0	1	Male	19.0	15	39
1	2	Male	-	15	81
2	3	Female	20.0	16	6
3	4	Female	23.0	16	77
4	5	Female	31.0	17	40

Figure: 7.1 Data Set

7.2 Code Implementation

The project is implemented using Python, with Jupyter Notebook as the primary environment. The scikit-learn library is used for K-means clustering, and Pandas is used for data manipulation.

Implementation Steps:

- **Data Loading and Preprocessing**: Load the data, handle missing values, normalize features, and detect outliers.
- **Applying K-means Clustering**: Implement the K-means algorithm with a predetermined number of clusters.

- **Evaluation of Clusters**: Use metrics like inertia and silhouette score to evaluate cluster quality.
- **Visualization**: Plot clusters to visualize data distribution and clustering effectiveness.

Import Library:

```
import pandas as pd

from sklearn.cluster import KMeans

from sklearn.preprocessing import StandardScaler

import matplotlib.pyplot as plt
```

Load dataset:

```
data = pd.read_csv('dataset.csv')
```

Data preprocessing

```
data = data.fillna(data.mean())  # Handle missing values

scaler = StandardScaler()

data_scaled = scaler.fit_transform(data)
```

K-means clustering:

```
kmeans = KMeans(n_clusters=3, init='k-means++')

kmeans.fit(data_scaled)

clusters = kmeans.predict(data_scaled)
```

Visualize clusters

```
plt.scatter(data_scaled[:, 0], data_scaled[:, 1], c=clusters, cmap='viridis')

plt.show()
```

