**CUSTOMER SEGMENTATION USING DATA SCIENCE**

**PHASE 2: INNOVATION**

**TEAM MEMBER**

**SAMUNDEESWARI D - 310521104103**

**INTRODUCTION:**

In this story, we are gonna go through three Dimensionality reduction techniques specifically used for *Data Visualization*: **PCA, t-SNE, LDA and UMAP.**We are going to explore them in details using the *Sign Language MNIST Dataset*, without going in-depth with the maths behind the algorithms.

**What is Dimensionality Reduction?**

Many Machine Learning problems involve thousands of features, having such a large number of features bring along many problems, the most important ones are:

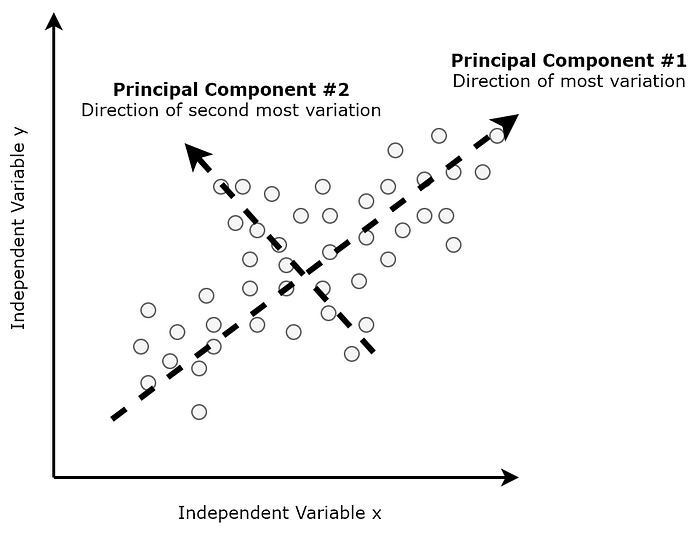
* *Makes the training extremely slow*
* *Makes it difficult to find a good solution*

This is known as the ***curse of dimensionality***and the Dimensionality Reduction is the process of reducing the number of features to the most relevant ones in simple terms.

Reducing the dimensionality does lose some information, however as most compressing processes it comes with some drawbacks, even though we get the training faster, we make the system perform slightly worse, but this is ok! “sometimes reducing the dimensionality can filter out some of the noise present and some of the unnecessary details”.

**1.PCA (Principal Component Analysis)**

One of the most known dimensionality reduction “unsupervised” algorithm is PCA(Principal Component Analysis. This works by identifying the hyperplane which lies closest to the data and then projects the data on that hyperplane while retaining most of the variation in the data set.



PCA is a linear dimensionality reduction technique. Here's an example of how to use it in Python:

**CODING:**

import numpy as np

from sklearn.decomposition import PCA

import matplotlib.pyplot as plt

X = np.random.rand(100, 5)

pca = PCA(n\_components=2)

X\_reduced = pca.fit\_transform(X)

plt.scatter(X\_reduced[:, 0], X\_reduced[:, 1])

plt.xlabel('Principal Component 1')

plt.ylabel('Principal Component 2')

plt.title('PCA Visualization')

plt.show()

**2. t-SNE ( T-distributed stochastic neighbour embedding )**

**(t-SNE)** or **T-distributed stochastic neighbour embedding**created in2008by**(**Laurens van der Maaten andGeoffrey Hinton) for dimensionality reduction that is particularly well suited for the visualization of high-dimensional datasets.

**(t-SNE)**takes a high dimensional data set and reduces it to a low dimensional graph that retains a lot of the original information. It does so by giving each data point a location in a two or three-dimensional map. This technique finds clusters in data thereby making sure that an embedding preserves the meaning in the data. t-SNE reduces dimensionality while trying to keep similar instances close and dissimilar instances apar

t-SNE is a non-linear dimensionality reduction technique. Here's an example of how to use it:

**CODING:**

import numpy as np

from sklearn.manifold import TSNE

import matplotlib.pyplot as plt

X = np.random.rand(100, 5)

tsne = TSNE(n\_components=2, perplexity=30)

X\_reduced = tsne.fit\_transform(X)

plt.scatter(X\_reduced[:, 0], X\_reduced[:, 1])

plt.xlabel('t-SNE Dimension 1')

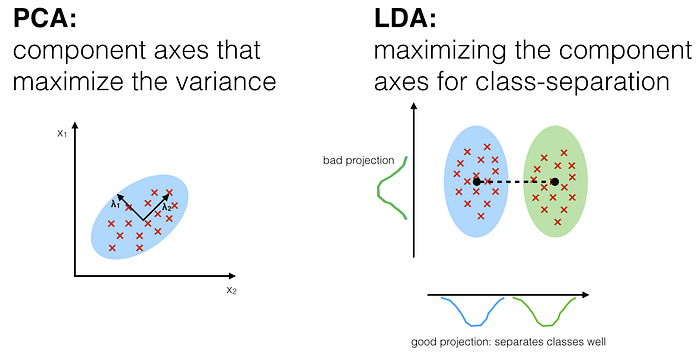
plt.ylabel('t-SNE Dimension 2')

plt.title('t-SNE Visualization')

plt.show()

