In [8]:
```python
import pandas as pd
import numpy as np
import tensorflow as tf
import tensorflow_hub as hub
from tensorflow.keras.models import Model
```

In [9]:
```python
reviews = pd.read_csv('/content/drive/MyDrive/NLP/preprocessed.csv')
```

In [10]:
```python
reviews.head()
```

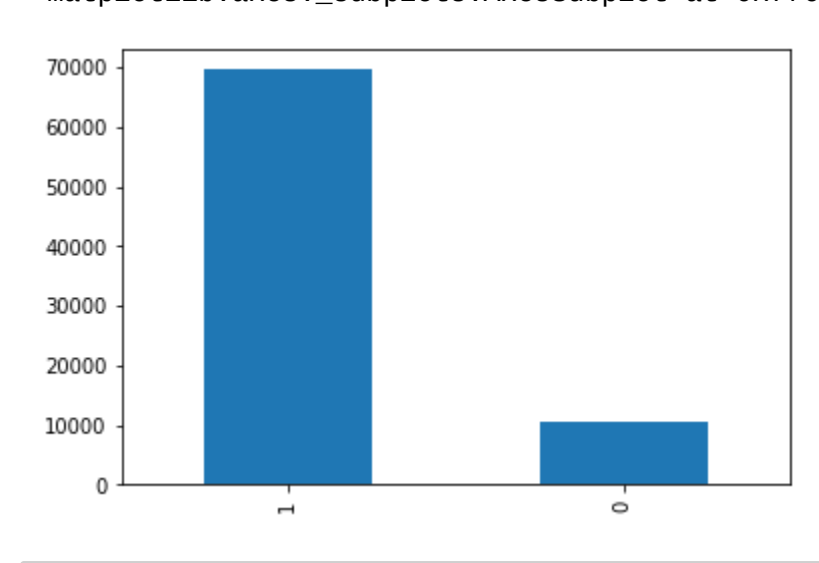
Out[10]:

	Text	Score	len
0	The tea was of great quality and it tasted lik...	1	30
1	My cat loves this. The pellets are nice and s...	1	31
2	Great product. Does not completely get rid of ...	1	41
3	This gum is my favorite! I would advise every...	1	27
4	I also found out about this product because of...	1	22

In [11]:
```python
from sklearn.model_selection import train_test_split
reviews_train,reviews_test = train_test_split(reviews,test_size=0.20, st
ratify = reviews['Score'],random_state=33)
```

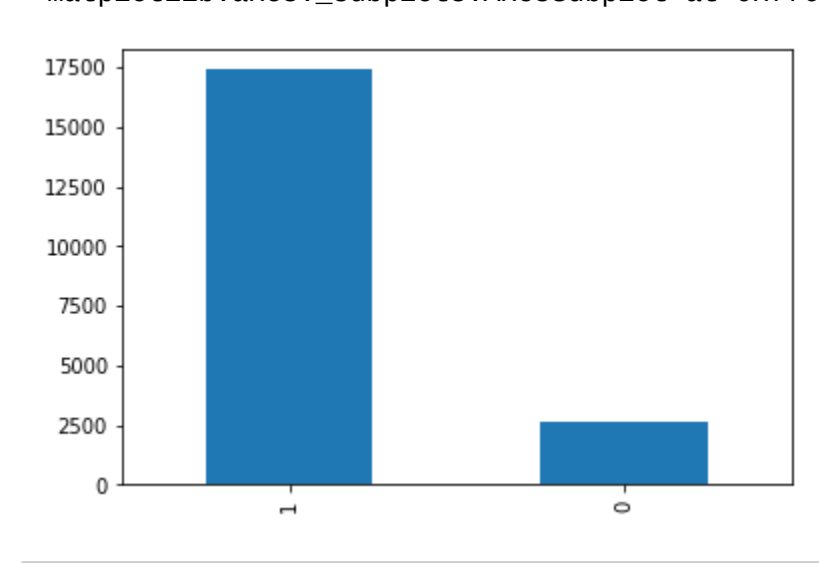
In [12]:
```python
#plot bar graphs of train and test Scores
reviews_train['Score'].value_counts().plot(kind='bar')
```

Out[12]: <matplotlib.axes._subplots.AxesSubplot at 0x7f695d330250>



In [13]:
```python
reviews_test['Score'].value_counts().plot(kind='bar')
```

Out[13]: <matplotlib.axes._subplots.AxesSubplot at 0x7f695d287c50>



In [14]:
```python
X_train,y_train = np.array(reviews_train['Text']),np.array(reviews_train
['Score'])
X_test,y_test   = np.array(reviews_test['Text']),np.array(reviews_test[
'Score'])
```

