chit-6-imd-reviews

May 6, 2025

[ ]:

[ ]:

[21]:
```
import tensorflow as tf
from mlxtend.plotting import plot_confusion_matrix
from sklearn import metrics
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
#%matplotlib inline
#from tqdm.notebook import tqdm
#import warnings
#warnings.filterwarnings("ignore")
```

[22]:
```
vocab_size = 10000
max_len = 200
(x_train, y_train), (x_test, y_test) = tf.keras.datasets.imdb.
 ↪load_data(num_words=vocab_size)
```

[23]:
```
x_train.shape, y_train.shape, x_test.shape, y_test.shape
```

[23]:
```
((25000,), (25000,), (25000,), (25000,))
```

[24]:
```
x_train = tf.keras.preprocessing.sequence.
 ↪pad_sequences(x_train,maxlen=max_len,padding='post')
x_test = tf.keras.preprocessing.sequence.
 ↪pad_sequences(x_test,maxlen=max_len,padding='post')
```

[25]:
```
x_train.shape, y_train.shape, x_test.shape, y_test.shape
```

[25]:
```
((25000, 200), (25000,), (25000, 200), (25000,))
```

[26]:
```
model = tf.keras.Sequential([
    tf.keras.layers.Embedding(vocab_size, 128, input_length=max_len),
    tf.keras.layers.Flatten(),
```

```
    tf.keras.layers.Dense(128, activation='relu'),
    tf.keras.layers.Dense(1, activation='sigmoid')
])
```

```
C:\Users\LENOVO.LAPTOP-K3FTEK88\AppData\Roaming\Python\Python310\site-
packages\keras\src\layers\core\embedding.py:90: UserWarning: Argument
`input_length` is deprecated. Just remove it.
  warnings.warn(
```

[27]:
```
model.compile(optimizer='adam', loss='binary_crossentropy',␣
 ↪metrics=['accuracy'])
```

[28]:
```
model.summary()
```

```
Model: "sequential_1"
```

Layer (type)	Output Shape	Param #
embedding_1 (Embedding) (unbuilt)	?	0
flatten_1 (Flatten) (unbuilt)	?	0
dense_2 (Dense) (unbuilt)	?	0
dense_3 (Dense) (unbuilt)	?	0

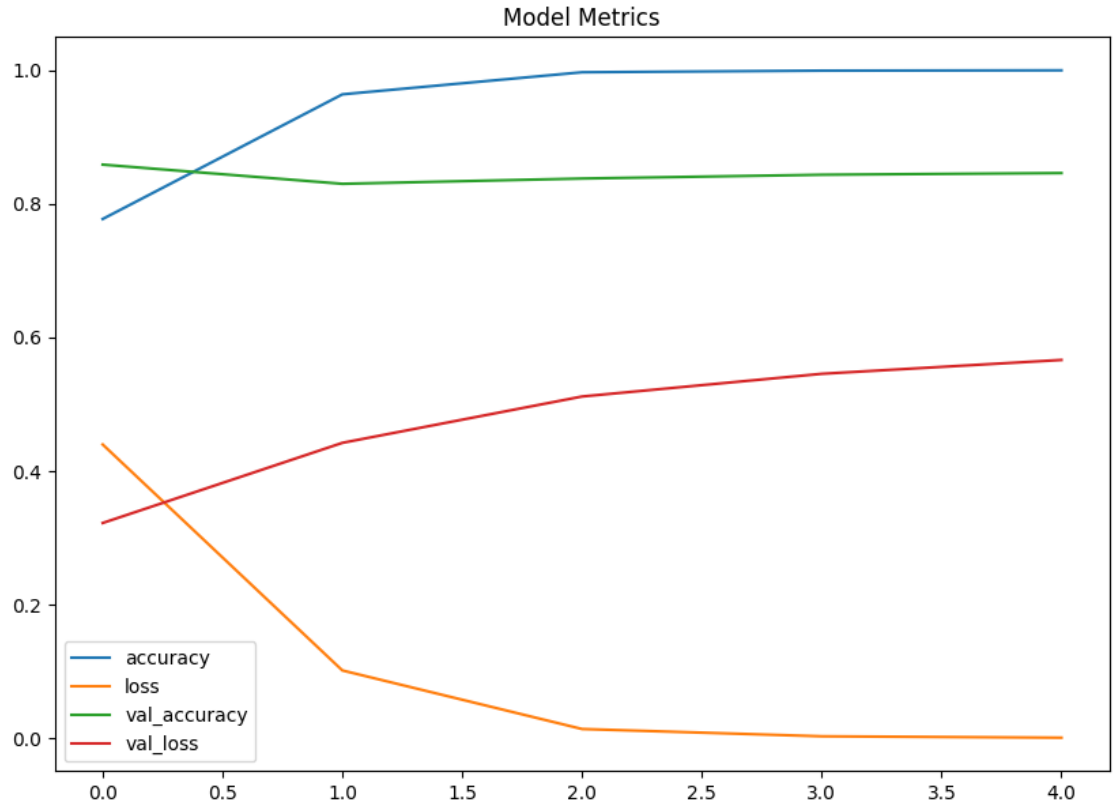
```
Total params: 0 (0.00 B)

Trainable params: 0 (0.00 B)

Non-trainable params: 0 (0.00 B)
```

