UNIVERSITÀ
DEGLI STUDI
DI PADOVA

Differential Privacy on a Diabetic Dataset

PRIVACY PRESERVING INFORMATION ACCESS: HOMEWORK 2

Ferrari Luca (id. 2166294)
Scalco Riccardo (id. 2155352)

Years: 2025/2026

UNIVERSITÀ
DEGLI STUDI
DI PADOVA

OUR DATASET

Dataset Overview

- Around 800 female patients (Pima Indians, age starting from 21)
- No direct sensitive Identifiers (no ID, name,...)
- All attributes act as Quasi-Identifiers
- Outcome is the only binary feature (Type-2 Diabetes: Yes/No)

Diabetes dataset: https://www.kaggle.com/datasets/uciml/pima-indians-diabetes-database

Università degli Studi di Padova

DIFFERENTIAL PRIVACY

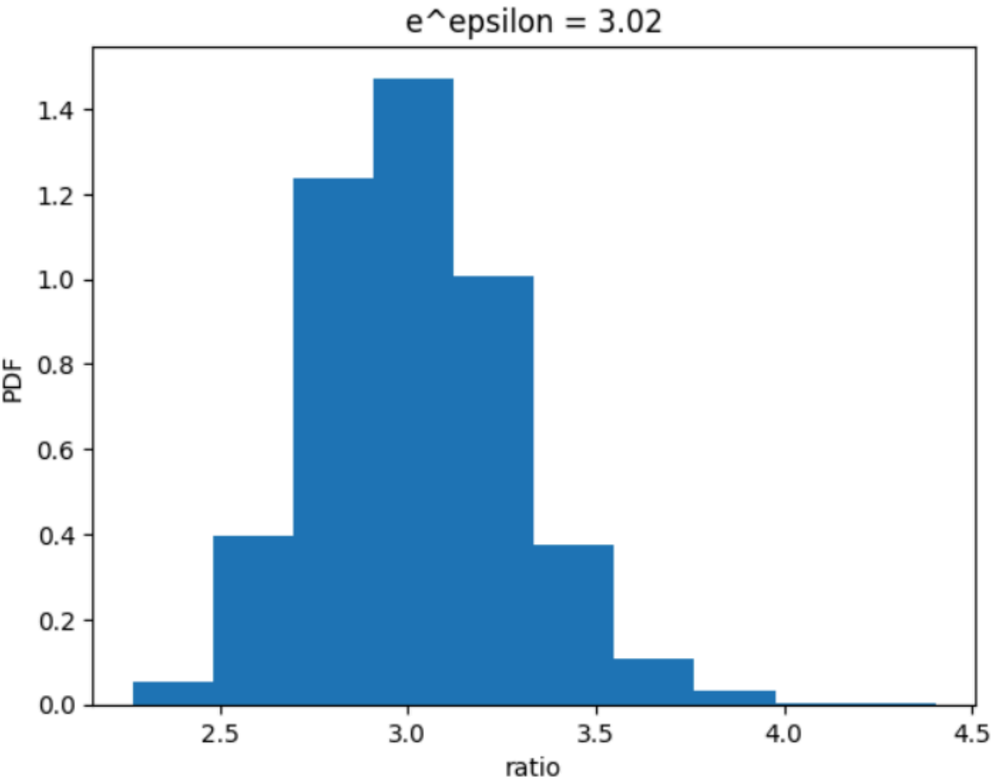
List of techinques Implemented:

- Coin toss & Epsilon coin toss
- Laplace mechanism on counting/histogram queries
- Exponential mechanism
- Sparse vector Algorithm:
  - Sparse
  - Numerical Sparse
- RAPPOR

COIN TOSS

QUERY: How many people are diabetic?

TRIVIAL COIN TOSS



Percent number of 1 in original coloum: 0.349
Percent number of 1 after coin toss: 0.346

EPSILON COIN TOSS

```python
def epsilonCointoss(data, epsilon=1e-5):
    size = len(data)
    outputs = []

    p_1_given_1 = np.exp(epsilon) / (np.exp(epsilon) + 1)

    for x in data:
        if np.random.random() < p_1_given_1:
            outputs.append(x)
        else:
            outputs.append(1 if x == 0 else 0)

    return outputs
```

```python
def getEpsilonOriginalProbabilities(data, epsilon):
    Y = np.sum(data)/len(data)
    e_eps = np.exp(epsilon)

    numerator = Y * (e_eps + 1) - 1
    denominator = e_eps - 1

    p = numerator / denominator
    return p
```