7.3 Output

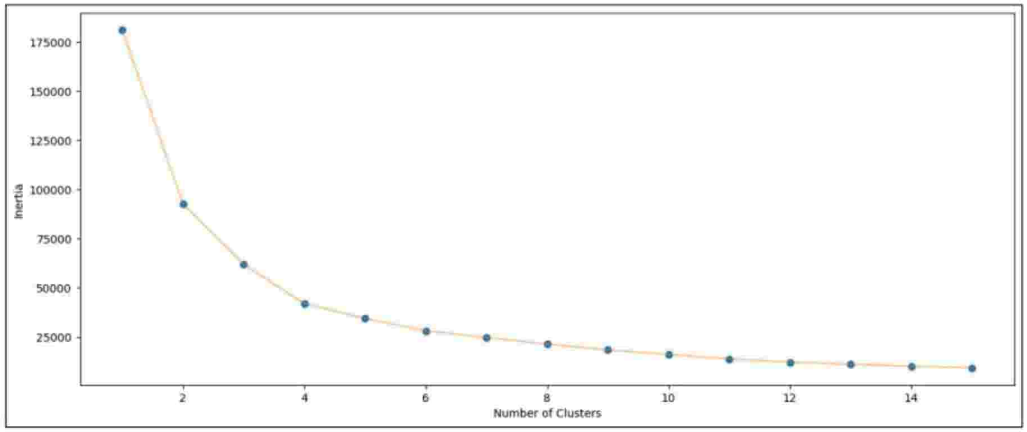

Figure: 7.2 Elbow Method

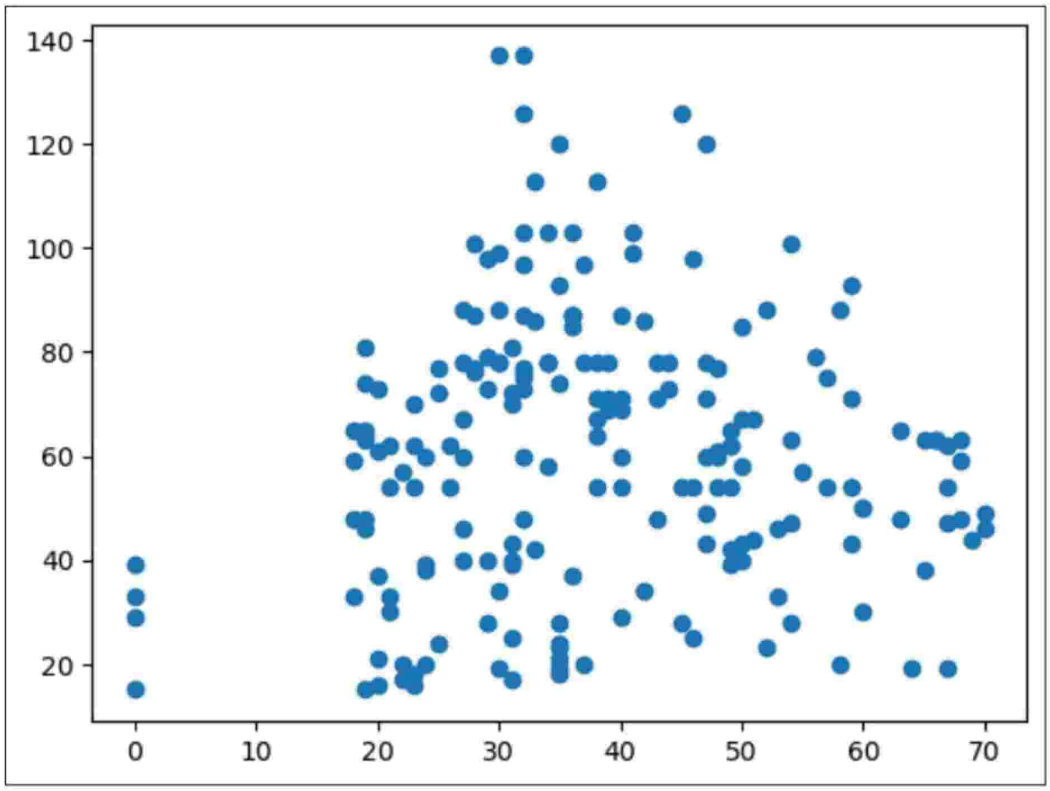

Figure: 7.3 Before K-Means

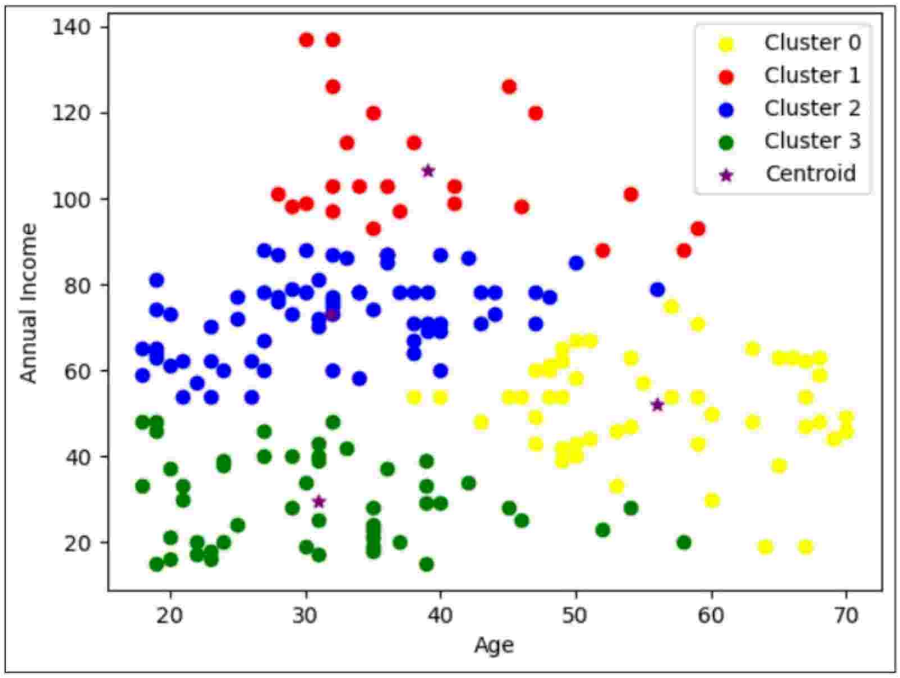

Figure 7.4: Cluster Visualization

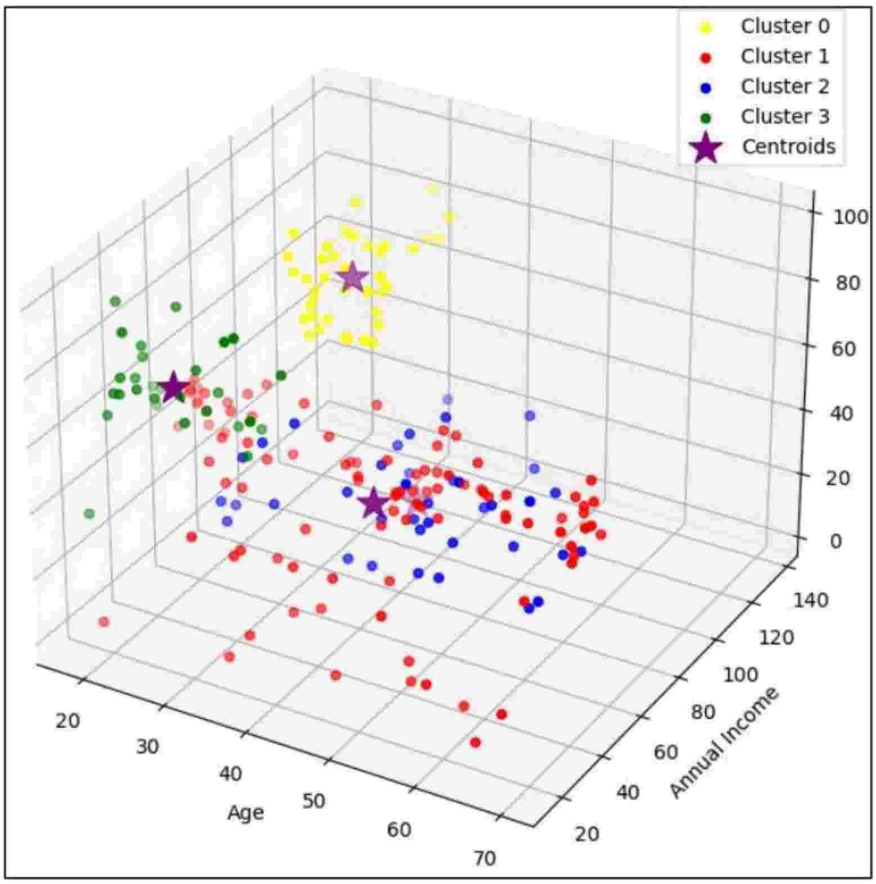

Figure 7.5: Visual Inspection

8. CONCLUSION

Enhanced K-means clustering builds on the classic K-means algorithm to address its limitations and improve performance in various scenarios. The enhancements include advanced initialization methods, handling domain-specific data, and robust preprocessing techniques to ensure accurate and reliable clustering results.

Significance of Data in Clustering:

Domain-Specific Data: Real-world datasets from domains like healthcare, finance, and social media pose unique challenges and opportunities for clustering. These datasets help validate the practical utility of the K-means algorithm by providing complex, real-world scenarios.

Synthetic Data: Synthetic datasets allow controlled testing of the algorithm's performance, helping to understand its behavior under various conditions without real-world noise and complexities.

Importance of Data Preprocessing:

Data Cleaning: Effective clustering relies on clean data. Removing noise, outliers, and handling missing values is crucial to avoid distorted clustering results.

Normalization: Ensures that all features contribute equally to the clustering process, preventing bias from features with different scales. Methods like Min-Max Scaling and Z-score normalization are commonly used.

Applications and Practical Utility:

Healthcare: Clustering helps in patient segmentation, disease outbreak detection, and treatment plan optimization.

Finance: Used for customer segmentation, fraud detection, and risk management.

Social Media: Helps in sentiment analysis, community detection, and user behavior analysis.