**LDA ( Linear Discriminant Analysis )**

Linear Discriminant Analysis (LDA) is most commonly used as a dimensionality reduction technique in the pre-processing step for**pattern-classification.** The goal is to project a dataset onto a lower-dimensional space with good class-separability in order to avoid overfitting and also reduce computational costs. The general approach is very similar to PCA, rather than finding the component axes that maximize the variance of our data, ***we are additionally*** . LDA is “supervised” and computes the directions (“linear discriminants”) that will represent the axes that maximize the separation between multiple classes.



**UMAP ( Uniform Manifold Approximation and Projection )**

Uniform Manifold Approximation and Projectioncreated in 2018 by **(**[Leland McInnes](https://arxiv.org/search/stat?searchtype=author&query=McInnes%2C+L), [John Healy](https://arxiv.org/search/stat?searchtype=author&query=Healy%2C+J), [James Melville](https://arxiv.org/search/stat?searchtype=author&query=Melville%2C+J)) is a general-purpose manifold learning and dimension reduction algorithm.UMAP is a *nonlinear* dimensionality reduction method, it is very effective for visualizing clusters or groups of data points and their relative proximities.

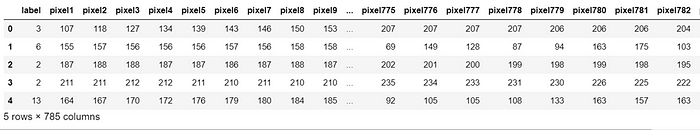
**Use Case**

Now we are going to go through the above-mentioned use case where all the three techniques will be applied: specifically, we will try to visualize a high dimensional dataset using these techniques: T***he Sign-Language-MNIST***Dataset



(Sign-Language-MNIST Dataset)

**The Data**



https://miro.medium.com/v2/resize:fit:700/1*8CAwfSdQeyLSKmObKcMvNA.png

Size of the train Data

https://miro.medium.com/v2/resize:fit:700/1*4nAkbwrMEJsc8cpDXuX9vA.png

The number of unique labels

**Implementing PCA**

After applying PCA, the new dimensionality of the data now has only 3 features compared to the 784 features of the x data.The number of dimension has been cut down drastically whilst trying to retain as much of the ‘variation’ in the information as possible.

**— PCA — 2D —**

plot\_2d(principalComponents[:, 0],principalComponents[:, 1])

Image by Author

From the 2D plot, we can see the two components definitely hold some information, especially for specific digits, but clearly not enough to set all of them apart.

**Implementing t-SNE**

One thing to note down is that **t-SNE** is very computationally expensive, hence it is mentioned in its documentation that :

*“It is highly recommended to use another dimensionality reduction method (e.g. PCA for dense data or TruncatedSVD for sparse data) to reduce the number of dimensions to a reasonable amount (e.g. 50) if the number of features is very high. This will suppress some noise and speed up the computation of pairwise distances between samples.*

Thus, I have applied PCA choosing to retain 50 principal components from the original data to cut down the need for more processing power and it will require time to compute the dimensionality reduction if we had considered the original data.

**Implementing UMAP**

UMAP has different hyperparameters that can have an impact on the resulting embeddings:

* n\_neighbors

This parameter controls how **UMAP** balances local versus global structure in the data. This low values of n\_neighbours forces **UMAP** to focus on very local structures while the higher values will make **UMAP** focus on the larger neighbourhoods.

* min\_dist

This parameter controls how tightly **UMAP** is allowed to pack points together. Lower values mean the points will be clustered closely and vice versa.

* n\_components

This parameter allows the user to determine the dimensionality of the reduced dimension space.

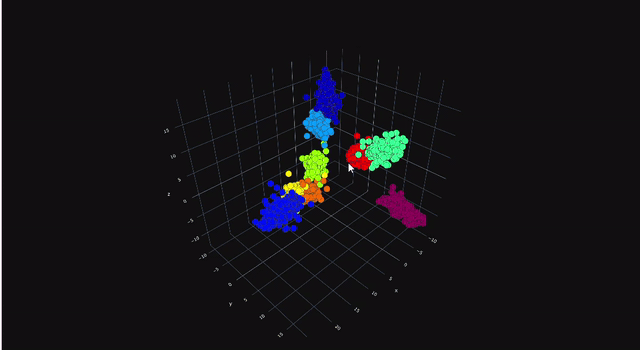
* metric

**IMPLEMENTING LDA**

With LDA we can clearly identify the presence of these nine clusters with a significant separation. If with UMAP and t-SNE we could barely see the backbone of the clusters with LDA we can see the whole clusters of data points agglomerated in the same cluster zones.

