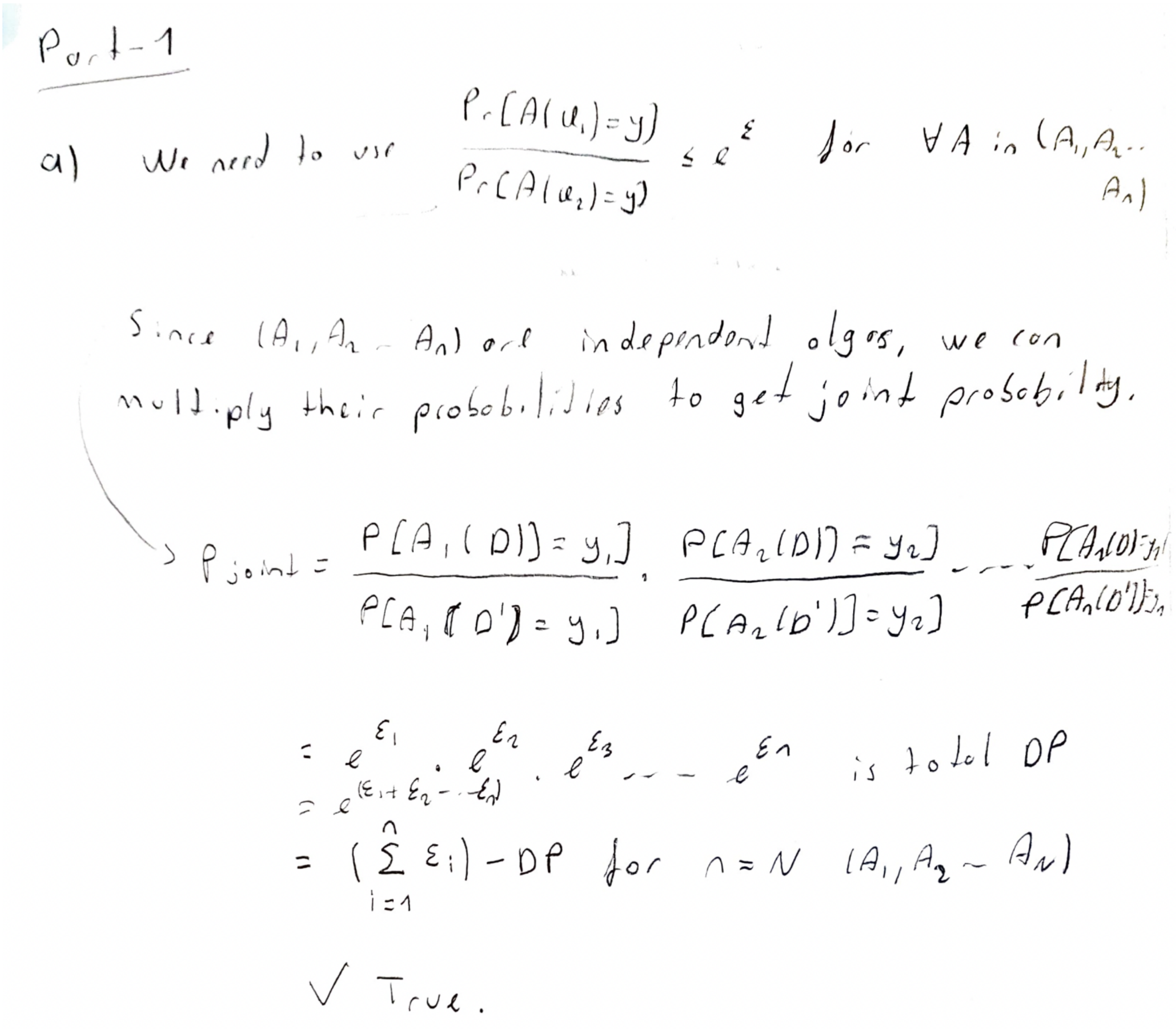
Privacy HW-2 Report

Berke Can Rizai - 69282

1 -



Part-1

a) We need to use $\frac{Pr[A(d_1)=y]}{Pr[A(d_2)=y]} \le e^{\varepsilon}$ for $\forall A$ in $(A_1, A_2 .. A_n)$

Since $(A_1, A_2 - A_n)$ are independent algos, we can multiply their probabilities to get joint probabilty.