Part 2 - Creating BERT Model

In [15]:
```python
## Loading the Pretrained Model from tensorflow HUB
tf.keras.backend.clear_session()

# maximum length of a seq in the data we have, for now i am making it as
55. You can change this
max_seq_length = 128  # maximum seq length should be less than 512

#BERT takes 3 inputs

#this is input words. Sequence of words represented as integers
input_word_ids = tf.keras.layers.Input(shape=(max_seq_length,), dtype=tf
.int32, name="input_word_ids")

#mask vector if you are padding anything
input_mask = tf.keras.layers.Input(shape=(max_seq_length,), dtype=tf.int
32, name="input_mask")

#segment vectors. If you are giving only one sentence for the classifica
tion, total seg vector is 0.
#If you are giving two sentenced with [sep] token separated, first seq s
egment vectors are zeros and
#second seq segment vector are 1's
segment_ids = tf.keras.layers.Input(shape=(max_seq_length,), dtype=tf.in
t32, name="segment_ids")

#bert layer
bert_layer = hub.KerasLayer("https://tfhub.dev/tensorflow/bert_en_uncase
d_L-12_H-768_A-12/1", trainable=False)
pooled_output, sequence_output = bert_layer([input_word_ids, input_mask,
segment_ids])

#Bert model
#We are using only pooled output not sequence out.
#If you want to know about those, please read https://www.kaggle.com/que
stions-and-answers/86510
bert_model = Model(inputs=[input_word_ids, input_mask, segment_ids], out
puts=pooled_output)
```

In [16]:
```python
bert_model.summary()
```

```
Model: "model"
__________________________________________________________________________________________________
 Layer (type)                   Output Shape         Param #     Connect
ed to
==================================================================================================
 input_word_ids (InputLayer)    [(None, 128)]        0           []

 input_mask (InputLayer)        [(None, 128)]        0           []

 segment_ids (InputLayer)       [(None, 128)]        0           []

 keras_layer (KerasLayer)       [(None, 768),        109482241   ['input
_word_ids[0][0]',
                                 (None, 128, 768)]                'input
_mask[0][0]',
                                                                  'segme
nt_ids[0][0]']

==================================================================================================
Total params: 109,482,241
Trainable params: 0
Non-trainable params: 109,482,241
__________________________________________________________________________________________________
```

In [17]:
```python
bert_model.output
```

Out[17]: <KerasTensor: shape=(None, 768) dtype=float32 (created by layer 'keras_layer')>

Part 3 - Tokenization

In [18]:
```python
#getting Vocab file
vocab_file = bert_layer.resolved_object.vocab_file.asset_path.numpy()
do_lower_case = bert_layer.resolved_object.do_lower_case.numpy()
```

In [20]:
```python
! pip install sentencepiece
```

```
Collecting sentencepiece
  Downloading sentencepiece-0.1.96-cp37-cp37m-manylinux_2_17_x86_64.many
linux2014_x86_64.whl (1.2 MB)
     |████████████████████████████████| 1.2 MB 12.5 MB/s
Installing collected packages: sentencepiece
Successfully installed sentencepiece-0.1.96
```

In [21]:
```python
import tokenization
```

In [22]:
```python
tokenizer = tokenization.FullTokenizer(vocab_file,do_lower_case)
```

Grader function 3

In [23]:
```python
#it has to give no error
def grader_tokenize(tokenizer):
    out = False
    try:
        out=('[CLS]' in tokenizer.vocab) and ('[SEP]' in tokenizer.vocab
)
    except:
        out = False
    assert(out==True)
    return out
grader_tokenize(tokenizer)
```

Out[23]: True

In [24]:
```python
X_train_tokens =  list(map(lambda t: ['[CLS]'] + tokenizer.tokenize(t) +
['[SEP]'],X_train))
X_test_tokens  =  list(map(lambda t: ['[CLS]'] + tokenizer.tokenize(t) +
['[SEP]'],X_test))
```

In [25]:
```python
for i in range(len(X_train_tokens)):
  offby = max_seq_length - len(X_train_tokens[i])
  if offby>0:
    X_train_tokens[i] = np.array(X_train_tokens[i] + ['[PAD]']*offby)
  else:
    X_train_tokens[i] = np.array(X_train_tokens[i][:max_seq_length-1] +
["[SEP]"])
```

In [26]:
```python
for i in range(len(X_test_tokens)):
  offby = max_seq_length - len(X_test_tokens[i])
  if offby>0:
    X_test_tokens[i] = np.array(X_test_tokens[i] + ['[PAD]']*offby)
  else:
    X_test_tokens[i] = np.array(X_test_tokens[i][:max_seq_length-1] + ["
[SEP]"])
```

In [27]:
```python
train_tokens = np.array(list(map(tokenizer.convert_tokens_to_ids, X_trai
n_tokens)))
test_tokens  = np.array(list(map(tokenizer.convert_tokens_to_ids, X_test
_tokens)))
```

