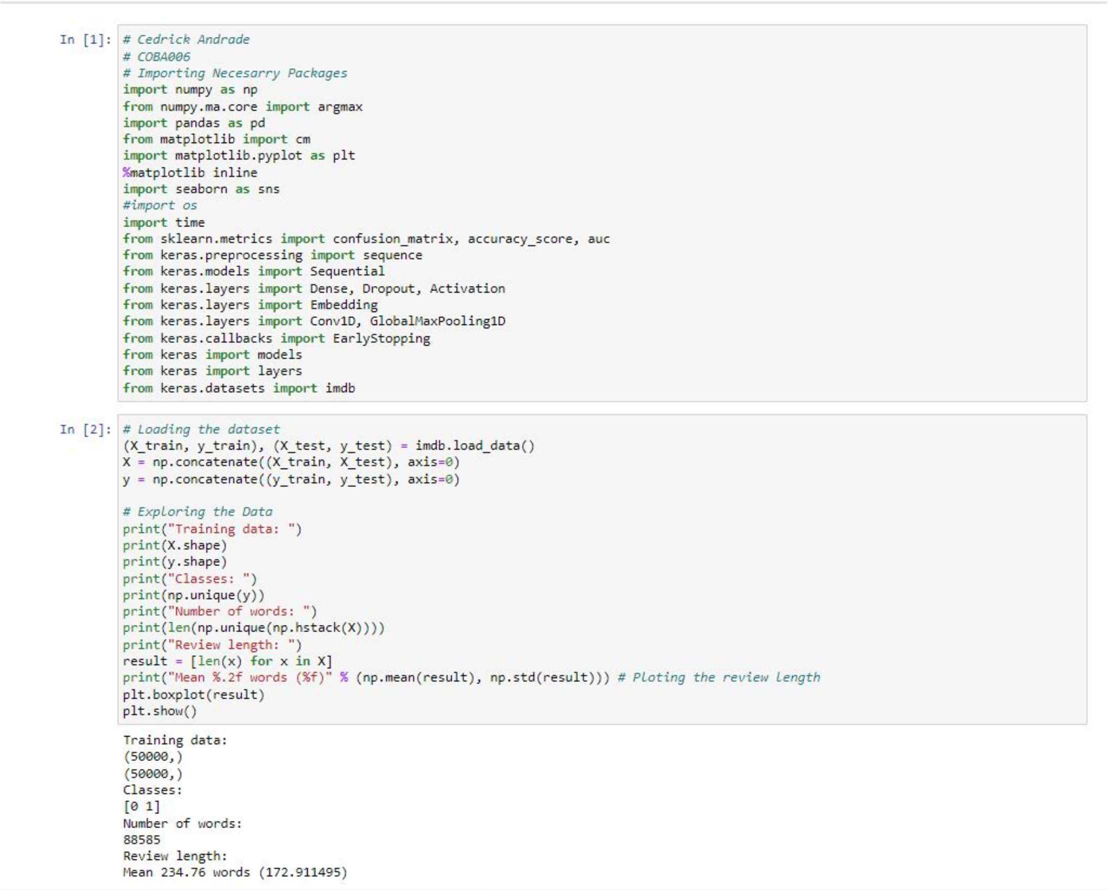
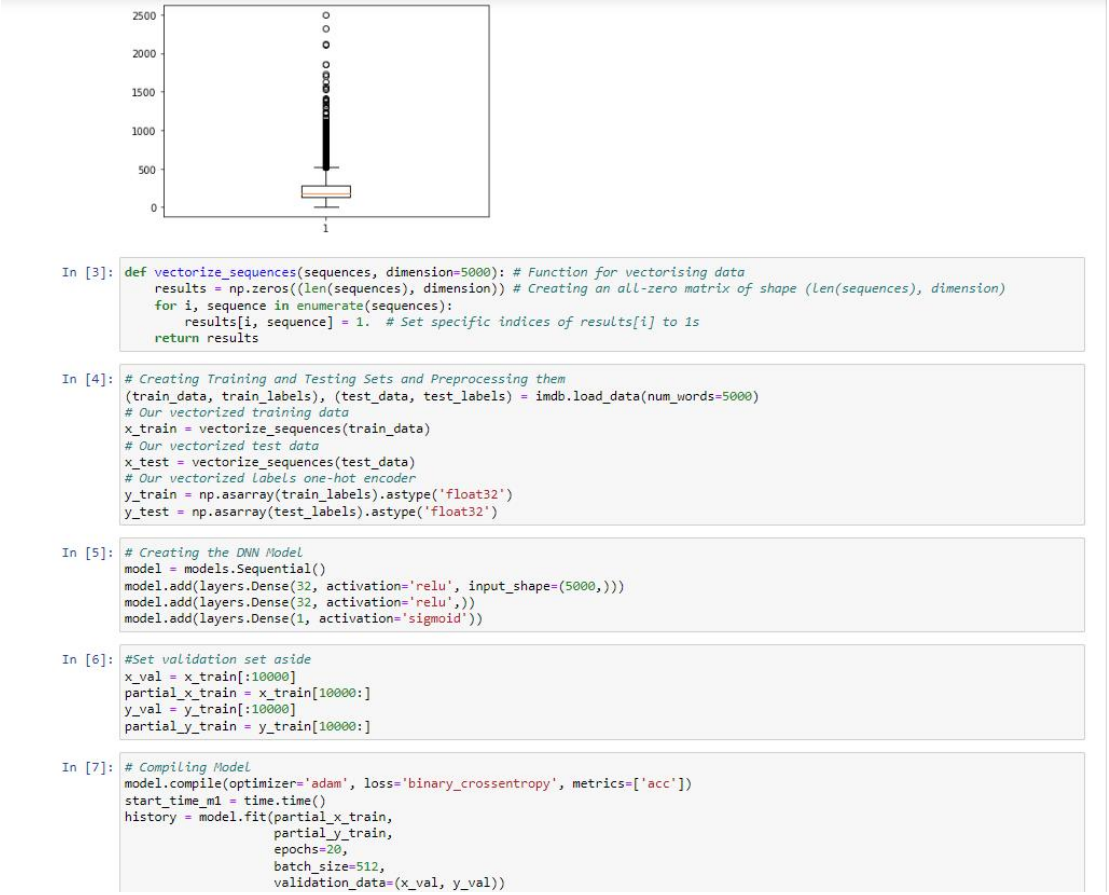
Cedrick Andrade
COBA006



