In this notebook I am going to use BERT on the text data of the dataset.

Reference : https://www.tensorflow.org/text/tutorials/classify_text_with_bert

```
# A dependency of the preprocessing for BERT inputs
!pip install -q -U tensorflow-text
```

```
     |████████████████████████████████| 4.3MB 28.4MB/s
```

```
!pip install -q tf-models-official
```

```
     |████████████████████████████████| 1.6MB 32.2MB/s
     |████████████████████████████████| 38.2MB 74kB/s
     |████████████████████████████████| 358kB 52.0MB/s
     |████████████████████████████████| 645kB 49.1MB/s
     |████████████████████████████████| 102kB 13.9MB/s
     |████████████████████████████████| 51kB 7.7MB/s
     |████████████████████████████████| 686kB 49.7MB/s
     |████████████████████████████████| 61kB 9.9MB/s
     |████████████████████████████████| 215kB 57.9MB/s
     |████████████████████████████████| 1.2MB 53.5MB/s
  Building wheel for py-cpuinfo (setup.py) ... done
  Building wheel for seqeval (setup.py) ... done
```

```
#importing libraries
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split

import os
import shutil

import tensorflow as tf
import tensorflow_hub as hub
import tensorflow_text as text
from tensorflow.keras.utils import plot_model
from official.nlp import optimization  # to create AdamW optimizer

import matplotlib.pyplot as plt
import seaborn as sns
from tensorflow.keras.metrics import Accuracy
accuracy = Accuracy()
from sklearn.metrics import confusion_matrix
from sklearn.metrics import roc_curve
from sklearn.metrics import roc_auc_score
from imblearn.over_sampling import SMOTE, RandomOverSampler

tf.get_logger().setLevel('ERROR')
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/externals/six.py:31: FutureWar
  "(https://pypi.org/project/six/).", FutureWarning)
/usr/local/lib/python3.7/dist-packages/sklearn/utils/deprecation.py:144: Futu
  warnings.warn(message, FutureWarning)
```

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

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

```
#reading the dataset
df = pd.read_csv('/content/drive/MyDrive/Applied_ai/df_text.csv')
```

```
df.head()
```

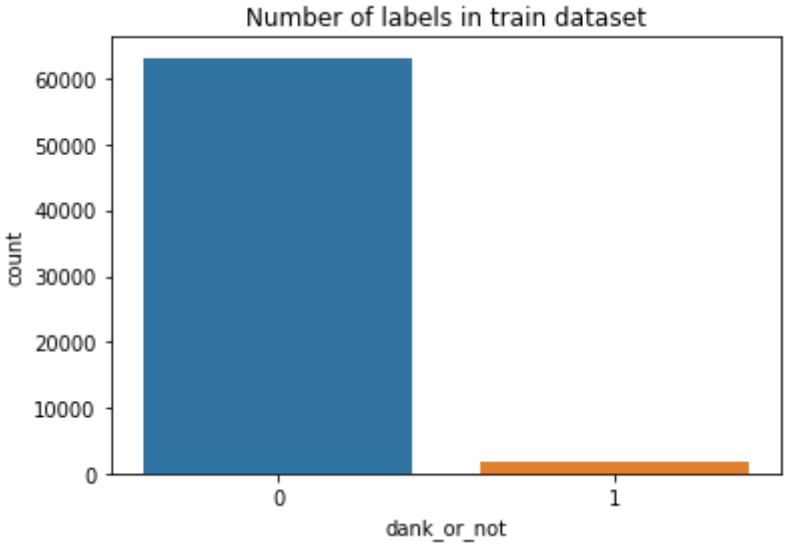
	text	dank_or_not
0	literally bros international feeling world ent...	0
1	looking differ	0
2	mematic young assimilated talented cuno made r...	0
3	commentgatewi diewithicovid joebiden voters caml	0
4	errr impostor funeral vent wisdom unless expla...	0

```
X = df['text']
y = df['dank_or_not']
```

```
#splitting the dataset in train and test data
X_train,  X_test,y_train, y_test = train_test_split(X,y, test_size=0.2, stratify=y
```

