A Novel Approach to Outlier-Aware K-means Clustering

Dhruv Chandna and Soham Jain

Thomas Jefferson High School for Science and Technology

TJ Machine Learning 2

Dr. Yilmaz

Quarter 2 Project

January 16, 2025

Abstract

Clustering algorithms like k-means often struggle to classify data containing outliers, as these extreme values can distort centroid calculations. To address this limitation, we developed an outlier-aware k-means algorithm that integrates both k-means and k-medians using a weighted mean. The weight is calculated based on the ratio of high outliers to the total size of the dataset, amplified by a cube root function to enhance k-medians' influence when extreme values are present. The model was evaluated on a synthetic dataset of 450 instances and two quantitative continuous features, generated using scikit-learn's make_blobs function. Compared to traditional k-means, the outlier-aware algorithm achieved higher accuracy (0.9044 vs. 0.8978), precision (0.9115 vs. 0.9065), recall (0.9140 vs. 0.9080), and F1-score (0.9044 vs. 0.8978). The results demonstrate that the adaptive weighting strategy effectively balanced outlier resistance and accurate centroid placement, leading to more consistent cluster boundaries. In the future, we will explore the scalability of this algorithm on higher-dimensional datasets and investigate other weighting strategies.

Introduction

Outlier detection is crucial in clustering algorithms, as they can significantly impact the performance of classification models. Traditional k-means clustering calculates centroids based on the mean of data points, making it highly sensitive to outliers. This sensitivity often leads to poor clustering performance when outliers are present, as centroids are skewed toward extreme values. Consequently, the resulting clusters may not accurately represent the underlying data distribution.

One common approach to mitigating the influence of outliers is to use k-medians clustering, which calculates centroids based on the median rather than the mean. This makes the model more robust to outliers, as the median is less affected by extreme values. However, while k-medians provides better resistance to outliers, it often struggles to accurately represent clusters with varying distributions. Specifically, it often results in poor centroid placement when clusters are unevenly distributed or have complex shapes. This trade-off between outlier resistance and accurate cluster representation presents a significant challenge in unsupervised learning.

This project introduces a novel approach that combines the strengths of both k-means and k-medians to effectively mitigate the influence of outliers without completely disregarding them. The input to our algorithm is a comma-separated value (CSV) file containing values for two quantitative continuous features, generated through scikit-learn's make_blobs function. We then use a weighted k-means clustering algorithm to output a binary classification (0 or 1), indicating the cluster assignment.

Related Work

Previous studies implemented k-means on datasets containing outliers, citing the algorithm's lack of resistance to extreme values as a limitation. For example, Song (2024) achieved an accuracy of 89% using k-means on the iris dataset. However, he noted a sensitivity to outliers and variance. This makes sense because the iris dataset includes five high outliers out of 150 total instances. These five points skew the centroids and result in worse overall performance. Similarly, Oyelade et al. (2010) applied k-means clustering to predict students' academic performance. They achieved a maximum cluster accuracy of 65% due to variance in the data, which k-means was not able to capture.

Other studies attempted to address k-means' sensitivity to outliers by completely removing them before applying the clustering algorithm. In fact, Zhang et al. (2020) did so, testing their model on six synthetic and three benchmark UCI and KEEL datasets. They found an improvement in Adjusted Rand Index (ARI), a metric for evaluating k-means, as well as an improvement in time. However, they noted that removing outliers resulted in a loss of information that may be necessary for real-world data. Likewise, Olukanmi and Twala (2017) used this approach to classify 10 different datasets, including Iris and Ruspini. They quantified an average of 90.2% accuracy and 86.8% precision. Although their results suggest relatively high performance, they also highlighted the trade-off between accuracy and generalizability as a limitation.

However, only a few studies in the past have attempted to integrate both k-means and k-medians to overcome this trade-off. One such example is Drias et al. (2016), who created their own "k-MM" clustering algorithm. It calls k-means, replaces each centroid with the closest object in the cluster, and then calls k-medians (which they call k-medoids). This approach resulted in a higher Normalized Mutual Information (NMI)—a metric used to evaluate clustering performance—of 0.72 compared to 0.69, demonstrating better clustering performance.

Our approach introduces a novel weighted centroid calculation that integrates both k-means and k-medians in a single step. Instead of assigning centroid locations by using k-means and implementing k-medians after it, we seek to simultaneously combine the advantages of both methods.

Dataset and Features

We created our own dataset using scikit-learn's make_blobs function, which generates synthetic data points in multidimensional space. In particular, our dataset consists of 450 instances and two quantitative continuous features, called "Feature 1" and "Feature 2." The dataset contains two large blobs of 200 instances each, corresponding to each of the two classes. There is another,

smaller cluster of 50 instances in between the other clusters, with points assigned to the class on the right. Hence, one class contains 200 instances while the other contains 250 instances.

