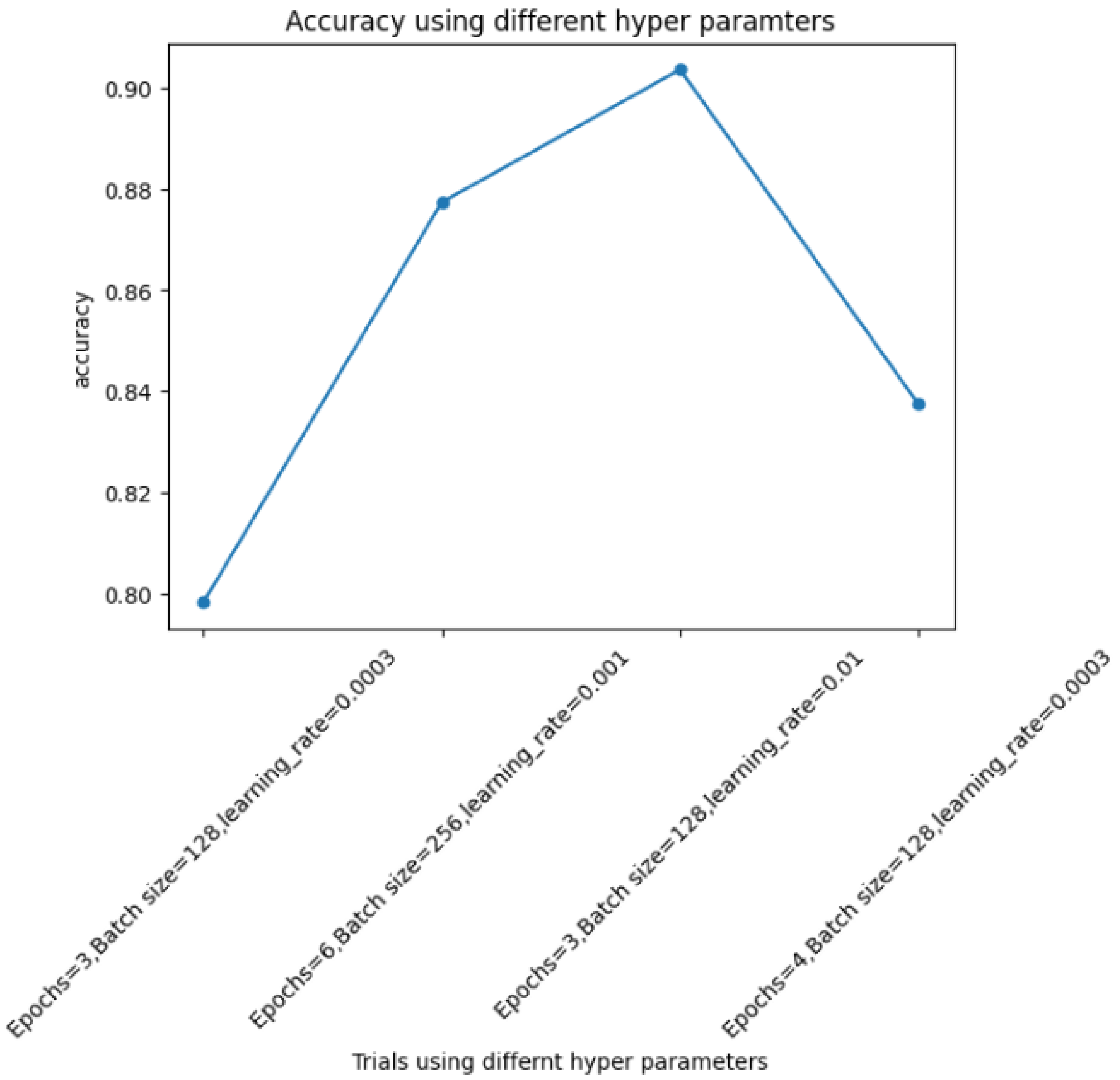
NLP Report

<u>I-Best accuracy results</u>

1-CNN MODELS



Model1 accuracy (0.798) layers:

```
embedding (Embedding)
dropout (Dropout
conv1d (Conv1D) relu
 max_pooling1d (MaxPooling1D  (  )
 conv1d_1 (Conv1D) relu
 max_pooling1d_1 (MaxPooling   )
flatten (Flatten)
dense (Dense)  relu
dropout_1 (Dropout
dense_1 (Dense)softmax
```

Model2 accuracy (0.877) layers :

```
embedding (Embedding)
dropout (Dropout
conv1d (Conv1D) relu
 max_pooling1d (MaxPooling1D  (  )
 conv1d_1 (Conv1D) relu
 max_pooling1d_1 (MaxPooling   )
flatten (Flatten)
dense (Dense)  relu
dropout_1 (Dropout
dense_1 (Dense)softmax
```

