Lab – The Laplace Mechanism
Gunnar Yonker

Introduction:

Differential privacy is a fundamental concept in data privacy that aims to protect sensitive information while still allowing useful insights to be extracted from a dataset. In this report, we will explore the implementation of differential privacy mechanisms in Python to safeguard the privacy of a given dataset. The goal of this assignment is to gain hands-on experience in applying differential privacy techniques and making informed decisions regarding privacy parameter selection and sensitive analysis.

Implementation Details:

The implementation of differential privacy hinges on the dp_mechanism function, which was designed to protect the privacy of our dataset. This function calculates the scale parameter b for the Laplace noise using the formula b = sensitivity / epsilon. This scale parameter ensures that the noise added is appropriately scaled to satisfy differential privacy requirements. The noise is added to the true_ans to produce a randomized answer, which is the protected result returned by the function.

```
def dp_mechanism(true_ans, sensitivity, epsilon):

   b = sensitivity / epsilon

   noise = np.random.laplace(0, b)

   rand_ans = true_ans + noise

   return rand_ans
```

dp_mechanism Function: This function is at the heart of the implementation. It employs the Laplace mechanism to introduce controlled randomness into query results while maintaining differential privacy. The function takes three parameters:

true_ans: This represents the true answer to a query without any privacy protection.

sensitivity: Sensitivity quantifies how much the query result can change when a single data point is added or removed from the dataset. It plays a pivotal role in determining the amount of noise to be added. For each query, we calculate sensitivity based on the nature of the query and the dataset's characteristics.

epsilon: Epsilon is a privacy parameter that controls the trade-off between privacy and utility. Smaller values of epsilon provide stronger privacy guarantees but may introduce more noise, potentially reducing utility. Epsilon values are allocated to each query to align with its sensitivity and importance. The total privacy budget is called epsilon_total, and is allocated among the three queries.

```
epsilon_total = 2.5

epsilon_query1 = epsilon_total / 3

epsilon_query2 = epsilon_total / 3

epsilon_query3 = epsilon_total / 3
```

Lab – The Laplace Mechanism
Gunnar Yonker

The implementation incorporates the composition theorem to guarantee that the total privacy budget is not exceeded. This allows us to combine multiple queries while staying within the specified epsilon total of 2.5.

Query Support and Results:

In this section, there are a few examples of the results of the three queries. Each example varies slightly due to the answers being randomized after applying differential privacy. Three query types are supported by this program:

Query 1: Return the number of individuals in the dataset with salaries between $90,000 and $120,000. The aim is to return the number of individuals in the dataset with salaries between the given range. The sensitivity of this query is set to 1, representing the change when one person is added or removed. The code computes the true query result, applies the differential privacy mechanism, and prints both the true and randomized answers.

```
sensitivity_query1 = 1

query1_result = df.loc[(df['Salary'] >= 90000) & (df['Salary'] <= 120000)].shape[0]

dp_query1_result = dp_mechanism(query1_result, sensitivity_query1, epsilon_query1)
```

Query 2: Return the highest salary in the dataset. To calculate sensitivity for Query 2, we find the difference between the current highest salary and the second-highest salary. We then compute the true query result, apply the differential privacy mechanism, and print both results.

```
sorted_salaries = df['Salary'].sort_values(ascending=False)

sensitivity_query2 = sorted_salaries.iloc[0] - sorted_salaries.iloc[1]

query2_result = sorted_salaries.iloc[0]

dp_query2_result = dp_mechanism(query2_result, sensitivity_query2, epsilon_query2)
```

Query 3: Return the median salary in the dataset. We calculate the sensitivity for Query 3 as the maximum change in median value when adding or removing a single individual. We then compute the true query result, apply the differential privacy mechanism, and print both results.

```
sorted_salaries = df['Salary'].sort_values(ascending=True)

num_individuals = len(sorted_salaries)

median_index = num_individuals // 2

sensitivity_query3 = max(

    abs(sorted_salaries.iloc[median_index] - sorted_salaries.iloc[median_index - 1]),
    abs(sorted_salaries.iloc[median_index] - sorted_salaries.iloc[median_index + 1])
)
query3_result = df['Salary'].median()
dp_query3_result = dp_mechanism(query3_result, sensitivity_query3, epsilon_query3)
```

Examples of Query Results Using the Program:

Run 1:

```
Query 1 - True answer: 60
Query 1 - Randomized answer: 58.71315812310854


Query 2 - True answer: 150000
Query 2 - Randomized answer: 150934.36329691036


Query 3 - True answer: 102500.0
Query 3 - Randomized answer: 107061.33148438769


Total privacy budget used: 2.5
```

