Differential Privacy for Vehicular Data Marketplaces

Master Thesis

Muhammad Aitsam

This work was submitted to the

Chair of Communication and Distributed Systems

RWTH Aachen University, Germany

Advisers:

Roman Matzutt, M. Sc.
Gonzalo Munilla Garrido, M. Sc.

Examiners:

Prof. Dr.-Ing. Klaus Wehrle
Prof. Dr. Florian Matthes

Registration date: April 27, 2020
Submission date: October 27, 2020

Zentrales Prüfungsamt/Central Examination Office

Eidesstattliche Versicherung
Statutory Declaration in Lieu of an Oath

Aitsam, Muhammad — 410342

Name, Vorname/Last Name, First Name — Matrikelnummer (freiwillige Angabe) / Matriculation No. (optional)

Ich versichere hiermit an Eides Statt, dass ich die vorliegende ~~Arbeit~~/~~Bachelorarbeit~~/ Masterarbeit* mit dem Titel

I hereby declare in lieu of an oath that I have completed the present ~~paper~~/~~Bachelor thesis~~/Master thesis* entitled

Differential Privacy for Vehicular Data Marketplaces

selbstständig und ohne unzulässige fremde Hilfe (insbes. akademisches Ghostwriting) erbracht habe. Ich habe keine anderen als die angegebenen Quellen und Hilfsmittel benutzt. Für den Fall, dass die Arbeit zusätzlich auf einem Datenträger eingereicht wird, erkläre ich, dass die schriftliche und die elektronische Form vollständig übereinstimmen. Die Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

independently and without illegitimate assistance from third parties (such as academic ghostwriters). I have used no other than the specified sources and aids. In case that the thesis is additionally submitted in an electronic format, I declare that the written and electronic versions are fully identical. The thesis has not been submitted to any examination body in this, or similar, form.

Aachen, 27.10.2020

Ort, Datum/City, Date — Unterschrift/Signature

*Nichtzutreffendes bitte streichen

*Please delete as appropriate

Belehrung:
Official Notification:

§ 156 StGB: Falsche Versicherung an Eides Statt
Wer vor einer zur Abnahme einer Versicherung an Eides Statt zuständigen Behörde eine solche Versicherung falsch abgibt oder unter Berufung auf eine solche Versicherung falsch aussagt, wird mit Freiheitsstrafe bis zu drei Jahren oder mit Geldstrafe bestraft.

Para. 156 StGB (German Criminal Code): False Statutory Declarations
Whoever before a public authority competent to administer statutory declarations falsely makes such a declaration or falsely testifies while referring to such a declaration shall be liable to imprisonment not exceeding three years or a fine.

§ 161 StGB: Fahrlässiger Falscheid; fahrlässige falsche Versicherung an Eides Statt
(1) Wenn eine der in den §§ 154 bis 156 bezeichneten Handlungen aus Fahrlässigkeit begangen worden ist, so tritt Freiheitsstrafe bis zu einem Jahr oder Geldstrafe ein.
(2) Straflosigkeit tritt ein, wenn der Täter die falsche Angabe rechtzeitig berichtigt. Die Vorschriften des § 158 Abs. 2 und 3 gelten entsprechend.

Para. 161 StGB (German Criminal Code): False Statutory Declarations Due to Negligence
(1) If a person commits one of the offences listed in sections 154 through 156 negligently the penalty shall be imprisonment not exceeding one year or a fine.
(2) The offender shall be exempt from liability if he or she corrects their false testimony in time. The provisions of section 158 (2) and (3) shall apply accordingly.

Die vorstehende Belehrung habe ich zur Kenntnis genommen:
I have read and understood the above official notification:

Aachen, 27.10.2020

Ort, Datum/City, Date — Unterschrift/Signature

Abstract

In 21st century world has become an explosion of information as data storage is cheap and accessible. Many organizations use this data to conduct research, track user's behavior, or to recommend products. Companies need to know as much as possible about their customer. This massive generation of data also gives birth to privacy concerns. Companies want to analyze data without compromising the user's privacy. Differential Privacy provides this guarantee during statistical analysis. Differential Privacy aims to attain the broadest range of data while achieving a robust, significant and mathematically accurate privacy definition. Many companies came up with their solution to use Differential Privacy for real-world applications. This thesis is divided into two parts. In part 1, we analyze available Differential Privacy libraries and frameworks for synthetically generated datasets and real-world use-case. In part 2, we design a differentially private query management system for the vehicular data marketplace. Our system tries to answer statistical questions while preserving the user's privacy. This thesis aims to provide a thorough comparison of Differential Privacy libraries and how they can be used for vehicular data marketplaces.

Acknowledgments

First of all I would like to thank Roman Matzutt and Gonzalo Munilla Garrido for the great supervision and the huge amount of time he dedicated to the support of this thesis. During my thesis, I asked a whole lot of questions and I thank you for the patience to answer them all. I also want to express my thanks to Prof. Dr.-Ing. Klaus Wehrle for the opportunity to write this thesis about such a fascinating topic. Apart from that, I am thankful for the resources that were provided by the chair. Additionally, I would like to thank Prof. Dr. Florian Matthes for kindly agreeing to take on the task of second examination.

I would like to thank my family for their support during the last month and especially my parents. A special thanks to Monis for pushing me to work hard and stay motivated. I also want to thank you for proofreading and notifying me about several grammatical errors. Finally, I want to thank my friends (Asad, Sana, Alvina, Amna) for their encouragement during the time of the thesis. In particular, I am grateful for the support and motivation during the very stressful last weeks.

Contents

1

Introduction

For many years, personal data is being collected by the government, hospitals, companies for different purposes, which sometimes help them in decision making to set or achieve their organizational goals. However, this data might be shared or sold to other organizations for further analysis. This data sharing helps researchers and companies to analyze data again for potential applications, but due to the presence of sensitive information in the dataset, it is crucial to preserve the privacy of an individual. The problem is how to analyze and share statistical data without compromising an individual's privacy. In a setting where the curator owns a database containing specific information, a privacy breach occurs when an adversary infers this information. An adversary can use different techniques or preys on the background knowledge, even when the released data is anonymized.

In the automotive industry, millions of cars are generating data every second. This data can be analyzed and shared to improve services and quality of future products, but it is not safe to share or sell these statistics due to privacy concerns. Figure 1.1 shows the basic idea of the vehicular data marketplace. The car sensor data is stored in a database, and data analysts can query that database to get the statistical results. These results can be converted into data products to be sell in the data marketplace. While querying the database, there is a possibility that data analysts get knowledge about someone whose data is present in the database. This leakage of user privacy is undesirable. That is why there is a need for a system that can protect an individual's privacy while sharing useful statistics. For many years different techniques are developed to achieve this goal, but we have seen privacy protection failures which lead to the re-identification of individuals e.g., Insurance Commission (GIS), and Netflix Award. Some privacy-preserving mechanisms like Data Anonymization [7] and Data Encryption [24] were proposed earlier, but later on, research shows that these methods could not stand some well-known attacks like Homogeneity Attack [43], Background Knowledge Attack [34], and Inference Attack [42]. More details about these privacy failures and privacy mechanisms are in Section 2.2. These privacy failures could be harmful to a company in many ways, and they also reduce customer confidence in sharing their data.

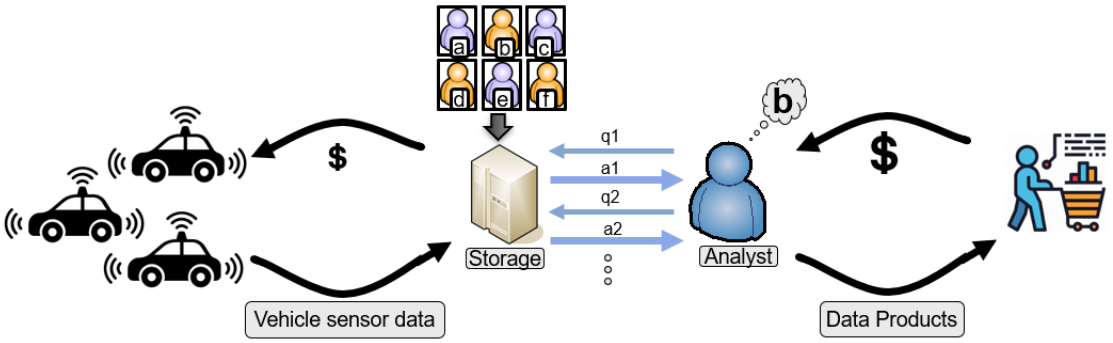

Figure 1.1 The general idea of how data marketplace works. The car sensor's data is submitted to the database. Data analysts query the database to get the statistical results. These results are then converted into data products it can sell in the data marketplace. The car owner gets the reward for sharing his car sensor's data

The Goal of this thesis is to give a solution to the problem, as mentioned above, by using differential privacy technique. The concept of Differential privacy was introduced by Cynthia Dwork [14] in 2006, which promises the privacy preservation of an individual's data. It is based on the idea that the outcome of a statistical analysis is equally likely independent of whether an individual's data is in the dataset or opt-out of the dataset. It has been studied theoretically and proved that it guarantees privacy even if adversaries have background knowledge [15]. Differential privacy mechanisms can make confidential data widely available for data analysis without compromising an individual's sensitive information. Some state-of-the-art research institute already released their differential privacy libraries and frameworks. Thus, we will conduct subjective, performance, and comparative analysis of these libraries and frameworks for ideal case and real-world use cases.

We also designed and implemented a query management system for vehicular data marketplace that maintains enough privacy while releasing the statistics. This study helps the reader to choose best library or framework for their specific use case and it will also benefit the automotive industry to share statistics while maintaining individual privacy.

The main contributions of this thesis include:

- Investigation of major privacy protection methodologies and privacy failures in the recent past.
- Theoretical and practical analysis of all available Differential Privacy libraries and frameworks.
- The basic query management system can be used by the automotive industry to share or sell their data products without compromising an individual's privacy.

First, we conducted a comparative analysis of Differential Privacy with other privacy techniques to prove the importance of Differential Privacy for statistical queries. Then we searched for state-of-the-art tools and carried out experiments on common grounds. We used them for a real world use case by using our privacy budget strategy.

Right now, there is no other benchmark that provides thorough comparison of all available libraries and frameworks. Our approach helps to distinguish different behaviors of libraries and frameworks under different conditions. It will also help the community to select appropriate Differential Privacy Library for their specific use case.

1.1 Outline

This thesis is organized as follows: In Chapter 2, we discuss pre-existing privacy-preserving methodologies before differential privacy. We present the reasons why Differential Privacy should be used instead of other techniques. Then we discuss different settings, mechanisms and techniques for achieving Differential Privacy theoretically and for a real-world problem.

In Chapter 3, we discuss the problem we are solving in our work and some general assumptions under which all experiments are conducted.

In Chapter 4, we conduct a detailed comparative analysis of all available differential privacy libraries and mechanisms. We analyzed them for their general characteristics, functionality and performance.

In Chapter 5, we discuss the setting of our real-world use case: Eco-Friendly Driving Model. We also showed how a synthetic dataset is generated from the initially available dataset and calculated Eco-Friendly scores. We used all possible Differential Privacy libraries for our use case. This helps us design a basic query system for data marketplace that could be used in the automotive industry to release statistical analysis without compromising individual privacy. We also introduced a our privacy budget calculation strategy.

In Chapter 6, we state possible future work and conclude the thesis.

2

Background

The first formal definition of statistical data privacy covering all essential aspects was given by Dalenius in 1977 [15]. According to him, anything that can be learned from the statistics about an individual should be determined with or without access to the database [15]. In other words, the difference in learning something new when someone is in the dataset and when someone is not in the dataset should be very tiny. This is precisely a definition of semantic security [48]. Practically, it is impossible to achieve this tiny difference because here, the advisory and legitimate recipient is the same person (data analyst). He/ She might have good reasons to get the results, or they could also have bad motivations. To solve this issue and to limit the harm to the teachings of the database instead of somebodies participation in it, Cyntia Dwork came up with a new definition of Privacy: the risk of an individual's privacy (e.g., risk of being denied automobile insurance) should not substantially increase due to participating in a database [17].

A significant amount of work has been done to protect privacy. This chapter will discuss some effective mechanisms, techniques, and models to preserve Privacy, along with some real-world examples of privacy failures. After that, we discuss why Differential Privacy is better than other privacy definitions, what are the significant properties and mechanisms of Differential Privacy. Finally, some statistical operations are discussed in the context of Differential Privacy.

2.1 Privacy Sensitivity of Attribute Types

The data collected from all sources is stored in a database, and it is categorized into four primary fields, including *non-sensitive attributes, sensitive attributes, quasi-identifiers,* and *specific identifier.*

Non-sensitive Attributes include attributes that are not listed in other fields. Sensitive data contain personal information, such as salary and medical records. Quasi-Identifier has characteristics which can be combined with other data to identify an

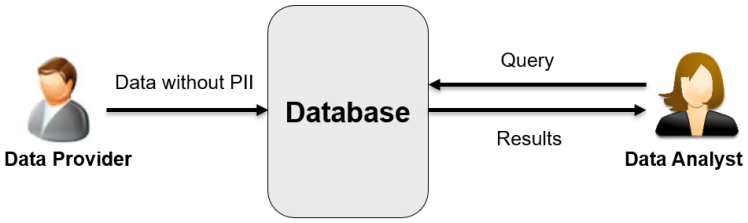

Figure 2.1 Basic Model for Data Querying

individual. Specific Identifiers include attributes that can be used to identify an individual (e.g., name, social security number). Other than these categories, personally identifiable information (PII) is any information that can be used stand-alone to identify an individual. Before releasing the data, the data provider must remove all such information while keeping the dataset valuable. Now we will discuss different privacy-preserving models and techniques. Figure 2.1 shows the basic model for data querying.

2.2 General Method for Privacy Preservation

Many models have been developed so far to preserve privacy, but they all have some limitations. In this section, we will briefly go through these models. The purpose of this discussion is to give provide motivation that why Differential Privacy is better than other techniques used for statistical analysis. Subsequently, one can realize the shortcoming of these approaches. Later in the section, we can argue why the Differential Privacy is better for statistical analysis.

2.2.1 Privacy Model

The process of removing personally identifiable information from the data to protect an individual's Privacy before sharing it or using it for data analysis is known as data anonymization [62]. The data collector (curator) modifies the data by removing the specific identifiers like name, phone number, address, security number, etc.

This approach showed promising results in some cases[9]. Still, many privacy failures occurred in the past, which shows that even when these specific identifiers are removed, the availability of an individual's background knowledge can be used by an adversary to re-identify him/her [42]. Once the data is released to a third party, data owners can't control how the data is used or manipulated. An MIT graduate student, Latanya Sweeney, had shown that linking anonymized data with publicly available data can reveal an individual's information. We will discuss more these privacy failures in the next section.

2.2.1.1 k-Anonymity

To solve the shortcomings of data anonymization, researchers proposed other methods to preserve Privacy. Samarati and Sweeney[49] proposed the idea of K-anonymity,

which says that data can be released with a scientific guarantee that a particular individual's information cannot be uniquely distinguished while utilizing data. The property of the K-anonymized dataset is that each person in the record is similar to at least another K-1 other records on the potentially identifying variables. K-anonymization prevents linking the released data to other information sources. But later, Machanavajjhala et al. [37] proved that K-anonymity does not guarantee privacy in some circumstances.

2.2.1.2 l-Diversity

Through K-anonymity, one can maintain Privacy against record identification; however, it is not successful for protecting against inference attacks of sensitive attributes. Machanavajjhala et al. proposed a new notion called l-diversity [37]. The l-diversity requires that each tuple with identical quasi-identifiers has at least l-diverse value for the sensitive attribute. It provides Privacy even when the curator does not know what kind of background knowledge is possessed by the adversary.

The l-diversity provides Privacy against linkage attacks, but Li et al. [33] showed that l-diversity does not cover all issues of attributes disclosure. They presented two attacks to demonstrate their claims. *Skewness attack* where the dataset has skewed distribution and *Similarlity attack* where the equivalence class contains semantically similar sensitive values. The dataset used to prove l-diversity privacy was much different from the real-world population; that is why it fails to prevent attributes disclosure.

2.2.1.3 t-Closeness

To overcome the deficiency of l-diversity, Li et al. proposed a new notion called t-closeness. The formal definition is stated below[33]:

An equivalence class is said to have t-closeness if the distance between the distribution of a sensitive attribute in this class and the distribution of the attribute in the whole table is no more than a threshold t. A table is said to have t-closeness if all equivalence classes have t-closeness.

The set of data that has the same value for their quasi-identifiers is known as an equivalence class. This reduces the risk of the adversary to learn something unique about individual. The distribution between sensitive attributes is measured using a metric called Earth Mover's Distance (EMD) which considers the semantic proximity of the feature values. This technique does preserve feature disclosure but in many cases identity is still disclosed. For some special cases, the combination of K-anonymity and t-closeness can be used to preserve the privacy of published data [46].

2.2.2 Data Encryption

In the previous section, we gave several examples of privacy attempts that went wrong and there are many more in the literature. According to Dwork [15], they all were definition failure. Those models do not cover all aspects of Privacy.

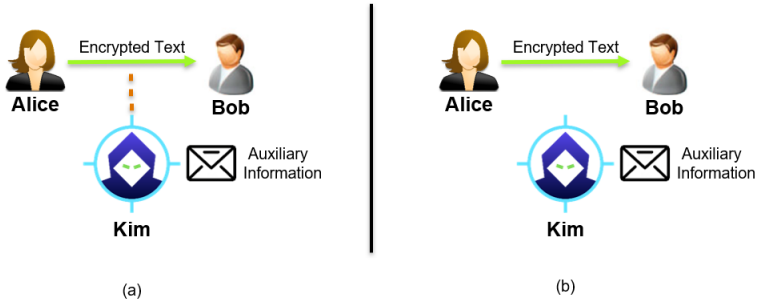

Figure 2.2 Alice sending encrypted text to Bob, Kim hacked communication channel to get encrypted text. (a) Kim successfully got encrypted text and he also has access to auxiliary information. (b) Kim only has access to auxiliary information. According to Goldwasser and Micali, the difference between (a) and (b) must be small.

But encryption system shows tremendous definition success. Goldwasser and Micali [48] gave the definition of modern cryptography, which is still famous in theoretical computer science. To understand their definition, consider the following case: Alice wants to share some plain text with Bob. To keep communication secure, Alice encrypted the plain text and sent it to Bob. So now, there is only one decryption key that can convert encrypted text into plain text. When Bob receives the encrypted text, an adversary Kim hacked the communication channel and successfully obtained the encrypted text.

According to Goldwasser and Micali, the difference between an adversary's ability to reach plain text, when he/she links encrypted data with auxiliary information and when he/she only has auxiliary information, should be minimal. For a better understanding of the scenario, see Figure 2.1 (a, b). It was assumed that Bob couldn't be an adversary because if Bob is malicious, then with the decryption key, he got access to the whole plain text.

Now the question arises that is there any similar definition for statistical databases? The answer is yes, in 1977, five years before Goldwasser and Micali came up with their definition, the data privacy definition by Dalenius already existed. We already discussed his definition in the first paragraph of this chapter and how it leads to Differential Privacy. Before jumping more into Differential Privacy, let us look at some attacks and privacy failures, which pushed researchers to come up with better solutions for data privacy. It will provide more motivation to prefer Differential Privacy over other techniques.

2.3 Privacy Attacks

This section will discuss some famous attacks used to breach privacy when models mentioned in section 2.2 are applied to preserve privacy. These attacks can be used either separately or in combination to re-identify individuals.

Homogeneity Attack

This attack showed that the adversary could identify the value of sensitive attributes of the k-group in the dataset when there is little diversity among them. Due to

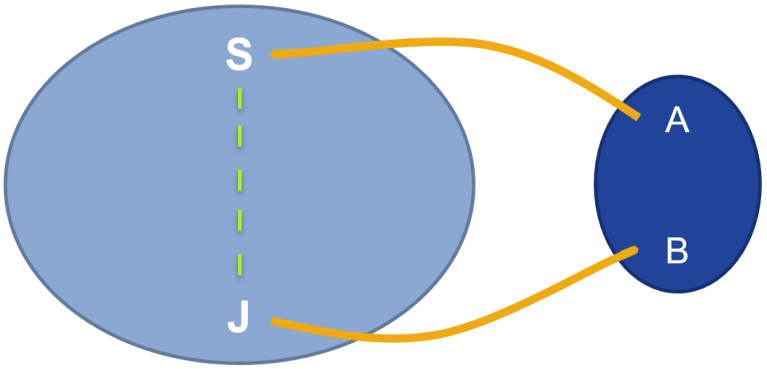

Figure 2.3 *A* and *B* are sybils, linked with *S* and *J* to find the connection between them. When the released dataset is anonymized, researchers used certain algorithms and techniques to identify accounts *A* and *B* and then through their link, they were able to identify that is there any connection between account *S* and *J*. Figure 2.3 shows these links.

similarity in most of the data, the adversary can get the exact value of sensitive data and use it for malicious activities. So, even when the data is anonymized, it can still leak Privacy.

Background Knowledge Attack

In such attacks, the adversary uses background knowledge to reveal useful information from an anonymized dataset. For instance, a man knows that his neighbor is very old. One night he saw an ambulance in his street. The next morning he read a local newspaper, which shows the graph that last night, three accident cases, two cardiac cases, and two diabetic cases were reported in the local hospital. From background knowledge, a man knows that either his old neighbor had a heart attack or she has a severe diabetic issue.

Therefore, from the above example, it can be seen that Privacy is compromised by using background knowledge.

Active Attack

The attack in which the adversary enrolls several fake users (sybils) to the system and creates unique structural patterns among them, which later can be identified after anonymization.

In one of the experiments, researchers created 12 new accounts and created artificially social links (e.g., sending an email back and forth). *A* and *B* are two of such accounts; their connection is built with accounts in system *S* and *J*. When the released dataset is anonymized, researchers used certain algorithms and techniques to identify accounts *A* and *B* and then through their link, they were able to identify that is there any connection between account *S* and *J*. Figure 2.3 shows these links.

All of these attacks can be neutralized by using the privacy definition of Differential Privacy, where the results occur irrespective of an individual's presence and absence.

2.4 Privacy Failures in Past

Every day a tremendous amount of data is collected and shared in multiple ways, which led to an increase in privacy concerns. Many studies in the recent past showed that most of the US population could be uniquely distinguished by joining their date of birth, gender, and zip code [52, 20]. If the individual is identified in the released data, their private data will be disclosed. The disclosure of personal information can be harmful to the user and it is also against GDPR.

It is tough to protect the dataset from background knowledge attack. It is impossible to predict what background information the adversary has, which makes privacy preservation models vulnerable. This section will discuss some famous privacy failures, which started a big debate regarding these issues.

AOL Privacy Debacle

In 2006, AOL disclosed anonymized data of over 650,000 users. The disclosure was done without users' consent because the AOL team thought they anonymized data well enough not to threaten anybody's Privacy. But, the New York Times demonstrated [6] that individuals can be identified based on these search queries. After this story was released, AOL removed that data from the site and apologized for its release. But copies of these records continue to circulate online, risking the Privacy of many Americans. This attack also taught how these companies could compile such data and study them on a micro level to learn more about their user behaviors. Besides this, the consequences of storing and compiling data at one place are what Marc Rotenberg, the executive director of the Electronic Privacy Information Center, called 'a ticking privacy time bomb [6].'

Insurance Commission (GIS)

The MIT graduate student Latanya Sweeny has shown that an individual's privacy can be compromised even when the released data is anonymized. She demonstrated this by revealing the medical records of Massachusetts' Governor. She accomplished this task by using some background knowledge and then linking the anonymized data released by the hospital with the publicly available data (e.g., voter registration list).

It was public that on May 18, 1996, the Governor of Massachusetts collapsed during a ceremony. As released, data by GIS doesn't contain this information. Sweeney used her background knowledge from media regarding hospital name and address. She also bought the registered voter list for tens of dollars. This list contains many attributes, including names, birth dates, zip codes, gender from the governor's residential area. She linked these quasi-identifiers such as zip code, birth date, gender with data released by GIC to re-identify the governor's medical data [7].

Netflix Prize

In 2006, one of the most extensive online DVD rental services, NETFLIX, released an anonymized dataset containing 100 million movie ratings provided by 500,000 of its subscribers. They started a competition called NETFLIX Prize to develop a better movie recommendation system.

NETFLIX claimed that they secured Privacy by removing all the personal information and only kept information like unique user ID, ratings and dates the subscriber

rated the movie. This time students from the University of Texas, Narayanan and Shmatikov, demonstrated that with very little background knowledge about the subscriber and linking the data with publicly available data, an individual's privacy could be compromised. They used IMDB (The Internet Movie Database) [43] as a source of background knowledge. By linking this database with the NETFLIX database, they were able to re-identify many subscribers. After this research was published, NETFLIX canceled its second competition[43].

Besides the above-mentioned privacy failures, much research confirms the privacy breach when the released data is considered to maintain user privacy. S.Ezzini et al. [19] showed that it is possible to re-identify the user's from her vehicle sensor data in the automotive industry. Another research by S.Lastyan et al. [31] showed that extracting vehicle sensor signals from can logs can lead to users' re-identification. All mentioned privacy failures are proof that even when the released data is anonymized, there is still the possibility of a privacy breach. This privacy breach can be harmful to the users participating in the study. To motivate maximum users for participating in any particular study, there was a need for better techniques to give the participants confidence that their privacy will not be compromised whatsoever. All the above mentioned privacy attacks can be neutralized and all these privacy failures could be prevented with Differential Privacy. In the next section, we will discuss Differential Privacy in depth.

