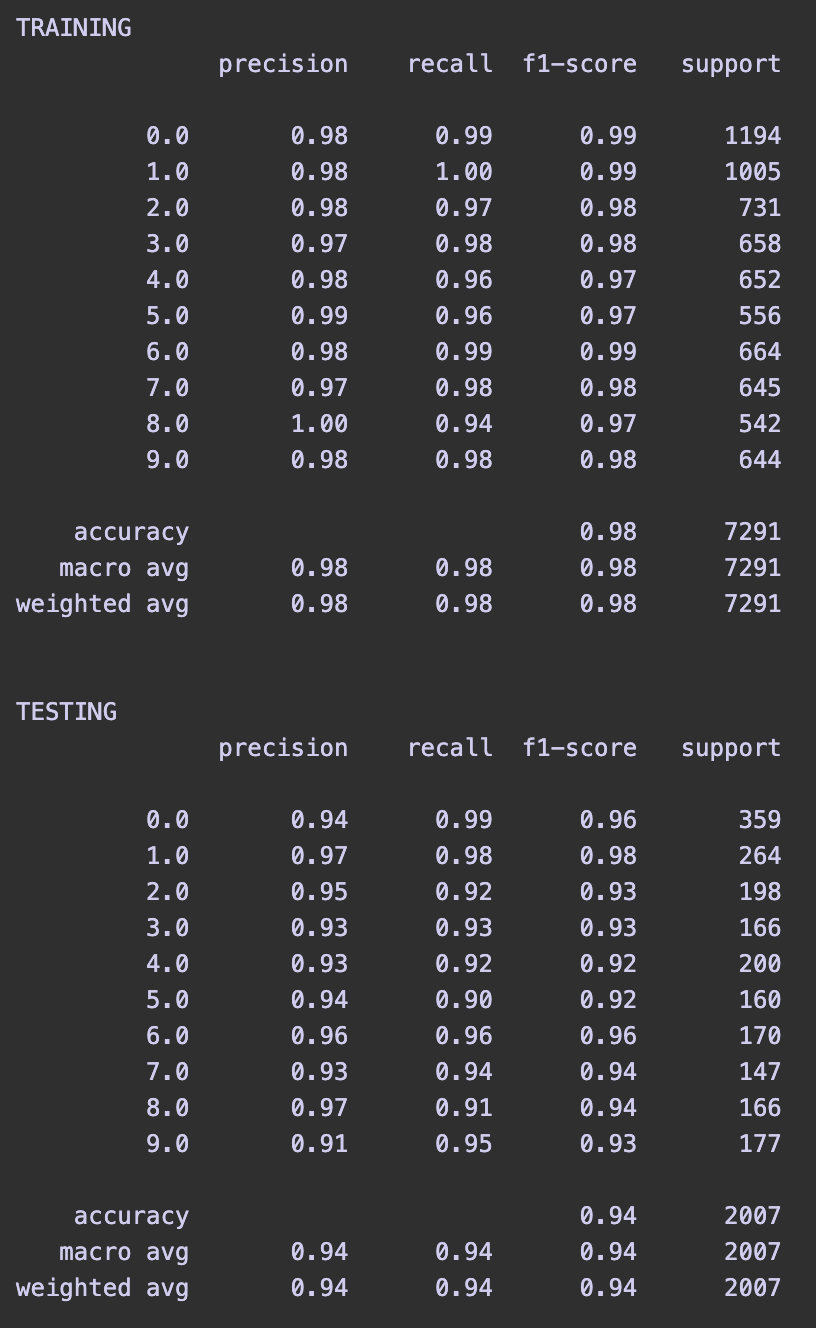
**MACHINE LEARNING FROM DATA**

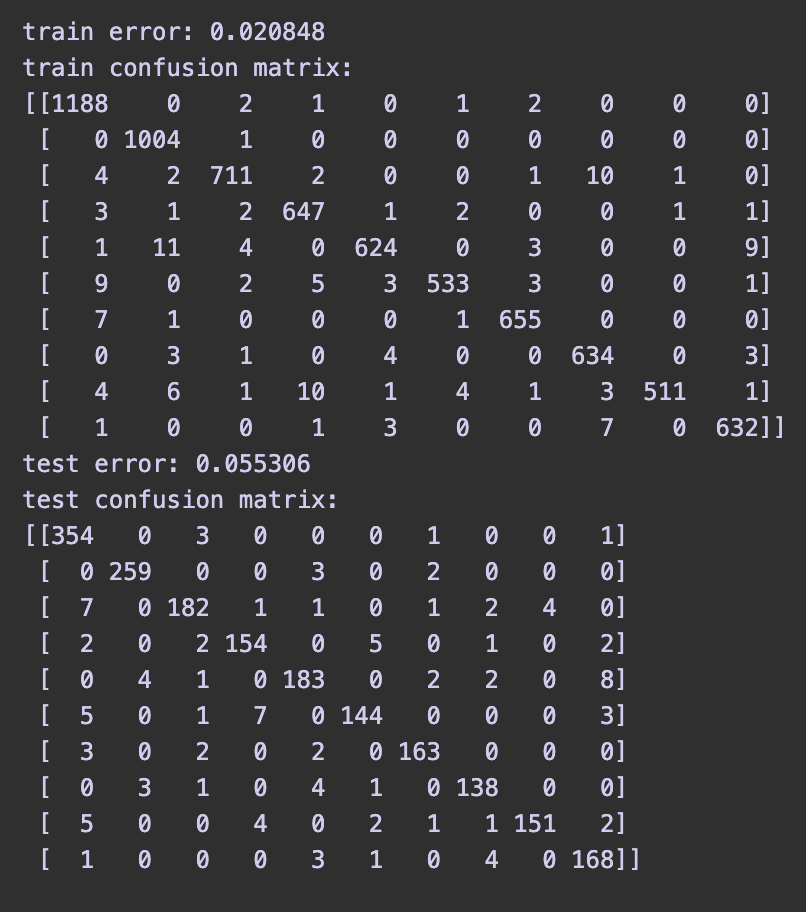
**Report: Lab Session 4 – K-Nearest Neighbors**

**Questions**

**Q1**. Copy the results obtained with kNN on the train and test sets, and discuss the results. What is the value of *k* (default)? Analyze the confusion matrices and identify the two most challenging classes.

Default value for k is 5.

