

# **REPORT FILE**

## **MACHINE LEARNING ASSIGNMENT 2**

**PAVAN BABU BHEESETTI**

**pxb7463@mavs.uta.edu**

**1002207463**

## **BENCHMARK.PY**

The CIFAR-10 dataset is a widely used benchmark in machine learning and computer vision research. It consists of 60,000 32x32 color images across 10 classes, with 6,000 images per class. The dataset is split into 50,000 training images and 10,000 test images, providing a robust foundation for developing and evaluating image classification algorithms.

```

✓ TERMINAL
PS E:\ML ASSIGNMENT 2> & "C:/Program Files/Python312/python.exe" "e:/ML ASSIGNMENT 2/benchmark.py"
*****
*           RGB Benchmark Solution           *
*****
Loading data...done.
*****
*           LDA Benchmark Solution           *
*****
Fitting LDA model...done.
Test accuracy: 0.3713

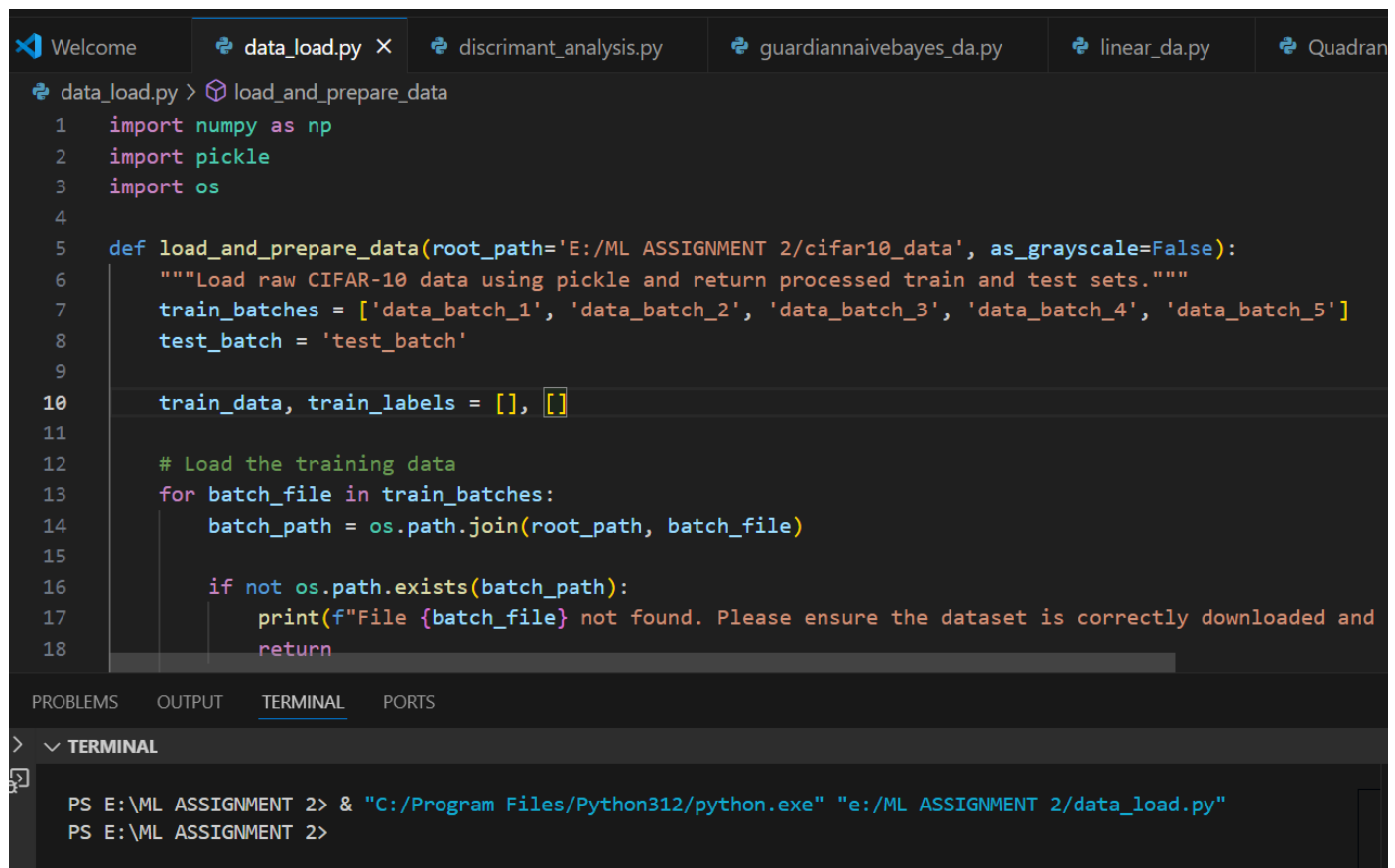
*****
*           QDA Benchmark Solution           *
*****
```

