Università di Genova

Mondrian Multidimensional K-Anonymity

Data Protection & Privacy
Computer Engineering

Authors:
Basso Alessandro
Perasso Bianca

Agenda

Università di Genova

1. *K-anonymity*
2. *Dataset generation*
3. *Mondrian Algorithm*
4. *Quality Metrics & Test*

Agenda

Università di Genova

1. *K-anonymity*
2. *Dataset generation*
3. *Mondrian Algorithm*
4. *Quality Metrics & Test*

k – Anonymity

Università di Genova

K-Anonymization is a technique for preserving individual identification by transforming the record set so that each record of a table identical to at least k-1 other records in terms of quasi-identifying attributes.

K-anonymization is granted by generalizing and suppressing the value of attributes.

gender	city	age	education	profession	annual_income
female	[Italy]	[45-77]	[ANY-EDUCATION]	[ANY-JOB]	55134
female	[Italy]	[45-77]	[ANY-EDUCATION]	[ANY-JOB]	56928
female	[Italy]	[45-77]	[ANY-EDUCATION]	[ANY-JOB]	23321
[ANY-GENDER]	[Italy]	[52-72]	High School	[ANY-JOB]	40449
[ANY-GENDER]	[Italy]	[52-72]	High School	[ANY-JOB]	33253
[ANY-GENDER]	[Italy]	[52-72]	High School	[ANY-JOB]	20848

