using-bert-distilbert-submitted-1

April 25, 2023

###Importing all the necessity Liabries for the building the MODEL

[1]:
```
##### Import all necessity functions for Machine Learning #####
import sys
import math
import string
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import scipy as shc
import warnings
import zipfile
import cv2
import os
import re
import nltk
import random
from collections import Counter
from functools import reduce
from itertools import chain
from wordcloud import WordCloud
from google.colab.patches import cv2_imshow
from keras.preprocessing import image
from sklearn.metrics._plot.confusion_matrix import confusion_matrix
from sklearn.model_selection import train_test_split, KFold, StratifiedKFold,␣
↪GridSearchCV, RandomizedSearchCV
from sklearn.preprocessing import StandardScaler, RobustScaler, MinMaxScaler
from sklearn.decomposition import PCA
from sklearn.cluster import KMeans, DBSCAN, AgglomerativeClustering
from sklearn.feature_selection import mutual_info_classif,␣
↪mutual_info_regression, SelectKBest, chi2, VarianceThreshold
from imblearn.under_sampling import RandomUnderSampler, NearMiss
from imblearn.over_sampling import RandomOverSampler, SMOTE, SMOTEN, SMOTENC,␣
↪SVMSMOTE, KMeansSMOTE, BorderlineSMOTE, ADASYN
from imblearn.ensemble import EasyEnsembleClassifier
from sklearn.feature_extraction.text import CountVectorizer, TfidfVectorizer
from sklearn.naive_bayes import GaussianNB, BernoulliNB, MultinomialNB
from sklearn.neighbors import KNeighborsClassifier, KNeighborsRegressor,␣
 ↪NearestNeighbors
from sklearn.linear_model import LinearRegression, LogisticRegression,␣
 ↪SGDClassifier, SGDRegressor, Perceptron
from sklearn.neural_network import MLPClassifier, MLPRegressor
from sklearn.svm import SVC, SVR
from sklearn.tree import DecisionTreeClassifier, DecisionTreeRegressor,␣
 ↪ExtraTreeClassifier, ExtraTreeRegressor
from sklearn.ensemble import BaggingClassifier, BaggingRegressor,␣
 ↪RandomForestClassifier, RandomForestRegressor, VotingClassifier,␣
 ↪VotingRegressor
from sklearn.ensemble import AdaBoostClassifier, AdaBoostRegressor,␣
 ↪GradientBoostingClassifier, GradientBoostingRegressor, StackingClassifier,␣
 ↪StackingRegressor
from sklearn.metrics import classification_report, mean_absolute_error,␣
 ↪mean_squared_error, r2_score, accuracy_score, recall_score, precision_score,␣
 ↪f1_score, silhouette_score
from xgboost import XGBClassifier, XGBRegressor

##### Download keras #####
!pip install keras

##### Remove all warnings #####
import warnings
warnings.filterwarnings("ignore")

##### Import all necessity functions for Neural Network #####
import tensorflow as tf
from keras.models import Sequential, Model
from keras.utils import plot_model
from keras.layers import Dense, Conv2D, LSTM, GRU, RNN, Flatten, AvgPool2D,␣
 ↪MaxPool2D, GlobalAveragePooling2D, BatchNormalization, Dropout, LeakyReLU,␣
 ↪ELU, PReLU,  Embedding
from keras.activations import tanh, relu, sigmoid, softmax, swish
from keras.regularizers import L1, L2, L1L2
from keras.optimizers import SGD, Adagrad, Adadelta, RMSprop, Adam, Adamax,␣
 ↪Nadam
from keras.initializers import HeNormal, HeUniform, GlorotNormal, GlorotUniform
from keras.losses import SparseCategoricalCrossentropy,␣
 ↪CategoricalCrossentropy, hinge, MSE, MAE, Huber, BinaryCrossentropy
import keras.utils as image
from google.colab.patches import cv2_imshow
from keras.utils import plot_model

##### Plotting the confusion matrix #####
from mlxtend.evaluate import confusion_matrix
from mlxtend.plotting import plot_confusion_matrix
from sklearn.metrics import multilabel_confusion_matrix
from sklearn.metrics import confusion_matrix

###### NLP Libraries ######

from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize, sent_tokenize
from nltk.stem.porter import PorterStemmer
from nltk.stem import WordNetLemmatizer

###### Download functions for nltk ######

nltk.download('stopwords')
nltk.download('punkt')
nltk.download('wordnet')
nltk.download('omw-1.4')
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-
wheels/public/simple/
Requirement already satisfied: keras in /usr/local/lib/python3.9/dist-packages
(2.12.0)
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data…
[nltk_data]   Unzipping corpora/stopwords.zip.
[nltk_data] Downloading package punkt to /root/nltk_data…
[nltk_data]   Unzipping tokenizers/punkt.zip.
[nltk_data] Downloading package wordnet to /root/nltk_data…
[nltk_data] Downloading package omw-1.4 to /root/nltk_data…
```

[1]: True

######Checking the GPU that I am using

[ ]:
```
!nvidia-smi
```

```
Mon Apr 10 18:38:22 2023
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 525.85.12    Driver Version: 525.85.12    CUDA Version: 12.0     |
|-------------------------------+----------------------+----------------------+
| 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 T4            Off  | 00000000:00:04.0 Off |                    0 |
| N/A   35C    P8     9W /  70W |      0MiB / 15360MiB |      0%      Default |
|                               |                      |                  N/A |
+-------------------------------+----------------------+----------------------+
+-----------------------------------------------------------------------------+
| Processes:                                                                  |
|  GPU   GI   CI        PID   Type   Process name                  GPU Memory |
|        ID   ID                                                   Usage      |
|=============================================================================|
|  No running processes found                                                 |
+-----------------------------------------------------------------------------+
```

#####Import the dataset

[2]:
```
#### Import the dataset ####
df = pd.read_csv('/content/dataset.csv')

#### Show the dataset ####
df.head()
```

[2]:
```
                                                text  humor
0  Joe biden rules out 2020 bid: 'guys, i'm not r…  False
1  Watch: darvish gave hitter whiplash with slow …  False
2  What do you call a turtle without its shell? d…   True
3      5 reasons the 2016 election feels so personal  False
4  Pasco police shot mexican migrant from behind,…  False
```

#Perform EDA - Exploratory data analysis

####Check the Shape of this dataset

[ ]:
```
#### Show the shape of this dataset ####
def find_shape(df):
  if df.empty:
    raise Exception('DataFrame is empty'.capitalize())
  else:
    return df.shape

#### Call the current function with this current directory ####
try:
  shape = find_shape(df)
except Exception as e:
  print(e.with_traceback)
else:
  print('The shape of this dataset is {}'.format(shape))
  print('*'*50)
  print('The number of instances of this dataset is = {}'.format(shape[0],'\n'))
  print('The number of features of this dataset is = {}'.format(shape[1]))
```

```
The shape of this dataset is (200000, 2)
**************************************************
The number of instances of this dataset is = 200000
The number of features of this dataset is = 2
```

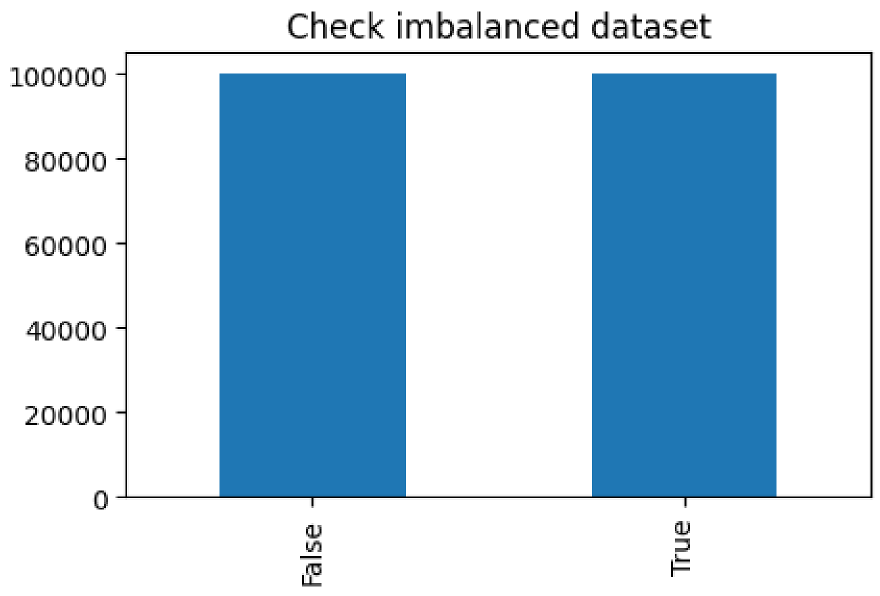
####Check `NaN` value presence or not

```
[ ]: #### define a function for NaN check ####
def check_NaN(df):
  if df.empty:
    raise Exception('The DataFrame is empty'.capitalize())
  else:
    if df.isnull().sum().sum() > 0:
      return "NaN value presence in this dataset.".capitalize()
    else:
      return "No NaN value in this dataset.".capitalize()

#### Call the current function with this current directory ####
try:
  checking = check_NaN(df)
  print("The final outcome is: "+checking)
except Exception as e:
  print(e.with_traceback)
```

```
The final outcome is: No nan value in this dataset.
```

####Checking dataset contains `Imbalanced` dataset or not

```
[ ]: #### Define a function for checking the imbalanced condition of dataset ####
def check_imbalanced(df):
  if df.empty:
    raise Exception('DataFrame is empty'.capitalize())
  else:
    return df.loc[:, 'humor'].value_counts()

#### Call the current function with this current directory ####
try:
  seris = check_imbalanced(df)
except Exception as e:
  print(e.with_traceback)
else:
  plt.figure(figsize = (5, 3))
  plt.title('Check Imbalanced dataset'.capitalize())
  seris.plot(kind = 'bar')
  plt.show()
```



####Checking the `duplicate` instances presence or not

```
[ ]: #### Define a function for checking the duplicate instances ####
def check_duplicate(df):
  if df.empty:
    raise Exception('DataFrame is empty'.capitalize())
  else:
    return df.duplicated().sum()

#### Call the current function with this current directory ####
try:
  duplicate_check = check_duplicate(df)
except Exception as e:
  print(e.with_traceback)
else:
  if duplicate_check == 0:
    print('There is No duplicated instances in this dataset'.capitalize())
  else:
    print('There is duplicated instances in this dataset'.capitalize())
```

```
There is no duplicated instances in this dataset
```

####Checking the `info` of this dataset

```
[ ]: #### Define a function for the displaying the info ####
def show_info(df):
  if df.empty:
    raise Exception('DataFrame is empty'.capitalize())
  else:
    print(df.info())

#### Call the current function with this current directory ####

try:
  show_info(df)
except Exception as e:
  print(e.with_traceback)
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 200000 entries, 0 to 199999
Data columns (total 2 columns):
 #   Column  Non-Null Count   Dtype
---  ------  --------------   -----
 0   text    200000 non-null  object
 1   humor   200000 non-null  bool
dtypes: bool(1), object(1)
memory usage: 1.7+ MB
None
```

#####`Pre-processing` the dataset for finding the insights of the dataset

[ ]:
```
#### Do the copy of the orginal dataset ####
dummy_df = df.copy()

#### Display the dataset ####
dummy_df.head()
```

[ ]:
```
                                                text  humor
0  Joe biden rules out 2020 bid: 'guys, i'm not r…  False
1  Watch: darvish gave hitter whiplash with slow …  False
2  What do you call a turtle without its shell? d…   True
3      5 reasons the 2016 election feels so personal  False
4  Pasco police shot mexican migrant from behind,…  False
```

####Remove `HTML` or `Number` from the dataset - for analysis

[ ]:
```
#### define a function to remove the HTML format from the dataset ####
def removeHTML(record):
  CLEANR = re.compile('<.*?>')
  cleantext = re.sub('[^a-zA-Z]', ' ', record)
  return cleantext
try:
  dummy_df.loc[:, 'text'] = dummy_df.loc[:, 'text'].apply(removeHTML)
except Exception as e:
  print(e)
else:
```

```
print(dummy_df.head())
```

```
                                                text  humor
0  Joe biden rules out      bid   guys  i m not r…  False
1  Watch  darvish gave hitter whiplash with slow …  False
2  What do you call a turtle without its shell  d…   True
3        reasons the      election feels so personal  False
4  Pasco police shot mexican migrant from behind …  False
```

####Remove `Stopwords` from the dataset - for analysis

```
[ ]: #### Define a function for remove the stopwords ####
def removestopwords(record):
  clean_text_ = []
  for word_ in word_tokenize(record):
    if word_ in stopwords.words('english'):
      pass
    else:
      clean_text_.append(word_)

  return ' '.join(clean_text_)

#### Call the current function with this current directory ####
try:
  dummy_df.loc[:, 'text'] = dummy_df.loc[:, 'text'].apply(removestopwords)
except Exception as e:
  print(e.with_traceback)
else:
  print(dummy_df.head())
```

```
                                                text  humor
0                     Joe biden rules bid guys running  False
1      Watch darvish gave hitter whiplash slow pitch  False
2                  What call turtle without shell dead   True
3                       reasons election feels personal  False
4  Pasco police shot mexican migrant behind new a…  False
```