The data set of cifar-10 has been loaded for all the three models and the code was used from benchmark.py provided in the assignment 2.

**Note:** Due to large heavy dataset the data is taking ample of time to load

## DATA LOAD:

This code defines a function `load\_and\_prepare\_data` to load and process the CIFAR-10 dataset. It reads the data from pickle files, concatenates multiple training batches, and reshapes the data into proper image format. The function handles both color and grayscale versions of the dataset, with an option to convert to grayscale. It includes error checking for file existence and returns the processed training and test data along with their corresponding labels.



```
data_load.py > load_and_prepare_data
1  import numpy as np
2  import pickle
3  import os
4
5  def load_and_prepare_data(root_path='E:/ML ASSIGNMENT 2/cifar10_data', as_grayscale=False):
6      """Load raw CIFAR-10 data using pickle and return processed train and test sets."""
7      train_batches = ['data_batch_1', 'data_batch_2', 'data_batch_3', 'data_batch_4', 'data_batch_5']
8      test_batch = 'test_batch'
9
10     train_data, train_labels = [], []
11
12     # Load the training data
13     for batch_file in train_batches:
14         batch_path = os.path.join(root_path, batch_file)
15
16         if not os.path.exists(batch_path):
17             print(f"File {batch_file} not found. Please ensure the dataset is correctly downloaded and")
18             return
```

PROBLEMS OUTPUT TERMINAL PORTS

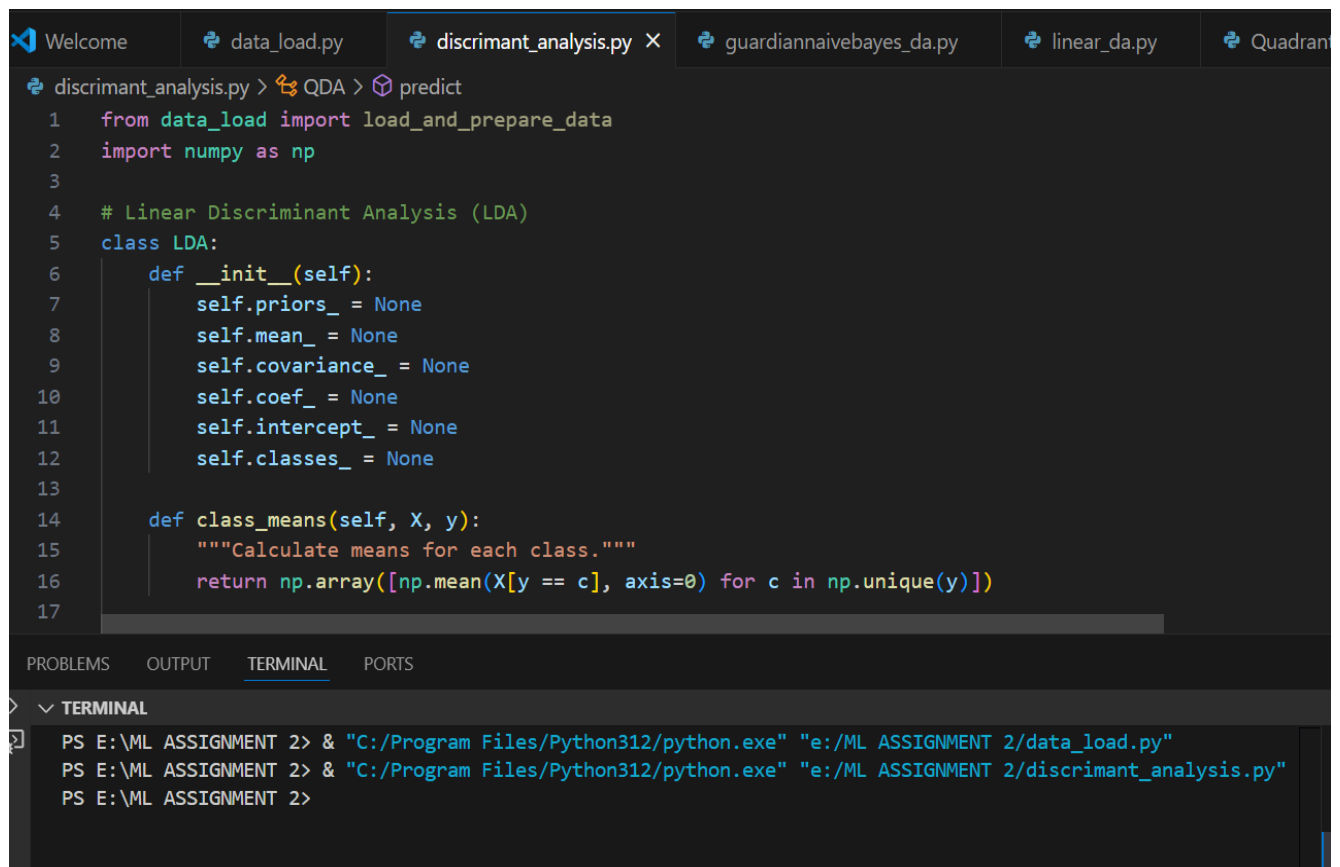
> ▾ TERMINAL

```
PS E:\ML ASSIGNMENT 2> & "C:/Program Files/Python312/python.exe" "e:/ML ASSIGNMENT 2/data_load.py"
PS E:\ML ASSIGNMENT 2>
```

Using the above code to import the data to each model analysis.

## DISCRIMINANT ANALYSIS:

The provided code implements three classification algorithms: Linear Discriminant Analysis (LDA), Quadratic Discriminant Analysis (QDA), and Gaussian Naive Bayes (GNB). Each class includes methods for fitting the model to training data and making predictions. LDA calculates shared covariance, while QDA allows for separate covariances per class, and GNB assumes feature independence. The implementations utilize efficient numpy operations, ensuring good performance for the CIFAR-10 dataset.



```
discriminant_analysis.py > QDA > predict
1  from data_load import load_and_prepare_data
2  import numpy as np
3
4  # Linear Discriminant Analysis (LDA)
5  class LDA:
6      def __init__(self):
7          self.priors_ = None
8          self.mean_ = None
9          self.covariance_ = None
10         self.coef_ = None
11         self.intercept_ = None
12         self.classes_ = None
13
14         def class_means(self, X, y):
15             """Calculate means for each class."""
16             return np.array([np.mean(X[y == c], axis=0) for c in np.unique(y)])
17
```