Agenda

Università di Genova

1 *K-anonymity*

Dataset generation 2

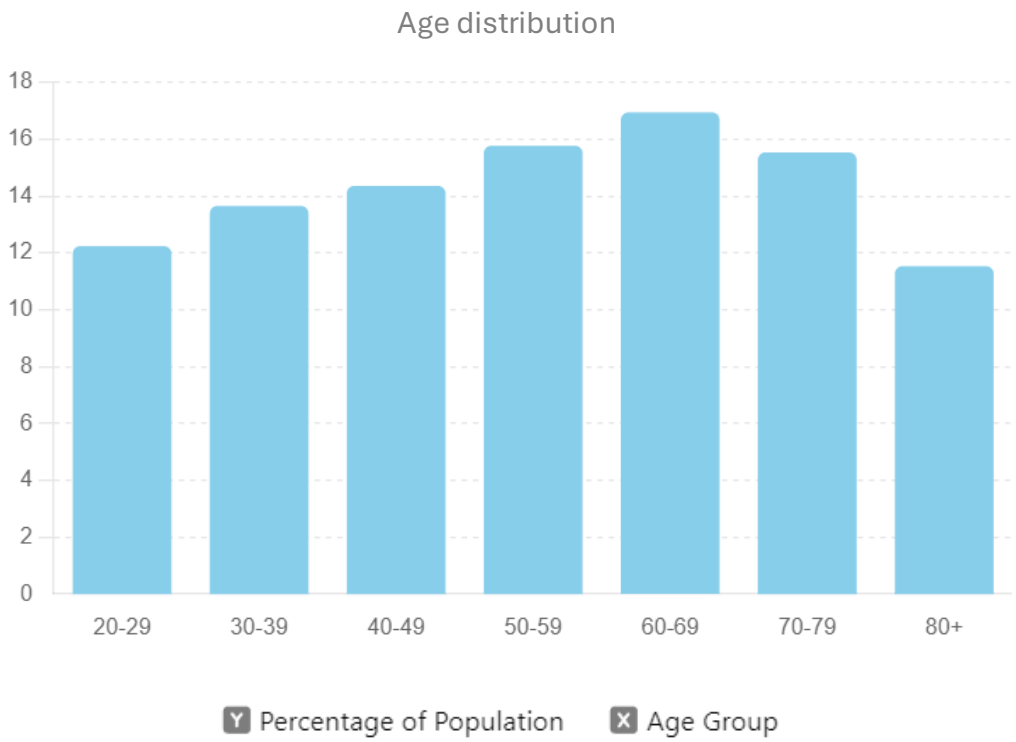
3 *Mondrian Algorithm*

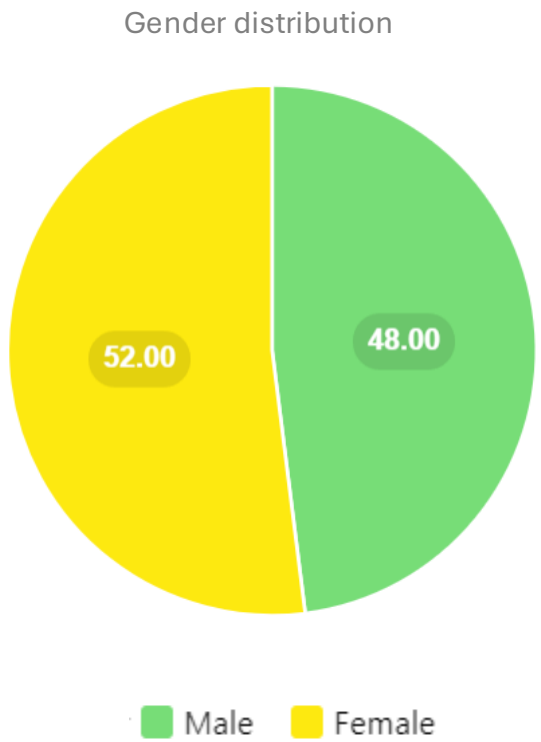
Quality Metrics & Test 4

Dataset Generation

Università di Genova

The dataset was generated following the Italian distribution regarding gender, age, level of education, type of work relative to the level of education and average annual salary relative to the type of work [1].

