In [1]:

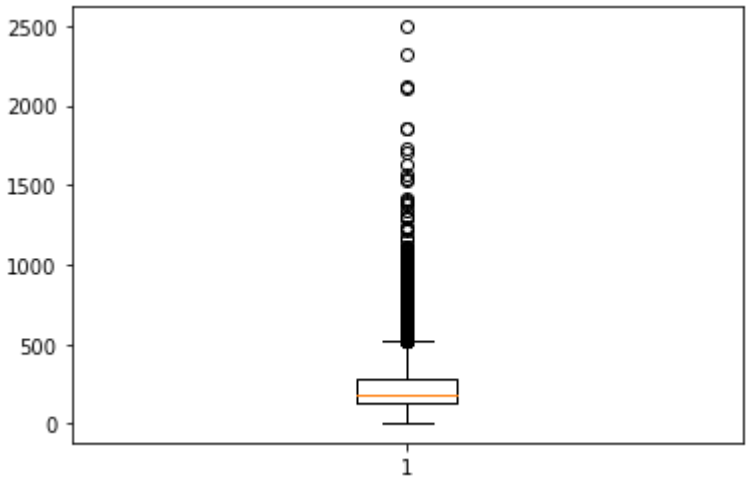
```
# Vishakha Dhonde
# COBB055
# Importing Necesarry Packages
import numpy as np
from numpy.ma.core import argmax
import pandas as pd
from matplotlib import cm
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
#import os
import time
from sklearn.metrics import confusion_matrix, accuracy_score, auc
from keras.preprocessing import sequence
from keras.models import Sequential
from keras.layers import Dense, Dropout, Activation
from keras.layers import Embedding
from keras.layers import Conv1D, GlobalMaxPooling1D
from keras.callbacks import EarlyStopping
from keras import models
from keras import layers
from keras.datasets import imdb
```

In [2]:

```
# Loading the dataset
(X_train, y_train), (X_test, y_test) = imdb.load_data()
X = np.concatenate((X_train, X_test), axis=0)
y = np.concatenate((y_train, y_test), axis=0)

# Exploring the Data
print("Training data: ")
print(X.shape)
print(y.shape)
print("Classes: ")
print(np.unique(y))
print("Number of words: ")
print(len(np.unique(np.hstack(X))))
print("Review length: ")
result = [len(x) for x in X]
print("Mean %.2f words (%f)" % (np.mean(result), np.std(result))) # Ploting the review
plt.boxplot(result)
plt.show()
```

```
<__array_function__ internals>:5: VisibleDeprecationWarning: Creating an ndarray from ra
gged nested sequences (which is a list-or-tuple of lists-or-tuples-or ndarrays with diff
erent lengths or shapes) is deprecated. If you meant to do this, you must specify 'dtype
=object' when creating the ndarray.
C:\Users\i-net computer\anaconda3\envs\tensorflow\lib\site-packages\tensorflow\python\ke
ras\datasets\imdb.py:159: VisibleDeprecationWarning: Creating an ndarray from ragged nes
ted sequences (which is a list-or-tuple of lists-or-tuples-or ndarrays with different le
ngths or shapes) is deprecated. If you meant to do this, you must specify 'dtype=object'
when creating the ndarray.
  x_train, y_train = np.array(xs[:idx]), np.array(labels[:idx])
C:\Users\i-net computer\anaconda3\envs\tensorflow\lib\site-packages\tensorflow\python\ke
ras\datasets\imdb.py:160: VisibleDeprecationWarning: Creating an ndarray from ragged nes
ted sequences (which is a list-or-tuple of lists-or-tuples-or ndarrays with different le
ngths or shapes) is deprecated. If you meant to do this, you must specify 'dtype=object'
```

```
when creating the ndarray.
  x_test, y_test = np.array(xs[idx:]), np.array(labels[idx:])
Training data:
(50000,)
(50000,)
Classes:
[0 1]
Number of words:
88585
Review length:
Mean 234.76 words (172.911495)
```



In [3]:

```
def vectorize_sequences(sequences, dimension=5000): # Function for vectorising data
    results = np.zeros((len(sequences), dimension)) # Creating an all-zero matrix of sh
    for i, sequence in enumerate(sequences):
        results[i, sequence] = 1.  # Set specific indices of results[i] to 1s
    return results
```