[29]:
```
model.compile(optimizer='adam', loss='binary_crossentropy',␣
 ↪metrics=['accuracy'])
```

[30]:
```
history = model.fit(x_train, y_train, batch_size=128, epochs=5,␣
 ↪validation_data=(x_test,y_test))
```

```
Epoch 1/5
196/196             13s 61ms/step -
accuracy: 0.6782 - loss: 0.5563 - val_accuracy: 0.8588 - val_loss: 0.3221
Epoch 2/5
196/196             12s 59ms/step -
accuracy: 0.9657 - loss: 0.1045 - val_accuracy: 0.8300 - val_loss: 0.4421
Epoch 3/5
196/196             11s 56ms/step -
accuracy: 0.9974 - loss: 0.0148 - val_accuracy: 0.8380 - val_loss: 0.5115
Epoch 4/5
196/196             11s 56ms/step -
accuracy: 0.9994 - loss: 0.0029 - val_accuracy: 0.8436 - val_loss: 0.5455
Epoch 5/5
196/196             11s 57ms/step -
accuracy: 1.0000 - loss: 6.1680e-04 - val_accuracy: 0.8462 - val_loss: 0.5663
```

[31]:
```
pd.DataFrame(history.history).plot(figsize=(10,7))
plt.title("Model Metrics")
plt.show()
```



[32]:
```
loss, accuracy = model.evaluate(x_test,y_test)
print("Test Accuracy:",accuracy)
```

```
782/782             3s 3ms/step -
accuracy: 0.8484 - loss: 0.5555
Test Accuracy: 0.8461599946022034
```

[33]:
```
y_pred = model.predict(x_test)
```

```
782/782             3s 3ms/step
```

[34]:
```
y_pred
```

[34]:
```
array([[0.00498131],
       [0.999999  ],
       [0.07536433],
       …,
       [0.00197054],
       [0.7076974 ],
       [0.11507031]], dtype=float32)
```

