Data Mining
Assignment 2:
Learn how to categorize digit images with IBL

Arnau Sanz Froiz
arnau.sf@students.salle.url.edu

Task 1

Analysis of the Digits Dataset

Dataset Overview

The Digits dataset consists of 1,797 samples, each representing an 8x8 grayscale image of a handwritten digit (0 through 9). Each image is flattened into a 64-dimensional feature vector, where each feature corresponds to the intensity of a pixel.

- Number of Samples: 1,797
- Number of Features: 64
- Number of Classes: 10 (digits 0-9)

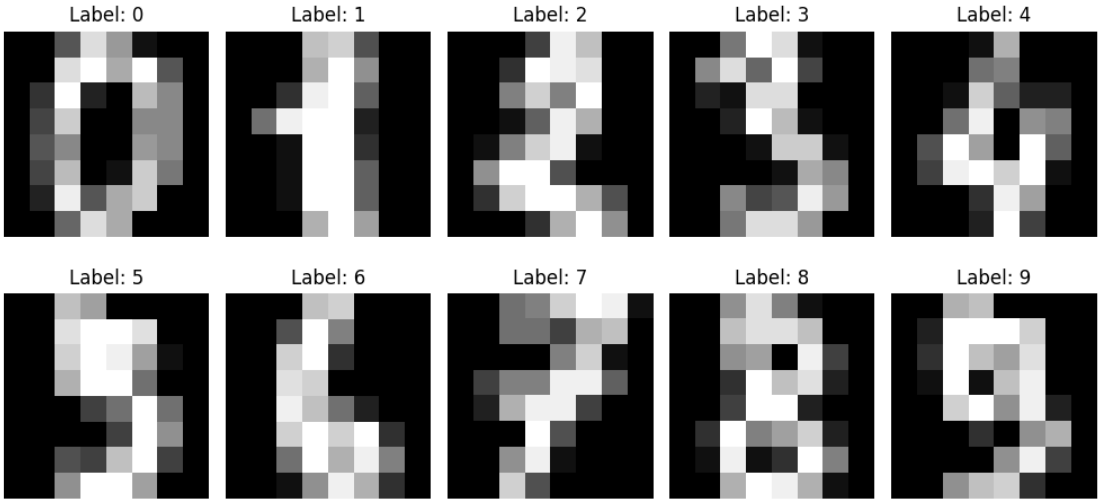
Basic Statistics

- Feature Statistics:
  - Mean (first 10 features): The average pixel intensity for the first 10 pixels across all images.
  - Standard Deviation (first 10 features): The variability in pixel intensity for the first 10 pixels.
- Class Distribution:
  - Class 0: 178 samples
  - Class 1: 182 samples
  - Class 2: 177 samples
  - Class 3: 183 samples
  - Class 4: 181 samples
  - Class 5: 181 samples
  - Class 6: 181 samples
  - Class 7: 179 samples
  - Class 8: 174 samples
  - Class 9: 180 samples