####Convert `Shortform` text to its `abbreviation`

```
[ ]: #### Define a function for conversion from short-text to its abbrevation form␣
 ↪####
def conversion_full_form(dummy_df):
  CONTRACTION_MAP = {
    "ain't": "is not",
    "aren't": "are not",
    "can't": "cannot",
    "can't've": "cannot have",
    "'cause": "because",
    "could've": "could have",
    "couldn't": "could not",
    "couldn't've": "could not have",
    "didn't": "did not",
    "doesn't": "does not",
    "don't": "do not",
    "hadn't": "had not",
    "hadn't've": "had not have",
    "hasn't": "has not",
    "haven't": "have not",
    "he'd": "he would",
    "he'd've": "he would have",
    "he'll": "he will",
    "he'll've": "he he will have",
    "he's": "he is",
    "how'd": "how did",
    "how'd'y": "how do you",
    "how'll": "how will",
    "how's": "how is",
    "i'd": "i would",
    "i'd've": "i would have",
    "i'll": "i will",
    "i'll've": "i will have",
    "i'm": "i am",
    "i've": "i have",
    "isn't": "is not",
    "it'd": "it would",
    "it'd've": "it would have",
    "it'll": "it will",
    "it'll've": "it will have",
    "it's": "it is",
    "let's": "let us",
    "ma'am": "madam",
    "mayn't": "may not",
    "might've": "might have",
    "mightn't": "might not",
    "mightn't've": "might not have",
    "must've": "must have",
    "mustn't": "must not",
    "mustn't've": "must not have",
    "needn't": "need not",
    "needn't've": "need not have",
    "o'clock": "of the clock",
    "oughtn't": "ought not",
    "oughtn't've": "ought not have",
    "shan't": "shall not",
    "sha'n't": "shall not",
    "shan't've": "shall not have",
    "she'd": "she would",
    "she'd've": "she would have",
    "she'll": "she will",
    "she'll've": "she will have",
    "she's": "she is",
    "should've": "should have",
    "shouldn't": "should not",
    "shouldn't've": "should not have",
    "so've": "so have",
    "so's": "so as",
    "that'd": "that would",
    "that'd've": "that would have",
    "that's": "that is",
    "there'd": "there would",
    "there'd've": "there would have",
    "there's": "there is",
    "they'd": "they would",
    "they'd've": "they would have",
    "they'll": "they will",
    "they'll've": "they will have",
    "they're": "they are",
    "they've": "they have",
    "to've": "to have",
    "wasn't": "was not",
    "we'd": "we would",
    "we'd've": "we would have",
    "we'll": "we will",
    "we'll've": "we will have",
    "we're": "we are",
    "we've": "we have",
    "weren't": "were not",
    "what'll": "what will",
    "what'll've": "what will have",
    "what're": "what are",
    "what's": "what is",
    "what've": "what have",
    "when's": "when is",
    "when've": "when have",
    "where'd": "where did",
    "where's": "where is",
    "where've": "where have",
    "who'll": "who will",
    "who'll've": "who will have",
    "who's": "who is",
    "who've": "who have",
    "why's": "why is",
    "why've": "why have",
    "will've": "will have",
    "won't": "will not",
    "won't've": "will not have",
    "would've": "would have",
    "wouldn't": "would not",
    "wouldn't've": "would not have",
    "y'all": "you all",
    "y'all'd": "you all would",
    "y'all'd've": "you all would have",
    "y'all're": "you all are",
    "y'all've": "you all have",
    "you'd": "you would",
    "you'd've": "you would have",
    "you'll": "you will",
    "you'll've": "you will have",
    "you're": "you are",
    "you've": "you have"
  }

  clean_text = []
  for word in dummy_df:
    if word in CONTRACTION_MAP:
      clean_text.append(CONTRACTION_MAP[word])
    else:
      clean_text.append(word)

  return ''.join(clean_text)

#### Call this function with this directory ####
try:
  dummy_df.loc[:, 'text'] = dummy_df.loc[:, 'text'].apply(lambda x:␣
 ↪conversion_full_form(x))
except Exception as e:
  print(e.args)
else:
  print('Short-form to abbrevation is done successfully'.capitalize())
```

```
Short-form to abbrevation is done successfully
```

####Use `Stemming` to convert to the `root` text of the dataset

[ ]:
```
#### Call the PorterStemmer ####
ps = PorterStemmer()

#### Define a stemming function ####
def stemming(eachRow):
  cleaned = []
  for word in word_tokenize(eachRow):
    cleaned.append(ps.stem(word))

  return ' '.join(cleaned)

#### Call the current function with this current directory ####
try:
  dummy_df.loc[:, 'text'] = dummy_df.loc[:, 'text'].apply(stemming)
except Exception as e:
  print(e.with_traceback)
else:
  print(dummy_df.head())
```

```
                                                text  humor
0                          joe biden rule bid guy run  False
1      watch darvish gave hitter whiplash slow pitch  False
2                    what call turtl without shell dead   True
3                              reason elect feel person  False
4  pasco polic shot mexican migrant behind new au…  False
```

####Install text-hero API

```
[ ]: !pip install -U texthero
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-
wheels/public/simple/
Collecting texthero
  Downloading texthero-1.1.0-py3-none-any.whl (24 kB)
Requirement already satisfied: numpy>=1.17 in /usr/local/lib/python3.9/dist-
packages (from texthero) (1.22.4)
Collecting unidecode>=1.1.1
  Downloading Unidecode-1.3.6-py3-none-any.whl (235 kB)
                           235.9/235.9 kB
18.4 MB/s eta 0:00:00
Collecting gensim<4.0,>=3.6.0
  Downloading gensim-3.8.3.tar.gz (23.4 MB)
                            23.4/23.4 MB
68.5 MB/s eta 0:00:00
  Preparing metadata (setup.py) … done
Requirement already satisfied: pandas>=1.0.2 in /usr/local/lib/python3.9/dist-
packages (from texthero) (1.4.4)
Requirement already satisfied: nltk>=3.3 in /usr/local/lib/python3.9/dist-
packages (from texthero) (3.8.1)
Collecting spacy<3.0.0
  Downloading
spacy-2.3.9-cp39-cp39-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (4.9 MB)
                            4.9/4.9 MB
108.0 MB/s eta 0:00:00
```

```
Requirement already satisfied: scikit-learn>=0.22 in
/usr/local/lib/python3.9/dist-packages (from texthero) (1.2.2)
Requirement already satisfied: plotly>=4.2.0 in /usr/local/lib/python3.9/dist-
packages (from texthero) (5.13.1)
Requirement already satisfied: wordcloud>=1.5.0 in
/usr/local/lib/python3.9/dist-packages (from texthero) (1.8.2.2)
Requirement already satisfied: matplotlib>=3.1.0 in
/usr/local/lib/python3.9/dist-packages (from texthero) (3.7.1)
Requirement already satisfied: tqdm>=4.3 in /usr/local/lib/python3.9/dist-
packages (from texthero) (4.65.0)
Requirement already satisfied: scipy>=0.18.1 in /usr/local/lib/python3.9/dist-
packages (from gensim<4.0,>=3.6.0->texthero) (1.10.1)
Requirement already satisfied: six>=1.5.0 in /usr/local/lib/python3.9/dist-
packages (from gensim<4.0,>=3.6.0->texthero) (1.16.0)
Requirement already satisfied: smart_open>=1.8.1 in
/usr/local/lib/python3.9/dist-packages (from gensim<4.0,>=3.6.0->texthero)
(6.3.0)
Requirement already satisfied: contourpy>=1.0.1 in
/usr/local/lib/python3.9/dist-packages (from matplotlib>=3.1.0->texthero)
(1.0.7)
Requirement already satisfied: importlib-resources>=3.2.0 in
/usr/local/lib/python3.9/dist-packages (from matplotlib>=3.1.0->texthero)
(5.12.0)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.9/dist-
packages (from matplotlib>=3.1.0->texthero) (23.0)
Requirement already satisfied: python-dateutil>=2.7 in
/usr/local/lib/python3.9/dist-packages (from matplotlib>=3.1.0->texthero)
(2.8.2)
Requirement already satisfied: pyparsing>=2.3.1 in
/usr/local/lib/python3.9/dist-packages (from matplotlib>=3.1.0->texthero)
(3.0.9)
Requirement already satisfied: kiwisolver>=1.0.1 in
/usr/local/lib/python3.9/dist-packages (from matplotlib>=3.1.0->texthero)
(1.4.4)
Requirement already satisfied: fonttools>=4.22.0 in
/usr/local/lib/python3.9/dist-packages (from matplotlib>=3.1.0->texthero)
(4.39.3)
Requirement already satisfied: pillow>=6.2.0 in /usr/local/lib/python3.9/dist-
packages (from matplotlib>=3.1.0->texthero) (8.4.0)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.9/dist-
packages (from matplotlib>=3.1.0->texthero) (0.11.0)
Requirement already satisfied: click in /usr/local/lib/python3.9/dist-packages
(from nltk>=3.3->texthero) (8.1.3)
Requirement already satisfied: regex>=2021.8.3 in /usr/local/lib/python3.9/dist-
packages (from nltk>=3.3->texthero) (2022.10.31)
Requirement already satisfied: joblib in /usr/local/lib/python3.9/dist-packages
(from nltk>=3.3->texthero) (1.1.1)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.9/dist-
```

```
packages (from pandas>=1.0.2->texthero) (2022.7.1)
Requirement already satisfied: tenacity>=6.2.0 in /usr/local/lib/python3.9/dist-
packages (from plotly>=4.2.0->texthero) (8.2.2)
Requirement already satisfied: threadpoolctl>=2.0.0 in
/usr/local/lib/python3.9/dist-packages (from scikit-learn>=0.22->texthero)
(3.1.0)
Requirement already satisfied: setuptools in /usr/local/lib/python3.9/dist-
packages (from spacy<3.0.0->texthero) (67.6.1)
Requirement already satisfied: murmurhash<1.1.0,>=0.28.0 in
/usr/local/lib/python3.9/dist-packages (from spacy<3.0.0->texthero) (1.0.9)
Collecting srsly<1.1.0,>=1.0.2
  Downloading
srsly-1.0.6-cp39-cp39-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (209 kB)
                          209.2/209.2 kB
27.3 MB/s eta 0:00:00
Collecting plac<1.2.0,>=0.9.6
  Downloading plac-1.1.3-py2.py3-none-any.whl (20 kB)
Requirement already satisfied: cymem<2.1.0,>=2.0.2 in
/usr/local/lib/python3.9/dist-packages (from spacy<3.0.0->texthero) (2.0.7)
Collecting catalogue<1.1.0,>=0.0.7
  Downloading catalogue-1.0.2-py2.py3-none-any.whl (16 kB)
Requirement already satisfied: blis<0.8.0,>=0.4.0 in
/usr/local/lib/python3.9/dist-packages (from spacy<3.0.0->texthero) (0.7.9)
Requirement already satisfied: requests<3.0.0,>=2.13.0 in
/usr/local/lib/python3.9/dist-packages (from spacy<3.0.0->texthero) (2.27.1)
Requirement already satisfied: preshed<3.1.0,>=3.0.2 in
/usr/local/lib/python3.9/dist-packages (from spacy<3.0.0->texthero) (3.0.8)
Collecting thinc<7.5.0,>=7.4.1
  Downloading
thinc-7.4.6-cp39-cp39-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (1.1 MB)
                           1.1/1.1 MB
75.0 MB/s eta 0:00:00
Collecting wasabi<1.1.0,>=0.4.0
  Downloading wasabi-0.10.1-py3-none-any.whl (26 kB)
Requirement already satisfied: zipp>=3.1.0 in /usr/local/lib/python3.9/dist-
packages (from importlib-resources>=3.2.0->matplotlib>=3.1.0->texthero) (3.15.0)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in
/usr/local/lib/python3.9/dist-packages (from
requests<3.0.0,>=2.13.0->spacy<3.0.0->texthero) (1.26.15)
Requirement already satisfied: certifi>=2017.4.17 in
/usr/local/lib/python3.9/dist-packages (from
requests<3.0.0,>=2.13.0->spacy<3.0.0->texthero) (2022.12.7)
Requirement already satisfied: charset-normalizer~=2.0.0 in
/usr/local/lib/python3.9/dist-packages (from
requests<3.0.0,>=2.13.0->spacy<3.0.0->texthero) (2.0.12)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.9/dist-
packages (from requests<3.0.0,>=2.13.0->spacy<3.0.0->texthero) (3.4)
Building wheels for collected packages: gensim
```

```
  Building wheel for gensim (setup.py) … done
  Created wheel for gensim: filename=gensim-3.8.3-cp39-cp39-linux_x86_64.whl
size=26528016
sha256=0625e9a68cb4f353bf150e8473b2b13cd523e6c5e028b0f3bb7b6dcd58ca43a7
  Stored in directory: /root/.cache/pip/wheels/ca/5d/af/618594ec2f28608c1d6ee7d2
b7e95a3e9b06551e3b80a491d6
Successfully built gensim
Installing collected packages: wasabi, plac, unidecode, srsly, catalogue, thinc,
gensim, spacy, texthero
  Attempting uninstall: wasabi
    Found existing installation: wasabi 1.1.1
    Uninstalling wasabi-1.1.1:
      Successfully uninstalled wasabi-1.1.1
  Attempting uninstall: srsly
    Found existing installation: srsly 2.4.6
    Uninstalling srsly-2.4.6:
      Successfully uninstalled srsly-2.4.6
  Attempting uninstall: catalogue
    Found existing installation: catalogue 2.0.8
    Uninstalling catalogue-2.0.8:
      Successfully uninstalled catalogue-2.0.8
  Attempting uninstall: thinc
    Found existing installation: thinc 8.1.9
    Uninstalling thinc-8.1.9:
      Successfully uninstalled thinc-8.1.9
  Attempting uninstall: gensim
    Found existing installation: gensim 4.3.1
    Uninstalling gensim-4.3.1:
      Successfully uninstalled gensim-4.3.1
  Attempting uninstall: spacy
    Found existing installation: spacy 3.5.1
    Uninstalling spacy-3.5.1:
      Successfully uninstalled spacy-3.5.1
ERROR: pip's dependency resolver does not currently take into account all
the packages that are installed. This behaviour is the source of the following
dependency conflicts.
en-core-web-sm 3.5.0 requires spacy<3.6.0,>=3.5.0, but you have spacy 2.3.9
which is incompatible.
confection 0.0.4 requires srsly<3.0.0,>=2.4.0, but you have srsly 1.0.6 which is
incompatible.
Successfully installed catalogue-1.0.2 gensim-3.8.3 plac-1.1.3 spacy-2.3.9
srsly-1.0.6 texthero-1.1.0 thinc-7.4.6 unidecode-1.3.6 wasabi-0.10.1
```

