Configuration Manual

Classification of PII and Non PII files using Machine learning (NER) for DLP

MSc Research Project
MSc Cyber Security

Shivraj Prithviraj Mohite
Student ID: x21171327

School of Computing
National College of Ireland

Supervisor: Prof. Apurva Vangujar

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	Shivraj Prithviraj Mohite		
Student ID:	x21171327		
Programme:	MSc Cyber Security	**Year:**	2022-2023
Module:	Research Project		
Lecturer:	Prof. Apurva Vangujar		
Submission Due Date:	14th August 2023		
Project Title:	Classification of PII and Non PII files using Machine learning (NER) for DLP		
Word Count:	1407	**Page Count:**	**11**

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: Shivraj Prithviraj Mohite

Date: 14th August 2023

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

Classification of PII and Non PII files Using Machine learning (NER) for DLP

Shivraj Prithviraj Mohite
X21171327

1 Introduction

Welcome to the Configuration Manual for the PII and Non PII files Classifier Application. This guide empowers you to efficiently set up and utilize this tool, simplifying the process of analysing security vulnerabilities. In this manual we will guide you on how to train and finetune BERT model according to your dataset and detect PII and Non PII data when input is provided and later using the finetuned model to create a basic python application which could read a .txt file and determine whether it's a PII and Non PII file. In this configuration manual we will show from scratch how to install all the libraries needed, the software's and everything.

2 Configurations

2.1 Hardware

- Operating System: Windows 11
- Processor: Intel i7
- Architecture: 64bit
- Storage: 1TB SSD
- Memory: 16GB
- GPU: Nvidia RTX 3060

2.2 Software

- Visual Studio Code
- Python
- Google collab

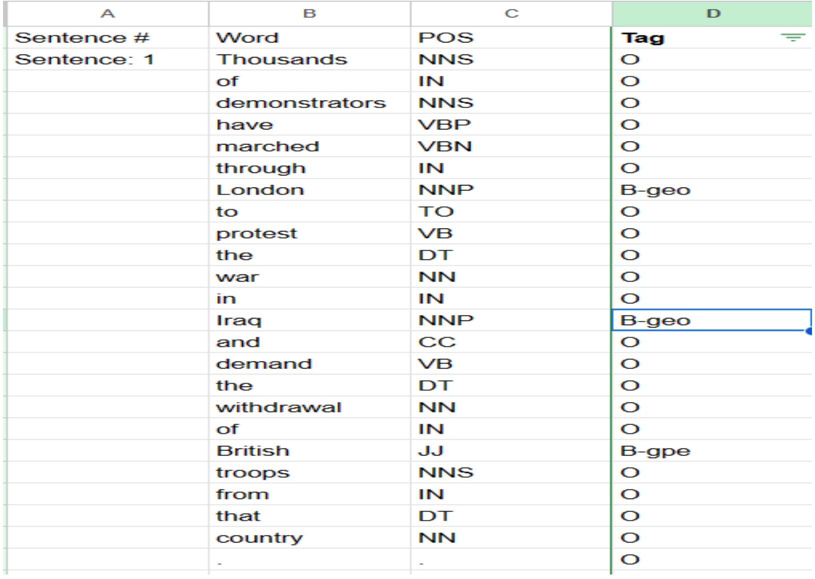
2.3 Model Used

- **BERT**

3 Implementation

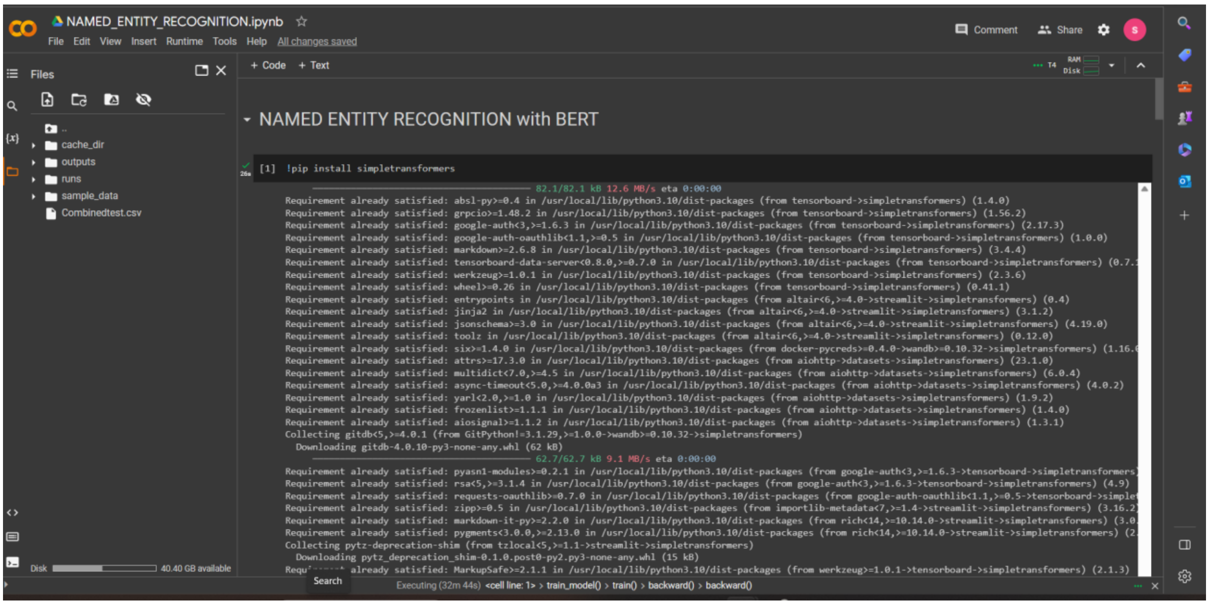
Prerequisites: Installing the latest python version along with Visual Studio Code IDE which is required for our implementation

Step 1: Creation of Dataset: Dataset consisted of 18lakh entries of finely labelled entities. The creation of the Dataset involved two steps, one was finding the format in which we wanted to create a dataset and then was to generate a dataset as we could not find one because PII data is not available on internet. A fake PII data generator code was developed using Faker library in python which created our data then we used AI tool to put that created data into a format which was suitable for BERT to be trained on. This finely labelled dataset of Non PII and PII data was used to train BERT model and achieve good accuracy.. Ill share you the dataset file where we have the content and their respective which are required to train or finetune the BERT model. Below is the image of dataset and its format:



A	B	C	D
Sentence #	Word	POS	Tag
Sentence: 1	Thousands	NNS	O
	of	IN	O
	demonstrators	NNS	O
	have	VBP	O
	marched	VBN	O
	through	IN	O
	London	NNP	B-geo
	to	TO	O
	protest	VB	O
	the	DT	O
	war	NN	O
	in	IN	O
	Iraq	NNP	B-geo
	and	CC	O
	demand	VB	O
	the	DT	O
	withdrawal	NN	O
	of	IN	O
	British	JJ	B-gpe
	troops	NNS	O
	from	IN	O
	that	DT	O
	country	NN	O
	.	.	O

Step 2: Open Google collab and import the Jupyter notebook file which I have uploaded: The name of the file is **Model_training.ipynb** which contains all the below code. This code is used to train of Pretrained BERT model to get good accuracy.



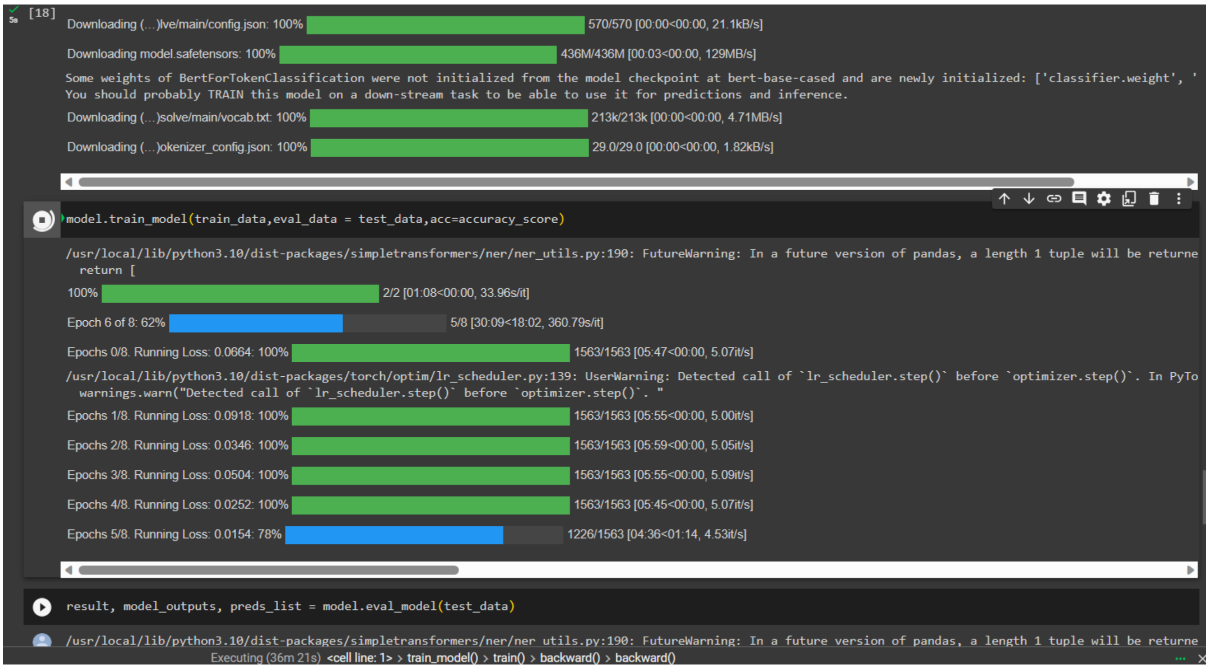
NAMED ENTITY RECOGNITION with BERT

```
[1] !pip install simpletransformers
```

```
                                          82.1/82.1 kB 12.6 MB/s eta 0:00:00
Requirement already satisfied: absl-py>=0.4 in /usr/local/lib/python3.10/dist-packages (from tensorboard->simpletransformers) (1.4.0)
Requirement already satisfied: grpcio>=1.48.2 in /usr/local/lib/python3.10/dist-packages (from tensorboard->simpletransformers) (1.56.2)
Requirement already satisfied: google-auth<3,>=1.6.3 in /usr/local/lib/python3.10/dist-packages (from tensorboard->simpletransformers) (2.17.3)
Requirement already satisfied: google-auth-oauthlib<1.1,>=0.5 in /usr/local/lib/python3.10/dist-packages (from tensorboard->simpletransformers) (1.0.0)
Requirement already satisfied: markdown>=2.6.8 in /usr/local/lib/python3.10/dist-packages (from tensorboard->simpletransformers) (3.4.4)
Requirement already satisfied: tensorboard-data-server<0.8.0,>=0.7.0 in /usr/local/lib/python3.10/dist-packages (from tensorboard->simpletransformers) (0.7.
Requirement already satisfied: werkzeug>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from tensorboard->simpletransformers) (2.3.6)
Requirement already satisfied: wheel>=0.26 in /usr/local/lib/python3.10/dist-packages (from tensorboard->simpletransformers) (0.41.1)
Requirement already satisfied: entrypoints in /usr/local/lib/python3.10/dist-packages (from altair<6,>=4.0->streamlit->simpletransformers) (0.4)
Requirement already satisfied: jinja2 in /usr/local/lib/python3.10/dist-packages (from altair<6,>=4.0->streamlit->simpletransformers) (3.1.2)
Requirement already satisfied: jsonschema>=3.0 in /usr/local/lib/python3.10/dist-packages (from altair<6,>=4.0->streamlit->simpletransformers) (4.19.0)
Requirement already satisfied: toolz in /usr/local/lib/python3.10/dist-packages (from altair<6,>=4.0->streamlit->simpletransformers) (0.12.0)
Requirement already satisfied: six>=1.4.0 in /usr/local/lib/python3.10/dist-packages (from docker-pycreds>=0.4.0->wandb>=0.10.32->simpletransformers) (1.16.
Requirement already satisfied: attrs>=17.3.0 in /usr/local/lib/python3.10/dist-packages (from aiohttp->datasets->simpletransformers) (23.1.0)
Requirement already satisfied: multidict<7.0,>=4.5 in /usr/local/lib/python3.10/dist-packages (from aiohttp->datasets->simpletransformers) (6.0.4)
Requirement already satisfied: async-timeout<5.0,>=4.0.0a3 in /usr/local/lib/python3.10/dist-packages (from aiohttp->datasets->simpletransformers) (4.0.2)
Requirement already satisfied: yarl<2.0,>=1.0 in /usr/local/lib/python3.10/dist-packages (from aiohttp->datasets->simpletransformers) (1.9.2)
Requirement already satisfied: frozenlist>=1.1.1 in /usr/local/lib/python3.10/dist-packages (from aiohttp->datasets->simpletransformers) (1.4.0)
Requirement already satisfied: aiosignal>=1.1.2 in /usr/local/lib/python3.10/dist-packages (from aiohttp->datasets->simpletransformers) (1.3.1)
Collecting gitdb<5,>=4.0.1 (from GitPython!=3.1.29,>=1.0.0->wandb>=0.10.32->simpletransformers)
  Downloading gitdb-4.0.10-py3-none-any.whl (62 kB)
                                          62.7/62.7 kB 9.1 MB/s eta 0:00:00
Requirement already satisfied: pyasn1-modules>=0.2.1 in /usr/local/lib/python3.10/dist-packages (from google-auth<3,>=1.6.3->tensorboard->simpletransformers
Requirement already satisfied: rsa<5,>=3.1.4 in /usr/local/lib/python3.10/dist-packages (from google-auth<3,>=1.6.3->tensorboard->simpletransformers) (4.9)
Requirement already satisfied: requests-oauthlib>=0.7.0 in /usr/local/lib/python3.10/dist-packages (from google-auth-oauthlib<1.1,>=0.5->tensorboard->simple
Requirement already satisfied: zipp>=0.5 in /usr/local/lib/python3.10/dist-packages (from importlib-metadata<7,>=1.4->streamlit->simpletransformers) (3.16.2
Requirement already satisfied: markdown-it-py>=2.2.0 in /usr/local/lib/python3.10/dist-packages (from rich<14,>=10.14.0->streamlit->simpletransformers) (3.0
Requirement already satisfied: pygments<3.0.0,>=2.13.0 in /usr/local/lib/python3.10/dist-packages (from rich<14,>=10.14.0->streamlit->simpletransformers) (2
Collecting pytz-deprecation-shim (from tzlocal<5,>=1.1->streamlit->simpletransformers)
  Downloading pytz_deprecation_shim-0.1.0.post0-py2.py3-none-any.whl (15 kB)
Requirement already satisfied: MarkupSafe>=2.1.1 in /usr/local/lib/python3.10/dist-packages (from werkzeug>=1.0.1->tensorboard->simpletransformers) (2.1.3)
```

Executing (32m 44s) <cell line: 1> > train_model() > train() > backward() > backward()

Step 3 : Upload the labelled dataset which I have uploaded in artefact on to the google collab notebook and then run all the commands in the Jupypter Notebook. When you run all the commands given below it will start training the model for 8 epoch which means it will train the model 8 times over the same dataset.



```
[18]
Downloading (…)lve/main/config.json: 100%  570/570 [00:00<00:00, 21.1kB/s]
Downloading model.safetensors: 100%  436M/436M [00:03<00:00, 129MB/s]
Some weights of BertForTokenClassification were not initialized from the model checkpoint at bert-base-cased and are newly initialized: ['classifier.weight', '
You should probably TRAIN this model on a down-stream task to be able to use it for predictions and inference.
Downloading (…)solve/main/vocab.txt: 100%  213k/213k [00:00<00:00, 4.71MB/s]
Downloading (…)okenizer_config.json: 100%  29.0/29.0 [00:00<00:00, 1.82kB/s]
```

```
model.train_model(train_data,eval_data = test_data,acc=accuracy_score)
```

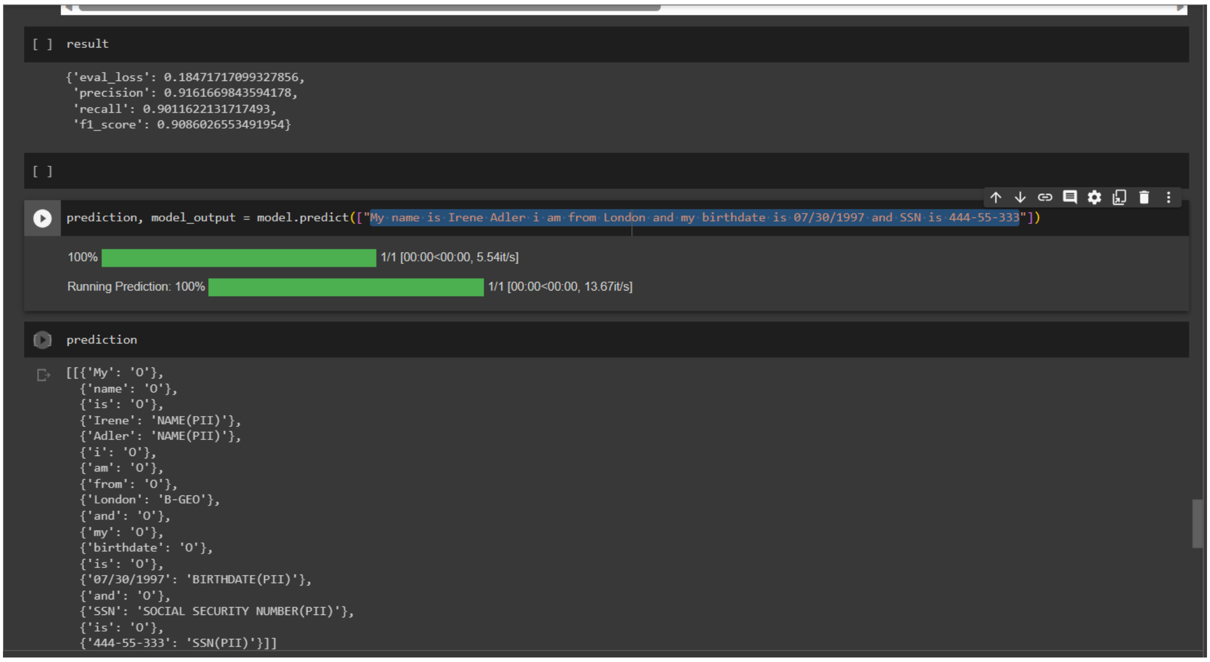
```
/usr/local/lib/python3.10/dist-packages/simpletransformers/ner/ner_utils.py:190: FutureWarning: In a future version of pandas, a length 1 tuple will be returne
  return [
100%  2/2 [01:08<00:00, 33.96s/it]
Epoch 6 of 8: 62%  5/8 [30:09<18:02, 360.79s/it]
Epochs 0/8. Running Loss: 0.0664: 100%  1563/1563 [05:47<00:00, 5.07it/s]
/usr/local/lib/python3.10/dist-packages/torch/optim/lr_scheduler.py:139: UserWarning: Detected call of `lr_scheduler.step()` before `optimizer.step()`. In PyTo
  warnings.warn("Detected call of `lr_scheduler.step()` before `optimizer.step()`. "
Epochs 1/8. Running Loss: 0.0918: 100%  1563/1563 [05:55<00:00, 5.00it/s]
Epochs 2/8. Running Loss: 0.0346: 100%  1563/1563 [05:59<00:00, 5.05it/s]
Epochs 3/8. Running Loss: 0.0504: 100%  1563/1563 [05:55<00:00, 5.09it/s]
Epochs 4/8. Running Loss: 0.0252: 100%  1563/1563 [05:45<00:00, 5.07it/s]
Epochs 5/8. Running Loss: 0.0154: 78%  1226/1563 [04:36<01:14, 4.53it/s]
```

```
result, model_outputs, preds_list = model.eval_model(test_data)
```

```
/usr/local/lib/python3.10/dist-packages/simpletransformers/ner/ner_utils.py:190: FutureWarning: In a future version of pandas, a length 1 tuple will be returne
```

Executing (36m 21s) <cell line: 1> > train_model() > train() > backward() > backward()

Step 4 : Once the model is trained , you can pass the text input using model.predict function: To give a random text to the model which the model will classify and give you output. In my case I have given the text as “My name is Irene Adler i am from London and my birthdate is 07/30/1997 and SSN is 444-55-333” which is a random text and is not associated with any human.



```
[ ] result
```

```
{'eval_loss': 0.18471717099327856,
 'precision': 0.9161669843594178,
 'recall': 0.9011622131717493,
 'f1_score': 0.9086026553491954}
```

```
[ ]
```

```
prediction, model_output = model.predict(["My name is Irene Adler i am from London and my birthdate is 07/30/1997 and SSN is 444-55-333"])
```

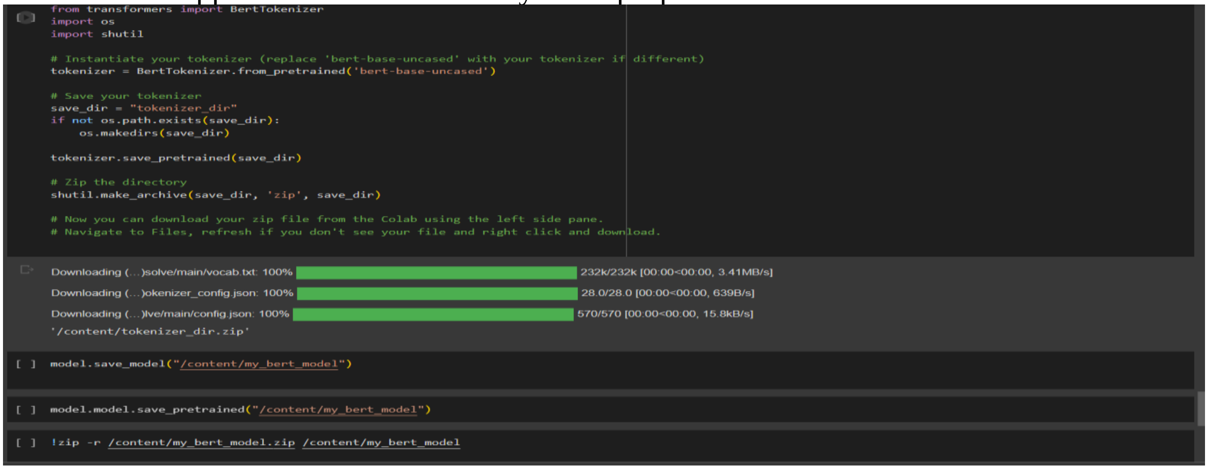
```
100%  1/1 [00:00<00:00, 5.54it/s]
Running Prediction: 100%  1/1 [00:00<00:00, 13.67it/s]
```

```
prediction
```

```
[[{'My': 'O'},
  {'name': 'O'},
  {'is': 'O'},
  {'Irene': 'NAME(PII)'},
  {'Adler': 'NAME(PII)'},
  {'i': 'O'},
  {'am': 'O'},
  {'from': 'O'},
  {'London': 'B-GEO'},
  {'and': 'O'},
  {'my': 'O'},
  {'birthdate': 'O'},
  {'is': 'O'},
  {'07/30/1997': 'BIRTHDATE(PII)'},
  {'and': 'O'},
  {'SSN': 'SOCIAL SECURITY NUMBER(PII)'},
  {'is': 'O'},
  {'444-55-333': 'SSN(PII)'}]]
```

As seen in the image above we gave a sentence to the model through model.predict function and the output was stored in prediction. When we displayed the output using prediction function it showed us how the model classifies each word and label its with an identifier. In the above scenario words like My, name are labelled as O which represents Others, London is labelled as B-Geo which represents Geo location, Irene Adler is tagged as Name(PII) , SSN is tagged as Social Security number(PII), 07/30/1997 is tagged as Birthdate(PII) these all labels show PII data has been classified with great accuracy which is our main goal.

Step 5: Importing the BERT model and classifier into our local machine so we could use it further in our application or use it for any other purpose.



```
from transformers import BertTokenizer
import os
import shutil

# Instantiate your tokenizer (replace 'bert-base-uncased' with your tokenizer if different)
tokenizer = BertTokenizer.from_pretrained('bert-base-uncased')

# Save your tokenizer
save_dir = "tokenizer_dir"
if not os.path.exists(save_dir):
    os.makedirs(save_dir)

tokenizer.save_pretrained(save_dir)

# Zip the directory
shutil.make_archive(save_dir, 'zip', save_dir)

# Now you can download your zip file from the Colab using the left side pane.
# Navigate to Files, refresh if you don't see your file and right click and download.
```

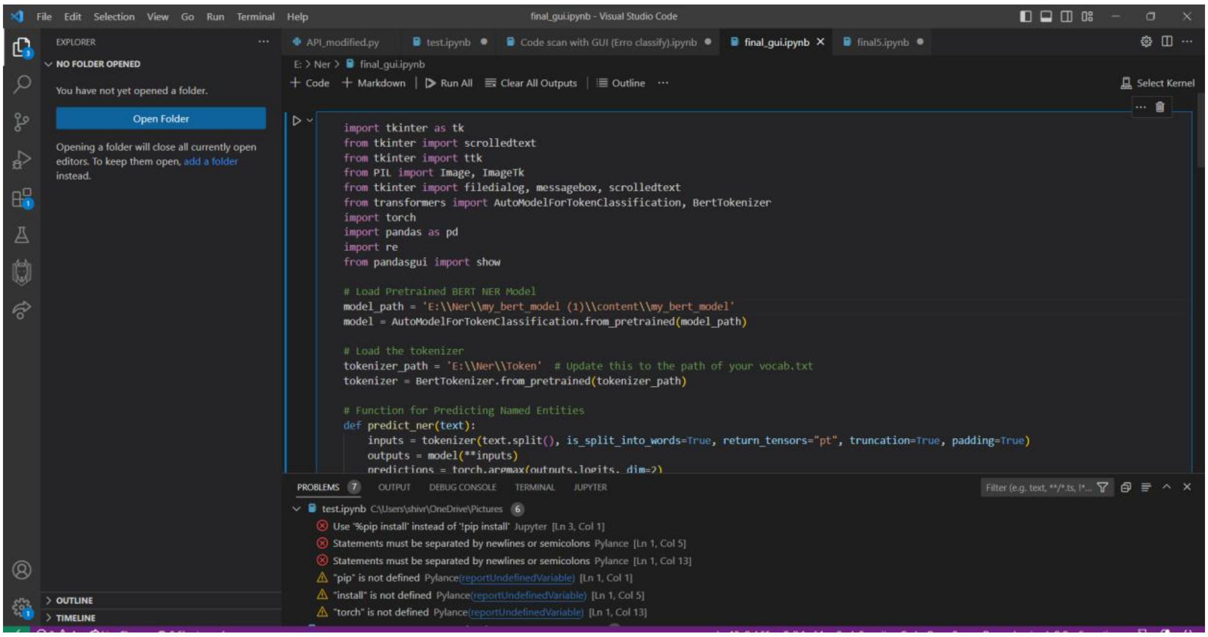
```
Downloading (…)solve/main/vocab.txt: 100%   232k/232k [00:00<00:00, 3.41MB/s]
Downloading (…)okenizer_config.json: 100%   28.0/28.0 [00:00<00:00, 639B/s]
Downloading (…)lve/main/config.json: 100%   570/570 [00:00<00:00, 15.8kB/s]
'/content/tokenizer_dir.zip'
```

```
[ ] model.save_model("/content/my_bert_model")
```

```
[ ] model.model.save_pretrained("/content/my_bert_model")
```

```
[ ] !zip -r /content/my_bert_model.zip /content/my_bert_model
```

Step 6: Open the a new jupypter notebook and import the file named **final_gui.ipynb** which is code which creates a application where it imports the finetuned BERT model and gives us a user interface to interact with the model.

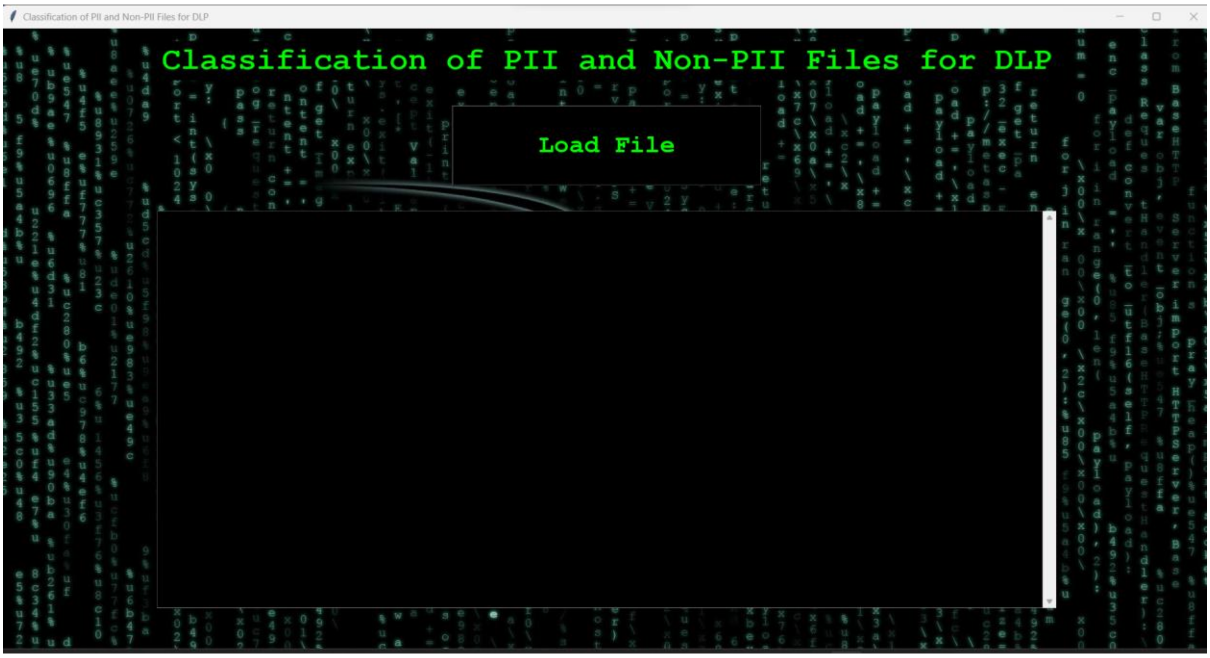

File Edit Selection View Go Run Terminal Help
final_gui.ipynb - Visual Studio Code
EXPLORER
NO FOLDER OPENED
You have not yet opened a folder.
Open Folder
Opening a folder will close all currently open editors. To keep them open, add a folder instead.
OUTLINE
TIMELINE
API_modified.py
test.ipynb
Code scan with GUI (Erro classify).ipynb
final_gui.ipynb
final5.ipynb
E: > Ner > final_gui.ipynb
+ Code + Markdown Run All Clear All Outputs Outline
Select Kernel
PROBLEMS 7 OUTPUT DEBUG CONSOLE TERMINAL JUPYTER
Filter (e.g. text, **/*.ts, !*...

```
import tkinter as tk
from tkinter import scrolledtext
from tkinter import ttk
from PIL import Image, ImageTk
from tkinter import filedialog, messagebox, scrolledtext
from transformers import AutoModelForTokenClassification, BertTokenizer
import torch
import pandas as pd
import re
from pandasgui import show

# Load Pretrained BERT NER Model
model_path = 'E:\\Ner\\my_bert_model (1)\\content\\my_bert_model'
model = AutoModelForTokenClassification.from_pretrained(model_path)

# Load the tokenizer
tokenizer_path = 'E:\\Ner\\Token'  # Update this to the path of your vocab.txt
tokenizer = BertTokenizer.from_pretrained(tokenizer_path)

# Function for Predicting Named Entities
def predict_ner(text):
    inputs = tokenizer(text.split(), is_split_into_words=True, return_tensors="pt", truncation=True, padding=True)
    outputs = model(**inputs)
    predictions = torch.argmax(outputs.logits, dim=2)
```

```
test.ipynb C:\Users\shivr\OneDrive\Pictures 6
Use '%pip install' instead of '!pip install' Jupyter [Ln 3, Col 1]
Statements must be separated by newlines or semicolons Pylance [Ln 1, Col 5]
Statements must be separated by newlines or semicolons Pylance [Ln 1, Col 13]
"pip" is not defined Pylance(reportUndefinedVariable) [Ln 1, Col 1]
"install" is not defined Pylance(reportUndefinedVariable) [Ln 1, Col 5]
"torch" is not defined Pylance(reportUndefinedVariable) [Ln 1, Col 13]
```

Here in this code snippet we can see we have imported the finetuned BERT model and its tokenizer by giving the path on your local computer where the files are present.

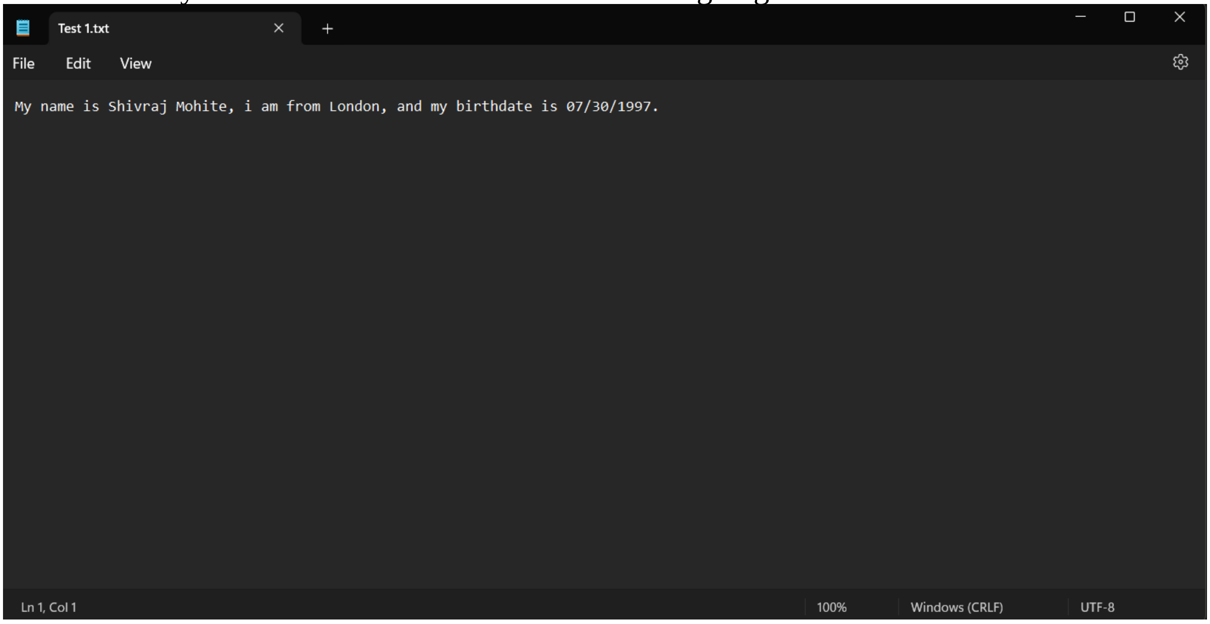
Step 7: Executing the code directly once the path of BERT model and tokenizer is set (Note: All the code has been explained in implementation part of the report)


Classification of PII and Non-PII Files for DLP
Classification of PII and Non-PII Files for DLP
Load File

This is the basic GUI of the application now you can use the load file button to upload a text file which the application will read and then will classify the content of the file using BERT and then give alert according to the contents present in the file.

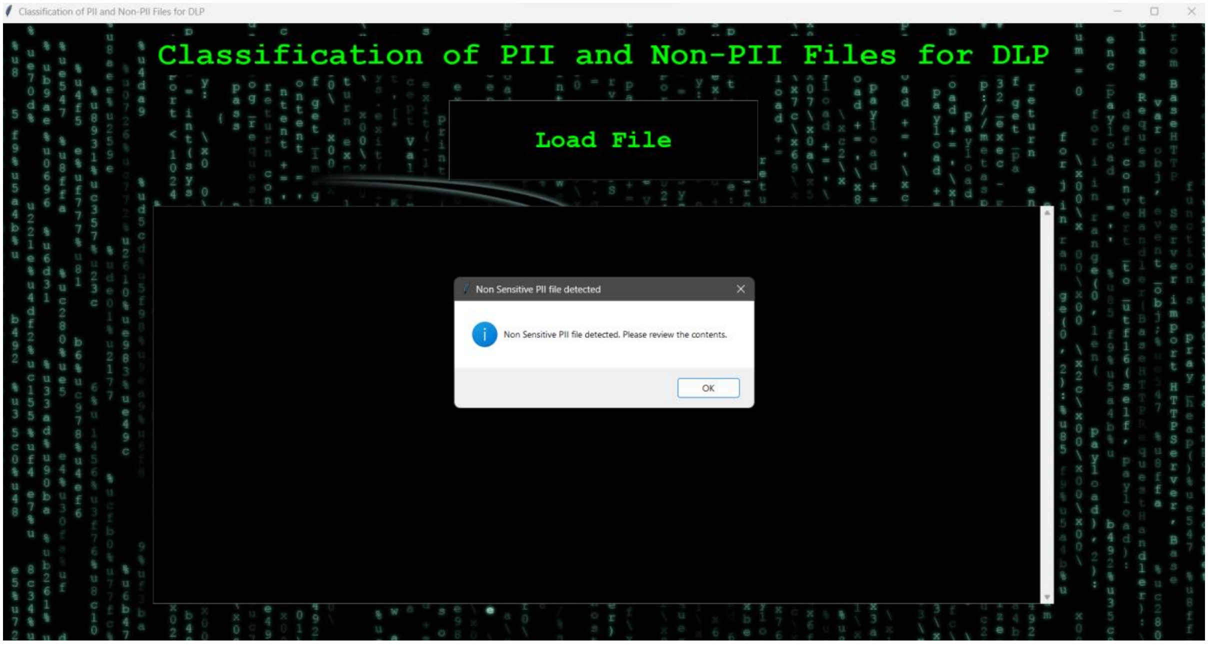
Step 8: Running the test cases to see whether the model and the application is running correctly or not.

Test case 1 : We will load a text file which will have Non sensitive PII information
First ill show you the content of the file which we are going to load:



```
My name is Shivraj Mohite, i am from London, and my birthdate is 07/30/1997.
```

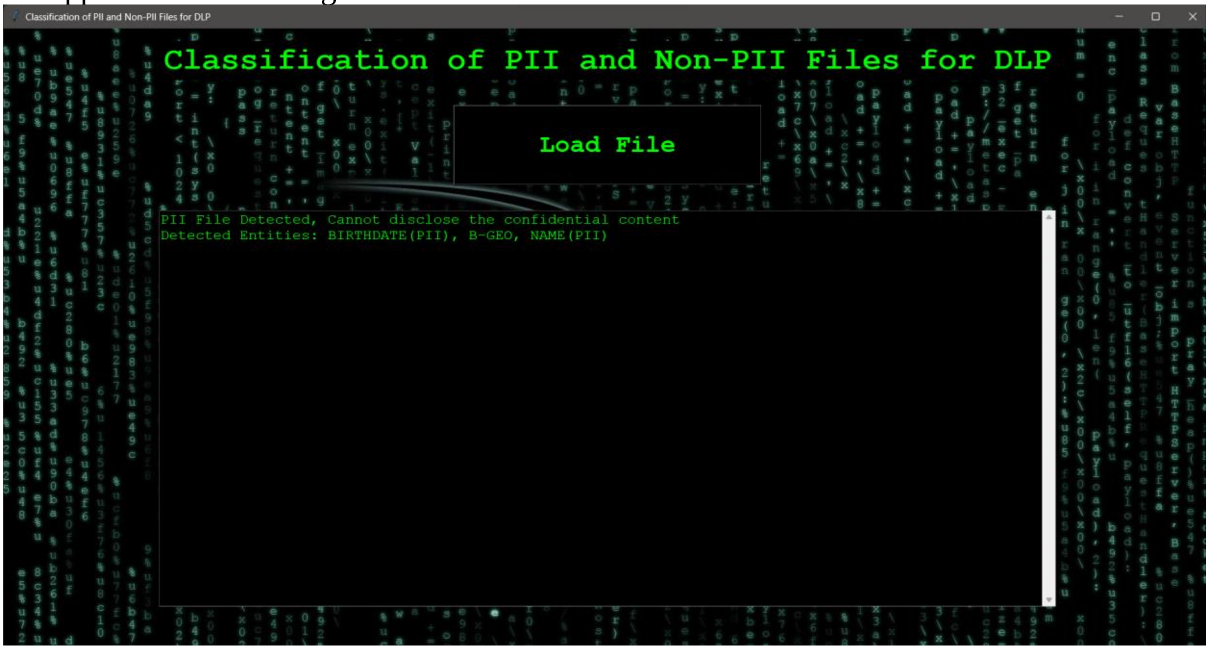
You can see that this file does not have Sensitive PII but has Non Sensitive PII, we will now check if it works correctly.


Classification of PII and Non-PII Files for DLP
Classification of PII and Non-PII Files for DLP
Load File
OK

Non Sensitive PII file detected

Non Sensitive PII file detected. Please review the contents.

It gives alert that Non sensitive PII detected as the file contains name, birthdate etc. Hence the application is working fine.

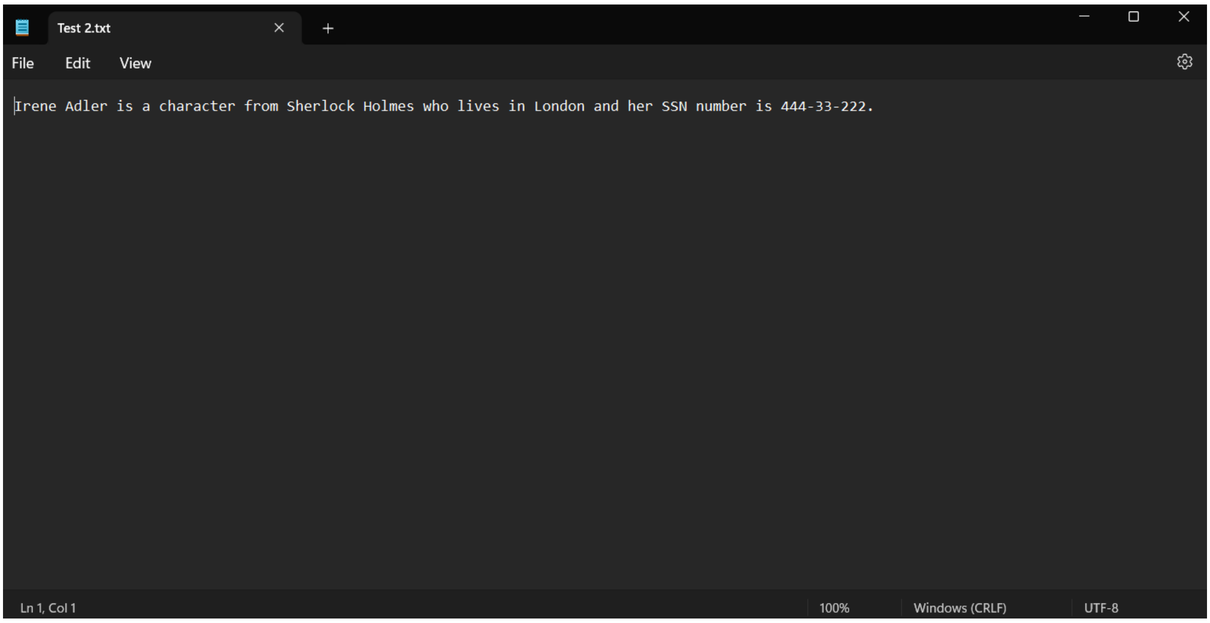

Classification of PII and Non-PII Files for DLP
Classification of PII and Non-PII Files for DLP
Load File

```
PII File Detected, Cannot disclose the confidential content
Detected Entities: BIRTHDATE(PII), B-GEO, NAME(PII)
```

It will show the detected entities as Birthdate(PII), B-Geo, Name (PII) and wont display the content of the file. In our scenario above 3 labels are classified as Non-Sensitive PII according to that the application is working accurately.

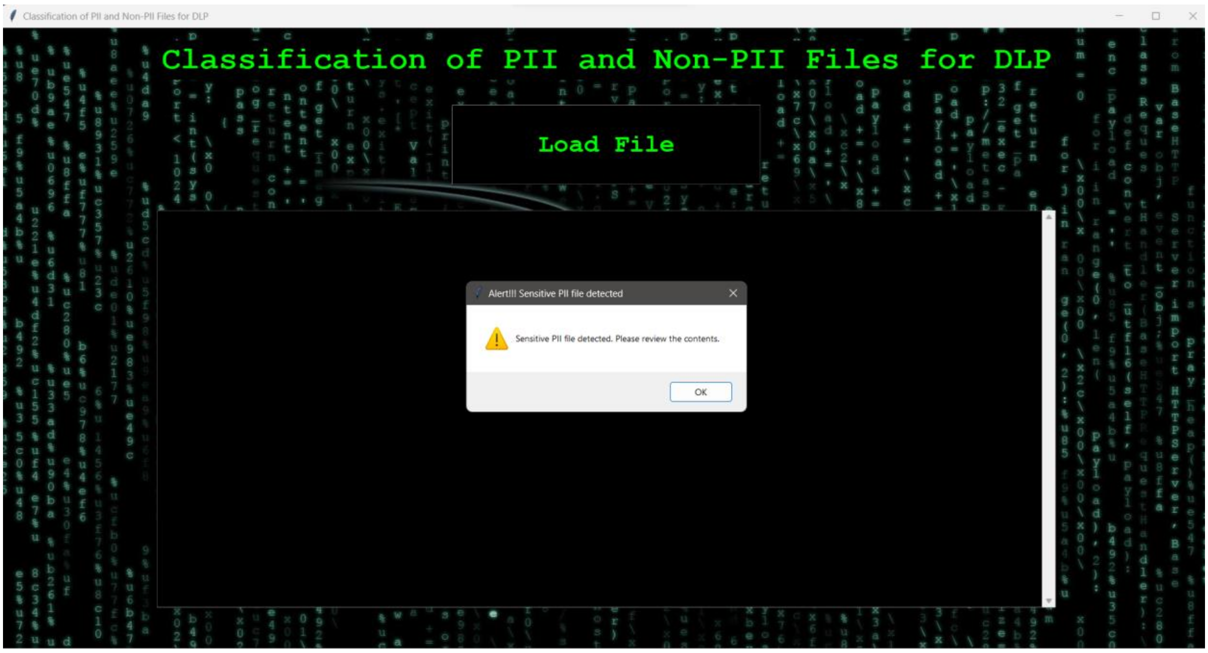
Test Case 2: Loading Sensitive File

Lets first show you content of the file we are going to load:



Irene Adler is a character from Sherlock Holmes who lives in London and her SSN number is 444-33-222.

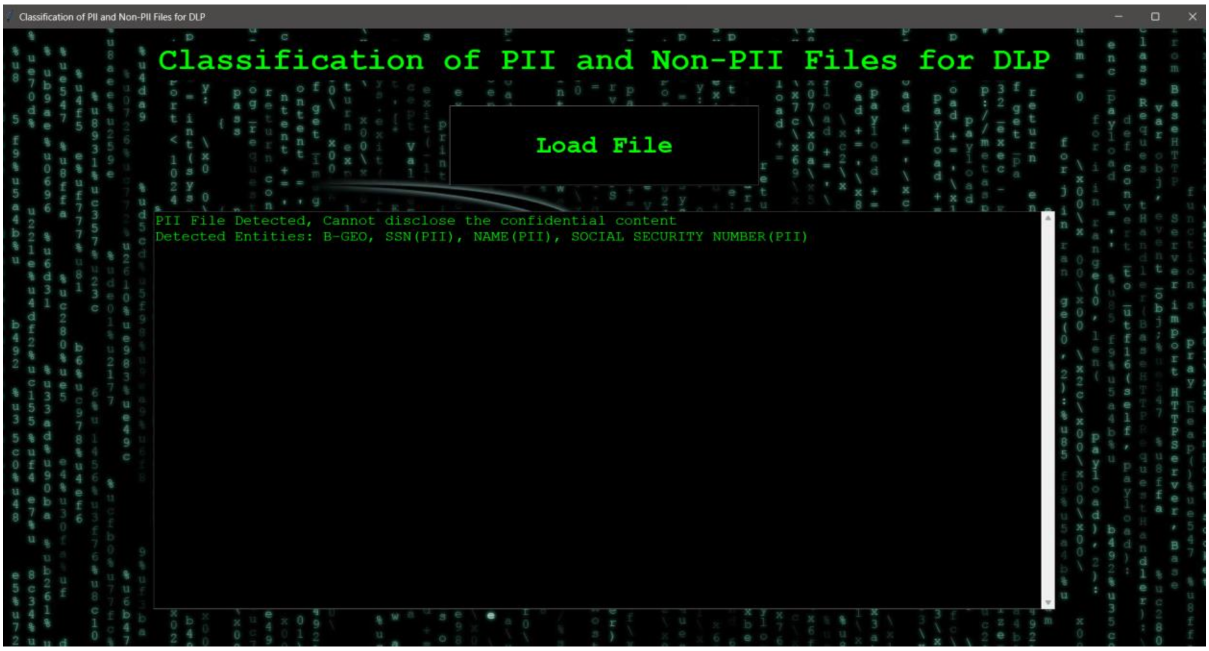
Here you can clearly see Sensitive PII information which is SSN number is present. Hence we will now test by uploading the file.


Classification of PII and Non-PII Files for DLP
Classification of PII and Non-PII Files for DLP
Load File
OK

Alert!!! Sensitive PII file detected

Sensitive PII file detected. Please review the contents.

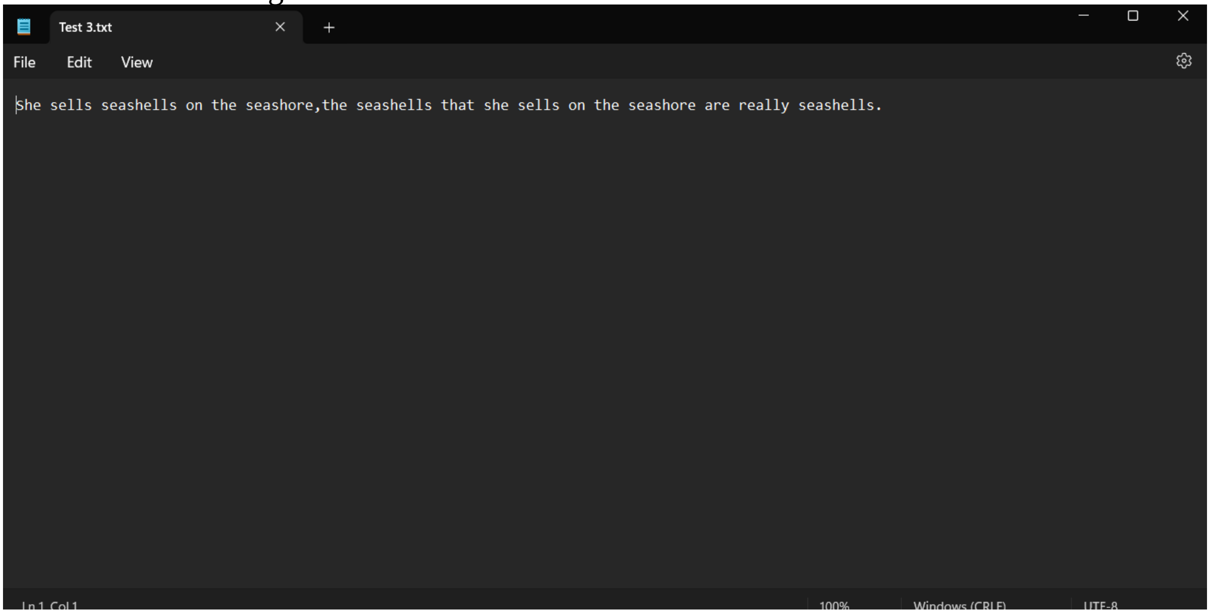
It shows that the file we loaded has Sensitive PII information present in it like social security number.


Classification of PII and Non-PII Files for DLP
Classification of PII and Non-PII Files for DLP
Load File

```
PII File Detected, Cannot disclose the confidential content
Detected Entities: B-GEO, SSN(PII), NAME(PII), SOCIAL SECURITY NUMBER(PII)
```

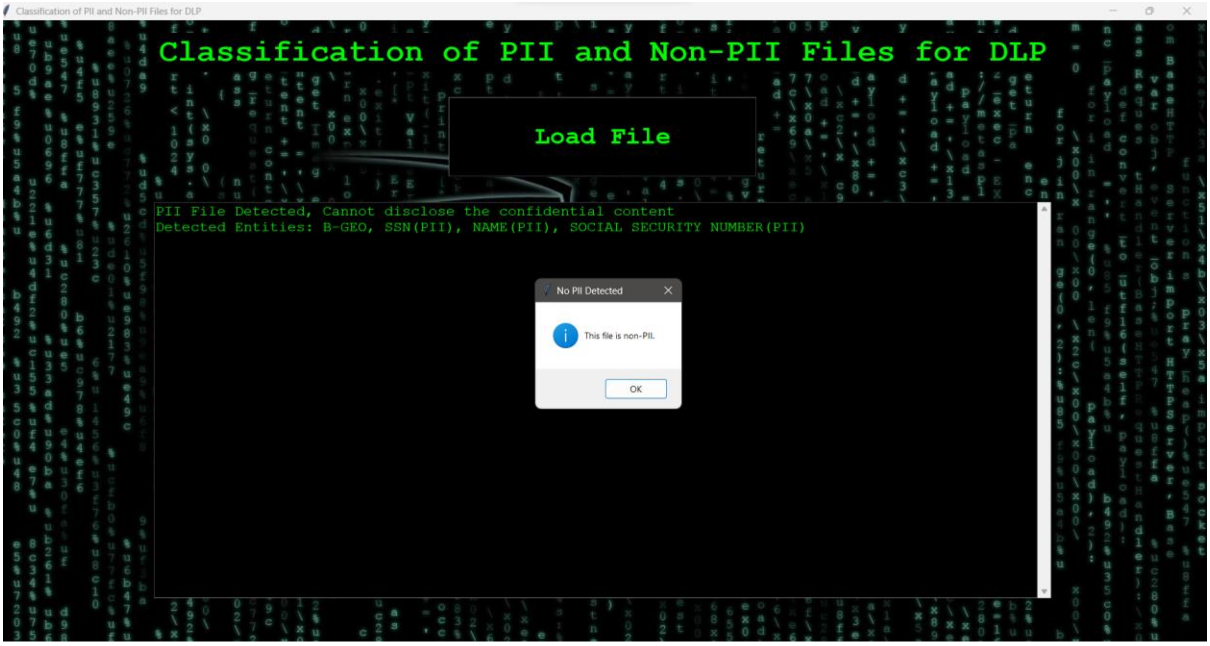
Here we can see in detected entities Social security number is detected which is considered as Sensitive PII which indicates our model is working fine.

Test Case 3 : Loading file which contains Non PII information.



```
She sells seashells on the seashore,the seashells that she sells on the seashore are really seashells.
```

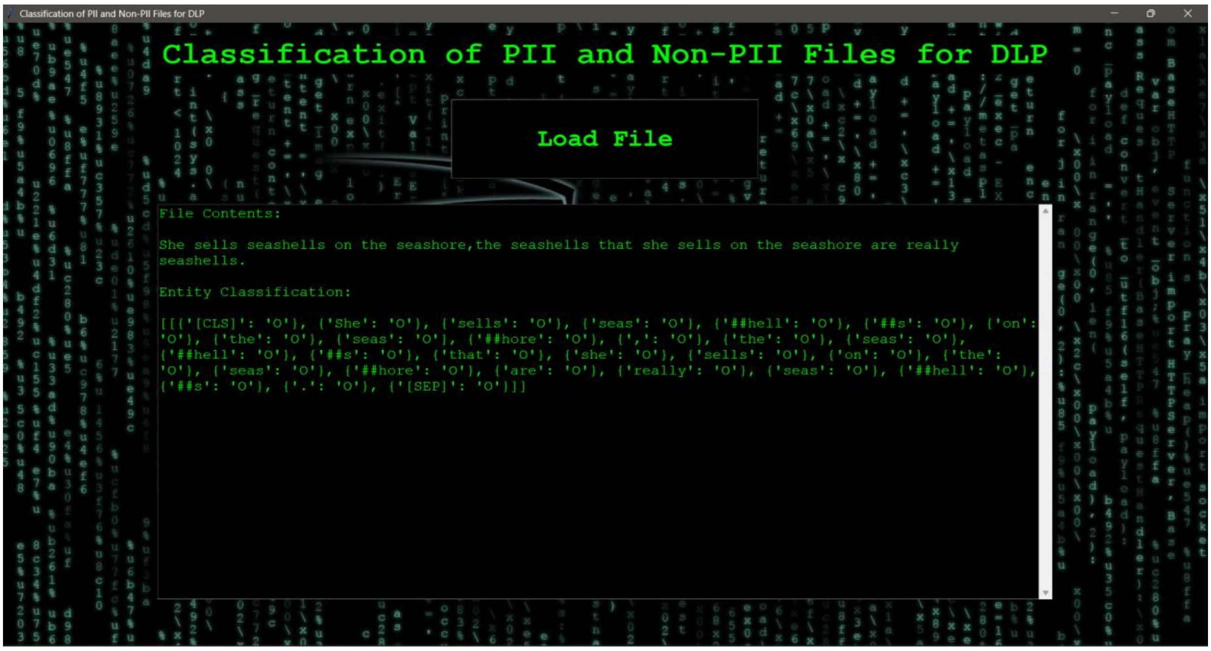
This is the content of the file and as you can see there are is no sensitive information in that. And after loading it up on our application it should show that the file is non PII and show display its content



```
PII File Detected, Cannot disclose the confidential content
Detected Entities: B-GEO, SSN(PII), NAME(PII), SOCIAL SECURITY NUMBER(PII)
```

This file is non-PII.

As you can see when we load the file the alert pops up saying its an Non PII file. And in the next screenshot you can see the contents of the file are displayed and classified.



```
File Contents:

She sells seashells on the seashore,the seashells that she sells on the seashore are really
seashells.

Entity Classification:

[[{'[CLS]': 'O'}, {'She': 'O'}, {'sells': 'O'}, {'seas': 'O'}, {'##hell': 'O'}, {'##s': 'O'}, {'on':
'O'}, {'the': 'O'}, {'seas': 'O'}, {'##hore': 'O'}, {',': 'O'}, {'the': 'O'}, {'seas': 'O'},
{'##hell': 'O'}, {'##s': 'O'}, {'that': 'O'}, {'she': 'O'}, {'sells': 'O'}, {'on': 'O'}, {'the':
'O'}, {'seas': 'O'}, {'##hore': 'O'}, {'are': 'O'}, {'really': 'O'}, {'seas': 'O'}, {'##hell': 'O'},
{'##s': 'O'}, {'.': 'O'}, {'[SEP]': 'O'}]]
```

Here we can actually see the content of loaded file along with its classification.