In [28]:
```python
train_masks = np.zeros((train_tokens.shape[0],max_seq_length))
test_masks  = np.zeros((test_tokens.shape[0],max_seq_length))
for k in range(len(train_tokens)):
  for j in range(len(train_tokens[k])):
    if train_tokens[k][j]!=0:
      train_masks[k][j] = 1
for m in range(len(test_tokens)):
  for l in range(len(test_tokens[m])):
    if test_tokens[m][l]!=0:
      test_masks[m][l] = 1
```

In [29]:
```python
train_segments = np.zeros((train_tokens.shape[0],train_tokens.shape[1]))
test_segments = np.zeros((test_tokens.shape[0],test_tokens.shape[1]))
```

In [30]:
```python
import pickle
pickle.dump((X_train, train_tokens, train_masks, train_segments, y_train
),open('/content/drive/MyDrive/NLP/train_data.pkl','wb'))
pickle.dump((X_test, test_tokens, test_masks,test_segments, y_test),open
('/content/drive/MyDrive/NLP/test_data.pkl','wb'))
```

In [31]:
```python
import pickle
X_train, X_train_tokens, X_train_mask, X_train_segment, y_train = pickle
.load(open("/content/drive/MyDrive/NLP/train_data.pkl", 'rb'))
X_test, X_test_tokens, X_test_mask, X_test_segment, y_test = pickle.load
(open("/content/drive/MyDrive/NLP/test_data.pkl", 'rb'))
```

Grader Function 4

In [32]:
```python
def grader_alltokens_train():
    out = False

    if type(X_train_tokens) == np.ndarray:

        temp_shapes = (X_train_tokens.shape[1]==max_seq_length) and (X_t
rain_mask.shape[1]==max_seq_length) and \
        (X_train_segment.shape[1]==max_seq_length)

        segment_temp = not np.any(X_train_segment)

        mask_temp = np.sum(X_train_mask==0) == np.sum(X_train_tokens==0)

        no_cls = np.sum(X_train_tokens==tokenizer.vocab['[CLS]'])==X_tra
in_tokens.shape[0]

        no_sep = np.sum(X_train_tokens==tokenizer.vocab['[SEP]'])==X_tra
in_tokens.shape[0]

        out = temp_shapes and segment_temp and mask_temp and no_cls and
no_sep

    else:
        print('Type of all above token arrays should be numpy array not
 list')
        out = False
    assert(out==True)
    return out

grader_alltokens_train()
```

Out[32]: True

Grader Function 5

In [33]:
```python
def grader_alltokens_test():
    out = False
    if type(X_test_tokens) == np.ndarray:

        temp_shapes = (X_test_tokens.shape[1]==max_seq_length) and (X_te
st_mask.shape[1]==max_seq_length) and \
        (X_test_segment.shape[1]==max_seq_length)

        segment_temp = not np.any(X_test_segment)

        mask_temp = np.sum(X_test_mask==0) == np.sum(X_test_tokens==0)

        no_cls = np.sum(X_test_tokens==tokenizer.vocab['[CLS]'])==X_test
_tokens.shape[0]

        no_sep = np.sum(X_test_tokens==tokenizer.vocab['[SEP]'])==X_test
_tokens.shape[0]

        out = temp_shapes and segment_temp and mask_temp and no_cls and
no_sep

    else:
        print('Type of all above token arrays should be numpy array not
 list')
        out = False
    assert(out==True)
    return out
grader_alltokens_test()
```

Out[33]: True

Part 4 - Getting Embeddings from BERT Model

In [34]:
```python
bert_model.input
```

Out[34]:
```
[<KerasTensor: shape=(None, 128) dtype=int32 (created by layer 'input_wo
rd_ids')>,
 <KerasTensor: shape=(None, 128) dtype=int32 (created by layer 'input_ma
sk')>,
 <KerasTensor: shape=(None, 128) dtype=int32 (created by layer 'segment_
ids')>]
```

In [35]:
```python
bert_model.output
```

Out[35]: <KerasTensor: shape=(None, 768) dtype=float32 (created by layer 'keras_layer')>

In [36]:
```python
X_train_pooled_output=bert_model.predict([X_train_tokens,X_train_mask,X_
train_segment])
```

In [37]:
```python
X_test_pooled_output=bert_model.predict([X_test_tokens,X_test_mask,X_tes
t_segment])
```

In [38]:
```python
pickle.dump((X_train_pooled_output, X_test_pooled_output),open('/conten
t/drive/MyDrive/NLP/final_output.pkl','wb'))
```

In [39]:
```python
X_train_pooled_output, X_test_pooled_output= pickle.load(open('/content/
drive/MyDrive/NLP/final_output.pkl', 'rb'))
```

