Comparative Analysis of ChatGPT and DeepSeek for Privacy-Preserving Code Generation using Differential Privacy

Felipe Castano Gonzalez
Department of Electrical and Software Engineering
University of Calgary
Calgary, Canada
felipe.castanogonzal@ucalgary.ca

Reyhaneh Farahmand
Department of Electrical and Software Engineering
University of Calgary
Calgary, Canada
reyhaneh.farahmand@ucalgary.ca

Abstract— **As artificial intelligence (AI) systems increasingly rely on sensitive data, implementing privacy-preserving mechanisms has become critical. Differential Privacy (DP) provides strong mathematical guarantees against individual re-identification, making it a key tool in secure data analysis. This study explores the ability of two large language models (LLMs), ChatGPT and DeepSeek, to generate executable Python code that applies DP to structured tabular data.**

Using a synthetic dataset simulating shelter records, both models received identical structured prompts requesting DP-protected counts by birth decade. Their outputs were evaluated in terms of correctness, structure, and statistical fidelity. Quantitative metrics, including Mean Squared Error, Mean Absolute Error, and Spearman correlation coefficient, confirmed that both implementations preserved data utility under a strict privacy budget ($\varepsilon = 0.1$).

While both LLMs generated functional code, DeepSeek provided more consistent and readable results. These findings demonstrate the feasibility of using LLMs for automated DP code generation in privacy-sensitive applications.

Keywords— Differential Privacy, Large Language Models, Data Security, Code Generation.

I. INTRODUCTION

The increasing reliance on artificial intelligence (AI) and machine learning (ML) applications has raised significant concerns regarding software security and data privacy. As AI-driven solutions become integral to sensitive domains such as healthcare, finance, and governance, ensuring robust privacy-preserving mechanisms is imperative. Privacy-Preserving AI aims to protect user data while maintaining the functionality and benefits of AI models. Several techniques contribute to this field, including Federated Learning, Homomorphic Encryption, Secure Multi-Party Computation, and Differential Privacy (DP). Among these, DP stands out as a mathematically rigorous approach to reducing the risk of individual data exposure while allowing meaningful statistical analysis. Its adoption in software security mechanisms highlights and potential to mitigate adversarial attacks and ensure data integrity without compromising privacy [1].

DP systematically modifies datasets or algorithms by incorporating controlled noise, thereby preventing adversaries from discerning whether a particular individual's data was included in the dataset. This approach is particularly advantageous in domains where data sensitivity is paramount. Unlike conventional anonymization techniques, which can be vulnerable to re-identification attacks, DP provides formal guarantees against privacy breaches by ensuring that query responses remain statistically indistinguishable regardless of an individual's data presence. The application of DP spans various computational paradigms, including ML, where differentially private mechanisms, such as differentially private stochastic gradient descent, have been integrated to enhance privacy-preserving AI models [2].

One significant application of DP is in protecting structured tabular data, a crucial requirement in data-driven industries. Traditional methods of safeguarding tabular data, such as data anonymization or encryption, often fail to balance privacy and utility. Feature hashing, a widely used technique in natural language processing (NLP), has been explored as a means of incorporating DP in tabular data processing [3]. This method enables the transformation of sensitive text features into a differentially private representation, thus preventing unauthorized access to personally identifiable information while maintaining data utility. Unlike standard DP implementations that rely on noise injection, feature hashing enables a noiseless yet effective privacy-preserving transformation, thereby offering a viable alternative for sensitive text and tabular data protection.

The emergence of Large Language Models (LLMs) has revolutionized AI applications, but their extensive data processing capabilities pose significant privacy challenges. Generative AI tools, such as OpenAI's ChatGPT, leverage billions of parameters to analyze vast datasets, often containing sensitive information. To address privacy concerns, researchers have developed various privacy-preserving techniques for LLMs, including DP, secure multi-party computation, and federated learning [4]. These methods aim to enhance privacy during both data curation and model training, thereby mitigating risks associated with data leakage and adversarial attacks. The use of DP within LLMs ensures that sensitive contextual information remains protected while preserving the model's ability to generate coherent and informative responses.

Recent advancements in open-source LLMs, such as DeepSeek, have further underscored the potential of privacy-preserving AI. DeepSeek's latest model, V3, has demonstrated performance comparable to leading closed-source models like OpenAI's GPT-4o, despite operating under constrained computational resources. By leveraging innovative architectures such as Mixture of Experts, DeepSeek optimizes computational efficiency while maintaining high accuracy. The model's open-source nature facilitates transparency, enabling researchers to explore privacy-preserving enhancements, including DP-based methodologies [5].

While LLMs have been extensively utilized in various AI-driven applications, their role in privacy-preserving mechanisms remains an emerging research area. Studies have explored the capabilities of LLMs in code generation, test automation, and validation of AI models [6]. However, the integration of DP within LLM validation processes has yet to be systematically examined. The necessity of rigorous testing for privacy-preserving AI tools is crucial to ensuring their reliability, particularly in security-sensitive applications.

Despite the advantages of DP in LLMs, certain limitations must be acknowledged. The effectiveness of DP in NLP applications depends on strict adherence to mathematical privacy guarantees, which can be challenging to maintain [7]. For instance, studies have demonstrated that improperly configured DP mechanisms can result in privacy leakage, particularly in complex model architectures such as autoencoders, raising concerns about their reliability in sensitive domains. Such findings underscore the importance of formal verification and rigorous evaluation of DP implementations within AI models. Furthermore, the trade-off between privacy and model utility remains a critical challenge, as excessive noise injection can degrade model performance. Addressing these limitations requires ongoing research into optimizing DP parameters and refining privacy-preserving algorithms tailored for LLMs applications.

The aim of this project is to compare the capabilities of ChatGPT and DeepSeek in generating code that incorporates DP for privacy-preserving applications. Specifically, this study seeks to determine which LLM produces more accurate, effective, and practically applicable code for privacy-preserving tasks. To enhance the quality of the generated outputs, prompt engineering techniques will be employed to ensure clarity, relevance, and precision. A synthetic dataset will be used throughout the semester to test and evaluate the generated code. Finally, the viability of these LLM-generated solutions will be assessed in the context of real-world databases, examining whether they can be directly applied or require further modifications for practical implementation.

