65070501037
Paweekorn Soratyathorn

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

```
df = pd.read_csv('IMDB Dataset.csv')
df.head()
```

	review	sentiment
0	One of the other reviewers has mentioned that ...	positive
1	A wonderful little production. The...	positive
2	I thought this was a wonderful way to spend ti...	positive
3	Basically there's a family where a little boy ...	negative
4	Petter Mattei's "Love in the Time of Money" is...	positive

Next steps: Generate code with df | View recommended plots | New interactive sheet

```
pd.crosstab(df.sentiment, columns="Count").sort_values(by='Count', ascending=False)
```

col_0	Count
sentiment	
negative	25000
positive	25000

Data cleansing

```
import nltk
from nltk.corpus import stopwords
import re

nltk.download('stopwords')

REPLACE_BY_SPLACE_RE = re.compile(r'[/(){}\[\]\|@,;]')
BAD_SYMBOLS_RE = re.compile(r'[^0-9a-z #+_]')
STOPWORDS = set(stopwords.words('english'))
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Unzipping corpora/stopwords.zip.
```

```
def clean_text(text):
  text = text.lower()
  text = REPLACE_BY_SPLACE_RE.sub(' ', text)
  text = BAD_SYMBOLS_RE.sub('', text)
  text = text.replace('x', '')
  text = ' '.join(word for word in text.split() if word not in STOPWORDS)
  return text
```

```
df['review'] = df['review'].apply(clean_text)
df['review'] = df['review'].str.replace(r'\d+','')
```

```
def print_plot(index):
  example = df[df.index == index][['review', 'sentiment']].values[0]
  if len(example) > 0:
    print(example[0])
    print('Review: ',example[1])

print_plot(10)
```

```
phil alien one quirky films humour based around oddness everything rather actual punchlinesbr br first odd pretty funny movie progre
Review:  negative
```

Preprocessing

Train Tokenizer

```
from tensorflow.keras.preprocessing.text import Tokenizer

MAX_NB_WORDS = 50000

MAX_SEQUENCE_LENGTH = 250

EMBEDDING_DIM = 100

tokenizer = Tokenizer(num_words = MAX_NB_WORDS,
                      filters = '!"#$%&()*+,-./:;<=>?@[\]^_`{|}~',
                      lower=True)

tokenizer.fit_on_texts(df['review'].values)
word_index = tokenizer.word_index
print('Found %s unique tokens.'%len(word_index))
```

```
Found 165306 unique tokens.
```

Pad sequence

```
from tensorflow.keras.preprocessing.sequence import pad_sequences

X = tokenizer.texts_to_sequences(df['review'].values)[:2500]
X = pad_sequences(X, maxlen = MAX_SEQUENCE_LENGTH)
print('Shape of data tensor:', X.shape)
```

```
Shape of data tensor: (2500, 250)
```

```
print(df['review'].values[0], '\n')
X[0]
```

```
one reviewers mentioned watching 1 oz episode youll hooked right eactly happened mebr br first thing struck oz brutality unflinching

array([    0,     0,     0,     0,     0,     0,     0,     0,     0,
           0,     0,     0,     0,     0,     0,     0,     0,     0,
           0,     0,     0,     0,     0,     0,     0,     0,     0,
           0,     0,     0,     0,     0,     0,     0,     0,     0,
           0,     0,     0,     0,     0,     0,     0,     0,     0,
           0,     0,     0,     0,     0,     0,     0,     0,     0,
           0,     0,     0,     0,     0,     0,     0,     0,     0,
           0,     0,     0,     0,     0,     0,     0,     0,     0,
           0,     0,     0,     0,     0,     0,     0,     0,     0,
           0,     0,     0,     0,     4,  1834,   950,    57,   233,
        3193,   288,   353,  3079,   110,   492,   480,  2107,     1,
          20,    58,  3138,  3193,  5451, 15271,    51,   461,   182,
         110,   560,    53,  1585,    42,  8154,  5657, 11761,    42,
        2394,  5953,  5452,  1337,   264,   461,  3267,   249,   239,
       23365,     1,   364,  3193, 11400,   237, 15905,  6763,  2415,
         947,  2521,  1257, 25213,   425,  4483,  2409,  1081,  6991,
        2860, 12894,   302, 17412,   214,  4942,  3559,   425,   241,
        8294, 40799, 15272,  5061,  7725,  2315, 18295,   224,  9040,
        7356, 13122,  8621, 34707,    35,   128,  5513,     1,     8,
          47,   171,  1191,    42,   557,    95,   163,   159,   439,
        2874,   706,    86,  1150,  4228,  2379,   984,   706,  1295,
         706,    60,   869,    89,    20,   288,    44,   106,  3138,
        1463,  2090,   293,    47,  1437,   178,  1356,  1138,  3193,
          92, 10168,   214,  1973,  1976,   461,   461,  7856,  6992,
        4842, 14080,  2861, 32394,  6934, 14080,   384,   515,    15,
         144,    14,  9815,   639,   703,  6934,   551,  1081, 20459,
         557,   440,   814,  1880,  1081,   448,    57,  3193,   102,
         308,  3653,  3161,    15,  1090,  3906,   394], dtype=int32)
```

```
Y = pd.get_dummies(df['sentiment']).values[:2500]
print('Shape of label tensor:', Y.shape)
```

```
Shape of label tensor: (2500, 2)
```

```
Y
```

```
array([[False,  True],
       [False,  True],
```

```
       [False,  True],
       ...,
       [False,  True],
       [ True, False],
       [False,  True]])
```

