Coats Data Science Assignement

Group No

G3

Group Member Names:

1. S. Guna

Dataset:

https://www.kaggle.com/datasets/gargmanas/sentimental-analysis-for-tweets

1. Business Understanding

Students are expected to identify a classification problem of given insurance_part2_data.csv dataset You have to detail the Business Understanding part of your problem under this heading which basically addresses the following questions.

1. What is the business problem that you are trying to solve?
2. What data do you need to answer the above problem?
3. What are the different sources of data?
4. What kind of analytics task are you performing?

Score: 1 Mark in total (0.25 mark each)

1. Finding if a person is depressed from their use of words in social media, Nowadays Social media becomes a platform where many people speaking about their mental health or their usage reflect their mental health, So identifying a person whether he/she is in depression could definitely save a life.
2. Their Social Media activty (Tweets they posted in twitter)
3. Twitter is the main source of data that being extracted.
4. Classification Whether a person is in depresssion or not based on their tweets.

2. Data Acquisition

2.1 Read the data directly

```
datafile = open("/content/sentiment_tweets3.csv", "r")
datafile.read()
```

```
'Index,message to examine,label (depression result)\n106,"just had a real good moment. i missssssssss him so much, ",0\n217,is reading manga  http://plurk.com/p/mzp1e,0\n220,@comeagainjen http://twitpic.com/2y2lx - http://www.youtube.com/watch?v=zoGfqvh2ME8 ,0\n288,"@lapcat Need to send \'em to my accountant tomorrow. Oddly, I wasn\'t even referring to my taxes. Those are supporting evidence, though. ",0\n540,ADD ME ON MYSPACE!!!  myspace.com/LookThunder,0\n624,so sleepy. good times tonight though ,0\n701,"@SilkCharm re: #nbn as someone already said, does fiber to the home mean we will all at least be regular now ",0\n808,23 or 24ï¿½C possible today. Nice ,0\n1193,nite twitterville  workout in the am  -ciao,0\n1324,"@daNanner Night, darlin\'!  Sweet dreams to you ",0\n1332,Good morning everybody! ,0\n1368,Finally! I just created my WordPress Blog. There\'s already a blog up on the Seattle Coffee Community  ... http://tinyurl.com/c5uufd,0\n1578,kisha they cnt get over u til they get out …'
```

2.2 Code for converting the above downloaded data into a dataframe

```
import pandas as pd
data = pd.read_csv("/content/sentiment_tweets3.csv")
```

2.3 Confirm the data has been correctly by displaying the first 5 and last 5 records.

```
data.head(5)
```

	Index	message to examine	label (depression result)
0	106	just had a real good moment. i missssssssss hi...	0
1	217	is reading manga http://plurk.com/p/mzp1e	0
2	220	@comeagainjen http://twitpic.com/2y2lx - http:...	0

```
data.tail(5)
```

	Index	message to examine	label (depression result)
10309	802309	No Depression by G Herbo is my mood from now o...	1
10310	802310	What do you do when depression succumbs the br...	1
10311	802311	Ketamine Nasal Spray Shows Promise Against Dep...	1
10312	802312	dont mistake a bad day with depression! everyo...	1
10313	802313	0	1

2.4 Display the column headings, statistical information, description and statistical summary of the data.

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10314 entries, 0 to 10313
Data columns (total 3 columns):
 #   Column                     Non-Null Count  Dtype 
---  ------                     --------------  ----- 
 0   Index                      10314 non-null  int64 
 1   message to examine         10314 non-null  object
 2   label (depression result)  10314 non-null  int64 
dtypes: int64(2), object(1)
memory usage: 241.9+ KB
```

2.5 Write your observations from the above.

1. Size of the dataset
2. What type of data attributes are there?
3. Is there any null data that has to be cleaned?

Score: 2 Marks in total (0.25 marks for 2.1, 0.25 marks for 2.2, 0.5 marks for 2.3, 0.25 marks for 2.4, 0.75 marks for 2.5)

1. Size of the dataset = 10,314 rows
2. (i) Index - unique Ids (Not Necessary)

   (ii) Message to examine - Text data or Tweets.

   (iii) Label - **1** if a text is classified as **depressing** text, **0** if a text is classified as **non-depressing** text
3. No null rows are there in the dataset to be cleaned. Index alone need to be dropped.

3. Data Preparation

3.1 Check for

- duplicate data
- missing data
- data inconsistencies

```
duplicate = data[data.duplicated()]

print("Duplicate_rows")
duplicate
```

```
Duplicate_rows
```

Index	message to examine	label (depression result)

There are no duplicate or missing rows in the current dataset.

Data is consistent as well.

3.2 Apply techiniques

- to remove duplicate data
- to impute or remove missing data
- to remove data inconsistencies

Since there are no duplicate or missing rows in the current dataset.

Data is consistent as well.So There is no need to clean data.

3.3 Encode categorical data

The Categorical data is already been encoded as 1 or 0

3.4 Text data

1. Remove special characters
2. Change the case (up-casing and down-casing).
3. Tokenization — process of discretizing words within a document.
4. Filter Stop Words.

```
import nltk
from nltk.stem import PorterStemmer
from nltk.stem import WordNetLemmatizer
from nltk.tokenize import word_tokenize
from nltk.corpus import stopwords, wordnet
import re
import string
nltk.download('omw-1.4')
nltk.download('punkt')
nltk.download('stopwords')
nltk.download('averaged_perceptron_tagger')
nltk.download('wordnet')
```

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

```
data['tokenized']=data['message to examine'].apply(word_tokenize)
data.head(5)
```

	Index	message to examine	label (depression result)	tokenized
0	106	just had a real good moment. i missssssssss hi...	0	[just, had, a, real, good, moment, ., i, misss...
1	217	is reading manga http://plurk.com/p/mzp1e	0	[is, reading, manga, http, :, //plurk.com/p/mz...
2	220	@comeagainjen http://twitpic.com/2y2lx - http:...	0	[@, comeagainjen, http, :, //twitpic.com/2y2lx...
3	288	@lapcat Need to send 'em to my accountant tomo...	0	[@, lapcat, Need, to, send, 'em, to, my, accou...
4	540	ADD ME ON MYSPACE!!!	0	[ADD, ME, ON, MYSPACE, !, !, !,

```
data['lower'] = data['tokenized'].apply(lambda x: [word.lower() for word in x])
data.head(5)
```

	Index	message to examine	label (depression result)	tokenized	lower
0	106	just had a real good moment. i missssssssss hi...	0	[just, had, a, real, good, moment, ., i, misss...	[just, had, a, real, good, moment, ., i, misss...
1	217	is reading manga http://plurk.com/p/mzp1e	0	[is, reading, manga, http, :, //plurk.com/p/mz...	[is, reading, manga, http, :, //plurk.com/p/mz...
2	220	@comeagainjen http://twitpic.com/2y2lx - http:...	0	[@, comeagainjen, http, :, //twitpic.com/2y2lx...	[@, comeagainjen, http, :, //twitpic.com/2y2lx...

```
punc = string.punctuation
data['no_punc'] = data['lower'].apply(lambda x: [word for word in x if word not in punc])
data.head(5)
```

	Index	message to examine	label (depression result)	tokenized	lower	no_punc
0	106	just had a real good moment. i missssssssss hi...	0	[just, had, a, real, good, moment, ., i, misss...	[just, had, a, real, good, moment, ., i, misss...	[just, had, a, real, good, moment, i, missssss...
1	217	is reading manga http://plurk.com/p/mzp1e	0	[is, reading, manga, http, :, //plurk.com/p/mz...	[is, reading, manga, http, :, //plurk.com/p/mz...	[is, reading, manga, http, //plurk.com/p/mzp1e]
2	220	@comeagainjen http://twitpic.com/2y2lx - http:...	0	[@, comeagainjen, http, :, //twitpic.com/2y2lx...	[@, comeagainjen, http, :, //twitpic.com/2y2lx...	[comeagainjen, http, //twitpic.com/2y2lx, http...
3	288	@lapcat Need to send 'em to my accountant tomo...	0	[@, lapcat, Need, to, send, 'em, to, my, accou...	[@, lapcat, need, to, send, 'em, to, my, accou...	[lapcat, need, to, send, 'em, to, my, accounta...
4	540	ADD ME ON MYSPACE!!!	0	[ADD, ME, ON, MYSPACE, !, !, !,	[add, me, on, myspace,	[add, me, on, myspace,