Grader Function 6

In [40]:
```python
#now we have X_train_pooled_output, y_train
#X_test_pooled_ouput, y_test

def greader_output():
    assert(X_train_pooled_output.shape[1]==768)
    assert(len(y_train)==len(X_train_pooled_output))
    assert(X_test_pooled_output.shape[1]==768)
    assert(len(y_test)==len(X_test_pooled_output))
    assert(len(y_train.shape)==1)
    assert(len(X_train_pooled_output.shape)==2)
    assert(len(y_test.shape)==1)
    assert(len(X_test_pooled_output.shape)==2)
    return True
greader_output()
```

Out[40]: True

Part 5 - Training a NN with 768 features

In [41]:
```python
from sklearn.metrics import roc_auc_score
def auc(y_true,y_pred):
  return roc_auc_score(y_true,y_pred,average='micro')
```

In [42]:
```python
from tensorflow.keras.layers import Input, Dense, Activation, Dropout, L
STM
from tensorflow.keras.models import Model
```

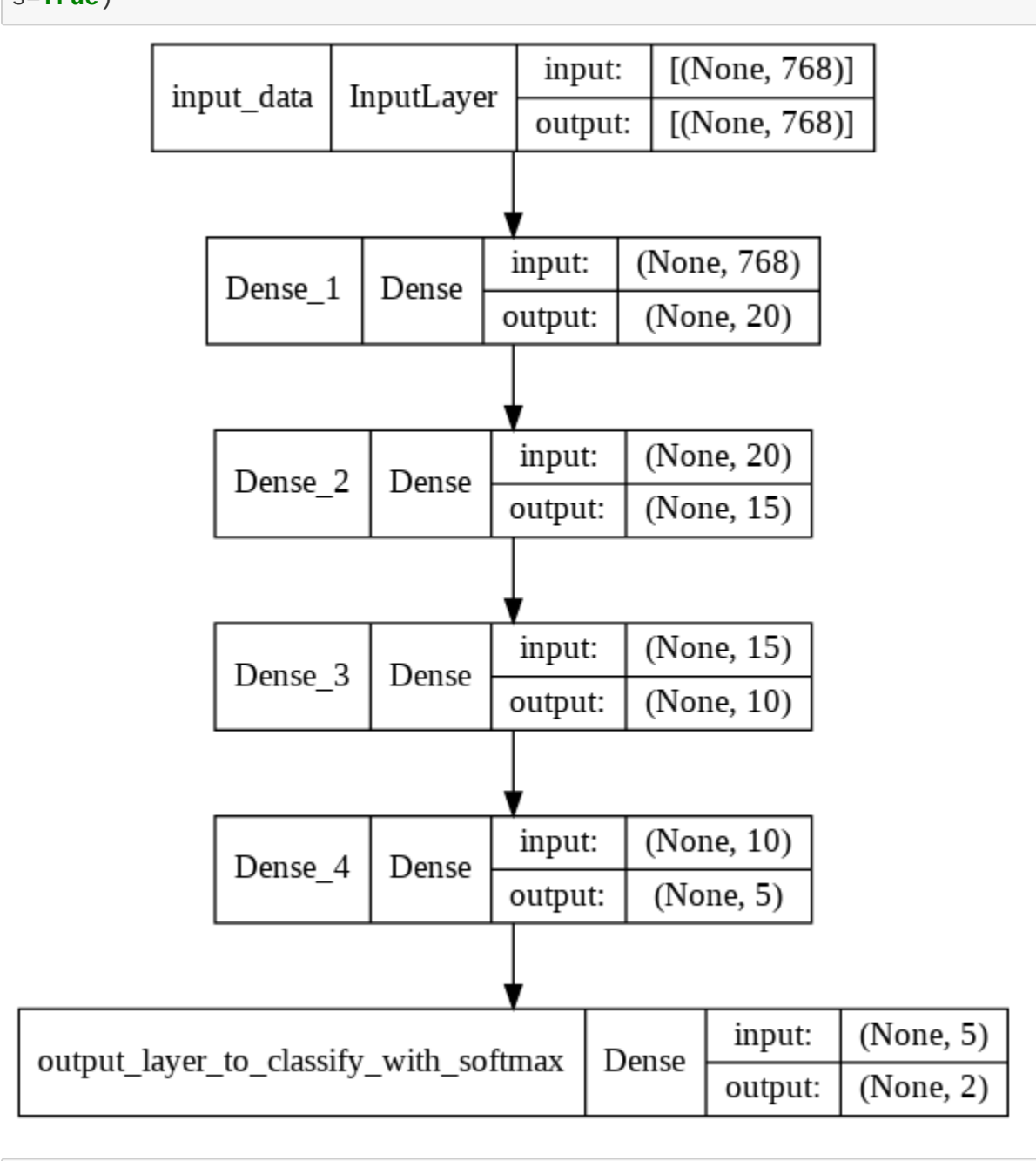
In [43]:
```python
import tensorflow as tf
y_train = tf.keras.utils.to_categorical(y_train,2)
y_test  = tf.keras.utils.to_categorical(y_test,2)
```

In [52]:
```python
initializer                    = tf.keras.initializers.he_normal()
input_layer                    = Input(shape=(X_train_pooled_output.shape
[1],),name='input_data')
Layer1                         = Dense(20, activation='relu',kernel_initi
alizer=initializer, name = 'Dense_1')(input_layer)
Layer2                         = Dense(15, activation='relu',kernel_initi
alizer=initializer, name = 'Dense_2')(Layer1)
Layer3                         = Dense(10, activation='relu',kernel_initi
alizer=initializer, name = 'Dense_3')(Layer2)
Layer4                         = Dense(5, activation='relu',kernel_initia
lizer=initializer, name = 'Dense_4')(Layer3)
output                         = Dense(2, activation='softmax',kernel_ini
tializer=initializer, name = 'output_layer_to_classify_with_softmax')(La
yer4)

model                          = Model(inputs=input_layer,outputs=output)
model.compile(optimizer=tf.keras.optimizers.Adam(learning_rate=0.0001, b
eta_1=0.9, beta_2=0.999, epsilon=1e-08),
               loss='categorical_crossentropy',metrics=[auc],run_eagerly=
True)
model.summary()
```

```
Model: "model_3"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_data (InputLayer)     [(None, 768)]             0

 Dense_1 (Dense)             (None, 20)                15380

 Dense_2 (Dense)             (None, 15)                315

 Dense_3 (Dense)             (None, 10)                160

 Dense_4 (Dense)             (None, 5)                 55

 output_layer_to_classify_wi  (None, 2)                12
 th_softmax (Dense)

=================================================================
Total params: 15,922
Trainable params: 15,922
Non-trainable params: 0
_________________________________________________________________
```

In [53]:
```python
from tensorflow.keras.utils import plot_model
plot_model(model, to_file='NN for NLP Transfer Learning.png', show_shape
s=True)
```

Out[53]:



In [54]:
```python
from tensorflow.keras.callbacks import TensorBoard,ModelCheckpoint,Early
Stopping,ReduceLROnPlateau
import datetime
log_dir = "logs/fit/" + datetime.datetime.now().strftime("%Y%m%d-%H%M%S"
)
tensorboard_callback = TensorBoard(log_dir=log_dir,histogram_freq=1)
es = EarlyStopping(monitor='val_auc', mode='max', verbose=1,patience=5)
mc = ModelCheckpoint('NLP.h5', monitor='val_auc', mode='max', save_best_
only=True,verbose=1)
reduce_lr = ReduceLROnPlateau(monitor='val_auc', factor=0.8, patience=2,
verbose=0)
callback_list = [tensorboard_callback,es,mc,reduce_lr]
```

In [55]:
```python
%load_ext tensorboard
```

```
The tensorboard extension is already loaded. To reload it, use:
  %reload_ext tensorboard
```