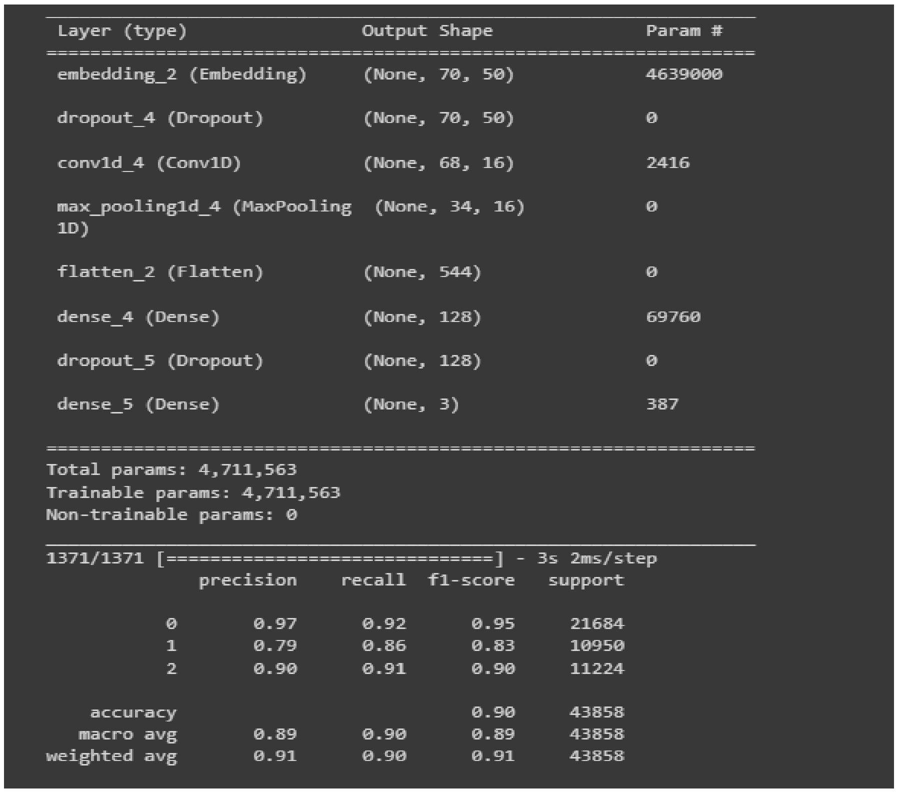
Model3 accuracy (0.90) layers(**Best accuracy):**

```
embedding (Embedding)
dropout (Dropout
conv1d (Conv1D) relu
 max_pooling1d (MaxPooling1D  (  )
flatten (Flatten)
dense (Dense)  relu
dropout_1 (Dropout
dense_1 (Dense)softmax
```

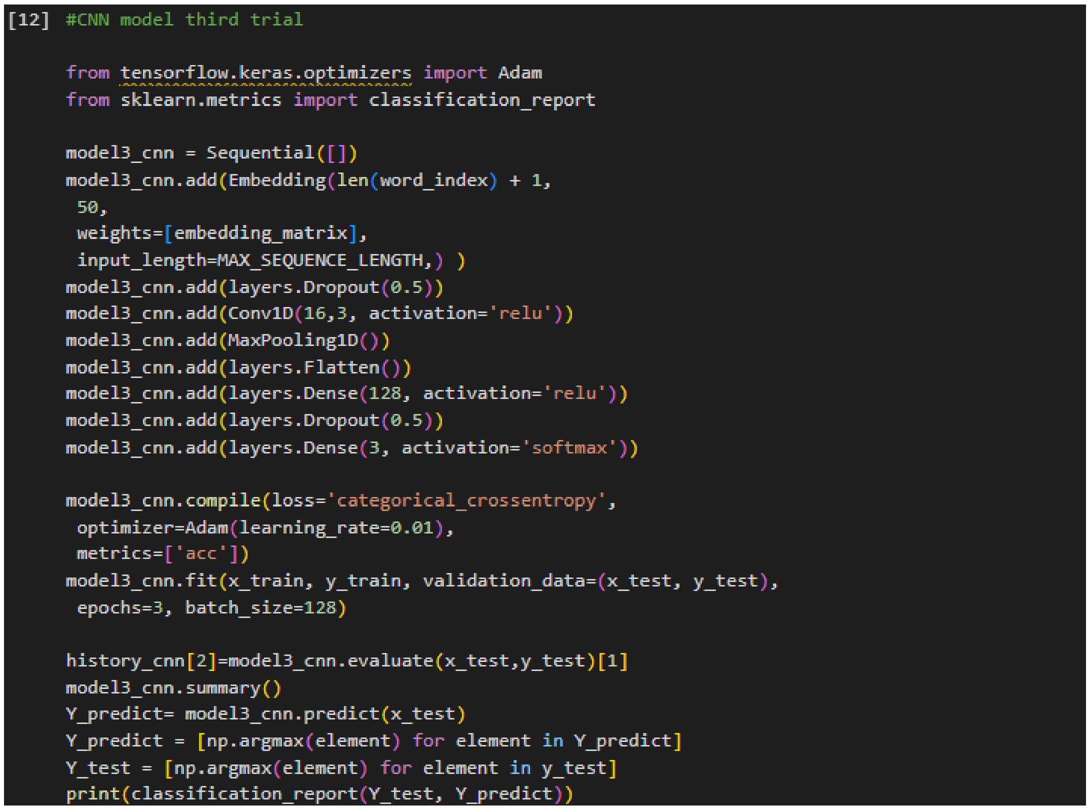
Model4 accuracy (0.83) layers:

```
embedding (Embedding)
dropout (Dropout
conv1d (Conv1D) relu
 max_pooling1d (MaxPooling1D  (  )
flatten (Flatten)
dense (Dense)  relu
dropout_1 (Dropout
dense_1 (Dense)softmax
```

Best CNN model (model3)

