Piyush Kumar

PII-1

Dissertation_MiniProject_Project

Dissertation_Project_Mini Project

Symbiosis International University

Document Details

Submission ID

trn:oid:::1:2907591093

Submission Date

May 2, 2024, 11:40 AM GMT+5:30

Download Date

May 2, 2024, 11:42 AM GMT+5:30

File Name

Final_Project_1.docx

File Size

1.0 MB

26 Pages

3,792 Words

24,019 Characters

How much of this submission has been generated by AI?

0%

of qualifying text in this submission has been determined to be generated by AI.

Caution: Percentage may not indicate academic misconduct. Review required.

It is essential to understand the limitations of AI detection before making decisions about a student's work. We encourage you to learn more about Turnitin's AI detection capabilities before using the tool.

Frequently Asked Questions

What does the percentage mean?
The percentage shown in the AI writing detection indicator and in the AI writing report is the amount of qualifying text within the submission that Turnitin's AI writing detection model determines was generated by AI.

Our testing has found that there is a higher incidence of false positives when the percentage is less than 20. In order to reduce the likelihood of misinterpretation, the AI indicator will display an asterisk for percentages less than 20 to call attention to the fact that the score is less reliable.

However, the final decision on whether any misconduct has occurred rests with the reviewer/instructor. They should use the percentage as a means to start a formative conversation with their student and/or use it to examine the submitted assignment in greater detail according to their school's policies.

How does Turnitin's indicator address false positives?
Our model only processes qualifying text in the form of long-form writing. Long-form writing means individual sentences contained in paragraphs that make up a longer piece of written work, such as an essay, a dissertation, or an article, etc. Qualifying text that has been determined to be AI-generated will be highlighted blue on the submission text.

Non-qualifying text, such as bullet points, annotated bibliographies, etc., will not be processed and can create disparity between the submission highlights and the percentage shown.

What does 'qualifying text' mean?
Sometimes false positives (incorrectly flagging human-written text as AI-generated), can include lists without a lot of structural variation, text that literally repeats itself, or text that has been paraphrased without developing new ideas. If our indicator shows a higher amount of AI writing in such text, we advise you to take that into consideration when looking at the percentage indicated.

In a longer document with a mix of authentic writing and AI generated text, it can be difficult to exactly determine where the AI writing begins and original writing ends, but our model should give you a reliable guide to start conversations with the submitting student.

Symbiosis Institute of Computer Studies and Research

A PROJECT REPORT

on

Personally Identifiable Information Analysis

in partial fulfillment
for the award of the degree of MSc-CA

Submitted by

PIYUSH KUMAR

PRN: 22030142020

Symbiosis International (Deemed University), Pune, India

MSc-CA- Batch 2022-2024

Date: 02/05/24

COMPLETION CERTIFICATE

This is to certify that the project report titled Personally Identifiable Information Analysis is the bonafide work of Piyush Kumar PRN: 22030142020, MSc-CA Batch 2022-2024 at Symbiosis Institute of Computer Studies and Research (SICSR) who carried out the project work under my supervision. He has completed the project during Start Date: 5/12/23 to End Date: 02/05/24.

Signature of the Supervisor/Mentor
Shubhashri Waghmare

MSc (CA)/SICSR, Pune

ACKNOWLEDGEMENT

I would like to thank Shubhashri Waghmare for her essential contributions to my final project on the investigation of Personally Identifiable Information (PII). Her active participation was critical in acquiring critical information and varied perspectives for my project, which delves into the complex environment of privacy and security issues linked with personal data.

Abstract

With the goal of improving system security, the "Personal Identifiable Information (PII) Scanner" is a Python-based utility that may locate and evaluate sensitive data that may be present in text files. The program looks for personally identifiable information (PII) like names, phone numbers, social security numbers, and email addresses using regular expressions. Based on the kind and volume of information found, the scanning procedure generates a PII score.

A visual depiction of the discovered PII in relation to the overall amount of PII in the scanned text is given by the scanning tool. Moreover, a Plotly and Matplotlib-implemented PII Score Indicator provides a brief evaluation of the overall system security. Before the PII scan begins, the user is asked for permission.

The PII Scanner's primary attributes comprise:

- Detection of several PII categories, including names, phone numbers, social security numbers, and email addresses.
- PII score calculation based on the amount and kind of information found.
- Results are displayed using bar charts and a PII Score Indicator.
- Based on their PII score, the tool urges users to take proactive steps to improve system security.
- The PII scan findings are recorded in the produced log file, which helps with system auditing and analysis.

Users can designate the file paths to be scanned when using the PII Scanner, and the program offers useful insights depending on the PII that is found. By providing outside resources for further ideas, the project also raises awareness of data security.

The PII Scanner employs a user-centric methodology, guaranteeing that consent is acquired prior to commencing any scans. This ethical concern demonstrates the tool's dedication to privacy and conforms to modern data protection guidelines. Users are empowered to determine whether or not their system is subject to PII scanning, encouraging an open and accountable use paradigm.

Plotly and Matplotlib integration improves the tool's visual appeal and makes it easier to comprehend the PII distribution in its whole. The bar charts provide users with a clear visual depiction of potential vulnerabilities by efficiently displaying the counts of PII that were found compared to the total PII in the scanned text.

As a result, the "Personal Identifiable Information (PII) " not only takes care of the urgent requirement for PII detection, but it also promotes responsible data management and greater user knowledge of security issues.

TABLE OF CONTENTS

1. Introduction: -

1.1 Background: -

In an era of privacy concerns and information breaches, the need for tools to safeguard and maintain robust systems is immediate. The Personal Identifiable Information PII Scanner is a tool built in Python to meet the growing need for identifying and assessing information in text files. The primary aim is to give users a force for healthy possibilities to promote and enhance system security. A tool in the detector camp's current landscape shines through as an excellent grade. The PII Scanner satisfies the need for an urgent, reliable tool to detect and assess information in text files. The simple fact it was created in Python means its development was motivated by a commitment to the highest degree of flexibility and ease of use — different computer expertise levels, using its powers.

1.2 Need: -

PII scanning technology put in place will be essential in addressing the need to protect sensitive data, considering the rapidly growing concerns about data privacy. The program whim permission from the user, conduct a pattern analysis search of the addressed files in a systematic approach. The use of a complex pattern-matching algorithm will help in the identification and analysis of names, social security numbers, phone numbers, addresses, and email addresses. The code will be able to efficiently log and present critical information, as seen case bar charts and a Plotly gauge indicator used for illustrative and educational purposes. This solution is critical because it will provide a comprehensive approach to the assessment and amelioration of the security of data stored that will benefit both people and businesses.

1.3 Problem Definition: -

A person is identifiable by the following information: name, phone number, e-mail, social security number , and other personally identifiable information . Today, the growing number of data gathered and stored digitally poses a major threat to personal data from data breaches and misuse. The "process for identifying and evaluating threats to personally identifiable information that can jeopardize its availability, integrity, and confidentiality" is known as "PII analysis".

1.4 Objectives: -

A Python Personal Identifiable Information scanning tool should be a straight-forward tool requiring consent to methodically analyze particular text files. Specifically, the tool is supposed to locate and quantify the number of sensitive data appearances, including phone numbers, e-mail addresses, social security numbers, and names that have occurred. The tool should compare the seriousness of these detections through absolute visual depictions such as bar graphs and a Plotly indicator. The device should also keep a report of the result and output evidence

that the customer can improve data kept security on. The main objective is to offer human beings and firms a dynamic and effective PII scan alternative to make data privacy more robust.

1.5 Technology Used: -

The program for scanning PII is implemented in Python, for the implementation of which a number of technologies are used. These technologies include regular expressions , for comparison with the pattern, which creates “fingerprints” when looking for patterns, using exact matching, allows to find such information as: phone numbers, e-mails, card numbers social security, and names with documents; Matplolib, plotly graph Objects for visualization, interactive gauge indications, and instructive bar charts on the severity of PII incidents found ; Python’s built-in logging package for comprehensive logging of the scan result, which eases the analysis and audit of the logs; Simple prompts for user interaction are provided, and with the help of the abilities of the Python file, cases are structured to scan particular text files.

2. Literature Survey: -

2.1 Existing System: -

The systems that are used to identify PII information include: Data loss prevention systems, are systems that detect and prevent the leakage of sensitive data. They may be set up to monitor several avenues, including emails, web traffic, file transfers, and other means, and then identify PII data depending on pre-defined rules. ML models that can identify patterns and locate personally identifying information can be used. Such models may be trained using labelled data, which is useful for identifying patterns in text or images used to identify personally identifiable information.

2.2 Proposed System: -

- User consent obtained: A priorly before scanning a user’s system or one’s files looking for Personally Identifiable Information (PII) The express permission is acquired from the user.
- PII Scanning: Using high-level computer languages and open-source tools or GitHub repositories, the user’s system and one’s files are cross-checked for the user-specified PII which is an email address and emails in the given system.
- PII score output: The PII score is tabulated and then fed back to the user in a translation piece giving the user a simplified Format of his/her PII score determined based on the amount and types of PII prodded from the user’s data.
- Recommendations: Based on the PII score, realistic improvements are recommended to the user’s system which may need solutions by Galaxkey Enterprises Solutions for a PII score of more than 20 and favorable improvement given at a score of 20 and below to achieve better or ensure security.
- In conclusion, our aim is to help customers comprehend what security problems they may confront and provide practical pointers on protective measures in respect of their vulnerable information.

2.3 Feasibility Study: -

The PII feasibility study aims to evaluate whether it is obtainable and suitable to create a detection and encryption system that allows the project to examine PII. Specifically, if a PII score scanning the system shows on the score, the technology is engineered to immediately encode or hide crucial information to block unauthorized access.

The proposed solution is feasible from a technological standpoint since the most recent technology supports this kind of encryption and discovery. Solutions from Galaxkey Enterprises, Cipher Cloud, and Secure Age Technologies may be built in the suggested system.

Financial viability: Considerable monetary investment will be required to develop and implement the proposed system. Hardware, software, and personnel costs have to be taken into account. A detailed financial analysis will be needed.

Operational suitability: The proposed system will need adepts employees to run and maintain it. The personnel will need sufficient training in the required technology and security measures. An operational strategy and training schedule must be formulated to guarantee the system's maximum usage.

2.4 Future Scope: -

In the future, analyzing data that is unstructured, or which includes audio, video, photographs, text files, etc., may greatly increase data security and privacy. Since the number of IoT devices is rising, so is the opportunity of crunching vital and personal data. Hence, it is crucial to ensure adequate user privacy measures.

PII discovery software on the network layer might be a viable option. In this case, sensitive data is removed and only data necessary will be sent to the analytics platform. The private information wouldn't be as likely to be placed in precarious places or shared with third parties.

Lastly, if PII is ever yelled when using IoT devices, the microphone could be disabled from the software side. Technologies for speech recognition or similar can be developed to recognize when someone says private information and restrict the device from speaking it. By pairing with PII detection technologies as an easily accessible API or SDKs integrated into other apps, one is able to extend another line of control against the acquisition and abuse of personal information.

Since the PII detection technology could be provided as an API or SDK, one could easily integrate its capabilities into an app. Therefore, it allows the developer to find and protect any sensitive data their apps could

process in an easy and quick way. For instance, the app developer could search the user-submitted data for sensitive data that needs to be protected – e.g., credit card numbers or social security numbers.

One of the tools that could potentially enable the above applications is a PII detection SDK. Overall, there are many possible solutions and combinations of solutions to ensure the protection of sensitive information during the era of IoT devices. As for the current case, ensuring user privacy can be achieved through a mixture of technological and legal solutions or regulatory measures.

Additionally, integrating PII detection as an API or SDK may increase users' trust in the application, indicating that the developer respects security, privacy of personal information, and protects users' sensitive data.

Generally, providing API integrations or SDKs for PII detection is a smart way of ensuring the privacy and protection of personalized information and aiding users in feeling safe using an application that interacts with their data. Therefore, it is a fair and good business practice.

2.5 Industry Requirements: -

- A variety of sectors not only prioritize personally identifiable information but also have special rules and requirements that a business should comply with in terms of ensuring the security and privacy of PII data. A few examples of industry standard involve the analysis of PII information
- Healthcare. The HIPAA law – the Health Insurance Portability and Accountability Act, introduces strict requirements to the security of patient information. Healthcare-related bodies state that all PII data – patient names, address or any medical indicators – need to be encrypted while in transit or at rest. Moreover, the access level should be restricted in terms of reading or editing PII data.
- The education sector is covered by the Family Educational Rights and Privacy Act, which provides protection to student information. Names, addresses, and academic records of students are examples of PII in this category. Educational institutions must make sure that this data is maintained securely and that only authorised individuals have access to it.
- We have learned from https://www.galaxkey.com/ that they are providing a desktop programme for PII information analysis. According to the description, their solution appears to be intended to assist enterprises in safeguarding their personally identifiable information (PII) by transferring it to a secure area within their system and then encrypting that location to deter illegal access.

3. Methodology: -

3.1 Methods adopted: -

Several diverse methods and technologies have to be integrated systematically for the successful development of the PII scanning tool used to find and evaluate the sensitive information contained in text files. The primary techniques leveraged are:

Expressions (Regex) for Pattern Matching:

- Utilized the Python re module to develop regular expressions for definitive pattern matching.
- Jerked regex patterns to spot the email addresses, phone numbers, social security numbers SSN , and names from the text files.
- Employed the obtained patterns of regex to scan the Text files and system detection of potential PII.OutputStream.

Data Visualization with Plotly Graph Objects and Matplotlib:

- The use of Plotly Graph Objects was for creating an interactive gauge indicator that can visually indicate the overall PII score.
- Additionally, Matplotlib was used for creating a bar chart that provides meaningful comparison of the detected PII instances versus the total PII occurrences identified.
- Consequently, these visualization ideas were crucial in enabling the user to understand the severity of the detected PII instances.

Logging Module for Results Recording:

- The Python logging module is a built-in facility used for recording detailed information about the PII scan results.
- This includes records of the file path, detected PII, and the overall PII score of each text file scanned.
- This logging approach provides the necessary logs that allow auditing and further analysis.

User Interaction and Consent Mechanism:

- The code incorporated the input function to ensure that users were prompt for consent to commence the PII scan, which aligns with ethical data privacy dimensions.
- Hence, the users freely participated in the scanning process through proper engagement agreement, and use of the scanning tool was explicit.

File Handling for Systematic Text File Scanning:

- Python’s file handling was key while reading and processing text files systematically.
- The code implemented the iteration features where the code was run on the file paths specified, allowing each file’s contents to be read for the PII scan.

- The code's systematic nature facilitated the systematic handling of the files hence the efficient scanning of multiple files based on several data storage practices.

Conditional Statements and Loops for Program Flow:

- This was achieved by: Systematic scanning of multiple files; ensuring each file is considered and processed independently.
- Conditional feedback based on the level of severity of the extracted PII information provided users with an understanding of the presented threat and associated reaction.

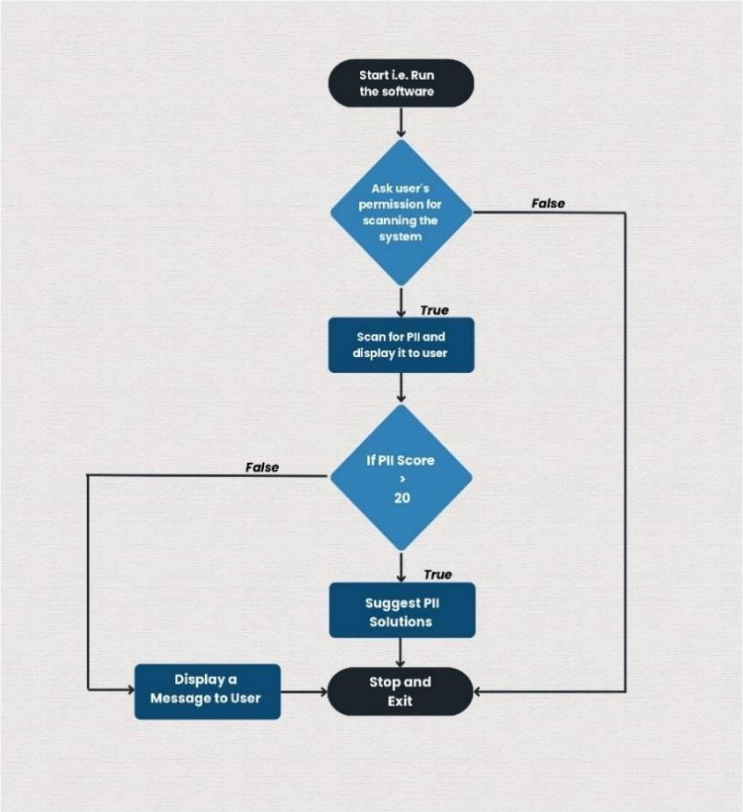
4. System Analysis and Design: -

4.1 User Requirement Specifications: -

The following are the user requirements for the PII analysis: -

- The system cannot scan the entire system for PII data without the user's consent.
- The scans will be used by the system to determine a PII score.
- The PII score will be shown to the consumer in an understandable graphical format.
- The optimal techniques for reducing privacy problems connected with PII data will be provided by the algorithm based on the PII score.
- The system will conduct research and analysis in accordance with moral standards.

4.2 Flow Chart: -



4.3 Code Execution and Output: -

```
import re

import plotly.graph_objects as go
import matplotlib.pyplot as plt
import logging
import sqlite3
from tabulate import tabulate
import datetime
import requests
from bs4 import BeautifulSoup
import PyPDF2
import csv
import json
```

```
# Configure logging settings
logging.basicConfig(filename='pii_scan_log.txt',
level=logging.INFO, format='%(asctime)s - %(levelname)s -
%(message)s')

patterns = {
    "Name": {"pattern": r"\b[A-Z][a-z]+\s[A-Z][a-z]+\b",
"weight": 1},
    "Email": {"pattern": r"\b[A-Za-z0-9._%+-]+@[A-Za-z0-9.-
]+\.[A-Z|a-z]{2,}\b", "weight": 1},
    "Phone Number": {"pattern": r"\b(?:\+?1[-
.\s]?)?\(?\d{3}\)?[-.\s]?\d{3}[-.\s]?\d{4}\b", "weight": 2},
    "Social Security Number": {"pattern": r"\b\d{3}[-
.\s]?\d{2}[-.\s]?\d{4}\b", "weight": 0}
}

pii_type_names = {
    "Name": "Name",
    "Email": "Email",
    "Phone Number": "Phone Number",
    "Social Security Number": "Social Security Number"
}

def detect_pii(text, file_name):
    detected_pii = {}
    for key, pattern_info in patterns.items():
        if key == "Name" and (file_name.endswith('.csv') or
file_name.endswith('.pdf') or isinstance(file_name, str) and
file_name.startswith('https://')):
            continue
        matches = re.findall(pattern_info["pattern"], text)
        if matches:
            detected_pii[(key, file_name)] = matches
    return detected_pii
```

```
def save_pii_to_database(detected_pii, source):
    conn = sqlite3.connect('pii_summary.db')
    c = conn.cursor()

    # Create a table for the source if it doesn't exist
    table_name = re.sub(r'\W+', '_', source)  # Sanitize the
URL to create a valid table name
    c.execute(f'''CREATE TABLE IF NOT EXISTS {table_name}
                 (name TEXT, email TEXT, phone_number TEXT,
social_security_number TEXT)''')

    # Iterate over detected PII
    for (_, _), pii_values in detected_pii.items():
        # Initialize variables for each type of PII
        name = email = phone_number = social_security_number =
None

        # Extract PII values from the detected_pii dictionary
and segregate them
        for pii_type, values in detected_pii.items():
            if pii_type[0] == "Name":
                name = '\n'.join(values)
            elif pii_type[0] == "Email":
                email = '\n'.join(values)
            elif pii_type[0] == "Phone Number":
                phone_number = '\n'.join(values)
            elif pii_type[0] == "Social Security Number":
                social_security_number = '\n'.join(values)

        # Check if the data already exists in the table before
inserting
        existing_data = c.execute(f"SELECT * FROM {table_name}
WHERE name=? AND email=? AND phone_number=? AND
social_security_number=?", (name, email, phone_number,
social_security_number)).fetchone()
```

```
        # Insert segregated PII values into the appropriate
columns only if the data doesn't exist
        if not existing_data:
            c.execute(f"INSERT INTO {table_name} (name, email,
phone_number, social_security_number) VALUES (?, ?, ?, ?)",
                      (name, email, phone_number,
social_security_number))

    conn.commit()
    conn.close()


def display_pii_table(detected_pii, pii_score, file_path):
    print("\nDetected PII:")
    headers = ["File Name", "PII Type", "Count"]
    rows = [[file_path, pii_type_names[pii_type],
len(pii_values)] for (pii_type, _), pii_values in
detected_pii.items()]

    # Replace empty values with "NA" in the detected_pii
dictionary
    for key, values in detected_pii.items():
        if not values:
            detected_pii[key] = ["NA"]

    rows = [[file_path, pii_type_names[pii_type],
len(pii_values)] for (pii_type, pii_file), pii_values in
detected_pii.items()]

    print(tabulate(rows, headers=headers))

def fetch_web_content(url):
    try:
        response = requests.get(url)
        if response.status_code == 200:
            return response.text
```

```
        else:
            logging.warning(f"Failed to fetch web content from
{url} - Status code: {response.status_code}")
            return None
    except Exception as e:
        logging.error(f"Error fetching web content from {url} -
Error: {e}")
        return None

def extract_pii_from_web_content(content, url):
    detected_pii = {}
    data = json.loads(content)
    Name = []
    Email = []
    Phone = []
    for item in data:
        firstname = item["name"]["firstname"]
        lastname = item["name"]["lastname"]
        email = item["email"]
        phone = item["phone"]
        name = f"{firstname} {lastname}"
        Name.append(name)
        Email.append(email)
        Phone.append(phone)

    print( '\n\nExtracted Data From
https://fakestoreapi.com/users:' )
    print("{:<30}{:<30}{:<30}".format("Name", "Email",
"Phone"))
    for i in range(len(Name)):
        print("{:<30}{:<30}{:<30}".format(Name[i], Email[i],
Phone[i]))

    for key, pattern_info in patterns.items():
        if key != "Name":  # Exclude the "Name" pattern
```

```
            matches = re.findall(pattern_info["pattern"],
content)
            if matches:
                detected_pii[(key, url)] = matches

    return detected_pii

def extract_text_from_pdf(file_path):
    with open(file_path, 'rb') as file:
        pdf_reader = PyPDF2.PdfReader(file)
        text = ''
        for page_num in range(len(pdf_reader.pages)):
            text += pdf_reader.pages[page_num].extract_text()
        return text

def display_results(scanned_text, detected_pii, pii_score,
file_path):
    save_pii_to_database(detected_pii, file_path)
    display_pii_table(detected_pii, pii_score, file_path)

    detected_pii_counts = {pii_type_names[key[0]]: len(matches)
for key, matches in detected_pii.items() if key[0] != "Name" or
not file_path.endswith('.csv') and not
file_path.endswith('.pdf')}
    total_pii_counts = {pii_type_names[key]:
len(re.findall(pattern_info["pattern"], scanned_text)) for key,
pattern_info in patterns.items() if key != "Name" or not
file_path.endswith('.csv') and not file_path.endswith('.pdf')}

    # Ensure both dictionaries have the same keys
     all_keys = set(detected_pii_counts.keys()) |
set(total_pii_counts.keys())
    detected_pii_counts = {key: detected_pii_counts.get(key, 0)
for key in all_keys}
    total_pii_counts = {key: total_pii_counts.get(key, 0) for
key in all_keys}
```

```
    plt.figure(figsize=(10, 6))
    bar_width = 0.4
    index = range(len(detected_pii_counts))
    plt.bar(index, detected_pii_counts.values(), bar_width,
color='black', label='Detected PII')
    plt.bar([i + bar_width for i in index],
total_pii_counts.values(), bar_width, color='teal', alpha=0.7,
label='Total PII in Text')
    plt.xlabel('PII Type')
    plt.ylabel('Count')
    plt.title(f'Detected PII vs Total PII - {file_path}')
    plt.xticks([i + bar_width / 2 for i in index],
detected_pii_counts.keys(), rotation=90, ha='right')
    plt.legend()
    plt.tight_layout()
    plt.show()

    fig = go.Figure(go.Indicator(
        mode="gauge+number+delta",
        value=pii_score,
        domain={'x': [0, 1], 'y': [0, 1]},
        title={'text': f"PII Score Indicator - {file_path}",
'font': {'size': 24}},
        delta={'reference': 20, 'increasing': {'color':
"RebeccaPurple"}},
        gauge={
            'axis': {'range': [0, 500], 'tickwidth': 1,
'tickcolor': "darkblue"},
            'bar': {'color': "darkblue"},
            'bgcolor': "white",
            'borderwidth': 2,
            'bordercolor': "gray",
            'steps': [
                {'range': [0, 250], 'color': 'cyan'},
                {'range': [250, 400], 'color': 'royalblue'}],
```

```
            'threshold': {
                'line': {'color': "red", 'width': 4},
                'thickness': 0.75,
                'value': 490}}))
    fig.update_layout(paper_bgcolor="lavender", font={'color':
"darkblue", 'family': "Arial"})
    fig.show()

def display_web_results(detected_pii, pii_score, url):
    # Sanitize the URL to create a valid table name
    table_name = re.sub(r'\W+', '_', url)

    # Save PII data to the respective table
    save_pii_to_database(detected_pii, table_name)

    # Display detected PII for the URL
    print("\nDetected PII for URL:")
    headers = ["URL", "PII Type", "Count"]
    rows = [[url, pii_type_names[pii_type], len(pii_values)]
for (pii_type, _), pii_values in detected_pii.items() if
pii_type != "Name"]
    print(tabulate(rows, headers=headers))

    # Filter out "Name" entries from detected PII counts
    detected_pii_counts = {pii_type_names[key[0]]: len(matches)
for key, matches in detected_pii.items() if key[0] != "Name"}

    # Filter out "Name" entries from total PII counts
    total_pii_counts = {pii_type_names[key]:
len(re.findall(pattern_info["pattern"], content)) for key,
pattern_info in patterns.items() if key != "Name"}

    plt.figure(figsize=(10, 6))
    bar_width = 0.4
    index = range(len(detected_pii_counts))
```

```
    plt.bar(index, detected_pii_counts.values(), bar_width,
color='black', label='Detected PII')
    plt.bar([i + bar_width for i in index],
[total_pii_counts[key] for key in detected_pii_counts.keys()],
bar_width, color='teal', alpha=0.7, label='Total PII in Web
Content')
    plt.xlabel('PII Type')
    plt.ylabel('Count')
    plt.title(f'Detected PII vs Total PII - {url}')
    plt.xticks([i + bar_width / 2 for i in index],
detected_pii_counts.keys(), rotation=90, ha='right')
    plt.legend()
    plt.tight_layout()
    plt.show()

    fig = go.Figure(go.Indicator(
        mode="gauge+number+delta",
        value=pii_score,
        domain={'x': [0, 1], 'y': [0, 1]},
        title={'text': f"PII Score Indicator - {url}", 'font':
{'size': 24}},
        delta={'reference': 20, 'increasing': {'color':
"RebeccaPurple"}},
        gauge={
            'axis': {'range': [0, 500], 'tickwidth': 1,
'tickcolor': "darkblue"},
            'bar': {'color': "darkblue"},
            'bgcolor': "white",
            'borderwidth': 2,
            'bordercolor': "gray",
            'steps': [
                {'range': [0, 250], 'color': 'cyan'},
                {'range': [250, 400], 'color': 'royalblue'}],
            'threshold': {
                'line': {'color': "red", 'width': 4},
                'thickness': 0.75,
```

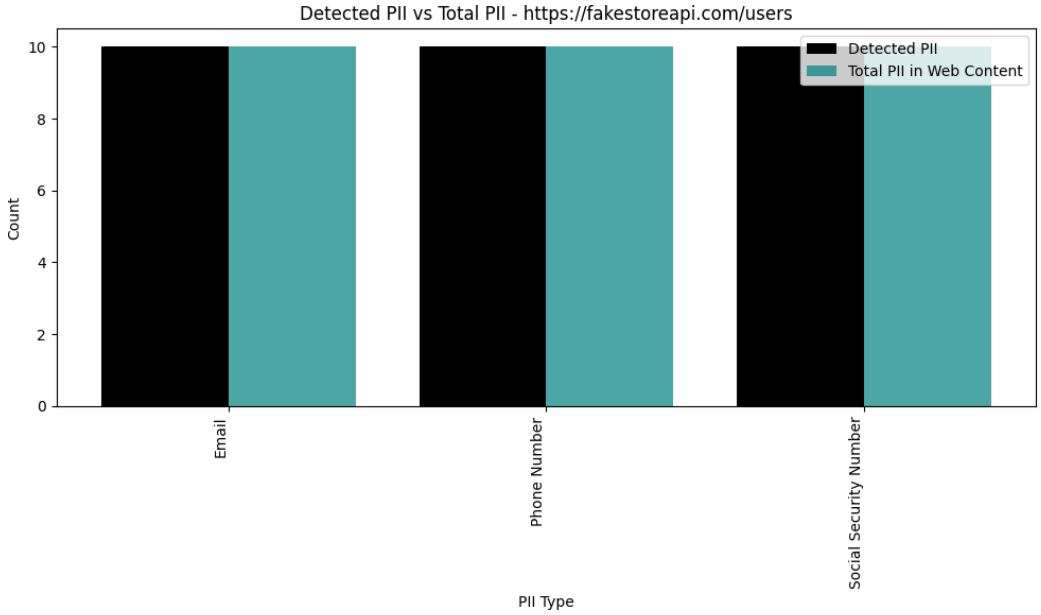
```
                'value': 490}}))
    fig.update_layout(paper_bgcolor="lavender", font={'color':
"darkblue", 'family': "Arial"})
    fig.show()

def scan_system_for_pii(file_paths):
    scanned_texts = {}
    for file_path in file_paths:
        if file_path.endswith('.csv'):
            with open(file_path, newline='') as csvfile:
                reader = csv.reader(csvfile)
                text = ''
                for row in reader:
                    text += ' ' + row[0] + '\t' + row[1] + '\t'
+ row[2] + '\n'
                scanned_texts[file_path] = text
                print(f' \n\nExtracted Data From: {file_path}')
                print(text)
        elif file_path.endswith('.pdf'):
             scanned_texts[file_path] =
extract_text_from_pdf(file_path)
        else:
            with open(file_path, "r") as file:
                text_to_scan = file.read()
            scanned_texts[file_path] = text_to_scan
    return scanned_texts

user_consent = input("Do you agree to a PII scan of your system
and websites? (yes/no): ").lower()

if user_consent == "yes":
    file_paths = ["PII_Data.txt", "PII_Data_2.txt",
"MSMEANNUALREPORT2022.pdf", "datainfo.csv"] # Example: Add
more file paths as needed
    urls = ["https://fakestoreapi.com/users"] # Example: Add
more URLs as needed
```

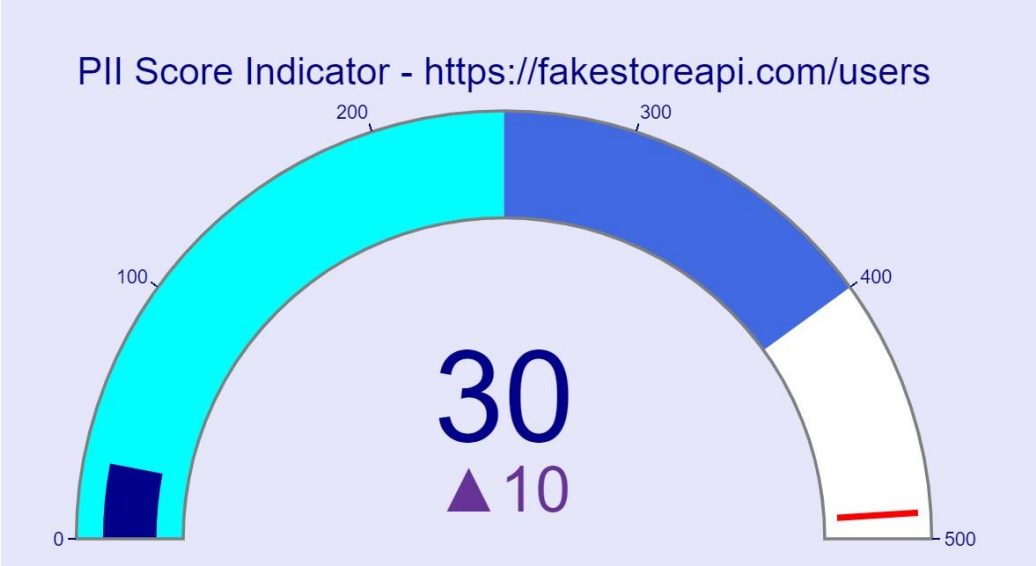
```
    for url in urls:
        content = fetch_web_content(url)
        if content:
            detected_pii =
extract_pii_from_web_content(content, url)
            pii_score = sum(len(matches) *
patterns[pii_type]["weight"] for (pii_type, _), matches in
detected_pii.items())

            display_web_results(detected_pii, pii_score, url)

            logging.info(f"PII Scan for URL: {url}")
            logging.info(f"Detected PII: {detected_pii}")
            logging.info(f"PII Score: {pii_score}")

            if pii_score > 20:
                print(f"For {url}: Your weighted PII score is
high. Consider taking actions to improve system security.")
                if "Social Security Number" in detected_pii:
                    print("Detected Social Security Numbers.
Follow best practices to secure this sensitive information.")
            else:
                print(f"For {url}: Your weighted PII score is
low. You can still take steps to enhance system security.")

        else:

            print(f"Failed to fetch web content from {url}")

    for file_path, scanned_text in
scan_system_for_pii(file_paths).items():
        detected_pii = detect_pii(scanned_text, file_path)
        pii_score = sum(len(matches) *
patterns[pii_type]["weight"] for (pii_type, _), matches in
detected_pii.items())
```

```
        display_results(scanned_text, detected_pii, pii_score,
file_path)

        logging.info(f"PII Scan for File: {file_path}")
        logging.info(f"Detected PII: {detected_pii}")
        logging.info(f"PII Score: {pii_score}")

        if pii_score > 20:
            print(f"For {file_path}: Your weighted PII score is
high. Consider taking actions to improve system security.")
            if "Social Security Number" in detected_pii:
                print("Detected Social Security Numbers. Follow
best practices to secure this sensitive information.")
        else:
            print(f"For {file_path}: Your weighted PII score is
low. You can still take steps to enhance system security.")

        print(f"For solutions, you can visit Galaxkey
Enterprises: https://www.galaxkey.com/")

else:
    print("PII scanning requires user consent. If you have
concerns, please review the privacy policy.")
```

Output: Consent – YES

```
Extracted Data From https://fakestoreapi.com/users:
Name                            Email                       Phone
john doe                        john@gmail.com              1-570-236-7033
david morrison                  morrison@gmail.com          1-570-236-7033
kevin ryan                      kevin@gmail.com             1-567-094-1345
don romer                       don@gmail.com               1-765-789-6734
derek powell                    derek@gmail.com             1-956-001-1945
david russell                   david_r@gmail.com           1-678-345-9856
miriam snyder                   miriam@gmail.com            1-123-943-0563
william hopkins                 william@gmail.com           1-478-001-0890
kate hale                       kate@gmail.com              1-678-456-1934
jimmie klein                    jimmie@gmail.com            1-104-001-4567

Detected PII for URL:
URL                             PII Type                      Count
------------------------------  ----------------------  -------
https://fakestoreapi.com/users  Email                         10
https://fakestoreapi.com/users  Phone Number                  10
https://fakestoreapi.com/users  Social Security Number        10
```

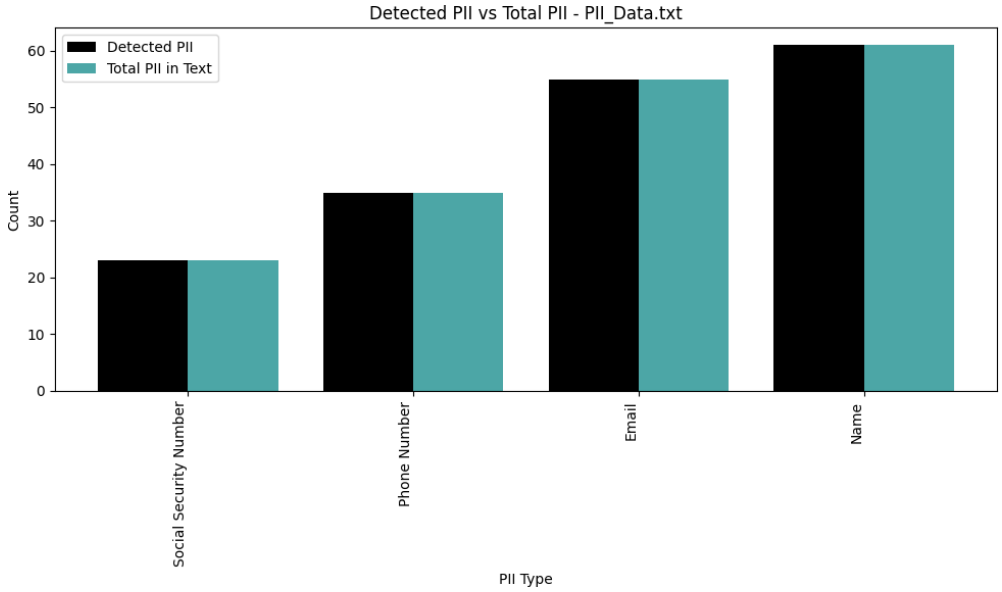


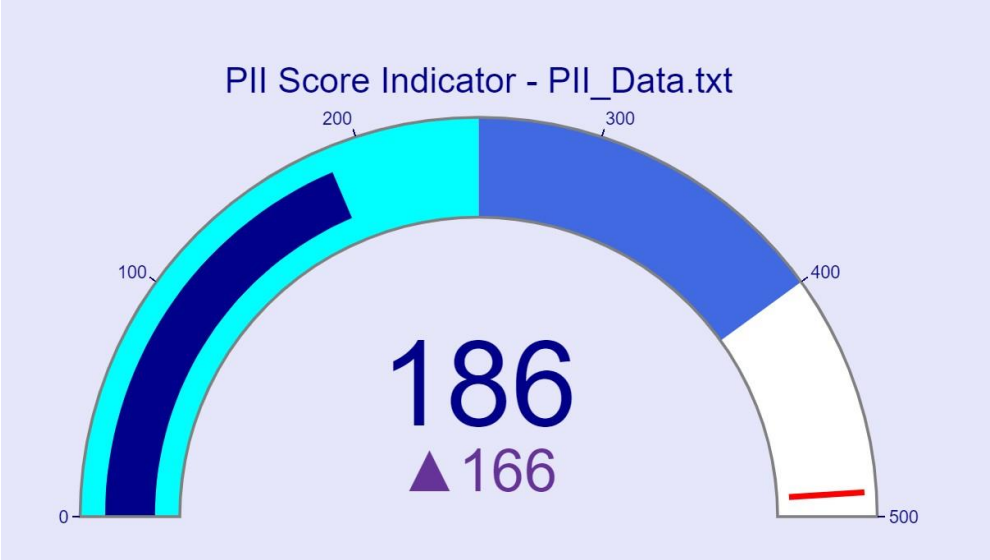


For https://fakestoreapi.com/users: Your weighted PII score is high. Consider taking actions to improve system security.

Detected PII:

File Name	PII Type	Count
PII_Data.txt	Name	61
PII_Data.txt	Email	55
PII_Data.txt	Phone Number	35
PII_Data.txt	Social Security Number	23

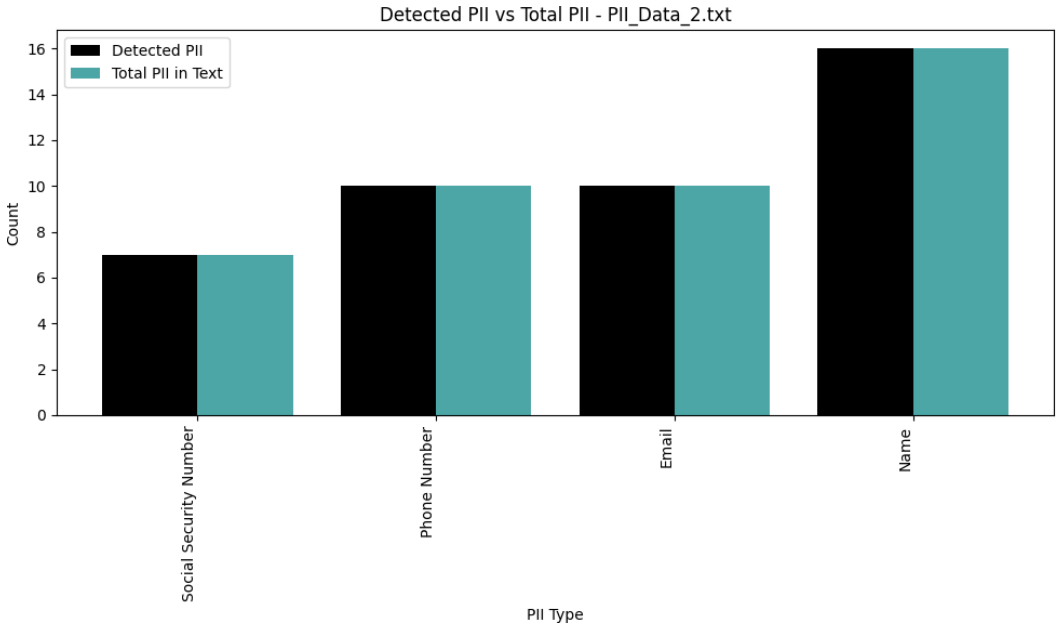


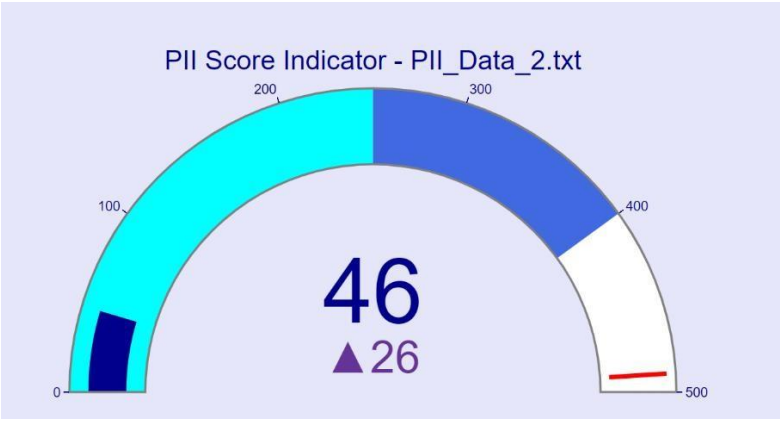


For PII_Data.txt: Your weighted PII score is high. Consider taking actions to improve system security.
For solutions, you can visit Galaxkey Enterprises: https://www.galaxkey.com/

Detected PII:

File Name	PII Type	Count
PII_Data_2.txt	Name	16
PII_Data_2.txt	Email	10
PII_Data_2.txt	Phone Number	10
PII_Data_2.txt	Social Security Number	7

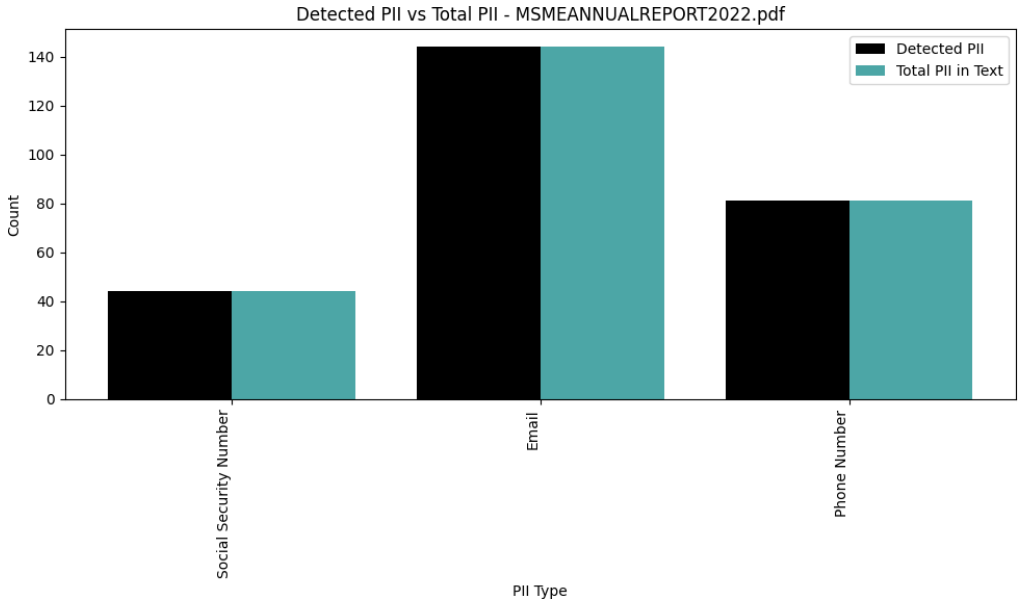


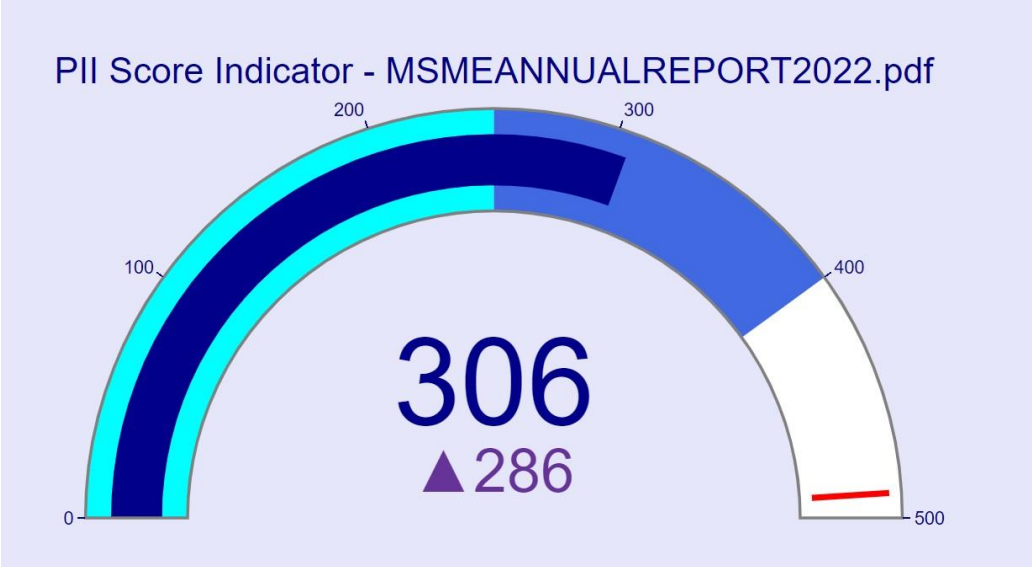


For PII_Data_2.txt: Your weighted PII score is high. Consider taking actions to improve system security.
For solutions, you can visit Galaxkey Enterprises: https://www.galaxkey.com/

Detected PII:

File Name	PII Type	Count
MSMEANNUALREPORT2022.pdf	Email	144
MSMEANNUALREPORT2022.pdf	Phone Number	81
MSMEANNUALREPORT2022.pdf	Social Security Number	44

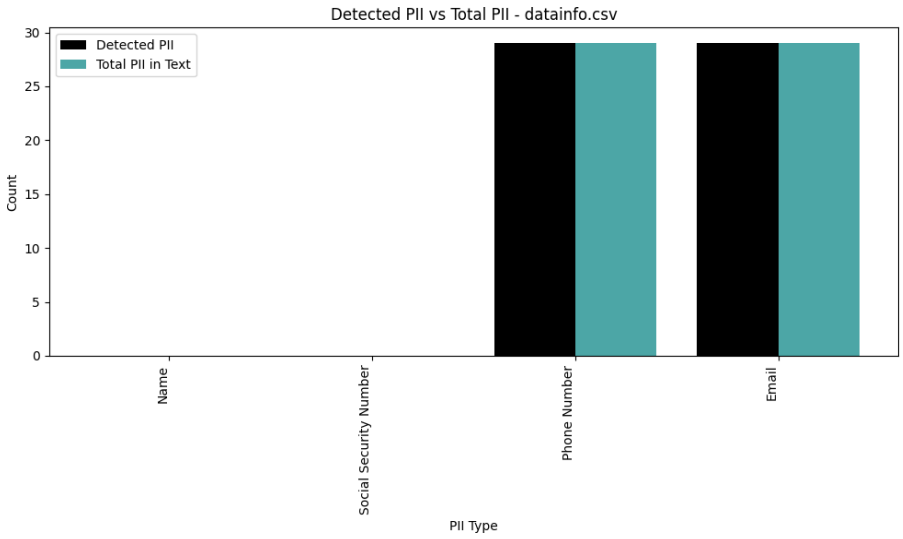


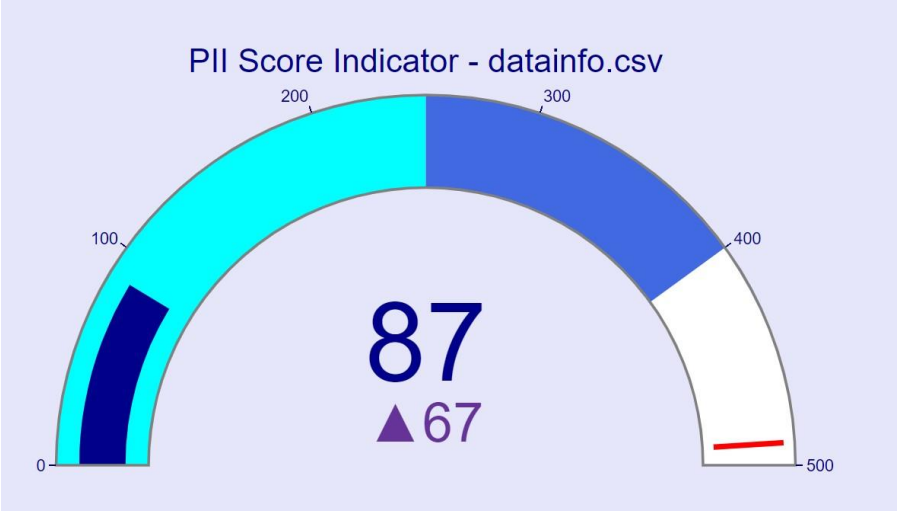


For MSMEANNUALREPORT2022.pdf: Your weighted PII score is high. Consider taking actions to improve system security.
For solutions, you can visit Galaxkey Enterprises: https://www.galaxkey.com/

```
Extracted Data From: datainfo.csv
 phone number    name     email
 9765432890      David    ydjrdqxj@example.com
 9546781230      Albert   ibnktb@example.com
 9856321470      Jhon     fzhdlf@example.com
 9123456789      Clave    dfqgvjlr@example.com
 9745698231      Mike     qnyvnz@example.com
 9876543210      Sam      rdfypvsgu@example.com
 9134567890      Tim      attcaxu@example.com
 9567890123      Fate     jgiqfvs@example.com
 9245678901      Ram      ihlxzmrhpr@example.com
 9345678901      Tonny    ppjbqgthje@example.com
 9761234567      Jack     rgbeoe@example.com
 9845678901      Kim      jhfufiw@example.com
 9276543210      Akram    bqtdbkko@example.com
 9789012345      Tuins    lhvpebndyb@example.com
 9156789012      Vivek    yoyrsx@example.com
 9870123456      Soham    hovvv@example.com
 9287654321      Donald   srugzq@example.com
 9734567890      Piyush   zytdxhpz@example.com
 9187654321      Baharat  juogweik@example.com
 9678901234      Rohit    ldwzwlethz@example.com
```

```
Detected PII:
File Name      PII Type         Count
-------------  -------------  -------
datainfo.csv   Email                29
datainfo.csv   Phone Number         29
```






```
For datainfo.csv: Your weighted PII score is high. Consider taking actions to improve system security.
For solutions, you can visit Galaxkey Enterprises: https://www.galaxkey.com/
```

Output: Consent – No

```
Do you agree to a PII scan of your system? (yes/no): no
PII scanning requires user consent. If you have concerns, please review the privacy policy.
```

5. Encountered Challenges: -

- One of the most significant challenges we had to face was the difficulty to gather data that contained PII. More importantly, the data containing PII was not readily available as this type of information is too sensitive to be exposed through standard data sources.
- Moreover, the majority of opensource platforms did not have PII-containing datasets due to security concerns. Alternatively, either data sources where it was available for a fee of other limitations were found. Hence, finding a proper dataset containing PII was challenging
- Complications with Regex Pattern Matching: One of the other major problems that we encountered was the preparation of regex patterns for PII identification. The PII data is formatted differently, which made them in the pattern process. For example, the address formatting for the USA and India are different. Poorer variation, such as the inclusion of a number in the name, made the regex pattern match fairly more complex.
- PII Identified based on Context: It was tough to find PII based on the document context. This could be solved if machine learning techniques used for it.

6. Conclusions: -

Moreover, the identification of people may harm their privacy due to the highly sensitive data set that PII represents. As a result, all entities and persons must understand that PII cannot fall into the wrong hands, and steps, such as protection mechanisms against unauthorized access, use, and disclosure, are necessary. Among other measures, one can identify the use of strong security measures such as encryption and access control, frequent testing of systems for vulnerabilities, and regular training of staff.

According to the Principle of People First, the consequences of failing to protect PII are generally negative and may involve damages to reputation, loss of financial resources, or legal implications. For this reason, all entities must remember that people first must be the first principle for all individuals and groups.

7. References: -

- Al-Roubaiey, A., Al-Sadi, J., & Abubaker, M. (2022). A Review of Personal Identifiable Information (PII) and Its Challenges. *International Journal of Computer Science and Network Security, 22*(6), 32-37.
- European Union Agency for Cybersecurity. (2021). Good practices for data protection. Retrieved from https://www.enisa.europa.eu/publications/good-practices-for-data-protection/good-practices-for-data-protection/view
- National Institute of Standards and Technology. (2021). NIST Privacy Framework: A Tool for Improving Privacy through Enterprise Risk Management. *NIST Special Publication 800-37 Rev. 2*.
- Office of the Privacy Commissioner of Canada. (2020). What is personal information? Retrieved from https://www.priv.gc.ca/en/privacy-topics/collecting-personal-information/02_05_d_16/
- Office of the Privacy Commissioner of Canada. (2021). Privacy and Cyber Security: Emphasizing Privacy Protection in Cyber Security Activities. Retrieved from https://www.priv.gc.ca/en/privacy-topics/technology-and-privacy/cyber-security-and-privacy/
- Solove, D. J. (2011). *Understanding privacy*. Harvard University Press.
- U.S. Department of Commerce. (2021). Privacy Shield Framework. Retrieved from https://www.privacyshield.gov/welcome