

**EXPT NO : 4            A python program to implement Single Layer**

**DATE:13/09/2024      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**

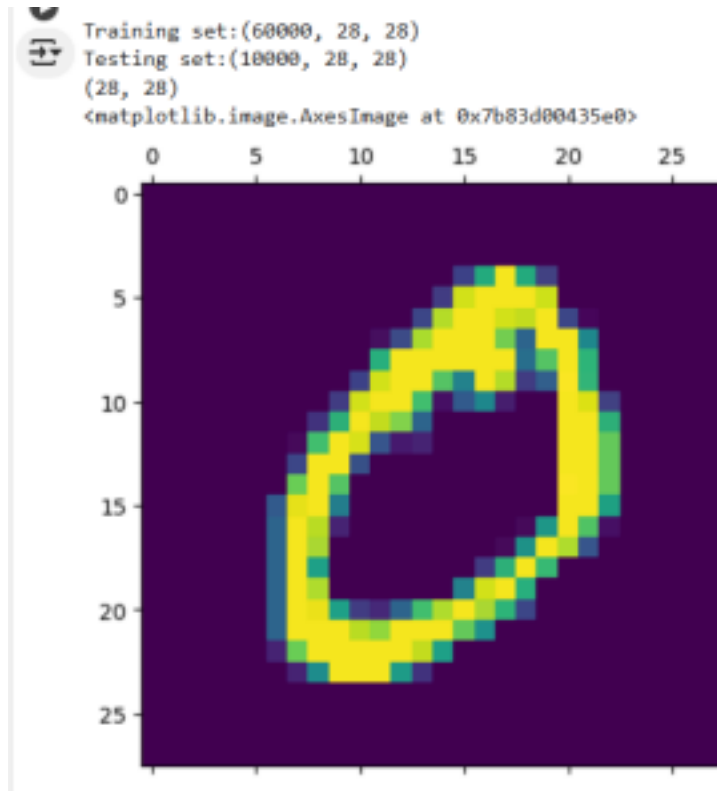
Ensure the data is clean and ready for modeling. Since the Iris dataset is clean, minimal preprocessing is needed.

```
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 :

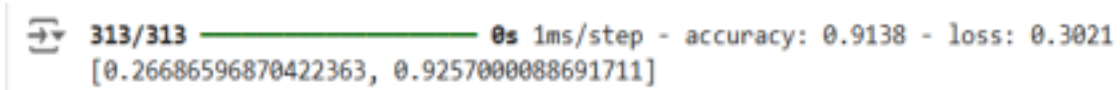
```
Epoch 1/5  
1875/1875 ————— 3s 1ms/step - accuracy: 0.8180 - loss: 0.7118  
Epoch 2/5  
1875/1875 ————— 3s 1ms/step - accuracy: 0.9148 - loss: 0.3101  
Epoch 3/5  
1875/1875 ————— 4s 956us/step - accuracy: 0.9238 - loss: 0.2769  
Epoch 4/5  
1875/1875 ————— 2s 940us/step - accuracy: 0.9250 - loss: 0.2744  
Epoch 5/5  
1875/1875 ————— 3s 990us/step - accuracy: 0.9239 - loss: 0.2706  
<keras.src.callbacks.history.History at 0x7b83d00c6a70>
```

## Step 6 : Evaluate the Model

Evaluate the model performance.

```
model.evaluate(x_test_flatten,y_test)
```

### OUTPUT :

A screenshot of a Jupyter Notebook cell output. It features a progress bar at the top, which is a horizontal line with a green segment on the left and a grey segment on the right. To the left of the bar is a small icon of a document with a checkmark. To the right of the bar is the text '313/313'. Below the progress bar, the text '0s 1ms/step - accuracy: 0.9138 - loss: 0.3021' is displayed. At the bottom, a list containing two numerical values is shown: [0.26686596870422363, 0.9257000088691711].

```
313/313 0s 1ms/step - accuracy: 0.9138 - loss: 0.3021  
[0.26686596870422363, 0.9257000088691711]
```

### RESULT:

This step-by-step process will help us to implement Single Layer Perceptron models

using the Keras dataset and analyze their performance.