Phishing Detection using Transformer

CS753 Practical Training
Under
Alwyn Roshan Pais

MASTER OF TECHNOLOGY in
COMPUTER SCIENCE AND ENGINEERING

by

Raju B Patat (212CS024)

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
NATIONAL INSTITUTE OF TECHNOLOGY - KARNATAKA
SURATHKAL, MANGALORE -575025
AUGUST, 2022

Contents

1 Topic **IV**

2 Motivation **IV**

3 Problem definition **V**

4 Literature Survey **VI**

5 Transformer **VIII**

5.1 Challenges with Traditional Deep Learning Models VIII
5.2 Transformer - Architecture VIII
5.3 Self Attention X
5.4 BERT Model XI

6 Dataset **XIII**

7 Implementation **XVI**

8 Results **XVIII**

8.1 Dataset1 XVIII
8.2 Dataset2 XXI
8.3 Dataset3 XXIV

9 Conclusion **XXVI**

10 References **XXVII**

1 Topic

Phishing Detection using Transformer.

2 Motivation

Attacks are of many types to disturb the network or any other websites. Phishing attacks (PA) are a type of attacks which attack the website and damage the website and may lose the data. Many types of research have been done to prevent the attacks. Phishing is the cyber attack that will destroy the website and may attack with the virus.Phishing may be a style of broad extortion that happens once a pernicious web site act sort of a real one memory that the last word objective to accumulate unstable info, as an example, passwords, account focal points, or MasterCard numbers. all the same, the means that there square measure some of contrary to phishing programming and techniques for recognizing potential phishing tries in messages and characteristic phishing substance on locales, phishes think about new and crossbreed procedures to bypass the open programming and frameworks. Phishing may be a fraud framework that uses a mixture of social designing what is additional, advancement to sensitive and personal data, as an example, passwords associate degree open-end credit unpretentious elements by presumptuous the highlights of a reliable individual or business in electronic correspondence. Phishing makes use of parody messages that square measure created to seem substantial and instructed to start out from true blue sources like money connected institutions, online business goals, etc, to draw in customers to go to phony destinations through joins gave within the phishing email.

Website Phishing costs internet users billions of dollars per year. Phishers steal personal information and financial account details such as usernames and passwords, leaving users vulnerable in the online space. The COVID-19 pandemic has boosted the use of technology in every sector, resulting in shifting of activities like organising official meetings, attending classes, shopping, payments, etc. from physical to online space. This means more opportunities for phishers to carry out attacks impacting the victim financially, psychologically and professionally. As per CheckPoint Research Security Report 2018, 77%of IT professionals feel their security teams are unprepared for today's cybersecurity challenge. The same report indicates that 64% of organizations have experienced a phishing attack in the past year.

Email has become one of the most important forms of communication. In 2014, there are estimated to be 4.1 billion email accounts worldwide, and about 196 billion emails are sent each day worldwide.[1] Spam is one of the major threats posed to email users. In 2013, 69.6% of all email flows were spam. Links in spam emails may lead to users to websites with malware or phishing schemes, which can access and disrupt the receiver's computer system. These sites can also gather sensitive information from. Additionally, spam costs businesses around $2000 per employee per year due to decreased productivity.Therefore, an effective spam filtering technology is a significant contribution to the sustainability of the cyberspace and to our society.

3 Problem definition

Emails are widely used as a means of communication for personal and professional use. The information exchanged over mails is often sensitive and confidential such as banking information, credit reports, login details etc. This makes them valuable to cyber criminals who can use the information for malicious purposes. Phishing is a strategy used by fraudsters to obtain sensitive information from people by pretending to be from recognized sources. In a phished email, the sender can convince you to provide personal information under false pretenses.

In this practical training I will try to classify the three Email Body data set.I will be using a Deep Learning Transformer model to classify the data as legitimate or Phished.

4 Literature Survey

In this section I have shown some of the similar work which have been already done.

- **Attention Is All You Need:** Recurrent neural networks, long short-term memory and gated recurrent neural networks in particular, have been firmly established as state of the art approaches in sequence modeling and transduction problems such as language modeling and machine translation. Numerous efforts have since continued to push the boundaries of recurrent language models and encoder-decoder architectures.ttention mechanisms have become an integral part of compelling sequence modeling and transduction models in various tasks, allowing modeling of dependencies without regard to their distance in the input or output sequences . In all but a few cases , however, such attention mechanisms are used in conjunction with a recurrent network. In this work they have propose the Transformer, a model architecture eschewing recurrence and instead relying entirely on an attention mechanism to draw global dependencies between input and output. The Transformer allows for significantly more parallelization and can reach a new state of the art in translation quality after being trained for as little.

- **Deep Learning in Phishing Detection** With the innovations in deep learning algorithms, effective phishing detection models using deep learning algorithms were recently developed. Although there are many different models developed so far, there are still some challenges with open solutions. In this study, they have performed a Systematic Literature Review (SLR) study, investigated 43 high-quality articles to respond to nine research questions, evaluated where and how deep learning algorithms were used. They also presented the challenges and open solutions in deep learning-based phishing detection models.

  The contributions of this article are five-fold:

  1. There was a lack of a general overview of deep learning-based phishing detection models and therefore, our SLR study filled this gap by addressing nine research questions.
  2. 43 high-quality deep learning-based phishing detection articles were investigated in detail and the required data were extracted and synthesized to respond to the research questions.
  3. Deep learning algorithms were briefly presented.
  4. The most used deep learning algorithms, the widely used datasets, machine learning types, development platforms, evaluation metrics, validation approaches, data sources, and feature selection algorithms were identified in detail.
  5. Challenges and research gaps were provided.

  We also identified the following directions for further research:

  1. The development of semi-supervised and unsupervised deep learning-based phishing detection models.
  2. The development of more hybrid deep learning-based models to improve the performance.
  3. Larger public datasets for phishing detection experiments
  4. A framework for comparing the performance of deep learning algorithms for phishing detection
  5. The development of Explainable Artificial Intelligence models for phishing detection
  6. The design of novel models for real-time phishing detection

  Researchers and practitioners can get benefit from the results of this study. Researchers can focus on the identified challenges and research gaps; practitioners can develop their tools based on the presented algorithms and models. As future work, they aimed to develop a novel deep learning-based phishing detection model by addressing one or more challenges discussed in this study.

5 Transformer

The Transformer in NLP is a novel architecture that aims to solve sequence-to-sequence tasks while handling long-range dependencies with ease. The Transformer was proposed in the paper "Attention Is All You Need".
Quoting from the paper:

“The Transformer is the first transduction model relying entirely on self-attention to compute representations of its input and output without using sequence-aligned RNNs or convolution.”

Here, “transduction” means the conversion of input sequences into output sequences. The idea behind Transformer is to handle the dependencies between input and output with attention and recurrence completely.

5.1 Challenges with Traditional Deep Learning Models

Sequence-to-sequence (seq2seq) models in NLP are used to convert sequences of Type A to sequences of Type B. For example, translation of English sentences to German sentences is a sequence-to-sequence task.
Recurrent Neural Network (RNN) based sequence-to-sequence models have garnered a lot of traction ever since they were introduced in 2014. Most of the data in the current world are in the form of sequences – it can be a number sequence, text sequence, a video frame sequence or an audio sequence. The performance of these seq2seq models was further enhanced with the addition of the Attention Mechanism in 2015.

Despite being so good at what it does, there are certain limitations of seq-2-seq models with attention:
1. Dealing with long-range dependencies is still challenging
2. The sequential nature of the model architecture prevents parallelization. These challenges are addressed by Google Brain’s Transformer concept

5.2 Transformer - Architecture

Begin by looking at the model as a single black box. In a machine translation application, it would take a sentence in one language, and output its translation in another.
We see an encoding component, a decoding component, and connections between them.
The encoding component is a stack of encoders (the paper stacks six of them on top of each other – there’s nothing magical about the number six, one can definitely experiment with other arrangements). The decoding component is a stack of decoders of the same number. The encoders are all identical in structure (yet they do not share weights). Each one is broken down into two sub-layers:

