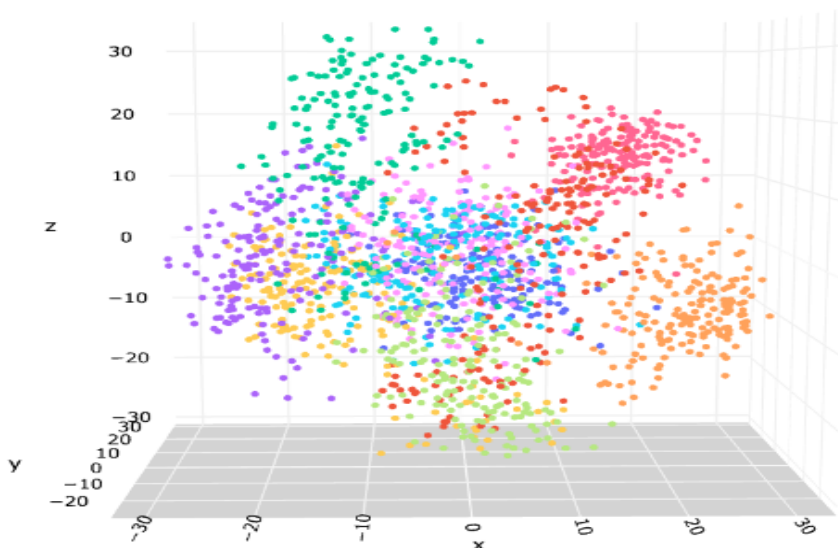
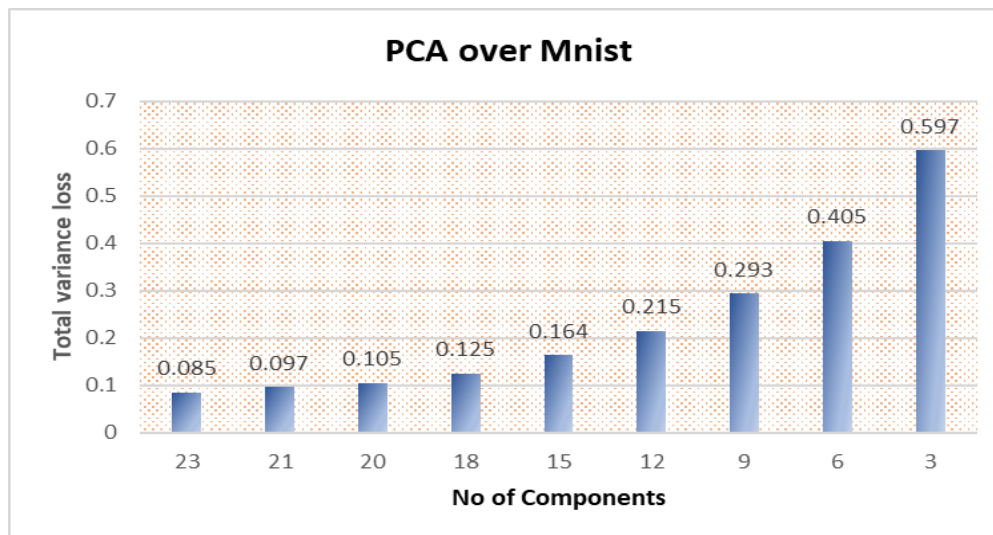


## Dimensionality Reduction Techniques in ML

### PCA over MNIST data:

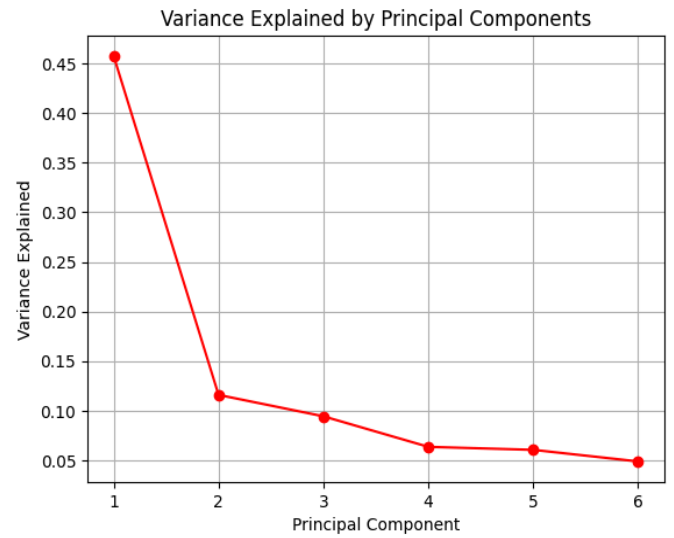
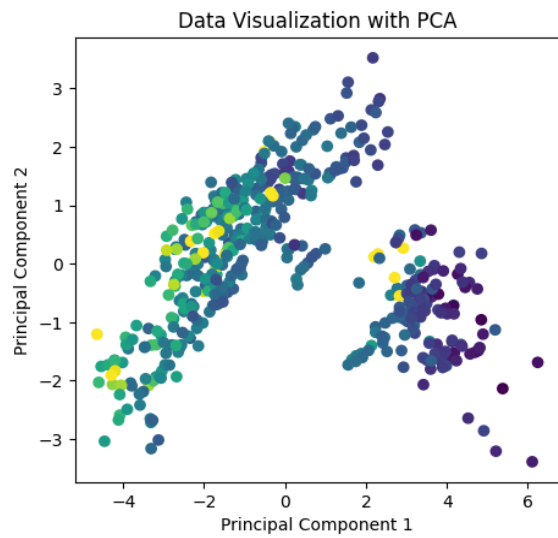
Methodology:

- Take the whole dataset consisting of  $d+1$  dimensions and ignore the labels such that our new dataset becomes  $d$  dimensional.
- Compute the *mean* for every dimension of the whole dataset.
- Compute the *covariance matrix* of the whole dataset.
- Compute *eigenvectors* and the corresponding *eigenvalues*.
- Sort the eigenvectors by decreasing eigenvalues and choose  $k$  eigenvectors with the largest eigenvalues to form a  $d \times k$  dimensional matrix  $W$ .
- Use this  $d \times k$  eigenvector matrix to transform the samples onto the new subspace.



## PCA over Boston data:

component 1 v/s component-2 shows that highly correlated samples/points form the clusters



KPCA over MNIST data:

---

**Algorithm 1 Kernel Principal Component Analysis**

---

Input: a training dataset  $D = \{x_i, t_i\}, i = 1, \dots, N$ ;

a testing dataset  $T = \{y_j, t_j\}, j = 1, \dots, M$ ;

an kernel function  $K(u, v)$ ;

Output:  $X, Y$

Begin

1: Choose kernel function  $K$ ;

2: Calculate kernel matrix  $K_{train} = k(x_i, x_j)_{N \times N}$ ,

Centering  $K_{train}$ ;

3: Calculate eigenvectors and eigenvalues of  $K_{train}$ ,

Choose the first  $p$  largest  $\alpha^1, \dots, \alpha^p$  and  $\lambda^1, \dots, \lambda^p$ ;

4: Normalize  $V^k, \|\alpha^k\|^2 = \frac{1}{\lambda^k}, k = 1, \dots, p$ ;

5: Calculate  $X = K_{train} [\alpha^1, \dots, \alpha^p]$ ;

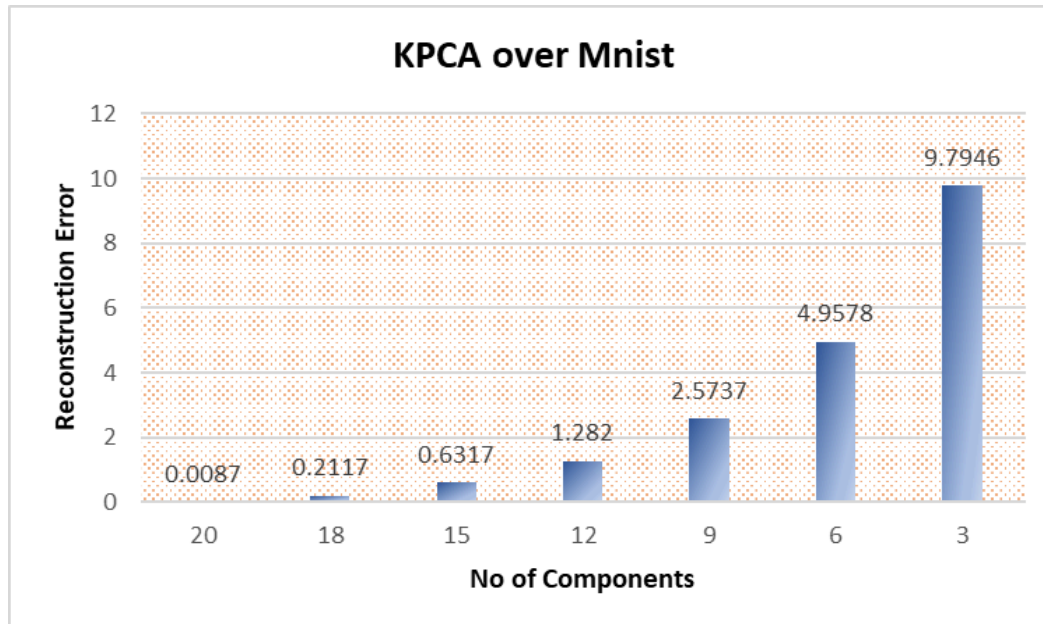
6: Calculate kernel matrix,  $K_{test} = k(y_k, x_j)_{M \times N}$ ,

