```
import pandas as pd

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

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 50000 entries, 0 to 49999
Data columns (total 2 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   review     50000 non-null  object
 1   sentiment  50000 non-null  object
dtypes: object(2)
memory usage: 781.4+ KB
```

Data Cleaning

```
import nltk
nltk.download('stopwords')
```

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

```
from nltk.corpus import stopwords
stopwords.words('english')[:10]
```

```
['i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves', 'you', "you're"]
```

```
import re

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


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('\d+','')
```

Make the label to be 1 and 0 for binary classification

```
df['sentiment'].value_counts()
```

sentiment	count
positive	25000
negative	25000

```
df['sentiment'] = df['sentiment'].map({'positive': 1, 'negative': 0})
```

```
df.head()
```

	review	sentiment
0	one reviewers mentioned watching 1 oz episode ...	1
1	wonderful little production br br filming tech...	1
2	thought wonderful way spend time hot summer we...	1
3	basically theres family little boy jake thinks...	0
4	petter mattei's love time money visually stunni...	1

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

Modeling

1. Vectorie input
2. Limit dataset to top 50000 words
3. Set max number of words in each review to 250

```
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.
```

```
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)
```

```
df['review'].values[0]
```

```
'one reviewers mentioned watching 1 oz episode youll hooked right eactly happened mebr br first thing struck oz brutality unflinchi
ng scenes violence set right word go trust show faint hearted timid show pulls punches regards drugs se violence hardcore classic u
se wordbr br called oz nickname given oswald maimum security state penitentary focuses mainly emerald city eperimental section pris
on cells glass fronts face inwards privacy high agenda em city home manyaryans muslims gangstas latinos christians italians irish m
oreso scuffles death stares dodgy dealings shady agreements never far awaybr br would say main appeal show due fact goes shows woul
dnt dare forget pretty pictures painted mainstream audiences forget charm forget romanceoz doesnt mess around first episode ever sa
```

```
X[0]
```

```
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)
```

Convert output label into numeric format

```
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]])
```

Split dataset to Training and Test set

```
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)
```

Construct LSTM Text Classifier

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

model = Sequential()
model.add(Embedding(MAX_NB_WORDS, EMBEDDING_DIM, input_length = X.shape[1]))
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'])

model.summary()
```

```
/usr/local/lib/python3.10/dist-packages/keras/src/layers/core/embedding.py:90: UserWarning: Argument `input_length` is deprecated. J
  warnings.warn(
```

Model: "sequential_2"

Layer (type)	Output Shape	Param #
embedding_2 (Embedding)	?	0 (unbuilt)
spatial_dropout1d_2 (SpatialDropout1D)	?	0 (unbuilt)
lstm_2 (LSTM)	?	0 (unbuilt)
dense_2 (Dense)	?	0 (unbuilt)

Total params: 0 (0.00 B)

```
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 ━━━━━━━━━━━━━━━━━━━━ 23s 748ms/step - accuracy: 0.5168 - loss: 0.6918 - val_accuracy: 0.5429 - val_loss: 0.6711
Epoch 2/3
25/25 ━━━━━━━━━━━━━━━━━━━━ 17s 677ms/step - accuracy: 0.8019 - loss: 0.5706 - val_accuracy: 0.7314 - val_loss: 0.5145
Epoch 3/3
25/25 ━━━━━━━━━━━━━━━━━━━━ 17s 668ms/step - accuracy: 0.9460 - loss: 0.2049 - val_accuracy: 0.8114 - val_loss: 0.4903
```

Evaluate the model

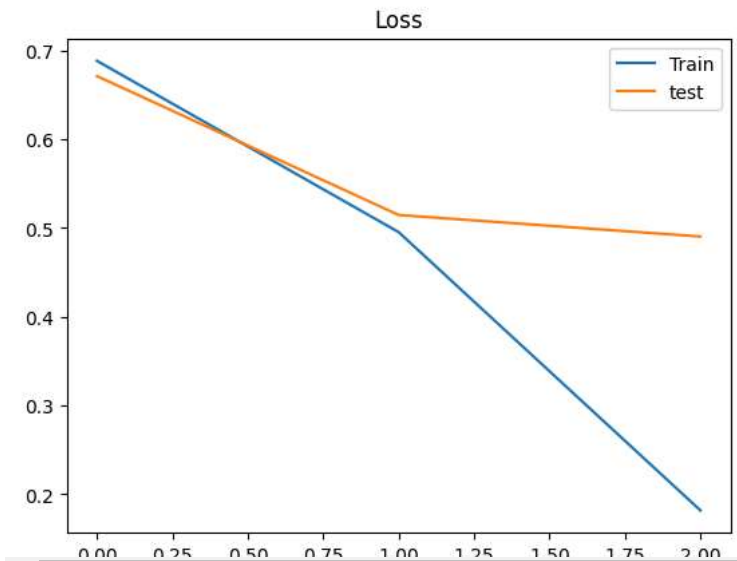
```
acc = model.evaluate(X_test, Y_test)
print('Test set \n\tLoss: {:0.3f}\n\tAccuracy: {:0.3f}'.format(acc[0], acc[1]))
```