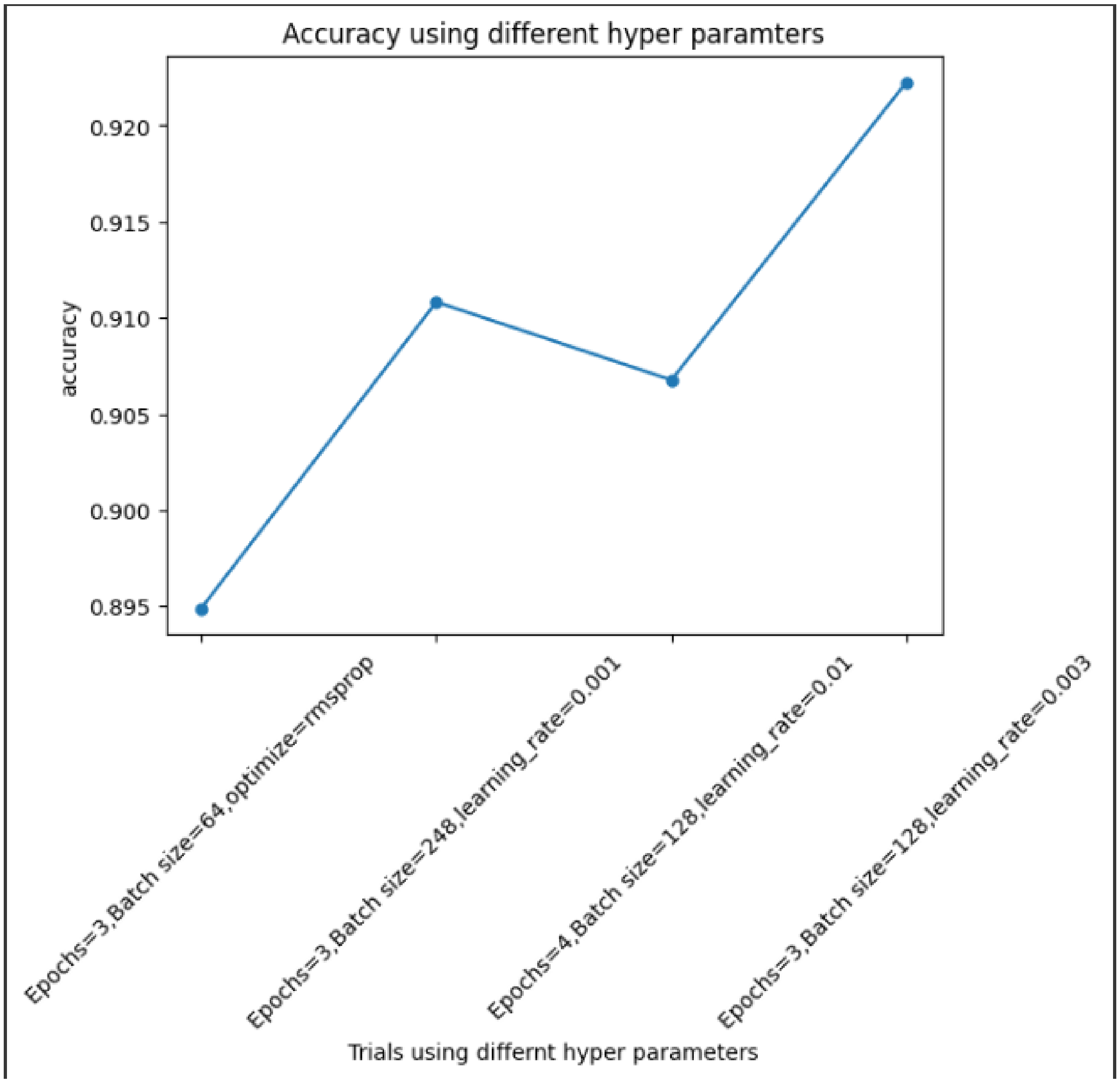


```
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 embedding_2 (Embedding)     (None, 70, 50)            4639000   
                                                                 
 dropout_4 (Dropout)         (None, 70, 50)            0         
                                                                 
 conv1d_4 (Conv1D)           (None, 68, 16)            2416      
                                                                 
 max_pooling1d_4 (MaxPooling  (None, 34, 16)           0         
 1D)                                                             
                                                                 
 flatten_2 (Flatten)         (None, 544)               0         
                                                                 
 dense_4 (Dense)             (None, 128)               69760     
                                                                 
 dropout_5 (Dropout)         (None, 128)               0         
                                                                 
 dense_5 (Dense)             (None, 3)                 387       
                                                                 
=================================================================
Total params: 4,711,563
Trainable params: 4,711,563
Non-trainable params: 0
_________________________________________________________________
1371/1371 [==============================] - 3s 2ms/step
              precision    recall  f1-score   support

           0       0.97      0.92      0.95     21684
           1       0.79      0.86      0.83     10950
           2       0.90      0.91      0.90     11224

    accuracy                           0.90     43858
   macro avg       0.89      0.90      0.89     43858
weighted avg       0.91      0.90      0.91     43858
```



```
[12] #CNN model third trial

     from tensorflow.keras.optimizers import Adam
     from sklearn.metrics import classification_report

     model3_cnn = Sequential([])
     model3_cnn.add(Embedding(len(word_index) + 1,
      50,
      weights=[embedding_matrix],
      input_length=MAX_SEQUENCE_LENGTH,) )
     model3_cnn.add(layers.Dropout(0.5))
     model3_cnn.add(Conv1D(16,3, activation='relu'))
     model3_cnn.add(MaxPooling1D())
     model3_cnn.add(layers.Flatten())
     model3_cnn.add(layers.Dense(128, activation='relu'))
     model3_cnn.add(layers.Dropout(0.5))
     model3_cnn.add(layers.Dense(3, activation='softmax'))

     model3_cnn.compile(loss='categorical_crossentropy',
      optimizer=Adam(learning_rate=0.01),
      metrics=['acc'])
     model3_cnn.fit(x_train, y_train, validation_data=(x_test, y_test),
      epochs=3, batch_size=128)

     history_cnn[2]=model3_cnn.evaluate(x_test,y_test)[1]
     model3_cnn.summary()
     Y_predict= model3_cnn.predict(x_test)
     Y_predict = [np.argmax(element) for element in Y_predict]
     Y_test = [np.argmax(element) for element in y_test]
     print(classification_report(Y_test, Y_predict))
```

