Regional Language Toxic comment classification

Technical Report

Yashkumar Parikh
parikhy@lakeheadu.ca
Student ID No.: 1138765

I. Abstract

Social media sites are gaining popularity day by day. They are best for communication, business, entertainment, and many other things. After more than a decade, social media have become very influential. On the flip side, fake news, hate speech, and online trolls are the biggest concerns because of social media. So, a solution to curb this issue is needed, especially in regional languages. Many social media platforms support regional languages. This paper will provide a machine learning-based solution to this problem. The focus of this paper is to classify comments written in regional languages. Firstly, a dataset has been created in Gujarati, Hindi, English, Marathi, and Punjabi languages. After that, different machine learning and deep learning models are applied to the multilingual dataset. At last, a comparison of all model performances was made.

II. Steps to Execute Project

1. Unzip the project folder
2. Upload the project folder (Data and Code files) to google drive
3. Open ipynb file and change ‘dir_path’ variable to your local path
4. Execute the code cells

III. Dataset

I have gathered data from various social media platforms in Indic languages like Gujarati, Hindi, Marathi, Punjabi, and English. The shape of the data is (14995, 2). It has two columns comment_text and toxic. The comment_text column contains actual text, and the toxic column contains '1' for toxic and '0' for non-toxic comments. The pre-processing step removes URLs, hashtags, mentions, punctuations, and extra white spaces from the comments. Plus, removed rows with empty or null values.

IV. Code Editor

Google Colab

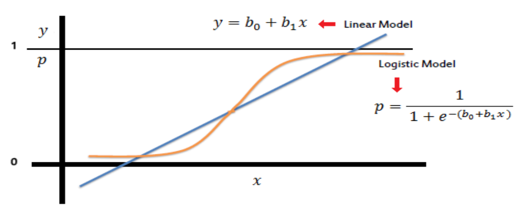
Colaboratory, sometimes called Colab, is a Google Research product. Colab is particularly well suited to machine learning, data analysis, and education. It enables anyone to create and execute arbitrary Python code through the browser. Technically speaking, Colab is a hosted Jupyter notebook service that offers free access to computer resources, including GPUs, and requires no setup [2].

V. Code

In code section I have used five different machine learning algorithms.

1. Logistic Regression

Logistic Regression is a supervised learning algorithm. It can model the probability of a specific class or category. It is applied when the outcome is binary, and the data may be linearly separated. That means problems involving binary classification are solved using Logistic Regression.



Referances

1. https://www.saedsayad.com/logistic_regression.htm

```
[ ] import pandas as pd
    import numpy as np
    import matplotlib.pyplot as plt
    import seaborn as sns
    import joblib
```

```
[ ] from google.colab import drive
    drive.mount('/content/gdrive', force_remount=True)

    Mounted at /content/gdrive
```

```
[ ] dir_path = "/content/gdrive/MyDrive/Colab Notebooks/Final_Project_Yashkumar_1138765/"
```

```
[ ] # Loading dataset
    data1=pd.read_excel(dir_path + "Data/Eng_Dataset.xlsx")
    data2=pd.read_excel(dir_path + "Data/Guj_Dataset.xlsx")
    data3=pd.read_excel(dir_path + "Data/Hin_Dataset.xlsx")
    data4=pd.read_excel(dir_path + "Data/Mar_Dataset.xlsx")
    data5=pd.read_excel(dir_path + "Data/Pun_Dataset.xlsx")
```

```
[ ] # Concatanating the files data
    data = pd.concat([
        data1[['comment_text', 'toxic']],
        data2[['comment_text', 'toxic']],
        data3[['comment_text', 'toxic']],
        data4[['comment_text', 'toxic']],
        data5[['comment_text', 'toxic']]
    ])
```

```
[ ] data.size # (14995 rows * 2 columns)

    29990
```

```
[ ] data.shape

    (14995, 2)
```

```
[ ] data['toxic'].value_counts()

    1.0    7500
    0.0    7495
    Name: toxic, dtype: int64
```

```
[ ] data = data.astype({"toxic": int, "comment_text": str})
```

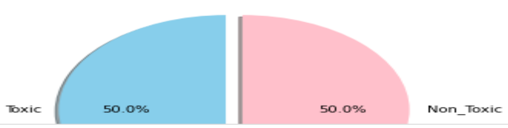
```
[ ] data.dtypes

    comment_text    object
    toxic            int64
    dtype: object
```

```
[ ] data.head()
```

	comment_text	toxic
0	Explanation Why the edits made under my userna...	0
1	D aww He matches this background colour I m se...	0
2	Hey man I m really not trying to edit war It s...	0
3	More I can t make any real suggestions on imp...	0
4	You sir are my hero Any chance you remember wh...	0

```
[ ] data.isnull().sum()

comment_text    0
toxic           0
dtype: int64
```

```
[ ] # Plotting the data distibution
fig=plt.figure(figsize=(5,5))
colors=["skyblue",'pink']
pos=data[data['toxic']==1]
neg=data[data['toxic']==0]
ck=[pos['toxic'].count(),neg['toxic'].count()]
legpie=plt.pie(ck,labels=["Toxic","Non_Toxic"],
                 autopct ='%1.1f%%',
                 shadow = True,
                 colors = colors,
                 startangle = 90,
                 explode=(0, 0.1))
```




```
# add new column to calculate the length of each comment
data['length'] = data['comment_text'].str.len()

# compute the summary statistics
data[["length"]].describe().T
```

	count	mean	std	min	25%	50%	75%	max
length	14995.0	346.164922	640.867558	2.0	72.0	158.0	356.0	15684.0

```
[ ] from sklearn.metrics import confusion_matrix, classification_report, accuracy_score
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegressionCV
```

```
[ ] def tokenizer(text):
        return text.split()
```

```
[ ] # tokenizing data
tfidf=TfidfVectorizer(strip_accents=None,lowercase=False,preprocessor=None,tokenizer=tokenizer,use_idf=True,norm='l2',smooth_idf=True)
y=data.toxic.values
x=tfidf.fit_transform(data.comment_text)
```

```
[ ] print(x[0])

  (0, 436)	0.13629397545473285
  (0, 487)	0.1516648558771504
  (0, 358)	0.14644587411427934
  (0, 643)	0.14397108902004568
  (0, 15161)	0.10990839624576292
  (0, 16929)	0.18820233374566683
  (0, 14358)	0.10178687548523987
  (0, 17590)	0.12276810984321544
  (0, 15502)	0.09225787054040967
  (0, 18390)	0.09929527054910753
  (0, 12499)	0.09429391208470374
  (0, 18481)	0.15003262861094374
  (0, 16762)	0.12983646313734015
  (0, 11339)	0.09975649758621642
```

```
[ ] # split the data in train and test set
    X_train,X_test,y_train,y_test=train_test_split(x,y,random_state=1,test_size=0.1,shuffle=True)
```

```
[ ] # training model
    clf=LogisticRegressionCV(cv=6,scoring='accuracy',random_state=0,n_jobs=-1,verbose=0,max_iter=500).fit(X_train,y_train)
    y_pred = clf.predict(X_test)

    # Model Train and Test Accuracy
    print("Train Accuracy: %.2f" % clf.score(X_train, y_train))
    print("Test Accuracy: %.2f" % accuracy_score(y_test, y_pred))
```

```
Train Accuracy: 1.00
Test Accuracy: 0.88
```

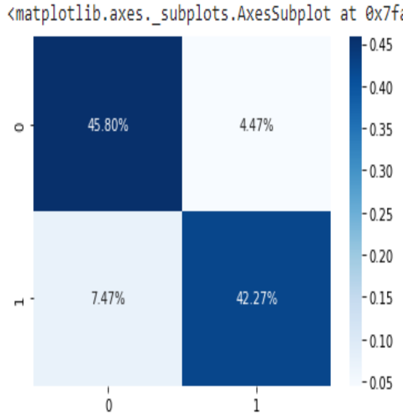
```
[ ] # Calculating Accuracy Score and Classification report
    print("Test Accuracy: %.2f%%" % (accuracy_score(y_test, y_pred) * 100))
    print(classification_report(y_test, y_pred))

    # Confusion matrix
    cf_matrix=confusion_matrix(y_test, y_pred)
    pd.DataFrame(cf_matrix)
```

```
Test Accuracy: 88.07%
              precision    recall  f1-score   support

           0       0.86      0.91      0.88       754
           1       0.90      0.85      0.88       746

    accuracy                           0.88      1500
   macro avg       0.88      0.88      0.88      1500
weighted avg       0.88      0.88      0.88      1500
```

	0	1
0	687	67
1	112	634

```
[ ] # Plot Confusion matrix
    sns.heatmap(cf_matrix/np.sum(cf_matrix), annot=True, fmt='.2%', cmap='Blues')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fa0dc074190>
```




```
[ ] # save model in pkl file
    joblib.dump(clf, dir_path + 'model/Logistic_Regression.pkl')
```

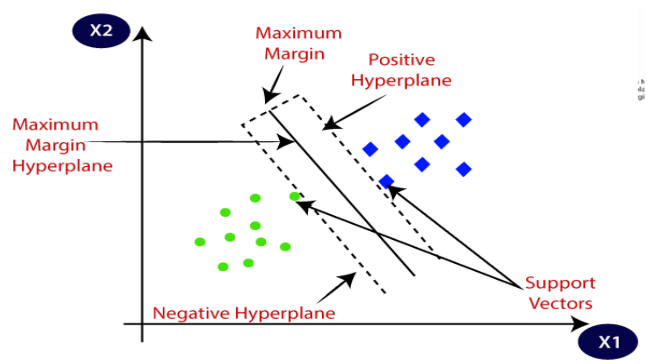
```
['/content/gdrive/MyDrive/Colab Notebooks/Final_Project_Yashkumar_1138765/model/Logistic_Regression.pkl']
```

```
# load model
model = joblib.load(dir_path + 'model/Logistic_Regression.pkl')

text= [" डी aww वह इस पृष्ठभूमि रंग से मेल खाता है मैं प्रतीत होता है कि धन्यवाद टॉक 21 51 जनवरी 11 2016 यूटीसी के साथ फंस गया है",
       "हाय मला वाटतं की तुम्ही सर्वात दुःखी व्यक्ती असाल ज्याच्याशी मी कधीही बोललो आहे एव्हर lgrainger199810",
       "ਤੁਹਾਨੂੰ ਇੱਕ ਬੇਵਕੂਫ ਹੋਣ ਲਈ ਟਰਾਉਟ ਕੀਤਾ ਗਿਆ ਹੈ",
       "You sir are my hero Any chance you remember what page that s on ",
       " મને લાગે છે કે હું ફ્રેડ ફેલ્પ્સની પાછળ ઉભો છું તે ક્યારેય તેની પાછળ કે સામે ઉભો રહેશે નહીં કારણ કે તે છે"]

# Tokenize the text
tok = tfidf.transform(text)

# predict using the model
prediction = model.predict(tok)

# Results of predications
for i in prediction:
  if(prediction[i] == 1):
    print("Toxic")
  else:
    print("Non-Toxic")
```

```
Non-Toxic
Toxic
Toxic
Non-Toxic
Toxic
```

2. Support Vector Machine

SVM is a supervised machine learning algorithm. It is helpful for both classification and regression challenges. SVM categorizes data points even when they are not linearly separable by mapping the data to a high-dimensional feature space. Once a separator divides the categories, the data are converted to make it possible to draw the separator as a hyperplane.



Referances

1. https://www.javatpoint.com/machine-learning-support-vector-machine-algorithm

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import joblib
```

```
from google.colab import drive
drive.mount('/content/gdrive', force_remount=True)
```

```
[ ] dir_path = "/content/gdrive/MyDrive/Colab Notebooks/Final_Project_Yashkumar_1138765/"
```

```
[ ] # Loading dataset
    data1=pd.read_excel(dir_path + "Data/Eng_Dataset.xlsx")
    data2=pd.read_excel(dir_path + "Data/Guj_Dataset.xlsx")
    data3=pd.read_excel(dir_path + "Data/Hin_Dataset.xlsx")
    data4=pd.read_excel(dir_path + "Data/Mar_Dataset.xlsx")
    data5=pd.read_excel(dir_path + "Data/Pun_Dataset.xlsx")
```

```
[ ] # Concatanating the files data
    data = pd.concat([
        data1[['comment_text', 'toxic']],
        data2[['comment_text', 'toxic']],
        data3[['comment_text', 'toxic']],
        data4[['comment_text', 'toxic']],
        data5[['comment_text', 'toxic']]
    ])
```

```
[ ] data.size
```

```
29990
```

```
[ ] data.shape
```

```
(14995, 2)
```

```
[ ] data['toxic'].value_counts()
```

```
1.0    7500
0.0    7495
Name: toxic, dtype: int64
```

```
[ ] data = data.astype({"toxic": int, "comment_text": str})
```

```
[ ] from sklearn.metrics import confusion_matrix, classification_report, accuracy_score
    from sklearn.feature_extraction.text import TfidfVectorizer
    from sklearn.model_selection import train_test_split
    from sklearn import svm
```

```
[ ] def tokenizer(text):
            return text.split()
```

```
[ ] # tokenizing data
    tfidf=TfidfVectorizer(strip_accents=None,lowercase=False,preprocessor=None,tokenizer=tokenizer,use_idf=True,norm='l2',smooth_idf=True)
    y=data.toxic.values
    x=tfidf.fit_transform(data.comment_text)
```

```
[ ] print(x[0])
```

```
  (0, 436)      0.13629397545473285
  (0, 487)      0.1516648558771504
  (0, 358)      0.14644587411427934
  (0, 643)      0.14397108902004568
  (0, 15161)    0.10990839624576292
  (0, 16929)    0.18820233374566683
  (0, 14358)    0.10178687548523987
  (0, 17590)    0.12276810984321544
  (0, 15502)    0.09225787054040967
  (0, 18390)    0.09929527054910753
  (0, 12499)    0.09429391208470374
  (0, 18481)    0.15003262861094374
  (0, 16762)    0.12983646313734015
  (0, 11339)    0.09975649758621642
  (0, 15868)    0.11737051127704197
  (0, 1012)     0.1131067208141196
  (0, 2877)     0.16757381381483724
  (0, 2577)     0.2023369596283723
  (0, 8267)     0.1767906739678217
  (0, 5339)     0.14780584159965618
  (0, 9008)     0.09384017867601845
  (0, 19433)    0.18820233374566683
  (0, 3819)     0.13350233622132449
  (0, 8506)     0.12471167070897449
  (0, 3234)     0.19406873352526693
```

```
[ ] # split the data in train and test set
    X_train,X_test,y_train,y_test=train_test_split(x,y,random_state=1,test_size=0.1,shuffle=True)
```

```
[ ] # training model
    clf = svm.SVC(C=1.0, kernel='linear', degree=3, gamma='auto').fit(X_train,y_train)
    y_pred = clf.predict(X_test)

    # Model Train and Test Accuracy
    print("Train Accuracy: %.2f" % clf.score(X_train, y_train))
    print("Test Accuracy: %.2f" % accuracy_score(y_test, y_pred))
```

```
Train Accuracy: 0.97
Test Accuracy: 0.87
```

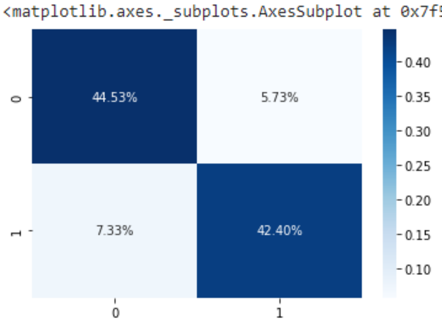
```
[ ] # Calculating Accuracy Score and Classification report
    print("Test Accuracy: %.2f%%" % (accuracy_score(y_test, y_pred) * 100))
    print(classification_report(y_test, y_pred))

    # Confusion matrix
    cf_matrix=confusion_matrix(y_test, y_pred)
    pd.DataFrame(cf_matrix)
```

```
Test Accuracy: 86.93%
              precision    recall  f1-score   support

           0       0.86      0.89      0.87       754
           1       0.88      0.85      0.87       746

    accuracy                           0.87      1500
   macro avg       0.87      0.87      0.87      1500
weighted avg       0.87      0.87      0.87      1500
```

	0	1
0	668	86
1	110	636

```
[ ] # Plot Confusion matrix
    sns.heatmap(cf_matrix/np.sum(cf_matrix), annot=True, fmt='.2%', cmap='Blues')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f54fc87d110>
```




```
[ ] # save model in pkl file
    joblib.dump(clf, dir_path + 'model/SVM.pkl')
```

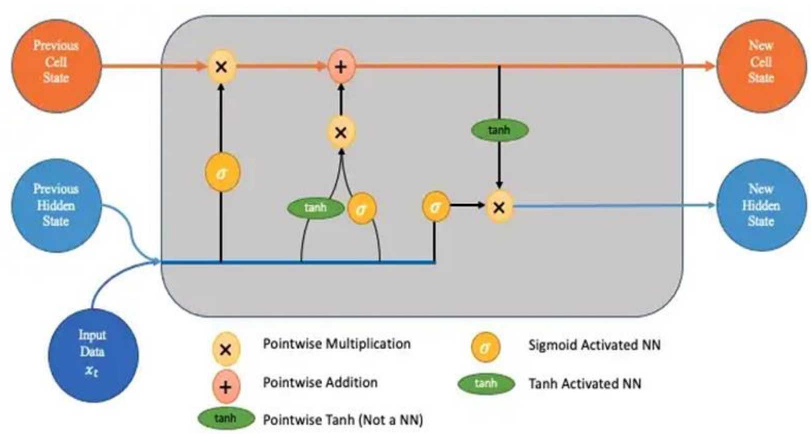
```
['/content/gdrive/MyDrive/Colab Notebooks/Final_Project_Yashkumar_1138765/model/SVM.pkl']
```

```
[ ] # load model
    model = joblib.load(dir_path + 'model/Logistic_Regression.pkl')

    text= [" डी aww वह इस पृष्ठभूमि रंग से मेल खाता है मैं प्रतीत होता है कि धन्यवाद टॉक 21 51 जनवरी 11 2016 यूटीसी के साथ फंस गया है",
           "हाय मला वाटतं की तुम्ही सर्वात दुःखी व्यक्ती असाल ज्याच्याशी मी कधीही बोललो आहे एव्हर lgrainger199810",
           "ਤੁਹਾਨੂੰ ਇੱਕ ਬੇਵਕੂਫ ਹੋਣ ਲਈ ਟਰਾਊਟ ਕੀਤਾ ਗਿਆ ਹੈ",
           "You sir are my hero Any chance you remember what page that s on ",
           " મને લાગે છે કે હું ફ્રેડ ફેલ્પ્સની પાછળ ઊભો છું તે ક્યારેય તેની પાછળ કે સામે ઊભો રહેશે નહીં કારણ કે તે છે"]

    # Tokenize the text
    tok = tfidf.transform(text)

    # predict using the model
    prediction = model.predict(tok)

    # Results of predications
    for i in prediction:
      if(prediction[i] == 1):
        print("Toxic")
      else:
        print("Non-Toxic")
```

```
Non-Toxic
Toxic
Toxic
Non-Toxic
Toxic
```

3. LSTM

Recurrent neural networks (RNNs) have a long-term dependency issue that LSTM networks solve. LSTMs include feedback connections.[26] With feedback connection property, LSTMs may process whole data sequences without considering each data point individually. Instead, they can process new data points by using the information from earlier data in the sequence to assist their processing. LSTMs are helpful for processing data sequences like text, audio, and general time series.



Referances

1. https://towardsdatascience.com/lstm-networks-a-detailed-explanation-8fae6aefc7f9

```
[ ] import numpy as np # linear algebra
    import pandas as pd # data processing, CSV file I/O
    from tqdm import tqdm
    from sklearn.model_selection import train_test_split
    import tensorflow as tf
    from keras.models import Sequential
    from keras.layers import LSTM
    from keras.layers.core import Dense, Activation, Dropout
    from keras.layers import Embedding
    from keras.utils import np_utils
    from sklearn import preprocessing, decomposition, model_selection, metrics, pipeline
    from keras.preprocessing import sequence, text
    from keras.callbacks import EarlyStopping
    from keras import regularizers

    import matplotlib.pyplot as plt
    import seaborn as sns
    %matplotlib inline
```

```
[ ] print("TF Version: ", tf.__version__)
    print("Keras Version: ", tf.keras.__version__)
```

```
TF Version:  2.9.2
Keras Version:  2.9.0
```

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

```
Mounted at /content/gdrive
```

```
[ ] from tensorflow.python.client import device_lib
    device_lib.list_local_devices()
```

```
[name: "/device:CPU:0"
 device_type: "CPU"
 memory_limit: 268435456
 locality {
 }
 incarnation: 6098051978135092642
 xla_global_id: -1, name: "/device:GPU:0"
```

```
[ ] dir_path = "/content/gdrive/MyDrive/Colab Notebooks/Final_Project_Yashkumar_1138765/"
```

```
[ ] # Loading dataset
    data1=pd.read_excel(dir_path + "Data/Eng_Dataset.xlsx")
    data2=pd.read_excel(dir_path + "Data/Guj_Dataset.xlsx")
    data3=pd.read_excel(dir_path + "Data/Hin_Dataset.xlsx")
    data4=pd.read_excel(dir_path + "Data/Mar_Dataset.xlsx")
    data5=pd.read_excel(dir_path + "Data/Pun_Dataset.xlsx")
```

```
[ ] # Concatanating the files data
    train = pd.concat([
        data1[['comment_text', 'toxic']],
        data2[['comment_text', 'toxic']],
        data3[['comment_text', 'toxic']],
        data4[['comment_text', 'toxic']],
        data5[['comment_text', 'toxic']]
    ])
```

```
[ ] train.head()
```

	comment_text	toxic
0	Explanation Why the edits made under my usema...	0.0
1	D aww He matches this background colour I m se...	0.0
2	Hey man I m really not trying to edit war It s...	0.0
3	More I can t make any real suggestions on imp...	0.0
4	You sir are my hero Any chance you remember wh...	0.0

```
[ ] train.tail()
```

	comment_text	toxic
2994	ਇਸ ਨੂੰ ਇਸ ਦੇ ਕਾਲੇ ਰਿੰਗ ਵਿੱਚ ਹਾਰਡ Fuck OOOOOOOO...	1.0
2995	ਪ੍ਰਸ਼ਾਸਕ ਕੁਲੀਨਵਾਦੀ ਹਨ, ਮੈਨੂੰ ਕੋਈ ਪਰਵਾਹ ਨਹੀਂ ਹ...	1.0
2996	ਇਸ ਵੇਸਲੇ ਆਊਟਰਿਗਰ ਨੂੰ ਫੱਕ ਕਰੋ ਅਤੇ ਬਾਅਦ ਵਿੱਚ ਜੌਨ...	1.0

```
# Shuffle dataset
train = train.sample(frac=1).reset_index(drop=True)
```

```
# split the data in train and test set
xtrain, xvalid, ytrain, yvalid = train_test_split(train.comment_text.values, train.toxic.values,
                                                  stratify=train.toxic.values,
                                                  random_state=42,
                                                  test_size=0.1, shuffle=True)
```

```
# Declaring Tokenizer
token = text.Tokenizer(num_words=None, lower=False)
max_len = 1500
```

```
# tokenizing data

token.fit_on_texts(list(xtrain) + list(xvalid))

# Converting texts to sequnces
xtrain_seq = token.texts_to_sequences(xtrain)
xvalid_seq = token.texts_to_sequences(xvalid)
```

```
xtrain_seq[0]
```

```
[1239,
 11416,
 5861,
 26510,
 26511,
 20862,
 26512,
 26513,
 11416,
 1,
 926,
 11417,
```

```
# Padding the sequences
xtrain_pad = tf.keras.utils.pad_sequences(xtrain_seq, maxlen=max_len)
xvalid_pad = tf.keras.utils.pad_sequences(xvalid_seq, maxlen=max_len)

word_index = token.word_index
```

```
xtrain_pad[0]
```

```
array([   0,    0,    0, ...,   96, 4100,  164], dtype=int32)
```

```
xtrain_pad[0].shape
```

```
(1500,)
```

```
# Regulazing Parameter
lamda = 0.01

# Sequential Neural Network
model = Sequential()
model.add(Embedding(len(word_index) + 1,
                    300,
                    input_length=max_len))
model.add(LSTM(128, kernel_regularizer=regularizers.L2(lamda)))
model.add(Dense(64, activation='relu'))
model.add(Dense(1, activation='sigmoid'))
```

```
# Compile model
model.compile(loss=tf.keras.losses.BinaryCrossentropy(), optimizer='adam', metrics=['accuracy'])
model.summary()
```

```
tf.keras.utils.plot_model(model, to_file="my_model.png", show_shapes=True)
```

embedding_input	input:	[(None, 1500)]
InputLayer	output:	[(None, 1500)]

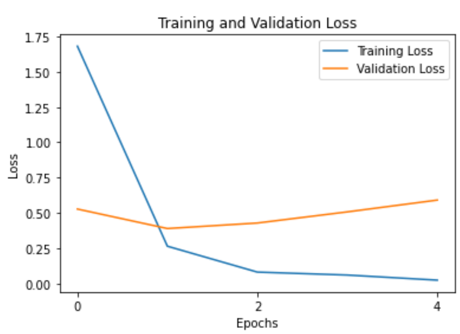
```python
# Callback EarlyStopping
es = tf.keras.callbacks.EarlyStopping(monitor='val_loss',
                                      patience=3,
                                      restore_best_weights=True,
                                      verbose=1)

# Fit the model
history=model.fit(xtrain_pad, ytrain,validation_split=0.2, epochs=5, batch_size=128, callbacks=[es])
```

```
Epoch 1/5
85/85 [==============================] - 21s 157ms/step - loss: 1.6810 - accuracy: 0.7147 - val_loss: 0.5279 - val_accuracy: 0.8396
Epoch 2/5
85/85 [==============================] - 13s 152ms/step - loss: 0.2647 - accuracy: 0.9250 - val_loss: 0.3899 - val_accuracy: 0.8637
Epoch 3/5
85/85 [==============================] - 13s 152ms/step - loss: 0.0813 - accuracy: 0.9840 - val_loss: 0.4289 - val_accuracy: 0.8540
Epoch 4/5
85/85 [==============================] - 13s 157ms/step - loss: 0.0605 - accuracy: 0.9893 - val_loss: 0.5078 - val_accuracy: 0.8585
Epoch 5/5
85/85 [==============================] - ETA: 0s - loss: 0.0244 - accuracy: 0.9979Restoring model weights from the end of the best epoch: 2.
85/85 [==============================] - 13s 154ms/step - loss: 0.0244 - accuracy: 0.9979 - val_loss: 0.5913 - val_accuracy: 0.8444
Epoch 5: early stopping
```

```python
# Taking values from history callback object
train_values = history.history['loss']
val_values = history.history['val_loss']
acc=history.history['accuracy']
val_acc=history.history['val_accuracy']

epoch=5
```

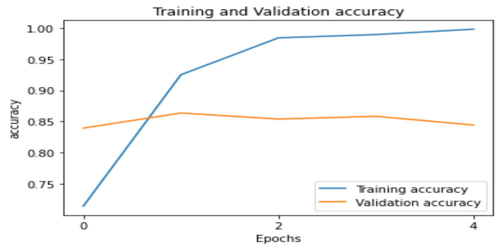
```python
from matplotlib.pylab import plt
from numpy import arange

# integers sequence to represent the epoch numbers
epochs = range(0, epoch)

# Plot and label loss values
plt.plot(epochs, train_values, label='Training Loss')
plt.plot(epochs, val_values, label='Validation Loss')
```

```python
plt.legend(loc='best')
plt.show()
```




```python
# integers sequence to represent the epoch numbers
epochs = range(0, epoch)

# Plot and label loss values
plt.plot(epochs, acc, label='Training accuracy')
plt.plot(epochs, val_acc, label='Validation accuracy')

# Add title and axes labels
plt.title('Training and Validation accuracy')
plt.xlabel('Epochs')
plt.ylabel('accuracy')

# tick locations
plt.xticks(arange(0, epoch, 2))

# Display plot
plt.legend(loc='best')
plt.show()
```

[ ]



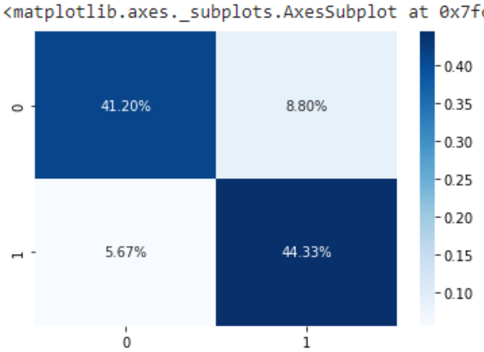
[ ]

```python
from sklearn.metrics import confusion_matrix, classification_report, accuracy_score

# Making predications on validation data
predictions = [1 if pr > 0.5 else 0
               for pr in model.predict(xvalid_pad, verbose=0).ravel()]