II. Related Work

A. *Differential Privacy*

DP has become an essential strategy for protecting sensitive data across a wide range of contexts. Its core premise lies in adding carefully calibrated noise to reduce the likelihood of re-identifying individuals, without unduly compromising the utility of the information. By effectively "dissociating" personal identities from published data, DP has overcome many of the pitfalls associated with traditional anonymization methods, which are often vulnerable to correlation or linkage attacks.

According to Janghyun et al. [8], DP was initially developed to address the limitations of conventional anonymization. The authors highlight that, over time, the literature has introduced nearly 200 variants and extensions of DP, revealing the need for organized taxonomies that can guide researchers in choosing the most suitable configurations. By providing different mathematical formulations, researchers can determine how much noise to add, and at which stages in the data pipeline (e.g., before, during, or after computation). This flexibility allows for precisely balancing privacy protection with data utility.

Moving beyond theoretical foundations, cyber-physical systems and big data applications illustrate the practical relevance of DP. Hassan et al. [9] performed a thorough survey on how DP mitigates privacy risks when managing large, real-time datasets in settings such as smart grids and industrial monitoring. In these scenarios, continuous data collection from numerous sensors complicates data protection. Nonetheless, DP proves robust against inferences drawn from aggregated statistics, making it particularly advantageous for publishing measurements, histograms, or synthetic data. This advantage is further underscored when analyzing extensive data streams that need ongoing monitoring or diagnostic analysis.

On the other hand, Zhu et al. [10] examined how Laplacian or Gaussian noise, coupled with hierarchical structures such as trees or quadtrees, can be used to progressively scale perturbations. This approach is highly relevant when datasets are high-dimensional or frequently updated. By subdividing the data space and injecting noise selectively, it becomes more difficult for adversaries to reconstruct exact user information, while maintaining a reasonable degree of accuracy for analytical purposes.

In parallel, ML presents another domain where DP has shown considerable impact. Liu et al. [11] explain how privacy mechanisms can be integrated at various stages of the ML process, from data preprocessing to parameter updates and loss functions. A prime development in this area is the DP Stochastic Gradient Descent (SGD) approach proposed by Abadi and colleagues, which continuously tracks privacy loss via the Moment Accountant. This facilitates a more adaptive handling of the privacy budget (ε), ensuring that organizations can tailor privacy levels to their specific sensitivity or accuracy demands. Liu et al. [11] also underscore the value of dynamic noise assignment throughout model training, highlighting that such strategies can avert severe computational overhead or drastic degradation in model performance.

When these ideas are applied to more advanced models such as evolutionary algorithms or fuzzy systems additional challenges arise, as noted by Gong et al. [12]. The high number of parameters and the complexity of evolutionary or reasoning operators can demand stronger safeguards to prevent information leakage. Nevertheless, the authors highlight an intriguing paradox: in certain cases, a moderately high noise level can enhance diversity in evolutionary populations. This suggests that, beyond privacy, noise might function as a factor that boosts exploration in these optimization methods, provided the noise levels are carefully managed.

One of the fields that has recently spurred significant research is that of LLMs. Coffey et al. [13] propose various strategies to integrate DP during both fine-tuning and inference phases. They outline a two-step training approach: first, the model is trained using partial or pre-filtered data to reduce exposure risks, and second, a private mechanism adds noise to the gradients. In doing so, they seek to safeguard any sensitive information that could otherwise be revealed via the model's responses. Additionally, these authors discuss the benefits of more refined privacy accounting methods, such as the Edgeworth accountant, which provides a granular view of how privacy loss accumulates throughout the process.

Finally, the healthcare domain has shown substantial gains from the adoption of DP. Ficek et al. [14], in their analysis of 54 studies, note that publishing aggregated data such as counts, or contingency tables embedded with Laplacian or Gaussian noise is a common technique to share information without exposing individual clinical records. This practice extends to wearable device data (e.g., heart rate monitors) and genomic sequences, both of which are complex and highly sensitive. However, the authors also warn that applying uniform perturbations to heterogeneous populations may lead to injustice or bias, since minority groups often experience greater data utility degradation.

While these works provide valuable insights into the application of DP in various domains, including healthcare and real-time data publishing, they primarily focus on structured data protection or model training scenarios. In contrast, our project explores a relatively underexplored area, leveraging LLMs such as ChatGPT and DeepSeek for automated code generation that incorporates DP mechanisms. Unlike prior studies, our work investigates not only the correctness and privacy guarantees of the generated code but also its practical applicability to real-world databases, filling a gap in the literature where LLM-driven privacy-preserving code generation has received limited attention.

B. *Large language Models*

The advent of LLMs has significantly transformed various domains, from healthcare and finance to scientific research and agriculture. However, the growing reliance on these models brings concerns about privacy and security, necessitating the exploration of privacy-preserving mechanisms such as DP. This section reviews prior research on LLMs, emphasizing their applications, limitations, and the importance of privacy-preserving techniques in code generation.

To begin with, Peng et al. [15] investigated the role of LLMs in healthcare, uncovering critical privacy gaps when deploying AI in medical applications. Their research underscores that while models such as ChatGPT and DeepSeek enhance clinical decision support, they also pose significant risks by potentially exposing sensitive patient data. This underscores the necessity of integrating privacy-preserving techniques, a challenge our study addresses in the context of software development.

Shifting to the financial sector, Du et al. [16] introduced FI-NL2PY2SQL, an LLM-based model designed to translate natural language financial queries into SQL. Their findings demonstrate the efficiency of LLMs in automating data-driven tasks but also highlight concerns regarding the handling of confidential financial information. Given these risks, our study extends this work by exploring how DP can be incorporated into LLM-based code generation to ensure both confidentiality and usability.

