GE - 461

Introduction to Data Science

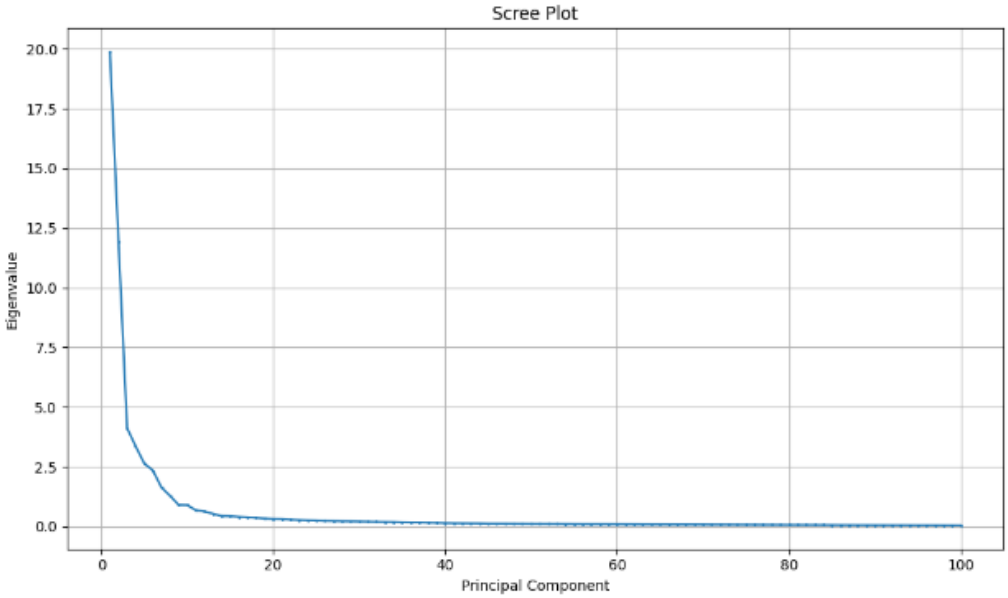
Project 2

Bahadır Yüzlü

7 April , 2024

In preparation for our analysis, I initiated by downloading the Fashion-MNIST dataset, which originally comprises 70,000 images. However, for our study, I chose a subset containing 10,000 images, ensuring a manageable dataset size conducive to our experimentation. To maintain a balanced representation across classes, I divided the data into training and testing sets with a 50/50 split [1]. This equitably distributed 500 images from each of the 10 classes into both the training and testing sets, supporting a robust evaluation framework. To facilitate processing, I flattened the images from their original 28 by 28 dimensions to a single vector of 784 elements. Additionally, I applied a rotation to the images. As a visual representation, an example image of a shirt from the dataset is depicted below, providing a glimpse into the Fashion-MNIST dataset.

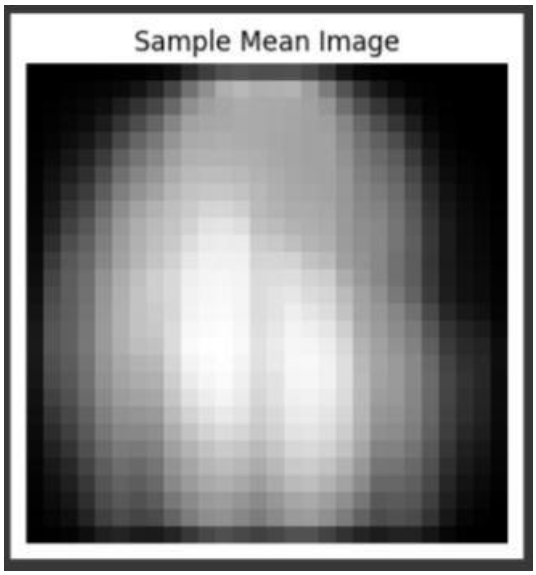
The data was centered by subtracting the mean of the entire dataset from each sample. This centering process ensures that the data is balanced around zero, which is a prerequisite for principal component analysis (PCA) to be effective. After centering the data, principal component analysis (PCA) of sklearn.decomposition. PCA [2] was applied to the training dataset consisting of 5,000 samples. PCA helps to identify the principal components that capture the maximum variance in the dataset, effectively reducing the dimensionality of the data while retaining important information. Following PCA, the eigenvalues associated with each principal component were calculated. These eigenvalues represent the amount of variance explained by each principal component. The eigenvalues were plotted in descending order, as shown in the figure below.



