6.Compare

January 7, 2022

[ ]:
```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

[ ]:
```
import os
import pathlib
from pathlib import Path
os.chdir("/content/drive/My Drive/Akarshan/BERT")
!ls -l
```

```
total 62833
-rw------- 1 root root  8388432 Dec 26 21:48  BERT5.hdf5
drwx------ 2 root root     4096 Dec  3 16:27  clr
-rw------- 1 root root   488019 Dec 26 22:59  Compare.ipynb
-rw------- 1 root root   476329 Dec 26 23:48 'copy EDA on results.ipynb'
-rw------- 1 root root   488019 Dec 28 17:32 'Copy of Compare.ipynb'
-rw------- 1 root root   255088 Dec 26 23:16 'Copy of Distllbert400000.ipynb'
drwx------ 2 root root     4096 Dec  3 16:27  Data
-rw------- 1 root root  8306584 Dec 24 07:57  DBert1hk.hdf5
-rw------- 1 root root 12719136 Dec 24 07:57  DBert4hk.hdf5
-rw------- 1 root root   251029 Dec 26 22:52  Distllbert400000.ipynb
-rw------- 1 root root   487917 Dec 27 00:17 'EDA on results.ipynb'
drwx------ 2 root root     4096 Dec 18 07:14 'misc model'
-rw------- 1 root root    42964 Dec 26 22:44  model.png
drwx------ 2 root root     4096 Dec  3 16:27  papers
-rw------- 1 root root  8306584 Dec 19 08:56  Rbert4.hdf5
-rw------- 1 root root    85578 Dec 26 22:54  Roberta.ipynb
-rw------- 1 root root  5551000 Dec 26 22:48  roBERT.hdf5
-rw------- 1 root root 12719160 Dec 25 10:35  SBert.hdf5
-rw------- 1 root root  5551000 Dec 26 22:28  scBERT.hdf5
-rw------- 1 root root   203468 Dec 26 22:53  SciBert400k.ipynb
```

[ ]:
```
from psutil import virtual_memory
ram_gb = virtual_memory().total / 1e9
print('Your runtime has {:.1f} gigabytes of available RAM\n'.format(ram_gb))
```

```
if ram_gb < 20:
  print('Not using a high-RAM runtime')
else:
  print('You are using a high-RAM runtime!')
```

```
Your runtime has 27.3 gigabytes of available RAM

You are using a high-RAM runtime!
```

[ ]:
```
gpu_info = !nvidia-smi
gpu_info = '\n'.join(gpu_info)
if gpu_info.find('failed') >= 0:
  print('Not connected to a GPU')
else:
  print(gpu_info)
```

```
Tue Dec 28 17:35:55 2021
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 495.44       Driver Version: 460.32.03    CUDA Version: 11.2     |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|                               |                      |               MIG M. |
|===============================+======================+======================|
|   0  Tesla P100-PCIE...  Off  | 00000000:00:04.0 Off |                    0 |
| N/A   43C    P0    28W / 250W |      0MiB / 16280MiB |      0%      Default |
|                               |                      |                  N/A |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                                  |
|  GPU   GI   CI        PID   Type   Process name                  GPU Memory |
|        ID   ID                                                   Usage      |
|=============================================================================|
|  No running processes found                                                 |
+-----------------------------------------------------------------------------+
```

[ ]:
```
!pip install transformers
!pip install pympler
!pip install tensorflow_addons
```

```
Collecting transformers
  Downloading transformers-4.15.0-py3-none-any.whl (3.4 MB)
     || 3.4 MB 5.1 MB/s
Collecting tokenizers<0.11,>=0.10.1
```

```
  Downloading tokenizers-0.10.3-cp37-cp37m-manylinux_2_5_x86_64.manylinux1_x86_6
4.manylinux_2_12_x86_64.manylinux2010_x86_64.whl (3.3 MB)
     || 3.3 MB 54.6 MB/s
Collecting pyyaml>=5.1
  Downloading PyYAML-6.0-cp37-cp37m-manylinux_2_5_x86_64.manylinux1_x86_64.manyl
inux_2_12_x86_64.manylinux2010_x86_64.whl (596 kB)
     || 596 kB 56.3 MB/s
Collecting huggingface-hub<1.0,>=0.1.0
  Downloading huggingface_hub-0.2.1-py3-none-any.whl (61 kB)
     || 61 kB 514 kB/s
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.7
/dist-packages (from transformers) (21.3)
Requirement already satisfied: regex!=2019.12.17 in /usr/local/lib/python3.7
/dist-packages (from transformers) (2019.12.20)
Requirement already satisfied: tqdm>=4.27 in /usr/local/lib/python3.7/dist-
packages (from transformers) (4.62.3)
Collecting sacremoses
  Downloading sacremoses-0.0.46-py3-none-any.whl (895 kB)
     || 895 kB 56.9 MB/s
Requirement already satisfied: requests in /usr/local/lib/python3.7/dist-
packages (from transformers) (2.23.0)
Requirement already satisfied: importlib-metadata in /usr/local/lib/python3.7
/dist-packages (from transformers) (4.8.2)
Requirement already satisfied: numpy>=1.17 in /usr/local/lib/python3.7/dist-
packages (from transformers) (1.19.5)
Requirement already satisfied: filelock in /usr/local/lib/python3.7/dist-
packages (from transformers) (3.4.0)
Requirement already satisfied: typing-extensions>=3.7.4.3 in
/usr/local/lib/python3.7/dist-packages (from huggingface-
hub<1.0,>=0.1.0->transformers) (3.10.0.2)
Requirement already satisfied: pyparsing!=3.0.5,>=2.0.2 in
/usr/local/lib/python3.7/dist-packages (from packaging>=20.0->transformers)
(3.0.6)
Requirement already satisfied: zipp>=0.5 in /usr/local/lib/python3.7/dist-
packages (from importlib-metadata->transformers) (3.6.0)
Requirement already satisfied: urllib3!=1.25.0,!=1.25.1,<1.26,>=1.21.1 in
/usr/local/lib/python3.7/dist-packages (from requests->transformers) (1.24.3)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.7
/dist-packages (from requests->transformers) (2021.10.8)
Requirement already satisfied: chardet<4,>=3.0.2 in /usr/local/lib/python3.7
/dist-packages (from requests->transformers) (3.0.4)
Requirement already satisfied: idna<3,>=2.5 in /usr/local/lib/python3.7/dist-
packages (from requests->transformers) (2.10)
Requirement already satisfied: six in /usr/local/lib/python3.7/dist-packages
(from sacremoses->transformers) (1.15.0)
Requirement already satisfied: click in /usr/local/lib/python3.7/dist-packages
(from sacremoses->transformers) (7.1.2)
Requirement already satisfied: joblib in /usr/local/lib/python3.7/dist-packages
```