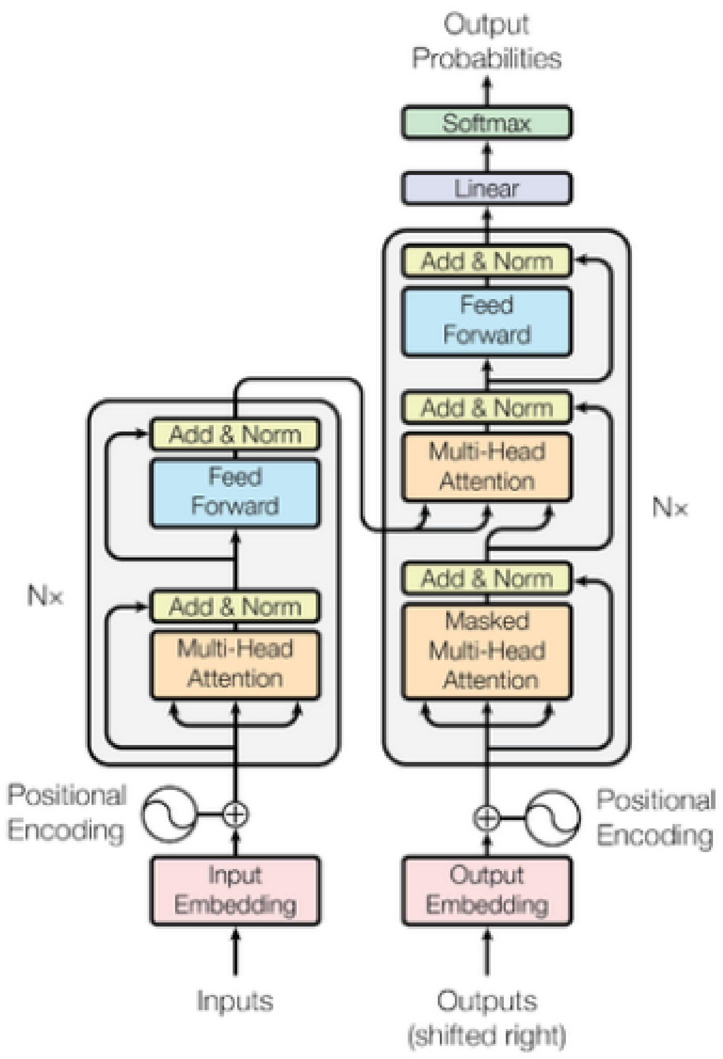


Figure 1: Architecture

The encoder's inputs first flow through a self-attention layer – a layer that helps the encoder look at other words in the input sentence as it encodes a specific word. We'll look closer at self-attention later in the post.

The outputs of the self-attention layer are fed to a feed-forward neural network. The exact same feed-forward network is independently applied to each position.

The decoder has both those layers, but between them is an attention layer that helps the decoder focus on relevant parts of the input sentence (similar what attention does in seq2seq models).

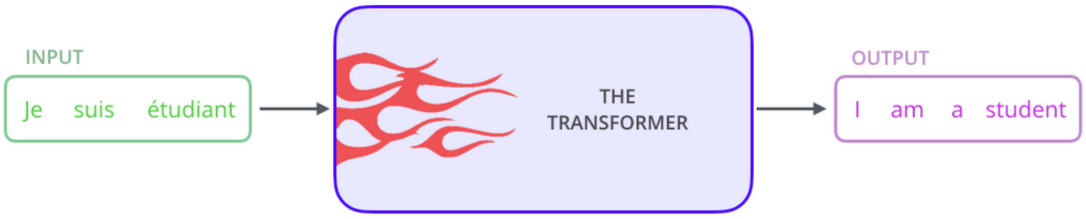


Figure 2: High Level View of Transformer

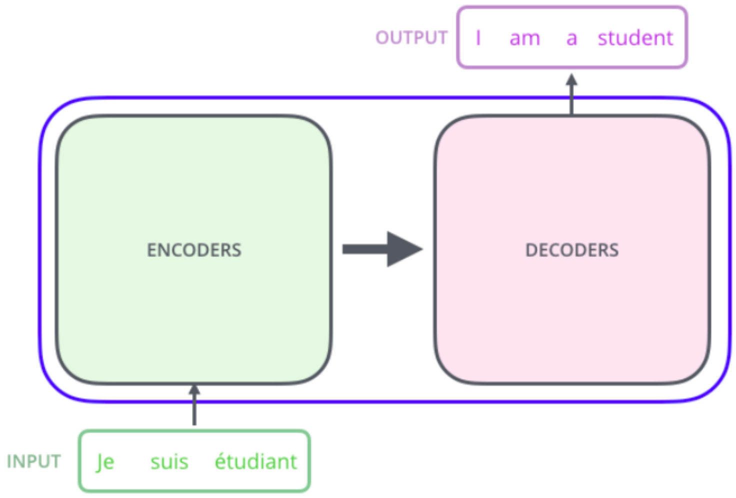


Figure 3: Encoder and Decoder 1

5.3 Self Attention

Say the following sentence is an input sentence we want to translate:

”The animal didn’t cross the street because it was too tired”

What does “it” in this sentence refer to? Is it referring to the street or to the animal? It’s a simple question to a human, but not as simple to an algorithm.

When the model is processing the word “it”, self-attention allows it to associate “it” with “animal”.

As the model processes each word (each position in the input sequence), self attention allows it to look at other positions in the input sequence for clues that can help lead to a better encoding for this word.

If you’re familiar with RNNs, think of how maintaining a hidden state allows an RNN to incorporate its representation of previous words/vectors it has processed with the current one it’s processing. Self-attention is the method the Transformer uses to brake the “understanding” of other relevant words into the one we’re currently processing.

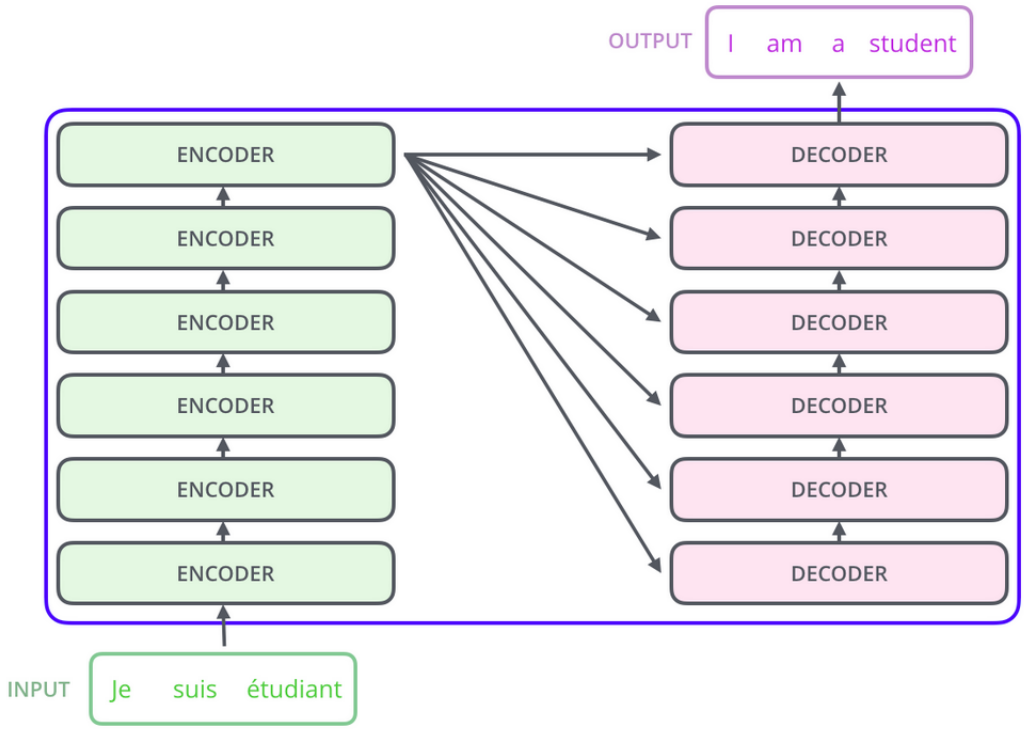


Figure 4: Encoder and Decoder 2

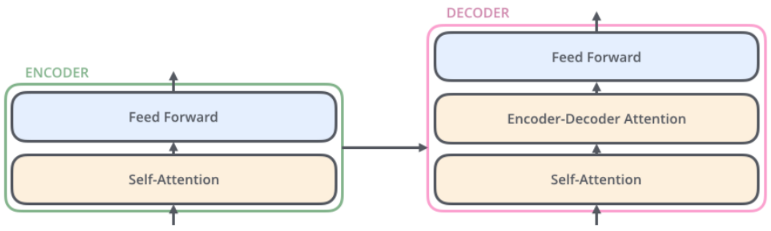


Figure 5: Inside Encoder and Decoder

5.4 BERT Model

Bidirectional Encoder Representations from Transformers (BERT) is a transformer-based machine learning technique for natural language processing (NLP) pre-training developed by Google. BERT was created and published in 2018 by Jacob Devlin and his colleagues from Google.In 2019, Google announced that it had begun leveraging BERT in its search engine, and by late 2020 it was using BERT in almost every English-language query. A 2020 literature survey concluded that "in a little over a year, BERT has become a ubiquitous baseline in NLP experiments", counting over 150 research publications analyzing and improving the model.

The original English-language BERT has two models:

(1) the BERTBASE: 12 encoders with 12 bidirectional self-attention heads, and

(2) the BERTLARGE: 24 encoders with 16 bidirectional self-attention heads. Both models are pre-trained from unlabeled data extracted from the BooksCorpus with 800M words and English Wikipedia with 2,500M words.

Architecture

BERT is at its core a transformer language model with a variable number of encoder layers and self-attention heads. The architecture is "almost identical" to the original transformer implementation in Vaswani et al. (2017).

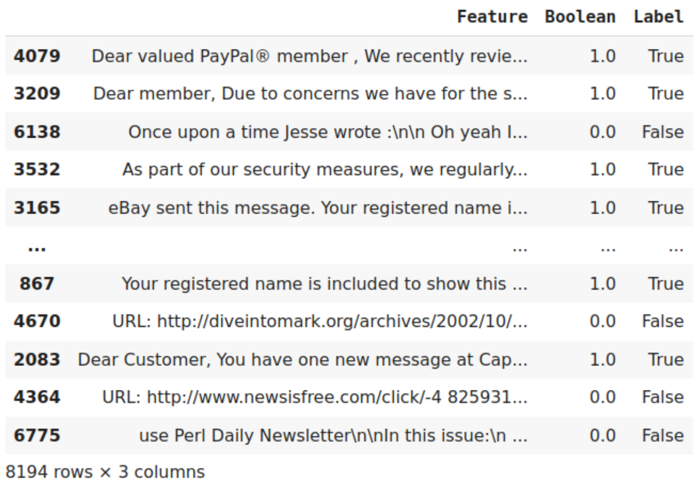
BERT was pretrained on two tasks: language modelling (15 Percent of tokens were masked and BERT was trained to predict them from context) and next sentence prediction (BERT was trained to predict if a chosen next sentence was probable or not given the first sentence). As a result of the training process, BERT learns contextual embeddings for words. After pretraining, which is computationally expensive, BERT can be finetuned with less resources on smaller datasets to optimize its performance on specific tasks.

How it Works

BERT makes use of Transformer, an attention mechanism that learns contextual relations between words (or sub-words) in a text. In its vanilla form, Transformer includes two separate mechanisms — an encoder that reads the text input and a decoder that produces a prediction for the task. Since BERT's goal is to generate a language model, only the encoder mechanism is necessary. As opposed to directional models, which read the text input sequentially (left-to-right or right-to-left), the Transformer encoder reads the entire sequence of words at once. Therefore it is considered bidirectional, though it would be more accurate to say that it's non-directional. This characteristic allows the model to learn the context of a word based on all of its surroundings (left and right of the word).

6 Dataset

I have used 3 datasets.
1. First dataset is the open dataset available on the internet.It has the body of the email and the Label i.e Phished or Legitimate. It has 8194 rows with 4229 as Phished and 3965 as Legitimate Label.



	Feature	Boolean	Label
4079	Dear valued PayPal® member , We recently revie...	1.0	True
3209	Dear member, Due to concerns we have for the s...	1.0	True
6138	Once upon a time Jesse wrote :\n\n Oh yeah I...	0.0	False
3532	As part of our security measures, we regularly...	1.0	True
3165	eBay sent this message. Your registered name i...	1.0	True
...	...	...	...
867	Your registered name is included to show this ...	1.0	True
4670	URL: http://diveintomark.org/archives/2002/10/...	0.0	False
2083	Dear Customer, You have one new message at Cap...	1.0	True
4364	URL: http://www.newsisfree.com/click/-4 825931...	0.0	False
6775	use Perl Daily Newsletter\n\nIn this issue:\n ...	0.0	False

8194 rows × 3 columns

Figure 6: Dataset 1

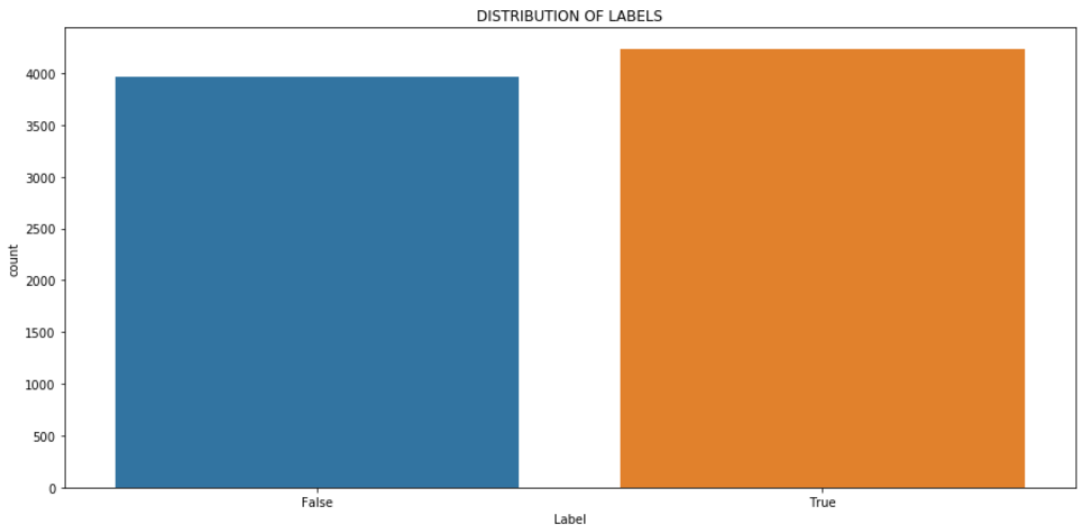
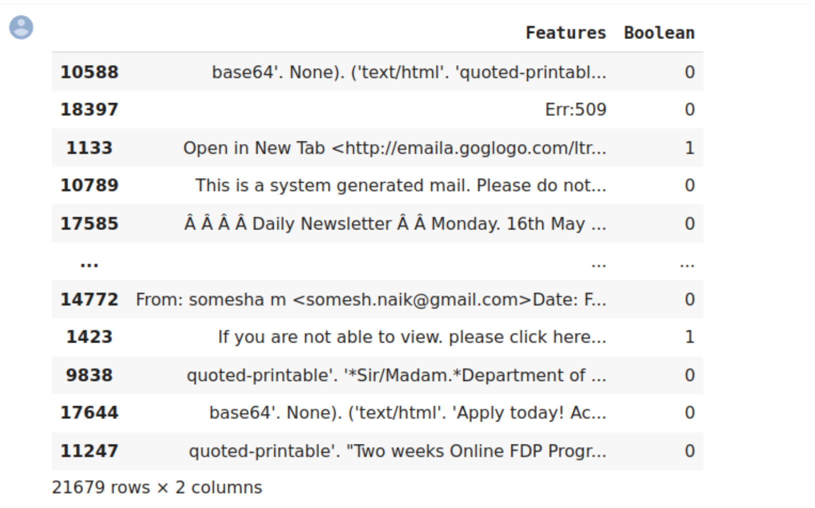


Figure 7: Label Classification of Dataset 1

2. Second dataset .It has the body of the email and Label i.e Phished or Legitimate.It has 21679 rows with 9392 as Phished and 12287 as Legitimate Label.



	Features	Boolean
10588	base64'. None). ('text/html'. 'quoted-printabl...	0
18397	Err:509	0
1133	Open in New Tab <http://emaila.goglogo.com/ltr...	1
10789	This is a system generated mail. Please do not...	0
17585	Â Â Â Â Daily Newsletter Â Â Monday. 16th May ...	0
...	...	...
14772	From: somesha m <somesh.naik@gmail.com>Date: F...	0
1423	If you are not able to view. please click here...	1
9838	quoted-printable'. '*Sir/Madam.*Department of ...	0
17644	base64'. None). ('text/html'. 'Apply today! Ac...	0
11247	quoted-printable'. "Two weeks Online FDP Progr...	0

21679 rows × 2 columns

Figure 8: Dataset 2

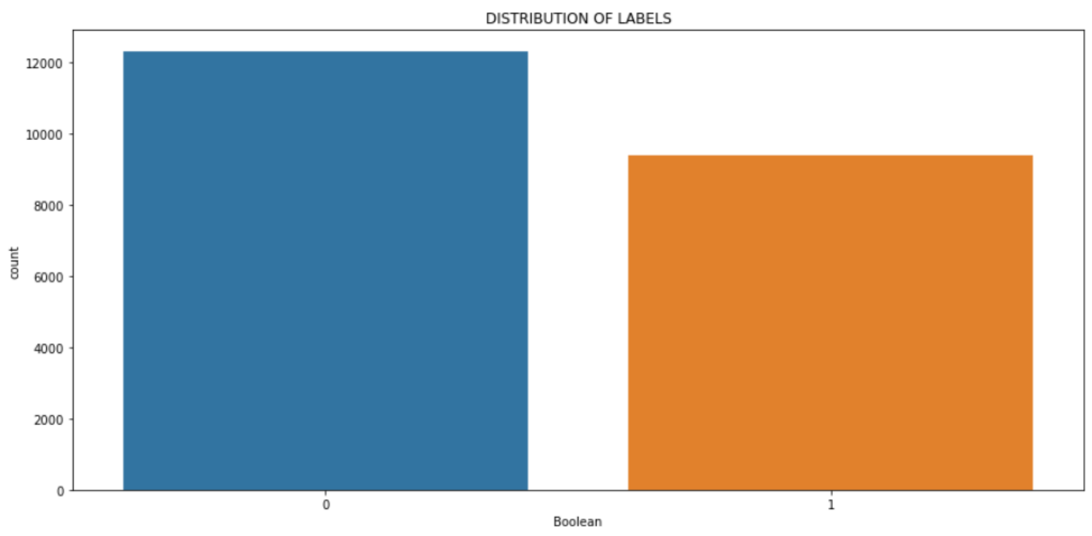
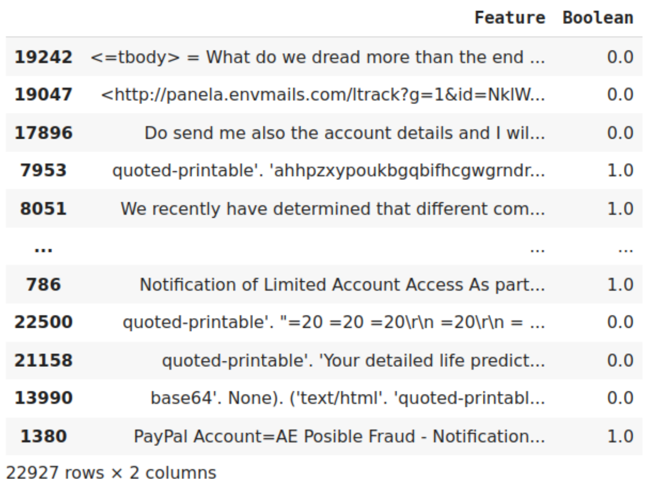


Figure 9: Label Classification of Dataset 2

3. Second dataset .It has the body of the email and Label i.e Phished or Legitimate.It has 22927 rows with 12287 as Phished and 10640 as Legitimate Label.



	Feature	Boolean
19242	<=tbody> = What do we dread more than the end ...	0.0
19047	<http://panela.envmails.com/ltrack?g=1&id=NklW...	0.0
17896	Do send me also the account details and I wil...	0.0
7953	quoted-printable'. 'ahhpzxypoukbgqbifhcgwgrndr...	1.0
8051	We recently have determined that different com...	1.0
...	...	...
786	Notification of Limited Account Access As part...	1.0
22500	quoted-printable'. "=20 =20 =20\r\n =20\r\n = ...	0.0
21158	quoted-printable'. 'Your detailed life predict...	0.0
13990	base64'. None). ('text/html'. 'quoted-printabl...	0.0
1380	PayPal Account=AE Posible Fraud - Notification...	1.0

22927 rows × 2 columns

Figure 10: Dataset 3

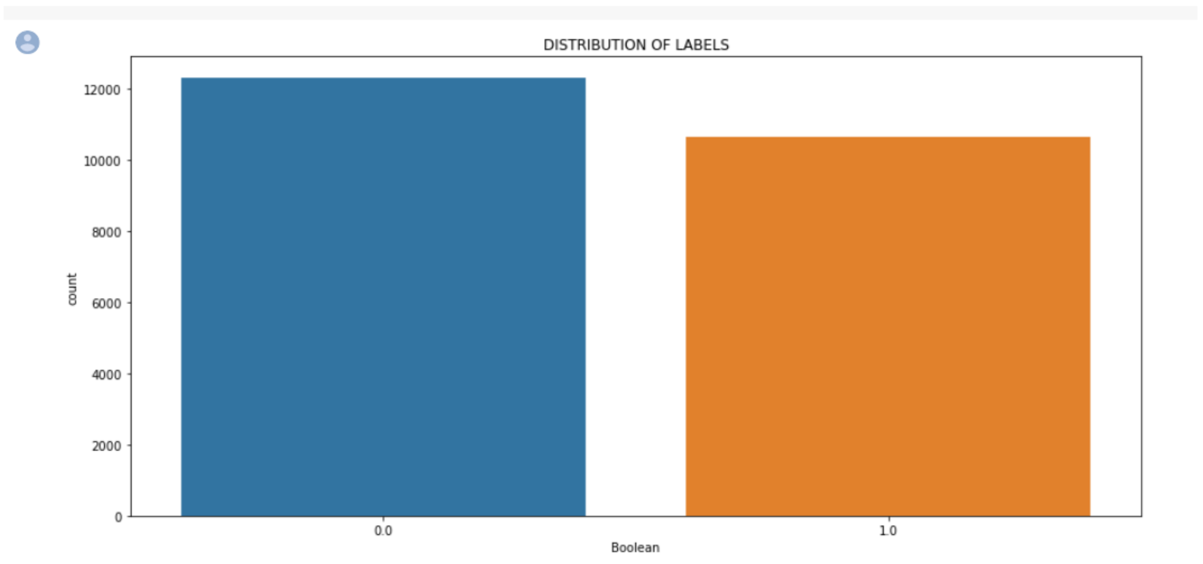
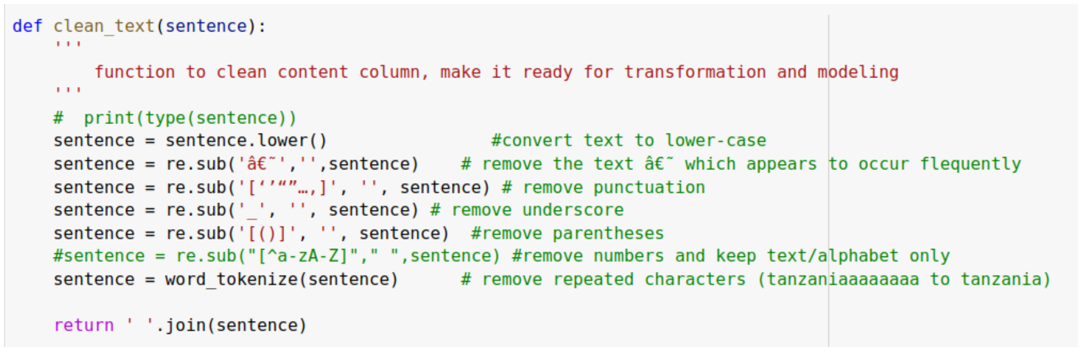


Figure 11: Label Classification of Dataset 3

7 Implementation

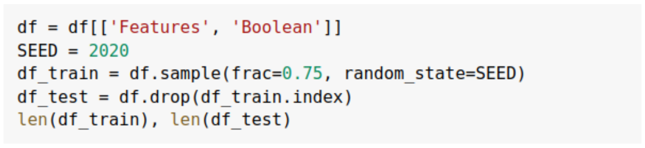
I have used the Ktrain library , so it must be installed. Along with that all the depending libraries need to be installed beforehand.

I also did cleaning of the data, in which I removed the punctiotions , some unknown symbols , underscores, parenthesis and converted everything to small letters. I have split the dataset in 75% as Training and 25% as Validation.



```
def clean_text(sentence):
    '''
        function to clean content column, make it ready for transformation and modeling
    '''
    #  print(type(sentence))
    sentence = sentence.lower()                #convert text to lower-case
    sentence = re.sub('â€˜','',sentence)    # remove the text â€˜ which appears to occur flequently
    sentence = re.sub('[‘’“”…,]', '', sentence) # remove punctuation
    sentence = re.sub('_', '', sentence) # remove underscore
    sentence = re.sub('[()]', '', sentence)  #remove parentheses
    #sentence = re.sub("[^a-zA-Z]"," ",sentence) #remove numbers and keep text/alphabet only
    sentence = word_tokenize(sentence)      # remove repeated characters (tanzaniaaaaaaaa to tanzania)

    return ' '.join(sentence)
```

Figure 12: Data Cleaning



```
df = df[['Features', 'Boolean']]
SEED = 2020
df_train = df.sample(frac=0.75, random_state=SEED)
df_test = df.drop(df_train.index)
len(df_train), len(df_test)
```

Figure 13: Training and Validation Dataset

I have used the Transformer Model : bert-base-uncase. I have got best result after fine tuning the parameters as :

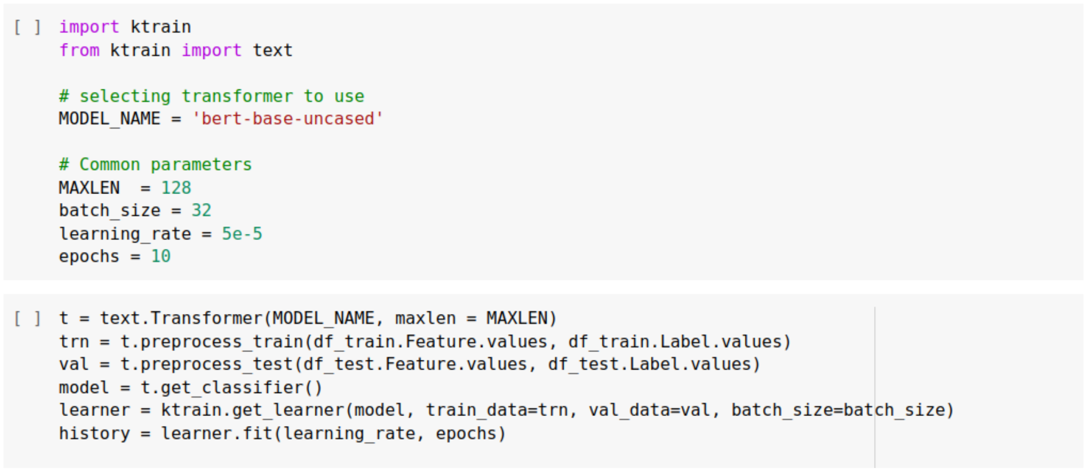
MAXLEN = 128

batch_size = 32

learning_rate = 5e-5

epochs = 10

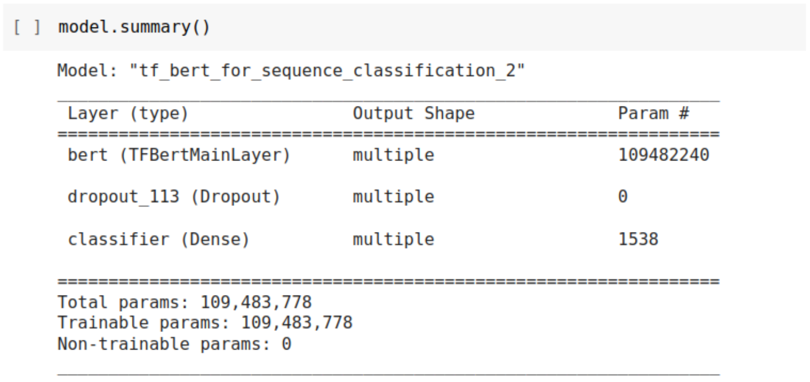
The Model summary is given in the figure below. It has total 109,483,778 Trainable parameters.



```
[ ] import ktrain
    from ktrain import text

    # selecting transformer to use
    MODEL_NAME = 'bert-base-uncased'

    # Common parameters
    MAXLEN  = 128
    batch_size = 32
    learning_rate = 5e-5
    epochs = 10
```

```
[ ] t = text.Transformer(MODEL_NAME, maxlen = MAXLEN)
    trn = t.preprocess_train(df_train.Feature.values, df_train.Label.values)
    val = t.preprocess_test(df_test.Feature.values, df_test.Label.values)
    model = t.get_classifier()
    learner = ktrain.get_learner(model, train_data=trn, val_data=val, batch_size=batch_size)
    history = learner.fit(learning_rate, epochs)
```

Figure 14: Model and Hyper Parameters



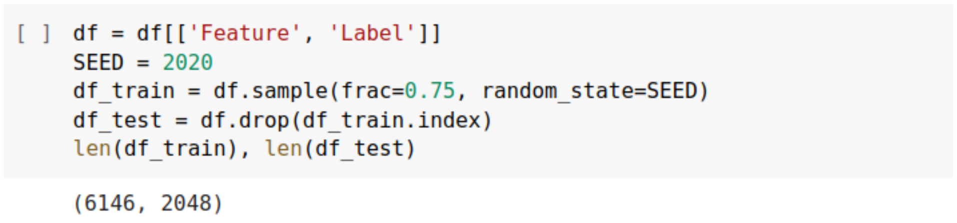
```
[ ] model.summary()

Model: "tf_bert_for_sequence_classification_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 bert (TFBertMainLayer)      multiple                  109482240

 dropout_113 (Dropout)       multiple                  0

 classifier (Dense)          multiple                  1538

=================================================================
Total params: 109,483,778
Trainable params: 109,483,778
Non-trainable params: 0
_________________________________________________________________
```

Figure 15: Model Summary

8 Results

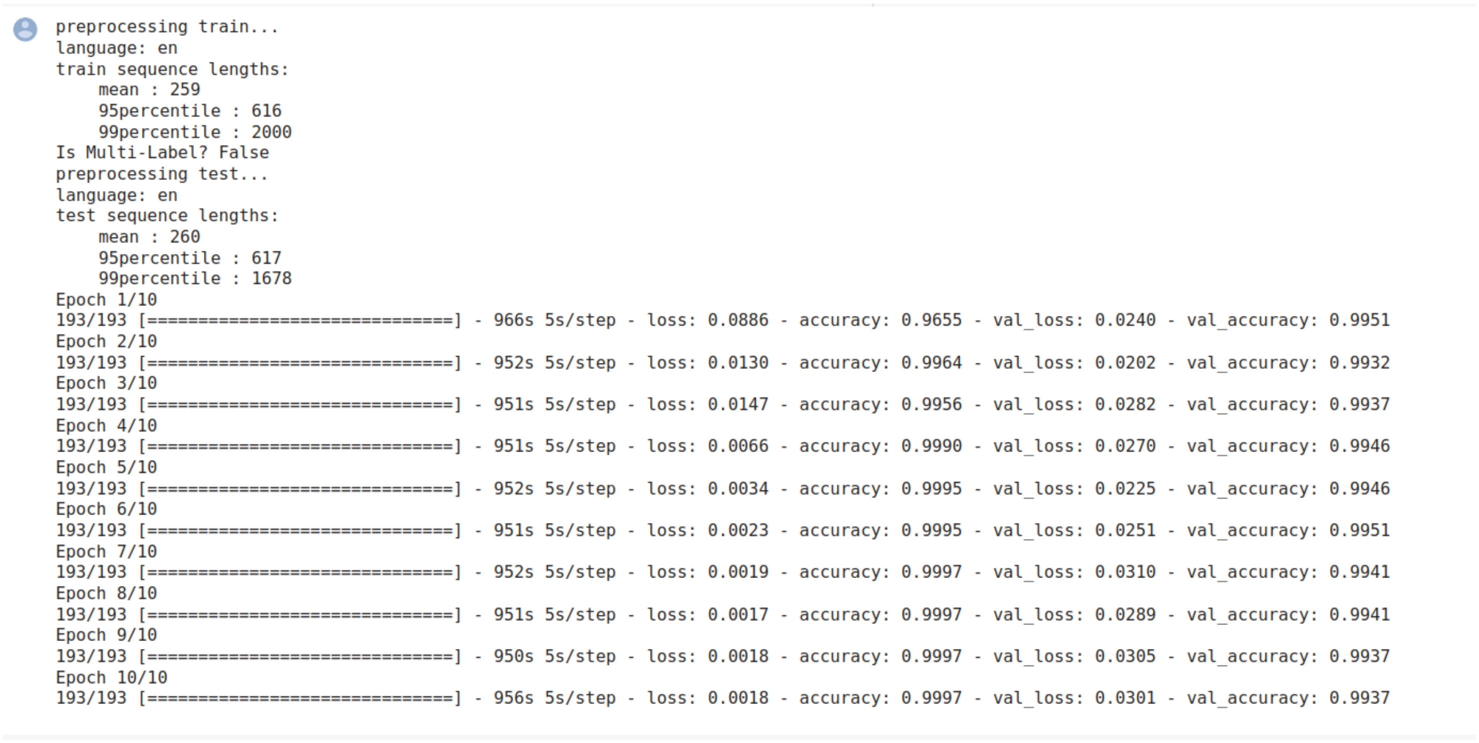
8.1 Dataset1

Time for each epoch : 16 minutes
Total time : 160 minutes.
Train Aaccuracy : 0.9997 Highest Validation Accuracy : 0.9951 Loss : 0.0018



```
[ ] df = df[['Feature', 'Label']]
    SEED = 2020
    df_train = df.sample(frac=0.75, random_state=SEED)
    df_test = df.drop(df_train.index)
    len(df_train), len(df_test)

    (6146, 2048)
```

Figure 16: Train and Test Data



```
preprocessing train...
language: en
train sequence lengths:
    mean : 259
    95percentile : 616
    99percentile : 2000
Is Multi-Label? False
preprocessing test...
language: en
test sequence lengths:
    mean : 260
    95percentile : 617
    99percentile : 1678
Epoch 1/10
193/193 [==============================] - 966s 5s/step - loss: 0.0886 - accuracy: 0.9655 - val_loss: 0.0240 - val_accuracy: 0.9951
Epoch 2/10
193/193 [==============================] - 952s 5s/step - loss: 0.0130 - accuracy: 0.9964 - val_loss: 0.0202 - val_accuracy: 0.9932
Epoch 3/10
193/193 [==============================] - 951s 5s/step - loss: 0.0147 - accuracy: 0.9956 - val_loss: 0.0282 - val_accuracy: 0.9937
Epoch 4/10
193/193 [==============================] - 951s 5s/step - loss: 0.0066 - accuracy: 0.9990 - val_loss: 0.0270 - val_accuracy: 0.9946
Epoch 5/10
193/193 [==============================] - 952s 5s/step - loss: 0.0034 - accuracy: 0.9995 - val_loss: 0.0225 - val_accuracy: 0.9946
Epoch 6/10
193/193 [==============================] - 951s 5s/step - loss: 0.0023 - accuracy: 0.9995 - val_loss: 0.0251 - val_accuracy: 0.9951
Epoch 7/10
193/193 [==============================] - 952s 5s/step - loss: 0.0019 - accuracy: 0.9997 - val_loss: 0.0310 - val_accuracy: 0.9941
Epoch 8/10
193/193 [==============================] - 951s 5s/step - loss: 0.0017 - accuracy: 0.9997 - val_loss: 0.0289 - val_accuracy: 0.9941
Epoch 9/10
193/193 [==============================] - 950s 5s/step - loss: 0.0018 - accuracy: 0.9997 - val_loss: 0.0305 - val_accuracy: 0.9937
Epoch 10/10
193/193 [==============================] - 956s 5s/step - loss: 0.0018 - accuracy: 0.9997 - val_loss: 0.0301 - val_accuracy: 0.9937
```

Figure 17: Model run

```
<matplotlib.legend.Legend at 0x7f077478f8b0>
```

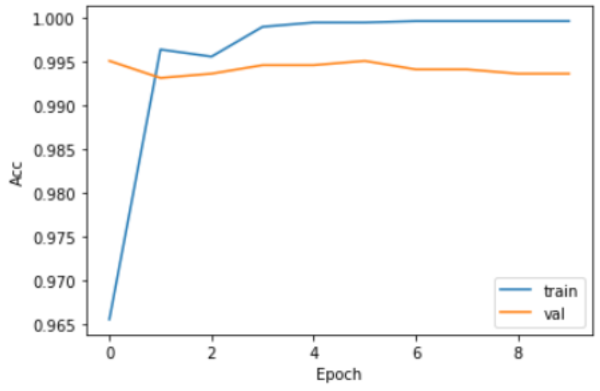


Figure 18: Accuracy Curve

```
<matplotlib.legend.Legend at 0x7f078d61f940>
```

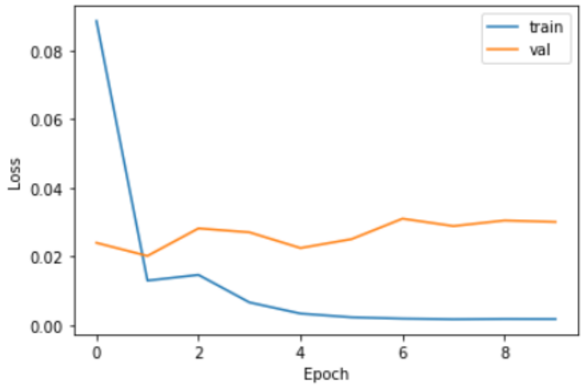
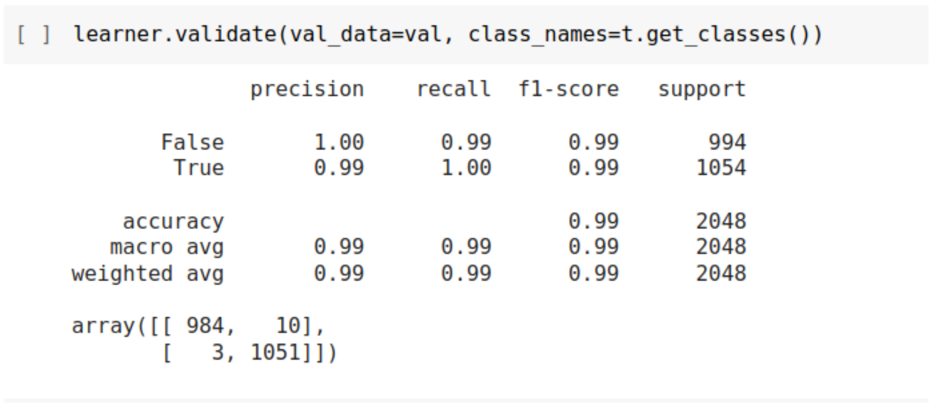


Figure 19: Loss Curve



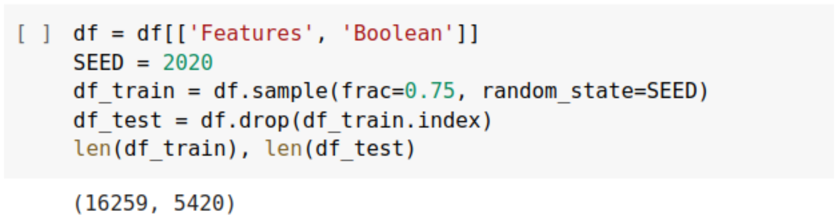
```
[ ]  learner.validate(val_data=val, class_names=t.get_classes())
```

```
              precision    recall  f1-score   support

       False       1.00      0.99      0.99       994
        True       0.99      1.00      0.99      1054

    accuracy                           0.99      2048
   macro avg       0.99      0.99      0.99      2048
weighted avg       0.99      0.99      0.99      2048

array([[ 984,   10],
       [   3, 1051]])
```

Figure 20: Confusion

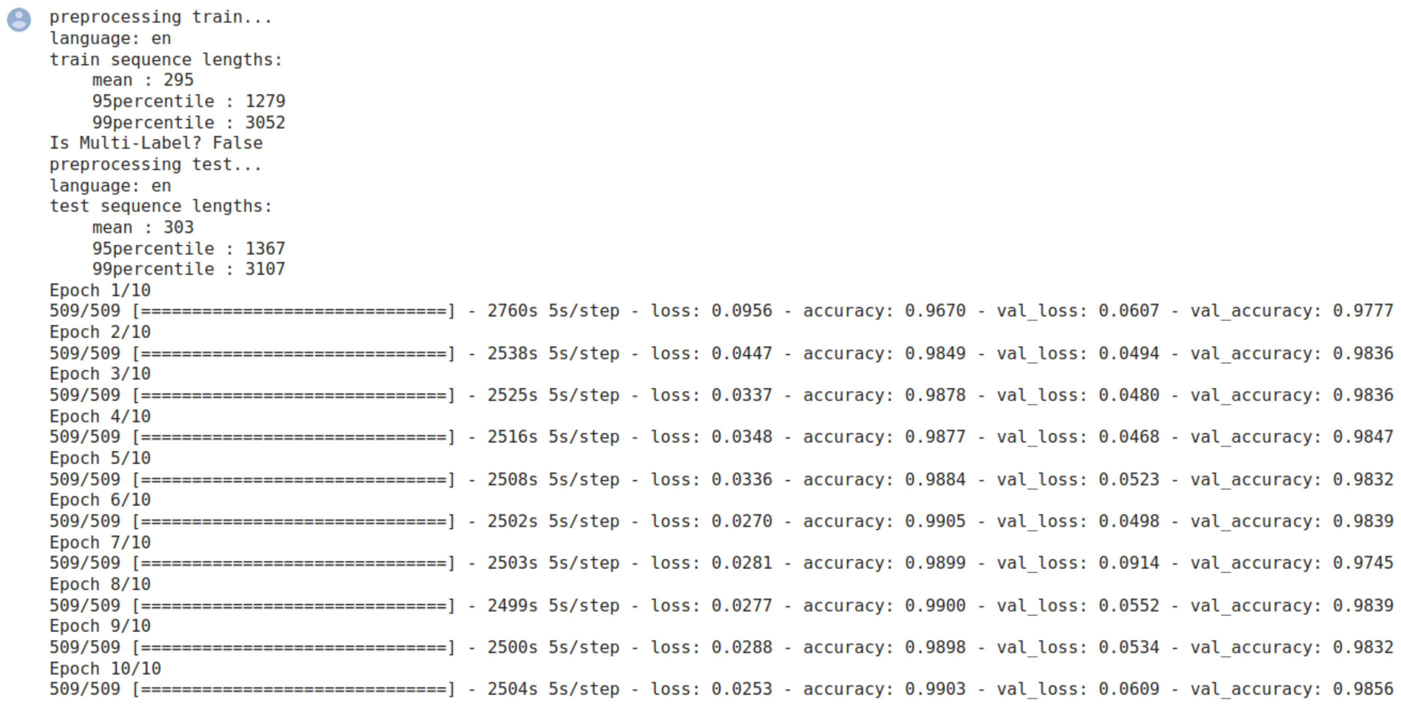
8.2 Dataset2

Time for each epoch : 41 minutes
Total time : 410 minutes.
Train Aaccuracy : 0.9903 Highest Validation Accuracy : 0.9856 Loss : 0.0253



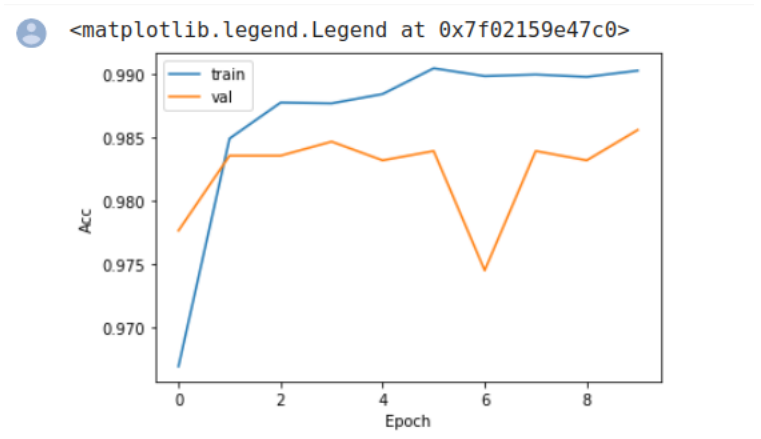
```
[ ] df = df[['Features', 'Boolean']]
    SEED = 2020
    df_train = df.sample(frac=0.75, random_state=SEED)
    df_test = df.drop(df_train.index)
    len(df_train), len(df_test)

    (16259, 5420)
```

Figure 21: Train and Test Data



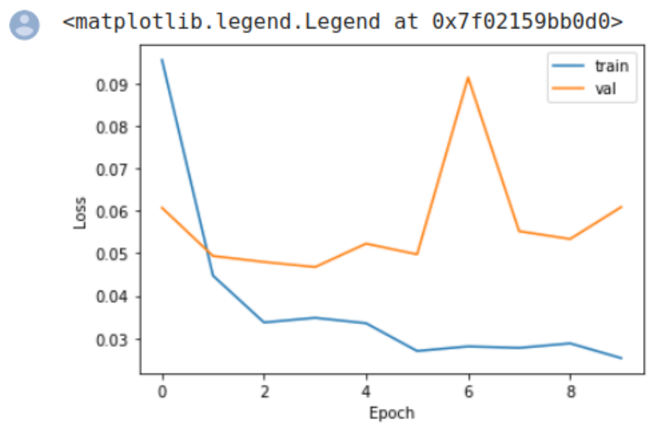
```
preprocessing train...
language: en
train sequence lengths:
    mean : 295
    95percentile : 1279
    99percentile : 3052
Is Multi-Label? False
preprocessing test...
language: en
test sequence lengths:
    mean : 303
    95percentile : 1367
    99percentile : 3107
Epoch 1/10
509/509 [==============================] - 2760s 5s/step - loss: 0.0956 - accuracy: 0.9670 - val_loss: 0.0607 - val_accuracy: 0.9777
Epoch 2/10
509/509 [==============================] - 2538s 5s/step - loss: 0.0447 - accuracy: 0.9849 - val_loss: 0.0494 - val_accuracy: 0.9836
Epoch 3/10
509/509 [==============================] - 2525s 5s/step - loss: 0.0337 - accuracy: 0.9878 - val_loss: 0.0480 - val_accuracy: 0.9836
Epoch 4/10
509/509 [==============================] - 2516s 5s/step - loss: 0.0348 - accuracy: 0.9877 - val_loss: 0.0468 - val_accuracy: 0.9847
Epoch 5/10
509/509 [==============================] - 2508s 5s/step - loss: 0.0336 - accuracy: 0.9884 - val_loss: 0.0523 - val_accuracy: 0.9832
Epoch 6/10
509/509 [==============================] - 2502s 5s/step - loss: 0.0270 - accuracy: 0.9905 - val_loss: 0.0498 - val_accuracy: 0.9839
Epoch 7/10
509/509 [==============================] - 2503s 5s/step - loss: 0.0281 - accuracy: 0.9899 - val_loss: 0.0914 - val_accuracy: 0.9745
Epoch 8/10
509/509 [==============================] - 2499s 5s/step - loss: 0.0277 - accuracy: 0.9900 - val_loss: 0.0552 - val_accuracy: 0.9839
Epoch 9/10
509/509 [==============================] - 2500s 5s/step - loss: 0.0288 - accuracy: 0.9898 - val_loss: 0.0534 - val_accuracy: 0.9832
Epoch 10/10
509/509 [==============================] - 2504s 5s/step - loss: 0.0253 - accuracy: 0.9903 - val_loss: 0.0609 - val_accuracy: 0.9856
```

Figure 22: Model run



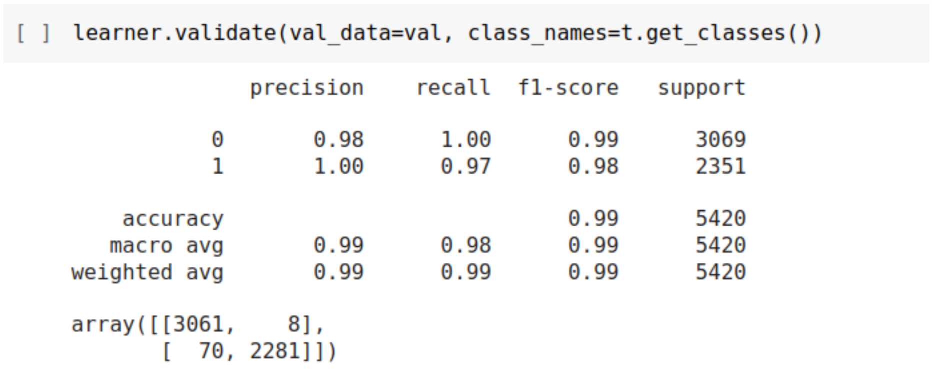
```
<matplotlib.legend.Legend at 0x7f02159e47c0>
```

Figure 23: Accuracy Curve



```
<matplotlib.legend.Legend at 0x7f02159bb0d0>
```

Figure 24: Loss Curve

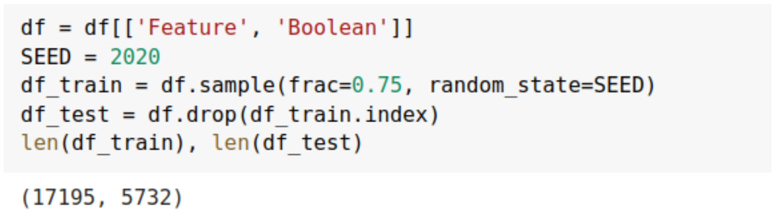


```
[ ]  learner.validate(val_data=val, class_names=t.get_classes())

                 precision    recall  f1-score   support

              0       0.98      1.00      0.99      3069
              1       1.00      0.97      0.98      2351

       accuracy                           0.99      5420
      macro avg       0.99      0.98      0.99      5420
   weighted avg       0.99      0.99      0.99      5420

   array([[3061,    8],
          [  70, 2281]])
```

Figure 25: Confusion

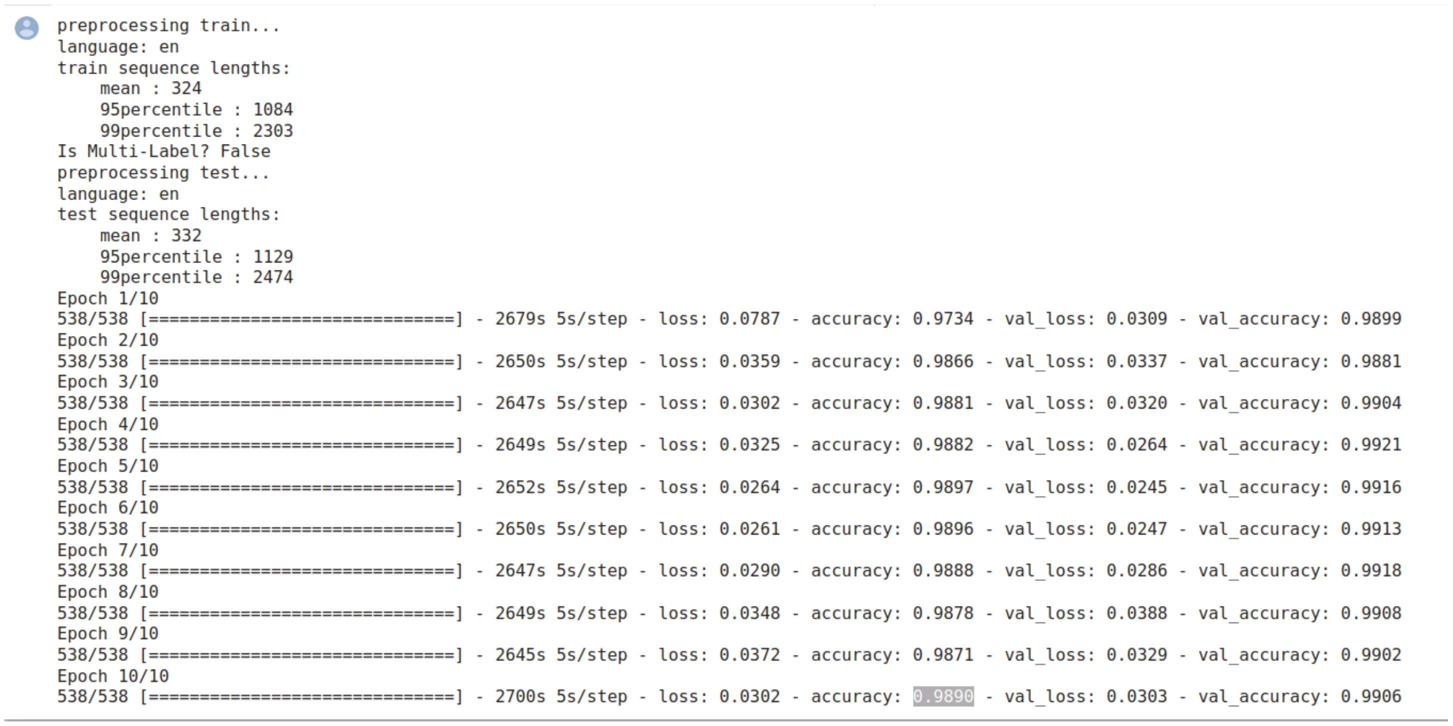
8.3 Dataset3

Time for each epoch : 45 minutes
Total time : 450 minutes.
Train Aaccuracy : 0.9890 Highest Validation Accuracy : 0.9921 Loss : 0.0302



```
df = df[['Feature', 'Boolean']]
SEED = 2020
df_train = df.sample(frac=0.75, random_state=SEED)
df_test = df.drop(df_train.index)
len(df_train), len(df_test)

(17195, 5732)
```

Figure 26: Train and Test Data



```
preprocessing train...
language: en
train sequence lengths:
    mean : 324
    95percentile : 1084
    99percentile : 2303
Is Multi-Label? False
preprocessing test...
language: en
test sequence lengths:
    mean : 332
    95percentile : 1129
    99percentile : 2474
Epoch 1/10
538/538 [==============================] - 2679s 5s/step - loss: 0.0787 - accuracy: 0.9734 - val_loss: 0.0309 - val_accuracy: 0.9899
Epoch 2/10
538/538 [==============================] - 2650s 5s/step - loss: 0.0359 - accuracy: 0.9866 - val_loss: 0.0337 - val_accuracy: 0.9881
Epoch 3/10
538/538 [==============================] - 2647s 5s/step - loss: 0.0302 - accuracy: 0.9881 - val_loss: 0.0320 - val_accuracy: 0.9904
Epoch 4/10
538/538 [==============================] - 2649s 5s/step - loss: 0.0325 - accuracy: 0.9882 - val_loss: 0.0264 - val_accuracy: 0.9921
Epoch 5/10
538/538 [==============================] - 2652s 5s/step - loss: 0.0264 - accuracy: 0.9897 - val_loss: 0.0245 - val_accuracy: 0.9916
Epoch 6/10
538/538 [==============================] - 2650s 5s/step - loss: 0.0261 - accuracy: 0.9896 - val_loss: 0.0247 - val_accuracy: 0.9913
Epoch 7/10
538/538 [==============================] - 2647s 5s/step - loss: 0.0290 - accuracy: 0.9888 - val_loss: 0.0286 - val_accuracy: 0.9918
Epoch 8/10
538/538 [==============================] - 2649s 5s/step - loss: 0.0348 - accuracy: 0.9878 - val_loss: 0.0388 - val_accuracy: 0.9908
Epoch 9/10
538/538 [==============================] - 2645s 5s/step - loss: 0.0372 - accuracy: 0.9871 - val_loss: 0.0329 - val_accuracy: 0.9902
Epoch 10/10
538/538 [==============================] - 2700s 5s/step - loss: 0.0302 - accuracy: 0.9890 - val_loss: 0.0303 - val_accuracy: 0.9906
```

Figure 27: Model run

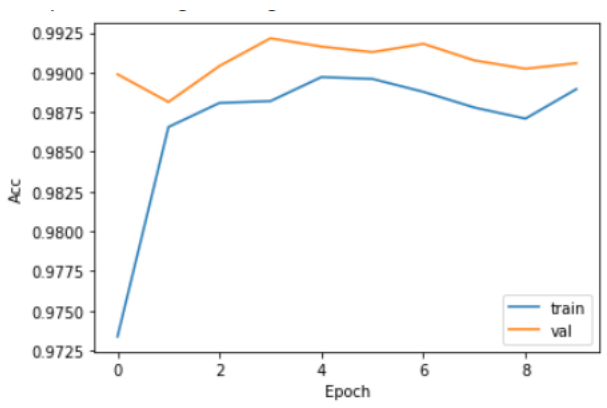


Figure 28: Accuracy Curve

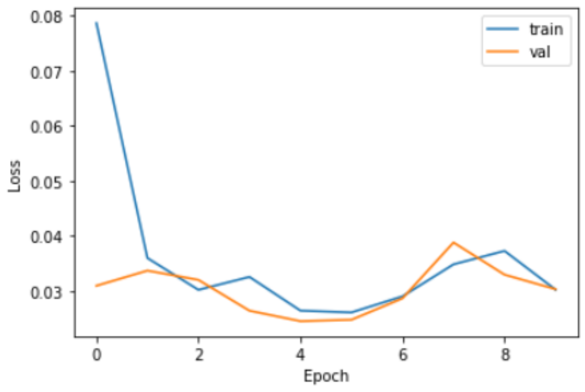
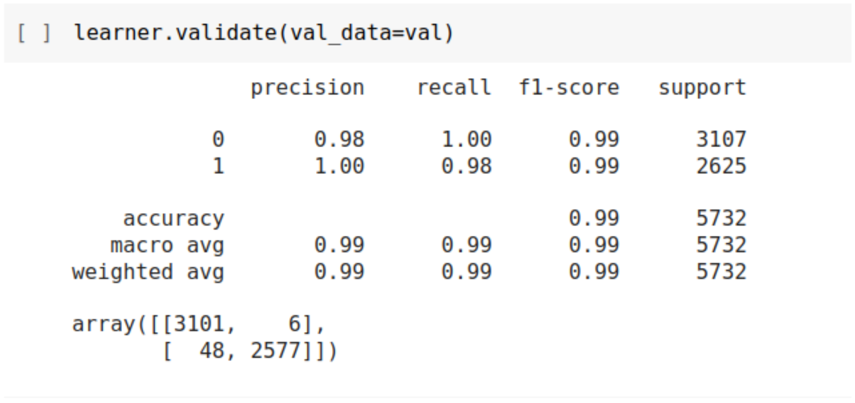


Figure 29: Loss Curve



```
[ ]  learner.validate(val_data=val)

              precision    recall  f1-score   support

           0       0.98      1.00      0.99      3107
           1       1.00      0.98      0.99      2625

    accuracy                           0.99      5732
   macro avg       0.99      0.99      0.99      5732
weighted avg       0.99      0.99      0.99      5732

array([[3101,    6],
       [  48, 2577]])
```

Figure 30: Confusion

9 Conclusion

I have classified the given three dataset comprising of the Email body into legitimate and phished. The model is savedd for each dataset so that it can be directly used . The respective results are shown in the result section.

10 References

References

[1] Attention Is All You Need. By: Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, Illia Polosukhin

[2] Jay Alammar - https://jalammar.github.io/illustrated-transformer/

[3] Googel Bert - https://ai.googleblog.com/2018/11/open-sourcing-bert-state-of-art-pre.html

[4] Applications of deep learning for phishing detection: a systematic literature review. By Cagatay,Bedir etc.

[5] Swahili Text Classification . https://dev.to/neurotech_africa/swahili-text-classification-using-transformers-4850

[6] Transformers - https://pypi.org/project/transformers/2.1.0/

[7] Transformers - https://towardsdatascience.com/transformers-89034557de14

[8] Transformers - https://www.analyticsvidhya.com/blog/2019/06/understanding-transformers-nlp-state-of-the-art-models/