This documentation provides a structured guide to implementing[***Sentiment Analysis using Deep Learning***](https://colab.research.google.com/drive/1gdSTM2yCa1QEj3MjQ2kld1DNIgpiKozu?usp=sharing). The model is trained to classify tweets into three categories:

**0 → Bearish**

**1 → Bullish**

**2 → Neutral**

The dataset is sourced from Hugging Face (***Hugging Face - zeroshot/twitter-financial-news-sentiment***), and the training is performed in ***Google Colab (T4 GPU).***

**Step 1: Environment Setup**

**Open Google Colab → Enable GPU support** : Runtime → Change runtime type → Select GPU (T4 recommended)

**Step 2: Load the Dataset**

**Dataset Source** : [Hugging Face - Twitter Financial News Sentiment](https://huggingface.co/datasets/zeroshot/twitter-financial-news-sentiment)

**Libraries Installation**

!pip install tensorflow datasets transformers pandas scikit-learn

**Code to Load Data**

import pandas as pd

splits = {'train': 'sent\_train.csv', 'validation': 'sent\_valid.csv'}

df\_train = pd.read\_csv("hf://datasets/zeroshot/twitter-financial-news-sentiment/" + splits["train"])

df\_valid = pd.read\_csv("hf://datasets/zeroshot/twitter-financial-news-sentiment/" + splits["validation"])

df\_train.head()

**Step 3: Data Preprocessing**

**Convert text labels to numerical format** : Bearish → 0, Bullish → 1 & Neutral → 2

**Tokenize and pad text sequences**

from sklearn.model\_selection import train\_test\_split

from tensorflow.keras.preprocessing.text import Tokenizer

from tensorflow.keras.preprocessing.sequence import pad\_sequences

import numpy as np

label\_map = {'Bearish': 0, 'Bullish': 1, 'Neutral': 2}

df\_train.columns = ['text', 'label']

df\_valid.columns = ['text', 'label']

df\_train['label'] = df\_train['label'].map(label\_map)

df\_valid['label'] = df\_valid['label'].map(label\_map)

tokenizer = Tokenizer(num\_words=5000, oov\_token='<OOV>')

tokenizer.fit\_on\_texts(df\_train['text'])

X\_train = tokenizer.texts\_to\_sequences(df\_train['text'])

X\_valid = tokenizer.texts\_to\_sequences(df\_valid['text'])

X\_train = pad\_sequences(X\_train, maxlen=100, padding='post', truncating='post')

X\_valid = pad\_sequences(X\_valid, maxlen=100, padding='post', truncating='post')

y\_train = np.array(df\_train['label'])

y\_valid = np.array(df\_valid['label'])

**Step 4: Build Deep Learning Model**

**Model Consists** :

* Embedding Layer for text representation
* LSTM Layers for sequential data processing
* Dense Layers for classification

import tensorflow as tf

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import Embedding, LSTM, Dense, Dropout

model = Sequential([

Embedding(input\_dim=5000, output\_dim=64, input\_length=100),

LSTM(128, return\_sequences=True),

Dropout(0.5),

LSTM(64),

Dense(32, activation='relu'),

Dropout(0.5),

Dense(3, activation='softmax')

])

model.compile(loss='sparse\_categorical\_crossentropy', optimizer='adam', metrics=['accuracy'])

model.summary()

**Step 5: Train the Model**

Training the model using 10 epochs with a batch size of 32

history = model.fit(

X\_train, y\_train,

validation\_data=(X\_valid, y\_valid),

epochs=10,

batch\_size=32

)

**Step 6: Save and Export the Model**

Once training is completed, save the model in PKL format for validation

import pickle

with open('tokenizer.pkl', 'wb') as handle:

pickle.dump(tokenizer, handle, protocol=pickle.HIGHEST\_PROTOCOL)

model.save('3681\_sentiment\_model\_Kshitiz\_Sharma.h5')

with open('3681\_sentiment\_model\_Kshitiz\_Sharma.pkl', 'wb') as f:

pickle.dump(model, f)

**Download the Model**

Downloading the 3681\_sentiment\_model\_Kshitiz\_Sharma.pkl file for validation

from google.colab import files

files.download('3681\_sentiment\_model\_Kshitiz\_Sharma.pkl')

**Step 7: Load and Test the Model**

After saving, the model can be reloaded for inference

from tensorflow.keras.models import load\_model

loaded\_model = load\_model('3681\_sentiment\_model\_Kshitiz\_Sharma.h5')

with open('tokenizer.pkl', 'rb') as handle:

loaded\_tokenizer = pickle.load(handle)

def predict\_sentiment(text):

sequence = loaded\_tokenizer.texts\_to\_sequences([text])

padded\_sequence = pad\_sequences(sequence, maxlen=100, padding='post', truncating='post')

prediction = loaded\_model.predict(padded\_sequence)

label\_map\_reverse = {0: "Bearish", 1: "Bullish", 2: "Neutral"}

return label\_map\_reverse[np.argmax(prediction)]

test\_text = input("Enter a Tweet: ")

print(predict\_sentiment(test\_text))

**Step 8: Model Validation**

* The model predicts sentiment for a given tweet.
* The trained model can be used for further testing.