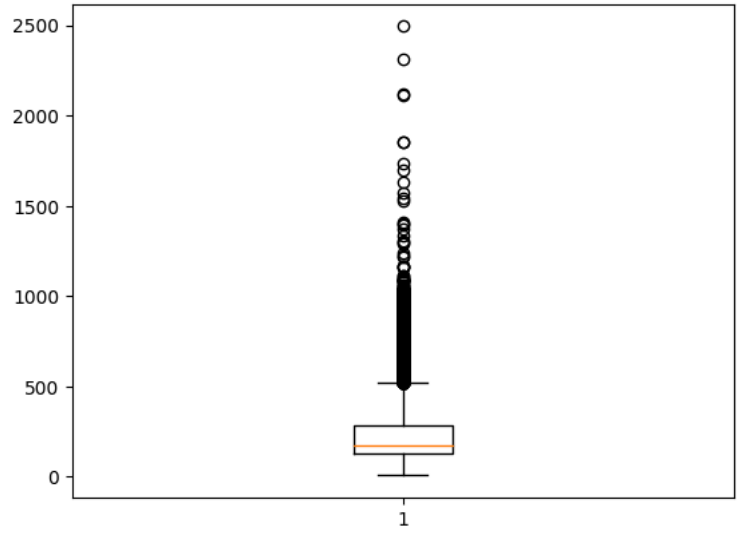
```
#Name: Saloni Satappa Bailkar
#Div: A Roll No. COBA013
#Sub: DL_Lab_02

# 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
```

```
# 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 length
plt.boxplot(result)
plt.show()
```

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/imdb.npz
17464789/17464789 [==============================] - 0s 0us/step
Training data:
(50000,)
(50000,)
Classes:
[0 1]
Number of words:
88585
Review length:
Mean 234.76 words (172.911495)
```




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

```
# 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')
```

```
# 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'))
```

```
#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:]
```

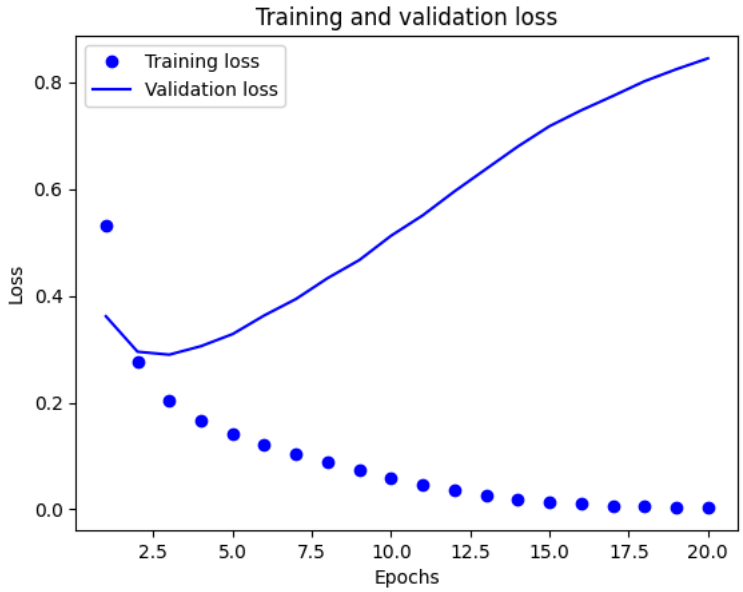
```
# Compiling Model
model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['acc'])
start_time_m1 = time.time()
history = model.fit(partial_x_train,
                    partial_y_train,
                    epochs=20,
                    batch_size=512,
                    validation_data=(x_val, y_val))
total_time_m1 = time.time() - start_time_m1
print("The Dense Convolutional Neural Network 1 layer took %.4f seconds to train." % (total_time_m1))
```