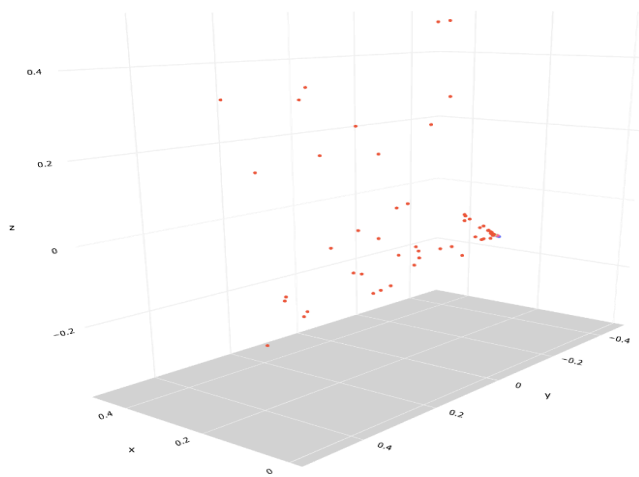
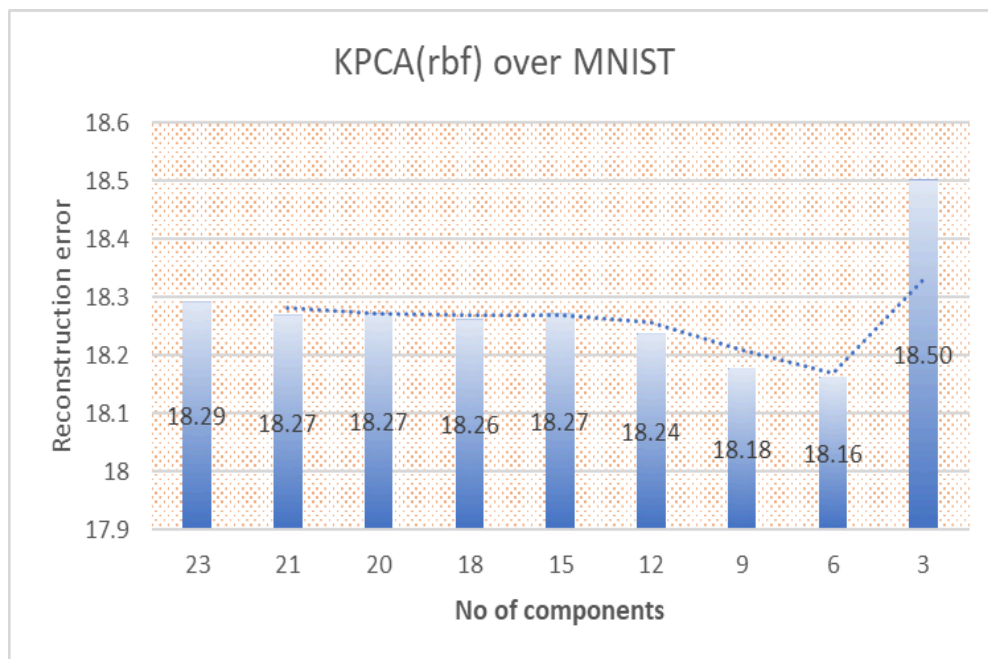
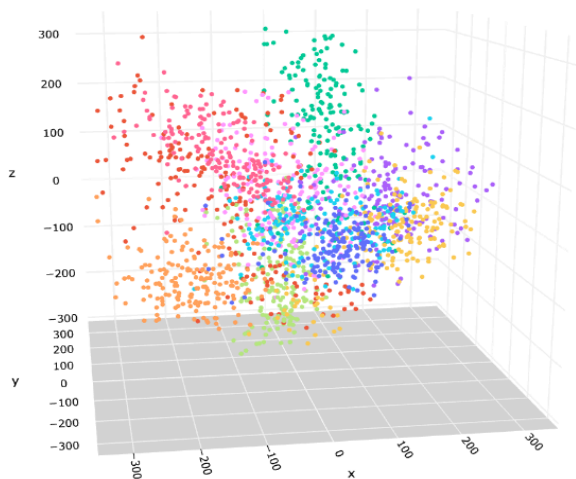
Centering  $K_{test}$ ;