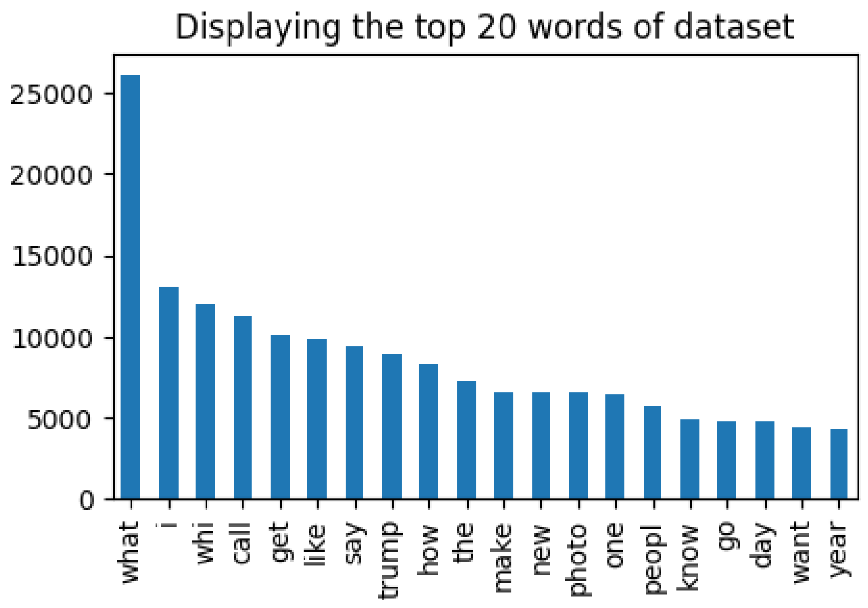
```
[ ]: !pip install https://github.com/explosion/spacy-models/releases/download/
     ↪en_core_web_sm-2.2.0/en_core_web_sm-2.2.0.tar.gz
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-
wheels/public/simple/
Collecting https://github.com/explosion/spacy-
models/releases/download/en_core_web_sm-2.2.0/en_core_web_sm-2.2.0.tar.gz
  Downloading https://github.com/explosion/spacy-
models/releases/download/en_core_web_sm-2.2.0/en_core_web_sm-2.2.0.tar.gz (12.0
MB)
                           12.0/12.0 MB
46.6 MB/s eta 0:00:00
  Preparing metadata (setup.py) … done
Requirement already satisfied: spacy>=2.2.0 in /usr/local/lib/python3.9/dist-
packages (from en-core-web-sm==2.2.0) (2.3.9)
Requirement already satisfied: blis<0.8.0,>=0.4.0 in
/usr/local/lib/python3.9/dist-packages (from spacy>=2.2.0->en-core-web-
sm==2.2.0) (0.7.9)
Requirement already satisfied: plac<1.2.0,>=0.9.6 in
/usr/local/lib/python3.9/dist-packages (from spacy>=2.2.0->en-core-web-
sm==2.2.0) (1.1.3)
Requirement already satisfied: tqdm<5.0.0,>=4.38.0 in
/usr/local/lib/python3.9/dist-packages (from spacy>=2.2.0->en-core-web-
sm==2.2.0) (4.65.0)
Requirement already satisfied: cymem<2.1.0,>=2.0.2 in
/usr/local/lib/python3.9/dist-packages (from spacy>=2.2.0->en-core-web-
sm==2.2.0) (2.0.7)
Requirement already satisfied: murmurhash<1.1.0,>=0.28.0 in
/usr/local/lib/python3.9/dist-packages (from spacy>=2.2.0->en-core-web-
sm==2.2.0) (1.0.9)
Requirement already satisfied: thinc<7.5.0,>=7.4.1 in
/usr/local/lib/python3.9/dist-packages (from spacy>=2.2.0->en-core-web-
sm==2.2.0) (7.4.6)
Requirement already satisfied: setuptools in /usr/local/lib/python3.9/dist-
packages (from spacy>=2.2.0->en-core-web-sm==2.2.0) (67.6.1)
Requirement already satisfied: requests<3.0.0,>=2.13.0 in
/usr/local/lib/python3.9/dist-packages (from spacy>=2.2.0->en-core-web-
sm==2.2.0) (2.27.1)
Requirement already satisfied: wasabi<1.1.0,>=0.4.0 in
/usr/local/lib/python3.9/dist-packages (from spacy>=2.2.0->en-core-web-
sm==2.2.0) (0.10.1)
Requirement already satisfied: numpy>=1.15.0 in /usr/local/lib/python3.9/dist-
packages (from spacy>=2.2.0->en-core-web-sm==2.2.0) (1.22.4)
Requirement already satisfied: catalogue<1.1.0,>=0.0.7 in
/usr/local/lib/python3.9/dist-packages (from spacy>=2.2.0->en-core-web-
sm==2.2.0) (1.0.2)
Requirement already satisfied: srsly<1.1.0,>=1.0.2 in
```

```
/usr/local/lib/python3.9/dist-packages (from spacy>=2.2.0->en-core-web-
sm==2.2.0) (1.0.6)
Requirement already satisfied: preshed<3.1.0,>=3.0.2 in
/usr/local/lib/python3.9/dist-packages (from spacy>=2.2.0->en-core-web-
sm==2.2.0) (3.0.8)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in
/usr/local/lib/python3.9/dist-packages (from
requests<3.0.0,>=2.13.0->spacy>=2.2.0->en-core-web-sm==2.2.0) (1.26.15)
Requirement already satisfied: charset-normalizer~=2.0.0 in
/usr/local/lib/python3.9/dist-packages (from
requests<3.0.0,>=2.13.0->spacy>=2.2.0->en-core-web-sm==2.2.0) (2.0.12)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.9/dist-
packages (from requests<3.0.0,>=2.13.0->spacy>=2.2.0->en-core-web-sm==2.2.0)
(3.4)
Requirement already satisfied: certifi>=2017.4.17 in
/usr/local/lib/python3.9/dist-packages (from
requests<3.0.0,>=2.13.0->spacy>=2.2.0->en-core-web-sm==2.2.0) (2022.12.7)
Building wheels for collected packages: en-core-web-sm
  Building wheel for en-core-web-sm (setup.py) … done
  Created wheel for en-core-web-sm: filename=en_core_web_sm-2.2.0-py3-none-
any.whl size=12019122
sha256=d84d59f921653ec00d212c2d7fd3774f7d67ee8cbabebfa5f3756b088812d550
  Stored in directory: /root/.cache/pip/wheels/02/87/47/4d729a97cc46afa46135595b
4de32d01461f05947df39166d7
Successfully built en-core-web-sm
Installing collected packages: en-core-web-sm
  Attempting uninstall: en-core-web-sm
    Found existing installation: en-core-web-sm 3.5.0
    Uninstalling en-core-web-sm-3.5.0:
      Successfully uninstalled en-core-web-sm-3.5.0
Successfully installed en-core-web-sm-2.2.0
```

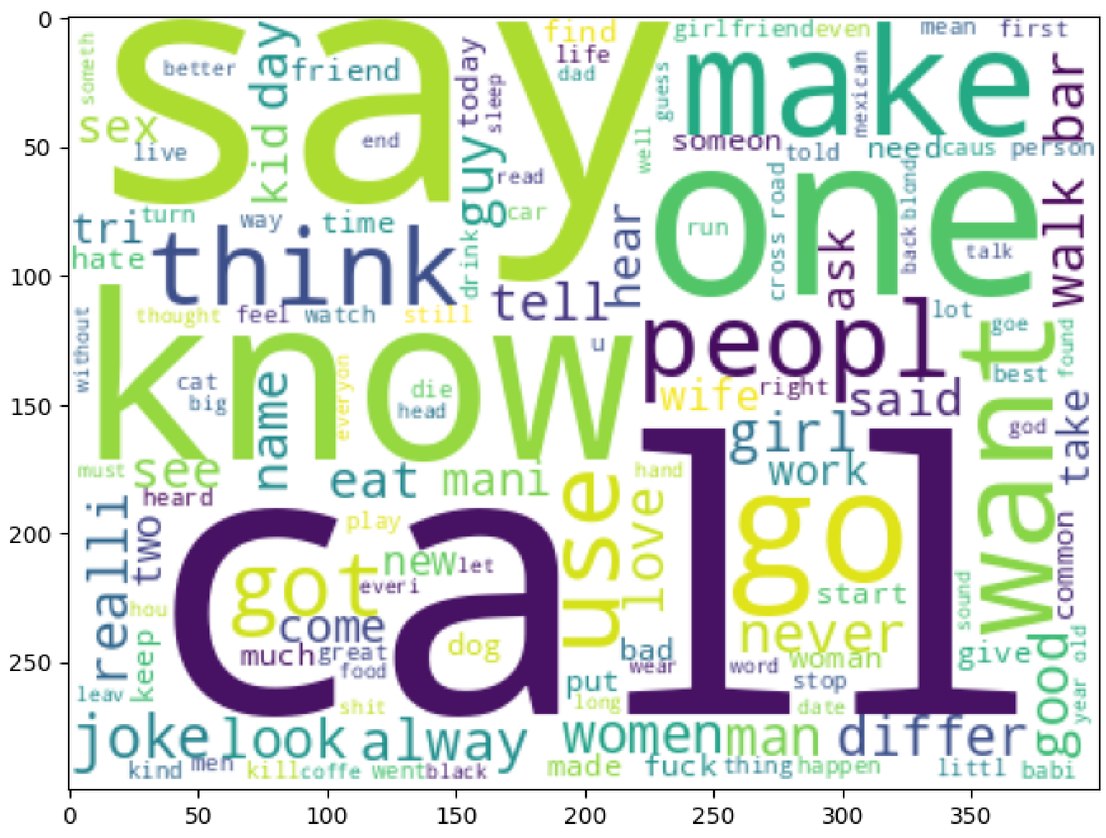
####Visualization of top 20 words of the dataset

[ ]:
```
#### Import the texthero ####
import texthero as hero
plt.figure(figsize = (5, 3))
plt.title('Displaying the top 20 words of dataset'.capitalize())
hero.visualization.top_words(dummy_df.loc[:, 'text'])[:20].plot(kind = 'bar')
plt.show()
```