```
(from sacremoses->transformers) (1.1.0)
Installing collected packages: pyyaml, tokenizers, sacremoses, huggingface-hub,
transformers
  Attempting uninstall: pyyaml
    Found existing installation: PyYAML 3.13
    Uninstalling PyYAML-3.13:
      Successfully uninstalled PyYAML-3.13
Successfully installed huggingface-hub-0.2.1 pyyaml-6.0 sacremoses-0.0.46
tokenizers-0.10.3 transformers-4.15.0
Collecting pympler
  Downloading Pympler-1.0.1-py3-none-any.whl (164 kB)
     || 164 kB 5.1 MB/s
Installing collected packages: pympler
Successfully installed pympler-1.0.1
Collecting tensorflow_addons
  Downloading tensorflow_addons-0.15.0-cp37-cp37m-
manylinux_2_12_x86_64.manylinux2010_x86_64.whl (1.1 MB)
     || 1.1 MB 5.1 MB/s
Requirement already satisfied: typeguard>=2.7 in /usr/local/lib/python3.7
/dist-packages (from tensorflow_addons) (2.7.1)
Installing collected packages: tensorflow-addons
Successfully installed tensorflow-addons-0.15.0
```

[ ]:
```
import numpy as np
import pickle
import pandas as pd
import pickle
import time
import matplotlib.pyplot as plt
import seaborn as sns
from pympler import asizeof
import tensorflow as tf
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report
import transformers
from transformers import pipeline
from tensorflow.keras.layers import concatenate
from transformers import TFAutoModel, AutoTokenizer,␣
 ↪AutoConfig,TFAutoModelForSequenceClassification
from tensorflow.keras.callbacks import ModelCheckpoint
from clr import clr_callback
import tensorflow_addons as tfa
```

[ ]:
```
csvfile = 'Data//data.csv'
dropna = 'Data//datadropna.csv'
sent_data_file = 'Data//sent_data.csv'
label_file = 'Data//label.csv'
vocab_file = 'Data//vocab_tr_w.txt'
```

[ ]:
```
df = pd.read_csv(dropna,usecols = ['SBE','Label'])
# df.dropna(inplace=True)
print(df.head())
print(df.shape)
```

```
   Label                                                SBE
0      1  To facilitate an easier notation throughout th...
1      0  Therefore _MATH_ defines a special order of ti...
2      0  This is important since only _MATH_ is the rea...
3      0  Note that in all contour time-integrals we ess...
4      0  Theorem _REF_ proves the equivalence of ensemb...
(1189321, 2)
```

0.0.1 loading embeddings From Distillbert

[ ]:
```
num = len(os.listdir('Data//embeddingBr//'))

with open('Data//embeddingBr//embeddings'+str(0),'rb') as f:
    dataD = pickle.load(f)

for idx in range(1,num):

  with open('Data//embeddingBr//embeddings'+str(idx),'rb') as f:
    mat = pickle.load(f)
    dataD=np.concatenate([dataD,mat],axis=0)
```

[ ]:
```
np.shape(dataD)
```

[ ]:
```
(400000, 768)
```

[ ]:
```
df =  df.iloc[:np.shape(dataD)[0],:]
```

[ ]:
```
_, temp_text, _, temp_labels = train_test_split(dataD, df['Label'],
                                                                 ␣
 ↪random_state=2018,
                                                                  test_size=0.
 ↪3,
                                                                 ␣
 ↪stratify=df['Label'])

# we will use temp_text and temp_labels to create validation and test set
_, test_text, _, test_labels = train_test_split(temp_text, temp_labels,
                                                               ␣
 ↪random_state=2018,
                                                                test_size=0.5,
                                                               ␣
 ↪stratify=temp_labels)
```

```
test_labels = tf.keras.utils.to_categorical(test_labels)

