The practice is performed following this tutorial: https://www.makeuseof.com/create-sentiment-analysis-model/, which used Trip Advisor Hotel Reviews dataset from Kaggle to build the sentiment analysis model

Dataset used to train the model in this practice: https://www.kaggle.com/datasets/abhi8923shriv/sentiment-analysis-dataset, which is is a Tweet Polarity dataset that is intented for sentiment analysis

```
! pip install tensorflow scikit-learn pandas numpy pickle5
```

```
Requirement already satisfied: tensorflow in /usr/local/lib/python3.10/dist-packages (2.15.0)
Requirement already satisfied: scikit-learn in /usr/local/lib/python3.10/dist-packages (1.2.2)
Requirement already satisfied: pandas in /usr/local/lib/python3.10/dist-packages (2.0.3)
Requirement already satisfied: numpy in /usr/local/lib/python3.10/dist-packages (1.25.2)
Requirement already satisfied: pickle5 in /usr/local/lib/python3.10/dist-packages (0.0.11)
Requirement already satisfied: absl-py>=1.0.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (1.4.0)
Requirement already satisfied: astunparse>=1.6.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (1.6.3)
Requirement already satisfied: flatbuffers>=23.5.26 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (24.3.25)
Requirement already satisfied: gast!=0.5.0,!=0.5.1,!=0.5.2,>=0.2.1 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (0.5.4)
Requirement already satisfied: google-pasta>=0.1.1 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (0.2.0)
Requirement already satisfied: h5py>=2.9.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (3.9.0)
Requirement already satisfied: libclang>=13.0.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (18.1.1)
Requirement already satisfied: ml-dtypes~=0.2.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (0.2.0)
Requirement already satisfied: opt-einsum>=2.3.2 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (3.3.0)
Requirement already satisfied: packaging in /usr/local/lib/python3.10/dist-packages (from tensorflow) (24.0)
Requirement already satisfied: protobuf!=4.21.0,!=4.21.1,!=4.21.2,!=4.21.3,!=4.21.4,!=4.21.5,<5.0.0dev,>=3.20.3 in /usr/local/lib/python
Requirement already satisfied: setuptools in /usr/local/lib/python3.10/dist-packages (from tensorflow) (67.7.2)
Requirement already satisfied: six>=1.12.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (1.16.0)
Requirement already satisfied: termcolor>=1.1.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (2.4.0)
Requirement already satisfied: typing-extensions>=3.6.6 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (4.11.0)
Requirement already satisfied: wrapt<1.15,>=1.11.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (1.14.1)
Requirement already satisfied: tensorflow-io-gcs-filesystem>=0.23.1 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (0.36.0
Requirement already satisfied: grpcio<2.0,>=1.24.3 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (1.62.2)
Requirement already satisfied: tensorboard<2.16,>=2.15 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (2.15.2)
Requirement already satisfied: tensorflow-estimator<2.16,>=2.15.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (2.15.0)
Requirement already satisfied: keras<2.16,>=2.15.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (2.15.0)
Requirement already satisfied: scipy>=1.3.2 in /usr/local/lib/python3.10/dist-packages (from scikit-learn) (1.11.4)
Requirement already satisfied: joblib>=1.1.1 in /usr/local/lib/python3.10/dist-packages (from scikit-learn) (1.4.0)
Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/local/lib/python3.10/dist-packages (from scikit-learn) (3.4.0)
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/python3.10/dist-packages (from pandas) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-packages (from pandas) (2023.4)
Requirement already satisfied: tzdata>=2022.1 in /usr/local/lib/python3.10/dist-packages (from pandas) (2024.1)
Requirement already satisfied: wheel<1.0,>=0.23.0 in /usr/local/lib/python3.10/dist-packages (from astunparse>=1.6.0->tensorflow) (0.43.
Requirement already satisfied: google-auth<3,>=1.6.3 in /usr/local/lib/python3.10/dist-packages (from tensorboard<2.16,>=2.15->tensorflo
Requirement already satisfied: google-auth-oauthlib<2,>=0.5 in /usr/local/lib/python3.10/dist-packages (from tensorboard<2.16,>=2.15->te
Requirement already satisfied: markdown>=2.6.8 in /usr/local/lib/python3.10/dist-packages (from tensorboard<2.16,>=2.15->tensorflow) (3.
Requirement already satisfied: requests<3,>=2.21.0 in /usr/local/lib/python3.10/dist-packages (from tensorboard<2.16,>=2.15->tensorflow)
Requirement already satisfied: tensorboard-data-server<0.8.0,>=0.7.0 in /usr/local/lib/python3.10/dist-packages (from tensorboard<2.16,>
Requirement already satisfied: werkzeug>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from tensorboard<2.16,>=2.15->tensorflow) (3.
Requirement already satisfied: cachetools<6.0,>=2.0.0 in /usr/local/lib/python3.10/dist-packages (from google-auth<3,>=1.6.3->tensorboar
Requirement already satisfied: pyasn1-modules>=0.2.1 in /usr/local/lib/python3.10/dist-packages (from google-auth<3,>=1.6.3->tensorboard
Requirement already satisfied: rsa<5,>=3.1.4 in /usr/local/lib/python3.10/dist-packages (from google-auth<3,>=1.6.3->tensorboard<2.16,>=
Requirement already satisfied: requests-oauthlib>=0.7.0 in /usr/local/lib/python3.10/dist-packages (from google-auth-oauthlib<2,>=0.5->t
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.10/dist-packages (from requests<3,>=2.21.0->tensorboar
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-packages (from requests<3,>=2.21.0->tensorboard<2.16,>=2.1
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.10/dist-packages (from requests<3,>=2.21.0->tensorboard<2.16
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.10/dist-packages (from requests<3,>=2.21.0->tensorboard<2.16
Requirement already satisfied: MarkupSafe>=2.1.1 in /usr/local/lib/python3.10/dist-packages (from werkzeug>=1.0.1->tensorboard<2.16,>=2.
Requirement already satisfied: pyasn1<0.7.0,>=0.4.6 in /usr/local/lib/python3.10/dist-packages (from pyasn1-modules>=0.2.1->google-auth<
Requirement already satisfied: oauthlib>=3.0.0 in /usr/local/lib/python3.10/dist-packages (from requests-oauthlib>=0.7.0->google-auth-oa
```

```
import numpy as np
import pandas as pd
import tensorflow as tf
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Embedding, Conv1D, GlobalMaxPooling1D, Dense, Dropout
import pickle5 as pickle
```

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
```

Loading the Dataset

```
# Load dataset
df_test = pd.read_csv("/content/drive/MyDrive/Sentiment Analysis Dataset/test.csv", encoding='latin1')
df_train = pd.read_csv("/content/drive/MyDrive/Sentiment Analysis Dataset/train.csv", encoding='latin1')

# Select only 'text' and 'sentiment' columns
df_test = df_test[['text', 'sentiment']]
df_train = df_train[['text', 'sentiment']]


# Display the first 5 rows of the datasets
print(df_test.head())
print(df_train.head())
```

```
                                           text sentiment
0  Last session of the day  http://twitpic.com/67ezh   neutral
1   Shanghai is also really exciting (precisely -...  positive
2  Recession hit Veronique Branquinho, she has to...  negative
3                                    happy bday!  positive
4            http://twitpic.com/4w75p - I like it!!  positive
                                           text sentiment
0               I`d have responded, if I were going   neutral
1      Sooo SAD I will miss you here in San Diego!!!  negative
2                          my boss is bullying me...  negative
3                     what interview! leave me alone  negative
4   Sons of ****, why couldn`t they put them on t...  negative
```

Data Preprocessing

```
# Check for missing values in the 'text' column
missing_values_train = df_train['text'].isnull().sum()
print("Number of missing values in 'text' column of training dataset:", missing_values_train)
missing_values_test = df_test['text'].isnull().sum()
print("Number of missing values in 'text' column of testing dataset:", missing_values_test)

# Drop rows with missing values
df_train = df_train.dropna(subset=['text'])
df_test = df_train.dropna(subset=['text'])
```

```
Number of missing values in 'text' column of training dataset: 1
Number of missing values in 'text' column of testing dataset: 1281
```

```
# Tokenization and Padding
tokenizer = Tokenizer(num_words=5000, oov_token='<OOV>')
tokenizer.fit_on_texts(df_train['text'])
word_index = tokenizer.word_index
sequences_train = tokenizer.texts_to_sequences(df_train['text'])
sequences_test = tokenizer.texts_to_sequences(df_test['text'])
padded_sequences_train = pad_sequences(sequences_train, maxlen=100, truncating='post')
padded_sequences_test = pad_sequences(sequences_test, maxlen=100, truncating='post')


# Convert sentiment labels to one-hot encoded vectors
train_sentiment_labels = pd.get_dummies(df_train['sentiment']).values
test_sentiment_labels = pd.get_dummies(df_test['sentiment']).values


# Split data into features and labels
x_train = padded_sequences_train
y_train = train_sentiment_labels
x_test = padded_sequences_test
y_test = test_sentiment_labels
```

Creating and Training the Neural Network

```
# Creating the Neural Network
model = Sequential()
model.add(Embedding(5000, 100, input_length=100))
model.add(Conv1D(64, 5, activation='relu'))
model.add(GlobalMaxPooling1D())
model.add(Dense(32, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(3, activation='softmax'))
model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding (Embedding)       (None, 100, 100)          500000

 conv1d (Conv1D)             (None, 96, 64)            32064

 global_max_pooling1d (Glob  (None, 64)                0
 alMaxPooling1D)

 dense (Dense)               (None, 32)                2080

 dropout (Dropout)           (None, 32)                0

 dense_1 (Dense)             (None, 3)                 99

=================================================================
Total params: 534243 (2.04 MB)
Trainable params: 534243 (2.04 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

```
# Training the Neural Network
model.fit(x_train, y_train, epochs=10, batch_size=32, validation_data=(x_test, y_test))
```

```
Epoch 1/10
859/859 [==============================] - 48s 55ms/step - loss: 0.8239 - accuracy: 0.6266 - val_loss: 0.5814 - val_accuracy: 0.7631
Epoch 2/10
859/859 [==============================] - 37s 43ms/step - loss: 0.6240 - accuracy: 0.7488 - val_loss: 0.4723 - val_accuracy: 0.8157
Epoch 3/10
859/859 [==============================] - 45s 52ms/step - loss: 0.5343 - accuracy: 0.7874 - val_loss: 0.3764 - val_accuracy: 0.8609
Epoch 4/10
859/859 [==============================] - 35s 41ms/step - loss: 0.4345 - accuracy: 0.8337 - val_loss: 0.2760 - val_accuracy: 0.9015
Epoch 5/10
859/859 [==============================] - 37s 43ms/step - loss: 0.3349 - accuracy: 0.8767 - val_loss: 0.2050 - val_accuracy: 0.9328
Epoch 6/10
859/859 [==============================] - 40s 46ms/step - loss: 0.2544 - accuracy: 0.9081 - val_loss: 0.1353 - val_accuracy: 0.9571
Epoch 7/10
859/859 [==============================] - 41s 48ms/step - loss: 0.1938 - accuracy: 0.9311 - val_loss: 0.1063 - val_accuracy: 0.9670
Epoch 8/10
859/859 [==============================] - 37s 43ms/step - loss: 0.1526 - accuracy: 0.9477 - val_loss: 0.0722 - val_accuracy: 0.9784
Epoch 9/10
859/859 [==============================] - 36s 41ms/step - loss: 0.1221 - accuracy: 0.9590 - val_loss: 0.0623 - val_accuracy: 0.9804
Epoch 10/10
859/859 [==============================] - 35s 41ms/step - loss: 0.1077 - accuracy: 0.9658 - val_loss: 0.0516 - val_accuracy: 0.9828
<keras.src.callbacks.History at 0x7a49098bfdf0>
```

```
# Evaluating the Performance of the Trained Model
from sklearn.metrics import f1_score

y_pred = np.argmax(model.predict(x_test), axis=-1)
y_true = np.argmax(y_test, axis=-1)
# Calculate accuracy score
print("Accuracy:", accuracy_score(y_true, y_pred))
# Calculate F1-score
print("F1-score:", f1_score(y_true, y_pred, average='macro'))
```

```
859/859 [==============================] - 8s 9ms/step
Accuracy: 0.9828238719068413
F1-score: 0.9832633729384904
```

```
# Saving the Model
model.save('my_sentiment_analysis_model.h5')
with open('tokenizer.pickle', 'wb') as handle:
    pickle.dump(tokenizer, handle, protocol=pickle.HIGHEST_PROTOCOL)
```

```
/usr/local/lib/python3.10/dist-packages/keras/src/engine/training.py:3103: UserWarning: You are saving your model as an HDF5 file via `m
  saving_api.save_model(
```

Using the Model to Classify the Sentiment of Given Text

```
# Load the saved model and tokenizer
import keras

model = keras.models.load_model('my_sentiment_analysis_model.h5')
with open('tokenizer.pickle', 'rb') as handle:
    tokenizer = pickle.load(handle)


# Define a function to predict the sentiment of input text
def predict_sentiment(text, model, tokenizer):
    # Tokenize and pad the input text
    text_sequence = tokenizer.texts_to_sequences([text])
    text_sequence = pad_sequences(text_sequence, maxlen=100)

    # Make a prediction using the trained model
    predicted_rating = model.predict(text_sequence)[0]

    # Map the predicted sentiment index to its corresponding label
    sentiment_mapping = {0: 'negative', 1: 'neutral', 2: 'positive'}
    predicted_index = np.argmax(predicted_rating)
    predicted_sentiment_label = sentiment_mapping[predicted_index]

    return predicted_index, predicted_sentiment_label


positive_texts = [
    "I loved the new book. It's amazing!",
    "The weather today is beautiful.",
    "I'm feeling great today.",
    "I had a fantastic time at the party last night!",
    "I'm really excited about the upcoming event.",
    "The customer service was excellent!",
    "I feel so happy right now.",
    "The meeting went well.",
    "The hotel room was clean and comfortable.",
    "I'm proud of my achievements."
]
print(len(positive_texts))
```

```
10
```

```
for text in positive_texts:
    predicted_index, predicted_sentiment = predict_sentiment(text, model, tokenizer)
    print("Text:", text)
    print("Predicted Sentiment Index:", predicted_index)
    print("Predicted Sentiment Label:", predicted_sentiment)
    print()
```

```
1/1 [==============================] - 0s 33ms/step
Text: I loved the new book. It's amazing!
Predicted Sentiment Index: 2
Predicted Sentiment Label: positive

1/1 [==============================] - 0s 38ms/step
Text: The weather today is beautiful.
Predicted Sentiment Index: 2
Predicted Sentiment Label: positive

1/1 [==============================] - 0s 39ms/step
Text: I'm feeling great today.
Predicted Sentiment Index: 2
Predicted Sentiment Label: positive

1/1 [==============================] - 0s 33ms/step
Text: I had a fantastic time at the party last night!
Predicted Sentiment Index: 2
Predicted Sentiment Label: positive

1/1 [==============================] - 0s 22ms/step
Text: I'm really excited about the upcoming event.
```

```
Predicted Sentiment Index: 2
Predicted Sentiment Label: positive

1/1 [==============================] - 0s 22ms/step
Text: The customer service was excellent!
Predicted Sentiment Index: 2
Predicted Sentiment Label: positive

1/1 [==============================] - 0s 23ms/step
Text: I feel so happy right now.
Predicted Sentiment Index: 2
Predicted Sentiment Label: positive

1/1 [==============================] - 0s 22ms/step
Text: The meeting went well.
Predicted Sentiment Index: 1
Predicted Sentiment Label: neutral

1/1 [==============================] - 0s 25ms/step
Text: The hotel room was clean and comfortable.
Predicted Sentiment Index: 1
Predicted Sentiment Label: neutral

1/1 [==============================] - 0s 24ms/step
Text: I'm proud of my achievements.
Predicted Sentiment Index: 2
Predicted Sentiment Label: positive
```

```
negative_texts = [
    "The service at the restaurant was terrible.",
    "The traffic was awful this morning.",
    "The food tasted awful.",
    "I'm so disappointed with the product quality.",
    "The flight got delayed again.",
    "I can't stand this waiting.",
    "I'm tired of all this drama.",
    "I'm annoyed by all the noise outside.",
    "The internet connection is so slow.",
    "I'm so frustrated with this project."
]
print(len(negative_texts))
```

```
10
```

```
for text in negative_texts:
    predicted_index, predicted_sentiment = predict_sentiment(text, model, tokenizer)
    print("Text:", text)
    print("Predicted Sentiment Index:", predicted_index)
    print("Predicted Sentiment Label:", predicted_sentiment)
    print()
```

```
1/1 [==============================] - 0s 56ms/step
Text: The service at the restaurant was terrible.
Predicted Sentiment Index: 0
Predicted Sentiment Label: negative

1/1 [==============================] - 0s 38ms/step
Text: The traffic was awful this morning.
Predicted Sentiment Index: 0
Predicted Sentiment Label: negative

1/1 [==============================] - 0s 44ms/step
Text: The food tasted awful.
Predicted Sentiment Index: 0
Predicted Sentiment Label: negative

1/1 [==============================] - 0s 35ms/step
Text: I'm so disappointed with the product quality.
Predicted Sentiment Index: 1
Predicted Sentiment Label: neutral

1/1 [==============================] - 0s 37ms/step
Text: The flight got delayed again.
Predicted Sentiment Index: 0
Predicted Sentiment Label: negative

1/1 [==============================] - 0s 30ms/step
Text: I can't stand this waiting.
Predicted Sentiment Index: 0
Predicted Sentiment Label: negative
```

```
1/1 [==============================] - 0s 26ms/step
Text: I'm tired of all this drama.
Predicted Sentiment Index: 0
Predicted Sentiment Label: negative

1/1 [==============================] - 0s 31ms/step
Text: I'm annoyed by all the noise outside.
Predicted Sentiment Index: 0
Predicted Sentiment Label: negative

1/1 [==============================] - 0s 38ms/step
Text: The internet connection is so slow.
Predicted Sentiment Index: 0
Predicted Sentiment Label: negative

1/1 [==============================] - 0s 30ms/step
Text: I'm so frustrated with this project.
Predicted Sentiment Index: 0
Predicted Sentiment Label: negative
```

```
neutral_texts = [
    "I am going to the store to buy some groceries.",
    "The meeting is scheduled for 2 PM in the conference room.",
    "I need to finish this report by the end of the day.",
    "The new software update includes several bug fixes and improvements.",
    "I'm planning to take a vacation next month.",
    "The book I'm reading is quite interesting.",
    "I'm going to try out a new recipe for dinner tonight.",
    "I'm considering joining a yoga class to improve my flexibility.",
    "The Industrial Revolution marked a significant turning point in human history.",
    "The discovery of penicillin by Alexander Fleming revolutionized the field of medicine."
]
print(len(neutral_texts))
```

```
10
```

```
for text in neutral_texts:
    predicted_index, predicted_sentiment = predict_sentiment(text, model, tokenizer)
    print("Text:", text)
    print("Predicted Sentiment Index:", predicted_index)
    print("Predicted Sentiment Label:", predicted_sentiment)
    print()
```

```
1/1 [==============================] - 0s 27ms/step
Text: I am going to the store to buy some groceries.
Predicted Sentiment Index: 1
Predicted Sentiment Label: neutral

1/1 [==============================] - 0s 45ms/step
Text: The meeting is scheduled for 2 PM in the conference room.
Predicted Sentiment Index: 1
Predicted Sentiment Label: neutral

1/1 [==============================] - 0s 35ms/step
Text: I need to finish this report by the end of the day.
Predicted Sentiment Index: 1
Predicted Sentiment Label: neutral

1/1 [==============================] - 0s 28ms/step
Text: The new software update includes several bug fixes and improvements.
Predicted Sentiment Index: 1
Predicted Sentiment Label: neutral

1/1 [==============================] - 0s 31ms/step
Text: I'm planning to take a vacation next month.
Predicted Sentiment Index: 1
Predicted Sentiment Label: neutral

1/1 [==============================] - 0s 28ms/step
Text: The book I'm reading is quite interesting.
Predicted Sentiment Index: 2
Predicted Sentiment Label: positive

1/1 [==============================] - 0s 28ms/step
Text: I'm going to try out a new recipe for dinner tonight.
Predicted Sentiment Index: 1
Predicted Sentiment Label: neutral

1/1 [==============================] - 0s 27ms/step
```

```
Text: I'm considering joining a yoga class to improve my flexibility.
Predicted Sentiment Index: 2
Predicted Sentiment Label: positive

1/1 [==============================] - 0s 29ms/step
Text: The Industrial Revolution marked a significant turning point in human history.
Predicted Sentiment Index: 1
Predicted Sentiment Label: neutral

1/1 [==============================] - 0s 30ms/step
Text: The discovery of penicillin by Alexander Fleming revolutionized the field of medicine.
Predicted Sentiment Index: 1
Predicted Sentiment Label: neutral
```

Based on the results, the model is struggling with identifying neutral texts, especially differentially between positive and neutral texts.

In the positive text set, 8/10 texts are labeled correctly, with the incorrect one labeled as neutral. In the negative text set, 9/10 texts are labeled correctly, with the incorrect one labeled as neutral. In the neutral text set, 8/10 texts are labeled correctly, and the remaining 2 texts are mislabeled as positive.

When analyzing the dataset, I found that it contains more neutral tweets (40.5% of the tweets are labeled neutral). The trained model has decent accuracy score and F1-score, but it struggles to accurately identify neutral input text. This may be due to some problems with the quality of the dataset and the labeling process for neutral tweets.