Bilkent University

GE-461

Introduction to Data Science

Spring 2024

Project 2: Dimensionality Reduction and Visualization Report

Kaan Tek

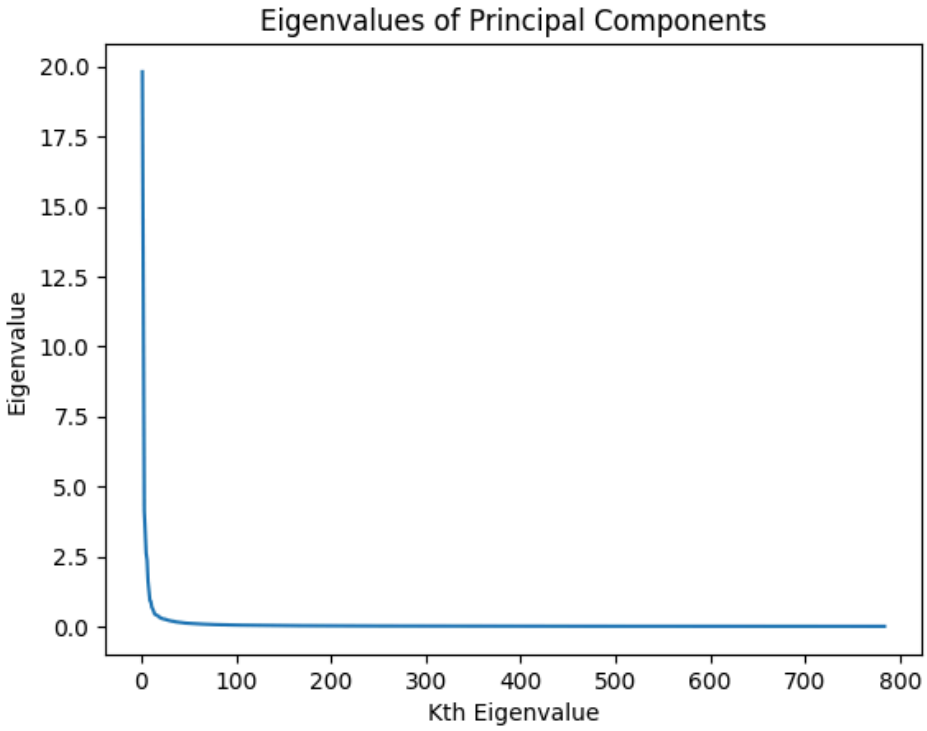
21901946

Question 1)

In this first part, I applied the PCA technique to transform the data from 784 dimensions to lower dimensional subspaces, in order to observe the impact of dimensionality on the Gaussian classifier's effectiveness.

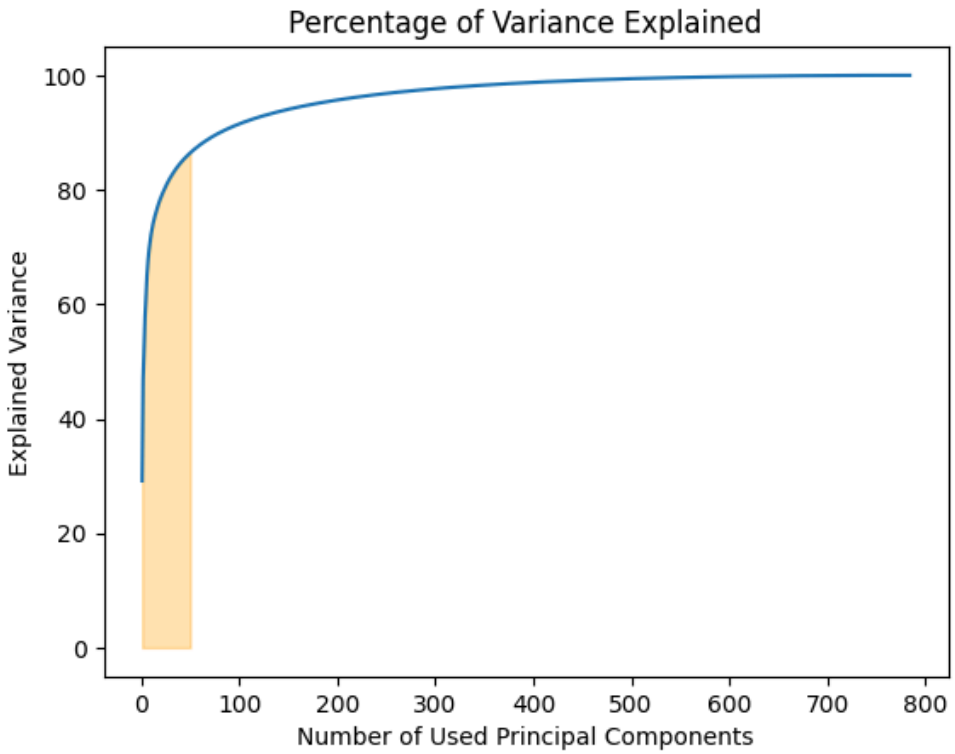
Question 1.1 & 1.2)