test_data = tf.data.Dataset.from_tensor_slices((test_text))
test_data = test_data.shuffle(50000).batch(128)
```

0.1 Performance of FineTuned Model on DistillBert embedding Trained on 4

400k data points

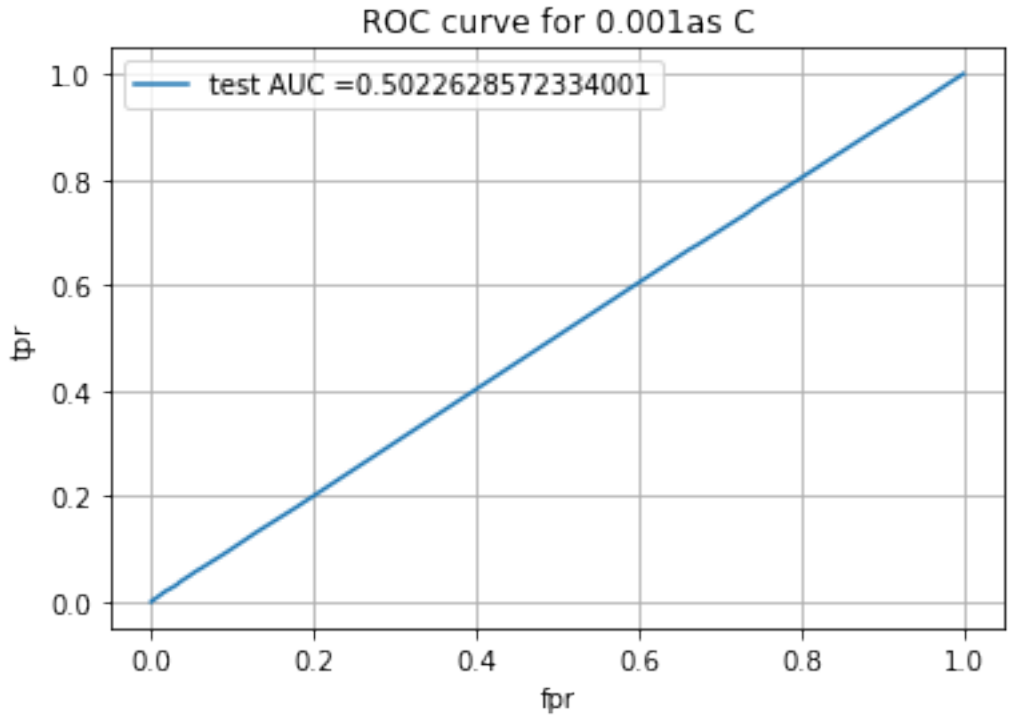
[ ]:
```
from keras.models import load_model
model = load_model("BERT5.hdf5")
model.summary()
```

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

 dense_10 (Dense)            (None, 768)               590592

 dropout_7 (Dropout)         (None, 768)               0

 batch_normalization_9 (Batc  (None, 768)              3072
 hNormalization)

 batch_normalization_10 (Bat  (None, 768)              3072
 chNormalization)

 dense_11 (Dense)            (None, 128)               98432

 dropout_8 (Dropout)         (None, 128)               0

 batch_normalization_11 (Bat  (None, 128)              512
 chNormalization)

 dense_12 (Dense)            (None, 2)                 258

=================================================================
Total params: 695,938
Trainable params: 692,610
Non-trainable params: 3,328
_________________________________________________________________
```

[ ]:
```
from keras.utils.vis_utils import plot_model
plot_model(model, show_shapes=True, show_layer_names=True)
```

[ ]:
```
y_pr_ts_all = model.predict(test_data)
```

[ ]:
```
y_pr_ts = y_pr_ts_all[:,0]
y_ts = test_labels[:,0]
from sklearn.metrics import␣
 ↪roc_curve,auc,confusion_matrix,accuracy_score,precision_score,recall_score,f1_score

test_fpr, test_tpr, te_thresholds = roc_curve(y_ts, y_pr_ts)
plt.plot(test_fpr, test_tpr, label="test AUC ="+str(auc(test_fpr, test_tpr)))
plt.xlabel("fpr")
plt.ylabel("tpr")
plt.title('ROC curve for '+str (0.001)+'as C')
plt.legend()
plt.grid()
plt.show()
```



[ ]:
```
# This section of code where ever implemented is taken from sample kNN python␣
 ↪notebook

def find_best_threshold(threshould, fpr, tpr):
    t = threshould[np.argmax(tpr*(1-fpr))]
    # (tpr*(1-fpr)) will be maximum if your fpr is very low and tpr is very␣
 ↪high
```

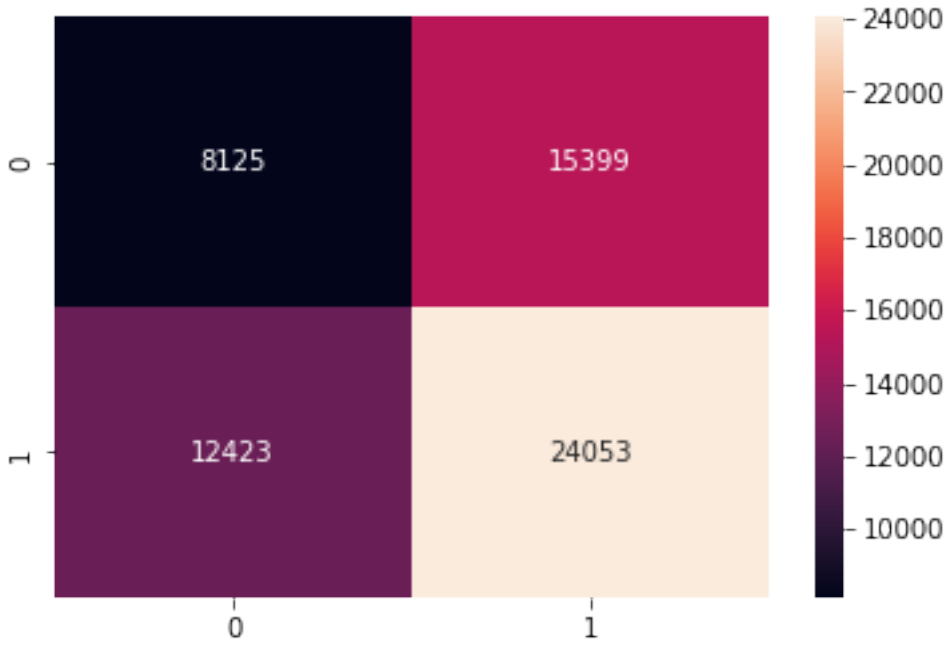
```
    print("the maximum value of tpr*(1-fpr)", max(tpr*(1-fpr)), "for␣
 ↪threshold", np.round(t,3))
    return t

def predict_with_best_t(proba, threshould):
    predictions = []
    for i in proba:
        if i>=threshould:
            predictions.append(1)
        else:
            predictions.append(0)
    return predictions


print('test')
best_ts_thres = find_best_threshold(te_thresholds, test_fpr, test_tpr)
```

```
test
the maximum value of tpr*(1-fpr) 0.22775831605267458 for threshold 0.718
```

[ ]:
```
print('Test Confusion Matrix')
cm2 = pd.DataFrame(confusion_matrix(y_ts, predict_with_best_t(y_pr_ts,␣
 ↪best_ts_thres)), range(2),range(2))
sns.heatmap(cm2, annot=True,fmt='g')
```

```
Test Confusion Matrix
```

[ ]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fd410093410>
```



[ ]:
```
acc=accuracy_score(y_ts, predict_with_best_t(y_pr_ts, best_ts_thres))*100
ps=precision_score(y_ts, predict_with_best_t(y_pr_ts, best_ts_thres))*100
rc=recall_score(y_ts, predict_with_best_t(y_pr_ts, best_ts_thres))*100
f1=f1_score(y_ts, predict_with_best_t(y_pr_ts, best_ts_thres))*100

print("Accuracy on test set: %0.2f%%"%(acc))
print("Precision on test set: %0.2f%%"%(ps))
print("recall score on test set: %0.2f%%"%(rc))
print("f1 score on test set: %0.2f%%"%(f1))
```

```
Accuracy on test set: 53.63%
Precision on test set: 60.97%
recall score on test set: 65.94%
f1 score on test set: 63.36%
```

1 Performance of FineTuned Model on Robert embedding Trained on 400k datapoints

1.0.1 Loading Embeddings form Roberta

[ ]:
```
num = len(os.listdir('Data//embeddingRo//'))

with open('Data//embeddingRo//embeddings'+str(0),'rb') as f:
    dataR = pickle.load(f)

for idx in range(1,num):

  with open('Data//embeddingRo//embeddings'+str(idx),'rb') as f:
    mat = pickle.load(f)
    dataR=np.concatenate([dataR,mat],axis=0)
```

[ ]:
```
asizeof.asizeof(dataR)/1024/1024/1024
```

[ ]:
```
2.2888184785842896
```

[ ]:
```
np.shape(dataR)
```

[ ]:
```
(400000, 768)
```

[ ]:
```
df =  df.iloc[:np.shape(dataR)[0],:]
```

[ ]:
```
_, temp_text, _, temp_labels = train_test_split(dataR, df['Label'],
                                                                 ␣
↪random_state=2018,
                                                                   test_size=0.
↪3,
```

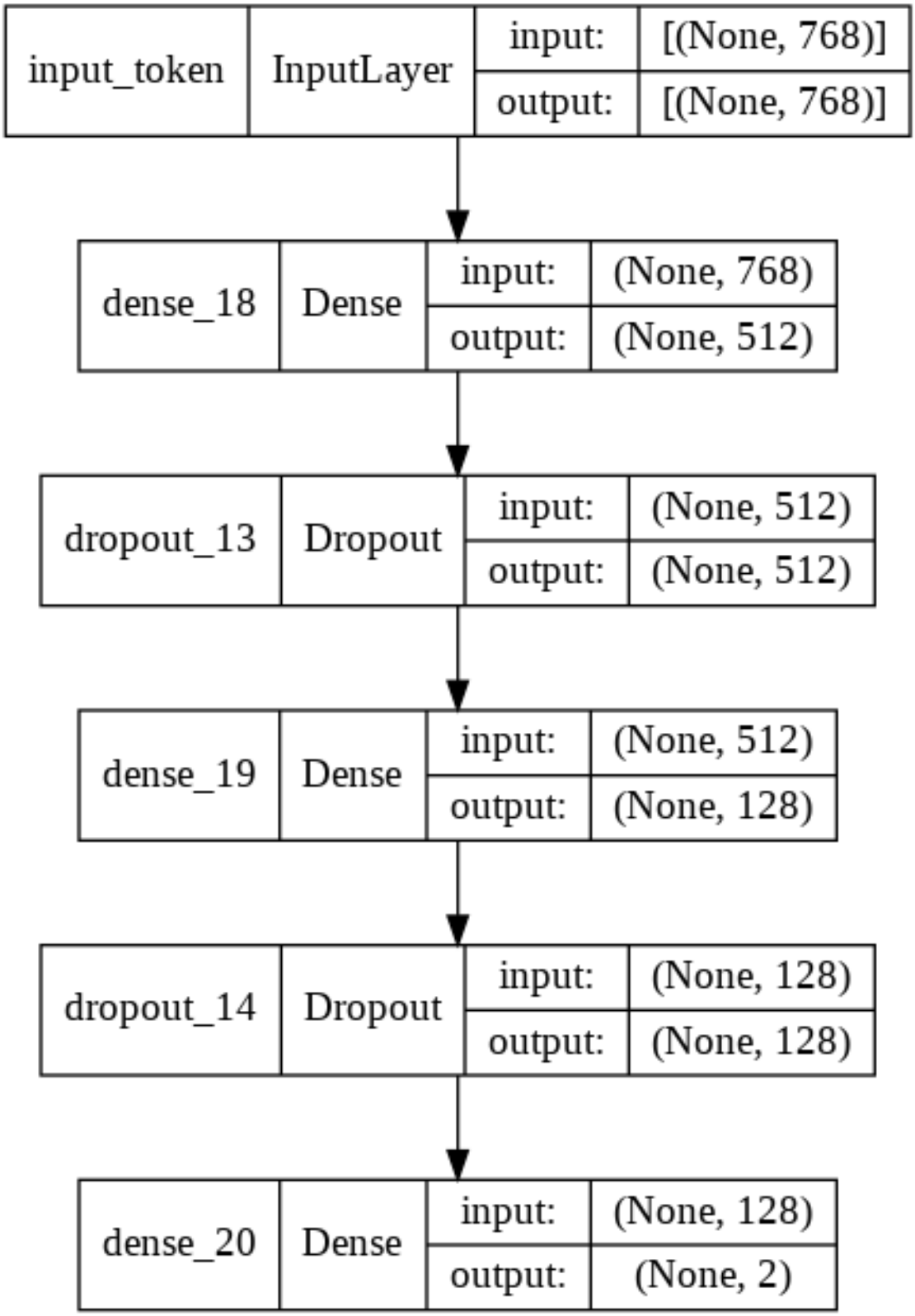
```
 ↪stratify=df['Label'])

# we will use temp_text and temp_labels to create validation and test set
_, test_text, _, test_labels = train_test_split(temp_text, temp_labels,
 ↪random_state=2018,
                                                 test_size=0.5,
 ↪stratify=temp_labels)



test_labels = tf.keras.utils.to_categorical(test_labels)

test_data = tf.data.Dataset.from_tensor_slices((test_text))
test_data = test_data.shuffle(50000).batch(128)
```

[ ]:
```
from keras.models import load_model
model = load_model("roBERT.hdf5")
model.summary()
```

```
Model: "model_5"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_token (InputLayer)    [(None, 768)]             0

 dense_18 (Dense)            (None, 512)               393728

 dropout_13 (Dropout)        (None, 512)               0

 dense_19 (Dense)            (None, 128)               65664

 dropout_14 (Dropout)        (None, 128)               0

 dense_20 (Dense)            (None, 2)                 258

=================================================================
Total params: 459,650
Trainable params: 459,650
Non-trainable params: 0
_________________________________________________________________
```

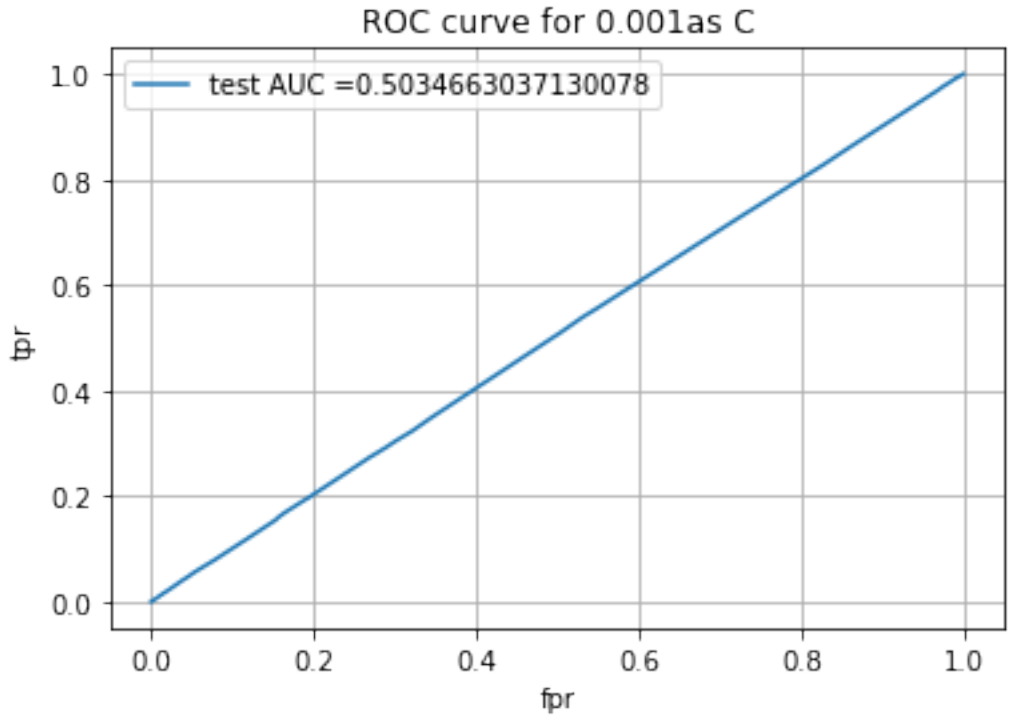
[ ]:
```
from keras.utils.vis_utils import plot_model
plot_model(model, show_shapes=True, show_layer_names=True)
```

[ ]:


input_token
InputLayer
input: [(None, 768)]
output: [(None, 768)]
dense_18
Dense
input: (None, 768)
output: (None, 512)
dropout_13
Dropout
input: (None, 512)
output: (None, 512)
dense_19
Dense
input: (None, 512)
output: (None, 128)
dropout_14
Dropout
input: (None, 128)
output: (None, 128)
dense_20
Dense
input: (None, 128)
output: (None, 2)

[ ]:
```
y_pr_ts_all = model.predict(test_data)
```

[ ]:
```
y_pr_ts = y_pr_ts_all[:,0]
y_ts = test_labels[:,0]
from sklearn.metrics import␣
 ↪roc_curve,auc,confusion_matrix,accuracy_score,precision_score,recall_score,f1_score

test_fpr, test_tpr, te_thresholds = roc_curve(y_ts, y_pr_ts)
plt.plot(test_fpr, test_tpr, label="test AUC ="+str(auc(test_fpr, test_tpr)))
plt.xlabel("fpr")
plt.ylabel("tpr")
plt.title('ROC curve for '+str (0.001)+'as C')
plt.legend()
plt.grid()
plt.show()
```



[ ]:
```
# This section of code where ever implemented is taken from sample kNN python␣
 ↪notebook