# Calculating Accuracy Score and Classification report
print("Accuracy: %.2f%%" % (accuracy_score(yvalid, predictions)*100))
print(classification_report(yvalid, predictions))

# Confusion matrix
cf_matrix =confusion_matrix(yvalid, predictions)
pd.DataFrame(cf_matrix)
```

```
Accuracy: 85.53%
              precision    recall  f1-score   support

           0       0.88      0.82      0.85       750
           1       0.83      0.89      0.86       750

    accuracy                           0.86      1500
   macro avg       0.86      0.86      0.86      1500
weighted avg       0.86      0.86      0.86      1500

       0    1
```

```python
# Plot Confusion matrix
sns.heatmap(cf_matrix/np.sum(cf_matrix), annot=True, fmt='.2%', cmap='Blues')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fcbb7a2f690>
```



[ ]

```python
path = dir_path + '/model/my_CNN_model.h5'
```

[ ]

```python
# save model in h5 file
model.save(path, save_format='h5', overwrite=True)
```

[ ]

```python
# load model
l_model = tf.keras.models.load_model(path)
l_model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding (Embedding)       (None, 1500, 300)         24357900

 lstm (LSTM)                 (None, 128)               219648

 dense (Dense)               (None, 64)                8256
```

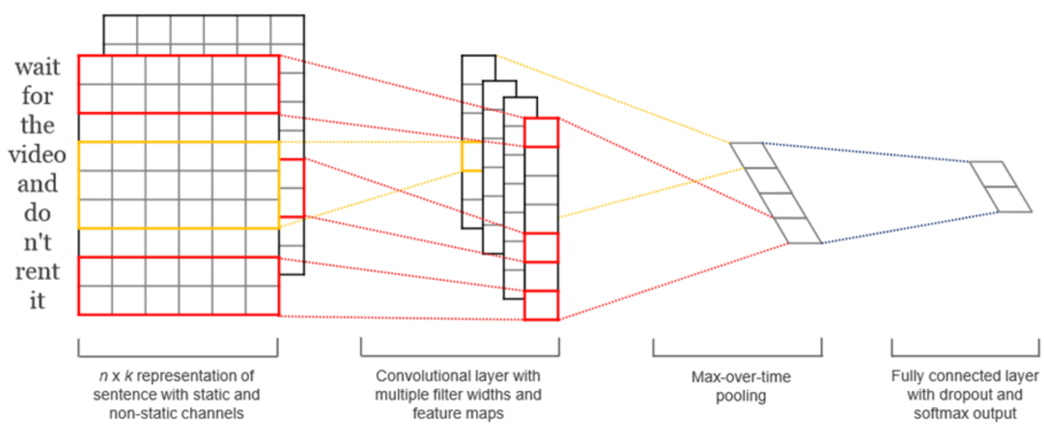
```
[ ]  # Evaluate model on actual data
     def evaluate(text):
       token.fit_on_texts(text)
       tokens = token.texts_to_sequences(text)
       seq = []
       for i in tokens:
         for j in i:
           seq.append(j)
       seq = np.array(seq)

       text_pad = tf.keras.utils.pad_sequences([seq,], maxlen=max_len)
       word_index = token.word_index

       result = l_model.predict(text_pad, verbose=0)
       print("%.5f" % result[0])

       if(result >=0.5):
         print("Toxic")
       else:
         print("Non-Toxic")
```

```
[ ]  text= " डी aww वह इस पृष्ठभूमि रंग से मेल खाता है में प्रतीत होता है कि धन्यवाद टॉक 21 51 जनवरी 11 2016 यूटीसी के साथ फंस गया है"

     evaluate(text)
```

```
0.16309
Non-Toxic
```

4. CNN

The most popular deep learning architectures for image processing and recognition are Convolutional neural networks (CNNs). Nevertheless, CNN's have recently become common in solving NLP-related issues. This technique treats each comment as an image by displaying the text in vector form and applying a CNN.



Referances

1. https://dennybritz.com/posts/wildml/implementing-a-cnn-for-text-classification-in-tensorflow/

```python
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O
from tqdm import tqdm
from sklearn.model_selection import train_test_split
import tensorflow as tf
from keras.models import Sequential
from keras.layers import LSTM, GRU,SimpleRNN
from keras.layers.core import Dense, Activation, Dropout
from keras.layers import Embedding
from keras.layers import BatchNormalization
from keras.utils import np_utils
from sklearn import preprocessing, decomposition, model_selection, metrics, pipeline
from keras.layers import Conv1D, MaxPooling1D, Flatten
from keras.preprocessing import sequence, text
from keras.callbacks import EarlyStopping
from keras import regularizers

import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

```python
[ ] print("TF Version: ", tf.__version__)
    print("Keras Version: ", tf.keras.__version__)
```

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

```
Drive already mounted at /content/gdrive; to attempt to forcibly remount, call drive.mount("/content/gdrive", force_remount=True).
```

```python
[ ] from tensorflow.python.client import device_lib
    device_lib.list_local_devices()
```

```
[name: "/device:CPU:0"
 device_type: "CPU"
 memory_limit: 268435456
 locality {
 }
 incarnation: 12354885265407430509
 xla_global_id: -1]
```

```python
[ ] dir_path = "/content/gdrive/MyDrive/Colab Notebooks/Final_Project_Yashkumar_1138765/"
```

```python
[ ] # Loading dataset
    data1=pd.read_excel(dir_path + "Data/Eng_Dataset.xlsx")
    data2=pd.read_excel(dir_path + "Data/Guj_Dataset.xlsx")
    data3=pd.read_excel(dir_path + "Data/Hin_Dataset.xlsx")
    data4=pd.read_excel(dir_path + "Data/Mar_Dataset.xlsx")
    data5=pd.read_excel(dir_path + "Data/Pun_Dataset.xlsx")
```

```python
[ ] # Concatanating the files data
    train = pd.concat([
        data1[['comment_text', 'toxic']],
        data2[['comment_text', 'toxic']],
        data3[['comment_text', 'toxic']],
        data4[['comment_text', 'toxic']],
        data5[['comment_text', 'toxic']]
    ])
```

```python
[ ] train.head()
```

	comment_text	toxic
0	Explanation Why the edits made under my userna...	0.0
1	D aww He matches this background colour I m se...	0.0
2	Hey man I m really not trying to edit war It s...	0.0
3	More I can t make any real suggestions on imp...	0.0
4	You sir are my hero Any chance you remember wh...	0.0

```python
[ ] train.tail()
```

```
[ ] # split the data in train and test set
    xtrain, xvalid, ytrain, yvalid = train_test_split(train.comment_text.values, train.toxic.values,
                                                      stratify=train.toxic.values,
                                                      random_state=42,
                                                      test_size=0.1, shuffle=True)
```

```
[ ] # Declaring Tokenizer
    token = text.Tokenizer(num_words=None, lower=False)
    max_len = 1500
```

```
[ ] # tokenizing data

    token.fit_on_texts(list(xtrain) + list(xvalid))

    # Converting texts to sequnces
    xtrain_seq = token.texts_to_sequences(xtrain)
    xvalid_seq = token.texts_to_sequences(xvalid)
```

```
[ ] xtrain_seq[0]
```

```
[31,
 185,
 81,
 214,
 14,
 76,
 789,
 3,
 21,
 31,
 11416,
 27,
 2195,
 3737,
 462,
 458,
 20862,
 271,
```

```
[ ] # Padding the sequences
    xtrain_pad = tf.keras.utils.pad_sequences(xtrain_seq, maxlen=max_len)
    xvalid_pad = tf.keras.utils.pad_sequences(xvalid_seq, maxlen=max_len)

    word_index = token.word_index
```

```
[ ] xtrain_pad[0]
```

```
array([  0,    0,    0, ..., 185, 759, 177], dtype=int32)
```

```
[ ] xtrain_pad[0].shape
```

```
(1500,)
```

```
[ ] # Sequential Neural Network
    model=Sequential()
    model.add(Embedding(len(word_index) + 1,
                        300,
                        input_length=max_len))
    model.add(Conv1D(64, 3, padding='same', activation='relu'))
    model.add(MaxPooling1D())
    model.add(Conv1D(32, 3, padding='same', activation='relu'))
    model.add(MaxPooling1D())
    model.add(Flatten())
    model.add(Dense(250, activation='relu'))
    model.add(Dense(1, activation='sigmoid'))
```

```
[ ] # Compile model
    model.compile(loss=tf.keras.losses.BinaryCrossentropy(), optimizer='adam', metrics=['accuracy'])
    model.summary()
```

```
Model: "sequential"
_________________________________________________________________
```

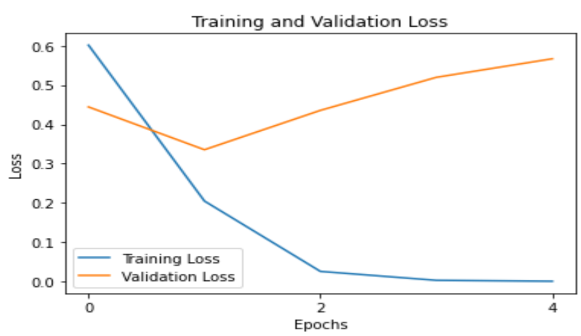
```
[ ] # Callback EarlyStopping
    es = tf.keras.callbacks.EarlyStopping(monitor='val_loss',
                                          patience=3,
                                          restore_best_weights=True,
                                          verbose=1)

    # Fit the model
    history=model.fit(xtrain_pad, ytrain,validation_split=0.2, epochs=5, batch_size=128, callbacks=[es])
```

```
Epoch 1/5
85/85 [==============================] - 211s 2s/step - loss: 0.6015 - accuracy: 0.6431 - val_loss: 0.4442 - val_accuracy: 0.7940
Epoch 2/5
85/85 [==============================] - 211s 2s/step - loss: 0.2046 - accuracy: 0.9190 - val_loss: 0.3352 - val_accuracy: 0.8555
Epoch 3/5
85/85 [==============================] - 209s 2s/step - loss: 0.0255 - accuracy: 0.9927 - val_loss: 0.4355 - val_accuracy: 0.8629
Epoch 4/5
85/85 [==============================] - 209s 2s/step - loss: 0.0028 - accuracy: 0.9994 - val_loss: 0.5196 - val_accuracy: 0.8614
Epoch 5/5
85/85 [==============================] - ETA: 0s - loss: 4.1384e-04 - accuracy: 1.0000Restoring model weights from the end of the best epoch: 2.
85/85 [==============================] - 212s 3s/step - loss: 4.1384e-04 - accuracy: 1.0000 - val_loss: 0.5668 - val_accuracy: 0.8607
Epoch 5: early stopping
```

```
[ ] # Taking values from history callback object
    train_values = history.history['loss']
    val_values = history.history['val_loss']
    acc=history.history['accuracy']
    val_acc=history.history['val_accuracy']

    epoch=5
```