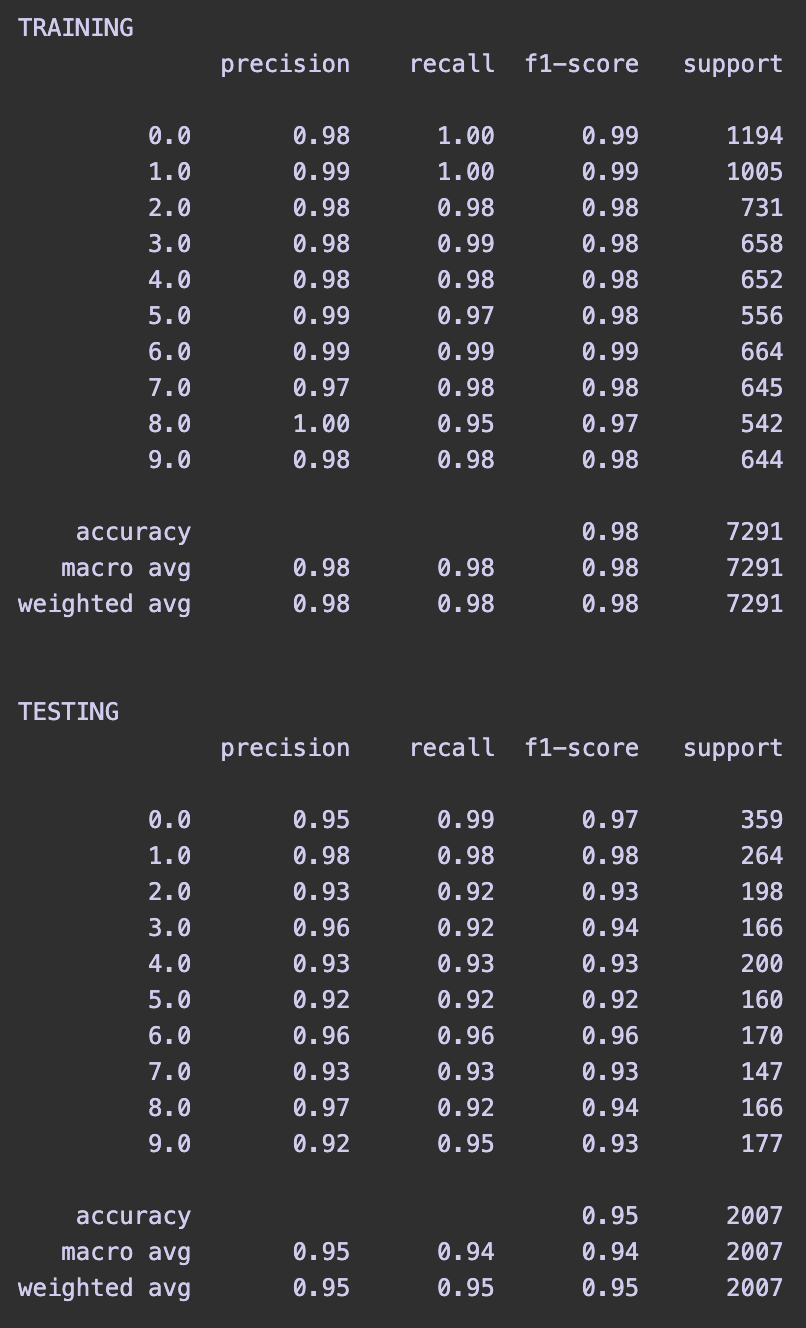


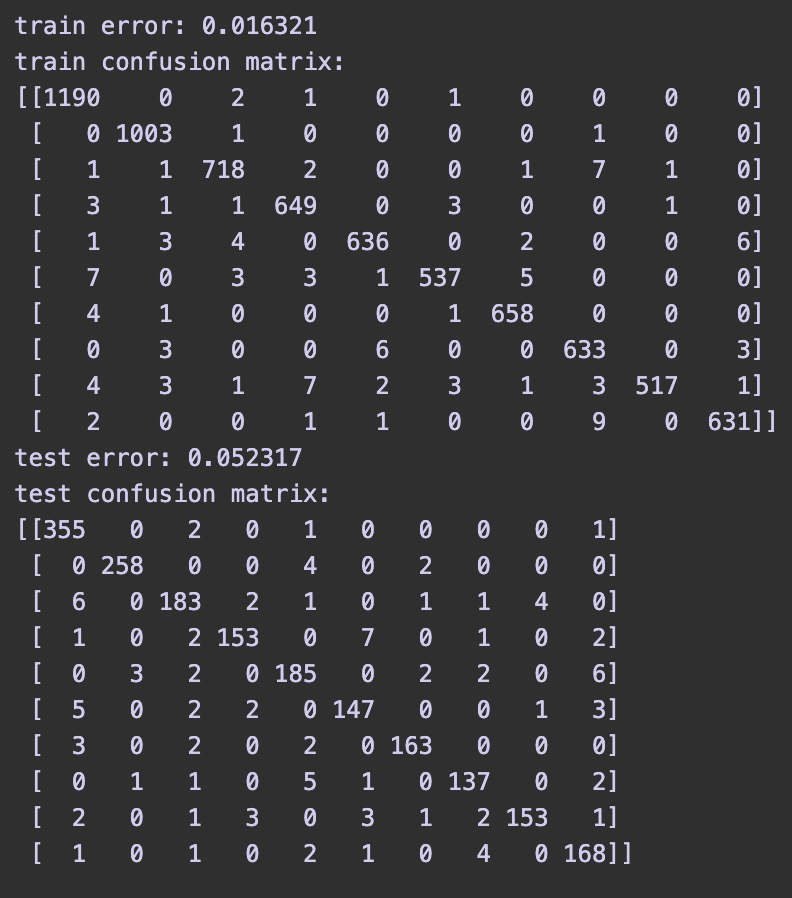


The model has very good accuracy in both training and testing. We cannot see signs of overfitting, the error is a bit higher in the testing but not that much. In general, the model generalizes and performs quite well for the test set.

The model is weak when classifying the classes 4 and 5, based on the value of f1 score, which is a combination of precision and recall. however there is not that big of a difference between these 2 classes and others.

**Q2**. Run again the script, using **PCA** to reduce the dimensionality of the feature space, selecting **d’=64 features**. Observe the eigenvectors and the images reconstructed using only the first d’ eigenvectors (those with the highest eigenvalues). Discuss. Copy train and test results. Discuss the results and compare with the previous case (no PCA).





