Sentiment Analysis using Deep Learning

Introduction

This code performs sentiment analysis using a deep learning model. Sentiment analysis, also known as opinion mining, is the process of determining the sentiment or emotion expressed in text data. In this project, we analyze sentiment in text reviews.

```
# Importing necessary libraries
import pandas as pd
import re
import string

from sklearn.feature_extraction.text import CountVectorizer, TfidfVectorizer

import tensorflow as tf
from tensorflow.keras.layers import Embedding, Bidirectional, LSTM, Dense
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences
from sklearn.model_selection import train_test_split
import numpy as np
```

Data Preprocessing

Text Cleaning: The text in the "Review Text" column is cleaned by removing emojis and converting the text to lowercase.
Punctuation Removal: Punctuation is removed from the cleaned text.
Tokenization: The text is tokenized for further analysis.
Label Encoding: The "Rating" column is label encoded to represent sentiment labels

```
# Defining a class for text cleaning and preprocessing
class CleanText():
    def __init__(self, clean_pattern=r"[^A-ZĞÜŞİÖÇIa-zğüı'şöç0-9.\"',()]"):
        self.clean_pattern = clean_pattern

    def __call__(self, text):
        if isinstance(text, str):
            docs = [[text]]

        if isinstance(text, list):
            docs = text

        text = [[re.sub(self.clean_pattern, " ", sent) for sent in sents] for sents in docs]

        return text


# Defining a function to remove emojis
def remove_emoji(data):
    emoj = re.compile("["
        u"\U0001F600-\U0001F64F"  # emoticons
        u"\U0001F300-\U0001F5FF"  # symbols & pictographs
        u"\U0001F680-\U0001F6FF"  # transport & map symbols
        u"\U0001F1E0-\U0001F1FF"  # flags (iOS)
        u"\U00002500-\U00002BEF"
        u"\U00002702-\U000027B0"
        u"\U00002702-\U000027B0"
        u"\U000024C2-\U0001F251"
        u"\U0001f926-\U0001f937"
        u"\U00010000-\U0010ffff"
        u"\u2640-\u2642"
        u"\u2600-\u2B55"
        u"\u200d"
        u"\u23cf"
        u"\u23e9"
        u"\u231a"
        u"\ufe0f"  # dingbats
        u"\u3030"
                      "]+", re.UNICODE)
    return re.sub(emoj, '', data)

# Defining a function to tokenize text
def tokenize(text):
    text = re.sub(r" +", " ", str(text))
    text = re.split(r"(\d+|[a-zA-Zğüşıöçğüşİöç]+|\W)", text)
    text = list(filter(lambda x: x != '' and x != ' ', text))
```

```
    sent_tokenized = ' '.join(text)
    return sent_tokenized

regex = re.compile('[%s]' % re.escape(string.punctuation))

# Defining a regular expression for punctuation removal
def remove_punct(text):
    text = regex.sub(" ", text)
    return text

clean = CleanText()

# Defining a function to remove punctuation
def label_encode(x):
    if x == 1 or x == 2:
        return 0
    if x == 3:
        return 1
    if x == 5 or x == 4:
        return 2

# Defining a function to label encode sentiment
def label2name(x):
    if x == 0:
        return "Negative"
    if x == 1:
        return "Neutral"
    if x == 2:
        return "Positive"



# Loading the dataset (replace with your dataset path)
df = pd.read_csv("/content/Womens Clothing E-Commerce Reviews.csv")
df.head()
```

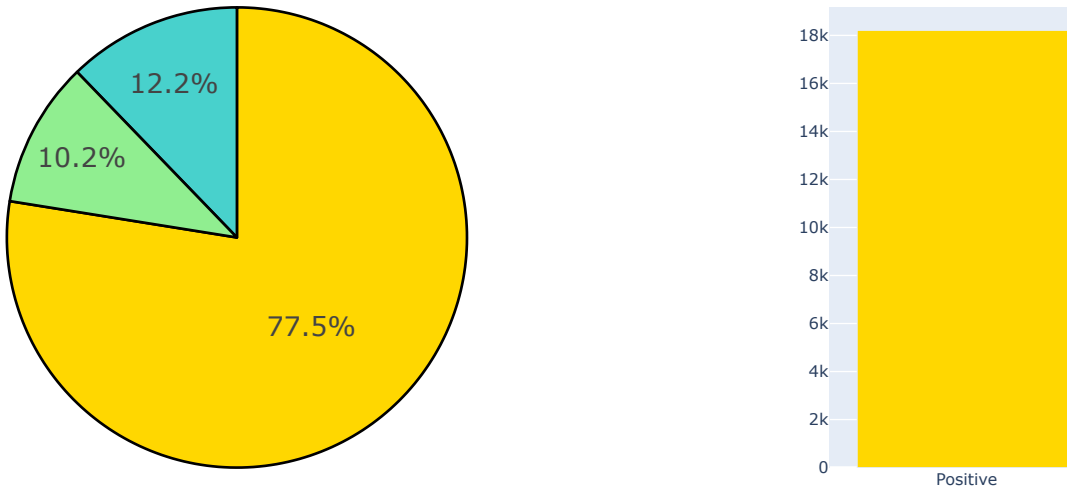
	Unnamed: 0	Clothing ID	Age	Title	Review Text	Rating	Recommended IND	Positive Feedback Count	Division Name	Department Name	Class Name
0	0	767	33	NaN	Absolutely wonderful - silky and sexy and comf...	4	1	0	Initmates	Intimate	Intimates
1	1	1080	34	NaN	Love this dress! it's sooo pretty. i happene...	5	1	4	General	Dresses	Dresses

```
# Label encode the 'Rating' column to convert it to sentiment labels
df["label"] = df["Rating"].apply(lambda x: label_encode(x))
# Map label values to sentiment names
df["label_name"] = df["label"].apply(lambda x: label2name(x))

# Preprocess the 'Review Text' column: lowercase, remove punctuation, and remove emojis
df["Review Text"] = df["Review Text"].apply(str)
df["Review Text"] = df["Review Text"].apply(lambda x: remove_punct(clean(remove_emoji(x).lower())[0][0]))

# Tokenize and pad sequences for model input
tokenizer = Tokenizer()
tokenizer.fit_on_texts(df["Review Text"])
word_index = tokenizer.word_index
sequences = tokenizer.texts_to_sequences(df["Review Text"])
padded_sequences = pad_sequences(sequences, maxlen=100, padding='post', truncating='post')

# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(padded_sequences, df["label"], test_size=0.2, random_state=42)

# Build a deep learning model for sentiment analysis
model = tf.keras.Sequential([
    Embedding(input_dim=len(word_index) + 1, output_dim=128, input_length=100),
    Bidirectional(LSTM(64, return_sequences=True)),
    Bidirectional(LSTM(64)),
    Dense(64, activation='relu'),
    Dense(3, activation='softmax')
])

# Compile the model
model.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics=['accuracy'])

# Train the model
model.fit(X_train, np.array(y_train), epochs=5, validation_data=(X_test, np.array(y_test)))

# Evaluate the model on the testing data
```

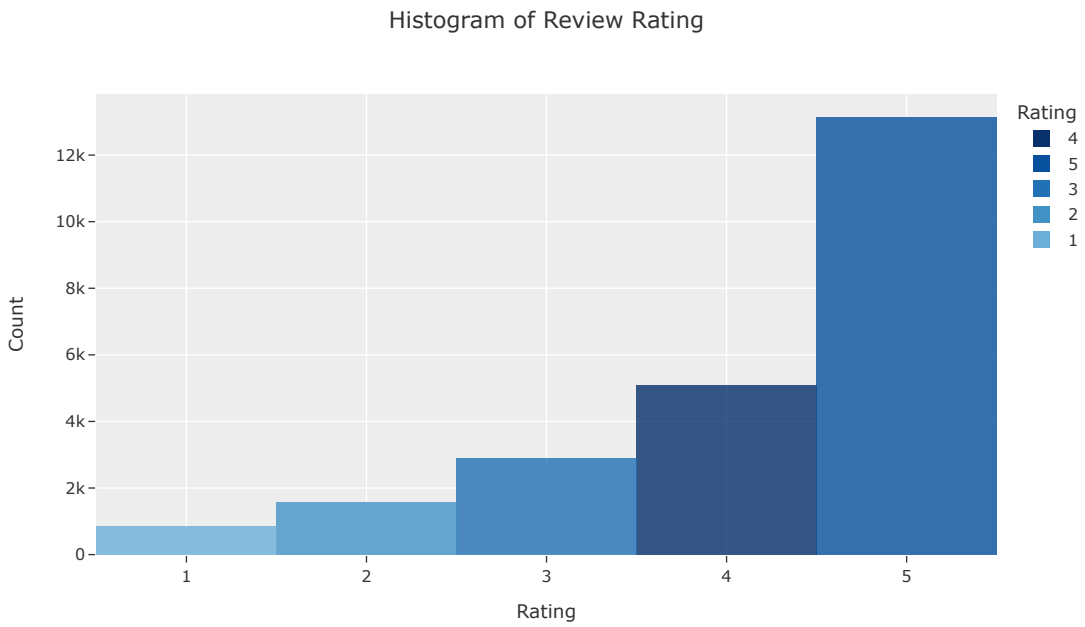
```
test_loss, test_accuracy = model.evaluate(X_test, np.array(y_test))
print(f"Test Loss: {test_loss}")
print(f"Test Accuracy: {test_accuracy}")
```

```
Epoch 1/5
588/588 [==============================] - 108s 175ms/step - loss: 0.5171 - accuracy: 0.7980 - val_loss: 0.4878 - val_accuracy: 0.77
Epoch 2/5
588/588 [==============================] - 98s 167ms/step - loss: 0.3980 - accuracy: 0.8325 - val_loss: 0.4755 - val_accuracy: 0.811
Epoch 3/5
588/588 [==============================] - 99s 168ms/step - loss: 0.3418 - accuracy: 0.8570 - val_loss: 0.4596 - val_accuracy: 0.807
Epoch 4/5
588/588 [==============================] - 100s 170ms/step - loss: 0.3082 - accuracy: 0.8720 - val_loss: 0.4824 - val_accuracy: 0.81
Epoch 5/5
588/588 [==============================] - 99s 168ms/step - loss: 0.2605 - accuracy: 0.8957 - val_loss: 0.5266 - val_accuracy: 0.811
147/147 [==============================] - 6s 41ms/step - loss: 0.5266 - accuracy: 0.8112
Test Loss: 0.5266285538673401
Test Accuracy: 0.8111962676048279
```

```
# The following visualization gives a pie chart of the types of reviews received
fig = make_subplots(rows=1, cols=2, specs=[[{"type": "pie"}, {"type": "bar"}]])
colors = ['gold', 'mediumturquoise', 'lightgreen'] # darkorange
fig.add_trace(go.Pie(labels=df.label_name.value_counts().index,
                             values=df.label.value_counts().values), 1, 1)

fig.update_traces(hoverinfo='label+percent', textfont_size=20,
                  marker=dict(colors=colors, line=dict(color='#000000', width=2)))

fig.add_trace(go.Bar(x=df.label_name.value_counts().index, y=df.label.value_counts().values, marker_color = colors), 1,2)

fig.show()
```




```
# Histogram showing the different ratings received
fig = px.histogram(df,
             x = 'Rating',
             title = 'Histogram of Review Rating',
             template = 'ggplot2',
             color = 'Rating',
             color_discrete_sequence= px.colors.sequential.Blues_r,
             opacity = 0.8,
             height = 525,
             width = 835,
            )

fig.update_yaxes(title='Count')
fig.show()
```


Histogram of Review Rating
Count
12k
10k
8k
6k
4k
2k
0
1
2
3
4
5
Rating
Rating
4
5
3
2
1