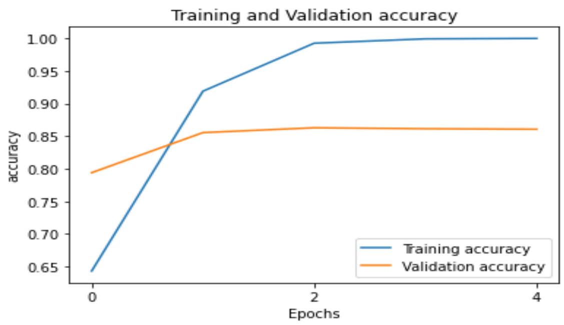
```
[ ] from matplotlib.pylab import plt
    from numpy import arange

    # integers sequence to represent the epoch numbers
    epochs = range(0, epoch)

    # Plot and label loss values
    plt.plot(epochs, train_values, label='Training Loss')
    plt.plot(epochs, val_values, label='Validation Loss')
```

```
[ ] # tick locations
    plt.xticks(arange(0, epoch, 2))

    # Display plot
    plt.legend(loc='best')
    plt.show()
```




```
[ ] # integers sequence to represent the epoch numbers
    epochs = range(0, epoch)

    # Plot and label loss values
    plt.plot(epochs, acc, label='Training accuracy')
    plt.plot(epochs, val_acc, label='Validation accuracy')

    # Add title and axes labels
    plt.title('Training and Validation accuracy')
    plt.xlabel('Epochs')
    plt.ylabel('accuracy')

    # tick locations
    plt.xticks(arange(0, epoch, 2))

    # Display plot
    plt.legend(loc='best')
    plt.show()
```

[ ]

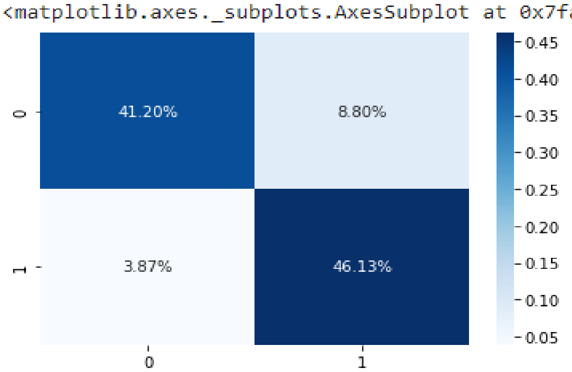



```
[ ] from sklearn.metrics import confusion_matrix, classification_report, accuracy_score

    # Making predications on validation data
    predictions = [1 if pr > 0.5 else 0
                   for pr in model.predict(xvalid_pad, verbose=0).ravel()]

    # Calculating Accuracy Score and Classification report
    print("Accuracy: %.2f%%" % (accuracy_score(yvalid, predictions)*100))
    print(classification_report(yvalid, predictions))

    # Confusion matrix
    cf_matrix =confusion_matrix(yvalid, predictions)
    pd.DataFrame(cf_matrix)
```

```
Accuracy: 87.33%
              precision    recall  f1-score   support

           0       0.91      0.82      0.87       750
           1       0.84      0.92      0.88       750

    accuracy                           0.87      1500
   macro avg       0.88      0.87      0.87      1500
weighted avg       0.88      0.87      0.87      1500
```

```
[ ] # Plot Confusion matrix
    sns.heatmap(cf_matrix/np.sum(cf_matrix), annot=True, fmt='.2%', cmap='Blues')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fafb91c3cd0>
```




```
[ ] path = dir_path + '/model/my_CNN_model.h5'
```

```
[ ] # save model in h5 file
    model.save(path, save_format='h5', overwrite=True)
```

```
[ ] # load model
    l_model = tf.keras.models.load_model(path)
    l_model.summary()
```

```
[ ] # Evaluate model on actual data
    def evaluate(text):
      token.fit_on_texts(text)
      tokens = token.texts_to_sequences(text)
      seq = []
      for i in tokens:
        for j in i:
          seq.append(j)
      seq = np.array(seq)

      text_pad = tf.keras.utils.pad_sequences([seq,], maxlen=max_len)
      word_index = token.word_index

      result = l_model.predict(text_pad, verbose=0)
      print("%.5f" % result[0])

      if(result >=0.5):
        print("Toxic")
      else:
        print("Non-Toxic")
```

```
[ ] text= " डी aww वह इस पृष्ठभूमि रंग से मेल खाता है में प्रतीत होता है कि धन्यवाद टॉक 21 51 जनवरी 11 2016 यूटीसी के साथ फंस गया है"

    evaluate(text)
```

```
0.01654
Non-Toxic
```

5. DistilBERT

Bidirectional Encoder Representations from Transformers are known as BERT. Towards the end of 2018, Google created and presented a brand-new language model. BERT is a multi-layer bidirectional Transformer encoder based on fine-tuning.

DistilBERT is a Transformer model based on the BERT architecture. To maximize training efficiency, DistilBERT strives to keep as much performance as possible while lowering the size of the BERT and speeding up the BERT. It is 40% smaller, 60% faster, and 97% functionally equivalent to the original BERT-base model.

Referances

1. https://www.marktechpost.com/2022/11/30/top-natural-language-processing-nlp-tools-platforms/
2. https://towardsdatascience.com/breaking-bert-down-430461f60efb

```
[ ] !pip3 install tqdm
    !pip3 install transformers
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Requirement already satisfied: tqdm in /usr/local/lib/python3.7/dist-packages (4.64.1)
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Collecting transformers
  Downloading transformers-4.24.0-py3-none-any.whl (5.5 MB)
     |████████████████████████████████| 5.5 MB 13.1 MB/s
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.7/dist-packages (from transformers) (21.3)
Collecting huggingface-hub<1.0,>=0.10.0
  Downloading huggingface_hub-0.11.1-py3-none-any.whl (182 kB)
     |████████████████████████████████| 182 kB 56.5 MB/s
Collecting tokenizers!=0.11.3,<0.14,>=0.11.1
  Downloading tokenizers-0.13.2-cp37-cp37m-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (7.6 MB)
     |████████████████████████████████| 7.6 MB 17.2 MB/s
Requirement already satisfied: filelock in /usr/local/lib/python3.7/dist-packages (from transformers) (3.8.0)
Requirement already satisfied: tqdm>=4.27 in /usr/local/lib/python3.7/dist-packages (from transformers) (4.64.1)
Requirement already satisfied: requests in /usr/local/lib/python3.7/dist-packages (from transformers) (2.23.0)
```

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

    Mounted at /content/gdrive
```

```
[ ] import numpy as np
    import pandas as pd
    import tensorflow as tf
    import transformers
    import matplotlib.pyplot as plt
    import tqdm
    import seaborn as sns

    %matplotlib inline

    # fix random seed for reproducibility
    seed = 42
    np.random.seed(seed)
    tf.random.set_seed(seed)

    print("TF Version: ", tf.__version__)
    print("Eager mode: ", tf.executing_eagerly())
    print("GPU is", "available" if tf.test.is_gpu_available() else "NOT AVAILABLE")

    WARNING:tensorflow:From <ipython-input-3-2df0a8d50504>:18: is_gpu_available (from tensorflow.python.framework.test_util) is deprecated and will be removed in a future version.
    Instructions for updating:
    Use `tf.config.list_physical_devices('GPU')` instead.
    TF Version:  2.9.2
    Eager mode:  True
    GPU is available
```

```
[ ] dir_path = "/content/gdrive/MyDrive/Colab Notebooks/Final_Project_Yashkumar_1138765/"
```

```
[ ] # Loading dataset
    data1=pd.read_excel(dir_path + "Data/Eng_Dataset.xlsx")
    data2=pd.read_excel(dir_path + "Data/Guj_Dataset.xlsx")
```

```
[ ] # split the data in train, validation and test set
    text = dataset['comment_text'].values
    toxic = dataset['toxic'].values

    x_train = text[:10000]
    x_val = text [10000:13000]
    x_test = text[13000:]

    y_train = toxic[:10000]
    y_val = toxic [10000:13000]
    y_test = toxic[13000:]
```

```
[ ] # Loading Distil-Bert tokenizer
    tokenizer = transformers.DistilBertTokenizer.from_pretrained('distilbert-base-multilingual-cased')

    Downloading: 100% 996k/996k [00:01<00:00, 967kB/s]
    Downloading: 100% 29.0/29.0 [00:00<00:00, 368B/s]
    Downloading: 100% 466/466 [00:00<00:00, 7.81kB/s]
```

This function combines multiple steps for us:

1. Split the sentence into tokens.
2. Add the special [CLS] and [SEP] tokens.
3. Map the tokens to their IDs.
4. Pad/truncate all sentences to the same length.
5. Create the attention masks which explicitly differentiate real tokens from [PAD] tokens.

[ ]
```
def create_distilbert_input_features(tokenizer, texts, max_seq_length):

    input_ids, input_masks= [], []

    for text in tqdm.tqdm(texts):

        # Tokeniz the text
        tokens = tokenizer.tokenize(text)

        # Truncate tokens with extra length
        if len(tokens) > max_seq_length-2:
            tokens = tokens[0 : (max_seq_length-2)]

        # Append [CLS] at start and [SEP] at end of sequence
        tokens = ['[CLS]'] + tokens + ['[SEP]']

        # Convert tokens to ids
        ids = tokenizer.convert_tokens_to_ids(tokens)

        # Create mask list of '1' same as sequence length
        masks = [1] * len(ids)

        # Pad sequences and make all same size
        while len(ids) < max_seq_length:
            ids.append(0)
            masks.append(0)

        input_ids.append(ids)
        input_masks.append(masks)

    return np.array([input_ids, input_masks])
```

[ ]
```
max_seq_len = 250

inp_id = tf.keras.layers.Input(shape=(max_seq_len,), dtype='int32', name="bert_input_ids")
inp_mask = tf.keras.layers.Input(shape=(max_seq_len,), dtype='int32', name="bert_input_masks")
inputs = [inp_id, inp_mask]

# Adding pre-trained model onto Sequential Neural Network for fine tuning
hidden_state = transformers.TFDistilBertModel.from_pretrained('distilbert-base-multilingual-cased',  num_labels=2)(inputs)[0]
pooled_output = hidden_state[:, 0]
dense1 = tf.keras.layers.Dense(256, activation='relu')(pooled_output)
drop1 = tf.keras.layers.Dropout(0.25)(dense1)
dense2 = tf.keras.layers.Dense(128, activation='relu')(drop1)
drop2 = tf.keras.layers.Dropout(0.25)(dense2)
dense3 = tf.keras.layers.Dense(64, activation='relu')(drop2)
drop3 = tf.keras.layers.Dropout(0.25)(dense3)
output = tf.keras.layers.Dense(1, activation='sigmoid')(drop3)

model = tf.keras.Model(inputs=inputs, outputs=output)
```

```
Downloading: 100% 911M/911M [00:18<00:00, 63.8MB/s]
Some layers from the model checkpoint at distilbert-base-multilingual-cased were not used when initializing TFDistilBertModel: ['vocab_
- This IS expected if you are initializing TFDistilBertModel from the checkpoint of a model trained on another task or with another arc
- This IS NOT expected if you are initializing TFDistilBertModel from the checkpoint of a model that you expect to be exactly identical
All the layers of TFDistilBertModel were initialized from the model checkpoint at distilbert-base-multilingual-cased.
If your task is similar to the task the model of the checkpoint was trained on, you can already use TFDistilBertModel for predictions w
```

[ ]
```
# Compile model
model.compile(optimizer=tf.optimizers.Adam(learning_rate=2e-5,
                                           epsilon=1e-08),
              loss='binary_crossentropy', metrics=['accuracy'])

model.summary()
```

```
Model: "model"
```

```python
# Converting train data into input ids and attention mask
train_features_ids, train_features_masks = create_distilbert_input_features(tokenizer, x_train,
                                                                             max_seq_length=max_seq_len)

# Converting validation data into input ids and attention mask
val_features_ids, val_features_masks = create_distilbert_input_features(tokenizer, x_val,
                                                                         max_seq_length=max_seq_len)

print('Train Features:', train_features_ids.shape, train_features_masks.shape)
print('Val Features:', val_features_ids.shape, val_features_masks.shape)
```

```
100%|██████████| 10000/10000 [00:24<00:00, 415.80it/s]
100%|██████████| 3000/3000 [00:05<00:00, 510.53it/s]
Train Features: (10000, 250) (10000, 250)
Val Features: (3000, 250) (3000, 250)
```

```python
print(train_features_ids)
```

```
[[  101 27746 31609 ...     0     0     0]
 [  101   141 56237 ...     0     0     0]
 [  101 35936 10817 ...     0     0     0]
 ...
 [  101 44551 18869 ...     0     0     0]
 [  101   885 18321 ... 15070 50051   102]
 [  101   898 13841 ... 77285   117   102]]
```

```python
# Attention masks
print(train_features_masks)
```

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

```python
# Callback EarlyStopping
es = tf.keras.callbacks.EarlyStopping(monitor='val_loss',
                                      patience=3,
                                      restore_best_weights=True,
                                      verbose=1)

# Fit the model
model.fit([train_features_ids,
           train_features_masks], y_train,
           validation_data=([val_features_ids,
                             val_features_masks], y_val),
          epochs=5,
          batch_size=5,
          callbacks=[es],
          shuffle=True,
          verbose=1)
```

```
Epoch 1/5
2000/2000 [==============================] - 391s 193ms/step - loss: 0.4468 - accuracy: 0.7832 - val_loss: 3.4007 - val_accuracy: 0.5407
Epoch 2/5
2000/2000 [==============================] - 383s 192ms/step - loss: 0.2656 - accuracy: 0.8933 - val_loss: 4.3603 - val_accuracy: 0.5383
Epoch 3/5
2000/2000 [==============================] - 398s 199ms/step - loss: 0.1718 - accuracy: 0.9313 - val_loss: 3.7490 - val_accuracy: 0.5350
Epoch 4/5
2000/2000 [==============================] - ETA: 0s - loss: 0.1123 - accuracy: 0.9596Restoring model weights from the end of the best epoch: 1.
2000/2000 [==============================] - 398s 199ms/step - loss: 0.1123 - accuracy: 0.9596 - val_loss: 4.4655 - val_accuracy: 0.5313
Epoch 4: early stopping
<keras.callbacks.History at 0x7f31420a39d0>
```

```python
# Converting test data into input ids and attention mask
test_features_ids, test_features_masks = create_distilbert_input_features(tokenizer, x_test,
                                                                           max_seq_length=max_seq_len)
print('Test Features:', test_features_ids.shape, test_features_masks.shape)
```

```
100%|██████████| 1995/1995 [00:02<00:00, 689.56it/s]
Test Features: (1995, 250) (1995, 250)
```

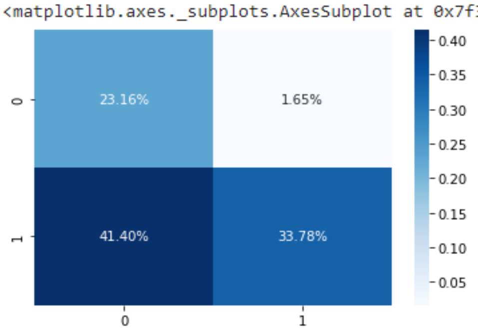
```
[ ] from sklearn.metrics import confusion_matrix, classification_report, accuracy_score

    # Making predications on validation data
    predictions = [1 if pr > 0.5 else 0
                   for pr in model.predict([test_features_ids,
                                            test_features_masks], verbose=0).ravel()]

    # Calculating Accuracy Score and Classification report
    print("Accuracy: %.2f%%" % (accuracy_score(y_test, predictions)*100))
    print(classification_report(y_test, predictions))

    # Confusion matrix
    cf_matrix =confusion_matrix(y_test, predictions)
    pd.DataFrame(cf_matrix)
```

```
Accuracy: 56.94%
              precision    recall  f1-score   support

         0.0       0.36      0.93      0.52       495
         1.0       0.95      0.45      0.61      1500

    accuracy                           0.57      1995
   macro avg       0.66      0.69      0.56      1995
weighted avg       0.81      0.57      0.59      1995
```

	0	1
0	462	33
1	826	674

```
[ ] # Plot Confusion matrix
    sns.heatmap(cf_matrix/np.sum(cf_matrix), annot=True, fmt='.2%', cmap='Blues')
```

<matplotlib.axes._subplots.AxesSubplot at 0x7f30b3e3ddd0>




```
[ ] path = dir_path + '/model/'

    # Save weights in h5 file
    model.save_weights(path + "my_Distil_Bert_model_Weights.h5", save_format='h5', overwrite=True)
```

```
[ ] l_model = model

    # Load Weights in the model
    l_model.load_weights(path + "my_Distil_Bert_model_Weights.h5")
    l_model.summary()
```

```
Model: "model"
__________________________________________________________________________________________________
 Layer (type)                   Output Shape         Param #     Connected to
==================================================================================================
 bert_input_ids (InputLayer)    [(None, 250)]        0           []

 bert_input_masks (InputLayer)  [(None, 250)]        0           []

 tf_distil_bert_model (TFDistil  TFBaseModelOutput(l  134734080  ['bert_input_ids[0][0]',
 BertModel)                     ast_hidden_state=(N               'bert_input_masks[0][0]']
                                one, 250, 768),
                                 hidden_states=None
```

```
==================================================================================================
Total params: 134,972,161
Trainable params: 134,972,161
Non-trainable params: 0
__________________________________________________________________________________________________
```

```python
text= [" डी aww वह इस पृष्ठभूमि रंग से मेल खाता है मैं प्रतीत होता है कि धन्यवाद टॉक 21 51 जनवरी 11 2016 यूटीसी के साथ फंस गया है",
       "हाय मला वाटतं की तुम्ही सर्वात दुःखी व्यक्ती असाल ज्याच्याशी मी कधीही बोललो आहे एव्हर 1grainger199810",
       "ਤੁਹਾਨੂੰ ਇੱਕ ਬੇਵਕੂਫ ਹੋਣ ਲਈ ਟਰਾਊਟ ਕੀਤਾ ਗਿਆ ਹੈ",
       "You sir are my hero Any chance you remember what page that s on ",
       " મને લાગે છે કે હું ફ્રેડ ફેલ્પ્સની પાછળ ઊભો છું તે ક્યારેય તેની પાછળ કે સામે ઊભો રહેશે નહીં કારણ કે તે છે"]