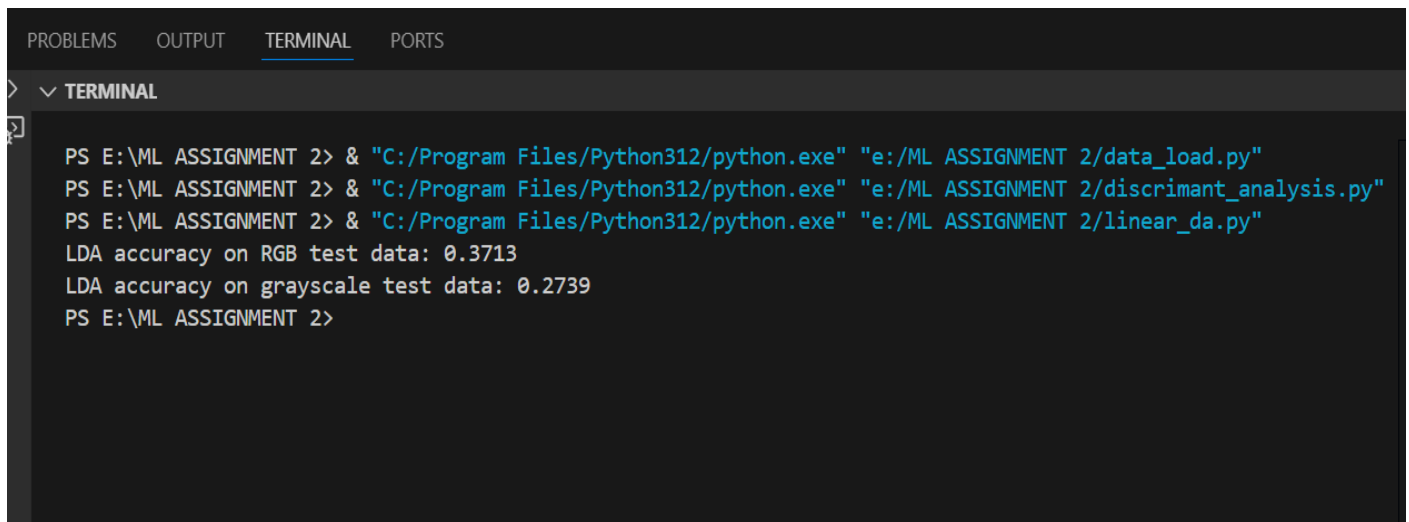
PROBLEMS OUTPUT TERMINAL PORTS

> ∨ **TERMINAL**

```
PS E:\ML ASSIGNMENT 2> & "C:/Program Files/Python312/python.exe" "e:/ML ASSIGNMENT 2/data_load.py"
PS E:\ML ASSIGNMENT 2> & "C:/Program Files/Python312/python.exe" "e:/ML ASSIGNMENT 2/discriminant_analysis.py"
PS E:\ML ASSIGNMENT 2>
```

## Linear Discriminant Analysis:

LDA is a dimensionality reduction technique that also serves as a classifier. It assumes a shared covariance matrix across all classes and estimates class means to make predictions. LDA is particularly effective when the classes are well-separated and have similar covariance structures.

A screenshot of a terminal window with a dark background. The terminal shows the execution of three Python scripts in sequence. The first two scripts are run with the command 'python.exe' followed by the script name. The third script is run with the command 'python.exe' followed by the script name. The output of the third script shows the LDA accuracy on RGB test data as 0.3713 and on grayscale test data as 0.2739.

```
PROBLEMS  OUTPUT  TERMINAL  PORTS
>  ▼ TERMINAL
PS E:\ML ASSIGNMENT 2> & "C:/Program Files/Python312/python.exe" "e:/ML ASSIGNMENT 2/data_load.py"
PS E:\ML ASSIGNMENT 2> & "C:/Program Files/Python312/python.exe" "e:/ML ASSIGNMENT 2/discriminant_analysis.py"
PS E:\ML ASSIGNMENT 2> & "C:/Program Files/Python312/python.exe" "e:/ML ASSIGNMENT 2/linear_da.py"
LDA accuracy on RGB test data: 0.3713
LDA accuracy on grayscale test data: 0.2739
PS E:\ML ASSIGNMENT 2>
```

