S. B. JAIN INSTITUTE OF TECHNOLOGY,

MANAGEMENT & RESEARCH, NAGPUR.

Practical No. 03

Aim: Apply and evaluate the perceptron model for any particular domain.

Name of Student : **Shrutika Pradeep Bagdi**

Roll No : **CS22130**

Semester/Year : VIIth Sem / IVth Year

Academic Session : 2025-2026 [ODD]

Date of Performance : _____

Date of Submission : _____

AIM: Apply and evaluate the perceptron model for any particular domain

OBJECTIVE/EXPECTED LEARNING OUTCOME:

The objectives and expected learning outcome of this practical are:

- The objective of perceptron learning is to adjust weight along with class identification;
- it raised the hopes and expectations for the field of neural networks;
- The objective is to reduce the error e, which is the difference between the neuron response a, and the target vector t;
- Develop a greater familiarity with a range of techniques and methods through a diverse set of theoretical and applied readings;
- Know where to go to learn more about the techniques in this class and those called for that were not covered in this class.

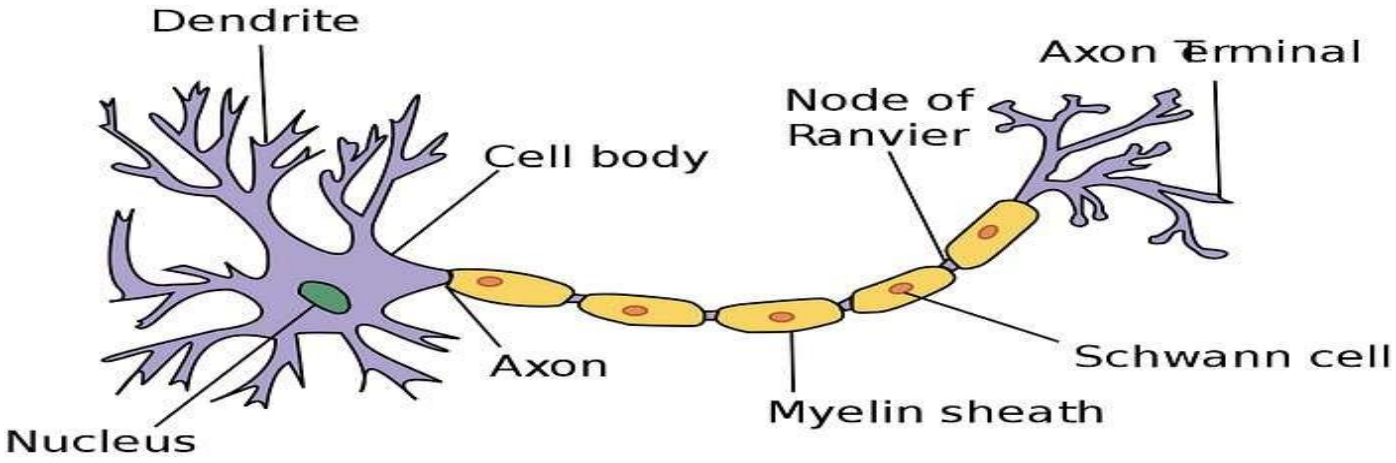
HARDWARE AND SOFTWARE REQUIREMENTS:

Hardware Requirement: Computer

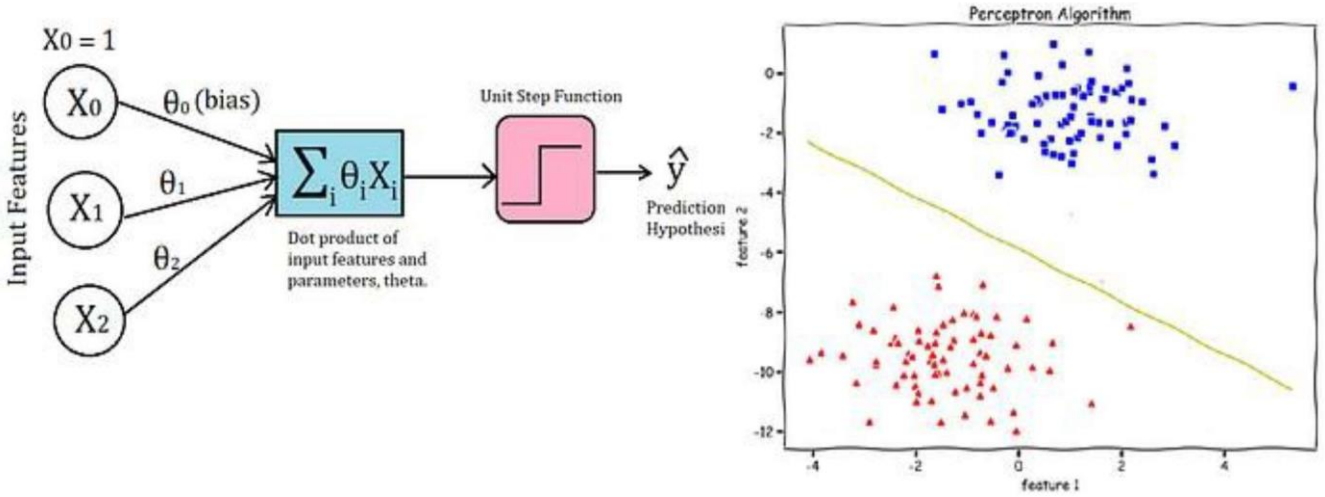
Software Requirement: Google colab, Python 3

THEORY:
Introduction

The Perceptron algorithm was inspired by the basic processing units in the brain, called neurons, and how they process signals. It was invented by **Frank Rosenblatt,** using the McCulloch-Pitts neuron and the findings of Hebb.



A Perceptron Algorithm is not something widely used in practice. We study it mostly for historical reasons and also because it is the most basic and simple single-layered neural network.



we are going to look at the Perceptron Algorithm, which is the most **basic single-layered neural network** used for **binary classification**. First, we will look at the **Unit Step Function** and see how the Perceptron Algorithm classifies and then have a look at the **perceptron update rule**.

Finally, we will plot the **decision boundary** for our data. We will use the data with only two features, and there will be two classes since Perceptron is a binary classifier. We will implement all the code using **Python NumPy**, and visualize/plot using **Matplotlib**.

Perceptron Algorithm

The Perceptron is inspired by the information processing of a single neural cell called a neuron.

A neuron accepts input signals via its dendrites, which pass the electrical signal down to the cell body.

In a similar way, the Perceptron receives input signals from examples of training data that we weight and combined in a linear equation called the activation.

activation = sum(weight * x_i) + bias

The activation is then transformed into an output value or prediction using a transfer function, such as the step transfer function.

prediction = 1.0 if activation >= 0.0 else 0.0

In this way, the Perceptron is a classification algorithm for problems with two classes (0 and 1) where a linear equation (like or hyperplane) can be used to separate the two classes.

It is closely related to linear regression and logistic regression that make predictions in a similar way (e.g. a weighted sum of inputs).

The weights of the Perceptron algorithm must be estimated from your training data using stochastic gradient descent.

CODE:

```
import numpy as np
import tensorflow as tf
from tensorflow import keras
import matplotlib.pyplot as plt
%matplotlib inline
(x_train, y_train), (x_test, y_test) = keras.datasets.mnist.load_data()
x_train
x_test
len(x_test)
len(x_train)
x_train[0].shape
plt.matshow(x_train[0])
x_train = x_train / 255
x_test = x_test / 255
x_train_flatten = x_train.reshape(len(x_train), 28*28)
x_test_flatten = x_test.reshape(len(x_test), 28*28)
model = keras.Sequential([
    keras.layers.Dense(10, input_shape=(784,), activation='sigmoid')
])
model.compile(optimizer='adamax',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])
model.fit(x_train_flatten, y_train, epochs=25)
model.evaluate(x_test_flatten, y_test)
```

Extra

```
import numpy as np
import tensorflow as tf
from tensorflow import keras
import matplotlib.pyplot as plt
%matplotlib inline
(x_train, y_train), (x_test, y_test) = keras.datasets.fashion_mnist.load_data()
x_train
x_test
len(x_test)
len(x_train)
x_train[0].shape
plt.matshow(x_train[0])
x_train = x_train / 255
x_test = x_test / 255
x_train_flatten = x_train.reshape(len(x_train), 28*28)
```

```
x_test_flatten = x_test.reshape(len(x_test), 28*28)
model = keras.Sequential([
   keras.layers.Dense(10, input_shape=(784,), activation='sigmoid')
])
model.compile(optimizer='adamax',
          loss='sparse_categorical_crossentropy',
          metrics=['accuracy'])
model.fit(x_train_flatten, y_train, epochs=25)
model.evaluate(x_test_flatten, y_test)
```

