```
In [1]: import numpy as np
        import pandas as pd
        from sklearn.model_selection import train_test_split
```

```
In [2]: from keras.datasets import imdb
```

```
In [3]: (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/imd
b.npz
17464789/17464789 [==============================] - 0s 0us/step
```

```
In [4]: data = np.concatenate((X_train, X_test), axis=0)
```

```
In [5]: label = np.concatenate((y_train, y_test), axis=0)
```

```
In [6]: X_train.shape
```

```
Out[6]: (25000,)
```

```
In [7]: X_test.shape
```

```
Out[7]: (25000,)
```

```
In [8]: y_train.shape
```

```
Out[8]: (25000,)
```

```
In [9]: y_test.shape
```

```
Out[9]: (25000,)
```

```
In [10]: print("Review is ",X_train[0])
         print("Review is ",y_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, 35, 480, 284, 5, 150, 4, 172, 112,
167, 2, 336, 385, 39, 4, 172, 4536, 1111, 17, 546, 38, 13, 447, 4, 192, 50, 16, 6, 14
7, 2025, 19, 14, 22, 4, 1920, 4613, 469, 4, 22, 71, 87, 12, 16, 43, 530, 38, 76, 15,
13, 1247, 4, 22, 17, 515, 17, 12, 16, 626, 18, 2, 5, 62, 386, 12, 8, 316, 8, 106, 5,
4, 2223, 5244, 16, 480, 66, 3785, 33, 4, 130, 12, 16, 38, 619, 5, 25, 124, 51, 36, 13
5, 48, 25, 1415, 33, 6, 22, 12, 215, 28, 77, 52, 5, 14, 407, 16, 82, 2, 8, 4, 107, 11
7, 5952, 15, 256, 4, 2, 7, 3766, 5, 723, 36, 71, 43, 530, 476, 26, 400, 317, 46, 7,
4, 2, 1029, 13, 104, 88, 4, 381, 15, 297, 98, 32, 2071, 56, 26, 141, 6, 194, 7486, 1
8, 4, 226, 22, 21, 134, 476, 26, 480, 5, 144, 30, 5535, 18, 51, 36, 28, 224, 92, 25,
104, 4, 226, 65, 16, 38, 1334, 88, 12, 16, 283, 5, 16, 4472, 113, 103, 32, 15, 16, 53
45, 19, 178, 32]
Review is  1
```

```
In [11]: vocab=imdb.get_word_index()
```

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/imd
b_word_index.json
1641221/1641221 [==============================] - 0s 0us/step
```

In [12]:
```
print(vocab)
```

In [13]:
```
def vectorize(sequences, dimension = 10000):
    results = np.zeros((len(sequences), dimension))
    for i, sequence in enumerate(sequences):
        results[i, sequence] = 1
    return results
```

In [14]:
```
test_x = data[:10000]
test_y = label[:10000]
train_x = data[10000:]
train_y = label[10000:]
test_x.shape
```

Out[14]:
```
(10000,)
```

In [15]:
```
print("Categories:", np.unique(label))
print("Number of unique words:", len(np.unique(np.hstack(data))))
```

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

In [16]:
```
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
```

In [17]:
```
print("Label:", label[0])
Label: 1
print("Label:", label[1])
Label: 0
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, 284, 5, 150, 4, 172, 112, 167, 2, 3
36, 385, 39, 4, 172, 4536, 1111, 17, 546, 38, 13, 447, 4, 192, 50, 16, 6, 147, 2025,
19, 14, 22, 4, 1920, 4613, 469, 4, 22, 71, 87, 12, 16, 43, 530, 38, 76, 15, 13, 1247,
4, 22, 17, 515, 17, 12, 16, 626, 18, 2, 5, 62, 386, 12, 8, 316, 8, 106, 5, 4, 2223, 5
244, 16, 480, 66, 3785, 33, 4, 130, 12, 16, 38, 619, 5, 25, 124, 51, 36, 135, 48, 25,
1415, 33, 6, 22, 12, 215, 28, 77, 52, 5, 14, 407, 16, 82, 2, 8, 4, 107, 117, 5952, 1
5, 256, 4, 2, 7, 3766, 5, 723, 36, 71, 43, 530, 476, 26, 400, 317, 46, 7, 4, 2, 1029,
13, 104, 88, 4, 381, 15, 297, 98, 32, 2071, 56, 26, 141, 6, 194, 7486, 18, 4, 226, 2
2, 21, 134, 476, 26, 480, 5, 144, 30, 5535, 18, 51, 36, 28, 224, 92, 25, 104, 4, 226,
65, 16, 38, 1334, 88, 12, 16, 283, 5, 16, 4472, 113, 103, 32, 15, 16, 5345, 19, 178,
32]
```

In [18]:
```
index = imdb.get_word_index()
```

In [19]:
```
reverse_index = dict([(value, key) for (key, value) in index.items()])
```

In [20]:
```
decoded = " ".join( [reverse_index.get(i - 3, "#") for i in data[0]] )
```

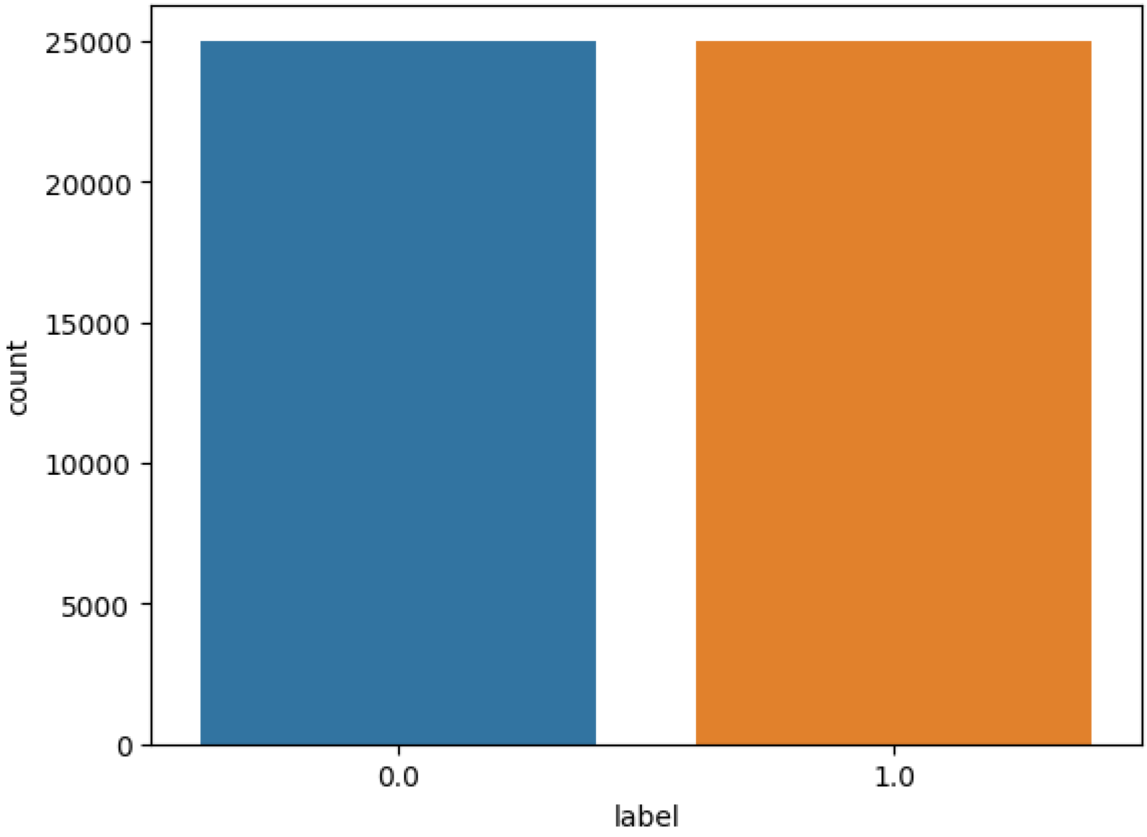
In [21]:
```
print(decoded)
```

```
# this film was just brilliant casting location scenery story direction everyone's re
ally suited the part they played and you could just imagine being there robert # is a
n amazing actor and now the same being director # father came from the same scottish
island as myself so i loved the fact there was a real connection with this film the w
itty remarks throughout the film were great it was just brilliant so much that i boug
ht the film as soon as it was released for # and would recommend it to everyone to wa
tch and the fly fishing was amazing really cried at the end it was so sad and you kno
w what they say if you cry at a film it must have been good and this definitely was a
lso # to the two little boy's that played the # of norman and paul they were just bri
lliant children are often left out of the # list i think because the stars that play
them all grown up are such a big profile for the whole film but these children are am
azing and should be praised for what they have done don't you think the whole story w
as so lovely because it was true and was someone's life after all that was shared wit
h us all
```

In [22]:
```
import seaborn as sns
```

In [23]:
```
data = vectorize(data)
label = np.array(label).astype("float32")
labelDF=pd.DataFrame({'label':label})
sns.countplot(x='label', data=labelDF)
```

Out[23]:
```
<Axes: xlabel='label', ylabel='count'>
```



In [25]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(data,label, test_size=0.20, randor
X_train.shape
X_test.shape
```

Out[25]:
```
(10000, 10000)
```

In [26]:
```
from keras.utils import to_categorical
from keras import models
from keras import layers
model = models.Sequential()
```

In [27]:
```
model.add(layers.Dense(50, activation = "relu", input_shape=(10000, )))
```

In [28]:
```
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"))
```

In [29]:
```
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
_________________________________________________________________
```

In [30]:
```
import tensorflow as tf
callback = tf.keras.callbacks.EarlyStopping(monitor='loss', patience=3)
```

In [31]:
```
model.compile(
optimizer = "adam",
loss = "binary_crossentropy",
metrics = ["accuracy"]
)
```

In [32]:
```
from sklearn.model_selection import train_test_split
```

In [33]:
```
results = model.fit(
X_train, y_train,
epochs= 2,
batch_size = 500,
validation_data = (X_test, y_test),
callbacks=[callback]
)
```

```
Epoch 1/2
80/80 [==============================] - 7s 74ms/step - loss: 0.3951 - accuracy: 0.82
63 - val_loss: 0.2631 - val_accuracy: 0.8935
Epoch 2/2
80/80 [==============================] - 4s 51ms/step - loss: 0.2172 - accuracy: 0.91
66 - val_loss: 0.2596 - val_accuracy: 0.8973
```

In [34]:

```
print(np.mean(results.history["val_accuracy"]))
```

```
0.8953999876976013
```

In [35]:

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

```
20/20 [==============================] - 1s 53ms/step - loss: 0.2596 - accuracy: 0.89
73
Test loss: 0.25958889722824097
Test accuracy: 0.8973000049591064
```

In [36]:

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

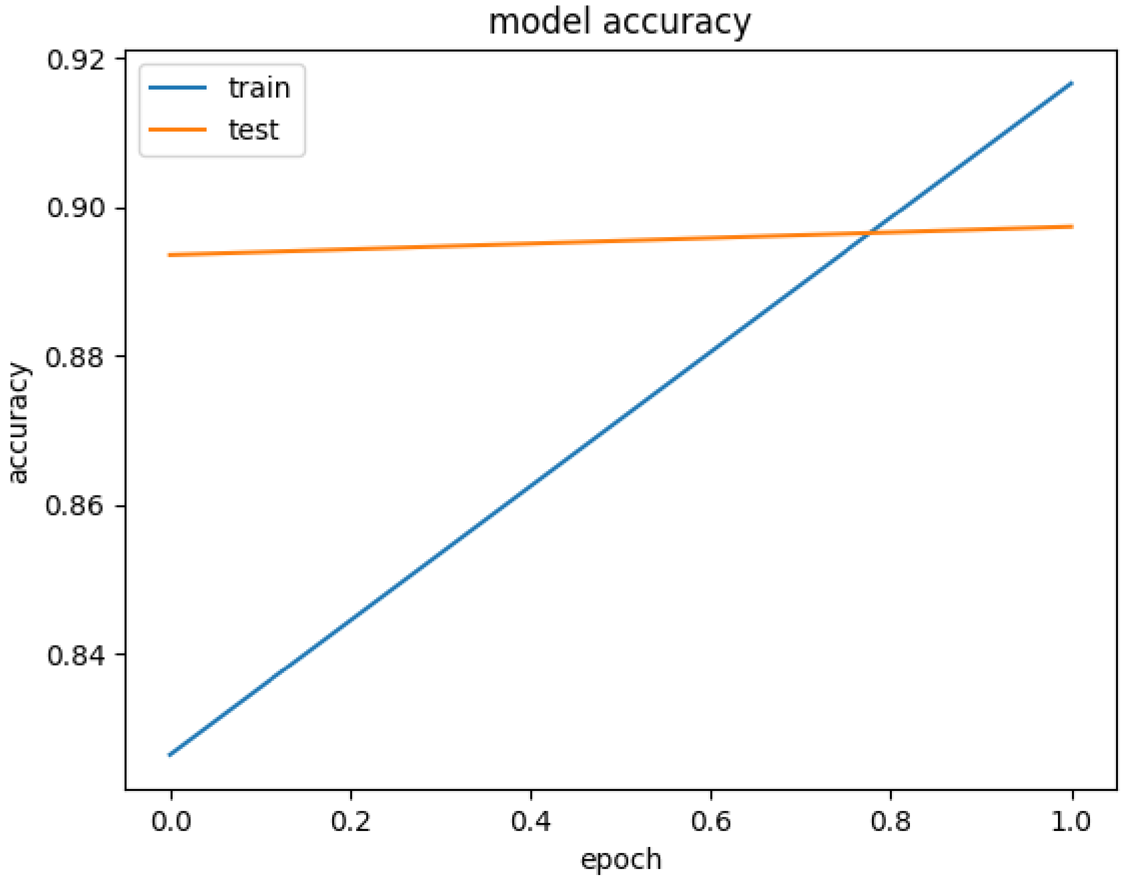
```
dict_keys(['loss', 'accuracy', 'val_loss', 'val_accuracy'])
```

In [38]:

```
import matplotlib.pyplot as plt
```

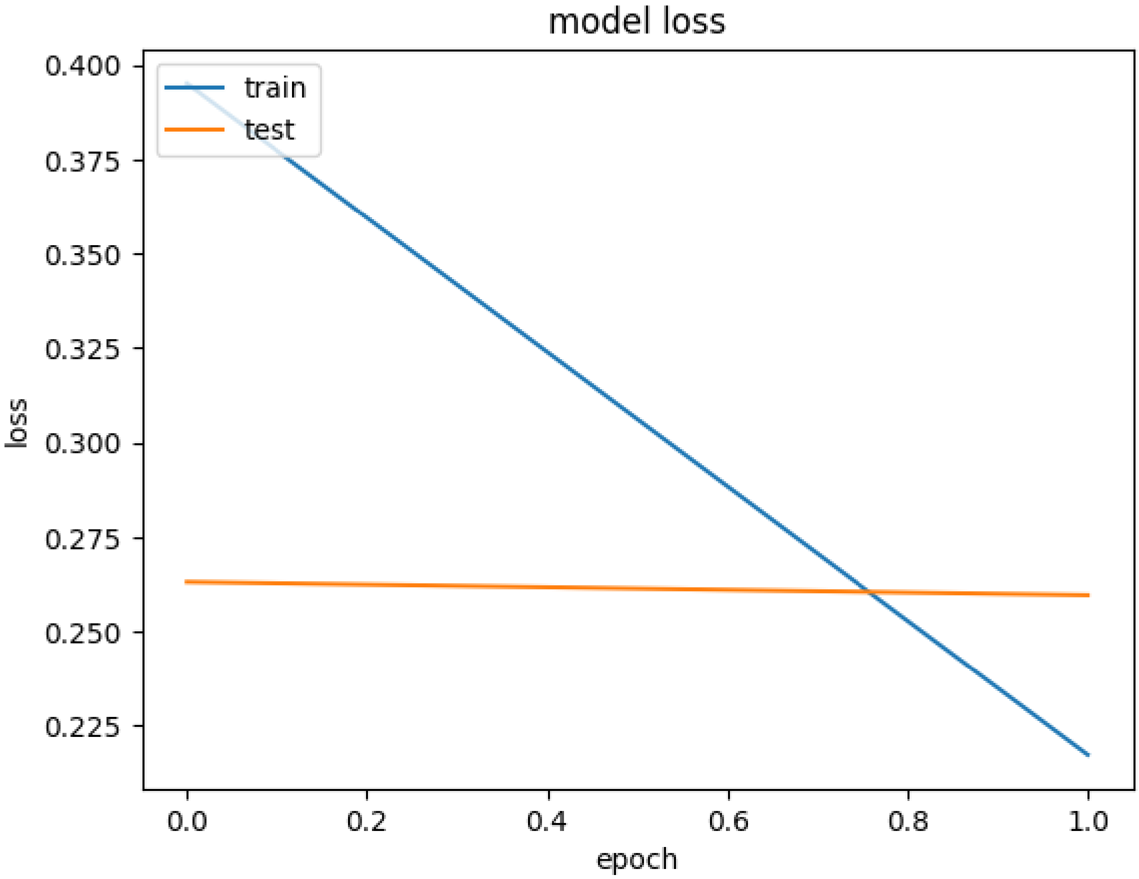
In [39]:

```
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()
```



In [40]:

```
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()
```


model loss
train
test
loss
epoch
0.400
0.375
0.350
0.325
0.300
0.275
0.250
0.225
0.0
0.2
0.4
0.6
0.8
1.0