PAREENITA ATUL SHIRSATH PRN : 221101062 ROLL.NO : 57 B.E.A.I.&.D.S.

DLL EXPERIMENT NO: 1

AIM - To explore python libraries for deep learning e.g. Theano, TensorFlow,etc.

1. Theano Example

```
import tensorflow as tf

# Define variables
x = tf.constant(2.0, dtype=tf.float32)
w = tf.constant(3.0, dtype=tf.float32)
b = tf.constant(4.0, dtype=tf.float32)

# Define model
y = w * x + b

# Run with example values
print("TensorFlow Output:", y.numpy())
```

```
TensorFlow Output: 10.0
```

2. TensorFlow (v2) Example (Using tf.keras API)

```
import tensorflow as tf
import numpy as np

# Generate synthetic data
X = np.array([[1], [2], [3], [4]], dtype=float)
y = np.array([[2], [4], [6], [8]], dtype=float)

# Build a simple linear model
model = tf.keras.Sequential([
    tf.keras.layers.Dense(units=1, input_shape=[1])
])

# Compile model
model.compile(optimizer='sgd', loss='mean_squared_error')

# Train model
model.fit(X, y, epochs=100, verbose=0)

# Predict
print("TensorFlow Prediction for 5:", model.predict(np.array([[5]])))  # Expected near 10
```

```
/usr/local/lib/python3.11/dist-packages/keras/src/layers/core/dense.py:87: UserWarning: Do not pass an `input_shape`/`input_dim` arg
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 78ms/step
TensorFlow Prediction for 5: [[9.768687]]
```

3. Keras Example

```
from keras.models import Sequential
from keras.layers import Dense
import numpy as np

# Input and output data
X = np.array([1, 2, 3, 4], dtype=float)
y = np.array([2, 4, 6, 8], dtype=float)

# Define the model
model = Sequential()
model.add(Dense(1, input_dim=1, activation='linear'))

# Compile and fit
model.compile(optimizer='adam', loss='mse')
model.fit(X, y, epochs=100, verbose=0)

# Predict
print("Keras Prediction for 5:", model.predict(np.array([5])))
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 80ms/step
Keras Prediction for 5: [[2.0212967]]
```

4. PyTorch Example

```
import torch
import torch.nn as nn
import torch.optim as optim

# Input and output data
X = torch.tensor([[1.0], [2.0], [3.0], [4.0]])
y = torch.tensor([[2.0], [4.0], [6.0], [8.0]])

# Define linear model
model = nn.Linear(1, 1)
criterion = nn.MSELoss()
optimizer = optim.SGD(model.parameters(), lr=0.01)

# Train model
for epoch in range(100):
    pred = model(X)
    loss = criterion(pred, y)
    loss.backward()
    optimizer.step()
    optimizer.zero_grad()

# Predict
with torch.no_grad():
    print("PyTorch Prediction for 5:", model(torch.tensor([[5.0]])))
```

```
PyTorch Prediction for 5: tensor([[9.7740]])
```

5. NLTK (Natural Language Toolkit) Tokenization and Stopword Removal

```
import nltk
from nltk.tokenize import word_tokenize
from nltk.corpus import stopwords

# Download resources (only once)
nltk.download('punkt')
nltk.download('stopwords')
nltk.download('punkt_tab')

text = "NLTK is a powerful toolkit for natural language processing with Python."
tokens = word_tokenize(text)

# Remove stopwords
filtered = [word for word in tokens if word.lower() not in stopwords.words('english')]
print("Filtered Tokens:", filtered)
```

```
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
[nltk_data] Downloading package punkt_tab to /root/nltk_data...
[nltk_data]   Unzipping tokenizers/punkt_tab.zip.
Filtered Tokens: ['NLTK', 'powerful', 'toolkit', 'natural', 'language', 'processing', 'Python', '.']
```