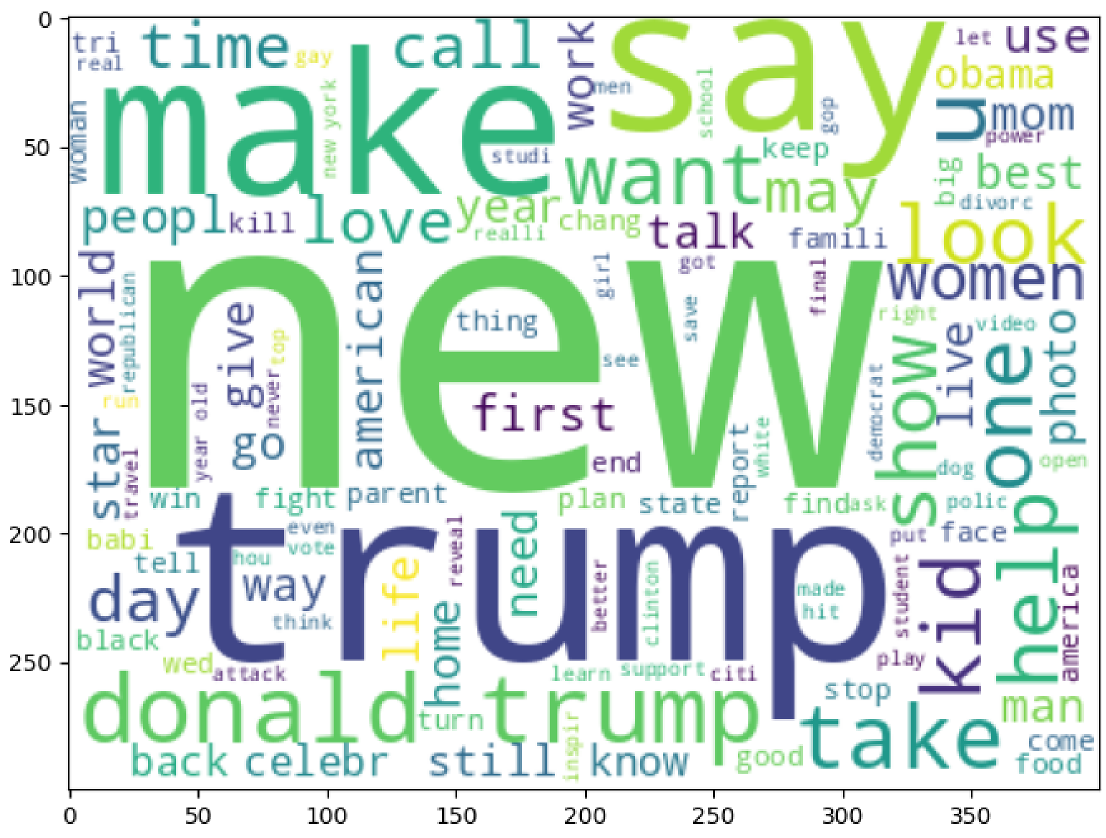


####Check `Most Frequent words` with respect to `target classes` using `WordCloud`

```
[ ]: #### show the most frequent word with respect to True ####
     plt.figure(figsize = (8, 10))
     word_cloud_ = WordCloud(height = 300, width = 400, min_font_size = 8,␣
      ↪background_color = 'white')
     word_cloud_.generate(dummy_df.loc[dummy_df.loc[:, 'humor'] == True, 'text'].str.
      ↪cat(sep = ''))

     plt.imshow(word_cloud_)
     plt.show()
```



```
[ ]: #### show the most frequent word with respect to False ####
     plt.figure(figsize = (8, 10))
     word_cloud_ = WordCloud(height = 300, width = 400, min_font_size = 8,␣
      ↪background_color = 'white')
     word_cloud_.generate(dummy_df.loc[dummy_df.loc[:, 'humor'] == False, 'text'].
      ↪str.cat(sep = ''))

     plt.imshow(word_cloud_)
     plt.show()
```



####Visualization of scatter plot to understand the data

```
[ ]: #### Create a NEW feature named PCA using neumerical conversion TFIDF to␣
      ↪convert the Higher dimension into Lower dimension ####
     def PCA(df):
       if df.empty:
         raise Exception('The dataset is empty'.capitalize())
       else:
         dummy_df['PCA'] = (hero.tfidf(dummy_df.loc[:, 'text'], max_features=500)).
      ↪pipe(hero.pca)
         return dummy_df

     #### This PCA would be n_components = 2 #####
     try:
       dummy_df = PCA(df)
     except Exception as e:
       print(e.with_traceback)
     else:
       print(dummy_df.head())
```

```
#### Visualization the dataset ####
plot = hero.scatterplot(df = dummy_df, col = 'PCA', color = 'humor', title =␣
 ↪'PCA with Scatter')
plt.show()
```

```
                                                   text  humor  \
0                           joe biden rule bid guy run  False
1       watch darvish gave hitter whiplash slow pitch  False
2                      what call turtl without shell dead   True
3                             reason elect feel person  False
4  pasco polic shot mexican migrant behind new au…  False

                                         PCA
0  [-0.040401537656080184, -0.014151492910330365]
1    [-0.04797831400216986, -0.01951040719741723]
2      [0.41988102033822045, 0.02699406957930625]
3    [-0.04560593615863668, -0.02244263774563714]
4   [-0.048008664292122305, -0.03480864579227028]
```

#Model Building using BERT- ARCHITECTURE

[ ]:
```
#### Convert the target column into Object ####
df.loc[:, 'humor'] = df.loc[:, 'humor'].astype('object')
#### Check the conversion is done successfully or not ####
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 200000 entries, 0 to 199999
Data columns (total 2 columns):
 #   Column  Non-Null Count   Dtype
---  ------  --------------   -----
 0   text    200000 non-null  object
 1   humor   200000 non-null  object
dtypes: object(2)
memory usage: 3.1+ MB
```

####Use last 10000 Dataset for the Validation Test

[4]:
```
#### Define a function for validation ####
def validation_dataset(df):
  return df.tail(10000)

val_data = validation_dataset(df)

####Show the shape of this validation dataset ####
try:
  shape = find_shape(val_data)
except Exception as e:
  print(e.with_traceback)
else:
  print('The shape of this dataset is {}'.format(shape))
  print('*'*50)
  print('The number of instances of this dataset is = {}'.format(shape[0],'\n'))
  print('The number of features of this dataset is = {}'.format(shape[1]))

#### Show the validation data ####
val_data.head()
```

```
<built-in method with_traceback of NameError object at 0x7fe76486f9f0>
```

[4]:

```
                                                      text  humor
190000  My friend took a whole bottle of chill pills a…   True
190001  What came first, the chicken or the egg? the r…   True
190002  Mommy, i don't wanna grow up and die! oh. well…   True
190003  Why are tennis equipment factories so loud? be…   True
190004  The cost of not investing in after-school prog…  False
```

[5]:

```
#### Rest of the dataset for using Train and Test ####
df = df[0:-10000]

#### Show the current shape of the dataset ####
try:
  shape = find_shape(df)
except Exception as e:
  print(e.with_traceback)
else:
  print('The shape of this dataset is {}'.format(shape))
  print('*'*50)
  print('The number of instances of this dataset is = {}'.format(shape[0],'\n'))
  print('The number of features of this dataset is = {}'.format(shape[1]))

#### Show the dataset ####
df.head()
```

```
<built-in method with_traceback of NameError object at 0x7fe6bb424860>
```

[5]:

```
                                                text  humor
0  Joe biden rules out 2020 bid: 'guys, i'm not r…  False
1  Watch: darvish gave hitter whiplash with slow …  False
2  What do you call a turtle without its shell? d…   True
3         5 reasons the 2016 election feels so personal  False
4  Pasco police shot mexican migrant from behind,…  False
```

####Make the dataset into Independent & Dependent Features

```
[6]: def conversion(df):
       if df.empty:
         raise Exception('DataFrame is empty'.capitalize())
       else:
         #### For the traing and testing purpose ####
         X = list(df['text'])
         y = list(df['humor'])
         #### For the validation ####
         X_val = list(val_data['text'])
         y_val = list(val_data['humor'])

         return X, y, X_val, y_val
     try:
       X, y, X_val, y_val = conversion(df)
     except Exception as e:
       print(e.with_traceback)
     else:
       print('The length of X(Independent) Feature is {}'.format(len(X),'\n'))
       print('The length of y(Independent) Feature is {}'.format(len(y),'\n'))
       print('*'*60)
       print('The length of X-validation(Independent) Feature is {}'.
      ↪format(len(X_val),'\n'))
       print('The length of y-validation(Independent) Feature is {}'.
      ↪format(len(y_val)))
```

```
The length of X(Independent) Feature is 190000
The length of y(Independent) Feature is 190000
************************************************************
The length of X-validation(Independent) Feature is 10000
The length of y-validation(Independent) Feature is 10000
```

####Convert the `target` - Feature into Neumerical

```
[7]: def convert_target(df):
       return list(pd.get_dummies(y, drop_first = True)[True]),\
       list(pd.get_dummies(y_val, drop_first = True)[True])

     y, y_val = convert_target(df)

     # print(y)
     # print(y_val)
```

####Split the dataset into train and test

```
[ ]: #### Define a function for splitting the dataset into train and test ####
     def split_train_test(df):
       X_train, X_test, y_train, y_test = train_test_split(X, y,\
                                                         test_size = 0.30,␣
      ↪random_state = 42)
       return X_train, X_test, y_train, y_test

     X_train, X_test, y_train, y_test = split_train_test(df)

     #### Display the shape of train and test ####
     print('The shape of X_train is = {}'.format(np.array(X_train).shape),'\n')
     print('The shape of y_train is = {}'.format(np.array(y_train).shape),'\n')
     print('The shape of X_test is  = {}'.format(np.array(X_test).shape),'\n')
     print('The shape of y_test is  = {}'.format(np.array(y_test).shape))
```

```
The shape of X_train is = (133000,)

The shape of y_train is = (133000,)

The shape of X_test is  = (57000,)

The shape of y_test is  = (57000,)
```

#Install the `Transformers`

The link of `transformers` image:

https://github.com/atikul-islam-sajib/200K-SHORT-TEXTS-FOR-HUMOR-DETECTION-for-Human-Resourceful-task-using-BERT—DistilBert—Update/blob/main/T1.png

https://github.com/atikul-islam-sajib/200K-SHORT-TEXTS-FOR-HUMOR-DETECTION-for-Human-Resourceful-task-using-BERT—DistilBert—Update/blob/main/T2.png

- Use `DistilBert` architecture to complete this `HUMOR` detection TASK

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

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-
wheels/public/simple/
Collecting transformers
  Downloading transformers-4.27.4-py3-none-any.whl (6.8 MB)
                           6.8/6.8 MB
86.9 MB/s eta 0:00:00
Collecting tokenizers!=0.11.3,<0.14,>=0.11.1
  Downloading
tokenizers-0.13.3-cp39-cp39-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (7.8
MB)
                           7.8/7.8 MB
90.0 MB/s eta 0:00:00
Requirement already satisfied: pyyaml>=5.1 in
```

```
/usr/local/lib/python3.9/dist-packages (from transformers) (6.0)
Requirement already satisfied: numpy>=1.17 in /usr/local/lib/python3.9/dist-
packages (from transformers) (1.22.4)
Requirement already satisfied: filelock in /usr/local/lib/python3.9/dist-
packages (from transformers) (3.10.7)
Requirement already satisfied: requests in /usr/local/lib/python3.9/dist-
packages (from transformers) (2.27.1)
Collecting huggingface-hub<1.0,>=0.11.0
  Downloading huggingface_hub-0.13.4-py3-none-any.whl (200 kB)
                         200.1/200.1 kB
26.3 MB/s eta 0:00:00
Requirement already satisfied: tqdm>=4.27 in
/usr/local/lib/python3.9/dist-packages (from transformers) (4.65.0)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.9/dist-
packages (from transformers) (23.0)
Requirement already satisfied: regex!=2019.12.17 in
/usr/local/lib/python3.9/dist-packages (from transformers) (2022.10.31)
Requirement already satisfied: typing-extensions>=3.7.4.3 in
/usr/local/lib/python3.9/dist-packages (from huggingface-
hub<1.0,>=0.11.0->transformers) (4.5.0)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.9/dist-
packages (from requests->transformers) (3.4)
Requirement already satisfied: certifi>=2017.4.17 in
/usr/local/lib/python3.9/dist-packages (from requests->transformers) (2022.12.7)
Requirement already satisfied: charset-normalizer~=2.0.0 in
/usr/local/lib/python3.9/dist-packages (from requests->transformers) (2.0.12)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in
/usr/local/lib/python3.9/dist-packages (from requests->transformers) (1.26.15)
Installing collected packages: tokenizers, huggingface-hub, transformers
Successfully installed huggingface-hub-0.13.4 tokenizers-0.13.3
transformers-4.27.4
```

####Call the Tokenizer of `DistilBert` with respect to convert the train and test in an appropraite format.

- We will use `distilbert-base-uncased` model for this `HUMOR DETECTION` problem. Because, DistilBert is `lighter, cheaper, smaller, and Faster` and able to provide the equivalent performance as like BERT. In shorts, DistilBert using BERT architecture in the concept of `distillation the knowledge.`*
- Using `DistilBert` - `distilbert-base-uncased` containing 6 layers, 768-heads, 65M parameters.
- The link of `DistilBERT` architecture is: https://github.com/atikul-islam-sajib/200K-SHORT-TEXTS-FOR-HUMOR-DETECTION-for-Human-Resourceful-task-using-BERT—DistilBert—Update/blob/main/unnamed.jpg

[ ]:
```python
from transformers import DistilBertTokenizerFast
tokenizer = DistilBertTokenizerFast.from_pretrained('distilbert-base-uncased')
```

```
try:
  train_encodings = tokenizer(list(X_train), padding = 'max_length', truncation␣
↪= True, max_length = 50)
  test_encodings  = tokenizer(list(X_test),  padding = 'max_length', truncation␣
↪= True, max_length = 50)
  val_encodings   = tokenizer(list(X_val),   padding = 'max_length', truncation␣
↪= True, max_length = 50)
except SyntaxError as e:
  print(e.with_traceback)
else:
  print(train_encodings.keys())
  print(test_encodings.keys())
  print(val_encodings.keys())
```

```
Downloading (…)okenizer_config.json:   0%|          | 0.00/28.0 [00:00<?, ?B/s]

Downloading (…)solve/main/vocab.txt:   0%|          | 0.00/232k [00:00<?, ?B/s]

Downloading (…)/main/tokenizer.json:   0%|          | 0.00/466k [00:00<?, ?B/s]

Downloading (…)lve/main/config.json:   0%|          | 0.00/483 [00:00<?, ?B/s]

dict_keys(['input_ids', 'attention_mask'])
dict_keys(['input_ids', 'attention_mask'])
dict_keys(['input_ids', 'attention_mask'])
```

####Convert `encodings` with respect to Dataset Objects

- We will convert those into `tensor` because we are going to use `TensorFlow`

```
[ ]: import tensorflow as tf
     #### Define a function that is able to convert the dataset-object ####
     def convert_dataset_object(df):
       if df.empty:
         raise Exception('DataFrame is empty'.capitalize())
       else:
         #### For the training dataset ####
         train_dataset = tf.data.Dataset.from_tensor_slices((
             dict(train_encodings),
             y_train
         ))
         #### For the testing dataset ####
         test_dataset = tf.data.Dataset.from_tensor_slices((
             dict(test_encodings),
             y_test
         ))
         #### For the validation dataset ####
         val_dataset = tf.data.Dataset.from_tensor_slices((
             dict(val_encodings),
             y_val
         ))

         return train_dataset, test_dataset, val_dataset

     #### Call this function with this current directory ####
     try:
       train_dataset, test_dataset, val_dataset = convert_dataset_object(df)
     except Exception as e:
       print(e.with_traceback)
     except TypeError as e:
       print(e)
     else:
       print('Conversion into Dataset objects is done successfully'.capitalize())
```

```
Conversion into dataset objects is done successfully
```

####Do the `Hyper Parameter Tuning` to find the best `Arguments` for the `DistilBert`.

- Define some value with respect to the arguments and check which one is able to provide the best model performance. I used few paramters due to `memory` issue.

