|  |
| --- |
| Group 7 |
| Analysis Report |
| UMIST CROPPED DATASET |

|  |
| --- |
| Professor: Merlin James  12-12-2024 |

# Dataset Overview

The dataset used in this project comprises facial images of multiple individuals. Each image is labeled with a unique identifier corresponding to the person depicted. The dataset exhibits variability in lighting, pose, and expression, making it ideal for testing the robustness of clustering algorithms and classification models. The images are grayscale and resized to a uniform dimension of pixels to ensure consistency across all preprocessing and modeling steps.

However, the dataset presents a significant **class imbalance** issue, where certain individuals are overrepresented while others have fewer images. This imbalance poses challenges for training models, as it can lead to biased predictions that favor the dominant classes. To address this, we incorporated data augmentation techniques such as rotation, shifting, and zooming, creating synthetic samples for underrepresented classes. This step ensures a more equitable distribution of data across all classes, enhancing model performance and generalization.

# Preprocessing Techniques:

## Normalization

Pixel values were normalized to the range by dividing by 255. This ensures uniformity and accelerates convergence during training.

## Data Augmentation

Using ImageDataGenerator, we augmented the dataset to address class imbalance. The transformations included:

* Rotation (20 degrees)
* Width and Height Shifts (20%)
* Zoom and Shear Transformations

Data Augmentation made sure that every class has same amount of images to fix the Class Imbalance and we added variation to the images to make sure that they are not the same.

## PCA

The main goal behind using PCA was Dimensionality Reduction of the Database while keeping 99 percent of the variance explained. This ensures that the important information is not lost while also significantly reducing dimensions.

Due to PCA, We were able to ***bring down the dimension of images from 10800 features to only 400 features*** which significantly reduces the processing time helping us save computing resources and time.

## CNN for Feature Extraction:

Using CNNs for feature extraction ensures that relevant spatial information is captured effectively. The model learns hierarchical feature representations, which are critical for distinguishing subtle differences between classes in the dataset. This approach eliminates the need for manual feature engineering and is well-suited for image data, where relationships between pixels are vital.

By using the Last Layer of CNN which has formed hierarchical relationships between pixels helps the KMEANS or any other clustering algorithm to map the clusters better.

Here is the Model Summary of our CNN:

A screenshot of a computer program

Description automatically generated

## Autoencoders

An autoencoder is used to extract meaningful features from high-dimensional image data, specifically from the UMIST faces dataset. The primary goal of using an autoencoder is to learn a compressed, lower-dimensional representation (bottleneck) of the input data that captures the most important features, while discarding irrelevant noise.

By training the model to reconstruct the input images, the autoencoder identifies patterns in the data that are useful for downstream tasks, such as clustering or classification. The model architecture includes several dense layers with Leaky ReLU activations and L2 regularization to prevent overfitting, while using Mean Squared Error (MSE) loss for reconstruction accuracy.

PCA is first applied to reduce the dimensionality of the dataset before training the autoencoder, enhancing the model’s efficiency and performance**. This method is particularly effective for reducing computational complexity and improving the quality of feature extraction, which is crucial for further tasks like clustering with Gaussian Mixture Models (GMM) and classification with a neural network**. The use of an autoencoder thus plays a critical role in dimensionality reduction, feature learning, and noise elimination in image-based datasets.

# Clustering Methods

## KMEANS:

**Rationale:**  
K-Means is widely used because it is simple and efficient, making it ideal for large datasets. The algorithm works by grouping data points into clusters based on how close they are to the center of the cluster (called a centroid). The intuition behind K-Means is that we want to minimize the differences within each cluster and maximize the differences between clusters. It's most effective when the clusters are roughly spherical and evenly distributed.

**Parameter Tuning:**  
The optimal number of clusters is chosen using the **silhouette score**, which measures how similar each point is to its own cluster versus other clusters. A high silhouette score indicates well-formed, distinct clusters.

**Clusters:**

The number of Clusters formed in KMEANS depended on Preprocessing we did. Three of our Team members used KMEANS with different Preprocessing techniques, Each team mate got different optimal clusters determined by the methods such as silhouette score and the Elbow method.

|  |  |
| --- | --- |
| Preprocessing Technique | Clusters |
| PCA | 19 |
| CNN | 28 |
| PCA + Autoencoders | 24 |

# DBSCAN

**Rationale:**  
Unlike K-Means, DBSCAN doesn’t require specifying the number of clusters. Instead, it groups together points that are close to each other based on a density criterion. This works well when clusters have irregular shapes or varying densities. DBSCAN can also detect noise—data points that don’t belong to any cluster. The intuition here is that dense regions of data should form clusters, while sparse areas are considered noise.

**Parameter Tuning:**  
The main parameters for DBSCAN are **ε (epsilon)**, the radius of a neighborhood around each point, and **MinPts**, the minimum number of points needed to form a cluster. These are tuned using the **k-distance graph**, where we look for the distance at which points start to become more isolated, helping us identify the best radius for clustering.

**Clusters**

One of our team members used DBCSAN and for different epsilon values, he calculated the silhouette scores of the cluster created. He found that the best Silhouette score was being formed for **33 Clusters**. This was done with PCA as a preprocessing step for Dimensionality Reduction.

## GMM

**Rationale:**  
GMM is a probabilistic model that assumes the data is generated from a mixture of several Gaussian distributions. Unlike K-Means, which assigns each point to one cluster, GMM provides probabilities for a point to belong to different clusters. This is especially useful when clusters overlap or have different shapes. Since GMM considers the distribution of data points rather than just their distances from centroids, it can handle more complex cluster structures.