Run 2:

```
Query 1 - True answer: 60
Query 1 - Randomized answer: 61.07986009790593


Query 2 - True answer: 150000
Query 2 - Randomized answer: 151808.20452394374


Query 3 - True answer: 102500.0
Query 3 - Randomized answer: 101670.06198553287


Total privacy budget used: 2.5
```

Lab – The Laplace Mechanism
Gunnar Yonker

Run 3:

Query 1 - True answer: 60

Query 1 - Randomized answer: 61.391971193647365

Query 2 - True answer: 150000

Query 2 - Randomized answer: 151026.37241439102

Query 3 - True answer: 102500.0

Query 3 - Randomized answer: 103521.48537632027

Total privacy budget used: 2.5

Run 4:

Query 1 - True answer: 60

Query 1 - Randomized answer: 59.79991129629425

Query 2 - True answer: 150000

Query 2 - Randomized answer: 148386.42640752203

Query 3 - True answer: 102500.0

Query 3 - Randomized answer: 103461.62682890019

Total privacy budget used: 2.5

Lab – The Laplace Mechanism
Gunnar Yonker

Privacy Parameter Explanation:

Query 1: We allocate a portion of the total privacy budget, epsilon_query1, to ensure differential privacy. The chosen value was based on equal distribution among queries, aiming for a balanced trade-off between privacy and utility. This value was determined as epsilon_total/3.

Query 2: Similarly, for Query 2, epsilon_query2 was allocated as epsilon_total/3, ensuring an equal share of the budget among queries.

Query 3: For Query 3, epsilon_query3 was also determined as epsilon_total/3, following the same principle as the other queries.

The allocation of epsilon to each query was divided equally among the queries to ensure a fair distribution. This allocation strategy simplifies the process and ensures that no single query consumes the entire budget. As more testing is done, moving forward the epsilon budget could be adjusted between each query if needed. For example, Query 2 could be considered more critical than the others, so it could be allocated a larger share of the budget. However, in this scenario, I chose to equally allocate the budget.

Sensitivity Analysis:

Sensitivity analysis is crucial in differential privacy as it measures how much the output of a query can change when a single individual's data is added or removed. Sensitivity affects the amount of noise introduced to maintain privacy. In this implementation:

Query 1 has a sensitivity of 1, representing a small change when one person is added or removed within the salary range.

Query 2 has a sensitivity equal to the difference between the highest and second-highest salary, capturing the significant change when the top earner changes because changes at the top end of the salary distribution have the highest impact on the result.

Query 3 considers the maximum change in median value when one person is added or removed, ensuring both privacy and accuracy of the median.

Sensitivity analysis helps in selecting appropriate privacy parameters and understanding the impact of data changes on query results. Sensitivity is vital in differential privacy because it quantifies how much a query's output can change when one individual's data is altered, providing a measure of a query's sensitivity to individual contributions. This information guides the amount of noise added to protect privacy, ensuring a balance between privacy and utility. Additionally, sensitivity underpins differential privacy's mathematical guarantees and plays a central role in privacy budget allocation when multiple queries are involved, making it a fundamental concept in preserving individual privacy while allowing for meaningful data analysis.

Lab – The Laplace Mechanism
Gunnar Yonker

Conclusion:

In this lab assignment, the code successfully implements differentially private mechanisms to protect the privacy of the given dataset. Three types of queries were supported while adhering to a total privacy budget of 2.5. The approach involved distributing the budget equally among the queries, conducting sensitivity analysis to determine appropriate sensitivities, and applying Laplace noise to ensure differential privacy. Through this exercise, I gained valuable insights into the practical implementation of differential privacy mechanisms and the trade-offs between privacy and utility.

A key takeaway from this lab is the importance of selecting appropriate privacy parameters and understanding the sensitivity of queries to achieve a balance between privacy protection and accurate query results. Another takeaway for me personally, is that determining the appropriate privacy parameters is a difficult task because there is technically no “right answer” because it is all dependent on various factors. The goal is to have that perfect balance of privacy protection and guarantees, while still having the utility part of the dataset be useful. One last key takeaway that I had from this assignment is that it is crucial to recognize that the data landscape is dynamic. As datasets evolve and new data points are added, the sensitivity of queries may change, and the privacy parameters that we initially chosen may need adjustment. This highlights the need for ongoing monitoring and adaptation to maintain the balance between privacy and utility.