

Dataset: Link(<https://www.kaggle.com/snap/amazon-fine-food-reviews/data>)

```
import keras
import pickle
import pandas as pd
import numpy as np
import tensorflow as tf
import tensorflow_hub as hub
from keras import layers, Model
import torch
from keras.preprocessing.sequence import pad_sequences
import warnings
warnings.filterwarnings('ignore')
```

```
print(tf.test.gpu_device_name())
```

```
/device:GPU:0
```

```
def grader_tf_version():
    assert((tf.__version__)>'2')
    return True
grader_tf_version()
```

```
True
```

```
!wget --header="Host: storage.googleapis.com" --header="User-Agent: Mozilla/5.0 (Windows N
```

```
!unzip /content/archive.zip
```

```
#Read the dataset - Amazon fine food reviews
reviews = pd.read_csv("/content/Reviews.csv")
#check the info of the dataset
reviews.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 568454 entries, 0 to 568453
Data columns (total 10 columns):
#   Column                                Non-Null Count  Dtype
---  -
0   Id                                     568454 non-null  int64
1   ProductId                             568454 non-null  object
2   UserId                                 568454 non-null  object
3   ProfileName                           568438 non-null  object
4   HelpfulnessNumerator                   568454 non-null  int64
5   HelpfulnessDenominator                 568454 non-null  int64
6   Score                                  568454 non-null  int64
7   Time                                  568454 non-null  int64
8   Summary                                568427 non-null  object
9   Text                                   568454 non-null  object
dtypes: int64(5), object(5)
memory usage: 43.4+ MB
```

```
review = reviews.drop(['Id', 'ProductId', 'UserId', 'ProfileName', 'HelpfulnessNumerator', 'Hel
review.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 568454 entries, 0 to 568453
Data columns (total 2 columns):
#   Column  Non-Null Count  Dtype
---  -
0    Score   568454 non-null    int64
1    Text     568454 non-null    object
dtypes: int64(1), object(1)
memory usage: 8.7+ MB
```

```
reviews = review[review['Score']!=3]
reviews.head(2)
```

|   | Score | Text  |
|---|-------|---|
| 0 | 5     | I have bought several of the Vitality canned d... |
| 1 | 1     | Product arrived labeled as Jumbo Salted Peanut... |

```
scores=[]
for score in reviews['Score']:
    if score>3:
        scores.append(1)
    if score<=2:
        scores.append(0)
reviews['Score']=scores
reviews.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 525814 entries, 0 to 568453
Data columns (total 2 columns):
#   Column  Non-Null Count  Dtype
---  -
0    Score   525814 non-null    int64
1    Text     525814 non-null    object
dtypes: int64(1), object(1)
memory usage: 12.0+ MB
```

```
def grader_reviews():
    temp_shape = (reviews.shape == (525814, 2)) and (reviews.Score.value_counts()[1]==4437)
    assert(temp_shape == True)
    return True
grader_reviews()
```

True

```
def get_wordlen(x):
    return len(x.split())
reviews['len'] = reviews.Text.apply(get_wordlen)
reviews = reviews[reviews.len<50]
reviews = reviews.sample(n=100000, random_state=30)
```

```
def remove_html(text):
    for i in range(len(text)):
        html = re.compile('<.*?>')
        text[i] = re.sub(html, ' ',text[i])
    return text
```

```
reviews['Text']=remove_html(reviews['Text'].values)
reviews.to_csv('/content/drive/MyDrive/Assignment/NLP with Transfer Learning/preprocessed.
```

```
y = reviews['Score']
X = reviews['Text']
```

```
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test = train_test_split(X,y, test_size=0.2 , stratify = y, random
X_train.head(2)
```

```
33523    I had never tried this brand before, so I was ...
10855    I love these for a snack. I get a nice taste o...
Name: Text, dtype: object
```

```
y_train.value_counts()
```

```
1    69603
0    10397
Name: Score, dtype: int64
```

```
import seaborn as sns
sns.set_theme(style="darkgrid")
sns.countplot(y_train).set_title('Train_Data')
```

```
Text(0.5, 1.0, 'Train_Data')
```



```
sns.set_theme(style="darkgrid")
sns.countplot(y_test).set_title('Test_Data')
```