Training Model

```
from sklearn.model_selection import train_test_split

X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.3, random_state=42)

print(X_train.shape, Y_train.shape)
print(X_test.shape, Y_test.shape)
```

```
(1750, 250) (1750, 2)
(750, 250) (750, 2)
```

```
from keras import Input
from keras.models import Sequential
from keras.layers import Embedding, SpatialDropout1D, LSTM, Dense
from keras.callbacks import EarlyStopping

model = Sequential()
model.add(Input(shape=(X.shape[1], )))
model.add(Embedding(MAX_NB_WORDS, EMBEDDING_DIM))
model.add(SpatialDropout1D(0.2))
model.add(LSTM(100, dropout=0.2, recurrent_dropout=0.2))
model.add(Dense(2, activation='softmax'))
model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])

epochs=3
batch_size=64

history = model.fit(X_train, Y_train,
                    epochs=epochs,
                    batch_size=batch_size,
                    validation_split=0.1,
                    callbacks=[EarlyStopping(monitor='val_loss', patience=3, min_delta=0.0001)])
```

```
Epoch 1/3
25/25 ━━━━━━━━━━━━━━━━━━━━ 16s 504ms/step - accuracy: 0.5439 - loss: 0.6913 - val_accuracy: 0.5943 - val_loss: 0.6696
Epoch 2/3
25/25 ━━━━━━━━━━━━━━━━━━━━ 25s 682ms/step - accuracy: 0.7983 - loss: 0.5829 - val_accuracy: 0.7429 - val_loss: 0.5268
Epoch 3/3
25/25 ━━━━━━━━━━━━━━━━━━━━ 16s 480ms/step - accuracy: 0.9142 - loss: 0.2584 - val_accuracy: 0.8057 - val_loss: 0.4707
```

```
model.summary()
```

Model: "sequential_1"

Layer (type)	Output Shape	Param #
embedding_1 (Embedding)	(None, 250, 100)	5,000,000
spatial_dropout1d_1 (SpatialDropout1D)	(None, 250, 100)	0
lstm_1 (LSTM)	(None, 100)	80,400
dense_1 (Dense)	(None, 2)	202

Total params: 15,241,808 (58.14 MB)
Trainable params: 5,080,602 (19.38 MB)
Non-trainable params: 0 (0.00 B)
Optimizer params: 10,161,206 (38.76 MB)

Evaluation

```
acc = model.evaluate(X_test, Y_test)
print('Test set \n\tLoss: {:0.3f}\n\tAccuracy: {:0.3f}'.format(acc[0], acc[1]))
```