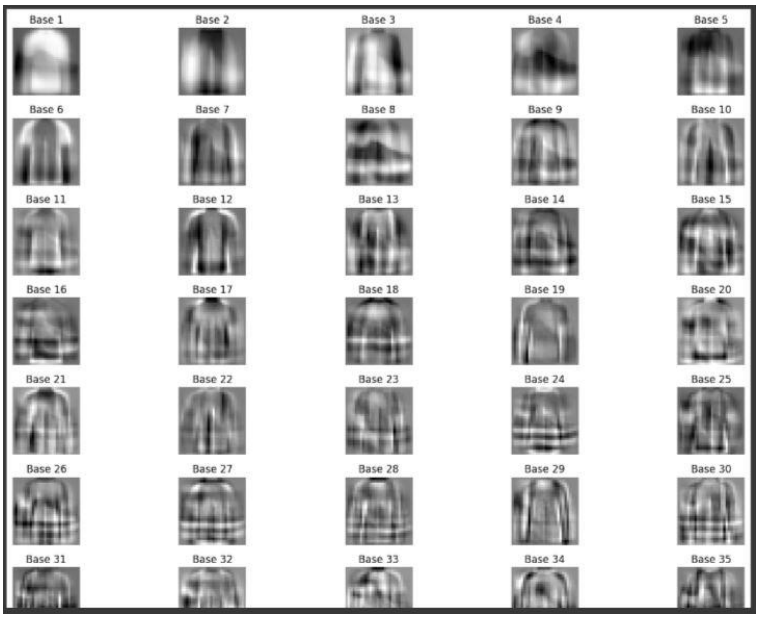
By examining the plot of eigenvalues, it is possible to determine the optimal number of components to retain. Typically, a 'elbow' or a sharp drop-off in the eigenvalue plot indicates the point

at which adding additional components does not significantly increase the explained variance. Therefore, the number of components chosen can be determined based on this criterion. According to this explanation, it should be around 10 components.

After computing the sample mean image and the eigenvectors, I proceeded to visualize and analyze these key components of our dataset. The sample mean image provides an average representation of pixel values across all samples in the training dataset. Upon examination, I observed that I noticed a concentration of white pixels at the center. This observation confirms the centralized nature of the dataset's features. These findings are consistent with our initial expectations. Here is the figure;



Subsequently, I visualized the eigenvectors obtained through principal component analysis (PCA) as images. Each eigenvector represents a principal component capturing the maximum variance in the dataset. Upon inspection of these eigenvector images, I observed that certain class silhouettes, such as shoes, dresses, and t-shirts, overlap in the middle.



I proceeded to create different subspace dimensions ranging from 1 to 400 with an interval of 16. Utilizing this set, Principal Component Analysis (PCA) was applied to the entire dataset consisting of