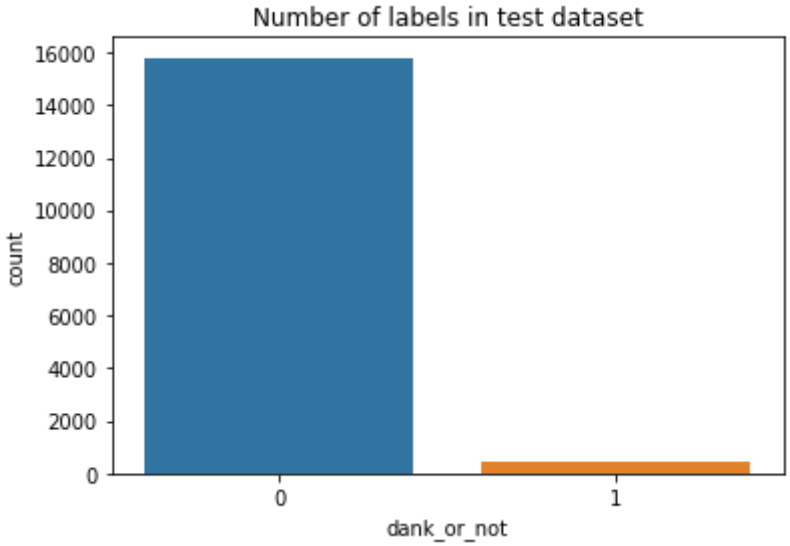
```
sns.countplot(y_train)
plt.title('Number of labels in train dataset')
plt.show()
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarni
  FutureWarning
```



```
sns.countplot(y_test)
plt.title('Number of labels in test dataset')
plt.show()
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarni
  FutureWarning
```




```
#Since our dataset is highly imbalanced, oversampling only train dataset
X_train, y_train = RandomOverSampler(random_state=777).fit_sample(np.array(X_train
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/utils/validation.py:760: DataC
  y = column_or_1d(y, warn=True)
/usr/local/lib/python3.7/dist-packages/sklearn/utils/deprecation.py:87: Futur
  warnings.warn(msg, category=FutureWarning)
```

```
#pre-steps to load tensorboard in colab
! wget https://bin.equinox.io/c/4VmDzA7iaHb/ngrok-stable-linux-amd64.zip > /dev/nu
! unzip ngrok-stable-linux-amd64.zip > /dev/null 2>&1


LOG_DIR = './log'
get_ipython().system_raw(
    'tensorboard --logdir {} --host 0.0.0.0 --port 6006 &'
    .format(LOG_DIR)
)


get_ipython().system_raw('./ngrok http 6006 &')


! curl -s http://localhost:4040/api/tunnels | python3 -c \
    "import sys, json; print(json.load(sys.stdin)['tunnels'][0]['public_url'])"
```

```
https://db0543002eed.ngrok.io
```

