Loading the dataset

```
Dataset Preview:
   label  pixel_0  pixel_1  pixel_2  pixel_3  pixel_4  pixel_5  pixel_6  \
0      0        0        0        0        0        0        0        0
1      1        0        0        0        0        0        0        0
2      1        0        0        0        0        0        0        0
3      1        0        0        0        0        0        0        0
4      1        0        0        0        0        0        0        0

   pixel_7  pixel_8  ...  pixel_774  pixel_775  pixel_776  pixel_777  \
0        0        0  ...          0          0          0          0
1        0        0  ...          0          0          0          0
2        0        0  ...          0          0          0          0
3        0        0  ...          0          0          0          0
4        0        0  ...          0          0          0          0

   pixel_778  pixel_779  pixel_780  pixel_781  pixel_782  pixel_783
0          0          0          0          0          0          0
1          0          0          0          0          0          0
2          0          0          0          0          0          0
3          0          0          0          0          0          0
4          0          0          0          0          0          0

[5 rows x 785 columns]

Dataset Information:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 12665 entries, 0 to 12664
Columns: 785 entries, label to pixel_783
dtypes: int64(785)
memory usage: 75.9 MB
None
```

Task-1: ImplementationofMCPNeurons:

MCP Neuron for AND Operation

```
# Print the result
print(f"Output of AND gate for inputs {X1} and {X2} with threshold {T}: {result}")
```

```
Output of AND gate for inputs [0, 0, 1, 1] and [0, 1, 0, 1] with threshold 2: [0, 0, 0, 1]
```

MCP Neuron for OR Operation

```
# Print the result
print(f"Output of OR gate for inputs {X1} and {X2} with threshold {T}: {result_or}")
```

```
Output of OR gate for inputs [0, 0, 1, 1] and [0, 1, 0, 1] with threshold 1: [0, 1, 1, 1]
```

Question- 2: Think if you can develop a logic to solve for XOR function using MCP Neuron. {Can you devise a if else rules.}

IF-ELSE Logic for XOR

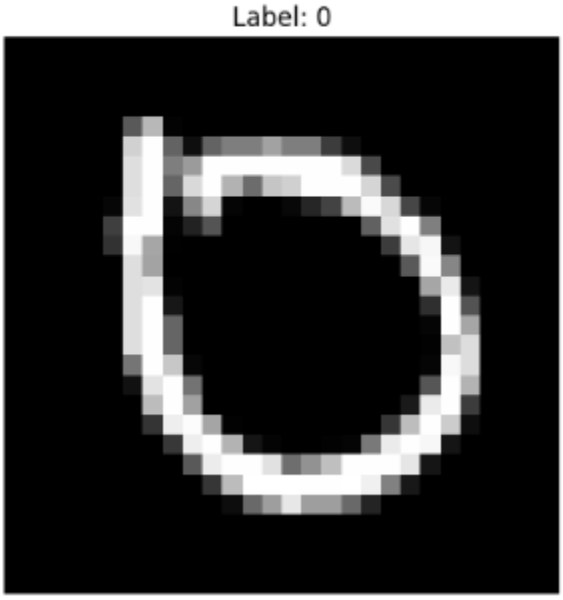
```
# Print result
print(f"Output of XOR gate for inputs {X1} and {X2}: {result_xor}")
```

```
Output of XOR gate for inputs [0, 0, 1, 1] and [0, 1, 0, 1]: [0, 1, 1, 0]
```

