Statistical learning for big data (MVE441 / MSA220)
Exam May 24 - June 7, 2024 (part 2)

Luca Modica

June 7, 2024

Question 2a: Dataset exploration through Linear and Nonlinear Dimension Reduction Techniques

The main goal of this task is a first exploration of this dataset, using proper dimensional reduction techniques and methods to handle its high-dimensionality.

The first linear dimensional reduction technique used was PCA, with its Mini batch variant (**Incremental PCA**, IPCA) to handle the high number of observations: the idea is to incrementally use batched data to reduce dimensions. To show its result, down below combinations of the first 4 components are visualized (Figure 1).

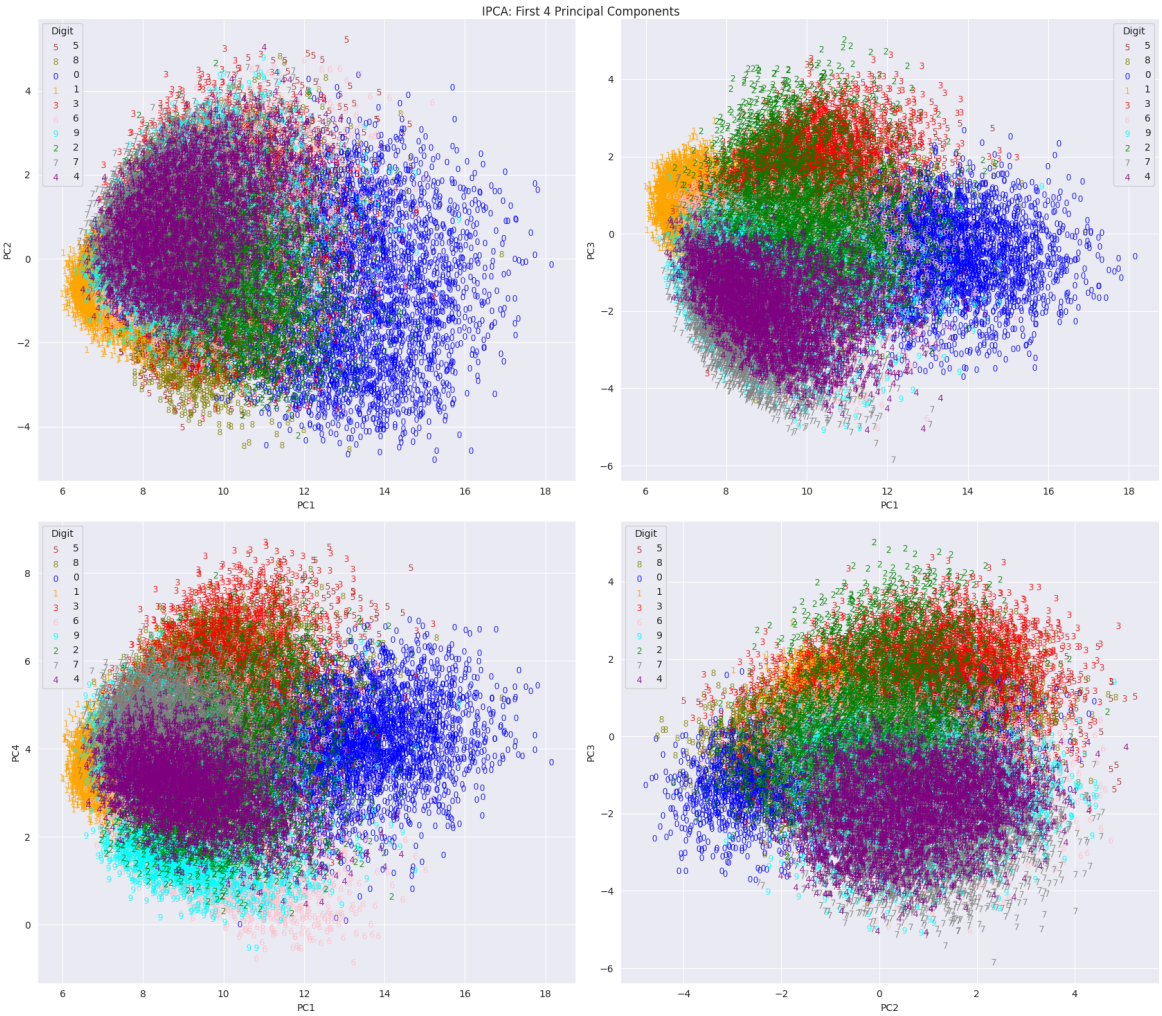

Figure 1: Visualization of the first 4 components of Incremental PCA.

