Assignment No. 11

import numpy as np

# Sample dataset: words and their labels (0 = short, 1 = long)

words = ['cat', 'dog', 'elephant', 'ant', 'giraffe', 'zebra']

labels = [0, 0, 1, 0, 1, 1] # Short words are labeled as 0, long words as 1

# Feature extraction function

def extract\_features(words):

features = []

for word in words:

length = len(word)

vowels = sum(1 for char in word if char in 'aeiou')

features.append([length, vowels]) # Length and number of vowels

return np.array(features)

# Prepare feature matrix and label vector

X = extract\_features(words)

y = np.array(labels).reshape(-1, 1)

# Neural Network Parameters

input\_size = X.shape[1] # Number of features

hidden\_size = 4 # Number of neurons in hidden layer

output\_size = 1 # Output layer size (binary classification)

learning\_rate = 0.01

epochs = 10000

# Initialize weights and biases

np.random.seed(42)

weights\_input\_hidden = np.random.rand(input\_size, hidden\_size)

bias\_hidden = np.random.rand(1, hidden\_size)

weights\_hidden\_output = np.random.rand(hidden\_size, output\_size)

bias\_output = np.random.rand(1, output\_size)

# Activation function (Sigmoid)

def sigmoid(x):

return 1 / (1 + np.exp(-x))

# Derivative of Sigmoid

def sigmoid\_derivative(x):

return x \* (1 - x)

# Training the neural network

for epoch in range(epochs):

# Forward Propagation

hidden\_layer\_activation = np.dot(X, weights\_input\_hidden) + bias\_hidden

hidden\_layer\_output = sigmoid(hidden\_layer\_activation)

output\_layer\_activation = np.dot(hidden\_layer\_output, weights\_hidden\_output) + bias\_output

predicted\_output = sigmoid(output\_layer\_activation)

# Calculate error

error = y - predicted\_output

# Backpropagation

d\_predicted\_output = error \* sigmoid\_derivative(predicted\_output)

error\_hidden\_layer = d\_predicted\_output.dot(weights\_hidden\_output.T)

d\_hidden\_layer = error\_hidden\_layer \* sigmoid\_derivative(hidden\_layer\_output)

# Update weights and biases

weights\_hidden\_output += hidden\_layer\_output.T.dot(d\_predicted\_output) \* learning\_rate

bias\_output += np.sum(d\_predicted\_output, axis=0, keepdims=True) \* learning\_rate

weights\_input\_hidden += X.T.dot(d\_hidden\_layer) \* learning\_rate

bias\_hidden += np.sum(d\_hidden\_layer, axis=0, keepdims=True) \* learning\_rate

# Testing the neural network with new data

test\_words = ['bat', 'hippopotamus', 'antelope']

test\_X = extract\_features(test\_words)

# Forward pass for testing

hidden\_layer\_activation\_test = np.dot(test\_X, weights\_input\_hidden) + bias\_hidden

hidden\_layer\_output\_test = sigmoid(hidden\_layer\_activation\_test)

output\_layer\_activation\_test = np.dot(hidden\_layer\_output\_test, weights\_hidden\_output) + bias\_output

predicted\_test\_output = sigmoid(output\_layer\_activation\_test)

# Classify based on threshold (0.5)

predictions = (predicted\_test\_output > 0.5).astype(int)

# Print results

for word, prediction in zip(test\_words, predictions):

print(f"Word: {word}, Predicted Class: {'Long' if prediction[0] == 1 else 'Short'}")

#OUTPUT

Word: bat, Predicted Class: Short

Word: hippopotamus, Predicted Class: Long

Word: antelope, Predicted Cl