2.5 Differential Privacy

Differential Privacy gives a firm definition for data privacy. It is based on the idea that the outcome of the statistical analysis is equally likely independent of whether an individual joins or refrains from joining the dataset. This statement covers all the requirements for data privacy. An adversary can bring harm or good to any individual or group regardless of their presence in the dataset. Due to this property of Differential Privacy, it is considered a promising privacy-preserving method. Since the birth of Differential Privacy, researchers have explored many corner cases in the theoretical world. For practical cases, there is still a vast area to explore. Usually, it is done by adding noise to the query results. This section will discuss the importance of Differential Privacy, its formal definition, properties, mechanisms, and some common statistical operations.

Differential Privacy encourages users to share their data for statistical analysis by promising them that the adversary will not be able to re-identify them irrespective of the auxiliary information he/she has. Differential Privacy claims that nothing could be learned about an individual while learning useful information about a population.

Compared to other privacy-preserving models we discussed in Section 2.2, the Differential Privacy definition covers all aspects of data privacy and provide strong theoretical guarantees for statistical analysis. With the passage of time, The General Data Protection Regulations (GDPR) are getting tough because sensitive data is becoming more vulnerable. That is why modern research on statistics is focusing on Differential Privacy.

2.6 Definitions of Differential Privacy

In this section, we will discuss the formal definitions of Differential Privacy. But before that, we must understand the critical parameters used in Differential Privacy.

- ε - The privacy parameter which can be controlled by the data analyst to maintain the trade-off between privacy and accuracy. ε-differential privacy is known as *pure differential privacy.*
- δ - The parameter which tells the probability of privacy leak (ε, δ)-differential privacy is known as *approximate differential privacy.*
- D1 and D2 - Neighboring dataset (differ by only one element).

Lets have a look on two formal definitions of Differential Privacy. The first definition is ε-Differential Privacy also known as pure Differential privacy. Here the δ is considered to be zero and ε is the only privacy parameter. The second definition is (ε, δ)-Differential Privacy also know as approximate Differential Privacy. Here the δ value is not equal to zero and it tell the probability of privacy breach. So, for approximate Differential Privacy we have two (ε, δ) privacy parameters.

2.6.1 ε - Differential Privacy

Let ε>0. *Define a randomized function M to be (ε)-differentially private if for all neighboring input datasets D1 and D2 differing on at most one element and* $\forall S \subseteq$ *Range(M), we have*[15].

$$\frac{Pr[M(D1) \in S]}{Pr[M(D2) \in S]} \leq e^{\varepsilon}$$

where the probability is taken over the coin tosses of M[17]. The above equation can also be written as:

$$\Pr[\mathrm{M(D1)} \in \mathrm{S}] \geq e^{\varepsilon} .\Pr[\mathrm{M(D2)} \in \mathrm{S}]$$

The probability of output in S on a D1 dataset is at least e^{ε} times to the probability of output in S on a D2 datasets. For better understanding of neighboring datasets in Differential Privacy, see Figure 2.1.

2.6.2 (ε, δ) - Differential Privacy

Define a randomized function M to be (ε, δ)-differentially private if for all neighboring input datasets D1 and D2 differing on at most one element and $\forall S \subseteq$ *Range(M), we have*[21].

$$\Pr[\mathrm{M(D1)} \in \mathrm{S}] \geq exp(\varepsilon) \times \Pr[\mathrm{M(D2)} \in \mathrm{S}] + \delta$$

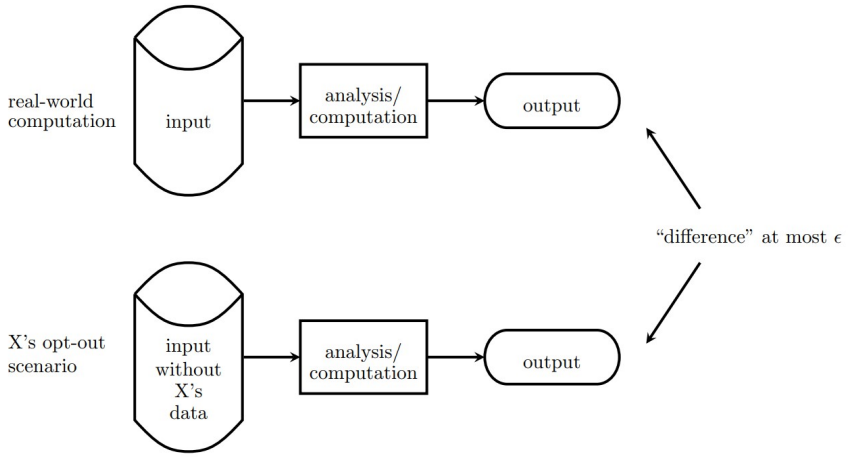

Figure 2.4 The first dataset is complete dataset from real world computation and second one is neighboring dataset where information of $'X'$ is removed or replaced. The difference of output is at most of ε.

In the above equation, we have two privacy parameters. Epsilon (ε) and delta (δ). Where delta (δ) is the probability of privacy leakage.

For instance, suppose J is an output that possibly discloses *K's* identity or data, where that parallel dataset D2 does not contain *K's* data, so we can say Pr[M(D2) $\in$ S] = 0. In such case ε - differential privacy, M can never output K on any dataset, while (ε, δ) - differential privacy may output K with probability up to δ.

From these definitions, we can conclude that information acquired regarding the participant by the output of some algorithm should be the same or no more than the information acquired regarding the participant without accessing the output. In a later section, we will see the practical use of these definitions.

2.7 Properties of Differential Privacy

2.7.1 Sensitivity

Sensitivity is a parameter which decides how much noise required in differential privacy mechanism. There are mainly two types of sensitivity being used in differential privacy.

2.7.1.1 Global Sensitivity

It is the maximum difference between the query results of two neighboring datasets that are going to be used in one of the differentially private mechanisms. The formal definition[18]:

For $f : D^n \rightarrow \Re^k$, *and use the l1 norm on* $\Re^k$ *as a distance metric on outcomes of* f. *Then, the global sensitivity of* f *is*

$$GS(f) = \max_{D_1, D_2} \|f(D_1) - f(D_2)\|_1$$

Usually, while using differential privacy for a real-world use case, global sensitivity is easy to implement, especially for queries like **count** and **sum**.

2.7.1.2 Local Sensitivity

Nissim[45] proposed the concept of local sensitivity that satisfies the differential privacy only by adjusting the difference between query results on the neighboring dataset. The formal definition of local sensitivity is as follows:

For $f : D^n \to \Re^k$ *and* $x \in D^n$*, the local sensitivity of* f

$$LS(f) = \max_{D_2} \|f(D_1) - f(D_2)\|_1$$

For queries like **count** or **range**, local sensitivity behaves almost the same as global sensitivity. For more complex queries like **median** and **average**, global sensitivity adds much higher noise than local sensitivity.

From the above definitions, we can observe that on many queries, every differentially private algorithm must add noise at least large as local sensitivity. It is hard and still a hot topic in the differential privacy community to decide how much noise is enough since noise can also leak information. That is why the concept of privacy budget was introduced.

2.7.2 Privacy Budget

If no information is leaked despite many computations carried out on the best-case scenario, one can claim that mechanism used provides an absolute privacy guarantee. But practically, this absolute privacy guarantee is impossible to achieve. The concept of privacy budget was introduced to restrict the number of queries[47].

2.7.3 Composability

Another important property of Differential Privacy is Composability. Currently, the *parallel composition* and the *sequential composition* are mostly used in the design of mechanisms[39].

2.7.3.1 Parallel Composability

When we have a sequence of queries made on non-intersecting sets, parallel composability could be used. The ultimate privacy in this composition will be guaranteed by the largest privacy budget. In 2009, McSherry[38] gave the proofs of this composition theorem.

Theorem: (Parallel composability $\mathcal{A}$ is $\|\varepsilon - differentially private$)

Let $k_i(D)$, for some $i \in I$, be computations over D_i providing ε-differential privacy. If each D_i contains data on a set of subjects disjoint from the sets of the sets of subjects of D_j for all $j \neq i$, then $(k_i(D))_{i\in I}$ provides ε_i-differential privacy.

Proof. Let D_1, D_2 be two neighboring datasets. Assume that the j-th partition contains the differing element. Then

$$\begin{aligned}\Pr\left[\mathcal{M}(\mathcal{D})_1 = x_k\right] &= \prod_{i=1}^{k} \Pr\left[\mathcal{M}_i\left(D_1; x_1, \ldots, x_{i-1}\right) = x_i\right] \\ &\leq e^{\varepsilon} \Pr\left[\mathcal{M}_j\left(D_{2j}; x_1, \ldots, x_j\right) = x_j\right] \prod_{i\neq j}^{k} \Pr\left[\mathcal{M}_j\left(D_1; x_1, \ldots, x_{i-1}\right)\right] \\ &= e^{\varepsilon} \Pr\left[\mathcal{M}\left(D_2\right) = x_k\right]\end{aligned}$$

2.7.3.2 Sequential Composability

The chance of computing the results of independent differentially private calculations in sequence on a dataset, without surrendering privacy could be accomplished in sequential composability, while the privacy budget is included for each progression.

Theorem: (Sequential composability $\mathcal{A}$ is $\|\varepsilon - differentially private$[39])

Let $k_i(D)$, for some $i \in I$, be computations over D providing ε_i-differential privacy. The sequence of computations $(k_i(D))_{i\in I}$ provides $\left(\sum_{i\in I} \varepsilon_i\right)$-differential privacy.

Proof. Let D_1, D_2 be two neighboring datasets. Then

$$\begin{aligned}\Pr\left[\mathcal{M}(\mathcal{D})_1 = x_k\right] &= \Pr\left[\mathcal{M}_1\left(D_1\right) = x_1\right] \Pr\left[\mathcal{M}_2\left(D_1; x_1\right) = x_2\right] . \Pr\left[\mathcal{M}_k\left(D_1; x_1..x_{k-1}\right)\right] \\ &\leq e^{k\varepsilon} \prod_{i=1}^{k} \Pr\left[\mathcal{M}_i\left(D_2; x_1, \ldots, x_{i-1}\right) = x_i\right] \\ &= e^{k\varepsilon} \Pr\left[\mathcal{M}\left(D_2\right) = x_k\right]\end{aligned}$$

2.7.4 Data Release Setting

There are two commonly used settings for data releasing, *Interactive* and *Non-Interactive*. Both settings have their advantages and disadvantages. In the process of private data release, there are two parties, *data provider*, that has dataset with

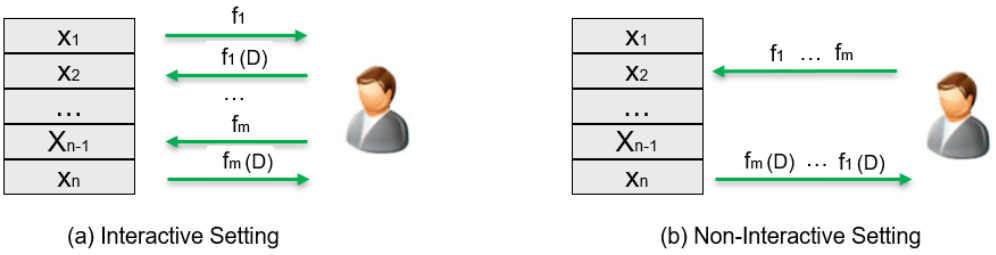

Figure 2.5 (a) interactive setting where noise is added to the answer of individual query. (b) non-interactive setting where set of queries are answered in batch and then noise is added.

sensitive information and *data analyst*, that lay out a sequence of queries. Both parties are trustworthy in this case. It is essential to have in-depth knowledge about specific use cases before finalizing the data release setting. Figure 4.1 (a,b) illustrate these two settings.

2.7.4.1 Interactive Setting

In this setting, one query is asked at a time. An interactive differential privacy interface is inserted between the *data analyst* and the dataset to answer the queries. Moreover, the query answer depends on the available privacy budget. Once the privacy budget is over, the curator will be unable to ask further queries[3]. For such a setting, even if we know all the queries in advance, we can run an interactive mechanism on each of them to get privacy-preserving answers. That is why this setting is preferred in many cases as compared to a non-interactive setting.

Throughout our analysis in this study, we used an interactive setting where noise is added to an individual query's answer.

2.7.4.2 Non-Interactive Setting

In this setting, a set of queries can be answered together, resulting in higher flexibility for data analysis than interactive settings. In a non-interactive setting, higher noise is added to the answers given for queries to confirm differential privacy. Thus, this setting's major issue is how to answer more queries with a limited privacy budget while maintaining the balance between privacy and utility. It is typically used with large datasets[3].

2.8 Differential Privacy Mechanisms

In this section, we will introduce some of the important mechanisms used in differential privacy. We will also discuss how sensitivity is calculated in each of these mechanisms.

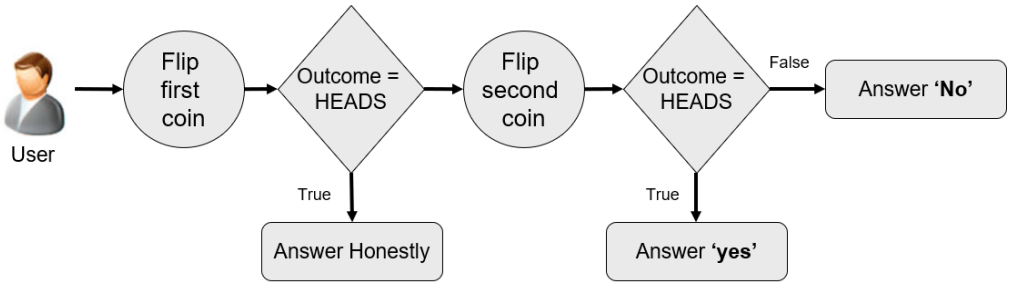

Figure 2.6 Randomized response procedure where decision is made on the results of coin flips. For first toss if heads come on tossing the coin, then a user has to answer honestly(*yes or no*. For second toss, If tails come on tossing, then the user will toss again, and this time if its heads, then, the user will answer *yes* otherwise the user will say *no*.

2.8.1 Randomized Response Mechanism

The randomized response is a case of non-interactive schemes, where each user data is perturbed individually based on the decision of coin flips. This procedure provides 'plausible deniability' to the respondent[51]. The algorithm of randomized response is as follows:

Whenever a question is asked, each user has to flip the coin.

- If heads come on tossing the coin, then a user has to answer honestly(*yes or no*).
- If tails come on tossing, then the user will toss again, and this time if its heads, then, the user will answer *yes* otherwise the user will say *no*.

Figure 4.2 is the illustration of the randomized response.

2.8.2 Laplace Mechanism

Laplace mechanism is one of the most widely used mechanisms in differential privacy. In this mechanism, the random noise that adjusts to Laplace distribution with mean 0 and sensitivity $GS(f)/\varepsilon$ is added to the response of each query to make it perturbed appropriately[18]. Usually, in ε - differential privacy, the Laplace mechanism is used because ε is the only the concerned parameter when computing noise with $l1$ sensitivity. Throughout this study, we are focusing on ε-differential privacy; that is why it is important to understand the basic definition and theorem proved by Dwork[18].

Definition: The Laplace Mechanism

Given any function $f : \mathcal{D}^n \to \mathbb{R}^k$, the Laplace mechanism is defined as:

$$\mathcal{M}_L(x, f(.), \varepsilon) = f(x) + (Y_1, \ldots, Y_k)$$

Following is the theorem for Laplace mechanism.

Theorem:

Let $f : \mathcal{D}^n \to \mathbb{R}^k$ be a real valued query of sensitivity 1. Then the mechanism $\mathcal{M} = f(D) + \frac{GS(f)}{\varepsilon}$ satisfies $(\varepsilon, 0)$ -differential privacy where $GS(f)$ was defined in previous section.

Proof. Let D_1, D_2 be neighboring databases at any point $x \in \mathbb{R}$. It is enough to compare the probability density f of $\mathcal{M}(D_1)$ with the g of $\mathcal{M}(D_2)$. Hence, without loss of generality we could assume that $GS(D_1) = 0$. Put $c = f(D_2)$ and note that $|c| \leq \Delta = \Delta(q)$. Hence, may assume that

$$\begin{aligned}\frac{f(x)}{g(x)} &= \frac{\exp(-\varepsilon|x|/GS(f))}{\exp(-\varepsilon|x-c|/GS(f)} \\ &\leq \frac{\exp(\varepsilon|x|/GS(f))}{\exp(-\varepsilon|x|/GS(f)\exp(-\varepsilon|c|/GS(f))} \\ &\leq \exp(\varepsilon)\end{aligned}$$

2.8.3 Gaussian Mechanism

Another mechanism that has received much attention in the recent past is the Gaussian mechanism[35] to achieve (ε, δ) - differential privacy. Here a certain amount of zero-mean Gaussian noise is added to the query result. The amount of noise added to the result scales with $l2$ sensitivity. Another factor that plays an important role here, along with ε, is δ, which determines the probability of privacy leakage. Liu et al[35] proved a generalized Gaussian mechanism for differential privacy.

2.8.4 Exponential Mechanism

Not all queries return numerical values to their output. Hence, McSherry and Talwar[40] came up with a method that can be applied to make non-numeric queries differentially private. After using the Exponential mechanism to non-numeric queries, the final output would be close to ideal answers since the mechanism appoints exponentially higher probabilities of being selected to the higher outcomes. The conclusion of theorem proved by McSherry[40] is:

$$\frac{\Pr\left[\mathcal{M}_E(D_1, q) = r\right]}{\Pr\left[\mathcal{M}_E(D_2, q) = r\right]} \leq \exp(\varepsilon)$$

2.8.5 Median Mechanism

It is an interactive differential privacy mechanism that answers arbitrary queries $f1, ..., fk$ without knowing the future queries. Theoretically, it performs better than other mechanisms (e.g., Laplace mechanism) when answering a large number of queries and gives fixed constraints. It is also suitable for identifying and defining queries equivalence in the interactive setting[56]. Median mechanism categorize queries as *hard* and *easy* with very low privacy cost.

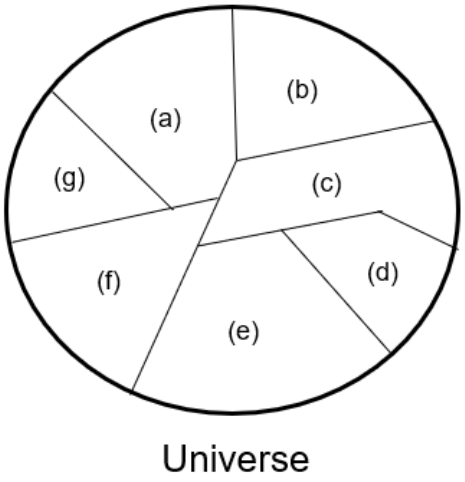

Figure 2.7 Universe is divided into seven cells. Each person can only be present in one cell at a time. So, sensitivity of Histogram query is 1.

2.9 Differential Privacy Statistical Operations

In this section, we will discuss some common statistical operations and their sensitivity calculation formulas that will be used to add noise to the query result. The major work to find out the equation for sensitivity calculation is done under Harvard Open Differential Privacy project[**?**].

Histogram Query

A histogram is a type of column statistic that sorts value into buckets. It is a great way to see and understand the distribution of data. To understand the sensitivity of histogram queries in the Differential Privacy world, assume that we have a massive universe from where the dataset elements are drawn. In the histogram query, we partitioned the universe in disjoint cells (cell(a) ...cell(g)), and the query is asking for each of these cells how many people live in each cell. Figure 4.3 illustrates this situation. Since a histogram is a partition of the universe, each person can only be in one cell. Someone's presence or absence in each cell can change the histogram query outcome by only one. This proves that the histogram query's sensitivity is always one, independent of the number of queries.

Counting Query

One of the most commonly used queries is the counting query. It helps to determine the total number of times a specific event happened. To make counting query differentially private, noise is added to the outcome. As the presence or absence of any user in one particular cell can change the outcome maximum by 1, so $l1$ and $l2$ sensitivity of counting query is always 1. The Laplace mechanism amount of noise added to the correct outcome is calculated by $\frac{1}{\varepsilon}$. In counting query, for $queryq, datasetD$ with size n. We use,

$$q(D) = \frac{1}{n} \sum_{i=1}^{n} q\left(x_i\right)$$

to evaluate counting queries on datasets.

Sum Query

It is an aggregate function that calculates the sum of desired events. It is widely used for answering statistical questions. For dataset x with n number of rows, Sum query will be:

$$s(x) = \sum_{i=1}^{n} x_i$$

The l_1 and l_2 sensitivity for sum query is proved in *Open Differential Privacy*[**?**] white paper. According to that *"Say the space of data points X is bounded above by M (maximum) and bounded below by m (minimum). Then s over X^n has l_1, l_2-sensitivity in the change-one model bounded above by:"*

$$M - m$$

Mean Query

Another important query in data analysis is the mean query. It is an aggregate function that calculates the mean of a numerical dataset. For dataset x with n number of rows, Mean query will be:

$$f(X) = \frac{1}{n} \sum_{i=1}^{n} x_i$$

The maximum difference between two possible answers of the mean query is known as sensitivity of mean query. Formally it is defined as *"Say the space of data points X is bounded above by M and bounded below by m. Then $f(.)$ has $l1$, $l2$-sensitivity in the change-one model bounded above by"*

$$\frac{M - m}{n}$$

Where n is the total number of rows in dataset.

Variance Query

In statistics, the measurement of the spread between numbers in a dataset is known as Variance. It helps to measure how far each number in the dataset is from the mean.

$$s^2(x) = \frac{1}{n-1} \sum_{i=1}^{n} (x_i - \bar{x})^2$$

where $\bar{x}$ refers to the sample mean of x

The $l1$, $l2$-sensitivity of the Variance query is calculated by the following equation:

For X bounded between m and M, $f(.)$ has $l1$, $l2$-sensitivity in the change one model bounded above by

$$\frac{n-1}{n^2}(M-m)^2$$

Standard Deviation Query

Standard deviation is derived from Variance. It is the measure of depression of a set of data from its mean. In simple words, the standard deviation is the square root of Variance. The following equation calculates the $l1$, $l2$-sensitivity of standard deviation:

For X bounded between m and M, $f(.)$ has $l1$, $l2$-sensitivity in the change one model bounded above by

$$\frac{M-m}{\sqrt{n}}$$

So far, we have briefly discussed the importance, properties and mechanisms of Differential privacy. Furthermore, we have also seen some general queries and their equations to calculate sensitivity. In upcoming chapters, we will discuss and analyze the available libraries and frameworks of differential privacy.

3

Problem Statement

After the idea of Differential Privacy got viral in 2006, many studies were conducted to prove it theoretically in different settings. Comparatively, fewer researchers and companies come up with the practical implementation of Differential Privacy for real-world use cases.

3.1 Related Work

All the companies who develop their solutions to use Differential Privacy practically claim to make dataset private for the desired accuracy level. Many pieces of research are available that compare different aspects and use cases of Differential Privacy. These are the studies where Differential Privacy libraries and frameworks are compared. N.Amiet[2] compared IBM-diffprivlib, Google-Differential Privacy, and diffpriv to check their behavior with different epsilon values and sizes of datasets. The comparison focuses only on the effect of sizes on the privacy provided by each library. Moreover, the comparison is brief and did not consider other available libraries. Another research was carried out by Asseffa and Seleshi[3] that focuses on the overall picture of Differential Privacy, but they also briefly discussed some Differential Privacy libraries. This study aimed to give general information about available libraries and frameworks instead of conducting a comparative analysis among them. The overall discussion is high level and not all the libraries are considered in the discussion. Stacy et al. [54] studied the effect of data skewness on inference vulnerability while doing machine learning with Differential Privacy. This study gave an excellent intuition skewness effect on inference vulnerability in machine learning with Differential Privacy. This study also had a high-level discussion on some of the Differential Privacy libraries that can be used for machine learning applications. Johnson et al.[28] compared the results of CHORUS with weighted-PINQ. In this paper CHORUS framework for Differential Privacy is introduced and to prove that CHORUS results are better, they compared it with weighted-PINQ, which is another Differential privacy framework. This study provides useful ideas for comparing the

performance of different Differential Privacy frameworks. They only focus on a few available frameworks, so we cannot use their idea to compare several libraries and frameworks. Linkovate Team [13] published an article in 2018 in which they compared the top organizations in Differential Privacy, the leaders in Differential Privacy and countries leading the advancement in this field. This comparison was mostly according to the business point of view to give readers an idea about the importance of Differential Privacy in business. In a recently published paper P.Jain et al [25], the research was conducted on how the Differential Privacy can be useful in big data analysis. The paper discussed the fundamental idea of Differential Privacy, along with that, they mentioned the importance of mechanism and other properties. The paper does not dig deep into the comparison of different tools under different conditions. Another survey was conducted by F.Boenisch[8] to analyze the predictability of results in the context of Machine Learning. In this survey, a brief theoretical comparison of Differential Privacy mechanisms is stated. In all the mentioned related work, there is a lack of thorough comparison for all available Differential Privacy libraries and frameworks.

These are the studies where improvement in Differential Privacy algorithms was made. Li et al. [32] proposed their Differential Privacy algorithm, which claims to improve the accuracy of the linear computation. For this purpose, they divided the queries into several workloads. They proved their results by comparing them with the results of other Differential Privacy algorithms. The proposed algorithm shows better results as compared to other existing algorithms. Hay et al.[22] proposed the evaluation framework for standardized evaluation of privacy algorithms. They conducted this study for 15 published algorithms on a total of 27 datasets. This study provide a detailed comparison of Differential Privacy algorithms. The overall research is in-depth and they included every vital aspect of Differential privacy algorithms. Ji et al. [26] explored the interplay between Differential Privacy, machine learning algorithms and learning-based data release mechanism. Their survey compared the Differentially Private machine learning results with non-Differentially Private results to calculate the utility. This survey paper helps to understand how to maintain the balance between privacy and utility while calculating privacy parameters in machine learning. Balle et al [5] gives optimal Gaussian mechanisms where the Variance can be calibrated directly by using the Gaussian cumulative density function. This improvement in the existing Gaussian mechanism opens a new chapter in the field of Differential Privacy. More research work is underway to improve the available tool for specific use cases. Subsequently, Zhang et al. [59] show how the results for the Histogram query can be made more accurate. They introduced a new clustering framework and evaluated the trade-off between approximation error due to clustering and Laplace error due to Laplacian noise's insertion, which was overlooked in the prior study. This paper gives a good idea about what factor should be taken into account when improving the existing algorithm. In research conducted by k.Minami et al. [41] the improvement was made in existed classical exponential mechanism of Differential Privacy. They also showed the results of how the Differential Privacy exponential mechanism behaves without sensitivity. These related work will help us to understand how existing mechanisms can be improved. Moreover, it will help us identify the issue if the results of specific mechanisms are not according to theoretical results.

These are the publications where Differential Privacy was analyzed for the real-world use-case. The book published by T.Zhu, G.Li et al. [61] focuses on differential privacy and its applications. It provides different strategies for researchers to implement Differential Privacy for real-world applications. Moreover, these books also discuss many practical scenarios where Differential Privacy can be useful. It provides a general checklist for the selection of appropriate values for Differential privacy parameters while applying it to real-world use-case. Besides this, the book covers all the essential aspects of Differential Privacy in detail. D.Vu et al. [57] demonstrated how Differential Privacy could be integrated with classical statistical testing in clinical trials where participants' information is sensitive. According to them, their developed methodology is concrete and can be used by researchers for all statistical analyses. They derived the rules for sample size adjustment to achieve better utility results while maintaining the individual's privacy. J.Liu et al. [36] showed how the Differential Privacy could be used to defend smart homes against Internet traffic analysis. They created a scenario where the adversary wants to know the resident's behavior by analyzing the sensors' data in smart homes. The proposed design considers the network energy consumption and resource constraints in IoT devices while achieving a strong Differential Privacy guarantee so the adversary cannot link any traffic flow to a specific home. These works are related to real-world applications of Differential Privacy. They gave good ideas for selecting appropriate privacy parameters while applying Differential Privacy in the real world.

These are the studies where Differential Privacy was used for the Automotive domain. B.Nelson and T.Olovsson [44] discussed how and when Differential Privacy could be used in the automotive domain. Furthermore, they elaborated on the challenges of using Differential Privacy in the automotive industry and gave recommendations for moving forward. Another publication of B.Nelson [4] is about data privacy for big automotive data. Here he used a Differential Privacy mechanism to prevent an individual's privacy in big data generated by connected vehicles. Initially, he describes the pros and cons of other privacy techniques and then showed how Differential Privacy is better for statistical analysis of vehicle data. The paper by F.Kargl et al. [29] focuses on how the concept of Differential Privacy can be applied to the intelligent transport system. They illustrated the integration of Differential Privacy with privacy policy language and policy-enforcement framework. They also showed how the appropriate mechanism could be integrated with their framework. P.Zhao et al. [60] conducted a local differential privacy survey to secure the internet of vehicles. Usually, in practical scenarios, global Differential Privacy is used. In local Differential Privacy (LDP), the noise is added to the individual dataset before it is submitted to centralized storage. Every user has to add the privacy of her choice. This survey gives an overview of existing LDP studies, their advantages and disadvantages, and computation complexity. Although the available literature on LDP is less than global Differential Privacy and focuses more on theoretical proofs, it gave us a good idea of conducting a comparative analysis for Differential Privacy libraries and frameworks. M.Johanson et al. [27] described a technological framework for capture and analysis of objective measurement and subjective use experience data in an automotive application. This research focuses on the first part of the data marketplace where data is fetched from car sensors and stored in a centralized database. These publications are related to the use of Differential Privacy in the automotive domain. They are useful to understand how and

when we can apply Differential Privacy and if we have more than one framework, how can we compare them.

These are the studies related to the privacy-preserving data marketplace. N.Hynes et al. [23] proposed *Sterling* which is privacy-preserving data marketplace. To preserve privacy in this decentralized marketplace, privacy-preserving smart contracts are used, which run on a permission-less blockchain. Data providers and consumers write these contracts. Sterling also provides a mechanism for data providers to control the use of their data. The research conducted by R.Cummings et al. [11] showed the appropriate use of Differential Privacy for growing databases. The databases are continuously growing in data-marketplace, but most of the Differential Privacy algorithms focus on static settings where queries are made on a fixed dataset. Here, they improved some of the existing algorithms to return differentially private results for growing databases with minimal loss of accuracy. Mengxiao Zhang [1] proposed an incentive mechanism for sharing the data in the privacy-preserving data marketplace. He explored this pricing mechanism for a buy-sided market, sell-sided market, two-sided market and two-sided platform. Agora [30] is another blockchain-based data marketplace that provides a facility for multiple parties to get compensated for sharing the data without relying on the trusted third party during an exchange. It uses cryptographic techniques to achieve data privacy and output verifiability. These related work give us a strong intuition of how our data marketplace should behave and the significant aspects we must focus on while designing our data marketplace.

So far, we have seen the use of Differential privacy in different domains. First, we discussed related work where the Differential Privacy libraries and frameworks are compared. We found that there is still a thorough comparison of all available libraries and frameworks that claims to confirm Differential Privacy. Then we discussed related work where improvement has been made in the existing mechanism. These papers will help us identify the mechanism's issue if our theoretical results mismatch with our practical results. After that, we saw some related works where Differential privacy was used for real-world applications. These studies help us to understand the essential aspects one should keep in mind while applying Differential Privacy. We also saw related work where Differential Privacy is used in the automotive domain. But we haven't found any paper which directly deals with the query management system in the automotive domain. Finally, we saw some papers related to the data marketplace. This helps us to understand how exactly the data marketplace works.

In this study, all available Differential Privacy libraries and frameworks are compared thoroughly. The work is divided into two parts. The first task is to evaluate libraries and frameworks for different types of datasets. The second task is to design a differentially private query management system for the vehicular data marketplace.

In the next section, we presented two scenarios for our analysis and assumptions and challenges related to our evaluation. All the libraries and frameworks will be evaluated under these scenarios. In Scenario 1 we have fully synthetically generated datasets while in scenario 2 we have datasets containing car sensors data. This chapter is concluded by the precise problem statement for this thesis.

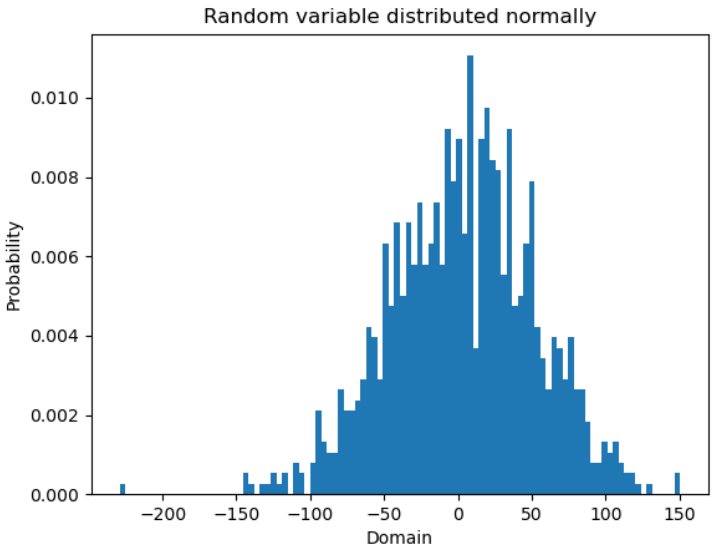

Figure 3.1 Gaussian Normal Distribution.Rows: 1000, Mean: 0, Std: 50.

3.2 Scenarios

Scenario 1

In this scenario, we have several datasets of Gaussian normal distribution. Each dataset either differ by scale, size or shape. We considered three scales (1000, 10000, 100000), three Std (50, 250, 500) and three shapes (skew: 0, 5, 50). So, in total, we have twenty-seven (27) unique datasets. It will help us to analyze libraries and frameworks under different scenarios. Figure 3.1 shows graph for one of these datasets, having n = 1000, std = 50 and skew = 0.

Scenario 2

In this scenario, we have data generated by car sensors. It will help us analyze behavior fo the drivers, so it can decided how many drivers are driving eco-friendly. Eco-friendly driving (or eco-driving) is an efficient way of driving that emphasizes fuel efficiency, speed, and safety. Anyone can eco-drive and save fuel. Whether you are driving a commercial truck or hybrid vehicle, eco-driving techniques improve fuel mileage. It has been verified that driving methods could affect fuel consumption and emission. According to De Vliger[12], aggressive driving consumes 12% - 40% more fuel. Another research by Mierlo et al.[55] showed that 5% - 25% fuel could be saved from acceptable driving practices. In conclusion, we can say that just by improving small driving habits, one can save a lot of money spent on fuel.

To evaluate the eco-driving behaviors, we created a synthetic dataset from the dataset provided by BMW. Initially available columns (*Time Stamp, Longitudinal Acceleration, Latitude Acceleration, Actual Speed, RPM Speed, RPM Torque, Brakes*) were insufficient to evaluate eco-friendly driving behavior. Especially, instantaneous fuel consumption column was missing, which we derived from other available attributes [58]. Other than that we used machine learning model, linear and multi-linear regression to extend the dataset from 1 lap to 4 laps. Finally, we landed with 25 unique datasets ready to be evaluated for eco-driving behavior. Each dataset contains following columns:

Time Stamp	Total Acceleration	Speed	Torque	Brakes	Fuel
sec	m/s^2	m/s	Nm	$1/0$	ltr

We assume that our synthetic datasets are true and similar to real-world datasets as they are derived from real datasets. Secondly, we assumed that the data-processor (curator) is semi-honest and curious, so she wants to get the answers statistical questions without learning something about individual whose data is present in dataset. Thirdly, we assumed that there is no traffic on roads and every car traveled the same distance of 4 laps. For the calculation of Eco-driving scores, we divided into two segments. The first segment is fuel-related a hundred-mark score and the second segment is cases related to hundred-mark score[10].

Fuel related score: Here the fuel consumption per hundred kilometer is calculated for every dataset and the answer is converted into hundred-mark score by using the following formula:

$$\text{Segment-FPH} = 40 + \left(1 - \frac{(\text{FPH} - 4.7621)}{6.9621 - 4.7621}\right) \times 60$$
$$\text{FPH} \in [4.7621, 6.9621]$$

For our current datasets, minimum and maximum true FPH is 4.7612 and 6.9612, respectively. This will change if we add more datasets.

Cases related score: Here, we considered five cases to determine the driving behavior.

1. If an increase in the vehicle's acceleration is more than or equal to 6m/2sec^2, it will be considered one event of *Acceleration Sharply*.

2. If a decrease in acceleration of vehicle is more than or equal to 6m/2sec^2, it will be considered one event of *Deceleration Sharply*.

3. If the growing acceleration lasts more than three seconds, then it is identified as one *Long-Time Accelerating* event.

4. If for six seconds, the average speed is more than or equal to 120 km/h or maximum speed is more than or equal to 130 km/h, or speed var is more than 676 km/h, then it will be considered as one event of *High-Cruising*.

5. If the brake is pressed straight for three seconds, it will be considered one *Brake-Pressing* event. The percentage of all the cases is averaged to reach the final Cases related score. The average of both cases is the total eco-driving score of the driver.

In conclusion, the described scenarios will be used to evaluate the libraries and frameworks at the same level. Particularly, scenario 2 will also help us to decide the further statistical question for the Differential Privacy query management system (second part of the thesis).

3.3 Problem Statement

This thesis is divided into two parts. In the first part, we conducted extensive comparative analysis of open source Differential Privacy libraries and frameworks

(*OpenDP-Whitenoise, Google-PyDP, Google-PySyft, IBM-diffprivlib, CHORUS-DP, diffpriv, PINQ, GUPT, Tensorflow-DP, MIT-PrivateMultiWeight*). We analyzed the behavior of each library or framework for scenario 1 and scenario 2. Analyzing with Scenario 1 will provide information regarding library or framework behavior with different dataset sizes, scales and shapes. In contrast, Scenario 2 will help us understand how libraries or frameworks behave for real-world use cases. Furthermore, we analyzed *error*, *scaled error*, *std of error and scaled error*, and *95% of error and scaled error* for Scenario 1. The challenge was to understand the architecture and the purpose of the library or framework so the common ground for comparison can be decided. The goal is to **1.**discuss non-technical features of every library or framework, **2.**evaluate and understand the library and framework's behavior, and **3.**identify best library or framework so it can be used in the 2nd part of the thesis.

For the 2nd part, the goal is to develop a basic query management system that confirms an individual's privacy while answering the statistical question. We have to use a library or framework we got from the analysis in the first part of the thesis. Our system must have these features: **1.**All statistical questions must be related to the automotive industry. **2.**The system should allow data analyst to choose from available statistical questions return a differentially private answer. **3.**It must not allow data analyst to ask the same question twice. **4.** The system must follow our defined privacy budget strategy. Here we are assuming that all statistical questions are independent of each other.

4

Analysis of Differential Privacy Libraries and Frameworks

In this part of the thesis, we analyze Differential Privacy libraries and frameworks. This results of this analysis will help to select the appropriate library or framework for a specific use-case. Furthermore, the limitations of libraries and frameworks can also be determined through this analysis. Differential Privacy libraries and frameworks considered in this study are *OpenDP-Whitenoise* [53], *Google-PyDP*, *Google-PySyft*, *IBM-diffprivlib*, *CHORUS-DP*[28], *diffpriv*[18], *PINQ*[38], *Tensorflow-DP*, *MIT-PrivateMultiWeight*[21], *GUPT*. All of them are open source and their code is available on Git.

4.1 Comparison Framework

Criteria for comparing the above-mentioned libraries and frameworks can be divided into general characteristics, functionality, performance and usability.

4.1.0.1 General Characteristics

Some available product features are considered by general characteristics, including product name, architecture, programming language, and the product's purpose. We will also look into non-technical aspects, like documentation and community.

4.1.0.2 Functionality

Every library and framework's desired task is to make the dataset differentially private and provide users enough flexibility to control the parameters. Even though all libraries and frameworks accomplish their desired task, they don't offer the same

functionality. In this section, a comparison will be conducted on the basis of functionality.

4.1.0.3 Performance

After understanding the functionalities, we will conduct a performance analysis. This section will see how the library or framework behaves in certain conditions and how much time they consume for each Differential Privacy query. All libraries and frameworks are compared under the common grounds. For the performance analysis of all libraries and frameworks, we had Gaussian normal distribution datasets with scales (1000, 10000, 100000), Std (50, 250, 500) and shapes (skew: 0, 5, 50). Each library is evaluated for its Laplace mechanism to obtain ε-Differential Privacy.

In Section 4.2, we will discuss each library separately and we will see their behavior for statistical operations *(count, sum, mean, standard Deviation and Variance)*. For the first library, we will discuss results for *Std of scaled error* of all queries to give the general overview of library behavior. After that, we only discuss the unusual scenarios when the library or framework behaves unexpectedly. Section 4.3 will analyze our finding by comparing the results of all libraries and framework. In section 4.4, we will discuss the results of eco-driving use case. Finally, we will analyze our all results and find out the best library among all.

4.2 Libraries and Frameworks

4.2.1 IBM-diffprivlib

4.2.1.1 General Characteristics

IBM-diffprivlib is a general-purpose, open-source library for experimenting, investigating and developing Differential Privacy applications. The programming language used is Python. It can also be used for machine learning applications. IBM-diffprivlib is designed in such a way that it can be modified to create new models and mechanisms. It is easy to install and use. Privacy parameters can be easily changed according to the user's desire. All the models are well documented and the community of this project is very active. IBM-diffprivlib only focuses on global sensitivity, which means the noise is added to the actual results. The amount of noise is decided by sensitivity and epsilon.

It is available under *MIT license*. Recently the new version *v0.3.0* of the library is released where they updated some general features. We initially analyzed it for *v0.2.0* and we noticed that results are similar. It currently supports Windows, Linux and Mac operating systems.

4.2.1.2 Functionality

IBM-diffprivlib doesn't require any specific pre-requisite libraries to be installed. It also works perfectly with some powerful data analysis libraries like *Pandas* and *Scipy*.

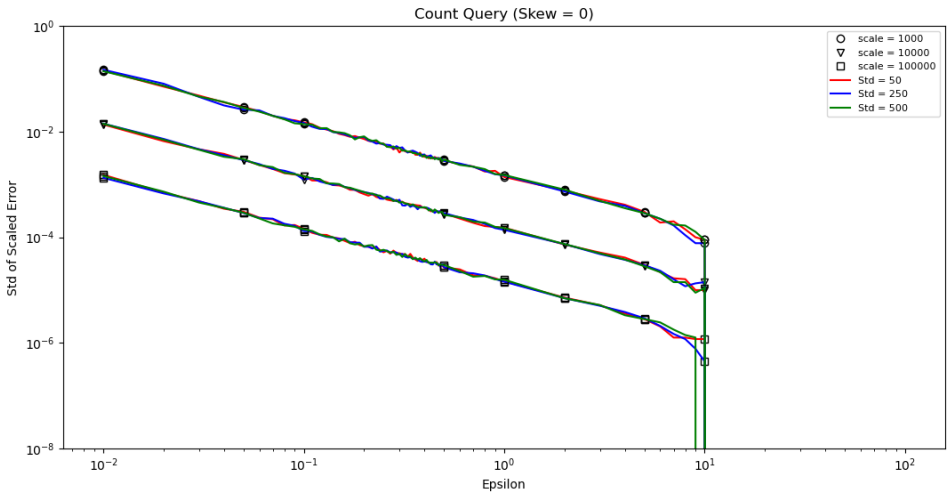

Figure 4.1 shows the behavior of library for *Count query* for datasets = 1000, 10000, 100000, Std = 5, 250, 500 and skew = 0. With increase in size of dataset, the standard deviation of scaled error decreases. Datasets with different standard deviation values does not effect the results.

IBM-diffprivlib has following built-in functionalities:

Mechanisms	Tools	Models	Sensitivity
Laplace	Histogram	Logistic Regression	Built-in function
Gaussian	Count	GaussianNB	float(m, M)
Exponential	Sum	Kmeans	
Staircase	Mean		
Uniform	Std		
	Var		

4.2.1.3 Performance Results

Now we will discuss the performance results of IBM-diffprivlib. As we can see in the table above that IBM-diffprivlib has six built-in tools. We are not considering *Histogram Query* in performance analysis because it behaves the same as *Count query* (both have a sensitivity of 1 for unbounded-Differential Privacy). The graphs for each query, when there is a change in size, std, or skew, are as follows:

Note: The terms, 'std of scaled error' and 'results' refers to privacy.

Count query

Figure 4.1 shows that the epsilon's logarithmic values (ε) are represented along the x-axis, while the y-axis represents the standard deviation of scaled error. We can notice that the std of scaled error (privacy) decreases with an increase in the dataset's size. For a dataset of size 100000, the minimum nose is added to the true answer. We can see the results are not affected by datasets having different values of standard deviation. For all cases the results reaches 0 for epsilon(ε) = 10. After $\varepsilon = 10$ the std of scaled error reaches 0.

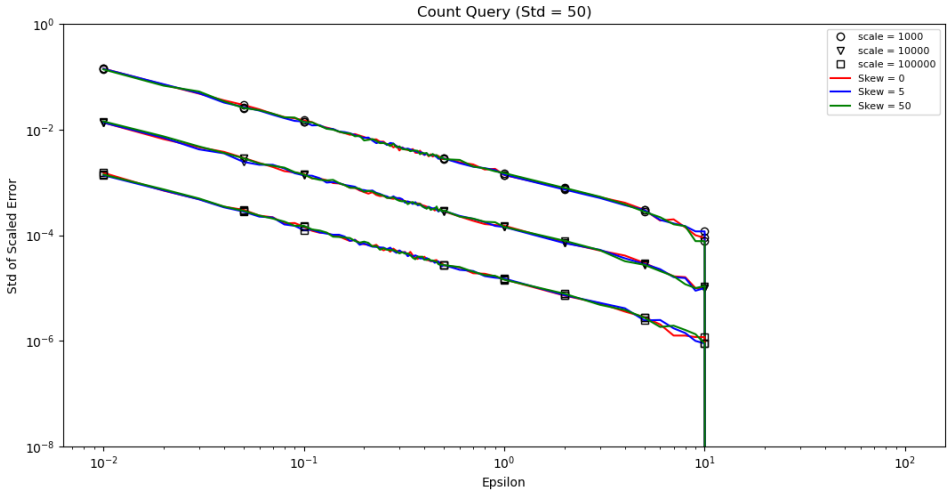

Figure 4.2 shows the behavior of library for *Count query* for datasets = 1000, 10000, 100000, Skew = 0, 5, 50 and std = 50. For datasets with different shapes does not effect the results much. Here the size of datasets it effecting the standard deviation of scaled error.

Figure 4.2 shows that the shape of the dataset does not affect the privacy much. Here again, the size of the dataset shows the prominent difference in the results. Finally, for all datasets, irrespective of their shape, the std of scaled error reaches 0 for epsilon(ε) 10 and onward.

Sum Query

Figure 4.3 shows that the std of scaled error reduces with an increase in epsilon value. This trend was expected because the higher the epsilon value, the lower the added noise, resulting in a lower std of scaled error. We can see the effect on std of scaled error for datasets with different standard deviations. The top line is for a dataset of 1000 rows with std 500 and it shows the std of scaled error is high when comparing it with the same size of the dataset with a lower standard deviation. The sum query results for a dataset of size 1000 with std 50 are very similar to the dataset results of size 10000 with std 500. The std of scaled error reduces more for datasets with many rows and less standard deviation.

Figure 4.4 shows the behavior of *Sum query* for datasets with different shapes. Here the standard deviation in the dataset is 50. We can see that shape of the dataset does not affect the results. The results keep on reducing with an increase in epsilon irrespective of dataset shape. Like previous results, std of scaled error for a big dataset is lower than for small datasets.

Mean Query

Figure 4.5 shows the results for the Mean query. We can see that behavior is a little similar to what we have seen in the Sum query. The dataset results with 1000 rows and 500 std are higher than other results but near the dataset results with 1000 rows and 250 std. The std of scaled error for a dataset with 100000 rows and 50 std is least. All dataset difference between std 250 and 500 is similar and smaller than their difference when std is 50.

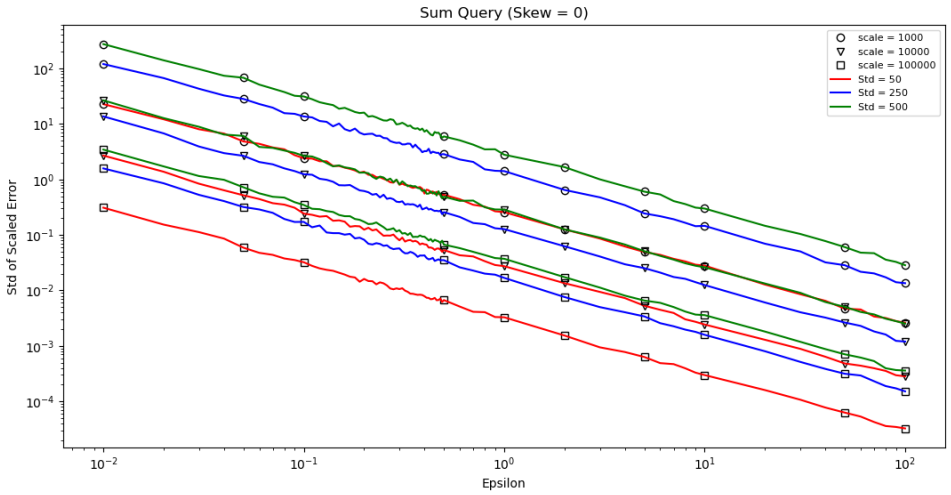

Figure 4.3 shows the behavior of library for *Sum query* for datasets = 1000, 10000, 100000, Std = 5, 250, 500 and skew = 0. The change in standard deviation of datasets effects the results. Less noise is induced in the results of big datasets.

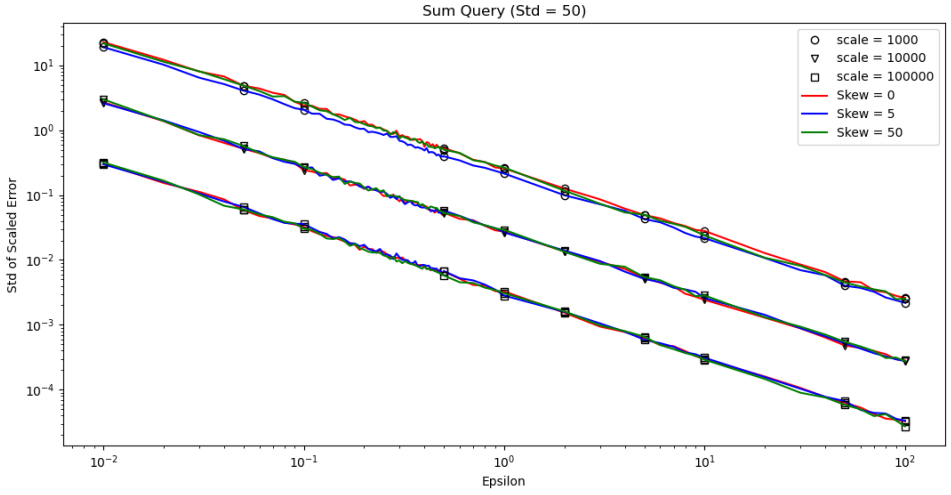

Figure 4.4 shows the behavior of library for *Sum query* for datasets = 1000, 10000, 100000, Skew = 0, 5, 50 and std = 50. For datasets with different shapes does not effect the results much. Here the size of datasets it effecting the standard deviation of scaled error.