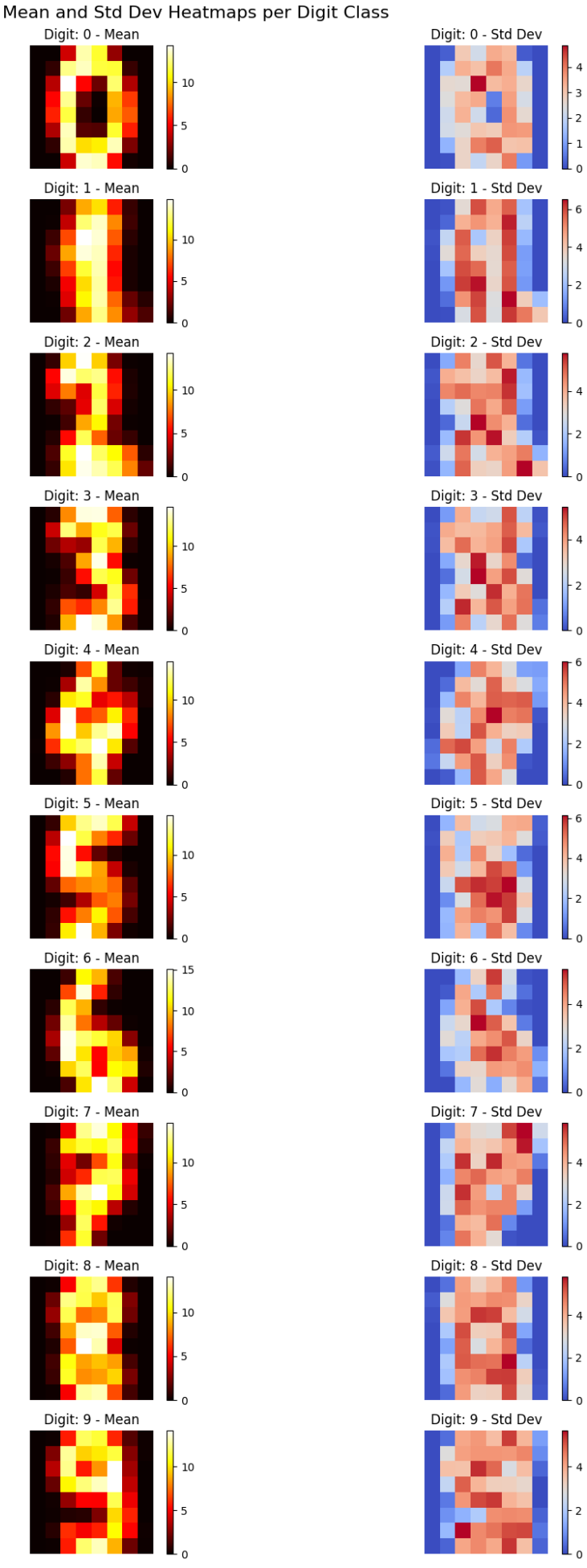
The dataset is relatively balanced across all classes, with each digit class containing approximately 180 samples.

Visual Inspection

An initial visual inspection of the dataset reveals a variety of handwriting styles. Below are examples of digits from classes 0 to 9:



A mean and standard deviation heat map representation of the dataset per-class shows the following visualization:


Mean and Std Dev Heatmaps per Digit Class
Digit: 0 - Mean
Digit: 0 - Std Dev
Digit: 1 - Mean
Digit: 1 - Std Dev
Digit: 2 - Mean
Digit: 2 - Std Dev
Digit: 3 - Mean
Digit: 3 - Std Dev
Digit: 4 - Mean
Digit: 4 - Std Dev
Digit: 5 - Mean
Digit: 5 - Std Dev
Digit: 6 - Mean
Digit: 6 - Std Dev
Digit: 7 - Mean
Digit: 7 - Std Dev
Digit: 8 - Mean
Digit: 8 - Std Dev
Digit: 9 - Mean
Digit: 9 - Std Dev

Task 2

Why Normalize Based on Training Data Only?

Key Reasons:

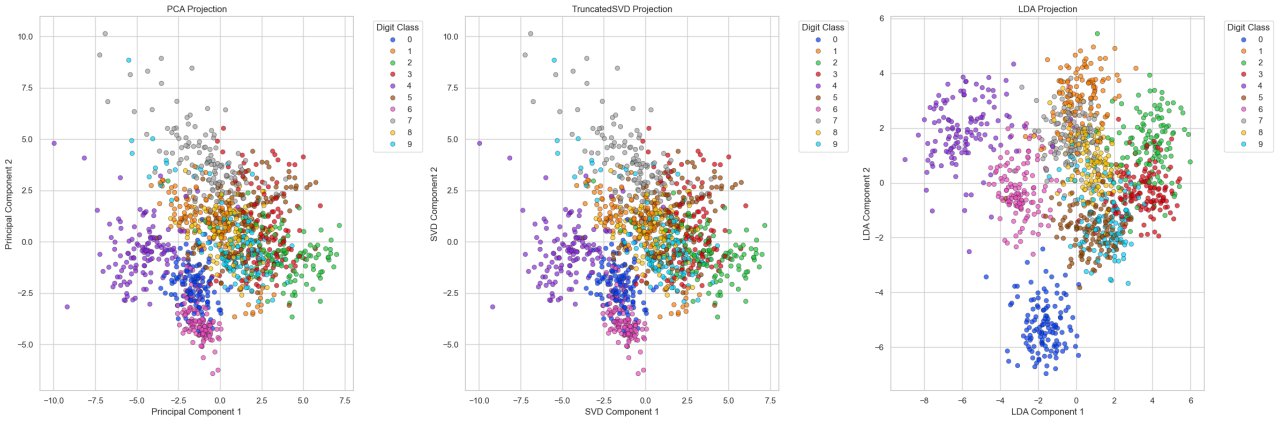
1. Prevent Data Leakage:
   - Data leakage occurs when information from the test set inadvertently influences the training process, leading to overly optimistic performance estimates.
   - Calculating normalization parameters (mean and standard deviation) using the entire dataset (both training and testing) would introduce information from the test set into the training process.
2. Reflect Real-World Scenarios:
   - In real-world applications, new (unseen) data arrives after the model has been trained. The model should rely solely on training data statistics to handle such scenarios.
3. Consistency:
   - The test data should be transformed using the same parameters derived from the training data to ensure consistency in feature scaling.