In [4]:

```
# Creating Training and Testing Sets and Preprocessing them
(train_data, train_labels), (test_data, test_labels) = imdb.load_data(num_words=5000)
# Our vectorized training data
x_train = vectorize_sequences(train_data)
# Our vectorized test data
x_test = vectorize_sequences(test_data)
# Our vectorized labels one-hot encoder
y_train = np.asarray(train_labels).astype('float32')
y_test = np.asarray(test_labels).astype('float32')
```

```
<__array_function__ internals>:5: VisibleDeprecationWarning: Creating an ndarray from ra
gged nested sequences (which is a list-or-tuple of lists-or-tuples-or ndarrays with diff
erent lengths or shapes) is deprecated. If you meant to do this, you must specify 'dtype
=object' when creating the ndarray.
C:\Users\i-net computer\anaconda3\envs\tensorflow\lib\site-packages\tensorflow\python\ke
ras\datasets\imdb.py:159: VisibleDeprecationWarning: Creating an ndarray from ragged nes
ted sequences (which is a list-or-tuple of lists-or-tuples-or ndarrays with different le
ngths or shapes) is deprecated. If you meant to do this, you must specify 'dtype=object'
when creating the ndarray.
  x_train, y_train = np.array(xs[:idx]), np.array(labels[:idx])
C:\Users\i-net computer\anaconda3\envs\tensorflow\lib\site-packages\tensorflow\python\ke
ras\datasets\imdb.py:160: VisibleDeprecationWarning: Creating an ndarray from ragged nes
ted sequences (which is a list-or-tuple of lists-or-tuples-or ndarrays with different le
ngths or shapes) is deprecated. If you meant to do this, you must specify 'dtype=object'
```

when creating the ndarray.
  x_test, y_test = np.array(xs[idx:]), np.array(labels[idx:])

In [5]:
```
# Creating the DNN Model
model = models.Sequential()
model.add(layers.Dense(32, activation='relu', input_shape=(5000,)))
model.add(layers.Dense(32, activation='relu',))
model.add(layers.Dense(1, activation='sigmoid'))
```

In [6]:
```
#Set validation set aside
x_val = x_train[:10000]
partial_x_train = x_train[10000:]
y_val = y_train[:10000]
partial_y_train = y_train[10000:]
```

