```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
import seaborn as sns
import matplotlib.pyplot as plt
import tensorflow as tf
from keras.datasets import imdb
```

```
(X_train, y_train), (X_test, y_test) = imdb.load_data(num_words=10000)
```

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/imdb.npz
17464789/17464789 [==============================] - 1s 0us/step
```

```
data = np.concatenate((X_train, X_test), axis=0)
```

```
label = np.concatenate((y_train, y_test), axis=0)
```

```
X_train.shape
```

```
(25000,)
```

```
X_test.shape
```

```
(25000,)
```

```
y_train.shape
```

```
(25000,)
```

```
y_test.shape
```

```
(25000,)
```

```
print("Review is ",X_train[0])
```

```
Review is  [1, 14, 22, 16, 43, 530, 973, 1622, 1385, 65, 458, 4468, 66, 3941, 4, 173, 36, 256, 5, 25, 100, 43, 838, 112, 50, 670, 2, 9,
```

```
print("Review is ",y_train[0])
```

```
Review is  1
```

```
vocab=imdb.get_word_index()
print(vocab)
```

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/imdb_word_index.json
1641221/1641221 [==============================] - 0s 0us/step
{'fawn': 34701, 'tsukino': 52006, 'nunnery': 52007, 'sonja': 16816, 'vani': 63951, 'woods': 1408, 'spiders': 16115, 'hanging': 2345, 'wo
```

```
y_train
```

```
array([1, 0, 0, ..., 0, 1, 0])
```

```
y_test
```

```
array([0, 1, 1, ..., 0, 0, 0])
```

```
def vectorize(sequences, dimension = 10000):
  results = np.zeros((len(sequences), dimension))
  for i, sequence in enumerate(sequences):
    results[i, np.array(sequence).astype(int)] = 1
  return results
```

```
test_x = data[:10000]
test_y = label[:10000]
train_x = data[10000:]
train_y = label[10000:]
```

+ Code + Text

```
test_x.shape
```

```
(10000,)
```

```
test_y.shape
```

```
(10000,)
```

```
train_x.shape
```

```
(40000,)
```

```
train_y.shape
```

```
(40000,)
```

```
print("Categories:", np.unique(label))
print("Number of unique words:", len(np.unique(np.hstack(data))))
```

```
Categories: [0 1]
Number of unique words: 9998
```

```
length = [len(i) for i in data]
print("Average Review length:", np.mean(length))
print("Standard Deviation:", round(np.std(length)))
```

```
Average Review length: 234.75892
Standard Deviation: 173
```

```
print("Label:", label[0])
print("Label:", label[1])
print(data[0])
```

```
Label: 1
Label: 0
[1, 14, 22, 16, 43, 530, 973, 1622, 1385, 65, 458, 4468, 66, 3941, 4, 173, 36, 256, 5, 25, 100, 43, 838, 112, 50, 670, 2, 9, 35, 480, 28
```

```
index = imdb.get_word_index()
reverse_index = dict([(value, key) for (key, value) in index.items()])
decoded = " ".join( [reverse_index.get(i - 3, "#") for i in data[0]] )
print(decoded)
```

```
# this film was just brilliant casting location scenery story direction everyone's really suited the part they played and you could just
```

```
data = vectorize(data)
label = np.array(label).astype("float32")
labelDF=pd.DataFrame({'label':label})
sns.countplot(x='label', data=labelDF)
```

```
<Axes: xlabel='label', ylabel='count'>
```


```
X_train, X_test, y_train, y_test = train_test_split(data,label, test_size=0.20, random_state=1)
```

```
X_train.shape
```

```
(40000, 10000)
```

```
X_test.shape
```

```
(10000, 10000)
```

```
from keras.utils import to_categorical
from keras import models
from keras import layers

model = models.Sequential()
model.add(layers.Dense(50, activation = "relu", input_shape=(10000, )))
model.add(layers.Dropout(0.3, noise_shape=None, seed=None))
model.add(layers.Dense(50, activation = "relu"))
model.add(layers.Dropout(0.2, noise_shape=None, seed=None))
model.add(layers.Dense(50, activation = "relu"))
model.add(layers.Dense(1, activation = "sigmoid"))
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (None, 50)                500050

 dropout (Dropout)           (None, 50)                0

 dense_1 (Dense)             (None, 50)                2550

 dropout_1 (Dropout)         (None, 50)                0

 dense_2 (Dense)             (None, 50)                2550

 dense_3 (Dense)             (None, 1)                 51

=================================================================
Total params: 505,201
Trainable params: 505,201
Non-trainable params: 0
_________________________________________________________________
```

```
callback = tf.keras.callbacks.EarlyStopping(monitor='loss', patience=3)

model.compile(
optimizer = "adam",
loss = "binary_crossentropy",
metrics = ["accuracy"]
)

results = model.fit(
X_train, y_train,
epochs= 2,
batch_size = 500,
validation_data = (X_test, y_test),
callbacks=[callback]
)

score = model.evaluate(X_test, y_test, batch_size=500)
print('Test loss:', score[0])
print('Test accuracy:', score[1])
```

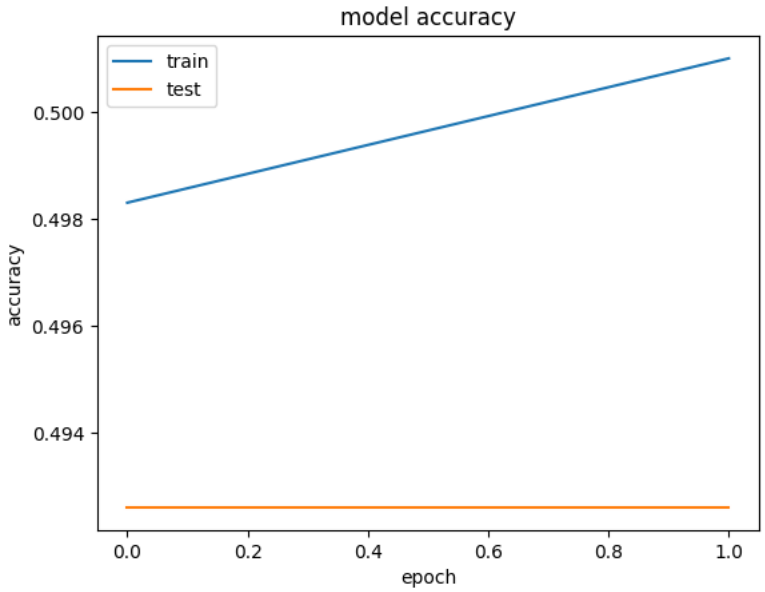
```
Epoch 1/2
80/80 [==============================] - 7s 70ms/step - loss: 0.6932 - accuracy: 0.4983 - val_loss: 0.6937 - val_accuracy: 0.4926
Epoch 2/2
80/80 [==============================] - 6s 78ms/step - loss: 0.6932 - accuracy: 0.5010 - val_loss: 0.6932 - val_accuracy: 0.4926
20/20 [==============================] - 1s 48ms/step - loss: 0.6932 - accuracy: 0.4926
```

```
Test loss: 0.6931878924369812
Test accuracy: 0.4925999939441681
```

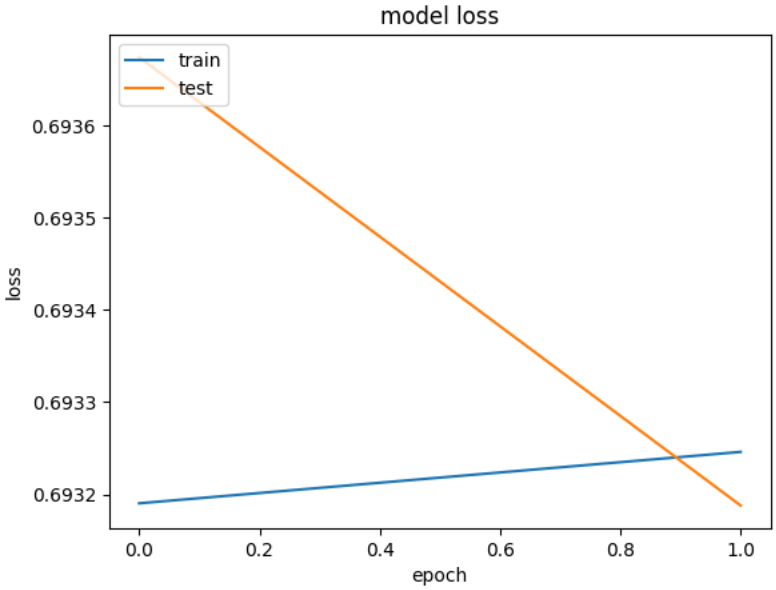
```
print(results.history.keys())
```

```
dict_keys(['loss', 'accuracy', 'val_loss', 'val_accuracy'])
```

```
plt.plot(results.history['accuracy'])
plt.plot(results.history['val_accuracy'])
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.show()
```




```
plt.plot(results.history['loss'])
plt.plot(results.history['val_loss'])
plt.title('model loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.show()
```



✓ 0s completed at 4:08 PM