2-LSTM Models



Model1 accuracy (accuracy :0.89)layers:

```
 (Embedding)
 (LSTM)
 (Dropout)
 (Flatten)
 (Dense)(relu)
 (Dropout)
 (Dense)(softmax)
```

Model2 accuracy (accuracy: 0.910)layers:

```
(Embedding)
 (LSTM)
 (Flatten)
 (Dense)(softmax)
```

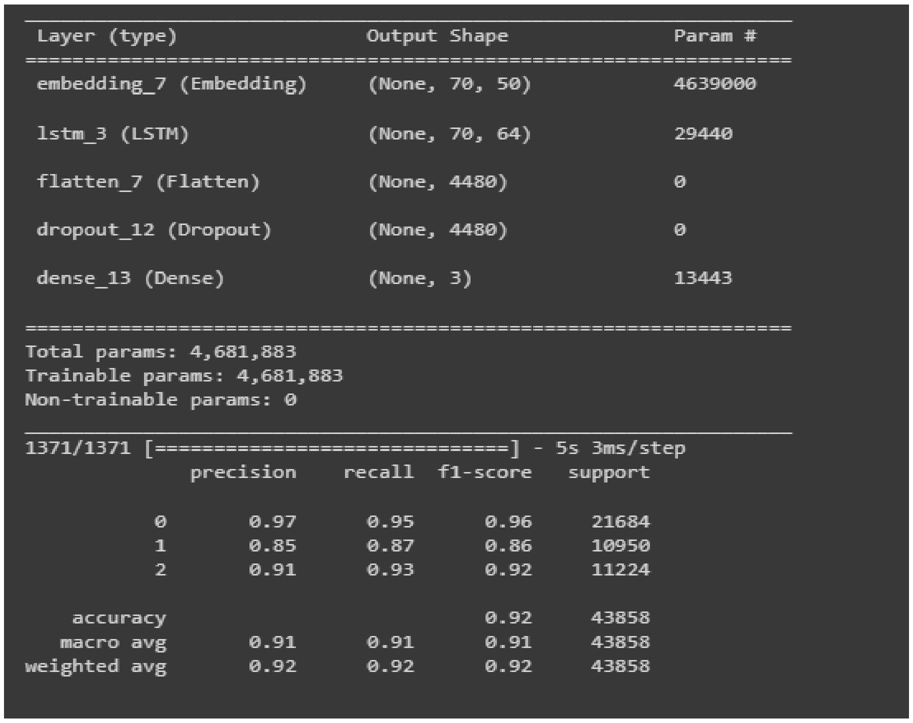
Model3 accuracy (accuracy : 0.906)layers:

```
(Embedding)
 (LSTM)
 (Dropout)
 (Flatten)
 (Dense)(relu)
 (Dropout)
 (Dense)(softmax)
```

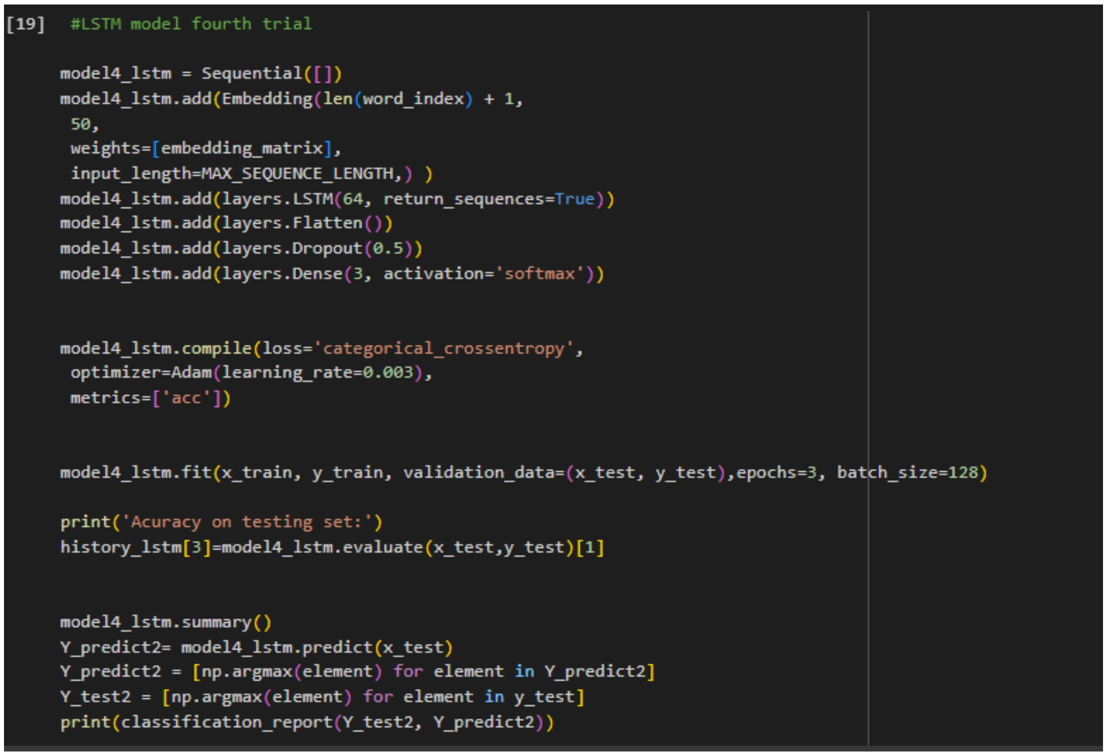
Model4 accuracy (accuracy :0.92)layers: **Best accuracy**

```
(Embedding)
 (LSTM)
 (Flatten)
 (Dropout)
 (Dense)(softmax)
```