```
24/24 ━━━━━━━━━━━━━━━━━━━━ 1s 62ms/step - accuracy: 0.8113 - loss: 0.4390
Test set
        Loss: 0.435
        Accuracy: 0.813
```

Loss

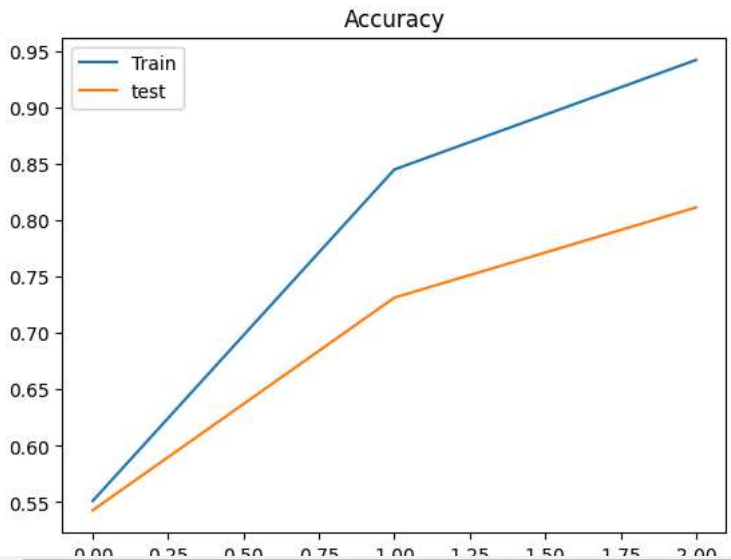
```
import matplotlib.pyplot as plt

plt.title('Loss')
plt.plot(history.history['loss'], label='Train')
plt.plot(history.history['val_loss'], label='test')
plt.legend()
plt.show()
```



Accuracy

```
plt.title('Accuracy')
plt.plot(history.history['accuracy'], label='Train')
plt.plot(history.history['val_accuracy'], label='test')
plt.legend()
plt.show()
```



Confusion Matrix

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

```
[0, 1]
```

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

# confusion_matrix(Y_test.argmax(axis=1),
#                  y_pred.argmax(axis=1))

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

```
24/24 ━━━━━━━━━━━━━━━━━━━━ 4s 162ms/step
```

	0	1
0	298	79
1	61	312

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 ━━━━━━━━━━━━━━━━━━━━ 3s 104ms/step
              precision    recall  f1-score   support

           0       0.83      0.79      0.81       377
           1       0.80      0.84      0.82       373

    accuracy                           0.81       750
   macro avg       0.81      0.81      0.81       750
weighted avg       0.81      0.81      0.81       750
```

Test using new review from Deadpool Wolverine.

```
new_review = ['''I’ve waited for more than 2 years for this cinematic masterpiece to ultimately debut in theaters as the ultimate live-ac
To elucidate this belief in my case, I myself watched this film the first at Reading Cinemas Town Square on Tuesday July 30th, 2024 at 6:
As for the prime antagonist, I wasn’t too agitated and irritably bothered by one of the antagonists being Cassandra Nova, (but that’s les
Overall, this true pure magnum opus is all the more breathtaking to say for the least that is truly, and might just practically even surp
```

```
import numpy as np

seq = tokenizer.texts_to_sequences(new_review)
padded = pad_sequences(seq, maxlen=MAX_SEQUENCE_LENGTH)
pred = model.predict(padded)
labels=pd.get_dummies(df['sentiment']).columns.values
print(pred, labels[np.argmax(pred)])
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 124ms/step
[[0.14448868 0.85551125]] 1
```

Start coding or generate with AI.