def find_best_threshold(threshould, fpr, tpr):
    t = threshould[np.argmax(tpr*(1-fpr))]
    # (tpr*(1-fpr)) will be maximum if your fpr is very low and tpr is very␣
 ↪high
```

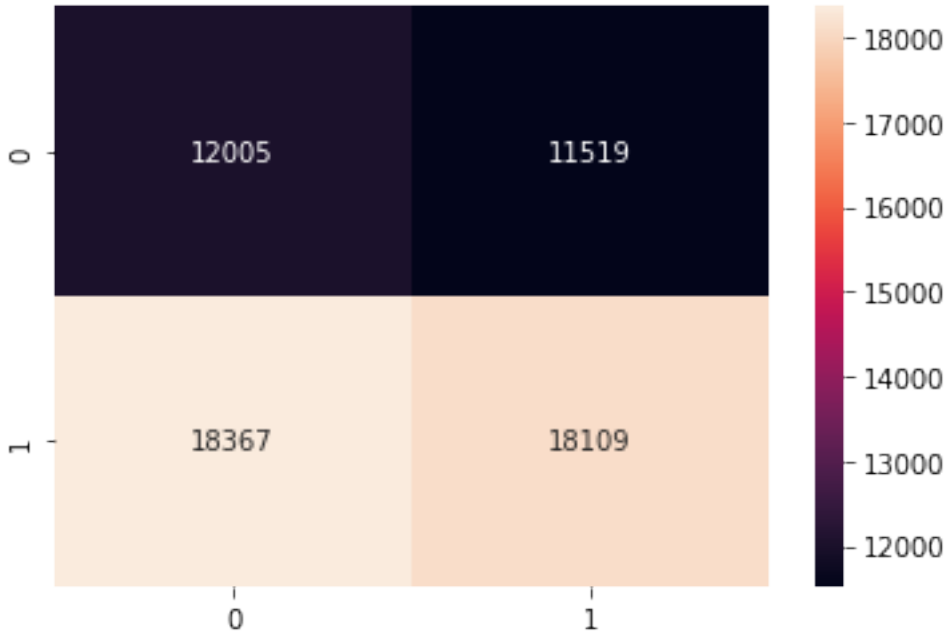
```
    print("the maximum value of tpr*(1-fpr)", max(tpr*(1-fpr)), "for␣
 ↪threshold", np.round(t,3))
    return t

def predict_with_best_t(proba, threshould):
    predictions = []
    for i in proba:
        if i>=threshould:
            predictions.append(1)
        else:
            predictions.append(0)
    return predictions


print('test')
best_ts_thres = find_best_threshold(te_thresholds, test_fpr, test_tpr)
```

```
test
the maximum value of tpr*(1-fpr) 0.25336011958976035 for threshold 0.678
```

[ ]:
```
print('Test Confusion Matrix')
cm2 = pd.DataFrame(confusion_matrix(y_ts, predict_with_best_t(y_pr_ts,␣
 ↪best_ts_thres)), range(2),range(2))
sns.heatmap(cm2, annot=True,fmt='g')
```

```
Test Confusion Matrix
```

[ ]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fd5a3930990>
```



[ ]:
```
acc=accuracy_score(y_ts, predict_with_best_t(y_pr_ts, best_ts_thres))*100
ps=precision_score(y_ts, predict_with_best_t(y_pr_ts, best_ts_thres))*100
rc=recall_score(y_ts, predict_with_best_t(y_pr_ts, best_ts_thres))*100
f1=f1_score(y_ts, predict_with_best_t(y_pr_ts, best_ts_thres))*100

print("Accuracy on test set: %0.2f%%"%(acc))
print("Precision on test set: %0.2f%%"%(ps))
print("recall score on test set: %0.2f%%"%(rc))
print("f1 score on test set: %0.2f%%"%(f1))
```

```
Accuracy on test set: 50.19%
Precision on test set: 61.12%
recall score on test set: 49.65%
f1 score on test set: 54.79%
```

2 Performance of FineTuned Model on SciBert embedding Trained on 400k datapoints

2.0.1 Loading Embeddings from Scibert

[ ]:
```
num = len(os.listdir('Data//embeddingSb//'))

with open('Data//embeddingSb//embeddings'+str(0),'rb') as f:
    dataS = pickle.load(f)

for idx in range(1,num):

  with open('Data//embeddingSb//embeddings'+str(idx),'rb') as f:
    mat = pickle.load(f)
    dataS=np.concatenate([dataS,mat],axis=0)
```

[ ]:
```
np.shape(dataS)
```

[ ]:
```
(400000, 768)
```

[ ]:
```
asizeof.asizeof(dataS)
```

[ ]:
```
2457600128
```

[ ]:
```
del dataR
```

[ ]:
```
datay = df.iloc[:np.shape(dataS)[0],:]
```

[ ]:
```
_, temp_text, _, temp_labels = train_test_split(dataS, datay['Label'],
                                                                                ␣
 ↪random_state=2018,
                                                                     test_size=0.
 ↪3,
```

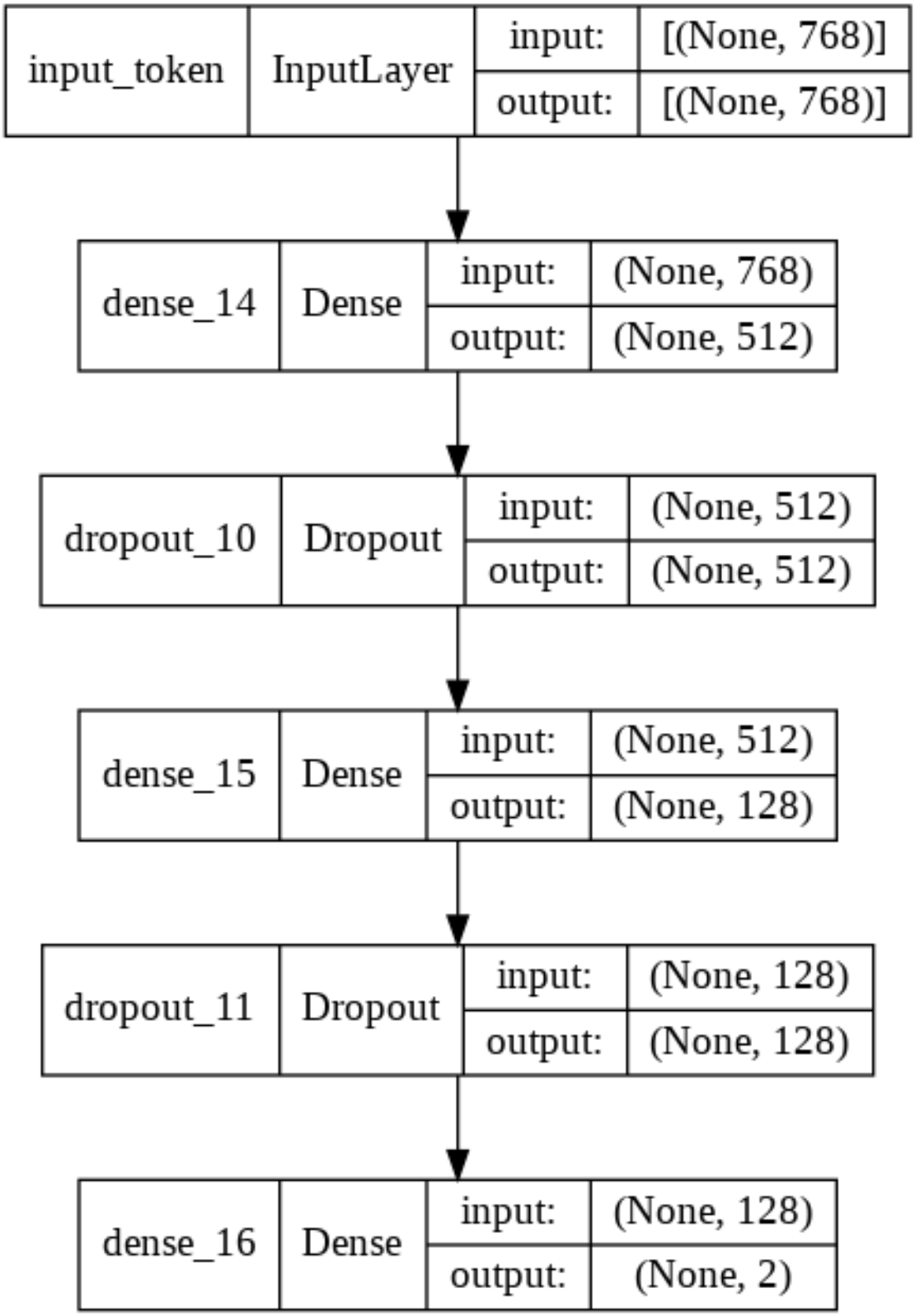
```
 ↪stratify=datay['Label'])

# we will use temp_text and temp_labels to create validation and test set
_, test_text, _, test_labels = train_test_split(temp_text, temp_labels,
 ↪random_state=2018,
                                                 test_size=0.5,
 ↪stratify=temp_labels)



test_labels = tf.keras.utils.to_categorical(test_labels)

test_data = tf.data.Dataset.from_tensor_slices((test_text))
test_data = test_data.shuffle(5000).batch(128)
```

[ ]:
```
from keras.models import load_model
model = load_model("scBERT.hdf5")
model.summary()
```

```
Model: "model_4"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_token (InputLayer)    [(None, 768)]             0

 dense_14 (Dense)            (None, 512)               393728

 dropout_10 (Dropout)        (None, 512)               0

 dense_15 (Dense)            (None, 128)               65664

 dropout_11 (Dropout)        (None, 128)               0

 dense_16 (Dense)            (None, 2)                 258

=================================================================
Total params: 459,650
Trainable params: 459,650
Non-trainable params: 0
_________________________________________________________________
```

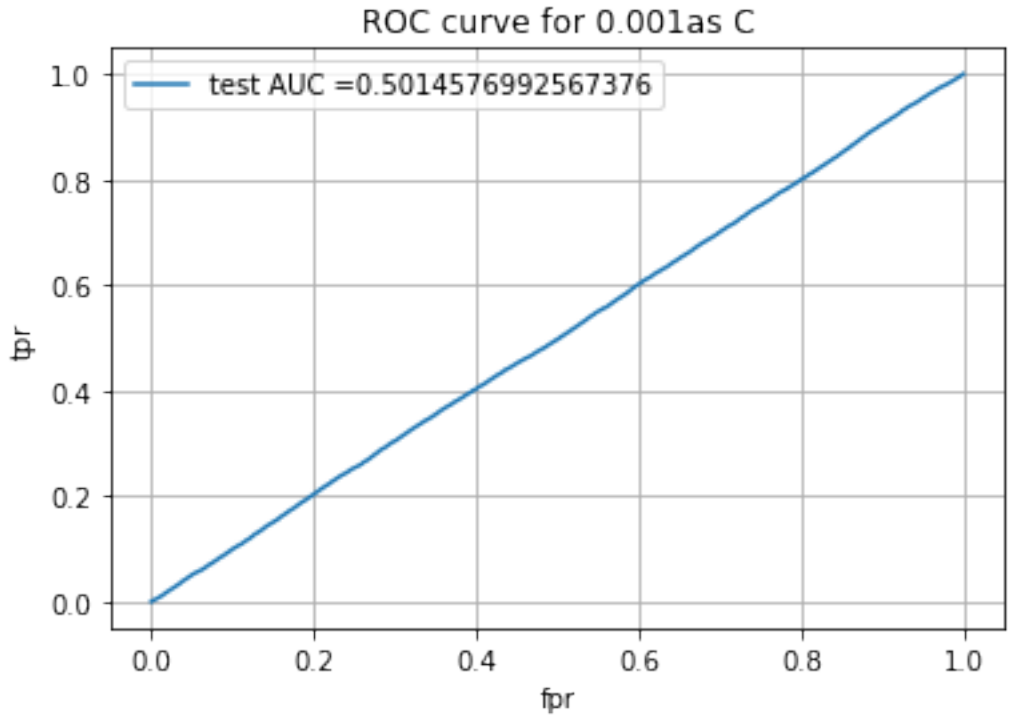
[ ]:
```
from keras.utils.vis_utils import plot_model
plot_model(model, show_shapes=True, show_layer_names=True)
```