Best LSTM model (Model 4)



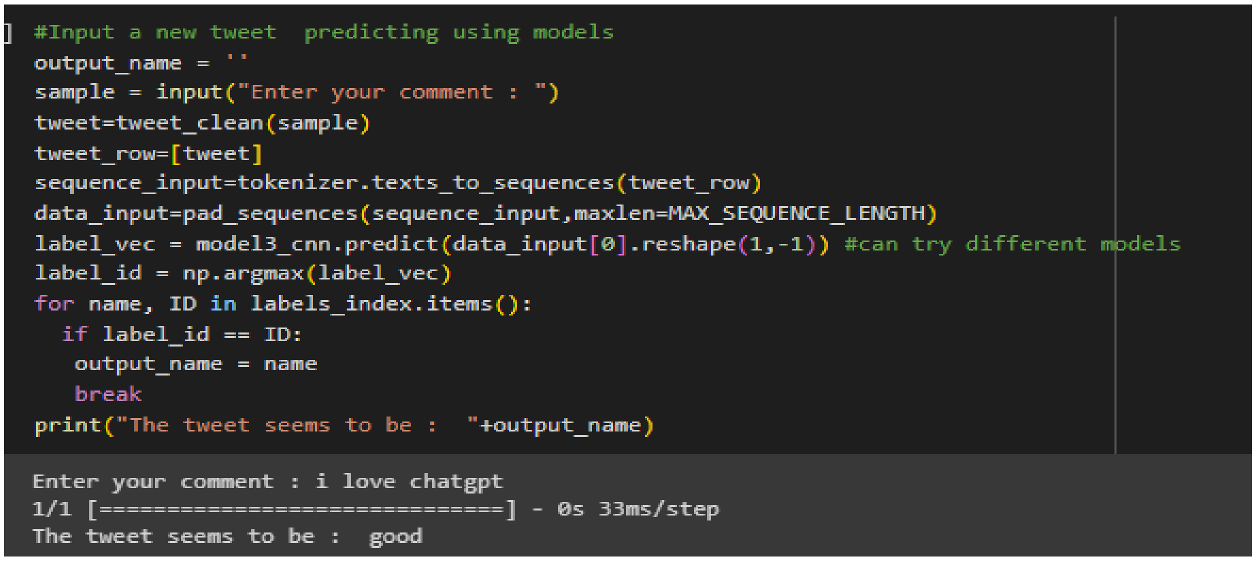
```
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding_7 (Embedding)     (None, 70, 50)            4639000

 lstm_3 (LSTM)               (None, 70, 64)            29440

 flatten_7 (Flatten)         (None, 4480)              0

 dropout_12 (Dropout)        (None, 4480)              0

 dense_13 (Dense)            (None, 3)                 13443

=================================================================
Total params: 4,681,883
Trainable params: 4,681,883
Non-trainable params: 0
_________________________________________________________________
1371/1371 [==============================] - 5s 3ms/step
              precision    recall  f1-score   support

           0       0.97      0.95      0.96     21684
           1       0.85      0.87      0.86     10950
           2       0.91      0.93      0.92     11224

    accuracy                           0.92     43858
   macro avg       0.91      0.91      0.91     43858
weighted avg       0.92      0.92      0.92     43858
```



```
[19]  #LSTM model fourth trial

    model4_lstm = Sequential([])
    model4_lstm.add(Embedding(len(word_index) + 1,
     50,
     weights=[embedding_matrix],
     input_length=MAX_SEQUENCE_LENGTH,) )
    model4_lstm.add(layers.LSTM(64, return_sequences=True))
    model4_lstm.add(layers.Flatten())
    model4_lstm.add(layers.Dropout(0.5))
    model4_lstm.add(layers.Dense(3, activation='softmax'))


    model4_lstm.compile(loss='categorical_crossentropy',
     optimizer=Adam(learning_rate=0.003),
     metrics=['acc'])


    model4_lstm.fit(x_train, y_train, validation_data=(x_test, y_test),epochs=3, batch_size=128)

    print('Acuracy on testing set:')
    history_lstm[3]=model4_lstm.evaluate(x_test,y_test)[1]


    model4_lstm.summary()
    Y_predict2= model4_lstm.predict(x_test)
    Y_predict2 = [np.argmax(element) for element in Y_predict2]
    Y_test2 = [np.argmax(element) for element in y_test]
    print(classification_report(Y_test2, Y_predict2))
```