```
Epoch 1/20
30/30 [==============================] - 2s 23ms/step - loss: 0.5304 - acc: 0.7687 - val_loss: 0.3616 - val_acc: 0.8604
Epoch 2/20
30/30 [==============================] - 0s 12ms/step - loss: 0.2769 - acc: 0.8957 - val_loss: 0.2952 - val_acc: 0.8814
Epoch 3/20
30/30 [==============================] - 0s 12ms/step - loss: 0.2033 - acc: 0.9267 - val_loss: 0.2897 - val_acc: 0.8827
Epoch 4/20
30/30 [==============================] - 0s 13ms/step - loss: 0.1666 - acc: 0.9401 - val_loss: 0.3056 - val_acc: 0.8790
Epoch 5/20
30/30 [==============================] - 0s 12ms/step - loss: 0.1401 - acc: 0.9521 - val_loss: 0.3283 - val_acc: 0.8745
Epoch 6/20
30/30 [==============================] - 0s 12ms/step - loss: 0.1205 - acc: 0.9595 - val_loss: 0.3632 - val_acc: 0.8693
Epoch 7/20
30/30 [==============================] - 0s 12ms/step - loss: 0.1025 - acc: 0.9669 - val_loss: 0.3940 - val_acc: 0.8688
Epoch 8/20
30/30 [==============================] - 0s 12ms/step - loss: 0.0877 - acc: 0.9710 - val_loss: 0.4331 - val_acc: 0.8658
Epoch 9/20
30/30 [==============================] - 1s 18ms/step - loss: 0.0737 - acc: 0.9781 - val_loss: 0.4670 - val_acc: 0.8613
Epoch 10/20
30/30 [==============================] - 1s 26ms/step - loss: 0.0589 - acc: 0.9841 - val_loss: 0.5124 - val_acc: 0.8597
Epoch 11/20
30/30 [==============================] - 1s 19ms/step - loss: 0.0462 - acc: 0.9895 - val_loss: 0.5507 - val_acc: 0.8568
Epoch 12/20
30/30 [==============================] - 1s 21ms/step - loss: 0.0350 - acc: 0.9935 - val_loss: 0.5955 - val_acc: 0.8568
Epoch 13/20
30/30 [==============================] - 1s 20ms/step - loss: 0.0255 - acc: 0.9965 - val_loss: 0.6375 - val_acc: 0.8550
Epoch 14/20
30/30 [==============================] - 1s 21ms/step - loss: 0.0192 - acc: 0.9979 - val_loss: 0.6795 - val_acc: 0.8532
Epoch 15/20
30/30 [==============================] - 1s 20ms/step - loss: 0.0137 - acc: 0.9995 - val_loss: 0.7176 - val_acc: 0.8506
Epoch 16/20
30/30 [==============================] - 1s 26ms/step - loss: 0.0099 - acc: 0.9998 - val_loss: 0.7473 - val_acc: 0.8529
Epoch 17/20
30/30 [==============================] - 1s 25ms/step - loss: 0.0072 - acc: 1.0000 - val_loss: 0.7741 - val_acc: 0.8543
Epoch 18/20
30/30 [==============================] - 1s 21ms/step - loss: 0.0056 - acc: 1.0000 - val_loss: 0.8018 - val_acc: 0.8525
Epoch 19/20
30/30 [==============================] - 1s 29ms/step - loss: 0.0046 - acc: 1.0000 - val_loss: 0.8243 - val_acc: 0.8531
Epoch 20/20
30/30 [==============================] - 1s 25ms/step - loss: 0.0037 - acc: 1.0000 - val_loss: 0.8446 - val_acc: 0.8532
The Dense Convolutional Neural Network 1 layer took 22.8725 seconds to train.
```

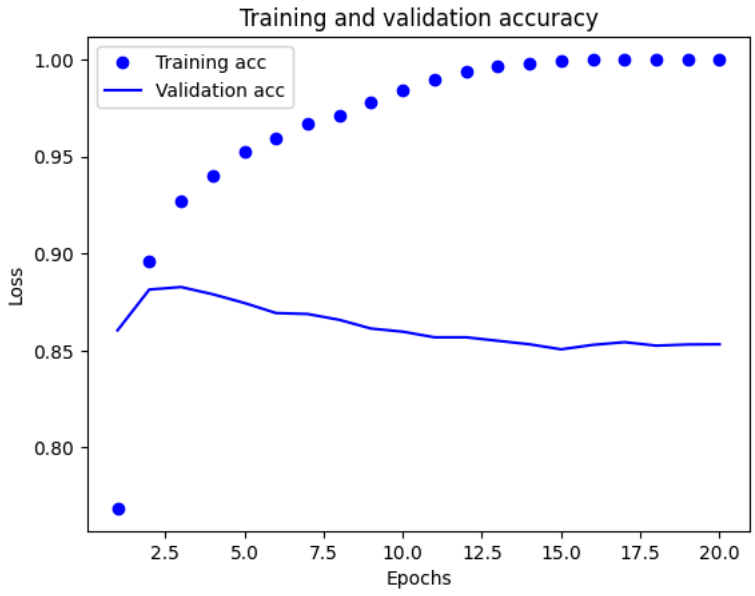
```
history_dict = history.history
history_dict.keys()
```

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