In [1]:
```
# Cedrick Andrade
# COBA006
# 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 length
plt.boxplot(result)
plt.show()
```

```
Training data:
(50000,)
(50000,)
Classes:
[0 1]
Number of words:
88585
Review length:
Mean 234.76 words (172.911495)
```



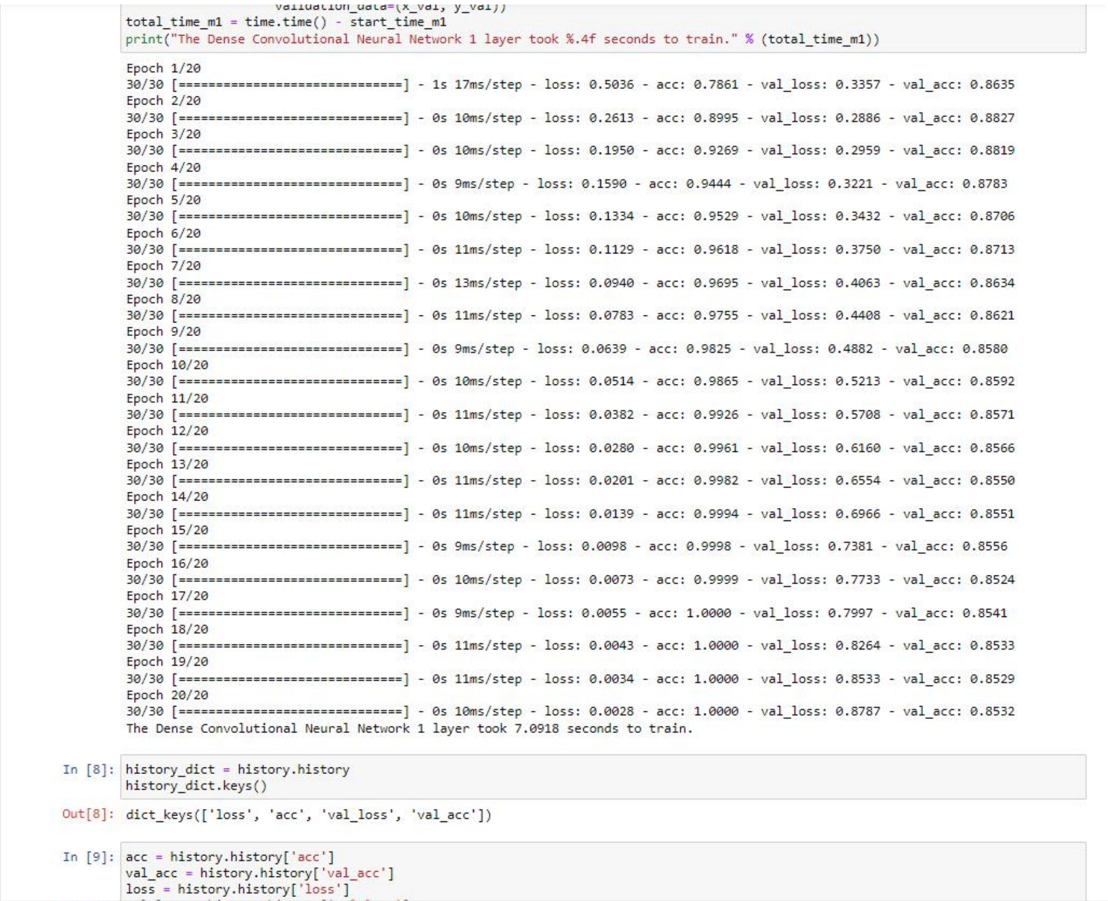
In [3]:
```
def vectorize_sequences(sequences, dimension=5000): # Function for vectorising data
    results = np.zeros((len(sequences), dimension)) # Creating an all-zero matrix of shape (len(sequences), dimension)
    for i, sequence in enumerate(sequences):
        results[i, sequence] = 1.  # Set 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')
```

