SOUTHERN JOURNAL OF SCIENCES

ESTABLISHED IN 1993

Original research paper

PRIVACY-PRESERVING DATA ANONYMIZATION TOOL FOR MEDICAL DATA

FERRAMENTA DE ANONIMIZAÇÃO DE DADOS MÉDICOS COM PRESERVAÇÃO DE PRIVACIDADE

BORKAKOTY, Sangeeta[1]; ISLAM, Atowar Ul[2]*; BORA, Kanak Chandra[3]

[1]University of Science & Technology Meghalaya, Department of Computer Science. India. ORCID: 0009-0003-1554-9717

[2]University of Science & Technology Meghalaya, Department of Computer Science. India. ORCID: 0000-0001-5345-0556

[3]University of Science & Technology Meghalaya, Department of Computer Science. India. ORCID: 0009-0009-0491-1235

** Corresponding author: e-mail: atowar91626@gmail.com*

Received xxx; received in revised form xxx; accepted xxx

ABSTRACT

Background: Medical institutions collect a vast amount of sensitive patient data for personalized treatments and health trend analysis. However, this raises concerns regarding the privacy of patient data, as it contains sensitive and confidential information. **Aims:** Develop an anonymization tool using diverse techniques to protect data while preserving its utility. **Methods:** A Python-based data anonymization tool for medical datasets supporting both categorical and numerical data is developed. It employs various methods, including data perturbation, binning, scaling, transformation, and differential privacy. **Results:** The tool was able to anonymize sensitive data, both categorical and numerical, while preserving its utility for further analysis. **Discussion:** The Privacy-Preserving Data Anonymization Tool advances sensitive medical data management by anonymizing both categorical and numerical data using various techniques while retaining data utility. **Conclusions:** The Anonymization Tool addresses patient data privacy concerns by balancing data utility with privacy, enabling secure medical data use in research.

Keywords: *anonymization, privacy-preservation, medical dataset, data utility, data analytics.*

RESUMO

Introdução: As instituições médicas coletam vastas quantidades de dados sensíveis de pacientes para tratamentos personalizados e análise de tendências de saúde. No entanto, isso levanta preocupações quanto à privacidade dos dados dos pacientes, pois contêm informações sensíveis e confidenciais. **Objetivos**: Desenvolver uma ferramenta de anonimização usando diversas técnicas para proteger dados enquanto preserva sua utilidade. **Métodos**: Uma ferramenta de anonimização de dados baseada em Python para conjuntos de dados médicos que suporta dados categóricos e numéricos foi desenvolvida. Ela emprega vários métodos, incluindo perturbação de dados, binning, escalonamento, transformação e privacidade diferencial. Resultados: A ferramenta foi capaz de anonimizar os dados sensíveis, tanto categóricos quanto numéricos, preservando ao mesmo tempo sua utilidade para análises posteriores. **Discussão**: A Ferramenta de Anonimização de Dados com Preservação de Privacidade avança o gerenciamento de dados médicos sensíveis ao anonimizar dados categóricos e numéricos usando várias técnicas enquanto mantém a utilidade dos dados. **Conclusões**: A Ferramenta de Anonimização aborda as preocupações com a privacidade dos dados dos pacientes equilibrando utilidade dos dados com privacidade, permitindo o uso seguro de dados médicos em pesquisas.

Palavras-chave: *anonimização, preservação de privacidade, conjunto de dados médicos, utilidade de dados, análise de dados.*

DOI: 10.48141/2764-5959.05.v33.n40.2025_05_BORA_pgs_75_85.pdf

1. INTRODUCTION:

Medical institutions collect sensitive patient information, including personal identification, medical histories, and genetic data, to enable personalized treatments, enhance diagnostic accuracy, and analyze health trends at both individual and population levels. This wealth of data holds the potential to drive significant advancements in medical research, improve patient outcomes, and inform public health policies. However, it also raises serious privacy concerns due to risks of unauthorized access, data breaches, and misuse. These issues are exacerbated by the increasing frequency and sophistication of cyberattacks targeting healthcare facilities, which can compromise patient trust and hinder the adoption of data-driven healthcare innovations.

Moreover, ethical challenges arise from the lack of transparency regarding data usage and sharing. Patients are often unaware of how their data is being stored, processed, or shared with third parties, raising concerns about consent and ownership. This highlights the critical need for robust privacy protections and ethical frameworks to ensure data is handled responsibly. Failure to address these concerns not only jeopardizes patient confidentiality but also threatens the integrity of the healthcare system.

There are two widely accepted methods to combat privacy breaches and ensure data privacy: data encryption and anonymization. In data encryption, the data is transformed into a ciphertext using cryptographic algorithms and keys. This makes the data unreadable to unauthorized. The encrypted data can be decrypted only by authorized parties who possess the corresponding decryption keys. This method ensures that the data remains protected if it is intercepted or accessed without authorization. However, data encryption is generally unsuitable when data needs to be frequently shared after processing. Data encryption and decryption add complexity and computational overhead to data-handling processes. Moreover, key management and ensuring compatibility and interoperability between systems and organizations may pose challenges.

In anonymization, the data is selectively removed or altered to prevent the identification of individuals and other details. However, if done indiscriminately, it can remove or conceal valuable information, thereby reducing the usefulness or accuracy of the data for analysis and decision-making. Various anonymization techniques have been developed to ensure data privacy without compromising its usability. Dhawas *et al.* (2024) have provided a survey of existing anonymization techniques, analyzing their advantages and disadvantages. Olatunji *et al.* (2022) have done an extensive review of anonymization for Healthcare Data.

Similar studies done by Kargupta *et al.* (2003) and Muralidhar & Sarathy (1999) focus on specific data anonymization techniques, like random data perturbation (RDP) methods.

One practical application of data analytics is in recommendation systems, widely used by e-commerce sites to suggest products to customers based on their buying habits. While such data analytics is valuable for business decision-making, it raises serious privacy concerns if it falls into the wrong hands. In this context, Murthy *et al.* (2019) have compared five anonymization techniques using the same dataset, reviewing the strengths and weaknesses of each approach. Marques & Bernardino (2020) have reviewed several anonymization techniques and efficient software tools, aiming to understand which methods offer higher levels of anonymization, along with their respective strengths and weaknesses. In the same lines, Majeed & Lee (2021) conducted a comprehensive survey of anonymization techniques for privacy-preserving data publishing, examining potential attacks on sanitized data and the different stakeholders involved in the anonymization process.

At a very basic level, privacy is about keeping personal information away from unauthorized access. It is essential for individual autonomy, individualism, and respect. There are four types of privacy: information, bodily, territorial, and communication.

This paper focuses on information privacy, which includes systems and infrastructures that collect, analyze, process, and publish users' data.

1.1 Aim of the project

To address the challenges of preserving privacy while maintaining the functionality of data, this paper aims to develop a Privacy-Preserving Anonymization tool for medical datasets, designed to safeguard sensitive patient information while enabling its meaningful use in research and healthcare analytics. This tool will employ anonymization techniques such as pseudonymization and binning to protect Personal Identifiable Information (PII) and other sensitive data while ensuring the functionality and utility of the datasets remain intact. Pseudonymization

replaces identifiable attributes with pseudonyms or unique codes, making it difficult to trace the information back to individuals without access to the specific mapping keys. Binning will group continuous or high-resolution data, such as ages or numerical test results, into broader categories or ranges, reducing the risk of re-identification while preserving the dataset's analytical value.

By integrating these techniques, the tool will strike a balance between robust privacy protection and the usability of the data for research and analysis. This approach ensures that medical datasets can be shared or analyzed without compromising patient confidentiality, enabling the secure and ethical use of health information across applications such as medical research, public health studies, and policy development.

2. MATERIALS AND METHODS:

2.1. Materials

The tool was developed in a Windows environment using Python version 3.12.2, selected for its comprehensive data processing and web application capabilities. Key libraries include Pandas for data manipulation, Flask for creating the web interface, Nest Asyncio for resolving conflicts in the event loop, and Random for pseudonymization. The application was programmed and tested using Jupyter Notebook version 7.1.3.

2.2. Methods

The anonymization techniques chosen for this tool are - binning for numerical data and pseudonymization for textual data. These are particularly well-suited given the specific use case and application context. One of the key reasons for their selection is their simplicity and ease of implementation. Unlike advanced techniques such as k-anonymity or differential privacy, binning and pseudonymization are straightforward to apply and require minimal computational resources. This makes them ideal for scenarios where quick, efficient anonymization is necessary, without relying on complex configurations or dependencies.

Additionally, these techniques excel in preserving the utility of the data. Binning obfuscates specific numerical values by grouping them into ranges, ensuring that patterns and trends remain observable, which is essential for exploratory data analysis or visualization. Pseudonymization, on the other hand, replaces text data with randomly generated identifiers while maintaining its structural integrity. This approach preserves categorical distinctions, making it highly effective in cases where understanding groupings or classifications is crucial, even when the actual identifiers are hidden.

The focus of this tool is on lightweight privacy needs, making these techniques particularly suitable. The use case involves anonymizing a dataset for general-purpose sharing or analysis rather than meeting the stringent privacy guarantees demanded in sensitive domains such as healthcare or finance. Advanced methods like differential privacy, while offering robust guarantees, are often unnecessary for simpler datasets and can be challenging to configure correctly. In contrast, binning and pseudonymization provide a practical and efficient solution.

Moreover, these techniques perform well in small-scale applications where computational efficiency is a priority. Methods such as k-anonymity or differential privacy can introduce performance bottlenecks, especially when deployed on lightweight systems or in real-time processing scenarios. The chosen techniques strike a balance between performance and effectiveness, ensuring the tool remains efficient and user-friendly.

The alignment of these techniques with the dataset's data types further justifies their use. Binning works effectively for numeric data by masking specific values while retaining ranges relevant for analysis. Pseudonymization, tailored for string data, successfully anonymizes identifiable text while preserving distinct categories. This alignment ensures that the anonymization process is both meaningful and contextually appropriate. Finally, the scalability and adaptability of these methods make them suitable for varying datasets. Without extensive preprocessing or parameter tuning, they can be applied across different structures and sizes, ensuring the tool's versatility for general-purpose applications.

2.2.1. Implementation of the Tool

The program is a Flask-based web application designed for uploading and processing CSV files. Its primary goal is to provide users with the ability to:

- Upload a CSV file and view its attributes (column names and data types).

- Select specific columns (features) for anonymization or binning:
  - Textual data is pseudonymized by replacing values with random alphanumeric strings.
  - Numeric data is binned into ranges.
- Download the anonymized version of the CSV file.

The program employs user-friendly HTML interfaces and leverages pandas for data manipulation.

2.2.2 Source Codes

Firstly, the necessary libraries are imported.

- **pandas:** For CSV file manipulation.
- **Flask:** To create a web application.
- **nest_asyncio:** To handle asyncio event loops, especially in environments like Jupyter Notebook.
- **multiprocessing:** Allows running the Flask app as a separate process.
- **random & string:** Generate random strings for pseudonymization.
- **traceback:** For detailed error logging.

```
import pandas as pd
from flask import Flask, request,
render_template_string, send_file
import nest_asyncio
from multiprocessing import Process
import random
import string
import traceback
```

The nest_asyncio.apply() function is called to resolve potential event loop conflicts, making the application more robust. Then, the Flask app is initialized with app = Flask(__name__), which creates an instance of the web application.

```
# Applying nest_asyncio to avoid event
loop issues in Jupyter Notebook
nest_asyncio.apply()

# Creating the Flask app
app = Flask(__name__)
```

Next, two utility functions are defined to handle data anonymization.

The pseudonymize function replaces sensitive textual data with random strings. For every input text, it generates an 8-character alphanumeric string using Python's random.choices() function. This ensures that text data remains anonymous while maintaining uniformity across the dataset.

```
# Function for pseudonymizing text
(replacing with random strings)
def pseudonymize(text):
    return
''.join(random.choices(string.ascii_letter
s + string.digits, k=8))
```

The binning function divides numerical data into equal ranges, known as bins. The range of values in the dataset is divided into bins (default: 5), and each numeric value is replaced by a range string (e.g., 10-20). This reduces the granularity of numerical data while preserving its distribution pattern.

```
# Function for binning numeric data
def binning(value, bins=5):
    bin_size = (max(value) - min(value))
/ bins
    binned_values = [
        f"{min(value) + (int((val -
min(value)) / bin_size) * bin_size)}-
{min(value) + ((int((val - min(value)) /
bin_size) + 1) * bin_size)}"
        for val in value
    ]
    return binned_values
```

The html_template variable contains an HTML structure for the web interface. It provides:

- A form for uploading a CSV file.
- A table displaying file attributes (column names and data types) after the upload.
- A list of checkboxes allowing the user to select columns for anonymization.