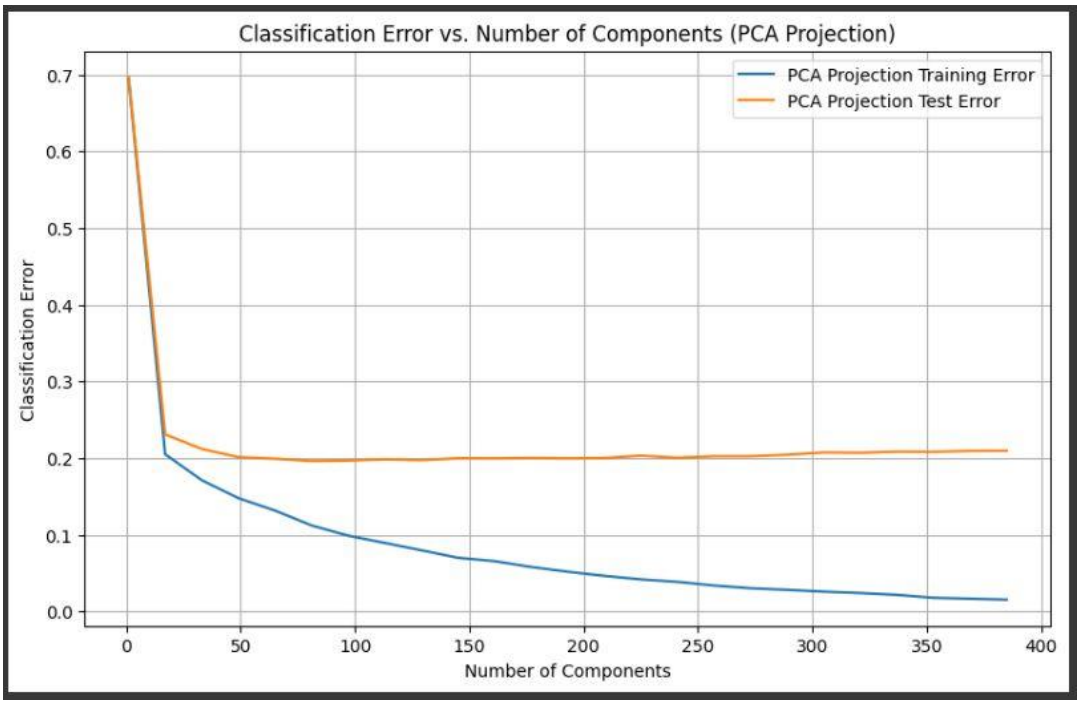
10,000 images. Subsequently, the reduced versions of the data corresponding to each subspace dimension were saved using a list structure. This approach allowed us to systematically explore the effects of varying subspace dimensions on the dimensionality-reduced representations of the Fashion-MNIST dataset.

For the Gaussian classifier, two key functions were implemented. The first function was responsible for training the class mean matrix and the class covariance matrix. This function ensured that the classifier could learn the underlying distribution of each class in the dataset. Additionally, a regularization coefficient was incorporated into the covariance matrix to prevent singularity, which could lead to errors during computation.

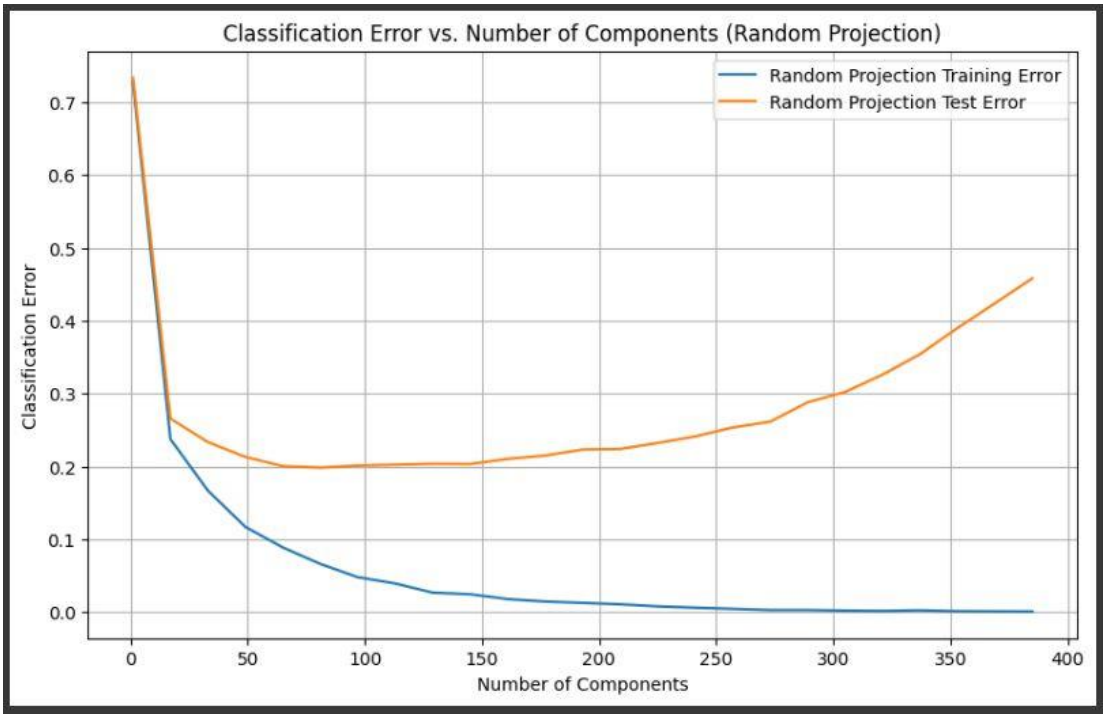
The second function was designed to calculate the probability density function of the **quadratic Gaussian distribution** [3, 4], as outlined in the document. This function enabled the classifier to determine the probabilities of an image belonging to each class based on its features. Subsequently, the maximum probability was selected as the prediction for the class label of the image.