Similarly, in the field of agriculture, Fei et al. [17] examined the role of LLMs in extracting disease related knowledge from research papers. While their study showcases the effectiveness of LLMs in specialized knowledge retrieval, it does not address the privacy concerns associated with proprietary data usage. This gap reinforces the need for privacy-preserving approaches like DP, which our research applies in secure code generation.

Moving toward a broader comparison, Kayaalp et al. [18] conducted an extensive evaluation of ChatGPT and DeepSeek, focusing on their multimodal AI capabilities in scientific content generation. Their study indicates that DeepSeek exhibits superior content generation abilities in specific applications. However, they do not examine privacy-related aspects, which remain the focal point of our research. By expanding their work, we aim to analyze how these models perform when integrating DP for secure and reliable code generation.

Further examining LLM development, Normile [19] discussed China's rapid progress in AI, particularly with models like DeepSeek. While this study highlights advancements in AI innovation, it also raises concerns about data security and regulatory compliance. Our research extends this discussion by evaluating how privacy-preserving mechanisms can be embedded in such models to enhance their security in real-world applications.

Another important perspective is provided by Rhomrasi et al. [20], who assessed LLM performance in mathematical reasoning tasks. Their findings emphasize that LLMs generalize differently across linguistic contexts, revealing performance variability based on task adaptation. This insight aligns with our research as we investigate the comparative performance of ChatGPT and DeepSeek in privacy-preserving code generation, particularly regarding how DP integration impacts model accuracy.

Beyond domain specific applications, several studies have explored privacy-preserving mechanisms in LLMs. Shi et al. [21] introduced Selective DP, demonstrating that carefully fine tuning LLMs with DP can achieve a balance between privacy and model utility. Expanding on this, Shi et al. [22] further refined DP applications, showcasing how selective implementation can mitigate utility loss while ensuring data security. Their contributions from the theoretical backbone of our research, as we assess the effectiveness of DP-enhanced ChatGPT and DeepSeek in secure code generation.

By synthesizing these prior works, our study aims to provide a comprehensive comparative analysis of ChatGPT and DeepSeek. We focus on their ability to generate code while preserving privacy through DP and prompt engineering. This research will contribute to the ongoing efforts to integrate robust privacy-preserving mechanisms into AI-driven software development.

III. Dataset

The dataset used in this research is a synthetic dataset based on the record structure of the Calgary Drop-In Centre, one of the leading organizations supporting individuals experiencing homelessness in Calgary. It contains 15,991 records, designed to reflect the structure of real data used in the management of shelters and assistance services, without compromising individual privacy.

This dataset is relevant as it represents the scale of homelessness in Calgary. According to the latest Calgary Drop-In Centre report (2023-2024), 8,731 individuals accessed the DI shelter services over the past year, marking a significant increase compared to previous years (6,839 in 2022-2023). Additionally, the average number of people staying at the shelter each night was 634, highlighting the ongoing strain on available resources [23].

In contexts where highly sensitive information is handled, such as shelter records, it is essential to implement effective DP mechanisms to prevent the reidentification of individuals and ensure compliance with ethical and legal principles in data management.

IV. Methodology

This section outlines the methodology employed to compare the capabilities of ChatGPT and DeepSeek in generating DP protected code for privacy-preserving applications. The methodology is designed to be fully reproducible, ensuring that other researchers can replicate the study under similar conditions. It consists of four key components: dataset preprocessing, application of DP, prompt engineering, and evaluation metrics.

A. *Dataset preparation*

The dataset used in this research is a synthetic dataset based on the record structure of the Calgary Drop-In Centre. To ensure the dataset is suitable for applying privacy-preserving techniques, the dataset will undergo cleaning, filtering, and organization. The cleaning process involves removing missing, inconsistent, or redundant records to improve data integrity. Filtering will focus on selecting only the relevant attributes necessary for privacy-preserving tasks, ensuring that unnecessary or non-essential data points do not contribute to privacy risks.

The dataset will then be organized into a structured format compatible with the DP implementation, ensuring uniformity in categorical and numerical attributes. Standardization of numerical values will be performed to prevent outlier effects in differentially private mechanisms. Additionally, feature hashing or tokenization will be applied to anonymize sensitive text attributes, preserving privacy while maintaining dataset utility. After these steps, the cleaned and structured dataset will serve as the input for DP applications.

B. *Privacy preserving implementation*

DP is the primary privacy-preserving mechanism in this study, implemented through a systematic approach that ensures robust privacy guarantees while maintaining data utility. A predefined privacy budget (ε), typically ranging from 0.1 to 1.0, is selected and iteratively tuned to balance privacy protection with data usability. Noise injection is applied using the Laplace mechanism for numerical attributes, adding noise sampled from a Laplace distribution centered at zero with a scale proportional to query sensitivity. For high-dimensional datasets, the Gaussian mechanism is employed, ensuring compliance with DP principles under established assumptions.

For ML applications, *Stochastic Gradient Descent (SGD)* is implemented, injecting calibrated noise into model training to preserve privacy while allowing convergence. Privacy loss tracking is performed using the Moment Accountant technique to ensure compliance with predefined privacy constraints. To prevent data reconstruction attacks, histogram-based DP aggregation techniques are applied, allowing statistical analyses while upholding strong privacy guarantees. Formal privacy checks validate DP implementation through compliance tests, statistical evaluations of noise effectiveness, and robustness assessments against adversarial attacks.

C. *Prompt engineering*

To maximize the effectiveness of LLM-generated code for DP, an iterative prompt engineering strategy will be employed. Initially, well-structured prompts will be designed with explicit instructions regarding DP implementation. If initial outputs are suboptimal, the prompts will be refined iteratively, testing variants that adjust specificity, format, and complexity to optimize model responses [24]. Few-shot prompting will be used to provide contextual examples that improve model understanding, while chain-of-thought prompting will enhance logical consistency in DP-related responses.

Prompts will be systematically modified and evaluated based on the relevance, completeness, and accuracy of the generated code. If intermediate results indicate insufficient quality, alternative prompt formats and additional context structuring will be explored to enhance reliability. Since LLM-generated responses can vary, prompt modifications will be continuously refined throughout the project, following best practices in applying DP techniques within NLP models to ensure privacy and robustness in generated outputs [25]. The final outputs of the models will consist of DP code implementations, intended to be executed by the user.

In this context, two LLMs will be employed and compared: ChatGPT (GPT-4o) and DeepSeek-V3. GPT-4o, OpenAI's flagship multimodal model, is designed for real-time reasoning across text, vision, and audio, which makes it particularly effective for generating and refining code through structured, few-shot, and chain-of-thought prompting strategies. Its advanced capabilities in logical reasoning and contextual understanding are expected to produce high-quality DP-related code. In contrast, DeepSeek-V3 is a cutting-edge text-based model specialized in extended textual comprehension and generation, capable of processing up to 128K tokens. It will be leveraged to analyze lengthy technical documents (e.g., PDFs, spreadsheets, and manuals) and extract relevant content to support prompt construction. Although it lacks multimodal capabilities, DeepSeek-V3 offers high precision and coherence in handling large-scale textual inputs. By comparing the outputs of these two models under the same iterative prompt engineering framework, the study aims to evaluate their relative effectiveness in generating accurate, relevant, and user-implementable code for DP applications.

D. Evaluation metrics

To assess the effectiveness of the DP code generated by ChatGPT and DeepSeek, this study implements a set of evaluation metrics focused on measuring the similarity between the original (true) data distribution and the differentially private output produced through the Laplace mechanism. The goal is to quantify the trade-off between data utility and privacy, ensuring that the added noise does not significantly distort meaningful statistical patterns.

The evaluation process begins with a visual comparison. Histograms of true and differentially private counts, grouped by birth decade, are plotted side by side. This graphical approach facilitates the identification of overall distributional trends, abrupt deviations, and the extent to which privacy-preserving mechanisms alter the underlying data structure.

Beyond visual inspection, three quantitative metrics are computed to compare true and noisy counts. Mean Squared Error (MSE) measures the average squared difference between true and private values, placing greater weight on larger errors and capturing the extent of distortion introduced by the noise. Mean Absolute Error (MAE) provides a complementary perspective by reporting the average magnitude of errors, regardless of direction, offering a more interpretable view of typical differences. Finally, the Spearman Correlation Coefficient assesses the relationship between the two distributions, indicating whether the overall structure and trend of the data are preserved after applying DP.

V. Results

A. Dataset preparation

The dataset used in this project is a synthetic collection designed to simulate personal records for 15,991 individuals and comprises 28 columns. It was generated using a structured approach that balances realism with privacy preservation. The source for the synthetic generation was an original database from the Calgary Drop-In Center. To protect the identity of real individuals, no original data was used directly. Instead, new entries were created by duplicating individuals and introducing realistic, statistically grounded variations in their personal attributes.

The generation process applied subtle modifications to key fields such as first names, last names, and dates of birth. These changes mirror natural variations found in real-world data, such as spelling differences, typographical errors, and date offsets. For example, a first name like “Jonathon” may be rendered as “Jonathan”, and a surname like “Peterson” may appear as “Patterson”, reflecting common transcription or pronunciation-based differences. Similarly, a date of birth originally recorded as “August 15, 1987” could be adjusted to “August 14, 1987” or “August 16, 1987”, simulating real data entry inconsistencies. These controlled variations ensure the dataset remains representative of real-world data while eliminating any direct identifiers.

This synthetic dataset is particularly useful for testing data processing pipelines, evaluating data anonymization techniques, and simulating real-world scenarios in a controlled environment. Its structure is both comprehensive and versatile, offering a dual-layer approach with both raw and encoded data. This allows researchers to explore not only privacy risks in raw personal information but also the behavior of encoded or transformed versions in privacy-preserving models.

At the core of the dataset are the primary identifiers and basic personal information. Each record features a unique identifier labeled as "*Id*" along with an accompanying “*IdLinked*” field, which mirrors the primary ID. These identifiers ensure data integrity and facilitate potential relational operations within larger databases. Additionally, the dataset includes straightforward personal details such as “*FirstName*” and “LastName” which provide the basic identity information of each individual.

Moreover, the dataset meticulously breaks down the date of birth into three distinct fields: “*DobDay*” for the day, “DobMonth” for the month, and “*DobYear*” for the year. This separation allows for more granular analysis and data manipulation, enabling users to isolate specific components of the date for targeted processing tasks (*Table I*).

TABLE I.

Id	IdLinked	FirstName	LastName	DobDay
1070	1070	Michaela	Neuman	11
1016	1016	Courtney	Painter	14

Table I. First 5 columns of the original database prior to pre-processing.

A key structural detail of the dataset is the presence of several additional columns that appear to be encoded versions of primary fields such as first name, last name, and date of birth components. These columns are labeled with suffixes like “*q2f32*” “q3f32” “*q2f64*” and “*q3f64*”. These suffixes refer to quantized Bloom filter encodings with different configurations, such as varying filter depths and bit precision (e.g., 32-bit vs. 64-bit). Bloom filters are often used in data anonymization for secure matching and linkage without revealing the original values. However, in the context of this project, these encoded columns are not utilized. Since they are not relevant to the analysis or implementation tasks performed here, they are excluded during preprocessing and will not be discussed further.

Additionally, the dataset includes an auxiliary column named “*index_level_0*” This auto-generated index reflects the original order of records as they were exported or saved. While not directly informative on its own, this index can be helpful as a reference to maintain consistency during data manipulation or when merging with other data sources.

The first step in using the dataset involves loading it from a CSV file into a data frame, enabling efficient manipulation in the programming environment. The preprocessing phase begins by removing duplicate entries and filtering out rows that contain missing values in key columns. This ensures the dataset is both reliable and consistent for downstream analysis. After cleaning, the focus shifts to preserving privacy by selecting only a subset of relevant columns. This target selection helps minimize exposure of sensitive information while still retaining the core attributes necessary for meaningful analysis. These steps collectively establish a solid foundation for all subsequent data processing and evaluation tasks (*Table II*).

TABLE II.