OUTPUT (ScreenShot):

Aim: Apply and evaluate the percertron model for any particular domain.

```
[ ]  #Name: Shrutika Bagdi_ CS22130
     import numpy as np
     import tensorflow as tf
     from tensorflow import keras
     import matplotlib.pyplot as plt
     %matplotlib inline
```

```
[ ]  #Name: Shrutika Bagdi_ CS22130
     (x_train, y_train), (x_test, y_test) = keras.datasets.mnist.load_data()
```

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/mnist.npz
11490434/11490434 ━━━━━━━━━━━━━━━━━━━━ 0s 0us/step
```

```
#Name: Shrutika Bagdi_ CS22130
x_train
```

```
array([[[0, 0, 0, ..., 0, 0, 0],
        [0, 0, 0, ..., 0, 0, 0],
        [0, 0, 0, ..., 0, 0, 0],
        ...,
        [0, 0, 0, ..., 0, 0, 0],
        [0, 0, 0, ..., 0, 0, 0],
        [0, 0, 0, ..., 0, 0, 0]],

       [[0, 0, 0, ..., 0, 0, 0],
        [0, 0, 0, ..., 0, 0, 0],
```

```
[ ]  len(x_test)
```

```
10000
```

```
[ ]  len(x_train)
```

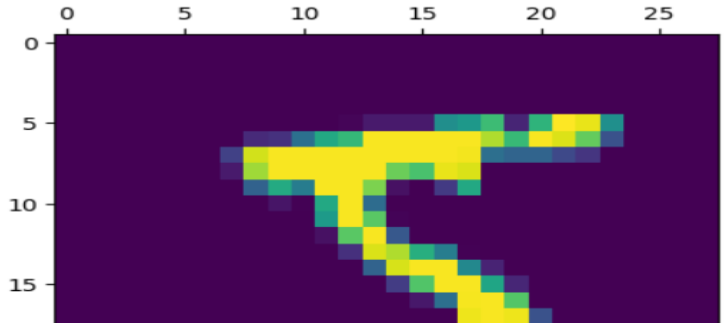
```
60000
```

```
[ ]  x_train[0].shape
```

```
(28, 28)
```

```
plt.matshow(x_train[0])
```

```
<matplotlib.image.AxesImage at 0x7a6d3ae4a710>
```



```
[ ]  #Name: Shrutika Bagdi_ CS22130
     #Normalization the dataset
     x_train = x_train / 255
     x_test = x_test / 255
```

```
[ ]  #Name: Shrutika Bagdi_ CS22130
     # Flattening 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)
```

```
     #Name: Shrutika Bagdi_ CS22130
     model = keras.Sequential([
         keras.layers.Dense(10, input_shape=(784,), activation='sigmoid')
     ])
```

```
/usr/local/lib/python3.11/dist-packages/keras/src/layers/core/dense.py:87: UserWarning: Do not pass
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
```

```
[ ]  #Name: Shrutika Bagdi_ CS22130
     # model.compile(optimizer='adamax',loss='sparse_categorical_crossentropy',metrics=['accuracy'])
     model.compile(optimizer='adamax',
                   loss='sparse_categorical_crossentropy',
                   metrics=['accuracy'])
```

```
     #Name: Shrutika Bagdi_ CS22130
     model.fit(x_train_flatten, y_train, epochs=25)
```