In [56]:
```python
!rm -rf ./logs/
```

In [57]:
```python
model.fit(X_train_pooled_output,y_train,epochs=70,batch_size=512,
          validation_data=(X_test_pooled_output,y_test),callbacks = call
back_list)
```

```
Epoch 1/70
157/157 [==============================] - ETA: 0s - loss: 0.3900 - auc:
0.8890
Epoch 00001: val_auc improved from -inf to 0.91188, saving model to NLP.
h5
157/157 [==============================] - 28s 150ms/step - loss: 0.3900
- auc: 0.8890 - val_loss: 0.3687 - val_auc: 0.9119 - lr: 1.0000e-04
Epoch 2/70
157/157 [==============================] - ETA: 0s - loss: 0.3534 - auc:
0.9291
Epoch 00002: val_auc improved from 0.91188 to 0.93778, saving model to N
LP.h5
157/157 [==============================] - 24s 155ms/step - loss: 0.3534
- auc: 0.9291 - val_loss: 0.3378 - val_auc: 0.9378 - lr: 1.0000e-04
Epoch 3/70
157/157 [==============================] - ETA: 0s - loss: 0.3136 - auc:
0.9467
Epoch 00003: val_auc improved from 0.93778 to 0.95035, saving model to N
LP.h5
157/157 [==============================] - 23s 148ms/step - loss: 0.3136
- auc: 0.9467 - val_loss: 0.2970 - val_auc: 0.9504 - lr: 1.0000e-04
Epoch 4/70
157/157 [==============================] - ETA: 0s - loss: 0.2847 - auc:
0.9554
Epoch 00004: val_auc improved from 0.95035 to 0.95708, saving model to N
LP.h5
157/157 [==============================] - 24s 155ms/step - loss: 0.2847
- auc: 0.9554 - val_loss: 0.2762 - val_auc: 0.9571 - lr: 8.0000e-05
Epoch 5/70
157/157 [==============================] - ETA: 0s - loss: 0.2657 - auc:
0.9607
Epoch 00005: val_auc improved from 0.95708 to 0.96264, saving model to N
LP.h5
157/157 [==============================] - 23s 148ms/step - loss: 0.2657
- auc: 0.9607 - val_loss: 0.2615 - val_auc: 0.9626 - lr: 8.0000e-05
Epoch 6/70
157/157 [==============================] - ETA: 0s - loss: 0.2505 - auc:
0.9648
Epoch 00006: val_auc improved from 0.96264 to 0.96515, saving model to N
LP.h5
157/157 [==============================] - 23s 147ms/step - loss: 0.2505
- auc: 0.9648 - val_loss: 0.2462 - val_auc: 0.9651 - lr: 6.4000e-05
Epoch 7/70
157/157 [==============================] - ETA: 0s - loss: 0.2387 - auc:
0.9677
Epoch 00007: val_auc improved from 0.96515 to 0.96743, saving model to N
LP.h5
157/157 [==============================] - 24s 154ms/step - loss: 0.2387
- auc: 0.9677 - val_loss: 0.2357 - val_auc: 0.9674 - lr: 6.4000e-05
Epoch 8/70
157/157 [==============================] - ETA: 0s - loss: 0.2294 - auc:
0.9697
Epoch 00008: val_auc improved from 0.96743 to 0.96992, saving model to N
LP.h5
157/157 [==============================] - 24s 155ms/step - loss: 0.2294
- auc: 0.9697 - val_loss: 0.2285 - val_auc: 0.9699 - lr: 5.1200e-05
Epoch 9/70
157/157 [==============================] - ETA: 0s - loss: 0.2231 - auc:
0.9715
Epoch 00009: val_auc did not improve from 0.96992
157/157 [==============================] - 24s 155ms/step - loss: 0.2231
- auc: 0.9715 - val_loss: 0.2257 - val_auc: 0.9697 - lr: 5.1200e-05
Epoch 10/70
157/157 [==============================] - ETA: 0s - loss: 0.2176 - auc:
0.9725
Epoch 00010: val_auc improved from 0.96992 to 0.97185, saving model to N
LP.h5
157/157 [==============================] - 24s 154ms/step - loss: 0.2176
- auc: 0.9725 - val_loss: 0.2181 - val_auc: 0.9718 - lr: 4.0960e-05
Epoch 11/70
157/157 [==============================] - ETA: 0s - loss: 0.2132 - auc:
0.9734
Epoch 00011: val_auc improved from 0.97185 to 0.97338, saving model to N
LP.h5
157/157 [==============================] - 24s 154ms/step - loss: 0.2132
- auc: 0.9734 - val_loss: 0.2145 - val_auc: 0.9734 - lr: 4.0960e-05
Epoch 12/70
157/157 [==============================] - ETA: 0s - loss: 0.2100 - auc:
0.9741
Epoch 00012: val_auc improved from 0.97338 to 0.97421, saving model to N
LP.h5
157/157 [==============================] - 24s 155ms/step - loss: 0.2100
- auc: 0.9741 - val_loss: 0.2127 - val_auc: 0.9742 - lr: 3.2768e-05
Epoch 13/70
157/157 [==============================] - ETA: 0s - loss: 0.2073 - auc:
0.9748
Epoch 00013: val_auc improved from 0.97421 to 0.97438, saving model to N
LP.h5
157/157 [==============================] - 24s 154ms/step - loss: 0.2073
- auc: 0.9748 - val_loss: 0.2087 - val_auc: 0.9744 - lr: 3.2768e-05
Epoch 14/70
157/157 [==============================] - ETA: 0s - loss: 0.2048 - auc:
0.9754
Epoch 00014: val_auc did not improve from 0.97438
157/157 [==============================] - 23s 148ms/step - loss: 0.2048
- auc: 0.9754 - val_loss: 0.2078 - val_auc: 0.9740 - lr: 2.6214e-05
Epoch 15/70
157/157 [==============================] - ETA: 0s - loss: 0.2032 - auc:
0.9758
Epoch 00015: val_auc improved from 0.97438 to 0.97538, saving model to N
LP.h5
157/157 [==============================] - 24s 154ms/step - loss: 0.2032
- auc: 0.9758 - val_loss: 0.2065 - val_auc: 0.9754 - lr: 2.6214e-05
Epoch 16/70
157/157 [==============================] - ETA: 0s - loss: 0.2015 - auc:
0.9761
Epoch 00016: val_auc improved from 0.97538 to 0.97545, saving model to N
