Name: Nupur Lalit Vartak
Class: BE/CSE(DS)
Roll No.: 62
Subject: Deep Learning Lab
Exp No.: 01

Aim: Implement Multilayer Perceptron algorithm to simulate XOR gate

Code:

```
#Import Libraries
import numpy as np
from keras.models import Sequential
from keras.layers import Dense
# XOR input data
X = np.array([[0, 0], [0, 1], [1, 0], [1, 1]])
# Corresponding XOR output data
Y = np.array([[0], [1], [1], [0]])
# Create a sequential model
model = Sequential()
# Add a hidden layer with 8 neurons and 'relu' activation function
model.add(Dense(8, input_dim=2, activation='relu'))
# Add the output layer with 1 neuron and 'sigmoid' activation function
model.add(Dense(1, activation='sigmoid'))
# Compile the model using binary cross-entropy loss and Adam optimizer
model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
# Train the model for 1000 epochs
model.fit(X, Y, epochs=1000, verbose=0)
# Evaluate the model
loss, accuracy = model.evaluate(X, Y)
print(f"Loss: {loss:.4f}, Accuracy: {accuracy:.4f}")
# Make predictions
predictions = model.predict(X)
rounded_predictions = np.round(predictions)
print("Predictions:")
print(rounded_predictions)
```

Output:

```
1/1 [==============================] - 0s 208ms/step - loss: 0.1793 - accuracy: 1.0000
Loss: 0.1793, Accuracy: 1.0000
1/1 [==============================] - 0s 125ms/step
Predictions:
[[0.]
 [1.]
 [1.]
 [0.]]
```

Conclusion:

In conclusion, implementing the Multilayer Perceptron algorithm to simulate the XOR gate demonstrates its capability to learn and approximate nonlinear functions through layered neural networks. This exercise underscores the versatility and power of neural networks in solving complex computational tasks, showcasing their effectiveness in handling problems that traditional linear models struggle with.