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))
```



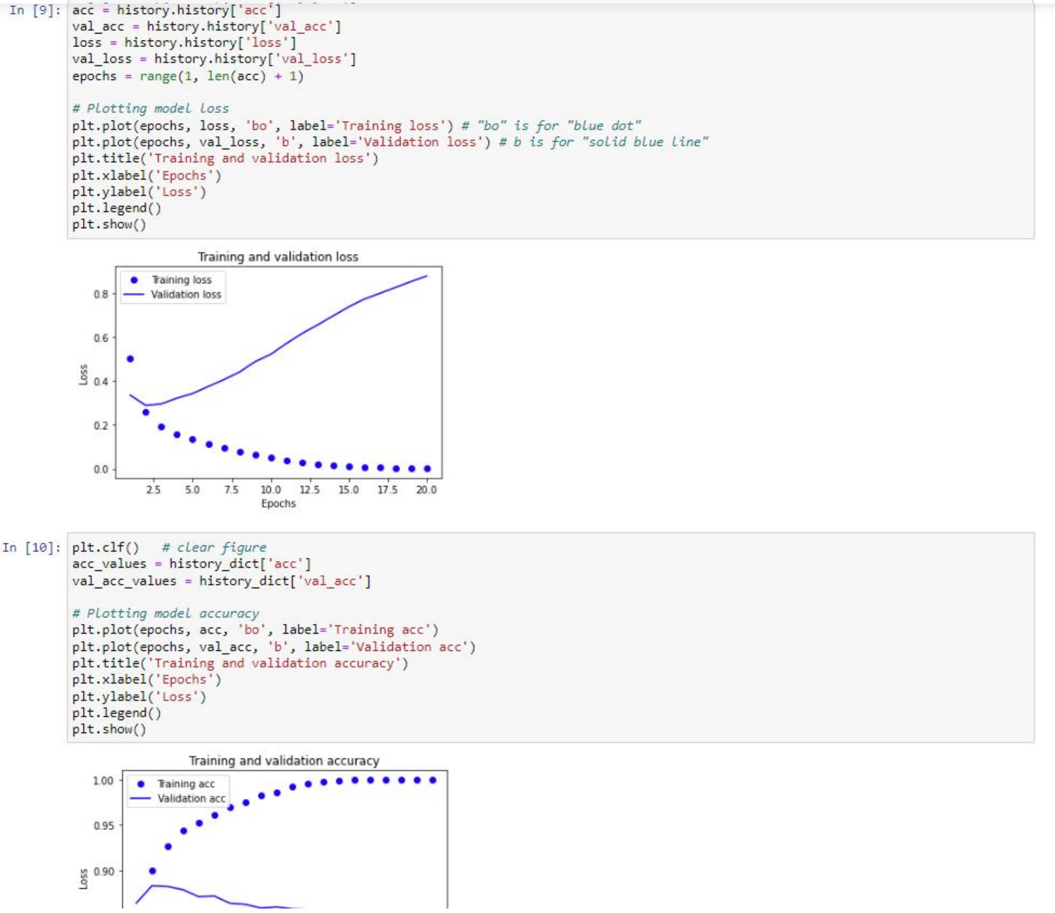
```
                    validation_data=(x_val, y_val))
total_time_m1 = time.time() - start_time_m1
print("The Dense Convolutional Neural Network 1 layer took %.4f seconds to train." % (total_time_m1))
```