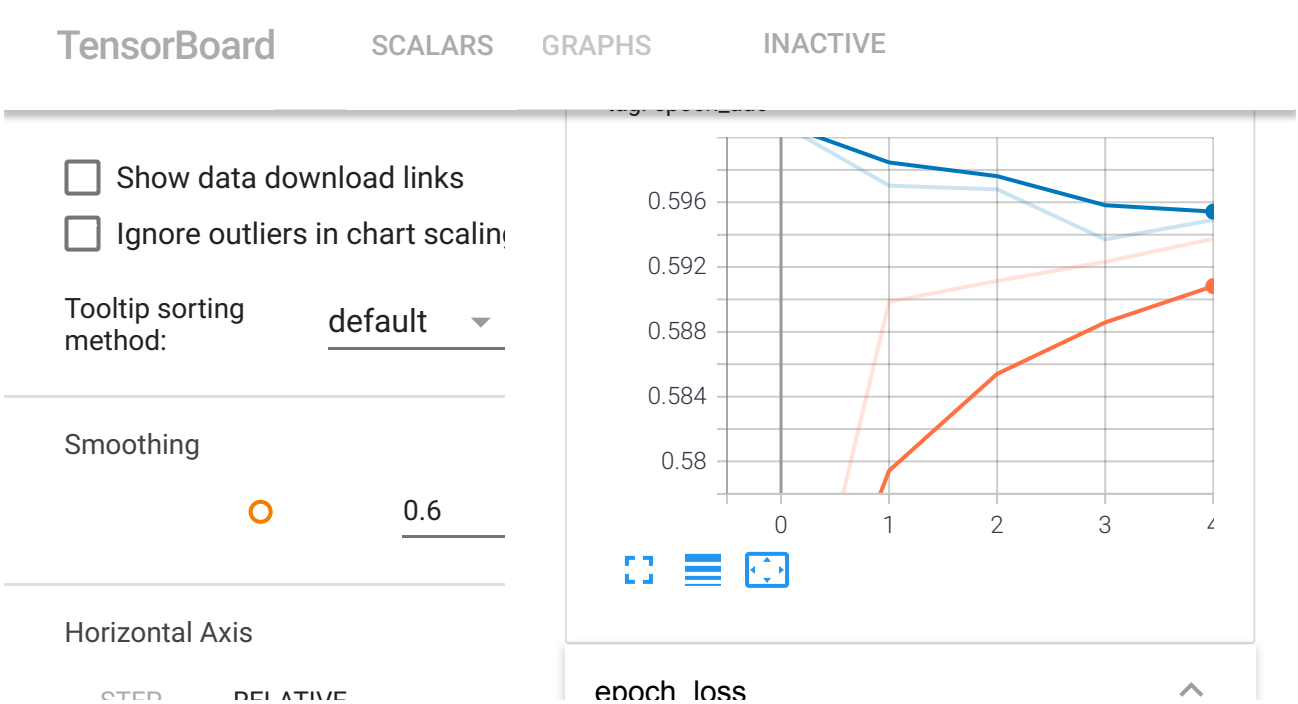
```
%load_ext tensorboard
# Clear any logs from previous runs
!rm -rf ./log/
```

```
#tensorboard callback to draw different metrices during training
tbCallBack = tf.keras.callbacks.TensorBoard(log_dir='./log', histogram_freq=0,
                         write_graph=True,
                         write_grads=True,
                         write_images=True)
callback_list = [tbCallBack]


#Loading models from TensorFlow Hub
tfhub_handle_encoder = "https://tfhub.dev/tensorflow/small_bert/bert_en_uncased_L-
tfhub_handle_preprocess = "https://tfhub.dev/tensorflow/bert_en_uncased_preprocess


#building classification model
def build_classifier_model():
  text_input = tf.keras.layers.Input(shape=(), dtype=tf.string, name='text')
  preprocessing_layer = hub.KerasLayer(tfhub_handle_preprocess, name='preprocessin
  encoder_inputs = preprocessing_layer(text_input)
  encoder = hub.KerasLayer(tfhub_handle_encoder, trainable=False, name='BERT_encod
  outputs = encoder(encoder_inputs)
  net = outputs['pooled_output']
  net = tf.keras.layers.Dropout(0.1)(net)
  net = tf.keras.layers.Dense(1, activation='sigmoid', name='classifier')(net)
  return tf.keras.Model(text_input, net)


model = build_classifier_model()


#plotting model
plot_model(model)
```

```
loss = tf.keras.losses.BinaryCrossentropy(from_logits=True)

#compiling model
model.compile(optimizer='adam', loss=loss, metrics=['AUC'])

#training the model
history = model.fit(X_train,y_train,
                            validation_data=(X_test, y_test),
                            epochs=5,
                            callbacks = callback_list)
```

```
Epoch 1/5
/usr/local/lib/python3.7/dist-packages/tensorflow/python/keras/backend.py:501
  '"`binary_crossentropy` received `from_logits=True`, but the `output`'
3948/3948 [==============================] - 274s 68ms/step - loss: 0.6908 -
Epoch 2/5
3948/3948 [==============================] - 268s 68ms/step - loss: 0.6809 -
Epoch 3/5
3948/3948 [==============================] - 267s 68ms/step - loss: 0.6805 -
Epoch 4/5
3948/3948 [==============================] - 267s 68ms/step - loss: 0.6801 -
Epoch 5/5
3948/3948 [==============================] - 267s 68ms/step - loss: 0.6798 -
```

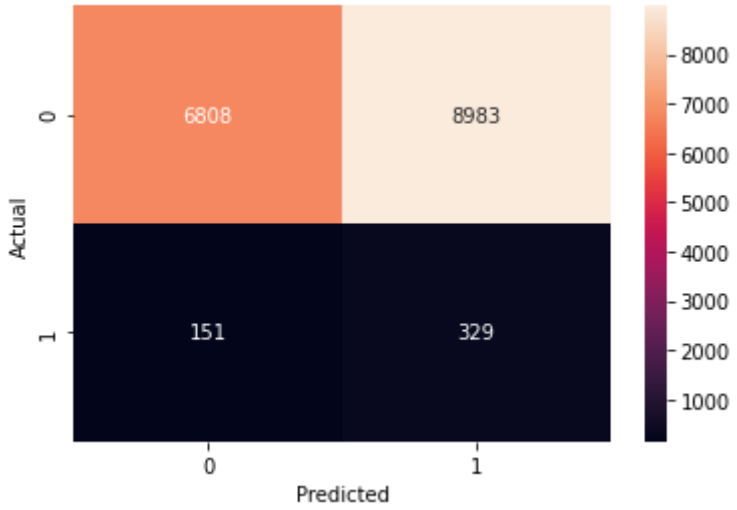
```
#drawing tensorboard
%tensorboard --logdir log
```




```
#predicting for test dataset
y_pred_b = model.predict(X_test)
```

```
#converting probabilities to labels
y_pred_b_df = pd.DataFrame(y_pred_b).round().astype('int')
```

```
#plotting confusion matrix
cm_b = confusion_matrix(y_test, y_pred_b_df[0])
sns.heatmap(cm_b, annot=True, fmt='g')
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.show()
```




```
#getting true negative, false positive, false negative and true positive values fr
tn_b, fp_b, fn_b, tp_b = cm_b.ravel()
```

