**Ex.no:1**

**PROGRAM:**

import tensorflow as tf

scalar = tf.constant(7)

scalar

scalar.ndim

vector = tf.constant([10, 10])

vector.ndim

matrix = tf.constant([[1, 2], [3, 4]])

print(matrix)

print('the number of dimensions of a matrix is :' + str(matrix.ndim))

matrix = tf.constant([[1, 2], [3, 4]])

matrix1 = tf.constant([[2, 4], [6, 8]])

print(matrix + matrix1)

**OUTPUT:**

tf.Tensor(

[[1 2]

[3 4]], shape=(2, 2), dtype=int32)

the number of dimensions of a matrix is :2

tf.Tensor(

[[ 3 6]

[ 9 12]], shape=(2, 2), dtype=int32)

**EX.no:2**

**PROGRAM:**  
# Importing necessary libraries

import numpy as np

import tensorflow as tf

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import Dense

# Generate some random data for demonstration

np.random.seed(0)

X = np.random.rand(100, 1) # Input features

y = 2 \* X.squeeze() + np.random.randn(100) # Target variable with some noise

# Splitting data into training and testing sets

split = int(0.8 \* len(X))

X\_train, X\_test = X[:split], X[split:]

y\_train, y\_test = y[:split], y[split:]

# Creating a Sequential model

model = Sequential()

# Adding a dense layer with one neuron (for regression)

model.add(Dense(1, input\_dim=1))

# Compiling the model

model.compile(optimizer='adam', loss='mean\_squared\_error')

# Training the model

model.fit(X\_train, y\_train, epochs=100, batch\_size=10, verbose=0)

# Evaluating the model

loss = model.evaluate(X\_test, y\_test, verbose=0)

print("Test Loss:", loss)

# Making predictions

predictions = model.predict(X\_test)

# Printing some predictions

for i in range(5):

print("Predicted:", predictions[i][0], "Actual:", y\_test[i])

**OUTPUT:**

Test Loss: 0.8871573209762573

1/1 [==============================] - ETA: 0s1/1 [==============================] - 0s 78ms/step

Predicted: 0.8866252 Actual: 0.13793390809564715

Predicted: 1.0223583 Actual: 2.7580580428463257

Predicted: 0.5287732 Actual: 1.0777157996233295

Predicted: 1.4145714 Actual: 1.4724954801252306

Predicted: 1.2371218 Actual: -0.09223261041701769

**EX.no:3**

**PROGRAM:**

import tensorflow as tf

from tensorflow import keras

from tensorflow.keras.layers import Dense

from tensorflow.keras.optimizers import SGD

import numpy as np

# Generate some sample data for a logical OR operation

X = np.array([[0, 0], [0, 1], [1, 0], [1, 1]]) # Input features

y = np.array([0, 1, 1, 1]) # Output labels (OR gate)

# Define a simple perceptron model

model = keras.Sequential([

Dense(units=1, input\_dim=2, activation='sigmoid') # 2 input features, 1 output unit, sigmoid activation

])

# Compile the model

model.compile(optimizer=SGD(learning\_rate=0.1), loss='mean\_squared\_error', metrics=['accuracy'])

# Train the model

model.fit(X, y, epochs=1000, verbose=0) # You can adjust the number of epochs

# Evaluate the model

loss, accuracy = model.evaluate(X, y)

print("Loss:", loss)

print("Accuracy:", accuracy)

# Make predictions

predictions = model.predict(X)

print("Predictions:")

print(predictions)

**OUTPUT:**

1/1 [==============================] - 0s 86ms/step - loss: 0.0488 - accuracy: 1.0000

Loss: 0.04884687811136246

Accuracy: 1.0

1/1 [==============================] - 0s 43ms/step

Predictions:

[[0.34876198]

[0.80799234]

[0.8101397 ]

[0.9710391 ]]