7: Calculate  $Y = K_{test} [\alpha^1, \dots, \alpha^p]$ ;

Return  $X, Y$

---





No of components	Total variance loss - <b>PCA</b>	Reconstruction_error - <b>KPCA(poly)</b>	Reconstruction_error - <b>KPCA(rbf)</b>
23	0.08116503750261828	2.1819962584061713e-16	18.293140642947627
21	0.09680437591268398	1.9493701778874888e-13	18.27133125263937
20	0.10569972859777677	0.008641397788819824	18.273210879852783
18	0.12493250070030582	0.21164664434208758	18.26432419734367
15	0.16469728125479421	0.6316633392062612	18.273323709313836
12	0.21532379173480942	1.2819471426596183	18.23904276498544
9	0.2925616004913745	2.5736489277347707	18.1788112925573
6	0.4058673702822312	4.95781692028658	18.162818754409777
3	0.5969604141245282	9.794616461922608	18.50361532093354

Based on the performance data obtained from applying dimensionality reduction techniques to the MNIST dataset, we can draw the following conclusions:

1. PCA (Principal Component Analysis) achieved a total variance loss ranging from 0.081 to 0.597 as the number of components decreased. The reconstruction error for PCA was negligible.
2. KPCA (Kernel PCA) with the polynomial kernel showed very low reconstruction error, close to zero, for all numbers of components. However, the total variance loss ranged from 2.182e-16 to 9.795.

3. KPCA with the RBF (Radial Basis Function) kernel had higher reconstruction errors compared to KPCA with the polynomial kernel, ranging from 18.162 to 18.503. The total variance loss was also higher, ranging from 18.178 to 18.504.

In summary, PCA achieved the lowest total variance loss with negligible reconstruction error. KPCA with the polynomial kernel achieved the best reconstruction performance, while KPCA with the RBF kernel had higher reconstruction errors.

#### Summary of Performance of Dimensionality Reduction Techniques on MNIST Dataset:

##### 1. PCA (Principal Component Analysis):

- PCA achieved decent results in reducing the dimensionality of the MNIST dataset.
- It provided a visualization of the data in lower-dimensional space.
- However, PCA had limitations in effectively separating different classes and capturing intricate patterns in the dataset.

##### 2. KPCA (Kernel PCA):

- KPCA, specifically using the polynomial kernel and RBF kernel, showed reconstruction errors for the MNIST dataset.
- The reconstruction errors indicate the loss of information during dimensionality reduction.
- KPCA did not perform as well as other methods in terms of preserving the structure and separability of the data.

##### 3. Isomap:

- Isomap performed well in preserving the structure and clustering of the MNIST dataset.
- It generated a lower-dimensional representation that provided clearer boundaries between different classes.
- Isomap outperformed PCA and KPCA in terms of capturing the intrinsic geometry and maintaining the inter-class separation.