FirstName	LastName	DobDay	DobMonth	DobYear
Michaela	Neuman	11	11	1915
Courtney	Painter	14	12	1916

Table II. First 5 columns of the original database after preprocessing.

B. *Prompt engineering*

As part of the process of automating code generation for the application of DP mechanisms to sensitive data, a prompt was designed for interaction with a general-purpose LLM with code-generation capabilities. The objective was to obtain, as a direct output of the prompt, a complete Python script capable of applying the Laplace mechanism to a real-world dataset, grouping individuals by decade of birth. The dataset included columns such as first name, last name, and components of date of birth.

To guide the model toward generating correct and executable code, several prompting strategies were tested. Each strategy was selected based on assumptions about how LLMs respond to different forms of instruction and was evaluated for its effectiveness in this complex, multi-step programming task.

Initially, a goal-oriented prompting strategy was attempted. This was chosen because it aligns well with tasks where the LLM is expected to reason abstractly and generate creative or open-ended outputs. We expected that describing the overall objective in conceptual terms would give the model enough flexibility to design an appropriate solution. However, this approach proved ineffective for detailed programming tasks. The outputs were often incomplete, misused libraries, or made assumptions that didn't align with the dataset, such as incorrect column names or file formats.

Next, we explored a few-shot prompting approach, which included a short example of the desired output format. This method was chosen based on its success in tasks requiring style imitation or structural consistency. The expectation was that showing a partial working code snippet would help the model mirror the logic while adapting to the new context. However, the model often copied the example too literally, ignoring differences in column names or failing to implement the required DP logic, revealing its difficulty with adapting examples to new scenarios.

A minimal prompting strategy was also tested, under the assumption that less constrained instructions might lead the LLM to "think more freely" and creatively complete the task. However, the lack of specificity introduced ambiguity. The model frequently used inappropriate libraries, skipped key preprocessing steps, and made assumptions that reduced the output's reliability.

To test whether natural language structure might help, we applied a narrative-style prompt, where the instructions were embedded in a paragraph rather than numbered or explicitly segmented. This style is sometimes recommended for general comprehension tasks, and we hypothesized that it could lead to holistic interpretation of the task. However, this approach underperformed in the context of programming. Without clearly delineated steps, the model often omitted critical operations such as handling missing values or coercing data types. The narrative format lacked the precision and control needed for procedural tasks.

These iterative findings led us to adopt a structured, stepwise prompting strategy, which broke the task into explicitly numbered and logically ordered actions. This approach was chosen because it aligns with how LLMs are trained to follow detailed, sequential instructions — particularly for tasks involving multiple dependencies. The final prompt included all necessary technical details: the exact file path, column names, parameters (e.g., epsilon, sensitivity), and the expected output format. This strategy yielded the most complete and correct results, producing scripts that were immediately executable and followed best practices in applying the Laplace mechanism.

Through this iterative process, we developed a robust prompting method that not only ensures accurate code generation but is also adaptable to similar DP-related use cases. The final prompt template provides a reusable foundation for future automated privacy-preserving data analyses.

The final prompt (*Figure 1*), designed using this structured approach, will be implemented and evaluated across two LLMs, specifically within the ChatGPT and DeepSeek platforms. This cross-platform testing will assess the generalizability of the prompt structure and its effectiveness in guiding different LLM architectures to produce privacy-preserving, executable code in reproducible conditions.

C. *Privacy preserving implementation*

This section compares the performance of ChatGPT and DeepSeek in generating executable Python code for applying DP. Both models received an identical structured prompt, and the full outputs are available at the following links: ChatGPT-generated code and DeepSeek generated code.

I have a CSV file located at this path: `/Users/felipecastanogonzalez/Downloads/ChfSynthData-13_09_2024.csv`

I want a complete Python script that applies Differential Privacy using the Laplace mechanism on counts by decade of birth.

Please generate a clean, ready-to-run Python script that performs the following steps:

1. Loads the dataset using `pandas`.
2. Filters and cleans the data by: -Dropping duplicates -Removing rows with missing values in: `"FirstName"`, `"LastName"`, `"DobDay"`, `"DobMonth"`, `"DobYear"` -Converting `"DobYear"` to integer
3. Keeps only these columns: `"FirstName"`, `"LastName"`, `"DobDay"`, `"DobMonth"`, `"DobYear"`
4. Creates a new column `"Decade"` from `"DobYear"` by flooring it to the nearest multiple of 10 (e.g., 1985 → 1980).
5. Counts the number of records in each decade (`true_counts`).
6. Applies the **Laplace mechanism** (from `diffprivlib`) with `epsilon = 0.1` and `sensitivity = 1` to each decade count.
7. Produces a `DataFrame` that compares true counts vs differentially private counts by decade.
8. Prints the resulting comparison table.

Please include: -All necessary imports (`pandas`, `numpy`, `diffprivlib.mechanisms.Laplace`) -Clear code structure with step comments - `max(0, ...)` to prevent negative counts -Output printed as a clean comparison table using `to_string(index=False)`

Do not use placeholders. Use the exact path and column names I provided

Figure I. Final used prompt.

Both models correctly followed the multi-step instructions, generating code that loads the dataset, cleans and filters relevant columns, computes true counts per decade, and injects noise using the Laplace mechanism from the diffprivlib library. The resulting scripts were functional and adhered to the specified parameter values, including ε = 0.1 and sensitivity = 1.