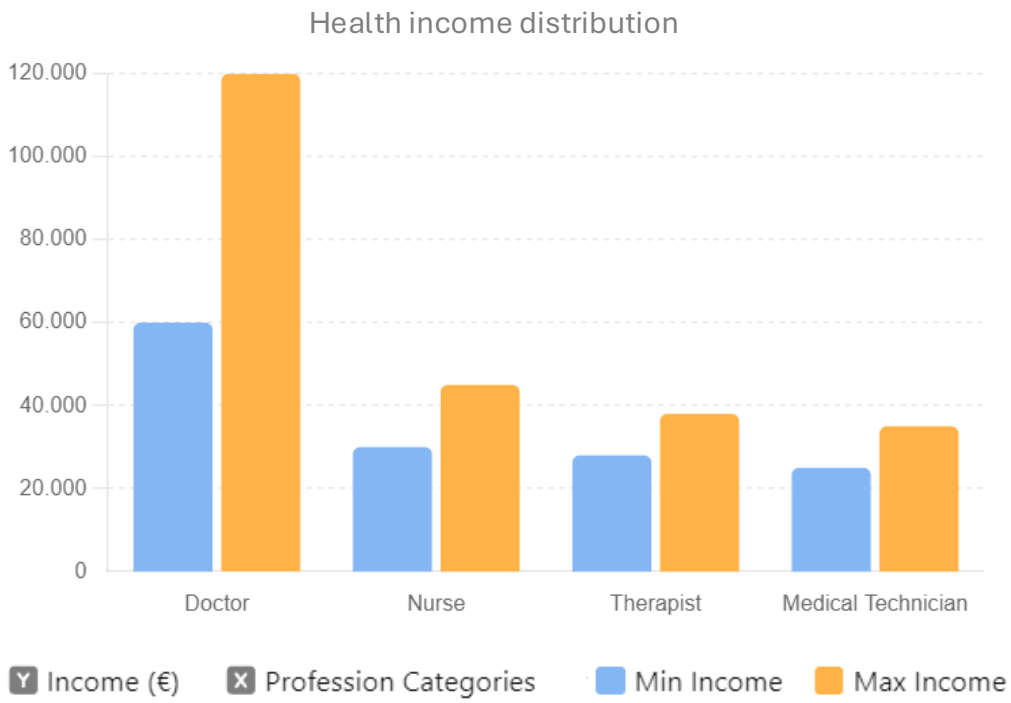


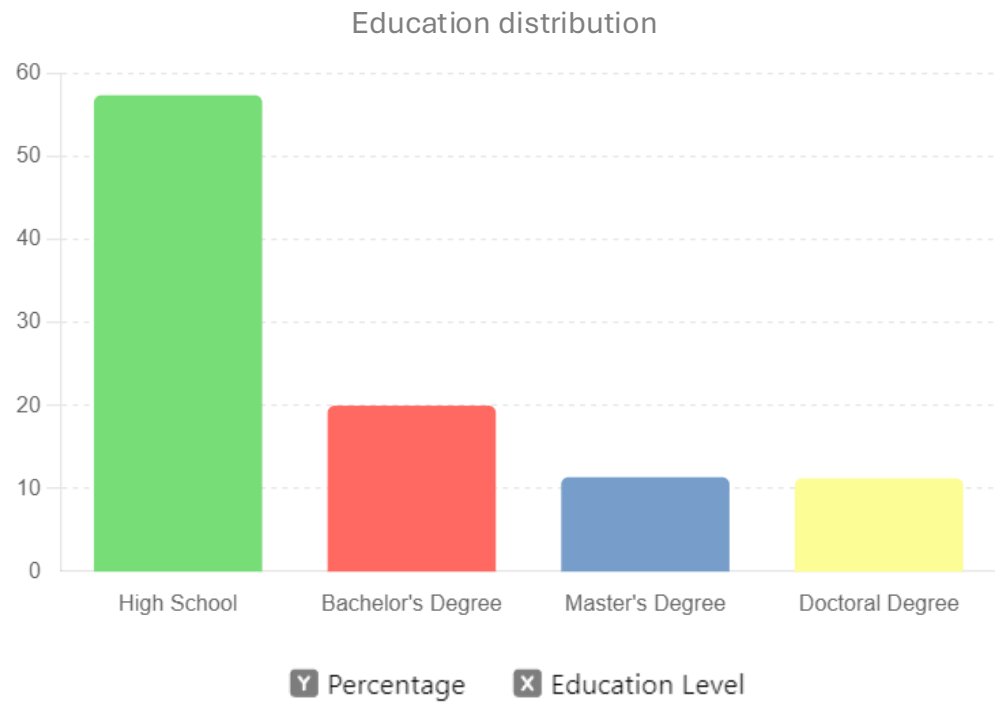


Dataset Generation

Università di Genova

The dataset was generated following the Italian distribution regarding gender, age, level of education, type of work relative to the level of education and average annual salary relative to the type of work [1].





Explicit Identifier

Università di Genova

Explicit identifer are attributes that uniquely allow to identify an individual record.
In order to protect individual privacy Explicit identifier must be removed.

PERSON ID	FIRST NAME	LAST NAME
1ce5b3d4	Tina	Carocci
a71fa5e8	Benvenuto	Santi
c9b2cffe	Rosa	Alighieri
688e0c28	Leonardo	Giannini
b0c56f4f	Margherita	Lanfranchi

Quasi Identifier

Università di Genova

Quasi identifier are attributes that can be joined with external information to identify an individual record. In order to protect individual privacy Explicit identifier must be anonymized:

- Numerical attributes with Range or Mean
- Categorical attributes with generalizitation

GENDER	CITY	AGE	EDUCATION	PROFESSION
Female	Avellino	48	High School	Restaurant Executive Chef
Male	Cremona	84	High School	Opera Singer
Female	Ascoli Piceno	46	Master's Degree	Geography Teacher
Female	Bressanone	78	Doctoral Degree	Cardiothoracic Surgeon

Categorical

Numerical

Sensitive Attributes

Università di Genova

Sensitive attributes contain confidential information which cannot be compromised at any cost.