```
stop_words = set(stopwords.words('english'))
data['stopwords_removed'] = data['no_punc'].apply(lambda x: [word for word in x if word not in stop_words])
data.head()
```

	Index	message to examine	label (depression result)	tokenized	lower	no_punc	stopwords_removed
0	106	just had a real good moment. i missssssssss hi...	0	[just, had, a, real, good, moment, ., i, misss...	[just, had, a, real, good, moment, ., i, misss...	[just, had, a, real, good, moment, i, missssss...	[real, good, moment, missssssssss, much]
1	217	is reading manga http://plurk.com/p/mzp1e	0	[is, reading, manga, http, :, //plurk.com/p/mz...	[is, reading, manga, http, :, //plurk.com/p/mz...	[is, reading, manga, http, //plurk.com/p/mzp1e]	[reading, manga, http, //plurk.com/p/mzp1e]
2	220	@comeagainjen http://twitpic.com/2y2lx - http:...	0	[@, comeagainjen, http, :, //twitpic.com/2y2lx...	[@, comeagainjen, http, :, //twitpic.com/2y2lx...	[comeagainjen, http, //twitpic.com/2y2lx, http...	[comeagainjen, http, //twitpic.com/2y2lx, http...
3	288	@lapcat Need to send 'em to my accountant tomo...	0	[@, lapcat, Need, to, send, 'em, to, my, accou...	[@, lapcat, need, to, send, 'em, to, my, accou...	[lapcat, need, to, send, 'em, to, my, accounta...	[lapcat, need, send, 'em, accountant, tomorrow...
4	540	ADD ME ON MYSPACE!!! myspace.com/LookThunder	0	[ADD, ME, ON, MYSPACE, !, !, !, myspace.com/Lo...	[add, me, on, myspace, !, !, !, myspace.com/lo...	[add, me, on, myspace, myspace.com/lookthunder]	[add, myspace, myspace.com/lookthunder]

3.4 Report

Mention and justify the method adopted

- to remove duplicate data, if present
- to impute or remove missing data, if present
- to remove data inconsistencies, if present

OR for textdata

- How many tokens after step 3?
- how may tokens after stop words filtering?

If the any of the above are not present, then also add in the report below.

Score: 2 Marks (based on the dataset you have, the data prepreation you had to do and report typed, marks will be distributed between 3.1, 3.2, 3.3 and 3.4)

I Have check whether there is any duplicate data is present or not. Since there are no duplicate or missing rows in the current dataset. Data is consistent as well.So There is no need to clean data.

Since I'm using text data I performed tokenization,lower case coversion, Specal character removal and Stopword removal.

```
print("Sentence After Tokenizing:", len(data['tokenized'].iloc[0]))
print("Sentence After Stopword Removal:",len(data['stopwords_removed'].iloc[0]))
```

```
Sentence After Tokenizing: 13
Sentence After Stopword Removal: 5
```

```
print("Sentence After Tokenizing:", data['tokenized'].iloc[0])
print("Sentence After Stopword Removal:",data['stopwords_removed'].iloc[0])
```

```
Sentence After Tokenizing: ['just', 'had', 'a', 'real', 'good', 'moment', '.', 'i', 'missssssssss', 'him', 'so', 'much', ',']
Sentence After Stopword Removal: ['real', 'good', 'moment', 'missssssssss', 'much']
```

3.5 Identify the target variables.

- Separate the data from the target such that the dataset is in the form of (X,y) or (Features, Label)
- Discretize / Encode the target variable or perform one-hot encoding on the target or any other as and if required.
- Report the observations

Score: 1 Mark

```
# Here x is independent variable contains text input
# Here Y is dependent variable contains 1 or 0 by considering whether a person in depression or not
x = data['message to examine']
y = data['label (depression result)']
```

```
#One Hot Encoding of target variable
import tensorflow
import keras
from tensorflow.keras.utils import to_categorical
dep = to_categorical(data['label (depression result)'])
print(dep)
```

```
[[1. 0.]
 [1. 0.]
 [1. 0.]
 ...
 [0. 1.]
 [0. 1.]
 [0. 1.]]
```

4. Data Exploration using various plots

4.1 Scatter plot of each quantitative attribute with the target.

Score: 1 Mark

4.2 EDA using visuals

- Use (minimum) 2 plots (pair plot, heat map, correlation plot, regression plot...) to identify the optimal set of attributes that can be used for classification.
- Name them, explain why you think they can be helpful in the task and perform the plot as well. Unless proper justification for the choice of plots given, no credit will be awarded.

Score: 2 Marks

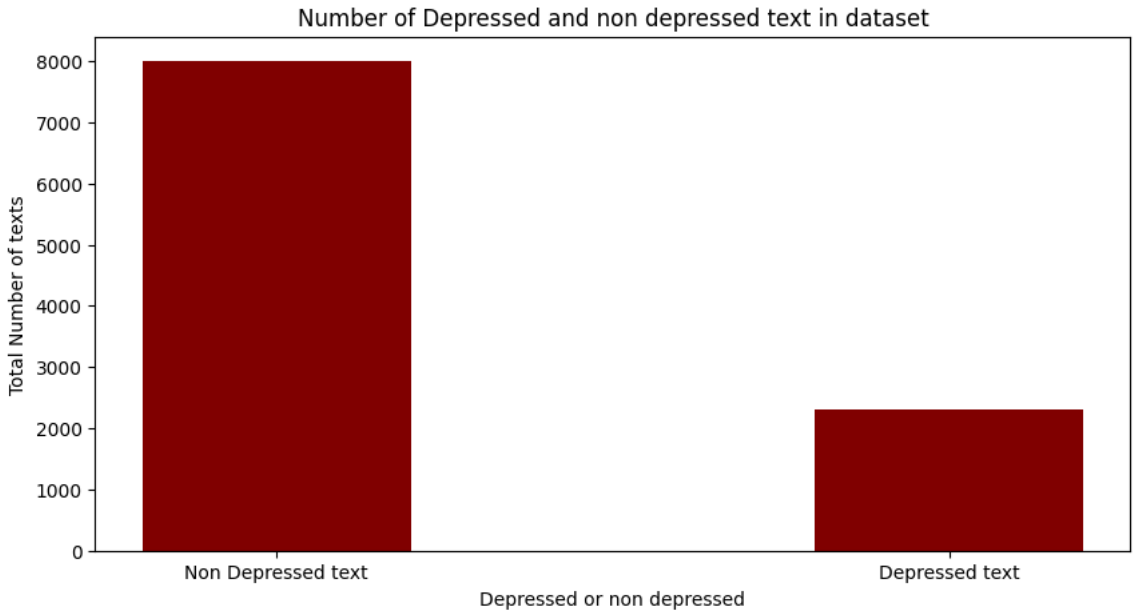
Since it's a text data these techniques cannot be applied

```
data['label (depression result)'].value_counts()
```

```
0    8000
1    2314
Name: label (depression result), dtype: int64
```

```
import matplotlib.pyplot as plt
import numpy as np
```

```
Non_depressed_text = ["Non Depressed text","Depressed text"]
depressed_text = [8000,2314]

fig = plt.figure(figsize = (10, 5))

# creating the bar plot
plt.bar(Non_depressed_text, depressed_text, color ='maroon',
        width = 0.4)

plt.xlabel("Depressed or non depressed")
plt.ylabel("Total Number of texts")
plt.title("Number of Depressed and non depressed text in dataset")
plt.show()
```



Here the data is clearly Imbalanced to balance the data the oversampled data is removed here.