Despite these similarities, notable differences were observed. ChatGPT introduced redundancy by calling the randomise function twice within the noise application loop, potentially introducing inconsistencies due to repeated sampling. *Figure 2* shows an excerpt from the code generated by ChatGPT. By contrast, *Figure 3* illustrates the DeepSeek code, which samples noise once per count and applies rounding and clipping consistently, making the output more deterministic and readable.

```python
#ChatGPT
# chf_differential_privacy.py

# Step 0: Imports
import pandas as pd
import numpy as np
from diffprivlib.mechanisms import Laplace

# Step 1: Load dataset
csv_path = "/Users/felipecastanogonzalez/Downloads/ChfSynthData-13_09_2024.csv"
df = pd.read_csv(csv_path)

# Step 2: Filter and clean data
df = df.drop_duplicates()
df = df.dropna(subset=["FirstName", "LastName", "DobDay", "DobMonth", "DobYear"])
df["DobYear"] = df["DobYear"].astype(int)

# Step 3: Keep only required columns
df = df[["FirstName", "LastName", "DobDay", "DobMonth", "DobYear"]]

# Step 4: Create 'Decade' column
df["Decade"] = (df["DobYear"] // 10) * 10

# Step 5: Count records per decade (true counts)
true_counts = df["Decade"].value_counts().sort_index()

# Step 6: Apply Laplace mechanism for Differential Privacy
epsilon = 0.1
sensitivity = 1

dp_counts = []
for count in true_counts:
    lap = Laplace(epsilon=epsilon, sensitivity=sensitivity)
    lap.randomise(count)
    noisy_count = max(0, int(round(lap.randomise(count))))
    dp_counts.append(noisy_count)

# Step 7: Create comparison DataFrame
comparison_df = pd.DataFrame({
    "Decade": true_counts.index,
    "True Count": true_counts.values,
    "DP Count (ε=0.1)": dp_counts
})

# Step 8: Print comparison table
print(comparison_df.to_string(index=False))
```

Figure II. Code snippet generated by ChatGPT showing redundant use of randomise ()

On the other hand, DeepSeek's implementation exhibited greater emphasis on readability and usability for downstream analysis. Specifically, it employed more descriptive variable names, such as *true_counts* and *private_counts* and included inline comments at each step to enhance interpretability for users unfamiliar with the differential privacy workflow.

Furthermore, DeepSeek structured its output within a DataFrame that explicitly labeled columns as "True Count" and "Private Count" offering greater semantic clarity. In contrast, while ChatGPT's script was more concise and functionally equivalent, it omitted contextual print messages and used less descriptive naming conventions, which may hinder code maintainability and interpretability.

Both implementations printed comparison tables without index labels using the "*to_string(index=False)*" method; however, DeepSeek improved user guidance by preceding the output with a descriptive message.

```python
#DeepSeek
import pandas as pd
import numpy as np
from diffprivlib.mechanisms import Laplace

# Step 1: Load the dataset
file_path = "/Users/felipecastanogonzalez/Downloads/ChfSynthData-13_09_2024.csv"
df = pd.read_csv(file_path)

# Step 2: Filter and clean the data
# Drop duplicates
df = df.drop_duplicates()

# Remove rows with missing values in specified columns
required_columns = ["FirstName", "LastName", "DobDay", "DobMonth", "DobYear"]
df = df.dropna(subset=required_columns)

# Convert DobYear to integer
df["DobYear"] = df["DobYear"].astype(int)

# Step 3: Keep only the required columns
df = df[required_columns]

# Step 4: Create Decade column
df["Decade"] = (df["DobYear"] // 10) * 10

# Step 5: Count records by decade
true_counts = df["Decade"].value_counts().sort_index().reset_index()
true_counts.columns = ["Decade", "TrueCount"]

# Step 6: Apply Laplace mechanism for differential privacy
epsilon = 0.1
sensitivity = 1
laplace = Laplace(epsilon=epsilon, sensitivity=sensitivity)

# Initialize list for private counts
private_counts = []

# Add noise to each count
for count in true_counts["TrueCount"]:
    noisy_count = laplace.randomise(count)
    private_counts.append(max(0, round(noisy_count)))  # Ensure non-negative counts

# Step 7: Create comparison DataFrame
result_df = true_counts.copy()
result_df["PrivateCount"] = private_counts

# Step 8: Print the comparison table
print("\nComparison of True vs Differentially Private Counts by Decade:")
print(result_df.to_string(index=False))
```

Figure III. Code snippet generated by Deepseek.

In summary, while both LLMs produced correct and executable code, DeepSeek demonstrated superior consistency, structure, and readability. These findings highlight the importance of evaluating not only code correctness but also implementation quality and reproducibility when applying LLMs in privacy-preserving applications.

D. *Evaluation metrics*

This section evaluates the effectiveness of DP when applied to a dataset grouped by birth decade, using a privacy budget of ε = 0.1. The analysis focuses on how well the DP transformation preserves the statistical structure of the original data while ensuring formal privacy guarantees.

Figure 4 presents the true histogram of birth decades, showing the original counts without any privacy-preserving transformation. The distribution is relatively uniform, with counts ranging between approximately 1430 and 1690 across decades from 1900 to 1990, with near-zero counts after 2000, as expected based on the dataset structure.

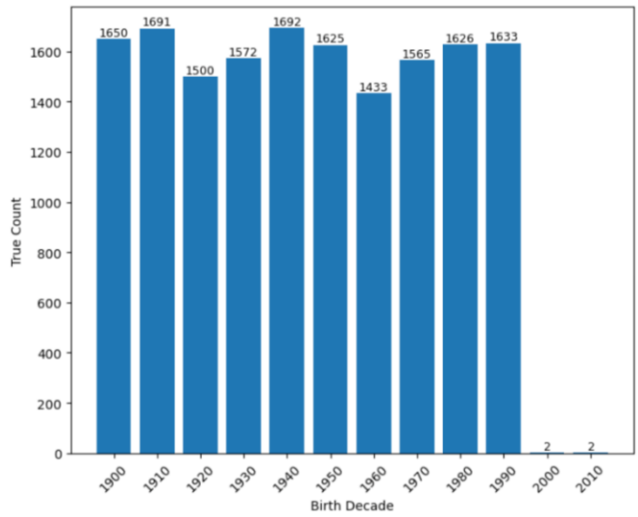

Figure IV. True histogram of birth decades.

The application of DP using the Laplace mechanism resulted in a modified histogram of birth decades that remains visually consistent with the original distribution. Although controlled noise was introduced, the global shape of the histogram, as well as the relative proportions across decades, remained intact (*Figure 5*). This suggests that the DP implementation successfully protected individual-level data without distorting the broader demographic trends.