I used *PCA* and *QuadraticDiscriminantAnalysis* libraries of scikit-learn, former in order to project my centered (mean-subtracted) data onto a lower dimensional space, and latter in order to fit a Gaussian that makes estimations in parallel to the formulas given in the project description [1, 2]. The *explained_variance_* and *explained_variance_ratio_* attributes of the PCA class are also useful in order to easily access the amount of variance explained by each selected component [1].



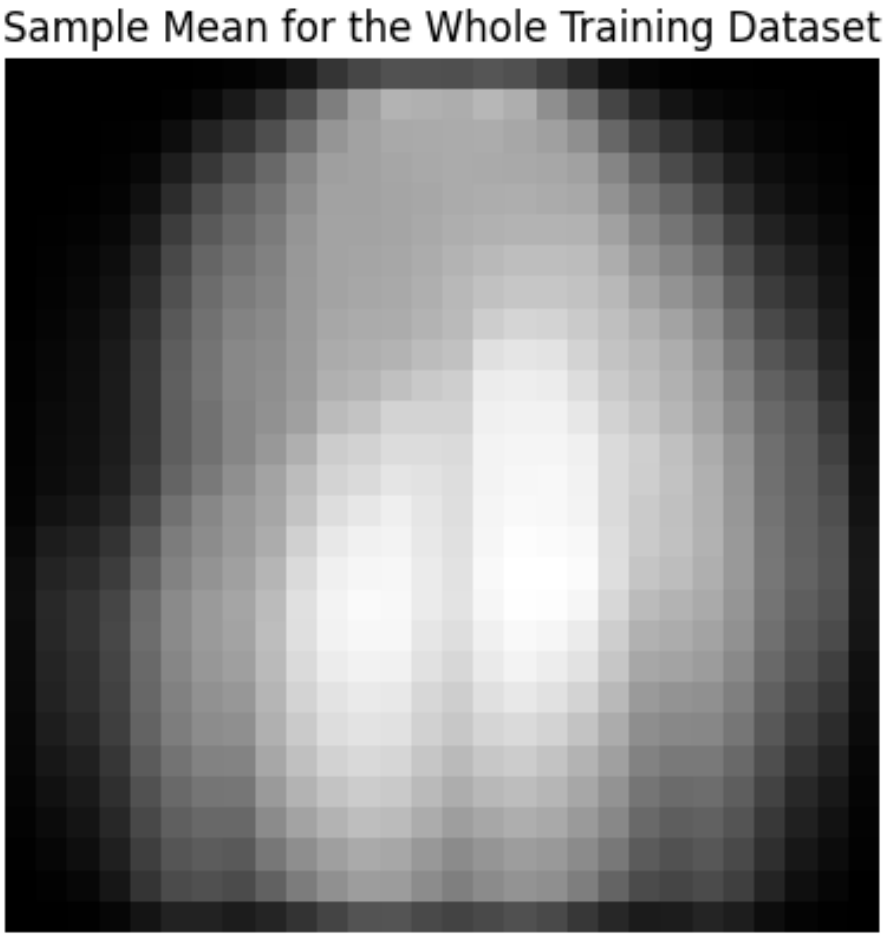
The above figure was generated by plotting our eigenvectors (principal components) based on their eigenvalues. These eigenvalues are indicative of the variance each principal component accounts for. Observing the plot, it is evident that the initial few principal components possess significantly larger eigenvalues. The decline in eigenvalues is non-linear, with a rapid decrease observed until the ~40th principal component. Since the 100th principal component, it can be said that the eigenvalues gradually converge to zero. Hence, from the plot, selecting the first 50 eigenvectors seems like a solid choice for reducing PCA dimensions.

