**Amrita Vishwa Vidyapeetham**

**School of Computing, Coimbatore**

**Fifth Semester**

**Department of Computer Science and Engineering**

**19CSE432 Pattern Recognition**

**Case Study**

**Team Members:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Registration No** | **Name** | **Email ID** | **Contribution** |
| CB.EN.U4CSE21053 | Samya Rebecca | [Cb.en.u4cse21053@cb.students.amrita.edu](mailto:Cb.en.u4cse21007@cb.students.amrita.edu) | Parametric and non parametric techniques, preprocessing MNIST, k-means, naive bayes, gaussian mixture model, LDA, PCA, ROC, evaluation metrics, hyperparameter tuning, error curve, challenges |
| CB.EN.U4CSE21460 | Srutija | [Cb.en.u4cse21460@cb.students.amrita.edu](mailto:Cb.en.u4cse21008@cb.students.amrita.edu) | Description of dataset, parzen window, hidden markov model, dimensionality reduction using PCA, evaluation metrics, posterior probability, challenges |

**DESCRIPTION OF DATASET:**

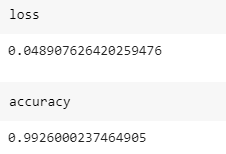
The MNIST dataset is a widely used benchmark dataset in machine learning and computer vision, consisting of grayscale images of handwritten digits (0-9). It's a collection of 28x28 pixel images, totaling 70,000 labeled examples, with 60,000 images for training and 10,000 for testing/validation.

* **Image Size**: Each image in the dataset is a 28x28 grayscale image, meaning each pixel has a value between 0 (black) and 255 (white), representing the intensity of the pixel.
* **Labeling**: Every image is associated with a label, indicating the digit it represents (0-9).

Recognizing handwritten digits in the MNIST dataset is a classic pattern recognition problem.

**The following are the steps we followed to use Python and Keras to train a simple Convolutional Neural Network (CNN) for handwritten digit recognition:**

* Loads the MNIST dataset using Keras.
* Preprocesses the data by reshaping and normalizing the pixel values to be between 0 and 1.
* Builds a simple CNN model using Keras' Sequential API.
* Compiles the model with an optimizer, loss function, and metrics.
* Trains the model on the training data for a specified number of epochs.
* Evaluates the trained model on the test data to calculate its accuracy.



**Other datasets used:**

**Daily activity metrics:** It records and describes the day to day activities of the user, like the step count, distance walked, heart points, duration.

**BMI\_train:** Dataset contains height and weight of people and the BMI index is calculated.

**Parametric and non-parametric techniques:**

In the context of pattern recognition, parametric and non-parametric techniques refer to different approaches for modeling and classifying patterns. Here's a brief overview of the differences between parametric and non-parametric techniques, along with examples:

**Parametric Techniques:**

Definition: Parametric techniques make assumptions about the underlying distribution of the data and use a fixed number of parameters to describe that distribution.

Examples:

Linear Discriminant Analysis (LDA): Assumes that the features follow a Gaussian distribution and estimates the parameters (mean and covariance) for each class.

Naive Bayes (Bayesian classifier): Assumes independence between features within each class and estimates probabilities based on this assumption.

SVM: The primary goal of an SVM is to find a hyperplane in an N-dimensional space. It includes concepts like linear separation, support vectors, kernel function.

**Non-parametric Techniques:**

Definition: Non-parametric techniques do not make strong assumptions about the underlying distribution of the data. They are more flexible and can adapt to different types of distributions.

Examples:

k-Nearest Neighbors (KNN): Classifies a data point based on the majority class among its k nearest neighbors, without assuming any specific distribution.

K-means: It partitions data into K clusters by iteratively assigning data points to the cluster with the nearest centroid and updating centroids based on the mean of assigned points. It seeks to minimize the sum of squared distances between data points and their assigned cluster centroids.

Agglomerative clustering:

is a hierarchical clustering algorithm that starts with individual data points as separate clusters and iteratively merges the closest clusters until only one cluster remains. It uses a linkage criterion (e.g., average linkage or complete linkage) to measure the distance between clusters and forms a tree-like hierarchy known as a dendrogram.

Decision Trees: Builds a tree structure to make decisions based on the data, without assuming a particular distribution.

Support Vector Machines (SVM): While SVM can be considered parametric in some contexts, it is often used with non-linear kernel functions, making it more flexible and less reliant on distribution assumptions.

Key Differences:

Flexibility: Parametric methods are more rigid, assuming a specific form for the underlying distribution, while non-parametric methods are more flexible and can adapt to different types of data.

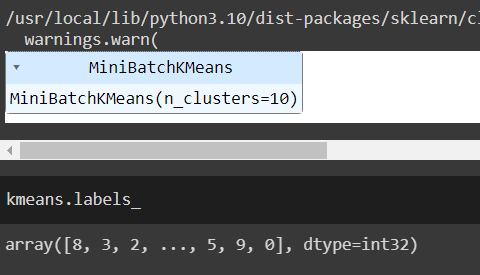
Assumptions: Parametric methods make assumptions about the form of the data distribution, which may or may not be accurate. Non-parametric methods make fewer assumptions, allowing them to handle a wider range of data distributions.