```
[ ]: #### Define some parameters with respect to the DistilBert ####
num_train_epochs = [2, 3]
per_device_train_batch_size = [8]
per_device_eval_batch_size = [16, 64]
eval_steps = [2]

#### Make the Combinations ####
import itertools

list_combined = [num_train_epochs,\
                 per_device_train_batch_size,\
                 per_device_eval_batch_size ,\
                 eval_steps]

combinations = list(itertools.product(*list_combined))

#### Print the combinations ####
print(combinations)
```

```
[(2, 8, 16, 2), (2, 8, 64, 2), (3, 8, 16, 2), (3, 8, 64, 2)]
```

```
[ ]: from transformers import TFDistilBertForSequenceClassification, TFTrainer,␣
  ↪TFTrainingArguments
#### Define an empty dictionary ####
dict_evaluation_matrix = {}
#### Define a function that is responsible for the finding the best Param ####
```

```
def find_best_arguments(df):
  if df.empty:
    raise Exception('DataFrame is empty'.capitalize())
  else:

    for index, each_combination in enumerate(combinations):
      print('{} comination is running'.format(index + 1))

      training_args = TFTrainingArguments(
        #### output directory ####
        output_dir='./results-check',
        #### total number of training epochs ####
        num_train_epochs = each_combination[0],
        #### batch size per device during training ####
        per_device_train_batch_size = each_combination[1],
        #### batch size for evaluation ####
        per_device_eval_batch_size = each_combination[2],
        #### number of warmup steps for learning rate scheduler ####
        warmup_steps = 500,
        #### strength of weight decay ####
        weight_decay = 0.01,
        #### directory for storing logs ####
        logging_dir = './logs-check',
        #### Logging steps ####
        logging_steps = 10,
        #### Eval_steps ####
        eval_steps = each_combination[3]
      )
      #### Call the model of DistilBert ####
      with training_args.strategy.scope():
        model = TFDistilBertForSequenceClassification.
↪from_pretrained("distilbert-base-uncased")

      #### This is responsible to train the DistilBert ####
      trainer = TFTrainer(
        #### the instantiated Transformers model to be trained ####
        model = model,
        #### training arguments, defined above ####
        args = training_args,
        #### training dataset ####
        train_dataset = train_dataset,
        ### evaluation dataset ####
        eval_dataset = test_dataset
      )

      #### This is responsible for training ####
      trainer.train()
      #### Predict the Model ####

      output = trainer.predict(test_dataset)[0]
      output = np.argmax(output, axis = -1)

      #### Print the Evalution Matrix ####
      print(accuracy_score(output, y_test))
      print(precision_score(output, y_test))
      print(recall_score(output, y_test))
      print(f1_score(output, y_test))

      #### Store this into a dictionary for further use ####
      dict_evaluation_matrix[index] = [accuracy_score(output, y_test)]

      print('\n\n')

try:
  dict_eval_matrix = find_best_arguments(df)
except Exception as e:
  print(e)
else:
  print('The dictonary contains: {}\n'.format(dict_eval_matrix))
```

```
1 comination is running

Downloading tf_model.h5:   0%|          | 0.00/363M [00:00<?, ?B/s]

Some layers from the model checkpoint at distilbert-base-uncased were not used
when initializing TFDistilBertForSequenceClassification: ['vocab_transform',
'vocab_layer_norm', 'vocab_projector', 'activation_13']
- This IS expected if you are initializing TFDistilBertForSequenceClassification
from the checkpoint of a model trained on another task or with another
architecture (e.g. initializing a BertForSequenceClassification model from a
BertForPreTraining model).
- This IS NOT expected if you are initializing
TFDistilBertForSequenceClassification from the checkpoint of a model that you
expect to be exactly identical (initializing a BertForSequenceClassification
model from a BertForSequenceClassification model).
Some layers of TFDistilBertForSequenceClassification were not initialized from
the model checkpoint at distilbert-base-uncased and are newly initialized:
['dropout_19', 'pre_classifier', 'classifier']
You should probably TRAIN this model on a down-stream task to be able to use it
for predictions and inference.

0.9852631578947368
0.9829637451601548
0.9874124885085921
0.9851830946165243
```

```
2 comination is running
```

```
Some layers from the model checkpoint at distilbert-base-uncased were not used
when initializing TFDistilBertForSequenceClassification: ['vocab_transform',
'vocab_layer_norm', 'vocab_projector', 'activation_13']
- This IS expected if you are initializing TFDistilBertForSequenceClassification
from the checkpoint of a model trained on another task or with another
architecture (e.g. initializing a BertForSequenceClassification model from a
BertForPreTraining model).
- This IS NOT expected if you are initializing
TFDistilBertForSequenceClassification from the checkpoint of a model that you
expect to be exactly identical (initializing a BertForSequenceClassification
model from a BertForSequenceClassification model).
Some layers of TFDistilBertForSequenceClassification were not initialized from
the model checkpoint at distilbert-base-uncased and are newly initialized:
['dropout_39', 'pre_classifier', 'classifier']
You should probably TRAIN this model on a down-stream task to be able to use it
for predictions and inference.
```

```
0.9851228070175438
0.9824005631819782
0.9876849033901904
0.9850356462200891
```

```
3 comination is running
```

```
Some layers from the model checkpoint at distilbert-base-uncased were not used
when initializing TFDistilBertForSequenceClassification: ['vocab_transform',
'vocab_layer_norm', 'vocab_projector', 'activation_13']
- This IS expected if you are initializing TFDistilBertForSequenceClassification
from the checkpoint of a model trained on another task or with another
architecture (e.g. initializing a BertForSequenceClassification model from a
BertForPreTraining model).
- This IS NOT expected if you are initializing
TFDistilBertForSequenceClassification from the checkpoint of a model that you
expect to be exactly identical (initializing a BertForSequenceClassification
model from a BertForSequenceClassification model).
Some layers of TFDistilBertForSequenceClassification were not initialized from
the model checkpoint at distilbert-base-uncased and are newly initialized:
['pre_classifier', 'dropout_59', 'classifier']
You should probably TRAIN this model on a down-stream task to be able to use it
for predictions and inference.
```

```
0.9854736842105263
0.9817317845828933
```

```
0.9890425531914894
0.9853736089030206
```

```
4 comination is running
```

```
Some layers from the model checkpoint at distilbert-base-uncased were not used
when initializing TFDistilBertForSequenceClassification: ['vocab_transform',
'vocab_layer_norm', 'vocab_projector', 'activation_13']
- This IS expected if you are initializing TFDistilBertForSequenceClassification
from the checkpoint of a model trained on another task or with another
architecture (e.g. initializing a BertForSequenceClassification model from a
BertForPreTraining model).
- This IS NOT expected if you are initializing
TFDistilBertForSequenceClassification from the checkpoint of a model that you
expect to be exactly identical (initializing a BertForSequenceClassification
model from a BertForSequenceClassification model).
Some layers of TFDistilBertForSequenceClassification were not initialized from
the model checkpoint at distilbert-base-uncased and are newly initialized:
['classifier', 'pre_classifier', 'dropout_79']
You should probably TRAIN this model on a down-stream task to be able to use it
for predictions and inference.
```

```
0.9854561403508771
0.9815557902147132
0.989180944272995
0.9853536156605008
```

```
The dictonary contains: None
```

[ ]:
```
#### Find the best parameters for the DistilBert ####
print("The dictionary contains:\n",\
      dict(sorted(dict_evaluation_matrix.items(),\
                  key = lambda item: item[1],\
                  reverse = True)))
```

```
The dictionary contains:
 {2: [0.9854736842105263], 3: [0.9854561403508771], 0: [0.9852631578947368], 1:
[0.9851228070175438]}
```

[ ]:
```
'''
Here, It is giving the highest accuracy that is 98.61% with the parameters
num_train_epochs = 2
per_device_train_batch_size = 8
```

```
per_device_eval_batch_size = 16
eval_steps = 2
'''
```

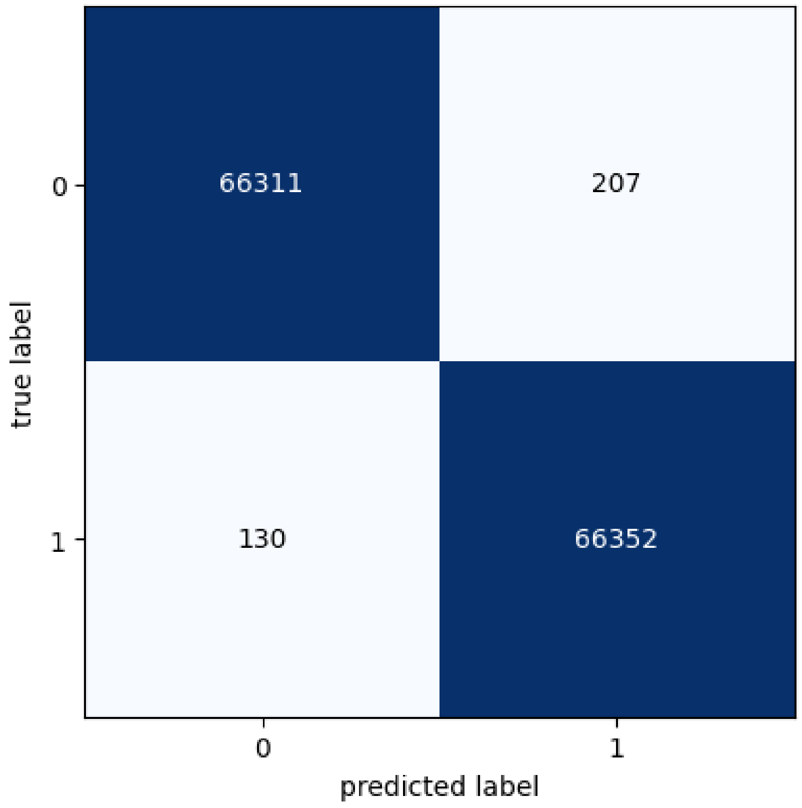
```
[ ]: '\nHere, It is giving the highest accuracy that is 98.61% with the
     parameters\nnum_train_epochs = 2\nper_device_train_batch_size =
     8\nper_device_eval_batch_size = 16\neval_steps = 2\n'
```

####Use those best parameters and check the `testing` and `validation` performance

- The `confusion matrix` is used to evaluate the model(`TESTING & VALIDATION AS WELL`). The link of this matrix's figure is: https://github.com/atikul-islam-sajib/200K-SHORT-TEXTS-FOR-HUMOR-DETECTION-for-Human-Resourceful-task-using-BERT—DistilBert—Update/blob/main/unnamed.png

```
[ ]: #### Define a function that would be responsible for training the model ####
def train_model(epochs = None, train_batch_size = None,\
                eval_batch_Size = None, eval_steps = None,\
                train_dataset = None, test_dataset = None):

    training_args = TFTrainingArguments(
        output_dir ='./results',
        num_train_epochs = epochs,
        per_device_train_batch_size = train_batch_size,
        per_device_eval_batch_size = eval_batch_Size,
        warmup_steps = 500,
        weight_decay = 0.01,
        logging_dir = './logs',
        logging_steps = 10,
        eval_steps = eval_steps
    )

    with training_args.strategy.scope():
        model = TFDistilBertForSequenceClassification.
 ↪from_pretrained("distilbert-base-uncased")

    trainer = TFTrainer(
        model = model,
        args = training_args,
        train_dataset = train_dataset,
        eval_dataset = test_dataset
    )

    trainer.train()

    output = trainer.predict(train_dataset)[0]
    output = np.argmax(output, axis = -1)
    print("The accuracy is  = ", accuracy_score(output, y_train),'\n')
    print("The precision is = ", precision_score(output, y_train),'\n')
    print("The recall is    = ", recall_score(output, y_train),'\n')
    print("The f1_score is  = ", f1_score(output, y_train),'\n')


    print('*'*120,'\n')

    print('The confusion Matrix of training is given below:\n')

    confusion_mat = confusion_matrix(output, y_train)

    fig, ax = plot_confusion_matrix(conf_mat = confusion_mat)

    plt.show()

    return trainer

try:
  trainer = train_model(epochs = 2,\
                        train_batch_size = 8,\
                        eval_batch_Size = 16,\
                        eval_steps = 2,\
                        train_dataset = train_dataset,\
                        test_dataset = test_dataset)
except Exception as e:
  print(e)
else:
  print('Training is completed'.title())
```

```
Some layers from the model checkpoint at distilbert-base-uncased were not used
when initializing TFDistilBertForSequenceClassification: ['vocab_transform',
'vocab_layer_norm', 'vocab_projector', 'activation_13']
- This IS expected if you are initializing TFDistilBertForSequenceClassification
from the checkpoint of a model trained on another task or with another
architecture (e.g. initializing a BertForSequenceClassification model from a
BertForPreTraining model).
- This IS NOT expected if you are initializing
TFDistilBertForSequenceClassification from the checkpoint of a model that you
expect to be exactly identical (initializing a BertForSequenceClassification
model from a BertForSequenceClassification model).
Some layers of TFDistilBertForSequenceClassification were not initialized from
the model checkpoint at distilbert-base-uncased and are newly initialized:
['dropout_99', 'pre_classifier', 'classifier']
You should probably TRAIN this model on a down-stream task to be able to use it
for predictions and inference.

