In [ ]:
```
#Name :- Omkar Balwade
#Roll N.o :- 4107
#Div :- A
#Practical N.o 2(A):- Text Classification with Neural Networks: IMDb Movie Review Sentiment Analysis
```

In [1]:
```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from keras.datasets import imdb
from keras.utils import to_categorical
from keras import models
from keras import layers
import tensorflow as tf
import seaborn as sns
import matplotlib.pyplot as plt
```

```
WARNING:tensorflow:From C:\Users\Omkar\AppData\Local\anaconda3\Lib\site-packages\keras\src\losses.py:2976: The name tf.losses.sparse_softmax_cross_e
```

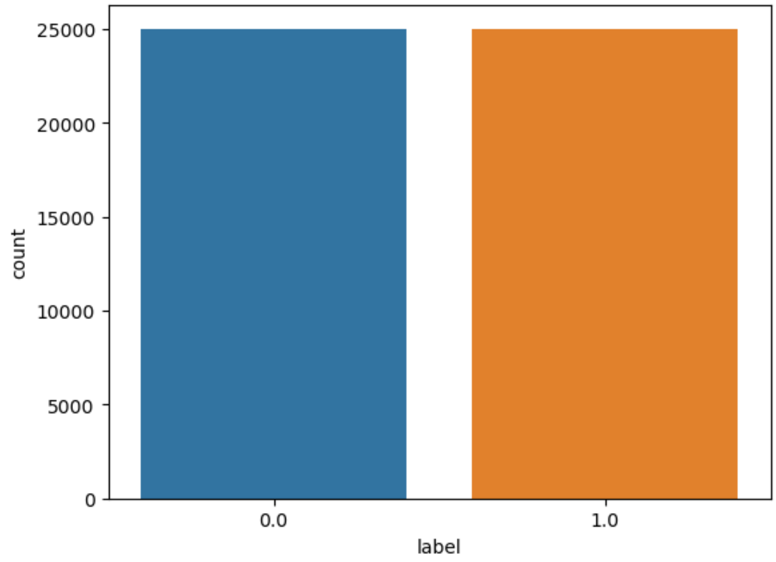
In [2]:
```
# loading imdb data with most frequent 10000 words
(X_train, y_train), (X_test, y_test) = imdb.load_data(num_words=10000)
```

In [3]:
```
# consolidating data for EDA
data = np.concatenate((X_train, X_test), axis=0)
label = np.concatenate((y_train, y_test), axis=0)
```

In [4]:
```
# sequences is name of method the review less than 10000 we perform padding overthere
def vectorize(sequences, dimension=10000):
    results = np.zeros((len(sequences), dimension))
    for i, sequence in enumerate(sequences):
        results[i, sequence] = 1
    return results
```

In [5]:
```
# Vectorization is the process of converting textual data into numerical vectors
data = vectorize(data)
label = np.array(label).astype("float32")
labelDF = pd.DataFrame({'label': label})
sns.countplot(x='label', data=labelDF)
```

Out [5]:
```
<Axes: xlabel='label', ylabel='count'>
```



In [6]:
```
# Creating train and test data set
X_train, X_test, y_train, y_test = train_test_split(data, label, test_size=0.20, random_state=1)
```

In [7]:
```
# Let's create sequential model
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"))
```

```
WARNING:tensorflow:From C:\Users\Omkar\AppData\Local\anaconda3\Lib\site-packages\keras\src\backend.py:873: The name tf.get_default_graph is deprecat
```

In [8]:

```python
# For early stopping
callback = tf.keras.callbacks.EarlyStopping(monitor='loss', patience=3)
```

In [9]:

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

```
WARNING:tensorflow:From C:\Users\Omkar\AppData\Local\anaconda3\Lib\site-packages\keras\src\optimizers\__init__.py:309: The name tf.train.Optimizer i
```

In [10]:

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

```
Epoch 1/2
WARNING:tensorflow:From C:\Users\Omkar\AppData\Local\anaconda3\Lib\site-packages\keras\src\utils\tf_utils.py:492: The name tf.ragged.RaggedTensorVal

WARNING:tensorflow:From C:\Users\Omkar\AppData\Local\anaconda3\Lib\site-packages\keras\src\engine\base_layer_utils.py:384: The name tf.executing_eag

80/80 [==============================] - 12s 123ms/step - loss: 0.4050 - accuracy: 0.8198 - val_loss: 0.2598 - val_accuracy: 0.8959
Epoch 2/2
80/80 [==============================] - 4s 54ms/step - loss: 0.2153 - accuracy: 0.9180 - val_loss: 0.2533 - val_accuracy: 0.8984
```

In [11]:

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

```
0.8971500098705292
```

In [12]:

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

```
20/20 [==============================] - 1s 27ms/step - loss: 0.2533 - accuracy: 0.8984
Test loss: 0.253330260515213
Test accuracy: 0.8984000086784363
```

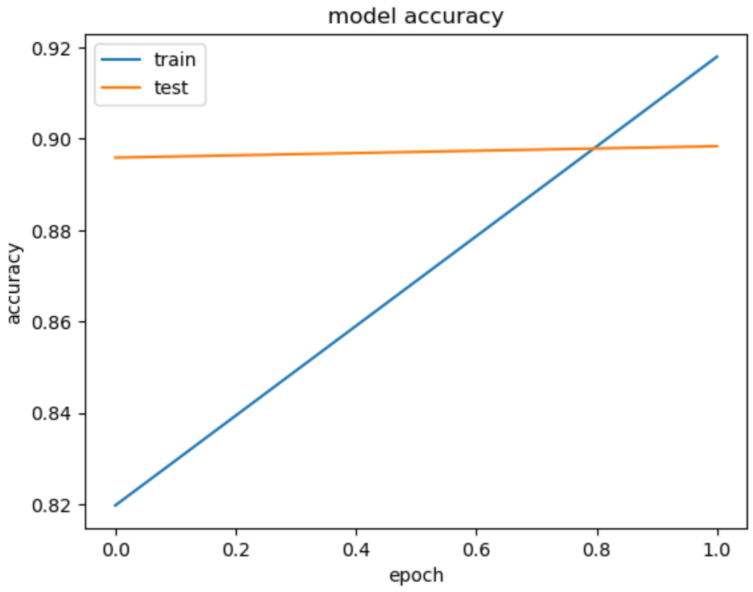
In [13]:

```python
# Plot training history of the model
print(results.history.keys())
```

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

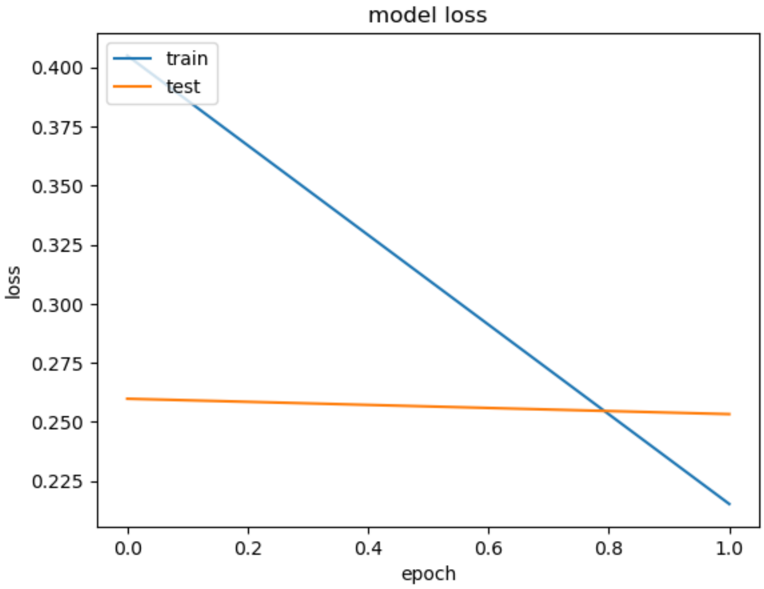
In [14]:

```python
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 [15]:

```python
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()
```



In [ ]:

In [ ]: