# MNIST Classification using PCA and Decision Trees

Tejash Prasad
2022539
SML Assignment 3

***Abstract*—This report presents a method for classifying digits 0, 1, and 2 from the MNIST dataset using Principal Component Analysis (PCA) for dimensionality reduction and Decision Trees for classification. The code structure and functionality, including data loading, preprocessing, model training, evaluation, and bagging, are detailed along with explanations of each component.**

## I. Introduction

The MNIST dataset is a widely used benchmark for handwritten digit classification tasks. This project aims to classify digits 0, 1, and 2 from the MNIST dataset using PCA and Decision Trees. The methodology involves preprocessing the data, performing PCA for dimensionality reduction, implementing Decision Trees for classification, and employing bagging to enhance model performance.

## II. Code Explanation and Functionality

### A. Data Loading and Preprocessing

The code begins by loading the MNIST dataset using NumPy's `np.load()` function. Training and testing images along with their corresponding labels are extracted. Data belonging to classes other than 0, 1, and 2 are filtered out. The filtered data is reshaped into a suitable format for further processing.

### B. Principal Component Analysis (PCA)

PCA is performed on the training data to reduce its dimensionality while retaining most of its variance. The `pca()` function computes principal components and reconstructs the data using a specified number of components. The top principal components are selected based on explained variance.

### C. Decision Trees and Gini Impurity

A basic Decision Tree classifier is implemented based on Gini impurity as the splitting criterion. Functions like `getGini()` and `calculateGini()` are used to compute Gini impurity for different splits and features. The splitting feature with the lowest Gini impurity is chosen to create child nodes.

### D. Classification and Evaluation

The trained PCA model and Decision Tree are used to predict the classes of the test data. Class-wise accuracy is computed to evaluate model performance. The accuracy is measured based on correct predictions made on the test data.

### E. Bagging for Model Improvement

Bagging (Bootstrap Aggregating) is employed to improve model robustness and accuracy. Multiple instances of the dataset are created by randomly selecting samples with replacement. Each instance is used to train a separate Decision Tree classifier. Final predictions are made by aggregating predictions from all classifiers. The accuracy of the bagged model is evaluated using class-wise accuracy.

## III. Conclusion

In conclusion, this project successfully classified digits 0, 1, and 2 from the MNIST dataset using PCA and Decision Trees. Bagging was employed to enhance model performance. The results demonstrate the effectiveness of PCA and Decision Trees in handling high-dimensional data and complex classification tasks.

```
Total accuracy: 0.7639021290117572
Class-wise correct: [ 829. 1134.  441.]
Class-wise total: [ 980. 1135. 1032.]
Class-wise accuracy: [0.84591837 0.99911894 0.42732558]
```

The results obtained without bagging

```
Total accuracy: 0.7604067365745154
Class-wise correct: [ 822. 1134.  437.]
Class-wise total: [ 980. 1135. 1032.]
Class-wise accuracy: [0.83877551 0.99911894 0.42344961]
```

The results obtained after bagging