```
Epoch 1/20
30/30 [==============================] - 1s 17ms/step - loss: 0.5036 - acc: 0.7861 - val_loss: 0.3357 - val_acc: 0.8635
Epoch 2/20
30/30 [==============================] - 0s 10ms/step - loss: 0.2613 - acc: 0.8995 - val_loss: 0.2886 - val_acc: 0.8827
Epoch 3/20
30/30 [==============================] - 0s 10ms/step - loss: 0.1950 - acc: 0.9269 - val_loss: 0.2959 - val_acc: 0.8819
Epoch 4/20
30/30 [==============================] - 0s 9ms/step - loss: 0.1590 - acc: 0.9444 - val_loss: 0.3221 - val_acc: 0.8783
Epoch 5/20
30/30 [==============================] - 0s 10ms/step - loss: 0.1334 - acc: 0.9529 - val_loss: 0.3432 - val_acc: 0.8706
Epoch 6/20
30/30 [==============================] - 0s 11ms/step - loss: 0.1129 - acc: 0.9618 - val_loss: 0.3750 - val_acc: 0.8713
Epoch 7/20
30/30 [==============================] - 0s 13ms/step - loss: 0.0940 - acc: 0.9695 - val_loss: 0.4063 - val_acc: 0.8634
Epoch 8/20
30/30 [==============================] - 0s 11ms/step - loss: 0.0783 - acc: 0.9755 - val_loss: 0.4408 - val_acc: 0.8621
Epoch 9/20
30/30 [==============================] - 0s 9ms/step - loss: 0.0639 - acc: 0.9825 - val_loss: 0.4882 - val_acc: 0.8580
Epoch 10/20
30/30 [==============================] - 0s 10ms/step - loss: 0.0514 - acc: 0.9865 - val_loss: 0.5213 - val_acc: 0.8592
Epoch 11/20
30/30 [==============================] - 0s 11ms/step - loss: 0.0382 - acc: 0.9926 - val_loss: 0.5708 - val_acc: 0.8571
Epoch 12/20
30/30 [==============================] - 0s 10ms/step - loss: 0.0280 - acc: 0.9961 - val_loss: 0.6160 - val_acc: 0.8566
Epoch 13/20
30/30 [==============================] - 0s 11ms/step - loss: 0.0201 - acc: 0.9982 - val_loss: 0.6554 - val_acc: 0.8550
Epoch 14/20
30/30 [==============================] - 0s 11ms/step - loss: 0.0139 - acc: 0.9994 - val_loss: 0.6966 - val_acc: 0.8551
Epoch 15/20
30/30 [==============================] - 0s 9ms/step - loss: 0.0098 - acc: 0.9998 - val_loss: 0.7381 - val_acc: 0.8556
Epoch 16/20
30/30 [==============================] - 0s 10ms/step - loss: 0.0073 - acc: 0.9999 - val_loss: 0.7733 - val_acc: 0.8524
Epoch 17/20
30/30 [==============================] - 0s 9ms/step - loss: 0.0055 - acc: 1.0000 - val_loss: 0.7997 - val_acc: 0.8541
Epoch 18/20
30/30 [==============================] - 0s 11ms/step - loss: 0.0043 - acc: 1.0000 - val_loss: 0.8264 - val_acc: 0.8533
Epoch 19/20
30/30 [==============================] - 0s 11ms/step - loss: 0.0034 - acc: 1.0000 - val_loss: 0.8533 - val_acc: 0.8529
Epoch 20/20
30/30 [==============================] - 0s 10ms/step - loss: 0.0028 - acc: 1.0000 - val_loss: 0.8787 - val_acc: 0.8532
The Dense Convolutional Neural Network 1 layer took 7.0918 seconds to train.
```