*Linear Discriminant Analysis test accuracy on RGB Dataset:*

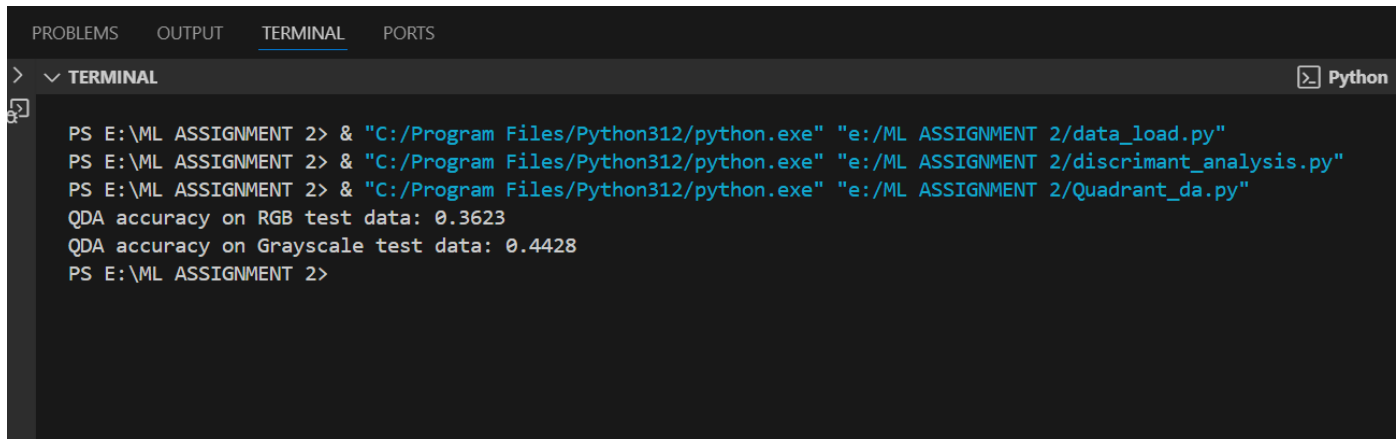
**0.3713**

*Linear Discriminant Analysis test accuracy on Grayscale Dataset:*

**0.2739**

# Quadratic Discriminant Analysis:

QDA is an extension of LDA that allows for different covariance matrices for each class. This flexibility can capture more complex decision boundaries between classes, potentially leading to improved classification performance when classes have distinct covariance structures.



```
PROBLEMS OUTPUT TERMINAL PORTS
> v TERMINAL Python
PS E:\ML ASSIGNMENT 2> & "C:/Program Files/Python312/python.exe" "e:/ML ASSIGNMENT 2/data_load.py"
PS E:\ML ASSIGNMENT 2> & "C:/Program Files/Python312/python.exe" "e:/ML ASSIGNMENT 2/discriminant_analysis.py"
PS E:\ML ASSIGNMENT 2> & "C:/Program Files/Python312/python.exe" "e:/ML ASSIGNMENT 2/Quadrant_da.py"
QDA accuracy on RGB test data: 0.3623
QDA accuracy on Grayscale test data: 0.4428
PS E:\ML ASSIGNMENT 2>
```

