**AML Algorithm #12 : neural network implementation to solve XOR**

import tensorflow as tf

import numpy as np

# Define XOR input and output

input\_data = np.array([[0, 0], [0, 1], [1, 0], [1, 1]], dtype=np.float32)

output\_data = np.array([[0], [1], [1], [0]], dtype=np.float32)

# Define the neural network architecture

model = tf.keras.Sequential([

tf.keras.layers.Dense(2, input\_dim=2, activation='sigmoid'),

tf.keras.layers.Dense(1, activation='sigmoid')

])

# Compile the model

model.compile(optimizer='adam', loss='mean\_squared\_error', metrics=['accuracy'])

# Train the model

model.fit(input\_data, output\_data, epochs=10000, verbose=0)

# Evaluate the model

loss, accuracy = model.evaluate(input\_data, output\_data)

print(f"Model Accuracy: {accuracy}")

# Make predictions

predictions = model.predict(input\_data)

rounded\_predictions = np.round(predictions)

print("Predictions:")

print(rounded\_predictions)

**OUTPUT :**

1/1 [==============================] - 0s 207ms/step - loss: 0.1258 - accuracy: 0.5000

Model Accuracy: 0.5

1/1 [==============================] - 0s 121ms/step

Predictions:

[[0.]

[1.]

[0.]

[1.]]