```
In [8]: history_dict = history.history
        history_dict.keys()
```

```
Out[8]: dict_keys(['loss', 'acc', 'val_loss', 'val_acc'])
```

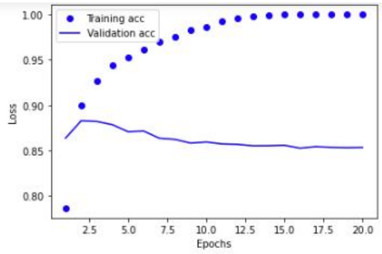
```
In [9]: acc = history.history['acc']
        val_acc = history.history['val_acc']
        loss = history.history['loss']
```

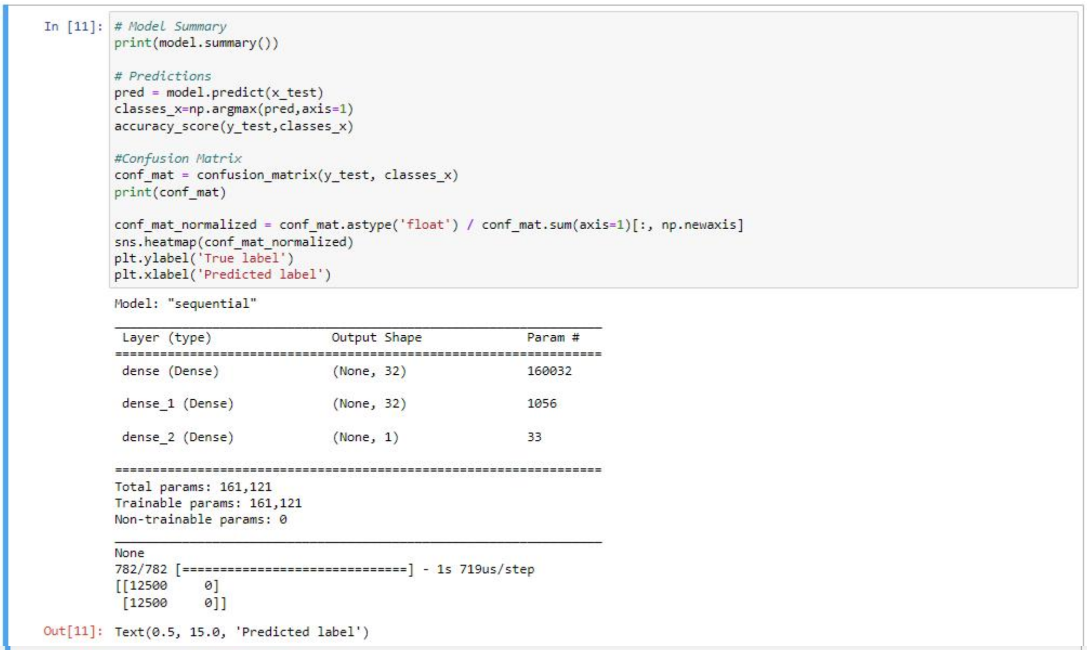


```
In [9]: acc = history.history['acc']
        val_acc = history.history['val_acc']
        loss = history.history['loss']
```

```
        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()
```

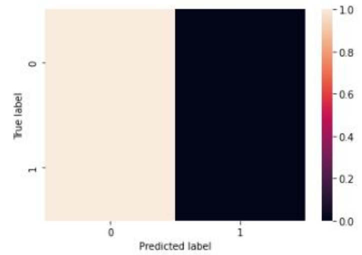


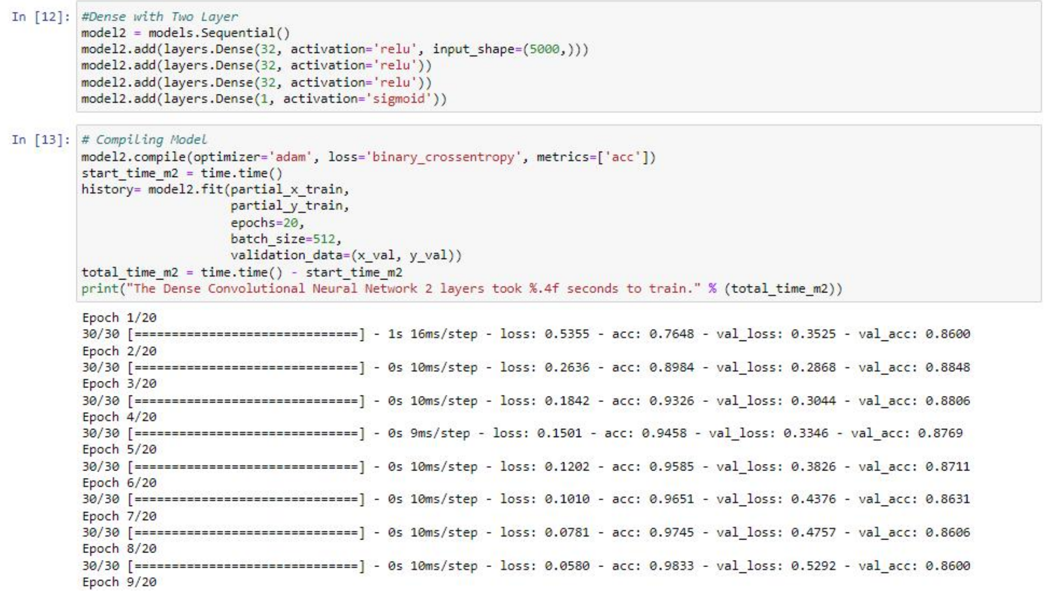