During the classification process, both the training and test data were utilized. This approach allowed for the evaluation of the model's performance on unseen data and facilitated the visualization of the error scores. Specifically, the error scores of both the test and train sets were plotted on a graph to observe their trends and assess the classifier's performance over iterations.

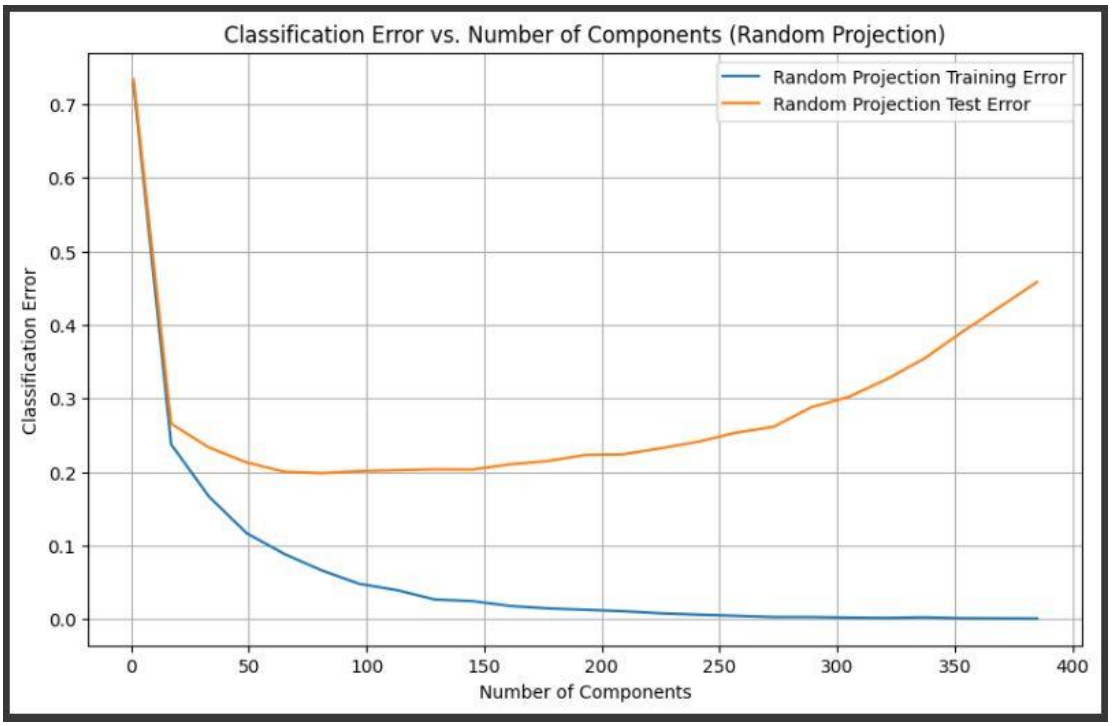
To quantify the performance of the classifier, the accuracy matrix was calculated using the **accuracy_score** function from the **sklearn.metrics** library [5].



From the observed graph, it becomes evident that the first 14 components are significantly important for dimension reduction. These components capture essential information crucial for effective representation of the Fashion-MNIST dataset. However, beyond approximately 50 components, the marginal gains in reducing test error diminish, indicating diminishing returns on increasing dimensionality.