In [7]:
```
# Compiling Model
model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['acc'])
start_time_m1 = time.time()
history = model.fit(partial_x_train,
                    partial_y_train,
                    epochs=20,
                    batch_size=512,
                    validation_data=(x_val, y_val))
total_time_m1 = time.time() - start_time_m1
print("The Dense Convolutional Neural Network 1 layer took %.4f seconds to train." % (t
```

```
Epoch 1/20
30/30 [==============================] - 30s 1s/step - loss: 0.5376 - acc: 0.7584 - val_
loss: 0.3651 - val_acc: 0.8600
Epoch 2/20
30/30 [==============================] - 2s 53ms/step - loss: 0.2836 - acc: 0.8962 - val
_loss: 0.2893 - val_acc: 0.8866
Epoch 3/20
30/30 [==============================] - 1s 29ms/step - loss: 0.2081 - acc: 0.9231 - val
_loss: 0.2872 - val_acc: 0.8849
Epoch 4/20
30/30 [==============================] - 2s 72ms/step - loss: 0.1679 - acc: 0.9410 - val
_loss: 0.3022 - val_acc: 0.8806
Epoch 5/20
30/30 [==============================] - 1s 39ms/step - loss: 0.1379 - acc: 0.9525 - val
_loss: 0.3252 - val_acc: 0.8768
Epoch 6/20
30/30 [==============================] - 1s 25ms/step - loss: 0.1132 - acc: 0.9637 - val
_loss: 0.3527 - val_acc: 0.8738
Epoch 7/20
30/30 [==============================] - 1s 27ms/step - loss: 0.0902 - acc: 0.9727 - val
_loss: 0.3859 - val_acc: 0.8693
Epoch 8/20
30/30 [==============================] - 2s 51ms/step - loss: 0.0689 - acc: 0.9818 - val
_loss: 0.4242 - val_acc: 0.8672
Epoch 9/20
30/30 [==============================] - 1s 27ms/step - loss: 0.0512 - acc: 0.9891 - val
_loss: 0.4713 - val_acc: 0.8616
Epoch 10/20
30/30 [==============================] - 1s 22ms/step - loss: 0.0369 - acc: 0.9945 - val
_loss: 0.5170 - val_acc: 0.8586
```

```
Epoch 11/20
30/30 [==============================] - 1s 27ms/step - loss: 0.0265 - acc: 0.9969 - val
_loss: 0.5468 - val_acc: 0.8620
Epoch 12/20
30/30 [==============================] - 1s 24ms/step - loss: 0.0174 - acc: 0.9989 - val
_loss: 0.5887 - val_acc: 0.8605
Epoch 13/20
30/30 [==============================] - 1s 22ms/step - loss: 0.0113 - acc: 0.9994 - val
_loss: 0.6214 - val_acc: 0.8615
Epoch 14/20
30/30 [==============================] - 1s 25ms/step - loss: 0.0078 - acc: 0.9998 - val
_loss: 0.6513 - val_acc: 0.8608
Epoch 15/20
30/30 [==============================] - 1s 30ms/step - loss: 0.0059 - acc: 0.9999 - val
_loss: 0.6795 - val_acc: 0.8611
Epoch 16/20
30/30 [==============================] - 1s 22ms/step - loss: 0.0044 - acc: 0.9999 - val
_loss: 0.7022 - val_acc: 0.8606
Epoch 17/20
30/30 [==============================] - 1s 40ms/step - loss: 0.0035 - acc: 0.9999 - val
_loss: 0.7238 - val_acc: 0.8606
Epoch 18/20
30/30 [==============================] - 1s 25ms/step - loss: 0.0029 - acc: 0.9999 - val
_loss: 0.7428 - val_acc: 0.8603
Epoch 19/20
30/30 [==============================] - 1s 23ms/step - loss: 0.0024 - acc: 1.0000 - val
_loss: 0.7619 - val_acc: 0.8601
Epoch 20/20
30/30 [==============================] - 1s 22ms/step - loss: 0.0020 - acc: 1.0000 - val
_loss: 0.7772 - val_acc: 0.8595
The Dense Convolutional Neural Network 1 layer took 100.5009 seconds to train.
```

In [8]:

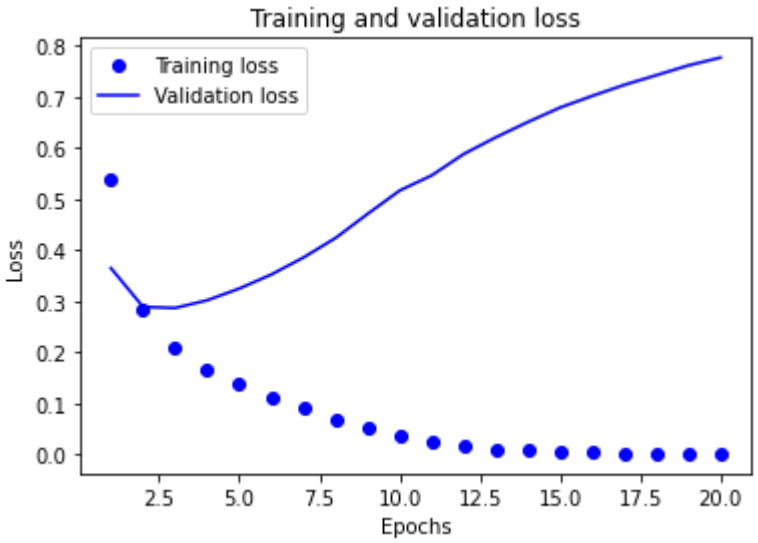
```
history_dict = history.history
history_dict.keys()
```