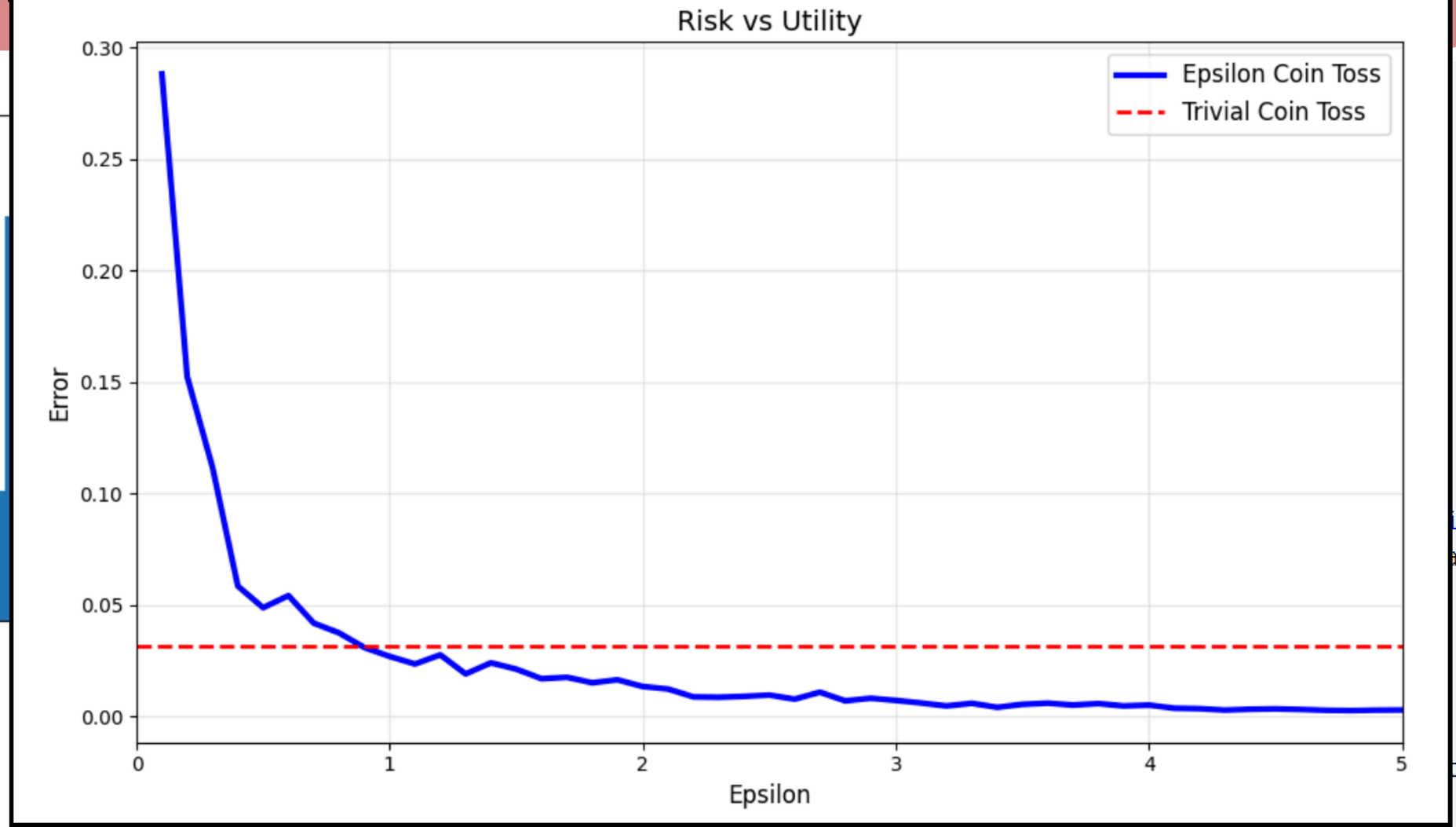
UNIVERSITÀ DEGLI STUDI DI PADOVA

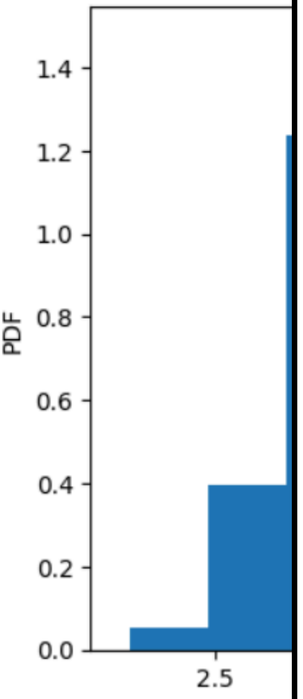
COIN TOSS

QUERY: How many people are diabetic?