Despite the simplicity of this linear unsupervised method, with Incremental PCA I could start retrieving specific class-based information and class separation insight: from all principal components is indeed possible to see a slight separation of the digit 0 and 1 from the rest, due to their particular shape. For class 0 however, there is overlap with classes that can share shape characteristics, such as 6.

Another useful insight given by IPCA can be its explained variance ratio. Down below (Table) I collected data related to the explained cumulative variance of every 100 components, alongside their relative Mean reconstruction Error.

Number of Components	Mean Reconstruction Error	Explained Variance Ratio
1	0.96	0.04
101	0.53	0.47
201	0.39	0.61
301	0.27	0.73
401	0.15	0.84
501	0.07	0.93
601	0.03	0.97
701	0.01	0.99

Table 1: Mean Reconstruction Error and Explained Variance Ratio by Number of Components (IPCA).

From this, we can conclude that 400 components can be a reasonable reduction to explain most of the dataset variance. Despite this, most of the classes are still overlapping, which suggests that linear methods couldn't be enough to fully separate the classes.

To further investigate, I proceeded with a more powerful linear reduction technique, which **LDA (Linear Discriminative Analysis)**. As we can see in Figure 2, using the class label information provides a more discriminative projection among the digits and less overlap.

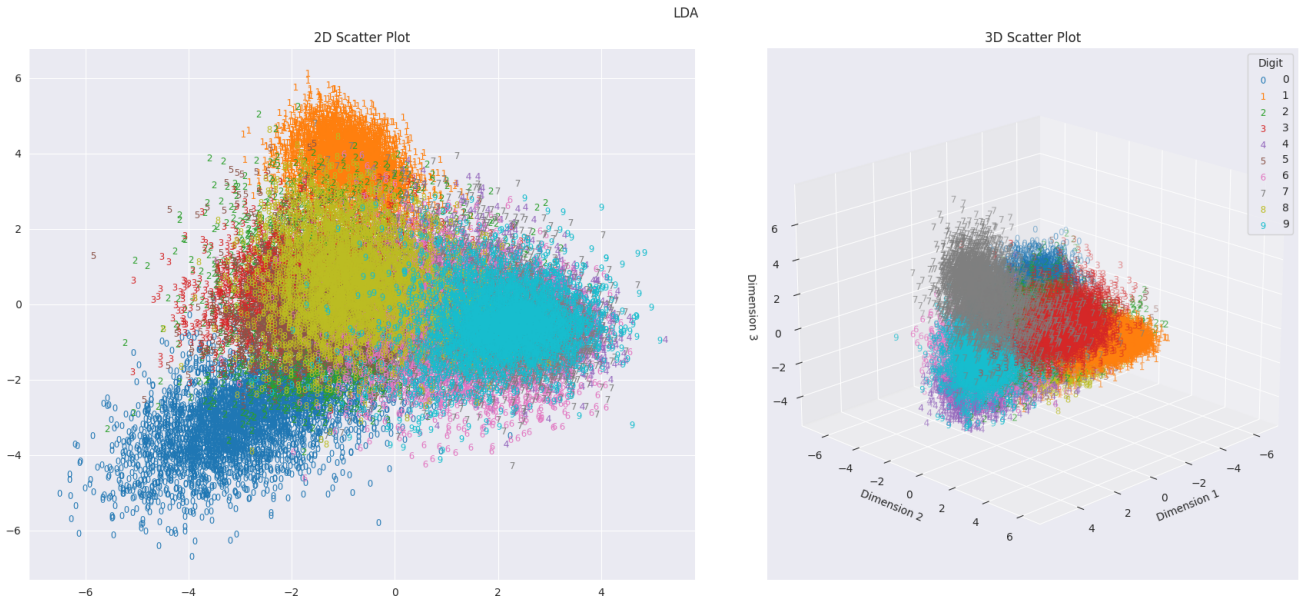

Figure 2: 2D and 3D visualization of the dataset reduced with LDA.