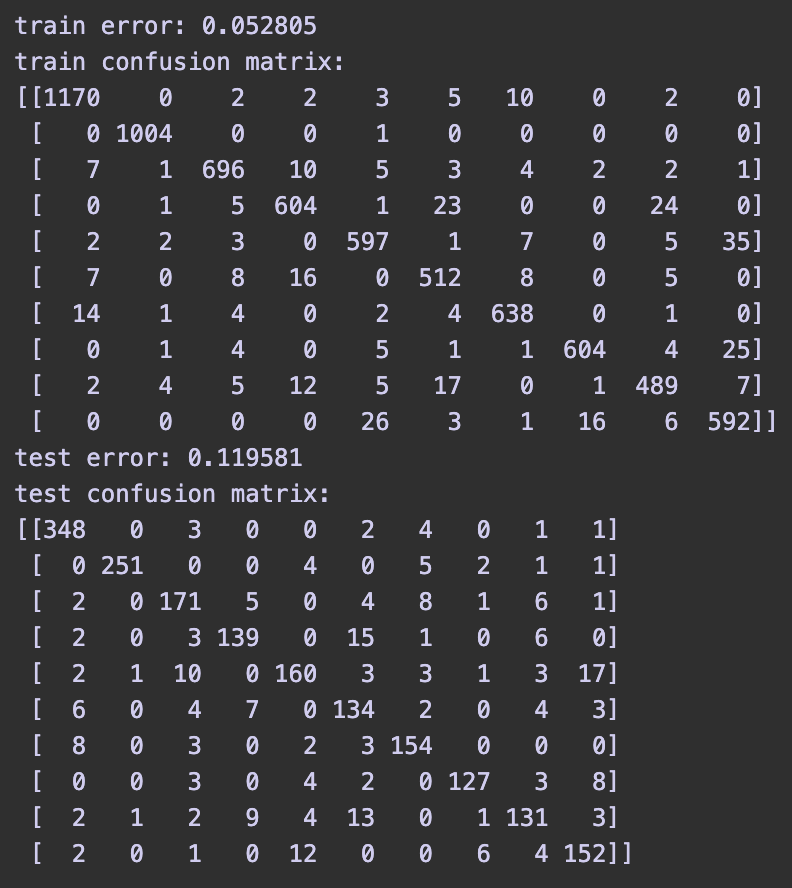
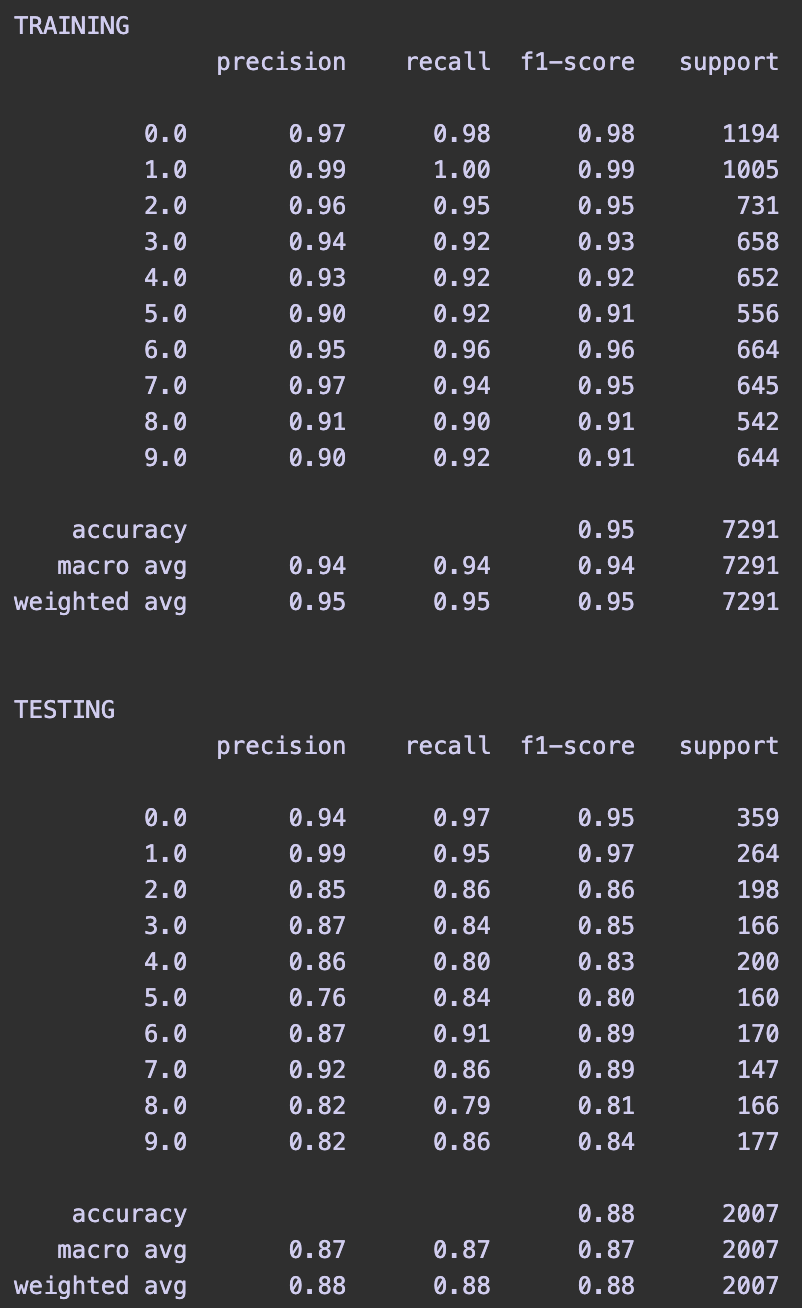
The results after the dimensionality reduction to 64 features is very similar to the previous example. This could mean that using only 64 features is enough to differentiate the classes just how the no reduction option does.