**— LDA— 3D —**

plot\_3d(X\_LDA[:, 0],X\_LDA[:, 1],X\_LDA[:, 2])



**Comparison between the Dimension Reduction Techniques: PCA vs t-SNE vs UMAP vs LDA**

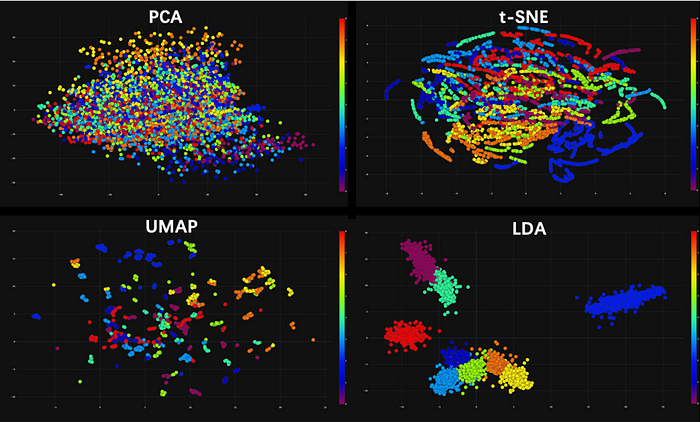
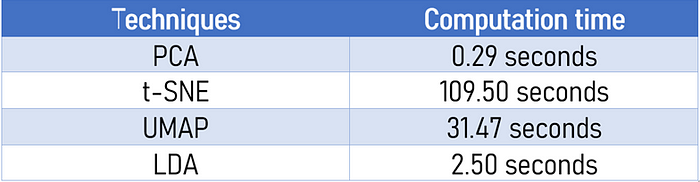