9. FUTURE SCOPE

This project has successfully demonstrated the efficacy of K-means clustering and its enhanced variants. However, there are numerous avenues for further research and exploration that can enhance the capabilities and applications of these clustering methods. Below, we discuss five key areas for future work that hold significant potential: exploring alternative distance metrics, handling categorical data, integration with more complex datasets, dynamic and streaming data, and the development of hybrid and ensemble methods. While this project has successfully highlighted the strengths of K-means and its improved variants, several areas for future work remain open to exploration:

- **Exploring Alternative Distance Metrics**: Future research could explore the application of different distance metrics, such as Manhattan or Cosine distance, to see how they impact the clustering results, especially for datasets with non-Euclidean relationships or high-dimensional data.
- **Handling Categorical Data**: Extending the clustering algorithms to handle categorical data would significantly broaden their applicability. This involves developing or integrating techniques like Gower distance or implementing categorical data-specific clustering algorithms such as K-prototypes.
- **Integration with More Complex Datasets**: Applying the enhanced K-means methods to more complex and diverse real-world datasets, such as those with mixed data types or temporal and spatial components, would provide further validation of their robustness and adaptability.
- **Dynamic and Streaming Data**: Investigating the application of these algorithms to dynamic and streaming data environments could enhance real-time clustering capabilities, essential for applications in fields like financial analysis, network security, and personalized recommendations.
- **Hybrid and Ensemble Methods**: Exploring hybrid and ensemble clustering methods that combine the strengths of multiple clustering techniques could lead to further improvements in clustering quality and the handling of complex data structures.

REFERENCES

1. **Analytics Vidhya. (2024).** Introduction to K-means Clustering Algorithm. Retrieved from https://www.analyticsvidhya.com/blog/2019/08/comprehensive-guide-k-means-clustering/
2. **Medium. (2024).** K-means Clustering: A Detailed Overview. Retrieved from https://medium.com/data-science-bootcamp/k-means-clustering-a-detailed-overview-d6b79214da73
3. **Kaggle. (2024).** K-means Clustering for Beginners. Retrieved from https://www.kaggle.com/code/faressayah/k-means-clustering-for-beginners
4. **Hartigan, J. A., & Wong, M. A. (1979).** Algorithm AS 136: A K-means clustering algorithm. *Journal of the Royal Statistical Society: Series C (Applied Statistics)*,28(1),100-108.
5. **Arthur, D., & Vassilvitskii, S. (2007).** K-means++: The advantages of careful seeding. *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*,1027-1035.
   DOI: 10.5555/1283383.1283494
6. **Aggarwal, C. C., & Reddy, C. K. (2023).** A Comprehensive Survey of Clustering Algorithms.*Springer. Available at: SpringerLink*

APPENDIX

Import Necessary Libraries

```
from sklearn.cluster import KMeans
import numpy as np
import seaborn as sns
from matplotlib import pyplot as plt
import warnings
warnings.filterwarnings("ignore")
%matplotlib inline
from sklearn.impute import SimpleImputer
from mpl_toolkits.mplot3d import Axes3D
```

```
# read data
df = pd.read_csv("C:/Users/aastha/Downloads/mini/Mall_Customers.csv")
df.head()
```

Missing Values Treatment

```
df.columns
Out[30]:

Index(['CustomerID', 'Gender', 'Age', 'Annual Income',
       'Spending Score (1-100)'],
      dtype='object')
```

```
#finding duplicates
df.duplicated().sum()
Out[32]:

0
In [33]:

df.isnull()
```

#finding missing values

df.isnull().sum()

Out[34]:

CustomerID	0
Gender	0
Age	4
Annual Income	0
Spending Score	0

dtype: int64

In [35]:

df.isnull().sum().sum()

Out[35]:

4

Filling Null Values

In [36]:

df1 = df.fillna(value = 0)

df1

Outlier Detection and Removal Treatment

In [38]:

columns_to_analyze = ['Age', 'Annual Income', 'Spending Score (1-100)']

 # Function to detect and remove outliers using IQR

def remove_outliers(df, columns):

  for col **in** columns:

    Q1 = df[col].quantile(0.25)

    Q3 = df[col].quantile(0.75)