The accuracy is  =  0.9974661654135338
The precision is =  0.996889977313361

The recall is     =  0.9980445834962847

The f1_score is  =  0.9974669462797183

********************************************************************************
****************************************

The confusion Matrix of training is given below:
```



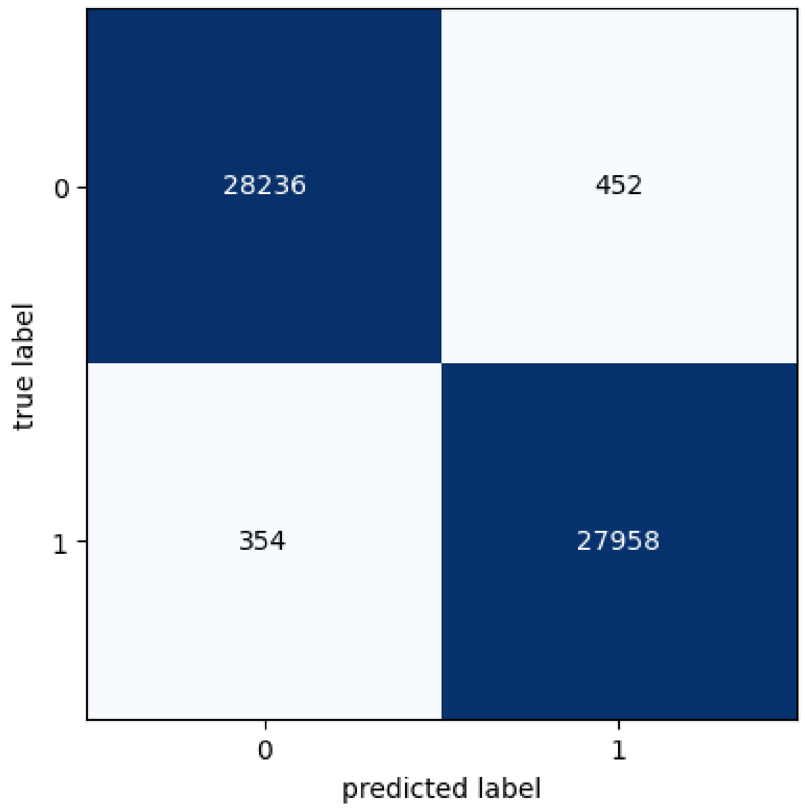
```
Training Is Completed
```

####Do the Evalution for the `Testing, & Validation` dataset

```
[ ]: #### Evalution for Testing dataset ####
     output = trainer.predict(test_dataset)[0]
     output = np.argmax(output, axis = -1)

     print("The accuracy is  = ", accuracy_score(output, y_test),'\n')
```

```python
print("The precision is = ", precision_score(output, y_test),'\n')
print("The recall is    = ", recall_score(output, y_test),'\n')
print("The f1_score is  = ", f1_score(output, y_test),'\n')

print('*'*120,'\n')

print('The classification Matrix is given below.\n')

print(classification_report(output, y_test),'\n')

print('*'*120,'\n')

print('The confusion Matrix is given below:\n')

confusion_mat = confusion_matrix(output, y_test)

fig, ax = plot_confusion_matrix(conf_mat = confusion_mat)

plt.show()
```

```
The accuracy is  =  0.985859649122807

The precision is =  0.9840901091165083

The recall is    =  0.9874964679287934

The f1_score is  =  0.9857903458975354

********************************************************************************
****************************************

The classification Matrix is given below.

              precision    recall  f1-score   support

           0       0.99      0.98      0.99     28688
           1       0.98      0.99      0.99     28312

    accuracy                           0.99     57000
   macro avg       0.99      0.99      0.99     57000
weighted avg       0.99      0.99      0.99     57000


********************************************************************************
****************************************

The confusion Matrix is given below:
```

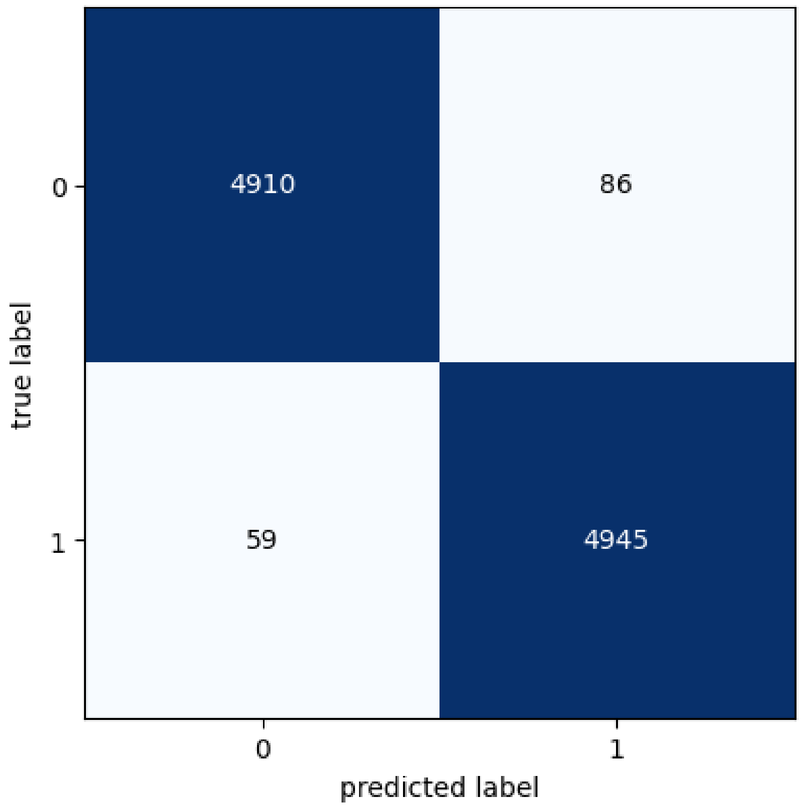


	0	1
0	28236	452
1	354	27958

```
[ ]: #### Evalution for Validation dataset ####
output = trainer.predict(val_dataset)[0]
output = np.argmax(output, axis = -1)

print("The accuracy is  = ", accuracy_score(output, y_val),'\n')
print("The precision is = ", precision_score(output, y_val),'\n')
print("The recall is    = ", recall_score(output, y_val),'\n')
print("The f1_score is  = ", f1_score(output, y_val),'\n')

print('*'*120,'\n')

print('The classification Matrix is given below.\n')

print(classification_report(output, y_val),'\n')

print('*'*120,'\n')

print('The confusion Matrix is given below:\n')
```

```
confusion_mat = confusion_matrix(output, y_val)

fig, ax = plot_confusion_matrix(conf_mat = confusion_mat)

plt.show()
```

```
The accuracy is  =  0.9855

The precision is =  0.9829059829059829

The recall is    =  0.9882094324540368

The f1_score is  =  0.9855505729945192

********************************************************************************
****************************************

The classification Matrix is given below.

              precision    recall  f1-score   support

           0       0.99      0.98      0.99      4996
           1       0.98      0.99      0.99      5004

    accuracy                           0.99     10000
   macro avg       0.99      0.99      0.99     10000
weighted avg       0.99      0.99      0.99     10000


********************************************************************************
****************************************

The confusion Matrix is given below:
```



####Do the KFold Cross validation with respect to K- value = 3.

- K - value 3 because if we increase K-value the memory issue comes and it would be computationaly expensive for this model.

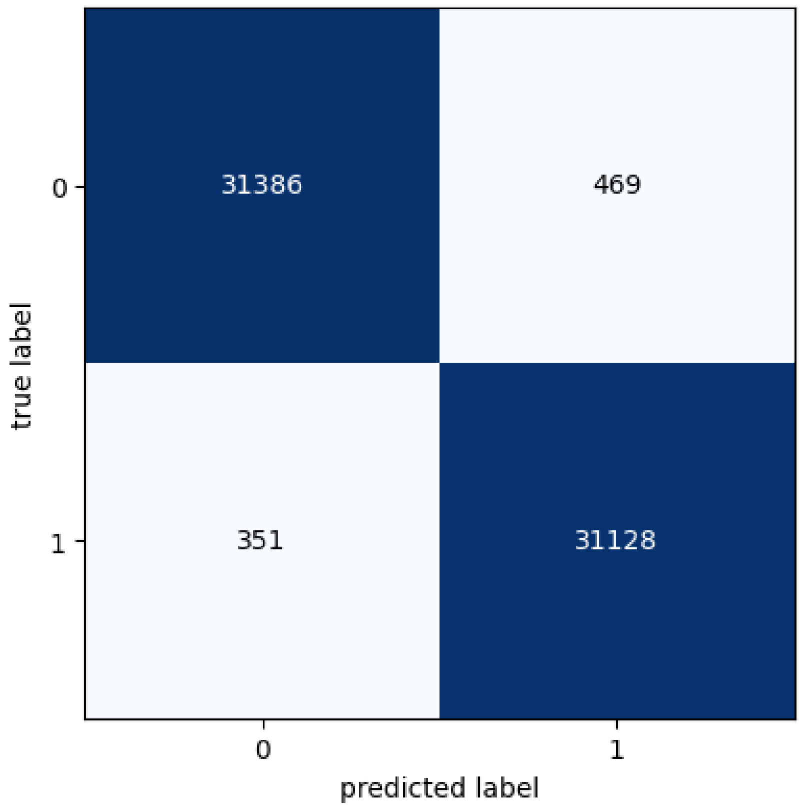
```
[ ]: Kfold = KFold(n_splits = 3, shuffle = True, random_state = 42)
     acc, pre, recall, f1 = [], [], [], []
     count = 1
     for train_index, test_index in Kfold.split(X):

       print('Cross validation is {} running'.format(count),'\n')

       X = np.array(X)
       y = np.array(y)

       X_train, X_test = X[train_index], X[test_index]
       y_train, y_test = y[train_index], y[test_index]

       X = list(X)
       y = list(y)
```

```
train_encodings = tokenizer(list(X_train), padding = 'max_length',␣
↪truncation=True, max_length = 50)
test_encodings  = tokenizer(list(X_test),  padding = 'max_length',␣
↪truncation=True, max_length = 50)

train_dataset = tf.data.Dataset.from_tensor_slices((
  dict(train_encodings),
  y_train
))

test_dataset = tf.data.Dataset.from_tensor_slices((
    dict(test_encodings),
    y_test
))

from transformers import TFDistilBertForSequenceClassification, TFTrainer,␣
↪TFTrainingArguments

training_args = TFTrainingArguments(
    output_dir = './results',          # output directory
    num_train_epochs = 2,              # total number of training epochs
    per_device_train_batch_size = 8,   # batch size per device during training
    per_device_eval_batch_size = 16,   # batch size for evaluation
    warmup_steps = 500,                # number of warmup steps for learning␣
↪rate scheduler
    weight_decay = 0.01,               # strength of weight decay
    logging_dir = './logs',            # directory for storing logs
    logging_steps = 10,
    eval_steps = 2
)

with training_args.strategy.scope():
  model = TFDistilBertForSequenceClassification.
↪from_pretrained("distilbert-base-uncased")

trainer = TFTrainer(
  model = model,                        # the instantiated Transformers␣
↪model to be trained
  args = training_args,                 # training arguments, defined above
  train_dataset = train_dataset,        # training dataset
  eval_dataset  = test_dataset          # evaluation dataset
)

trainer.train()
```

```
output = trainer.predict(test_dataset)[0]
output = np.argmax(output, axis = -1)

print("The accuracy score  = ", accuracy_score(output, y_test))
print("The precision score = ",precision_score(output, y_test))
print("The recall score = ",recall_score(output, y_test))
print("The f1_score score = ",f1_score(output, y_test))

print('*'*120,'\n')

print('The confusion Matrix is given below:\n')

confusion_mat = confusion_matrix(output, y_test)

fig, ax = plot_confusion_matrix(conf_mat = confusion_mat)

plt.show()

acc.append(accuracy_score(output, y_test))
pre.append(precision_score(output, y_test))
recall.append(recall_score(output, y_test))
f1.append(f1_score(output, y_test))