```
In [11]: # Model Summary
         print(model.summary())

         # 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
782/782 [==============================] - 1s 719us/step
[[12500     0]
 [12500     0]]
Out[11]: Text(0.5, 15.0, 'Predicted label')
```

Out[11]: Text(0.5, 15.0, 'Predicted label')

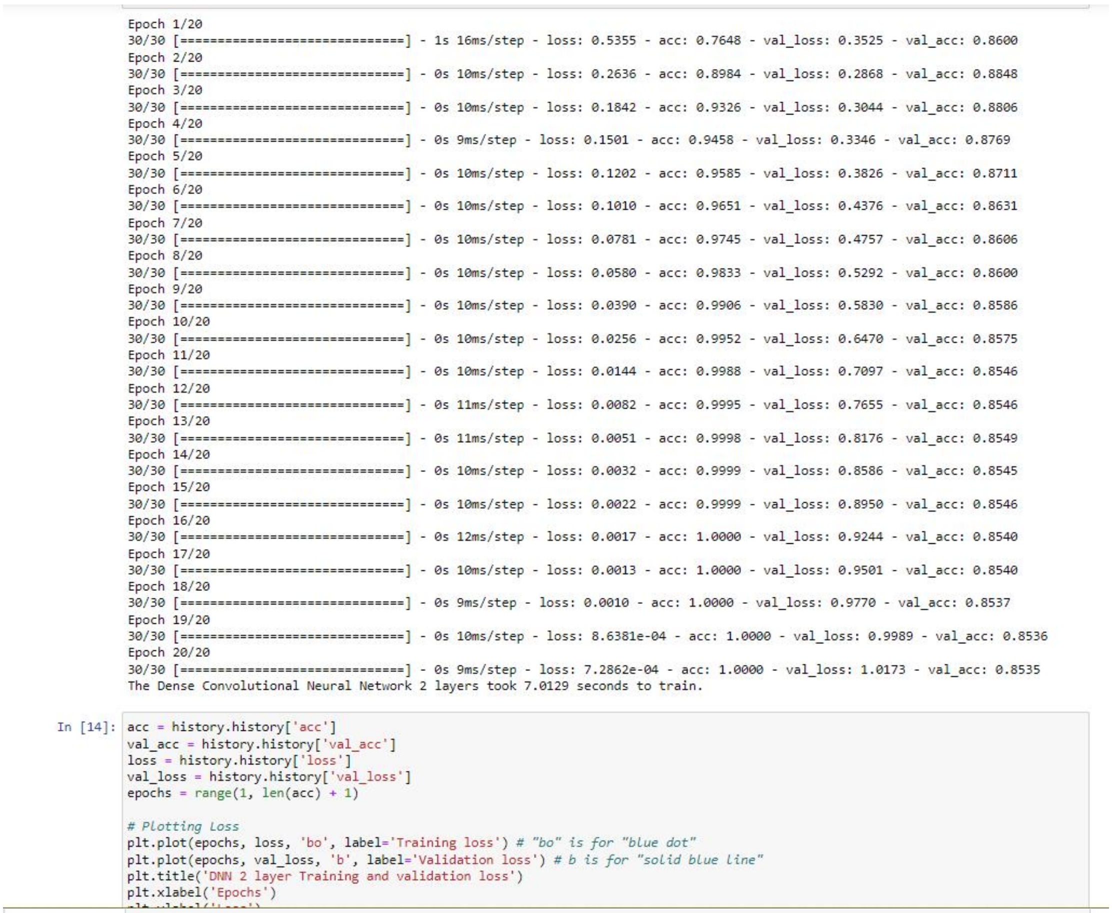




```
In [12]: #Dense with Two Layer
         model2 = models.Sequential()
         model2.add(layers.Dense(32, activation='relu', input_shape=(5000,)))
         model2.add(layers.Dense(32, activation='relu'))
         model2.add(layers.Dense(32, activation='relu'))
         model2.add(layers.Dense(1, activation='sigmoid'))
```