LP.h5
157/157 [==============================] - 23s 148ms/step - loss: 0.2015
- auc: 0.9761 - val_loss: 0.2039 - val_auc: 0.9755 - lr: 2.0972e-05
Epoch 17/70
157/157 [==============================] - ETA: 0s - loss: 0.2000 - auc:
0.9764
Epoch 00017: val_auc did not improve from 0.97545
157/157 [==============================] - 23s 148ms/step - loss: 0.2000
- auc: 0.9764 - val_loss: 0.2027 - val_auc: 0.9754 - lr: 2.0972e-05
Epoch 18/70
157/157 [==============================] - ETA: 0s - loss: 0.1988 - auc:
0.9766
Epoch 00018: val_auc improved from 0.97545 to 0.97571, saving model to N
LP.h5
157/157 [==============================] - 24s 155ms/step - loss: 0.1988
- auc: 0.9766 - val_loss: 0.2017 - val_auc: 0.9757 - lr: 1.6777e-05
Epoch 19/70
157/157 [==============================] - ETA: 0s - loss: 0.1978 - auc:
0.9767
Epoch 00019: val_auc improved from 0.97571 to 0.97596, saving model to N
LP.h5
157/157 [==============================] - 23s 149ms/step - loss: 0.1978
- auc: 0.9767 - val_loss: 0.2007 - val_auc: 0.9760 - lr: 1.6777e-05
Epoch 20/70
157/157 [==============================] - ETA: 0s - loss: 0.1969 - auc:
0.9769
Epoch 00020: val_auc did not improve from 0.97596
157/157 [==============================] - 24s 155ms/step - loss: 0.1969
- auc: 0.9769 - val_loss: 0.2005 - val_auc: 0.9757 - lr: 1.3422e-05
Epoch 21/70
157/157 [==============================] - ETA: 0s - loss: 0.1963 - auc:
0.9771
Epoch 00021: val_auc improved from 0.97596 to 0.97620, saving model to N
LP.h5
157/157 [==============================] - 24s 156ms/step - loss: 0.1963
- auc: 0.9771 - val_loss: 0.1993 - val_auc: 0.9762 - lr: 1.3422e-05
Epoch 22/70
157/157 [==============================] - ETA: 0s - loss: 0.1956 - auc:
0.9772
Epoch 00022: val_auc improved from 0.97620 to 0.97627, saving model to N
LP.h5
157/157 [==============================] - 24s 154ms/step - loss: 0.1956
- auc: 0.9772 - val_loss: 0.1988 - val_auc: 0.9763 - lr: 1.0737e-05
Epoch 23/70
157/157 [==============================] - ETA: 0s - loss: 0.1949 - auc:
0.9773
Epoch 00023: val_auc did not improve from 0.97627
157/157 [==============================] - 23s 150ms/step - loss: 0.1949
- auc: 0.9773 - val_loss: 0.1986 - val_auc: 0.9762 - lr: 1.0737e-05
Epoch 24/70
157/157 [==============================] - ETA: 0s - loss: 0.1944 - auc:
0.9774
Epoch 00024: val_auc improved from 0.97627 to 0.97649, saving model to N
LP.h5
157/157 [==============================] - 24s 156ms/step - loss: 0.1944
- auc: 0.9774 - val_loss: 0.1979 - val_auc: 0.9765 - lr: 8.5899e-06
Epoch 25/70
157/157 [==============================] - ETA: 0s - loss: 0.1940 - auc:
0.9776
Epoch 00025: val_auc improved from 0.97649 to 0.97663, saving model to N
LP.h5
157/157 [==============================] - 24s 155ms/step - loss: 0.1940
- auc: 0.9776 - val_loss: 0.1975 - val_auc: 0.9766 - lr: 8.5899e-06
Epoch 26/70
157/157 [==============================] - ETA: 0s - loss: 0.1935 - auc:
0.9776
Epoch 00026: val_auc improved from 0.97663 to 0.97679, saving model to N
LP.h5
157/157 [==============================] - 24s 155ms/step - loss: 0.1935
- auc: 0.9776 - val_loss: 0.1973 - val_auc: 0.9768 - lr: 6.8719e-06
Epoch 27/70
157/157 [==============================] - ETA: 0s - loss: 0.1933 - auc:
0.9777
Epoch 00027: val_auc improved from 0.97679 to 0.97689, saving model to N
LP.h5
157/157 [==============================] - 23s 149ms/step - loss: 0.1933
- auc: 0.9777 - val_loss: 0.1972 - val_auc: 0.9769 - lr: 6.8719e-06
Epoch 28/70
157/157 [==============================] - ETA: 0s - loss: 0.1929 - auc:
0.9777
Epoch 00028: val_auc did not improve from 0.97689
157/157 [==============================] - 24s 154ms/step - loss: 0.1929
- auc: 0.9777 - val_loss: 0.1969 - val_auc: 0.9766 - lr: 5.4976e-06
Epoch 29/70
157/157 [==============================] - ETA: 0s - loss: 0.1927 - auc:
0.9778
Epoch 00029: val_auc did not improve from 0.97689
157/157 [==============================] - 24s 155ms/step - loss: 0.1927
- auc: 0.9778 - val_loss: 0.1966 - val_auc: 0.9766 - lr: 5.4976e-06
Epoch 30/70
157/157 [==============================] - ETA: 0s - loss: 0.1925 - auc:
0.9778
Epoch 00030: val_auc did not improve from 0.97689
157/157 [==============================] - 24s 154ms/step - loss: 0.1925
- auc: 0.9778 - val_loss: 0.1963 - val_auc: 0.9767 - lr: 4.3980e-06
Epoch 31/70
157/157 [==============================] - ETA: 0s - loss: 0.1922 - auc:
0.9777
Epoch 00031: val_auc did not improve from 0.97689
157/157 [==============================] - 23s 147ms/step - loss: 0.1922
- auc: 0.9777 - val_loss: 0.1960 - val_auc: 0.9768 - lr: 4.3980e-06
Epoch 32/70
157/157 [==============================] - ETA: 0s - loss: 0.1921 - auc:
0.9779
Epoch 00032: val_auc did not improve from 0.97689
157/157 [==============================] - 23s 148ms/step - loss: 0.1921
- auc: 0.9779 - val_loss: 0.1959 - val_auc: 0.9768 - lr: 3.5184e-06
Epoch 00032: early stopping
```