count = count + 1
```

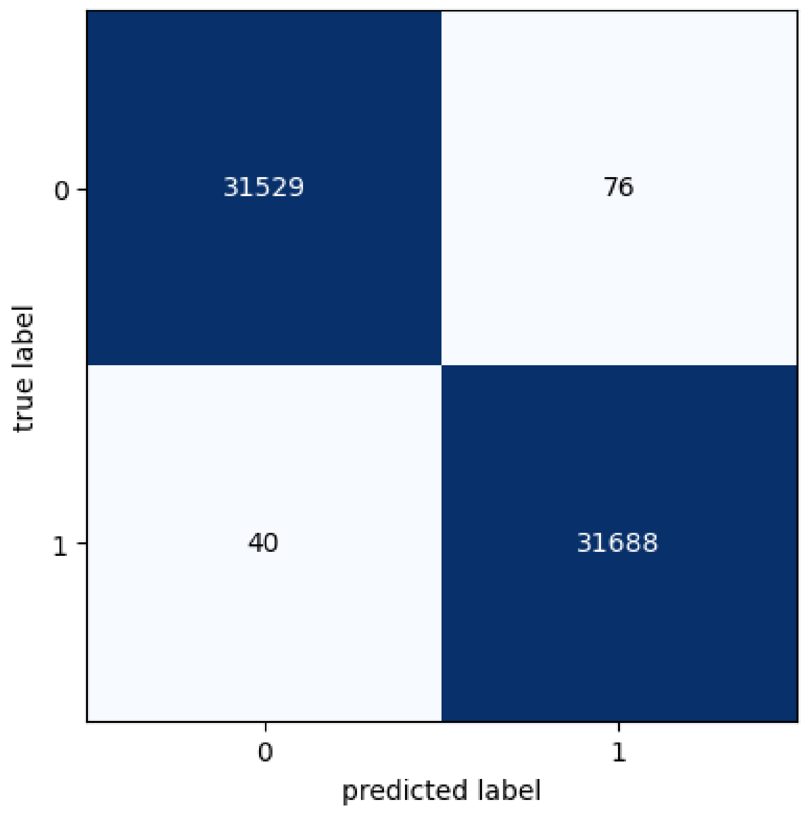
```
Cross validation is 1 running


Some layers from the model checkpoint at distilbert-base-uncased were not used
when initializing TFDistilBertForSequenceClassification: ['vocab_transform',
'vocab_layer_norm', 'vocab_projector', 'activation_13']
- This IS expected if you are initializing TFDistilBertForSequenceClassification
from the checkpoint of a model trained on another task or with another
architecture (e.g. initializing a BertForSequenceClassification model from a
BertForPreTraining model).
- This IS NOT expected if you are initializing
TFDistilBertForSequenceClassification from the checkpoint of a model that you
expect to be exactly identical (initializing a BertForSequenceClassification
model from a BertForSequenceClassification model).
Some layers of TFDistilBertForSequenceClassification were not initialized from
the model checkpoint at distilbert-base-uncased and are newly initialized:
['dropout_119', 'pre_classifier', 'classifier']
You should probably TRAIN this model on a down-stream task to be able to use it
for predictions and inference.
```

```
The accuracy score  =  0.9870527678656014
The precision score =  0.9851568186853182
The recall score =  0.9888497093300296
The f1_score score =  0.9869998097533135
********************************************************************************
****************************************

The confusion Matrix is given below:
```



```
Cross validation is 2 running


Some layers from the model checkpoint at distilbert-base-uncased were not used
when initializing TFDistilBertForSequenceClassification: ['vocab_transform',
'vocab_layer_norm', 'vocab_projector', 'activation_13']
- This IS expected if you are initializing TFDistilBertForSequenceClassification
from the checkpoint of a model trained on another task or with another
architecture (e.g. initializing a BertForSequenceClassification model from a
BertForPreTraining model).
- This IS NOT expected if you are initializing
TFDistilBertForSequenceClassification from the checkpoint of a model that you
expect to be exactly identical (initializing a BertForSequenceClassification
model from a BertForSequenceClassification model).
Some layers of TFDistilBertForSequenceClassification were not initialized from
```

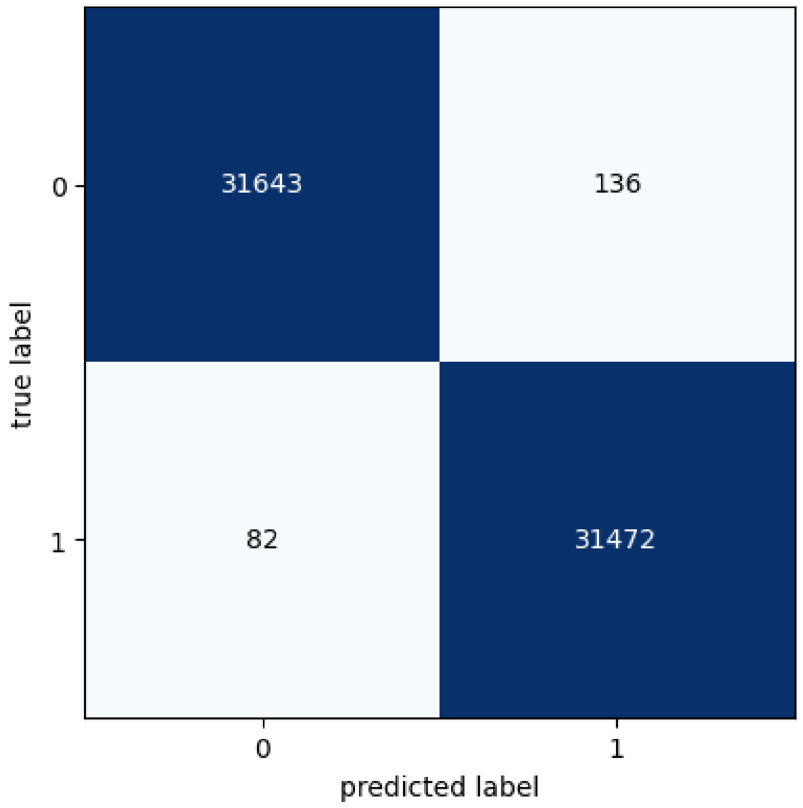
```
the model checkpoint at distilbert-base-uncased and are newly initialized:
['dropout_139', 'pre_classifier', 'classifier']
You should probably TRAIN this model on a down-stream task to be able to use it
for predictions and inference.

The accuracy score  =  0.9981684114126916
The precision score =  0.9976073542375016
The recall score =  0.9987392839132627
The f1_score score =  0.9981729981729982
********************************************************************************
****************************************

The confusion Matrix is given below:
```



```
Cross validation is 3 running


Some layers from the model checkpoint at distilbert-base-uncased were not used
when initializing TFDistilBertForSequenceClassification: ['vocab_transform',
'vocab_layer_norm', 'vocab_projector', 'activation_13']
- This IS expected if you are initializing TFDistilBertForSequenceClassification
from the checkpoint of a model trained on another task or with another
```

```
architecture (e.g. initializing a BertForSequenceClassification model from a
BertForPreTraining model).
- This IS NOT expected if you are initializing
TFDistilBertForSequenceClassification from the checkpoint of a model that you
expect to be exactly identical (initializing a BertForSequenceClassification
model from a BertForSequenceClassification model).
Some layers of TFDistilBertForSequenceClassification were not initialized from
the model checkpoint at distilbert-base-uncased and are newly initialized:
['dropout_159', 'pre_classifier', 'classifier']
You should probably TRAIN this model on a down-stream task to be able to use it
for predictions and inference.
```

```
The accuracy score  =  0.9965578766204033
The precision score =  0.9956972918248544
The recall score =  0.9974012803448057
The f1_score score =  0.9965485576770843
********************************************************************************
****************************************

The confusion Matrix is given below:
```



```
[ ]: #### Show the Evaluation Matrix after using KFold -3 ####
     print('Using KFold - 3, the accuracy of this model is {}'.format(np.array(acc).
      ↪mean()),'\n')
     print('Using KFold - 3, the precision of this model is {}'.format(np.array(pre).
      ↪mean()),'\n')
     print('Using KFold - 3, the recall of this model is {}'.format(np.array(recall).
      ↪mean()),'\n')
     print('Using KFold - 3, the f1 score of this model is {}'.format(np.array(f1).
      ↪mean()))
```

```
Using KFold - 3, the accuracy of this model is 0.9939263519662321

Using KFold - 3, the precision of this model is 0.9928204882492248

Using KFold - 3, the recall of this model is 0.9949967578626993

Using KFold - 3, the f1 score of this model is 0.9939071218677986
```

#For Single Data Prediction - HUMOR DETECTION

```
[ ]: from transformers import DistilBertTokenizerFast
     tokenizer = DistilBertTokenizerFast.from_pretrained('distilbert-base-uncased')

     class humor_prediction:

       def __init__(self, text):
         print('\n\nThe Prediction is Given below'.title())
         print('*'*70,'\n\n')
         self.text = text
         self.text_preprocessing()

       def text_preprocessing(self):
         preprocessing = tf.data.Dataset.from_tensor_slices((
           dict(tokenizer((self.text),\
                          padding = 'max_length',\
                          truncation=True,\
                          max_length = 50)),
                          [1]
               ))

         self.prediction(preprocessing)

       def prediction(self, preprocessing_result):
         output = trainer.predict(preprocessing_result)[0]
         output = np.argmax(output, axis = -1)
         if output[0] == 0:
           print("The prediction is False".capitalize())
         else:
```

```
      print("The prediction is True".capitalize())

#### Call the class and predict the model ####
try:
  print('Welcome to the detect Humor'.title())
  print('*'*70,'\n')
  text = (str(input("Enter the text:\n\n")))
except NameError as e:
  print(e.with_traceback)
else:
  humor_detect = humor_prediction([text])
```

```
Welcome To The Detect Humor
**********************************************************************

Enter the text:

Why didn't the mentally challenged kid finish his math test in time? because he
was too slow.


The Prediction Is Given Below
**********************************************************************



The prediction is true
```

THIS IS END OF TEST TASK - Using BERT ar

#Use LSTM - to solve this Problem - It is APARTfrom the test task. It's done for my analysis purpose

- Use it For understanding how our RNN - LSTM variants react with respect to this dataset.

[ ]:
```
#### Show the dummy_df dataset ####
dummy_df.head()
```

[ ]:
```
                                                text  humor  \
0                          joe biden rule bid guy run  False
1       watch darvish gave hitter whiplash slow pitch  False
2                  what call turtl without shell dead   True
3                           reason elect feel person  False
4  pasco polic shot mexican migrant behind new au…  False

                                          PCA
0  [-0.040401537656080184, -0.014151492910330365]
1    [-0.04797831400216986, -0.01951040719741723]
2      [0.41988102033822045, 0.02699406957930625]
3    [-0.04560593615863668, -0.02244263774563714]
```

```
4   [-0.048008664292122305, -0.03480864579227028]
```

[ ]:
```
#### Define a class for user defined Exception ####
class Conversion_Exception(Exception):
  def __init__(self, message):
    self.message = message.capitalize()

#### Convert the target column into Label Encoding ####
try:
  dummy_df.loc[:, 'humor'] = dummy_df.loc[:, 'humor'].map({True: 1, False: 0})
except Conversion_Exception as e:
  print(e)
else:
  print(dummy_df.head())
```

```
                                              text  humor  \
0                          joe biden rule bid guy run      0
1      watch darvish gave hitter whiplash slow pitch      0
2                    what call turtl without shell dead      1
3                             reason elect feel person      0
4  pasco polic shot mexican migrant behind new au…      0

                                              PCA
0  [-0.040401537656080184, -0.014151492910330365]
1    [-0.04797831400216986, -0.01951040719741723]
2      [0.41988102033822045, 0.02699406957930625]
3    [-0.04560593615863668, -0.02244263774563714]
4   [-0.048008664292122305, -0.03480864579227028]
```

[ ]:
```
##### Find the number of unique vocabulary in this dataframe ######
from tensorflow.keras.preprocessing.text import one_hot, Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences

class TokenException(Exception):
  def __init__(self, message):
    self.message = message.capitalize()
try:
  tokenizer = Tokenizer()
  tokenizer.fit_on_texts(dummy_df.loc[:, 'text'])
  voc_size = tokenizer.word_index
  voc_size = len(voc_size) + 1

except TokenException as e:
  print(e)
else:
  print("# of unique vocabulary size in this dataframe is : ", voc_size)
```

```
# of unique vocabulary size in this dataframe is :  42470
```

[ ]:
```
##### One Hot Representation #####
def OHE(dummy_df):
  return [one_hot(record, voc_size) for record in dummy_df.loc[:, 'text']]

one_hot = OHE(dummy_df)
```

[ ]:
```
##### Find the maxlen of the list #####
maxlen = -1
for list in one_hot:
  if len(list) > maxlen:
    maxlen = len(list)
  else:
    pass

print("The maximum length of this list = ", maxlen)
```

```
The maximum length of this list =  16
```

[ ]:
```
##### Embedded doc #####
embedded_doc = pad_sequences(one_hot, padding = 'pre', maxlen = maxlen + 1)

print(embedded_doc)
```

```
[[    0     0     0 … 18012  5226 26771]
 [    0     0     0 … 30364 10351 32279]
 [    0     0     0 … 34690  9254 10363]
 …
 [    0     0     0 …  6682  7386  6682]
 [    0     0     0 … 24301 20250 23374]
 [    0     0     0 … 25872  6725  4833]]
```

[ ]:
```
###### Split the dataset into train and test #######
X_train, X_test, y_train, y_test = train_test_split(embedded_doc,\
                                                    dummy_df.loc[:, 'humor'],\
                                                    test_size = 0.30,\
                                                    random_state = 42)


print("X_train shape is = ", X_train.shape)
print("y_train shape is = ", y_train.shape)
print("X_test  shape is = ", X_test.shape)
print("y_test  shape is = ", y_test.shape)
```