This means that some features are redundant and they are **highly correlated** with others. As a result, the variance of some features is explained by the variance of others and they end up being **redundant**.

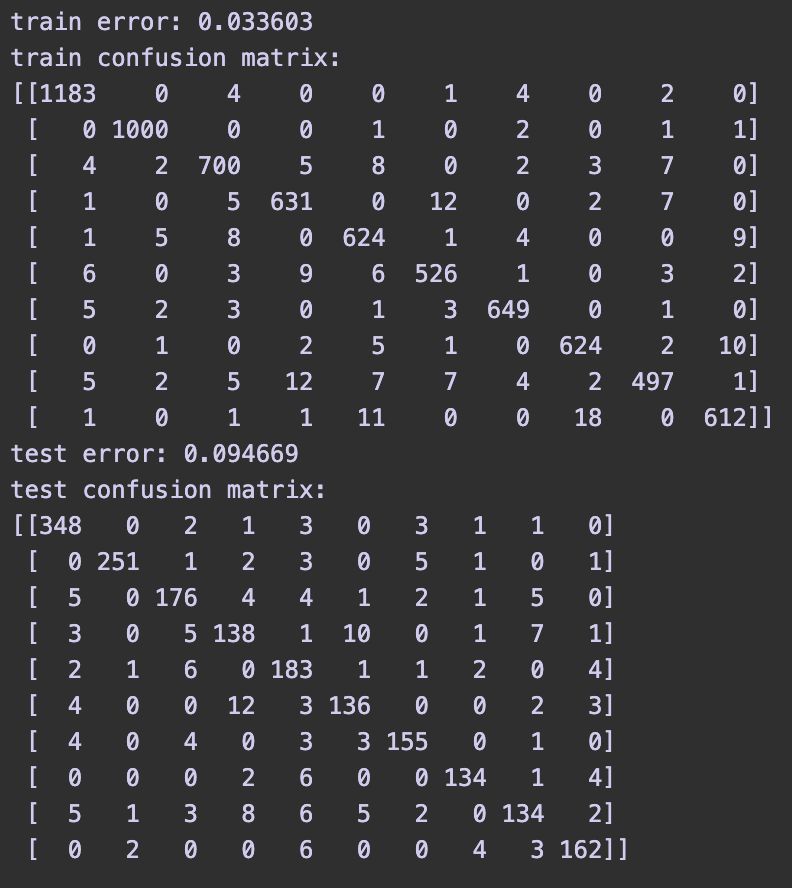
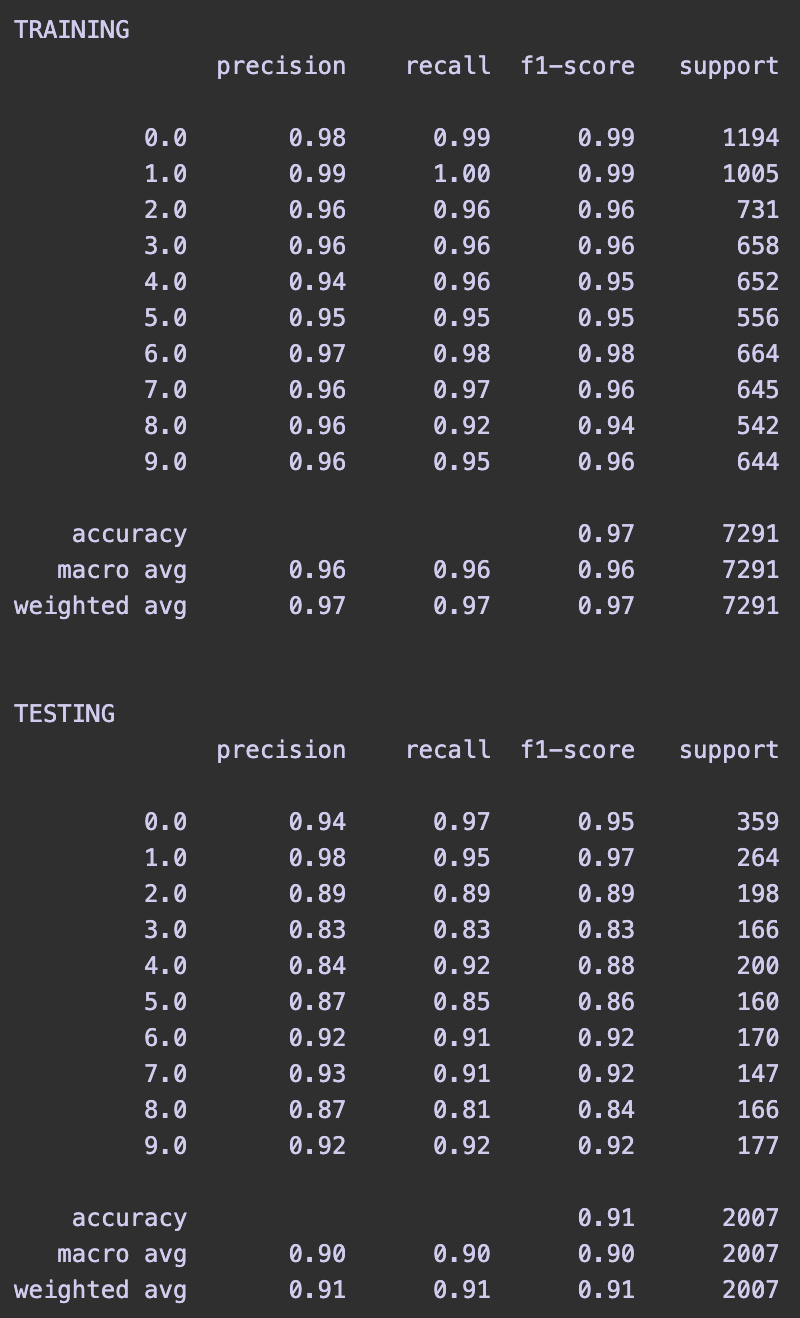
When checking the **eigen-images** we see that they resemble the form of the original images of the set. That is because the most important eigenvectors (related to the highest eigenvalues) **retain the most amount of variance** from the original features and in consequence the KNN classifier can then classify with a very low error rate after the reduction using these eigenvectors.