[35]:
```
y_pred = y_pred.flatten()
```

[36]:
```
y_pred
```

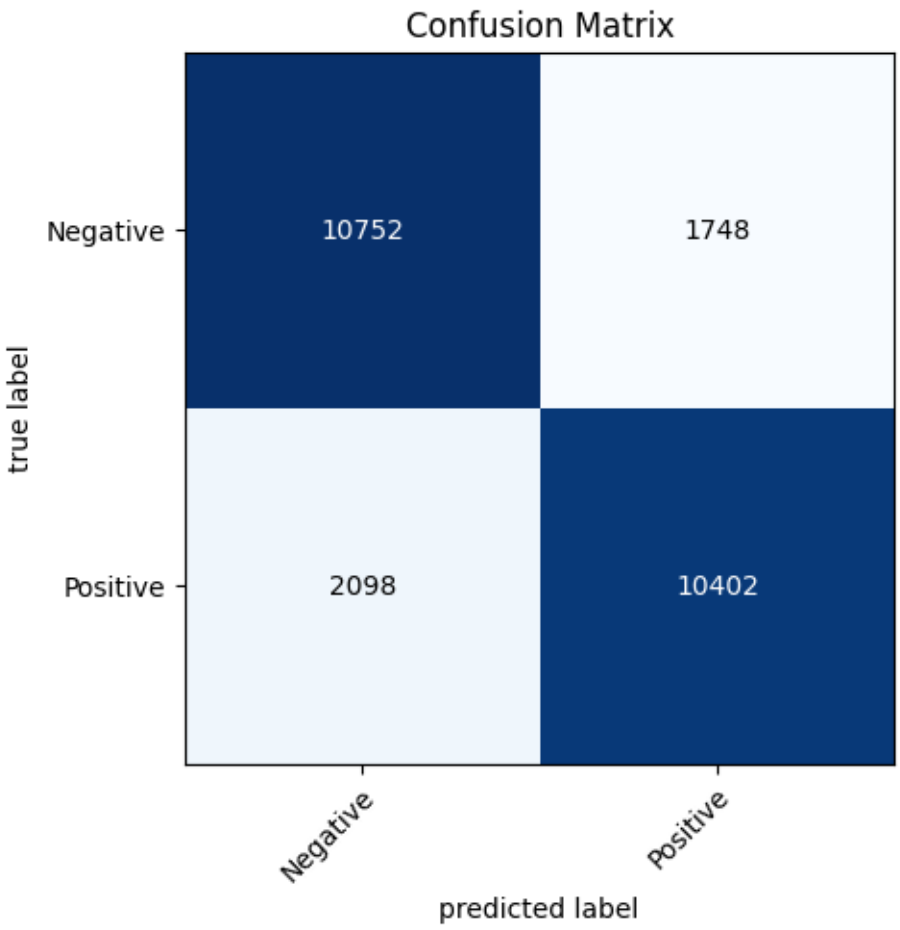
[36]:
```
array([0.00498131, 0.999999  , 0.07536433, …, 0.00197054, 0.7076974 ,
       0.11507031], dtype=float32)
```

[37]:
```
y_pred = (y_pred > 0.5).astype(int)
```

[38]:
```
print(metrics.classification_report(y_test,y_pred))
```

```
              precision    recall  f1-score   support

           0       0.84      0.86      0.85     12500
           1       0.86      0.83      0.84     12500

    accuracy                           0.85     25000
   macro avg       0.85      0.85      0.85     25000
weighted avg       0.85      0.85      0.85     25000
```

[43]:
```
cm = metrics.confusion_matrix(y_test, y_pred)
plot_confusion_matrix(cm, class_names=['Negative', 'Positive'])
plt.title("Confusion Matrix")
plt.show()
```



[42]:
```
import numpy as np
from tensorflow.keras.datasets import imdb
from tensorflow.keras.preprocessing.sequence import pad_sequences

# Load IMDB word index (word -> integer mapping)
word_index = imdb.get_word_index()

# Function to preprocess and predict sentiment for external text input
def predict_sentiment(text):
    # Convert text to lowercase and split into words
    words = text.lower().split()

    # Convert words to their corresponding indices, using 2 as the index for␣
 ↪unknown words
    sequence = [word_index.get(word, 2) for word in words]

    # Pad the sequence to match the model's input length
    padded_sequence = pad_sequences([sequence], maxlen=max_len)
```

```
    # Predict sentiment
    prediction = model.predict(padded_sequence)

    sentiment = 'Positive' if prediction[0][0] >= 0.5 else 'Negative'
    print(f"Sentiment: {sentiment} (Confidence: {prediction[0][0]:.4f})")

# Example usage
sample_review =  "absolutely fantastic with great performances"
predict_sentiment(sample_review)
```

```
1/1             0s 21ms/step
Sentiment: Negative (Confidence: 0.2201)
```

[52]:
```
y_pred[505]
```

[52]:
```
0
```

[53]:
```
y_test[505]
```

[53]:
```
0
```

[ ]:
```

```