```
In [13]: # 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))
```

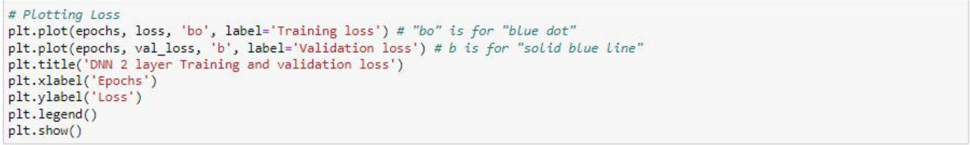
```
Epoch 1/20
30/30 [==============================] - 1s 16ms/step - loss: 0.5355 - acc: 0.7648 - val_loss: 0.3525 - val_acc: 0.8600
Epoch 2/20
30/30 [==============================] - 0s 10ms/step - loss: 0.2636 - acc: 0.8984 - val_loss: 0.2868 - val_acc: 0.8848
Epoch 3/20
30/30 [==============================] - 0s 10ms/step - loss: 0.1842 - acc: 0.9326 - val_loss: 0.3044 - val_acc: 0.8806
Epoch 4/20
30/30 [==============================] - 0s 9ms/step - loss: 0.1501 - acc: 0.9458 - val_loss: 0.3346 - val_acc: 0.8769
Epoch 5/20
30/30 [==============================] - 0s 10ms/step - loss: 0.1202 - acc: 0.9585 - val_loss: 0.3826 - val_acc: 0.8711
Epoch 6/20
30/30 [==============================] - 0s 10ms/step - loss: 0.1010 - acc: 0.9651 - val_loss: 0.4376 - val_acc: 0.8631
Epoch 7/20
30/30 [==============================] - 0s 10ms/step - loss: 0.0781 - acc: 0.9745 - val_loss: 0.4757 - val_acc: 0.8606
Epoch 8/20
30/30 [==============================] - 0s 10ms/step - loss: 0.0580 - acc: 0.9833 - val_loss: 0.5292 - val_acc: 0.8600
Epoch 9/20
```



```
Epoch 1/20
30/30 [==============================] - 1s 16ms/step - loss: 0.5355 - acc: 0.7648 - val_loss: 0.3525 - val_acc: 0.8600
Epoch 2/20
30/30 [==============================] - 0s 10ms/step - loss: 0.2636 - acc: 0.8984 - val_loss: 0.2868 - val_acc: 0.8848
Epoch 3/20
30/30 [==============================] - 0s 10ms/step - loss: 0.1842 - acc: 0.9326 - val_loss: 0.3044 - val_acc: 0.8806
Epoch 4/20
30/30 [==============================] - 0s 9ms/step - loss: 0.1501 - acc: 0.9458 - val_loss: 0.3346 - val_acc: 0.8769
Epoch 5/20
30/30 [==============================] - 0s 10ms/step - loss: 0.1202 - acc: 0.9585 - val_loss: 0.3826 - val_acc: 0.8711
Epoch 6/20
30/30 [==============================] - 0s 10ms/step - loss: 0.1010 - acc: 0.9651 - val_loss: 0.4376 - val_acc: 0.8631
Epoch 7/20
30/30 [==============================] - 0s 10ms/step - loss: 0.0781 - acc: 0.9745 - val_loss: 0.4757 - val_acc: 0.8606
Epoch 8/20
30/30 [==============================] - 0s 10ms/step - loss: 0.0580 - acc: 0.9833 - val_loss: 0.5292 - val_acc: 0.8600
Epoch 9/20
30/30 [==============================] - 0s 10ms/step - loss: 0.0390 - acc: 0.9906 - val_loss: 0.5830 - val_acc: 0.8586
Epoch 10/20
30/30 [==============================] - 0s 10ms/step - loss: 0.0256 - acc: 0.9952 - val_loss: 0.6470 - val_acc: 0.8575
Epoch 11/20
30/30 [==============================] - 0s 10ms/step - loss: 0.0144 - acc: 0.9988 - val_loss: 0.7097 - val_acc: 0.8546
Epoch 12/20
30/30 [==============================] - 0s 11ms/step - loss: 0.0082 - acc: 0.9995 - val_loss: 0.7655 - val_acc: 0.8546
Epoch 13/20
30/30 [==============================] - 0s 11ms/step - loss: 0.0051 - acc: 0.9998 - val_loss: 0.8176 - val_acc: 0.8549
Epoch 14/20
30/30 [==============================] - 0s 10ms/step - loss: 0.0032 - acc: 0.9999 - val_loss: 0.8586 - val_acc: 0.8545
Epoch 15/20
30/30 [==============================] - 0s 10ms/step - loss: 0.0022 - acc: 0.9999 - val_loss: 0.8950 - val_acc: 0.8546
Epoch 16/20
30/30 [==============================] - 0s 12ms/step - loss: 0.0017 - acc: 1.0000 - val_loss: 0.9244 - val_acc: 0.8540
Epoch 17/20
30/30 [==============================] - 0s 10ms/step - loss: 0.0013 - acc: 1.0000 - val_loss: 0.9501 - val_acc: 0.8540
Epoch 18/20
30/30 [==============================] - 0s 9ms/step - loss: 0.0010 - acc: 1.0000 - val_loss: 0.9770 - val_acc: 0.8537
Epoch 19/20
30/30 [==============================] - 0s 10ms/step - loss: 8.6381e-04 - acc: 1.0000 - val_loss: 0.9989 - val_acc: 0.8536
Epoch 20/20
30/30 [==============================] - 0s 9ms/step - loss: 7.2862e-04 - acc: 1.0000 - val_loss: 1.0173 - val_acc: 0.8535
The Dense Convolutional Neural Network 2 layers took 7.0129 seconds to train.
```

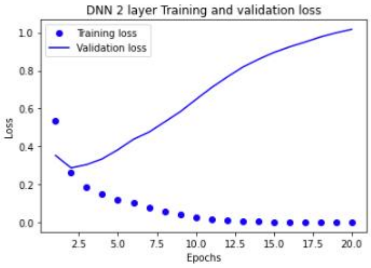
In [14]:

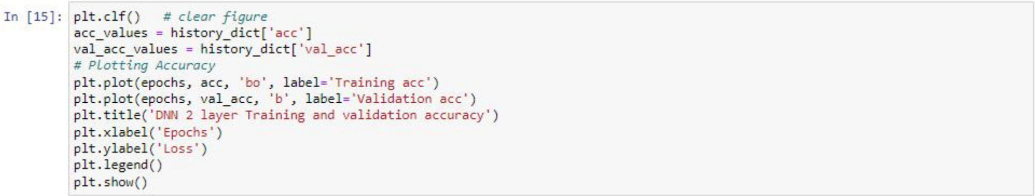
```python
acc = history.history['acc']
val_acc = history.history['val_acc']
loss = history.history['loss']
val_loss = history.history['val_loss']
epochs = range(1, len(acc) + 1)