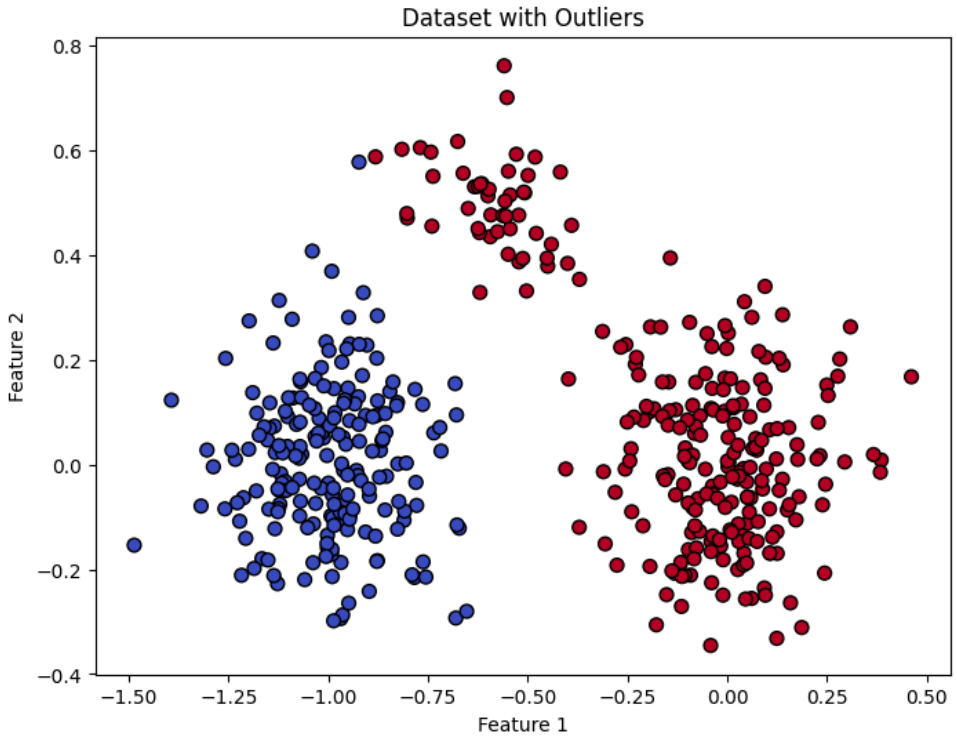

Fig. 1. Dataset constructed using scikit-learn's make_blobs function

Fig. 1 visualizes the two-dimensional dataset constructed using make_blobs. Although the data initially has labels, the model was constructed using unsupervised learning. Then, the performance was evaluated by comparing the predicted clusters to the original labels. Hence, we did not split the dataset into training, testing, or validation subsets.

Methods

Implementing k-means requires choosing a k value that corresponds to the number of centroids. Since our data contains two different labels (0 or 1) for binary classification, we set k = 2 for this problem. Next, traditional k-means implementation involves randomly selecting centroid positions, which we did by implementing Pandas' DataFrame.sample function.

Then, the algorithm calculates Euclidean distances between each data point and both centroids to figure out the closest cluster. All 450 data points are assigned to the nearest cluster. In k-means, the centroid location is updated with the mean of all data points assigned to that cluster. Conversely, k-medians updates the centroid with the median.

However, our proposed algorithm integrates k-means and k-medians clustering using a weighted mean. The new centroid is calculated as follows:

$$(x_f, y_f, \ldots) = (1 - \lambda) \times (x_{mean}, y_{mean}, \ldots) + \lambda \times (x_{median}, y_{median}, \ldots) \quad (1)$$

where:

- $(x_f, y_f, \ldots)$ are the updated centroid coordinates
- $(x_{mean}, y_{mean}, \ldots)$ is the centroid calculated using k-means
- $(x_{median}, y_{median}, \ldots)$ is the centroid calculated using k-medians
- λ is the weight that determines how much influence k-medians has

Since the weight is multiplied with the centroid calculated by k-medians, it should be higher when the dataset contains more outliers. So, when there are more extreme values, our algorithm gives more influence to k-medians. On the other hand, it gives higher weightage to k-means when there are fewer extreme values. The weight calculation is given as follows:

$$\lambda = \sqrt[3]{\frac{O}{N}} \quad (2)$$

where:

- O = total number of high outliers (whose value is greater than Q3 + 1.5 × IQR)
    - Q3 = third quartile, below which 75% of the data falls
    - IQR = interquartile range, calculated by taking the difference between Q3 and Q1
        - Q1 = first quartile, below which 25% of the data falls
- N = total number of instances in the dataset

We calculated the weight using the ratio between the number of high outliers and the total number of instances. We also amplified this ratio using a cube root function. Therefore, if the data contains 5% outliers, $\lambda = 0.368$ instead of 0.05, which gives more influence to k-medians.