Especially by looking at 3D visualization, we can notice a slightly better separation between other classes: the digit 7 (colored in grey), for example, seems to be a bit separated from the rest of the other classes, which is reasonable due to its almost unique shape. Another insightful observation is given by the partial overlap between class 8 (in light green), class 9 (in light blue), and class 6 (in pink), due to their partially circular part. Even tho, the silhouette coefficient of this linear method is approximately 0.11, which still indicates a major overall overlap between the classes.

To conclude, I performed a dimensional reduction using a non-linear method: **Uniform Manifold Approximation and Projection (UMAP)**. This is a method widely used to reduce very high-dimensional datasets while preserving the

global and local structure of the data.

The results of the data project using this method are shown below (Figure 3).

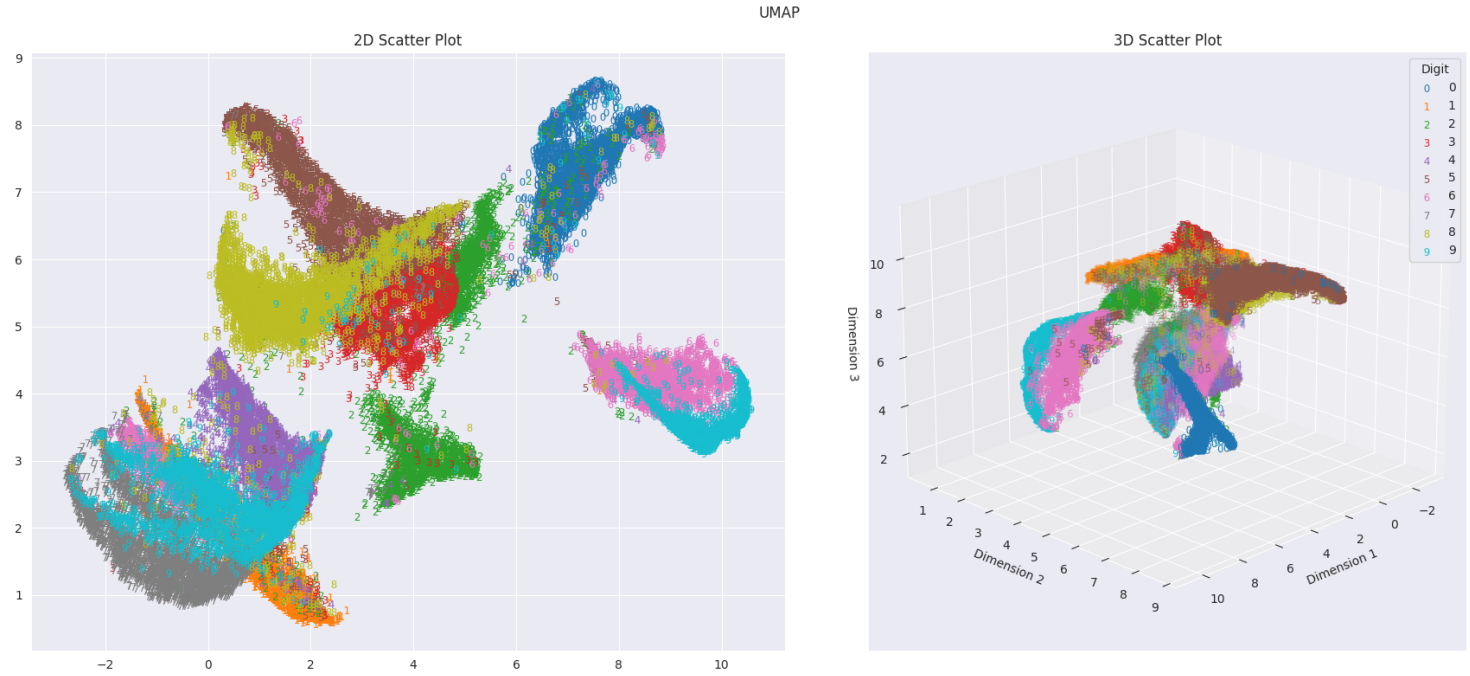

Figure 3: 2D and 3D visualization of the dataset reduced with UMAP.

As we can notice in the plots above, UMAP shows a way more intricate structure of the dataset. While we can see the digit 0 almost clearly separated (with few close 6 digits due to their shape similarity in some cases), we can also notice on the right of the 2D plot 2 close clusters of digit 6 and 9, which also make sense for the way they look. All in all, UMAP shows both a way clearer class separation, in a way that can shed light on class separation and reveal some regions of the images. This is also supported by a high Trustworthiness of the dimensional reduction, which is approximately 0.93.

1 Task 2b: Classification Performance as a Function of Training Sample Size