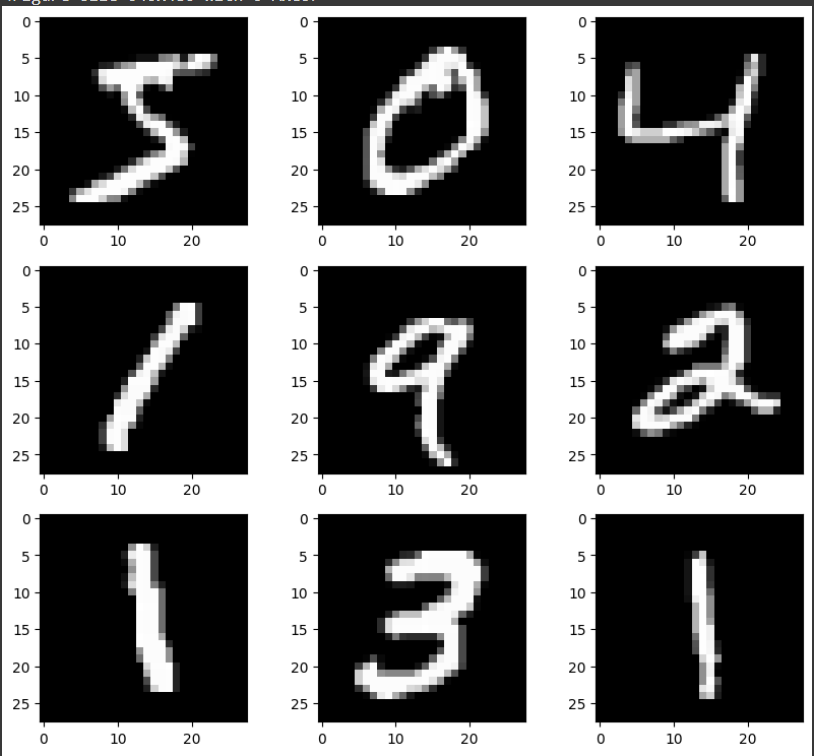
Data Efficiency: Parametric methods may be more data-efficient when the underlying assumptions hold, but they can perform poorly if the assumptions are violated. Non-parametric methods are generally more robust in the face of diverse data.

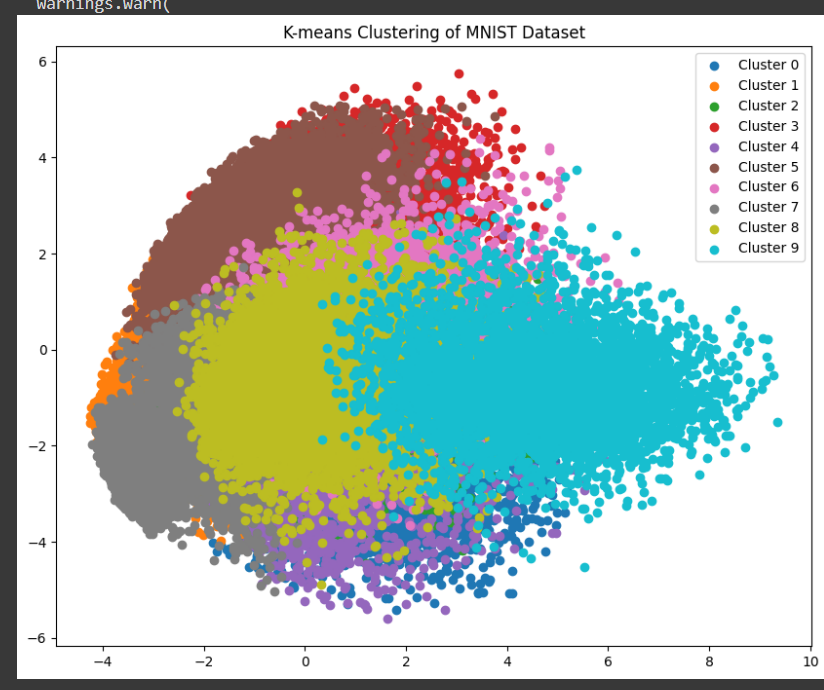
In practice, the choice between parametric and non-parametric techniques depends on the characteristics of the data and the problem at hand. If the underlying distribution is well-known and the assumptions are justified, parametric methods can be more efficient. However, if there is uncertainty about the data distribution or if it is non-standard, non-parametric methods may be preferred.

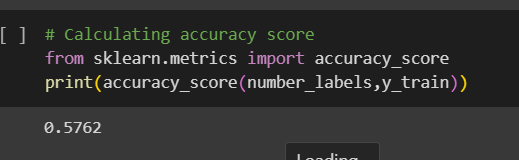
**K-means:**

Partitions a dataset into K clusters based on similarity patterns among data points.





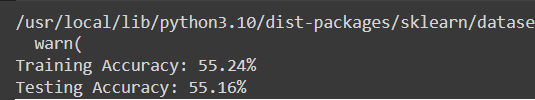


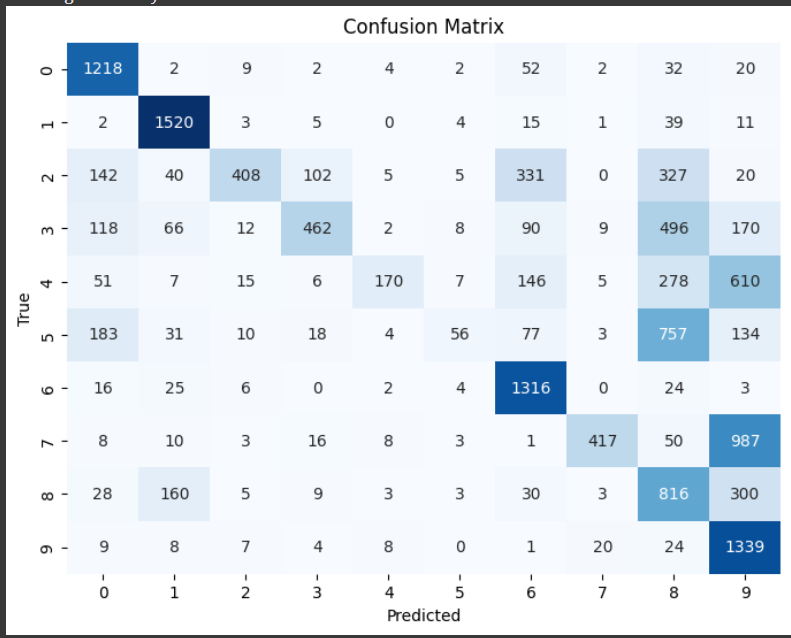


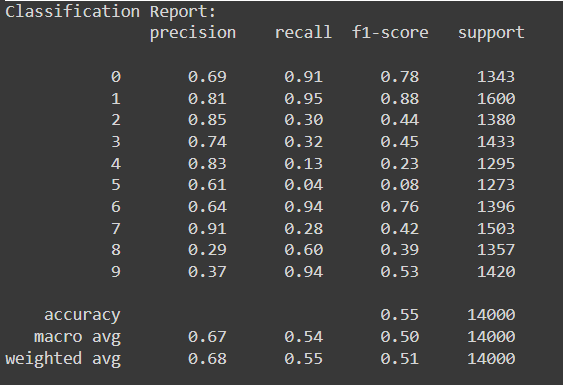
**Naïve bayes:**

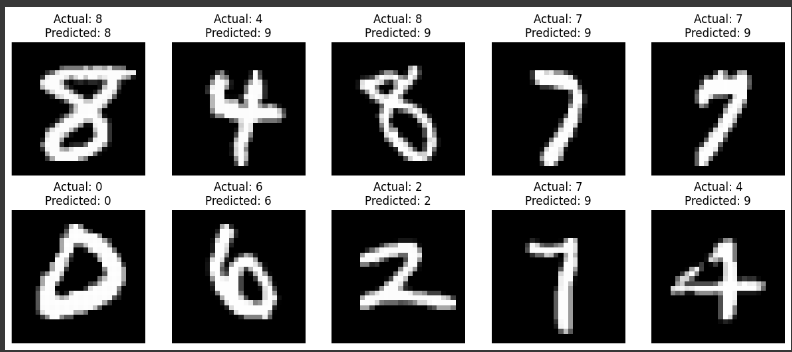
Supervised machine learning algorithm, which is used for classification tasks, like text classification.

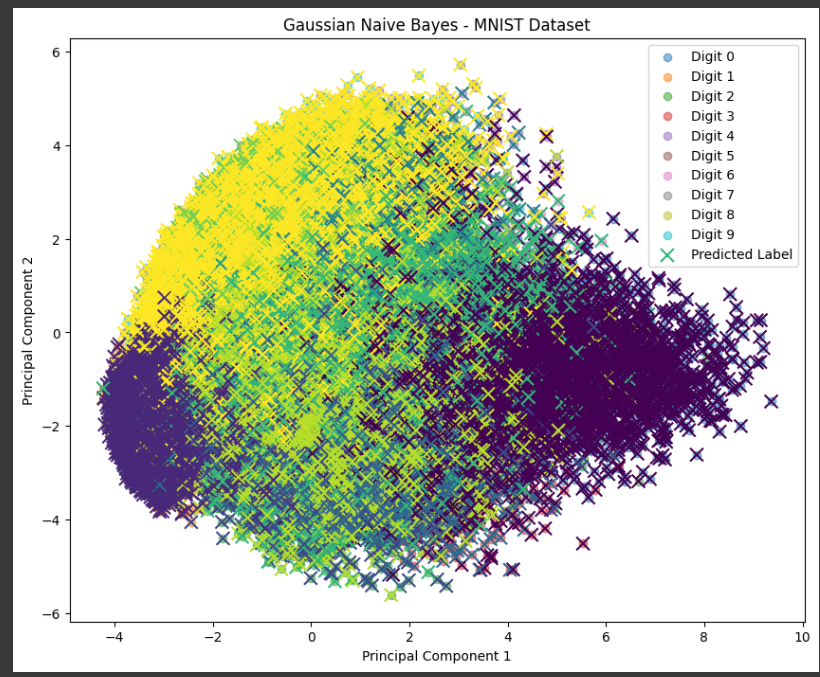
It is a collection of classification algorithms based on Bayes’ Theorem. It is not a single algorithm but a family of algorithms where all of them share a common principle, i.e. every pair of features being classified is independent of each other.



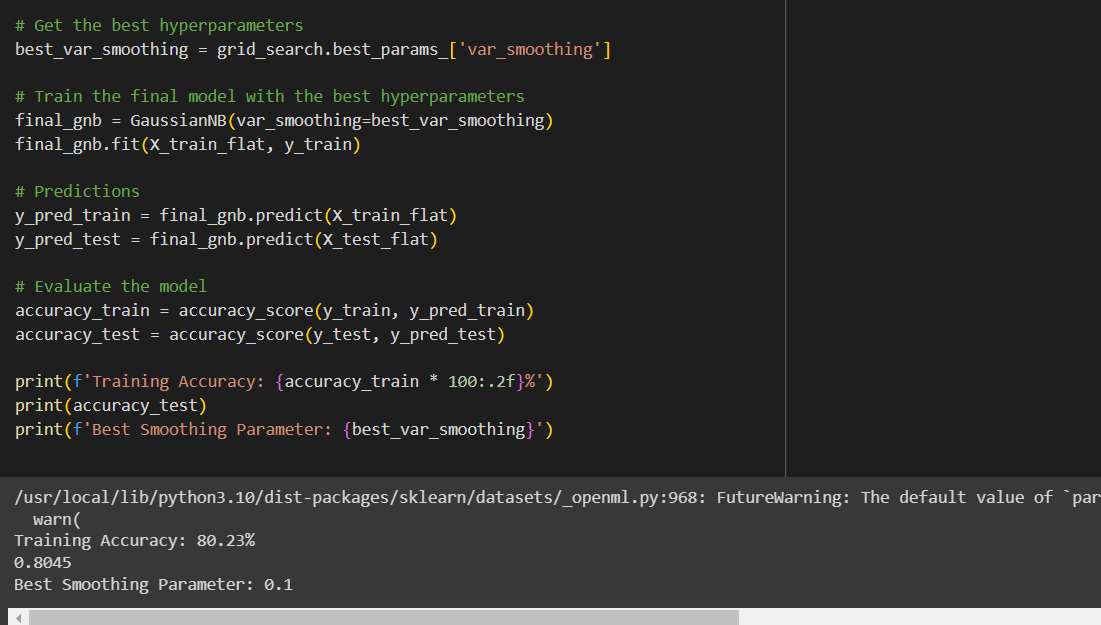






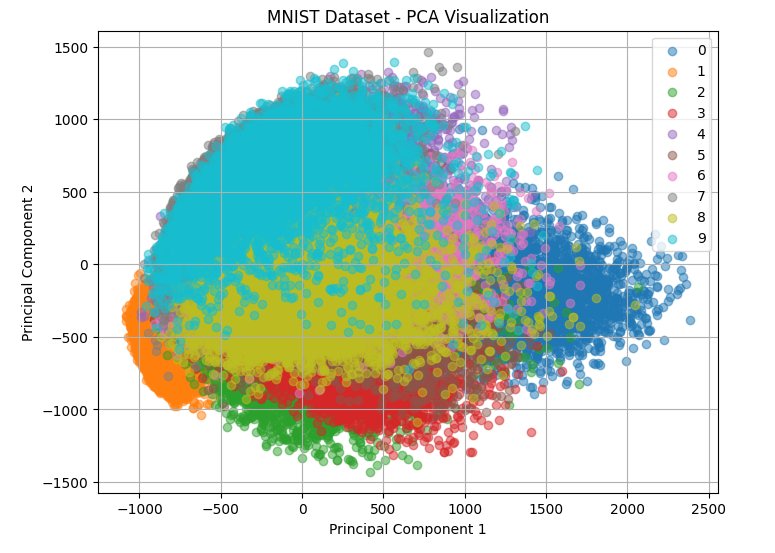


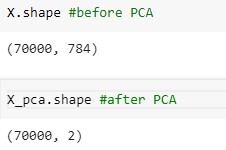
**Hyper Parameter Tuning:**



**DIMENSIONALITY REDUCTION USING PCA:**

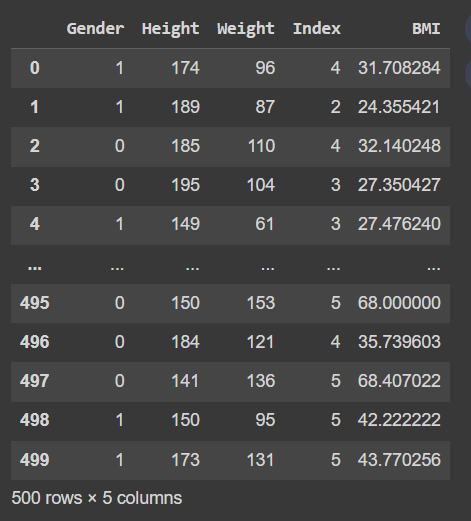
Dimensionality reduction using PCA (Principal Component Analysis) is a technique used to reduce the number of features (or dimensions) in a dataset while preserving the most important information.

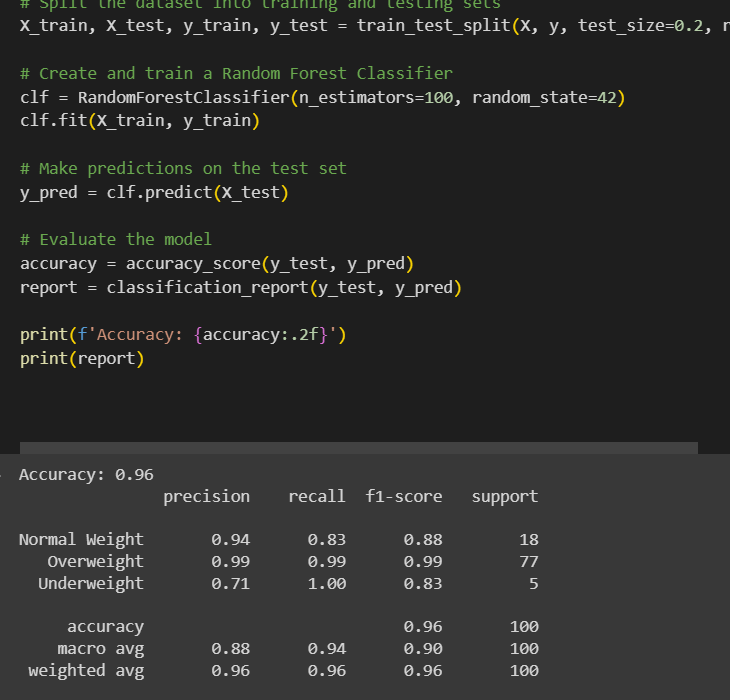


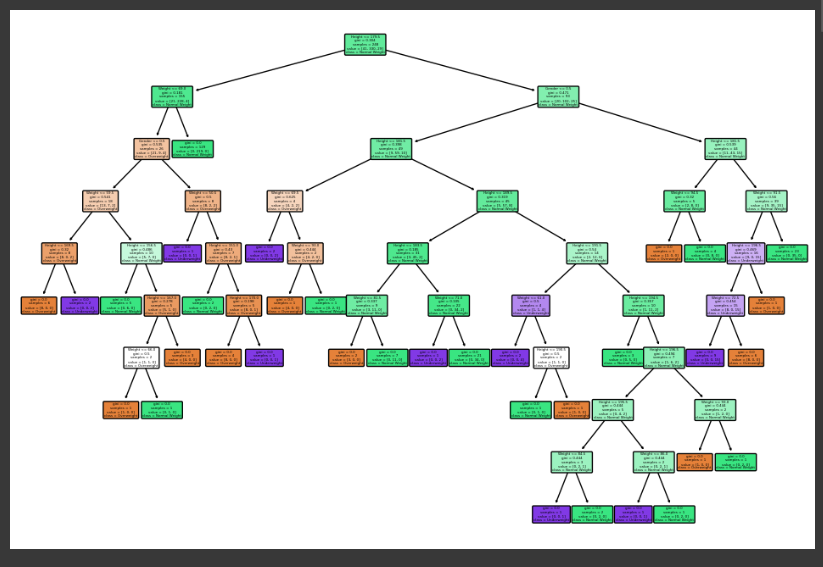


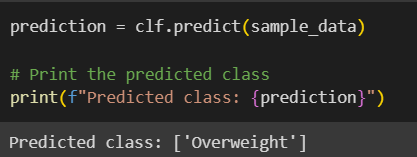
**ROC\_LDA using Random Forest classifier:**

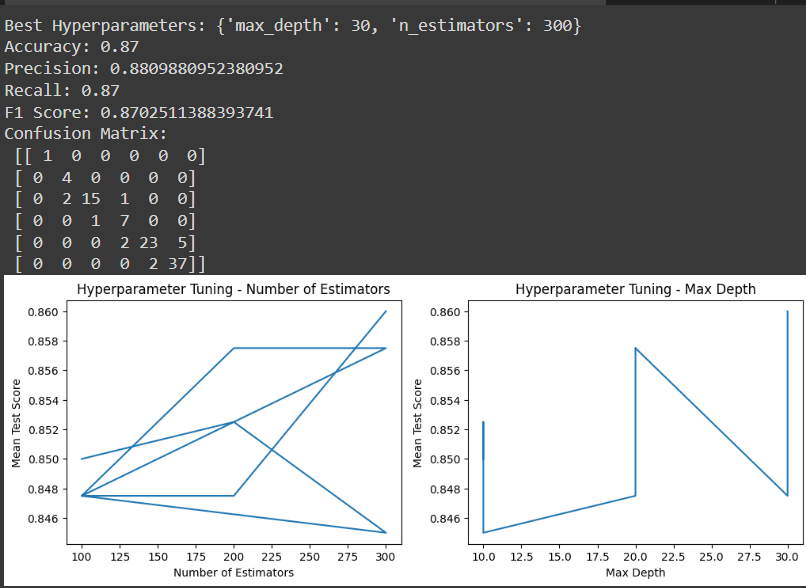
BMI\_train dataset:



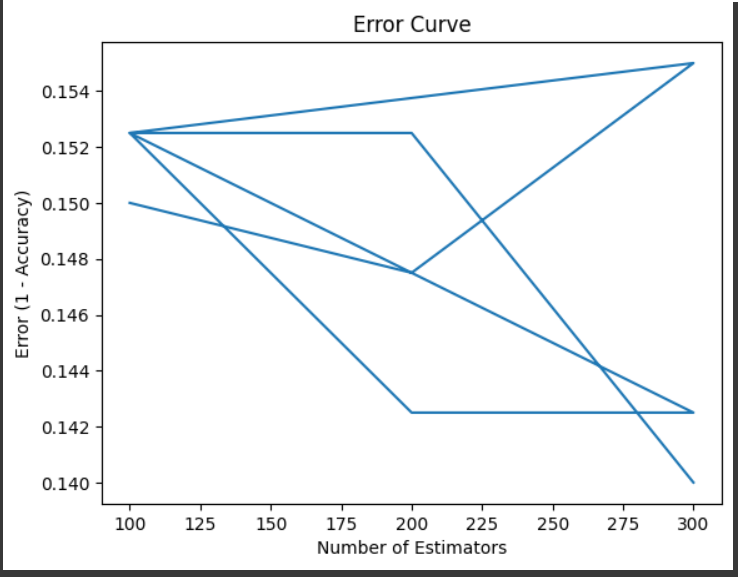


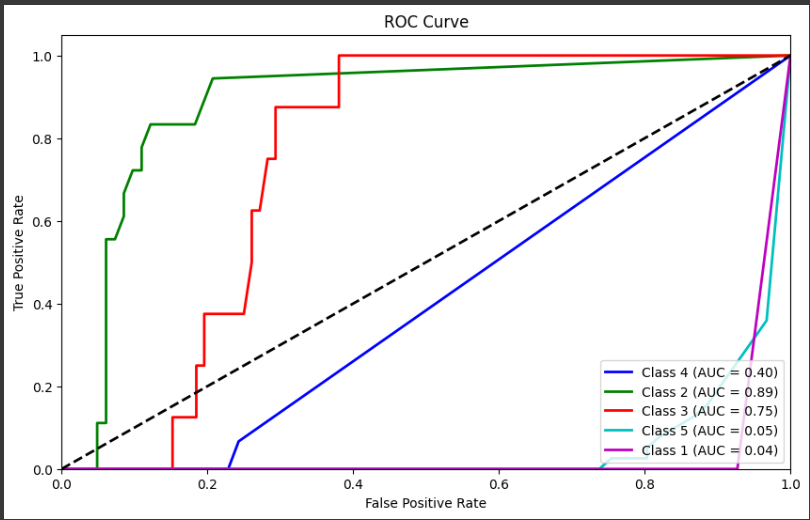








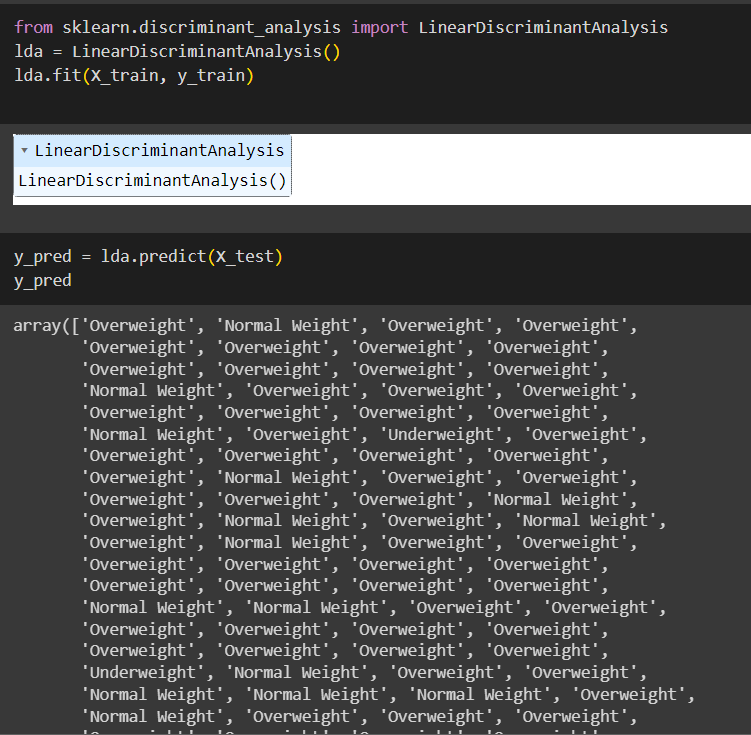


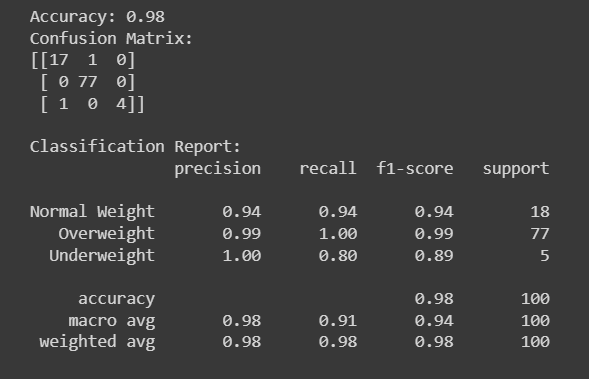


**Linear Discriminant Analysis:**

Generalization of Fisher's linear discriminant, a method used in statistics and other fields, to find a linear combination of features that characterizes or separates two or more classes of objects or events.

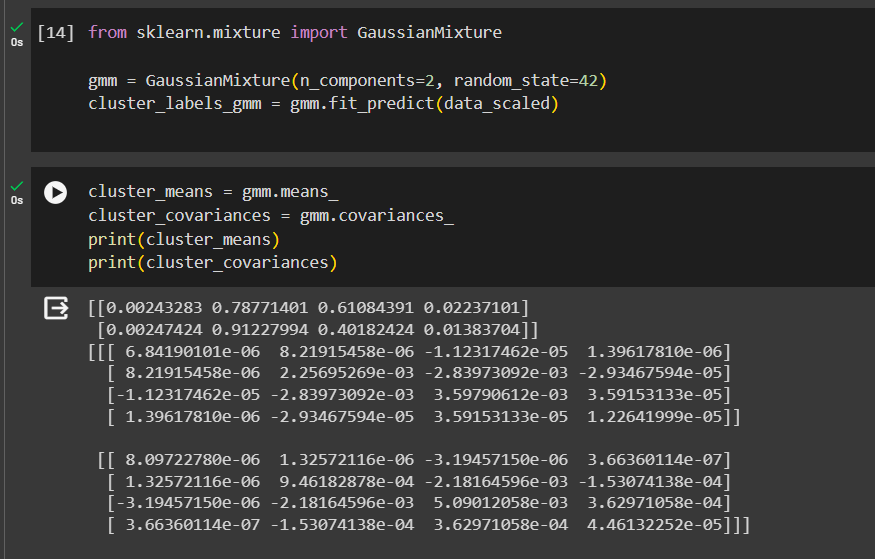
LDA works by projecting the data onto a lower-dimensional space that maximizes the separation between the classes. It does this by finding a set of linear discriminants that maximize the ratio of between-class variance to within-class variance. In other words, it finds the directions in the feature space that best separate the different classes of data.