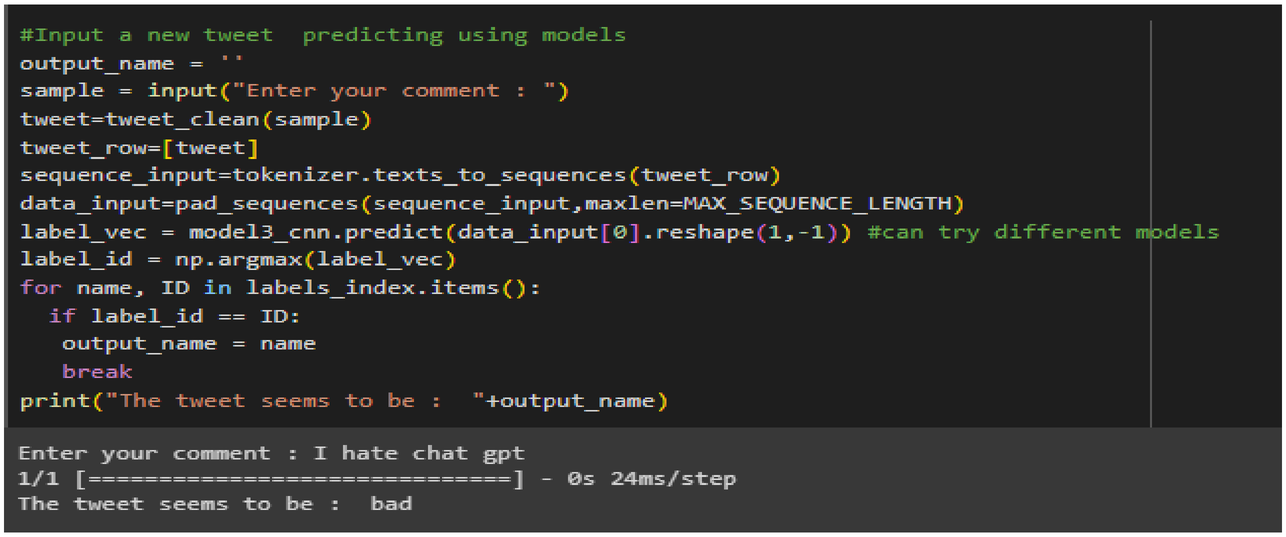
II-New tweets (inputs)to be classified

1-Input using CNN model



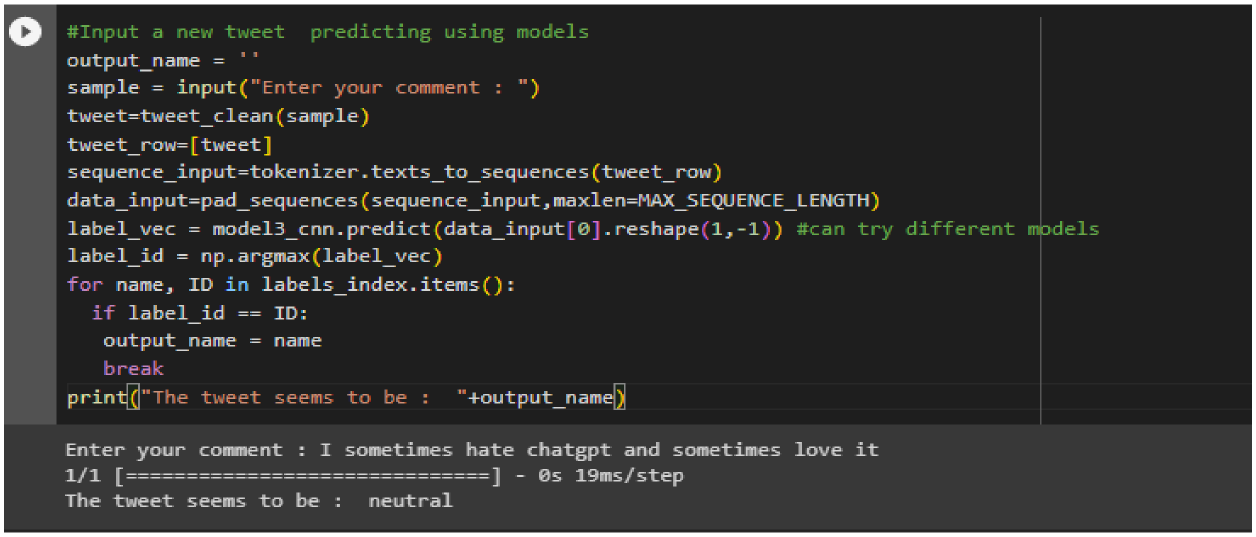
```
#Input a new tweet  predicting using models
output_name = ''
sample = input("Enter your comment : ")
tweet=tweet_clean(sample)
tweet_row=[tweet]
sequence_input=tokenizer.texts_to_sequences(tweet_row)
data_input=pad_sequences(sequence_input,maxlen=MAX_SEQUENCE_LENGTH)
label_vec = model3_cnn.predict(data_input[0].reshape(1,-1)) #can try different models
label_id = np.argmax(label_vec)
for name, ID in labels_index.items():
  if label_id == ID:
   output_name = name
   break
print("The tweet seems to be :  "+output_name)
```

```
Enter your comment : i love chatgpt
1/1 [==============================] - 0s 33ms/step
The tweet seems to be :  good
```



```
#Input a new tweet  predicting using models
output_name = ''
sample = input("Enter your comment : ")
tweet=tweet_clean(sample)
tweet_row=[tweet]
sequence_input=tokenizer.texts_to_sequences(tweet_row)
data_input=pad_sequences(sequence_input,maxlen=MAX_SEQUENCE_LENGTH)
label_vec = model3_cnn.predict(data_input[0].reshape(1,-1)) #can try different models
label_id = np.argmax(label_vec)
for name, ID in labels_index.items():
  if label_id == ID:
   output_name = name
   break
print("The tweet seems to be :  "+output_name)
```

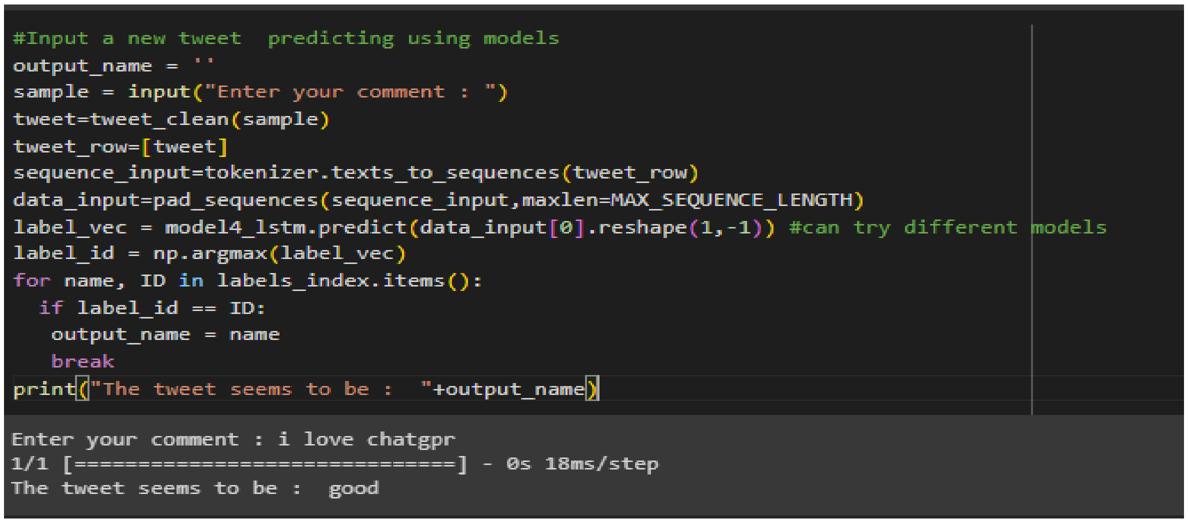
```
Enter your comment : I hate chat gpt
1/1 [==============================] - 0s 24ms/step
The tweet seems to be :  bad
```



```
#Input a new tweet  predicting using models
output_name = ''
sample = input("Enter your comment : ")
tweet=tweet_clean(sample)
tweet_row=[tweet]
sequence_input=tokenizer.texts_to_sequences(tweet_row)
data_input=pad_sequences(sequence_input,maxlen=MAX_SEQUENCE_LENGTH)
label_vec = model3_cnn.predict(data_input[0].reshape(1,-1)) #can try different models
label_id = np.argmax(label_vec)
for name, ID in labels_index.items():
  if label_id == ID:
   output_name = name
   break
print("The tweet seems to be :  "+output_name)
```

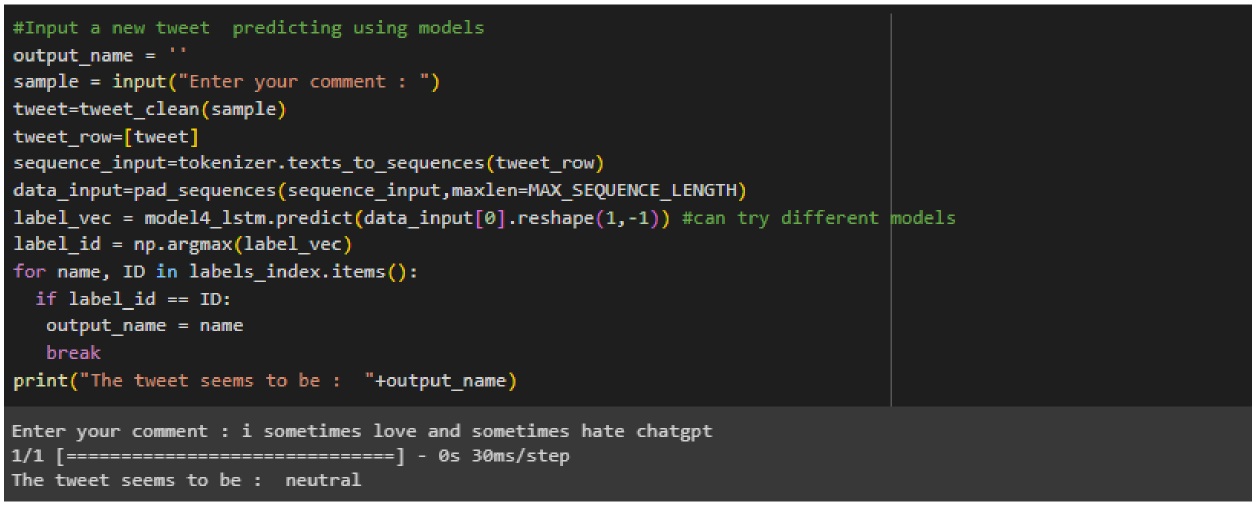
```
Enter your comment : I sometimes hate chatgpt and sometimes love it
1/1 [==============================] - 0s 19ms/step
The tweet seems to be :  neutral
```