The next task involves investigating classification performance as a function of training sample size on this data set. This represented a task with some challenges, both for its high dimensionality and the non-linear relationship / overlapping highlighted in 2a. The dimensions of the data heavily influenced the choices of the 5 estimators and the methods used for this experiment. The classifiers used are the following:

- Decision Tree
- Random Forest
- Logistic Regression
- K-Nearest Neighbor (KNN)
- Stochastic Gradient Descent - based classifier with a Linear Support Vector Machine.

Although the training sample sizes (the ones tried are: [1000, 2000, 5000, 10000, 20000, 30000] samples), the main approach used to train all the models is **Divide-and-conquer**: that is, in the training process the training set was divided into 10 parts where 10 different estimators of the same kind were trained and their results combined with different criteria. In particular:

- for Logistic regression, and K-Nearest Neighbor, the strategy was to combine average the results of the 10 different learners trained on different contiguous bathes of the training sample;
- for decision tree and random forest, each learner is combined using a Voting classifier, where the final prediction is made by majority voting;
- a slight exception is made by the SGD classifier, where instead of one single estimator used 10 different batches are used in a progressive way for training.

To assess the different models used, I used 3 different metrics:

- balanced accuracy and F1-score for prediction performance (test scores) and learning trend in the function of the increasing training sample size (training scores)
- training time to also discuss the computational expense.

After also reducing the dimension of the dataset with 400 principal components using IPCA, the results of the training process (on a macro perspective) with 3 metrics mentioned are shown below (Figure 4).