```
def balance_df(df, target_col):
  majority_class_size = df[df[target_col] == 1].shape[0]
  minority_class_samples = df[df[target_col] == 0]
  balanced_df = df[df[target_col] == 0].head(majority_class_size)
  return balanced_df

balanced_data = balance_df(data, 'label (depression result)')

balanced_data.tail(5)
```

	Index	message to examine	label (depression result)	tokenized	lower	no_punc	stopwords_removed
2309	235935	my BIS connection is KapuT, no BBM, feels lonely	0	[my, BIS, connection, is, KapuT, ,, no, BBM, ,...	[my, bis, connection, is, kaput, ,, no, bbm, ,...	[my, bis, connection, is, kaput, no, bbm, feel...	[bis, connection, kaput, bbm, feels, lonely]
2310	236037	I love how non-chalant & blunt Tony Montan...	0	[I, love, how, non-chalant, &, amp, ;, blunt, ...	[i, love, how, non-chalant, &, amp, ;, blunt, ...	[i, love, how, non-chalant, amp, blunt, tony, ...	[love, non-chalant, amp, blunt, tony, montana,...
2311	236412	Glad to have gotten outta bed on my way back ...	0	[Glad, to, have, gotten, outta, bed, on, my, w...	[glad, to, have, gotten, outta, bed, on, my, w...	[glad, to, have, gotten, outta, bed, on, my, w...	[glad, gotten, outta, bed, way, back, home, re...
2312	236419	@FRin323 none of that!! I can't wait!!! so wh...	0	[@, FRin323, none, of, that, !, !, I, ca, n't,...	[@, frin323, none, of, that, !, !, i, ca, n't,...	[frin323, none, of, that, i, ca, n't, wait, so...	[frin323, none, ca, n't, wait, big, day]
2313	236420	@brendax <333333333 love you!	0	[@, brendax, &, lt, ;, 333333333, love, you, !]	[@, brendax, &, lt, ;, 333333333, love, you, !]	[brendax, lt, 333333333, love, you]	[brendax, lt, 333333333, love]

```
for i in range(len(data)):
 if data["label (depression result)"].iloc[i] == 1:
  balanced_data = balanced_data.append(data.iloc[i])
```

```
<ipython-input-20-3aaf6573539b>:3: FutureWarning: The frame.append method is deprecated and will be removed from pandas in a fut
  balanced_data = balanced_data.append(data.iloc[i])
<ipython-input-20-3aaf6573539b>:3: FutureWarning: The frame.append method is deprecated and will be removed from pandas in a fut
  balanced_data = balanced_data.append(data.iloc[i])
<ipython-input-20-3aaf6573539b>:3: FutureWarning: The frame.append method is deprecated and will be removed from pandas in a fut
  balanced_data = balanced_data.append(data.iloc[i])
<ipython-input-20-3aaf6573539b>:3: FutureWarning: The frame.append method is deprecated and will be removed from pandas in a fut
  balanced_data = balanced_data.append(data.iloc[i])
<ipython-input-20-3aaf6573539b>:3: FutureWarning: The frame.append method is deprecated and will be removed from pandas in a fut
  balanced_data = balanced_data.append(data.iloc[i])
<ipython-input-20-3aaf6573539b>:3: FutureWarning: The frame.append method is deprecated and will be removed from pandas in a fut
  balanced_data = balanced_data.append(data.iloc[i])
<ipython-input-20-3aaf6573539b>:3: FutureWarning: The frame.append method is deprecated and will be removed from pandas in a fut
  balanced_data = balanced_data.append(data.iloc[i])
<ipython-input-20-3aaf6573539b>:3: FutureWarning: The frame.append method is deprecated and will be removed from pandas in a fut
  balanced_data = balanced_data.append(data.iloc[i])
<ipython-input-20-3aaf6573539b>:3: FutureWarning: The frame.append method is deprecated and will be removed from pandas in a fut
  balanced_data = balanced_data.append(data.iloc[i])
<ipython-input-20-3aaf6573539b>:3: FutureWarning: The frame.append method is deprecated and will be removed from pandas in a fut
  balanced_data = balanced_data.append(data.iloc[i])
<ipython-input-20-3aaf6573539b>:3: FutureWarning: The frame.append method is deprecated and will be removed from pandas in a fut
  balanced_data = balanced_data.append(data.iloc[i])
<ipython-input-20-3aaf6573539b>:3: FutureWarning: The frame.append method is deprecated and will be removed from pandas in a fut
  balanced_data = balanced_data.append(data.iloc[i])
<ipython-input-20-3aaf6573539b>:3: FutureWarning: The frame.append method is deprecated and will be removed from pandas in a fut
  balanced_data = balanced_data.append(data.iloc[i])
<ipython-input-20-3aaf6573539b>:3: FutureWarning: The frame.append method is deprecated and will be removed from pandas in a fut
  balanced_data = balanced_data.append(data.iloc[i])
<ipython-input-20-3aaf6573539b>:3: FutureWarning: The frame.append method is deprecated and will be removed from pandas in a fut
  balanced_data = balanced_data.append(data.iloc[i])
<ipython-input-20-3aaf6573539b>:3: FutureWarning: The frame.append method is deprecated and will be removed from pandas in a fut
  balanced_data = balanced_data.append(data.iloc[i])
<ipython-input-20-3aaf6573539b>:3: FutureWarning: The frame.append method is deprecated and will be removed from pandas in a fut
  balanced_data = balanced_data.append(data.iloc[i])
<ipython-input-20-3aaf6573539b>:3: FutureWarning: The frame.append method is deprecated and will be removed from pandas in a fut
  balanced_data = balanced_data.append(data.iloc[i])
<ipython-input-20-3aaf6573539b>:3: FutureWarning: The frame.append method is deprecated and will be removed from pandas in a fut
  balanced_data = balanced_data.append(data.iloc[i])
<ipython-input-20-3aaf6573539b>:3: FutureWarning: The frame.append method is deprecated and will be removed from pandas in a fut
  balanced_data = balanced_data.append(data.iloc[i])
<ipython-input-20-3aaf6573539b>:3: FutureWarning: The frame.append method is deprecated and will be removed from pandas in a fut
  balanced_data = balanced_data.append(data.iloc[i])
<ipython-input-20-3aaf6573539b>:3: FutureWarning: The frame.append method is deprecated and will be removed from pandas in a fut
  balanced_data = balanced_data.append(data.iloc[i])
<ipython-input-20-3aaf6573539b>:3: FutureWarning: The frame.append method is deprecated and will be removed from pandas in a fut
  balanced_data = balanced_data.append(data.iloc[i])
<ipython-input-20-3aaf6573539b>:3: FutureWarning: The frame.append method is deprecated and will be removed from pandas in a fut
  balanced_data = balanced_data.append(data.iloc[i])
<ipython-input-20-3aaf6573539b>:3: FutureWarning: The frame.append method is deprecated and will be removed from pandas in a fut
  balanced_data = balanced_data.append(data.iloc[i])
<ipython-input-20-3aaf6573539b>:3: FutureWarning: The frame.append method is deprecated and will be removed from pandas in a fut
  balanced_data = balanced_data.append(data.iloc[i])
<ipython-input-20-3aaf6573539b>:3: FutureWarning: The frame.append method is deprecated and will be removed from pandas in a fut
  balanced_data = balanced_data.append(data.iloc[i])
```

Balanced Data with equal number of values in both attributes of dependent variable

```
balanced_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 4628 entries, 0 to 10313
Data columns (total 7 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   Index                     4628 non-null   int64
 1   message to examine        4628 non-null   object
 2   label (depression result) 4628 non-null   int64
 3   tokenized                 4628 non-null   object
 4   lower                     4628 non-null   object
 5   no_punc                   4628 non-null   object
 6   stopwords_removed         4628 non-null   object
dtypes: int64(2), object(5)
memory usage: 289.2+ KB
```

```
balanced_data['label (depression result)'].value_counts()
```

```
0    2314
1    2314
Name: label (depression result), dtype: int64
```

```
# Here x is independent variable contains text input
# Here Y is dependent variable contains 1 or 0 by considering whether a person in depression or not
x = balanced_data['message to examine']
y = balanced_data['label (depression result)']
```

```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x, y,
                                    test_size=0.2, random_state=42)
print ("Training set shapes:", x_train.shape, y_train.shape)
print ("Test set shapes:", x_test.shape, y_test.shape)
```

```
Training set shapes: (3702,) (3702,)
Test set shapes: (926,) (926,)
```

5. Data Wrangling

5.1 Univariate Filters

Numerical and Categorical Data

- Identify top 5 significant features by evaluating each feature independently with respect to the target variable by exploring

1. Mutual Information (Information Gain)
2. Gini index
3. Gain Ratio
4. Chi-Squared test
5. Fisher Score (From the above 5 you are required to use only any **two**)

For Text data

1. Stemming / Lemmatization.
2. Forming n-grams and storing them in the document vector.
3. TF-IDF (From the above 2 you are required to use only any **two**)

Score: 3 Marks

```
def get_wordnet_pos(tag):
    if tag.startswith('J'):
        return wordnet.ADJ
    elif tag.startswith('V'):
        return wordnet.VERB
    elif tag.startswith('N'):
        return wordnet.NOUN
    elif tag.startswith('R'):
        return wordnet.ADV
    else:
        return wordnet.NOUN
```

