# EXPT NO : 4 A python program to implement Single Layer DATE:13.9.24 Perceptron

**AIM:**

To write a python program to implement Single layer perceptron.

# PROCEDURE:

Implementing Single layer perceptron method 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.

import numpy as np import pandas as pd

from tensorflow import keras

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)=keras.datasets.mnist.load\_data()

# Step 3: Data Preprocessing

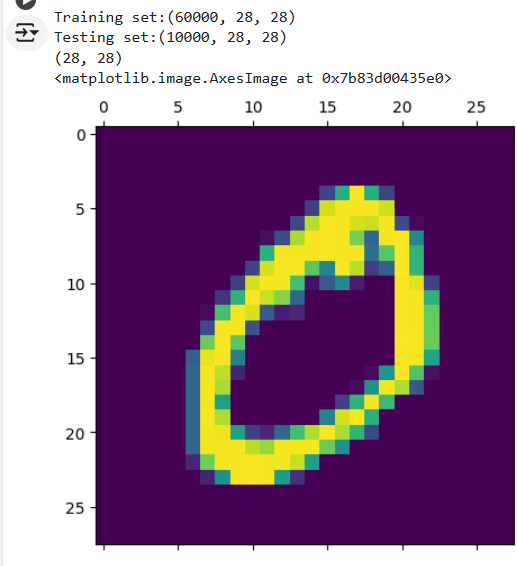
Ensure the data is clean and ready for modeling. Since the Iris dataset is clean, minimal preprocessing is needed.

print(f"Training set:{X\_train.shape}") print(f"Testing set:{X\_test.shape}")

print(X\_train[1].shape)

plt.matshow(X\_train[1])

# OUTPUT :



**Step 4 : Train a Model**

**#Normalizing the dataset x\_train=X\_train/255 x\_test=X\_test/255**

**#Flatting the dataset in order to compute for model building x\_train\_flatten=x\_train.reshape(len(x\_train),28\*28) x\_test\_flatten=x\_test.reshape(len(x\_test),28\*28)**

**x\_train\_flatten.shape**

# Step 5 : Make Predictions

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

model=keras.Sequential([

keras.layers.Dense(10,input\_shape=(784,),

activation='sigmoid')

])

model.compile(

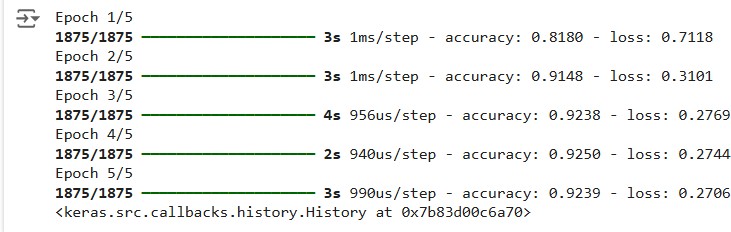
optimizer='adam',

loss='sparse\_categorical\_crossentropy', metrics=['accuracy'])

model.fit(x\_train\_flatten,y\_train,epochs=5

)

# OUTPUT :



**Step 6 : Evaluate the Model**

Evaluate the model performance.

model.evaluate(x\_test\_flatten,y\_test)

# OUTPUT :



**RESULT:**

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