Below plot would give us a better idea, since it represents the Cumulative Proportion of Variance Explained (PVE) for using increasing number of principal components:

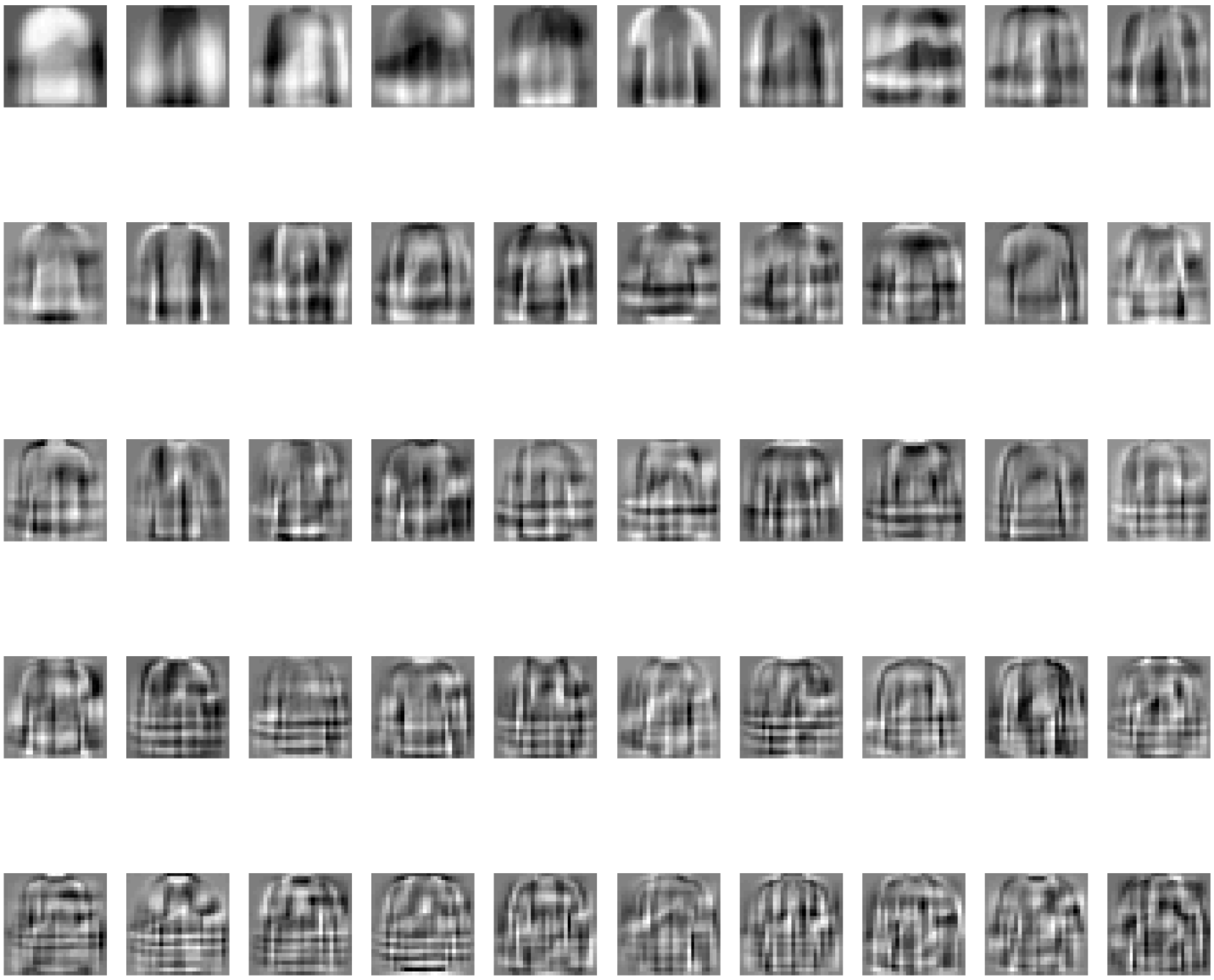


Observing the above plot, the cumulative variance explained by the first 50 eigenvectors reveals that they account for almost 90% of the total variance. This percentage signifies that we can maintain most of the data's information with just 1/16 of the total eigenvectors, indicating an adequate dimensionality reduction for our analytical needs. Therefore, using about 50 principal components would seem to be a reasonable number.