It is important to note that O only represents the number of high outliers, and does not include low outliers. Low outliers are significantly closer to the median of the dataset and do not distort centroid calculations. Since high outliers are the extreme values that can skew the mean in k-means clustering, they are the primary focus of the weight calculation in our algorithm.

Results and Discussion

We created a custom performance metric called “homogeneity.” It measures the consistency of labels within each cluster and is defined as follows:

$$Homogeneity = \frac{Max\ count\ of\ label\ in\ C}{Total\ points\ in\ C} \quad (3)$$

Where C is a cluster.

Furthermore, we also evaluated our model on four other performance metrics: accuracy, precision, recall, and F1-score. The formulas for each of these are given below.

Accuracy:

$$\frac{TP + TN}{TP + TN + FP + FN} \quad (4)$$

Precision:

$$\frac{TP}{TP + FP} \quad (5)$$

Recall:

$$\frac{TP}{TP + FN} \quad (6)$$

F1-score:

$$\frac{2}{\frac{1}{Precision} + \frac{1}{Recall}} \quad (7)$$

where TP, TN, FP, and FN represent true positives, true negatives, false positives, and false negatives, respectively.

We evaluated both our algorithm and the k-means algorithm, as a benchmark, on the make_blobs dataset containing 450 instances. The table below shows the results that we achieved:

	K-means	Outlier-Aware K-means
Accuracy	0.8978	0.9044
Precision	0.9065	0.9115
Recall	0.9080	0.9140
F1-score	0.8978	0.9044
Cluster 1 Homogeneity	1.000	1.000
Cluster 2 Homogeneity	0.813	0.823

Table 1. Performance metrics for k-means and outlier-aware k-means

Notably, our outlier-aware k-means algorithm achieved better performance for all of the metrics that we measured.

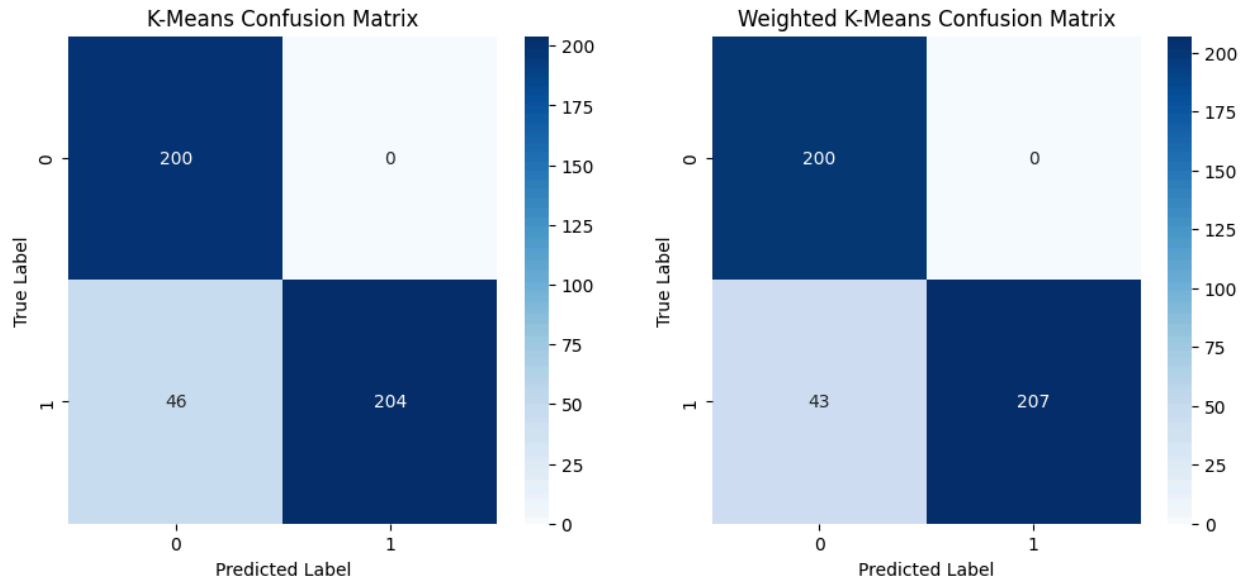

Fig. 3. Confusion matrices for k-means and weighted k-means

In order to understand why our model performed better than traditional k-means, we constructed the confusion matrices shown in Fig. 3. The label 0 corresponds to the blue class in Fig. 1 while the label 1 represents the red class. In both confusion matrices, it is clear that these clustering algorithms misclassify some of the instances in the blob between both clusters, where they predict label 0 when the true label is 1.