Splitting the dataset into training and testing sets ensures that the model is trained on one subset and evaluated on another, providing an unbiased estimate of its performance. Normalizing the data using z-score normalization based solely on the training data prevents data leakage and ensures that the model generalizes effectively to new, unseen data.

Task 3

Data decomposition techniques

Trying Linear Discriminant Analysis (LDA) as an alternate method to decompose the data into projections to achieve a better data separation yields a clearer clustering of the different digits while allowing for a greater separation between the clusters.



Scikit provides an efficient implementation of LDA through the LinearDiscriminantAnalysis class.

Task 4

Cross-validation to estimate optimal number of K neighbours

It was aimed to develop an effective K-Nearest Neighbours (KNN) classifier for the Digits dataset by optimizing its hyper parameters. Utilizing cross-validation and GridSearchCV, I systematically searched for the best combination of preprocessing methods, dimensionality reduction techniques, and KNN hyper parameters to achieve high classification accuracy.

Data Preparation

- Dataset: Digits dataset from scikit-learn, comprising 1,797 samples of 8x8 pixel images representing digits 0 through 9.
- Train-Test Split: The dataset was split into 70% training and 30% testing sets, ensuring that the class distribution was maintained using stratification.

Preprocessing and Dimensionality Reduction

We explored various preprocessing and dimensionality reduction techniques to enhance the classifier's performance:

- Preprocessing Methods:
  - StandardScaler: Standardizes features by removing the mean and scaling to unit variance.
  - MinMaxScaler: Transforms features by scaling each feature to a given range (default is [0, 1]).
  - Passthrough: No scaling applied.
- Dimensionality Reduction Techniques:
  - Principal Component Analysis (PCA): Unsupervised method that projects data onto directions maximizing variance.
  - Truncated Singular Value Decomposition (TruncatedSVD): Unsupervised method suitable for large and sparse datasets.
  - Linear Discriminant Analysis (LDA): Supervised method that maximizes class separability.

Results

After exhaustive search, the best model was identified by using GridSearchCV with the following parameters:

- Preprocessing Method: StandardScaler
- Dimensionality Reduction: LinearDiscriminantAnalysis (LDA)
- Number of Components: 5
- Number of Neighbors (K): 9
- Weight Function: distance