[ ]:


input_token
InputLayer
input: [(None, 768)]
output: [(None, 768)]
dense_14
Dense
input: (None, 768)
output: (None, 512)
dropout_10
Dropout
input: (None, 512)
output: (None, 512)
dense_15
Dense
input: (None, 512)
output: (None, 128)
dropout_11
Dropout
input: (None, 128)
output: (None, 128)
dense_16
Dense
input: (None, 128)
output: (None, 2)

[ ]:
```
y_pr_ts_all = model.predict(test_data)
```

[ ]:
```
y_pr_ts = y_pr_ts_all[:,0]
y_ts = test_labels[:,0]
from sklearn.metrics import␣
 ↪roc_curve,auc,confusion_matrix,accuracy_score,precision_score,recall_score,f1_score

test_fpr, test_tpr, te_thresholds = roc_curve(y_ts, y_pr_ts)
plt.plot(test_fpr, test_tpr, label="test AUC ="+str(auc(test_fpr, test_tpr)))
plt.xlabel("fpr")
plt.ylabel("tpr")
plt.title('ROC curve for '+str (0.001)+'as C')
plt.legend()
plt.grid()
plt.show()
```



[ ]:
```
# This section of code where ever implemented is taken from sample kNN python␣
 ↪notebook

def find_best_threshold(threshould, fpr, tpr):
    t = threshould[np.argmax(tpr*(1-fpr))]
    # (tpr*(1-fpr)) will be maximum if your fpr is very low and tpr is very␣
 ↪high
```

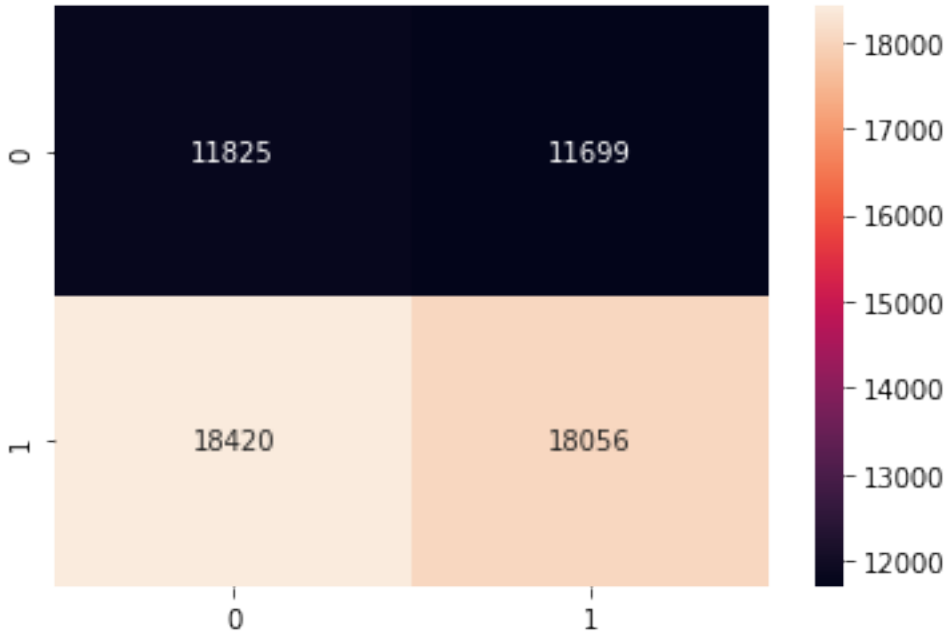
```
    print("the maximum value of tpr*(1-fpr)", max(tpr*(1-fpr)), "for␣
 ↪threshold", np.round(t,3))
    return t

def predict_with_best_t(proba, threshould):
    predictions = []
    for i in proba:
        if i>=threshould:
            predictions.append(1)
        else:
            predictions.append(0)
    return predictions


print('test')
best_ts_thres = find_best_threshold(te_thresholds, test_fpr, test_tpr)
```

```
test
the maximum value of tpr*(1-fpr) 0.24883090420808848 for threshold 0.763
```

[ ]:
```
print('Test Confusion Matrix')
cm2 = pd.DataFrame(confusion_matrix(y_ts, predict_with_best_t(y_pr_ts,␣
 ↪best_ts_thres)), range(2),range(2))
sns.heatmap(cm2, annot=True,fmt='g')
```

```
Test Confusion Matrix
```

[ ]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fd5a37813d0>
```



[ ]:
```
acc=accuracy_score(y_ts, predict_with_best_t(y_pr_ts, best_ts_thres))*100
ps=precision_score(y_ts, predict_with_best_t(y_pr_ts, best_ts_thres))*100
rc=recall_score(y_ts, predict_with_best_t(y_pr_ts, best_ts_thres))*100
f1=f1_score(y_ts, predict_with_best_t(y_pr_ts, best_ts_thres))*100

print("Accuracy on test set: %0.2f%%"%(acc))
print("Precision on test set: %0.2f%%"%(ps))
print("recall score on test set: %0.2f%%"%(rc))
print("f1 score on test set: %0.2f%%"%(f1))
```

```
Accuracy on test set: 49.80%
Precision on test set: 60.68%
recall score on test set: 49.50%
f1 score on test set: 54.52%
```

2.1 Comparison

[ ]:
```
from prettytable import PrettyTable
x = PrettyTable()
x.field_names = ["Model", "F1 score(%)"]
x.add_row(["DistillBert400k", "63.36"])
x.add_row(["Roberta", "54.79"])
x.add_row(["Scibert", "54.52"])

print(x)
```

```
+-----------------+-------------+
|      Model      | F1 score(%) |
+-----------------+-------------+
| DistillBert400k |    63.36    |
|     Roberta     |    54.79    |
|     Scibert     |    54.52    |
+-----------------+-------------+
```