```
#Pos tagging
data['pos_tags'] = data['stopwords_removed'].apply(nltk.tag.pos_tag)
data['wordnet_pos'] = data['pos_tags'].apply(lambda x: [(word, get_wordnet_pos(pos_tag)) for (word, pos_tag) in x])
data.head()
```

	Index	message to examine	label (depression result)	tokenized	lower	no_punc	stopwords_removed
0	106	just had a real good moment. i missssssssss hi...	0	[just, had, a, real, good, moment, ., i, misss...	[just, had, a, real, good, moment, ., i, misss...	[just, had, a, real, good, moment, i, missssss...	[real, good, moment, missssssssss, much]
1	217	is reading manga http://plurk.com/p/mzp1e	0	[is, reading, manga, http, :, //plurk.com/p/mz...	[is, reading, manga, http, :, //plurk.com/p/mz...	[is, reading, manga, http, //plurk.com/p/mzp1e]	[reading, manga, http, //plurk.com/p/mzp1e]
2	220	@comeagainjen http://twitpic.com/2y2lx - http:...	0	[@, comeagainjen, http, :, //twitpic.com/2y2lx...	[@, comeagainjen, http, :, //twitpic.com/2y2lx...	[comeagainjen, http, //twitpic.com/2y2lx, http...	[comeagainjen, http, //twitpic.com/2y2lx, http...
3	288	@lapcat Need to send 'em to my accountant tomo...	0	[@, lapcat, Need, to, send, 'em, to, my, accou...	[@, lapcat, need, to, send, 'em, to, my, accou...	[lapcat, need, to, send, 'em, to, my, accounta...	[lapcat, need, send, 'em, accountant, tomorrow...
4	540	ADD ME ON MYSPACE!!! myspace.com/LookThunder	0	[ADD, ME, ON, MYSPACE, !, !, !, myspace.com/Lo...	[add, me, on, myspace, !, !, !, myspace.com/lo...	[add, me, on, myspace, myspace.com/lookthunder]	[add, myspace, myspace.com/lookthunder]

```
#Lemmatization
wnl = WordNetLemmatizer()
```

```
data['lemmatized'] = data['wordnet_pos'].apply(lambda x: [wnl.lemmatize(word, tag) for word, tag in x])
data.head()
```

kenized	lower	no_punc	stopwords_removed	pos_tags	wordnet_pos	le
l, a, real, nent, ., i, misss...	[just, had, a, real, good, moment, ., i, misss...	[just, had, a, real, good, moment, i, missssss...	[real, good, moment, missssssssss, much]	[(real, JJ), (good, JJ), (moment, NN), (missss...	[(real, a), (good, a), (moment, n), (missssssss...	[real, good, mom
, manga, http, :, n/p/mz...	[is, reading, manga, http, :, //plurk.com/p/mz...	[is, reading, manga, http, //plurk.com/p/mzp1e]	[reading, manga, http, //plurk.com/p/mzp1e]	[(reading, VBG), (manga, NN), (http, NN), (//p...	[(reading, v), (manga, n), (http, n), (//plurk...	[read, ma //plurk.com/
againjen, http, :, n/2y2lx...	[@, comeagainjen, http, :, //twitpic.com/2y2lx...	[comeagainjen, http, //twitpic.com/2y2lx, http...	[comeagainjen, http, //twitpic.com/2y2lx, http...	[(comeagainjen, NN), (http, NN), (//twitpic.co...	[(comeagainjen, n), (http, n), (//twitpic.com/...	[comeagai //twitpic.com/2y
at, Need, , 'em, to, accou...	[@, lapcat, need, to, send, 'em, to, my, accou...	[lapcat, need, to, send, 'em, to, my, accounta...	[lapcat, need, send, 'em, accountant, tomorrow...	[(lapcat, NNS), (need, VBP), (send, VBP), ('em...	[(lapcat, n), (need, v), (send, v), ('em, n), ...	[lapcat, need, s accountant, tc
ME, ON, CE, !, !, !, com/Lo...	[add, me, on, myspace, !, !, !, myspace.com/lo...	[add, me, on, myspace, myspace.com/lookthunder]	[add, myspace, myspace.com/lookthunder]	[(add, VB), (myspace, NN), (myspace.com/lookth...	[(add, v), (myspace, n), (myspace.com/lookthun...	[add, myspace.com/loo

```
from sklearn.feature_extraction.text import TfidfVectorizer
vectorizer = TfidfVectorizer()

vectorizer.fit(data['message to examine'])

tfidf = vectorizer.transform(data['message to examine'])

for i, row in data.iterrows():
  print(tfidf[i].toarray())
```

```
[[0. 0. 0. ... 0. 0. 0.]]
[[0. 0. 0. ... 0. 0. 0.]]
[[0. 0. 0. ... 0. 0. 0.]]
[[0. 0. 0. ... 0. 0. 0.]]
[[0. 0. 0. ... 0. 0. 0.]]
[[0. 0. 0. ... 0. 0. 0.]]
[[0. 0. 0. ... 0. 0. 0.]]
```

5.2 Report observations

Write your observations from the results of each method. Clearly justify your choice of the method.

Score 1 mark

Here I Initially performed Lemmatization, Since Lemmatization is better than Stemming, Stemming only chops off word ending without considering any context, But lemmatizattion keeps the context of word. It also conssiders the part of speech of a word for better understanding.

Then I used TF - IDF for vectorization words, Where TF refers to term frequency it only considers frequence of word in a sentence and IDF refers to Inverse Document Frequency considers the word appeared throught the document. It's one of the frequency based vectorization techniques where it does not keep the context of a sentence.

6. Implement Machine Learning Techniques

Use any 2 ML algorithms

A clear justification have to be given for why a certain algorithm was chosen to address your problem.

Score: 4 Marks (2 marks each for each algorithm)

6.1 ML technique 1 + Justification

distilBERT

DistilBERT is a distilled version of the BERT model. It is smaller and faster than BERT, while still retaining most of its performance. DistilBERT is trained using a technique called knowledge distillation, which involves training a smaller model to mimic the output of a larger model. This allows the smaller model to learn the important features of the data without having to store as much information.

Since in the refered paper distil bert works good with classification process I choose this one.

(i) It is 40% smaller than BERT, with 66M parameters compared to BERT's 110M parameters.

(ii) It is 60% faster than BERT, with a 100ms latency compared to BERT's 160ms latency.

```
!pip install sentence-transformers
```

```
Collecting sentence-transformers
  Downloading sentence-transformers-2.2.2.tar.gz (85 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 86.0/86.0 kB 1.8 MB/s eta 0:00:00
  Preparing metadata (setup.py) ... done
Collecting transformers<5.0.0,>=4.6.0 (from sentence-transformers)
  Downloading transformers-4.33.1-py3-none-any.whl (7.6 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 7.6/7.6 MB 32.3 MB/s eta 0:00:00
Requirement already satisfied: tqdm in /usr/local/lib/python3.10/dist-packages (from sentence-transformers) (4.66.1)
Requirement already satisfied: torch>=1.6.0 in /usr/local/lib/python3.10/dist-packages (from sentence-transformers) (2.0.1+cu118)
Requirement already satisfied: torchvision in /usr/local/lib/python3.10/dist-packages (from sentence-transformers) (0.15.2+cu118)
Requirement already satisfied: numpy in /usr/local/lib/python3.10/dist-packages (from sentence-transformers) (1.23.5)
Requirement already satisfied: scikit-learn in /usr/local/lib/python3.10/dist-packages (from sentence-transformers) (1.2.2)
Requirement already satisfied: scipy in /usr/local/lib/python3.10/dist-packages (from sentence-transformers) (1.10.1)
Requirement already satisfied: nltk in /usr/local/lib/python3.10/dist-packages (from sentence-transformers) (3.8.1)
Collecting sentencepiece (from sentence-transformers)
  Downloading sentencepiece-0.1.99-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (1.3 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1.3/1.3 MB 66.7 MB/s eta 0:00:00
Collecting huggingface-hub>=0.4.0 (from sentence-transformers)
  Downloading huggingface_hub-0.16.4-py3-none-any.whl (268 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 268.8/268.8 kB 31.0 MB/s eta 0:00:00
Requirement already satisfied: filelock in /usr/local/lib/python3.10/dist-packages (from huggingface-hub>=0.4.0->sentence-transforn
Requirement already satisfied: fsspec in /usr/local/lib/python3.10/dist-packages (from huggingface-hub>=0.4.0->sentence-transformer
Requirement already satisfied: requests in /usr/local/lib/python3.10/dist-packages (from huggingface-hub>=0.4.0->sentence-transforn
Requirement already satisfied: pyyaml>=5.1 in /usr/local/lib/python3.10/dist-packages (from huggingface-hub>=0.4.0->sentence-transf
Requirement already satisfied: typing-extensions>=3.7.4.3 in /usr/local/lib/python3.10/dist-packages (from huggingface-hub>=0.4.0->
Requirement already satisfied: packaging>=20.9 in /usr/local/lib/python3.10/dist-packages (from huggingface-hub>=0.4.0->sentence-tr
Requirement already satisfied: sympy in /usr/local/lib/python3.10/dist-packages (from torch>=1.6.0->sentence-transformers) (1.12)
Requirement already satisfied: networkx in /usr/local/lib/python3.10/dist-packages (from torch>=1.6.0->sentence-transformers) (3.1)
Requirement already satisfied: jinja2 in /usr/local/lib/python3.10/dist-packages (from torch>=1.6.0->sentence-transformers) (3.1.2)
Requirement already satisfied: triton==2.0.0 in /usr/local/lib/python3.10/dist-packages (from torch>=1.6.0->sentence-transformers)
```

```
Requirement already satisfied: cmake in /usr/local/lib/python3.10/dist-packages (from triton==2.0.0->torch>=1.6.0->sentence-transfo
Requirement already satisfied: lit in /usr/local/lib/python3.10/dist-packages (from triton==2.0.0->torch>=1.6.0->sentence-transform
Requirement already satisfied: regex!=2019.12.17 in /usr/local/lib/python3.10/dist-packages (from transformers<5.0.0,>=4.6.0->sente
Collecting tokenizers!=0.11.3,<0.14,>=0.11.1 (from transformers<5.0.0,>=4.6.0->sentence-transformers)
  Downloading tokenizers-0.13.3-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (7.8 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 7.8/7.8 MB 94.5 MB/s eta 0:00:00
Collecting safetensors>=0.3.1 (from transformers<5.0.0,>=4.6.0->sentence-transformers)
  Downloading safetensors-0.3.3-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (1.3 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1.3/1.3 MB 68.4 MB/s eta 0:00:00
Requirement already satisfied: click in /usr/local/lib/python3.10/dist-packages (from nltk->sentence-transformers) (8.1.7)
Requirement already satisfied: joblib in /usr/local/lib/python3.10/dist-packages (from nltk->sentence-transformers) (1.3.2)
Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/local/lib/python3.10/dist-packages (from scikit-learn->sentence-transfo
Requirement already satisfied: pillow!=8.3.*,>=5.3.0 in /usr/local/lib/python3.10/dist-packages (from torchvision->sentence-transfo
Requirement already satisfied: MarkupSafe>=2.0 in /usr/local/lib/python3.10/dist-packages (from jinja2->torch>=1.6.0->sentence-trar
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.10/dist-packages (from requests->huggingface-hub>
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-packages (from requests->huggingface-hub>=0.4.0->sent
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.10/dist-packages (from requests->huggingface-hub>=0.4.0
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.10/dist-packages (from requests->huggingface-hub>=0.4.0
Requirement already satisfied: mpmath>=0.19 in /usr/local/lib/python3.10/dist-packages (from sympy->torch>=1.6.0->sentence-transfor
Building wheels for collected packages: sentence-transformers
  Building wheel for sentence-transformers (setup.py) ... done
  Created wheel for sentence-transformers: filename=sentence_transformers-2.2.2-py3-none-any.whl size=125923 sha256=1f4d056f2d29a0
  Stored in directory: /root/.cache/pip/wheels/62/f2/10/1e606fd5f02395388f74e7462910fe851042f97238cbbd902f
Successfully built sentence-transformers
Installing collected packages: tokenizers, sentencepiece, safetensors, huggingface-hub, transformers, sentence-transformers
Successfully installed huggingface-hub-0.16.4 safetensors-0.3.3 sentence-transformers-2.2.2 sentencepiece-0.1.99 tokenizers-0.13.3
```

```
from sentence_transformers import SentenceTransformer
distilbert_model = SentenceTransformer('distilbert-base-nli-mean-tokens')
```

```
tweets = balanced_data.values[:,1]
labels = balanced_data.values[:,2].astype(float)
```

```
embeddings1 = distilbert_model.encode(tweets, show_progress_bar=True)
print (embeddings1.shape)
```

Batches: 100% 145/145 [00:06<00:00, 51.63it/s]

```
(4628, 768)
```

```
from sklearn.model_selection import train_test_split
X_train1, X_test1, y_train1, y_test1 = train_test_split(embeddings1, labels,
                                                    test_size=0.2, random_state=42)
print ("Training set shapes:", X_train1.shape, y_train1.shape)
print ("Test set shapes:", X_test1.shape, y_test1.shape)
```

```
Training set shapes: (3702, 768) (3702,)
Test set shapes: (926, 768) (926,)
```

```
from tensorflow.keras import Sequential, layers

classifier = Sequential()
classifier.add (layers.Dense(256, activation='relu', input_shape=(768,)))
classifier.add (layers.Dense(256, activation='relu', input_shape=(768,)))
classifier.add (layers.Dense(1, activation='sigmoid'))
classifier.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])

hist1 = classifier.fit (X_train1, y_train1, epochs=50, batch_size=16,
                       validation_data=(X_test1, y_test1))
```

```
Epoch 1/50
232/232 [==============================] - 2s 5ms/step - loss: 0.1224 - accuracy: 0.9525 - val_loss: 0.0923 - val_accuracy: 0.96
Epoch 2/50
232/232 [==============================] - 1s 4ms/step - loss: 0.0473 - accuracy: 0.9846 - val_loss: 0.0811 - val_accuracy: 0.96
Epoch 3/50
232/232 [==============================] - 1s 4ms/step - loss: 0.0358 - accuracy: 0.9878 - val_loss: 0.0663 - val_accuracy: 0.97
Epoch 4/50
232/232 [==============================] - 1s 4ms/step - loss: 0.0308 - accuracy: 0.9903 - val_loss: 0.0664 - val_accuracy: 0.98
Epoch 5/50
232/232 [==============================] - 1s 4ms/step - loss: 0.0139 - accuracy: 0.9954 - val_loss: 0.0867 - val_accuracy: 0.97
Epoch 6/50
232/232 [==============================] - 1s 5ms/step - loss: 0.0221 - accuracy: 0.9916 - val_loss: 0.0702 - val_accuracy: 0.97
Epoch 7/50
232/232 [==============================] - 1s 6ms/step - loss: 0.0175 - accuracy: 0.9943 - val_loss: 0.0950 - val_accuracy: 0.98
Epoch 8/50
232/232 [==============================] - 1s 6ms/step - loss: 0.0134 - accuracy: 0.9946 - val_loss: 0.1641 - val_accuracy: 0.95
Epoch 9/50
232/232 [==============================] - 1s 6ms/step - loss: 0.0148 - accuracy: 0.9968 - val_loss: 0.0861 - val_accuracy: 0.97
Epoch 10/50
232/232 [==============================] - 1s 4ms/step - loss: 0.0021 - accuracy: 0.9997 - val_loss: 0.0880 - val_accuracy: 0.98
Epoch 11/50
232/232 [==============================] - 1s 4ms/step - loss: 1.9481e-04 - accuracy: 1.0000 - val_loss: 0.0915 - val_accuracy:
```

```
Epoch 12/50
232/232 [==============================] - 1s 4ms/step - loss: 6.8436e-05 - accuracy: 1.0000 - val_loss: 0.0947 - val_accuracy:
Epoch 13/50
232/232 [==============================] - 1s 4ms/step - loss: 4.7446e-05 - accuracy: 1.0000 - val_loss: 0.0976 - val_accuracy:
Epoch 14/50
232/232 [==============================] - 1s 4ms/step - loss: 3.5103e-05 - accuracy: 1.0000 - val_loss: 0.1003 - val_accuracy:
Epoch 15/50
232/232 [==============================] - 1s 4ms/step - loss: 2.6932e-05 - accuracy: 1.0000 - val_loss: 0.1027 - val_accuracy:
Epoch 16/50
232/232 [==============================] - 1s 4ms/step - loss: 2.1382e-05 - accuracy: 1.0000 - val_loss: 0.1049 - val_accuracy:
Epoch 17/50
232/232 [==============================] - 1s 4ms/step - loss: 1.7252e-05 - accuracy: 1.0000 - val_loss: 0.1069 - val_accuracy:
Epoch 18/50
232/232 [==============================] - 1s 4ms/step - loss: 1.4133e-05 - accuracy: 1.0000 - val_loss: 0.1087 - val_accuracy:
Epoch 19/50
232/232 [==============================] - 1s 4ms/step - loss: 1.1724e-05 - accuracy: 1.0000 - val_loss: 0.1107 - val_accuracy:
Epoch 20/50
232/232 [==============================] - 1s 6ms/step - loss: 9.7688e-06 - accuracy: 1.0000 - val_loss: 0.1124 - val_accuracy:
Epoch 21/50
232/232 [==============================] - 2s 7ms/step - loss: 8.2841e-06 - accuracy: 1.0000 - val_loss: 0.1141 - val_accuracy:
Epoch 22/50
232/232 [==============================] - 2s 7ms/step - loss: 7.0300e-06 - accuracy: 1.0000 - val_loss: 0.1157 - val_accuracy:
Epoch 23/50
232/232 [==============================] - 1s 4ms/step - loss: 5.9636e-06 - accuracy: 1.0000 - val_loss: 0.1171 - val_accuracy:
Epoch 24/50
232/232 [==============================] - 1s 4ms/step - loss: 5.0800e-06 - accuracy: 1.0000 - val_loss: 0.1187 - val_accuracy:
Epoch 25/50
232/232 [==============================] - 1s 4ms/step - loss: 4.3296e-06 - accuracy: 1.0000 - val_loss: 0.1201 - val_accuracy:
Epoch 26/50
232/232 [==============================] - 1s 4ms/step - loss: 3.7533e-06 - accuracy: 1.0000 - val_loss: 0.1216 - val_accuracy:
Epoch 27/50
232/232 [==============================] - 1s 4ms/step - loss: 3.1984e-06 - accuracy: 1.0000 - val_loss: 0.1229 - val_accuracy:
Epoch 28/50
232/232 [==============================] - 1s 4ms/step - loss: 2.8129e-06 - accuracy: 1.0000 - val_loss: 0.1244 - val_accuracy:
```

6.2 ML technique 2 + Justification

BERT

BERT is trained on a massive dataset of text and code, which allows it to learn the contextual meaning of words. This makes BERT very effective for a variety of natural language processing tasks, such as text classification, question answering, and natural language inference.

Here are some of the key features of BERT:

(i) It is a bidirectional model, which means that it can learn the meaning of words both before and after they appear in a sentence.

(ii) It is trained on a massive dataset of text and code, which allows it to learn the contextual meaning of words.

(iii) It is a transformer-based model, which means that it uses attention mechanisms to learn long-range dependencies in text.

(iv) It has been shown to be effective for a variety of natural language processing tasks, such as text classification, question answering, and natural language inference.

It is also one of the main algorithm which is been used in the reference paper.

```
from sentence_transformers import SentenceTransformer
bert_model = SentenceTransformer('bert-base-uncased')
```

```
WARNING:sentence_transformers.SentenceTransformer:No sentence-transformers model found with name /root/.cache/torch/sentence_trans1
```

```
embeddings2 = bert_model.encode(tweets, show_progress_bar=True)
print (embeddings2.shape)
```

```
Batches: 100%                    145/145 [00:13<00:00, 33.91it/s]
(4628, 768)
```

```
from sklearn.model_selection import train_test_split
X_train2, X_test2, y_train2, y_test2 = train_test_split(embeddings1, labels,
                                                        test_size=0.2, random_state=42)
print ("Training set shapes:", X_train2.shape, y_train2.shape)
print ("Test set shapes:", X_test2.shape, y_test2.shape)
```

```
Training set shapes: (3702, 768) (3702,)
Test set shapes: (926, 768) (926,)
```

```
from tensorflow.keras import Sequential, layers
```

```
classifier2 = Sequential()
classifier2.add (layers.Dense(128, activation='relu', input_shape=(768,)))
classifier2.add (layers.Dense(128, activation='relu', input_shape=(768,)))
classifier2.add (layers.Dense(1, activation='sigmoid'))
classifier2.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])

hist2 = classifier2.fit (X_train2, y_train2, epochs=50, batch_size=16,
                      validation_data=(X_test2, y_test2))
```

```
Epoch 1/50
232/232 [==============================] - 5s 7ms/step - loss: 0.1181 - accuracy: 0.9557 - val_loss: 0.0878 - val_accuracy: 0.97
Epoch 2/50
232/232 [==============================] - 1s 4ms/step - loss: 0.0566 - accuracy: 0.9781 - val_loss: 0.0843 - val_accuracy: 0.97
Epoch 3/50
232/232 [==============================] - 1s 4ms/step - loss: 0.0394 - accuracy: 0.9876 - val_loss: 0.0781 - val_accuracy: 0.97
Epoch 4/50
232/232 [==============================] - 1s 4ms/step - loss: 0.0347 - accuracy: 0.9868 - val_loss: 0.0712 - val_accuracy: 0.97
Epoch 5/50
232/232 [==============================] - 1s 4ms/step - loss: 0.0220 - accuracy: 0.9927 - val_loss: 0.0774 - val_accuracy: 0.97
Epoch 6/50
232/232 [==============================] - 1s 5ms/step - loss: 0.0178 - accuracy: 0.9927 - val_loss: 0.0802 - val_accuracy: 0.98
Epoch 7/50
232/232 [==============================] - 1s 6ms/step - loss: 0.0114 - accuracy: 0.9954 - val_loss: 0.0857 - val_accuracy: 0.97
Epoch 8/50
232/232 [==============================] - 1s 6ms/step - loss: 0.0140 - accuracy: 0.9941 - val_loss: 0.0824 - val_accuracy: 0.97
Epoch 9/50
232/232 [==============================] - 1s 6ms/step - loss: 0.0076 - accuracy: 0.9970 - val_loss: 0.0788 - val_accuracy: 0.97
Epoch 10/50
232/232 [==============================] - 1s 4ms/step - loss: 0.0011 - accuracy: 1.0000 - val_loss: 0.1057 - val_accuracy: 0.98
Epoch 11/50
232/232 [==============================] - 2s 8ms/step - loss: 0.0012 - accuracy: 0.9997 - val_loss: 0.1091 - val_accuracy: 0.98
Epoch 12/50
232/232 [==============================] - 1s 5ms/step - loss: 1.5394e-04 - accuracy: 1.0000 - val_loss: 0.1117 - val_accuracy:
Epoch 13/50
232/232 [==============================] - 2s 8ms/step - loss: 6.3683e-05 - accuracy: 1.0000 - val_loss: 0.1147 - val_accuracy:
Epoch 14/50
232/232 [==============================] - 2s 7ms/step - loss: 4.6031e-05 - accuracy: 1.0000 - val_loss: 0.1158 - val_accuracy:
Epoch 15/50
232/232 [==============================] - 2s 7ms/step - loss: 3.3810e-05 - accuracy: 1.0000 - val_loss: 0.1184 - val_accuracy:
Epoch 16/50
232/232 [==============================] - 2s 7ms/step - loss: 2.5108e-05 - accuracy: 1.0000 - val_loss: 0.1213 - val_accuracy:
Epoch 17/50
232/232 [==============================] - 3s 11ms/step - loss: 1.9155e-05 - accuracy: 1.0000 - val_loss: 0.1238 - val_accuracy:
Epoch 18/50
232/232 [==============================] - 3s 15ms/step - loss: 1.4306e-05 - accuracy: 1.0000 - val_loss: 0.1267 - val_accuracy:
Epoch 19/50
232/232 [==============================] - 3s 12ms/step - loss: 1.0878e-05 - accuracy: 1.0000 - val_loss: 0.1295 - val_accuracy:
Epoch 20/50
232/232 [==============================] - 2s 8ms/step - loss: 8.4372e-06 - accuracy: 1.0000 - val_loss: 0.1324 - val_accuracy:
Epoch 21/50
232/232 [==============================] - 2s 10ms/step - loss: 6.6928e-06 - accuracy: 1.0000 - val_loss: 0.1348 - val_accuracy:
Epoch 22/50
232/232 [==============================] - 2s 8ms/step - loss: 5.2692e-06 - accuracy: 1.0000 - val_loss: 0.1378 - val_accuracy:
Epoch 23/50
232/232 [==============================] - 2s 10ms/step - loss: 4.1826e-06 - accuracy: 1.0000 - val_loss: 0.1400 - val_accuracy:
Epoch 24/50
232/232 [==============================] - 2s 9ms/step - loss: 3.3731e-06 - accuracy: 1.0000 - val_loss: 0.1426 - val_accuracy:
Epoch 25/50
232/232 [==============================] - 2s 9ms/step - loss: 2.7439e-06 - accuracy: 1.0000 - val_loss: 0.1451 - val_accuracy:
Epoch 26/50
232/232 [==============================] - 2s 8ms/step - loss: 2.2348e-06 - accuracy: 1.0000 - val_loss: 0.1480 - val_accuracy:
Epoch 27/50
232/232 [==============================] - 2s 8ms/step - loss: 1.8422e-06 - accuracy: 1.0000 - val_loss: 0.1504 - val_accuracy:
Epoch 28/50
232/232 [==============================] - 2s 7ms/step - loss: 1.5374e-06 - accuracy: 1.0000 - val_loss: 0.1528 - val_accuracy:
Epoch 29/50
```

7. Conclusion

Compare the performance of the ML techniques used.

Derive values for preformance study metrics like accuracy, precision, recall, F1 Score, AUC-ROC etc to compare the ML algos and plot them. A proper comparision based on different metrics should be done and not just accuracy alone, only then the comparision becomes authentic. You may use Confusion matrix, classification report, Word cloud etc as per the requirement of your application/problem.

Score 1 Mark