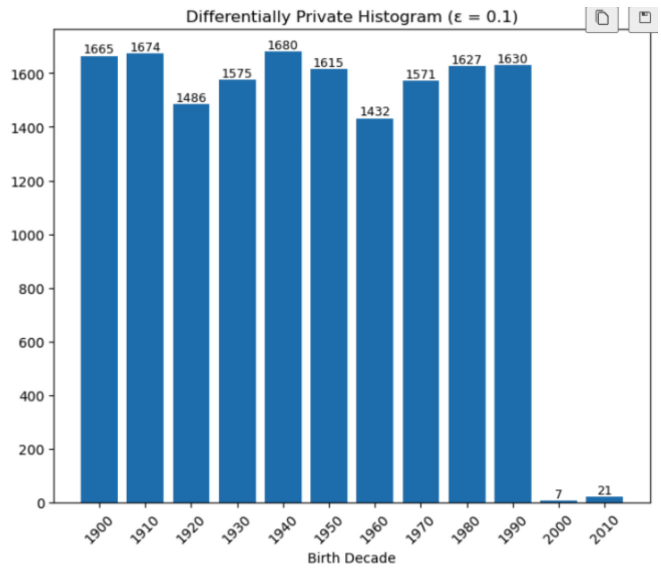

Figure V. Differential Privacy histogram (ε = 0.1).

Quantitative evaluation supports this observation. The MSE between the true and DP transformed counts was 116.33, indicating that the differences introduced by the privacy mechanism were modest and well-contained. The MAE was 8.83, meaning that, on average, the DP output deviated from the original count by approximately eight individuals per decade. Considering that the original group sizes exceed 1400 records per decade, this level of error represents a very minor perturbation relative to the data's scale.

Importantly, although the Pearson correlation coefficient was initially calculated as 1.0, suggesting a perfect linear relationship between the original and DP-protected distributions, this metric assumes that the underlying data follows a normal distribution, which is not the case for our dataset. Since our data consists of discrete, grouped birth decades, it violates the assumptions of normality, making the Pearson metric less appropriate for evaluating similarity in this context.

To address this, we employed the Spearman rank correlation coefficient, a non-parametric measure that assesses the monotonic relationship between variables based on their rank orders rather than their absolute values. Spearman correlation is particularly well-suited for non-normally distributed and ordinal data, making it more robust in our scenario.

The Spearman correlation coefficient was computed as 0.998, indicating a very strong positive association between the ranks of the true and DP-protected histograms. This high value suggests that the differentially private mechanism preserved the overall structure and trends of the original data, even if exact counts were perturbed. Consequently, the DP-protected dataset remains reliable for downstream analysis, including pattern recognition and group-level comparisons.

These findings demonstrate that DP, even under a strict privacy budget, can provide strong guarantees against individual re-identification while maintaining the analytical utility of the data. Furthermore, the results confirm the ability of LLMs to generate accurate, executable code that effectively implements DP in practice.

VI. Conclusion

Overall, both ChatGPT and DeepSeek demonstrated the ability to interpret structured prompts and generate executable Python code that integrates DP for protecting sensitive data. The outputs adhered to the intended structure, correctly applied the Laplace mechanism, and produced statistically valid results that maintained the analytical utility of the dataset. The findings support the growing evidence that LLMs can serve as effective assistants in the development of privacy-preserving software solutions.

From a technical standpoint, DeepSeek delivered code with superior internal consistency, particularly in its correct handling of the noise sampling process and its use of well-structured DataFrames with clearly labeled columns. These practices contribute to the reproducibility and transparency of DP implementations key principles in both research and production environments. Furthermore, DeepSeek's descriptive variable names and comprehensive documentation improve the interpretability and maintainability of the code, making it more accessible for developers and researchers working in sensitive domains such as healthcare or social services.

In contrast, ChatGPT's output, while functional and correctly applying the Laplace mechanism, introduced a redundancy in noise sampling that may lead to variation in results across repeated runs. Although this issue does not directly compromise privacy, it may affect consistency and reproducibility two crucial aspects in high-stakes applications where verifiability of results is essential. Additionally, ChatGPT's code was more concise but lacked the same level of contextual messaging and structure present in DeepSeek's output.

Importantly, the quantitative results showed minimal distortion introduced by the DP mechanism: the MAE was low, and the Spearman Correlation Coefficient was high. This indicates that the DP-protected data retained essential distributional properties, suggesting that both models can effectively produce code that satisfies the dual objectives of privacy and utility. These findings underscore the feasibility of using LLMs not just as code generators, but as tools for embedding formal privacy techniques directly into software pipelines.

Nevertheless, practical deployment of LLM-generated DP code requires careful validation. Neither model incorporated formal verification steps or exception handling mechanisms, which are critical for ensuring reliability in production settings. Future work could explore how prompt engineering or post-processing scripts can be used to enforce such standards automatically.

In conclusion, this study provides empirical evidence that LLMs like ChatGPT and DeepSeek can be leveraged as privacy-preserving development tools, capable of generating code that aligns with DP principles. While both models are viable, DeepSeek currently demonstrates stronger performance in terms of structure, clarity, and reproducibility. With further refinements, LLMs have the potential to accelerate the adoption of DP by lowering the technical barrier for implementation, especially in domains where data sensitivity and regulatory compliance are paramount.

References

[1] A. Aslan, M. Greve, T. O. Diesterhöft, and L. M. Kolbe, "Can Our Health Data Stay Private? A Review and Future Directions for IS Research on Privacy-Preserving AI in Healthcare," *Wirtsch. 2022*, Feb. 2022.

[2] C. Dwork and A. Roth, "The Algorithmic Foundations of Differential Privacy," *Found. Trends® Theor. Comput. Sci.*, vol. 9, no. 3–4, pp. 211–407, 2013, doi: 10.1561/0400000042.

[3] S. Fletcher, A. Roegiest, and A. K. Hudek, "Towards Protecting Sensitive Text with Differential Privacy," in *2021 IEEE 20th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)*, Shenyang, China: IEEE, Oct. 2021, pp. 468–475. doi: 10.1109/TrustCom53373.2021.00076.