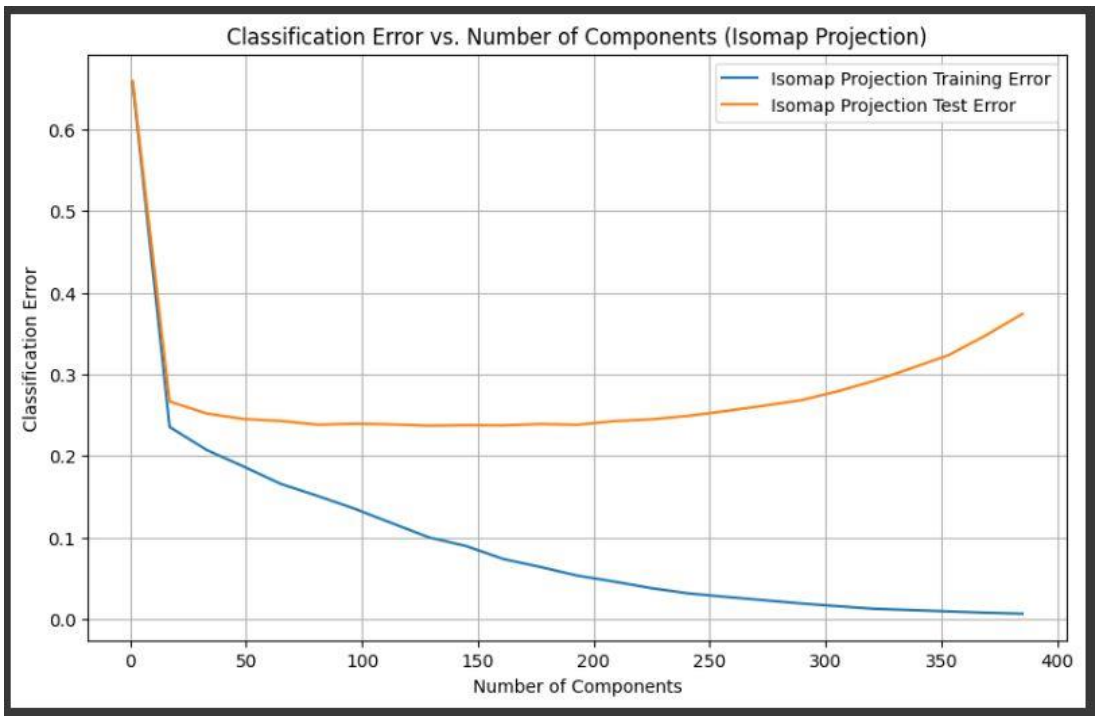


In the random matrix generation phase, the **np.random** library was utilized to generate random eigenvectors. These random vectors were then multiplied with the image data to project the dataset into a reduced-dimensional space. Surprisingly, the results obtained from this random projection were comparable to those achieved with PCA up to a certain point. Specifically, the projection analysis suggests that around 65 dimensions suffice for effective dimensionality reduction.