TRIV

Real Probability: 0.34896



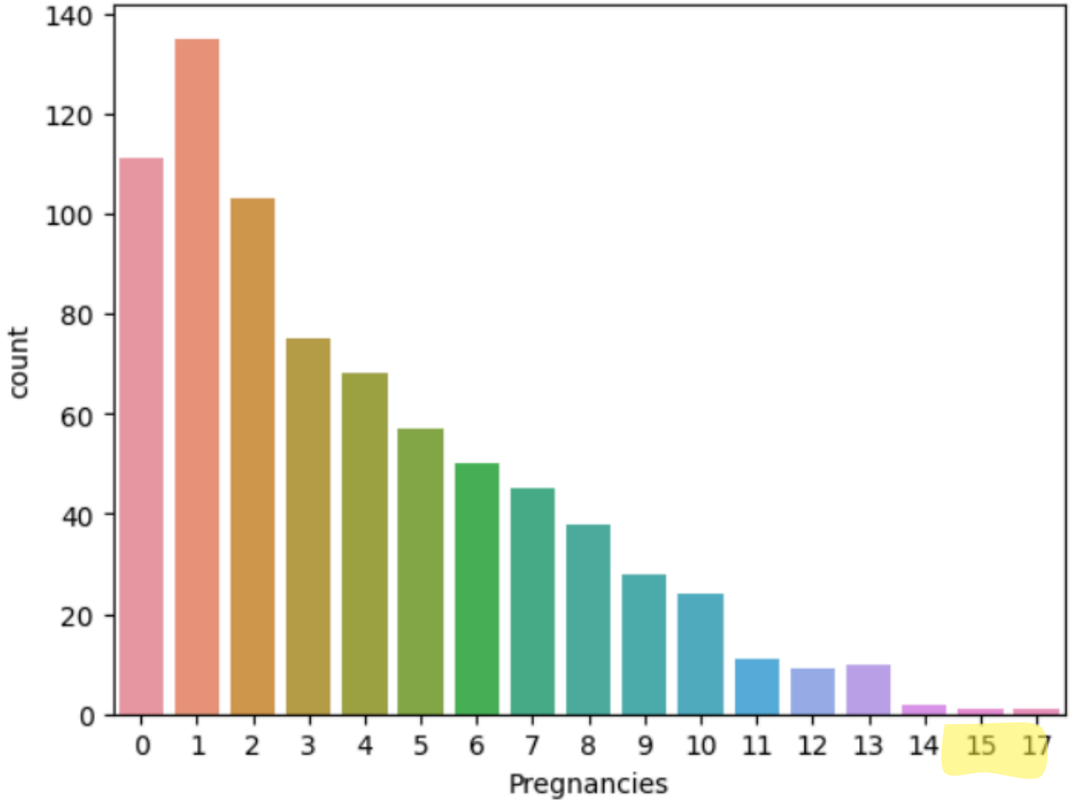



```
1)

ilities(data, epsilon):
a)

1) - 1

tor
```

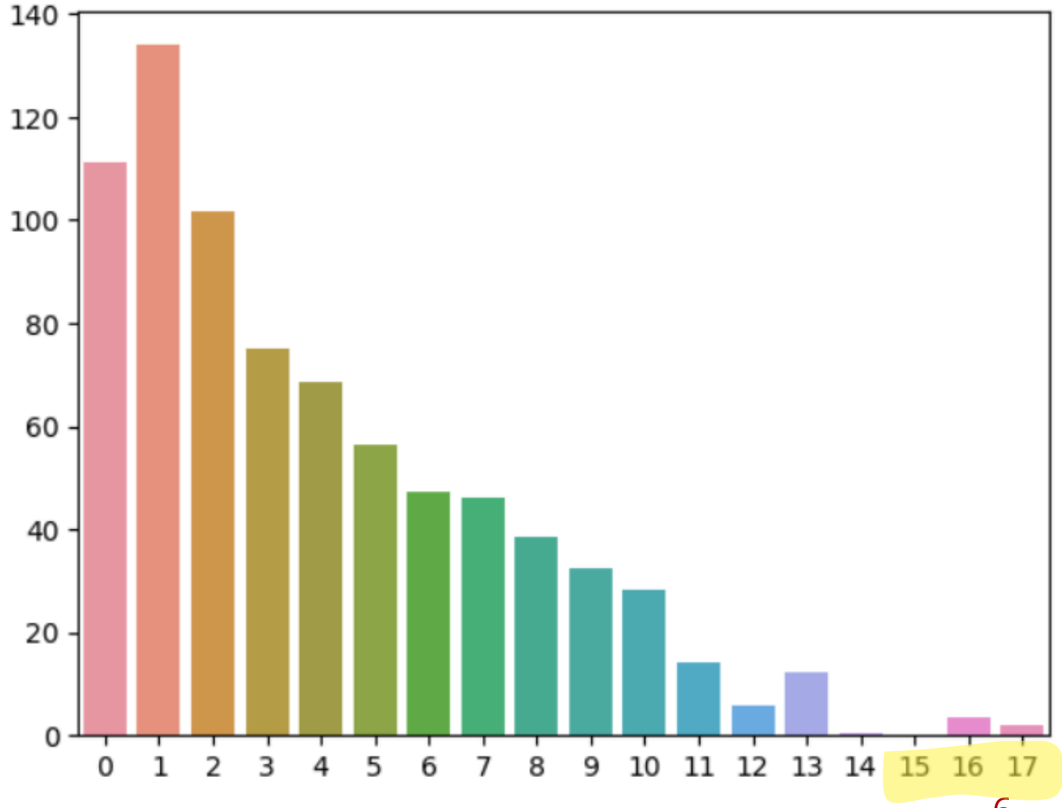
Università degli Studi di Padova

LAPLACE MECHANISM
HISTOGRAM QUERY: How many pregnancies have girls?

before Laplace



after Laplace



UNIVERSITÀ DEGLI STUDI DI PADOVA

LAPLACE MECHANISM HISTOGRAM QUERY: How many pregnancies have girls?