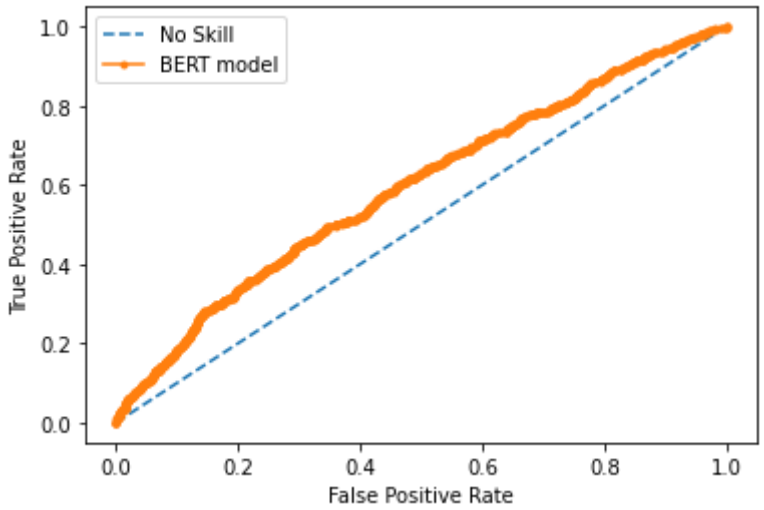
```
#computing sensetivity and specificity
sensitivity_b = (tp_b/(tp_b+fn_b)).round(4)
```

```
specificity_b = (tn_b/(tn_b+fp_b)).round(4)
print('sensitivity    : ',sensitivity_b)
print('specificity    : ',specificity_b)
print('accuracy       : ',accuracy(y_test, y_pred_b_df[0]))
```

```
sensitivity    :  0.6854
specificity    :  0.4311
accuracy       :  tf.Tensor(0.43863314, shape=(), dtype=float32)
```

```
'''Plotting ROC curve for both Random Forest and Gradient Boosting Classifiers tra

#no-skill values
ns_probs = [0 for _ in range(len(y_test))]
ns_auc = roc_auc_score(y_test, ns_probs)

# calculate roc curves
ns_fpr, ns_tpr, _ = roc_curve(y_test, ns_probs)
fpr, tpr, _ = roc_curve(y_test, y_pred_b)
print('AUC : ',roc_auc_score(y_test, y_pred_b))

# plot the roc curve for the model
plt.plot(ns_fpr, ns_tpr, linestyle='--', label='No Skill')
plt.plot(fpr, tpr, marker='.', label='BERT model')

# axis labels
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
# show the legend
plt.legend()
# show the plot
plt.show()
```

```
AUC :  0.5951647694889494
```




```
#saving the model
model.save('bert_model', include_optimizer=False)
```

```
WARNING:absl:Found untraced functions such as restored_function_body, restore
```

```
#reloading the saved model
reloaded_model = tf.saved_model.load('bert_model')
```

```
reloaded_results = tf.sigmoid(reloaded_model(tf.constant([X_test.iloc[0]])))
```

```
reloaded_results
```

```
<tf.Tensor: shape=(1, 1), dtype=float32, numpy=array([[0.6195241]], dtype=flo
```

```
!zip -r /content/bert_model.zip /content/bert_model
```

```
  adding: content/bert_model/ (stored 0%)
  adding: content/bert_model/saved_model.pb (deflated 91%)
  adding: content/bert_model/variables/ (stored 0%)
  adding: content/bert_model/variables/variables.index (deflated 69%)
  adding: content/bert_model/variables/variables.data-00000-of-00001 (deflate
  adding: content/bert_model/keras_metadata.pb (deflated 81%)
  adding: content/bert_model/assets/ (stored 0%)
  adding: content/bert_model/assets/vocab.txt (deflated 53%)
```

✓ 0s completed at 8:21 AM