Question 1.3)



First 50 PCs



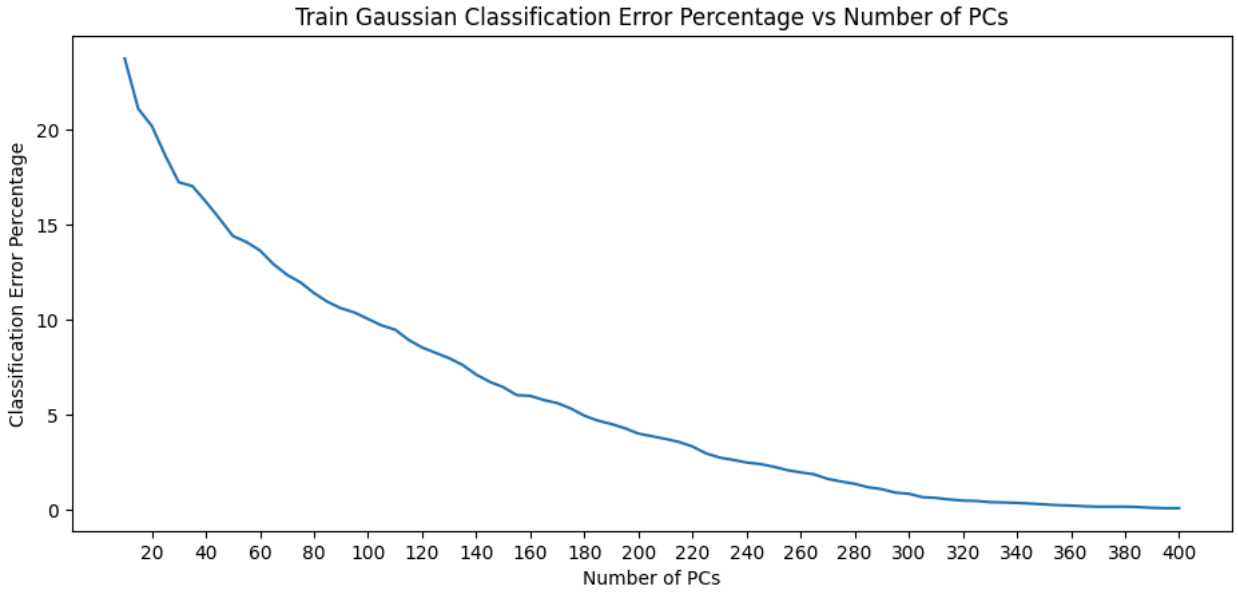
I visualized the mean of all the data as above. At first glance, one might think that the mean image is basically a blurred image that does not represent anything, however, that assumption would be wrong. The mean image is basically the mean of all pixels, so it basically represents a blurred

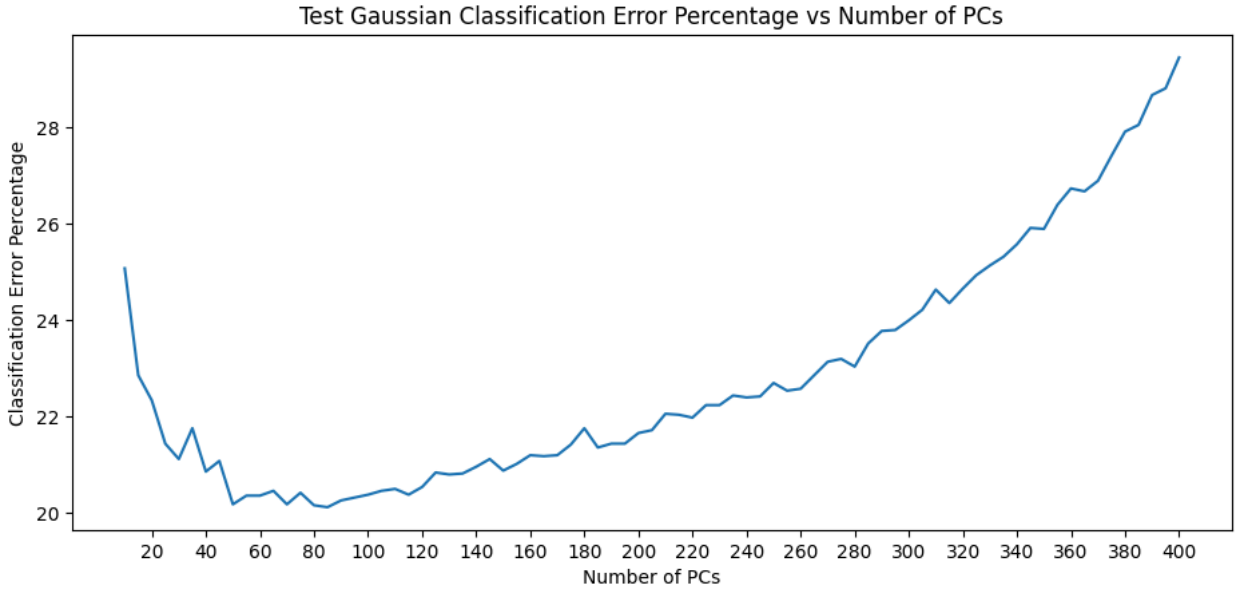
composite clothing image. The borders and corners of the mean image are simply black pixels, which again makes sense since the original images in the train data have those black pixels around the cloth images as well. Additionally, the center of the mean is somewhat a representation of the combination of all clothing images, upon examining, one can notice a coat-like image and sleeves.