- $k = 17$
- $\delta = [0.001, 0.01, 0.1]$
- $\Delta f = 1$
- $\epsilon = [0.001, 0.01, 0.05, 0.1, 0.5, 1.0]$

$$\Pr\left[||f(x) - y||_\infty \geq \ln\left(\frac{k}{\delta}\right) \cdot \left(\frac{\Delta f}{\varepsilon}\right)\right] \leq \delta$$

```
(epsilon= 0.01) - probability of having an error bigger than 974.097: 0.000800 -- expected <= 0.001
(epsilon= 0.01) - probability of having an error bigger than 743.838: 0.009200 -- expected <= 0.01
(epsilon= 0.01) - probability of having an error bigger than 513.580: 0.094600 -- expected <= 0.1
(epsilon=  0.1) - probability of having an error bigger than 97.410: 0.001200 -- expected <= 0.001
(epsilon=  0.1) - probability of having an error bigger than 74.384: 0.009200 -- expected <= 0.01
(epsilon=  0.1) - probability of having an error bigger than 51.358: 0.099000 -- expected <= 0.1
(epsilon=  0.5) - probability of having an error bigger than 19.482: 0.001300 -- expected <= 0.001
(epsilon=  0.5) - probability of having an error bigger than 14.877: 0.010000 -- expected <= 0.01
(epsilon=  0.5) - probability of having an error bigger than 10.272: 0.092600 -- expected <= 0.1
(epsilon=    1) - probability of having an error bigger than 9.741: 0.001400 -- expected <= 0.001
(epsilon=    1) - probability of having an error bigger than 7.438: 0.008800 -- expected <= 0.01
(epsilon=    1) - probability of having an error bigger than 5.136: 0.096300 -- expected <= 0.1
(epsilon=   10) - probability of having an error bigger than 0.974: 0.001300 -- expected <= 0.001
(epsilon=   10) - probability of having an error bigger than 0.744: 0.012200 -- expected <= 0.01
(epsilon=   10) - probability of having an error bigger than 0.514: 0.091800 -- expected <= 0.1
```

UNIVERSITÀ DEGLI STUDI DI PADOVA

EXPONENTIAL MECHANISM

QUERY: We want to divide the people in two group with the same size based on the age

Sensitivity

```
perturbed_utils = []
for i in range(len(age)):
    new_age = np.concatenate((age[:i], age[i+1:]))
    perturbed_utils.append(
        utility_function(R, new_age))

perturbed_utils.append(
    utility_function(R,np.concatenate(([70], age))))

sensitivity = np.max([
    np.max(np.abs(np.array(real_utility) - np.array(pu)))
    for pu in perturbed_utils
])
print(sensitivity)
```

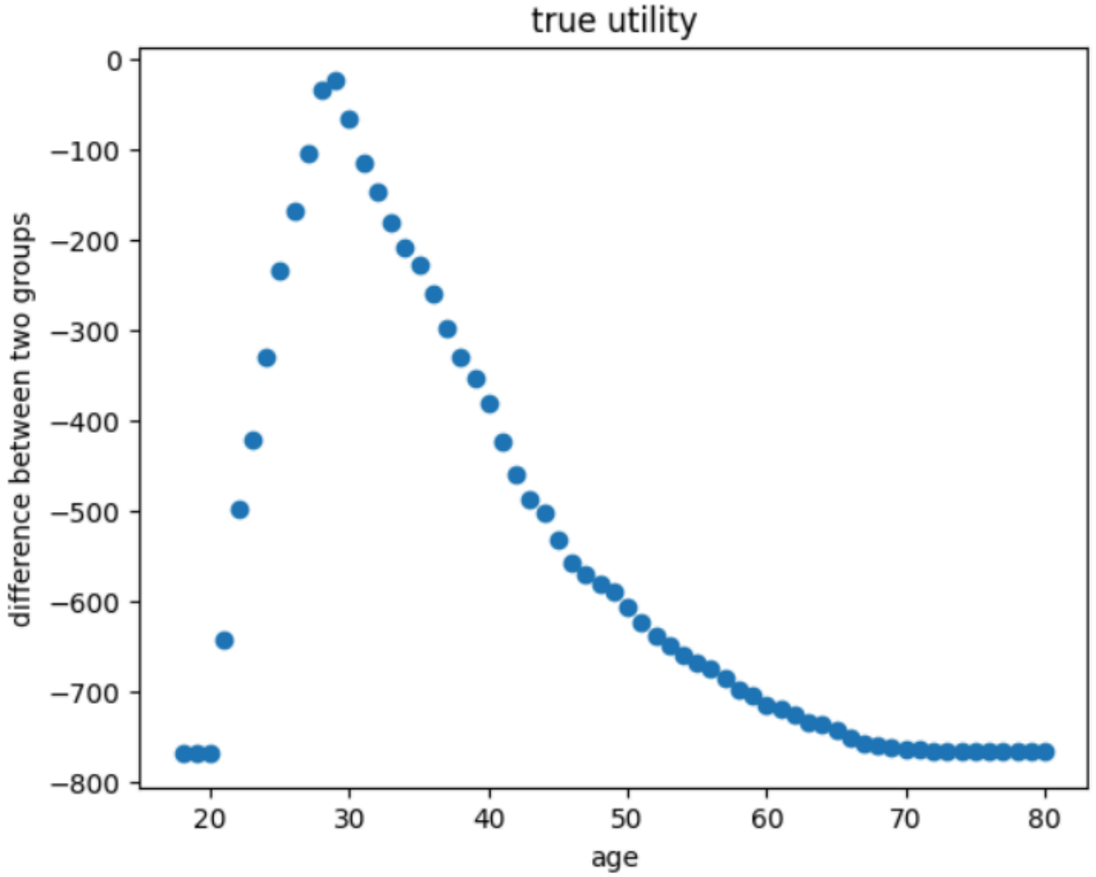
Utility Function

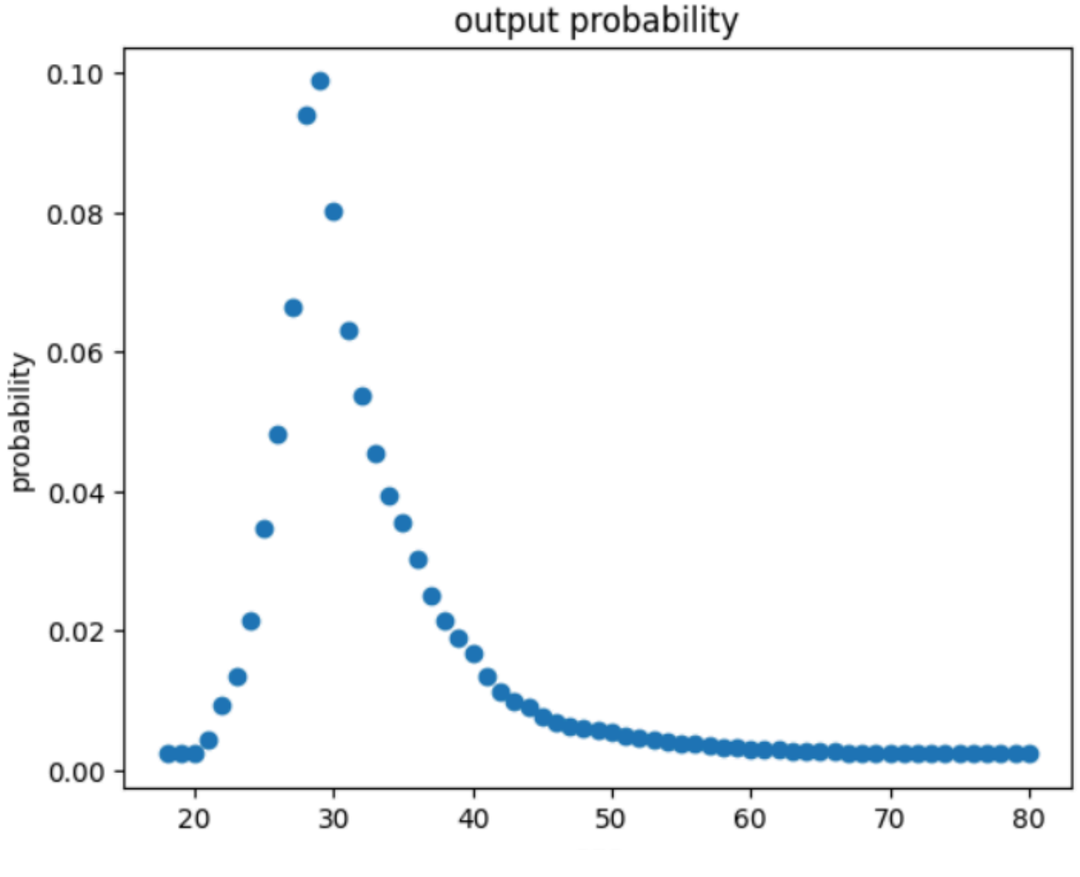
```
def utility_function(R, data):
    data = data
    utilities = []
    for r in R:
        group1 = data[data <= r]
        group2 = data[data > r]

        error = abs(len(group1) - len(group2))
        utilities.append(-error)

    return utilities
```

UNIVERSITÀ DEGLI STUDI DI PADOVA

EXPONENTIAL MECHANISM

QUERY: Divide the people in two group with the same size based on the age






```
best age: 29 - optimal utility: -24.00
output value: 32.00 - revenue: -146.00
```

```
best age: 29 - optimal utility: -24.00
output value: 31.00 - revenue: -114.00
```

UNIVERSITÀ DEGLI STUDI DI PADOVA

SPARSE VECTOR MECHANISM
QUERY: How is the Glucose value of the girls

SparseVector

```python
def sparse_vectors(data, queries, threshold, epsilon):
    results = {}

    for eps in epsilons:
        threshold_perturbed = np.random.laplace(threshold, 2/eps)
        run_output = []

        for q in queries:
            qnoise = np.random.laplace(0, 4/eps)
            noisy_value = q(data) + qnoise

            if noisy_value >= threshold_perturbed:
                run_output.append("ABOVE")
                break
            else:
                run_output.append("BELOW")

        results[eps] = run_output

    return results
```

Parameters

```python
glucose = diabetes_df["Glucose"].values
epsilons = [0.001, 0.01, 0.1, 0.25, 0.5, 1.0, 10.0]
threshold = 195
```

Results

```
epsilon = 0.001, output = ['ABOVE']
epsilon = 0.01, output = ['BELOW', 'BELOW', 'ABOVE']
epsilon = 0.1, output = ['ABOVE']
epsilon = 0.25, output = ['BELOW', 'BELOW', 'ABOVE']
epsilon = 0.5, output = ['BELOW', 'BELOW', 'BELOW', 'BELOW', 'BELOW', 'BELOW', 'BELOW', 'BEL
epsilon = 1.0, output = ['BELOW', 'BELOW', 'BELOW', 'BELOW', 'BELOW', 'BELOW', 'BELOW', 'BEL
epsilon = 10.0, output = ['BELOW', 'BELOW', 'BELOW', 'BELOW', 'BELOW', 'BELOW', 'BELOW', 'BE
```

Università degli Studi di Padova

SPARSE VECTOR MECHANISM
QUERY: How is the Glucose value of the girls

Sparse

```python
def sparse(data, queries, threshold, epsilon, limit_c):
    results = {}
    for eps in epsilons:
        count = 0
        threshold_perturbed = np.random.laplace(threshold, 2/eps)
        run_output = []
        for q in queries:
            qnoise = np.random.laplace(0, 4/eps)
            noisy_value = q(data) + qnoise
            if noisy_value >= threshold_perturbed:
                run_output.append("ABOVE")
                count += 1
                if count == limit_c: break
            else: run_output.append("BELOW")
        results[eps] = run_output
    return results
```

NumericSparse

```python
def numeric_sparse(data, queries, threshold, epsilon, limit_c):
    results = {}
    for eps in epsilons:
        count = 0
        threshold_perturbed = np.random.laplace(threshold, 2/eps)
        run_output = []
        for q in queries:
            qnoise = np.random.laplace(0, 4/eps)
            noisy_value = q(data) + qnoise
            if noisy_value >= threshold_perturbed:
                run_output.append(("ABOVE", noisy_value))
                count += 1
                if count == limit_c: break
            else: run_output.append("BELOW")
        results[eps] = run_output
    return results
```

UNIVERSITÀ DEGLI STUDI DI PADOVA

SPARSE VECTOR MECHANISM
QUERY: How is the Glucose value of the girls

Sparse (c = 3)

epsilon = 0.001, output = ['BELOW', 'ABOVE', 'ABOVE', 'BELOW', 'BELOW', 'BELOW', 'BELOW', 'BELOW', 'ABOVE']
epsilon = 0.01, output = ['ABOVE', 'BELOW', 'BELOW', 'BELOW', 'BELOW', 'BELOW', 'BELOW', 'BELOW', 'BELOW', 'ABOVE', 'BELOW', 'BELOW', 'ABOVE']
epsilon = 0.1, output = ['BELOW', 'BELOW', 'BELOW', 'BELOW', 'BELOW', 'BELOW', 'BELOW', 'BELOW', 'BELOW', 'BELOW', 'BELOW', 'BELOW', 'BELOW', 'BELOW', 'BELOW', 'BELOW', 'BELOW',
'BELOW', 'BELOW', 'BELOW', 'BELOW', 'BELOW', 'BELOW', 'ABOVE', 'BELOW', 'BELOW', 'BELOW', 'BELOW', 'BELOW', 'BELOW', 'BELOW', 'BELOW', 'BELOW', 'BELOW', 'BELOW', 'BELO
W', 'BELOW', 'BELOW', 'BELOW', 'BELOW', 'BELOW', 'BELOW', 'BELOW', 'BELOW', 'BELOW', 'BELOW', 'BELOW', 'BELOW', 'BELOW', 'BELOW', 'BELOW', 'BELOW', 'BELOW', 'BELOW', 'E
LOW', 'BELOW', 'BELOW', 'BELOW', 'BELOW', 'BELOW', 'BELOW', 'BELOW', 'BELOW', 'BELOW', 'BELOW', 'BELOW', 'BELOW', 'BELOW', 'BELOW', 'BELOW', 'BELOW', 'BELOW', 'BELOW',
'BELOW', 'BELOW', 'BELOW', 'BELOW', 'BELOW', 'BELOW', 'BELOW', 'BELOW', 'BELOW', 'BELOW', 'BELOW', 'BELOW', 'BELOW', 'BELOW', 'BELOW', 'BELOW', 'BELOW', 'BELOW', 'BELO
W', 'BELOW', 'BELOW', 'BELOW', 'BELOW', 'BELOW', 'ABOVE', 'BELOW', 'BELOW', 'BELOW', 'BELOW', 'BELOW', 'BELOW', 'BELOW', 'BELOW', 'BELOW', 'BELOW', 'BELOW', 'BELOW', 'E
LOW', 'BELOW', 'BELOW', 'BELOW', 'BELOW', 'BELOW', 'BELOW', 'BELOW', 'BELOW', 'BELOW', 'BELOW', 'BELOW', 'BELOW', 'BELOW', 'BELOW', 'BELOW', 'BELOW', 'BELOW', 'BELOW',
'BELOW', 'BELOW', 'BELOW', 'BELOW', 'BELOW', 'BELOW', 'BELOW', 'BELOW', 'BELOW', 'BELOW', 'BELOW', 'BELOW', 'BELOW', 'BELOW', 'BELOW', 'BELOW', 'BELOW', 'BELOW', 'BELO
W', 'BELOW', 'BELOW', 'BELOW', 'BELOW', 'BELOW', 'BELOW', 'BELOW', 'BELOW', 'BELOW', 'BELOW', 'BELOW', 'BELOW', 'BELOW', 'BELOW', 'BELOW', 'BELOW', 'BELOW', 'BELOW', 'E
LOW', 'BELOW', 'BELOW', 'BELOW', 'BELOW', 'BELOW', 'BELOW', 'BELOW', 'BELOW', 'BELOW', 'BELOW', 'BELOW', 'BELOW', 'BELOW', 'BELOW', 'BELOW', 'BELOW', 'BELOW', 'BELOW',
'BELOW', 'BELOW', 'BELOW', 'BELOW', 'BELOW', 'ABOVE']
epsilon = 0.25, output = ['BELOW', 'BELOW', 'BELOW', 'BELOW', 'BELOW', 'BELOW', 'BELOW', 'BELOW', 'ABOVE', 'BELOW', 'BELOW', 'BELOW', 'BELOW', 'BELOW', 'BELOW', 'BELOW', 'BELOW',
'BELOW', 'BELOW', 'BELOW', 'BELOW', 'BELOW', 'BELOW', 'BELOW', 'BELOW', 'BELOW', 'BELOW', 'BELOW', 'BELOW', 'BELOW', 'ABOVE', 'BELOW', 'BELOW', 'BELOW', 'BELOW', 'BELOW', 'BELO

NumericSparse

epsilon = 0.001, output = ['BELOW', 'BELOW', ('ABOVE', 4616.794430102126), ('ABOVE', 12839.079778369474), 'BELOW', ('ABOVE', 3426.4233049457303)]
epsilon = 0.01, output = [('ABOVE', 674.2579619126356), ('ABOVE', 755.9377483034375), ('ABOVE', 642.4728193708339)]
epsilon = 0.1, output = ['BELOW', 'BELOW', ('ABOVE', 230.6315581402661), 'BELOW', 'BELOW', 'BELOW', 'BELOW', 'BELOW', ('ABOVE', 221.01335697139436), 'BELOW', 'BELO
ELOW', 'BELOW', ('ABOVE', 295.66817004200027)]
epsilon = 0.25, output = ['BELOW', 'BELOW', ('ABOVE', 214.58048399381724), 'BELOW', 'BELOW', 'BELOW', 'BELOW', 'BELOW', 'BELOW', 'BELOW', 'BELOW', ('ABOVE', 241.29
048463), 'BELOW', 'BELOW', 'BELOW', 'BELOW', 'BELOW', 'BELOW', 'BELOW', 'BELOW', 'BELOW', 'BELOW', ('ABOVE', 264.9194171009416)]
epsilon = 0.5, output = ['BELOW', 'BELOW', ('ABOVE', 209.05465343330692), 'BELOW', 'BELOW', 'BELOW', 'BELOW', 'BELOW', 'BELOW', 'BELOW', 'BELOW', 'BELOW'

UNIVERSITÀ DEGLI STUDI DI PADOVA

RAPPOR

Estimating BMI category counts Without revealing Individual identities

1) Encode the value v in a vector B via a bloom filter:

using one hash function encode a value in a vector of size k

```python
def hash_function_for_BMI(category_name):
    mapp = {
        "Underweight"   : 0,
        "Normal" : 1,
        "Overweight"    : 2,
        "Obese" : 3,
        "SeverelyObese": 4,
    }
    return mapp[category_name]
```

```python
def BloomFiltering(category_name, K=5):
    idx = hash_function_for_BMI(category_name)
    B = [0] * K
    B[idx] = 1
    return B
```

- h=1
- k=5

Università degli Studi di Padova

RAPPOR

2) randomize (and memoize) B in a vector B’

$$B'_i = \begin{cases} 1, & \text{with probability } \frac{1}{2}f \\ 0, & \text{with probability } \frac{1}{2}f \\ B_i, & \text{with probability } 1-f \end{cases}$$

- f = 0.5

```
def coinTossRappor(data, f):
    size = len(data)
    outputs = []
    for i in range(size):
        if random.random() < 1 - f:
            outputs.append(data[i])
        else:
            outputs.append(1 if random.random() > 0.5 else 0)

    return outputs
```