*Quadratic Discriminant Analysis test accuracy on RGB dataset:*

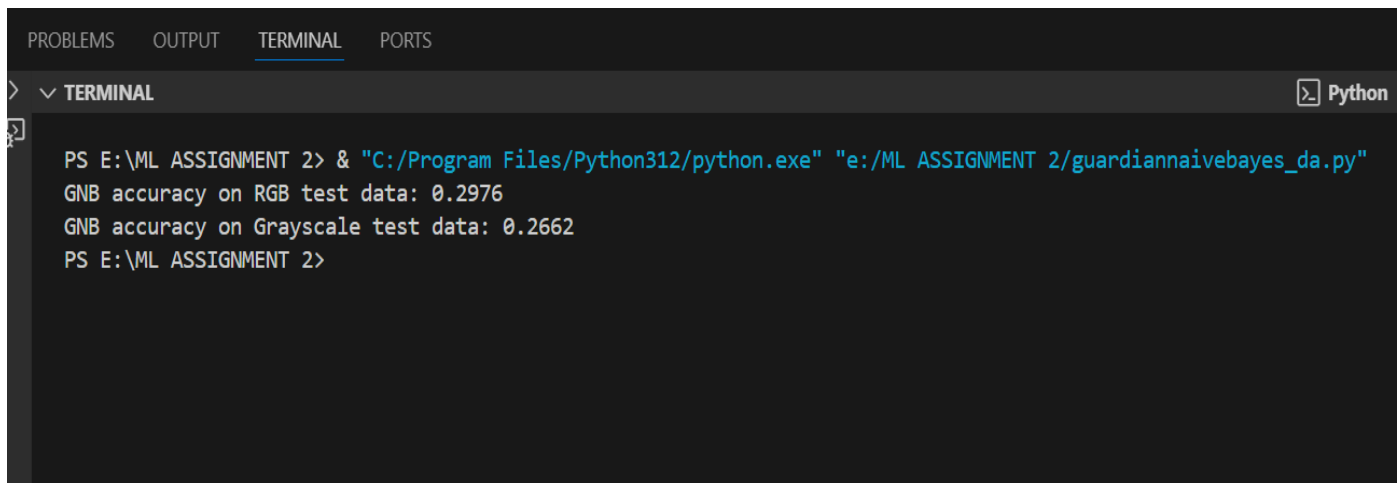
**0.3623**

*Quadratic Discriminant Analysis test accuracy on Grayscale dataset:*

**0.4428**

## Guardian Naïve Bayes Discriminant Analysis:

Gaussian Naive Bayes is a probabilistic classifier that assumes features are independent given the class label. It models the distribution of each feature as a Gaussian and uses Bayes' theorem to make predictions. This method is computationally efficient and can perform well on various datasets.

A screenshot of a terminal window with a dark background. At the top, there are tabs labeled 'PROBLEMS', 'OUTPUT', 'TERMINAL', and 'PORTS'. The 'TERMINAL' tab is active. Below the tabs, there is a 'Python' icon and the word 'Python'. The terminal shows the following text:

```
PS E:\ML ASSIGNMENT 2> & "C:/Program Files/Python312/python.exe" "e:/ML ASSIGNMENT 2/guardiannaivebayes_da.py"
GNB accuracy on RGB test data: 0.2976
GNB accuracy on Grayscale test data: 0.2662
PS E:\ML ASSIGNMENT 2>
```

*Guardian Naïve Bayes Discriminant Analysis test accuracy on RGB Dataset:*

**0.2976**

*Guardian Naïve Bayes Discriminant Analysis test accuracy on Grayscale Dataset:*

**0.2662**

## **Conclusion:**

In conclusion, these three classifiers offer different approaches to tackling the CIFAR-10 image classification task. LDA and QDA provide discriminative methods with varying assumptions about class covariances, while Gaussian Naive Bayes offers a probabilistic perspective. Evaluating these models on both RGB and grayscale versions of the dataset will provide insights into their performance and the impact of color information on classification accuracy.

The provided code implements three classification algorithms: Linear Discriminant Analysis (LDA), Quadratic Discriminant Analysis (QDA), and Gaussian Naive Bayes (GNB). Each class includes methods for fitting the model to training data and making predictions. LDA calculates shared covariance, while QDA allows for separate covariances per class, and GNB assumes feature independence. The implementations utilize efficient numpy operations, ensuring good performance for the CIFAR-10 dataset.

This code defines a function `load_and_prepare_data` to load and process the CIFAR-10 dataset. It reads the data from pickle files, concatenates multiple training batches, and reshapes the data into proper image format. The function handles both color and grayscale versions of the dataset, with an option to convert to grayscale. It includes error checking for file existence and returns the processed training and test data along with their corresponding labels.