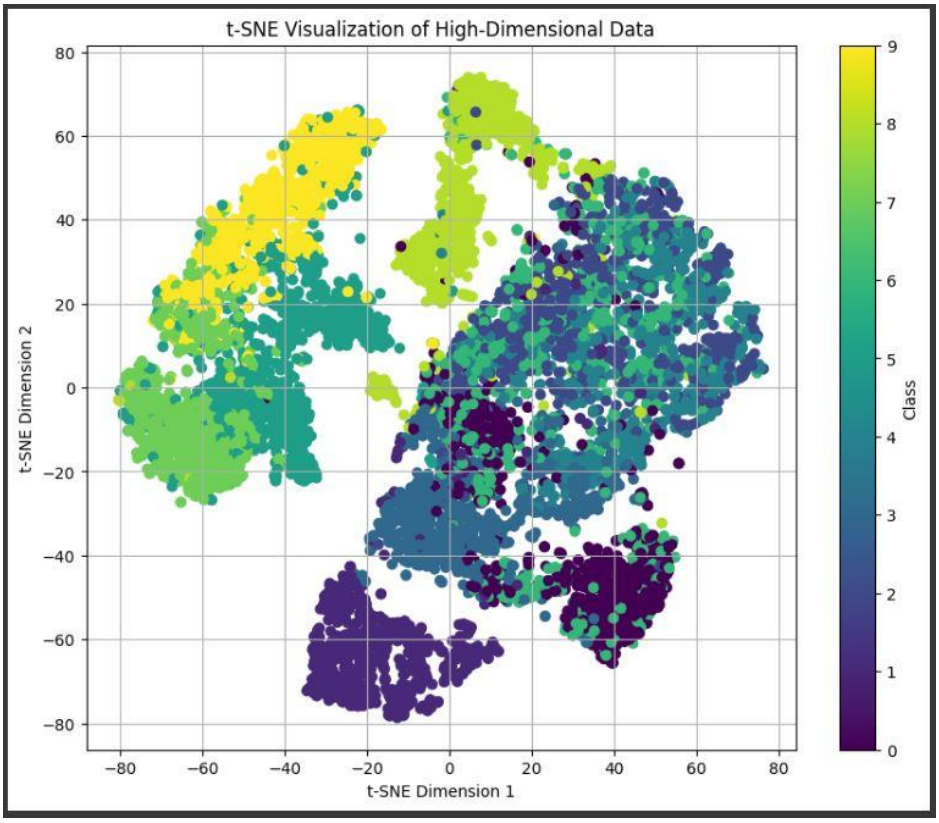


This observation highlights the potential effectiveness of random projection techniques as an alternative to traditional dimensionality reduction methods like PCA. By leveraging random matrices, it is possible to achieve significant reductions in dimensionality with minimal loss of information.

For this question, Isomap projection will be done by sklearn's manifold learning library [6]. During the application of Isomap projection, overfitting became apparent, particularly with higher dimensions, approximately 400, as evidenced by excessively low training errors, around 0.007. Despite experimenting with various parameters such as adjusting **n_neighbors**, **eigen_solver**, **path_method**, and **neighbors_algorithm** to simplify the model and mitigate overfitting, consistent results were obtained in higher dimensions. However, convergence between test and train errors was observed in relatively lower dimensions. The final parameter set selected included **n_neighbors = 10** to control the number of neighbors considered, **eigen_solver = 'arpack'** for eigenvalue decomposition, **path_method = 'D'** for geodesic distance calculation using the shortest path algorithm, and **neighbors_algorithm = 'ball_tree'** for efficient neighbor search. This parameter configuration was chosen for its lower solving time and convergence occurring at around 24% error approximately, striking a balance between model complexity and generalization performance.



Lastly, t-distributed Stochastic Neighbor Embedding (t-SNE) is applied by the use of sklearn's manifold library [6]. I employed t-SNE with tailored parameter settings. Setting **n_components** to 2 allowed for the projection of high-dimensional data onto a two-dimensional space, enhancing interpretability. A perplexity value of 30 struck a balance between capturing local and global structures, promoting coherent clustering in the visualization. With **n_iter** set to 1000, adequate iterations were ensured for comprehensive exploration of the data space, facilitating convergence without excessive computational demands. Additionally, fixing **random_state** at 42 ensured result reproducibility, maintaining consistency in visualizations across different runs.


t-SNE Visualization of High-Dimensional Data
t-SNE Dimension 1
t-SNE Dimension 2
Class
80
60
40
20
0
−20
−40
−60
−80
9
8
7
6
5
4
3
2
1
0

Reference List

[1] “Sklearn.model_selection.train_test_split,” scikit, https://scikitlearn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html (accessed Apr. 4, 2024).

[2] “SKLEARN.DECOMPOSITION.PCA,” scikit, https://scikitlearn.org/stable/modules/generated/sklearn.decomposition.PCA.html (accessed Apr. 6, 2024).

[3] “Mma, “Univariate/multivariate gaussian distribution and their properties,” Mustafa Murat ARAT, https://mmuratarat.github.io/2019-10-05/univariate-multivariate_gaussian (accessed Apr. 7, 2024).

[4] “Statistical Functions (scipy.stats),” Statistical functions (scipy.stats) - SciPy v1.13.0 Manual, https://docs.scipy.org/doc/scipy/reference/stats.html (accessed Apr. 6, 2024).

[5] “3.3. metrics and scoring: Quantifying the quality of predictions,” scikit, https://scikit-learn.org/stable/modules/model_evaluation.html (accessed Apr. 7, 2024).

[6] “2.2. Manifold Learning,” scikit, https://scikit-learn.org/stable/modules/manifold.html# (accessed Apr. 7, 2024).