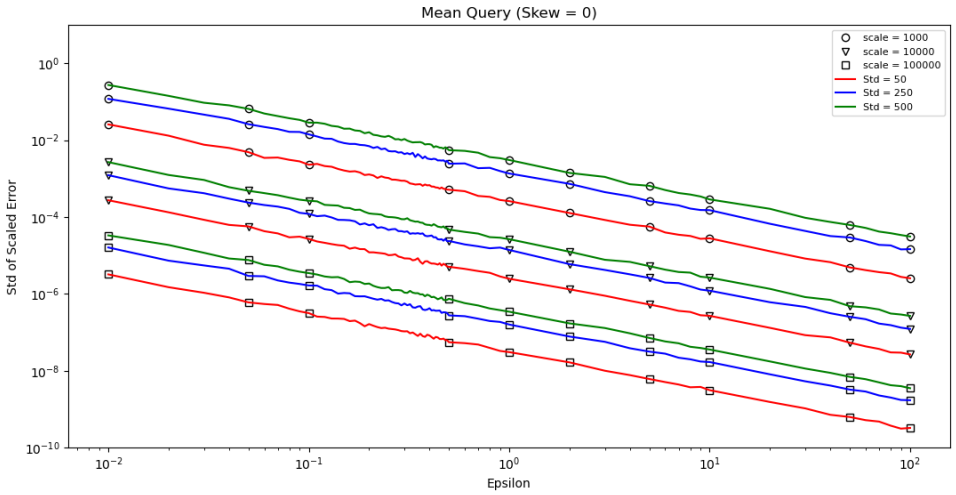

Figure 4.5 shows the behavior of library for *Mean query* for datasets = 1000, 10000, 100000, Std = 5, 250, 500 and skew = 0. The change in standard deviation of datasets effects the results. Less noise is induced in the results of big datasets.

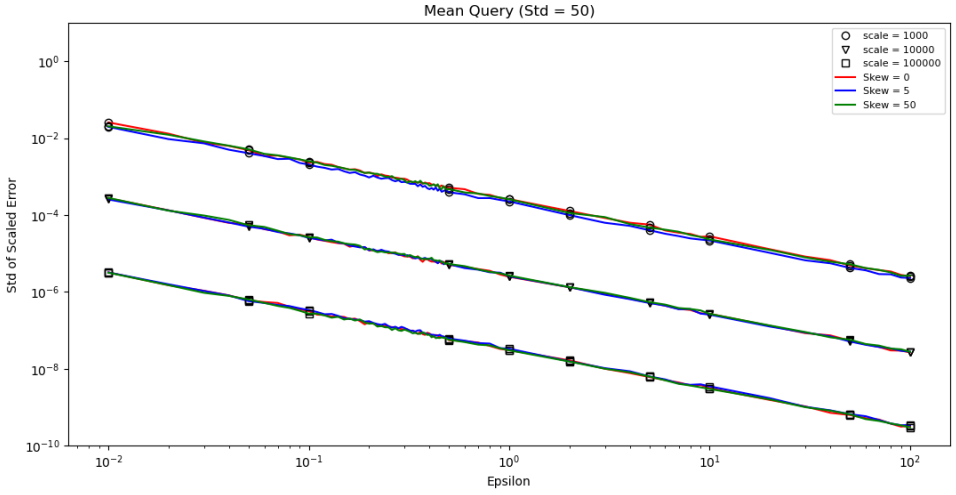

Figure 4.6 shows the behavior of library for *Mean query* for datasets = 1000, 10000, 100000, Skew = 0, 5, 50 and std = 50. For datasets with different shapes does not effect the results. Here the size of datasets it effecting the standard deviation of scaled error.

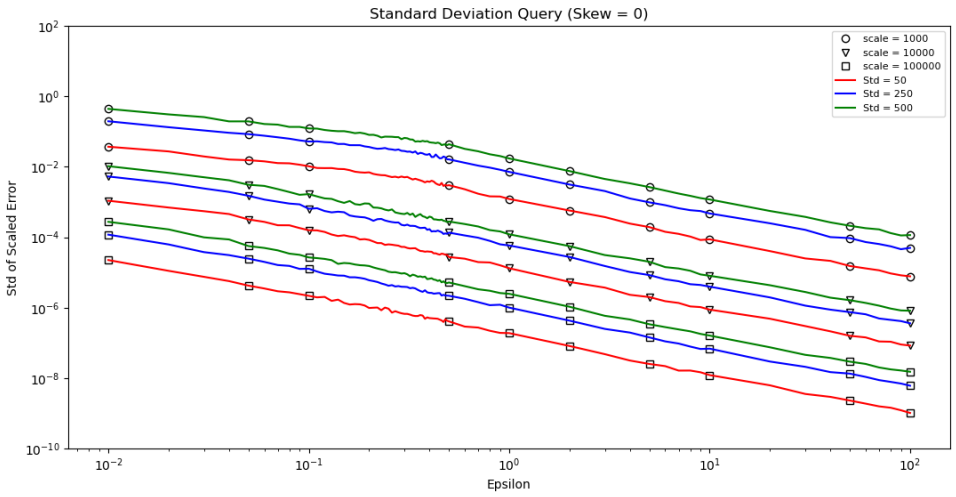

Figure 4.7 shows the behavior of library for *Standard Deviation query* for datasets = 1000, 10000, 100000, Std = 5, 250, 500 and skew = 0. The change in standard deviation of datasets effects the results.

Similar to *Count* and *Sum* query, Figure 4.6 shows that the shape of the dataset does not affect the results. For small datasets (rows 1000), we can see some hindrance, but for more massive datasets, results are strictly following the line, but overall results get smaller with the increase in the dataset size. The decrease in the std of scaled error means that privacy is reducing. For very high value of epsilon(ε) the privacy provided by library get very low and this behavior is theoretically proven.

Standard Deviation Query

Figure 4.7 shows the behavior of the *Standard Deviation query* for datasets of different sizes and std values. The std of scaled error for a dataset with 1000 rows and 500 std is almost equal to 1, but when std is reduced to 50 for the same size of a dataset, the std of scaled error reduces ten times for epsilon(ε) 0.01. Secondly, for a dataset of size 1000, there is a curve between epsilon(ε) 10^-1 and 10^0, which shows that std of scaled error is slightly higher at these values. Thirdly, std of scaled error is comparatively lower for the largest dataset with minimum standard deviation.

Figure 4.8 shows the effect of shape on *Standard Deviation query* results. As we can see, except for small dataset (rows 1000), the results are not affected much due to differences in shape. Unlike results of other queries (*Count, Sum, Mean*) for a dataset with 1000 rows, Standard Deviation shows that for skew = 5, results are slightly lower than for skew = 0 and 50. We don't know the reason for this behavior, but we can conclude that for Standard Deviation, query change in shape can affect the results for small datasets.

Variance Query

Figure 4.9 shows the *Variance query* behavior for different datasets. Though the trend is similar, std of scaled error for smaller datasets is higher than other queries. The dataset results of rows 1000 and std 50 are identical to the size 10000 and std 500. The large dataset with less std has a low value for std of scaled error.

In Figure 4.10, we can see that the behavior of *Variance query* for a dataset of different shapes and sizes are the same as of *Standard Deviation query*. Standard

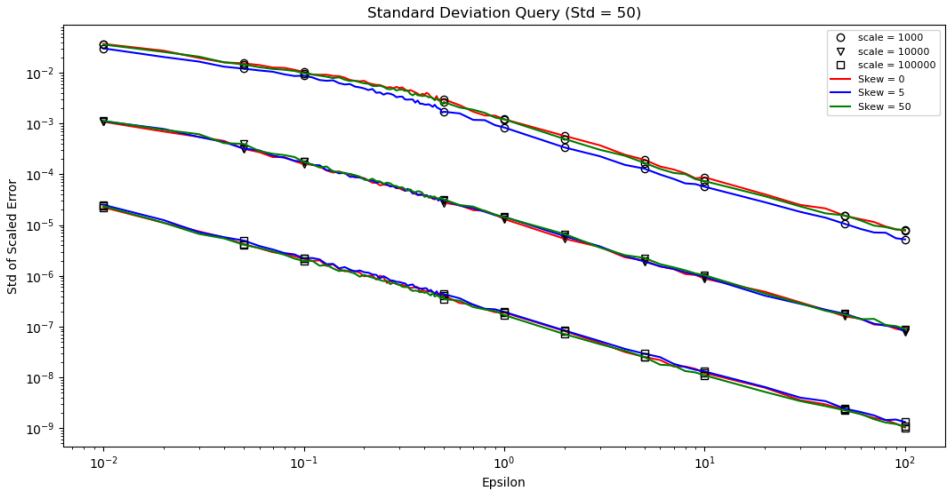

Figure 4.8 shows the behavior of library for *Standard Deviation query* for datasets = 1000, 10000, 100000, Skew = 0, 5, 50 and std = 50. For datasets with different shapes does not effect the results. Here the size of datasets it effecting the standard deviation of scaled error.

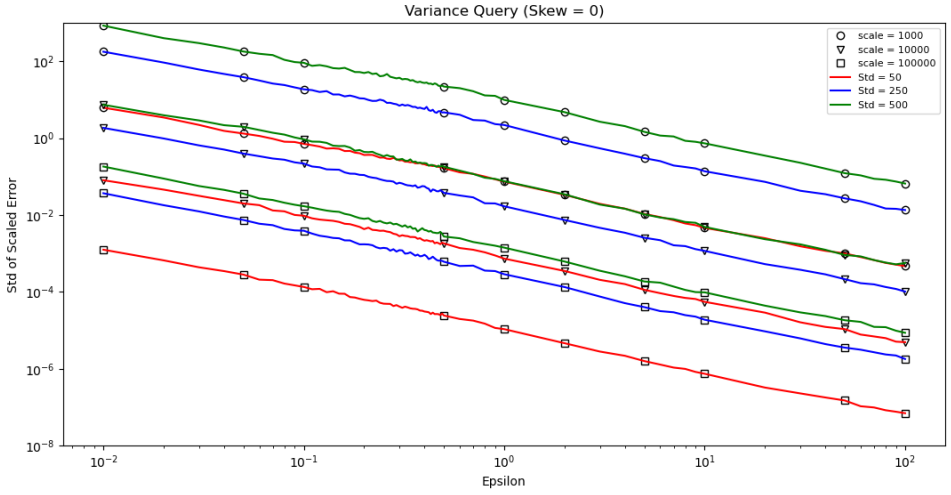

Figure 4.9 shows the behavior of library for *Variance query* for datasets = 1000, 10000, 100000, Std = 5, 250, 500 and skew = 0. The change in standard deviation of datasets effects the results.

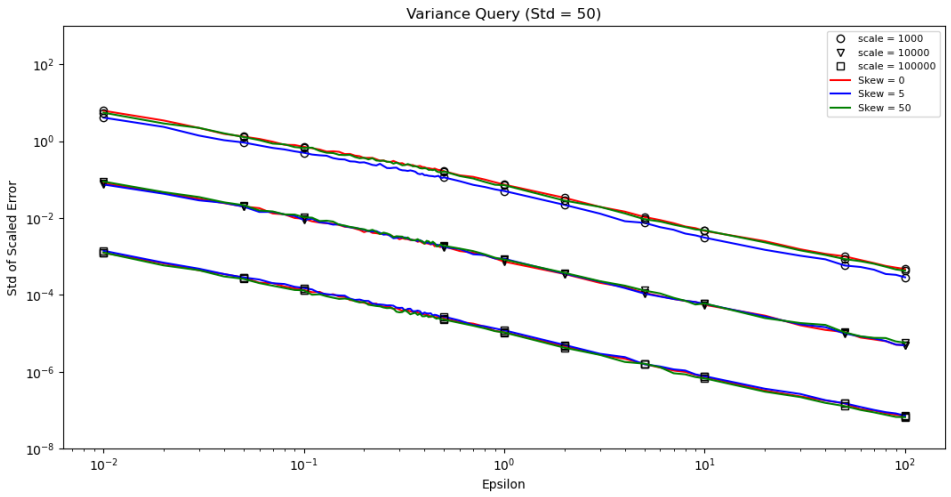

Figure 4.10 shows the behavior of library for *Standard Deviation query* for datasets = 1000, 10000, 100000, Skew = 0, 5, 50 and std = 50. For datasets with different shapes does not effect the results. Here the size of datasets it effecting the std of scaled error.

deviation is a square root of Variance, so the overall std of scaled error will be the same for both queries.

In conclusion, we can say that the Standard Deviation and dataset size are major factors that affect the amount of noise added to the query's answer. At the same time, the shape of the dataset does not have any significant effect on results. From now on, we will only discuss the graphs where we see any unusual behavior of query. If the query results in other libraries are similar to the results of IBM-diffprivlib, we will refer to that Figure. After discussing all libraries and frameworks individually, we will compare them with each other to analyze the performance. Finally, we will see the results of all libraries for our Eco-friendly driving model.

Note: The term 'Results' is referred to as 'Std of Scaled' error. Both times (results and std of scaled error) are used interchangeably throughout the thesis.

4.2.2 OpenDP-Whitenoise Core

4.2.2.1 General Characteristics

Whitenoise core is a new library introduced by *Open Differential Privacy* to bridge the gap between theoretical solutions from the academic community and real-world problems. It provides essential building blocks used by people involved with sensitive datasets, with mature Differential Privacy research. The white paper provides sensitivity proofs for all queries. It focus of global sensitivity for both *Bounded* and *Un-bounded* definitions. Everything is well documented and examples are available to understand the basic features of the library. Though it is majorly developed with *Rust* framework but its *PyPi* package is also available, which make it easy to install, but using it for the first time might cause some issues (e.g., You can not use Pandas library to upload the dataset; instead, you strictly have to follow the pattern provided by Whitenoise-Core itself). So, if one is dealing with multiple datasets, then

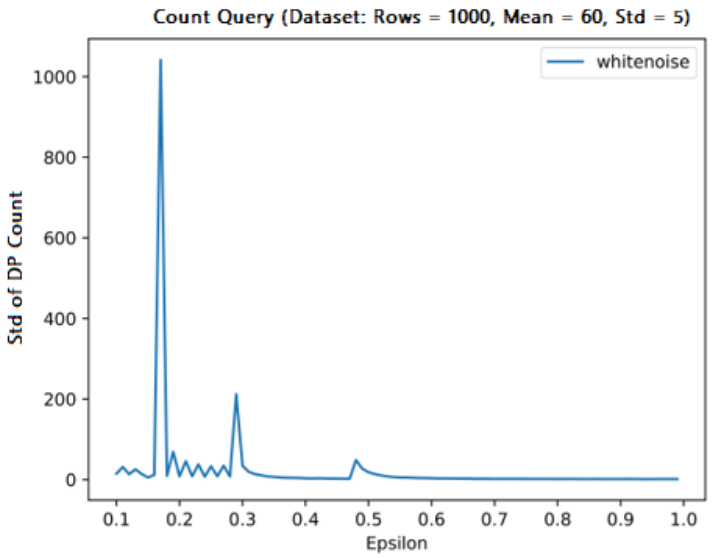

Figure 4.11 Abnormal behavior of Count query for some values of Epsilon (ε)

she has to apply Whitenoise queries separately, which could be hectic in some cases. Due to such behavior, one can say that it is designed for big datasets. Besides this, the *Open Differential Privacy* community is young and they are still trying to fix all bugs.

Whitenoise Core is under *MIT license* and there current version is $v0.2.0$. Recently they added a note in their Git repository that they are planning to introduce a new name for this project in the coming weeks. It currently supports Linux and Mac operating systems.

4.2.2.2 Functionality

Whitenoise focuses on all aspects of Differential Privacy. They encourage you to avoid providing the correct information at some places to maintain privacy, even when you know the dataset. For example, you want to get a Differentially Private sum for *Speed* column in your datasets. It is encouraged to provide the bounds (when applying sum query) as a sensible person, instead of giving the exact values. This action will provide additional privacy of data.

OpenDP-Whitenoise Core has following built-in functionalities:

Mechanisms	Tools	Sensitivity
Laplace	Histogram	Bounded and Un-bounded
Gaussian	Count	Built-in function
Geometric	Sum	float(m, M)
	Mean	
	Quantiles	
	Var/ Co-Var	

4.2.2.3 Performance Results

During our initial analysis of the library, we found abnormal behavior of *counting query*. For some values of ε, we witnessed spikes. Figure 4.1 shows this odd behavior. Later we reported this issue and they fixed it in their new release.

Count Query

Count query for OpenDP-Whitenoise shows similar results that we have seen for IBM-diffprivlib (Figure 4.1). All values are reaching almost 0 for epsilon(ε) 10^0 and onward. Different standard deviation in the dataset does not affect the std of scaled error. The increase in the size of the dataset will result in a lower std of scaled error. As the sensitivity of *Count query* is 1, so the similarity in results was expected.

Sum Query

The *Sum query* results are also similar to the results of IBM-diffprivlib, which tells that both libraries use the same formulas and mechanism for *Sum query*. Furthermore, both libraries are using global sensitivity to induce noise in the answer. OpenDP-Whitenoise has an option for both bounded and un-bounded sensitivity but by default, it calculates results with bounded sensitivity formula. We haven't compared the results for unbounded sensitivity here.

Mean Query

As we seen in *Count* and *Sum query*, results of *Mean query* in OpenDP-whitenoise are almost same as in IBM-diffprivlib. This similar behavior is due to the same approach taken in both projects to calculate the sensitivity. As the formula of bounded and unbounded sensitivity for *Mean query* is the same, results will not be affected at all.

Variance Query

Now we will discuss the results for *Variance query* as OpenDP-Whitenoise does not have *Standard Deviation query*. Again the results are similar to IBM-diffprivlib *Variance query* results. For the largest dataset with minimum standard deviation has the least value for std of scaled error. With an increase in epsilon value (ε), the std of scaled error for all datasets decreases with the same trend.

4.2.3 OpenMined-PyDP

4.2.3.1 General Characteristics

OpenMined-PyDP is the *Python* wrapper for *Google-Differential Privacy (C++)*. It focuses on ε-Differential Privacy algorithms, which can be used to answer statistical queries over numeric datasets containing sensitive information. It uses global sensitivity for only *Bounded* definition. The code is well commented and the repository is well documented. It does not require any pre-requisite libraries to be installed and works fine with other libraries like *Pandas* and *Scikit-learn*. Slack's active community and available examples in the Git repository make this library easy to use user friendly.

It is available under *Apache-2.0 license* and its current version is *v1.0.0*. It currently supports Linux and Mac operating systems. As mentioned in the Git repository, Windows support is also coming soon.

4.2.3.2 Functionality

The parent project *Google-differential Privacy* provides more functionalities as compared to its *Python* wrapper. *Google-Differential Privacy* includes implementation of the Laplace and Gaussian mechanisms while OpenMined-PyDP offers the implementation of the Laplace mechanism only.

OpenMined-PyDP has following built-in functionalities:

Mechanisms	Tools	Sensitivity
Laplace	Histogram	Bounded
	Count	Built-in function
	Sum	int(m, M)
	Mean	
	Std	
	Var	
	Min/Max	
	Median	

4.2.3.3 Performance Results

Count Query

The results for *Count query* are almost similar to the results of IBM-diffprivlib and OpenDP-whitenoise, but here std of scaled error is not reaching 0 for epsilon(ε) 10^0, instead, it is getting 0 slightly after the expected value. Figure 4.11 shows the behavior of *Count query* for OpenMined-PyDP.

Sum Query

Again the results are similar to IBM-diffprivlib (Figure 4.3) and OpenDP-Whitenoise, which means that they used the same formula for sensitivity calculation. So we can say performance wise OpenMined-PyDP, IBM-diffprivlib and OpenDP-Whitenoise are very similar for *Sum query*. For better understanding of their results we recommend to check Section 4.4 where we compared all these libraries.

Mean Query

The *Mean query* for OpenMined-PyDP shows unusual results, as you can see in Figure 4.13. For a dataset with 1000 rows and 500 standard deviations, the std of scaled error dropped abruptly when epsilon(ε) reaches 20. For a similar dataset with standard deviation of 250, this drop happened at 10^1 and for standard deviation of 50 std of scaled error dropped before epsilon(ε) reached 10^1. For a dataset of size 10000 and a standard deviation 500, the drop in std of scaled error values occurred between 10^0 and 10^1, with a decrease in the standard deviation value the fall in happening earlier. There are some unexpected spikes in the dataset results with 10000 and

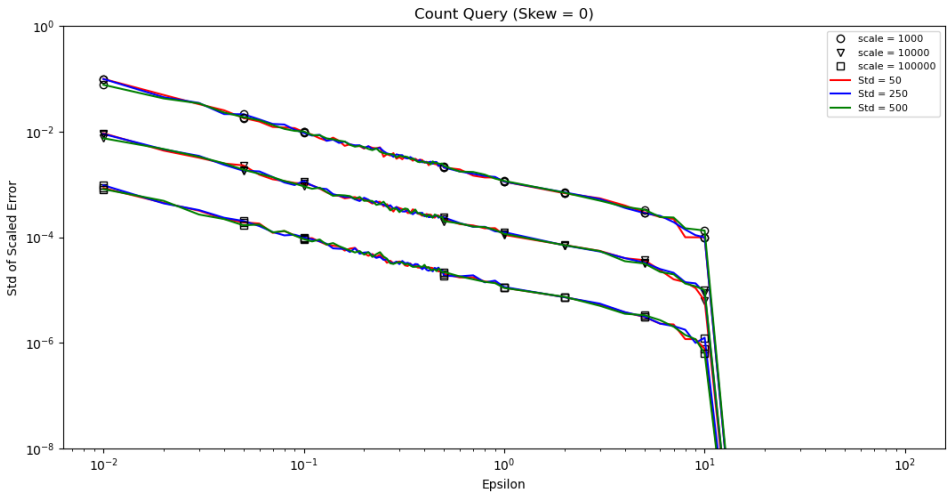

Figure 4.12 shows the behavior of OpenMined-PyDP for *Count query* when datasets = 1000, 10000, 100000, Std = 5, 250, 500 and skew = 0. The change in standard deviation of datasets effects the results.

100000 rows and 50 standard deviations. For 10000 rows, there is one spike after results reach 0, but for 100000 rows dataset, there are multiple spikes. Usually, once the std of scaled error reaches 0, it remains 0 for the rest of the epsilon(ε) values, but here these spikes make the results uncertain and we couldn't find any logical reason for this behavior.

Standard Deviation Query

Figure 4.14 shows the behavior of OpenMined-PyDP for *Standard Deviation query*. Here again, the behavior is unusual as compared to what we have seen for earlier libraries. The starting point for all datasets is similar to IBM-diffprivlib and OpenDP-Whitenoise. Std of scaled error for a dataset with 1000 rows and 50 standard deviations has shown almost no drop between epsilon(ε) 10^{-2} and 10^{-1}. After epsilon(ε) 10^{-1}, it started decreasing, but before reaching 10^{1}, the std of scaled error started rising again. The results for a dataset with 1000 rows and 250 standard deviations dropped drastically just before epsilon(ε) reaches 10^{2}. Similar behavior is shown by a dataset with rows 10000 and 500 std. The overall trend shows that instant drop in scaled error occurs early for larger datasets with smaller standard deviation.

Similarly, unlike IBM-diffprivlib and OpenDP-whitenoise, the shape of the dataset affects the results of *Standard Deviation query* in OpenMined-PyDP. Figure 4.14 shows this behavior. With an increase in skew value, the std of scaled error drop happens early, except for dataset with 10000 rows where the drop for skew 50 happened after the drop for skew 5. For datasets 10000 and 100000, we can also witness the unusual spikes in results.

Variance Query

Figure 4.15 shows the results of *Variance query* for datasets = 1000, 10000, 100000, Std = 50, 250, 500 and Skew = 0. We can see two unusual behaviors here. First, for dataset 10000 with std 50, the std of scaled error decreased gradually, but before reaching epsilon(ε) = 10^{2}, it started rising again. Second, for 100000 rows and 50 std

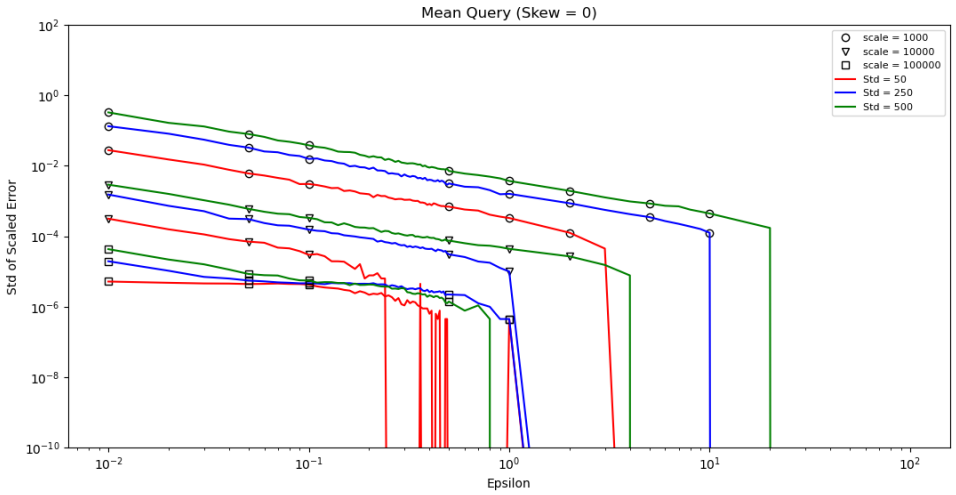

Figure 4.13 shows the behavior of OpenMined-PyDP for *Count query* when datasets = 1000, 10000, 100000, Std = 5, 250, 500 and skew = 0.

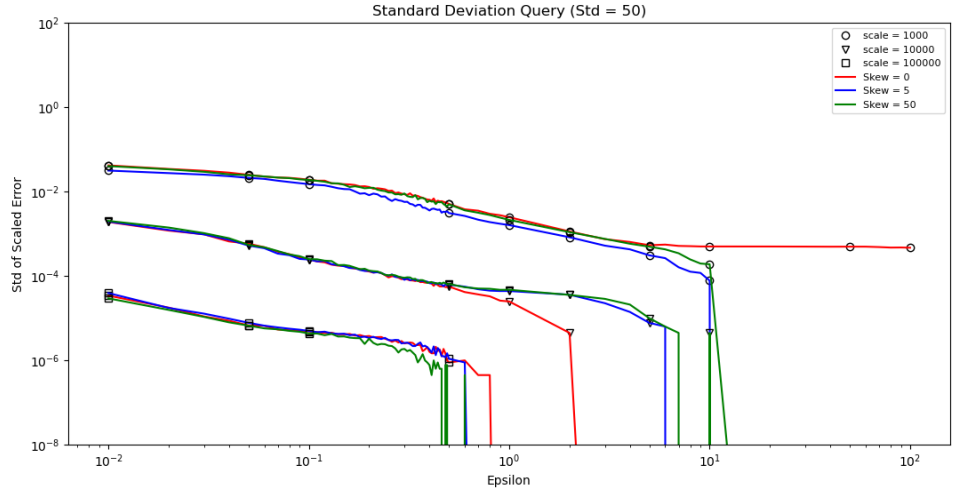

Figure 4.14 shows the behavior of OpenMined-PyDP for *Standard Deviation query* for datasets = 1000, 10000, 100000, Skew = 0, 5, 50 and std = 50. For datasets with different shapes does effect the results.

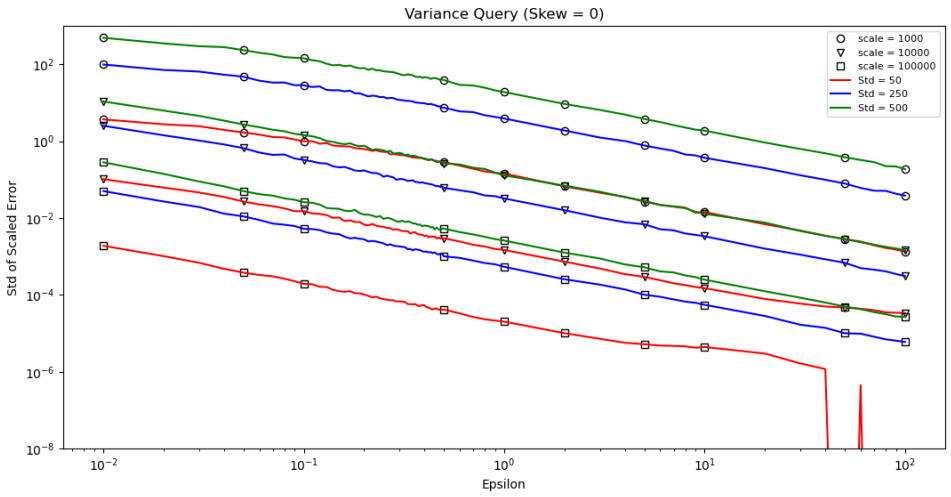

Figure 4.15 shows the behavior of OpenMined-PyDP for *Standard Deviation query* for datasets = 1000, 10000, 100000, Std = 50, 250, 500 and Skew = 0. For large dataset, results are unexpected. Instead of gradual decrease in results for dataset with 100000 rows and 50 std, it shows sudden drop before epsilon(ε) reaches 10^2.

dataset, we witnessed slight rise and an abrupt drop in results between epsilon(ε) 10^1 and 10^2.

4.2.4 OpenMined-PySyft

4.2.4.1 General Characteristics

PySyft is a private Deep Learning library. It uses Federated Learning, Encrypted Computation and Differential privacy within the Deep Learning framework like TensorFlow and PyTorch. Though the library's major focus is Federated Learning, it also uses Differential Privacy to keep the results private. Our focus is to analyze this Differential Privacy feature. It can only be used for global sensitivity for both *bounded* and *Un-bounded* definitions. The programming language used for this library is *Python*. PySyft has a large active community, but its main focus is on Deep Learning algorithms and talks very less regarding Differential Privacy. They also have a complete course on Udacity[50].

It is available under *Apachi-2.0 license* and it current version is *v0.2.9*. PySyft requires some pre-requisite libraries like TensorFlow and PyTorch. It works fine with only *Python 3.6, 3.7*. The library is compatible with all operating systems.

4.2.4.2 Functionality

Though Differential Privacy is claimed as one of the main features of PySyft, it does not has any built-in function for Differential Privacy. Instead, it uses Numpy Laplace or Gaussian function to add Laplacian or Gaussian noise to the answer. Furthermore, sensitivity has to be calculated manually for all queries, so its user prefers to use bounded sensitivity or un-bounded sensitivity.

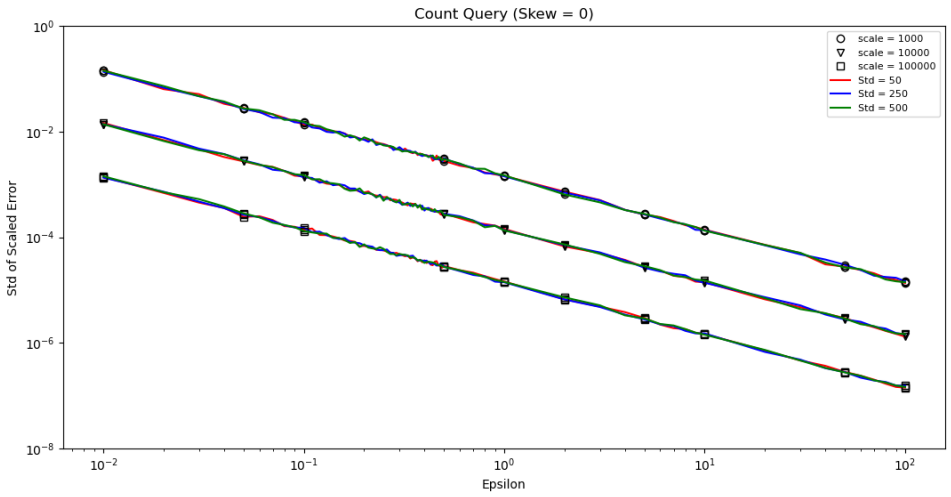

Figure 4.16 shows the behavior of library for *Standard Deviation query* for datasets = 1000, 10000, 100000, Std = 5, 250, 500 and skew = 0. The change in standard deviation of datasets effects the results. Unlike *Count query* results for other libraries, here results are not reaching 0 for epsilon(ε) 10^1 onward.

PySyft has following functionalities:

Mechanisms*	Tools*	Sensitivity
Laplace	Histogram	Manual or
Gaussian	Count	sensitivity sampler
	Sum	
	Mean	
	Std	
	Var	

* These features are supported but don't have a built-in function in the library.

4.2.4.3 Performance Results

Count Query

Figure 4.16 shows the *Count query* results for OpenMined-PySyft. As mentioned in the library's general characteristics, it uses the NumPy Laplace function to add noise in the true answer. That is why we can see that its behavior is different from other libraries' behavior for *Count query*. Here, the scaled error is not reaching 0 even when epsilon(ε) reaches 10^2. But similar to other libraries, standard deviation in datasets does not affect the results at all. For larger datasets, the std of scaled error is lower than smaller datasets. The shape of the dataset does not affect the results here too.

Sum Query

The results for *Sum query* are similar to that of IBM-diffprivlib and OpenDP-Whitenoise. The change in results due to the standard deviation in datasets of different sizes also matches other libraries' results under the same conditions.

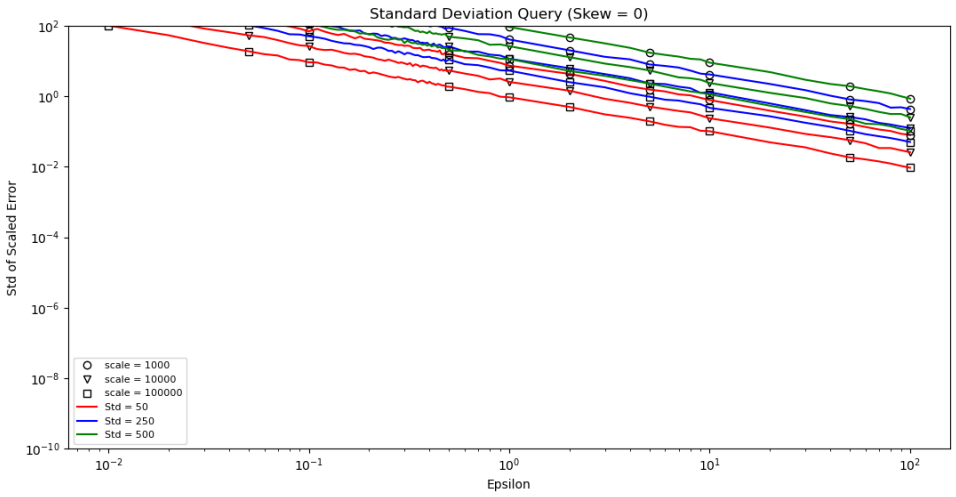

Figure 4.17 shows the behavior of library for *Standard Deviation query* for datasets = 1000, 10000, 100000, Std = 5, 250, 500 and skew = 0. Overall std of scaled error is much higher if it is compared with results of other libraries under same condition.

Mean Query

Mean query results for OpenMined-PySyft are also similar to the results of other libraries that we have discussed earlier. We can conclude that using NumPy Laplace functions is the same as the results for Laplace functions used in other libraries. See Figure 4.5 to find a detailed discussion regarding results.

Standard Deviation Query

Figure 4.17 shows the std of scaled error for *Standard Deviation query*. We can see that results are much higher, much higher than other libraries' results under the same condition. In IBM-diffprivlib, OpenDP-whitenoise and OpenMined-PyDP, the std of scaled error (y-axis) starts from 10^0 for small datasets and 10^-4 for large dataset. But here it results are starting well before 10^2 and ending at 10^-2. Although the results' trend is similar to previous results with the same conditions, the whole graph shifted upwards. Once again, the shape of the dataset does not affect overall results.

Variance Query

Figure 4.17 shows the results for *Variance query*. It is noticed that the std of scaled error is very high and the trend is also different than what we have seen in previous *Variance query* results. Here, to analyze the graph properly, we changed the y-scale. As we can see, the standard deviation in the dataset has no significant effect on results. The difference between std of scaled error for a dataset with 100000 rows and 10000 rows is comparatively higher than the difference between results with 1000 rows and 10000 rows. In conclusion, we can say that the NumPy Laplace function induce very high noise for *Variance query*.

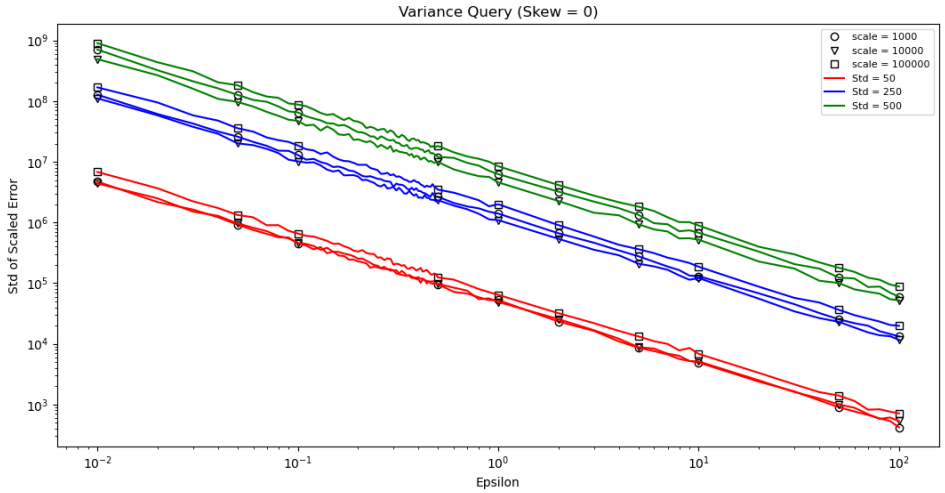

Figure 4.18 shows the behavior of library for *Variance query* for datasets = 1000, 10000, 100000, Std = 5, 250, 500 and skew = 0. The results are much higher if it is compared with results of other libraries under same condition. Due to unusual results the scale on y-axis is not same as it was in previous libraries.

4.2.5 CHORUS-DP

4.2.5.1 General Characteristics

CHORUS is a framework developed for building scalable Differential Privacy mechanisms. It is based on cooperation between mechanism and high-performance database management system (DBMS). It supports scalable implementation by leveraging the database management system for data processing tasks instead of custom code. This is known as cooperative architecture for differential privacy mechanisms[28]. It uses global sensitivity for only *Bounded* definition. As it take data from the connected databases so it is hard to evaluate it for multiple datasets. In our case we had to run the code manually for each dataset stored in our MySQL database. The complete framework is written in *Scala* language and built using *Maven*. It has a small but active community. Its Git repository provides basic examples, but overall code is not well commented. During our analysis, we found a bug in the Average mechanism where bounds were hard-coded[28].

It is available under *MIT license* and has only one release so far. It supports Linux and Mac operating systems.

4.2.5.2 Functionality

CHORUS has three major components: *rewriting, analysis* and *post-processing*. *Rewriting* is used to modify queries to perform functions like clipping, *analysis* is used to determine the amount of noise to be added to make results Differentially Private and *post-processing*, to process the results of executing queries.

CHORUS-DP has following built-in functionalities:

Mechanisms	Tools	Sensitivity
Laplace	Histogram	Bounded
Exponential	Count	Built-in function
Sparse Vector Mechanism	Sum	float(m, M)
	Mean	

4.2.5.3 Performance Results

Count Query

The results for *Count query* are similar to the results of other libraries. The strange thing is that CHORUS required lower and upper bounds even for *Count query* to calculate the sensitivity when it is known that sensitivity for *Count query* is 1. The results show that the required parameters to run the query are not used while adding noise to the total count.

Sum query

The overall trend of the results is same as the results for other libraries. CHORUS provides little bit more privacy as compared to IBM-diffprivlib and OpenDP-whitenoise for all sizes of dataset. For more information regarding how much more privacy is provided by CHORUS, we refer to the Section 4.4 where we compared the results of different libraries.

Mean Query

The trend of results for *Mean query* is similar to the *Sum query* results but here results are more higher than other results. This is due to insertion of Laplacian noise two times, one for *Count query* and other for *Sum quesry*. Due to the similarity in trend of results we recommend to see the discussion for other libraries.

4.2.6 diffpriv

4.2.6.1 General Characteristics

diffpriv is an *R* package that provides tools for statistical analysis and machine learning under Differential Privacy. This library follows the basic principles of Differential Privacy, proved by Dwork[16]. It can only be used for global sensitivity for both *Bounded* and *Un-bounded* definitions. Regardless of having a tiny project community, the library's simple architecture and documentation make it easy to use. diffpriv does not require any pre-requisite libraries.

It is available under *MIT license* and its current version is *v0.4.2*. It is compatible with all operating systems.

4.2.6.2 Functionality

Unlike other Differential Privacy libraries, for statistical analysis in diffpriv, one has to calculate sensitivity, which then can be used in built-in mechanisms to release Differentially Private results. So, it is the user's choice to include bounded sensitivity or un-bounded sensitivity. diffpriv also provide *sensitivity sampler* to estimate the target sensitivity by repeated calculation of sensitivity on random neighboring database pairs.

diffpriv has following functionalities:

Mechanisms	Tools*	Sensitivity
Laplace	Histogram	Manual or
Gaussian	Count	sensitivity sampler
Exponential	Sum	
	Mean	
	Std	
	Var	

* These features are supported but don't have a built-in function in the library.

4.2.6.3 Performance Results

Count Query

The results for *Count query* are similar to the results of other libraries. Due to the same sensitivity of count query all results are the same. To see the discussion on results, one can refer to Figure 4.16.

Sum Query

Figure 4.19 shows the behavior of diffpriv for *Sum query* for datasets = 1000, 10000, 100000, Std = 5, 250, 500 and skew = 0. The results are higher than the results of other libraries under the same condition. Generally, std of scaled error for larger datasets is less than std of scaled error for smaller datasets, but it's opposite. The results for a dataset with 100000 rows and 500 standard deviation shows the highest value. With the decrease in the dataset's standard deviation value, the results' overall value decreases for the largest dataset. A dataset of 1000 rows with 50 std and 100000 rows with 50 std are the same. In contrast, results for the 10000 rows dataset with 50 std and 1000 rows with 250 std are similar. The std of scaled error is lowest for a dataset with 10000 rows and 250 std. As sensitivity is calculated manually in diffpriv, these unexpected results could be due to the built-in Laplace function. This behavior can be analyzed better when we compare different libraries at the end of this section.

Mean Query

The results for *Mean query* are also interesting to analyze. Figure 4.20 shows these results. Std of scaled error for the largest dataset is higher, which is the opposite of the library's expected behavior. There is less difference between results for a dataset

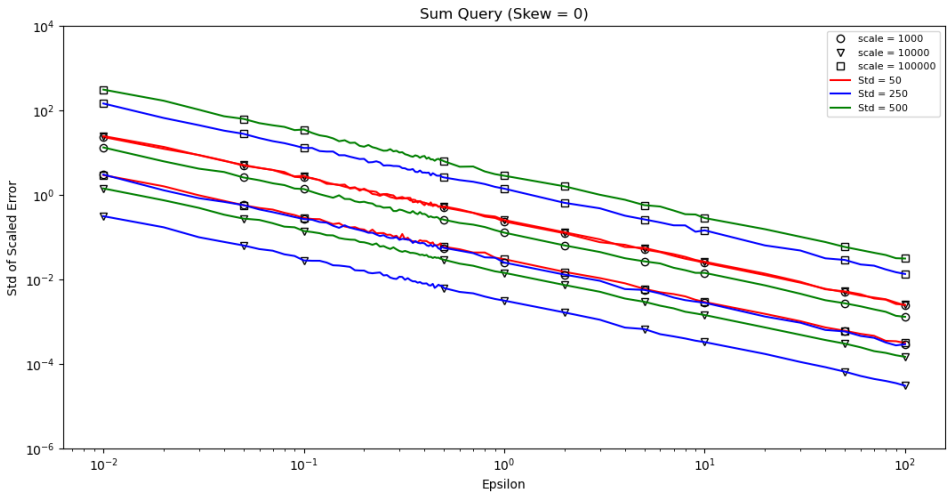

Figure 4.19 shows the behavior of diffpriv for *Sum query* for datasets = 1000, 10000, 100000, Std = 5, 250, 500 and skew = 0. The results are much higher if it is compared with results of other libraries under same condition. Generally std of scaled error for larger datasets is less than std of scaled error for smaller datasets but here its opposite.

of 100000 rows with 500 std and 250 std. But interestingly, std of scaled error for a dataset of 100000 rows with 50 std is very low. The smallest dataset results with 500 std are higher than the results of the same dataset with 250 std. Similar is the behavior for a dataset of 10000 rows with 50, 250 and 500 std. The lowest results are for the dataset of rows 10000 and std 250.

Standard Deviation Query

The overall trend of results for *Standard Deviation query* is similar to the results of *Mean query*. Here the whole graph is shifted a little upwards with a starting point between 10^2 and 10^0 on the y-axis (std of scaled error) for a dataset of 100000 rows and 500 std. Again, std of scaled error for dataset with 10000 rows and 250 std is lowest.

Variance Query

In Figure 4.21, we can see the diffpriv results for *Variance query*. The general behavior is similar to *Mean* and *Standard Deviation query* behavior, but the noticeable part is with an increase in the size of the dataset results in low std of scaled error, but if the size of the dataset exceeds the limit, than more noise is added to the true answers, which leads to higher std of scaled error.

4.2.7 Private Integrated Query (PINQ)

4.2.7.1 General Characteristics

Private Integrated Query (PINQ)[38] is similar to a LINQ-like API for analyzing and computing sensitive datasets while providing guarantees of Differential Privacy. It is a programming language as well as an execution platform in which expressible programs satisfy Differential Privacy. It is designed so that both analysts and data

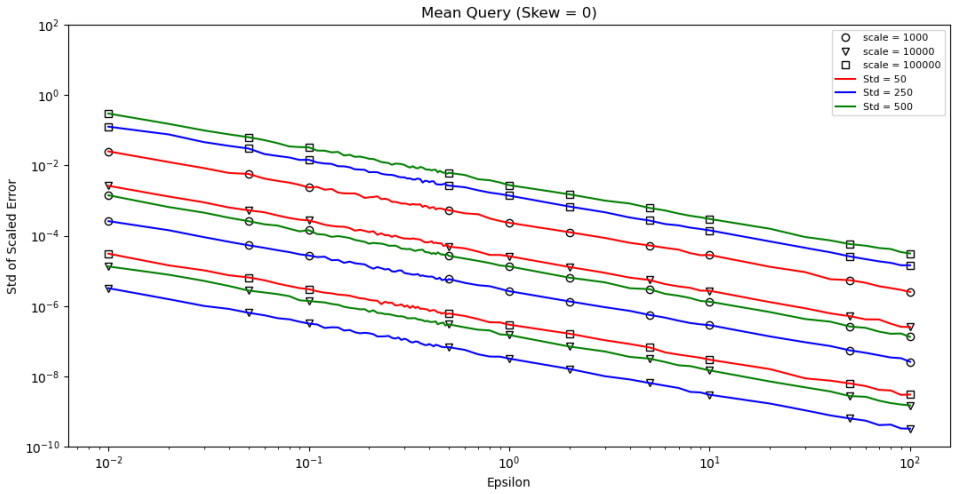

Figure 4.20 shows the behavior of diffpriv for *Mean query* for datasets = 1000, 10000, 100000, Std = 5, 250, 500 and skew = 0. Here again the results are deviating from expected results. For largest dataset (100000 rows) more noise is added as compare to dataset with 10000 rows.

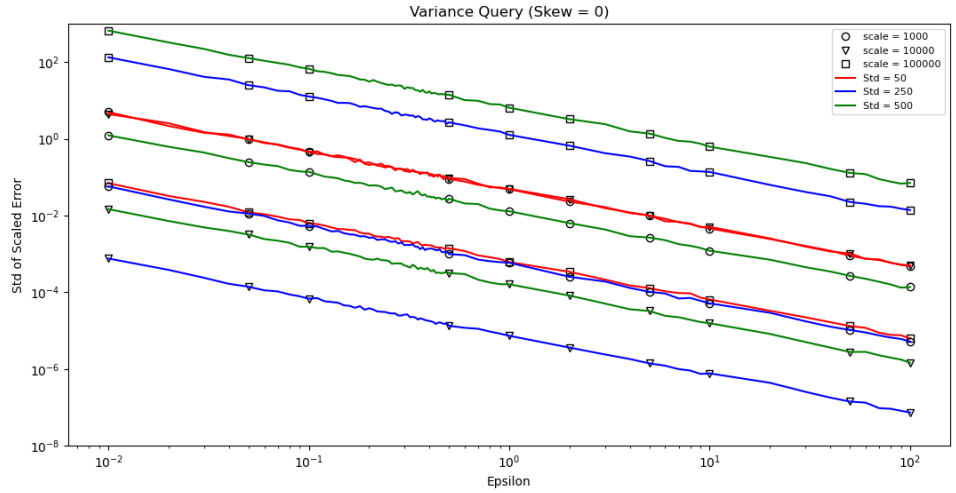

Figure 4.21 shows the behavior of diffpriv for *Variance query* for datasets = 1000, 10000, 100000, Std = 5, 250, 500 and skew = 0. Less noise is added to the results of dataset with 10000 rows as compare to the results of dataset with 100000 rows.

providers can start using PINQ with any complicated structure and without any prior data privacy training. The complete language is written in *C#*. The documentation of this project is not well maintained and only the PINQ prototype is available for download.

It is available under *MIT license* and has only one version (*v0.1*). It is compatible with all operating systems.

4.2.7.2 Functionality

PINQ is designed in such a way that non-experts can also use it. That is why it requires minimum parameters from the user (query and Epsilon(ε) value). It does not demand any parameters to calculate sensitivity. Instead, it already has hardcoded formulas for sensitivity. You can also create your query to get Differentially Private results with PINQ.

PINQ has following built-in functionalities:

Mechanisms	Tools	Models
Laplace	Count	K-means,
Exponential	Sum	Perceptron classification,
	Mean	Contingency table
	Median	measurement

4.2.7.3 Performance Results

Count Query

The results of *Count query* for PINQ is similar to the results of other libraries. As sensitivity for *Count query* is 1, so results of all libraries are almost the same. The std of scaled error is inversely proportional to epsilon(ε). When epsilon(ε) reaches 10^2 the results for largest dataset reaches 10^{-7}.

Sum Query

PINQ does not ask for bounds to calculate sensitivity; instead, sensitivity is hard-coded in the library. For *Sum query*, sensitivity is equal to 1.0, so the results are according to this fixed sensitivity. Figure 4.22 shows that results for *Sum query* are very similar to *Count query* results. Because the library is tested on a dataset having a column of 1 and 0, the sensitivity of *Sum query* ($Max - min$) will be equal to 1. In real-world datasets (columns of datasets are not 1's and 0's), the sensitivity will not be equal to 1, but it will be $Max - min$.

Mean Query

Sensitivity for *Mean query* in PINQ is equal to 2. This value is fixed and can't be changed while applying a query to the dataset. That is why its results are not comparable with *Mean query* results of other libraries. In our final comparison graphs, to analyze all libraries and frameworks properly, we will not compare PINQ *Mean query* results with different results.