Out[8]:

```
dict_keys(['loss', 'acc', 'val_loss', 'val_acc'])
```

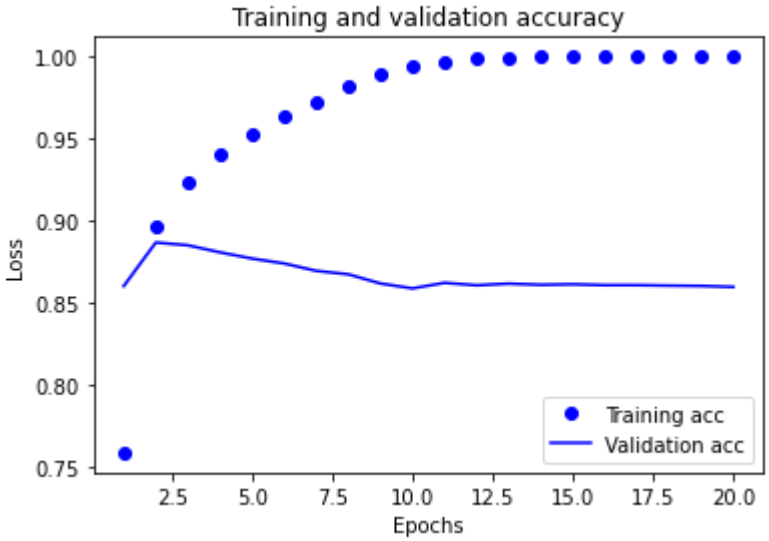
In [9]:

```
acc = history.history['acc']
val_acc = history.history['val_acc']
loss = history.history['loss']
val_loss = history.history['val_loss']
epochs = range(1, len(acc) + 1)

# Plotting model loss
plt.plot(epochs, loss, 'bo', label='Training loss') # "bo" is for "blue dot"
plt.plot(epochs, val_loss, 'b', label='Validation loss') # b is for "solid blue line"
plt.title('Training and validation loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()
plt.show()
```



In [10]:

```
plt.clf()   # clear figure
acc_values = history_dict['acc']
val_acc_values = history_dict['val_acc']

# Plotting model accuracy
plt.plot(epochs, acc, 'bo', label='Training acc')
plt.plot(epochs, val_acc, 'b', label='Validation acc')
plt.title('Training and validation accuracy')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()
plt.show()
```



In [11]:

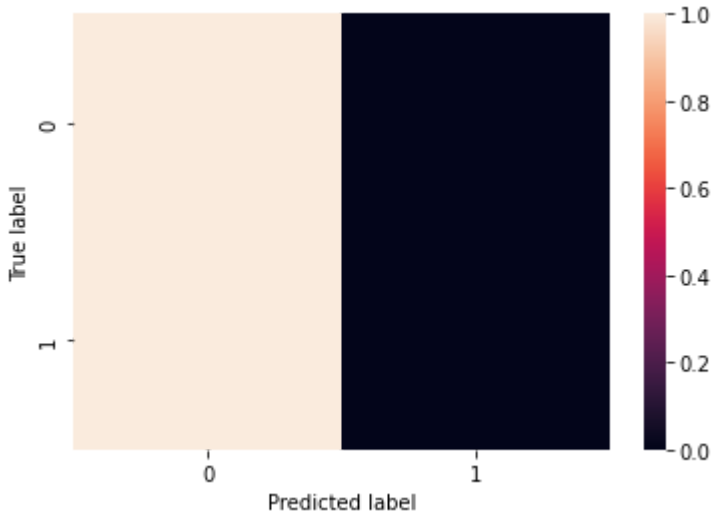
```
# Model Summary
print(model.summary())

# Predictions
pred = model.predict(x_test)
classes_x=np.argmax(pred,axis=1)
accuracy_score(y_test,classes_x)

#Confusion Matrix
```

```
conf_mat = confusion_matrix(y_test, classes_x)
print(conf_mat)

conf_mat_normalized = conf_mat.astype('float') / conf_mat.sum(axis=1)[:, np.newaxis]
sns.heatmap(conf_mat_normalized)
plt.ylabel('True label')
plt.xlabel('Predicted label')
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense (Dense)                (None, 32)                160032
_________________________________________________________________
dense_1 (Dense)              (None, 32)                1056
_________________________________________________________________
dense_2 (Dense)              (None, 1)                 33
=================================================================
Total params: 161,121
Trainable params: 161,121
Non-trainable params: 0
_________________________________________________________________
None
[[12500     0]
 [12500     0]]
```

Out[11]:

```
Text(0.5, 15.0, 'Predicted label')
```



In [12]:

```
#Dense with Two Layer
model2 = models.Sequential()
model2.add(layers.Dense(32, activation='relu', input_shape=(5000,)))
model2.add(layers.Dense(32, activation='relu'))
model2.add(layers.Dense(32, activation='relu'))
model2.add(layers.Dense(1, activation='sigmoid'))
```