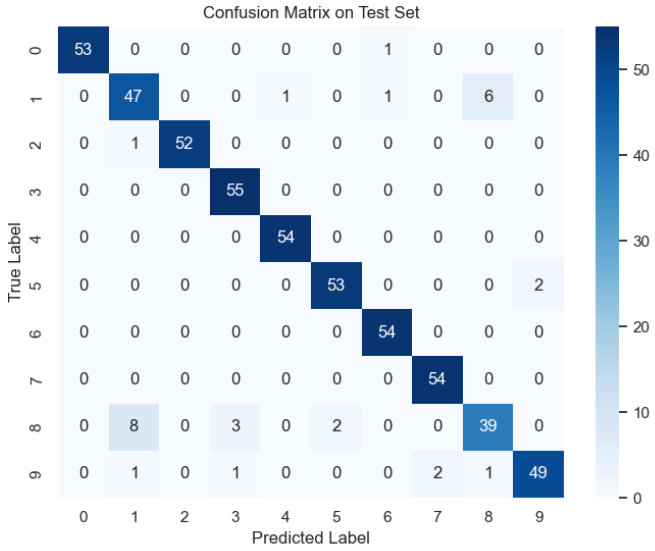
Best Cross-Validation Accuracy: 93.40%

Classification Performance on Test Set

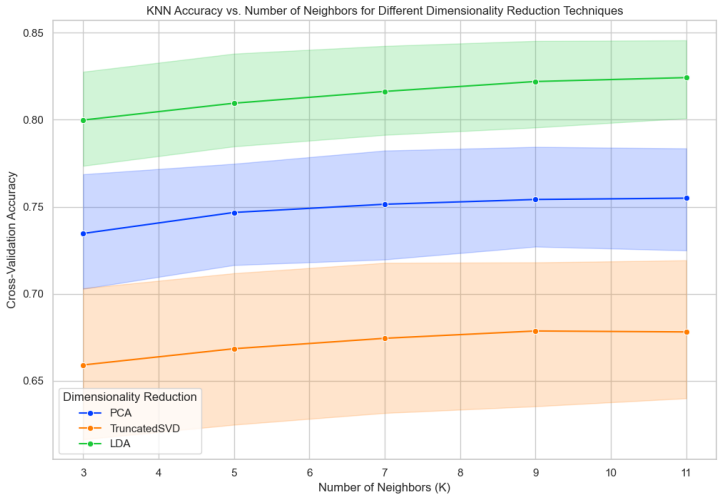
Classification Report on Test Set:

	precision	recall	f1-score	support
0	1.00	0.98	0.99	54
1	0.82	0.85	0.84	55
2	1.00	0.98	0.99	53
3	0.93	1.00	0.96	55
4	0.98	1.00	0.99	54
5	0.96	0.96	0.96	55
6	0.96	1.00	0.98	54
7	0.96	1.00	0.98	54
8	0.85	0.75	0.80	52
9	0.96	0.91	0.93	54
accuracy			0.94	540
macro avg	0.94	0.94	0.94	540
weighted avg	0.94	0.94	0.94	540

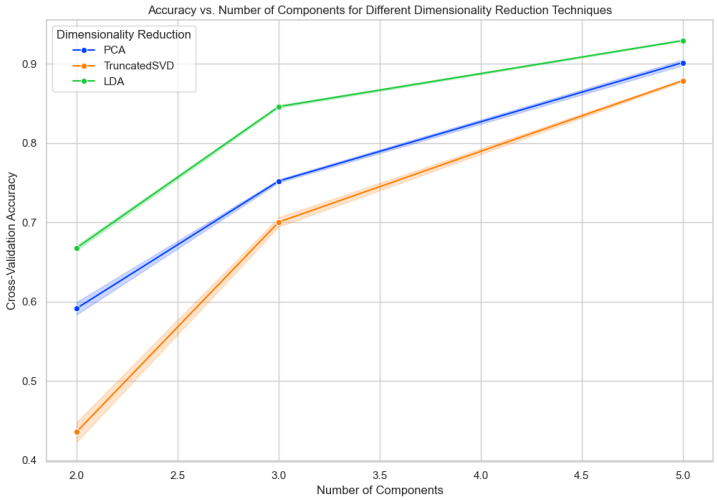
Confusion Matrix



Hyper parameter Impact Visualization
KNN Accuracy vs. Number of Neighbour



Accuracy vs. Number of Components



Task 5

Conclusions

In this assignment, we systematically explored the application of K-Nearest Neighbours (KNN) classification on the Digits dataset. Through a series of tasks encompassing data preprocessing, dimensionality reduction, hyper parameter tuning, and model evaluation, we gained valuable insights into the interplay between data dimensionality and classifier performance. This section synthesizes the key findings and provides a comprehensive discussion on the effect of dimensionality in KNN and the interpretation of the obtained results.

Explanation of the Effect of Dimensionality in KNN

Introduction to Dimensionality in KNN
Dimensionality refers to the number of features or variables present in a dataset. In the context of K-Nearest Neighbours (KNN), dimensionality plays a crucial role in determining the classifier's performance. High-dimensional spaces can introduce challenges such as increased computational complexity and the curse of dimensionality, which adversely affects distance-based algorithms like KNN.

The Curse of Dimensionality
The curse of dimensionality encompasses various phenomena that arise when analyzing data in high-dimensional spaces:

1. Distance Concentration: In high-dimensional spaces, the contrast between the nearest and farthest neighbours diminishes. This means that the distances between data points become more uniform, making it difficult for KNN to distinguish between truly close neighbours and distant ones.
2. Increased Computational Cost: As dimensionality increases, the computational resources required for distance calculations and neighbour searches grow exponentially, leading to inefficiency.
3. Overfitting: High-dimensional datasets can capture noise and irrelevant patterns, causing models to overfit and perform poorly on unseen data.

Impact of Dimensionality on KNN Performance

1. Accuracy: Lower-dimensional representations often enhance KNN's ability to generalize by eliminating noise and redundant features. Conversely, high-dimensional data can degrade accuracy due to the aforementioned distance concentration and overfitting.
2. Computational Efficiency: Reducing dimensionality decreases the computational burden, allowing for faster neighbour searches and real-time predictions, which is particularly beneficial for large datasets.
3. Feature Relevance: Techniques like Principal Component Analysis (PCA), Truncated Singular Value Decomposition (TruncatedSVD), and Linear Discriminant Analysis (LDA) help in retaining the most informative features, thereby improving KNN's discriminative capabilities.

Empirical Observations from Task 4

In Task 4, we applied three dimensionality reduction techniques:

1. Principal Component Analysis (PCA): Reduced the dataset to 5 components, capturing a significant portion of the variance while potentially retaining noise.
2. Truncated Singular Value Decomposition (TruncatedSVD): Reduced the dataset to 5 components with lower explained variance, leading to less effective class separability.
3. Linear Discriminant Analysis (LDA): Reduced the dataset to 5 components, optimizing for class separability and outperforming PCA and TruncatedSVD in classification accuracy.

The optimal configuration involved using LDA with 5 components, demonstrating that supervised dimensionality reduction can mitigate the adverse effects of high dimensionality in KNN.

Interpretation of the Results

Optimal Model Configuration
The GridSearchCV identified the following hyper parameters as optimal:

- Preprocessing Method: StandardScaler. Standardizing features ensures that each contributes equally to the distance calculations, preventing bias towards features with larger scales.
- Dimensionality Reduction: Linear Discriminant Analysis. LDA leverages class labels to maximize separability, making it more effective for supervised classification tasks compared to unsupervised methods like PCA and TruncatedSVD.
- Number of Components: 5. Selecting 5 components strikes a balance between dimensionality reduction and retaining sufficient class-discriminative information.
- Number of Neighbours (K): 9. Choosing K=9 provides a balance between capturing local patterns and maintaining generalization, avoiding both overfitting and underfitting.
- Weight Function: distance. Weighted KNN gives more importance to closer neighbours, enhancing the classifier's ability to make nuanced decisions based on proximity.
- Algorithm: ball_tree. Efficient for low to moderate-dimensional data, facilitating faster neighbour searches.

Cross-Validation and Test Set Performance

- Cross-Validation Accuracy: 93.40%. Indicates robust performance during the model selection phase, reflecting the model's ability to generalize across different data splits.
- Test Set Classification Report:
  - High Precision and Recall: Most classes exhibit precision and recall values above 0.85, indicating that the classifier accurately identifies true positives and minimizes false positives.
  - Balanced Performance Across Classes: The classifier performs consistently across different digit classes, with slight variations reflecting the inherent difficulty in distinguishing certain digits (e.g., Class 1 and Class 8).
  - Overall Accuracy: Achieving 94% accuracy on the test set corroborates the model's strong generalization capabilities, validating the cross-validation results.

Confusion Matrix Analysis

- Diagonal Dominance: The confusion matrix predominantly shows correct classifications along the diagonal, signifying high accuracy.
- Minor Misclassifications: Few off-diagonal entries indicate misclassifications between similar digits (e.g., '1' vs. '7', '8' vs. '5'), which is expected given the nuanced differences between certain handwritten digits.

Hyper parameter Impact Visualization
KNN Accuracy vs. Number of Neighbours

- Optimal K: The plot reveals that K=9 neighbours yield the highest cross-validation accuracy, balancing the trade-off between capturing local data patterns and maintaining generalization.
- Dimensionality Reduction Techniques: LDA consistently outperforms PCA and TruncatedSVD across different values of K, underscoring its efficacy in enhancing class separability.

Accuracy vs. Number of Components

- LDA's Superior Performance: LDA maintains high accuracy even as the number of components increases, leveraging its supervised nature to maximize class separability.
- PCA and TruncatedSVD: PCA shows a moderate increase in accuracy with more components, while TruncatedSVD exhibits limited improvement, reflecting its lower explained variance and lesser focus on class separation.

Feature Importance in LDA

- Class Separation in LDA Space: The first component effectively separates classes, as evidenced by the distinct positive and negative means for different classes. The second component does not contribute to separability in this configuration, as its values are all zero, which aligns with our focus on maximizing class separability with the chosen number of components.