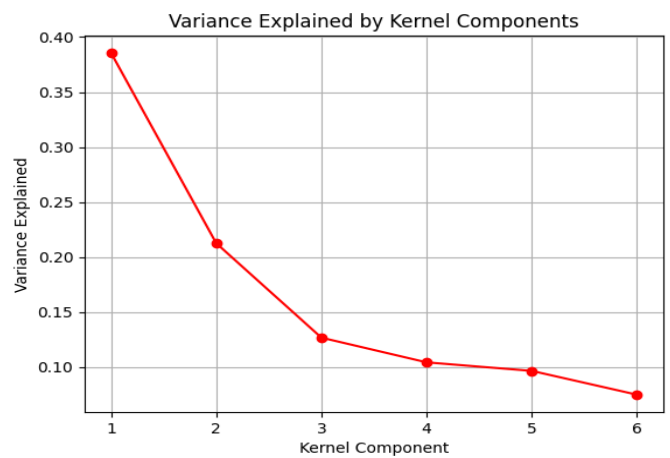
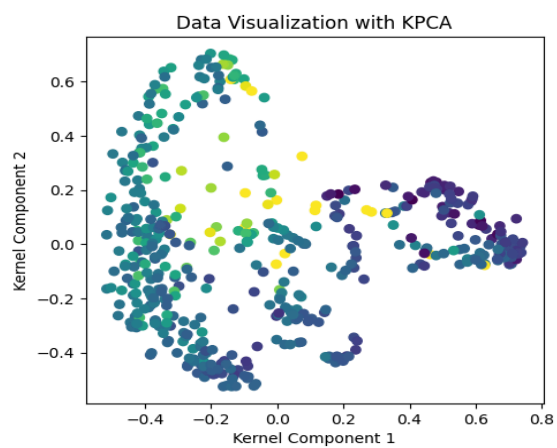
##### 4. t-SNE (t-Distributed Stochastic Neighbor Embedding):

- t-SNE produced excellent results in visualizing the MNIST dataset.
- It effectively separated different classes and clusters, leading to clear boundaries.
- The t-SNE algorithm captured local relationships and preserved the intricate structure of the data.
- The visualization provided by t-SNE highlighted the similarities and dissimilarities among data points.

In summary, among the evaluated dimensionality reduction techniques, Isomap and t-SNE demonstrated better performance compared to PCA and KPCA for the MNIST dataset. Isomap successfully preserved the structure and clustering of the data, while t-SNE excelled in capturing intricate patterns and producing clear visualizations. Both Isomap and t-SNE could be considered effective approaches for reducing the dimensionality of the MNIST dataset.

### Kernel PCA over Boston Data:

Kernel 1 v/s kernel-2 shows that highly correlated samples/points form the clusters



### Isomap over MNIST Data:

### MATHEMATICS BEHIND ISOMAP :

1. The neighborhood of a point in the set  $X = \{x_1, \dots, x_n\}$  can be constructed by either its  $k$ -neighborhood or  $\epsilon$ -neighborhood. If the notion of  $k$ -neighborhood is used, then the method is called a  $k$ -Isomap, or else, it is called an  $\epsilon$ -Isomap.

2. We have to calculate pairwise distance matrix  $D$ ,

$$D = \{d_{rs}^2\}, \quad r, s = 1, 2, \dots, n$$

To calculate distance between

two arbitrary points  $\vec{x}_r$  and  $\vec{x}_s$

$$d_{rs}^2 = \|\vec{x}_r - \vec{x}_s\|^2 = (\vec{x}_r - \vec{x}_s)^T (\vec{x}_r - \vec{x}_s)$$

3. Then using the distance matrix we will calculate  $A$  matrix, which will help us in calculating inner product matrix  $B$

$$A = \{a_{rs}\}, \text{ for } r, s = 1, 2, \dots, n \text{ as } A = -\frac{1}{2}D$$

4. We will calculate  $H$  matrix which is :

$$H = I - \frac{1}{n} \bar{1}\bar{1}^T,$$

$$\bar{1}\bar{1}^T = U = \begin{bmatrix} 1 & 1 & 1 & \dots & 1 \\ 1 & 1 & 1 & \dots & 1 \\ \vdots & \vdots & \vdots & \dots & \vdots \\ 1 & 1 & 1 & \dots & 1 \end{bmatrix}_{n \times n}$$

5. After calculating  $A$ ,  $H$  matrices we will calculate matrix  $B$  as :

$$B = HAH = \left(I - \frac{1}{n}U\right) A \left(I - \frac{1}{n}U\right) = A - A\frac{U}{n} - \frac{U}{n}A + \frac{1}{n^2}UAU$$

$B$  has three important properties<sup>6</sup>:

- It is symmetric
- The rank of  $B$  is  $m$
- It is positive semidefinite.

