**EXPT NO : 5 A python program to implement Multi Layer DATE: 29.9.24 Perceptron With Backpropagation**

# AIM:

To write a python program to implement Multilayer perceptron with backpropagation .

# PROCEDURE:

Implementing Multilayer perceptron with backpropagation using the Keras dataset involve the following steps:

## Step 1: Import Necessary Libraries

First, import the libraries that are essential for data manipulation, visualization, and model building.

# importing modules

import tensorflow as tf import numpy as np

from tensorflow.keras.models import Sequential from tensorflow.keras.layers import Flatten

from tensorflow.keras.layers import Dense

from tensorflow.keras.layers import Activation import matplotlib.pyplot as plt

## Step 2: Load the Keras Dataset

The Keras dataset can be loaded.

(x\_train, y\_train), (x\_test, y\_test) = tf.keras.datasets.mnist.load\_data()

# OUTPUT :



## Step 3: Data Preprocessing

Ensure the data is clean and ready for modeling. Since the Iris dataset is clean, minimal preprocessing is needed.

# Cast the records into float values x\_train = x\_train.astype('float32') x\_test = x\_test.astype('float32')

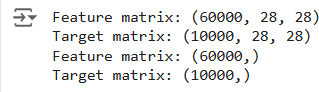
# normalize image pixel values by dividing # by 255

gray\_scale = 255 x\_train /= gray\_scale x\_test /= gray\_scale

print("Feature matrix:", x\_train.shape) print("Target matrix:", x\_test.shape)

print("Feature matrix:", y\_train.shape) print("Target matrix:", y\_test.shape)

# OUTPUT :



**Step 4 : Train a Model**

**model = Sequential([**

**# reshape 28 row \* 28 column data to 28\*28 rows Flatten(input\_shape=(28, 28)),**

**# dense layer 1**

**Dense(256, activation='sigmoid'),**

**# dense layer 2**

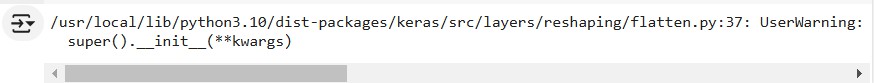
**Dense(128, activation='sigmoid'),**

**# output layer**

**Dense(10, activation='sigmoid'),**

**])**

# OUTPUT:



## Step 5 : Make Predictions

Use the model to make predictions based on the independent variable.

model.compile(optimizer='adam',

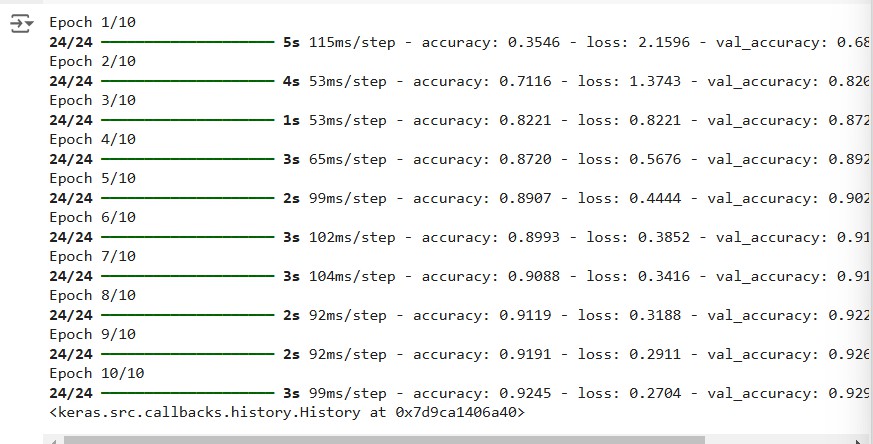
loss='sparse\_categorical\_crossentropy',

metrics=['accuracy']) model.fit(x\_train, y\_train, epochs=10,

batch\_size=2000,

validation\_split=0.2)

# OUTPUT:



## Step 6 : Evaluate the Model

Evaluate the model performance.

results = model.evaluate(x\_test, y\_test, verbose = 0) print('test loss, test acc:', results)

fig, ax = plt.subplots(10, 10) k = 0

for i in range(10):

for j in range(10): ax[i][j].imshow(x\_train[k].reshape(28, 28),

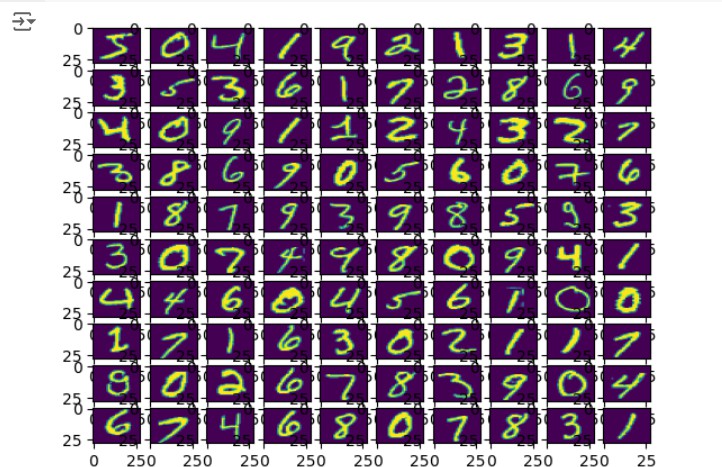
aspect='auto')

k += 1

plt.show()

# OUTPUT :





**RESULT:**

This step-by-step process will help us to implement MultiLayer Perceptron with Backpropagation models using the Keras dataset and analyze their performance.