**Q3**. Repeat the previous analysis using PCA with d’=9 features, and MDA with d’=9 features. Discuss which method is the best for image reconstruction and which one is preferable for classification.

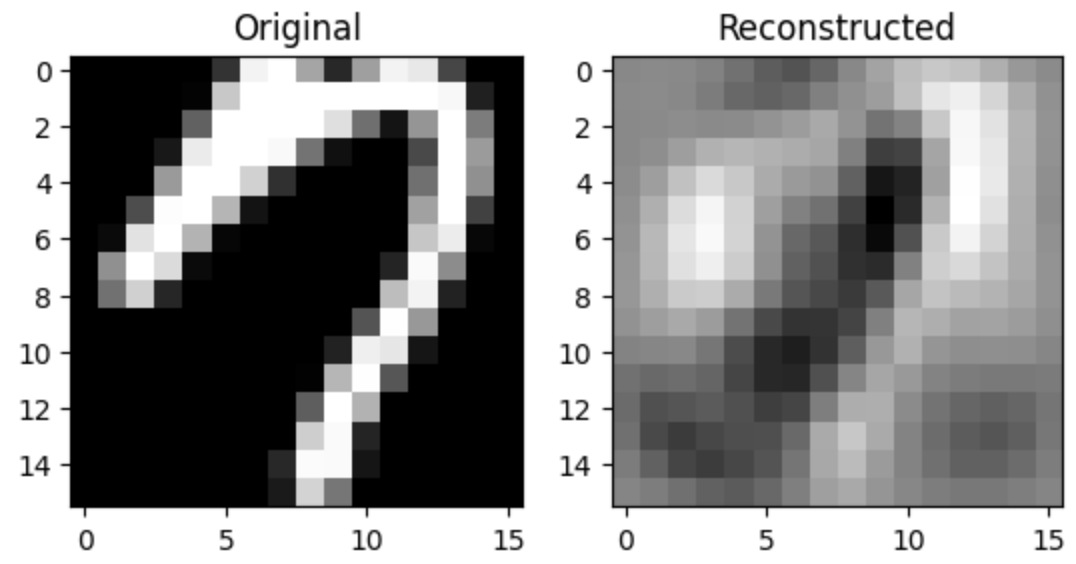
**PCA - d’=9 features**



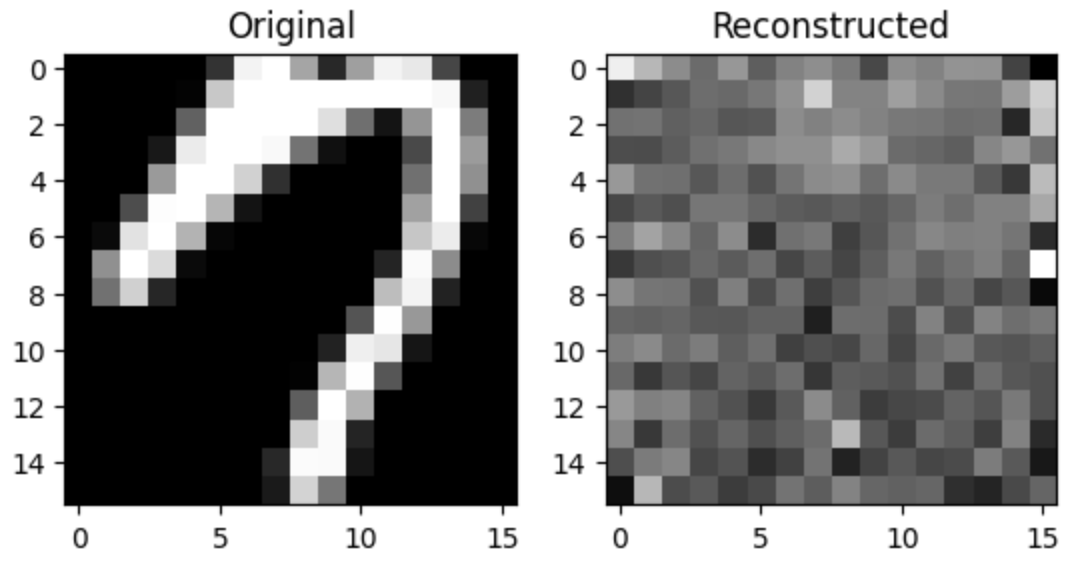
**MDA - d’=9 features**



**PCA reconstruction image:**



**MDA reconstruction image:**

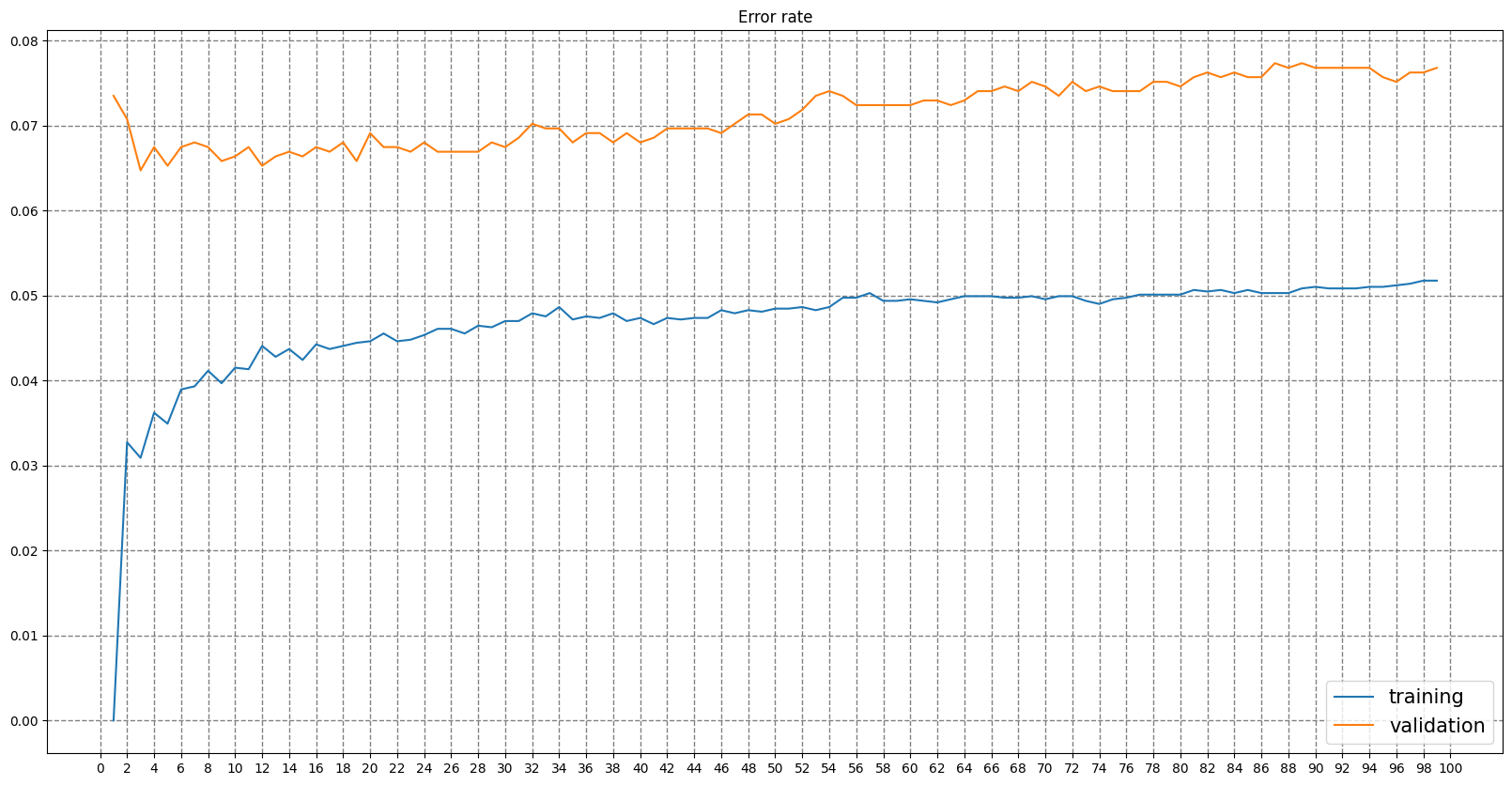


**PCA** is much more useful for **image reconstruction** because the process itself **retains the variance** of the features and that can later be used for the posterior reconstruction of images.

**MDA** is not very useful for image recognition as it **does not preserve the original structure** and only focuses on **enhancing and maximizing the variance related to class