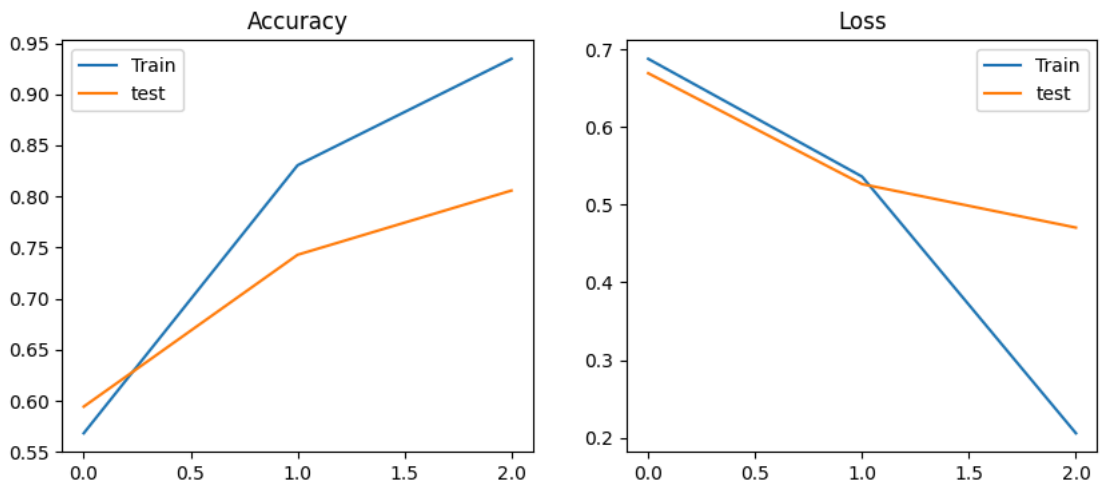
```
24/24 ━━━━━━━━━━━━━━━━━━━━ 1s 55ms/step - accuracy: 0.8079 - loss: 0.4661
Test set
        Loss: 0.476
        Accuracy: 0.801
```

```
def plot_history(history):
  plt.figure(figsize=(10, 4))

  plt.subplot(1, 2, 1)
  plt.title('Accuracy')
  plt.plot(history.history['accuracy'], label='Train')
  plt.plot(history.history['val_accuracy'], label='test')
  plt.legend()

  plt.subplot(1, 2, 2)
  plt.title('Loss')
  plt.plot(history.history['loss'], label='Train')
  plt.plot(history.history['val_loss'], label='test')
  plt.legend()


plot_history(history)
```



Confusion matrix

```
labels = pd.get_dummies(df['sentiment']).columns

from sklearn.metrics import confusion_matrix
y_pred = model.predict(X_test)

pd.DataFrame(confusion_matrix(Y_test.argmax(axis=1),
                              y_pred.argmax(axis=1)),
             index=labels, columns=labels)
```

24/24 ━━━━━━━━━━━━━━━━━━━━ 6s 224ms/step

	negative	positive
negative	295	82
positive	67	306

Classification Report

```
from sklearn.metrics import classification_report

y_pred = model.predict(X_test)
print(classification_report(y_true=Y_test.argmax(axis=1),
                            y_pred=y_pred.argmax(axis=1)))
```

```
24/24 ━━━━━━━━━━━━━━━━━━━━ 2s 87ms/step
              precision    recall  f1-score   support

           0       0.81      0.78      0.80       377
           1       0.79      0.82      0.80       373

    accuracy                           0.80       750
   macro avg       0.80      0.80      0.80       750
weighted avg       0.80      0.80      0.80       750
```

Test using new review

```
new_review = ['''Visually beautiful but unfortunately the dialogues are poorly written and superficial,
Galedriel who has a lot of screen time and massive importance unfortunately is portrayed very badly makes me questic
how could they allow such performance, not to insult the actress at all but the performance was horrible almost
everyone in the show was better performing than her. Overall fun to watch but very superficial and Shallow.
The plot is yet to reveal itself but so far nothing interesting. Good to mention the good portrayal of elrond.
elendil. Music is also good and appropriate. The dark theme is a bit overlooked.''']
```

```
seq = tokenizer.texts_to_sequences(new_review)
padded = pad_sequences(seq, maxlen=MAX_SEQUENCE_LENGTH)
pred = model.predict(padded)
print(pred, labels[np.argmax(pred)])
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 151ms/step
[[0.6472341 0.3527659]] negative
```

Start coding or generate with AI.