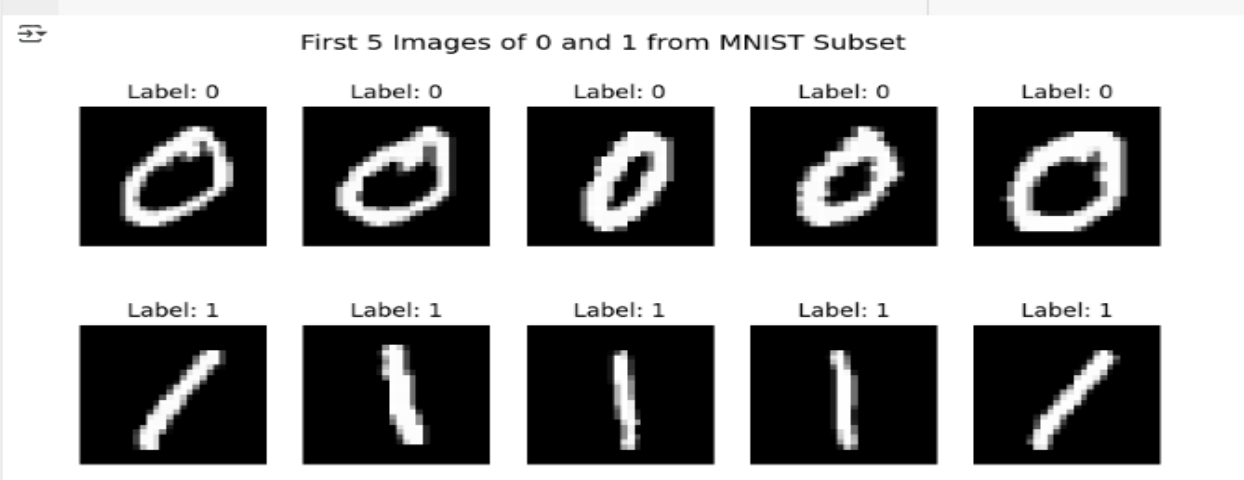
Task 2: Perceptron Algorithm for 0 vs 1 Classification.

Load the Dataset:

```
Feature matrix shape: (12665, 784)
Label vector shape: (12665,)
Training set shape: (10132, 784)
Testing set shape: (2533, 784)
```



Visualize the Dataset:



3. Initialize Weights and Bias:

```
Epoch 10/100 completed
Epoch 20/100 completed
Epoch 30/100 completed
Epoch 40/100 completed
Epoch 50/100 completed
Epoch 60/100 completed
Epoch 70/100 completed
Epoch 80/100 completed
Epoch 90/100 completed
Epoch 100/100 completed
Training complete!
```

4. ImplementaDecisionFunctionorActivationFunction:

```
Input Features (X_sample):
 [[0.37454012 0.95071431 0.73199394]
 [0.59865848 0.15601864 0.15599452]
 [0.05808361 0.86617615 0.60111501]
 [0.70807258 0.02058449 0.96990985]
 [0.83244264 0.21233911 0.18182497]]

Weights:
 [0.18340451 0.30424224 0.52475643]

Bias:
 0.43194501864211576

Predicted Labels:
 [1 1 1 1 1]
```

