**Comprehensive Report on Inverted Binarization Clustering of Digit Data**

# Abstract

This study explores an innovative approach to clustering digit data, which involves the application of reversed binarization before dimensionality reduction and clustering. The conventional binarization, typically resulting in high contrast images with significant loss of detail, is modified by inverting the binarization threshold. This report discusses the methodological approach and the implications of dataset limitations on the clustering outcomes.

# 1. Introduction

Clustering stands as a pivotal process in the field of machine learning, often serving as the bedrock of pattern recognition and unsupervised data analysis. In the context of digit recognition, clustering aids in categorizing digit representations without prior labels. Traditional binarization methods are known for amplifying noise by converting grayscale images to binary contrast. Our methodological pivot focuses on a reversed binarization process, hypothesizing that it accentuates the key features of digits, minimizes noise, and aids in clearer cluster identification.

# **2. Methodology**: **2.1 Data Preprocessing**

Data was sourced from a CSV file, omitting the 'ID' column as it held no analytical value. A reversed binarization approach was adopted, converting lower grayscale values to '1' and higher values to '0', thus highlighting the digits and reducing noise.

# **2. Methodology**: 2.2 Dimensionality Reduction

PCA was utilized to capture 75% of the data's variance, followed by UMAP to preserve data topology while reducing dimensions for effective clustering.

# **2. Methodology**: 2.3 Clustering with KMedoids

KMedoids was selected for clustering due to its robustness to noise and ability to use actual data points as cluster centers. Ten clusters were targeted to represent the ten digits, initialized with 'k-medoids++' for efficient partitioning.

# 3. Results

Visualization via UMAP revealed clear separations among clusters with each digit ostensibly forming its own cluster, evidencing the efficacy of the reversed binarization. A score of 0.63776 was achieved in a Kaggle competition, reflecting limitations due to the dataset size.

# 4. Discussion

The reversed binarization method significantly improved clustering quality by emphasizing digit features and reducing noise. The size of the dataset emerged as a constraint, suggesting that a larger, more varied dataset could enhance performance and generalizability.

# 5. Conclusion

This report presented an unconventional preprocessing strategy for digit data clustering that outperformed traditional methods in terms of clarity and noise reduction. The size of the dataset was noted as a limiting factor, which was evident from the moderate score obtained in a Kaggle competition.

# 6. Future Work

To advance this research, several avenues are proposed:

* **Validation with Ground Truth**: To quantitatively assess clustering performance.
* **Algorithmic Experimentation**: Exploring other clustering algorithms for potential improvements.
* **Hyperparameter Optimization**: Fine-tuning UMAP and KMedoids parameters.
* **Comparative Binarization Studies**: Assessing traditional vs. inverted binarization methods.
* **Integration with Supervised Learning**: Refining clustering with a classification phase.
* **Robustness to Noise and Outliers**: Testing stability against induced noise.
* **Deep Learning Approaches**: Employing autoencoders for feature extraction.
* **Real-world Application**: Extending the methodology to tasks like postal code recognition.

# 7.different methods with the score coresponding to them

|  |  |  |
| --- | --- | --- |
| model | parameters | kaggle score |
| *KMedoids with binarization* | *n\_clusters=10, init='k-medoids++', metric="euclidean", random\_state=42, max\_iter=1000* | Score: 0.63776 |
| *AffinityPropagation* | *random\_state=42* | Score: 0.35169 |
| *KMedoids with no binarization* | *n\_clusters=10, init='k-medoids++', metric="euclidean", random\_state=42, max\_iter=1000* | Score: 0.60240 |
| *kmeans* | *n\_clusters=10, init='k-means++', random\_state=42, n\_init=10, max\_iter=300* | Score: 0.55544 |
| *kmeans* | *n\_clusters=10, random\_state=0* | Score: 0.37699 |
| kmeans | n\_clusters=10, n\_init=20, max\_iter=500, random\_state=42 | Score : 0.32449 |