Out[57]: <keras.callbacks.History at 0x7f68e2e45d90>

In [58]:
```python
%tensorboard --logdir logs/fit
```

In [59]:
```python
! unzip /content/test.csv-20211125T113253Z-001.zip
```

```
Archive:  /content/test.csv-20211125T113253Z-001.zip
  inflating: test.csv
```

In [60]:
```
test_df = pd.read_csv('/content/test.csv')
```

In [61]:
```
test_df.head()
```

Out[61]:

	Text
0	Just opened Greenies Joint Care (individually ...
1	This product rocks :) My mom was very happy w/...
2	The product was fine, but the cost of shipping...
3	I love this soup. It's great as part of a meal...
4	Getting ready to order again. These are great ...

In [62]:
```
import re as re
def remove_tags(string):
    result = re.sub('<.*?>','',string)
    return result
```

In [63]:
```
def transfer(df,model,max_seq_length,bert_model):
  df['Text']  = df['Text'].apply(lambda cw : remove_tags(cw))
  test        = np.array(df['Text'])
  pre_token   = list(map(lambda t: ['[CLS]'] + tokenizer.tokenize(t) + [
'[SEP]'],test))
  for i in range(len(pre_token)):
    offby     = max_seq_length - len(pre_token[i])
    if offby>0:
      pre_token[i] = np.array(pre_token[i] + ['[PAD]']*offby)
    else:
      pre_token[i] = np.array(pre_token[i][:max_seq_length-1] + ["[SEP]"
])
  tokens_array   = np.array(list(map(tokenizer.convert_tokens_to_ids, pr
e_token)))
  masks_array    = np.zeros((tokens_array.shape[0],max_seq_length))
  for k in range(len(tokens_array)):
    for j in range(len(tokens_array[k])):
      if tokens_array[k][j]!=0:
        masks_array[k][j] = 1
  segments_array = np.zeros((tokens_array.shape[0],tokens_array.shape[1
]))
  pooled_output  = bert_model.predict([tokens_array,masks_array,segments
_array])
  predictions    =   model.predict(pooled_output)
  # Generate arg maxes for predictions
  classes        = np.argmax(predictions, axis = 1)
  unique, counts = np.unique(classes, return_counts=True)
  return dict(zip(unique, counts))
```

In [64]:
```
# count of datapoints classified as 1 or 0 in the form of dictionary
out  = transfer(test_df,model,max_seq_length,bert_model)
print(out)
```

```
{0: 36, 1: 316}
```

Observations:

1. Out of 352 datapoints in the test dataset, model classified 36 points as 0 and 316 datapoints as 1.
   The following are the steps performed while solving this assignment:
2. Preprocessing the reviews csv file by removing all the html tags, assigning the score values as 1 if score is >3 and score value as 0 if score is <=2. Also dropping all the rows if the score value is 3.
3. Splitting into train and test data which gives X_train,X_test,y_train and y_test and checking if train and test data have the same distribution.
4. Creating tokens, masked array and segment arrays for X_train and X_test.
5. Getting Embeddings from BERT Model and extracting pooled_output, out of two outputs pooled_output, sequence_output which it produces.
6. Training the Neural Network using the embeddings obtained using BERT model without overfitting and underfitting.
7. Then, predicting the output for the test data which has not seen by the model.
8. Extracting the number of occurences of classes 0 or 1.