The template uses placeholders ({% ... %}) for dynamic rendering via Flask's render_template_string() function.

```
# HTML template for the web page
html_template = """
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport"
content="width=device-width, initial-
scale=1.0">
    <title>CSV Attribute Viewer</title>
    <style>
        body {
            font-family: Arial, sans-
serif;
            margin: 20px;
        }
        h1 {
            color: #333;
        }
        table {
            width: 100%;
            border-collapse: collapse;
            margin-top: 20px;
        }
        th, td {
            border: 1px solid #ddd;
            padding: 8px;
            text-align: left;
        }
```

```
        th {
            background-color: #f4f4f4;
        }
        .checkbox-container {
            display: flex;
            flex-direction: column;
        }
    </style>
</head>
<body>
    <h1>Upload a CSV File</h1>
    <form action="/" method="POST"
enctype="multipart/form-data">
        <input type="file" name="file"
accept=".csv" required>
        <button
type="submit">Upload</button>
    </form>
    {% if attributes %}
    <h2>Select Features to
Anonymize</h2>
    <form action="/anonymize"
method="POST">
        <div class="checkbox-container">
            {% for attr, dtype in
attributes %}
                <label>
                    <input
type="checkbox" name="features"
value="{{ attr }}"> {{ attr }} ({{ dtype
}})
                </label>
            {% endfor %}
        </div>
        <button
type="submit">Anonymize</button>
    </form>
    {% endif %}
</body>
</html>
"""
```

The upload_file route handles both GET and POST requests.

- For a GET request, the route simply renders the initial web page, displaying a form to upload a CSV file.
- For a POST request, the program retrieves the uploaded file using request.files.get("file") and processes it. The file is read into a pandas DataFrame using pd.read_csv(file), and its columns and data types are extracted into a list of tuples (attributes). These attributes are dynamically displayed in the web interface.
- If an error occurs while reading the file (e.g., an invalid CSV format), the program logs the error and displays a message indicating that the file could not be processed.

```
@app.route("/", methods=["GET", "POST"])
def upload_file():
    attributes = None
    if request.method == "POST":
        file = request.files.get("file")
        if file:
            try:
                # Read the CSV file into a
DataFrame
                df = pd.read_csv(file)
                # Extract columns and data
types
                attributes = [(col,
str(dtype)) for col, dtype in
df.dtypes.items()]
            except Exception as e:
                app.logger.error(f"Error
reading file: {e}")
                attributes = [("Error",
f"Unable to process file: {e}")]
    return
render_template_string(html_template,
attributes=attributes)
```

The /anonymize route is responsible for anonymizing the uploaded file.

- First, the selected features (columns) for anonymization are retrieved using request.form.getlist("features").
- The uploaded file is read into a pandas DataFrame. If the file loads successfully, the program iterates through the selected columns. Depending on the column's data type:
    - Textual columns (object or string type) are pseudonymized using the pseudonymize function.
    - Numeric columns (int64 or float64 type) are binned into ranges using the binning function.
- Once the anonymization is complete, the modified DataFrame is saved as a new CSV file (output.csv) using df.to_csv(output_file, index=False).
- A success message, *"Anonymized file saved as output.csv. You can download it below."*, is displayed on the web page. The page also provides a link to download the file by directing users to the /download route.
- The /download route allows users to download the anonymized file. The route uses Flask's send_file() function to serve the saved output.csv file as an attachment, enabling the user to save it locally. This ensures a seamless download experience directly from the web interface.

```
@app.route("/anonymize", methods=["POST"])
def anonymize_file():
    selected_features =
request.form.getlist("features")
    file = request.files.get("file")
```

```
    if file:
        try:
            # Read the CSV file into a
DataFrame
            df = pd.read_csv(file)
            app.logger.info(f"File loaded
successfully. Features selected:
{selected_features}")

            # Apply anonymization based on
feature type
            for feature in
selected_features:
                if feature in df.columns:
                    if df[feature].dtype
== 'object' or df[feature].dtype ==
'string':  # Text data
                        df[feature] =
df[feature].apply(pseudonymize)
                    elif df[feature].dtype
in ['int64', 'float64']:  # Numeric data
                        df[feature] =
binning(df[feature])

            # Save the anonymized
DataFrame to a new CSV file
            output_file = "output.csv"
            df.to_csv(output_file,
index=False)

app.logger.info(f"Anonymization completed.
Output saved to {output_file}")

            # Render a success message in
the HTML
            message = "Anonymized file
saved as output.csv. You can download it
below."
            download_link = f'<a
href="/download" target="_blank">Download
Anonymized File</a>'
            return
f"<p>{message}</p><p>{download_link}</p>"

        except Exception as e:
            app.logger.error(f"Error
during anonymization: {e}")

app.logger.error(traceback.format_exc())
            return "Error: An error
occurred while anonymizing the file."
    return "No file uploaded."
```