# Converting text into ids and masks
test_features_ids, test_features_masks = create_distilbert_input_features(tokenizer, text,
                                                                          max_seq_length=max_seq_len)

# Classifying the text
predictions = ["Toxic" if pr > 0.5 else "Non-Toxic"
               for pr in l_model.predict([test_features_ids,
                                          test_features_masks], verbose=0).ravel()]

print("\n")
print(predictions)
```

```
100%|██████████| 5/5 [00:00<00:00, 1133.84it/s]


['Non-Toxic', 'Non-Toxic', 'Non-Toxic', 'Non-Toxic', 'Non-Toxic']
```

```python
# Loading Pre-trained DistilBert Sequnence classifier
distil_Model = transformers.TFDistilBertForSequenceClassification.from_pretrained('distilbert-base-multilingual-cased',  num_labels=2)
```

```
Some layers from the model checkpoint at distilbert-base-multilingual-cased were not used when initializing TFDistilBertForSequenceClassification: ['vocab_trans
- This IS expected if you are initializing TFDistilBertForSequenceClassification from the checkpoint of a model trained on another task or with another architec
- This IS NOT expected if you are initializing TFDistilBertForSequenceClassification from the checkpoint of a model that you expect to be exactly identical (ini
Some layers of TFDistilBertForSequenceClassification were not initialized from the model checkpoint at distilbert-base-multilingual-cased and are newly initiali
You should probably TRAIN this model on a down-stream task to be able to use it for predictions and inference.
```

```python
# Compile pre-trained classification model
distil_Model.compile(optimizer=tf.optimizers.Adam(learning_rate=2e-5, epsilon=1e-08),
                     loss='binary_crossentropy', metrics=['accuracy'])

# Callback EarlyStopping
es = tf.keras.callbacks.EarlyStopping(monitor='val_loss',
                                      patience=3,
                                      restore_best_weights=True,
                                      verbose=1)

# Fit the model
distil_Model.fit([train_features_ids,
           train_features_masks], y_train,
           validation_data=([val_features_ids,
                             val_features_masks], y_val),
          epochs=5,
          batch_size=5,
          callbacks=[es],
          shuffle=True,
          verbose=1)
```

```
Epoch 1/5
2000/2000 [==============================] - 405s 200ms/step - loss: 0.5292 - accuracy: 0.4498 - val_loss: 6.0128 - val_accuracy: 0.5100
Epoch 2/5
2000/2000 [==============================] - 383s 191ms/step - loss: 0.4191 - accuracy: 0.5597 - val_loss: 6.6585 - val_accuracy: 0.3827
Epoch 3/5
2000/2000 [==============================] - 396s 198ms/step - loss: 0.3994 - accuracy: 0.6133 - val_loss: 7.0184 - val_accuracy: 0.1697
Epoch 4/5
2000/2000 [==============================] - ETA: 0s - loss: 0.4592 - accuracy: 0.5167Restoring model weights from the end of the best epoch: 1.
2000/2000 [==============================] - 383s 191ms/step - loss: 0.4592 - accuracy: 0.5167 - val_loss: 6.7905 - val_accuracy: 0.4883
Epoch 4: early stopping
<keras.callbacks.History at 0x7f30b38116d0>
```

```
[ ] # saving pre-trained model
    path = dir_path + 'model/Distil'
    distil_Model.save_pretrained(path, saved_model=True)

    WARNING:tensorflow:Skipping full serialization of Keras layer <keras.layers.regularization.dropout.Dropout object at 0x7f3142839d90>, because it is not built.
    WARNING:tensorflow:Skipping full serialization of Keras layer <keras.layers.regularization.dropout.Dropout object at 0x7f3142814210>, because it is not built.
    WARNING:tensorflow:Skipping full serialization of Keras layer <keras.layers.regularization.dropout.Dropout object at 0x7f31426cdd90>, because it is not built.
    WARNING:tensorflow:Skipping full serialization of Keras layer <keras.layers.regularization.dropout.Dropout object at 0x7f31425694d0>, because it is not built.
    WARNING:tensorflow:Skipping full serialization of Keras layer <keras.layers.regularization.dropout.Dropout object at 0x7f31420d2c90>, because it is not built.
    WARNING:tensorflow:Skipping full serialization of Keras layer <keras.layers.regularization.dropout.Dropout object at 0x7f3142415690>, because it is not built.
    WARNING:absl:Found untraced functions such as embeddings_layer_call_fn, embeddings_layer_call_and_return_conditional_losses, transformer_layer_call_fn, transformer_layer_call_an

[ ] # Loading the pre-trained model
    distil_Model = transformers.TFDistilBertForSequenceClassification.from_pretrained(path,  id2label={0: 'Non-Toxic', 1: 'Toxic'})

    Some layers from the model checkpoint at /content/gdrive/MyDrive/Colab Notebooks/Final_Project_Yashkumar_1138765/model/Distil were not used when initializing TFDistilBertForSequ
    - This IS expected if you are initializing TFDistilBertForSequenceClassification from the checkpoint of a model trained on another task or with another architecture (e.g. initia
    - This IS NOT expected if you are initializing TFDistilBertForSequenceClassification from the checkpoint of a model that you expect to be exactly identical (initializing a BertF
    Some layers of TFDistilBertForSequenceClassification were not initialized from the model checkpoint at /content/gdrive/MyDrive/Colab Notebooks/Final_Project_Yashkumar_1138765/mo
    You should probably TRAIN this model on a down-stream task to be able to use it for predictions and inference.

[ ] text = text= [" डी aww वह इस पृष्ठभूमि रंग से मेल खाता है में प्रतीत होता है कि धन्यवाद टॉक 21 51 जनवरी 11 2016 यूटीसी के साथ फंस गया है",
            "हाय मला वाटतं की तुम्ही सर्वात दुःखी व्यक्ती असाल ज्याच्याशी मी कधीही बोललो आहे एव्हर 1grainger199810",
            "ਤੁਹਾਨੂੰ ਇੱਕ ਬੇਵਕੂਫ ਹੋਣ ਲਈ ਟਰਾਉਟ ਕੀਤਾ ਗਿਆ ਹੈ",
            "You sir are my hero Any chance you remember what page that s on ",
            " મને લાગે છે કે હું ફ્રેડ ફ્રેલ્સની પાછળ ઊભો છું તે ક્યારેય તેની પાછળ કે સામે ઊભો રહેશે નહીં કારણ કે તે છે"]

    # Classifying the text
    pipe = transformers.TextClassificationPipeline(model=distil_Model, tokenizer=tokenizer)
    pipe(text)

    [{'label': 'Non-Toxic', 'score': 0.5465138554573059},
     {'label': 'Non-Toxic', 'score': 0.5586320161819458},
     {'label': 'Non-Toxic', 'score': 0.5388920903205872},
     {'label': 'Non-Toxic', 'score': 0.5120198726654053},
     {'label': 'Non-Toxic', 'score': 0.5395165085792542}]
```

Referances:

1. https://www.kaggle.com/code/afi1289/nlp-with-disaster-tweets-cleaning-tf-idf-and-bert#7.-Fine-Tuning-BERT:
2. https://victordibia.com/blog/text-classification-hf-tf2/
3. https://stackoverflow.com/questions/47266383/save-and-load-weights-in-keras
4. https://github.com/keras-team/keras/issues/6447
5. https://github.com/dipanjanS/deep_transfer_learning_nlp_dhs2019/tree/master/notebooks
6. https://huggingface.co/
7. https://thecleverprogrammer.com/2020/12/06/resume-screening-with-python/#google_vignette
8. https://thecleverprogrammer.com/2020/12/07/sentiment-analysis-with-python/#google_vignette

VI. Libraries

1. TensorFlow

TensorFlow is an end-to-end open-source platform for machine learning. Researchers can advance the state-of-the-art in machine learning thanks to its extensive, adaptable ecosystem of tools, libraries, and community resources. At the same time, developers can create and deploy ML-powered applications [4].

To undertake machine learning and deep neural network research, the Google Brain team, a group of researchers and engineers within Google's Machine Intelligence Research division, created TensorFlow. The system is broad enough to work in several additional domains.

2. Keras

Keras is an open-source high-level Neural Network library, and it is written in Python. Francois Chollet, a Google developer, created it. It is user-friendly, expandable, and modular, enabling quicker experimentation with deep neural networks. It supports both Convolutional and Recurrent Networks separately as well as in combination. [3]

The sequential method of TensorFlow groups a linear stack of layers into a tf.keras.Model. So, in the below image Sequential method has grouped one embedding layer: one LSTM layer and two Dense layers. Kernel Regularization is used to apply penalties on layer parameters or layer activity during optimization. Because of regularization, the model does not overfit.

The Compile method configures the model for training. The Loss function and optimizers are defined in this method.

```
[ ] # Regulazing Parameter
    lamda = 0.01

    # Sequential Neural Network
    model = Sequential()
    model.add(Embedding(len(word_index) + 1,
                        300,
                        input_length=max_len))
    model.add(LSTM(128, kernel_regularizer=regularizers.L2(lamda)))
    model.add(Dense(64, activation='relu'))
    model.add(Dense(1, activation='sigmoid'))
```

```
# Compile model
model.compile(loss=tf.keras.losses.BinaryCrossentropy(), optimizer='adam', metrics=['accuracy'])
model.summary()
```

The Early Stopping is a callback, and it stops model training when a monitored metric has stopped improving. The model.fit() method trains the model for a fixed number of epochs. The validation split, epochs, batch size, and callbacks are parameters of the fit method.

```
# Callback EarlyStopping
es = tf.keras.callbacks.EarlyStopping(monitor='val_loss',
                                      patience=3,
                                      restore_best_weights=True,
                                      verbose=1)

# Fit the model
history=model.fit(xtrain_pad, ytrain,validation_split=0.2, epochs=5, batch_size=128, callbacks=[es])
```

The model.save() method is used to save model architecture, weights, and optimizer rates. The model.load() method loads model and all the parameters.

```
[ ] path = dir_path + '/model/my_CNN_model.h5'
```

```
[ ] # save model in h5 file
    model.save(path, save_format='h5', overwrite=True)
```

```
[ ] # load model
    l_model = tf.keras.models.load_model(path)
    l_model.summary()
```

In production, model.predict() method is used to predict the results.

```
result = l_model.predict(text_pad, verbose=0)
print("%.5f" % result[0])
```

3. SKLearn

Scikit-learn is the most helpful library for machine learning in Python. Classification, regression, clustering, and dimensionality reduction are just a few of the practical machine learning and statistical modelling algorithms in the SKLearn toolkit. [5]

The TfidfVectorizer() method of SKLearn converts a collection of raw documents to a matrix of TF-IDF features. These features are input to the model.

```
[ ] def tokenizer(text):
            return text.split()
```

```
[ ] # tokenizing data
    tfidf=TfidfVectorizer(strip_accents=None,lowercase=False,preprocessor=None,tokenizer=tokenizer,use_idf=True,norm='l2',smooth_idf=True)
    y=data.toxic.values
    x=tfidf.fit_transform(data.comment_text)
```

The train_test_split() method is used to split the data into random test and train matrics. The LogisticRegressionCV() method trains the model using Logistic Regression algorithm. To evaluate the model, SKLearn provides bunch of methods like score(), accuracy_score() and classification_report().

```
[ ] # split the data in train and test set
    X_train,X_test,y_train,y_test=train_test_split(x,y,random_state=1,test_size=0.1,shuffle=True)
```

```
[ ] # training model
    clf=LogisticRegressionCV(cv=6,scoring='accuracy',random_state=0,n_jobs=-1,verbose=0,max_iter=500).fit(X_train,y_train)
    y_pred = clf.predict(X_test)

    # Model Train and Test Accuracy
    print("Train Accuracy: %.2f" % clf.score(X_train, y_train))
    print("Test Accuracy: %.2f" % accuracy_score(y_test, y_pred))
```

```
Train Accuracy: 1.00
Test Accuracy: 0.88
```

```
[ ] # Calculating Accuracy Score and Classification report
    print("Test Accuracy: %.2f%%" % (accuracy_score(y_test, y_pred) * 100))
    print(classification_report(y_test, y_pred))

    # Confusion matrix
    cf_matrix=confusion_matrix(y_test, y_pred)
    pd.DataFrame(cf_matrix)
```

```
Test Accuracy: 88.07%
              precision    recall  f1-score   support

           0       0.86      0.91      0.88       754
           1       0.90      0.85      0.88       746

    accuracy                           0.88      1500
   macro avg       0.88      0.88      0.88      1500
weighted avg       0.88      0.88      0.88      1500
```

	0	1
0	687	67
1	112	634

4. Hugging Faces - Transformers

Transformers provides APIs and tools to Pretrained models that can save the time and resources needed to train a model from scratch while lowering computing expenses and carbon footprint. These models facilitate typical tasks in several modalities, including NLP, CV, and Audio, quickly download and train state-of-the-art pre-trained models.

```
[ ] # Loading Distil-Bert tokenizer
    tokenizer = transformers.DistilBertTokenizer.from_pretrained('distilbert-base-multilingual-cased')
```

```
Downloading: 100% 996k/996k [00:01<00:00, 967kB/s]
Downloading: 100% 29.0/29.0 [00:00<00:00, 368B/s]
Downloading: 100% 466/466 [00:00<00:00, 7.81kB/s]
```

```
[ ] max_seq_len = 250

    inp_id = tf.keras.layers.Input(shape=(max_seq_len,), dtype='int32', name="bert_input_ids")
    inp_mask = tf.keras.layers.Input(shape=(max_seq_len,), dtype='int32', name="bert_input_masks")
    inputs = [inp_id, inp_mask]

    # Adding pre-trained model onto Sequential Neural Network for fine tuning
    hidden_state = transformers.TFDistilBertModel.from_pretrained('distilbert-base-multilingual-cased',  num_labels=2)(inputs)[0]
    pooled_output = hidden_state[:, 0]
    dense1 = tf.keras.layers.Dense(256, activation='relu')(pooled_output)
    drop1 = tf.keras.layers.Dropout(0.25)(dense1)
    dense2 = tf.keras.layers.Dense(128, activation='relu')(drop1)
    drop2 = tf.keras.layers.Dropout(0.25)(dense2)
    dense3 = tf.keras.layers.Dense(64, activation='relu')(drop2)
    drop3 = tf.keras.layers.Dropout(0.25)(dense3)
    output = tf.keras.layers.Dense(1, activation='sigmoid')(drop3)

    model = tf.keras.Model(inputs=inputs, outputs=output)
```

5. NumPy

NumPy stands for Numerical Python, the standard Python library for working with arrays (vectors and matrices), linear algebra, and other numerical computations. NumPy is written in C. Its arrays are quicker and use less memory than Python lists or arrays. [7]

6. Pandas

Pandas is an open-source Python package. It is fast, powerful, flexible, and easy for data science/analysis and machine learning tasks. Moreover, It is built on top of the Python programming language.

```
[ ] dir_path = "/content/gdrive/MyDrive/Colab Notebooks/Final_Project_Yashkumar_1138765/"
```

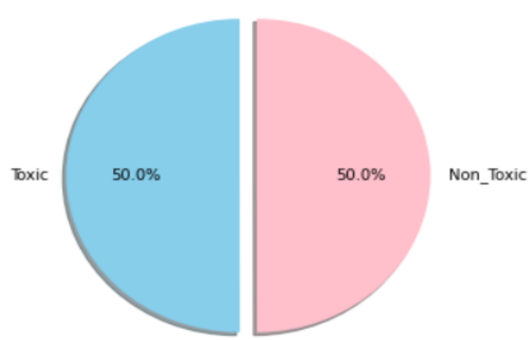
```
# Loading dataset
data1=pd.read_excel(dir_path + "Data/Eng_Dataset.xlsx")
data2=pd.read_excel(dir_path + "Data/Guj_Dataset.xlsx")
data3=pd.read_excel(dir_path + "Data/Hin_Dataset.xlsx")
data4=pd.read_excel(dir_path + "Data/Mar_Dataset.xlsx")
data5=pd.read_excel(dir_path + "Data/Pun_Dataset.xlsx")
```

```
[ ] # Concatanating the files data
data = pd.concat([
    data1[['comment_text', 'toxic']],
    data2[['comment_text', 'toxic']],
    data3[['comment_text', 'toxic']],
    data4[['comment_text', 'toxic']],
    data5[['comment_text', 'toxic']]
])
```

7. Matplotlib

Matplotlib is a comprehensive Python library for creating static, animated, and interactive visualizations. Matplotlib can is used in Python scripts, Python shells, web application servers, and various graphical user interface toolkits.

```
# Plotting the data distibution
fig=plt.figure(figsize=(5,5))
colors=["skyblue",'pink']
pos=data[data['toxic']==1]
neg=data[data['toxic']==0]
ck=[pos['toxic'].count(),neg['toxic'].count()]
legpie=plt.pie(ck,labels=["Toxic","Non_Toxic"],
                autopct ='%1.1f%%',
                shadow = True,
                colors = colors,
                startangle = 90,
                explode=(0, 0.1))
```



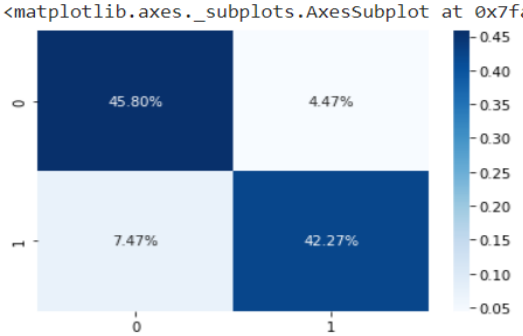
```python
from matplotlib.pylab import plt
from numpy import arange

# integers sequence to represent the epoch numbers
epochs = range(0, epoch)

# Plot and label loss values
plt.plot(epochs, train_values, label='Training Loss')
plt.plot(epochs, val_values, label='Validation Loss')

# Add title and axes labels
plt.title('Training and Validation Loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')

# tick locations
plt.xticks(arange(0, epoch, 2))

# Display plot
plt.legend(loc='best')
plt.show()
```

8. Seaborn

Seaborn is a Python data visualization library based on Matplotlib. It provides a high-level interface for drawing attractive and informative statistical graphics. [10] The heatmap () method plots the confusion matrix.

```python
# Plot Confusion matrix
sns.heatmap(cf_matrix/np.sum(cf_matrix), annot=True, fmt='.2%', cmap='Blues')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fa0dc074190>
```



9. Joblib

Joblib is a set of tools to provide lightweight pipelining in Python. In particular: transparent disk-caching of functions and lazy re-evaluation is accessible, simple parallel computing. [11]

The dump() method saves all the model and weights into .pkl file and load() method is used to load the model back using .pkl file.

```
[ ] # save model in pkl file
    joblib.dump(clf, dir_path + 'model/Logistic_Regression.pkl')

    ['/content/gdrive/MyDrive/Colab Notebooks/Final_Project_Yashkumar_1138765/model/Logistic_Regression.pkl']
```

```
# load model
model = joblib.load(dir_path + 'model/Logistic_Regression.pkl')
```

VII. References

1. https://colab.research.google.com
2. https://research.google.com/colaboratory/faq.html
3. https://keras.io
4. https://www.tensorflow.org/learn
5. https://scikit-learn.org/stable/
6. https://huggingface.co/docs/transformers/main/en/index
7. https://numpy.org/doc/stable/user/quickstart.html
8. https://pandas.pydata.org/
9. https://pypi.org/project/matplotlib/
10. https://seaborn.pydata.org/
11. https://joblib.readthedocs.io/en/latest/