$$\to P_{joint} = \frac{P[A_1(D)] = y_1]}{P[A_1[D'] = y_1]} \cdot \frac{P[A_2(D)] = y_2]}{P[A_2(D')] = y_2]} - - - \frac{P[A_n(D)=y_n]}{P[A_n(D')]=y_n}$$

$$= e^{\varepsilon_1} \cdot e^{\varepsilon_2} \cdot e^{\varepsilon_3} - - - e^{\varepsilon_n} \text{ is total DP}$$

$$= e^{(\varepsilon_1 + \varepsilon_2 - - \varepsilon_n)}$$

$$= \left(\sum_{i=1}^{n} \varepsilon_i\right) - DP \text{ for } n = N \ (A_1, A_2 \sim A_N)$$

✓ True.

b) No, it does not satisfy because,

Lets change the constant number e^{ε} with some number such as 15, since it wouldn't affect the problem.

Assume two neighbouring datasets are defined as remove, add single record.

We have D → } 15 and D' → } 14

that we got by removing a record.

With $\frac{P[A(D)=O]}{P[A(D')=O]} \le e^{\varepsilon}$ lets say A checks if record count is ≥ 15. And, O = 'Large'

$\frac{P[A(D) = \text{'large]}}{P[A(D') = \text{'large')}} = \frac{1}{0}$ since in D' we have 14 records, algorithm will exclusively return 'small' query on D'.

Since $\underbrace{\frac{1}{0}}_{\infty} > e^{\varepsilon}$ this doesn't satisfy DP.

c) In the original laplace mechanism, we add noise Laplace$\left(\frac{sens}{\varepsilon}\right)$ but here noise we add with function is Laplace(ε). We nee to prove this also holds.

From formal definition we should have $\frac{s(q)}{\varepsilon} = \varepsilon$

so sensitivity $s(q) = \varepsilon^2$

$$\frac{P[A(D)=0]}{P[A(D')=0]} \le e^{\varepsilon}$$

$$\longrightarrow P[r = 0 - q(D)] \rightarrow \frac{1}{2b} \cdot e^{-\frac{|0-q(D)|}{b}}$$

$$\longrightarrow P[r^k = 0 - q(D')] \rightarrow \frac{1}{2b} \cdot e^{-\frac{|0-q(D')|}{b}}$$

$$\downarrow$$

denom $\rightarrow e^{-\frac{|0-q(D')|}{\varepsilon}}$

$$e^{\cdot \frac{|q(D)-q(D')|}{s(q)}} \le e^{\varepsilon} \rightarrow e^{\frac{|q(D)-q(D')|}{s(q)}}$$

$$\downarrow$$

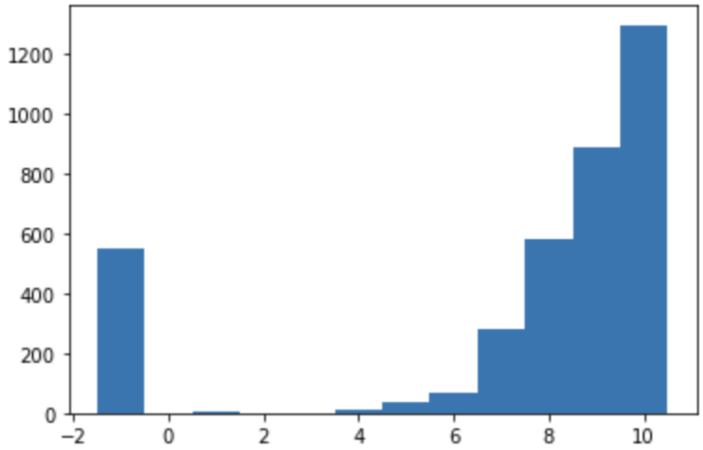
$$e^{\frac{|q(D)-q(D')|}{\varepsilon}} \quad s(q)$$

$$\downarrow$$

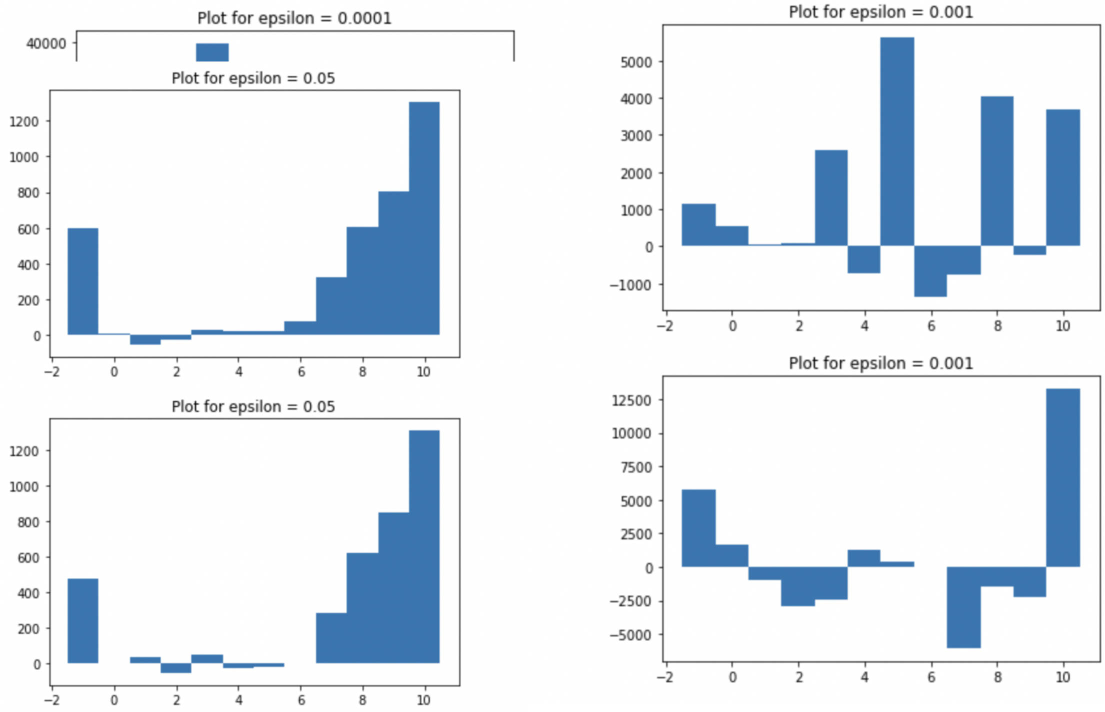
$e^{\left(\frac{\varepsilon^2}{\varepsilon}\right)} = e^{\varepsilon}$ which is $\le e^{\varepsilon}$ and it is DP.

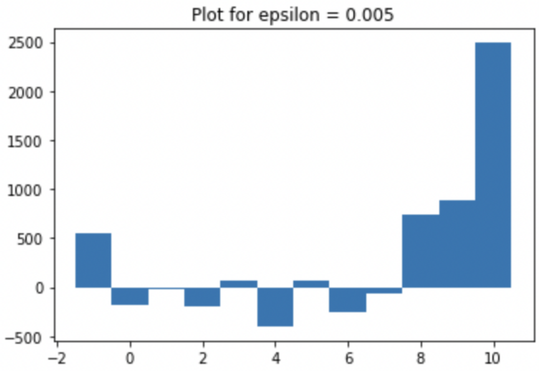
2-

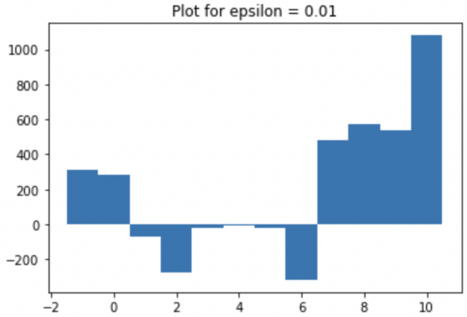
a) Histogram without privacy for the default anime. Scores and corresponding counts.

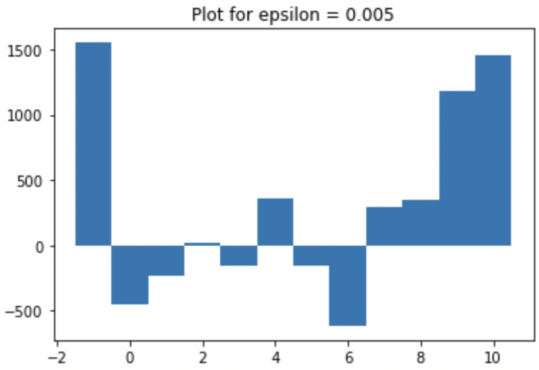


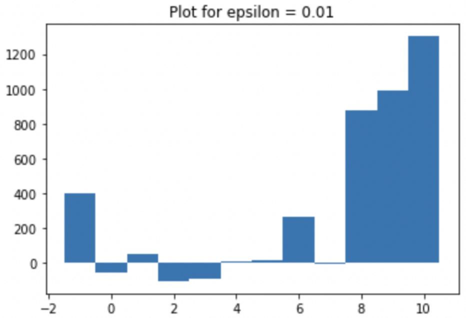
Some example histograms starting with epsilon = 0.0001 and other epsilon values.