In the realm of principal components, the leading ones encapsulate most of the variance, suggesting that they would represent the clothing's fundamental shapes. The eigenvalues decrease across the image grid, with the upper images, especially the first principal component, depicting basic forms like sneakers (shoes), and shirts. Subsequent images, such as the second principal component, capture additional details like trousers. It is seen in most of the images to have a short or long sleeved tops which makes sense since most of the generic clothing items such as t-shirts, coats, and dresses contain sleeves in common.

Deeper in the sequence, the images become increasingly abstract, likely highlighting minor nuances in the samples, such as unique curves and outlines. The dominance of short-sleeved tops in these representations is consistent with their variance significance, reflecting their foundational presence in the dataset's spectrum. As principal components with lesser variance are examined, they tend to reveal more detailed features of the images, diverging from the general structure.

Question 1.4 & 1.5)





The first plot above reveals a consistent increase in training accuracy with the inclusion of more principal components, where the classification error nearly converges to zero with nearly 400 components. This occurs because principal components represent maximum variance directions

within the data, and adding more of them integrates comprehensive information about the data's structure into the model. Consequently, the model's complexity increases, enabling it to identify subtle data patterns and variations, thus enhancing training set accuracy due to a more precise data fit.

The data from the second plot recommend around 50 principal components as ideal for attaining peak test accuracy in our model. Using a very low number of components might not capture enough of the variance which obviously results in poor performance, whereas an excessive count may lead to overfitting, weakening the model’s adaptability to unfamiliar data. Overall, the findings indicate the importance of finding a balance in selecting a sufficient number of components to encapsulate the data's important features while preventing underfitting and overfitting of the training data.

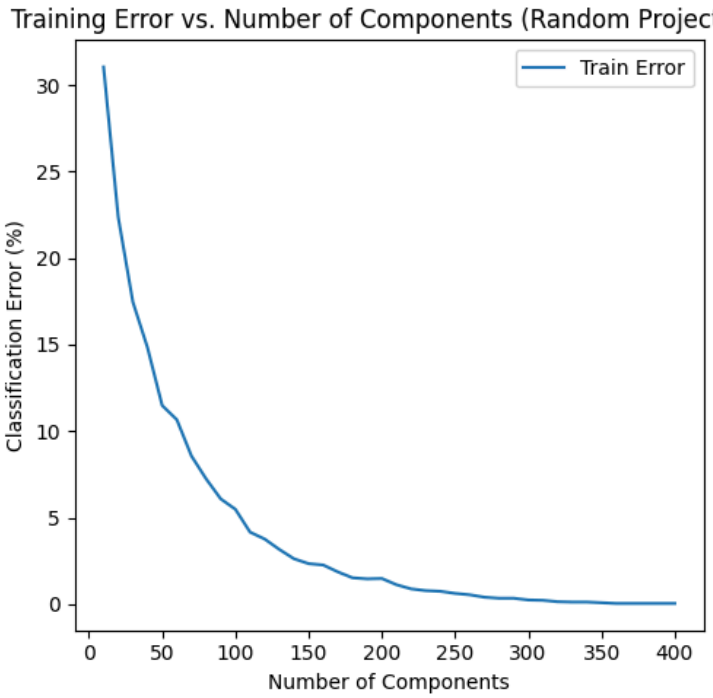
Question 2)

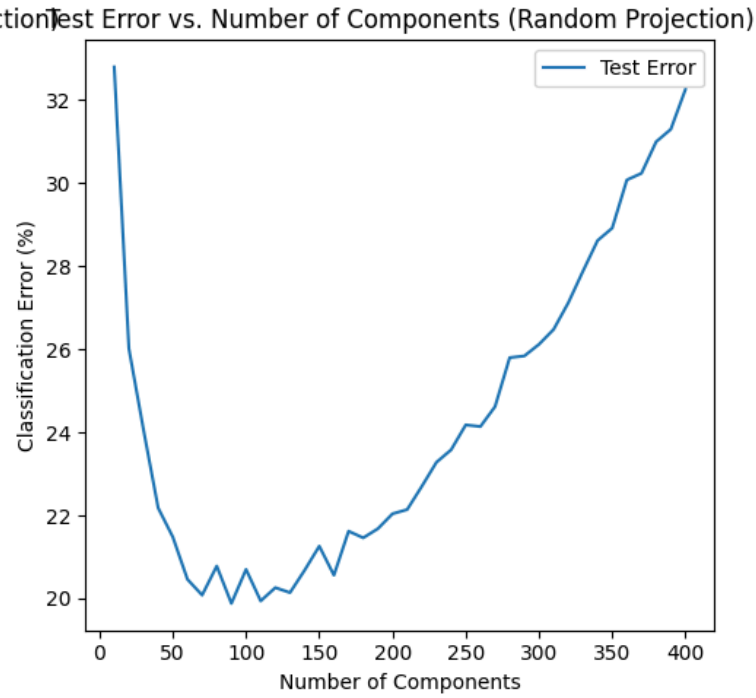
In this question, I used random matrices to project my data onto, instead of principal components, offering a straightforward and computationally less intensive alternative compared to PCA.