```
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()
```

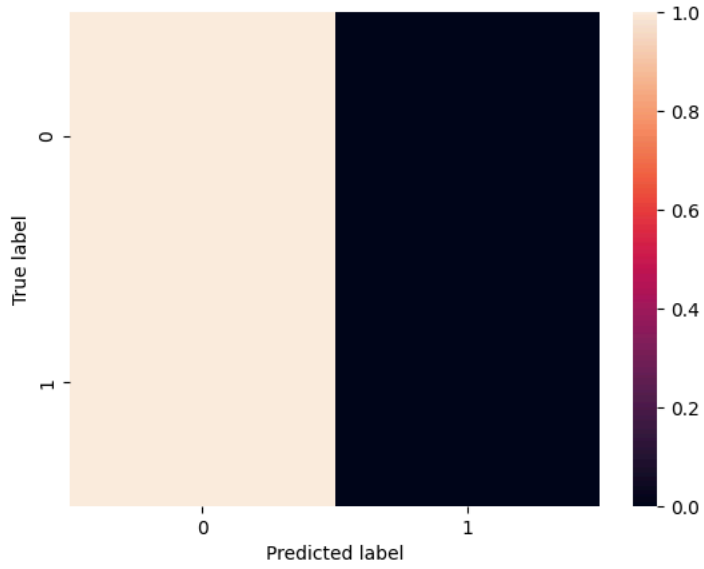



```
plt.clf()   # clear figure
acc_values = history_dict['acc']
val_acc_values = history_dict['val_acc']

# 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()
```



```
# 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: 161121 (629.38 KB)
Trainable params: 161121 (629.38 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
None
782/782 [==============================] - 2s 2ms/step
[[12500     0]
 [12500     0]]
Text(0.5, 23.52222222222222, 'Predicted label')
```

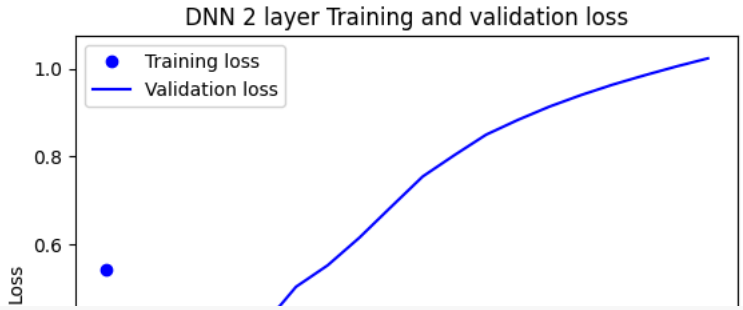



```
#Dense with Two Layer
model2 = models.Sequential()
model2.add(layers.Dense(32, activation='relu', input_shape=(5000,)))
model2.add(layers.Dense(32, activation='relu'))
model2.add(layers.Dense(32, activation='relu'))
model2.add(layers.Dense(1, activation='sigmoid'))
```