In [17]:

```
# Compiling Model
model2.compile(optimizer='adam', loss='binary_crossentropy', metrics=['acc'])
start_time_m2 = time.time()
history= model2.fit(partial_x_train,
                    partial_y_train,
                    epochs=20,
                    batch_size=512,
```

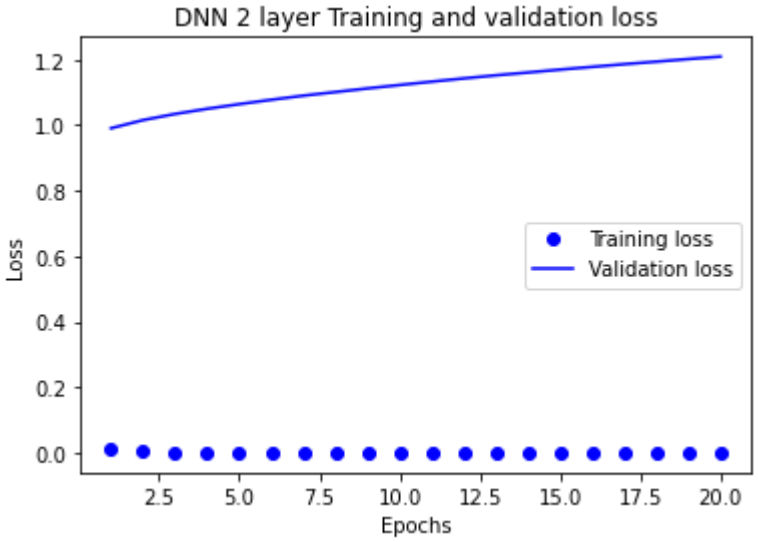
```
                    validation_data=(x_val, y_val))
total_time_m2 = time.time() - start_time_m2
print("The Dense Convolutional Neural Network 2 layers took %.4f seconds to train." % (
```

```
Epoch 1/20
30/30 [==============================] - 8s 276ms/step - loss: 0.0124 - acc: 0.9963 - va
l_loss: 0.9904 - val_acc: 0.8529
Epoch 2/20
30/30 [==============================] - 1s 42ms/step - loss: 0.0028 - acc: 0.9996 - val
_loss: 1.0154 - val_acc: 0.8521
Epoch 3/20
30/30 [==============================] - 1s 31ms/step - loss: 9.8876e-04 - acc: 1.0000 -
val_loss: 1.0341 - val_acc: 0.8521
Epoch 4/20
30/30 [==============================] - 1s 39ms/step - loss: 6.3576e-04 - acc: 1.0000 -
val_loss: 1.0499 - val_acc: 0.8519
Epoch 5/20
30/30 [==============================] - 1s 26ms/step - loss: 4.9221e-04 - acc: 1.0000 -
val_loss: 1.0641 - val_acc: 0.8515
Epoch 6/20
30/30 [==============================] - 1s 23ms/step - loss: 4.0827e-04 - acc: 1.0000 -
val_loss: 1.0774 - val_acc: 0.8515
Epoch 7/20
30/30 [==============================] - 1s 31ms/step - loss: 3.4753e-04 - acc: 1.0000 -
val_loss: 1.0900 - val_acc: 0.8525
Epoch 8/20
30/30 [==============================] - 1s 23ms/step - loss: 3.0112e-04 - acc: 1.0000 -
val_loss: 1.1010 - val_acc: 0.8525
Epoch 9/20
30/30 [==============================] - 1s 23ms/step - loss: 2.6444e-04 - acc: 1.0000 -
val_loss: 1.1121 - val_acc: 0.8530
Epoch 10/20
30/30 [==============================] - 1s 23ms/step - loss: 2.3421e-04 - acc: 1.0000 -
val_loss: 1.1225 - val_acc: 0.8530
Epoch 11/20
30/30 [==============================] - 1s 24ms/step - loss: 2.0866e-04 - acc: 1.0000 -
val_loss: 1.1327 - val_acc: 0.8530
Epoch 12/20
30/30 [==============================] - 1s 25ms/step - loss: 1.8680e-04 - acc: 1.0000 -
val_loss: 1.1427 - val_acc: 0.8528
Epoch 13/20
30/30 [==============================] - 1s 26ms/step - loss: 1.6813e-04 - acc: 1.0000 -
val_loss: 1.1521 - val_acc: 0.8526
Epoch 14/20
30/30 [==============================] - 1s 26ms/step - loss: 1.5275e-04 - acc: 1.0000 -
val_loss: 1.1606 - val_acc: 0.8526
Epoch 15/20
30/30 [==============================] - 1s 23ms/step - loss: 1.3893e-04 - acc: 1.0000 -
val_loss: 1.1695 - val_acc: 0.8531
Epoch 16/20
30/30 [==============================] - 1s 22ms/step - loss: 1.2706e-04 - acc: 1.0000 -
val_loss: 1.1778 - val_acc: 0.8533
Epoch 17/20
30/30 [==============================] - 1s 28ms/step - loss: 1.1684e-04 - acc: 1.0000 -
val_loss: 1.1861 - val_acc: 0.8531
Epoch 18/20
30/30 [==============================] - 1s 33ms/step - loss: 1.0736e-04 - acc: 1.0000 -
val_loss: 1.1937 - val_acc: 0.8529
Epoch 19/20
30/30 [==============================] - 1s 28ms/step - loss: 9.9063e-05 - acc: 1.0000 -
```

```
val_loss: 1.2017 - val_acc: 0.8531
Epoch 20/20
30/30 [==============================] - 1s 30ms/step - loss: 9.1767e-05 - acc: 1.0000 -
val_loss: 1.2092 - val_acc: 0.8531
The Dense Convolutional Neural Network 2 layers took 42.0632 seconds to train.
```

