Phishing Email Detection using BERT

A Deep Learning Approach with Transformers

Introduction

Phishing emails are one of the most common and dangerous cyber threats today. Attackers use deceptive messages to trick users into revealing sensitive information such as passwords, bank details, and OTPs. Traditional rule-based or keyword-based detection systems often fail to generalize against evolving phishing techniques.

In this project, we build a **BERT-based phishing email detection system** using **Transformer architecture** that can understand the contextual meaning of emails and accurately classify them as **phishing** or **legitimate**.

Dataset Description

The dataset used in this project is a real-world **Phishing Email Dataset** that contains labeled email messages for binary classification. It includes both legitimate and phishing emails, making it suitable for training and evaluating phishing detection models.

Dataset Source

- **Platform:** Kaggle
- **Dataset Name:** Phishing Email Dataset
- **Download Link:**
  https://www.kaggle.com/datasets/naserabdullahalam/phishing-email-dataset

Dataset Structure

The dataset consists of the following key columns:

Column Name	Description
`text`	The raw content of the email message
`label`	Target variable where `0` represents a legitimate email and `1` represents a phishing email

Dataset Characteristics

- Contains real-world phishing and legitimate email samples
- Text-based dataset suitable for NLP and deep learning models
- Binary classification problem
- Supports contextual learning using Transformer-based models such as BERT

This dataset provides a strong foundation for building and evaluating robust phishing email detection systems.

Required Packages and Dependencie

Core Programming Language

- **Python 3.12**

Deep Learning Framework

- **PyTorch**
- **Transformers (Hugging Face)**

Dataset Handling

- **Datasets (Hugging Face)**

Data Processing & Analysis

- **Pandas**
- **NumPy**
- **Regular Expressions (`re`)**

Model Evaluation

- **Scikit-learn**

Visualization

- **Matplotlib**
- **Seaborn**

GPU Acceleration (Colab)

- **CUDA**
- **FP16 (Mixed Precision)**

Environment & Reproducibility

- **pip**
- **requirements.txt**

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

```
Mounted at /content/drive
```

```
import numpy as np
import pandas as pd
import re
import torch

import seaborn as sns
import matplotlib.pyplot as plt

from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score,
precision_recall_fscore_support

from transformers import (
    BertTokenizer,
    BertForSequenceClassification,
    Trainer,
    TrainingArguments,
    DataCollatorWithPadding
)

from datasets import Dataset
import warnings
warnings.filterwarnings("ignore", category=UserWarning)

data = pd.read_csv('/content/drive/MyDrive/phishing_email.csv')
data.sample(10)
```


```
{"summary":"{\n  \"name\": \"data\",\n  \"rows\": 10,\n  \"fields\":
[\n    {\n      \"column\": \"text_combined\",\n      \"properties\":
{\n        \"dtype\": \"string\",\n        \"num_unique_values\": 10,\
n        \"samples\": [\n          \"small cap preview advantage
capital development corp avcp business development company operates
specifically meet needs small emerging companies need capital grow
current price 0 175 continue grow higher please view exactly company
watch trade week exciting one hot press release advantage capital
development corp advantage capital development corp portfolio company
pays senior secured debentureaug 17 2005 advantage capital development
corp announced today one portfolio companies global holdings inc
reduced debt company recent payment 250 000 senior secured debenture
advantage capital development corp recently increased stake global
holdings 15 22 1 2 percent global recently completed merger high road
international inc accordance transaction new public entity named
global holdings inc global holdings inc owns eighty five 85 percent
fully diluted shares pink sheet company company trades symbol pleased
investment global said jeffrey sternberg president ceo advantage
capital development corp leveraged relationship fullest extent utilize
funding enhance growth strategies according sternberg global pursuing
growth strategy currently various stages negotiations four possible
```

acquisitions staffing industry subsidiaries platinum consulting parker clark data processing global holdings managing internal growth actively seeking acquisitions compatible core business sternberg said platinum consulting associated company parker clark data processing served new york new jersey markets 25 years combined annual revenues excess 5 million according american staff association u annual sales temporary help totaled 63 3 billion 2004 nearly par industry sales peak 2000 12 5 2003 according association past three decades industry grown rate 10 percent year additionally ninety percent u companies use temporary staffing services advantage capital development corp advantage capital business development company operates specifically meet needs small emerging companies need capital grow business development companies defined investment act 1940 specifically designed encourage growth small businesses rules provide certain financing advantages companies invest small emerging businesses result include investing public private entities using certain types debt equity financing normally available public companies conclusion examples show awesome earning potential little known companies explode onto investors radar screens many already familiar avcp poised positioned may feel time come act please watch one trade thursday week go avcp _ penny stocks considered highly speculative may unsuitable aggressive investors profile way affiliated featured company compensated 3000 dollarsto distribute report report entertainment advertising purposes used investm 3 nt advice wish stop future mailings feel wrongfully placed membership send blank e mail thanks subject noth 4 nkso 0 yahoo com\",\n \"power plant model ken rdi gone model together clear idea going step step believe almost start recoding need c programmer familiar access data base e knows extract data access data table certain format output results access data table certain format programmer assigned go model incorporate ideas ken start coding alex\",\n \"meeting feb 8 2001 dear mr nur azmin abu bakar thanks prompt reply please let us know many members team visit enron look forward meeting february 8 vince kaminski azminab petronas com 01 02 2001 06 38 33 pm vince j kaminski enron com khairuddinbmjaafar petronas com shirley crenshaw enron com cc subject meeting feb 8 2001 dear kaminski happy new year thank reply honored lunch team however another appointment 2 30 p regards vince j kaminski enron com 03 01 2001 07 38 19 azminab petronas com cc vince j kaminski enron com shirley crenshaw enron com subject meeting feb 8 2001 dear sir would like apologize delay responding fax vacation last days shall honored meet delegation thursday february 8 10 00 please let know free lunch meeting vince kaminski\"\n],\n \"semantic_type\": \"\",\n \"description\": \"\"\n }\n },\n {\n \"column\": \"label\",\n \"properties\": {\n \"dtype\": \"number\",\n \"std\": 0,\n \"min\": 0,\n \"max\": 1,\n \"num_unique_values\": 2,\n \"samples\": [\n 1,\n 0\n],\n

```
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    }\n  ]\n}","type":"dataframe"}
```

```
data.shape
```

```
(82486, 2)
```

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 82486 entries, 0 to 82485
Data columns (total 2 columns):
 #   Column         Non-Null Count  Dtype 
---  ------         --------------  ----- 
 0   text_combined  82486 non-null  object
 1   label          82486 non-null  int64 
dtypes: int64(1), object(1)
memory usage: 1.3+ MB
```

```
data.isnull().sum()
```

```
text_combined    0
label            0
dtype: int64
```

```
data = data.dropna()
```

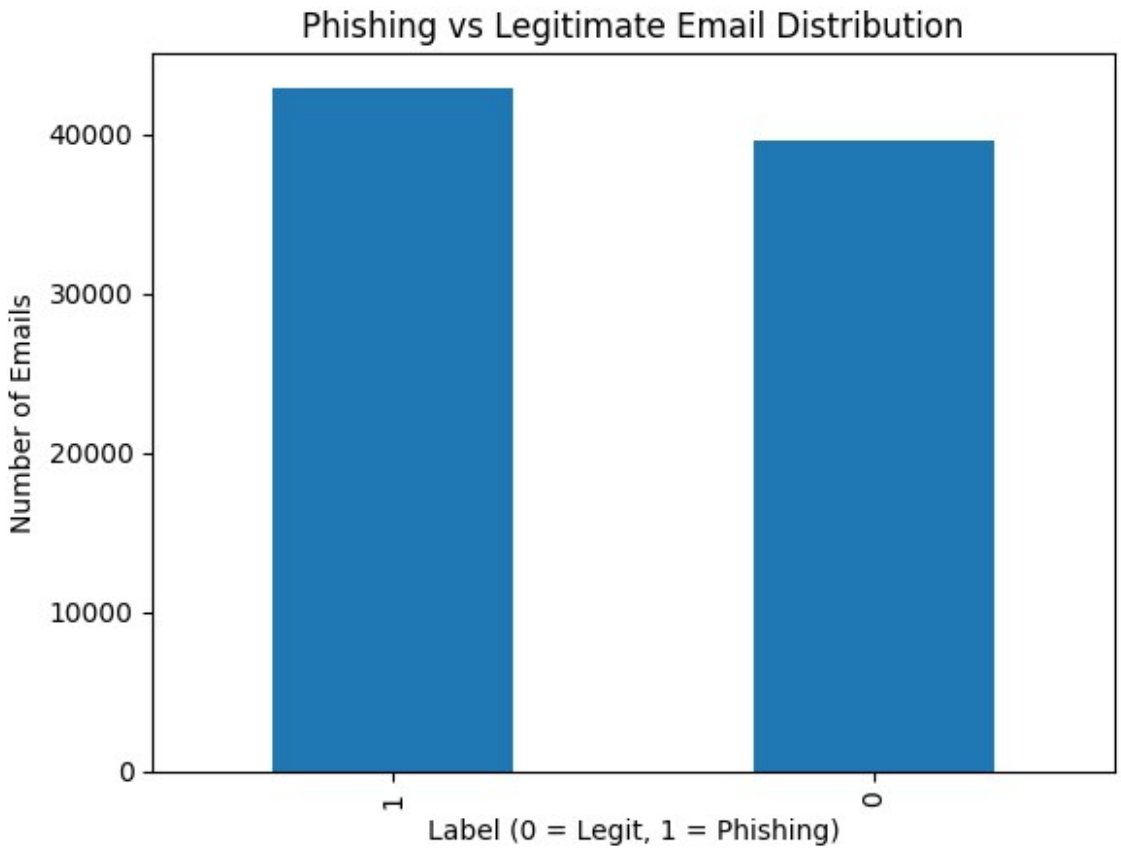
```
data["label"].value_counts()
```

```
label
1    42891
0    39595
Name: count, dtype: int64
```

```
data["label"].value_counts(normalize=True) * 100
```

```
label
1    51.997915
0    48.002085
Name: proportion, dtype: float64
```

```
import matplotlib.pyplot as plt

data["label"].value_counts().plot(kind="bar")
plt.title("Phishing vs Legitimate Email Distribution")
plt.xlabel("Label (0 = Legit, 1 = Phishing)")
plt.ylabel("Number of Emails")
plt.show()
```




```
data["text_length"] = data["text_combined"].apply(lambda x:
len(str(x).split()))

data["text_length"]
```

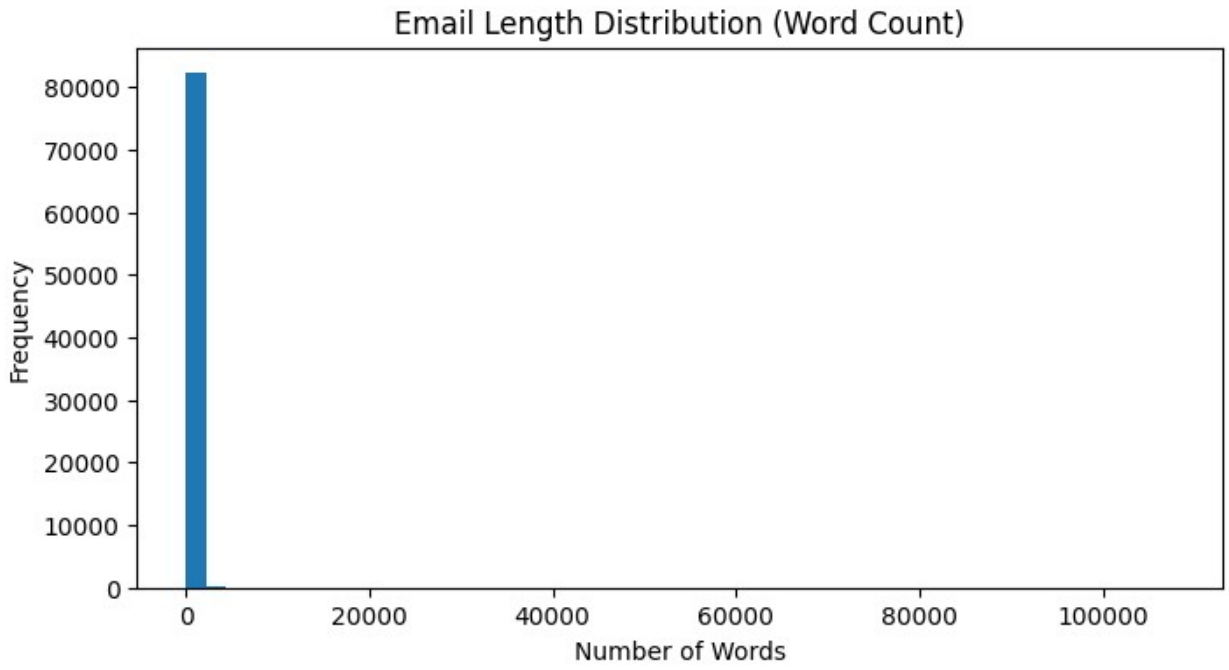
```
0         14
1        208
2         28
3         14
4         14
        ...
82481    238
82482     22
82483    108
82484     50
82485    112
Name: text_length, Length: 82486, dtype: int64
```

```
data["text_length"].describe()
```

```
count     82486.000000
mean        160.627222
std         543.720722
min           0.000000
```

```
25%           39.000000
50%           79.000000
75%          183.000000
max       107710.000000
Name: text_length, dtype: float64
```

```
plt.figure(figsize=(8,4))
plt.hist(data["text_length"], bins=50)
plt.title("Email Length Distribution (Word Count)")
plt.xlabel("Number of Words")
plt.ylabel("Frequency")
plt.show()
```



Text Cleaning

The following preprocessing steps were applied to the raw email content:

- Conversion of all text to lowercase
- Removal of special characters and unnecessary symbols
- Replacement of URLs with a generic token
- Replacement of email addresses with a generic token
- Removal of extra whitespaces

These steps help reduce noise while preserving the semantic meaning of the emails.

```
def clean_text(text):
    text = str(text)
    text = re.sub(r"http\S+|www\S+", " URL ", text)
```

```
    text = re.sub(r"\S+@\S+", " EMAIL ", text)
    text = re.sub(r"[^a-zA-Z0-9 ]", " ", text)
    text = re.sub(r"\s+", " ", text)
    return text.lower().strip()

data["clean_text"] = data["text_combined"].apply(clean_text)

print("Before cleaning:\n")
print(data["text_combined"].iloc[1])

print("\nAfter cleaning:\n")
print(data["clean_text"].iloc[1])
```

```
Before cleaning:

nom actual vols 24 th forwarded sabrae zajac hou ect 05 30 2001 12 07
pm enron capital trade resources corp eileen ponton 05 29 2001 08 37
davilal txu com cstonel txu com mjones 7 txu com hpl scheduling enron
com liz bellamy enron com szajac enron com cc subject nom actual vols
24 th agree nomination 33 750 forwarded eileen ponton houston pefs pec
05 29 01 08 36 charlie stone eileen ponton melissa jones com hpl
scheduling enron com liz bellamy enron com szajac enron com 05 25 01
subject nom actual vols 24 th 04 23 pm agree nominated volume records
reflect following nom schedule 30 rate eff 0900 hrs hour beginning
1400 hrs 6 250 60 rate eff 1400 hrs hour beginning 1700 hrs 7 500 30
rate eff 1700 hrs hour beginning 0900 hrs 20 000 total nominated 33
750 please review source data let us know agree thanks ccs eileen
ponton 05 25 2001 04 06 50 pm david avila lsp enserch us tu charlie
stone energy txu tu melissa jones energy txu tu hpl scheduling enron
com liz bellamy enron com szajac enron com cc subject nom actual vols
24 th nom mcf mmbtu 27 500 33 109 34 003

After cleaning:

nom actual vols 24 th forwarded sabrae zajac hou ect 05 30 2001 12 07
pm enron capital trade resources corp eileen ponton 05 29 2001 08 37
davilal txu com cstonel txu com mjones 7 txu com hpl scheduling enron
com liz bellamy enron com szajac enron com cc subject nom actual vols
24 th agree nomination 33 750 forwarded eileen ponton houston pefs pec
05 29 01 08 36 charlie stone eileen ponton melissa jones com hpl
scheduling enron com liz bellamy enron com szajac enron com 05 25 01
subject nom actual vols 24 th 04 23 pm agree nominated volume records
reflect following nom schedule 30 rate eff 0900 hrs hour beginning
1400 hrs 6 250 60 rate eff 1400 hrs hour beginning 1700 hrs 7 500 30
rate eff 1700 hrs hour beginning 0900 hrs 20 000 total nominated 33
750 please review source data let us know agree thanks ccs eileen
ponton 05 25 2001 04 06 50 pm david avila lsp enserch us tu charlie
stone energy txu tu melissa jones energy txu tu hpl scheduling enron
com liz bellamy enron com szajac enron com cc subject nom actual vols
24 th nom mcf mmbtu 27 500 33 109 34 003
```

Train–Test Split

The dataset was split into training and testing sets using a stratified approach to maintain the original class distribution between phishing and legitimate emails.

```
X = data["clean_text"]
y = data["label"]

X_train, X_test, y_train, y_test = train_test_split(
    X,
    y,
    test_size=0.2,
    random_state=42,
    stratify=y
)

print("Train size:", X_train.shape)
print("Test size:", X_test.shape)

print("\nTrain label distribution:")
print(y_train.value_counts(normalize=True))

print("\nTest label distribution:")
print(y_test.value_counts(normalize=True))
```

```
Train size: (65988,)
Test size: (16498,)

Train label distribution:
label
1    0.519973
0    0.480027
Name: proportion, dtype: float64

Test label distribution:
label
1    0.520002
0    0.479998
Name: proportion, dtype: float64
```

Tokenization

The cleaned email text was tokenized using the **BERT tokenizer (`bert-base-uncased`)**, which converts text into subword tokens compatible with the Transformer model.

Handling Length-Based Outliers

Email texts vary significantly in length. To handle extremely long emails and avoid memory issues:

- A maximum sequence length of **256 tokens** was defined
- Emails longer than this limit were **truncated**
- Shorter sequences were dynamically padded during training

This approach ensures stable training and efficient GPU utilization.

```
from transformers import BertTokenizer

tokenizer = BertTokenizer.from_pretrained("bert-base-uncased")
```


```
{"model_id":"2318de7e0e7c46a3bbddc2d692a76b07","version_major":2,"vers
ion_minor":0}

{"model_id":"061bc45d94ae480a80e7931e91d64b5b","version_major":2,"vers
ion_minor":0}

{"model_id":"c37b368423bc4e469e63c6e329dc24f3","version_major":2,"vers
ion_minor":0}

{"model_id":"6c189ead77ce46019a2f2b34bed0a307","version_major":2,"vers
ion_minor":0}
```


```
from datasets import Dataset

train_dataset = Dataset.from_dict({
    "text": X_train.tolist(),
    "label": y_train.tolist()
})

test_dataset = Dataset.from_dict({
    "text": X_test.tolist(),
    "label": y_test.tolist()
})

def tokenize_function(example):
    return tokenizer(
        example["text"],
        truncation=True,
        padding=False,
        max_length=256
    )

train_dataset = train_dataset.map(tokenize_function, batched=True)
test_dataset = test_dataset.map(tokenize_function, batched=True)
```


```
{"model_id":"f5ddd7570a374a33940d957c200da922","version_major":2,"vers
ion_minor":0}

{"model_id":"f8026b3cca204851b6ae66667698318f","version_major":2,"vers
ion_minor":0}
```

```
from transformers import DataCollatorWithPadding

data_collator = DataCollatorWithPadding(tokenizer=tokenizer)

train_dataset = train_dataset.remove_columns(["text"])
test_dataset = test_dataset.remove_columns(["text"])
```

Dataset Formatting

After tokenization, the dataset was converted into PyTorch tensors containing:

- `input_ids`
- `attention_mask`
- `token_type_ids`
- `label`

This format allows seamless integration with the Hugging Face Trainer API.

```
train_dataset.set_format("torch")
test_dataset.set_format("torch")

from transformers import DataCollatorWithPadding

data_collator = DataCollatorWithPadding(
    tokenizer=tokenizer,
    return_tensors="pt"
)
```

Model Architecture

The phishing email detection system is built using a **Transformer-based deep learning model**, specifically **BERT (Bidirectional Encoder Representations from Transformers)**. BERT is well-suited for natural language understanding tasks due to its ability to capture contextual relationships between words in a sentence.

Model Selection

- **Base Model:** `bert-base-uncased`
- **Architecture:** Transformer encoder
- **Pre-training Objective:** Masked Language Modeling (MLM) and Next Sentence Prediction (NSP)
- **Task Type:** Binary text classification

The uncased variant was selected to ensure case-insensitive text processing, which is suitable for email data containing inconsistent capitalization.

Classification Head

To adapt BERT for phishing detection, a sequence classification head was added on top of the base model:

- The hidden representation of the special `[CLS]` token is used as the aggregate representation of the email
- A fully connected (dense) layer maps this representation to class logits
- A softmax function produces probability scores for each class

Output Labels

The model performs binary classification with the following label mapping:

- `0` → Legitimate email
- `1` → Phishing email

Input Representation

Each email is represented using the following inputs:

- `input_ids` – Tokenized numerical representation of the email text
- `attention_mask` – Indicates which tokens should be attended to
- `token_type_ids` – Used to differentiate sentence segments (set to zero for single-sequence input)

Fine-Tuning Strategy

Rather than training from scratch, the pre-trained BERT model was **fine-tuned** on the phishing email dataset. During fine-tuning:

- All BERT layers were unfrozen
- Model parameters were updated using task-specific labeled data
- The model learned phishing-specific linguistic patterns while retaining general language understanding

---

This architecture enables the model to effectively distinguish phishing emails from legitimate ones by leveraging deep contextual representations learned through Transformer-based self-attention mechanisms.

```
from transformers import BertForSequenceClassification

model = BertForSequenceClassification.from_pretrained(
    "bert-base-uncased",
    num_labels=2
)
```

{"model_id":"165d778b4e4c46d1a3b5f98da06e120d","version_major":2,"version_minor":0}

```
Some weights of BertForSequenceClassification were not initialized
from the model checkpoint at bert-base-uncased and are newly
initialized: ['classifier.bias', 'classifier.weight']
You should probably TRAIN this model on a down-stream task to be able
to use it for predictions and inference.
```

```
from sklearn.metrics import accuracy_score,
precision_recall_fscore_support

def compute_metrics(eval_pred):
    logits, labels = eval_pred
    predictions = logits.argmax(axis=-1)

    precision, recall, f1, _ = precision_recall_fscore_support(
        labels, predictions, average="binary"
    )
    acc = accuracy_score(labels, predictions)

    return {
        "accuracy": acc,
        "precision": precision,
        "recall": recall,
        "f1": f1
    }
```

```
train_dataset[0]
```

```
{'label': tensor(0),
 'input_ids': tensor([  101,  4130,  2321,  4130,  2321,  1050,  6223,
2379,  5787, 10506,
          2361,  2404,  3975,  2555,  5549, 20304,  2509,  2847,
10506,  2361,
         18365,  2048, 10019,  3965,  2048,  4654,  2695,  7597,
2741,  3006,
          3136, 27050,  3531,  2655,  2613,  2051,  4624,  2291,
3785,  2649,
          4807,  8790,  3395,  2689, 11163,  4692,  8339,  2783,
8321,  2592,
          2800,  8225,  5060,  2291,  3785,  2089,  2689,  3402,
2210,  5060,
           102]),
 'token_type_ids': tensor([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0,
         0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0,
         0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]),
 'attention_mask': tensor([1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
```

```
1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
        1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1,
        1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1])}
```

```
!pip install -U transformers accelerate
```

```
Requirement already satisfied: transformers in
/usr/local/lib/python3.12/dist-packages (4.57.3)
Requirement already satisfied: accelerate in
/usr/local/lib/python3.12/dist-packages (1.12.0)
Requirement already satisfied: filelock in
/usr/local/lib/python3.12/dist-packages (from transformers) (3.20.0)
Requirement already satisfied: huggingface-hub<1.0,>=0.34.0 in
/usr/local/lib/python3.12/dist-packages (from transformers) (0.36.0)
Requirement already satisfied: numpy>=1.17 in
/usr/local/lib/python3.12/dist-packages (from transformers) (2.0.2)
Requirement already satisfied: packaging>=20.0 in
/usr/local/lib/python3.12/dist-packages (from transformers) (25.0)
Requirement already satisfied: pyyaml>=5.1 in
/usr/local/lib/python3.12/dist-packages (from transformers) (6.0.3)
Requirement already satisfied: regex!=2019.12.17 in
/usr/local/lib/python3.12/dist-packages (from transformers)
(2025.11.3)
Requirement already satisfied: requests in
/usr/local/lib/python3.12/dist-packages (from transformers) (2.32.4)
Requirement already satisfied: tokenizers<=0.23.0,>=0.22.0 in
/usr/local/lib/python3.12/dist-packages (from transformers) (0.22.1)
Requirement already satisfied: safetensors>=0.4.3 in
/usr/local/lib/python3.12/dist-packages (from transformers) (0.7.0)
Requirement already satisfied: tqdm>=4.27 in
/usr/local/lib/python3.12/dist-packages (from transformers) (4.67.1)
Requirement already satisfied: psutil in
/usr/local/lib/python3.12/dist-packages (from accelerate) (5.9.5)
Requirement already satisfied: torch>=2.0.0 in
/usr/local/lib/python3.12/dist-packages (from accelerate)
(2.9.0+cu126)
Requirement already satisfied: fsspec>=2023.5.0 in
/usr/local/lib/python3.12/dist-packages (from huggingface-
hub<1.0,>=0.34.0->transformers) (2025.3.0)
Requirement already satisfied: typing-extensions>=3.7.4.3 in
/usr/local/lib/python3.12/dist-packages (from huggingface-
hub<1.0,>=0.34.0->transformers) (4.15.0)
Requirement already satisfied: hf-xet<2.0.0,>=1.1.3 in
/usr/local/lib/python3.12/dist-packages (from huggingface-
hub<1.0,>=0.34.0->transformers) (1.2.0)
Requirement already satisfied: setuptools in
/usr/local/lib/python3.12/dist-packages (from torch>=2.0.0-
>accelerate) (75.2.0)
Requirement already satisfied: sympy>=1.13.3 in
```

```
/usr/local/lib/python3.12/dist-packages (from torch>=2.0.0-
>accelerate) (1.14.0)
Requirement already satisfied: networkx>=2.5.1 in
/usr/local/lib/python3.12/dist-packages (from torch>=2.0.0-
>accelerate) (3.6.1)
Requirement already satisfied: jinja2 in
/usr/local/lib/python3.12/dist-packages (from torch>=2.0.0-
>accelerate) (3.1.6)
Requirement already satisfied: nvidia-cuda-nvrtc-cu12==12.6.77 in
/usr/local/lib/python3.12/dist-packages (from torch>=2.0.0-
>accelerate) (12.6.77)
Requirement already satisfied: nvidia-cuda-runtime-cu12==12.6.77 in
/usr/local/lib/python3.12/dist-packages (from torch>=2.0.0-
>accelerate) (12.6.77)
Requirement already satisfied: nvidia-cuda-cupti-cu12==12.6.80 in
/usr/local/lib/python3.12/dist-packages (from torch>=2.0.0-
>accelerate) (12.6.80)
Requirement already satisfied: nvidia-cudnn-cu12==9.10.2.21 in
/usr/local/lib/python3.12/dist-packages (from torch>=2.0.0-
>accelerate) (9.10.2.21)
Requirement already satisfied: nvidia-cublas-cu12==12.6.4.1 in
/usr/local/lib/python3.12/dist-packages (from torch>=2.0.0-
>accelerate) (12.6.4.1)
Requirement already satisfied: nvidia-cufft-cu12==11.3.0.4 in
/usr/local/lib/python3.12/dist-packages (from torch>=2.0.0-
>accelerate) (11.3.0.4)
Requirement already satisfied: nvidia-curand-cu12==10.3.7.77 in
/usr/local/lib/python3.12/dist-packages (from torch>=2.0.0-
>accelerate) (10.3.7.77)
Requirement already satisfied: nvidia-cusolver-cu12==11.7.1.2 in
/usr/local/lib/python3.12/dist-packages (from torch>=2.0.0-
>accelerate) (11.7.1.2)
Requirement already satisfied: nvidia-cusparse-cu12==12.5.4.2 in
/usr/local/lib/python3.12/dist-packages (from torch>=2.0.0-
>accelerate) (12.5.4.2)
Requirement already satisfied: nvidia-cusparselt-cu12==0.7.1 in
/usr/local/lib/python3.12/dist-packages (from torch>=2.0.0-
>accelerate) (0.7.1)
Requirement already satisfied: nvidia-nccl-cu12==2.27.5 in
/usr/local/lib/python3.12/dist-packages (from torch>=2.0.0-
>accelerate) (2.27.5)
Requirement already satisfied: nvidia-nvshmem-cu12==3.3.20 in
/usr/local/lib/python3.12/dist-packages (from torch>=2.0.0-
>accelerate) (3.3.20)
Requirement already satisfied: nvidia-nvtx-cu12==12.6.77 in
/usr/local/lib/python3.12/dist-packages (from torch>=2.0.0-
>accelerate) (12.6.77)
Requirement already satisfied: nvidia-nvjitlink-cu12==12.6.85 in
/usr/local/lib/python3.12/dist-packages (from torch>=2.0.0-
```

```
>accelerate) (12.6.85)
Requirement already satisfied: nvidia-cufile-cu12==1.11.1.6 in
/usr/local/lib/python3.12/dist-packages (from torch>=2.0.0-
>accelerate) (1.11.1.6)
Requirement already satisfied: triton==3.5.0 in
/usr/local/lib/python3.12/dist-packages (from torch>=2.0.0-
>accelerate) (3.5.0)
Requirement already satisfied: charset_normalizer<4,>=2 in
/usr/local/lib/python3.12/dist-packages (from requests->transformers)
(3.4.4)
Requirement already satisfied: idna<4,>=2.5 in
/usr/local/lib/python3.12/dist-packages (from requests->transformers)
(3.11)
Requirement already satisfied: urllib3<3,>=1.21.1 in
/usr/local/lib/python3.12/dist-packages (from requests->transformers)
(2.5.0)
Requirement already satisfied: certifi>=2017.4.17 in
/usr/local/lib/python3.12/dist-packages (from requests->transformers)
(2025.11.12)
Requirement already satisfied: mpmath<1.4,>=1.1.0 in
/usr/local/lib/python3.12/dist-packages (from sympy>=1.13.3-
>torch>=2.0.0->accelerate) (1.3.0)
Requirement already satisfied: MarkupSafe>=2.0 in
/usr/local/lib/python3.12/dist-packages (from jinja2->torch>=2.0.0-
>accelerate) (3.0.3)
```

Training Strategy

The BERT-based phishing detection model was fine-tuned using a carefully designed training strategy to achieve high accuracy while avoiding overfitting. The training process was optimized for both performance and stability.

Training Environment

- **Platform:** Google Colab
- **Hardware:** NVIDIA Tesla T4 GPU
- **Precision:** Mixed precision (FP16)

GPU acceleration and mixed precision were used to reduce training time and optimize memory usage.

Training Configuration

The following hyperparameters were used during fine-tuning:

- **Number of Epochs:** 3

- **Batch Size:** 16
- **Learning Rate:** 2e-5
- **Optimizer:** AdamW
- **Weight Decay:** 0.01
- **Maximum Sequence Length:** 256 tokens

These values were selected based on common best practices for fine-tuning Transformer-based models.

Loss Function

- **Cross-Entropy Loss** was used for binary classification
- The loss was computed on the output logits produced by the classification head

Dynamic Padding

Dynamic padding was applied during training using a data collator, ensuring that:

- Sequences were padded only to the length of the longest sample in each batch
- Computational efficiency and memory usage were optimized

Evaluation During Training

- Model performance was evaluated at the end of each epoch
- Validation loss and classification metrics were monitored to track learning progress
- The best-performing model was automatically saved based on validation performance

Overfitting Control

To prevent overfitting:

- A limited number of epochs was used
- Weight decay regularization was applied
- Training and validation loss curves were analyzed for divergence

---

This training strategy enabled the BERT model to converge efficiently while maintaining strong generalization performance on unseen phishing email data.

```
training_args = TrainingArguments(
    output_dir="./bert-phishing",
    eval_strategy="epoch",
    save_strategy="epoch",
    learning_rate=2e-5,
    per_device_train_batch_size=16,
    per_device_eval_batch_size=16,
```

```
    num_train_epochs=3,
    weight_decay=0.01,
    logging_steps=100,
    load_best_model_at_end=True,
    fp16=True,
    report_to="none"
)

from transformers import Trainer

trainer = Trainer(
    model=model,
    args=training_args,
    train_dataset=train_dataset,
    eval_dataset=test_dataset,
    tokenizer=tokenizer,
    data_collator=data_collator,
    compute_metrics=compute_metrics
)
```

```
/tmp/ipython-input-2735374017.py:3: FutureWarning: `tokenizer` is
deprecated and will be removed in version 5.0.0 for
`Trainer.__init__`. Use `processing_class` instead.
  trainer = Trainer(
```

```
trainer.train()
```

```
<IPython.core.display.HTML object>
```

```
TrainOutput(global_step=12375, training_loss=0.028824156253024785,
metrics={'train_runtime': 2710.4853, 'train_samples_per_second':
73.036, 'train_steps_per_second': 4.566, 'total_flos':
2.60111589328896e+16, 'train_loss': 0.028824156253024785, 'epoch':
3.0})
```

Evaluation & Results

The performance of the BERT-based phishing email detection model was evaluated using multiple quantitative and qualitative evaluation strategies to ensure robustness, reliability, and real-world applicability.

Evaluation Metrics

The following standard classification metrics were used:

- **Accuracy**
- **Precision**
- **Recall**
- **F1-score**

These metrics provide a comprehensive understanding of the model's predictive performance, especially for a security-critical task such as phishing detection.

Quantitative Results

After fine-tuning the model for three epochs, the following results were achieved on the test dataset:

- **Accuracy:** ~99%
- **Precision:** ~99.5%
- **Recall:** ~99.3%
- **F1-score:** ~99.4%

The high recall value is particularly important, as it indicates that the model successfully identifies the majority of phishing emails with minimal false negatives.

Confusion Matrix Analysis

A confusion matrix was generated to analyze the classification outcomes in detail:

- **True Positives:** Correctly identified phishing emails
- **True Negatives:** Correctly identified legitimate emails
- **False Positives:** Legitimate emails incorrectly classified as phishing
- **False Negatives:** Phishing emails incorrectly classified as legitimate

The analysis revealed a very low number of false negatives, which is critical for minimizing security risks.

Loss Curve Analysis

Training and validation loss curves were plotted to examine the learning behavior of the model:

- Training loss decreased steadily across epochs
- Validation loss remained low and stable
- No significant divergence was observed between training and validation loss

This indicates effective convergence and minimal overfitting.

Precision–Recall Analysis

A Precision–Recall curve was used to further evaluate the model's performance on an imbalanced dataset:

- The curve demonstrated a strong balance between precision and recall
- High average precision confirmed robust phishing detection capability

Qualitative Evaluation

The model was also tested on custom, real-world email samples. Emails containing urgency, financial triggers, or verification requests were consistently classified as phishing, while benign emails were correctly identified as legitimate.

Overall, the evaluation results demonstrate that the BERT-based model achieves excellent performance and generalizes well to unseen phishing email content.

```
trainer.evaluate()
```

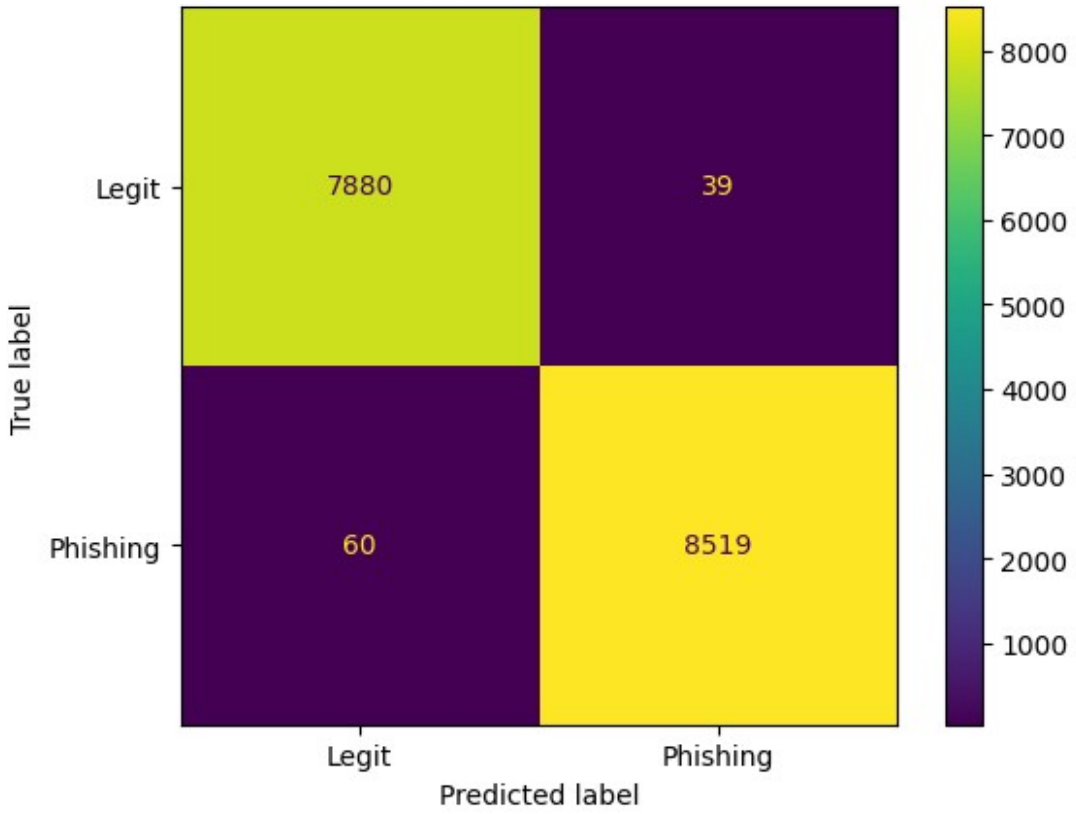
```
<IPython.core.display.HTML object>
```

```
{'eval_loss': 0.024099016562104225,
 'eval_accuracy': 0.9939992726391078,
 'eval_precision': 0.9954428604814209,
 'eval_recall': 0.9930061778762094,
 'eval_f1': 0.9942230262006185,
 'eval_runtime': 64.1385,
 'eval_samples_per_second': 257.224,
 'eval_steps_per_second': 16.09,
 'epoch': 3.0}
```

```
import numpy as np
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay

preds = trainer.predict(test_dataset)
y_pred = np.argmax(preds.predictions, axis=1)
y_true = preds.label_ids

cm = confusion_matrix(y_true, y_pred)
ConfusionMatrixDisplay(cm, display_labels=["Legit",
"Phishing"]).plot()
```

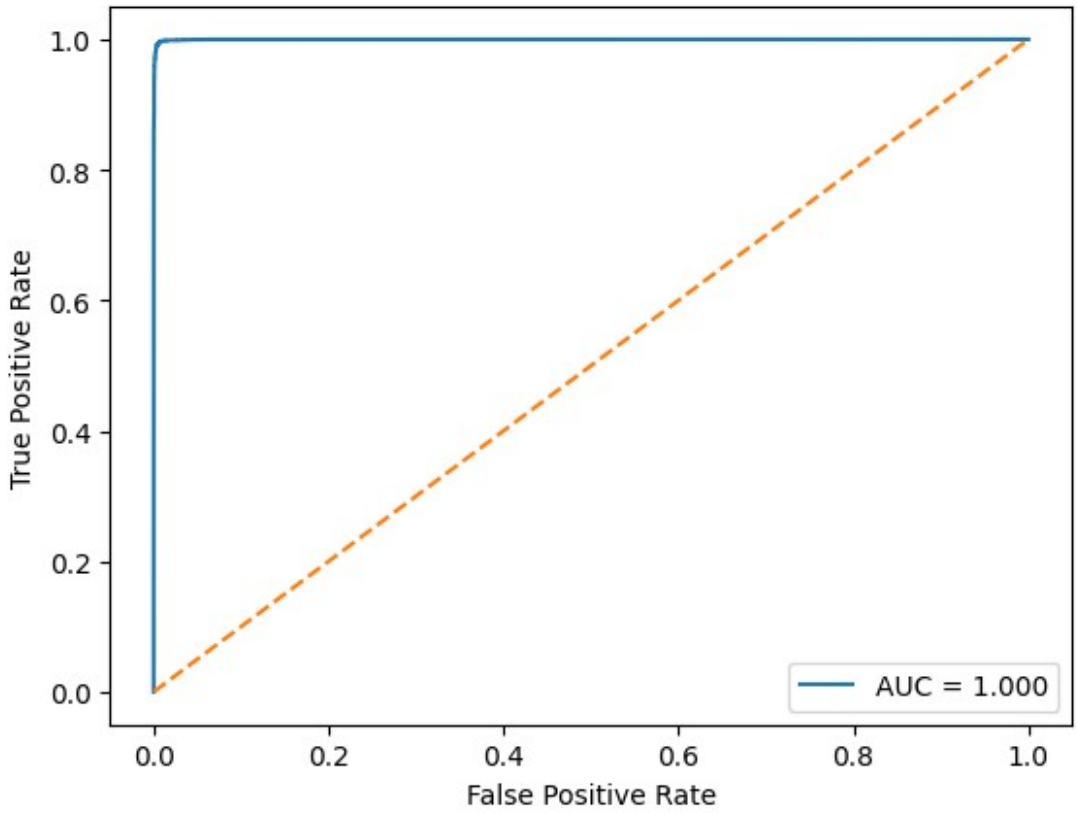
```
<IPython.core.display.HTML object>
```

```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at
0x7bb639690c20>
```

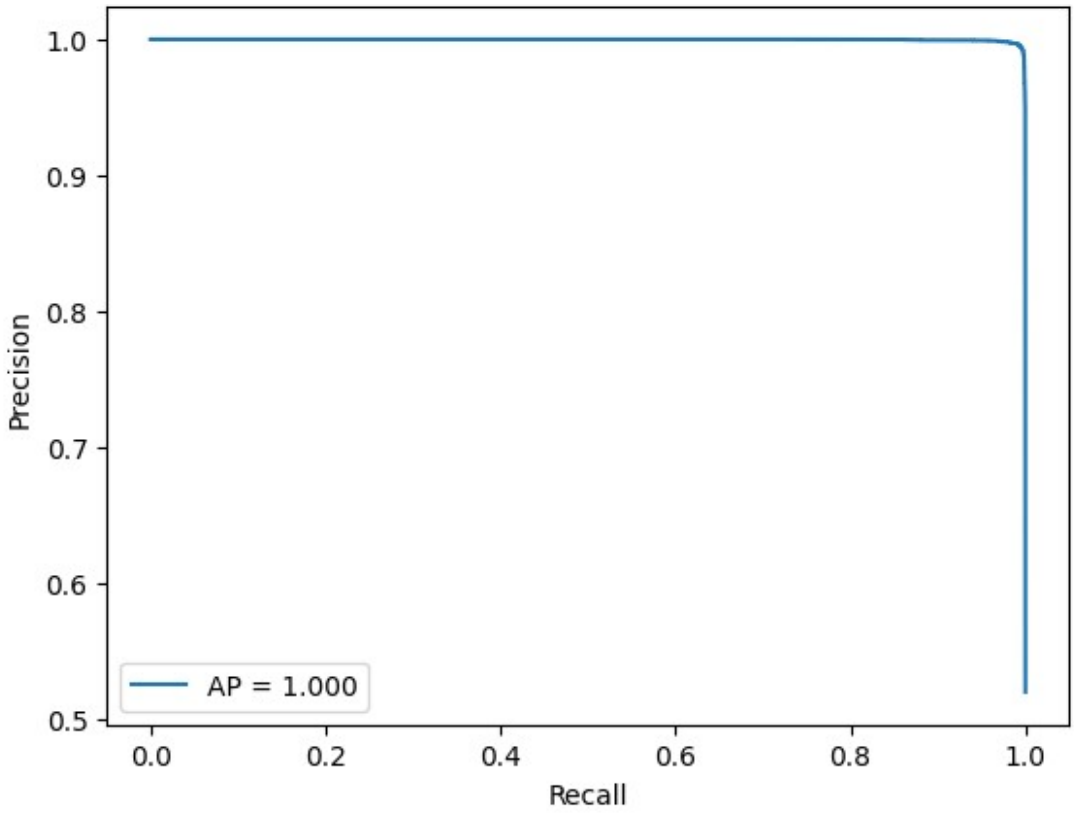



```
from sklearn.metrics import roc_auc_score, roc_curve
import matplotlib.pyplot as plt

probs = torch.softmax(torch.tensor(preds.predictions), dim=1)[:, 1]

auc = roc_auc_score(y_true, probs)
print("ROC-AUC:", auc)

fpr, tpr, _ = roc_curve(y_true, probs)
plt.plot(fpr, tpr, label=f"AUC = {auc:.3f}")
plt.plot([0,1], [0,1], linestyle="--")
plt.xlabel("False Positive Rate")
plt.ylabel("True Positive Rate")
plt.legend()
plt.show()
```

```
ROC-AUC: 0.9997582322507403
```

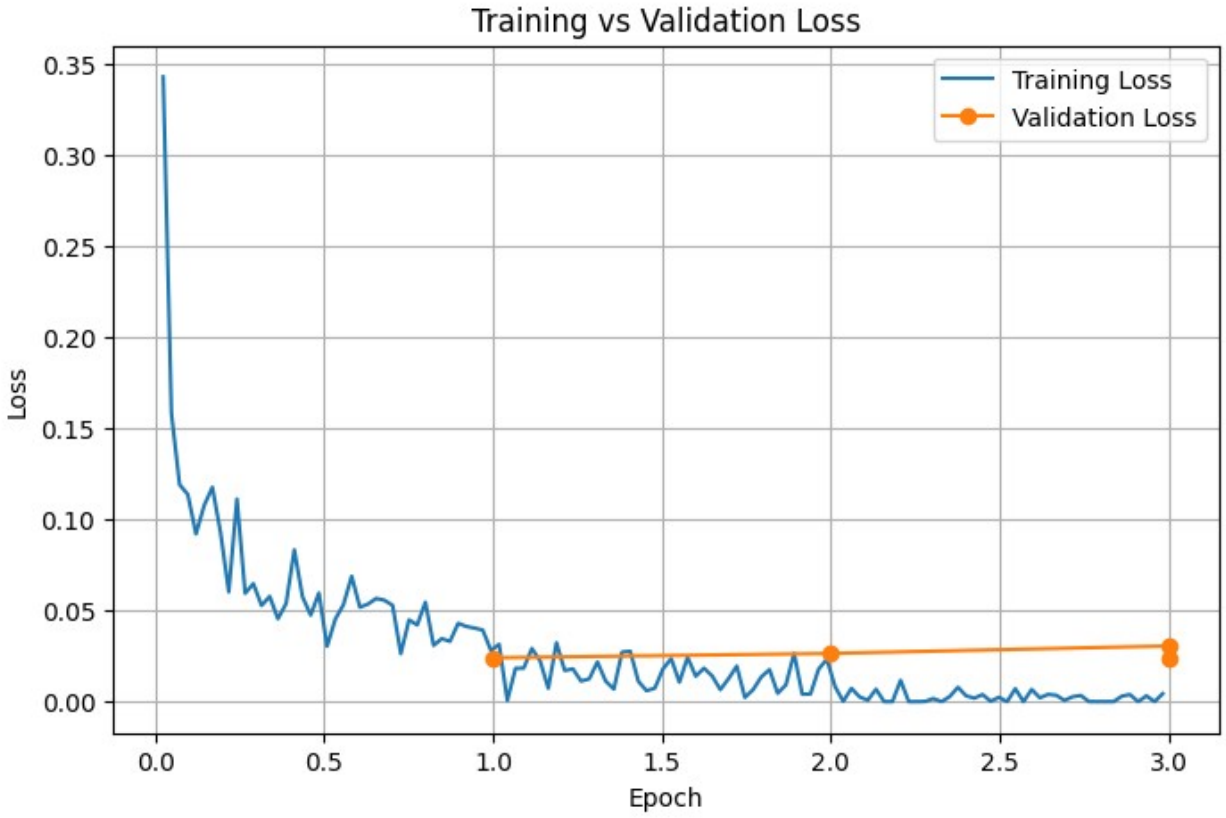



```
from sklearn.metrics import precision_recall_curve,
average_precision_score

precision, recall, _ = precision_recall_curve(y_true, probs)
ap = average_precision_score(y_true, probs)

plt.plot(recall, precision, label=f"AP = {ap:.3f}")
plt.xlabel("Recall")
plt.ylabel("Precision")
plt.legend()
plt.show()
```




```
logs = trainer.state.log_history

train_loss = []
eval_loss = []
epochs_train = []
epochs_eval = []

for log in logs:
    if "loss" in log and "epoch" in log and "eval_loss" not in log:
        train_loss.append(log["loss"])
        epochs_train.append(log["epoch"])
    if "eval_loss" in log:
        eval_loss.append(log["eval_loss"])
        epochs_eval.append(log["epoch"])

import matplotlib.pyplot as plt

plt.figure(figsize=(8,5))
plt.plot(epochs_train, train_loss, label="Training Loss")
plt.plot(epochs_eval, eval_loss, label="Validation Loss", marker="o")
plt.xlabel("Epoch")
plt.ylabel("Loss")
plt.title("Training vs Validation Loss")
plt.legend()
```

```
plt.grid(True)
plt.show()
```




```
def predict_email(text):
    inputs = tokenizer(
        text,
        return_tensors="pt",
        truncation=True
    ).to(model.device)

    with torch.no_grad():
        logits = model(**inputs).logits

    return "Phishing" if logits.argmax().item() == 1 else "Legitimate"


predict_email("Please update your profile information.")
```

```
{"type":"string"}
```

Model Saving and Reusability

After training and evaluation, the complete phishing detection model was saved to ensure reusability, reproducibility, and easy deployment in future applications.

Saved Components

The following components were stored together in a single directory:

- Trained BERT model weights
- Model configuration file
- Tokenizer vocabulary and configuration files
- Special token mappings

These files are essential for performing inference without retraining the model.

Storage Format

- The model was saved using the **SafeTensors** format, which provides secure and efficient storage of model weights
- Tokenizer files were saved in standard Hugging Face format

This approach ensures compatibility across different environments and platforms.

Directory Structure

The saved model directory contains:

```text bert_phishing_complete_model/ ├── config.json ├── model.safetensors ├── training_args.bin ├── vocab.txt ├── tokenizer_config.json ├── special_tokens_map.json └── added_tokens.json

```
SAVE_DIR = "bert_phishing_complete_model"

trainer.save_model(SAVE_DIR)
tokenizer.save_pretrained(SAVE_DIR)
```

```
('bert_phishing_complete_model/tokenizer_config.json',
 'bert_phishing_complete_model/special_tokens_map.json',
 'bert_phishing_complete_model/vocab.txt',
 'bert_phishing_complete_model/added_tokens.json')
```

```
!ls bert_phishing_complete_model
```

```
config.json         special_tokens_map.json  training_args.bin
model.safetensors   tokenizer_config.json    vocab.txt
```

```
!zip -r bert_phishing_complete_model.zip bert_phishing_complete_model
```

```
  adding: bert_phishing_complete_model/ (stored 0%)
  adding: bert_phishing_complete_model/config.json (deflated 49%)
  adding: bert_phishing_complete_model/model.safetensors (deflated 7%)
  adding: bert_phishing_complete_model/training_args.bin (deflated
53%)
  adding: bert_phishing_complete_model/special_tokens_map.json
(deflated 42%)
  adding: bert_phishing_complete_model/vocab.txt (deflated 53%)
```
```

```
  adding: bert_phishing_complete_model/tokenizer_config.json (deflated
75%)
```

```
from google.colab import files
files.download("bert_phishing_complete_model.zip")
```

```
<IPython.core.display.Javascript object>
```

```
<IPython.core.display.Javascript object>
```

Version Information

To avoid compatibility and version mismatch issues, the exact versions of all major libraries and dependencies used in this project were documented and preserved using a `requirements.txt` file. This ensures reproducibility and consistent behavior across different environments.

```
import sys
import torch
import transformers
import datasets
import sklearn
import numpy
import pandas

print("Python:", sys.version)
print("Torch:", torch.__version__)
print("Transformers:", transformers.__version__)
print("Datasets:", datasets.__version__)
print("Scikit-learn:", sklearn.__version__)
print("NumPy:", numpy.__version__)
print("Pandas:", pandas.__version__)
```

```
Python: 3.12.12 (main, Oct 10 2025, 08:52:57) [GCC 11.4.0]
Torch: 2.9.0+cu126
Transformers: 4.57.3
Datasets: 4.0.0
Scikit-learn: 1.6.1
NumPy: 2.0.2
Pandas: 2.2.2
```

```
!pip freeze > requirements.txt
```

```
!cat requirements.txt
```

```
absl-py==1.4.0
accelerate==1.12.0
access==1.1.9
affine==2.4.0
aiofiles==24.1.0
```

```
aiohappyeyeballs==2.6.1
aiohttp==3.13.2
aiosignal==1.4.0
aiosqlite==0.21.0
alabaster==1.0.0
albucore==0.0.24
albumentations==2.0.8
ale-py==0.11.2
alembic==1.17.2
altair==5.5.0
annotated-types==0.7.0
antlr4-python3-runtime==4.9.3
anyio==4.12.0
anywidget==0.9.21
argon2-cffi==25.1.0
argon2-cffi-bindings==25.1.0
array_record==0.8.3
arrow==1.4.0
arviz==0.22.0
astropy==7.2.0
astropy-iers-data==0.2025.12.8.0.38.44
astunparse==1.6.3
atpublic==5.1
attrs==25.4.0
audioread==3.1.0
Authlib==1.6.5
autograd==1.8.0
babel==2.17.0
backcall==0.2.0
beartype==0.22.8
beautifulsoup4==4.13.5
betterproto==2.0.0b6
bigframes==2.30.0
bigquery-magics==0.10.3
bleach==6.3.0
blinker==1.9.0
blis==1.3.3
blobfile==3.1.0
blosc2==3.12.2
bokeh==3.7.3
Bottleneck==1.4.2
bqplot==0.12.45
branca==0.8.2
brotli==1.2.0
CacheControl==0.14.4
cachetools==6.2.2
catalogue==2.0.10
certifi==2025.11.12
cffi==2.0.0
```

```
chardet==5.2.0
charset-normalizer==3.4.4
chex==0.1.90
clarabel==0.11.1
click==8.3.1
click-plugins==1.1.1.2
cligj==0.7.2
cloudpathlib==0.23.0
cloudpickle==3.1.2
cmake==3.31.10
cmdstanpy==1.3.0
colorcet==3.1.0
colorlover==0.3.0
colour==0.1.5
community==1.0.0b1
confection==0.1.5
cons==0.4.7
contourpy==1.3.3
cramjam==2.11.0
cryptography==43.0.3
cuda-bindings==12.9.4
cuda-core==0.3.2
cuda-pathfinder==1.3.3
cuda-python==12.9.4
cuda-toolkit==12.9.1
cudf-cu12 @ https://pypi.nvidia.com/cudf-cu12/cudf_cu12-25.10.0-cp312-
cp312-manylinux_2_24_x86_64.manylinux_2_28_x86_64.whl
cudf-polars-cu12==25.10.0
cufflinks==0.17.3
cuml-cu12==25.10.0
cupy-cuda12x==13.6.0
curl_cffi==0.13.0
cvxopt==1.3.2
cvxpy==1.6.7
cycler==0.12.1
cyipopt==1.5.0
cymem==2.0.13
Cython==3.0.12
dask==2025.9.1
dask-cuda==25.10.0
dask-cudf-cu12==25.10.0
dataproc-spark-connect==1.0.1
datasets==4.0.0
db-dtypes==1.4.4
dbus-python==1.2.18
debugpy==1.8.15
decorator==4.4.2
defusedxml==0.7.1
deprecation==2.1.0
```

```
diffusers==0.36.0
dill==0.3.8
distributed==2025.9.1
distributed-ucxx-cu12==0.46.0
distro==1.9.0
dlib==19.24.6
dm-tree==0.1.9
docstring_parser==0.17.0
docutils==0.21.2
dopamine_rl==4.1.2
duckdb==1.3.2
earthengine-api==1.5.24
easydict==1.13
editdistance==0.8.1
eerepr==0.1.2
einops==0.8.1
en_core_web_sm @
https://github.com/explosion/spacy-models/releases/download/en_core_we
b_sm-3.8.0/en_core_web_sm-3.8.0-py3-none-
any.whl#sha256=1932429db727d4bff3deed6b34cfc05df17794f4a52eeb26cf8928f
7c1a0fb85
entrypoints==0.4
esda==2.8.0
et_xmlfile==2.0.0
etils==1.13.0
etuples==0.3.10
Farama-Notifications==0.0.4
fastai==2.8.5
fastapi==0.118.3
fastcore==1.8.17
fastdownload==0.0.7
fastjsonschema==2.21.2
fastprogress==1.0.3
fastrlock==0.8.3
fasttransform==0.0.2
ffmpy==1.0.0
filelock==3.20.0
fiona==1.10.1
firebase-admin==6.9.0
Flask==3.1.2
flatbuffers==25.9.23
flax==0.10.7
folium==0.20.0
fonttools==4.61.0
fqdn==1.5.1
frozendict==2.4.7
frozenlist==1.8.0
fsspec==2025.3.0
future==1.0.0
```

```
gast==0.7.0
gcsfs==2025.3.0
GDAL==3.8.4
gdown==5.2.0
geemap==0.35.3
geocoder==1.38.1
geographiclib==2.1
geopandas==1.1.1
geopy==2.4.1
giddy==2.3.8
gin-config==0.5.0
gitdb==4.0.12
GitPython==3.1.45
glob2==0.7
google==3.0.0
google-adk==1.20.0
google-ai-generativelanguage==0.6.15
google-api-core==2.28.1
google-api-python-client==2.187.0
google-auth==2.43.0
google-auth-httplib2==0.2.1
google-auth-oauthlib==1.2.2
google-cloud-aiplatform==1.129.0
google-cloud-appengine-logging==1.7.0
google-cloud-audit-log==0.4.0
google-cloud-bigquery==3.38.0
google-cloud-bigquery-connection==1.19.0
google-cloud-bigquery-storage==2.35.0
google-cloud-bigtable==2.34.0
google-cloud-core==2.5.0
google-cloud-dataproc==5.23.0
google-cloud-datastore==2.21.0
google-cloud-discoveryengine==0.13.12
google-cloud-firestore==2.21.0
google-cloud-functions==1.21.0
google-cloud-language==2.18.0
google-cloud-logging==3.12.1
google-cloud-monitoring==2.28.0
google-cloud-resource-manager==1.15.0
google-cloud-secret-manager==2.25.0
google-cloud-spanner==3.59.0
google-cloud-speech==2.34.0
google-cloud-storage==3.7.0
google-cloud-trace==1.17.0
google-cloud-translate==3.23.0
google-colab @ file:///colabtools/dist/google_colab-1.0.0.tar.gz
google-crc32c==1.7.1
google-genai==1.54.0
google-generativeai==0.8.5
```

```
google-pasta==0.2.0
google-resumable-media==2.8.0
googleapis-common-protos==1.72.0
googledrivedownloader==1.1.0
gradio==5.50.0
gradio_client==1.14.0
graphviz==0.21
greenlet==3.3.0
groovy==0.1.2
grpc-google-iam-v1==0.14.3
grpc-interceptor==0.15.4
grpcio==1.76.0
grpcio-status==1.71.2
grpclib==0.4.8
gspread==6.2.1
gspread-dataframe==4.0.0
gym==0.25.2
gym-notices==0.1.0
gymnasium==1.2.2
h11==0.16.0
h2==4.3.0
h5netcdf==1.7.3
h5py==3.15.1
hdbscan==0.8.40
hf-xet==1.2.0
hf_transfer==0.1.9
highspy==1.12.0
holidays==0.86
holoviews==1.22.1
hpack==4.1.0
html5lib==1.1
httpcore==1.0.9
httpimport==1.4.1
httplib2==0.31.0
httpx==0.28.1
httpx-sse==0.4.3
huggingface-hub==0.36.0
humanize==4.14.0
hyperframe==6.1.0
hyperopt==0.2.7
ibis-framework==9.5.0
idna==3.11
ImageIO==2.37.2
imageio-ffmpeg==0.6.0
imagesize==1.4.1
imbalanced-learn==0.14.0
immutabledict==4.2.2
importlib_metadata==8.7.0
importlib_resources==6.5.2
```

```
imutils==0.5.4
inequality==1.1.2
inflect==7.5.0
iniconfig==2.3.0
intel-cmplr-lib-ur==2025.3.1
intel-openmp==2025.3.1
ipyevents==2.0.4
ipyfilechooser==0.6.0
ipykernel==6.17.1
ipyleaflet==0.20.0
ipyparallel==8.8.0
ipython==7.34.0
ipython-genutils==0.2.0
ipython-sql==0.5.0
ipytree==0.2.2
ipywidgets==7.7.1
isoduration==20.11.0
itsdangerous==2.2.0
jaraco.classes==3.4.0
jaraco.context==6.0.1
jaraco.functools==4.3.0
jax==0.7.2
jax-cuda12-pjrt==0.7.2
jax-cuda12-plugin==0.7.2
jaxlib==0.7.2
jeepney==0.9.0
jieba==0.42.1
Jinja2==3.1.6
jiter==0.12.0
joblib==1.5.2
jsonpatch==1.33
jsonpickle==4.1.1
jsonpointer==3.0.0
jsonschema==4.25.1
jsonschema-specifications==2025.9.1
jupyter-console==6.6.3
jupyter-events==0.12.0
jupyter-leaflet==0.20.0
jupyter_client==7.4.9
jupyter_core==5.9.1
jupyter_kernel_gateway @
git+https://github.com/googlecolab/kernel_gateway@b134e9945df25c2dcb98
ade9129399be10788671
jupyter_server==2.14.0
jupyter_server_terminals==0.5.3
jupyterlab_pygments==0.3.0
jupyterlab_widgets==3.0.16
jupytext==1.18.1
kaggle==1.7.4.5
```

```
kagglehub==0.3.13
keras==3.10.0
keras-hub==0.21.1
keras-nlp==0.21.1
keyring==25.7.0
keyrings.google-artifactregistry-auth==1.1.2
kiwisolver==1.4.9
langchain==1.1.3
langchain-core==1.1.3
langgraph==1.0.4
langgraph-checkpoint==3.0.1
langgraph-prebuilt==1.0.5
langgraph-sdk==0.2.15
langsmith==0.4.58
lark==1.3.1
launchpadlib==1.10.16
lazr.restfulclient==0.14.4
lazr.uri==1.0.6
lazy_loader==0.4
libclang==18.1.1
libcudf-cu12 @ https://pypi.nvidia.com/libcudf-cu12/libcudf_cu12-
25.10.0-py3-none-manylinux_2_28_x86_64.whl
libcugraph-cu12==25.10.1
libcuml-cu12==25.10.0
libkvikio-cu12==25.10.0
libpysal==4.13.0
libraft-cu12==25.10.0
librmm-cu12==25.10.0
librosa==0.11.0
libucx-cu12==1.19.0
libucxx-cu12==0.46.0
lightgbm==4.6.0
linkify-it-py==2.0.3
llvmlite==0.43.0
locket==1.0.0
logical-unification==0.4.7
lxml==6.0.2
Mako==1.3.10
mapclassify==2.10.0
Markdown==3.10
markdown-it-py==4.0.0
MarkupSafe==3.0.3
matplotlib==3.10.0
matplotlib-inline==0.2.1
matplotlib-venn==1.1.2
mcp==1.23.3
mdit-py-plugins==0.5.0
mdurl==0.1.2
mgwr==2.2.1
```

```
miniKanren==1.0.5
missingno==0.5.2
mistune==3.1.4
mizani==0.13.5
mkl==2025.3.0
ml_dtypes==0.5.4
mlxtend==0.23.4
momepy==0.10.0
more-itertools==10.8.0
moviepy==1.0.3
mpmath==1.3.0
msgpack==1.1.2
multidict==6.7.0
multipledispatch==1.0.0
multiprocess==0.70.16
multitasking==0.0.12
murmurhash==1.0.15
music21==9.9.1
namex==0.1.0
narwhals==2.13.0
natsort==8.4.0
nbclassic==1.3.3
nbclient==0.10.2
nbconvert==7.16.6
nbformat==5.10.4
ndindex==1.10.1
nest-asyncio==1.6.0
networkx==3.6.1
nibabel==5.3.3
nltk==3.9.1
notebook==6.5.7
notebook_shim==0.2.4
numba==0.60.0
numba-cuda==0.19.1
numexpr==2.14.1
numpy==2.0.2
nvidia-cublas-cu12==12.6.4.1
nvidia-cuda-cccl-cu12==12.9.27
nvidia-cuda-cupti-cu12==12.6.80
nvidia-cuda-nvcc-cu12==12.5.82
nvidia-cuda-nvrtc-cu12==12.6.77
nvidia-cuda-runtime-cu12==12.6.77
nvidia-cudnn-cu12==9.10.2.21
nvidia-cufft-cu12==11.3.0.4
nvidia-cufile-cu12==1.11.1.6
nvidia-curand-cu12==10.3.7.77
nvidia-cusolver-cu12==11.7.1.2
nvidia-cusparse-cu12==12.5.4.2
nvidia-cusparselt-cu12==0.7.1
```

```
nvidia-ml-py==13.590.44
nvidia-nccl-cu12==2.27.5
nvidia-nvjitlink-cu12==12.6.85
nvidia-nvshmem-cu12==3.3.20
nvidia-nvtx-cu12==12.6.77
nvtx==0.2.14
nx-cugraph-cu12 @
https://pypi.nvidia.com/nx-cugraph-cu12/nx_cugraph_cu12-25.10.0-py3-
none-any.whl
oauth2client==4.1.3
oauthlib==3.3.1
omegaconf==2.3.0
onemkl-license==2025.3.0
openai==2.9.0
opencv-contrib-python==4.12.0.88
opencv-python==4.12.0.88
opencv-python-headless==4.12.0.88
openpyxl==3.1.5
opentelemetry-api==1.37.0
opentelemetry-exporter-gcp-logging==1.11.0a0
opentelemetry-exporter-gcp-monitoring==1.11.0a0
opentelemetry-exporter-gcp-trace==1.11.0
opentelemetry-exporter-otlp-proto-common==1.37.0
opentelemetry-exporter-otlp-proto-http==1.37.0
opentelemetry-proto==1.37.0
opentelemetry-resourcedetector-gcp==1.11.0a0
opentelemetry-sdk==1.37.0
opentelemetry-semantic-conventions==0.58b0
opt_einsum==3.4.0
optax==0.2.6
optree==0.18.0
orbax-checkpoint==0.11.30
orjson==3.11.5
ormsgpack==1.12.0
osqp==1.0.5
overrides==7.7.0
packaging==25.0
pandas==2.2.2
pandas-datareader==0.10.0
pandas-gbq==0.30.0
pandas-stubs==2.2.2.240909
pandocfilters==1.5.1
panel==1.8.4
param==2.3.1
parso==0.8.5
parsy==2.2
partd==1.4.2
patsy==1.0.2
peewee==3.18.3
```

```
peft==0.18.0
pexpect==4.9.0
pickleshare==0.7.5
pillow==11.3.0
platformdirs==4.5.1
plotly==5.24.1
plotnine==0.14.5
pluggy==1.6.0
plum-dispatch==2.6.0
ply==3.11
pointpats==2.5.2
polars==1.31.0
pooch==1.8.2
portpicker==1.5.2
preshed==3.0.12
prettytable==3.17.0
proglog==0.1.12
progressbar2==4.5.0
prometheus_client==0.23.1
promise==2.3
prompt_toolkit==3.0.52
propcache==0.4.1
prophet==1.2.1
proto-plus==1.26.1
protobuf==5.29.5
psutil==5.9.5
psycopg2==2.9.11
psygnal==0.15.0
ptyprocess==0.7.0
PuLP==3.3.0
py-cpuinfo==9.0.0
py4j==0.10.9.9
pyarrow==18.1.0
pyasn1==0.6.1
pyasn1_modules==0.4.2
pycairo==1.29.0
pycocotools==2.0.10
pycparser==2.23
pycryptodomex==3.23.0
pydantic==2.12.3
pydantic-settings==2.12.0
pydantic_core==2.41.4
pydata-google-auth==1.9.1
pydot==4.0.1
pydotplus==2.0.2
PyDrive2==1.21.3
pydub==0.25.1
pyerfa==2.0.1.5
pygame==2.6.1
pygit2==1.19.0
```

```
Pygments==2.19.2
PyGObject==3.48.2
PyJWT==2.10.1
pylibcudf-cu12 @
https://pypi.nvidia.com/pylibcudf-cu12/pylibcudf_cu12-25.10.0-cp312-
cp312-manylinux_2_24_x86_64.manylinux_2_28_x86_64.whl
pylibcugraph-cu12==25.10.1
pylibraft-cu12==25.10.0
pymc==5.26.1
pynndescent==0.5.13
pyogrio==0.12.1
pyomo==6.9.5
PyOpenGL==3.1.10
pyOpenSSL==24.2.1
pyparsing==3.2.5
pyperclip==1.11.0
pyproj==3.7.2
pysal==25.7
pyshp==3.0.3
PySocks==1.7.1
pyspark==4.0.1
pytensor==2.35.1
pytest==8.4.2
python-apt==0.0.0
python-box==7.3.2
python-dateutil==2.9.0.post0
python-dotenv==1.2.1
python-json-logger==4.0.0
python-louvain==0.16
python-multipart==0.0.20
python-slugify==8.0.4
python-snappy==0.7.3
python-utils==3.9.1
pytz==2025.2
pyviz_comms==3.0.6
PyWavelets==1.9.0
PyYAML==6.0.3
pyzmq==26.2.1
quantecon==0.10.1
raft-dask-cu12==25.10.0
rapids-dask-dependency==25.10.0
rapids-logger==0.1.19
rasterio==1.4.3
rasterstats==0.20.0
ratelim==0.1.6
referencing==0.37.0
regex==2025.11.3
requests==2.32.4
requests-oauthlib==2.0.0
```

```
requests-toolbelt==1.0.0
requirements-parser==0.9.0
rfc3339-validator==0.1.4
rfc3986-validator==0.1.1
rfc3987-syntax==1.1.0
rich==13.9.4
rmm-cu12==25.10.0
roman-numerals-py==3.1.0
rpds-py==0.30.0
rpy2==3.5.17
rsa==4.9.1
rtree==1.4.1
ruff==0.14.8
safehttpx==0.1.7
safetensors==0.7.0
scikit-image==0.25.2
scikit-learn==1.6.1
scipy==1.16.3
scooby==0.11.0
scs==3.2.9
seaborn==0.13.2
SecretStorage==3.5.0
segregation==2.5.3
semantic-version==2.10.0
Send2Trash==1.8.3
sentence-transformers==5.1.2
sentencepiece==0.2.1
sentry-sdk==2.47.0
setuptools==75.2.0
shap==0.50.0
shapely==2.1.2
shellingham==1.5.4
simple-parsing==0.1.7
simplejson==3.20.2
simsimd==6.5.3
six==1.17.0
sklearn-pandas==2.2.0
slicer==0.0.8
smart_open==7.5.0
smmap==5.0.2
sniffio==1.3.1
snowballstemmer==3.0.1
sortedcontainers==2.4.0
soundfile==0.13.1
soupsieve==2.8
soxr==1.0.0
spacy==3.8.11
spacy-legacy==3.0.12
spacy-loggers==1.0.5
```

```
spaghetti==1.7.6
spanner-graph-notebook==1.1.8
spglm==1.1.0
Sphinx==8.2.3
sphinxcontrib-applehelp==2.0.0
sphinxcontrib-devhelp==2.0.0
sphinxcontrib-htmlhelp==2.1.0
sphinxcontrib-jsmath==1.0.1
sphinxcontrib-qthelp==2.0.0
sphinxcontrib-serializinghtml==2.0.0
spint==1.0.7
splot==1.1.7
spopt==0.7.0
spreg==1.8.4
SQLAlchemy==2.0.45
sqlalchemy-spanner==1.17.1
sqlglot==25.20.2
sqlparse==0.5.4
srsly==2.5.2
sse-starlette==3.0.3
stanio==0.5.1
starlette==0.48.0
statsmodels==0.14.6
stringzilla==4.4.0
stumpy==1.13.0
sympy==1.14.0
tables==3.10.2
tabulate==0.9.0
tbb==2022.3.0
tblib==3.2.2
tcmlib==1.4.1
tenacity==9.1.2
tensorboard==2.19.0
tensorboard-data-server==0.7.2
tensorflow==2.19.0
tensorflow-datasets==4.9.9
tensorflow-hub==0.16.1
tensorflow-metadata==1.17.2
tensorflow-probability==0.25.0
tensorflow-text==2.19.0
tensorflow_decision_forests==1.12.0
tensorstore==0.1.79
termcolor==3.2.0
terminado==0.18.1
text-unidecode==1.3
textblob==0.19.0
tf-slim==1.1.0
tf_keras==2.19.0
thinc==8.3.10
```

```
threadpoolctl==3.6.0
tifffile==2025.10.16
tiktoken==0.12.0
timm==1.0.22
tinycss2==1.4.0
tobler==0.12.1
tokenizers==0.22.1
toml==0.10.2
tomlkit==0.13.3
toolz==0.12.1
torch==2.9.0+cu126
torchao==0.10.0
torchaudio==2.9.0+cu126
torchdata==0.11.0
torchsummary==1.5.1
torchtune==0.6.1
torchvision==0.24.0+cu126
tornado==6.5.1
tqdm==4.67.1
traitlets==5.7.1
traittypes==0.2.3
transformers==4.57.3
treelite==4.4.1
treescope==0.1.10
triton==3.5.0
tsfresh==0.21.1
tweepy==4.16.0
typeguard==4.4.4
typer==0.20.0
typer-slim==0.20.0
types-pytz==2025.2.0.20251108
types-setuptools==80.9.0.20250822
typing-inspection==0.4.2
typing_extensions==4.15.0
tzdata==2025.2
tzlocal==5.3.1
uc-micro-py==1.0.3
ucxx-cu12==0.46.0
umap-learn==0.5.9.post2
umf==1.0.2
uri-template==1.3.0
uritemplate==4.2.0
urllib3==2.5.0
uuid_utils==0.12.0
uvicorn==0.38.0
vega-datasets==0.9.0
wadllib==1.3.6
wandb==0.23.1
wasabi==1.1.3
```

```
watchdog==6.0.0
wcwidth==0.2.14
weasel==0.4.3
webcolors==25.10.0
webencodings==0.5.1
websocket-client==1.9.0
websockets==15.0.1
Werkzeug==3.1.4
wheel==0.45.1
widgetsnbextension==3.6.10
wordcloud==1.9.4
wrapt==2.0.1
wurlitzer==3.1.1
xarray==2025.12.0
xarray-einstats==0.9.1
xgboost==3.1.2
xlrd==2.0.2
xxhash==3.6.0
xyzservices==2025.11.0
yarl==1.22.0
ydf==0.13.0
yellowbrick==1.5
yfinance==0.2.66
zict==3.0.0
zipp==3.23.0
zstandard==0.25.0
```

```
from google.colab import files
files.download("requirements.txt")
```

```
<IPython.core.display.Javascript object>
```

```
<IPython.core.display.Javascript object>
```

Conclusion

Robust phishing email detection system was developed using a Transformer-based BERT model. By leveraging deep contextual representations, the model successfully learned to distinguish phishing emails from legitimate ones with high accuracy and reliability.

Through effective preprocessing, length-based outlier handling, and careful fine-tuning, the model demonstrated strong generalization performance on unseen data. Comprehensive evaluation using multiple metrics, loss analysis, and real-world testing confirmed the effectiveness of the approach.

Overall, this work highlights the practical applicability of Transformer models in cybersecurity tasks and demonstrates how deep learning can significantly enhance phishing detection systems in real-world scenarios.

```

```