Handwritten Digit Classification with the MNIST Dataset

Objective: To apply various preprocessing and machine learning techniques to classify handwritten digits from the MNIST (alternative link) dataset without using deep learning methods. Additionally, it explores advanced preprocessing, interpretability, and scalability techniques for a comprehensive learning experience.

Project Description: The MNIST dataset is a collection of 70,000 grayscale images of handwritten digits (0-9), 60,000 train and 10,000 test datasets. Each image is 28x28 pixels. The goal of this project is to build a machine learning model that can accurately classify these images into the correct digit categories. Additionally, the project will explore advanced techniques such as data augmentation, dimensionality reduction, and hyperparameter tuning.

Part 1: Data Preprocessing

1. Data Loading:
   - Use fetch_openml from scikit-learn to load the MNIST dataset.
   - Explore the dataset to understand its structure (number of samples, feature dimensions, etc.).
2. Data Normalization:
   - Normalize the pixel values to a range of 0 to 1 to ensure uniformity in the input data scale.
3. Data Visualization:
   - Visualize a subset of images using matplotlib to get a sense of what the handwritten digits look like.
4. Dimensionality Reduction (Optional):
   - Apply PCA to reduce the dimensionality of the dataset while preserving most of the variance.
5. Data Augmentation (Optional):

- Perform transformations such as rotations, scaling, and translations to enhance dataset variability.

Part 2: Machine Learning Models

1. Model Selection:
   - Choose several machine learning algorithms (e.g., SVM, Decision Trees, Random Forest, Gradient Boosting, k-NN, etc.).
2. Model Training:
   - Split the dataset into training and testing sets. (60,000 train and 10,000 test)
   - Train each model on the training set.
3. Model Evaluation:
   - Evaluate each model's performance on the testing set using metrics such as accuracy, precision, recall, and F1-score.
   - Use a confusion matrix to visualize the performance of each model.

Part 3: Hyperparameter Tuning

1. Grid Search:
   - Perform grid search cross-validation to find the optimal hyperparameters for each model.
2. Advanced Optimization:
   - Experiment with Bayesian Optimization using libraries like Optuna.
   - Discuss trade-offs between model complexity and overfitting risks.

Part 4: Reporting

1. Results Compilation:
   - Compile the results from each model into a report.

- Include visualizations such as ROC curves and Precision-Recall curves.

2. Discussion:
   - Discuss the performance of each model and the impact of preprocessing techniques.
   - Provide insights on which model worked best and hypothesize why.
3. Error Analysis:
   - Analyze misclassified samples to identify common patterns or ambiguities.
4. Comparison with Benchmarks:
   - Compare the results with benchmark accuracy for traditional ML methods on MNIST.
5. Interpretable Models (Optional):
   - Use interpretability techniques like SHAP or LIME to understand the decision-making process of models.

Deliverables:

- ***We have no deliverables in this course—no need to submit anything.***

This project will give you hands-on experience with the fundamental steps of a machine learning project, from preprocessing to model evaluation. Encourage you to document your process and findings, as this will help you understand the impact of each step on the final model performance. Good luck to you!