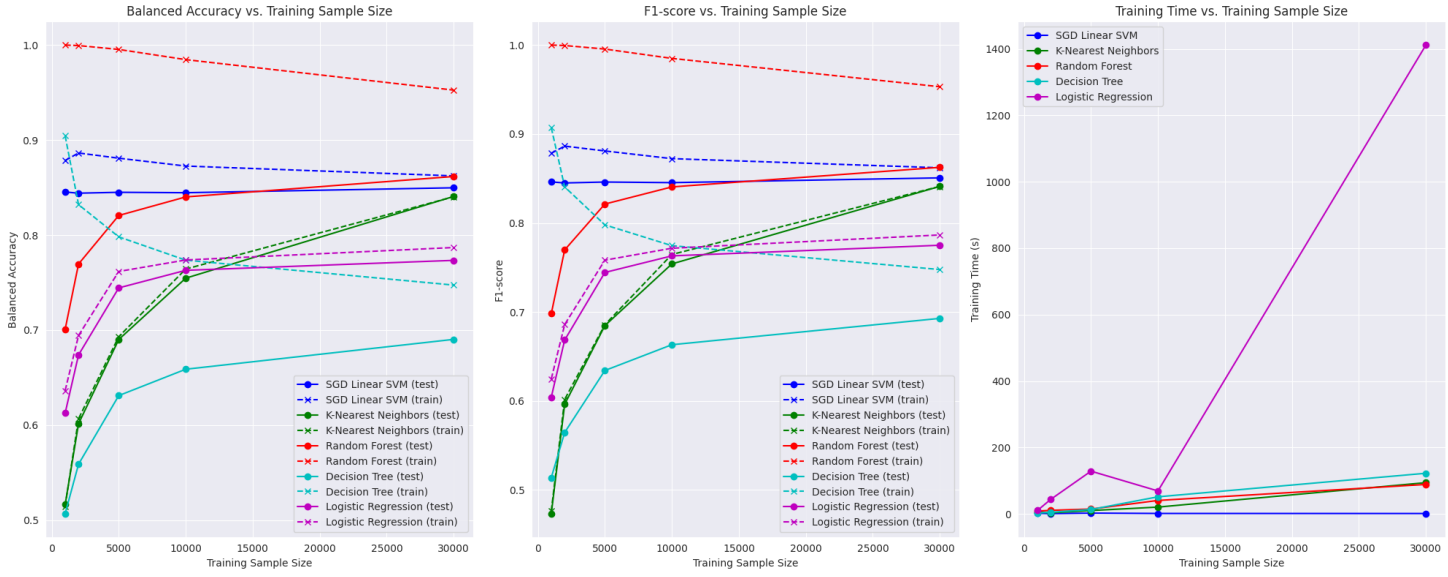

Figure 4: Overall results of the training process in function of increasing training sample size, for all 5 different models.

As we can see above, overall we can see a common pattern of increasing predictive performance across all models, with some exceptions and interesting insights.

- For the decision tree we can see a decreasing training performance alongside a progressive increase in the test scores: this represents a signal of reduced generalization gap.
- surprisingly, the performances of SGD with linear SVM remains almost unchanged with increasing training sample size. This can be due to its internal batch-learning mechanism, which keeps macro performances to a reasonable score.
- KNN seems to benefit the most from the increasing training sample size, suggesting its sensitivity to the amount of labeled data.
- Finally, by looking at the training time, all of the models kept a reasonable amount of seconds but Logistic regression, which demonstrates an exponential increase.

Overall, across all models, we can also see an overall improvement in performance per class. Table 1 shows the difference between the smallest and the biggest training sample size in the experiment.

	2000 training samples	30000 training samples
F1-score for digit 0	0.83	0.90
F1-score for digit 1	0.81	0.89
F1-score for digit 2	0.60	0.75
F1-score for digit 3	0.69	0.79
F1-score for digit 4	0.74	0.83
F1-score for digit 5	0.62	0.76
F1-score for digit 6	0.58	0.76
F1-score for digit 7	0.76	0.84
F1-score for digit 8	0.66	0.78
F1-score for digit 9	0.59	0.72

Table 2: Average F1-scores per digit for different training sample sizes and models.

The table above also shows relatively low performance on the most overlapping digits (mainly 5, 6, and 9), confirming the discovery made by the data exploration in 2a. To conclude the classifier analysis, 3 of the models are put comparison by showing their confusion matrices, for 2 different training sample sizes (2000 and 30000) (Figure 5)