For the training set, the use of random matrices resulted in a higher initial classification error compared to PCA. This is primarily because random projections, by their nature, do not retain the detailed variance structure as effectively as the carefully computed principal components. Nevertheless, as the number of random matrices increased, there was a notable convergence

towards zero classification error, which makes sense as we basically kept using more information of the data, mirroring the trend observed with PCA.

On the test dataset, the pattern was similar; the classification error was initially higher with random matrices due to their less precise variance capture. However, as the dimensionality increased, the error rate eventually escalated, indicative of overfitting, a phenomenon also seen in PCA but more pronounced here due to the random nature of the projections. This experiment underscores the balance between dimensionality reduction and the retention of meaningful variance, highlighting the strengths and limitations of random projections in comparison to PCA.



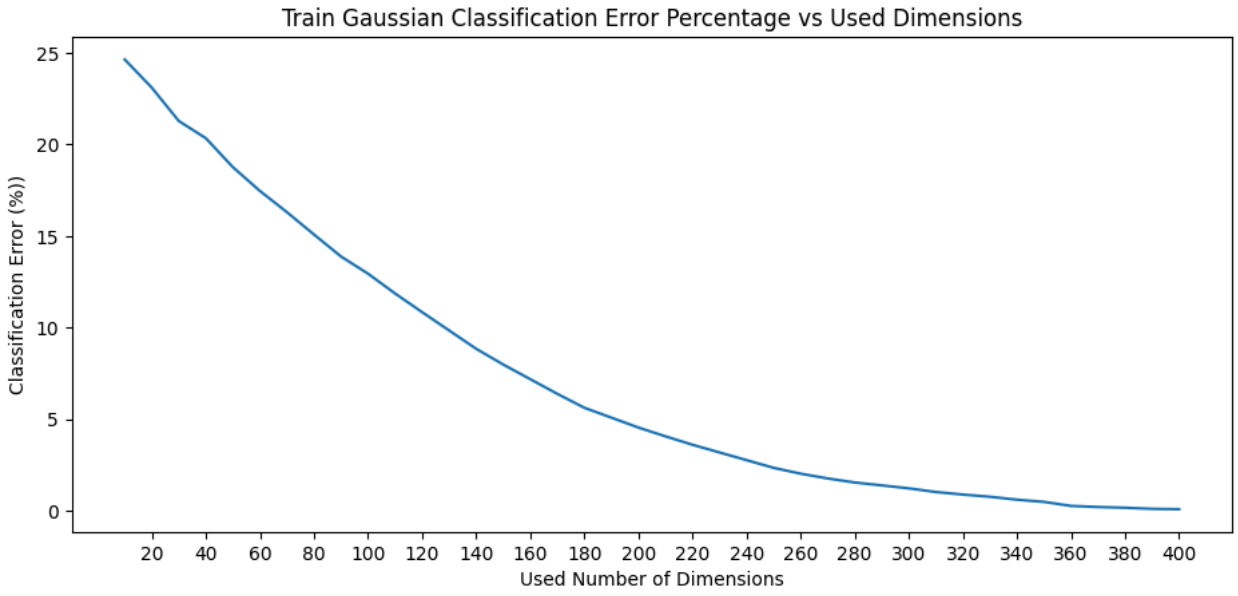


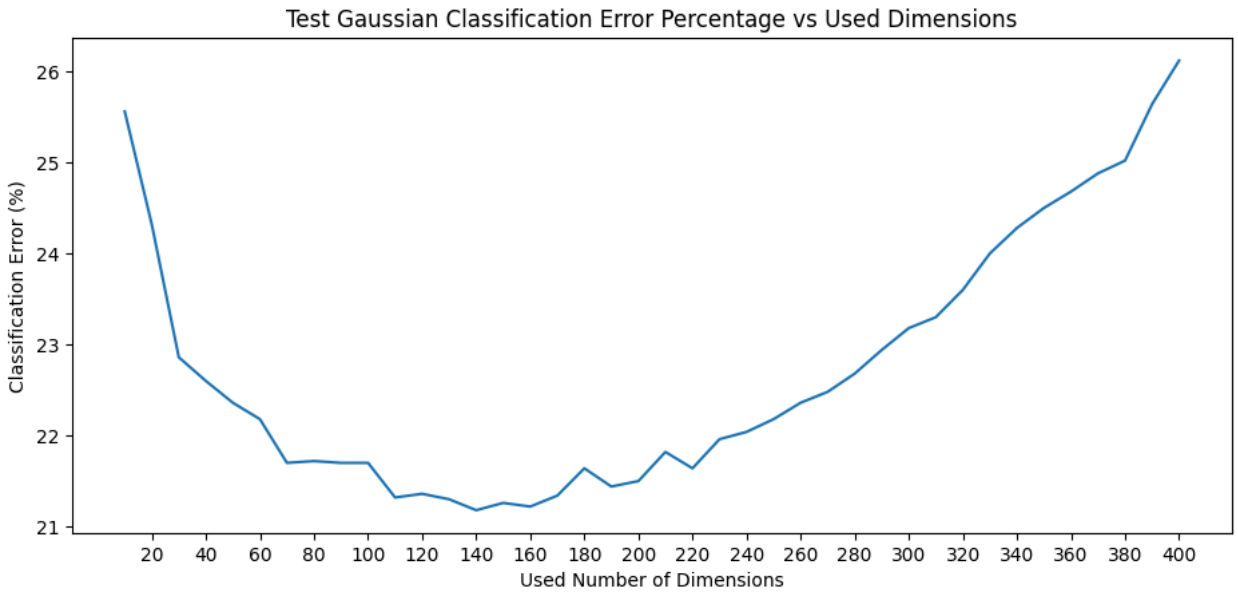
Question 3)

Question 3.1)

In this part, I utilized the *Isomap* algorithm from scikit-learn, adjusting the *n_component* parameter in 10-step increments from 10 to 400 to assess the number of dimensions [3]. I also modified the *n_neighbours* parameter to be 10, instead of the default 5, in order to achieve better results.

Question 3.2 & 3.3)





The error percentage trends observed with PCA and Isomap were notably similar. In the training phase, errors decreased steadily with an increase in dimensions, aiming towards zero. This again makes sense, as we keep using more and more connected graphs.

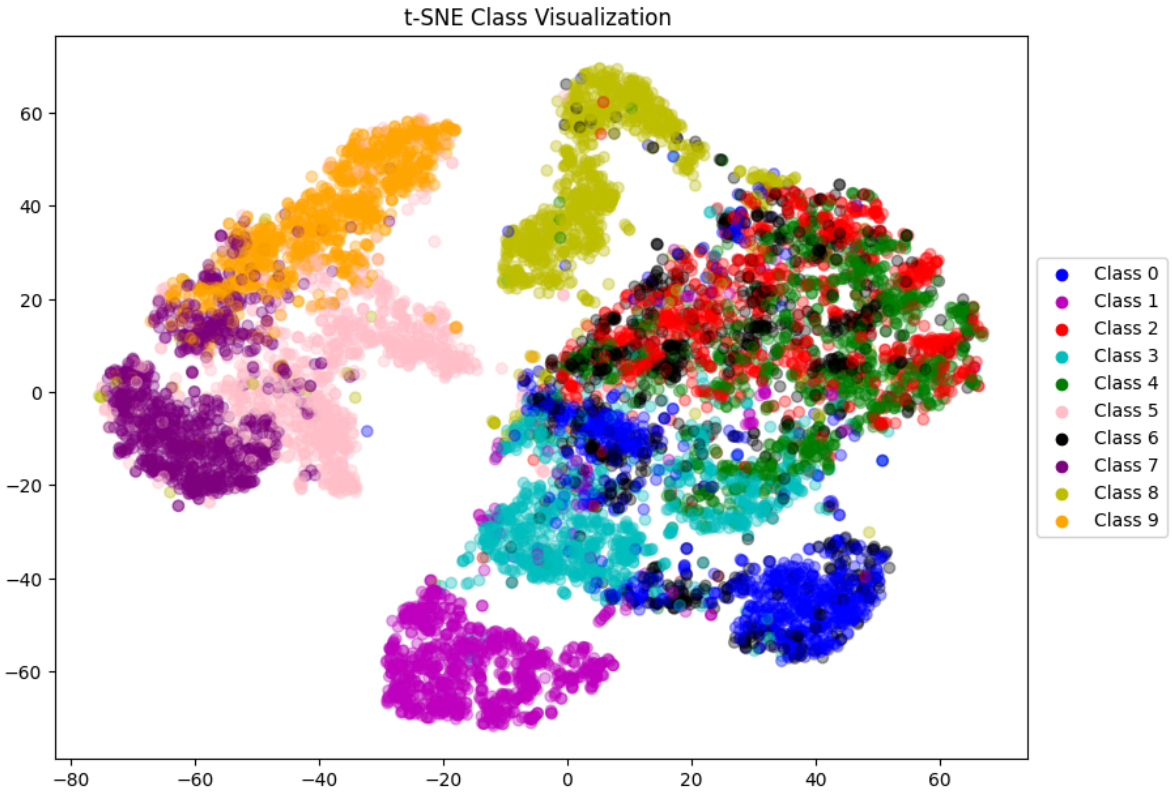
This decline was also noted in the test data, although it was followed by a subsequent rise in error rates, signifying overfitting at higher dimensions, similar to PCA. It can be said that the minimum error rate for Isomap is seen at around 140 dimensions (compared to PCA's 50).