Text(0.5, 1.0, 'Test\_Data')



```
length=[]
for t in X_train:
    length.append(len(t.split()))
max_seq_length = max(length)
print(max_seq_length)
```

49

## 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  
max\_seq\_length = 49

#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.int32, name="input_mask")
```

#segment vectors. If you are giving only one sentence for the classification, total segment vectors are 1

#If you are giving two sentences with [sep] token separated, first seq segment vectors are 0's

#second seq segment vector are 1's

```
segment_ids = tf.keras.layers.Input(shape=(max_seq_length,), dtype=tf.int32, name="segment_ids")
```

#bert layer

```
bert_layer = hub.KerasLayer("https://tfhub.dev/tensorflow/bert_en_uncased_L-12_H-768_A-12/")
pooled_output, sequence_output = bert_layer([input_word_ids, input_mask, segment_ids])
```

#Bert model

#We are using only pooled output not sequence output.

#If you want to know about those, please read <https://www.kaggle.com/questions-and-answers/bert-model>

```
bert_model = Model(inputs=[input_word_ids, input_mask, segment_ids], outputs=pooled_output)
```

```
bert_model.summary()
```

```
Model: "model"
```

| Layer (type)                      | Output Shape | Param # | Connected to |
|-----------------------------------|--------------|---------|--------------|
| =====                             |              |         |              |
| Total params: 109,482,241         |              |         |              |
| Trainable params: 0               |              |         |              |
| Non-trainable params: 109,482,241 |              |         |              |



```
bert_model.output
```

```
<KerasTensor: shape=(None, 768) dtype=float32 (created by layer 'keras_layer')>
```

```
vocab_file = bert_layer.resolved_object.vocab_file.asset_path.numpy()
do_lower_case = bert_layer.resolved_object.do_lower_case.numpy()
```

```
!pip install bert-for-tf2
from bert import bert_tokenization
```

```
Installing collected packages: py-params, params-flow, bert-for-tf2
Successfully installed bert-for-tf2-0.14.9 params-flow-0.8.2 py-params-0.10.2
```

```
tokenizer = bert_tokenization.FullTokenizer(vocab_file,do_lower_case)
```

```
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)
```

```
True
```

```
X_train_tokens,X_train_mask, X_train_segment=[],[],[]
for i in range(len(X_train)):
    tokens = tokenizer.tokenize(X_train.values[i])
    if len(tokens)<(max_seq_length-2):
        tokens=tokens
    else:
        tokens = tokens[0:(max_seq_length-2)]
    tokens = ['[CLS]',*tokens,'[SEP]']
    X_train_tokens.append(np.array(tokenizer.convert_tokens_to_ids(tokens)))
    X_train_mask.append(np.array([1]*len(tokens)+[0]*(max_seq_length-len(tokens))))
    X_train_segment.append([0]*max_seq_length)
X_train_token=pad_sequences(X_train_tokens , maxlen=max_seq_length, padding='post')
X_train_tokens = np.array(X_train_token)
```

```

X_train_mask=np.array(X_train_mask)
X_train_segment = np.array(X_train_segment)

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_train_mask.shape[1]
        (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_train_tokens.shape[0]

        no_sep = np.sum(X_train_tokens==tokenizer.vocab['[SEP]'])==X_train_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()
```

```
True
```

```

X_test_tokens,X_test_mask, X_test_segment=[],[],[]
for i in range(len(X_test)):
    tokens = tokenizer.tokenize(X_test.values[i])
    if len(tokens)<(max_seq_length-2):
        tokens=tokens
    else:
        tokens = tokens[0:(max_seq_length-2)]
        tokens = ['[CLS]',*tokens,'[SEP]']
        X_test_tokens.append(np.array(tokenizer.convert_tokens_to_ids(tokens)))
        X_test_mask.append(np.array([1]*len(tokens)+[0]*(max_seq_length-len(tokens))))
        X_test_segment.append([0]*max_seq_length)
X_test_token=pad_sequences(X_test_tokens , maxlen=max_seq_length, padding='post')
X_test_tokens = np.array(X_test_token)
X_test_mask=np.array(X_test_mask)
X_test_segment = np.array(X_test_segment)

```

```

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_test_mask.shape[1]==
        (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()

```

☞ True

bert\_model.input

```

[<KerasTensor: shape=(None, 49) dtype=int32 (created by layer 'input_word_ids')>,
 <KerasTensor: shape=(None, 49) dtype=int32 (created by layer 'input_mask')>,
 <KerasTensor: shape=(None, 49) dtype=int32 (created by layer 'segment_ids')>]

```

bert\_model.output

```

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

```

```

X_train_pooled_output=bert_model.predict([X_train_tokens,X_train_mask,X_train_segment])

```

```

X_test_pooled_output=bert_model.predict([X_test_tokens,X_test_mask,X_test_segment])

```

```

pickle.dump((X_train_pooled_output, X_test_pooled_output),open('/content/drive/MyDrive/Ass
X_train_pooled_output, X_test_pooled_output= pickle.load(open('/content/drive/MyDrive/Assi

```

```

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
grader_output()

```

True

```

from keras.utils import np_utils

```

```

from keras.utils.np_utils import to_categorical
y_tr = to_categorical(y_train)
y_te = to_categorical(y_test)

from sklearn.metrics import roc_auc_score
def auc( y_true, y_pred ) :
    score = tf.py_function( lambda y_true, y_pred : roc_auc_score( y_true, y_pred, average
        [y_true, y_pred],
        'float32',
        name='sklearnAUC' )
    return score

pooled_input = keras.Input(shape=(768,) , name='pooled')
model = keras.layers.Dense(512, activation='relu')(pooled_input)
model = keras.layers.Dropout(0.1)(model)
output = keras.layers.Dense(2 , activation='softmax' , name='class')(model)

model = keras.Model(inputs=[pooled_input],outputs=[output])
model.compile(optimizer=keras.optimizers.Adam(),loss='categorical_crossentropy',metrics=['

%load_ext tensorboard
from keras.callbacks import TensorBoard
import datetime
logs = "logs/fit/" + datetime.datetime.now().strftime('%Y%m%d-%H%M%S')
tensor = TensorBoard(log_dir='logs',histogram_freq=1,write_graph=True,write_grads=True)

WARNING:tensorflow:`write_grads` will be ignored in TensorFlow 2.0 for the `TensorBoard`
WARNING:tensorflow:`write_grads` will be ignored in TensorFlow 2.0 for the `TensorBoard`

from keras.callbacks import EarlyStopping,ModelCheckpoint
estop = EarlyStopping( monitor='val_loss',patience=5 )
filepath='/content/drive/MyDrive/Assignment/NLP with Transfer Learning/Model.hd5f'
checkpoint = ModelCheckpoint(filepath=filepath,monitor='val_loss',save_best_only=True , mo

model.fit(X_train_pooled_output,y_tr,validation_split=0.2,batch_size=128,epochs=50,callbacks=[estop,tensorboard])
:tensorflow:Assets written to: /content/drive/MyDrive/Assignment/NLP with Transfer Learning/Model
:tensorflow:Assets written to: /content/drive/MyDrive/Assignment/NLP with Transfer Learning/Model
h 26/50
500 [=====] - 4s 9ms/step - loss: 0.1796 - accuracy: 0.926
h 27/50

500 [=====] - 5s 10ms/step - loss: 0.1761 - accuracy: 0.92
h 28/50
500 [=====] - 5s 9ms/step - loss: 0.1779 - accuracy: 0.928
:tensorflow:Assets written to: /content/drive/MyDrive/Assignment/NLP with Transfer Learning/Model
:tensorflow:Assets written to: /content/drive/MyDrive/Assignment/NLP with Transfer Learning/Model
h 29/50
500 [=====] - 5s 10ms/step - loss: 0.1750 - accuracy: 0.92
h 30/50
500 [=====] - 5s 10ms/step - loss: 0.1748 - accuracy: 0.92
h 31/50
500 [=====] - 5s 9ms/step - loss: 0.1808 - accuracy: 0.926
h 32/50

```



```

500 [=====] - 5s 10ms/step - loss: 0.1702 - accuracy: 0.93
:tensorflow:Assets written to: /content/drive/MyDrive/Assignment/NLP with Transfer
:tensorflow:Assets written to: /content/drive/MyDrive/Assignment/NLP with Transfer
h 33/50
500 [=====] - 5s 9ms/step - loss: 0.1791 - accuracy: 0.926
h 34/50
500 [=====] - 4s 9ms/step - loss: 0.1724 - accuracy: 0.929
h 35/50
500 [=====] - 5s 10ms/step - loss: 0.1748 - accuracy: 0.92
:tensorflow:Assets written to: /content/drive/MyDrive/Assignment/NLP with Transfer
:tensorflow:Assets written to: /content/drive/MyDrive/Assignment/NLP with Transfer
h 36/50
500 [=====] - 5s 10ms/step - loss: 0.1696 - accuracy: 0.93
h 37/50
500 [=====] - 5s 9ms/step - loss: 0.1757 - accuracy: 0.926
h 38/50
500 [=====] - 4s 9ms/step - loss: 0.1728 - accuracy: 0.930
h 39/50
500 [=====] - 4s 8ms/step - loss: 0.1813 - accuracy: 0.926
:tensorflow:Assets written to: /content/drive/MyDrive/Assignment/NLP with Transfer
:tensorflow:Assets written to: /content/drive/MyDrive/Assignment/NLP with Transfer
h 40/50
500 [=====] - 5s 10ms/step - loss: 0.1765 - accuracy: 0.92
:tensorflow:Assets written to: /content/drive/MyDrive/Assignment/NLP with Transfer
:tensorflow:Assets written to: /content/drive/MyDrive/Assignment/NLP with Transfer
h 41/50
500 [=====] - 5s 10ms/step - loss: 0.1752 - accuracy: 0.92
h 42/50
500 [=====] - 4s 9ms/step - loss: 0.1722 - accuracy: 0.929
:tensorflow:Assets written to: /content/drive/MyDrive/Assignment/NLP with Transfer
:tensorflow:Assets written to: /content/drive/MyDrive/Assignment/NLP with Transfer
h 43/50
500 [=====] - 4s 9ms/step - loss: 0.1734 - accuracy: 0.929
h 44/50
500 [=====] - 4s 9ms/step - loss: 0.1743 - accuracy: 0.929
h 45/50
500 [=====] - 5s 9ms/step - loss: 0.1742 - accuracy: 0.929
h 46/50
500 [=====] - 5s 9ms/step - loss: 0.1749 - accuracy: 0.929
h 47/50
500 [=====] - 5s 9ms/step - loss: 0.1728 - accuracy: 0.929
as.callbacks.History at 0x7ff529d77790>

```

```
print('Test AUC is: ',model.evaluate(X_test_pooled_output,y_te,batch_size=64)[2])
```

```

313/313 [=====] - 2s 7ms/step - loss: 0.1693 - accuracy: 0.9
Test AUC is: 0.9553685784339905

```

```
%tensorboard--logdir /content/logs
```

```

test_data=pd.read_csv('/content/drive/MyDrive/Assignment/NLP with Transfer Learning/test.c
test_data.head(2)

```



---

✓ 0s completed at 12:11 AM ● ✕