CS394S Homework 2

Juexiao Zhou 175615

Date: Sep 29, 2021

Q1.

Consider any two neighbouring datasets D and D', which differ in only x_i, let event $E : x_i$ is outputed and $\bar{E}$ is the complement event of E, let A to be the Name and Shame mechanism and T be the universe of all outputs.

so we have

$$\begin{aligned} Pr[A(D) \in T] &= Pr[A(D) \in T|\bar{E}]Pr[\bar{E}] + Pr[A(D) \in T|E]Pr[E] \\ &= Pr[A(D') \in T|\bar{E}]Pr[\bar{E}] + Pr[A(D) \in T|E]Pr[E] \\ &\leq Pr[A(D') \in T|\bar{E}]Pr[\bar{E}] + 1 * \delta \\ &\leq Pr[A(D') \in T] + \delta \\ &= e^0 Pr[A(D') \in T] + \delta \end{aligned}$$

So, we have proved the the mechanism A is $(0, \delta)$-DP

Q2.

To show Noisy-max with Laplace Noise is also ϵ-DP. Following the prove in note

Consider the case where $Y = i \in [d]$, we will first fix the noises $Z_1, \ldots, Z_{i-1}, Z_{i+1}, \ldots, Z_d$ and denote it as Z_{-i}

so we show for any neighbouring dataset D and D':

$\mathbb{P}(Y = i|D, Z_{-i}) \leq e^{\epsilon}\mathbb{P}(Y = i|D', Z_{-i})$

if so, we can proof the theorem by taking the integral w.r.t Z_{-i}

To show the above unequality, we define

$z^* = argmin_z q(i; D) + z > q(j; D) + Z_j,\ \forall j \neq i$, which means the smallest value of $q(i; D) + Z_i$ should still be larger than all other values.

Thus we have D and Z_{-j} and the output $Y = i$ if and only if $z_i \geq z^*$

Thus,

$$\begin{aligned} & q(i; D) + z^* \geq q(j; D) + z_j \\ \implies & q(i; D') + \Delta + z^* \geq q(j; D') + z_j - \Delta \\ \implies & q(i; D') + (z^* + 2\Delta) \geq q(j; D') + z_j \end{aligned}$$

Thus, for dataset D' with fixed Z_{-i}, if $z_i \geq z^* + 2\Delta$, then the output for D' will also be i=i

As $Z_i \sim Lap(\frac{\Delta}{\epsilon})$. Then, we have

$$
\begin{aligned}
\mathbb{P}(Y=i|D',Z_{-i}) &\geq \mathbb{P}(z_i \geq z^* + 2\Delta) \\
&= \int_{z^*+2\Delta}^{\infty} Lap(\frac{\Delta}{\epsilon})dy \\
&= \int_{z^*+2\Delta}^{\infty} \frac{\epsilon}{2\Delta} exp(-\frac{\epsilon|y|}{\Delta})dy \\
&= \int_{z^*+2\Delta}^{\infty} \frac{\epsilon}{2\Delta} exp(-\frac{\epsilon y}{\Delta})dy \\
&= e^{-\epsilon} \int_{z^*}^{\infty} \frac{\epsilon}{2\Delta} exp(-\frac{\epsilon y}{\Delta})dy \\
&= e^{-\epsilon}\mathbb{P}(Y=i|D,Z_{-i})
\end{aligned}
$$

So, we have proved $\mathbb{P}(Y=i|D,Z_{-i}) \leq e^{\epsilon}\mathbb{P}(Y=i|D',Z_{-i})$

Then, by integrating w.r.t Z_{-i}, we can show Noisy-max with Laplace Noise is also ϵ-DP

Q4.

Here we define an election task as the selection problem:

A election task with $Y = \{y_1, y_2, \ldots, y_n\}$ candidates, each candidate has votes stored in the database as $D = \{x_1, x_2, \ldots x_n\}$. We need to find the candidate with the highest votes among all candidates.

Output space: any y in Y

Score function: $q(y:D)$ if y is the real candidate with the highest votes, then return 10, otherwise 0.

We know the sensitivity of the score function is 10.

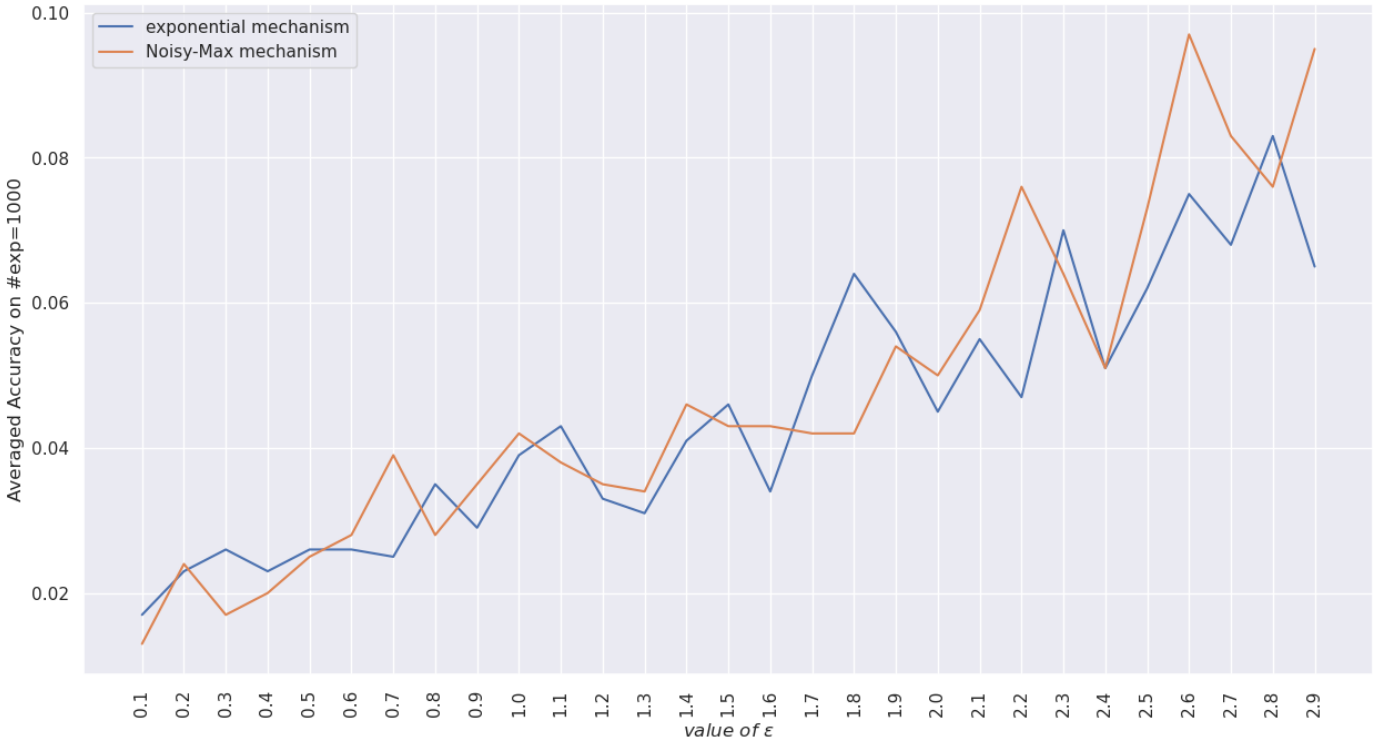
For epsilon in range(0.1,3,0.1), we run independent experiment 1000 times. For each time, we generate a database with size 50, which follow the Gaussian distribution with $\mu = 100$ and output the answer with either exponential mechanism or noisy-max mechanism. For each epsilon, we calculated the accuracy among 1000 experiments to count for the percentage of correct answers.

Here is the code for all experiments:

```
import numpy as np
from math import exp

# Task definition: D: votes for all candiates (follow normal distribution, int), f(D): output the candidate
with the highest votes, for each y in candidates, q(y;D)=1 if y has the highest votes, otherwise 0

def generate_database(mu, n):
    return [np.floor(np.abs(x)) for x in np.random.normal(mu,100,n)]

def f(database):
    return database.index(max(database))

def q(y, database):
    real_answer=database.index(max(database))
    if y==real_answer:
        return 10
    else:
        return 0
```

```
# from the setting, we know the Delta=1
def exponential_mechanism(database, epsilon, Delta=1):
    P=[]
    for idx in range(len(database)):
        P.append(exp((epsilon/(2*Delta))*q(idx,database)))
    P/=np.sum(P)
    return np.random.choice(range(len(database)),p=P)

def noisy_max_with_Exp(database, epsilon, Delta=1):
    new_database=[]
    for idx in range(len(database)):
        new_data = q(idx, database) + np.random.exponential((2*Delta)/epsilon)
        new_database.append(new_data)
    return f(new_database)

epsilons=np.arange(0.1,3,0.1)
#epsilons=[0.1]
delta=10
repeats=1000
for epsilon_idx in range(len(epsilons)):
    exec('real_{}=[]'.format(epsilon_idx))
    exec('exp_{}=[]'.format(epsilon_idx))
    exec('noisy_{}=[]'.format(epsilon_idx))
    epsilon=epsilons[epsilon_idx]
    for repeat in range(repeats):
        database=generate_database(100,50)
        #print(database)
        #print('Real answer: {}'.format(f(database)))
        #print('Exponential Mechanism:
{}'.format(exponential_mechanism(database,epsilon=epsilon,Delta=delta)))
        #print('Noisy-Max Mechanism: {}'.format(noisy_max_with_Exp(database,epsilon=epsilon,Delta=delta)))
        eval('real_{}'.format(epsilon_idx)).append(f(database))
        eval('exp_{}'.format(epsilon_idx)).append(exponential_mechanism(database,epsilon=epsilon,Delta=delta))
        eval('noisy_{}'.format(epsilon_idx)).append(noisy_max_with_Exp(database,epsilon=epsilon,Delta=delta))
    #print('epsilon: {}'.format(epsilon))
    #print(eval('real_{}'.format(epsilon_idx)))
    #print(eval('exp_{}'.format(epsilon_idx)))
    #print(eval('noisy_{}'.format(epsilon_idx)))

def accuracy(x,y):
    count=0
    for i in range(len(x)):
        if x[i]==y[i]:
            count+=1
    return count/len(x)

exp_accs=[accuracy(eval('real_{}'.format(epsilon_idx)),eval('exp_{}'.format(epsilon_idx))) for epsilon_idx in
range(len(epsilons))]
noisy_accs=[accuracy(eval('real_{}'.format(epsilon_idx)),eval('noisy_{}'.format(epsilon_idx))) for epsilon_idx
in range(len(epsilons))]
print(exp_accs,noisy_accs)
```

Here is the result:



From this result, we observed:

1. the performance of the exponential mechanism and the noisy-max mechanism is very similar.
2. Under our setting, the average accuracy is not stable, but we can still find the increasing trend when ϵ increases.
3. When the value of ϵ increases, the average accuracy will also increase, which means the increasing of ϵ -> less noise added -> better utility (less privacy)

Q5.

We generate the database by randomly choosing from {0,1} with equal probability. As we know the $f(D)$ is a vector with size d, so the global sensitivity of $f(D)$ is d/n.

We define the estimation error as the averaged L1-norm between the real answers and the answers given by both mechanisms. Therefore, larger estimation error means better accuracy but less utility.

We compare the laplacian mechanism under the setting of ϵ-DP and the gaussian mechanism under the setting of $(\epsilon, 0.001)$-DP.

Here is the code of all functions:

```
import numpy as np
from math import exp

def generate_database(n,d):
    database=[]
    for i in range(n):
        database.append(np.random.choice([0,1], replace = True, size=d, p=[0.5,0.5]))
    return database
```

```
# f(D)=mean(D)

def f(D):
    return np.mean(D,axis=0)

# as we know the f(D) is a vector of dimension d, GS=||f(D)-f(D')||_1=d/n

# laplacian mechanism
def laplacian_mechanism(d, epsilon, database, GS):
    _result = f(database)
    _noise = np.random.laplace(0, GS/epsilon, d)
    return _result + _noise[0]

# gaussian mechanism
def gaussian_mechanism(d, epsilon, database, GS, delta=0.001):
    _result = f(database)
    _noise = np.random.normal(0, 32*GS*GS*np.log(2/delta)/(9*epsilon*epsilon), d)
    return _result + _noise[0]

def EstimationError(A,B):
    #print('A: {}'.format(A))
    #print('B: {}'.format(B))
    return np.abs(np.subtract(A, B)).mean()
```

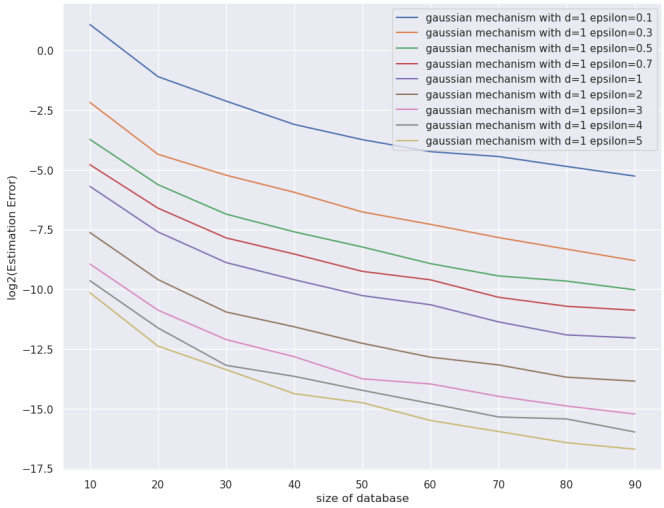
We tested all n from range(10,100,10), d from [1,10,100,1000] and ϵ from [0.1,0.3,0.5,0.7,1,2,3,4,5]. For each independent experiment, we repeat 100 times to calculate the averaged estimation error.

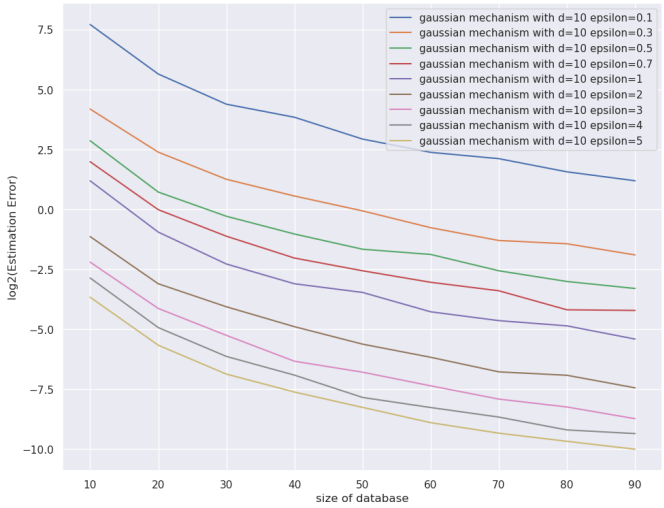
Here is the code for experiments:

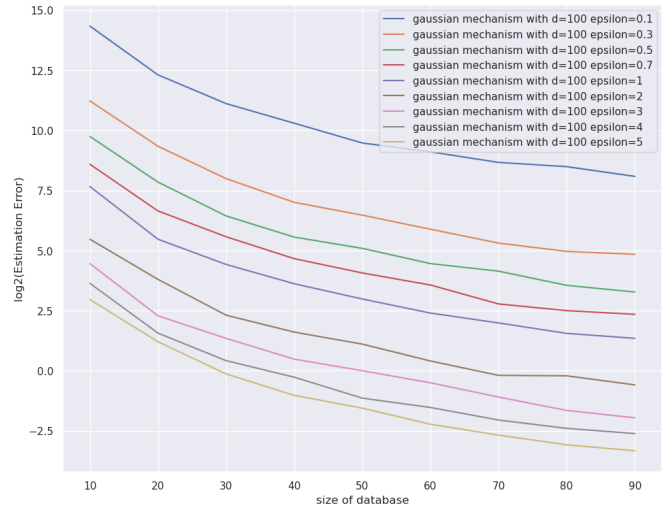
```
ns=np.arange(10,100,10)
ds=[1,10,100,1000]
epsilons=[0.1,0.3,0.5,0.7,1,2,3,4,5]
repeats=100

for d in ds:
    for epsilon in epsilons:
        print(d,epsilon)
        exec('lap_{}_{}_MSEs=[]'.format(d,epsilons.index(epsilon)))
        exec('gau_{}_{}_MSEs=[]'.format(d,epsilons.index(epsilon)))
        for n in ns:
            lap_tmp=[]
            gau_tmp=[]
            for repeat in range(repeats):
                database=generate_database(n=n,d=d)
                lap_tmp.append(EstimationError(f(database),laplacian_mechanism(d,epsilon,database,GS=d/n)))
                gau_tmp.append(EstimationError(f(database),gaussian_mechanism(d,epsilon,database,GS=d/n)))
            eval('lap_{}_{}_MSEs'.format(d,epsilons.index(epsilon))).append(np.mean(lap_tmp))
            eval('gau_{}_{}_MSEs'.format(d,epsilons.index(epsilon))).append(np.mean(gau_tmp))
```

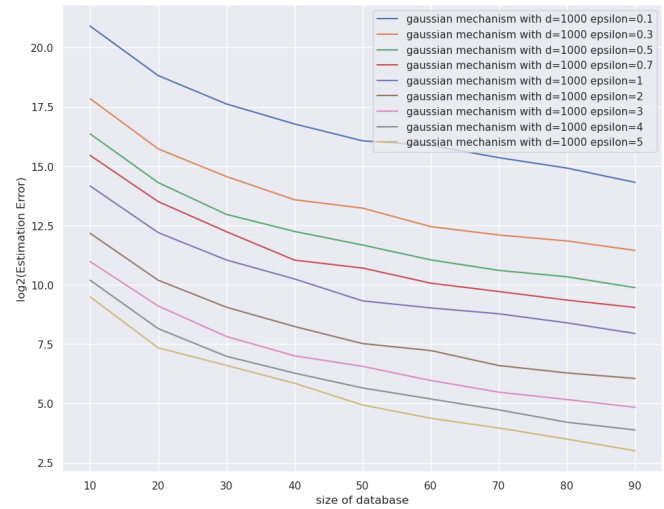
Here is the result:

- Gaussian mechanism with δ=1

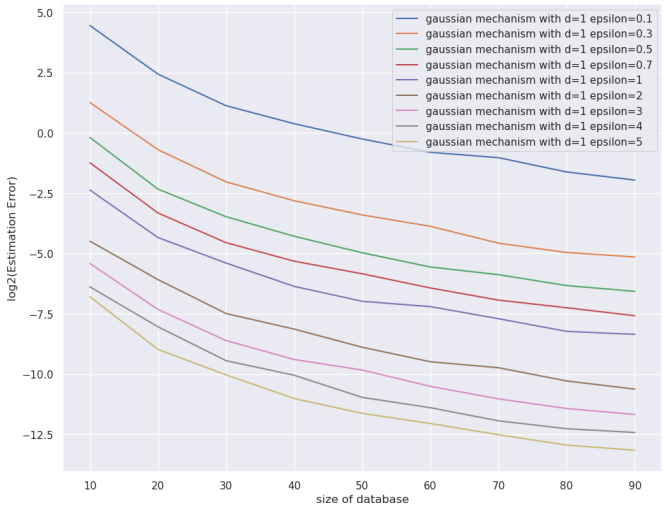


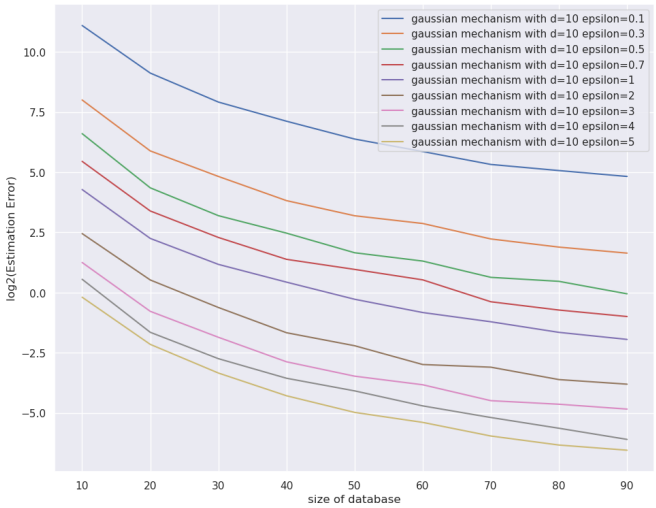


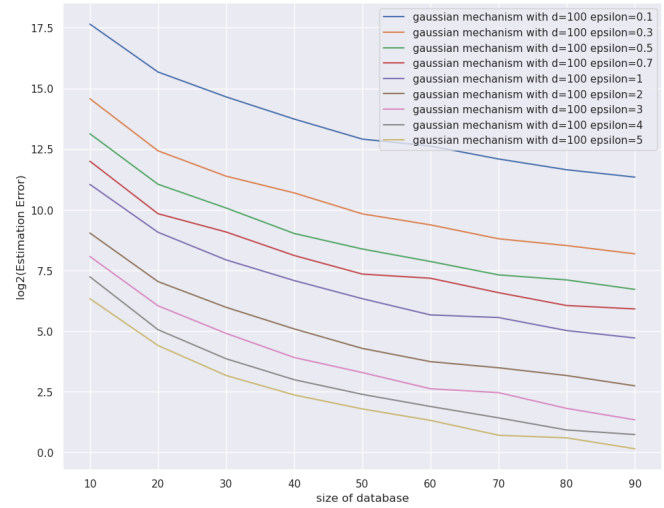


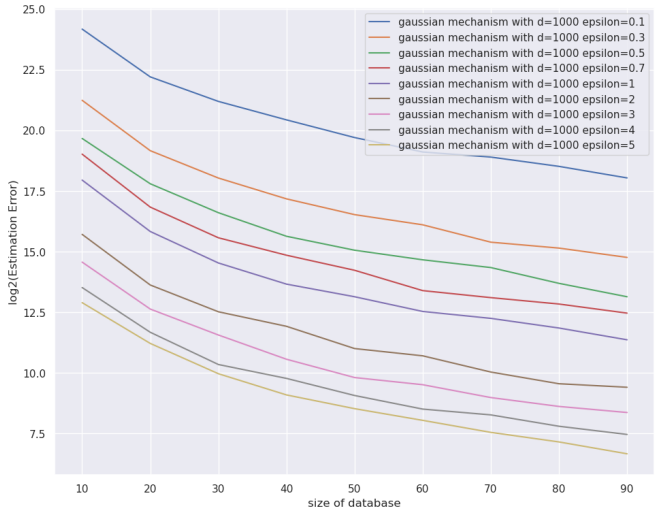


- Gaussian mechanism with δ=0.001

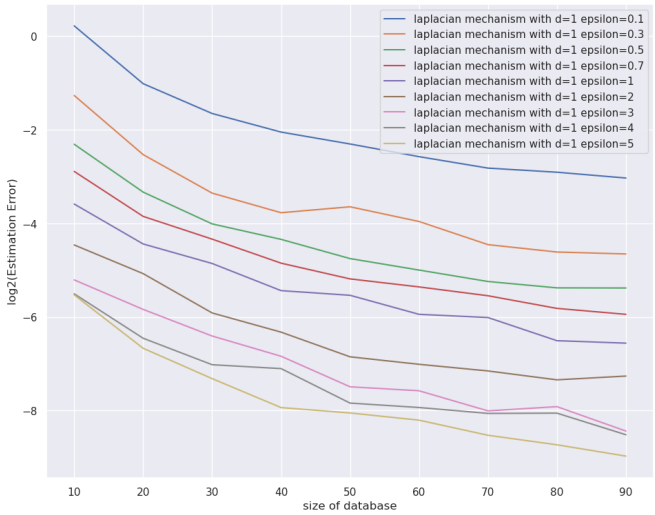


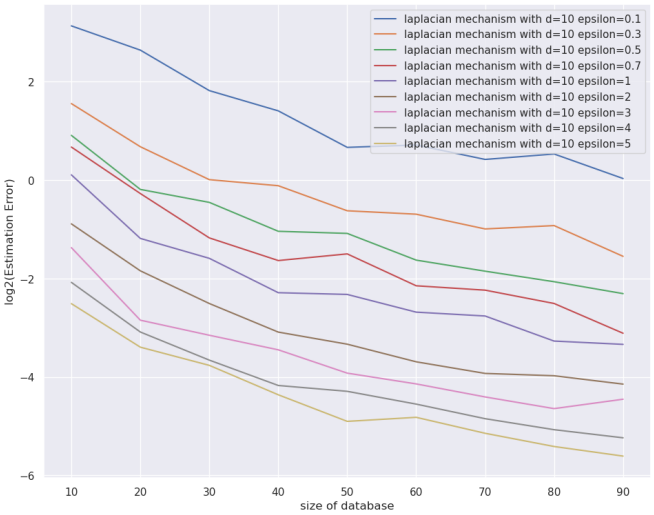


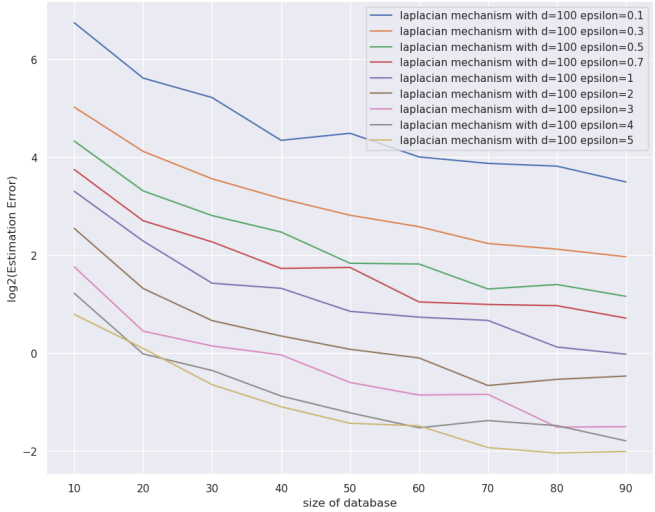


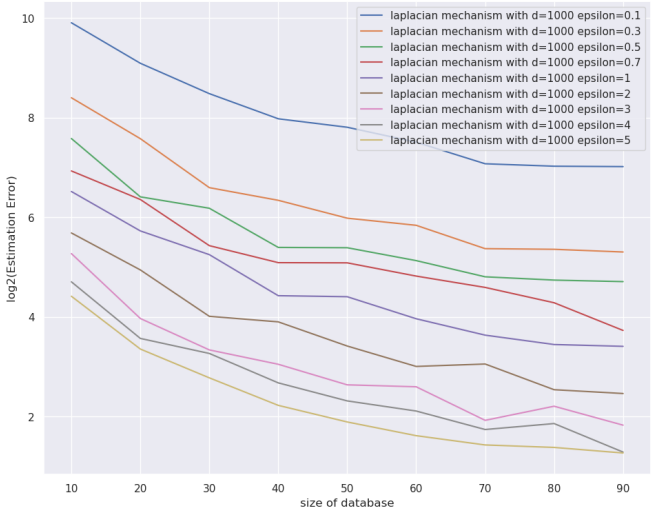


- Laplacian Mechanism









From the result, we observed:

1. For all individual cases, when the size of the database (n) increases, the log2(estimation error) will decrease.
2. With the size of databases (n) and d fixed, for both the laplacian mechanism and the gaussian mechanism, the log2(estimation error) decreases when ϵ increases.
3. With the size of databases (n) and epsilon fixed, for both the laplacian mechanism and the gaussian mechanism, the log2(estimation error) increases when d increases.
4. With all parameters fixed, the laplacian mechanism seems to give smaller log2(estimation error) compared to the gaussian mechanism (δ=1), however, for δ=0.001, the gaussian mechanism leads to a much higher log2(estimation error).