separability**. In other words, its focus is to maximize the distance between classes ignoring the structure. The resulting set of elements after this reduction makes it easier to then separate between the classes. So in general **MDA** feature reduction is **better for classification**.

We can also see it clearly in the two images above, with the reconstruction of an image for both PCA and MDA feature reduction. **PCA** still retains a very similar structure, but **MDA** does not resemble at all the original structure.

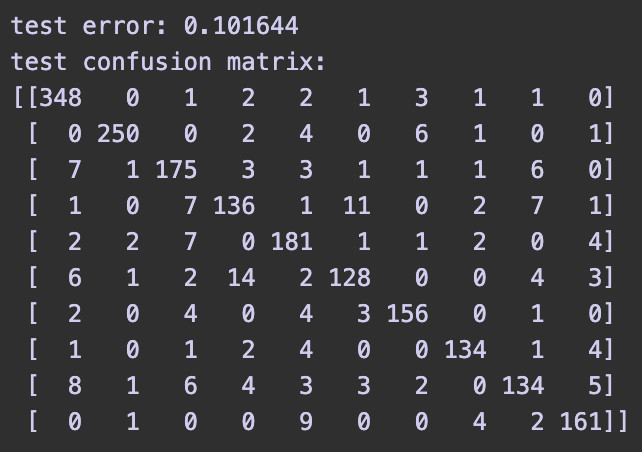
**Q4**. Find the optimal value of *k* on the training set. Use at least 10 values for *k*. Plot the train and validation errors as a function of *k*. Use the optimal value of *k* to compute the error on the test set. Discuss the results.



We see the error rate for the training quickly rises from 0 to 0.04 for the first values of k (from 1 to 8). After that, it is quite stable, but overall we can see a tendency to higher training errors for higher numbers of k.

When analyzing the error rate for the validation set, we see that it is quite stable as well, as it ranges from 0.05 to 0.08. But it is also clear that it steadily rises with the higher values for k we have.

Overall, no matter the value used for k, the resulting classifier is quite good, as the highest errors we find are lower than 0.08. However, for this exercise, the **optimal value is k=3**.



**Results for test set:**

When analyzing the optimal classifier using the optimal values found in the previous exercise, we see that it generalizes and performs quite well, as it has an error of 0.102 for the test set.