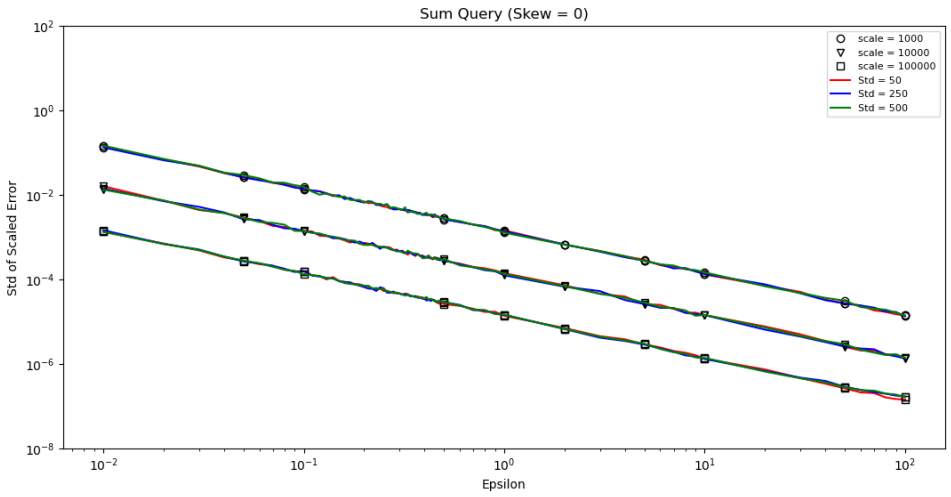

Figure 4.22 shows the behavior of PINQ for *Sum query* with datasets = 1000, 10000, 100000, Std = 5, 250, 500 and skew = 0. As the sensitivity is 1.0 so results are very similar to *Count query* results.

4.2.8 GUPT

4.2.8.1 General Characteristics

GUPT is a privacy-preserving data mining platform that guarantees Differentially private results. This library focuses on global sensitivity (add noise to the query results). The architecture is straight forward: Data owner provides data and privacy budget, while the data analyst query this dataset via web front-end. The computation manager executes these queries and returns Differentially private results to data analyst. It required some pre-requisite libraries like *Numpy, Scipy* and *Scikit-learn*. GUPT is written in *Python* and works only with *Python 2*. As *Python 2* is obsolete, we can say that era of GUPT is ended until a new update is released. The Git repository of GUPT is not updated for nine years, and it does not have an active community.

4.2.8.2 Functionality

This project's significant contribution was distributing the privacy budget, describing the privacy budget in terms of accuracy and maintaining the utility while providing enough privacy. This project is obsolete, so we did not investigate it further.

4.2.9 Tensorflow-Privacy

4.2.9.1 General Characteristics

TensorFlow-Privacy is *Python* library that includes TensorFlow optimizers' implementation for training machine learning models with Differential Privacy. With a

large and active community of TensorFlow, this project is under continual development. To get started with TensorFlow-Privacy, there are some pre-requisites like TensorFlow(>=1.14), scipy (>=0.17) and mpmath. It focuses on (ε, δ)-Differential Privacy and l_2-sensitivity.

It is available under *Apachi-2.0 license* and current version is *v0.5.1*.

4.2.9.2 Functionality

It focuses on (ε, δ)-Differential Privacy for Gaussian mechanism and l_2-sensitivity. It is designed specifically for the machine learning model and can't be used for single queries. There are two major reasons for not conducting a performance analysis of TensorFlow-Privacy. **1.**Throughout in this study, we consider the Laplace mechanism (ε-Differential Privacy), but TensorFlow has only the Gaussian mechanism. **2.**We can't use the TensorFlow optimizer for single queries like *Count, Sum, Mean, etc*, and they are designed only for machine learning models.

4.2.10 MIT-PrivateMultiplicativeWeights

4.2.10.1 General Characteristics

PrivateMultiplicativeWeights (MWEM) is a simple algorithm for Differentially Private data release. It is written entirely in *Julia* programming language. Differentially Private synthetic data preserving, optimized in-memory implementation for a small number of data attributes, a scalable heuristic for a large number of data attributes and an easy-to-use interface for data representation are some of its main features. It does not have an active community and developers provide few examples. The Git repository is not updated for the last two years.

It is available under *MIT Expat license* and its current version is *v0.1.1*. It is compatible with all operating systems.

4.2.10.2 Functionality

MWEM focuses mostly on Histogram queries and their representations. In the Git repository, there are only examples of Histograms in different scenarios. Due to the lack of built-in functions, documentation and examples, we can not use it for our single query analysis. It only deals with the Laplace mechanism.

4.3 Non-Functionality Comparison

This section gives an overview of the non-technical aspects of considered libraries and frameworks.

As we can see in Figure 4.23, that most of the projects are currently active and recently released new version. But there is one library (GUPT) which is not maintained or updated for several years. Moreover, some libraries (TensorFlow-Privacy

Evaluation Parameters	1) IBM-diffprivlib	2) openDP - White Noise	3) Google PyDP	4) Google Pysyft	5) CHORUS - DP	6) diffpriv	7) PINQ	8) GUPT	9) TensorFlow DP	10) MIT-PrivateMultiWeight
Programming Language	Pyhton	Rust	Pyhton	Pyhton	Scala	R	C#	Pyhton	Pyhton	Julia
Current Version	v0.3.0	v0.2.0	v1.0.0	v0.2.9	N/A	v0.4.2	v0.1	N/A	v0.5.1	v0.1.1
Release Date/ Last update	26-Jan-20	25-Aug-20	2-Sep-20	15-Sep-20	19-Aug-20	18-Jul-17	27-Jun-19	11-Jul-12	3-Sep-20	4-Dec-18
License	MIT	MIT	Apache-2.0	Apache-2.0	MIT	MIT	MIT	ACM Digital Library	Apache-2.0	MIT Expat
Operating System	Windows, Linux & Mac	Linux & Mac	Linux & Mac	Windows, Linux & Mac	Linux & Mac	Windows, Linux & Mac	Windows, Linux & Mac	Windows, Linux & Mac	Windows, Linux & Mac	Windows, Linux & Mac

Figure 4.23 shows these general information like major programming language, current version of library or framework, release data, license and what type of system it is compatible with.

and MIT-PrivacyMultiplicativeWeight) are not included in performance analysis because they are designed for different purposes and can not be used for our scenarios.

4.4 Findings

In this section, we will compare the results of the libraries mentioned above under the same conditions. Moreover, we will also see the results of each library for our Eco-Friendly Driving Model. Finally, we will discuss the overall findings of our analysis.

So far, we have seen that different libraries have different builtin tools and mechanisms. Before comparing the libraries and frameworks, let's look at the possible logical comparison.

Sr. No.	Libraries & Frameworks	**Count**	**Sum**	**Mean**	**Std**	**Var**
1	IBM-diffprivlib	Yes	Yes	Yes	Yes	Yes
2	OpenDP-Whitenoise	Yes	Yes	Yes	No	Yes
3	OpenMined-PySyft[1]	Yes	Yes	Yes	Yes	Yes
4	OpenMined-PyDP	Yes	Yes	Yes	Yes	Yes
5	PINQ[2]	Yes	Yes	Yes	No	No
6	CHORUS	Yes	Yes	Yes	No	No
7	diffpriv[1]	Yes	Yes	Yes	Yes	Yes

[1] Here, sensitivity is calculated manually. In OpenMined-PySyft, the NumPy Laplace function is used for adding Laplacian noise to the actual answer, while diffpriv has a built-in Laplace function for this purpose.

[2] Here, the sensitivity is fixed and the user is not allowed to provide bounds for sensitivity calculations.

Now we will compare the results of libraries and frameworks according to the above chart. This comparison will help us to decide which library is behaving better than others under certain conditions. In the individual analysis, we witnessed that shape

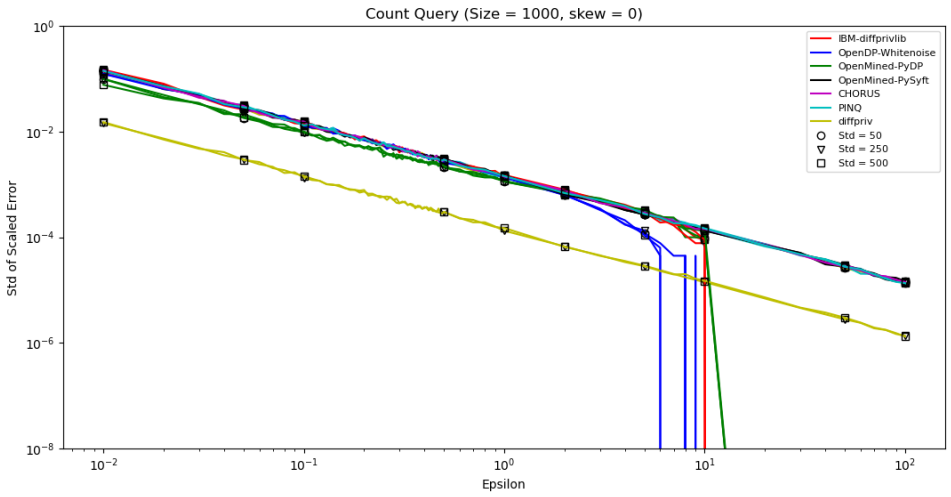

Figure 4.24 shows the results of *Count query* for all libraries and frameworks. The size and shape of dataset is 1000 and 0 while values of standard deviation in dataset are 50,250 and 500, represented by different shapes.

of the dataset den not affect the std of scaled error, so now we will analyze all libraries for the difference in standard deviation values and size of the datasets.

Count Query

Figure 4.24 shows the results of *Count query* for all libraries and frameworks. The shapes are representing different values of standard deviation in the dataset white line colors representing each library. The size of the dataset is 1000, with 0 skews. With an increase in the value of epsilon(ε), privacy decreases gradually. The results show that the standard deviation values of the dataset do not have much effect except for diffpriv library. For standard deviation = 50 in a dataset, the results are similar to other libraries' results. Still, for standard deviation = 250 and 500 in dataset, the overall std of scaled is lower than expected. Another noticeable behavior is when the epsilon(ε) value reaches or about to reach 10^1; some library's results show a sudden drop. For OpenDP-Whitenoise, this drop started right after epsilon(ε) 10^0. Besides this, after a drastic drop, it shows one unexpected spike just before epsilon(ε) 10^1. For IBM-diffprivlib, this drop happened exactly at 10^1 irrespective of the standard deviation value in the dataset. For OpenMined-PyDP, this fall happens a little bit after 10^1.

In Figure 4.25, we see the std of scaled error for different sizes of the dataset. Different shapes represent the size of the dataset, while other color lines represent libraries. The overall trend was as expected, with high epsilon(ε) value; less noise is added, which results in a lower std of scaled error. But there are four observable behaviors. First, the results for OpenMined-PyDP are always slightly lower than other results. Second, when epsilon(ε) value reaches or about to reach 10^1 three out of seven libraries shows a sudden drop, third, the drop for OpenDP-Whitenoise started right after epsilon(ε) 10^0, forth, irrespective of size, OpenMined-PyDP drops a little bit after epsilon(ε) 10^1. This unexpected behavior is possibly due to the programming language they are written in. OpenMined-PyDP, OpenDP-Whitenoise and IBM-diffprivlib are written in Python language. Python's extremely low values round them off to 0; after a certain amount, we see an abrupt drop.

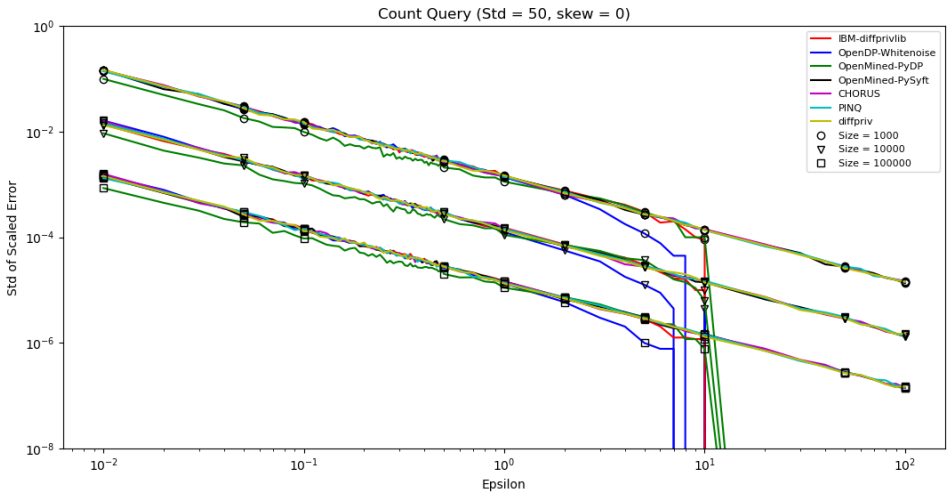

Figure 4.25 shows the results of *Count query* for all libraries and frameworks. The std and shape of dataset is 50 and 0 while size of dataset varies from 1000 to 100000, represented by different shapes.

Sum Query

Figure 4.26 shows the results of *Sum query* for all libraries when the dataset is 1000 rows, 0 skew and standard deviation 50, 250, 500. PINQ results show that std of scale is not affected by the difference in standard deviation in datasets. Moreover, std of scaled error is far less than the other results because sensitivity for *Sum query* in PINQ is 1. diffpriv results shows that for 500 standard deviations in the dataset, std of scaled is higher than when a standard deviation in a dataset is 250. The results of IBM-diffprivlib, OpenDP-Whitenoise, OpenMined-PySyft and OpenMined-PyDP are precisely the same for each value of the standard deviation dataset. CHORUS shows the same trend, but for each value of standard deviation in the dataset, std of scaled error is slightly higher than other results.

In Figure 4.27, we can get the response of all libraries for different dataset sizes. Except for diffpriv, all libraries and framework show that with an increase in the datasets' size, it results in a lower std of scaled error for all epsilon(ε). In diffpriv, results for a dataset with 1000 rows and 10000 rows are similar, while for a dataset with 100000, yields are lower. As sensitivity is 1 for PINQ, the results are lower than the results of other libraries. CHORUS results are always slightly higher than the results of IBM-diffprivlib, OpenMined-PySyft and OpenMined-PyDP.

Mean Query

Figure 4.28 shows the results of libraries and framework for *Mean query*. For most the libraries, results are according to our expectations. We found abnormality in the results of CHORUS, diffpriv and OpenMined-PyDp. CHORUS results are higher because it added noise two times (one time for *Count query* and one time for *Sum query*) to the actual answers. In diffpriv, for sizes of datasets, privacy is lower than expected while in OpenMined-PyDP we saw a sudden drop in all cases.

Standard Deviation Query

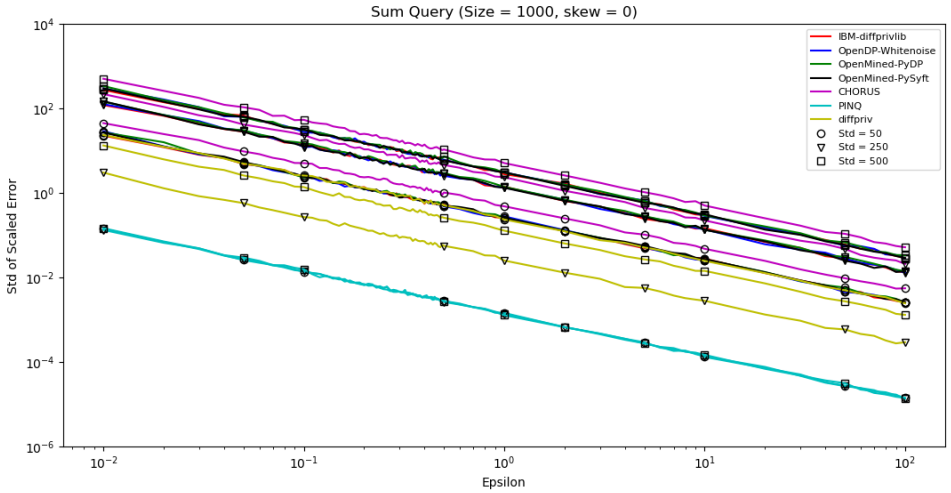

Figure 4.26 shows the results of *Sum query* for all libraries when dataset is of 1000 rows and 0 skew. Standard deviation of dataset is 50, 250, 500. PINQ results shows that std of scaled is not effected by the difference of standard deviation in datasets. Moreover, std of scaled error is far less than the other results because sensitivity for *Sum query* in PINQ is 1. diffpriv results shows that for 500 standard deviation in dataset, std of scaled is higher than when standard deviation in dataset is 250.

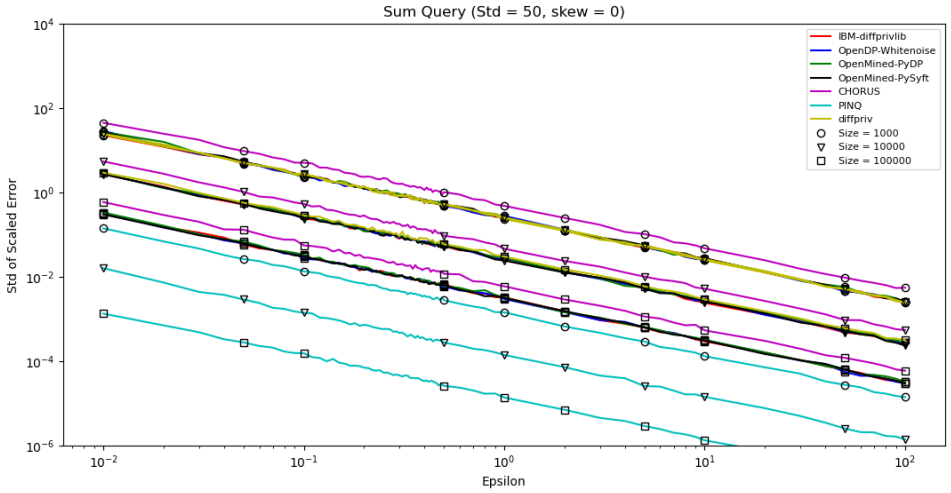

Figure 4.27 shows the responses of *Variance query* for dataset of different sizes when standard deviation and skewness of dataset is fixes (std = 50, skew = 0).

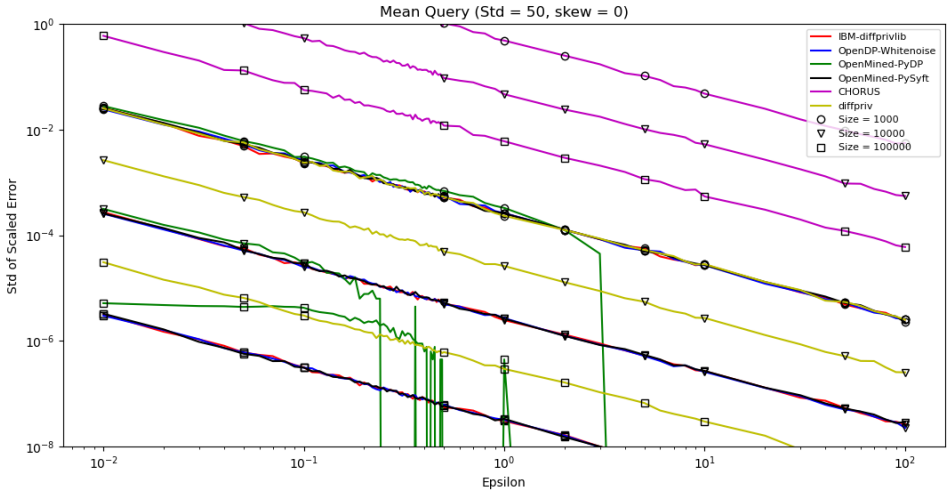

Figure 4.28 shows the response of libraries and framework for *Mean query*. For most the libraries, results are according to our expectations. We found abnormality in the results of diffpriv and OpenMined-PyDp.

As mentioned in the above table, only four libraries have *Standard Deviation query*. We will analyze the results for different values of standard deviation in the dataset and results for datasets of various sizes.

Figure 4.29 shows the results of the dataset for different values of standard deviations. The dataset has 1000 rows and 0 skewness. For IBM-diffprivlib, the std of scaled error decreases with an increase in the epsilon(ε) value. For higher standard deviation value in the dataset, lower is the std of scaled error. The results for OpenMined-PySyft are very high compared to other libraries' results as it uses the NumPy Laplace function. The diffpriv results are unexpected as the std of scaled error is more when the standard deviation of the dataset is 500 compared to results when the standard deviation in the dataset is 250. For OpenMined-PyDP, the std of scaled error for 500 standard deviations is higher than 250 and 50 standard deviation, which is the opposite of what we have seen for other libraries. For a dataset with 50 standard deviations, std of scaled error starts rising before epsilon(ε) value reaches 10^1, which means that more noise is added to the answer. For std = 250, a sudden drop before epsilon(ε) reaches 10^2, which is unique behavior compared to other results. The diffpriv results show that for 250 standard deviations in the dataset, std of scaled error is lowest. We can conclude that for a particular value of size and std, the diffpriv library's behavior is expected.

In Figure 4.30, we can see the result of all libraries and frameworks for *Standard Deviation query*. Similar to previous results, here again, OpenMined-PySyft results are high compared to other libraries' results. The trend of results for diffpriv and IBM-diffprivlib is identical. Diffpriv library results are a little bit higher than IBM-diffprivlib results.

Variance Query

Figure 4.31 shows the responses of *Variance query* when the dataset has 1000 rows, 0 skew and 50, 250, and 500 standard deviations. IBM-diffprivlib results are as per expectation. With an increase in epsilon value (ε), the std of scaled error decreases

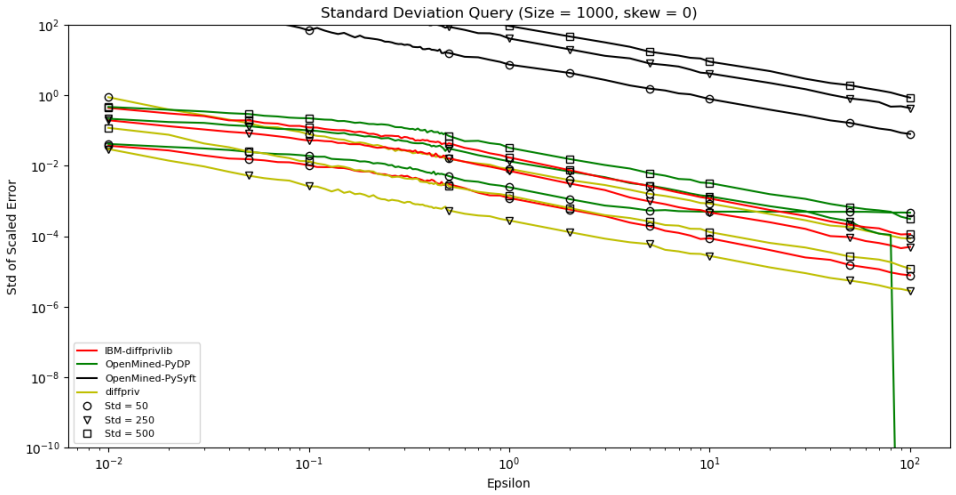

Figure 4.29 shows the results of *Standard Deviation query* for all libraries. The size and shape of dataset is 1000 and 0 while values of standard deviation in dataset are 50,250 and 500, represented by different shapes.

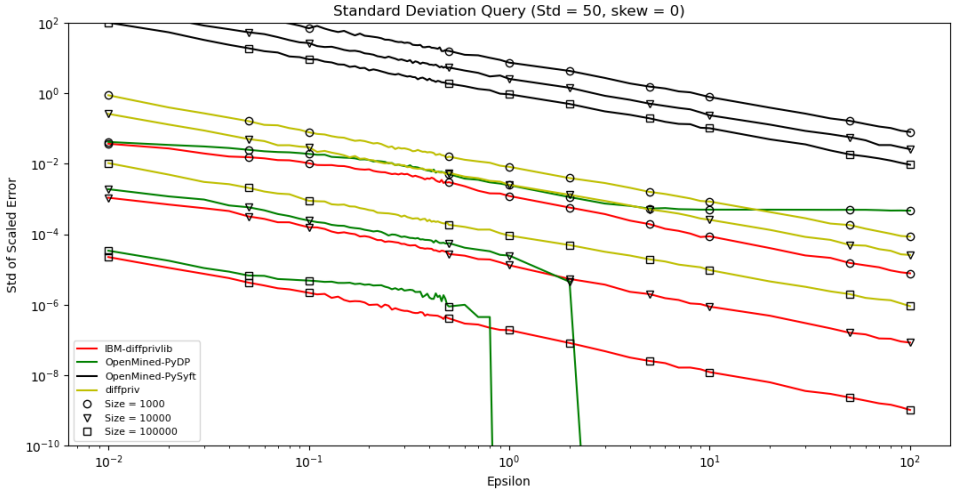

Figure 4.30 shows the responses of *Standard Deviation query* for dataset of different sizes when standard deviation and skewness of dataset is fixes (std = 50, skew = 0). OpenMined-PySyft results are much higher as compared to results of other libraries. OpenMined-PyDP and diffpriv shows abnormalities in their results.

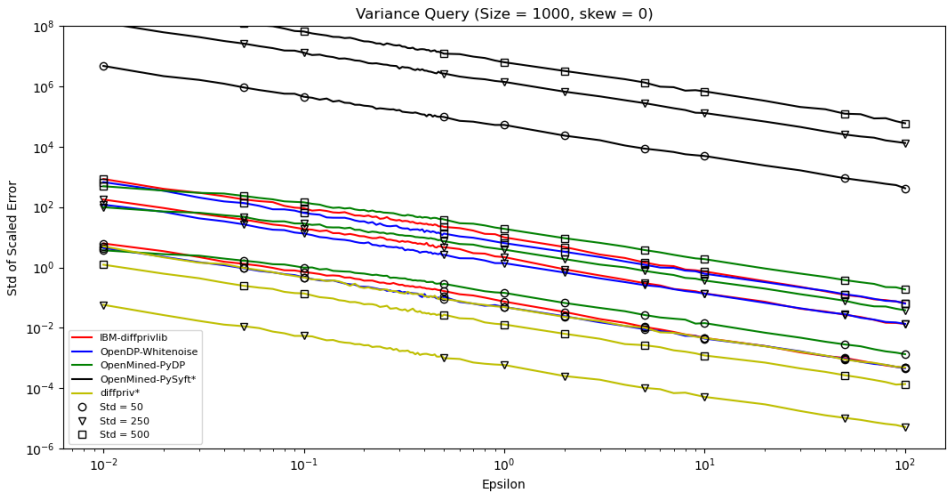

Figure 4.31 shows the responses of *Variance query* when dataset have 1000 rows, 0 skew and 50, 250, 500 standard deviations. The OpenMined-PySyft results are very high as compared to the results of other libraries. The results for diffpriv shows that less noise is added in an answer of dataset with 250 standard deviation. Generally std of scaled error is more for dataset with higher standard deviation.

gradually. OpenMined-PySyft results are very high as compared to the results of other libraries. The results for diffpriv shows that less noise is added in an answer of a dataset with 250 standard deviations. Generally, std of scaled error is more for a dataset with higher standard deviation.