```
# 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." % (total_time_m2))
```

```
Epoch 1/20
30/30 [==============================] - 2s 31ms/step - loss: 0.5423 - acc: 0.7644 - val_loss: 0.3510 - val_acc: 0.8609
Epoch 2/20
30/30 [==============================] - 1s 18ms/step - loss: 0.2681 - acc: 0.8953 - val_loss: 0.2831 - val_acc: 0.8867
Epoch 3/20
30/30 [==============================] - 0s 14ms/step - loss: 0.1924 - acc: 0.9284 - val_loss: 0.2961 - val_acc: 0.8813
Epoch 4/20
30/30 [==============================] - 0s 12ms/step - loss: 0.1534 - acc: 0.9445 - val_loss: 0.3248 - val_acc: 0.8766
Epoch 5/20
30/30 [==============================] - 0s 13ms/step - loss: 0.1190 - acc: 0.9585 - val_loss: 0.3676 - val_acc: 0.8708
Epoch 6/20
30/30 [==============================] - 0s 13ms/step - loss: 0.0897 - acc: 0.9713 - val_loss: 0.4175 - val_acc: 0.8669
Epoch 7/20
30/30 [==============================] - 0s 12ms/step - loss: 0.0641 - acc: 0.9813 - val_loss: 0.5032 - val_acc: 0.8597
Epoch 8/20
30/30 [==============================] - 0s 13ms/step - loss: 0.0484 - acc: 0.9859 - val_loss: 0.5522 - val_acc: 0.8578
Epoch 9/20
30/30 [==============================] - 0s 13ms/step - loss: 0.0297 - acc: 0.9937 - val_loss: 0.6152 - val_acc: 0.8613
Epoch 10/20
30/30 [==============================] - 0s 13ms/step - loss: 0.0148 - acc: 0.9981 - val_loss: 0.6853 - val_acc: 0.8599
Epoch 11/20
30/30 [==============================] - 0s 12ms/step - loss: 0.0074 - acc: 0.9996 - val_loss: 0.7547 - val_acc: 0.8586
Epoch 12/20
30/30 [==============================] - 0s 12ms/step - loss: 0.0037 - acc: 0.9999 - val_loss: 0.8034 - val_acc: 0.8589
Epoch 13/20
30/30 [==============================] - 0s 12ms/step - loss: 0.0023 - acc: 0.9999 - val_loss: 0.8500 - val_acc: 0.8588
Epoch 14/20
30/30 [==============================] - 0s 13ms/step - loss: 0.0016 - acc: 1.0000 - val_loss: 0.8836 - val_acc: 0.8597
Epoch 15/20
30/30 [==============================] - 0s 13ms/step - loss: 0.0012 - acc: 1.0000 - val_loss: 0.9140 - val_acc: 0.8587
Epoch 16/20
30/30 [==============================] - 0s 12ms/step - loss: 9.0390e-04 - acc: 1.0000 - val_loss: 0.9404 - val_acc: 0.8588
Epoch 17/20
30/30 [==============================] - 0s 13ms/step - loss: 7.2729e-04 - acc: 1.0000 - val_loss: 0.9641 - val_acc: 0.8586
Epoch 18/20
30/30 [==============================] - 1s 23ms/step - loss: 5.9766e-04 - acc: 1.0000 - val_loss: 0.9853 - val_acc: 0.8586
Epoch 19/20
30/30 [==============================] - 1s 18ms/step - loss: 5.0253e-04 - acc: 1.0000 - val_loss: 1.0054 - val_acc: 0.8591
Epoch 20/20
30/30 [==============================] - 0s 14ms/step - loss: 4.2890e-04 - acc: 1.0000 - val_loss: 1.0239 - val_acc: 0.8584
The Dense Convolutional Neural Network 2 layers took 10.4195 seconds to train.
```

```
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()
```




```
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()
```