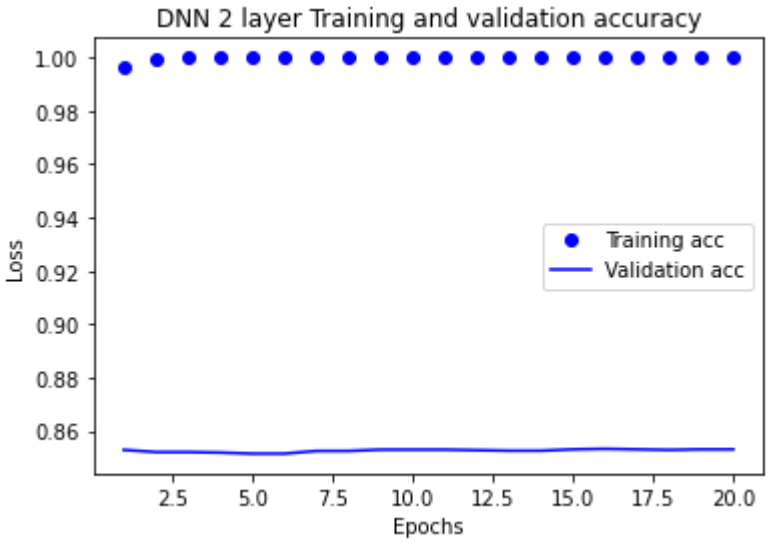
In [18]:

```
acc = history.history['acc']
val_acc = history.history['val_acc']
loss = history.history['loss']
val_loss = history.history['val_loss']
epochs = range(1, len(acc) + 1)

# Plotting Loss
plt.plot(epochs, loss, 'bo', label='Training loss') # "bo" is for "blue dot"
plt.plot(epochs, val_loss, 'b', label='Validation loss') # b is for "solid blue line"
plt.title('DNN 2 layer Training and validation loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()
plt.show()
```



In [19]:

```
plt.clf()   # clear figure
acc_values = history_dict['acc']
val_acc_values = history_dict['val_acc']
# Plotting Accuracy
plt.plot(epochs, acc, 'bo', label='Training acc')
plt.plot(epochs, val_acc, 'b', label='Validation acc')
plt.title('DNN 2 layer Training and validation accuracy')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()
plt.show()
```



In [20]:

```
print(model2.summary())
# Predictions
pred = model2.predict(x_test)
classes_x=np.argmax(pred,axis=-1)
accuracy_score(y_test,classes_x)
```

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_3 (Dense)              (None, 32)                160032
_________________________________________________________________
dense_4 (Dense)              (None, 32)                1056
_________________________________________________________________
dense_5 (Dense)              (None, 32)                1056
_________________________________________________________________
dense_6 (Dense)              (None, 1)                 33
=================================================================
Total params: 162,177
Trainable params: 162,177
Non-trainable params: 0
_________________________________________________________________
None
```

Out[20]:

```
0.5
```

In [ ]: