

Effective Anonymization of Sensitive Data in the Large-Scale Systems Using Privacy Enhancing Technology

MSc Research Project
Cloud Computing

Tarini Beeruka
Student ID: 21117314

School of Computing
National College of Ireland

Supervisor: Dr. Punit Gupta


**National College of Ireland
Project Submission Sheet
School of Computing**



Student Name:	Tarini Beeruka
Student ID:	21117314
Programme:	Cloud Computing
Year:	2022
Module:	MSc Research Project
Supervisor:	Dr. Punit Gupta
Submission Due Date:	01/02/2023
Project Title:	Effective Anonymization of Sensitive Data in the Large-Scale Systems Using Privacy Enhancing Technology
Word Count:	1334
Page Count:	7

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

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

Signature:	
Date:	1st February 2023

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST:

Attach a completed copy of this sheet to each project (including multiple copies).	<input type="checkbox"/>
Attach a Moodle submission receipt of the online project submission , to each project (including multiple copies).	<input type="checkbox"/>
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.	<input type="checkbox"/>

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):	

Effective Anonymization of Sensitive Data in the Large-Scale Systems Using Privacy Enhancing Technology

Tarini Beeruka
21117314

Table of Content

1	Introduction	2
2	Requirements	2
3	Installation	3
4	Connecting to S3 Bucket	4
5	Splitting and encryption	4
6	Opting the results from Postman	5
7	Evaluating the result	6
8	Final Results	6

1 Introduction

Data Anonymization is the process of securing confidential or sensitive information by hiding or encrypting identifiers that connect a specific person to stored data. Various privacy-preserving strategies, methodologies, frameworks, and prototypes have been proposed or developed for disclosing data while preserving user privacy.

The author developed an optimal anonymization tool that fetches the dataset, splitting the data based on sensitive and personal attributes, only sending essential columns for computation while adding encryption for the files for added security. For greater performance and security, the Amazon S3 bucket is used for storing and retrieving data and CryptPandas has been used for the encryption and decryption of pandas data frames. The author compares the datasets after using a data anonymization tool and the raw dataset. Execution time and memory consumption are the parameters considered in this study, the well-being of patient is shown as output after performing computation of dataset. According to the results, the dataset showed a 45.5 percentage decrease in execution time and a 33.3 percentage decrease in memory consumption after utilizing the data anonymization tool.

2 Requirements

1. Set up Python environment.
2. Set up Postman API platform from figure 2
3. Set up AWS account with access to S3 buckets from figure 1

To check the version of python:

```
$ python --version
```

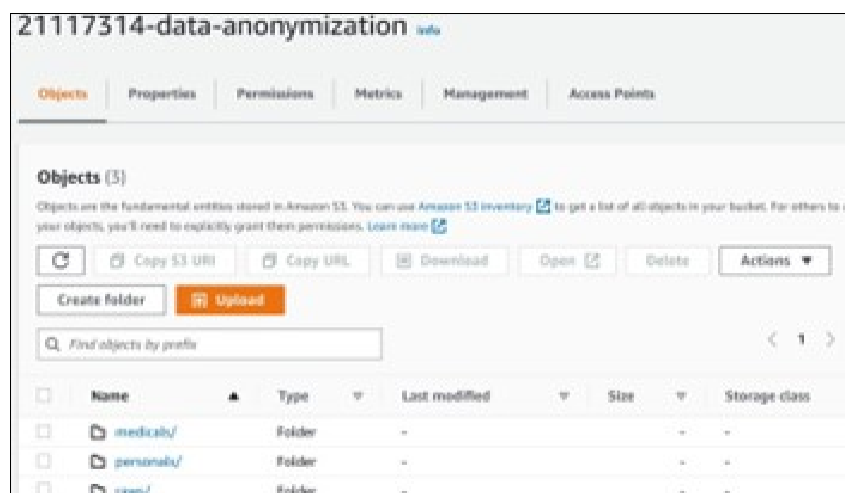


Figure 1: Author's S3 setup and access to Buckets

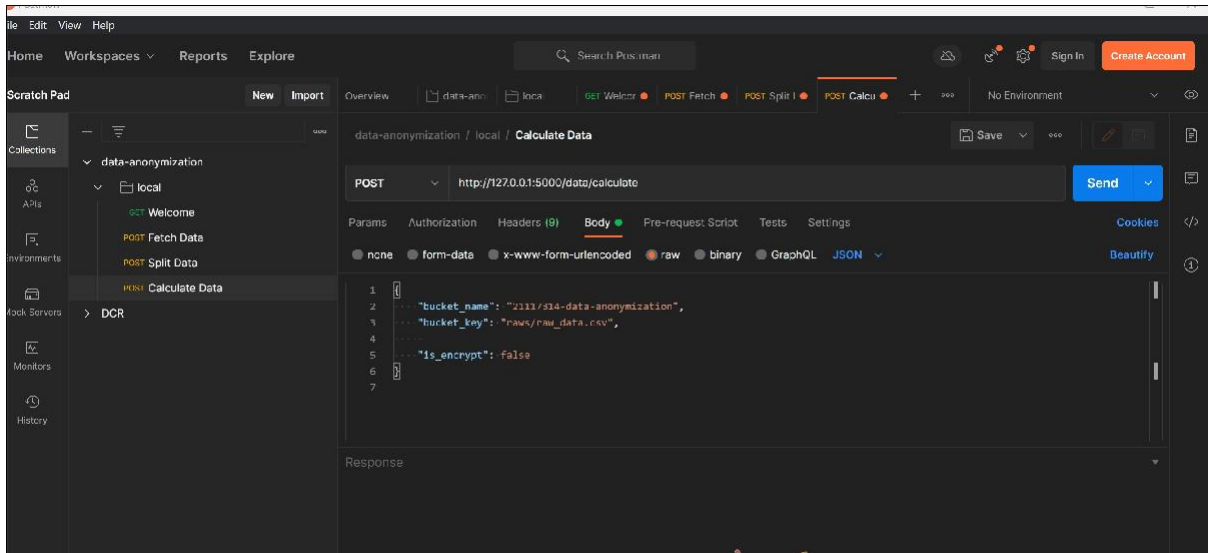


Figure 2: Author’s Postman API

3 Installation

Install the required libraries are mentioned in the figure 3.

```
main.py > ...
1  from flask import Flask, jsonify, request, g, Response
2  from boto3 import client
3  import boto3
4  import pandas as pd
5  import cryptopandas as crp
6  import sys
7  import time
8  import math
9  import os, psutil
10 import json
11 import botocore
```

Figure 3: Required libraries

1. Python-based Flask is a microweb framework. Due to the fact that it doesn’t require any special methods or libraries, it is categorized as a microframework. (Garnaat, 2011)
2. The flask.json module in Flask contains the function jsonify. In a Response object with the implementation mimetype, jsonify serializes data to the JavaScript Object Notation (JSON) standard. (VanderPlas et al., 2018b)

3. The Python application, library, or script could be seamlessly integrated with AWS services like Amazon S3, Amazon DynamoDB, and more. (Saeed, Baras and Hajjdiab, 2019)

4 Connecting to S3 Bucket

```
conn = client('s3')
s3_resource = boto3.resource('s3')

obj = conn.get_object(
    Bucket = bucket_name,
    Key = bucket_key
)
body = obj['Body']
csv_string = body.read().decode('utf-8')
```

Figure 4: Connecting to S3

```
conn = client(' s3' )
s3_resource = boto3.resource(' s3' )

obj = conn.get_object(
    Bucket = bucket_name,
    Key = bucket_key
)
body = obj[' Body' ]
csv_string = body.read().decode(' utf-8' )

# Load test CSV
df = pd.read_csv(StringIO(csv_string))

# Add Patient ID column
df.insert(0, ' patient_id' , range(0, 0 + len(df)))
```

5 Splitting and encryption

Figure 5 gives the information on splitting the data based on attributes. Based on the variables BMI and Smoker, the health of the patient is determined. the memory utilization is estimated, although Flask will correctly update the content-length if response.set

```

# Filtering unexpected columns based on whitelist
for val in columns_name:
    if (val.lower() not in personal_record_col_wl):
        personal_record_col_bl.append(val)

    if (val.lower() not in medical_record_col_wl):
        medical_record_col_bl.append(val)

# Duplicate, export to different CSV and encrypt it
if output_key_personal:
    df_personal_record = df.copy()
    df_personal_record.drop(personal_record_col_bl, axis = 1, inplace=True)
    df_personal_record_csv_buffer = StringIO()
    df_personal_record.to_csv(df_personal_record_csv_buffer, index=False)
    s3_resource.Object(bucket_name, output_key_personal).put(Body=df_personal_record_csv_buffer.getvalue())

if output_key_personal_encrypt:
    crp.to_encrypted(df_personal_record, password=local_key_personal_encrypt_pwd, path=local_key_personal_encrypt)
    s3_resource.Bucket(bucket_name).upload_file(local_key_personal_encrypt, output_key_personal_encrypt)

```

Figure 5: Performing splitting of the data based on labels

However, when it comes to measuring memory utilization per request, they both face the same challenge. One Python worker processes a lot of requests (concurrently or sequentially) during the course of his lifetime, which is why. So it's challenging to determine a request's precise memory use. The author calculates the increase in percentage between two sets of raw information and the anonymized data using the original formula. The results are then examined based on each parameter to comprehend how the tool should be used. Further more, the parameters are individually compared for greater comprehension

```

$ # Define whitelist
personal_record_col_wl = ["patient_id", "medical_id", "age", "sex", "
    ↔ children", "region", "charges", "first_name", "last_name", "
    ↔ email", "phone_number", "religion", "race"]
medical_record_col_wl = ["patient_id", "medical_id", "age", "bmi", "
    ↔ smoker", "charges"]

# Get columns name
columns_name = df.columns.values

```

6 Opting the results from Postman

```

$ {
    "bucket_name": "21117314-data-anonymization",
    "bucket_key": "raws/raw_data.csv",

    "is_encrypt": false
}

```

Figure6 gives the data about the values of the result obtained.

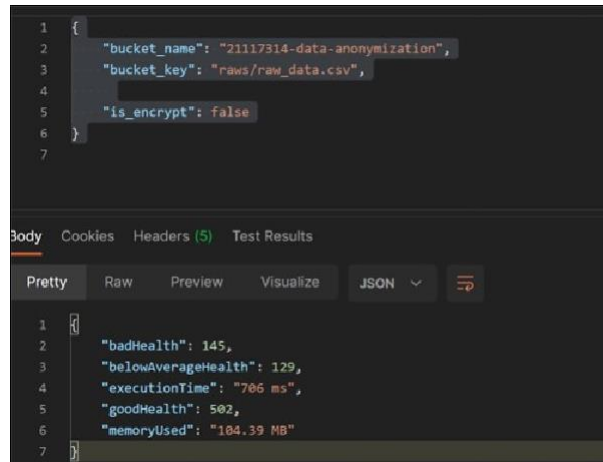


Figure 6: Fetching the results from Postman

7 Evaluating the result

Through figure 7, two significant results are found during evaluation for the performance measures. The patient’s health comes first, followed by memory requirements and the time needed to execute the findings. The comparison is based on the assessment of raw data while accounting for the patient’s well-being, obtaining its memory consumption and execution time, and using medical data that has already been anonymized while also obtaining the same comparison-based metrics

Results Obtained	Raw Data	Medical Data
badHealth	145	145
belowAverageHealth	129	129
executionTime in ms	1055	335
goodHealth	502	502
memoryUsed in MB	43.867	37.5

Figure 7: Evaluating the result

8 Final Results

1. When the raw data is directly executed the result has been 1055ms and the anonymized data show 355ms. It clearly says, as larger data set is being processed the execution time is more. Therefore it is suggestable to use data anonymization tool from figure 8

2. The comparison of memory usage of the raw and anonymized data bar chart illustrated below figure. The clearly shows that memory usage of raw data was more than Medical data from figure 9
3. Comparison of memory usage of the raw and anonymized data are shown in the bar chart in above Bar graph. This is the prime example that, results have been accurate which tells us that it is no need to have personal data items in the dataset, and it is better to eliminate if the evaluation does include personal data from figure 10.

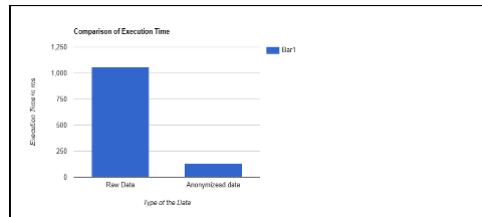


Figure 8: Comparison of the Execution Time in ms

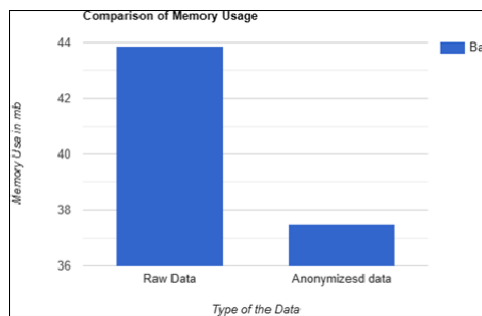


Figure 9: Comparison of the Memory usage in MB

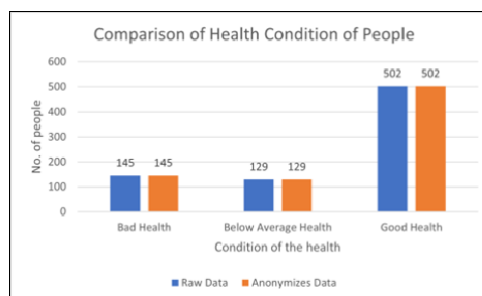


Figure 10: Result Summary on comparing the results obtained by Raw and Anonymized Data

References

Garnaat, M. (2011) *Python and AWS Cookbook*. O'Reilly Media.

Saeed, I., Baras, S. and Hajjdiab, H. (2019) "Security and Privacy of AWS S3 and Azure Blob Storage Services," *2019 IEEE 4th International Conference on Computer and Communication Systems (ICCCS)* [Preprint]. Available at: <https://doi.org/10.1109/ccoms.2019.8821735>.

VanderPlas, J. *et al.* (2018) "Altair: Interactive Statistical Visualizations for Python," *Journal of Open Source Software*, 3(32), p. 1057. Available at: <https://doi.org/10.21105/joss.01057>.