The run_app function initializes the Flask app on port 5000 in debug mode. To ensure the web server runs independently, the program uses Python's multiprocessing module. This allows the Flask app to run in a separate process without blocking other operations. By using the Process class, the Flask app starts in parallel with the main Python program, making it suitable for environments that might require additional processing tasks.

```
# Function to run Flask app in a separate
process
def run_app():
    app.run(port=5000, debug=True)

# Start the Flask app in a separate
process
if __name__ == "__main__":
    process = Process(target=run_app)
    process.start()
```

The proposed approach has been implemented as a web-based application to facilitate practical evaluation and reproducibility. The live application is publicly accessible at:

https://csv-anonymizer.onrender.com

The complete source code, including implementation details and documentation, is available in a public GitHub repository at:

https://github.com/sborkakoty-ustm/csv-anonymizer

3. RESULTS AND DISCUSSION:

3.1. Results

Upon executing the program, users are prompted to upload a CSV file containing data.

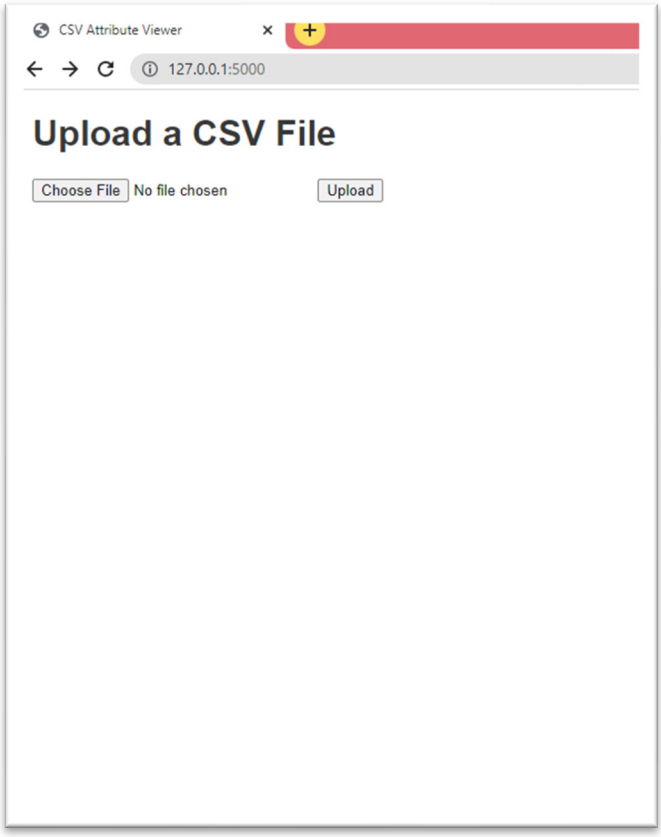

Figure 1: Interface to upload CSV file

After uploading the file, the application reads and processes the file, displaying the attributes (column names and data types) of the dataset. Users can then select which features

(columns) they wish to anonymize.

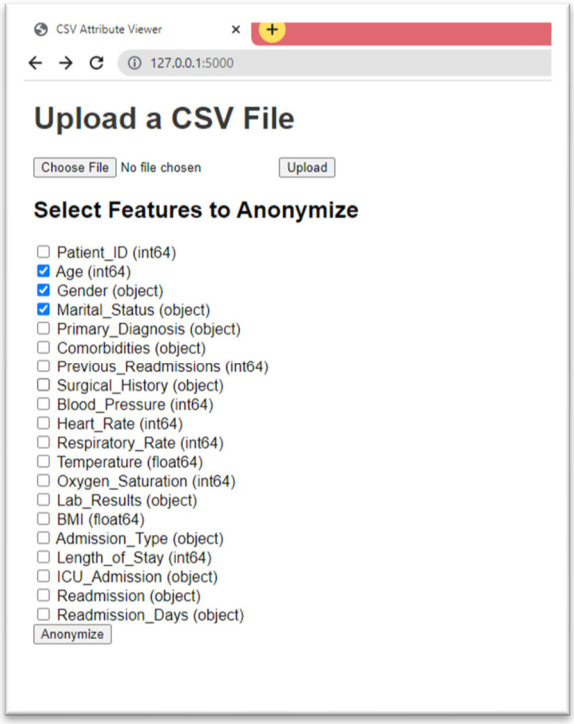

Figure 2: Interface to select features to anonymize

After anonymizing the selected features, the modified dataset is saved as output.csv, and the user is provided with a download link. A confirmation message is displayed, stating, "Anonymized file saved as output.csv. You can download it below." This seamless workflow enables users to efficiently anonymize their datasets while providing the necessary feedback about the operation's success.

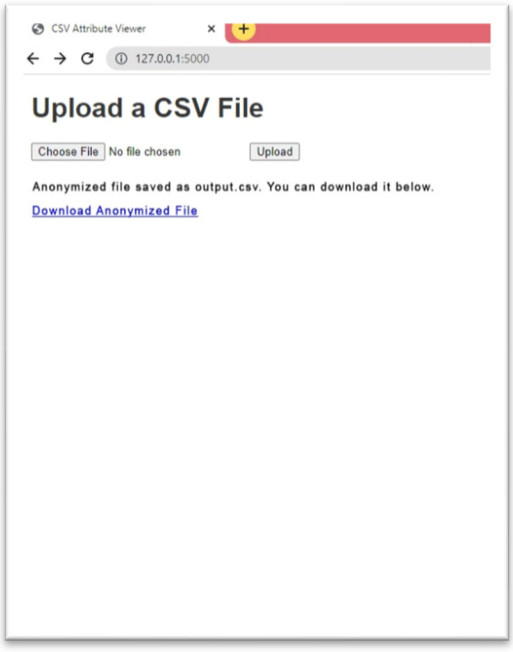

Figure 3: Interface to download nonymized file

3.2. Discussion

The implementation of pseudonymization and binning as anonymization techniques effectively reduces the risk of re-identifying individuals within a dataset while maintaining the analytical usefulness of the data. These techniques are particularly relevant for scenarios where data privacy and confidentiality are paramount, such as in healthcare.

1. **Pseudonymization**: Pseudonymization is a widely recognized technique for safeguarding personal data. By replacing sensitive textual data (such as names, addresses, or identifiers) with randomized strings, the application prevents unauthorized access to personal information. However, pseudonymization does not fully anonymize the data, as the generated pseudonyms can still be traced back to the original data if additional information is available. Therefore, it is critical to ensure that pseudonymization is combined with other privacy measures, such as secure storage or data encryption, to achieve a higher level of security.
2. **Binning**: The binning technique effectively reduces the precision of numeric data while preserving the overall distribution and patterns. It mitigates the risks associated with sharing fine-grained numerical values that could potentially lead to the identification of individuals. However, binning also introduces a trade-off between privacy and data utility. For example, while a binned dataset may obscure exact values, it may also make certain types of analysis, such as precise trend analysis or regression modeling, more difficult. The choice of the number of bins and their sizes is crucial to ensuring that the binning process effectively balances privacy and data usefulness.
3. **User Experience**: The user interface is designed to be simple and user-friendly. Users can easily upload CSV files, select features for anonymization, and download the processed file with minimal effort. This design ensures that non-technical users can perform data anonymization tasks without needing extensive programming knowledge. Furthermore, including feedback messages, such as confirmation that the anonymized file has been saved, enhances the overall user experience by providing clarity on the operation's status.

3.2.1 Areas of improvement

While the system has achieved positive results during the testing phase, there are certain areas of future scope and potential improvements that can further enhance its effectiveness:

1. **Integration of Advanced Anonymization Techniques**: To enhance the system's privacy guarantees, future work could explore integrating more sophisticated anonymization methods. Techniques such as **differential privacy** (Friedman & Schuster, 2010; Geng & Viswanath, 2016), which add noise to data to make it harder to trace back to individuals, could be implemented to further protect privacy. Additionally, incorporating **generalization** (i.e., transforming data into broader categories) could help to mitigate the risks of re-identification while maintaining data utility. (Ahsan *et al.*, 2021; Evans, 2005).
2. **Scalability Improvements**: To better handle large datasets, future improvements could focus on optimizing the application's performance. Techniques such as **chunk-based processing** (Sharma, 2022), where large files are processed in smaller, manageable segments, can be incorporated to reduce memory consumption and enhance efficiency. Additionally, parallel or distributed computing techniques could be explored to speed up the processing of large datasets, making the application more scalable and suitable for enterprise-level use cases (Sedgwick, 2022).
3. **Automated Feature Selection**: The current system requires users to manually select features for anonymization. To improve the user experience and reduce errors, automated feature-selection algorithms could be integrated to recommend which features should be anonymized based on predefined privacy rules or data sensitivity levels. Machine learning algorithms could be used to classify and prioritize sensitive attributes, streamlining the process for non-technical users.
4. **Extended Data Formats**: The current implementation supports only CSV files. Future iterations of the system could expand its functionality to support additional file formats, such as Excel and JSON, and to enable database connections, making the application more versatile and accessible to a wider range of users and industries.
5. **Enhanced User Interface and Experience**: Future versions of the application could improve the user interface (UI) by providing more granular control over the anonymization process, such as allowing users to define custom binning ranges or pseudonymization patterns. Additionally, incorporating a more intuitive UI with visual feedback (e.g., progress bars and alerts for successful anonymization) could enhance the overall user experience, especially for users with limited technical expertise.

4. CONCLUSIONS:

This paper presents an anonymization tool designed to protect the privacy of individuals in medical datasets, specifically focusing on pseudonymization and binning techniques for text and numeric data. By integrating Flask for web-based file uploads and processing, the tool provides an intuitive interface that lets users upload CSV files, select anonymization features, and download the modified dataset. By applying pseudonymization to text-based data and binning to numeric values, the system reduces the risk of re-identification while preserving the dataset's general structure and utility.

While the tool demonstrates the feasibility and effectiveness of basic anonymization techniques, it also highlights several limitations, including potential loss of data precision, reversibility of pseudonymization, and scalability challenges with large datasets. Additionally, the current implementation does not address advanced privacy threats, such as inference attacks, and lacks support for data formats beyond CSV.

Despite these limitations, the tool represents a valuable step toward ensuring data privacy in the era of big data and analytics. Future work in this area could expand the system’s capabilities by incorporating advanced anonymization techniques, improving scalability, automating feature selection, and ensuring compliance with international privacy regulations. By building on the current framework, this tool can evolve into a more robust, secure solution that addresses the growing concerns about data privacy in an increasingly connected world.

5. DECLARATIONS

5.1. Study Limitations

a) **Loss of Data Precision**: One key limitation of the current anonymization method, particularly the binning technique, is the potential loss of precision in numeric data. By grouping numerical values into bins, exact values are obscured, which may reduce the ability to perform certain analyses that require precise numerical data. For example, regression models or detailed statistical analyses may be less accurate because finer data details are lost during binning.

b) **Reversibility of Pseudonymization**: While pseudonymization is an effective method for anonymizing text-based data, it remains vulnerable to reversibility if the pseudonymization key or algorithm is exposed. When the mapping between original and pseudonymized data is known or can be inferred, the pseudonymized data can be linked back to individuals. Therefore, this technique alone may not be sufficient to fully safeguard privacy in high-risk environments.

c) **Limited Anonymization Techniques**: The current system employs only pseudonymization for textual data and binning for numerical data. These two techniques may not be robust enough for certain complex privacy concerns. The application does not address advanced privacy threats, such as inference or linkage attacks, where anonymized data may still be linked to individuals using external information.

d) **Scalability and Performance**: While the application performs effectively on small to medium-sized datasets, it may experience performance issues when handling large volumes of data. The current implementation loads the entire dataset into memory before applying anonymization, which could lead to high memory consumption and slow processing times for large datasets. As the dataset size increases, this could affect the application's responsiveness and efficiency.

5.2. Acknowledgements

I want to express my sincere gratitude to everyone who has offered their support during the research journey. While not directly involved in this project, the indirect contributions and unwavering support from various individuals have greatly contributed to its success. I also want to acknowledge the reviewers and editors for their valuable feedback and suggestions, which have significantly enhanced the quality of this research paper.

5.3. Funding source

The authors funded this research. In accordance with the ethical guidelines of the Southern Journal of Sciences, which do not allow donations from authors with manuscripts under evaluation (even when research funds are available), or in cases of authors' financial constraints, publication costs were fully absorbed by the journal under our Platinum Open Access policy, through the support of the Araucária Scientific Association (https://acaria.org/). This policy aims to ensure complete independence between the editorial process and any financial aspects, reinforcing our commitment to scientific integrity and equity in knowledge dissemination.

5.4. Competing Interests

The authors declare no potential conflict of interest in this publication.

5.5. Open Access

5.6. AI usage declaration

The authors declare that no Artificial Intelligence (AI) or AI-assisted technologies were used in the conception, data analysis, or preparation of this manuscript. All content presented is the original

work of the authors.

5.7. Author's contribution

All authors contributed to the design and implementation of the research, to the analysis of the results, and to the writing of the manuscript.

6. HUMAN AND ANIMAL-RELATED STUDIES

6.1. Ethical Approval

Not applicable.

6.2. Informed Consent

Not applicable.

7. REFERENCES:

1. Majeed, A., & Lee, S. (2021). Anonymization techniques for privacy preserving data publishing: A comprehensive survey. *IEEE Access*, *9*, 8512–8545. https://doi.org/10.1109/access.2020.3045700
2. Marques, J., & Bernardino, J. (2020). Analysis of data anonymization techniques. In *Proceedings of the 12th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management* (pp. 235–241). SCITEPRESS – Science and Technology Publications. https://doi.org/10.5220/0010142302350241
3. Murthy, S., Abu Bakar, A., Abdul Rahim, F., & Ramli, R. (2019). A comparative study of data anonymization techniques. In *2019 IEEE 5th International Conference on Big Data Security on Cloud (BigDataSecurity)* (pp. 306–309). IEEE. https://doi.org/10.1109/bigdatasecurity-hpsc-ids.2019.00063
4. Kargupta, H., Datta, S., Wang, Q., & Sivakumar, K. (2003). On the privacy preserving properties of random data perturbation techniques. In *Proceedings of the Third IEEE International Conference on Data Mining* (pp. 99–106). IEEE Computer Society. https://doi.org/10.1109/icdm.2003.1250908
5. Muralidhar, K., & Sarathy, R. (1999). Security of random data perturbation methods. *ACM Transactions on Database Systems*, *24*(4), 487–493. https://doi.org/10.1145/331983.331986
6. Dhawas, P., Dhore, A., Bhagat, D., Pawar, R. D., Kukade, A., & Kalbande, K. (2024). Big data preprocessing, techniques, integration, transformation, normalisation, cleaning, discretization, and binning. In *Advances in Business Information Systems and Analytics* (pp. 159–182). IGI Global. https://doi.org/10.4018/979-8-3693-0413-6.ch006
7. Ahsan, M., Mahmud, M., Saha, P., Gupta, K., & Siddique, Z. (2021). Effect of data scaling methods on machine learning algorithms and model performance. *Technologies*, *9*(3), 52. https://doi.org/10.3390/technologies9030052
8. Evans, P. (2005). Scaling and assessment of data quality. *Acta Crystallographica Section D Biological Crystallography*, *62*(1), 72–82. https://doi.org/10.1107/s0907444905036693
9. Sharma, V. (2022). A study on data scaling methods for machine learning. *International Journal for Global Academic & Scientific Research*, *1*(1). https://doi.org/10.55938/ijgasr.v1i1.4
10. Sedgwick, P. (2012). Log transformation of data. *BMJ*, *345*, Article e6727. https://doi.org/10.1136/bmj.e6727
11. Dwork, C. (2006). Differential privacy. In *Lecture Notes in Computer Science* (pp. 1–12). Springer Berlin Heidelberg. https://doi.org/10.1007/11787006_1
12. Friedman, A., & Schuster, A. (2010). Data mining with differential privacy. In *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (pp. 493–502). Association for Computing Machinery. https://doi.org/10.1145/1835804.1835868
13. Geng, Q., & Viswanath, P. (2016). The optimal noise-adding mechanism in differential privacy. *IEEE Transactions on Information Theory*, *62*(2), 925–951. https://doi.org/10.1109/tit.2015.2504967
14. Olatunji, I. E., Rauch, J., Katzensteiner, M., & Khosla, M. (2022). A review of anonymization for healthcare data. *Big Data*. Mary Ann Liebert Inc. https://doi.org/10.1089/big.2021.0169