2- Input using LSTM model



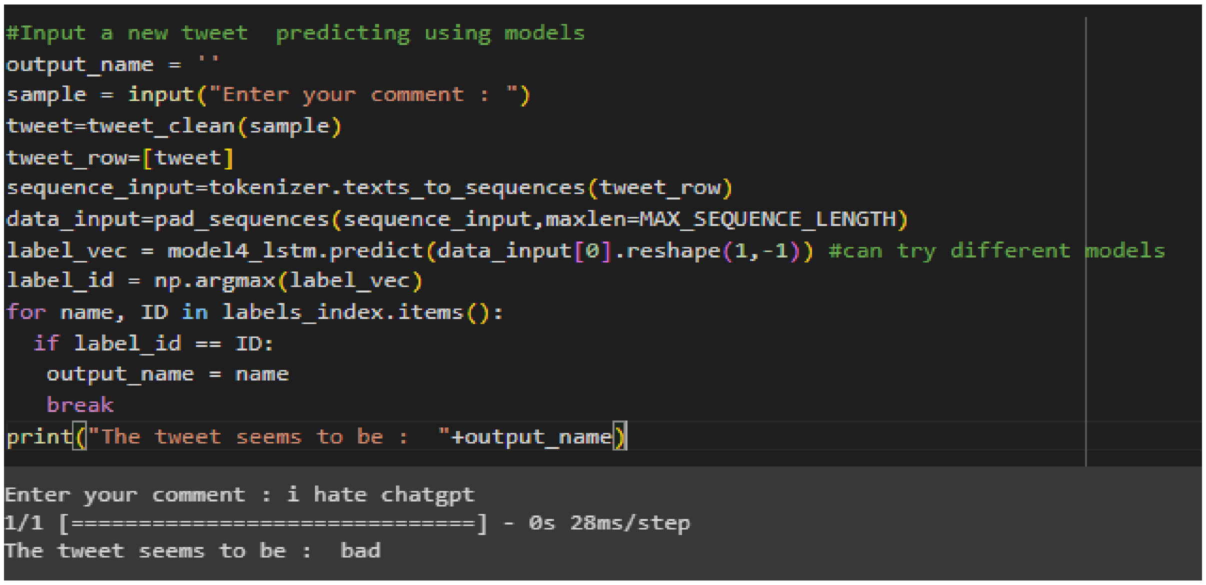
```
#Input a new tweet  predicting using models
output_name = ''
sample = input("Enter your comment : ")
tweet=tweet_clean(sample)
tweet_row=[tweet]
sequence_input=tokenizer.texts_to_sequences(tweet_row)
data_input=pad_sequences(sequence_input,maxlen=MAX_SEQUENCE_LENGTH)
label_vec = model4_lstm.predict(data_input[0].reshape(1,-1)) #can try different models
label_id = np.argmax(label_vec)
for name, ID in labels_index.items():
  if label_id == ID:
   output_name = name
   break
print("The tweet seems to be :  "+output_name)
```

```
Enter your comment : i love chatgpr
1/1 [==============================] - 0s 18ms/step
The tweet seems to be :  good
```



```
#Input a new tweet  predicting using models
output_name = ''
sample = input("Enter your comment : ")
tweet=tweet_clean(sample)
tweet_row=[tweet]
sequence_input=tokenizer.texts_to_sequences(tweet_row)
data_input=pad_sequences(sequence_input,maxlen=MAX_SEQUENCE_LENGTH)
label_vec = model4_lstm.predict(data_input[0].reshape(1,-1)) #can try different models
label_id = np.argmax(label_vec)
for name, ID in labels_index.items():
  if label_id == ID:
   output_name = name
   break
print("The tweet seems to be :  "+output_name)
```

```
Enter your comment : i sometimes love and sometimes hate chatgpt
1/1 [==============================] - 0s 30ms/step
The tweet seems to be :  neutral
```



```
#Input a new tweet  predicting using models
output_name = ''
sample = input("Enter your comment : ")
tweet=tweet_clean(sample)
tweet_row=[tweet]
sequence_input=tokenizer.texts_to_sequences(tweet_row)
data_input=pad_sequences(sequence_input,maxlen=MAX_SEQUENCE_LENGTH)
label_vec = model4_lstm.predict(data_input[0].reshape(1,-1)) #can try different models
label_id = np.argmax(label_vec)
for name, ID in labels_index.items():
  if label_id == ID:
   output_name = name
   break
print("The tweet seems to be :  "+output_name)
```

```
Enter your comment : i hate chatgpt
1/1 [==============================] - 0s 28ms/step
The tweet seems to be :  bad
```