```
Epoch 1/25
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step - accuracy: 0.7688 - loss: 0.9463
Epoch 2/25
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 6s 2ms/step - accuracy: 0.8947 - loss: 0.3926
Epoch 3/25
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step - accuracy: 0.9069 - loss: 0.3441
Epoch 4/25
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 5s 2ms/step - accuracy: 0.9108 - loss: 0.3208
Epoch 5/25
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 6s 2ms/step - accuracy: 0.9143 - loss: 0.3092
Epoch 6/25
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step - accuracy: 0.9164 - loss: 0.3017
Epoch 7/25
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 5s 2ms/step - accuracy: 0.9172 - loss: 0.2947
Epoch 8/25
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 5s 2ms/step - accuracy: 0.9218 - loss: 0.2824
Epoch 9/25
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 3s 2ms/step - accuracy: 0.9196 - loss: 0.2856
Epoch 10/25
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 6s 2ms/step - accuracy: 0.9214 - loss: 0.2808
Epoch 11/25
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step - accuracy: 0.9215 - loss: 0.2782
Epoch 12/25
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 3s 2ms/step - accuracy: 0.9212 - loss: 0.2787
Epoch 13/25
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 6s 2ms/step - accuracy: 0.9229 - loss: 0.2797
Epoch 14/25
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step - accuracy: 0.9249 - loss: 0.2699
Epoch 15/25
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 5s 2ms/step - accuracy: 0.9256 - loss: 0.2700
Epoch 18/25
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 3s 2ms/step - accuracy: 0.9260 - loss: 0.2684
Epoch 19/25
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 5s 2ms/step - accuracy: 0.9271 - loss: 0.2672
Epoch 20/25
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 3s 2ms/step - accuracy: 0.9282 - loss: 0.2667
Epoch 21/25
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 3s 2ms/step - accuracy: 0.9276 - loss: 0.2618
Epoch 22/25
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 5s 2ms/step - accuracy: 0.9278 - loss: 0.2609
Epoch 23/25
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 3s 2ms/step - accuracy: 0.9262 - loss: 0.2653
Epoch 24/25
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 3s 2ms/step - accuracy: 0.9280 - loss: 0.2638
Epoch 25/25
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 5s 2ms/step - accuracy: 0.9264 - loss: 0.2642
<keras.src.callbacks.history.History at 0x7a6d3a46a490>
```

```
[ ]  #Name: Shrutika Bagdi_ CS22130
     model.evaluate(x_test_flatten, y_test)
```

```
313/313 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - accuracy: 0.9158 - loss: 0.3003
[0.26502102613449097, 0.9269999861717224]
```

Extra:

```
[ ] #Name: Shrutika Bagdi_ CS22130
    import numpy as np
    import tensorflow as tf
    from tensorflow import keras
    import matplotlib.pyplot as plt
    %matplotlib inline
```

```
[ ] #Name: Shrutika Bagdi_ CS22130
    (x_train, y_train), (x_test, y_test) = keras.datasets.fashion_mnist.load_data()
```

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/train-labels-idx1-ubyte.gz
29515/29515 ━━━━━━━━━━━━━━━━━━━━ 0s 0us/step
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/train-images-idx3-ubyte.gz
26421880/26421880 ━━━━━━━━━━━━━━━━━━━━ 2s 0us/step
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/t10k-labels-idx1-ubyte.gz
5148/5148 ━━━━━━━━━━━━━━━━━━━━ 0s 0us/step
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/t10k-images-idx3-ubyte.gz
4422102/4422102 ━━━━━━━━━━━━━━━━━━━━ 1s 0us/step
```

```
#Name: Shrutika Bagdi_ CS22130
x_train
```