 # Computing the IQR

    IQR = Q3 - Q1

```
#      Defining the lower and upper bounds
       lower_bound = Q1 - 1.5 * IQR
       upper_bound = Q3 + 1.5 * IQR

#      Filtering the DataFrame to remove outliers
       df = df[(df[col] >= lower_bound) & (df[col] <= upper_bound)]
   return df

df1_cleaned = remove_outliers(df1, columns_to_analyze)

print(df1_cleaned.head())
```

Normalization

In [39]:

```
scaler = MinMaxScaler()
df1_cleaned[columns_to_analyze]
scaler.fit_transform(df1_cleaned[columns_to_analyze])
print(df1_cleaned.head)
```

Elbow Method

In [40]:

```
# Extract the relevant columns
X1 = df1[['Age', 'Annual Income']].values

# Determine the range dynamically from the length of the data or any other criteria
max_clusters = min(len(df1), 15)

# Loop over the range of cluster values
for n in range(1, max_clusters + 1):
   algorithm = KMeans(n_clusters=n, init='k-means++', n_init=10, max_iter=300,
                      tol=0.0001, random_state=111, algorithm='elkan')
   algorithm.fit(X1)
   inertia.append(algorithm.inertia_)
```

 # Plot the inertia values

```
plt.figure(1, figsize=(15, 6))
plt.plot(np.arange(1, max_clusters + 1), inertia, 'o')
plt.plot(np.arange(1, max_clusters + 1), inertia, '-', alpha=0.5)
plt.xlabel('Number of Clusters')
plt.ylabel('Inertia')
plt.show()
```

Implementation Of K-Mean

In [41]:

```
plt.scatter(df1['Age'], df1['Annual Income'])
```

In [42]:

 # Define an imputer to fill in the missing values

```
imputer = SimpleImputer(strategy='mean')
```

 # Apply the imputer to the columns

```
df[['Age', 'Annual Income']] = imputer.fit_transform(df[['Age', 'Annual Income']])
```

 # Apply KMeans

```
km = KMeans(n_clusters=4)
y_predicted = km.fit_predict(df[['Age', 'Annual Income']])
print(y_predicted)
```

In [43]:

```
km.cluster_centers_
```

In [44]:

 # Perform clustering

```
km = KMeans(n_clusters=4)
df['cluster'] = km.fit_predict(df[['Age', 'Annual Income']])
```

```
# Create data subsets
df1 = df[df.cluster == 0]
df2 = df[df.cluster == 1]
df3 = df[df.cluster == 2]
df4 = df[df.cluster == 3]

# Plot clusters
plt.scatter(df1['Age'], df1['Annual Income'], color='yellow', label='Cluster 0')
plt.scatter(df2['Age'], df2['Annual Income'], color='red', label='Cluster 1')
plt.scatter(df3['Age'], df3['Annual Income'], color='blue', label='Cluster 2')
plt.scatter(df4['Age'], df4['Annual Income'], color='green', label='Cluster 3')

# Plot centroids
plt.scatter(km.cluster_centers_[:, 0], km.cluster_centers_[:, 1], color='purple',
marker='*', label='Centroid')

# Show plot
plt.xlabel('Age')
plt.ylabel('Annual Income')
plt.title('Cluster Plot')
plt.legend()
plt.show()
```

Evaluating the Effectiveness of K-means Clustering

```
# Calculate Sum of Squared Errors (SSE)
sse = 0
for i in range(len(df)):
    centroid = km.cluster_centers_[df.loc[i, 'cluster']]
    sse += np.sum((df.loc[i, ['Age', 'Annual Income']] - centroid) ** 2)
print(f"Sum of Squared Errors (SSE): {sse}")
In [46]:

# Perform clustering
km = KMeans(n_clusters=4)
```

```
df['cluster'] = km.fit_predict(df[['Age', 'Annual Income', 'Spending Score (1-100)']])

# Create data subsets
df1 = df[df.cluster == 0]
df2 = df[df.cluster == 1]
df3 = df[df.cluster == 2]
df4 = df[df.cluster == 3]

# Plot 3D scatter plot with first three principal components
plt.show()
```