Configuration Manual

MSc Research Project
Cybersecurity

Samrat Shah
Student ID: X21189919

School of Computing
National College of Ireland

Supervisor: Prof. Raza Ul Mustafa

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	Samrat Sanjaykumar Shah
Student ID:	X21189919
Programme:	MSc. Cybersecurity **Year:** 2024
Module:	MSc Research Project
Lecturer:	Prof. Raza Ul Mustafa
Submission Due Date:	27/05/2024
Project Title:	Email Spam Detection: Leveraging Fine-Tuned Transformer Models with Attention Mechanism.
Word Count:	1084 **Page Count: 18**

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.

ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature:……………………………………………………………………………………………………

Date: ……………………………………………………………………………………………………

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	□
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	□
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	□

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Configuration Manual

Samrat Shah X21189919

1 Introduction

This article provides more detail about the planned system's specs as well as the software and hardware that were used to make the idea happen. It also provides steps taken part in the study project's growth, “Email Spam Detection: Leveraging Fine-Tuned Transformer Models with Attention Mechanism”.

2 System Configuration

This is a detailed section that provide guideline for the use of software. It gives a general outline of the programme or application, including what it does, how it works, what hardware and software is needs, and how it communicated to its environment and users.

2.1 Hardware Requirements

2.1.1 The experiment was carried out on the following hardware:

- Processor: Intel(R) Core (TM) i5-10300H CPU @ 2.50GHz   2.50 GHz
- RAM: 16.0 GB
- System type: 64-bit operating system, x64-based processor
- Edition: Windows 11 Home Single Language

2.1.2 Minimum hardware requirements are:

- Modern Operating System:
  1. Windows 10 or 11
  2. Mac OS X 10.11 or higher 64-bit

2.2 Software Requirements

- Google Collaborator: cloud-based jupyter notebook, python version 3.10.
- Email: Gmail account needed for accessing the drive.
- Browser: Any web browser.
- Other Software: VS code, Word.

3 Project Recreation

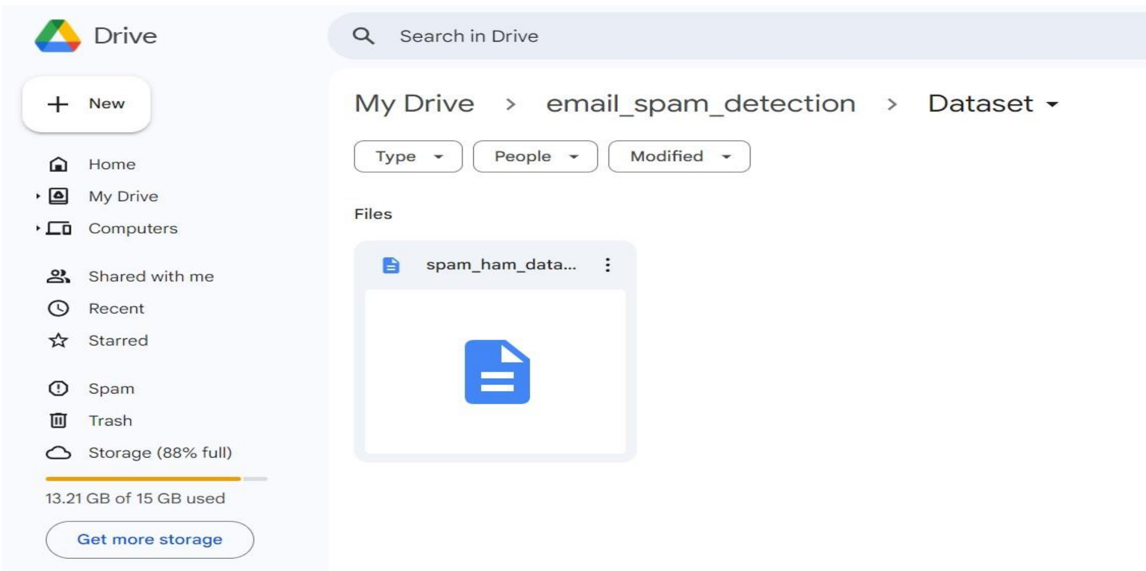
The aim for the challenge was to provide a safe approach for detecting email spam using deep learning models. We have mounted google drive with Collab.

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

     Mounted at /content/drive
```

3.1.1 Data Collection

This directory contains the Enron-Spam datasets, as described in the paper: V. Metsis, I. Androutsopoulos and G. Paliouras, "Spam Filtering with Naive Bayes - Which Naive Bayes?". Proceedings of the 3rd Conference on Email and Anti-Spam (CEAS 2006), Mountain View, CA, USA, 2006.

We have downloaded dataset from following URL and uploaded it on drive.
https://www2.aueb.gr/users/ion/data/enron-spam/



3.1.2 Installing Libraries

We have installed all the Necessary Libraries to our environment to run the project (Transformer, Kears- tuner and Word ninja).

```
[ ] !pip install transformers==4.31.0
    # Keras tuner
    !pip install -q -U keras-tuner
    !pip install wordninja

    Collecting transformers==4.31.0
      Downloading transformers-4.31.0-py3-none-any.whl (7.4 MB)
         ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 7.4/7.4 MB 23.1 MB/s eta 0:00:00
    Requirement already satisfied: filelock in /usr/local/lib/python3.10/dist-packages (from transformers==4.31.0) (3.13.1)
    Requirement already satisfied: huggingface-hub<1.0,>=0.14.1 in /usr/local/lib/python3.10/dist-packages (from transformers==4.31.0) (0.20.3)
    Requirement already satisfied: numpy>=1.17 in /usr/local/lib/python3.10/dist-packages (from transformers==4.31.0) (1.25.2)
    Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.10/dist-packages (from transformers==4.31.0) (24.0)
    Requirement already satisfied: pyyaml>=5.1 in /usr/local/lib/python3.10/dist-packages (from transformers==4.31.0) (6.0.1)
    Requirement already satisfied: regex!=2019.12.17 in /usr/local/lib/python3.10/dist-packages (from transformers==4.31.0) (2023.12.25)
    Requirement already satisfied: requests in /usr/local/lib/python3.10/dist-packages (from transformers==4.31.0) (2.31.0)
    Collecting tokenizers!=0.11.3,<0.14,>=0.11.1 (from transformers==4.31.0)
      Downloading tokenizers-0.13.3-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (7.8 MB)
         ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 7.8/7.8 MB 43.8 MB/s eta 0:00:00
    Requirement already satisfied: safetensors>=0.3.1 in /usr/local/lib/python3.10/dist-packages (from transformers==4.31.0) (0.4.2)
    Requirement already satisfied: tqdm>=4.27 in /usr/local/lib/python3.10/dist-packages (from transformers==4.31.0) (4.66.2)
    Requirement already satisfied: fsspec>=2023.5.0 in /usr/local/lib/python3.10/dist-packages (from huggingface-hub<1.0,>=0.14.1->transformers==4.31.0) (2023.6.0)
    Requirement already satisfied: typing-extensions>=3.7.4.3 in /usr/local/lib/python3.10/dist-packages (from huggingface-hub<1.0,>=0.14.1->transformers==4.31.0) (4.10.0)
    Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.10/dist-packages (from requests->transformers==4.31.0) (3.3.2)
    Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-packages (from requests->transformers==4.31.0) (3.6)
    Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.10/dist-packages (from requests->transformers==4.31.0) (2.0.7)
    Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.10/dist-packages (from requests->transformers==4.31.0) (2024.2.2)
    Installing collected packages: tokenizers, transformers
      Attempting uninstall: tokenizers
        Found existing installation: tokenizers 0.15.2
        Uninstalling tokenizers-0.15.2:
          Successfully uninstalled tokenizers-0.15.2
      Attempting uninstall: transformers
        Found existing installation: transformers 4.38.2
        Uninstalling transformers-4.38.2:
          Successfully uninstalled transformers-4.38.2
    Successfully installed tokenizers-0.13.3 transformers-4.31.0
         ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 129.1/129.1 kB 1.6 MB/s eta 0:00:00
    Collecting wordninja
      Downloading wordninja-2.0.0.tar.gz (541 kB)
         ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 541.6/541.6 kB 3.5 MB/s eta 0:00:00
      Preparing metadata (setup.py) ... done
```

3.1.3 Importing Libraries

We have imported all the libraries and listed in the figure 3.1.3. 1) matplotlib - this was used for data visualization and plotting of graphs 2) Pandas for data analysis and manipulation of data 3) Scikit learn is the machine learning library 4) Nltk Set of libraries for NLP tasks like tokenization, stemming, tagging, parsing 5) TensorFlow is Open-source library for numerical computation and large-scale machine learning 6) Keras is High-level API for building and training deep learning models on top of TensorFlow 7) word ninja is Python library for splitting text into words 8) pickle is Python module for serializing and de-serializing Python object structures (for saving/loading data).

```
[ ] import re
    import json
    import string
    import pickle
    import wordninja
    import math, nltk
    import numpy as np
    import pandas as pd
    import seaborn as sns
    from tqdm import tqdm
    import keras_tuner as kt
    import tensorflow as tf
    import keras
    import matplotlib.pyplot as plt
    from wordcloud import WordCloud
    from nltk.corpus import stopwords
    from sklearn import feature_extraction
    from transformers import AutoTokenizer
    from imblearn.over_sampling import SMOTE
    from sklearn.preprocessing import LabelBinarizer
    from sklearn.model_selection import train_test_split
    from keras.models import Sequential, model_from_json
    from tensorflow.keras.utils import to_categorical
    from sklearn.metrics import confusion_matrix,classification_report
    from transformers import TFDistilBertModel, DistilBertConfig, TFXLMRobertaModel, XLMRobertaConfig, TFRobertaModel, RobertaConfig
```

3.1.4 Importing Dependencies

We have downloaded Nltk dependencies for which we have used for data Cleaning in further steps.

```
nltk.download('all')
```

```
[nltk_data] Downloading collection 'all'
[nltk_data]    |
[nltk_data]    | Downloading package abc to /root/nltk_data...
[nltk_data]    |   Unzipping corpora/abc.zip.
[nltk_data]    | Downloading package alpino to /root/nltk_data...
[nltk_data]    |   Unzipping corpora/alpino.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 averaged_perceptron_tagger_ru to
[nltk_data]    |     /root/nltk_data...
[nltk_data]    |   Unzipping
[nltk_data]    |       taggers/averaged_perceptron_tagger_ru.zip.
[nltk_data]    | Downloading package basque_grammars to
[nltk_data]    |     /root/nltk_data...
[nltk_data]    |   Unzipping grammars/basque_grammars.zip.
[nltk_data]    | Downloading package bcp47 to /root/nltk_data...
[nltk_data]    | Downloading package biocreative_ppi to
[nltk_data]    |     /root/nltk_data...
[nltk_data]    |   Unzipping corpora/biocreative_ppi.zip.
[nltk_data]    | Downloading package bllip_wsj_no_aux to
[nltk_data]    |     /root/nltk_data...
[nltk_data]    |   Unzipping models/bllip_wsj_no_aux.zip.
[nltk_data]    | Downloading package book_grammars to
[nltk_data]    |     /root/nltk_data...
[nltk_data]    |   Unzipping grammars/book_grammars.zip.
[nltk_data]    | Downloading package brown to /root/nltk_data...
[nltk_data]    |   Unzipping corpora/brown.zip.
[nltk_data]    | Downloading package brown_tei to /root/nltk_data...
[nltk_data]    |   Unzipping corpora/brown_tei.zip.
[nltk_data]    | Downloading package cess_cat to /root/nltk_data...
[nltk_data]    |   Unzipping corpora/cess_cat.zip.
```

3.1.5 Data Loading

We have pulled the data from drive to the environment to work on it.

```
[ ] dataframe = pd.read_csv('/content/drive/MyDrive/email_spam_detection/Dataset/spam_ham_dataset.csv')
    dataframe.head()
```

	Unnamed: 0	label	text	label_num
0	605	ham	Subject: enron methanol ; meter # : 988291\r\n...	0
1	2349	ham	Subject: hpl nom for january 9 , 2001\r\n(see...	0
2	3624	ham	Subject: neon retreat\r\nho ho ho , we ' re ar...	0
3	4685	spam	Subject: photoshop , windows , office . cheap ...	1
4	2030	ham	Subject: re : indian springs\r\nthis deal is t...	0

```
[ ] dataframe.shape
```

```
(5171, 4)
```

3.1.6 Data Cleaning

We first convert the tweets CSV file into a Pandas data frame, Following are the steps we will perform for the preprocessing the data using the NLTK:

1. Remove HTML entities.
2. Substitute @mentions, URLs, etc. with whitespace using regular expressions.
3. Substitute any non-alphabetic whitespace.
4. All the words in lower case.
5. Removing stop words.
6. Punctuations.

```
[ ] dataframe = dataframe.drop('Unnamed: 0',axis='columns')
```

```
[ ] dataframe.info()

    <class 'pandas.core.frame.DataFrame'>
    RangeIndex: 5171 entries, 0 to 5170
    Data columns (total 3 columns):
     #   Column     Non-Null Count  Dtype
    ---  ------     --------------  -----
     0   label      5171 non-null   object
     1   text       5171 non-null   object
     2   label_num  5171 non-null   int64
    dtypes: int64(1), object(2)
    memory usage: 121.3+ KB
```

```
#creating text cleaning function
def html_references(tweets):
    texts = tweets
    # remove url - references to websites
    url_remove  = r'http[s]?://(?:[a-zA-Z]|[0-9]|[$-_@.&+]|[!*\(\),]|(?:%[0-9a-fA-F][0-9a-fA-F]))+'
    texts  = re.sub(url_remove, '', texts)
    # remove common html entity references in utf-8 as '&lt;', '&gt;', '&amp;'
    entities_remove = r'&amp;|&gt;|&lt'
    texts = re.sub(entities_remove, "", texts)
    # split into words by white space
    words = texts.split()
    #convert to lower case
    words = [word.lower() for word in words]
    return " ".join(words)

def decontraction(tweet):
    # specific
    tweet = re.sub(r"won\'t", " will not", tweet)
    tweet = re.sub(r"won\'t've", " will not have", tweet)
    tweet = re.sub(r"can\'t", " can not", tweet)
    tweet = re.sub(r"don\'t", " do not", tweet)

    tweet = re.sub(r"can\'t've", " can not have", tweet)
    tweet = re.sub(r"ma\'am", " madam", tweet)
    tweet = re.sub(r"let\'s", " let us", tweet)
    tweet = re.sub(r"ain\'t", " am not", tweet)
    tweet = re.sub(r"shan\'t", " shall not", tweet)
    tweet = re.sub(r"sha\n't", " shall not", tweet)
    tweet = re.sub(r"o\'clock", " of the clock", tweet)
    tweet = re.sub(r"y\'all", " you all", tweet)
    # general
    tweet = re.sub(r"n\'t", " not", tweet)
    tweet = re.sub(r"n\'t've", " not have", tweet)
    tweet = re.sub(r"\'re", " are", tweet)
    tweet = re.sub(r"\'s", " is", tweet)
    tweet = re.sub(r"\'d", " would", tweet)
    tweet = re.sub(r"\'d've", " would have", tweet)
    tweet = re.sub(r"\'ll", " will", tweet)
    tweet = re.sub(r"\'ll've", " will have", tweet)
    tweet = re.sub(r"\'t", " not", tweet)
    tweet = re.sub(r"\'ve", " have", tweet)
    tweet = re.sub(r"\'m", " am", tweet)
    tweet = re.sub(r"\'re", " are", tweet)
    return tweet

def filter_punctuations_etc(tweets):
    words = tweets.split()
    # prepare regex for char filtering
    re_punc = re.compile( '[%s]' % re.escape(string.punctuation))
    # remove punctuation from each word
    words = [re_punc.sub('', w) for w in words]
    # filter out non-printable characters
    re_print = re.compile( '[^%s]' % re.escape(string.printable))
    words = [re_print.sub(' ', w) for w in words]
    return " ".join(words)

def separate_alphanumeric(tweets):
    words = tweets
    # separate alphanumeric
    words = re.findall(r"[^\W\d_]+|\d+", words)
    return " ".join(words)

def cont_rep_char(text):
```

```
[ ] #text cleaning
    dataframe['clean_text'] = dataframe['text'].apply(lambda x : html_references(x))
    dataframe['clean_text'] = dataframe['clean_text'].apply(lambda x : decontraction(x))
    dataframe['clean_text'] = dataframe['clean_text'].apply(lambda x : filter_punctuations_etc(x))
    dataframe['clean_text'] = dataframe['clean_text'].apply(lambda x : separate_alphanumeric(x))
    dataframe['clean_text'] = dataframe['clean_text'].apply(lambda x : unique_char(cont_rep_char, x))
    dataframe['clean_text'] = dataframe['clean_text'].apply(lambda x : split_attached_words(x))
    dataframe['clean_text'] = dataframe['clean_text'].apply(lambda x : stopwords_shortwords(x))
```

```
[ ] dataframe['clean_text'][3]

    'subject photoshop windows office cheap main trending basements dare prudently fortuitous undergone lighthearted charm orinoco taster railroad affluent p
    rophyll robed diagrammatic fog arty clears bay da inconveniencing managing represented smartness hashish academies shareholders unload badness danielson
```

```
[ ] dataframe = dataframe[['clean_text','label']]
    dataframe.head(5)
```

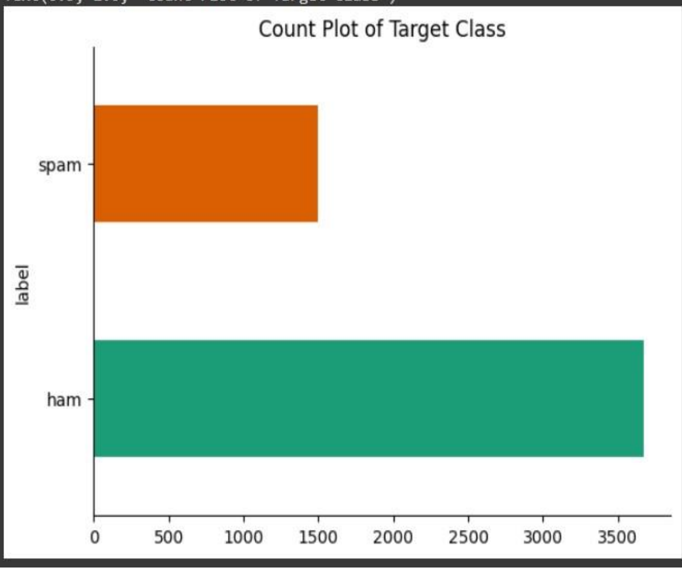
	clean_text	label
0	subject enron methanol meter 988291 follow not...	ham
1	subject hp nom january 2001 see attached file ...	ham
2	subject neon retreat ho ho ho around wonderful...	ham
3	subject photoshop windows office cheap main tr...	spam
4	subject indian springs deal book te co pv reve...	ham

3.1.7 Data Visualisation

After data preprocessing, we have generated a Count Plot graph we mentioned segregated number of Spam & Ham data present in our dataset.