```
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score
from sklearn.metrics import recall_score
from sklearn.metrics import precision_score
from sklearn.metrics import classification_report
from sklearn.metrics import roc_auc_score
```

Distil - BERT

```
def predict_sentence1(sentence, classifier):
    predict=sentence
    predict_embedding = distilbert_model.encode(predict)
    # print (predict_embedding.shape)
    predict_embedding = predict_embedding.reshape(1,768)
    # print (predict_embedding.shape)
    result = classifier.predict(predict_embedding)
    if result[0][0] <= 0.5:
        return 0
    else:
        return 1
```

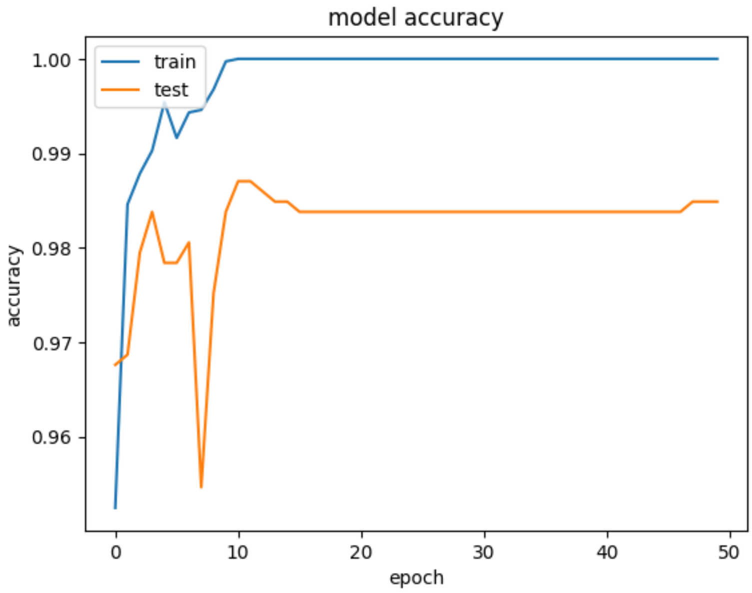
```
predict1 = []
# print(x_test.iloc[i])
for i in range(len(x_test)):
  predict1.append(predict_sentence1(x_test.iloc[i],classifier))
```

```
print("Accuracy Score:", accuracy_score(y_test, predict1))
print("F1 Score: ",classification_report(y_test, predict1))
print("ROC Curve: ",roc_auc_score(y_test, predict1))
```

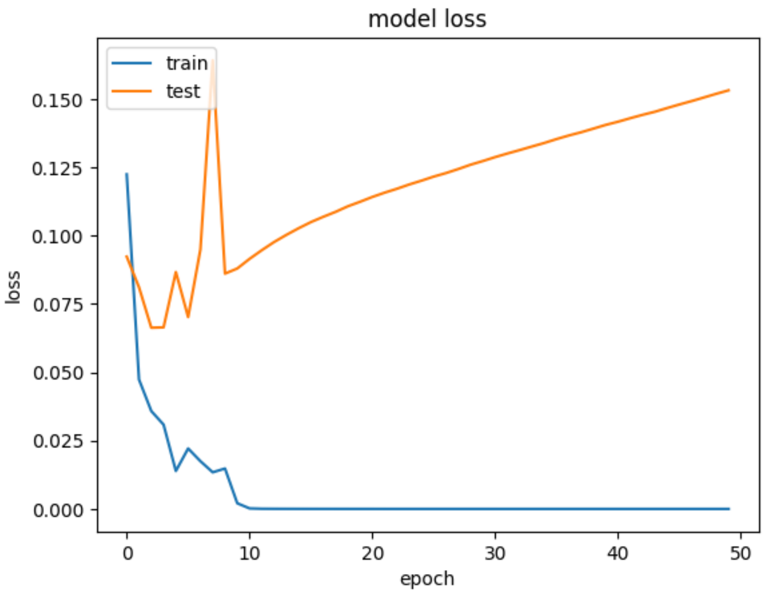
```
Accuracy Score: 0.9827213822894169
Recall Score: [0.98526316 0.98004435]
precision_score:  [0.98113208 0.9844098 ]
F1 Score:                precision    recall  f1-score   support

           0       0.98      0.99      0.98       475
           1       0.98      0.98      0.98       451

    accuracy                           0.98       926
   macro avg       0.98      0.98      0.98       926
weighted avg       0.98      0.98      0.98       926

ROC Curve:  0.9826537518963706
```

```
import matplotlib.pyplot as plt
from sklearn import metrics

depression = np.random.binomial(1,.9,size = 90)
Non_depression = np.random.binomial(1,.9,size = 90)
confusion_matrix = metrics.confusion_matrix(y_test,predict1)

cm_display = metrics.ConfusionMatrixDisplay(confusion_matrix = confusion_matrix, display_labels = ["Depression","Non_depression"])

cm_display.plot()
plt.show()
```

```
#Summarize history for accuracy
plt.plot(hist1.history['accuracy'])
plt.plot(hist1.history['val_accuracy'])
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.show()
```




```
# summarize history for loss
plt.plot(hist1.history['loss'])
plt.plot(hist1.history['val_loss'])
plt.title('model loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.show()
```



Conclusion

DistilBert Works excellent with this application. It is almost have perfect training accuracy of 1 and validation accuracy of 98.49%. It is having a good score of 0.98% for all precison, recall and f1 score. The validation loss tends to increase after 10 epoch and accuracy does not also increases so it is optimal to stop after 10 epochs.

BERT

```
def predict_sentence2(sentence, classifier):
    predict=sentence
    predict_embedding = bert_model.encode(predict)
    # print (predict_embedding.shape)
    predict_embedding = predict_embedding.reshape(1,768)
    # print (predict_embedding.shape)
    result = classifier.predict(predict_embedding)
    if result[0][0] <= 0.5:
        return 0
    else:
        return 1