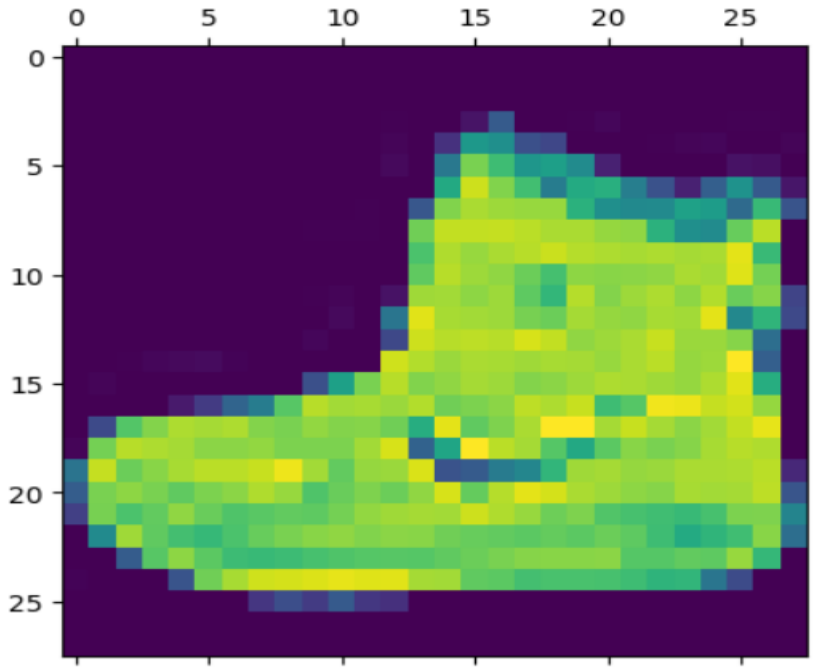
```
array([[[0, 0, 0, ..., 0, 0, 0],
        [0, 0, 0, ..., 0, 0, 0],
        [0, 0, 0, ..., 0, 0, 0],
        ...,
        [0, 0, 0, ..., 0, 0, 0],
        [0, 0, 0, ..., 0, 0, 0],
        [0, 0, 0, ..., 0, 0, 0]],
```

```
[ ] x_train[0].shape
```

```
(28, 28)
```

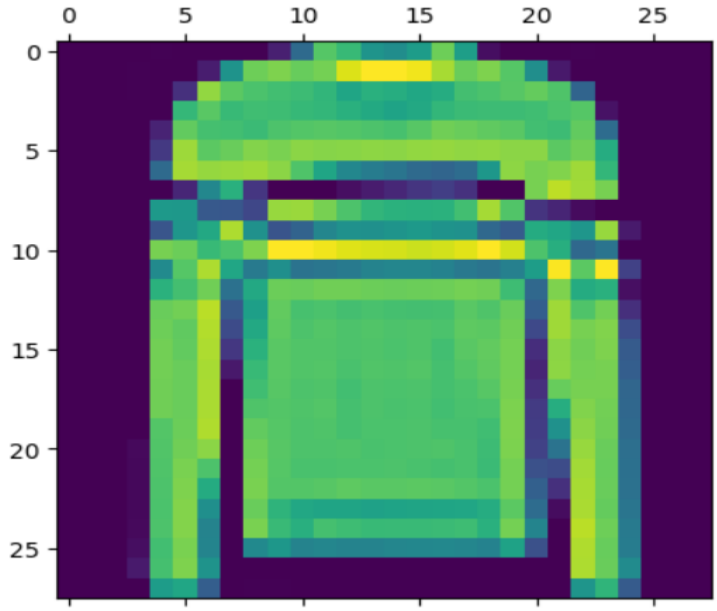
```
plt.matshow(x_train[0])
```

```
<matplotlib.image.AxesImage at 0x7c5f5af32850>
```



```
[ ] plt.matshow(x_train[5])
```

```
<matplotlib.image.AxesImage at 0x7c5f5af8a850>
```




```
x_train = x_train / 255
x_test = x_test / 255
```

```
[ ] x_train_flatten = x_train.reshape(len(x_train), 28*28)
    x_test_flatten = x_test.reshape(len(x_test), 28*28)
```

```
[ ] #Name: Shrutika Bagdi_ CS22130
    model = keras.Sequential([
        keras.layers.Dense(10, input_shape=(784,), activation='sigmoid')
    ])
```

```
/usr/local/lib/python3.11/dist-packages/keras/src/layers/core/dense.py:87: UserWarning: Do not pass an `inp
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
```

```
[ ] #Name: Shrutika Bagdi_ CS22130
    # model.compile(optimizer='adamax',loss='sparse_categorical_crossentropy',metrics=['accuracy'])
    model.compile(optimizer='adamax',
                  loss='sparse_categorical_crossentropy',
                  metrics=['accuracy'])
```

```
[ ] #Name: Shrutika Bagdi_ CS22130
    model.fit(x_train_flatten, y_train, epochs=25)
```

```
Epoch 1/25
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step - accuracy: 0.6819 - loss: 0.9557
Epoch 2/25
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 5s 2ms/step - accuracy: 0.8207 - loss: 0.5400
Epoch 3/25
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step - accuracy: 0.8313 - loss: 0.4972
Epoch 4/25
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 5s 2ms/step - accuracy: 0.8412 - loss: 0.4701
Epoch 5/25
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step - accuracy: 0.8462 - loss: 0.4550
```

```
Epoch 6/25
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 5s 2ms/step - accuracy: 0.8473 - loss: 0.4463
Epoch 7/25
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 5s 2ms/step - accuracy: 0.8521 - loss: 0.4351
Epoch 8/25
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 5s 2ms/step - accuracy: 0.8539 - loss: 0.4295
Epoch 9/25
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step - accuracy: 0.8549 - loss: 0.4234
Epoch 10/25
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 5s 2ms/step - accuracy: 0.8565 - loss: 0.4220
Epoch 11/25
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 5s 2ms/step - accuracy: 0.8568 - loss: 0.4173
Epoch 12/25
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 3s 2ms/step - accuracy: 0.8593 - loss: 0.4109
Epoch 13/25
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 5s 2ms/step - accuracy: 0.8591 - loss: 0.4133
Epoch 14/25
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 6s 2ms/step - accuracy: 0.8613 - loss: 0.4053
Epoch 15/25
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 5s 2ms/step - accuracy: 0.8620 - loss: 0.4064
Epoch 16/25
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step - accuracy: 0.8629 - loss: 0.4024
Epoch 17/25
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 5s 2ms/step - accuracy: 0.8636 - loss: 0.4002
Epoch 18/25
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 5s 2ms/step - accuracy: 0.8631 - loss: 0.4005
Epoch 19/25
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 6s 2ms/step - accuracy: 0.8615 - loss: 0.4021
Epoch 20/25
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 3s 2ms/step - accuracy: 0.8610 - loss: 0.3958
Epoch 21/25
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 6s 2ms/step - accuracy: 0.8635 - loss: 0.3926
Epoch 22/25
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 5s 2ms/step - accuracy: 0.8658 - loss: 0.3928
```

```
Epoch 23/25
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step - accuracy: 0.8678 - loss: 0.3866
Epoch 24/25
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 6s 2ms/step - accuracy: 0.8622 - loss: 0.3969
Epoch 25/25
1875/1875 ━━━━━━━━━━━━━━━━━━━━ 4s 2ms/step - accuracy: 0.8637 - loss: 0.3967
<keras.src.callbacks.history.History at 0x7c5f559cb590>
```

```
[ ] #Name: Shrutika Bagdi_ CS22130
    model.evaluate(x_test_flatten, y_test)
```

```
313/313 ━━━━━━━━━━━━━━━━━━━━ 1s 2ms/step - accuracy: 0.8496 - loss: 0.4323
[0.44083550572395325, 0.8457000255584717]
```