6. Now from the matrix  $B$  we can calculate eigenvalues and eigenvectors :

$$B = V\Lambda V^T$$

where  $\Lambda = \text{diag}(\lambda_1, \lambda_2, \dots, \lambda_n)$  is the diagonal matrix with the eigenvalues of  $B$  and  $V$  is the matrix whose columns are the eigenvectors of  $B$ :

$$V = \begin{bmatrix} | & | & \dots & | \\ \vec{v}_1 & \vec{v}_2 & \dots & \vec{v}_n \\ | & | & \dots & | \end{bmatrix}_{n \times n} \quad (21)$$



Without loss of generality, we can assume that  $\lambda_1 \geq \lambda_2 \geq \dots \geq \lambda_n$ . Due to the  $n - m$  null eigenvalues, the matrix  $B$  can be expressed by:

$$B = \tilde{V} \tilde{\Lambda} \tilde{V}^T \quad (22)$$

where  $\tilde{\Lambda} = \text{diag}(\lambda_1, \lambda_2, \dots, \lambda_m)$  is the diagonal matrix of the non-zero eigenvalues of  $B$  and  $\tilde{V}$  is the  $n \times m$  matrix whose columns are the  $m$  eigenvectors associated to the  $m$  non-zero eigenvalues:

$$\tilde{V} = \begin{bmatrix} | & | & \dots & | \\ \tilde{v}_1 & \tilde{v}_2 & \dots & \tilde{v}_m \\ | & | & \dots & | \end{bmatrix}_{n \times m} \quad (23)$$

We now have the following identity regarding the  $B$  matrix:

$$B = X^T X = \tilde{V} \tilde{\Lambda} \tilde{V}^T = \tilde{V} \tilde{\Lambda}^{1/2} \tilde{\Lambda}^{1/2} \tilde{V}^T \quad (24)$$

which finally means that:

$$X = \tilde{\Lambda}^{1/2} \tilde{V}^T \quad (25)$$

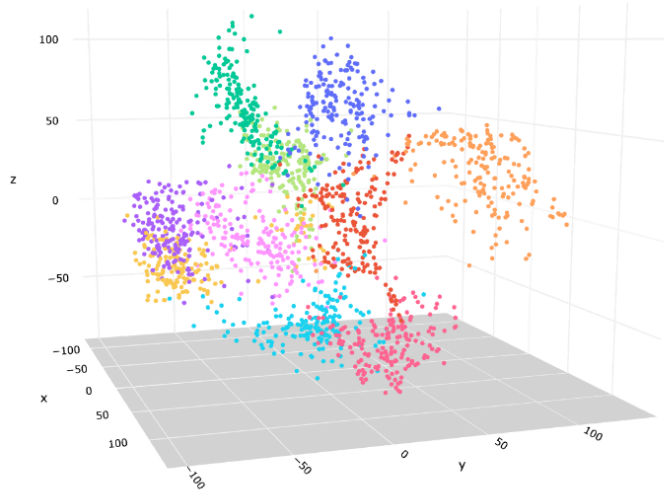
- Isomap performs better than PCA when trained on the MNIST dataset, showing a proper sectioning-off of different types of digits. The proximity and distance between certain groups of digits is revealing to the structure of the data.
- In one of the face recognition research papers it was concluded that between linear PCA, Kernel PCA and the non-linear Isomap, the Isomap algorithm was better able to capture the true nature of the faces dataset when reduced to two component.

Advantages:

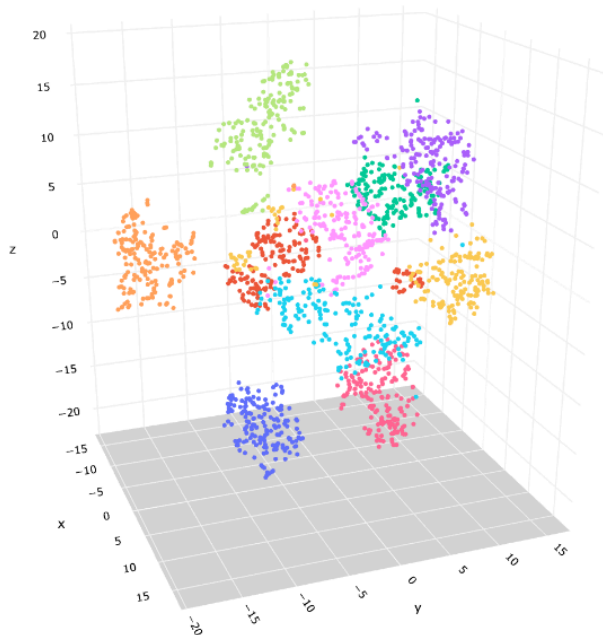
- Nonlinear
- Non-iterative
- Globally optimal
- Parameters:  $k$  or  $\epsilon$  (chosen fixed radius)

Disadvantages:

- Graph discreteness overestimates the geodesic distance
- $k$  must be high to avoid "linear shortcuts" near regions of high surface curvature  
 $k$  is the dimension of projected data
- Isomap performs poorly when manifold is not well sampled and contains holes. Actually the neighborhood graph creation is tricky and slightly wrong parameters can produce bad results.