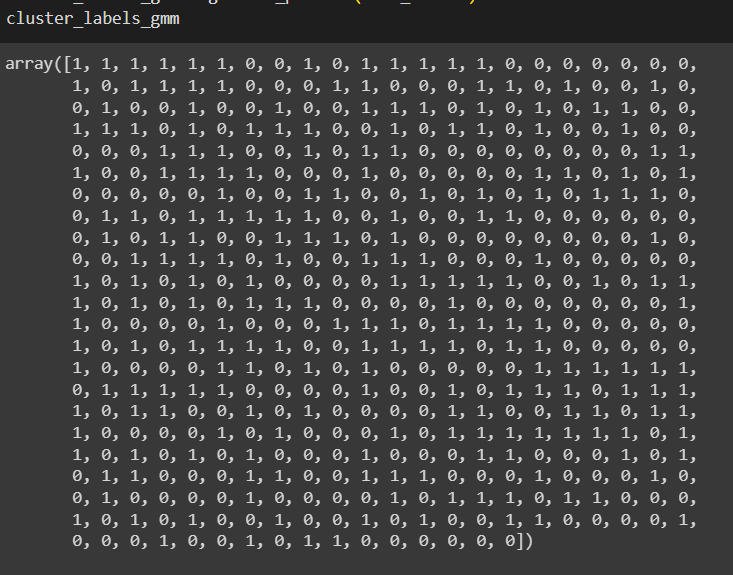




**Gaussian Mixture Model:**

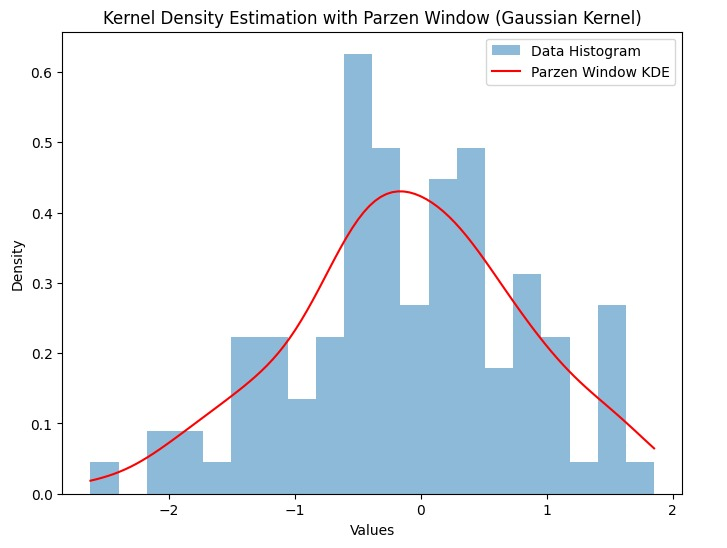
It is a probabilistic model that represents a mixture of Gaussian (normal) distributions. It is a generative probabilistic model that assumes that the data is generated by a mixture of several Gaussian distributions with unknown parameters. Each Gaussian distribution in the mixture is termed a component, and the GMM models the overall distribution as a weighted sum of these components.





**PARZEN WINDOW:**

The Parzen Window method is a non-parametric technique used for density estimation. It approximates the probability density function of a random variable based on its observed data.



**Values:**

[0.01853683758633841, 0.02048648549380472, 0.022658456089541992, 0.02509402111141079, 0.027832356966219822, 0.030907026256946506, 0.03434270324007762...

**HIDDEN MARKOV MODEL:**

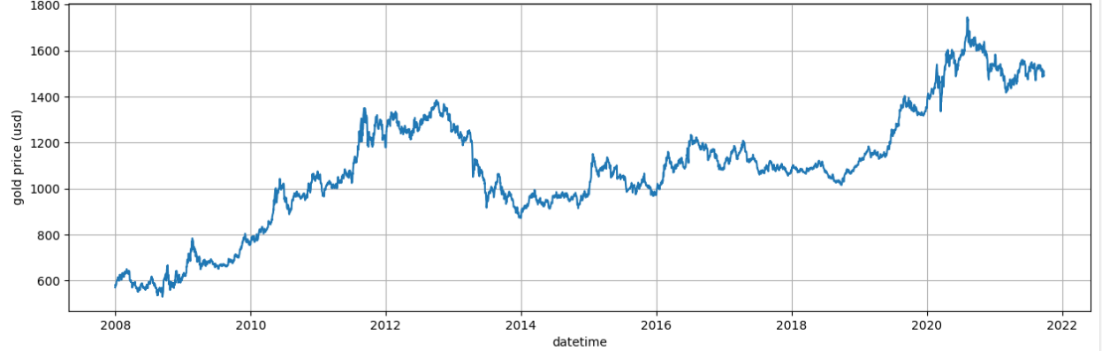
Implementing a Hidden Markov Model (HMM) directly on the MNIST dataset might not be the most suitable approach. HMMs are typically applied in sequential data modeling, particularly for time series or sequential observations where the underlying process can be modeled as a sequence of hidden states transitioning over time.