In comparing PCA with Isomap, both methods efficiently represented the dataset into fewer dimensions while sustaining low classification errors. PCA employs a linear approach for dimensionality reduction, whereas Isomap is non-linear. Therefore, the data patterns captured by these methods vary, with PCA potentially being more efficient in extracting the linear and simpler aspects of the dataset, like consistent patterns in clothing, thus requiring fewer components than Isomap to achieve similar or better classification accuracy. Therefore, it can be said that, even though both

methodologies performed well, Isomap needed more components to reach the minimum classification error compared to PCA.

Question 4)

For this part, I utilized the *t-SNE* technique available in the scikit-learn package, applying the *fit_transform* method on the TSNE object with the centered dataset, and selecting 2 for *n_components*, in order to visually represent our data on 2 dimensions [4]. I also increased the *perplexity* parameter to 40 from its default value 30, in order to include more number of nearest neighbors in the computation, since we have a comparably large dataset. Default values were maintained for other parameters such as *early_exaggeration*, *learning_rate*, and *n_iter* (1000).



The t-SNE plot captures the non-linear high-dimensional relationships among clothing items in a two-dimensional space, showcasing the correlations that linear methods like PCA might have overlooked. Notably, the plot distinctly clusters related clothing classes; for instance, classes 5, 7, and 9 are closely grouped, since they are all kinds of shoes: sandal, sneaker, and ankle boot, respectively. One can also notice classes 1 and 8, trouser and bag respectively, are distinct from other classes, since they are the only kind of clothing items in their own category.

t-SNE visualization aids in detecting related clothing clusters and enhances our understanding of the relationships between different clothing categories. However, it's important to recognize that t-SNE serves primarily as a visualization tool, and further statistical analysis is required to comprehensively evaluate the data.

Libraries and Techniques Used

Throughout the project, various libraries were employed for their specialized functions. The *numpy* library provided the essential array structures needed for data operations, ensuring efficient numerical computations [5]. For data partitioning, *sklearn.model_selection* was utilized, specifically its *train_test_split* function, which facilitated the generation of randomized and balanced train/test sets [6]. In terms of classification, *sklearn.discriminant_analysis* offered the QuadraticDiscriminantAnalysis (Quadratic Gaussian Classifier), aligning well with theoretical concepts [2]. For dimensionality reduction in the first question, sklearn.decomposition's PCA was used, where I applied the transform and fit methods before feeding the PCA output to the Gaussian Classifier for prediction, as outlined in the project guidelines [1].

Dimensionality reduction techniques like Isomap and t-SNE were accessed through *sklearn.manifold* [3, 4]. Finally, matplotlib.pyplot was the tool of choice for visualizing the data through various plots, aiding in the interpretation and presentation of the findings [7].

References

[1] “sklearn.decomposition.PCA”. [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.PCA.html.

[2] “sklearn.discriminant_analysis.QuadraticDiscriminantAnalysis”. [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.discriminant_analysis.QuadraticDiscriminantAnalysis.html

[3] “sklearn.manifold.Isomap”.[Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.manifold.Isomap.html

[4] “sklearn.manifold.TSNE”.[Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.manifold.TSNE.html

[5] “NumPy”.[Online]. Available: https://numpy.org/

[6] “sklearn.model_selection.train_test_split”.[Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html

[7] “Pyplot tutorial”.[Online]. Available: https://matplotlib.org/stable/tutorials/pyplot.html#pyplot-tutorial