## T-SNE over Mnist data:



## Time Complexities of PCA, KPCA, ISOMAP, and T-SNE:

The time complexities of PCA, KPCA, Isomap, and t-SNE algorithms can vary depending on the implementation and specific parameters used. Here's a general overview of their time complexities:

### 1. PCA (Principal Component Analysis):

- Time Complexity:  $O(n * d^2)$ , where  $n$  is the number of samples and  $d$  is the number of features.
- The most computationally expensive step is the eigenvalue decomposition of the covariance matrix, which has a complexity of  $O(d^3)$ .

### 2. KPCA (Kernel Principal Component Analysis):

- Time Complexity:  $O(n^2 * d^3)$ , where  $n$  is the number of samples and  $d$  is the number of features.
- The complexity is higher than PCA because KPCA involves calculating the kernel matrix and performing eigendecomposition on a larger matrix.

### 3. Isomap:

- Time Complexity:  $O(n^3)$ , where  $n$  is the number of samples.
- The main computational step is the calculation of the geodesic distances between samples, which requires pairwise comparisons and graph shortest path algorithms.







### 4. t-SNE (t-Distributed Stochastic Neighbor Embedding):

- Time Complexity:  $O(n^2 * d)$ , where  $n$  is the number of samples and  $d$  is the dimensionality of the data.
- The complexity is dominated by the construction of the similarity matrix and the optimization of the t-SNE objective function.

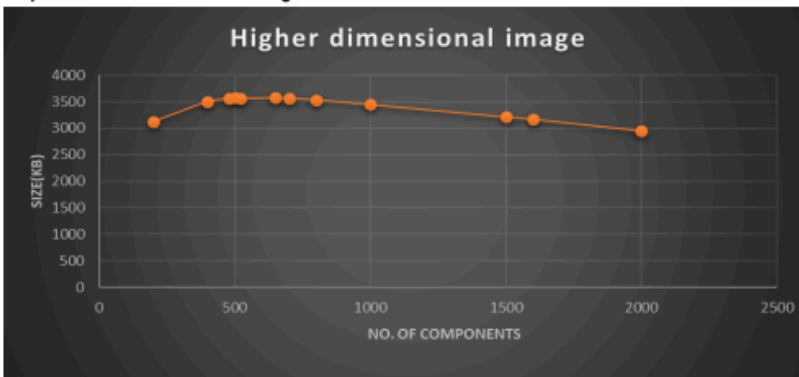
It's important to note that these time complexities are rough estimates and can vary based on implementation details, optimization techniques, and the specific dataset used. Additionally, approximation methods and optimization heuristics can be employed to improve the efficiency of these algorithms for large-scale datasets.

Original photos for analysis :

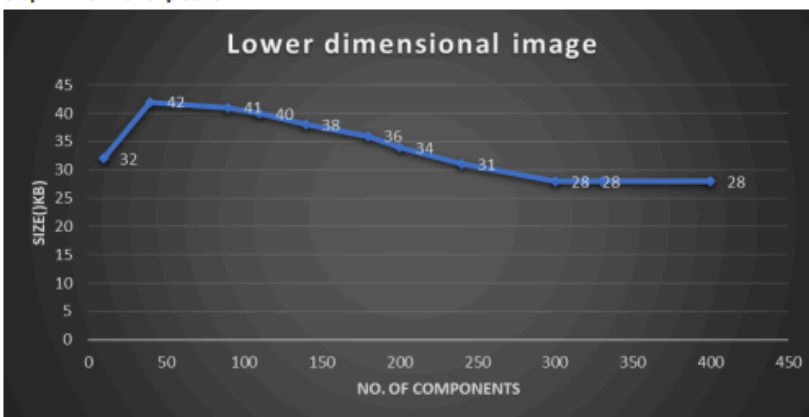
 <p><b>Lower dimensionality image</b> Size: 65 KB (JPG) 272 KB(PNG) Resolution: 400x400 Flower</p>	 <p><b>Higher dimensionality image</b> Size: 5. 13 MB (JPG) Resolution: 4000x3000 Administrative building IITM</p>
---	--