# 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')
```



```python
# 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')
```

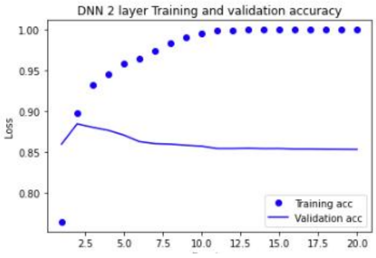
```python
plt.ylabel('Loss')
plt.legend()
plt.show()
```

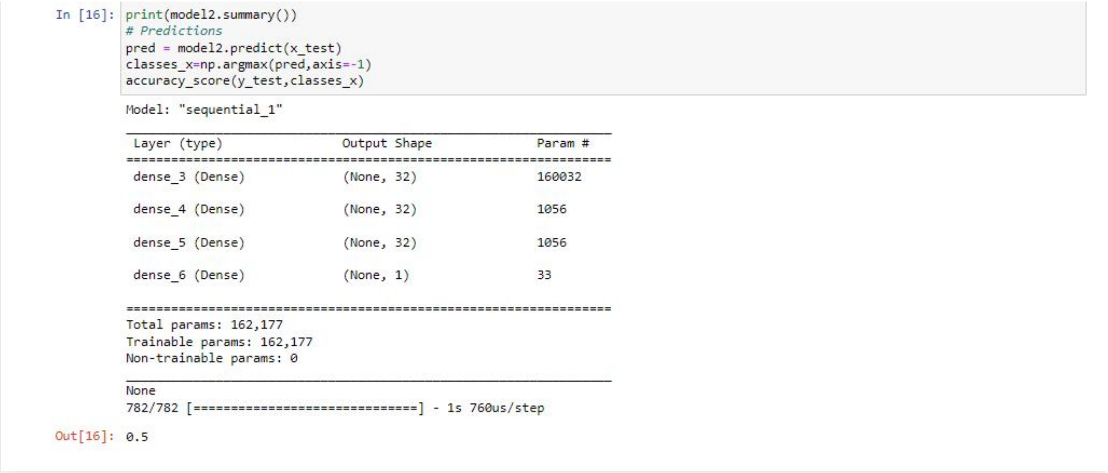




In [15]:

```python
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 [16]:

```
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
782/782 [==============================] - 1s 760us/step
```

Out[16]:

```
0.5
```