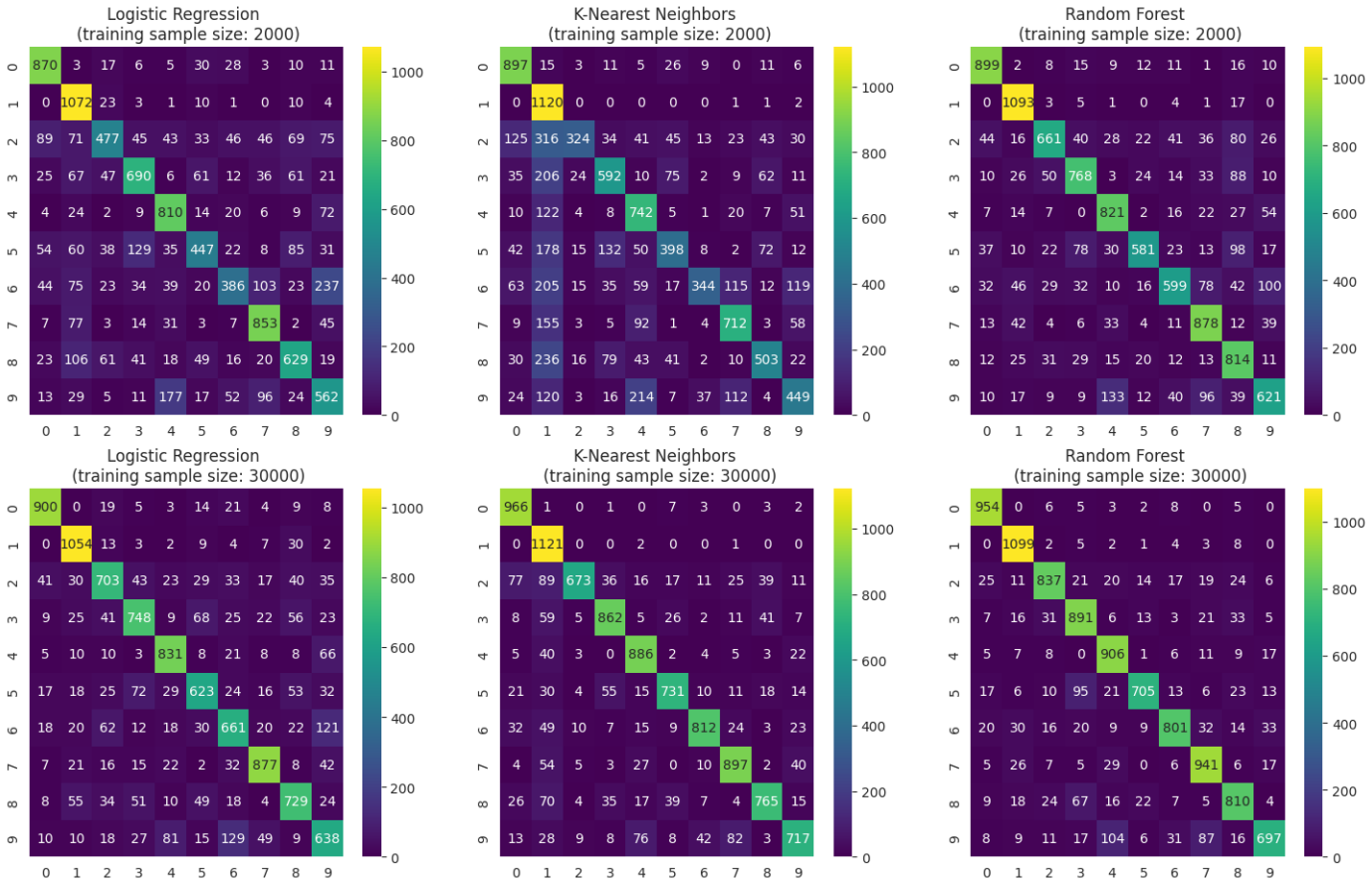

Figure 5: Confusion matrices for a subset of models, for a training sample size of 2000 and 30000.

One of the main findings from the confusion matrices above is the large improvement of KNN in comparison to the other 2 classifiers shown, especially with the digit "1".

2 Task 2c: Classes Retrieval through Clustering

Clustering method used:

- **Kmeans with mini-batch clustering** to deal with the high dimensional dataset
- **DBSCAN**, to try to discern non-linear patterns with a density-based clustering method
- BIRCH, an alternative to MiniBatch K means for efficient clustering with high dimensional data.

the high dimensionality of the data was also handled with a clustering aggregate technique (**local clustering**): that is, after clustering an arbitrary number of local clusters, the result will be then clustered again with 10 clusters, which is a number chosen for the number of classes we want to retrieve from the results. An example of visualization of the results of K-Means is shown below, by using PCA for visualization:

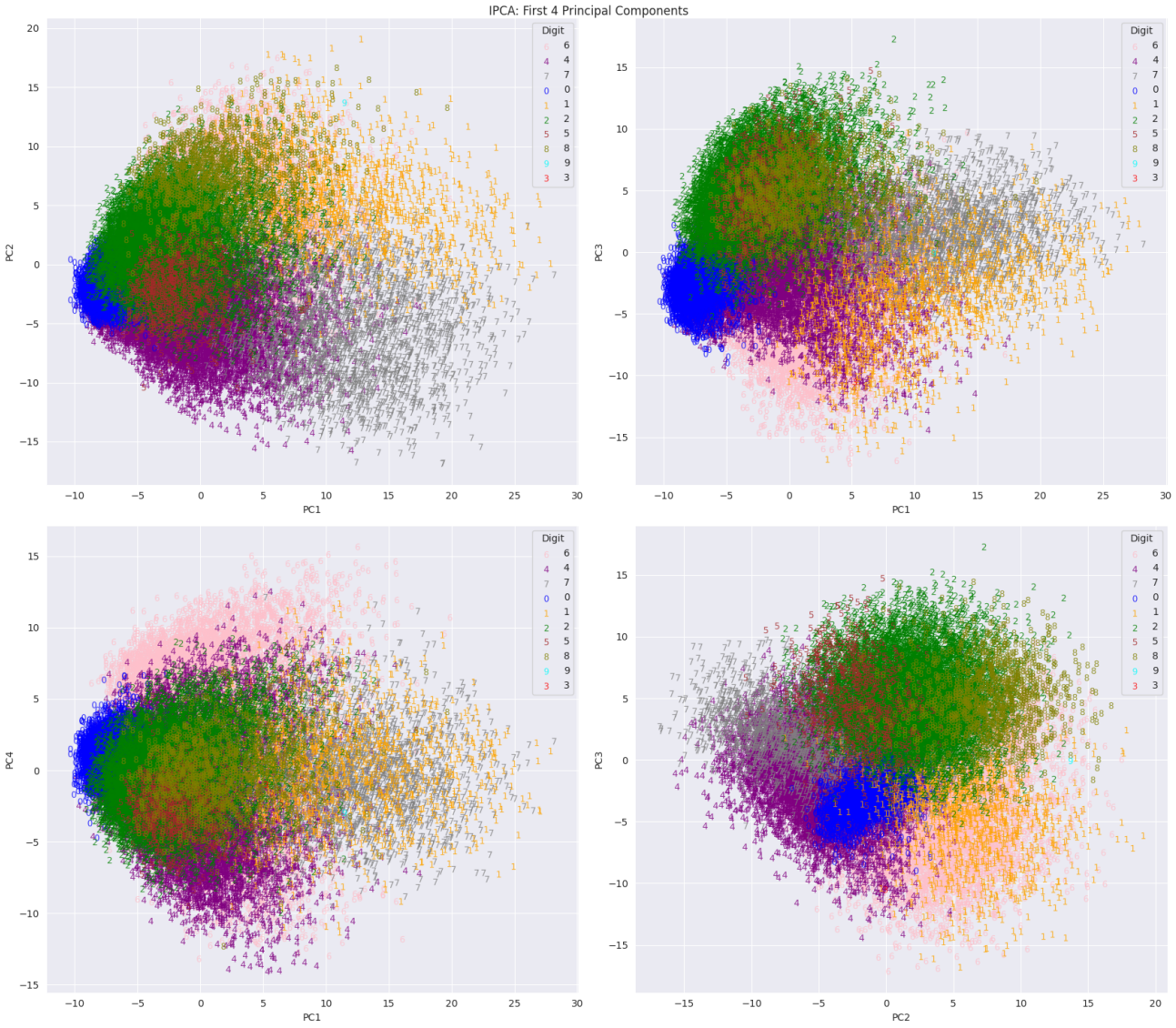

Figure 6: Visualization with the first 4 components of IPCA of the result of Kmeans, using the local clustering.

As can be seen, some of the digits in the dataset can be retrievable, from the digit 4 to the digit 0, even with significant overlapping as highlighted in task 2a.