```
import numpy as np
import pandas as pd
import tensorflow as tf
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
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
```

```
# Replace 'your_file.csv' with the actual filename you uploaded

df = pd.read_csv("Review_db.csv")  # Read the CSV file into a DataFrame

# Display the first 5 rows
print(df.head())
```

```
                 City            Place  \
0  Aamby Valley City  19 Degree North
1  Aamby Valley City  19 Degree North
2  Aamby Valley City  19 Degree North
3  Aamby Valley City  19 Degree North
4  Aamby Valley City  19 Degree North

                                              Review  Rating       Name  Date  \
0  aamby valley beautiful place clear blue skies ...       5  Anonymous   NaN
1  executed obt akshay thanx team thoroughly enjo...       4  Anonymous   NaN
2  awesome experience atv tracts obstacles mainta...       5  Anonymous   NaN
3  visited aamby valley yesterday short excursion...       4  Anonymous   NaN
4  far mumbai place finest adventure places visit...       5  Anonymous   NaN

                                          Raw_Review
0  Aamby valley is a beautiful place with its cle...
1  Very well executed obt by Akshay.... Thanx as ...
2  Awesome experience at the ATV\nTracts and obst...
3  we visited the Aamby Valley yesterday for shor...
4  Not far from Mumbai, this place is one of the ...
```

```
# Define a function to map ratings to sentiment categories
def map_rating_to_sentiment(rating):
    if rating >= 4:
        return "Positive"
    elif rating == 3:
        return "Neutral"
    else:
        return "Negative"

# Apply the function to create the sentiment column
df['sentiment'] = df['Rating'].apply(map_rating_to_sentiment)

# Drop the rating column (optional)
df.drop(columns=['Rating'], inplace=True)
```

```
# Display first few rows to verify
print(df.head())
```

```
                 City            Place  \
0  Aamby Valley City  19 Degree North
1  Aamby Valley City  19 Degree North
2  Aamby Valley City  19 Degree North
3  Aamby Valley City  19 Degree North
4  Aamby Valley City  19 Degree North

                                              Review       Name  Date  \
0  aamby valley beautiful place clear blue skies ...  Anonymous   NaN
1  executed obt akshay thanx team thoroughly enjo...  Anonymous   NaN
2  awesome experience atv tracts obstacles mainta...  Anonymous   NaN
3  visited aamby valley yesterday short excursion...  Anonymous   NaN
4  far mumbai place finest adventure places visit...  Anonymous   NaN

                                          Raw_Review sentiment
0  Aamby valley is a beautiful place with its cle...  Positive
1  Very well executed obt by Akshay.... Thanx as ...  Positive
2  Awesome experience at the ATV\nTracts and obst...  Positive
3  we visited the Aamby Valley yesterday for shor...  Positive
4  Not far from Mumbai, this place is one of the ...  Positive
```

```
from sklearn.preprocessing import LabelEncoder

label_encoder = LabelEncoder()
df['sentiment'] = label_encoder.fit_transform(df['sentiment'])

# Show mapping of labels
label_mapping = dict(zip(label_encoder.classes_, label_encoder.transform(label_encoder.cl
print(label_mapping)  # Example Output: {'Negative': 0, 'Neutral': 1, 'Positive': 2}
```

```
{'Negative': np.int64(0), 'Neutral': np.int64(1), 'Positive': np.int64(2)}
```

```
# Parameters
max_words = 10000  # Consider top 10,000 words
max_len = 100  # Maximum words per review

# Tokenization
tokenizer = Tokenizer(num_words=max_words, oov_token="<OOV>")
tokenizer.fit_on_texts(df['Review'])

# Convert text to sequences
sequences = tokenizer.texts_to_sequences(df['Review'])
padded_sequences = pad_sequences(sequences, maxlen=max_len, padding='post')

# Splitting data into train and test sets
X_train, X_test, y_train, y_test = train_test_split(padded_sequences, df['sentiment'], te
```

```
model = Sequential([
    Embedding(input_dim=max_words, output_dim=128, input_length=max_len),
    Conv1D(filters=128, kernel_size=5, activation='relu'),
    GlobalMaxPooling1D(),
    Dense(64, activation='relu'),
    Dropout(0.5),
    Dense(3, activation='softmax')  # 3 output classes (Negative, Neutral, Positive)
])

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

# Summary of the model
model.summary()
```

```
/usr/local/lib/python3.11/dist-packages/keras/src/layers/core/embedding.py:90: UserWa
  warnings.warn(
```

Model: "sequential"

Layer (type)	Output Shape	Param
embedding (Embedding)	?	0 (unbuilt
conv1d (Conv1D)	?	0 (unbuilt
global_max_pooling1d (GlobalMaxPooling1D)	?	
dense (Dense)	?	0 (unbuilt
dropout (Dropout)	?	
dense_1 (Dense)	?	0 (unbuilt

Total params: 0 (0.00 B)
Trainable params: 0 (0.00 B)
Non-trainable params: 0 (0.00 B)

```
epochs = 1
batch_size = 32

history = model.fit(X_train, y_train, validation_data=(X_test, y_test), epochs=epochs, ba
```

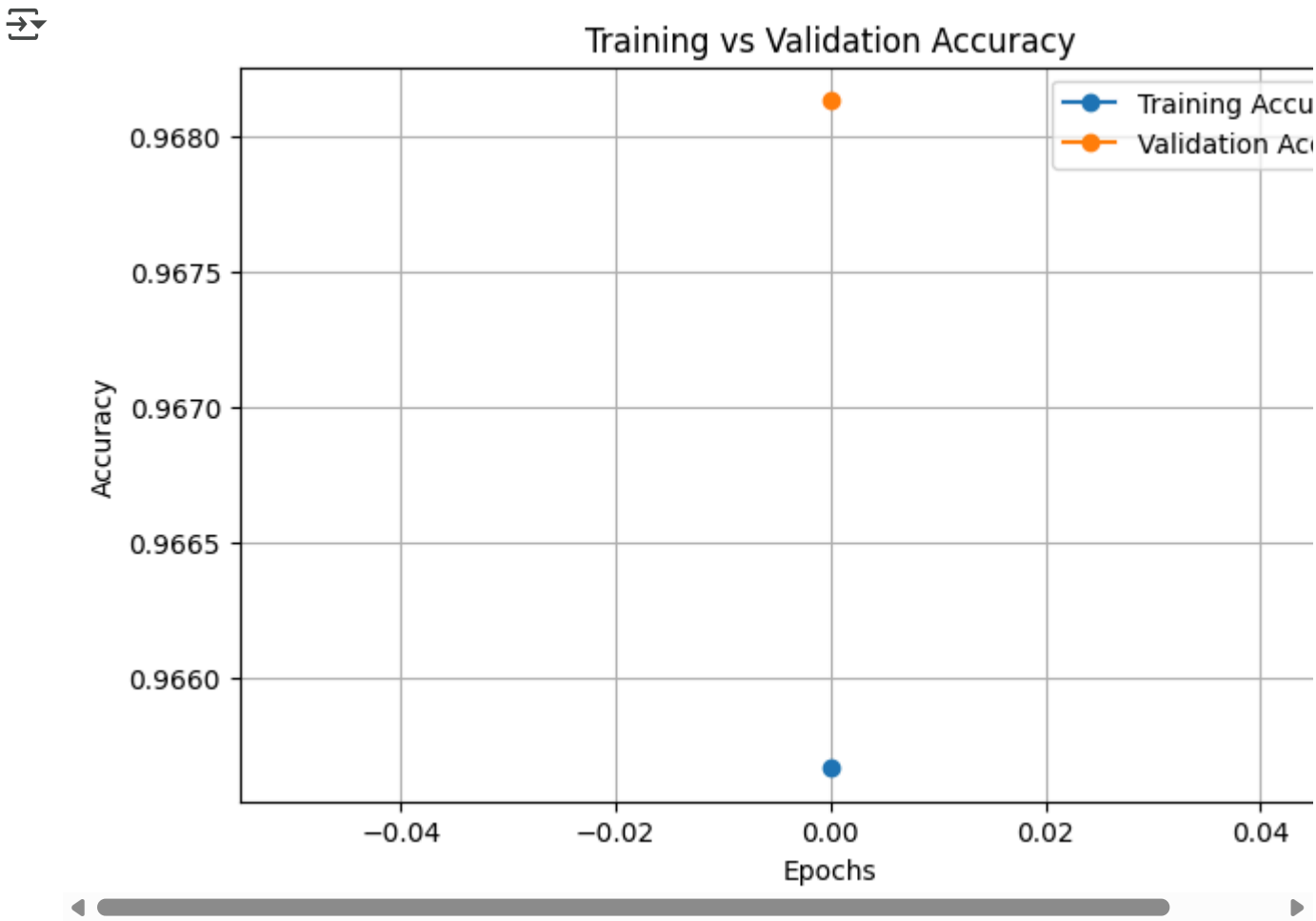
```
37062/37062 ━━━━━━━━━━━━━━━━━━━━ 2116s 57ms/step - accuracy: 0.9657 - loss: 0.1015 -
```

```
test_loss, test_accuracy = model.evaluate(X_test, y_test)
print(f"Test Accuracy: {test_accuracy:.4f}")
```

```
9266/9266 ━━━━━━━━━━━━━━━━━━━━ 128s 14ms/step - accuracy: 0.9679 - loss: 0.0957
Test Accuracy: 0.9681
```

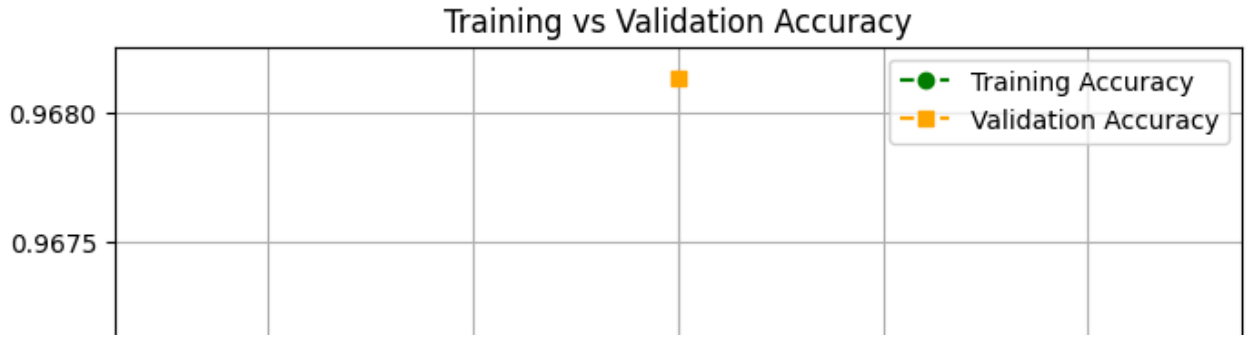
```
import matplotlib.pyplot as plt

plt.figure(figsize=(8, 5))
plt.plot(history.history['accuracy'], label='Training Accuracy', marker='o')
plt.plot(history.history['val_accuracy'], label='Validation Accuracy', marker='o')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.title('Training vs Validation Accuracy')
plt.legend()
plt.grid(True)
plt.show()
```




```
plt.figure(figsize=(8, 5))
plt.plot(history.history['loss'], label='Training Loss', marker='o', color='red')
plt.plot(history.history['val_loss'], label='Validation Loss', marker='o', color='blue')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.title('Training vs Validation Loss')
plt.legend()
plt.grid(True)
plt.show()
```


```
plt.figure(figsize=(8, 5))
plt.plot(history.history['accuracy'], label='Training Accuracy', linestyle='dashed', mark
plt.plot(history.history['val_accuracy'], label='Validation Accuracy', linestyle='dashed'
plt.fill_between(range(len(history.history['accuracy'])), history.history['accuracy'], hi
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.title('Training vs Validation Accuracy')
plt.legend()
plt.grid(True)
plt.show()
```




```
def predict_review(review):
    sequence = tokenizer.texts_to_sequences([review])
    padded_sequence = pad_sequences(sequence, maxlen=max_len, padding='post')
    prediction = model.predict(padded_sequence)
    sentiment = np.argmax(prediction)

    sentiment_label = {v: k for k, v in label_mapping.items()}  # Reverse mapping
    return sentiment_label[sentiment]

# Example review prediction
new_review = "The place was amazing, I had a great experience!"
print("Predicted Sentiment:", predict_review(new_review))
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 134ms/step
Predicted Sentiment: Positive
```