However, our algorithm classified more of these points accurately as the centroids were less resistant to outliers. Instead of these centroids skewing toward extreme values, a hybrid k-medians approach lead to more accurate cluster boundaries. Additionally, our model showed greater robustness in scenarios where traditional k-means struggled with cluster overlap. The improved homogeneity in Cluster 2 suggests that the adaptive weighting strategy effectively balanced the sensitivity to outliers with the need for accurate centroid placement.

Conclusion and Future Work

Our outlier-aware k-means algorithm effectively addresses the limitations of traditional k-means by integrating a weighted mean that balances the influence of outliers and maintains accurate cluster representation. This novel approach preserves data integrity while enhancing clustering performance, as demonstrated by improvements in precision, recall, and F1-scores. The homogeneity metric introduced in this study provides deeper insights into cluster consistency, revealing that our algorithm achieves more reliable and interpretable outcomes compared to other methods.

One limitation of our study is that the algorithm was only evaluated on a synthetic dataset with two quantitative features, limiting its generalizability to more complex, real-world datasets. In the future, we aim to test our model on more diverse, higher-dimensional datasets to evaluate its scalability. Moreover, we aim to explore more sophisticated weighting strategies, such as adaptive kernel methods and Fermat-Weber approximation, to further optimize outlier influence. Extending our approach to multi-class clustering problems and integrating semi-supervised learning techniques could also enhance cluster validation. We intend to investigate alternative distance metrics in the future as well, like Mahalanobis distance, to account for feature correlations.

Contributions

Soham: I worked on constructing the dataset with 450 instances using scikit-learn's make_blobs function. Furthermore, I identified related work to our project and noted limitations that we addressed in our research. I also came up with the formula for calculating the weight (λ), which involved taking the cube root of the outlier ratio to give more influence to k-medians. Finally, I evaluated accuracy, precision, recall, and F1-scores, and created confusion matrices to show the predicted vs. true labels for both our model as well as traditional k-means.

Dhruv: I focused on the methodology of our project by designing our algorithm, which takes the weighted average of k-means and k-medians to assign new centroids. Using our lecture notes and knowledge from previous labs, I wrote code to construct the k-means algorithm from scratch. Then, I modified it to implement k-medians and use the weight as a parameter. I also came up

with the homogeneity performance metric, which identifies how much of a cluster belongs to the same class. Additionally, I summarized our findings and proposed further improvements in the conclusion and future work sections.

References

Drias, H., Cherif, N., & Kechid, A. (2016). *k-MM: A Hybrid Clustering Algorithm Based on k-Means and k-Medoids*. ResearchGate. https://www.researchgate.net/publication/314632891_k-MM_A_Hybrid_Clustering_Algorithm_Based_on_k-Means_and_k-Medoids.

K means Clustering - Introduction. GeeksforGeeks. (2025). https://www.geeksforgeeks.org/k-means-clustering-introduction/.

K-Medians Clustering. Velog. (2024). https://velog.io/@greensox284/Clustering-K-Medians-Clustering.

make_blobs. Scikit-learn. (2025). https://scikit-learn.org/stable/modules/generated/sklearn.datasets.make_blobs.html.

Oyelade, O., Oladipupo, O., & Obagbuwa, I. (2010). *Application of k-Means Clustering algorithm for prediction of Students' Academic Performance*. arXiv. https://arxiv.org/pdf/1002.2425.

P. O. Olukanmi and B. Twala, "K-means-sharp: Modified centroid update for outlier-robust k-means clustering," *2017 Pattern Recognition Association of South Africa and Robotics and Mechatronics (PRASA-RobMech)*, Bloemfontein, South Africa, 2017, pp. 14-19, doi: 10.1109/RoboMech.2017.8261116.

pandas.DataFrame.sample. pandas. (2025). https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.sample.html.

sklearn.metrics. Scikit-learn. (2025). https://scikit-learn.org/stable/api/sklearn.metrics.html.

Song, J. (2024). *Research on clustering algorithms based on the Iris dataset*. ResearchGate. https://www.researchgate.net/publication/380208588_Research_on_clustering_algorithms_based_on_the_Iris_dataset.

Zhang, X., He, Y., Jin, Y., Qin, H., Azhar, M., & Huang, J. (2020). *A Robust k-Means Clustering Algorithm Based on Observation Point Mechanism*. Wiley Online Library. https://onlinelibrary.wiley.com/doi/10.1155/2020/3650926.