```
dataframe.groupby('label').size().plot(kind='barh', color=sns.palettes.mpl_palette('Dark2'))
plt.gca().spines[['top', 'right',]].set_visible(False)
plt.title('Count Plot of Target Class')
```

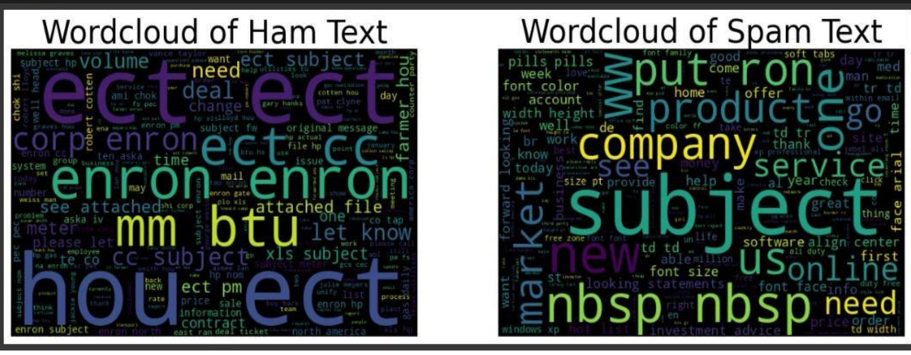
```
Text(0.5, 1.0, 'Count Plot of Target Class')
```



We have plotted Word cloud which mention words frequency in each Class (Ham & Spam).

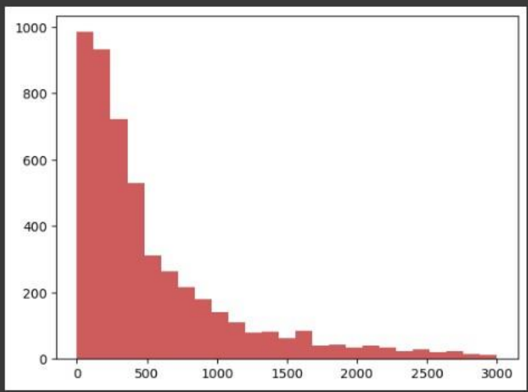
```
#Wordcloud
class1 = dataframe[dataframe['label']=='ham']['clean_text']
class2 = dataframe[dataframe['label']=='spam']['clean_text']
```

```
#---wordcloudplot
fig, (ax1, ax2) = plt.subplots(1, 2, figsize=[16, 8])
wordcloud1 = WordCloud( background_color='black',
                        width=600,
                        height=400).generate(" ".join(class1))
ax1.imshow(wordcloud1)
ax1.axis('off')
ax1.set_title('Wordcloud of Ham Text',fontsize=30);
#---
wordcloud2 = WordCloud( background_color='black',
                        width=600,
                        height=400).generate(" ".join(class2))
ax2.imshow(wordcloud2)
ax2.axis('off')
ax2.set_title('Wordcloud of Spam Text',fontsize=30);
plt.show()
```



Below graph shows text length in each data (Email).

```
[ ] #value count of text lenght
    lengthsoftext = []
    for clean_text in dataframe['clean_text']:
     lengthsoftext.append(len(clean_text))
    plt.hist(lengthsoftext, bins=25, color='indianred',range=(0,3000))
    plt.show()
```



3.1.8 Data Preprocessing

We have converted Ham & Spam Classes into numbers.

```
[ ] #converting categorical data into number
    class_labels = LabelBinarizer()
    targetlabel = class_labels.fit_transform(dataframe['label'])
    pickle.dump(class_labels,open('/content/drive/MyDrive/email_spam_detection/label_transform.pkl', 'wb'))
```

```
[ ] cls = len(class_labels.classes_)
    class_labels.classes_

    array(['ham', 'spam'], dtype='<U4')
```

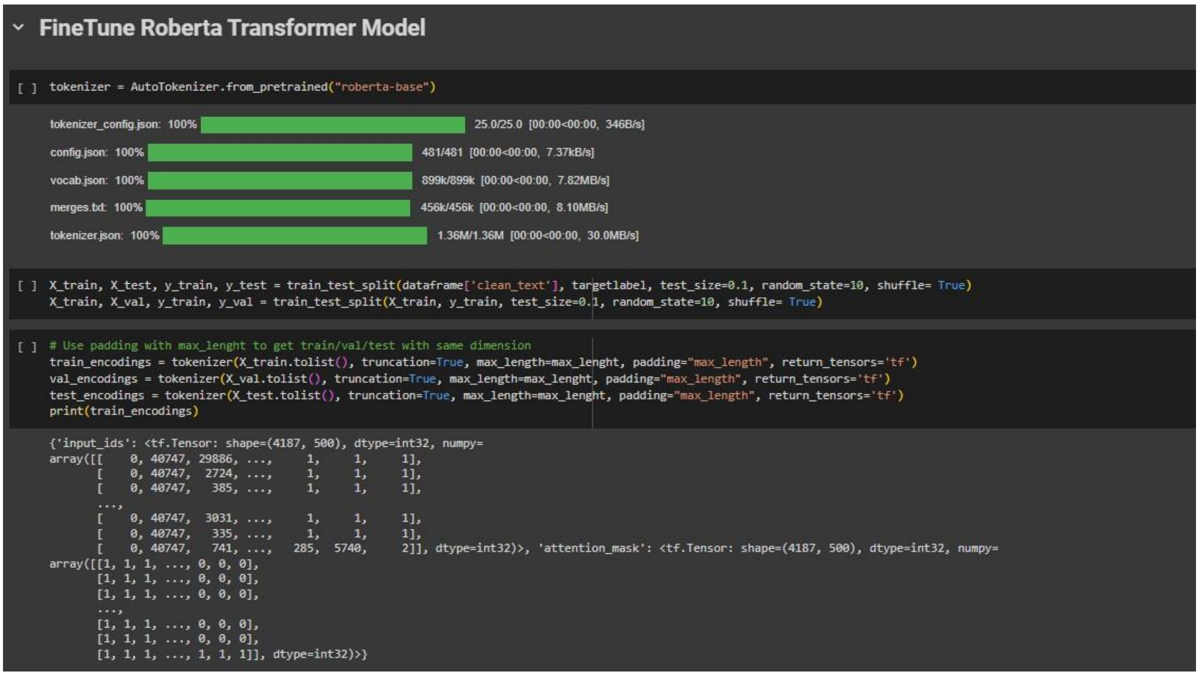
```
[ ] max_lenght=500
    BATCH_SIZE = 64
```

```
[ ] targetlabel = to_categorical(targetlabel)
```

4 Model Implementation

4.1.1 Fine tune Roberta model

We have loaded the Tokenizer on Fine tune Roberta model along with that we have split the data in a ratio of 80:10:10 For Training, validation, and Testing.



We have also applied Tokenizer to convert text values from data to into numerical values and Vector as shown in below figure.

```python
# Configure RobertaConfig's initialization
config = RobertaConfig(dropout=0.2,
                          attention_dropout=0.2,
                          output_hidden_states=True)
```

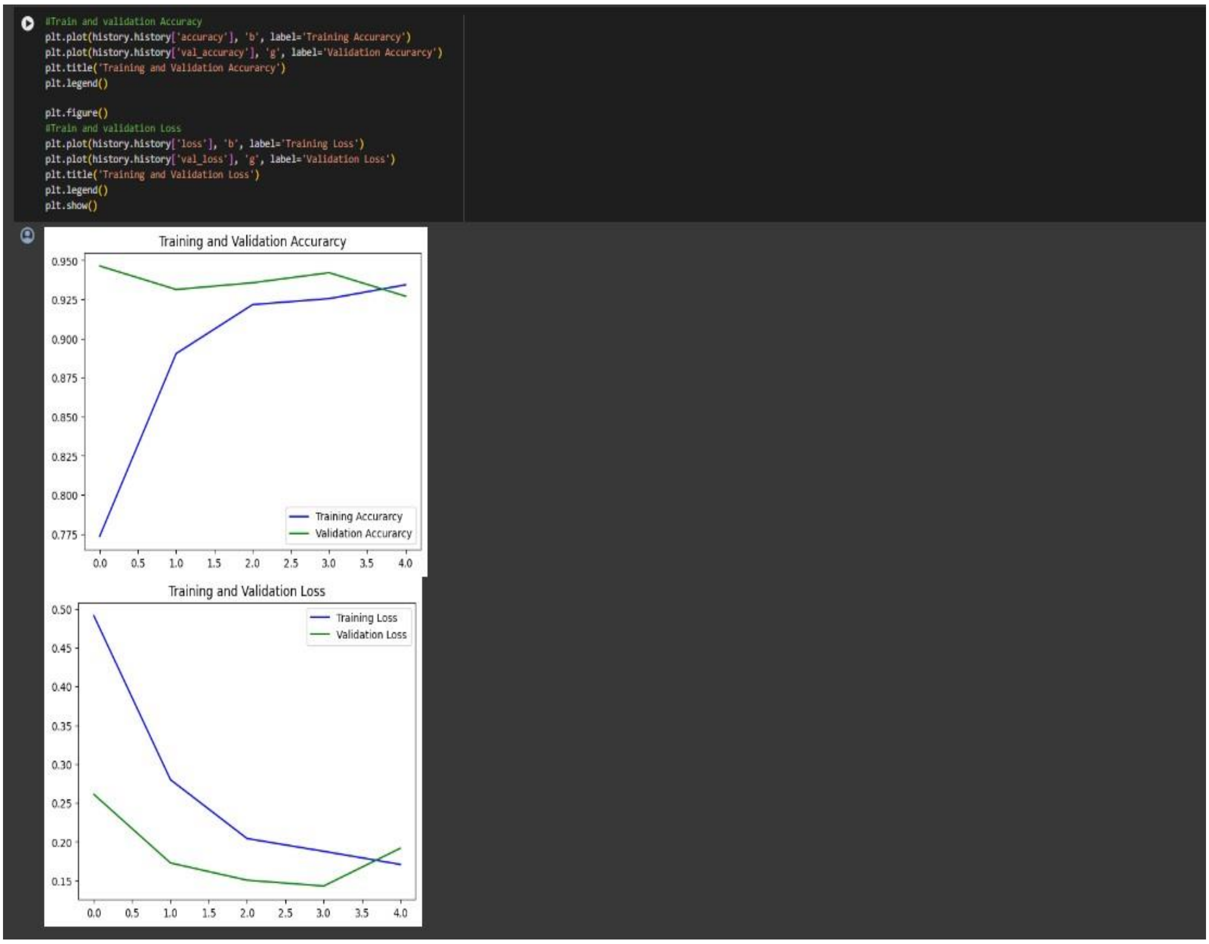
```python
# Model function
def create_model(transformer):

    # Make Transformer layers untrainable
    for layer in transformer.layers:
        layer.trainable = False
    # Input layers
    input_ids_layer = keras.Input(shape =(max_lenght,),
                           dtype=tf.int32,
                           name='input_ids')
    input_attention_layer = keras.Input(shape=(max_lenght,),
                                   dtype=tf.int32,
                                   name='attention_mask')

    #  outputs a tuple where the first element at index 0
    # represents the hidden-state at the output of the model's last layer.
    # It is a tf.Tensor of shape (batch_size, sequence_length, hidden_size=768).
    last_hidden_state = transformer([input_ids_layer, input_attention_layer])[0]

    # We only care about 's output for the [CLS] token,
    # which is located at index 0 of every encoded sequence.
    # Splicing out the [CLS] tokens gives us 2D data.
    cls_token = last_hidden_state[:, 0, :]
    # Hidden layers
    output = keras.layers.Dense(256,
                            kernel_initializer=keras.initializers.GlorotUniform(seed=1),
                            kernel_constraint=None,
                            bias_initializer='zeros',
                            activation='relu')(cls_token)
    output = keras.layers.Dropout(0.2)(output)
    output = keras.layers.Dense(128, activation = 'relu')(output)
    # Output layer
    output = keras.layers.Dense(cls, activation='sigmoid')(output)
    # Define the model
    model = keras.Model([input_ids_layer, input_attention_layer],
                   output)
    model.summary()
    keras.utils.plot_model(model)

    return model
```

We have loaded weights of pretrained model, and we complied the model for 5 Epoch for training the model to get better accuracy.

```python
Roberta_model = TFRobertaModel.from_pretrained('roberta-base')
model = create_model(Roberta_model)
# Compile the model
model.compile(optimizer="rmsprop", loss="categorical_crossentropy", metrics=['accuracy'])
#Training
history = model.fit(train_tf_dataset, epochs=5, batch_size=BATCH_SIZE, validation_data=val_tf_dataset,verbose=1)
```

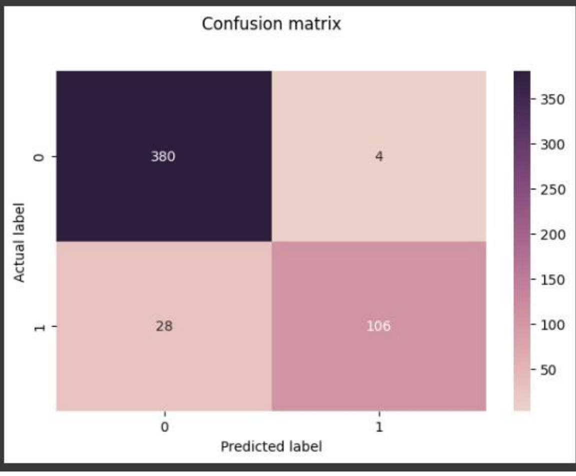
```
model.safetensors: 100% 499M/499M [00:02<00:00, 190MB/s]
Some weights of the PyTorch model were not used when initializing the TF 2.0 model TFRobertaModel: ['lm_head.layer_norm.weight', 'lm_head.layer_norm.bias', 'lm_head.bias', 'lm_head.dense.weight', 'lm_head.dense.bias', 'roberta.embeddings.position_ids']
- This IS expected if you are initializing TFRobertaModel from a PyTorch model trained on another task or with another architecture (e.g. initializing a TFBertForSequenceClassification model from a BertForPreTraining model).
- This IS NOT expected if you are initializing TFRobertaModel from a PyTorch model that you expect to be exactly identical (e.g. initializing a TFBertForSequenceClassification model from a BertForSequenceClassification model).
Some weights or buffers of the TF 2.0 model TFRobertaModel were not initialized from the PyTorch model and are newly initialized: ['roberta.pooler.dense.weight', 'roberta.pooler.dense.bias']
You should probably TRAIN this model on a down-stream task to be able to use it for predictions and inference.
Model: "model"
__________________________________________________________________________________________________
 Layer (type)                Output Shape                 Param #   Connected to
==================================================================================================
 input_ids (InputLayer)      [(None, 500)]                0         []

 attention_mask (InputLayer  [(None, 500)]                0         []
 )

 tf_roberta_model (TFRobert  TFBaseModelOutputWithPooli   1246456   ['input_ids[0][0]',
 aModel)                     ngAndCrossAttentions(last_   32         'attention_mask[0][0]']
                             hidden_state=(None, 500, 7
                             68),
                              pooler_output=(None, 768)
                             , past_key_values=None, hi
                             dden_states=None, attentio
                             ns=None, cross_attentions=
                             None)

 tf.__operators__.getitem (  (None, 768)                  0         ['tf_roberta_model[0][0]']
 SlicingOpLambda)

 dense (Dense)               (None, 256)                  196864    ['tf.__operators__.getitem[0][
                                                                    0]']

 dropout_37 (Dropout)        (None, 256)                  0         ['dense[0][0]']

 dense_1 (Dense)             (None, 128)                  32896     ['dropout_37[0][0]']

 dense_2 (Dense)             (None, 2)                    258       ['dense_1[0][0]']

==================================================================================================
Total params: 124875650 (476.36 MB)
Trainable params: 230018 (898.51 KB)
Non-trainable params: 124645632 (475.49 MB)
__________________________________________________________________________________________________
Epoch 1/5
66/66 [==============================] - 208s 3s/step - loss: 0.4910 - accuracy: 0.7736 - val_loss: 0.2610 - val_accuracy: 0.9464
Epoch 2/5
66/66 [==============================] - 193s 3s/step - loss: 0.2799 - accuracy: 0.8904 - val_loss: 0.1727 - val_accuracy: 0.9313
Epoch 3/5
66/66 [==============================] - 193s 3s/step - loss: 0.2041 - accuracy: 0.9217 - val_loss: 0.1505 - val_accuracy: 0.9356
Epoch 4/5
66/66 [==============================] - 197s 3s/step - loss: 0.1877 - accuracy: 0.9255 - val_loss: 0.1430 - val_accuracy: 0.9421
Epoch 5/5
66/66 [==============================] - 193s 3s/step - loss: 0.1708 - accuracy: 0.9343 - val_loss: 0.1916 - val_accuracy: 0.9270
```

We have generated below graph on basis of Every Epoch values.



Below generated Confusion Matrix shows Comparison of predicted output value and Actual Value of Test data.

```
[ ] #confusion Matrix
    plt.figure(figsize=(6,4))
    matrix =confusion_matrix(y_test_new, predictions)
    class_names=[0,1]
    tick_marks = np.arange(len(class_names))
    plt.xticks(tick_marks, class_names)
    plt.yticks(tick_marks, class_names)
    sns.heatmap(pd.DataFrame(matrix), annot=True, cmap=sns.cubehelix_palette(as_cmap=True), fmt='g')
    plt.tight_layout()
    plt.title('Confusion matrix', y=1.1)
    plt.ylabel('Actual label')
    plt.xlabel('Predicted label')
    plt.show()
```



Evaluation metrics results.

```
[ ] #classification report
    print(classification_report(y_test_new, predictions))

              precision    recall  f1-score   support

           0       0.93      0.99      0.96       384
           1       0.96      0.79      0.87       134

    accuracy                           0.94       518
   macro avg       0.95      0.89      0.91       518
weighted avg       0.94      0.94      0.94       518
```

Similar steps are performed for other two Transformer models (Fine-tuned XLMRoBERTa & DistilBERT).

4.1.2 Fine-tuned XLM-RoBERTa Model

FineTune XLMRoberta Transformer Model

```
tokenizer = AutoTokenizer.from_pretrained("xlm-roberta-base")
```

```
tokenizer_config.json: 100% 25.0/25.0 [00:00<00:00, 825B/s]
config.json: 100% 615/615 [00:00<00:00, 26.7kB/s]
sentencepiece.bpe.model: 100% 5.07M/5.07M [00:00<00:00, 22.7MB/s]
tokenizer.json: 100% 9.10M/9.10M [00:00<00:00, 39.1MB/s]
```

```
X_train, X_test, y_train, y_test = train_test_split(dataframe['clean_text'], targetlabel, test_size=0.1, random_state=10, shuffle= True)
X_train, X_val, y_train, y_val = train_test_split(X_train, y_train, test_size=0.1, random_state=10, shuffle= True)
```

```
# Use padding with max_lenght to get train/val/test with same dimension
train_encodings = tokenizer(X_train.tolist(), truncation=True, max_length=max_lenght, padding="max_length", return_tensors='tf')
val_encodings = tokenizer(X_val.tolist(), truncation=True, max_length=max_lenght, padding="max_length", return_tensors='tf')
test_encodings = tokenizer(X_test.tolist(), truncation=True, max_length=max_lenght, padding="max_length", return_tensors='tf')
print(train_encodings)
```

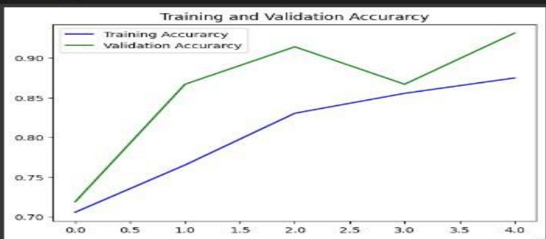
```
{'input_ids': <tf.Tensor: shape=(4187, 500), dtype=int32, numpy=
array([[     0,  28368, 118258, ...,      1,      1,      1],
       [     0,  28368,   1492, ...,      1,      1,      1],
       [     0,  28368,   5080, ...,      1,      1,      1],
       ...,
       [     0,  28368,   8561, ...,      1,      1,      1],
       [     0,  28368,   4677, ...,      1,      1,      1],
       [     0,  28368, 111467, ...,  11034,  60212,      2]], dtype=int32)>, 'attention_mask': <tf.Tensor: shape=(4187, 500), dtype=int32, numpy=
array([[1, 1, 1, ..., 0, 0, 0],
       [1, 1, 1, ..., 0, 0, 0],
```

```
# Configure XLMRobertaConfig's initialization
config = XLMRobertaConfig(dropout=0.2,
                          attention_dropout=0.2,
                          output_hidden_states=True)
```

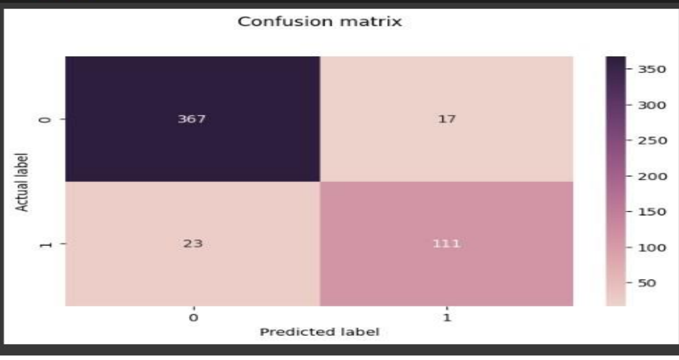
```
# Model function
def create_model(transformer):

    # Make Transformer layers untrainable
    for layer in transformer.layers:
        layer.trainable = False
    # Input layers
    input_ids_layer = keras.Input(shape =(max_lenght,),
                                  dtype=tf.int32,
                                  name='input_ids')
    input_attention_layer = keras.Input(shape=(max_lenght,),
                                        dtype=tf.int32,
                                        name='attention_mask')

    #  outputs a tuple where the first element at index 0
    # represents the hidden-state at the output of the model's last layer.
    # It is a tf.Tensor of shape (batch_size, sequence_length, hidden_size=768).
    last_hidden_state = transformer([input_ids_layer, input_attention_layer])[0]

    # We only care about 's output for the [CLS] token,
    # which is located at index 0 of every encoded sequence.
    # Splicing out the [CLS] tokens gives us 2D data.
    cls_token = last_hidden_state[:, 0, :]
    # Hidden layers
    output = keras.layers.Dense(256,
                                kernel_initializer=keras.initializers.GlorotUniform(seed=1),
                                kernel_constraint=None,
                                bias_initializer='zeros',
                                activation='relu')(cls_token)
    output = keras.layers.Dropout(0.2)(output)
    output = keras.layers.Dense(128, activation = 'relu')(output)
    # Output layer
    output = keras.layers.Dense(cls, activation='sigmoid')(output)
    # Define the model
    model = keras.Model([input_ids_layer, input_attention_layer],
                        output)
    model.summary()
    keras.utils.plot_model(model)

    return model
```

```
model.safetensors: 100% 1.12G/1.12G [00:11<00:00, 157MB/s]
Some weights of the PyTorch model were not used when initializing the TF 2.0 model TFXLMRobertaModel: ['lm_head.layer_norm.weight', 'lm_head.layer_norm.bias', 'lm_head.bias', 'lm_head.dense.weight', 'lm_head.dense.bias']
- This IS expected if you are initializing TFXLMRobertaModel from a PyTorch model trained on another task or with another architecture (e.g. initializing a TFBertForSequenceClassification model from a BertForPreTraining model).
- This IS NOT expected if you are initializing TFXLMRobertaModel from a PyTorch model that you expect to be exactly identical (e.g. initializing a TFBertForSequenceClassification model from a BertForSequenceClassification model).
All the weights of TFXLMRobertaModel were initialized from the PyTorch model.
If your task is similar to the task the model of the checkpoint was trained on, you can already use TFXLMRobertaModel for predictions without further training.
Model: "model_1"
__________________________________________________________________________________________________
 Layer (type)                Output Shape                 Param #   Connected to
==================================================================================================
 input_ids (InputLayer)      [(None, 500)]                0         []

 attention_mask (InputLayer  [(None, 500)]                0         []
 )

 tfxlm_roberta_model (TFXLM  TFBaseModelOutputWithPooli   2780436   ['input_ids[0][0]',
 RobertaModel)               ngAndCrossAttentions(last_   48         'attention_mask[0][0]']
                             hidden_state=(None, 500, 7
                             68),
                              pooler_output=(None, 768)
                             , past_key_values=None, hi
                             dden_states=None, attentio
                             ns=None, cross_attentions=
                             None)

 tf.__operators__.getitem_1  (None, 768)                  0         ['tfxlm_roberta_model[0][0]']
  (SlicingOpLambda)

 dense_3 (Dense)             (None, 256)                  196864    ['tf.__operators__.getitem_1[0
                                                                    ][0]']

 dropout_75 (Dropout)        (None, 256)                  0         ['dense_3[0][0]']

 dense_4 (Dense)             (None, 128)                  32896     ['dropout_75[0][0]']

 dense_5 (Dense)             (None, 2)                    258       ['dense_4[0][0]']

==================================================================================================
Total params: 278273666 (1.04 GB)
Trainable params: 230018 (898.51 KB)
Non-trainable params: 278043648 (1.04 GB)
__________________________________________________________________________________________________
Epoch 1/5
66/66 [==============================] - 209s 3s/step - loss: 0.6053 - accuracy: 0.7058 - val_loss: 0.5131 - val_accuracy: 0.7189
Epoch 2/5
66/66 [==============================] - 193s 3s/step - loss: 0.4907 - accuracy: 0.7652 - val_loss: 0.3268 - val_accuracy: 0.8670
Epoch 3/5
66/66 [==============================] - 193s 3s/step - loss: 0.3885 - accuracy: 0.8304 - val_loss: 0.2421 - val_accuracy: 0.9142
Epoch 4/5
66/66 [==============================] - 193s 3s/step - loss: 0.3308 - accuracy: 0.8555 - val_loss: 0.2919 - val_accuracy: 0.8670
Epoch 5/5
66/66 [==============================] - 193s 3s/step - loss: 0.2845 - accuracy: 0.8749 - val_loss: 0.1791 - val_accuracy: 0.9313
```

```python
#Train and validation Accuracy
plt.plot(history.history['accuracy'], 'b', label='Training Accurarcy')
plt.plot(history.history['val_accuracy'], 'g', label='Validation Accurarcy')
plt.title('Training and Validation Accurarcy')
plt.legend()

plt.figure()
#Train and validation Loss
plt.plot(history.history['loss'], 'b', label='Training Loss')
plt.plot(history.history['val_loss'], 'g', label='Validation Loss')
plt.title('Training and Validation Loss')
plt.legend()
plt.show()
```




```python
#confusion Matrix
plt.figure(figsize=(6,4))
matrix =confusion_matrix(y_test_new, predictions)
class_names=[0,1]
tick_marks = np.arange(len(class_names))
plt.xticks(tick_marks, class_names)
plt.yticks(tick_marks, class_names)
sns.heatmap(pd.DataFrame(matrix), annot=True, cmap=sns.cubehelix_palette(as_cmap=True), fmt='g')
plt.tight_layout()
plt.title('Confusion matrix', y=1.1)
plt.ylabel('Actual label')
plt.xlabel('Predicted label')
plt.show()
```



```
[ ] #classification report
    print(classification_report(y_test_new, predictions))

              precision    recall  f1-score   support

           0       0.94      0.96      0.95       384
           1       0.87      0.83      0.85       134

    accuracy                           0.92       518
   macro avg       0.90      0.89      0.90       518
weighted avg       0.92      0.92      0.92       518
```

4.1.3 Fine-tuned Distilbert Model

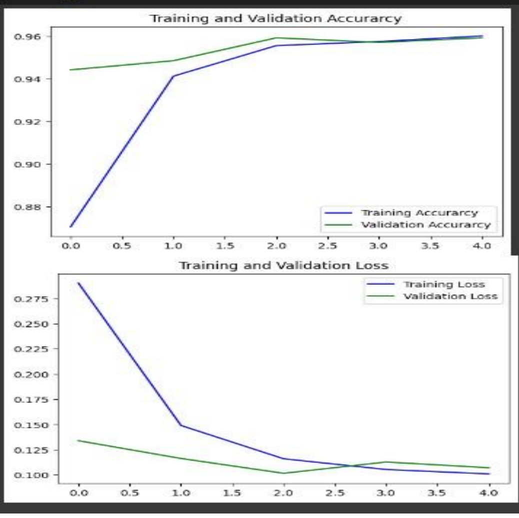
FineTune Distilbert Transformer Model

```
[ ] tokenizer = AutoTokenizer.from_pretrained("distilbert-base-uncased")

    tokenizer_config.json: 100% 28.0/28.0 [00:00<00:00, 2.10kB/s]
    config.json: 100% 483/483 [00:00<00:00, 35.8kB/s]
    vocab.txt: 100% 232k/232k [00:00<00:00, 4.36MB/s]
    tokenizer.json: 100% 466k/466k [00:00<00:00, 3.84MB/s]

[ ] X_train, X_test, y_train, y_test = train_test_split(dataframe['clean_text'], targetlabel, test_size=0.1, random_state=10, shuffle= True)
    X_train, X_val, y_train, y_val = train_test_split(X_train, y_train, test_size=0.1, random_state=10, shuffle= True)

[ ] # Use padding with max_len to get train/val/test with same dimension
    train_encodings = tokenizer(X_train.tolist(), truncation=True, max_length=max_lenght, padding="max_length", return_tensors='tf')
    val_encodings = tokenizer(X_val.tolist(), truncation=True, max_length=max_lenght, padding="max_length", return_tensors='tf')
    test_encodings = tokenizer(X_test.tolist(), truncation=True, max_length=max_lenght, padding="max_length", return_tensors='tf')
    print(train_encodings)

    {'input_ids': <tf.Tensor: shape=(4187, 500), dtype=int32, numpy=
    array([[  101,  3395,  6522, ...,     0,     0,     0],
           [  101,  3395,  2702, ...,     0,     0,     0],
           [  101,  3395,  3804, ...,     0,     0,     0],
           ...,
           [  101,  3395,  5025, ...,     0,     0,     0],
           [  101,  3395,  2592, ...,     0,     0,     0],
           [  101,  3395, 17531, ...,  6726,  3446,   102]], dtype=int32)>, 'attention_mask': <tf.Tensor: shape=(4187, 500), dtype=int32, numpy=
    array([[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, ..., 0, 0, 0],
           [1, 1, 1, ..., 1, 1, 1]], dtype=int32)>}
```

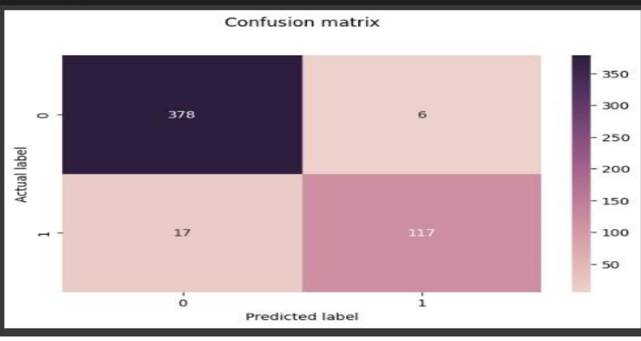
```
# Configure DistilBERT's initialization
config = DistilBertConfig(dropout=BERT_DROPOUT,
                          attention_dropout=BERT_ATT_DROPOUT,
                          output_hidden_states=True)

[ ] # Model function
    def create_model(transformer):

        # Make Transformer layers untrainable
        for layer in transformer.layers:
            layer.trainable = False
        # Input layers
        input_ids_layer = keras.Input(shape =(max_lenght,),
                                      dtype=tf.int32,
                                      name='input_ids')
        input_attention_layer = keras.Input(shape=(max_lenght,),
                                            dtype=tf.int32,
                                            name='attention_mask')

        # DistilBERT outputs a tuple where the first element at index 0
        # represents the hidden-state at the output of the model's last layer.
        # It is a tf.Tensor of shape (batch_size, sequence_length, hidden_size=768).
        last_hidden_state = transformer([input_ids_layer, input_attention_layer])[0]

        # We only care about DistilBERT's output for the [CLS] token,
        # which is located at index 0 of every encoded sequence.
        # Splicing out the [CLS] tokens gives us 2D data.
        cls_token = last_hidden_state[:, 0, :]
        # Hidden layers
        output = keras.layers.Dense(128,
                                    kernel_initializer=keras.initializers.GlorotUniform(seed=1),
                                    kernel_constraint=None,
                                    bias_initializer='zeros',
                                    activation='relu')(cls_token)
        output = keras.layers.Dropout(0.2)(output)
        output = keras.layers.Dense(64, activation = 'relu')(output)
        # Output layer
        output = keras.layers.Dense(cls, activation='sigmoid')(output)
```

```
model.safetensors: 100% 268M/268M [00:02<00:00, 58.4MB/s]
Some weights of the PyTorch model were not used when initializing the TF 2.0 model TFDistilBertModel: ['vocab_transform.bias', 'vocab_projector.bias', 'vocab_transform.weight', 'vocab_layer_norm.bias', 'vocab_layer_norm.weight']
- This IS expected if you are initializing TFDistilBertModel from a PyTorch model trained on another task or with another architecture (e.g. initializing a TFBertForSequenceClassification model from a BertForPreTraining model).
- This IS NOT expected if you are initializing TFDistilBertModel from a PyTorch model that you expect to be exactly identical (e.g. initializing a TFBertForSequenceClassification model from a BertForSequenceClassification model).
All the weights of TFDistilBertModel were initialized from the PyTorch model.
If your task is similar to the task the model of the checkpoint was trained on, you can already use TFDistilBertModel for predictions without further training.
Model: "model_2"
______________________________________________________________________________________
 Layer (type)                Output Shape                 Param #   Connected to
======================================================================================
 input_ids (InputLayer)      [(None, 500)]                0         []

 attention_mask (InputLayer  [(None, 500)]                0         []
 )

 tf_distil_bert_model (TFDi  TFBaseModelOutput(last_hid   6636288   ['input_ids[0][0]',
 stilBertModel)              den_state=(None, 500, 768)   0          'attention_mask[0][0]']
                             , hidden_states=((None, 50
                             0, 768),
                              (None, 500, 768),
                              (None, 500, 768),
                              (None, 500, 768),
                              (None, 500, 768),
                              (None, 500, 768),
                              (None, 500, 768)),
                              attentions=None)

 tf.__operators__.getitem_2  (None, 768)                  0         ['tf_distil_bert_model[0][7]']
  (SlicingOpLambda)

 dense_6 (Dense)             (None, 128)                  98432     ['tf.__operators__.getitem_2[0
                                                                    ][0]']

 dropout_95 (Dropout)        (None, 128)                  0         ['dense_6[0][0]']

 dense_7 (Dense)             (None, 64)                   8256      ['dropout_95[0][0]']

 dense_8 (Dense)             (None, 2)                    130       ['dense_7[0][0]']

======================================================================================
Total params: 66469698 (253.56 MB)
Trainable params: 106818 (417.26 KB)
Non-trainable params: 66362880 (253.15 MB)
______________________________________________________________________________________
Epoch 1/5
66/66 [==============================] - 104s 1s/step - loss: 0.2902 - accuracy: 0.8706 - val_loss: 0.1339 - val_accuracy: 0.9442
Epoch 2/5
66/66 [==============================] - 95s 1s/step - loss: 0.1490 - accuracy: 0.9412 - val_loss: 0.1163 - val_accuracy: 0.9485
Epoch 3/5
66/66 [==============================] - 94s 1s/step - loss: 0.1160 - accuracy: 0.9556 - val_loss: 0.1016 - val_accuracy: 0.9592
Epoch 4/5
66/66 [==============================] - 93s 1s/step - loss: 0.1053 - accuracy: 0.9575 - val_loss: 0.1128 - val_accuracy: 0.9571
Epoch 5/5
66/66 [==============================] - 95s 1s/step - loss: 0.1010 - accuracy: 0.9601 - val_loss: 0.1070 - val_accuracy: 0.9592
```

```
#Train and validation Accuracy
plt.plot(history.history['accuracy'], 'b', label='Training Accurarcy')
plt.plot(history.history['val_accuracy'], 'g', label='Validation Accurarcy')
plt.title('Training and Validation Accurarcy')
plt.legend()

plt.figure()
#Train and validation Loss
plt.plot(history.history['loss'], 'b', label='Training Loss')
plt.plot(history.history['val_loss'], 'g', label='Validation Loss')
plt.title('Training and Validation Loss')
plt.legend()
plt.show()
```




```
#confusion Matrix
plt.figure(figsize=(6,4))
matrix =confusion_matrix(y_test_new, predictions)
class_names=[0,1]
tick_marks = np.arange(len(class_names))
plt.xticks(tick_marks, class_names)
plt.yticks(tick_marks, class_names)
sns.heatmap(pd.DataFrame(matrix), annot=True, cmap=sns.cubehelix_palette(as_cmap=True), fmt='g')
plt.tight_layout()
plt.title('Confusion matrix', y=1.1)
plt.ylabel('Actual label')
plt.xlabel('Predicted label')
plt.show()
```

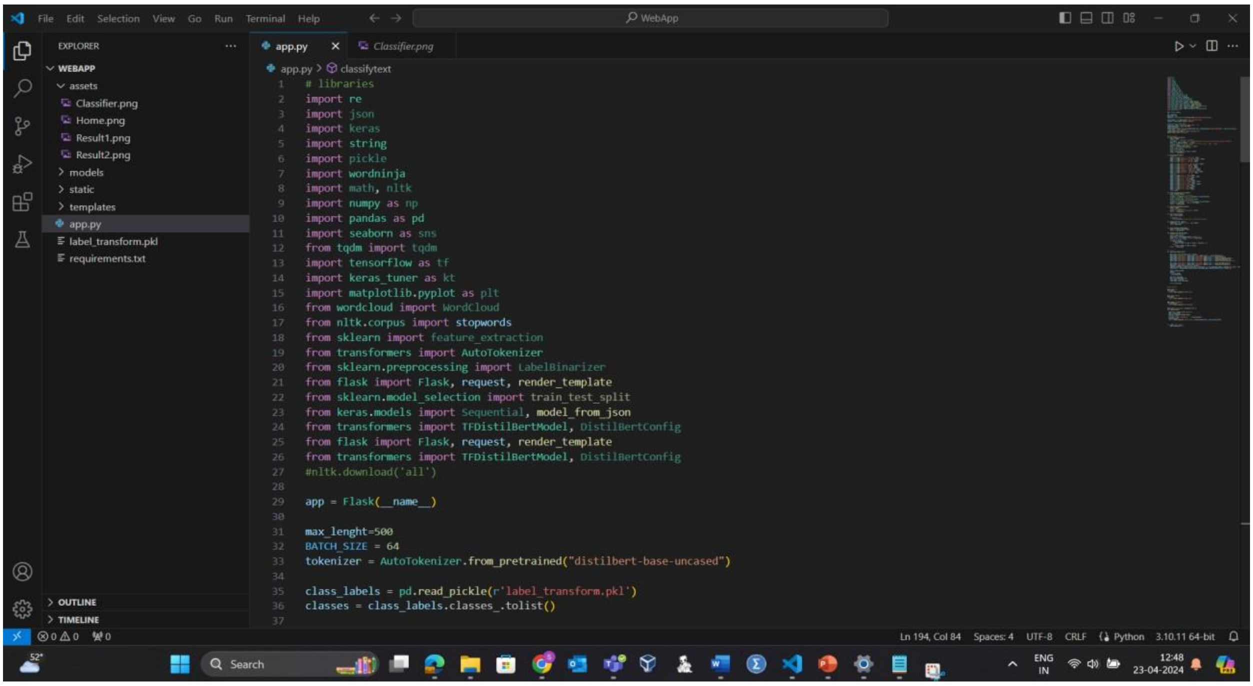


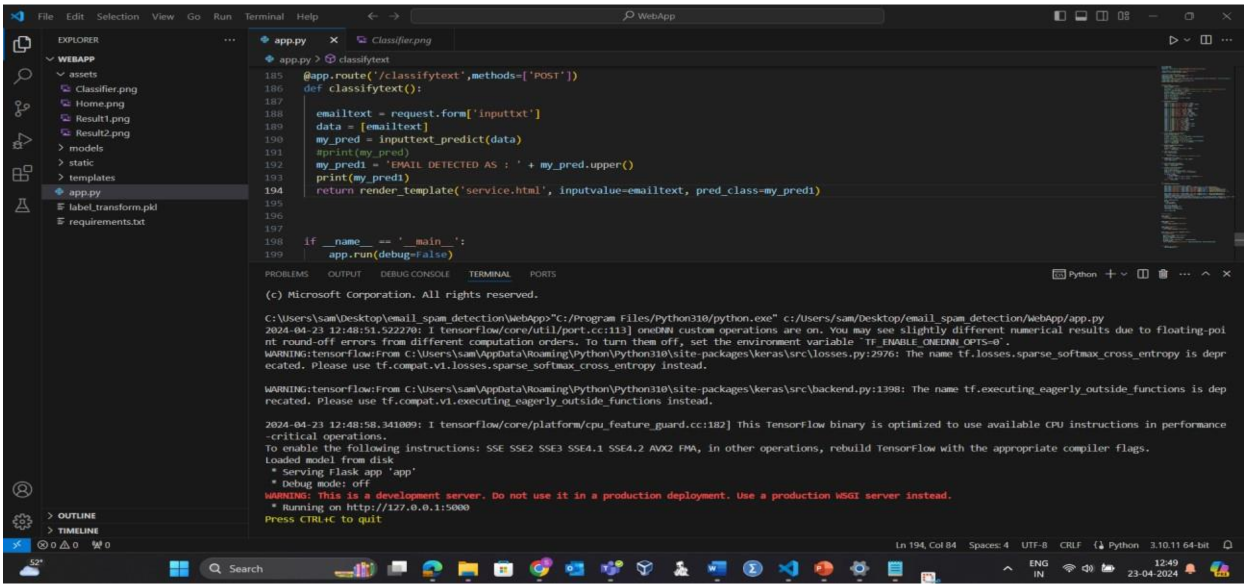
```
[ ] #classification report
    print(classification_report(y_test_new, predictions))

                  precision    recall  f1-score   support

               0       0.96      0.98      0.97       384
               1       0.95      0.87      0.91       134

        accuracy                           0.96       518
       macro avg       0.95      0.93      0.94       518
    weighted avg       0.96      0.96      0.95       518
```

Here Output of Best model is stored.

```
[ ] model_json = model.to_json()
    with open("/content/drive/MyDrive/email_spam_detection/models/bestmodel.json", "w") as json_file:
        json_file.write(model_json)
    # serialize weights to Hdataframe5
    model.save_weights("/content/drive/MyDrive/email_spam_detection/models/bestmodel.h5")
    print("Saved model to disk")

    Saved model to disk
```

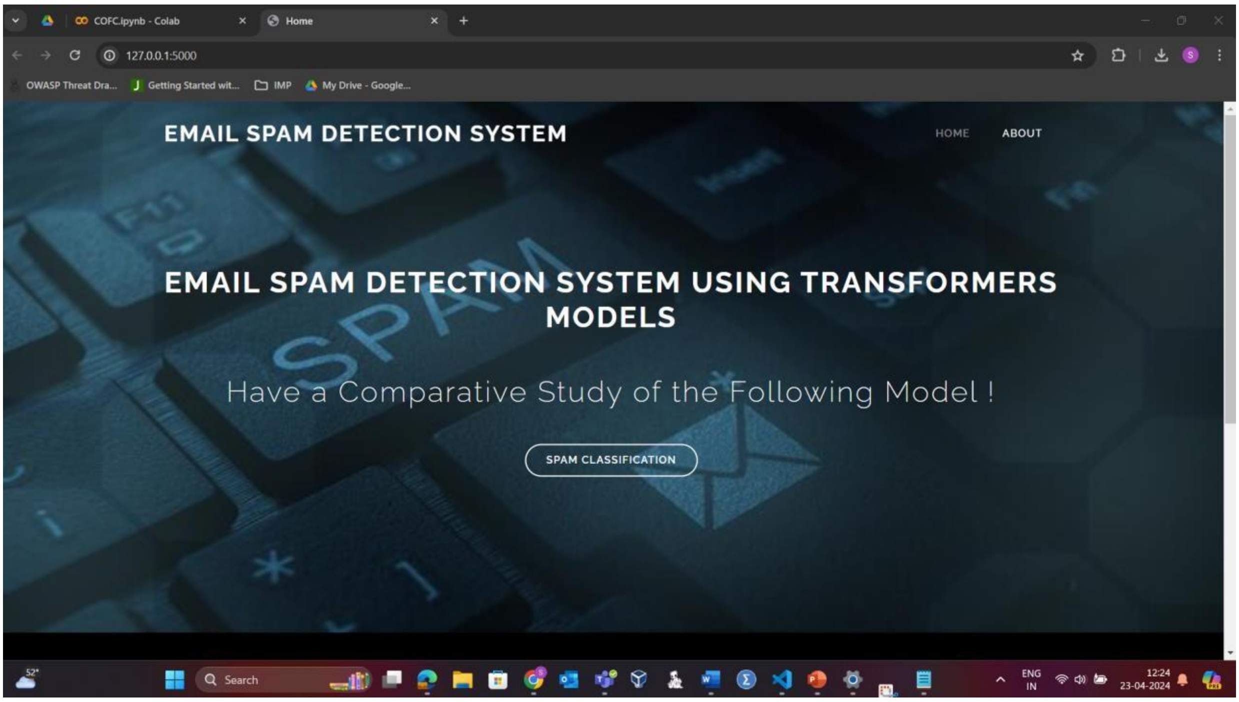
Visual Studio Code:



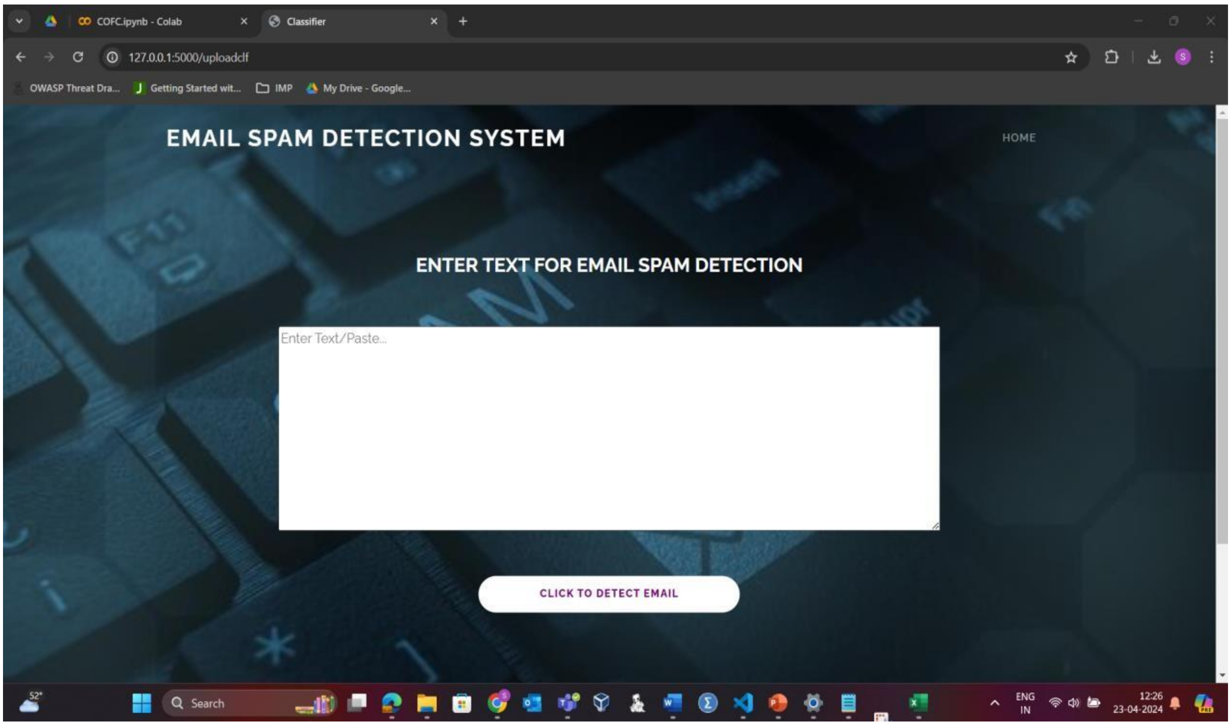


GUI Flask:

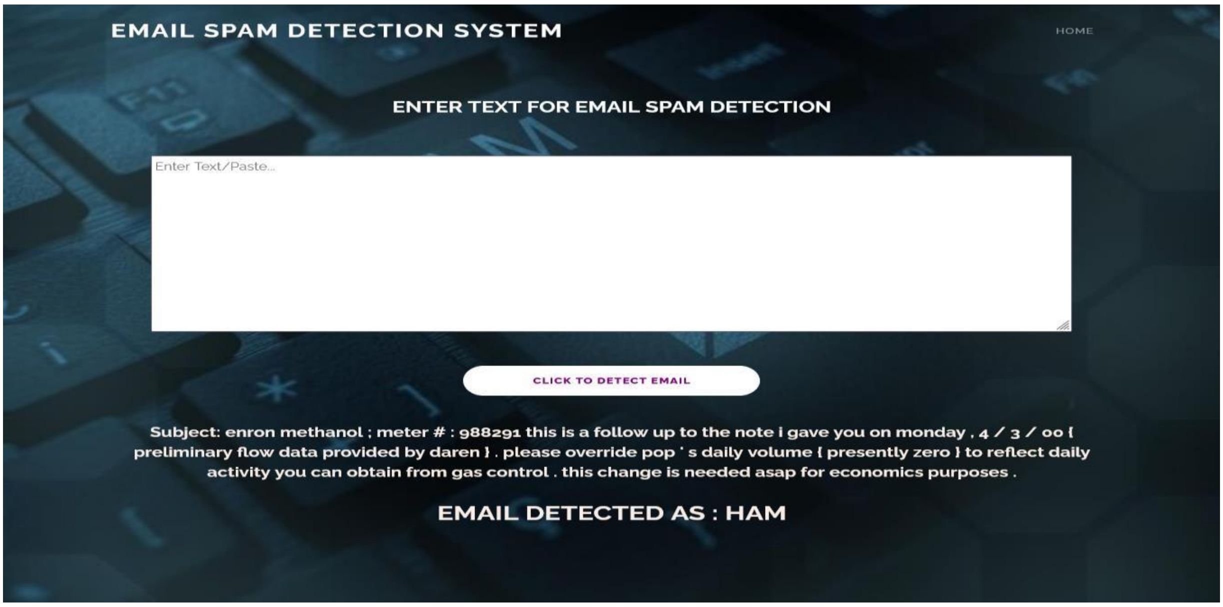
Shows the home Screen for the GUI application on Email Spam Detection.



Enter text for Email Spam Detection.



It concludes the entered text is Spam or Ham.



5 Conclusion

Researchers can get the same result by using the same code implementation and the datasets as those found in this work by following the steps outlined in this Configuration manual.

6 References

Anon., n.d. *The python tutorial.* [Online]
Available at: https://docs.python.org/3/tutorial
Anon., n.d. *Visual Studio Code.* [Online]
Available at: https://code.visualstudio.com/docs
I, R., n.d. *Google colab - a step-by-step guide - algotrading101 blog.* [Online]
Available at: https://algotrading101.com/learn/google-colab-guide/ [Accessed 2022].