predict2 = []
for i in range(len(x_test)):
  predict2.append(predict_sentence2(x_test.iloc[i],classifier2))
```

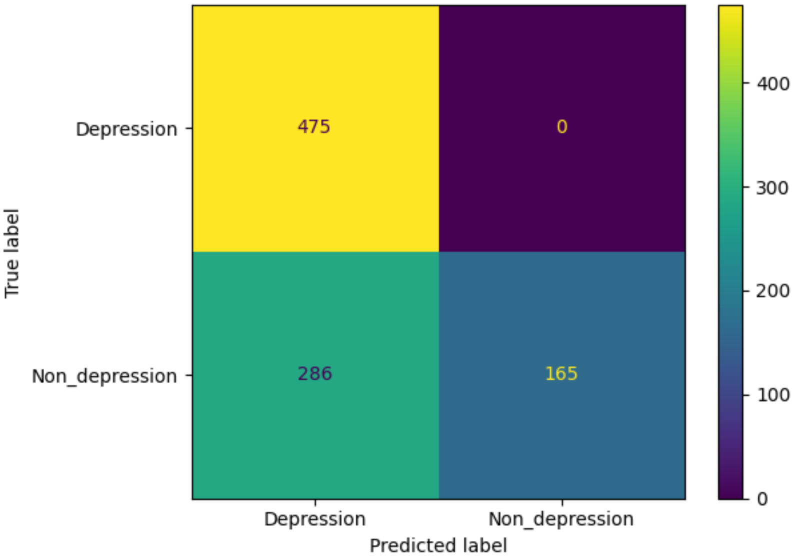
```
1/1 [==============================] - 0s 120ms/step
1/1 [==============================] - 0s 117ms/step
1/1 [==============================] - 0s 162ms/step
1/1 [==============================] - 0s 178ms/step
1/1 [==============================] - 0s 54ms/step
1/1 [==============================] - 0s 69ms/step
1/1 [==============================] - 0s 72ms/step
1/1 [==============================] - 0s 49ms/step
1/1 [==============================] - 0s 57ms/step
1/1 [==============================] - 0s 36ms/step
1/1 [==============================] - 0s 26ms/step
1/1 [==============================] - 0s 27ms/step
1/1 [==============================] - 0s 32ms/step
1/1 [==============================] - 0s 28ms/step
1/1 [==============================] - 0s 27ms/step
1/1 [==============================] - 0s 32ms/step
1/1 [==============================] - 0s 30ms/step
1/1 [==============================] - 0s 31ms/step
1/1 [==============================] - 0s 50ms/step
1/1 [==============================] - 0s 29ms/step
1/1 [==============================] - 0s 28ms/step
1/1 [==============================] - 0s 36ms/step
1/1 [==============================] - 0s 30ms/step
1/1 [==============================] - 0s 32ms/step
1/1 [==============================] - 0s 33ms/step
1/1 [==============================] - 0s 35ms/step
1/1 [==============================] - 0s 34ms/step
1/1 [==============================] - 0s 29ms/step
1/1 [==============================] - 0s 32ms/step
1/1 [==============================] - 0s 32ms/step
1/1 [==============================] - 0s 32ms/step
1/1 [==============================] - 0s 33ms/step
1/1 [==============================] - 0s 32ms/step
1/1 [==============================] - 0s 28ms/step
1/1 [==============================] - 0s 30ms/step
1/1 [==============================] - 0s 40ms/step
1/1 [==============================] - 0s 26ms/step
1/1 [==============================] - 0s 43ms/step
1/1 [==============================] - 0s 27ms/step
1/1 [==============================] - 0s 40ms/step
1/1 [==============================] - 0s 29ms/step
1/1 [==============================] - 0s 21ms/step
1/1 [==============================] - 0s 19ms/step
1/1 [==============================] - 0s 25ms/step
1/1 [==============================] - 0s 18ms/step
1/1 [==============================] - 0s 18ms/step
1/1 [==============================] - 0s 18ms/step
1/1 [==============================] - 0s 21ms/step
1/1 [==============================] - 0s 20ms/step
1/1 [==============================] - 0s 17ms/step
1/1 [==============================] - 0s 17ms/step
1/1 [==============================] - 0s 17ms/step
1/1 [==============================] - 0s 30ms/step
1/1 [==============================] - 0s 26ms/step
1/1 [==============================] - 0s 33ms/step
1/1 [==============================] - 0s 28ms/step
1/1 [==============================] - 0s 32ms/step
1/1 [==============================] - 0s 36ms/step
```

```
print("Accuracy Score:", accuracy_score(y_test, predict2))
print("Recall Score:", recall_score(y_test, predict2, average=None))
print("precision_score: ",precision_score(y_test, predict2, average=None))
print("F1 Score: ",classification_report(y_test, predict2))
print("ROC Curve: ",roc_auc_score(y_test, predict2))
```

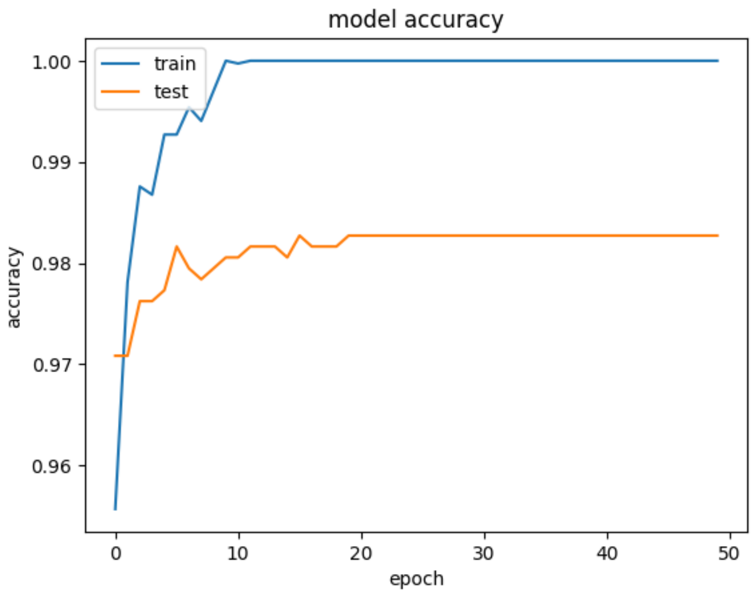
```
Accuracy Score: 0.6911447084233261
Recall Score: [1.         0.36585366]
precision_score:  [0.62417871 1.        ]
F1 Score:                precision    recall  f1-score   support
```

```
               0       0.62      1.00      0.77       475
               1       1.00      0.37      0.54       451

        accuracy                           0.69       926
       macro avg       0.81      0.68      0.65       926
    weighted avg       0.81      0.69      0.66       926

    ROC Curve:  0.6829268292682926
```

```
depression = np.random.binomial(1,.9,size = 90)
Non_depression = np.random.binomial(1,.9,size = 90)
confusion_matrix = metrics.confusion_matrix(y_test,predict2)

cm_display = metrics.ConfusionMatrixDisplay(confusion_matrix = confusion_matrix, display_labels = ["Depression","Non_depression"])

cm_display.plot()
plt.show()
```

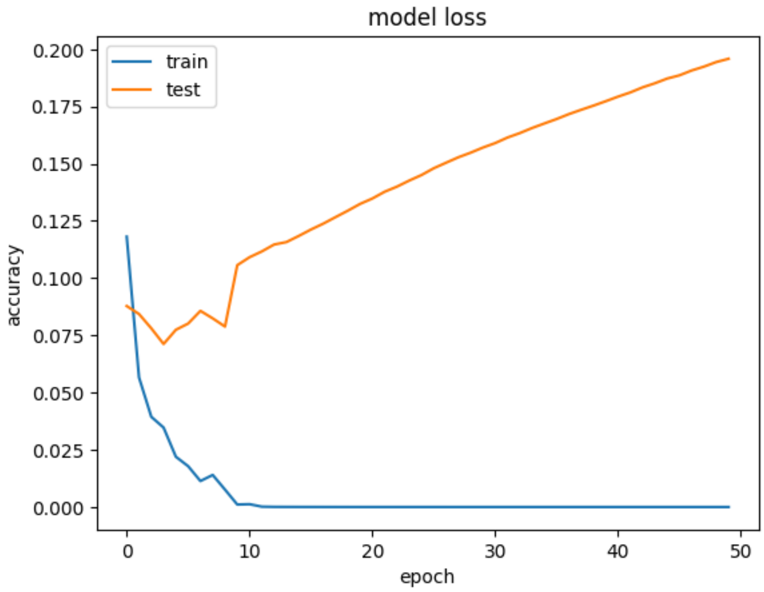



```
#Summarize history for accuracy
plt.plot(hist2.history['accuracy'])
plt.plot(hist2.history['val_accuracy'])
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.show()
```




```
#Summarize history for accuracy
plt.plot(hist2.history['loss'])
plt.plot(hist2.history['val_loss'])
plt.title('model loss')
plt.ylabel('accuracy')
plt.xlabel('epoch')
```

```
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.show()
```



Conclusion

Bert Works comparatively bad with this application. It is almost have perfect training accuracy of 1 and validation accuraccy of 98.27%. It is having a average score of 0.69% for all precison, recall and f1 score. The validation loss tends to increase after 10 epoch and accuracy does not also increases so it is optimal to stop after 10 epochs.

Since BERT performs works well with big dataset. It is optimal to use it for big dataset

8. Solution

What is the solution that is proposed to solve the business problem discussed in Section 1. Also share your learnings while working through solving the problem in terms of challenges, observations, decisions made etc.

Score 2 Marks

Depresssion is one of the most common mental health disease faced by our generation, detecting this early based on social media platform activity may benefit the humanity. Since many suicides are happening because of this it is become a very sensive case to be dealt with.

Now with the help of NLP techniques and abundant resource of data we can analyze the pattern of depressed people's social media activities and based on that we can save a life.

Challenges

One of the main challenges in handling twitter dataset is to identify the modern acronyms used in the tweets, but it can be handled better with transformer architecture.

Observations

The dataset is initially imbalanced it is then balnced, for bias free model training, No duplicates or any missing values are present in this data, The data is also consistent.

Decisions Made Distilbert Works best with this one since it is designed for to handle samller datasets. Bert performs comparatively worse than distilbert. So it is optimal to use distilbert in this use case scenario. It is also optimal to stop the model training with 10 epochs since validation loss increasing after that.