CONCLUSION:

DISCUSSION AND VIVA VOCE:

- How Human neurons differ from Artificial Neuron?

- What are the activities done by a single perceptron?
- Is there any drawback of single layer perceptrons?
- How does a single perceptron work?
- What is the role of the Activation function in single Layer perceptron?

REFERENCE:

- https://www.google.com/search?q=perceptron+model+domain&sca_esv=556652858&ei=ObvZZK3BIrmc4EP6dST2AI&ved=0ahUKEwjtwcyItduAAxU5zjgGHWnqBCsQ4dUDCA8&uact=5&oq=perceptron+model+domain&gs_lp=Egxnd3Mtd2l6LXNlcnAiF3BlcmNlcHRyb24gbW9kZWwgZG9tYWluMggQIRigARjDBEjdF1C8AliGFXABeAGQAQCYAcEBoAG0AqoBAzAuMrgBA8gBAPgBAclCChAAGEcY1gQYsAPiAwQYACBBiAYBkAYI&sclient=gws-wiz-serp
- https://www.geeksforgeeks.org/single-layer-perceptron-in-tensorflow/
- https://www.javatpoint.com/single-layer-perceptron-in-tensorflow
- https://blog.knoldus.com/complete-guide-to-single-layer-perceptron-with-implementation/
- https://www.educba.com/single-layer-perceptron/
- https://www.bogotobogo.com/python/scikit-learn/Perceptron_Model_with_Iris_DataSet.php