**Parameter Tuning:**  
We used the **Akaike Information Criterion (AIC)** to determine the optimal number of clusters, which turned out to be 15. AIC is preferred over BIC in this case because it is more focused on model fit, without over-penalizing the number of parameters. Since GMM uses multiple parameters to model each cluster, AIC strikes a good balance between fitting the model to the data and avoiding overfitting. A lower AIC value suggests a better model, making it ideal for choosing the right number of clusters in this case.

# H-DBSCAN

**Rationale:**  
HDBSCAN (Hierarchical DBSCAN) extends DBSCAN by providing a hierarchical clustering structure, which allows it to identify clusters at varying densities. It overcomes DBSCAN's limitation of a single density parameter by considering varying densities across different parts of the data. This makes HDBSCAN particularly effective for datasets with clusters that exhibit significant variance in density. Additionally, HDBSCAN does not require specifying the number of clusters in advance and can identify noise as well. The model works by first constructing a hierarchy of clusters and then selecting the most stable clusters based on their persistence across the hierarchy.  
**Parameter Tuning:**  
The key parameters for HDBSCAN are the minimum cluster size and the minimum samples required for a point to be considered as part of a cluster. These parameters are tuned by evaluating the stability of clusters at different levels of the hierarchy. Our team applied a Convolutional Neural Network (CNN) as a preprocessing step for dimensionality reduction before applying HDBSCAN. **Through this approach, the best results were achieved with 28 clusters, effectively capturing the underlying structure of the data.**

# Neural Network Classifier Accuracy for Different Combinations of Preprocessing and Clustering:

|  |  |
| --- | --- |
| **Preprocessing and Clustering** | **Test Accuracy** |
| PCA + KMEANS | 0.991150438785553 |
| CNN + KMEANS | 0.7396 |
| AUTOENCODERS+PCA + GMM | 0.8495575189590454 |
| DBSCAN + PCA | 0.982300877571106 |
| CNN + HDBSCAN | 0.9323 |

# Conclusion

In this project, we explored different clustering techniques and preprocessing methods to better understand and organize facial image data from the UMIST faces dataset. This dataset posed several challenges, such as class imbalance, lighting variations, and facial expressions, making it an ideal test case for evaluating the effectiveness of various approaches.

**Clustering Insights:**  
The results from the clustering techniques helped us understand how the dataset is structured. For K-Means, the number of clusters varied depending on the preprocessing method used. We found 19 clusters when PCA was applied, 28 clusters when using CNN, and 24 clusters with a combination of PCA and autoencoders. These differences show how preprocessing techniques affect the clustering results, with CNN-based features likely providing more detailed representations of the data. DBSCAN, which doesn’t require a set number of clusters, identified 33 clusters when tuned with PCA, demonstrating its ability to handle irregular cluster shapes and noise well. The Gaussian Mixture Model (GMM) produced results with 15 clusters, with the Akaike Information Criterion (AIC) guiding the model selection to balance fit and complexity. HDBSCAN, which provides hierarchical clustering, also identified 28 clusters with CNN as a preprocessing step. This further emphasizes the importance of feature extraction in improving clustering results.

**Preprocessing and Feature Extraction:**  
The preprocessing techniques we used, like PCA, Data Augmentation, and Autoencoders, played a big role in improving the performance of our models. PCA helped reduce the high-dimensional image data while keeping 99% of the variance, making the data more manageable and reducing computational costs. The Autoencoders also helped refine the feature extraction process by creating a compressed, meaningful representation of the data, making clustering and classification easier. Additionally, Data Augmentation addressed the class imbalance by generating synthetic samples for underrepresented individuals, leading to a more balanced dataset and making our models more robust.

**Clustering and Classification Performance:**  
When we combined the clustering results with neural network classifiers, we saw that **PCA + K-Means** achieved the highest accuracy of 99.11%. This combination worked the best, likely because PCA reduced the data's dimensions effectively, and K-Means clustered the data well. **CNN + K-Means** and **Autoencoders + PCA + GMM** had lower accuracies—73.96% and 84.96% respectively. This suggests that while CNNs and autoencoders were good for extracting features, GMM might not have handled the complex cluster structures of this dataset as well, which might have caused the lower accuracy. On the other hand, **DBSCAN + PCA** showed a strong accuracy of 98.23%, highlighting DBSCAN’s strength in identifying clusters without needing a set number of clusters, especially when combined with PCA for dimensionality reduction. **CNN + HDBSCAN** achieved an accuracy of **93.23%**, showcasing the potential of combining CNN-based feature extraction with hierarchical clustering.

The biggest surprise came from the fact that **CNN + K-Means** resulted in lower accuracy (73.96%) than **PCA + K-Means** (99.11%), despite CNN being widely recognized for image classification tasks. This anomaly could be due to the nature of the dataset, where CNN’s feature extraction did not lead to better clustering results for this particular scenario, or due to the specific way CNN-based features were used in conjunction with K-Means.

**Overall Summary:**  
This project demonstrated the impact of various preprocessing techniques and clustering methods on facial image data. While **K-Means with PCA** gave the best classification accuracy, other methods like **DBSCAN**, **GMM**, and **HDBSCAN** also showed strengths in handling complex datasets. Using **Autoencoders with PCA** improved feature extraction and clustering, leading to better overall results. The project suggests that a hybrid approach—combining different techniques—can be useful for building more robust clustering and classification models.

In conclusion, the performance of each method depended on the dataset’s characteristics and the preprocessing steps applied. By addressing challenges like class imbalance and computational complexity, we improved both the clustering quality and the accuracy of our classifiers. This highlights the importance of carefully selecting preprocessing methods and clustering algorithms to tackle real-world machine learning problems.