```
X_train shape is =  (140000, 17)
y_train shape is =  (140000,)
X_test  shape is =  (60000, 17)
y_test  shape is =  (60000,)
```

####LSTM used to predict the model #####

```
[ ]: class activation_exception(Exception):
       def __init__(self, message):
         self.message = message.capitalize()

     def LSTM_MODEL(activated = None):

       if activated == True:
         model = Sequential()
         ##### Create an embedded layer #####
         model.add(Embedding(input_dim = voc_size, output_dim = 300, input_length =␣
      ↪maxlen + 1))
         model.add(Dropout(0.6))

         ##### Create LSTM with 100 neurons #####
         model.add(LSTM(units = 32))
         model.add(Dropout(0.7))

         model.add(Dense(units = 128, activation = 'relu', kernel_initializer =␣
      ↪'he_normal'))
         model.add(Dropout(rate = 0.9))

         ##### Create an output layer #####
         model.add(Dense(units = 1, activation = 'sigmoid'))

         ##### Compile the model #####
         model.compile(optimizer = Adam(learning_rate = 0.0001), loss =␣
      ↪BinaryCrossentropy(), metrics = ['accuracy'])

         ##### Fit the model and run #####
         history_ = model.fit(x = X_train, y = y_train, validation_data = (X_test,␣
      ↪y_test), epochs = 20, batch_size = 1024, verbose = 1)
       else:
         raise Exception('Activated is not possible, Try again !'.capitalize())

       return history_, model

     try:
       history_, model_LSTM = LSTM_MODEL(activated = True)
     except activation_exception as message:
       print(message)
```

```
Epoch 1/20
137/137 [==============================] - 7s 22ms/step - loss: 0.6944 -
accuracy: 0.5098 - val_loss: 0.6886 - val_accuracy: 0.6705
Epoch 2/20
137/137 [==============================] - 2s 15ms/step - loss: 0.6848 -
accuracy: 0.5487 - val_loss: 0.6639 - val_accuracy: 0.7845
```

```
Epoch 3/20
137/137 [==============================] - 2s 15ms/step - loss: 0.6126 -
accuracy: 0.6837 - val_loss: 0.4517 - val_accuracy: 0.8748
Epoch 4/20
137/137 [==============================] - 2s 16ms/step - loss: 0.4239 -
accuracy: 0.8319 - val_loss: 0.2692 - val_accuracy: 0.8999
Epoch 5/20
137/137 [==============================] - 2s 15ms/step - loss: 0.3338 -
accuracy: 0.8756 - val_loss: 0.2346 - val_accuracy: 0.9064
Epoch 6/20
137/137 [==============================] - 2s 15ms/step - loss: 0.2975 -
accuracy: 0.8941 - val_loss: 0.2210 - val_accuracy: 0.9110
Epoch 7/20
137/137 [==============================] - 2s 15ms/step - loss: 0.2749 -
accuracy: 0.9042 - val_loss: 0.2141 - val_accuracy: 0.9138
Epoch 8/20
137/137 [==============================] - 2s 15ms/step - loss: 0.2598 -
accuracy: 0.9098 - val_loss: 0.2111 - val_accuracy: 0.9148
Epoch 9/20
137/137 [==============================] - 2s 15ms/step - loss: 0.2485 -
accuracy: 0.9147 - val_loss: 0.2087 - val_accuracy: 0.9164
Epoch 10/20
137/137 [==============================] - 2s 16ms/step - loss: 0.2387 -
accuracy: 0.9182 - val_loss: 0.2076 - val_accuracy: 0.9166
Epoch 11/20
137/137 [==============================] - 2s 15ms/step - loss: 0.2333 -
accuracy: 0.9202 - val_loss: 0.2076 - val_accuracy: 0.9169
Epoch 12/20
137/137 [==============================] - 2s 15ms/step - loss: 0.2251 -
accuracy: 0.9229 - val_loss: 0.2081 - val_accuracy: 0.9169
Epoch 13/20
137/137 [==============================] - 2s 15ms/step - loss: 0.2196 -
accuracy: 0.9249 - val_loss: 0.2091 - val_accuracy: 0.9173
Epoch 14/20
137/137 [==============================] - 2s 15ms/step - loss: 0.2165 -
accuracy: 0.9262 - val_loss: 0.2095 - val_accuracy: 0.9177
Epoch 15/20
137/137 [==============================] - 2s 15ms/step - loss: 0.2113 -
accuracy: 0.9278 - val_loss: 0.2105 - val_accuracy: 0.9175
Epoch 16/20
137/137 [==============================] - 2s 15ms/step - loss: 0.2068 -
accuracy: 0.9298 - val_loss: 0.2102 - val_accuracy: 0.9174
Epoch 17/20
137/137 [==============================] - 2s 15ms/step - loss: 0.2051 -
accuracy: 0.9295 - val_loss: 0.2113 - val_accuracy: 0.9176
Epoch 18/20
137/137 [==============================] - 2s 16ms/step - loss: 0.1993 -
accuracy: 0.9323 - val_loss: 0.2121 - val_accuracy: 0.9180
```

```
Epoch 19/20
137/137 [==============================] - 2s 15ms/step - loss: 0.1959 -
accuracy: 0.9331 - val_loss: 0.2126 - val_accuracy: 0.9179
Epoch 20/20
137/137 [==============================] - 2s 15ms/step - loss: 0.1955 -
accuracy: 0.9333 - val_loss: 0.2130 - val_accuracy: 0.9166
```

####Check the Performance of LSTM

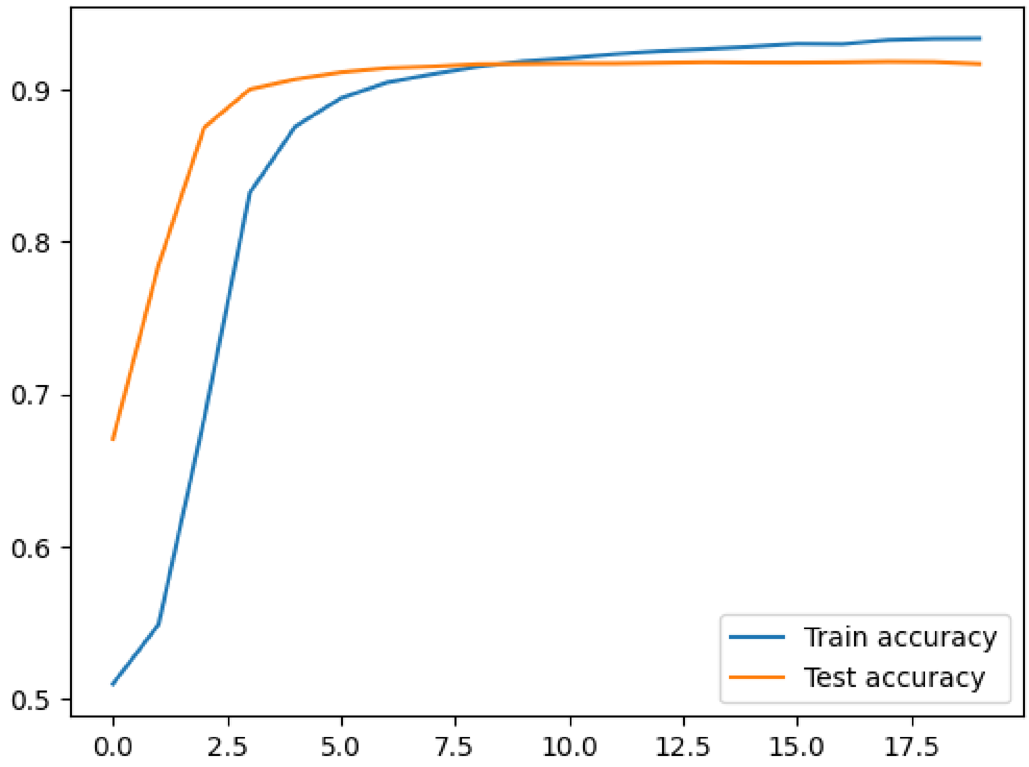
```
[ ]: y_pred = [1 if predicted > 0.5 else 0 for predicted in model_LSTM.
      ↪predict(X_test)]

     print("Using LSTM, accuracy is   = ", accuracy_score(y_pred, y_test))
     print("Using LSTM, recall is     = ", recall_score(y_pred, y_test))
     print("Using LSTM, precision is  = ", precision_score(y_pred, y_test))
     print("Using LSTM, f1 score is   = ", f1_score(y_pred, y_test))
```

```
1875/1875 [==============================] - 4s 2ms/step
Using LSTM, accuracy is   =  0.9166166666666666
Using LSTM, recall is     =  0.9090523707037109
Using LSTM, precision is  =  0.9264231830115317
Using LSTM, f1 score is   =  0.9176555787810456
```

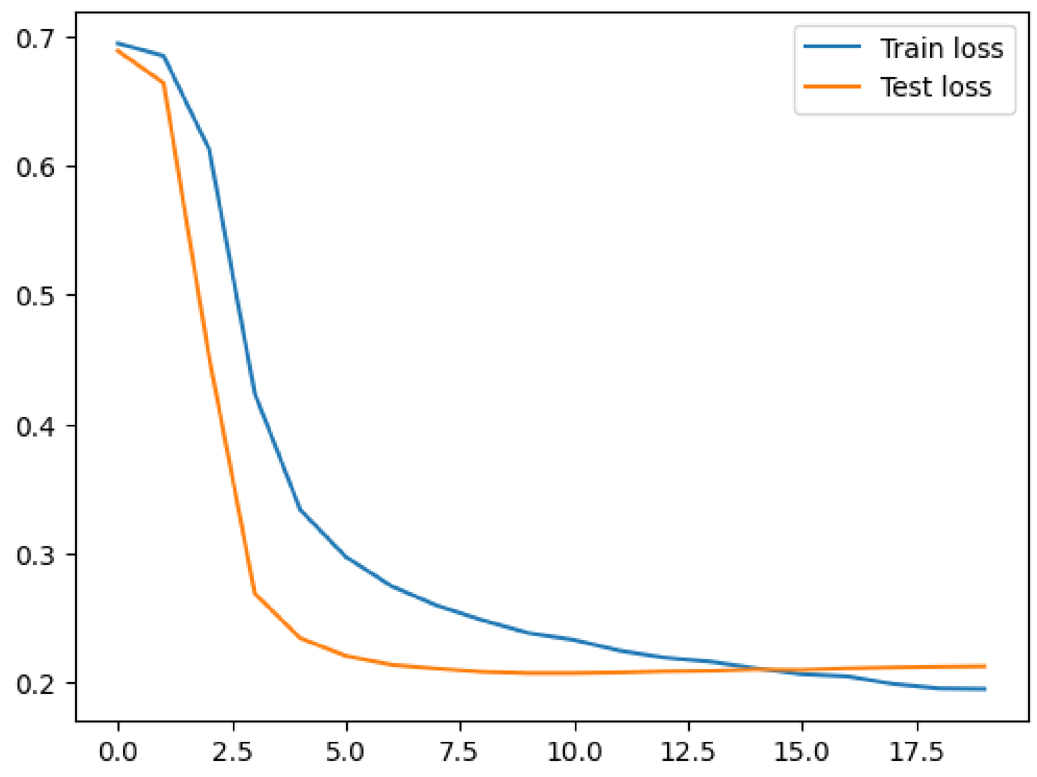
```
[ ]: #### Print the classification matrix ####
     print(classification_report(y_pred, y_test))
```

```
              precision    recall  f1-score   support

           0       0.91      0.92      0.92     29334
           1       0.93      0.91      0.92     30666

    accuracy                           0.92     60000
   macro avg       0.92      0.92      0.92     60000
weighted avg       0.92      0.92      0.92     60000
```

```
[ ]: ###### plot the accuracy and val_accuracy ######
     plt.plot(history_.history['accuracy'], label = 'Train accuracy')
     plt.plot(history_.history['val_accuracy'], label = 'Test accuracy')
     plt.legend()
     plt.show()
```



```
[ ]: ###### plot the train loss and validation loss ######
     plt.plot(history_.history['loss'], label = 'Train loss')
     plt.plot(history_.history['val_loss'], label = 'Test loss')
     plt.legend()
     plt.show()
```



[ ]:
```
!pip install colorama
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-
wheels/public/simple/
Collecting colorama
  Downloading colorama-0.4.6-py2.py3-none-any.whl (25 kB)
Installing collected packages: colorama
Successfully installed colorama-0.4.6
```

#Comments after Building the Model & Using the `DistilBert & LSTM`

[ ]:
```
import colorama
from colorama import Fore

print("It is clearly see that "+ Fore.RED + 'Distilbert Model performace '+ "is␣
↪outperfomred than LSTM Model".title())
```

```
It is clearly see that Distilbert Model performace Is Outperfomred Than

Lstm Model
```