Università degli Studi di Padova

RAPPOR

3) randomize B’ in a response S and send it to the server $P(S_i = 1) = \begin{cases} q, & \text{if } B'_i = 1 \\ p, & \text{if } B'_i = 0 \end{cases}$

```python
def randomizeRespondeRappor(B_prime, q, p):
    i = 0
    report_response = []
    for i, bit in enumerate(B_prime):
        if bit == 1:
            report_response.append(1 if random.random() < q else 0)
        else:
            report_response.append(1 if random.random() < p else 0)
    return report_response
```

Università degli Studi di Padova

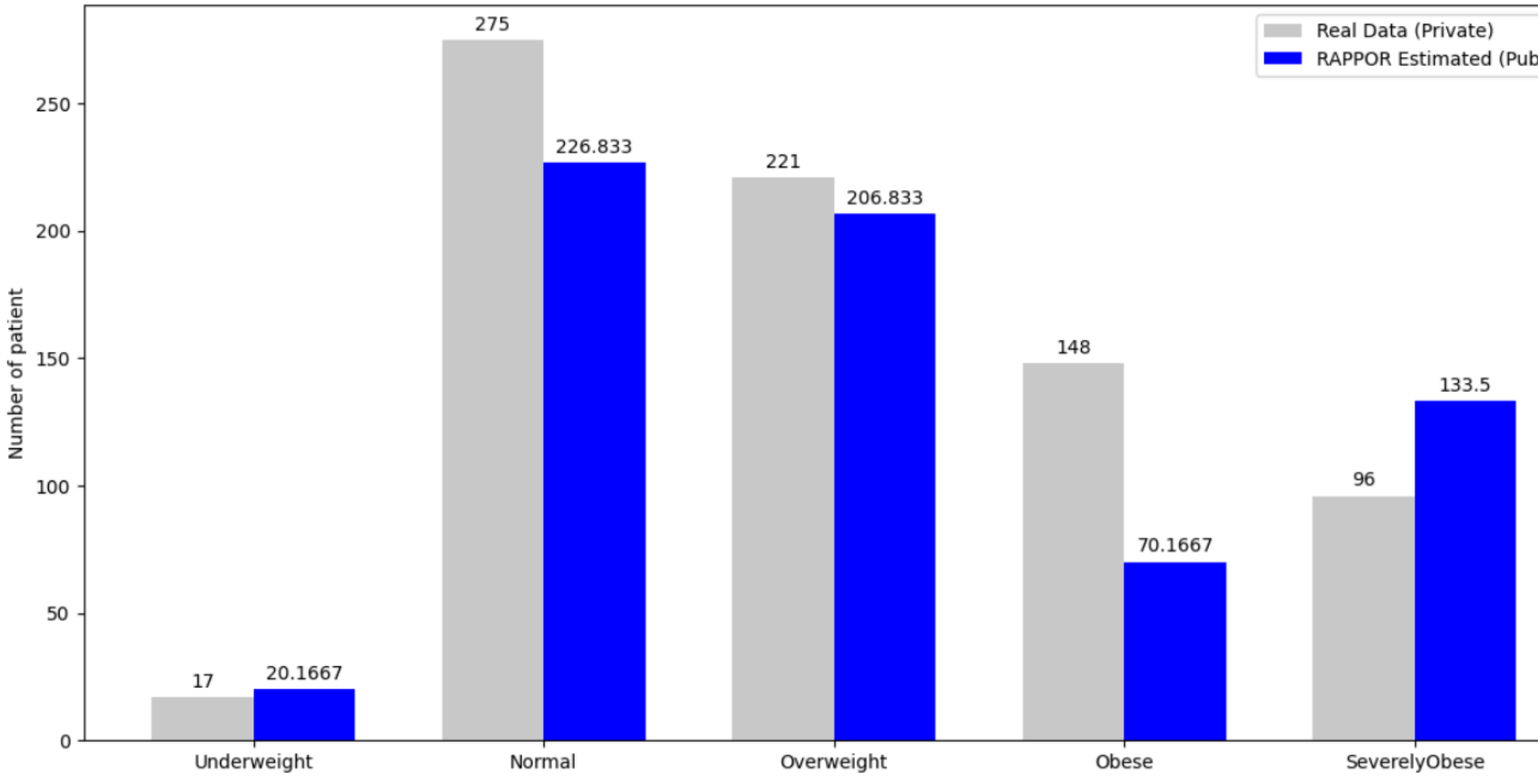
RAPPOR & Results

$$\varepsilon = 2h \log\left(\frac{1 - \frac{1}{2}f}{\frac{1}{2}f}\right)$$

ε = 0.95424

$$t_{ij} = \frac{c_{ij} - (p + \frac{1}{2}fq - \frac{1}{2}fp)N_j}{(1-f)(q-p)}$$

```
def decode_rappor(reports, f, q, p):
    N_j = len(reports)
    c_ij = np.sum(reports, axis=0)

    num = p + 0.5 * f * q - 0.5 * f * p
    den = (1 - f) * (q - p)

    est = (c_ij - (num * N_j)) / (den)

    est = np.maximum(est, 0)
    return est
```



- h = 1
- f = 0.5
- q = 0.8
- p = 0.2

UNIVERSITÀ
DEGLI STUDI
DI PADOVA

Thanks for your attention