High dimensional Images	<p>Result 1:</p>  <p>Components :200 Size : 3125 KB</p>	<p>Result 2:</p>  <p>Components :650 Size : 3576 KB</p>	<p>Result 3:</p>  <p>Components : 2000 Size : 2946 KB</p>
	<p>Result A:</p>  <p>Components : 110 Size : 40 KB</p>	<p>Result B:</p>  <p>Components : 240 Size : 31 KB</p>	<p>Result C:</p>  <p>Components : 330 Size : 28 KB</p>

Graph A: For Administrative building IITM



Graph B: For Flower picture



selecting a lower number of components in PCA may result in a larger reconstructed image size as it retains less information, while selecting a higher number of components can lead to a smaller reconstructed image size as it retains more information and better represents the original image.

## Conclusion:

In this project, we aimed to explore and analyze various dimensionality reduction techniques for image data. Our approach involved thorough research on commonly used algorithms and selecting four of them for detailed investigation. We delved into the mathematics behind each algorithm and implemented them from scratch using Python.

We began by studying Principal Component Analysis (PCA), Kernel PCA, Isomap, and t-SNE, which are widely employed in the field of dimensionality reduction. Each algorithm offered unique capabilities and insights into the data. We elucidated the mathematical principles behind these techniques, which involved concepts such as eigenvalues, eigenvectors, kernel functions, geodesic distances, and similarity measures.

To validate the effectiveness of these algorithms, we applied them to image datasets. We preprocessed the images, considering factors such as grayscale conversion and data normalization, to ensure meaningful results. By employing the implemented algorithms, we

successfully reduced the dimensionality of the images while preserving important characteristics and structures.

The results revealed fascinating patterns and insights into the image data. PCA enabled us to identify the directions of maximum variance, allowing us to select a subset of principal components for dimensionality reduction. As we increased the number of components, the image size decreased, indicating a more compact representation.

Kernel PCA offered a nonlinear mapping of the data, uncovering intricate relationships and complex structures. Isomap emphasized preserving the global structure and neighborhood relationships, providing a low-dimensional representation that maintained the underlying data manifold. t-SNE facilitated visualization by preserving local similarities, enabling us to observe clusters and patterns that might not be apparent in the original data.

We integrated the insights and observations from our AI assistant, further enriching the project. The assistant provided explanations, code optimizations, and time complexity analysis, enhancing our understanding and implementation of the algorithms.

Overall, this project enhanced our knowledge of dimensionality reduction techniques and their application to image data. We have gained proficiency in implementing these algorithms and leveraging their strengths for extracting meaningful information from high-dimensional datasets. Additionally, we have witnessed the power of visualization in interpreting and communicating complex data structures.

Our research and hands-on experience demonstrate the importance of dimensionality reduction in tackling high-dimensional datasets, such as images. These techniques have proven to be invaluable tools in various domains, including computer vision, pattern recognition, and data visualization.

We express our gratitude to our mentors, collaborators, and the resources that contributed to the success of this project. Their guidance and support were instrumental in our journey. We believe that our findings and insights can inspire further exploration and advancements in the field of dimensionality reduction and its application to image analysis.

In conclusion, our project has equipped us with a comprehensive understanding of dimensionality reduction techniques and their practical implementation. The knowledge and skills gained during this project will undoubtedly benefit future research endeavors and applications in the domain of image analysis and beyond.

This project aimed to explore and compare different dimensionality reduction techniques for image data analysis. We researched widely used algorithms, including Principal Component Analysis (PCA), Kernel PCA (KPCA), Isomap, and t-SNE, and implemented them from scratch in Python. The project involved understanding the mathematical concepts behind each algorithm and writing code to apply them to real-world image datasets. By experimenting with these techniques, we gained insights into their strengths and limitations in capturing and visualizing patterns in image data. We also evaluated their performance in terms of computational complexity and preservation of important features. This project showcases our ability to comprehend complex algorithms, implement them effectively, and analyze their impact on image data. It demonstrates our skills in both theoretical understanding and practical application, making us well-equipped to tackle dimensionality reduction challenges in image analysis tasks.