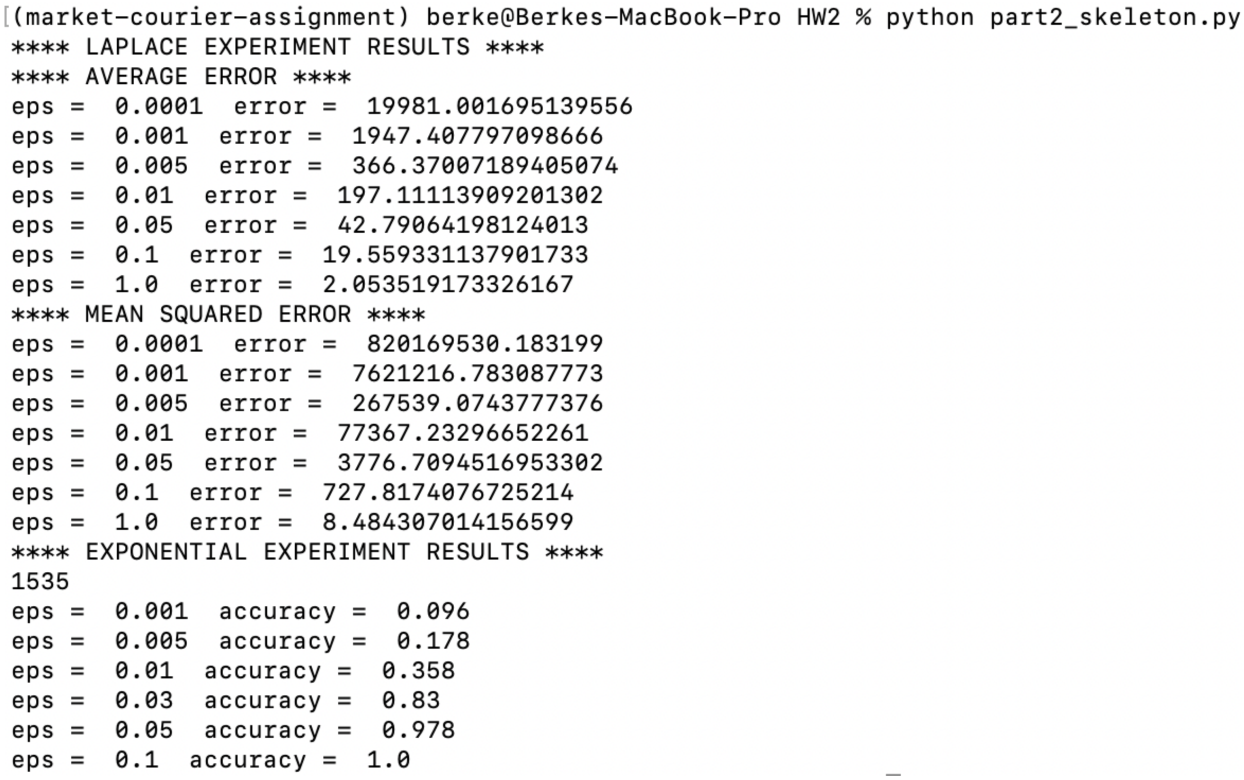



Plot for epsilon = 0.005


Plot for epsilon = 0.01


Plot for epsilon = 0.005

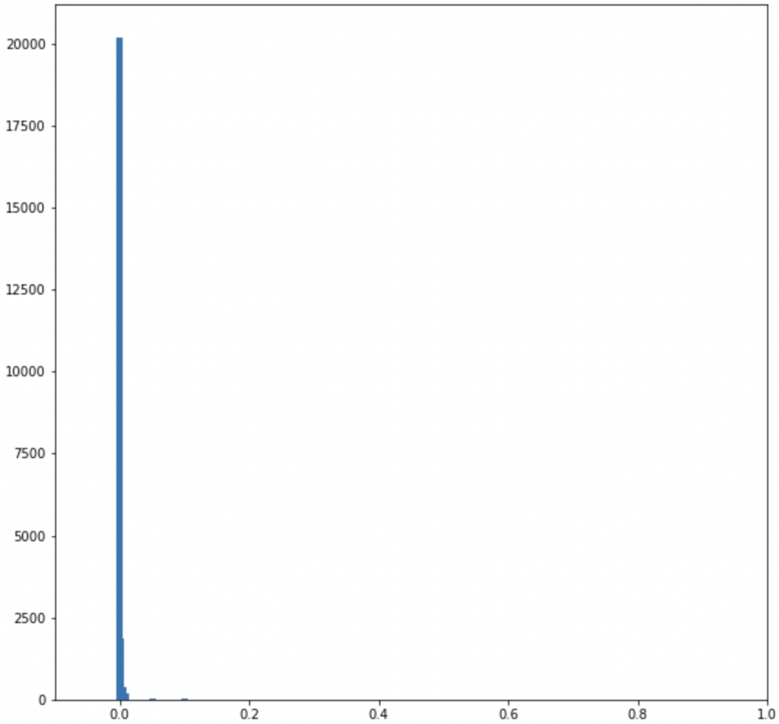

Plot for epsilon = 0.01



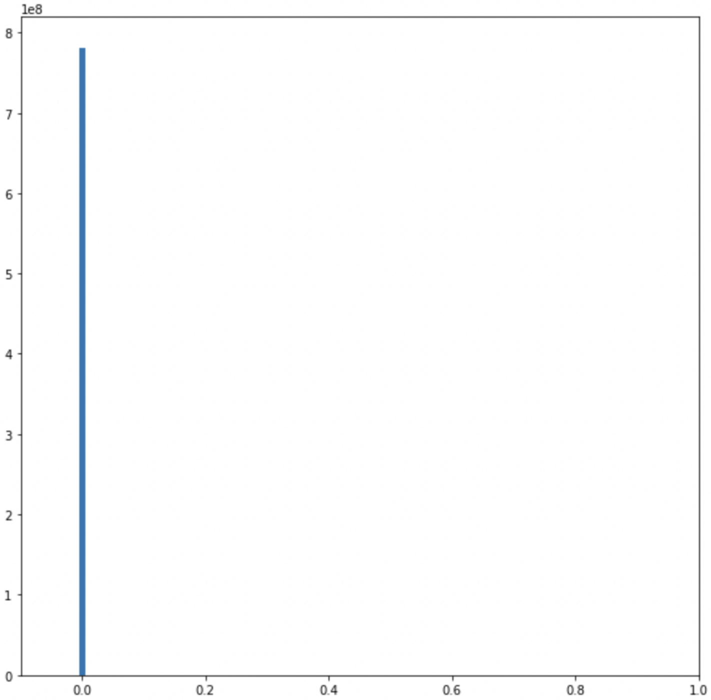
```
[(market-courier-assignment) berke@Berkes-MacBook-Pro HW2 % python part2_skeleton.py
 **** LAPLACE EXPERIMENT RESULTS ****
 **** AVERAGE ERROR ****
 eps =  0.0001  error =  19981.001695139556
 eps =  0.001  error =  1947.407797098666
 eps =  0.005  error =  366.37007189405074
 eps =  0.01  error =  197.11113909201302
 eps =  0.05  error =  42.79064198124013
 eps =  0.1  error =  19.559331137901733
 eps =  1.0  error =  2.053519173326167
 **** MEAN SQUARED ERROR ****
 eps =  0.0001  error =  820169530.183199
 eps =  0.001  error =  7621216.783087773
 eps =  0.005  error =  267539.0743777376
 eps =  0.01  error =  77367.23296652261
 eps =  0.05  error =  3776.7094516953302
 eps =  0.1  error =  727.8174076725214
 eps =  1.0  error =  8.484307014156599
 **** EXPONENTIAL EXPERIMENT RESULTS ****
 1535
 eps =  0.001  accuracy =  0.096
 eps =  0.005  accuracy =  0.178
 eps =  0.01  accuracy =  0.358
 eps =  0.03  accuracy =  0.83
 eps =  0.05  accuracy =  0.978
 eps =  0.1  accuracy =  1.0
```

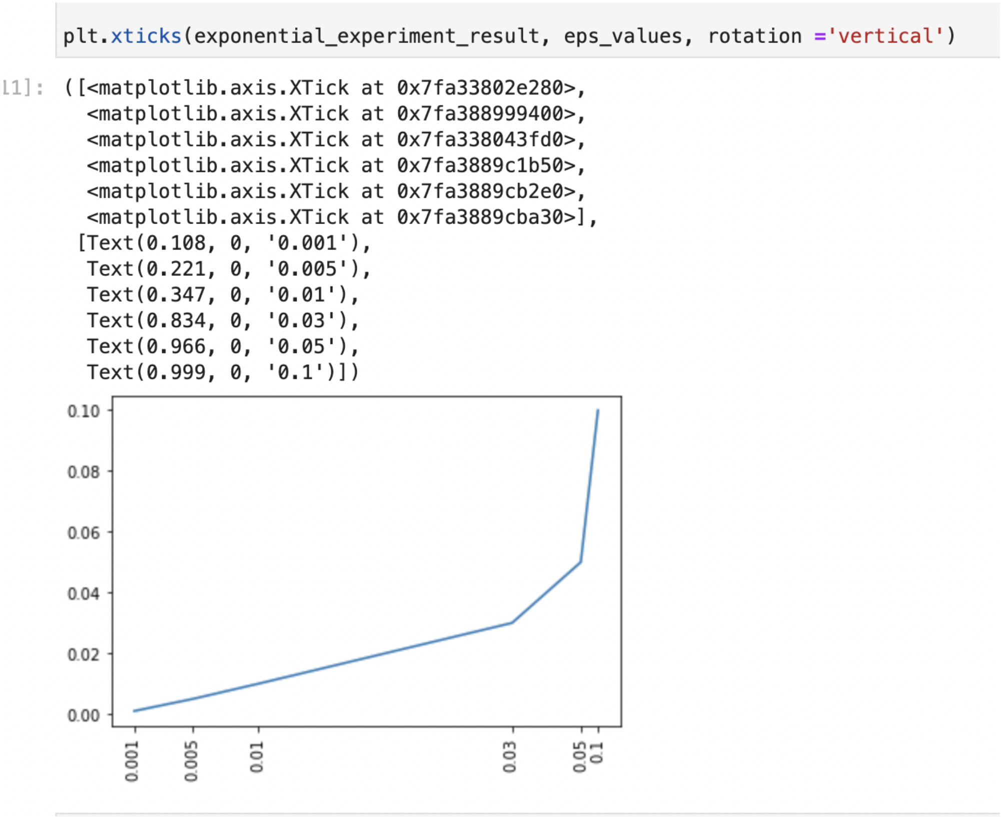
We can observe that when epsilon is lower, error is larger and this means that when epsilon is lower, we have more privacy, less utility.

Plot of average error,



Plot of MSE,





```
plt.xticks(exponential_experiment_result, eps_values, rotation ='vertical')
```

```
l1]: ([<matplotlib.axis.XTick at 0x7fa33802e280>,
       <matplotlib.axis.XTick at 0x7fa388999400>,
       <matplotlib.axis.XTick at 0x7fa338043fd0>,
       <matplotlib.axis.XTick at 0x7fa3889c1b50>,
       <matplotlib.axis.XTick at 0x7fa3889cb2e0>,
       <matplotlib.axis.XTick at 0x7fa3889cba30>],
      [Text(0.108, 0, '0.001'),
       Text(0.221, 0, '0.005'),
       Text(0.347, 0, '0.01'),
       Text(0.834, 0, '0.03'),
       Text(0.966, 0, '0.05'),
       Text(0.999, 0, '0.1')])
```

We can see the results for each epsilon in exponential experiment results. We can also see that larger epsilon corresponds to better accuracy here as well. One thing to note is that since we have less options to choose from here compared to the random number generator, error is less variable between same epsilon values.
These align with our expectations.

Part 3 - Some of the formulas and equations that I have implemented in 3rd part.

RAPPOR:

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

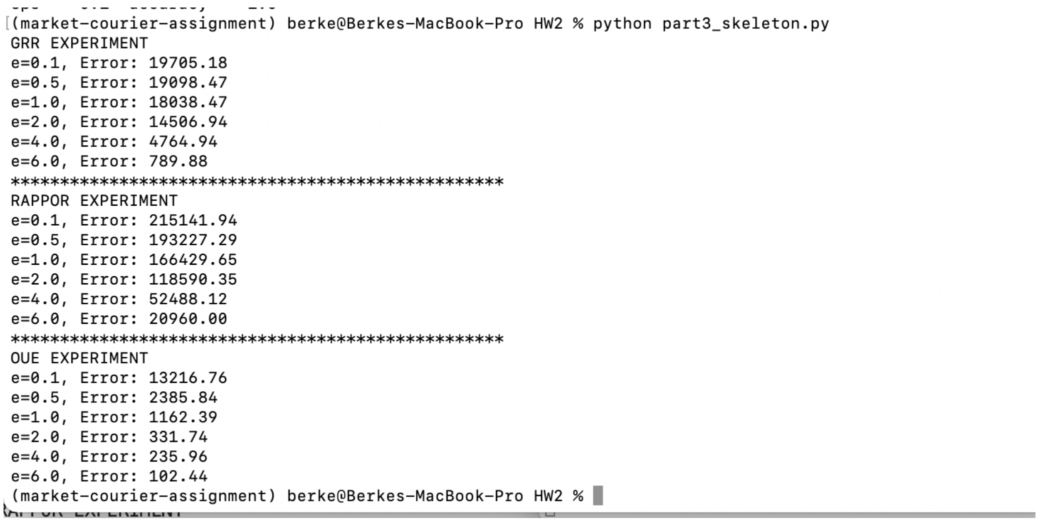
p = e ε e ε+d−1
$c(v)$ = Iv−n·q p−q
$E[Iv]$ = $nv \cdot p$ + n − nv · �

OUE:

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

Flip probabilities.

Results are in the following page.

