Experiment No. 2: Deep neural network on IMDB Dataset

1. The dataset is the Large Movie Review Dataset, often referred to as the IMDB dataset.

2. The IMDB dataset contains 50,000 highly polar movie reviews (good or bad) for training and the same amount again for testing. The problem is to determine whether a given movie review has a positive or negative sentiment.

#Load the IMDB Dataset with Keras

```
import numpy as np
from tensorflow.keras.datasets import imdb
import matplotlib.pyplot as plt

# load the dataset
(X_train, y_train), (X_test, y_test) = imdb.load_data()
X= np.concatenate((X_train, X_test), axis=0)
y= np.concatenate((y_train, y_test), axis=0)

# summarize size
print("Training data: ")
print(X.shape)
print(y.shape)
```

OUT:

Training data:

(50000,)

(50000,)

#Word Embedding

imdbload_data (nb_words=5000)

#truncate or pad the dataset to a length of 500 for each observation

X_train = sequence.pad_sequences(X_train, maxlen=500)

X_test = sequence.pad_sequences(X_test, maxlen=500)

Embedding(5000, 32, input_length=500)

#Simple Multi-Layer Perceptron Model for the IMDB Dataset

MLP for the IMDB problem

from tensorflow.keras.datasets import imdb

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import Dense

from tensorflow.keras.layers import Flatten

from tensorflow.keras.layers import Embedding

from tensorflow.keras.preprocessing import sequence

load the dataset but only keep the top n words, zero the rest

top_words = 5000

(X_train, y_train),(X_test, y_test) = imdb.load_data (num_words=top_words)

```
#bound reviews at 500 words, truncating longer reviews and zero-padding shorter one

max_words =500
X_train = sequence.pad_sequences (X_train, maxlen=max_words)
X_ test = sequence.pad_sequences (X_test, maxlen=max_words)

# create the model
model = Sequential()
model.add (Embedding (top_words, 32, input length=max_words))
model.add (Flatten ())
model.add (Dense [250, activation='relu'))
model.add (Dense(1, activation='sigmoid'))
model.compile (loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
model.summary()

# Fit the model
nodel.fit(X_train, y_train, validation_data=(X_test, y_test), epochs=2, batch_size=128,
verbose=2)

# Final evaluation of the model
Scores = model.evaluate(X_test, y_test, verbose=0)
print("Accuracy: %.2f%%" % (scores[1]*100))
```

OUT:

Epoch ½

196/196-4s - loss: 0.5579 - accuracy: 0.6664 - val loss: 0.3060 - val_accuracy: 0.8700 - 4s/epoch - 2Oms/step

Epoch 2/2

196/196 - 4s - loss: 0.2108 - accuracy: 0.9165 - val_loss: 0.3006 - val_accuracy: 0.8731 - 4s/epoch - 19ms/step

Accuracy: 87.319%