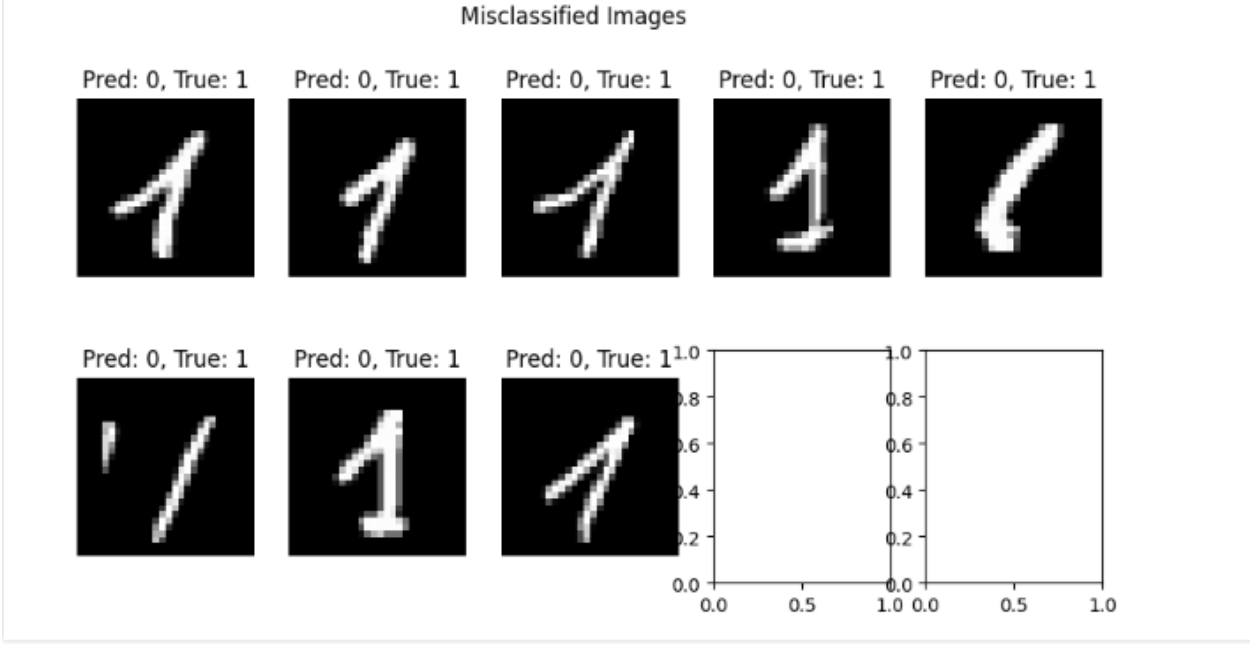
5. ImplementthePerceptronLearningAlgorithm:

```
Feature matrix shape: (12665, 784)
Label vector shape: (12665,)
Epoch 1/10 - Accuracy: 0.9962
Epoch 2/10 - Accuracy: 0.9982
Epoch 3/10 - Accuracy: 0.9984
Epoch 4/10 - Accuracy: 0.9993
Epoch 5/10 - Accuracy: 0.9991
Epoch 6/10 - Accuracy: 0.9995
Epoch 7/10 - Accuracy: 1.0000
Epoch 8/10 - Accuracy: 1.0000
Epoch 9/10 - Accuracy: 1.0000
Epoch 10/10 - Accuracy: 1.0000

Final Weights: [0. 0. 0. 0. 0.]
Final Bias: 1.500000000000002
```

7. Visualizing the Misclassified Image:

```
Final Accuracy: 0.9994
```



Task 3: Perceptron Algorithm for 3 vs 5 Classification.

Step 1: Load and Preprocess the Dataset

```
Feature matrix shape: (12665, 784)
Label vector shape: (12665,)
```

Step 2: Initialize Weights and Bias

```
weights = np.zeros(X.shape[1])  # 784 weights (one per pixel)
bias = 0
learning_rate = 0.1
epochs = 100
```

Step 3: Define the Decision Function

```
def decision_function(X, weights, bias):
    """
    Computes the predicted labels for input data using perceptron activation (step function).
    """
    predictions = np.dot(X, weights) + bias
    return np.where(predictions >= 0, 1, 0)  # Apply step function
```

Step 4: Train the Perceptron

```
    for epoch in range(epochs):
        for i in range(len(y)):
            # Compute prediction
            output = np.dot(X[i], weights) + bias
            prediction = 1 if output >= 0 else 0

            # Update weights and bias if prediction is incorrect
            if prediction != y[i]:
                update = learning_rate * (y[i] - prediction)
                weights += update * X[i]
                bias += update

    return weights, bias
```

Step 5: Train the Model

```
# Train perceptron
weights, bias = train_perceptron(X, y, weights, bias, learning_rate, epochs)
```

Step 6: Evaluate Model Performance

```
# Calculate accuracy
accuracy = np.mean(y_pred == y)
print(f"Final Accuracy: {accuracy:.4f}")
```

```
Final Accuracy: 1.0000
```

Step 7: Visualize Misclassified Images

```
All images were correctly classified!
```

Final output

```
Final Accuracy: 0.9613
```