To analyze the response of different libraries for *Variance query* are graphs generated for dataset sizes 1000, 10000, 100000 and standard Deviation 50, 500. Figure 4.32 shows the results for a dataset with different sizes with 50 standard deviation. Again OpenMined-PySyft results are higher than other results. Other than that, all results are according to our expectation (for a larger dataset, std of scaled error should be low), except the results of diffpriv. The std of scaled error is the same for the dataset of 1000 and 100000, while for 100000, it reduces but far less than the other libraries. The results of diffpriv for 100000 rows are similar to the results of IBM-diffprivlib, OpenDP-Whitenoise, and OpenMined-PyDP for 10000 rows. Other discuss-able results are of OpenMined-PyDP for a dataset with 1000 rows. An std of scaled error started rising before epsilon(ε) reaches 10^1 and then drop at once before epsilon(ε) reaches 10^2. The reason for this behavior is unclear and needs further investigation.

In Figure 4.33, we can see the results for different sizes of datasets with 500 standard deviations. OpenMined-PySyft results are higher than before. For IBM-diffprivlib, OpenDP-Whitenoise and OpenMined-PyDP results are according to our expectations. Here diffpriv results need discussion because for a dataset with 100000 rows and 500 standard deviation, the std of scaled error gets higher instead of lower. A diffpriv dataset with 10000 rows shows the lowest value for std of scaled error.

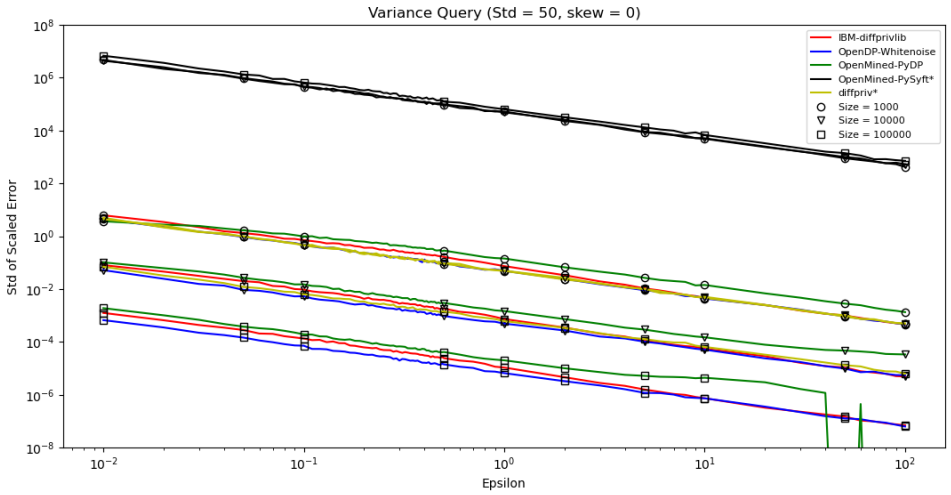

Figure 4.32 shows the responses of *Variance query* for dataset of different sizes when standard deviation and skewness of dataset is fixes (std = 50, skew = 0). The results for IBM-diffprivlib, OpenDP-Whitenoise are as per our expectation while OpenMined-PySyft, OpenMined-PyDP and diffpriv shows some abnormalities in results.

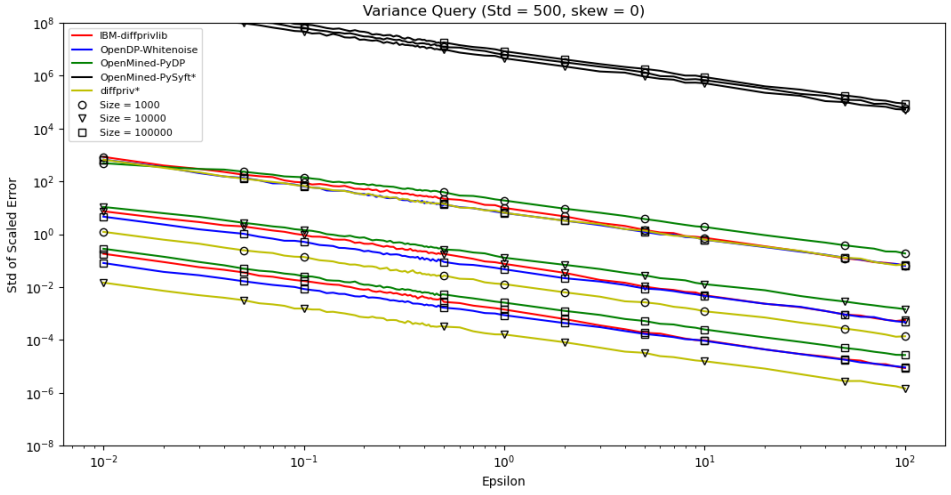

Figure 4.33 shows the responses of *Variance query* for dataset of different sizes when standard deviation and skewness of dataset is fixes (std = 500, skew = 0). The results for IBM-diffprivlib, OpenDP-Whitenoise and OpenMined-PyDP are as per our expectation while OpenMined-PySyft and diffpriv shows some abnormalities in results.

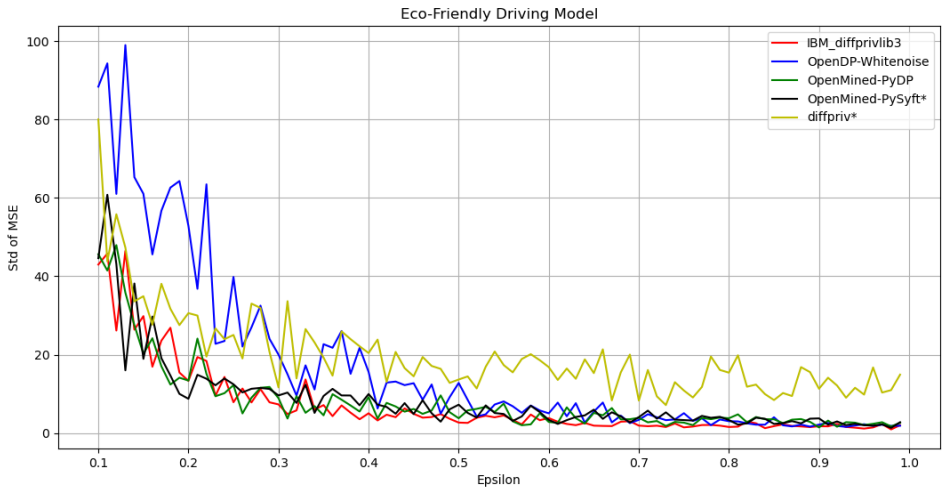

Figure 4.34 shows the results of Eco-Friendly Driving Model for all libraries. Epsilon is considered from 0.1 to 1.0 with step size of 0.01. This analysis gives are larger picture of libraries behavior for real-world use case.

4.4.1 Results for Eco-Friendly Driving Model

The Eco-Friendly Driving Model described in Section 3.2 (Scenario 2) is used to analyze the behavior of Differential Privacy libraries for the real world use case. We used our synthetically created 25 datasets of car sensor's data. For evaluation we considered epsilon(ε) from 0.1 to 1.0 with step of 0.01. To satisfy all Eco-Friendly Driving Model requirements, we require *Count, Sum, Mean* and *Variance queries*. IBM-diffprivlib, OpenDP-Whitenoise, OpenMined-PySyft, OpenMined-PyDP and diffpriv are the libraries containing mentioned queries Eco-Friendly Driving Model (Table in Section 4.4 shows a list of available queries in each library). All libraries are run 10 times for each value of epsilon(ε), then the Mean Squared Error (MSE) was calculated with non-differentially private results. The standard Deviation of MSE is then plotted.

Figure 4.34 shows the results of all libraries for the Eco-Friendly Driving Model. With an increase in the value of epsilon(ε), the std of MSE reduces for all libraries, which was expected. When epsilon(ε) reaches 0.5 the std of MSE for IBM-diffprivlib, OpenDP-Whitenoise, OpenMined-PySyft, OpenMined-PyDP are similar. It can be seen std of MSE for diffpriv is higher than other results. For diffpriv, this is mainly because of its unusual behavior due to the size of the dataset. As the datasets' size are different from each other and we have seen in the previous section, diffpriv add more noise to the answers when the size of the dataset exceeds a certain limit. From epsilon(ε) 0.1 to 0.5, the std of MSE for OpenDP-Whitenoise is higher than other libraries (except diffpriv) results in the same values of epsilon(ε). The OpenDp-Whitenoise results for a single query slightly higher than the results for IBM-diffprivlib. That is why for our real-world use case where multiple Differential Privacy queries are used, these slightly higher results combine and result in a high std of MSE.

$*$ The sensitivity is calculated manually.

4.4.2 Conclusion

This chapter analyzed Differential Privacy libraries and frameworks for their general characteristics, functionality, and performance under different scenarios. Figure 4.35 shows combined information for all the libraries and frameworks. The datasets described in Section 3.2 (scenario 1) are considered for performance analysis of individual queries. We discussed epsilon(ε) vs std of scaled error for all libraries and frameworks. After that, these results are compared among libraries and frameworks to determine which one is performing better. Finally, we have seen the results for Eco-Friendly Driving Model and discussed the possible reason behind the unexpected behavior of some libraries.

From our analysis, we can conclude that IBM-diffprivlib is better than other libraries because it contains all basic queries, its query execution time is faster, individual query results and real-world use case shows that its results are much stable as compared to many other libraries and frameworks last but not the least, it has a very active community that maintain the Git repository as well the documentation. For the next path of our thesis, we will use IBM-diffprivlib to apply Differential Privacy queries.

Evaluation Parameters		1) IBM-diffprivlib	2) openDP - White Noise	3) Google PyDP	4) Google Pysyft	5) PINQ	6) CHORUS - DP	7) diffpriv	8) GUPT	9) Tensorflow DP	10) MIT-PrivateMultiWeight
Mechanism	Laplace	●	●	●	◐	●	●	●	◌	○	●
	Gaussian	●	●	○	◐	○	○	●	○	●	○
	Exponential	●	○	○	○	●	●	●	◌	○	○
	Other (name)	Staircase, Uniform, Vector	Geometric	○	○	○	SparseVector Mechanism	Bernstein	○	○	○
Tools	Histogram	●	●	○	○	○	●	◐	◌	○	●/○
	Mean	●	●	●	◐	●	●	◐	◌	●/○	●/○
	Count	●	●	●	◐	●	●	◐	◌	○	●/○
	Sum	●	●	●	◐	●	●	◐	○	●/○	○
	Variance	●	●	●	◐	○	○	◐	○	○	○
	Standard Deviation	●	○	●	◐	○	○	◐	○	○	○
	Other (name)	○	Covarience , Max, Min, Median	min, max, median	min, max, median	median	○	○	time series	○	min, max
Bounded/ Unbounded		Bounded	Bounded & Unbounded	Bounded	manual	○	Bounded	manual	Bounded	○	○

Figure 4.35 shows combined information for all the libraries and frameworks. Empty circle represents that functionality does not exist. Dotted circle represents that functionality exist but obsolete now. Semi filled circle represents that library does not has builtin function and sensitivity is calculated manually for available tools. Fully filled circle represents that functionality exist with builtin function. Fully filled circle/empty circle represents that tool exist but we can't use it for single queries.

5

Differentially Private Query Management System for Vehicular Data Marketplaces

This part of the thesis aims to design and implement a basic query management system for vehicular data market that allows data analysts or data processors to receive answers to statistical questions while maintaining an individual's privacy. We propose a system that provides a solution for all the challenges outlined in Section 3.3. In this chapter, we elaborate on the system's design and implementation strategies. The next section will see an overview of our solution and then a description of design goals. Section 5.1 states the overview of our design along with the design goals. In Section 5.2, we discussed our proposed design and our privacy budget strategy. Section 5.3 states list of statistical questions our system is capable to answer and what are the queries particular question is composed of. Finally we discussed briefly about system implementation and execution.

5.1 Overview

As described in the problem statement (Section 3.3), there is a need for a differentially private query system for vehicular data marketplaces. The data analyst wants to issue a statistical query to the system and in return, she receives Differentially Private results while data providers wish to benefit from sharing the data. As a solution to our scenario, we propose to offer the data analyst a marketplace and set of statistical queries. Although we consider data providers and data analysts to be semi-honest, our target is to prevent an individual's privacy. To achieve the target, we provide pre-defined statistical questions related to the automotive industry.

Since data analyst has a vital role in our system, it has access to the system's primary section, where she can select the desired statistical question. In contrast,

the data provider has a minimal role in submitting the data to the database and waiting for an approximated reward once the data is utilized. The reasons for giving minimal control to data analysts are because there is a possibility that she has some background knowledge about the overall population, which can lead to the disclosure of individual data. For our analysis, we considered that all pre-defined statistical questions to be independent of each other and the privacy budget are equally divided among them. That is why we don't allow data analysts to ask the same question more than once because the average of several answers for the same statistical question might reveal the individual's privacy. This is done by caching the statistical queries numbers, already asked by the data analyst, in the database. In a real-world scenario, there is a possibility of the relation among questions; that is why we used the privacy budget concept.

5.1.1 Design Goals

In the last section, we discussed an overview of our proposed system. However, in this section, we discuss more precisely our features and system architecture. In conjunction with the challenges stated in Section 3.3, we set the following design goals:

Statistical Question

All questions must be related to the automotive industry and help data analysts to make sound decisions based on the Differentially Private results. As mentioned earlier, all questions are considered independent, which means that answers to more than one question can not be combined to reveal individuals' information. A data analyst should not be allowed to select the same statistical question twice.

Access to Queries

As described in the problem statement, out of two parties (data provider and data analyst), only the data analyst has access to choose statistical questions. The role of the data provider is to submit the dataset. Furthermore, unlike other systems, our system must not allow data analysts to choose privacy parameters.

Privacy Budget

According to the defined strategy, the privacy budget should be specified for each statistical question (containing Differential Privacy queries). The privacy budget must reduce every time a statistical question is selected. Once it reaches zero, the data analyst must not be able to ask further questions.

Extensibility and Scalability

Another goal is to make the system extensible and scale-able so that more features like incentive system, in-built data anonymization systems, and other privacy protection mechanisms could be added. Moreover, the system has to scale in a way that it can handle a massive amount of data.

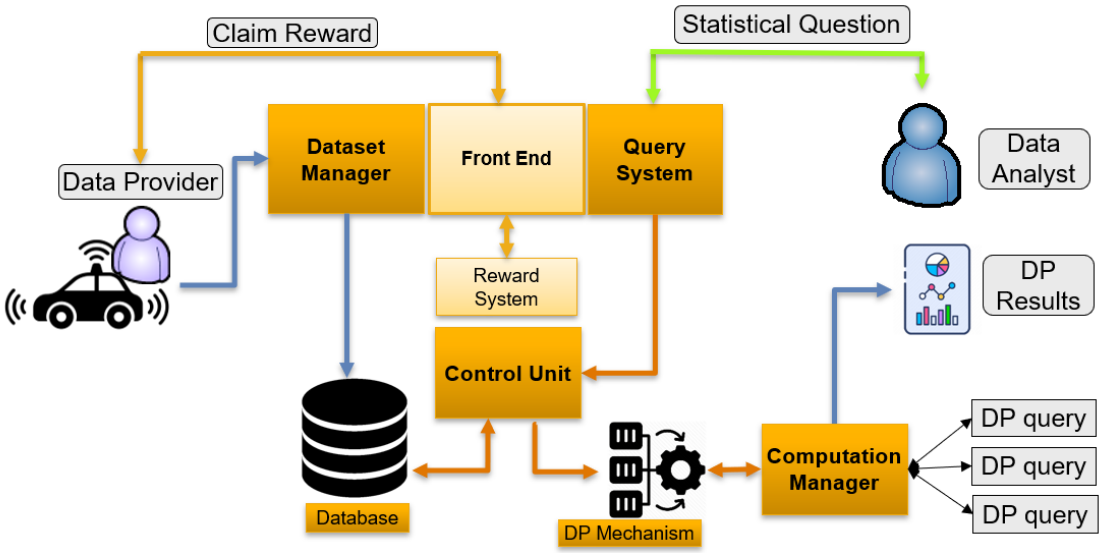

Figure 5.1 System Design: The system consists of four major parts. All these parts are related to the backend of the system. Frontend parts are shown in the design, but they are left to be integrated in the future. *Query system* receive a statistical question from the data analyst and check either this question has been asked earlier or not. *Control unit*, control the number of queries in every question. It communicates with the database and provides all required datasets along with queries to the Differential Private mechanism. *Dataset Manager* keeps the record of all datasets before sending it to the central database. The *Computation Manager* interacts with the DP mechanism to compute Differentially Private results for all inherited queries in statistical questions. It is also responsible for managing the privacy budget.

5.2 System Design

In this section, we will describe the design of our system we propose to tackle our questions. Figure 5.1 shows our proposed design. The system consist of four major parts **Dataset Manger**, **Query System**, **Control unit** and **Computation Manager**. All these parts are related to the backend of the system. The query system receives statistical questions from the data analyst and checks whether this question has been asked earlier. The control unit controls the number of queries in every question. It communicates with the database and provides all required datasets along with queries to the Differential Private mechanism. Dataset Manager keeps the record of all datasets before sending it to the central database. The Computation Manager interacts with the DP mechanism to compute Differentially Private results for all inherited queries in statistical questions. It is also responsible for managing the privacy budget.

5.2.1 Backend

The backend of the system contains most of the functionality described in the overview. The query unit manages the statistical questions asked by the data analyst. It determines either the user is requesting the selected statistical question for the first or not. If the user is asking the same statistical question twice, then the query unit will stop the execution return an error message to the data analyst. The dataset manager receives the user's dataset then checks it either with all the

required sensors data. The communication with the frontend and handling of the reward system is also done by the control unit. Furthermore, it collect all available dataset from storage and send it to Computation Manager along with queries in the statistical question. The computation manager communicates with the Differential Privacy mechanism and returns DP results to the data analyst. The reward system and frontend can be integrated later in the system. Now we will describe the functionality of these components in more detail. We will also discuss the flow of operations in Query unit and Computation Manager.

Control Unit

The control unit is the central administrative unit of the backend. It is the communication partner of the frontend's communicator. Furthermore, it handles the query unit's statistical questions by dividing it into queries to be sent to the DP mechanism. The control unit is also responsible for communicating with the primary database. It fetches the datasets of all cars except the last dataset because there is a possibility that the data analyst himself/herself first act as a data provider and now acting as a data analyst by asking the statistical question, so there is a chance in a real-world scenario that he/she can learn something new about an individual present in the dataset. It is impossible to prove mathematically practically the possibility of such a privacy breach because we don't know the data analyst's background knowledge. Here we are just getting extra cautious. Besides this, if our data store has a large number of datasets, then excluding one dataset will not affect the results much, but we will be assured that there is no possibility of a privacy breach.

To maintain every user's privacy, we are not accepting queries from users. Instead, we provide pre-defined statistical questions so data analysts can choose from them. The selected question when reaches the control unit from the query unit, the queries of which statistical question comprised of are separated. The number of queries in the statistical questions varies from 1 to 4. These queries, along with the datasets fetched from the data store, are sent to the DP mechanism. Furthermore, It also has an authentication unit where a user can log in with the provided username and password. The record of statistical query execution is kept in the SQL database. If the user exit from the system and then login again, the user will only be able to execute a statistical question that was not answered previously.

Query Unit

Query unit is responsible for two major tasks: first, to ensure the availability of all statistical questions so that the data analyst can choose from them. Secondly, it is responsible for validating the data analyst's statistical question before sending it to the control unit. If a data analyst asks the same query again, further computation will be canceled and the error message will be returned. If the selected question is passes the verification, then it will be sent to the control unit. Figure 5.2 depict these steps.

Computation Manager

The computation manager's role is to receive datasets and queries generated by control unit for the asked statistical question while communicating with DP-mechanism. The DP-mechanism contains our proposed privacy budget strategy and communicates it with the computation manager. The computation manager has DP query

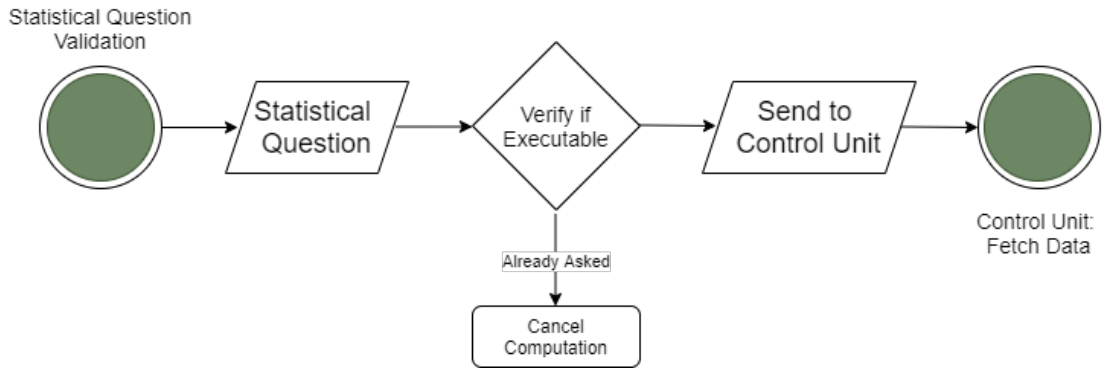

Figure 5.2 Statistical Question Validation: this figure depicts the role of the Query unit in our proposed system. First, the data analyst selects the query unit; then, the query unit verifies if it is executable by checking the data analyst's previous query. If the data analyst asks the same query again, further computation will be canceled and an error message will be returned. If the selected question is passes the verification, then it will be sent to the control unit.

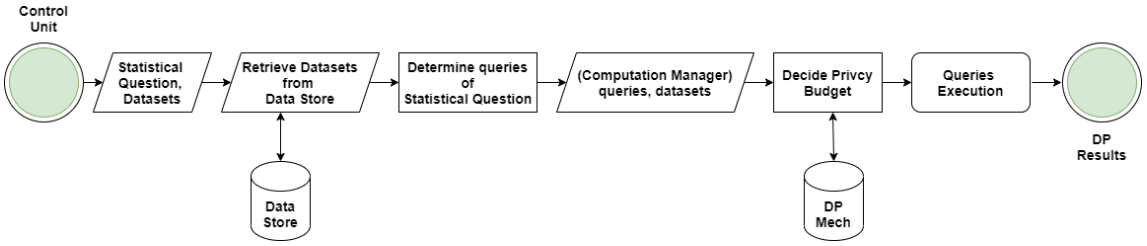

Figure 5.3 Statistical Question Execution: This figure depicts the process for statistical query execution. The control unit has access to the data store and DP-mechanism. First, the control unit will retrieve the data sets from the store; then it determines the statistical question queries. These queries and datasets will reach the computation manager, communicating with DP-mechanism to decide the privacy parameters. The computation manager has query execution chambers where noise is added to the query answer according to our privacy budget strategy. Finally, these results are combined to return the privacy preserved answer of the statistical question.

execution chambers where the query is executed against the privacy parameters provided by DP-mechanism. When all the queries are executed, the results are then combined to answer the statistical question. Though we considered that all statistical questions are independent of each other. However, we still decided to release the results not to leak any information about the individual. Either these releases are in the form of percentages or in the way of averages. Figure 5.3 depicts the overall process.

Dataset Manager

Dataset Manager is responsible for receiving the car sensor dataset, validate the data and submit it to the data store. It makes sure that the submitted dataset contains at least all the required columns *(time-stamp, acceleration, speed, fuel consumption and brakes)*. If any of these columns are missing, then the dataset will be rejected. Dataset managers also record the total number of datasets in the data store to instruct the data provider to rename the dataset accordingly. It provides all datasets to the control unit when required.

5.2.2 Privacy Budget Strategy

For pure Differential Privacy, the epsilon(ε) parameter can be considered a privacy budget. It determines the amount of noise to be added to make the results private. But adding a wrong amount of noise can leak privacy. The danger is an aggregating average of Differentially Private results by asking the same query much time; one can reach the real answer and possibly reveal some sensitive information. This danger is tackled by not allowing the data analyst to ask the same question twice. So, the data analyst will not be able to average results and have to settle with one answer. As we assumed that all statistical questions in our system are independent of each other, so we decided to provide a privacy budget(ε), which return results with an accuracy of 85% - 95% for each statistical question. If any statistical question has more than one Differential Privacy query, then ε will be equally divided among them (e.g., In question 2 where two *Count queries* are needed to get the answer. Value of privacy budget will$\varepsilon/2$ for each query), then the sum of all privacy budget will be equal to the total epsilon. In the implementation section, we will discuss all statistical questions according to our privacy budget strategy. We will calculate the execution time for each statistical question.

In the next section, we state all statistical questions for our system.

5.3 Statistical Questions

For our system, we considered ten statistical questions that could be most beneficial to improving the automotive industry's decision-making. Each statistical question contains at least one Differential Privacy query. These statistical questions, along with their Differential Privacy queries, are stated as follows:

1. How much is the average fuel consumption per hundred kilometers?

Here we are using Differential Privacy *Sum query* to get the aggregate sum of instantaneous fuel for every dataset. This Differentially Private sum is used to calculate *Fuel Consumption per hundred kilometers [FPH]*[12]. The average of all FPHs is returned.

2. What is the percentage of aggressive drivers?

Here we are considering two cases from the eco-friendly driving model, *Acceleration Sharply* and *Deceleration Sharply* (Chapter 3). Differential Privacy *Count query* is applied to the final count of both cases.

3. What is the percentage of eco-friendly drivers?

In this question, we considered a complete eco-friendly driving model. The total score of all datasets is averaged and its percentage is returned. Here we are using Differential Privacy *Count query, Sum query, Mean query*, and *Variance query.*

4. What is the percentage of cars required maintenance?

Car maintenance is usually related to its fuel consumption. So, if the fuel consumption per 100km of a car is more than the average fuel consumption per 100km of all

vehicles, we consider it a vehicle requiring maintenance. Here Differential Privacy *Sum query* is used in the calculation of fuel consumption per 100km.

5. For the 2nd segment of the eco-driving model, which case has the most and least influence on the final score?

This question will help decision-makers understand the driver's behavior more clearly and decide their future strategies accordingly. Here we are using Differential Privacy *Count query, Sum query, Mean query* and *Variance query* in different cases.

6. How much money can be saved if everyone drives eco-friendly?

We calculated the average of actual fuel consumption per 100km for all the datasets to answer this question. If any car consumes more than the actual average, the cost of this extra fuel is calculated. Finally, we returned the sum of all these costs. On 10 Sep 2020, the price of Gasoline was 1.294 euros/ltr. Here only Differential Privacy *Sum query* is used.

7. What is the effect of *Acceleration Sharply* on eco-driving results?

From here onward, we will see the impact of a single case on eco-driving results. Here we are calculating the effect caused by *Acceleration Sharply* on the total eco-driving score. It requires Differential Privacy *Count query, Sum query, Mean query* and *Variance query.*

8. What is the effect of *Deceleration Sharply* on eco-driving results?

The answer to this question will determine the effect of *Deceleration Sharply* on the overall eco-driving score. If this effect is not high, then decision-makers might focus on other, more convincing cases. Similar to the previous case, this case also requires Differential Privacy *Count query, Sum query, Mean query* and *Variance query.*

9. What is the effect of *High Cruising* on eco-driving results?

Considering all sub-scenarios in *High Cruising*, we calculated the eco-driving score and then subtracted it from the score where we did not consider *High Cruising* case. This difference is the effect of *High Cruising* on the eco-driving score.

10. How *Brake Pressing* affects eco-driving results?

More fuel is consumed when brakes are pressed. It will affect the overall eco-driving score because of its first segment, the fuel-related hundred-mark score.

Before jumping to the design section, let us discuss our privacy budget strategy.

5.4 Implementation and Evaluation

After we described our system's design in the last section, we focus on the implementation of our prototype in this section. Here we give an overview of the used technologies and frameworks. Afterward, we describe how the data analyst submits statistical questions and the result returned by our system. Then we will evaluate the computation time, single query execution time, and accuracy of an answer for all the statistical questions.

Our prototype is implemented in Python3 and we used IBM-diffprivlib v0.3.0 to apply Differential Privacy. The reason for selecting this library is discussed in Section 4.4.2. Our evaluation is based on 25 datasets and more could be added, but it will increase the computation time. To determine the privacy budget (ε), we run all statistical questions (that returns numeric results) 100 times for each value of ε and calculate the average accuracy. The ε value where average accuracy is between 85% - 95%, is selected for that statistical question. E.g. For statistical question 1, accuracy is 95% at epsilon(ε) 0.13. We will select this value of epsilon for first question.

User Authentication

When our prototype system is executed, it will first ask the data analyst to login. We provide these login credentials. We avoid the method where the user has to register and then login because there is a possibility that the user will register many accounts with different names. In our proposed system, we will provide a username and password to the user. After a successful login user will able to see all options.

Selecting Statistical Question

As the system's front end is not the focus here, our system is running on interactive terminal where you can find all the statistical question which we mentioned Section 5.3. Besides this it also have option for dataset submission. By selecting the *Submit Dataset* option, user can provide the dataset path. The system will automatically rename the dataset and store in main database.

Statistical Question Execution

For the demonstration purposes, we are considering statistical question 1 in which the average fuel consumption of all cars available in the dataset is calculated. To select the appropriate value of epsilon(ε), we run the statistical question 1, 100 times for each epsilon(ε) and then average the accuracy. The value of epsilon(ε) with an average accuracy of 95% is selected. Figure 5.4 shows the average accuracy for each value of ε.

In conclusion, if a data analyst runs our system 100 times, he/she can not reach 100% accuracy. We repeated the same process for other statistical questions to select the epsilon(ε) value. As mentioned in the previous section, we do not allow the user to ask the same question twice. Every user who wants to use the system has to login and then he/she can choose the available options. When a data analyst selects any statistical question and gets the results, he/she will not be allowed to select that option again even after exiting from the system and login in again. This additional feature will allow the user to return and use the system anytime by just simple login.

The execution time depends on the number of DP queries for each statistical question. For statistical questions where only one DP query is used, their execution time is far less than the satistical question. The whole eco-friendly driving model is executed to return the real and differentially private results.

New Dataset Insertion

Dataset managers handle the insertion of new datasets. When data analyst run the system, 12 option appears. The user has to select option 11 to add a new dataset and provide the dataset path. Now, if the dataset exists, the dataset manager will

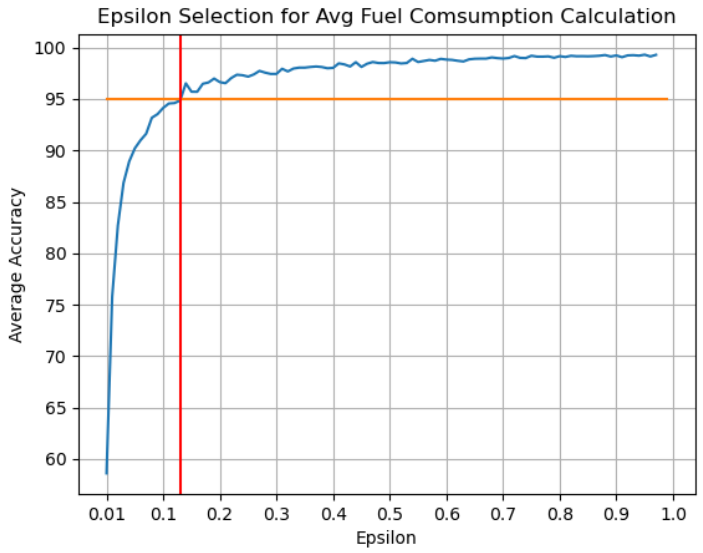

Figure 5.4 shows the epsilon(ε) vs average accuracy for statistical question 1. At epsilon = 0.13 the average accuracy will be 95%.

check the dataset's columns. A dataset at least have these columns: *Time Stamp, Acceleration, Speed, Brakes, Fuel*

If any of these columns are missing in the dataset, it will be rejected; otherwise, the file will be added with other datasets and an appropriate name will be allotted to it. Now, results for the new user will include this added dataset.

Example: Outcome of Question 1

Figure 5.5 shows the results for Statistical Question 1 in which data analyst want to find out what is fuel consumption of cars per 100 km. The bar graph shows what is the minmum, maximum, average and idle fuel consumption. All results are differentially private. If we run this case again then the answers will be different.

In conclusion, we can say that our system provides a first glimpse of how Differential Privacy system for vehicular data market place should look like. Our proposed design cover some important features like dataset manager, control unit, computation manager and privacy budget strategy. There is still room for imporvement, new features like user and data analyst registration, validation of submitted data and dynamic privacy budget selection, can be added to the current system.

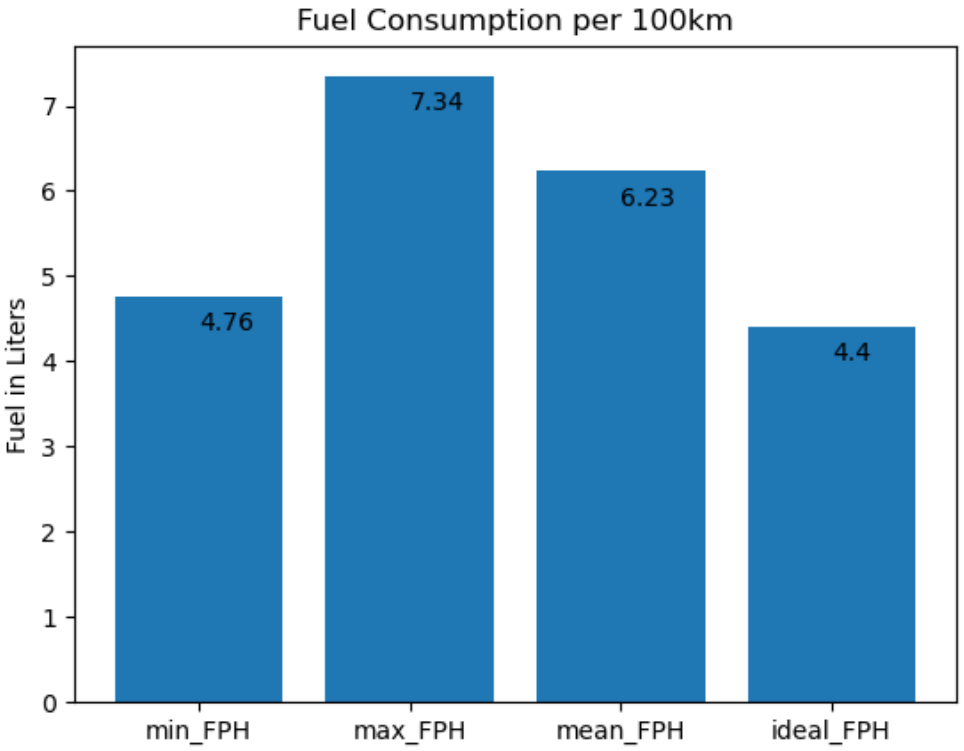

Figure 5.5 Outcome of a question 1. Fuel consumption of cars per 100km. These results are differentially private and their value will change if we re-run the question.

6

Conclusion

In this thesis, we conducted a comparative analysis of available Differential Privacy libraries and frameworks to select the better out of them which then can be used in Differential Privacy Query Management System for Vehicular Data Marketplace. In the current situation, cars sensor's data is collected by the companies for analysis and also sell it to other entities, which help them in decision making for their companies. There are many techniques that claim secure the individual's privacy, but in past we have seen many cases where the privacy was breach during the analysis and sharing of results. For example, Data Anonymization is one of the privacy preserving technique where personal identifiable information of the users is removed from the dataset to secure the privacy of person whose data is present in dataset. Some researches showed that it is possible to re-identify the person even when the dataset is anonymized. This invoked the need of technique which preserve the individual's privacy while doing statistical analysis on available data. The state-of-the art data privacy technique known as Differential Privacy promises this individual's privacy. According to Differential Privacy, the outcome of any analysis is essentially equally likely, independent of whether any individual joins, or refrain from joining the dataset. Practically this privacy is achieved by adding the calculated amount of noise in the original results.

So far my research organizations and companies have developed their solution that confirms that outcome of the analysis is differentially private. During our initial research we realized that thorough comparison of available libraries and framework which confirms differentially private results, is missing. So in first part of our study, we did the comparative analysis of available Differential Privacy libraries and frameworks. They were analyzed on the basis of their general characteristics, functionality and performance under certain scenarios. Based on results, we concluded that overall performance of IBM-diffprivlib is better than other libraries and frameworks.

The second part of the study we proposed a differentially private query management system for vehicular data marketplace, that answers the statistical questions while maintaining the individual's privacy. Our proposed design consist of dataset manager, query system, control unit and computation manager. The system allow data

analyst to select the desired statistical question related to automotive industry. The query system validates the question and send it to control unit. The Control unit gets the available datasets from database and send queries in statistical questions to computation manager along with retrieved dataset. At computation manager queries are computed differentially private results are returned. The system ensures that no user's data can retrieved by data analyst while querying the dataset. Furthermore, it follows our privacy budget strategy which select the appropriate amount to noise to be added in results and it also restrict the data analyst to issue a statistical question only one time. As our prototype system is extensible, more features can be added to make it more interactive and user friendly. Right now we have interactive console where data analyst can login and select the question to get the differentially private results.

6.1 Future Work

For future work we suggest to analyze Differential Privacy libraries for Gaussian and Exponential mechanisms. As our analysis focuses on pure Differential Privacy where epsilon(ϵ) is the only privacy parameter, we suggest to consider (δ) parameter to get differentially private results. For now our evaluation focuses mainly on statistical operation, for future we suggest to evaluate considered libraries for more complex queries and machine learning.

For the second path we suggest to develop frontend and integrate it with our proposed design. We suggest to especially focus on the registration process of data analyst and data provider. In our study we assumed that data provider is non-malicious while data analyst is semi-malicious but in practical scenario it is important to make sure that receiving data is not tempered. Furthermore, if the data analyst is malicious then out privacy budget strategy should to strong enough to prevent the individual's privacy.

For our analysis we used fully synthetically generated data as well as semi-synthetically generated data (where data was generated from initially provided datasets). Thus, it would be interesting to see how our prototype handle real datasets. Additionally, as future work further aggregations functions could be implemented that allow more utility in results while maintaining user's privacy. The current prototype does only support fixed sets of privacy parameters. Hence, future work could modify the processing pipeline in a way that these privacy parameters are calculated by system itself and return the results accordingly. Besides this, there is huge scope of improvement in query system, as our current prototype has pre-defined set of question. It is desirable to improve query system in a way that it allow data analyst to design her own statistical question.

Bibliography

[1] M. Alaggan, S. Gambs, and A.-M. Kermarrec. Heterogeneous Differential Privacy. *Journal of Privacy and Confidentiality*, 7(2), 2017.

[2] N. Amiet. Differential Privacy: A Comparison of Libraries (Accessed on 1.8.2020), https://research.kudelskisecurity.com/2020/03/11/differential-privacy-a-comparison-of-libraries/, 2020.

[3] S. Asseffa and B. Seleshi. A Case Study on Differential Privacy (Accessed on 10.7.2020), http://www.diva-portal.org/smash/record.jsf, 2017.

[4] B. NELSON. Thesis for the degree of licentiate of engineering (Accessed on 26.10.2020), http://publications.lib.chalmers.se/records/fulltext/235187/235187.pdf. 2016.

[5] B. Balle and Y. X. Wang. Improving the Gaussian mechanism for differential privacy: Analytical calibration and optimal denoising. In *35th International Conference on Machine Learning, ICML 2018*, volume 1, pages 678–692. International Machine Learning Society (IMLS), 2018.

[6] M. Barbaro and T. Zeller Jr. A Face Is Exposed for AOL Searcher No. 4417749 - The New York Times, 2006.

[7] D. C. Barth-Jones. The 'Re-Identification' of Governor William Weld's Medical Information: A Critical Re-Examination of Health Data Identification Risks and Privacy Protections, Then and Now. *SSRN Electronic Journal*, 2012.

[8] F. Boenisch. Differential Privacy: General Survey and Analysis of Practicability in the Context of Machine Learning. Technical report, 2019.

[9] G. Bondel, G. Garrido, K. Baumer, and F. Matthes. The Use of De-identification Methods for Secure and Privacy-enhancing Big Data Analytics in Cloud Environments. In *Proceedings of the 22nd International Conference on Enterprise Information Systems*, pages 338–344. SCITEPRESS - Science and Technology Publications, may 2020.

[10] C. Chen, X. Zhao, Y. Yao, Y. Zhang, J. Rong, and X. Liu. Driver's eco-driving behavior evaluation modeling based on driving events. *Journal of Advanced Transportation*, 2018, 2018.

[11] R. Cummings, K. A. Lai, S. Krehbiel, and U. Tantipongpipat. Differential privacy for growing databases. In *Advances in Neural Information Processing Systems*, volume 2018-Decem, pages 8864–8873, 2018.

[12] I. De Vlieger. On-board emission and fuel consumption measurement campaign on petrol-driven passenger cars. *Atmospheric Environment*, 31(22):3753–3761, nov 1997.

[13] Differential Privacy Leaders you Must Know (Accessed on 15.10.2020). https://blog.linknovate.com/differential-privacy-leaders-must-know/, 2018.

[14] C. Dwork. LNCS 4052 - Differential Privacy. *Automata, Languages and Programming*, 2006.

[15] C. Dwork. An ad omnia approach to defining and achieving private data analysis. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 4890 LNCS:1–13, 2008.

[16] C. Dwork. Differential privacy: A survey of results. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 4978 LNCS:1–19, 2008.

[17] C. Dwork. The promise of differential privacy: A tutorial on algorithmic techniques. In *Proceedings - Annual IEEE Symposium on Foundations of Computer Science, FOCS*, pages 1–2, 2011.

[18] C. Dwork, F. McSherry, K. Nissim, and A. Smith. Calibrating noise to sensitivity in private data analysis. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 3876 LNCS, pages 265–284, 2006.

[19] S. Ezzini, I. Berrada, and M. Ghogho. Who is behind the wheel? Driver identification and fingerprinting. *Journal of Big Data*, 5(1):1–15, dec 2018.

[20] P. Golle. Revisiting the uniqueness of simple demographics in the US population. In *Proceedings of the ACM Conference on Computer and Communications Security*, pages 77–80, New York, New York, USA, 2006. ACM Press.

[21] M. Hardt and G. N. Rothblum. A multiplicative weights mechanism for privacy-preserving data analysis. In *Proceedings - Annual IEEE Symposium on Foundations of Computer Science, FOCS*, pages 61–70, 2010.

[22] M. Hay, A. Machanavajjhala, G. Miklau, Y. Chen, and D. Zhang. Principled evaluation of differentially private algorithms using DPBENCH. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, volume 26-June-20, pages 139–154, 2016.

[23] N. Hynes, D. Dao, D. Yan, R. Cheng, and D. Song. A demonstration of sterling: A privacy-preserving data marketplace. In *Proceedings of the VLDB Endowment*, volume 11, pages 2086–2089. PVLDB, 2018.

[24] B. P. U. Ivy. A Simple and Efficient Counting Algorithm for Data Encryption and Decryption. 14:2950–2956, 2018.

[25] P. Jain, M. Gyanchandani, and N. Khare. Differential privacy: its technological prescriptive using big data. *Journal of Big Data*, 5(1), dec 2018.

[26] Z. Ji, Z. C. Lipton, and C. Elkan. Differential Privacy and Machine Learning: a Survey and Review. 2014.

[27] M. Johanson, J. Jalminger, E. Frécon, B. Nelson, T. Olovsson, and M. Gjertz. Joint subjective and objective data capture and analytics for automotive applications. *IEEE Vehicular Technology Conference*, 2017-Septe:1–5, 2018.

[28] N. Johnson, J. P. Near, J. M. Hellerstein, and D. Song. Chorus: Differential Privacy via Query Rewriting. sep 2018.

[29] F. Kargl, A. Friedman, and R. Boreli. Differential privacy in Intelligent Transportation Systems. In *WiSec 2013 - Proceedings of the 6th ACM Conference on Security and Privacy in Wireless and Mobile Networks*, pages 107–112, 2013.

[30] V. Koutsos, D. Papadopoulos, D. Chatzopoulos, S. Tarkoma, and P. Hui. Agora: A Privacy-Aware Data Marketplace. Technical report.

[31] S. Lestyán, G. Acs, G. Biczók, and Z. Szalay. Extracting vehicle sensor signals from can logs for driver re-identification. In *ICISSP 2019 - Proceedings of the 5th International Conference on Information Systems Security and Privacy*, pages 136–145, 2019.

[32] J. Li, H. Ma, G. Wu, Y. Zhang, B. Ma, Z. Hui, L. Zhang, and B. Zhu. A workload division differential privacy algorithm to improve the accuracy for linear computations. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, volume 12138 LNCS, pages 439–452. Springer, jun 2020.

[33] N. Li, T. Li, and S. Venkatasubramanian. t-Closeness: Privacy Beyond k-Anonymity and-Diversity. Technical report.

[34] T. Li and N. Li. Injector: Mining background knowledge for data anonymization. In *Proceedings - International Conference on Data Engineering*, pages 446–455, 2008.

[35] F. Liu. Generalized Gaussian Mechanism for Differential Privacy. *IEEE Transactions on Knowledge and Data Engineering*, 31(4):747–756, 2019.

[36] J. Liu, C. Zhang, and Y. Fang. EPIC: A Differential Privacy Framework to Defend Smart Homes Against Internet Traffic Analysis. *IEEE Internet of Things Journal*, 5(2):1206–1217, 2018.

[37] A. Machanavajjhala, D. Kifer, J. Gehrke, and M. Venkitasubramaniam. l-diversity: Privacy beyond k-anonymity. *ACM Transactions on Knowledge Discovery from Data*, 1(1), 2007.

[38] F. Mcsherry. An Extensible Platform for Privacy-Preserving Data Analysis. *Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD 2009*, pages 89–97, 2009.

[39] F. McSherry. Privacy integrated queries: An extensible platform for privacy-preserving data analysis. *Communications of the ACM*, 2010.

[40] F. McSherry and K. Talwar. Mechanism design via differential privacy. In *Proceedings - Annual IEEE Symposium on Foundations of Computer Science, FOCS*, pages 94–103, 2007.

[41] K. Minami, H. Arai, I. Sato, and H. Nakagawa. Differential privacy without sensitivity. *Advances in Neural Information Processing Systems*, 0(Nips):964–972, 2016.

[42] S. K. Murakonda, R. Shokri, and G. Theodorakopoulos. Ultimate Power of Inference Attacks: Privacy Risks of Learning High-Dimensional Graphical Models. may 2019.

[43] A. Narayanan and V. Shmatikov. How To Break Anonymity of the Netflix Prize Dataset. oct 2006.

[44] B. Nelson and T. Olovsson. Introducing differential privacy to the automotive domain: Opportunities and challenges. *IEEE Vehicular Technology Conference*, 2017-Septe:1–7, 2018.

[45] K. Nissim, S. Raskhodnikova, and A. Smith. Smooth sensitivity and sampling in private data analysis. In *Proceedings of the Annual ACM Symposium on Theory of Computing*, pages 75–84, 2007.

[46] R. M. E. Rajendran Keerthana, Jayabalan Manoj. A Study on k-anonymity, l-diversity, and t-closeness Techniques focusing Medical Data. *IJCSNS International Journal of Computer Science and Network Security*, 17(12), 2017.

[47] A. Roth and T. Roughgarden. Interactive privacy via the median mechanism. In *Proceedings of the Annual ACM Symposium on Theory of Computing*, pages 765–774, nov 2010.

[48] K. Sako. Semantic Security. In *Encyclopedia of Cryptography and Security*, pages 1176–1177. Springer US, 2011.

[49] P. Samarati and L. Sweeney. Protecting Privacy when Disclosing Information: k-Anonymity and its Enforcement Through Generalization and Suppresion. *Proc of the IEEE Symposium on Research in Security and Privacy*, 1998.

[50] Secure and Private AI - Udacity (Accessed on 15.10.2020). https://classroom.udacity.com/courses/ud185.

[51] B. Steephenson. Implementing Differential Privacy Using Randomized Response Algorithms. 2017.

[52] L. Sweeney. Simple demographics often identify people uniquely. *Carnegie Mellon University, Data Privacy Working Paper 3. Pittsburgh 2000*, pages 1–34, 2000.

[53] Team Opendp. The OpenDP White Paper (Accessed on 20.9.2020), https://projects.iq.harvard.edu/files/opendp/files/opendp_white_paper_11may2020.pdf. 2020.

[54] S. Truex, L. Liu, M. E. Gursoy, W. Wei, and L. Yu. Effects of differential privacy and data skewness on membership inference vulnerability. In *Proceedings - 1st IEEE International Conference on Trust, Privacy and Security in Intelligent Systems and Applications, TPS-ISA 2019*, pages 82–91, 2019.

[55] J. Van Mierlo, G. Maggetto, E. Van De Burgwal, and R. Gense. Driving style and traffic measures - Influence on vehicle emissions and fuel consumption. In *Proceedings of the Institution of Mechanical Engineers, Part D: Journal of Automobile Engineering*, volume 218, pages 43–50, jan 2004.

[56] V. N. Vapnik and A. Y. Chervonenkis. On the Uniform Convergence of Relative Frequencies of Events to Their Probabilities. *Theory of Probability & Its Applications*, 16(2):264–280, jan 1971.

[57] D. Vu and A. Slavković. Differential privacy for clinical trial data: Preliminary evaluations. In *ICDM Workshops 2009 - IEEE International Conference on Data Mining*, pages 138–143, 2009.

[58] H. Wallentowitz. Lecture Longitudinal Dynamics of Vehicles.

[59] X. Zhang, R. Ghent, J. Xu, X. Meng, and Y. Xie. Towards accurate Histogram publication under differential privacy. In *SIAM International Conference on Data Mining 2014, SDM 2014*, volume 2, pages 587–595. Society for Industrial and Applied Mathematics Publications, 2014.

[60] P. Zhao, G. Zhang, S. Wan, G. Liu, and T. Umer. A survey of local differential privacy for securing internet of vehicles. *Journal of Supercomputing*, 76(11):8391–8412, nov 2020.

[61] T. Zhu, G. Li, W. Zhou, and P. S. Yu. *Differential Privacy and Applications*. 2017.

[62] S. Zouinina, Y. Bennani, N. Rogovschi, and A. Lyhyaoui. A two-levels data anonymization approach. In *IFIP Advances in Information and Communication Technology*, volume 583 IFIP, pages 85–95. Springer, jun 2020.