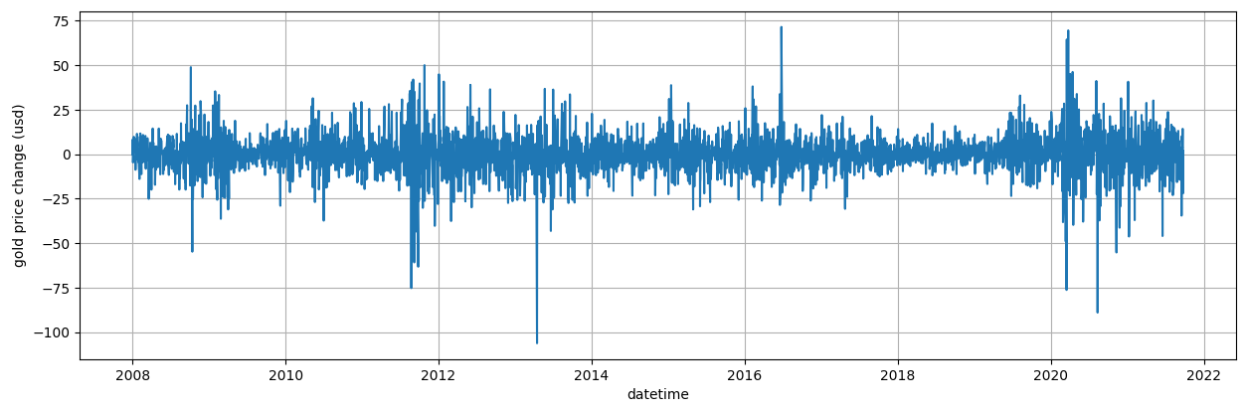
The MNIST dataset consists of static images of handwritten digits, and the pixels' spatial arrangement is not a temporal sequence. Therefore, using an HMM directly on the entire MNIST dataset wouldn't be an appropriate choice. (**challenge**)

**Sequential data** refers to any kind of data where the order or sequence of observations matters and carries important information. It's characterized by a sequence of elements or events that are dependent on each other and have a temporal or sequential relationship.

So, the dataset we were previously using is not the best to run this model. Hence, we are going to use the following dataset: <https://github.com/natsunoyuki/Data_Science/blob/master/gold/gold/gold_price_usd.csv?raw=True>

It shows the change in gold prices over the years.





Unique states: [1 0 2]

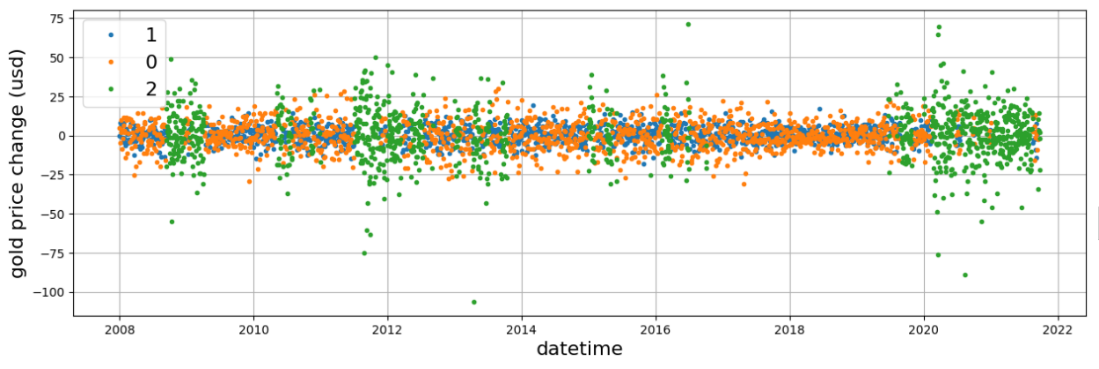
Start probabilities: [2.25217394e-05 9.99977478e-01 2.27517901e-54]

Transition matrix: [[1.18601839e-02 9.87982221e-01 1.57595016e-04] [7.39251441e-01 2.11663333e-01 4.90852259e-02] [6.21144512e-02 1.76534234e-02 9.20232125e-01]]

Gaussian distribution means: [[0.18754376] [0.28531535] [0.29985414]]

Gaussian distribution covariances: [[[ 84.22877262]] [[ 29.75024958]] [[321.61159569]]]







**CHALLENGES:**

**DBSCAN:**

ValueError: Expected 2D array, got scalar array instead: array=<module 'keras.api.\_v2.keras.datasets.mnist' from '/usr/local/lib/python3.10/dist-packages/keras/api/\_v2/keras/datasets/mnist/\_\_init\_\_.py'>. Reshape your data either using array.reshape(-1, 1) if your data has a single feature or array.reshape(1, -1) if it contains a single sample.

While the MNIST dataset is widely used for digit recognition and is suitable for many machine learning models, there are **scenarios where certain types of models may not be the most appropriate or effective.**

Some complex models, especially those with a large number of parameters, may require more memory and computational resources than are necessary for the MNIST dataset. Training large models may not be justified for a dataset of this size.

**Overfitting**: Due to its small size, the MNIST dataset is susceptible to overfitting. Models trained on MNIST may achieve very high accuracy on the test set but may struggle when applied to new and unseen data. It's important to use appropriate regularization techniques and consider more complex datasets to address this challenge.

**Generalization to Other Tasks**: MNIST is specific to digit recognition. While it's a valuable benchmark for evaluating the performance of machine learning models, it may not generalize well to other computer vision tasks. Practitioners often face the challenge of adapting models trained on MNIST to more diverse image datasets.

**Availability of Larger Datasets**: MNIST's limited size becomes a bottleneck. Researchers often seek larger and more diverse datasets to train models effectively.

**Data Augmentation**: Generating diverse training examples is crucial for training robust models. With MNIST, data augmentation techniques like rotation, scaling, and translation may not be as impactful as with more complex datasets.

**Class Imbalance**: While MNIST has a balanced distribution of digits, some real-world datasets may exhibit class imbalance. Practitioners may need to handle class imbalance issues and choose appropriate evaluation metrics in scenarios where certain classes are underrepresented.