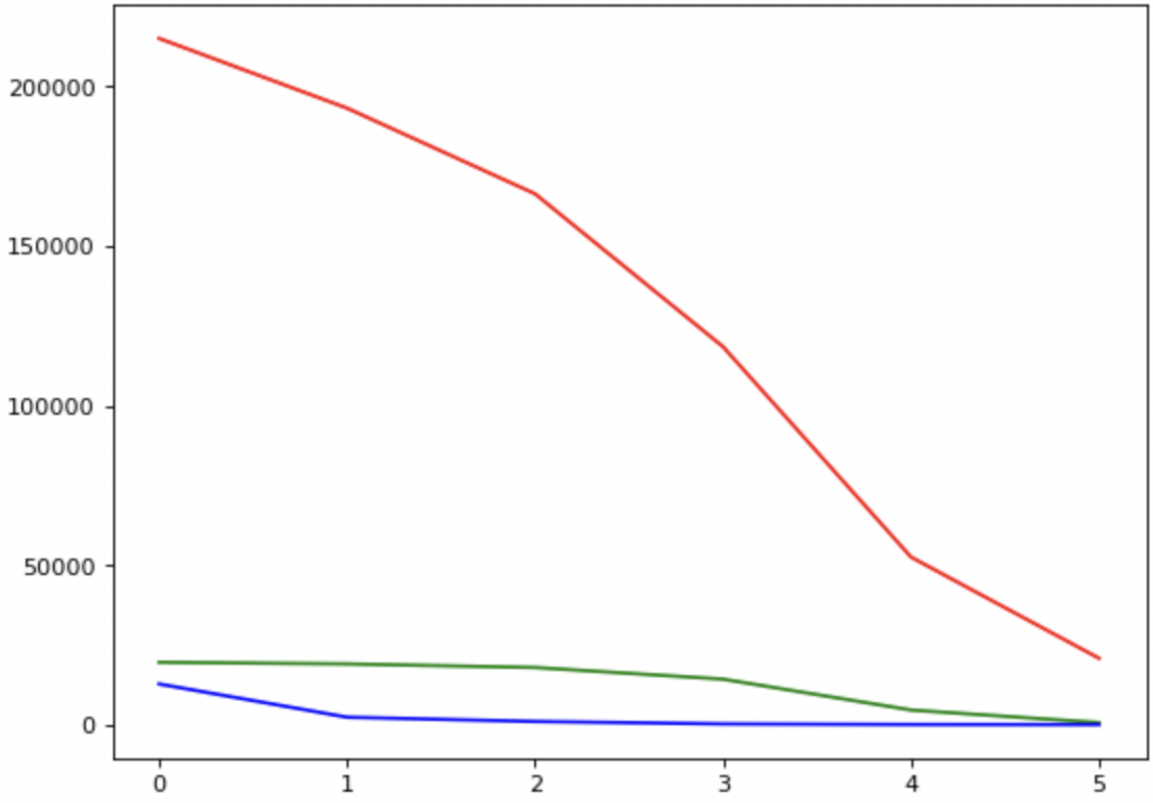


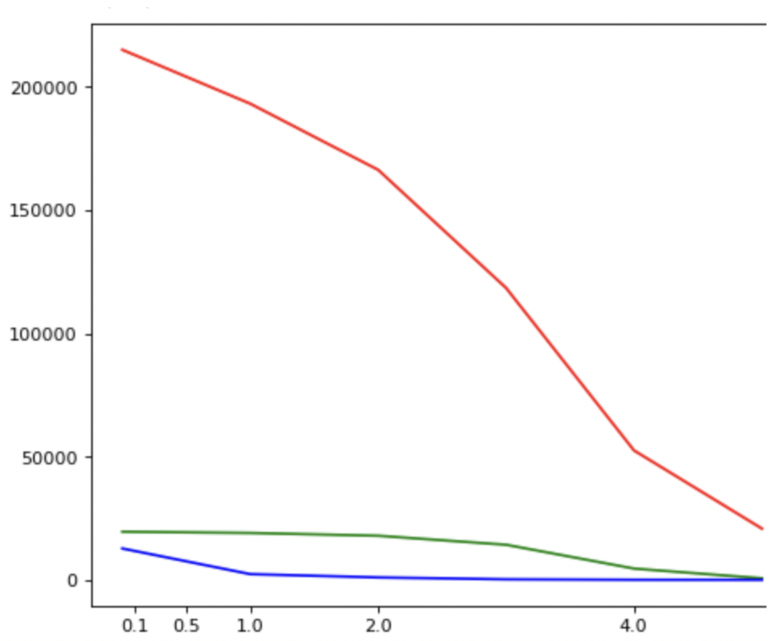
```
(market-courier-assignment) berke@Berkes-MacBook-Pro HW2 % python part3_skeleton.py
GRR EXPERIMENT
e=0.1, Error: 19705.18
e=0.5, Error: 19098.47
e=1.0, Error: 18038.47
e=2.0, Error: 14506.94
e=4.0, Error: 4764.94
e=6.0, Error: 789.88
**************************************************
RAPPOR EXPERIMENT
e=0.1, Error: 215141.94
e=0.5, Error: 193227.29
e=1.0, Error: 166429.65
e=2.0, Error: 118590.35
e=4.0, Error: 52488.12
e=6.0, Error: 20960.00
**************************************************
OUE EXPERIMENT
e=0.1, Error: 13216.76
e=0.5, Error: 2385.84
e=1.0, Error: 1162.39
e=2.0, Error: 331.74
e=4.0, Error: 235.96
e=6.0, Error: 102.44
(market-courier-assignment) berke@Berkes-MacBook-Pro HW2 %
```

We can visualize this with the following graph,

```
plt.plot(grr, c='g')
plt.plot(rappor, c='r')
plt.plot(oue, c='b')
```

```
[<matplotlib.lines.Line2D at 0x7fa3685586a0>]
```





We can observe that same principle that is, for larger epsilon values, we have less privacy and more accuracy and for smaller values of epsilon, we get larger errors. In the graph, the green line is GRR, blue is OUE and red is RAPPOR, and we can see that OUE gives the best results in terms of accuracy overall. As epsilon increases, we expect them to go together because bigger epsilon means less privacy and if epsilon is too big, accuracy will be high no matter which algorithm we choose.

Another observation is that the red line (RAPPOR) is the most sensitive one with the epsilon, and it varies greatly with accuracy for different epsilon values.

Credits:
Equations (flip probability, expected number of estimate etc.) are taken from the Course slides