Image by Author



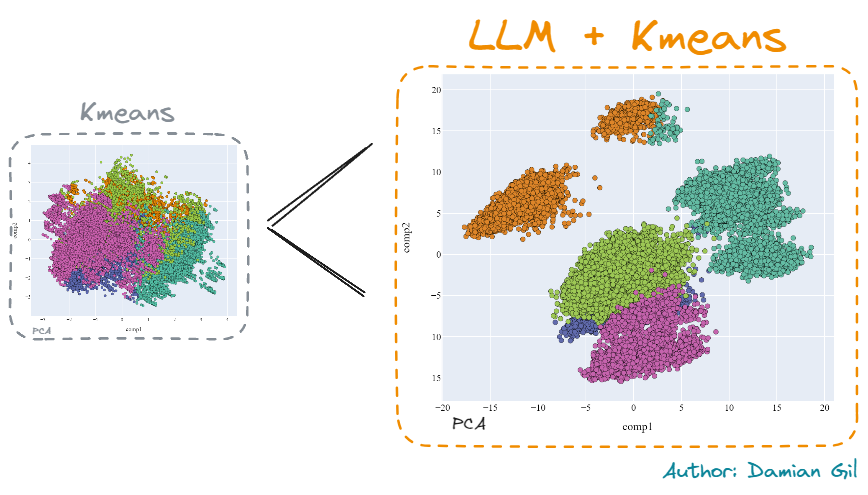
Speed Comparison

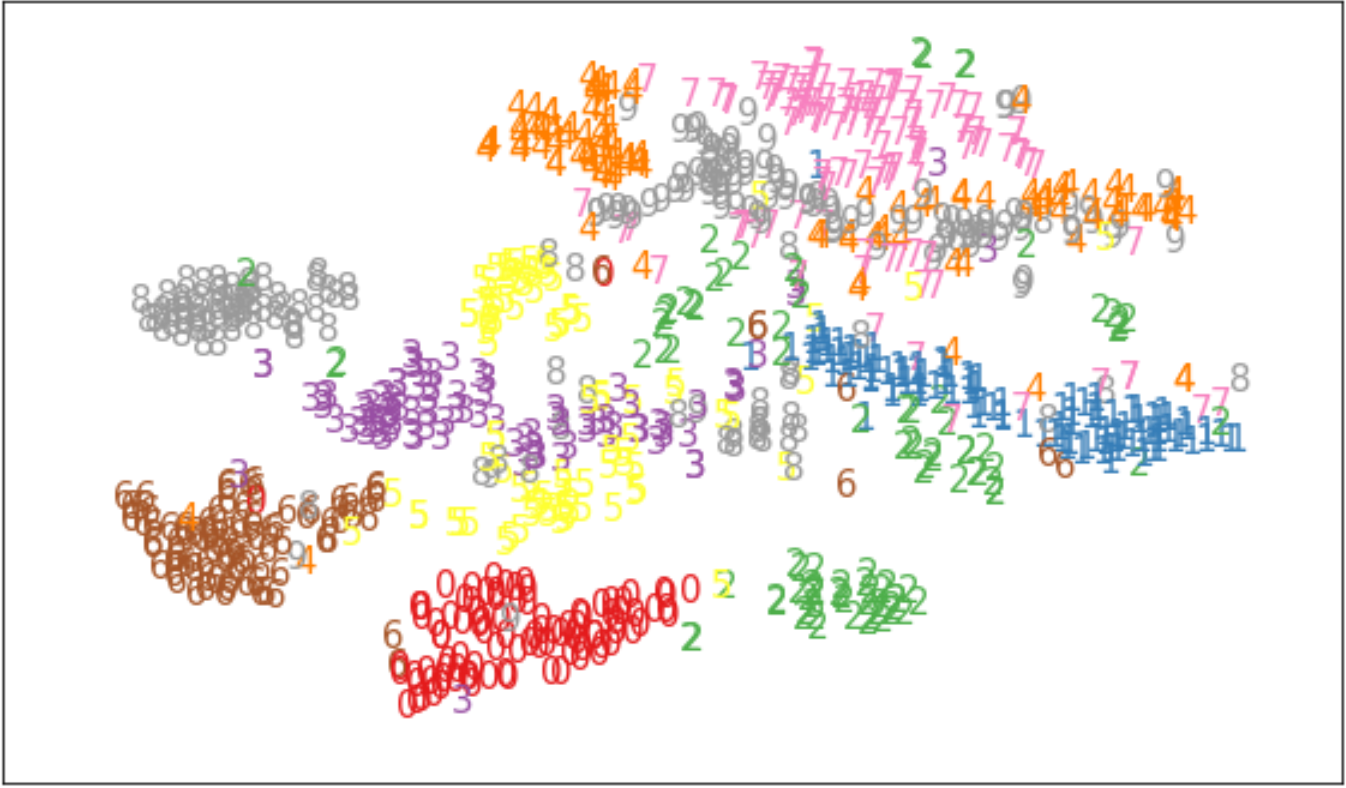
* PCA was not able to do such a good job in differentiating the signs. The main drawback of PCA is that it is highly influenced by outliers present in the data. PCA is a **linear projection**, which means it can’t capture non-linear dependencies, its goal is to find the directions (the so-called principal components) that maximize the variance in a dataset.
* **t-SNE** does a better job(it tries to preserve topology neighbourhood structure) as compared to**PCA** when it comes to visualising the different patterns of the clusters. Similar labels are clustered together, even though there are big agglomerates of data points on top of each other, certainly not good enough to expect a clustering algorithm to perform well.
* **UMAP** outperformed **t-SNE** and **PCA**, if we look at the 2d and 3d plot, we can see mini-clusters that are being separated well. It is very effective for visualizing clusters or **groups of data points and their relative proximities**. However, for this use case certainly not good enough to expect a clustering algorithm to distinguish the patterns.**UMAP** is much faster than **t-SNE**, another problem faced by the latter is the need for *another dimensionality reduction method prior, otherwise, it would take a longer time to compute.*
* Finally, **LDA** outperformed all the previous techniques in all aspects. Excellent computation time (second fastest) as well as proving the well-separated clusters we were expecting.

**Summary**

We have explored four dimensionality reduction techniques for data visualization : (PCA, t-SNE, UMAP, LDA)and tried to use them to visualize a high-dimensional dataset in 2d and 3d plots.

Thanks again for reaching until here, hope it has been an informative post! worth your time. There are many other variants and many other use cases, I highly encourage you to explore this amazing and well-developed area of Science.

[[](https://towardsdatascience.com/mastering-customer-segmentation-with-llm-3d9008235f41?source=author_recirc-----be4aa7b1cb29----1---------------------8cb6b505_f310_4977_9edc_2a373037be8d-------)](https://towardsdatascience.com/mastering-customer-segmentation-with-llm-3d9008235f41?source=author_recirc-----be4aa7b1cb29----1---------------------8cb6b505_f310_4977_9edc_2a373037be8d-------)

[[](https://medium.com/@pajakamy/dimensionality-reduction-t-sne-7865808b4e6a?source=read_next_recirc-----be4aa7b1cb29----1---------------------07b0aeb8_26cc_4a5b_9606_e8980e814a83-------)](https://medium.com/@pajakamy/dimensionality-reduction-t-sne-7865808b4e6a?source=read_next_recirc-----be4aa7b1cb29----1---------------------07b0aeb8_26cc_4a5b_9606_e8980e814a83-------)