**MNIST Dataset Dimensionality Reduction with PCA and Incremental PCA (Question 1)**

**1. Retrieve and Load the MNIST Dataset**

We retrieved the **MNIST\_784** dataset consisting of **70,000 images** of handwritten digits (28x28 pixels, flattened to 784 features).

**2. Display Each Digit**

Each digit was successfully visualized using **Matplotlib**, where each digit (0-9) from the dataset was plotted as a 28x28 grayscale image.

A number set of numbers

Description automatically generated with medium confidence

**3. PCA and Principal Components**

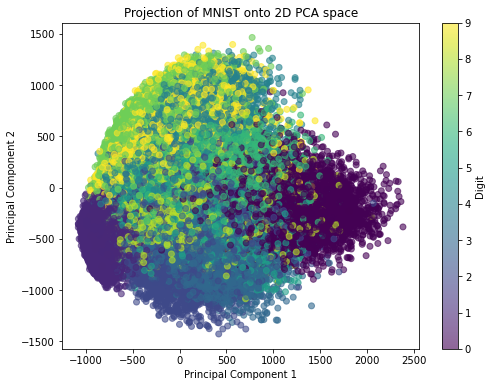
Using **Principal Component Analysis (PCA)**, we reduced the dataset to retrieve the first two principal components:

* **Explained variance of PC1**: 0.0975
* **Explained variance of PC2**: 0.0716

These two components capture approximately **16.91%** of the variance in the dataset.

**4. 1D Projections on the Principal Components**

We projected the data onto the first two principal components and plotted the resulting projections onto a **1D hyperplane**. The distribution of digits along these two components was visualized, showing clustering based on the digit labels.



**5. Incremental PCA to Reduce Dimensionality**

Using **Incremental PCA**, we reduced the dimensionality of the dataset down to **154 components**. The cumulative explained variance with 154 components was **94.95%**, meaning that this reduced dataset retains most of the original dataset's variability.

**6. Original vs Compressed Digits**

The original digits were compared with their compressed representations. The compressed digits retained their recognizable structure despite the reduction in dimensions, confirming the efficiency of PCA for this task.

A number written in black

Description automatically generated with medium confidence

**Challenges and Solutions [15 points]**

* **Challenge**: Handling large datasets with regular PCA could lead to memory issues.
  + **Solution**: We used **Incremental PCA**, which processes data in mini batches, solving the memory issue.
* **Challenge**: Preserving most of the variance while reducing dimensions.
  + **Solution**: We tuned the number of components until we captured over **94%** of the variance with 154 components

**Swiss Roll Dataset with Kernel PCA and Logistic Regression (Question 2)**

**1. Generate Swiss Roll Dataset**

The **Swiss Roll** dataset was generated using Scikit-learn’s make\_swiss\_roll function, producing a 3D spiral-like dataset with 1,000 instances. Each point was assigned a color based on its position for better visualization.

**2. Plot the Swiss Roll Dataset**

The dataset was plotted in 3D, revealing the classic Swiss Roll structure: a spiral shape unrolled in 3D space.

A graph of a graph with dots

Description automatically generated with medium confidence

**3. Apply Kernel PCA (kPCA) with Different Kernels**

We applied **Kernel PCA** using three different kernels:

* **Linear kernel**
* **Radial Basis Function (RBF) kernel**
* **Sigmoid kernel**

**4. Plot kPCA Results and Comparison**

The projections of the data after applying kPCA with different kernels were plotted:

* **Linear kernel**: Preserves linear structures but struggled with the dataset's nonlinearity. A colorful circle with text

  Description automatically generated with medium confidence
* **RBF kernel**: Successfully unraveled the Swiss Roll, projecting it into a meaningful 2D plane that clearly separated the rolled sections.A diagram of a triangle with different colored dots

  Description automatically generated
* **Sigmoid kernel**: Did not perform as well as RBF and was unable to capture the complexity of the dataset.A diagram of a colorful circle

  Description automatically generated with medium confidence

**5. Apply Logistic Regression and Tune Parameters Using GridSearchCV**

Using **RBF kernel** in kPCA, we applied **Logistic Regression** for classification. A **GridSearchCV** was used to optimize the hyperparameters, specifically the gamma value for the RBF kernel and the regularization strength C for Logistic Regression.

* **Best parameters found**:
  + kpca\_\_gamma: 0.020
  + kpca\_\_kernel: RBF
  + log\_reg\_\_C: 1000.0

**6. Plot GridSearchCV Results**

The results of GridSearchCV were visualized, showing the best hyperparameters and their impact on classification accuracy.A graph with dots and numbers

Description automatically generated

**Challenges and Solutions**

* **Challenge**: Finding the optimal gamma and C values.
  + **Solution**: Using **GridSearchCV** to automatically explore a wide range of parameters.
* **Challenge**: Handling nonlinearity in the dataset.
  + **Solution**: The **RBF kernel** in kPCA effectively handled the complex nonlinear structure of the Swiss Roll.
* **Challenge:** Discretizing the continuous target variable.
  + **Solution:** The **KBinsDiscretizer** was used to transform the continuous target variable **t into 10 discrete bins**, making it compatible with logistic regression for classification.