COMP430 PROJECT2 REPORT

Name-Surname: Barış KAPLAN
KU ID Number: 0069054

Part 2:

Task 1:

a)

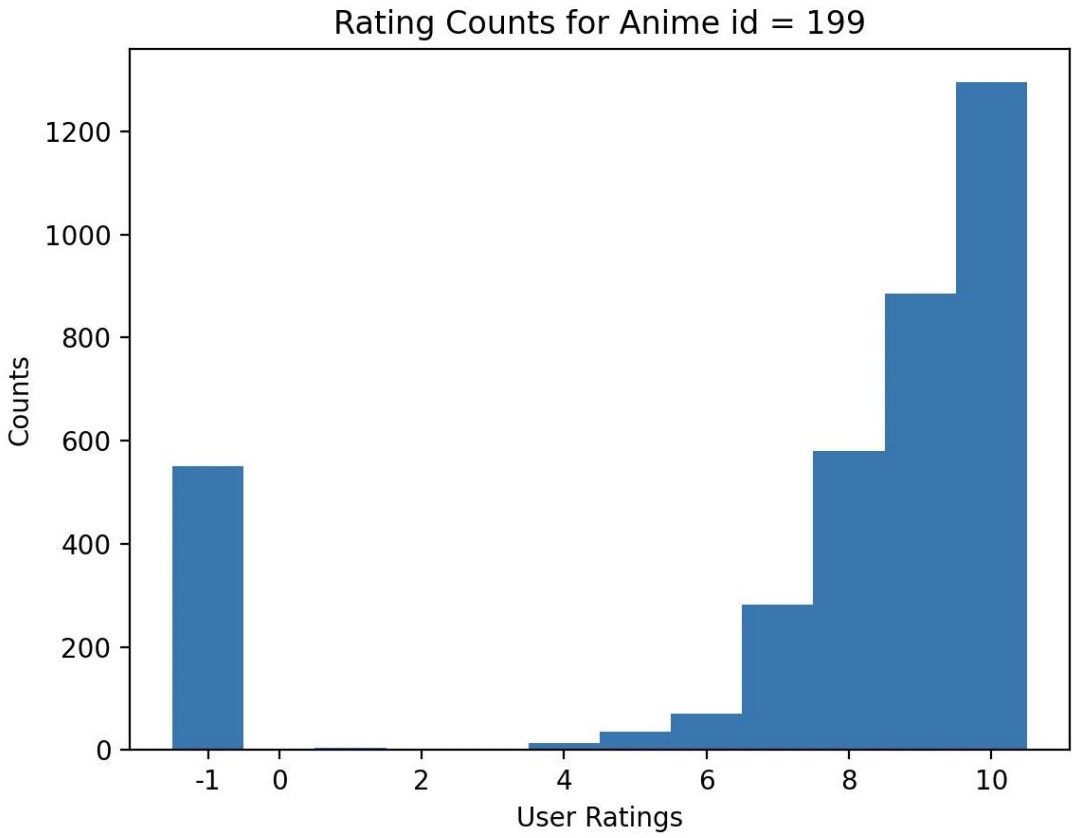


Figure 1: *My non-private histogram that I have obtained as a result of the part a of the task1 of Part 2*

d)

For Average Error:

Value Of Epsilon	Value Of Error
0.0001	21240.501543493512
0.001	1930.1887268792805
0.005	391.70665638738365
0.01	189.28140785665764
0.05	41.44396338015043
0.1	19.291770619567245
1.0	2.032834232192712

For Mean Squared Error:

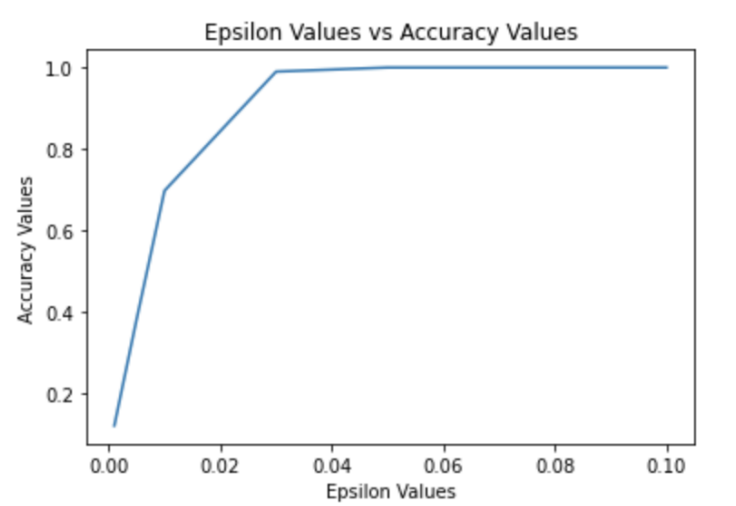
Value Of Epsilon	Value Of Error
0.0001	880321014.1476209
0.001	7216308.510779086
0.005	311455.90166564623
0.01	77191.90248633853
0.05	3382.849047341285
0.1	758.1427218813619
1.0	7.972391878245496

My discussion about the relationship between the value of epsilon and the value of error:
For both the average error and the mean squared error, I have observed the same relation between the value of epsilon and the value of error. For both of the errors, I have observed that as the value of epsilon increases, the value of error decreases. In other words, I have observed that the value of error is inversely proportional to the value of epsilon. This relation can also be clearly seen from the above tables. When I increase the value of epsilon, as expected, I have observed more alterations (changes) in Mean Squared Error compared to Average Error.

Task 2:

Epsilon Value	Accuracy
0.001	0.122
0.005	0.383
0.01	0.698
0.03	0.99
0.05	1.0
0.1	1.0

My discussion about the relationship between the value of epsilon and the accuracy:
As the value of epsilon increases, I have observed that the value of accuracy also increases. In other words, I have observed that the value of accuracy is directly proportional to the value of epsilon. The relation between the epsilon values and accuracy can be observed from the above table and from the below graph. For the epsilon values between 0.001 and 0.03 (0.001, 0.005, 0.01, 0.03); when I increase the epsilon value, I have observed higher changes in the accuracy. However; for the epsilon values between 0.03 and 0.1 (0.03, 0.05, 0.10); when I increase the epsilon value, I observed much smaller changes in the value of accuracy.



The graph which shows the relations between the epsilon values and the accuracy values

PART3

TABLE FOR GRR PROTOCOL:

Epsilon Values	Error Values of GRR
0.1	19797.06
0.5	19120.71
1.0	18049.53
2.0	14402.24
4.0	4788.59
6.0	799.55

TABLE FOR RAPPOR PROTOCOL:

Epsilon Values	Error Values of RAPPOR
0.1	209307.35
0.5	188775.47
1.0	163481.53
2.0	117028.00
4.0	52169.65
6.0	20791.82

TABLE FOR OUE PROTOCOL:

Epsilon Values	Error Values of OUE
0.1	10831.03
0.5	2131.78
1.0	1330.88
2.0	512.75
4.0	212.96
6.0	145.08

My discussion about the relation between the value of epsilon and the value of protocol error:

For all of the protocols (GRR, RAPPOR, and OUE), I have observed that as the value of epsilon increases (starting from 0.1 and ending at 6.0), the error value of the protocol decreases. In other words, I have observed that the error value of each protocol is **inversely proportional** to the value of epsilon. Moreover; for all protocols, as I increase the epsilon value, I observed that the percentage of change in the error value becomes higher. Yes, **OUE (Optimized Unary Encoding) is better** among all protocols for all epsilon values. Because for each epsilon value, OUE has the **lowest** amount of error (compared to GRR and RAPPOR). The estimation formula of OUE, the bit flipping probability of OUE, and the bit preservation probability of OUE **are not depended on the domain size (Here, domain size = d)**. In other words, the OUE protocol **does not care how small or how large the domain is.** Therefore, the perturbation and estimation results of OUE **are also not depended** on the domain size (Here, the domain size = d). This is an improvement over the GRR Protocol. In the GRR Protocol, the probability of heads, and the probability of tails are depended on the value of the domain size (Here, the domain size = d). The probability of heads (reporting true value) in GRR is inversely proportional to the domain size. In other words; as the domain size increases, the probability of reporting true value (the probability of heads) decreases. Moreover, the probability of tails in GRR is also inversely proportional to the domain size. In other words; as the domain size increases, the probability of tails decreases. By considering these, we can say that the **GRR protocol does not execute/operate efficiently and effectively in large domains.** In other words, **the performance and results of the GRR Protocol are not well inside the large domains. The bit preservation and bit flipping probabilities of RAPPOR are also not depended on the domain size (Here, the domain size = d). However, the OUE Protocol involves more randomization than the RAPPOR and GRR since the 1 bits and 0 bits are unequally/unevenly treated in the perturbation process of OUE.** The unequal treatment of 1 bits and 0 bits mean that the bit flipping probabilities and the bit preservation probabilities change depending on which bit (0 or 1) the protocol will perturb. This is also a reason why OUE works better than the GRR and RAPPOR. In order to better understand my above discussion; you can see the formulas I followed while implementing GRR, OUE, and RAPPOR, and the relationships of the formulas I used with the epsilon and d.

$$p = \frac{e^{\varepsilon}}{e^{\varepsilon}+d-1}$$

The formula I used for the true situation probability (namely p) of GRR ***(Screenshot taken from COMP430 Lecture08 Notes)***

$$q = \frac{1-p}{d-1}$$

The formula I used for the false/fake situation probability (namely q) of GRR ***(Screenshot taken from COMP430 Data Privacy & Security Lecture08 Notes)***

$$\Pr[B'_\ell[i] = 1] = \begin{cases} \frac{e^{\varepsilon/2}}{e^{\varepsilon/2}+1} & \text{if } B_\ell[i] = 1 \\ \frac{1}{e^{\varepsilon/2}+1} & \text{if } B_\ell[i] = 0 \end{cases}$$

→ **Rappor**

$$\Pr[B'_\ell[i] = 1] = \begin{cases} \frac{1}{2} & \text{if } B_\ell[i] = 1 \\ \frac{1}{e^{\varepsilon}+1} & \text{if } B_\ell[i] = 0 \end{cases}$$

→ **Oue**

The flipping and preservation probabilities I used for the OUE and RAPPOR Protocols ***(This screenshot is taken from the COMP430 Data Privacy & Security Lecture08 Notes)***

$$E[I_v] = n_v \cdot p + (n - n_v) \cdot q$$

The formula I used in GRR and Rappor for calculating expected value of Iv ***(This screenshot is taken from the COMP430 Data Privacy & Security Lecture08 Notes)***

$$\bar{C}(i) = \frac{2 \cdot \left((e^{\varepsilon}+1) \cdot \widehat{C}(i) - n\right)}{e^{\varepsilon}-1}$$

The estimation formula I used for estimating OUE. ***(this screenshot is taken from the COMP430 Data Privacy & Security Lecture Notes)***

$$c(v) = \frac{I_v - n \cdot q}{p-q}$$

The estimation formula I used for estimating GRR and Rappor ***(this screenshot is taken from the COMP430 Data Privacy & Security Lecture Notes)***

THE OUTPUTS I GOT FROM THE PART2 AND PART3

Note: The output screenshots I provided in this page and the outputs I provided in the above tables of Part-2 and Part-3 are the results of different runs. Moreover; the error calculations, exponential mechanism, and the implementations of the GRR, OUE, and RAPPOR Protocols involve probabilities / probabilistic events and the implementation of the get_dp_histogram() function involves randomized laplace noises. Due to these factors, the output screenshots (for Part-2 and Part-3) which I provided in this page may not be exactly same with the Part-2 and Part-3 outputs which I provided in the above tables of Part-2 and Part-3. However, the output screenshots (for Part-2 and Part-3) which I provided in this page may not match with the Part-2 and Part-3 outputs which I provided in the above tables of Part-2 and Part-3.

```
/usr/bin/python3 /Users/barissss/Desktop/HW2 2/part2_skeleton.py
User Ratings: [10.0, 9.0, 8.0, -1.0, 7.0, 6.0, 5.0, 4.0, 1.0, 2.0, 3.0]
Counts of user ratings for anime id = 199: [1295, 886, 580, 551, 282, 70, 35, 14, 5, 1, 1]
**** LAPLACE EXPERIMENT RESULTS ****
**** AVERAGE ERROR ****
eps =  0.0001  error =  19462.094462711255
eps =  0.001  error =  2155.435713576079
eps =  0.005  error =  402.716550912368
eps =  0.01  error =  205.1605175750479
eps =  0.05  error =  37.86950923869892
eps =  0.1  error =  19.177475293721578
eps =  1.0  error =  1.913037964134076
**** MEAN SQUARED ERROR ****
eps =  0.0001  error =  736613269.121948
eps =  0.001  error =  9848089.61744446
eps =  0.005  error =  319856.63735415484
eps =  0.01  error =  87917.35981045604
eps =  0.05  error =  2595.5087523088014
eps =  0.1  error =  731.1326787714652
eps =  1.0  error =  7.323284396696012
**** EXPONENTIAL EXPERIMENT RESULTS ****
eps =  0.001  accuracy =  0.102
eps =  0.005  accuracy =  0.351
eps =  0.01  accuracy =  0.651
eps =  0.03  accuracy =  0.987
eps =  0.05  accuracy =  0.999
eps =  0.1  accuracy =  1.0

Process finished with exit code 0
```

Output Screenshot 1: *The outputs I obtained from the Part-2*

```
/usr/bin/python3 /Users/barissss/Desktop/HW2 2/part3_skeleton.py
GRR EXPERIMENT
e=0.1, Error: 19749.53
e=0.5, Error: 19091.88
e=1.0, Error: 17981.76
e=2.0, Error: 14451.29
e=4.0, Error: 4763.29
e=6.0, Error: 795.76
************************************************
RAPPOR EXPERIMENT
e=0.1, Error: 209375.00
e=0.5, Error: 188645.76
e=1.0, Error: 163385.65
e=2.0, Error: 116984.76
e=4.0, Error: 52283.65
e=6.0, Error: 20853.94
************************************************
OUE EXPERIMENT
e=0.1, Error: 8185.69
e=0.5, Error: 2520.49
e=1.0, Error: 890.13
e=2.0, Error: 464.73
e=4.0, Error: 200.08
e=6.0, Error: 133.73

Process finished with exit code 0
```

Output Screenshot 2: *The outputs I obtained from the Part-3*

IMPORTANT NOTE: Please read the comments I have written inside the get_histogram() and get_dp_histogram() functions. These comments are about where to comment for observing noisy and actual histograms and what to do in order to see error results (MSE & Average Error) and exponential mechanism results after observing histograms. You can find more details in the comments I have written inside the get_histogram() and get_dp_histogram() function. Using the "plt.show()" function before the return statement in a function somehow interrupts the execution of that function / the entire program by nature (Using the "plt.show()" function before the return statement in a function does not bring any errors or warnings to the execution of that function or the entire program). Thus, for only observing the error results (MSE & Average Error) and the exponential mechanism results; you should comment all the source code lines , which are related to constructing, plotting, and drawing histograms , inside the get_histogram() and get_dp_histogram() function. In other words, you should comment all the source code lines which are based on matplotlib.pyplot library of python and which have plt prefixes in front of them. For individually observing noisy histograms or actual histograms, you can refer to the comments I have written inside the get_histogram() and get_dp_histogram() functions to understand what to do.

Name – Surname: Barış KAPLAN
KU Login: bkaplan18
KU ID Number: 0069054 (69054)
PROJECT NUMBER: PROJECT #2