[4] G. Feretzakis, K. Papaspyridis, A. Gkoulalas-Divanis, and V. S. Verykios, "Privacy-Preserving Techniques in Generative AI and Large Language Models: A Narrative Review," *Information*, vol. 15, no. 11, p. 697, 2024.

[5] D. Normile, "Chinese firm's large language model makes a splash," *Science*, vol. 387, no. 6731, pp. 238–238, Jan. 2025, doi: 10.1126/science.adv9836.

[6] C. Munley, A. Jarmusch, and S. Chandrasekaran, "LLM4VV: Developing LLM-driven testsuite for compiler validation," *Future Gener. Comput. Syst.*, vol. 160, pp. 1–13, Nov. 2024, doi: 10.1016/j.future.2024.05.034.

[7] I. Habernal, "When differential privacy meets NLP: The devil is in the detail," in *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, Online and Punta Cana, Dominican Republic: Association for Computational Linguistics, 2021, pp. 1522–1528. doi: 10.18653/v1/2021.emnlp-main.114.

[8] J. K, B. H, T. H, and M. A. P, “A review of preserving privacy in data collected from buildings with differential privacy,” *J. Build. Eng.*, vol. 56, p. 104724, Sep. 2022, doi: 10.1016/j.jobe.2022.104724.

[9] M. U. Hassan, M. H. Rehmani, and J. Chen, “Differential Privacy Techniques for Cyber Physical Systems: A Survey,” *IEEE Commun. Surv. Tutor.*, vol. 22, no. 1, pp. 746–789, 2020, doi: 10.1109/COMST.2019.2944748.

[10] T. Zhu, G. Li, W. Zhou, and P. S. Yu, “Differentially Private Data Publishing and Analysis: A Survey,” *IEEE Trans. Knowl. Data Eng.*, vol. 29, no. 8, pp. 1619–1638, Aug. 2017, doi: 10.1109/TKDE.2017.2697856.

[11] W. Liu, Y. Zhang, H. Yang, and Q. Meng, “A Survey on Differential Privacy for Medical Data Analysis,” *Ann. Data Sci.*, vol. 11, no. 2, pp. 733–747, Apr. 2024, doi: 10.1007/s40745-023-00475-3.

[12] M. Gong, Y. Xie, K. Pan, K. Feng, and A. K. Qin, “A Survey on Differentially Private Machine Learning [Review Article],” *IEEE Comput. Intell. Mag.*, vol. 15, no. 2, pp. 49–64, May 2020, doi: 10.1109/MCI.2020.2976185.

[13] S. Coffey, J. Catudal, and N. D. Bastian, “Differential privacy to mathematically secure fine-tuned large language models for linguistic steganography,” in *Assurance and Security for AI-enabled Systems*, J. D. Harguess, N. D. Bastian, and T. L. Pace, Eds., National Harbor, United States: SPIE, Jun. 2024, p. 19. doi: 10.1117/12.3013129.

[14] J. Ficek, W. Wang, H. Chen, G. Dagne, and E. Daley, “Differential privacy in health research: A scoping review,” *J. Am. Med. Inform. Assoc.*, vol. 28, no. 10, pp. 2269–2276, Sep. 2021, doi: 10.1093/jamia/ocab135.

[15] Y. Peng *et al.*, “From GPT to DeepSeek: Significant gaps remain in realizing AI in healthcare,” *J. Biomed. Inform.*, p. 104791, Feb. 2025, doi: 10.1016/j.jbi.2025.104791.

[16] X. Du, S. Hu, F. Zhou, C. Wang, and B. M. Nguyen, “FI-NL2PY2SQL: Financial Industry NL2SQL Innovation Model Based on Python and Large Language Model,” *Future Internet*, vol. 17, no. 1, p. 12, Jan. 2025, doi: 10.3390/fi17010012.

[17] Y. Fei, J. Fan, and G. Zhou, “Extracting Fruit Disease Knowledge from Research Papers Based on Large Language Models and Prompt Engineering,” *Appl. Sci.*, vol. 15, no. 2, p. 628, Jan. 2025, doi: 10.3390/app15020628.

[18] M. E. Kayaalp, R. Prill, E. A. Sezgin, T. Cong, A. Królikowska, and M. T. Hirschmann, “DeepSeek versus ChatGPT: Multimodal artificial intelligence revolutionizing scientific discovery. From language editing to autonomous content generation—Redefining innovation in research and practice,” *Knee Surg. Sports Traumatol. Arthrosc.*, p. ksa.12628, Feb. 2025, doi: 10.1002/ksa.12628.

[19] D. Normile, “Chinese firm’s large language model makes a splash,” *Science*, vol. 387, no. 6731, pp. 238–238, Jan. 2025, doi: 10.1126/science.adv9836.

[20] L. Rhomrasi *et al.*, “LLM performance on mathematical reasoning in Catalan language,” *Results Eng.*, vol. 25, p. 104366, Mar. 2025, doi: 10.1016/j.rineng.2025.104366.

[21] W. Shi, R. Shea, S. Chen, C. Zhang, R. Jia, and Z. Yu, “Just Fine-tune Twice: Selective Differential Privacy for Large Language Models”.

[22] W. Shi, A. Cui, E. Li, R. Jia, and Z. Yu, “Selective Differential Privacy for Language Modeling”.

[23] “Our Impact,” Calgary Drop-In Centre. Accessed: Feb. 03, 2025. [Online]. Available: https://calgarydropin.ca/who-we-are/our-impact/

[24] T. Wu, A. Panda, J. T. Wang, and P. Mittal, “Privacy-Preserving In-Context Learning for Large Language Models,” in Proc. 12th Int. Conf. Learn. Representations (ICLR), Vienna, Austria, 2024.

[25] L. Hu, I. Habernal, L. Shen, and D. Wang, “Differentially Private Natural Language Models: Recent Advances and Future Directions,” in Findings of the Association for Computational Linguistics: EACL 2024, St. Julian’s, Malta, Mar. 2024, pp. 478–499.