In order to preserve the utility they should be kept original.

ANNUAL INCOME
84158
31045
48926
23677
35907
43031

Agenda

Università di Genova

1 *K-anonymity*

Dataset generation 2

3 ***Mondrian Algorithm***

Quality Metrics & Test 4

Mondrian Alghoritm

Università di Genova

The mondrian algorithm executes:

1. REMOVAL EXPLICIT IDENTIFIER
2. PARTITIONING
3. GENERALIZATION
4. PERTURBATION

```python
def mondrian(database, k, qis, sd, ei, json_files, ordinal_qis, check=None):
    global dataframe_partitions
    dataframe_partitions = []

    database = drop_column(database, ei)
    iterative_partition(database, k, sd)

    generalized_partitions = []
    for partition in dataframe_partitions:
        generalized_partition = generalize_partition(partition, qis, json_files, statistic='range')
        generalized_partitions.append(generalized_partition)

    anonymized_data = pd.concat(generalized_partitions, ignore_index=True)
    perturbated_data = add_perturbation(anonymized_data, sensitive_data)
    perturbated_data.to_csv('anonymized.csv', index=False)
```

Partitioning

Università di Genova

This function does:

- Splitting by median if column is numerical
- Splitting by middle value if columns is categorical

```python
def splitter(dataframe, column, k):
    """Splits the dataframe along a certain column respecting k value"""
    if column in dataframe.select_dtypes(include='number').columns:
        median_value = dataframe[column].median()
        left_partition = dataframe[dataframe[column] <= median_value]
        right_partition = dataframe[dataframe[column] > median_value]
    elif column in dataframe.select_dtypes(include='object').columns:
        sorted_values = sorted(dataframe[column])
        middle_index = len(sorted_values) // 2
        middle_value = sorted_values[middle_index]
        left_partition = dataframe[dataframe[column] <= middle_value]
        right_partition = dataframe[dataframe[column] > middle_value]
    else:
        # Skip columns that are neither numeric nor categorical
        return None, None

    if len(left_partition) >= k and len(right_partition) >= k and len(left_partition) < k*2 and len(right_partition)
        return left_partition, right_partition
    else:
        # Adjust partitions if lengths are less than k
        left_partition = dataframe.iloc[:k]
        right_partition = dataframe.iloc[k:]
        return left_partition, right_partition
```

Generalization

Università di Genova

Categorical columns are generalized by finding the lowest common ancestor in a JSON file.

```python
def generalize_partition(partition, qis, json_files, statistic):
    """ numerical quasi-identifiers are generalized using the statistic provided.
        categorical quasi-identifiers are generalized using the LCA """
    numerical_qis = [qi for qi in qis if partition[qi].dtype in ['int64', 'float64']]

    for qi in qis:
        partition = partition.sort_values(by=qi)

        if partition[qi].iloc[0] != partition[qi].iloc[-1]:
            if qi in numerical_qis:
                if statistic == 'range':
                    min_val = partition[qi].iloc[0]
                    max_val = partition[qi].iloc[-1]
                    s = f"[{min_val}-{max_val}]"
                elif statistic == 'mean':
                    mean_val = partition[qi].mean()
                    s = f"[{mean_val}]"
                else:
                    raise ValueError("Statistic must be 'range' or 'mean'")
            else:
                unique_values = sorted(set(partition[qi]))
                if len(unique_values) == 1:
                    lca = unique_values[0]
                else:
                    lca = unique_values[0]
                    for value in unique_values[1:]:
                        lca_candidate = find_lowest_common_ancestor(json_files[qi], lca, value, key='name')
                        if lca_candidate:
                            lca = lca_candidate
                        else:
                            lca = 'ANY'
                            break

                s = f"[{lca}]"

            partition[qi] = [s] * partition[qi].size

    return partition
```

Random Perturbation

Università di Genova

A small random disturbance to sensitive data is applied.

The disturbance is generated as a normal distribution with mean zero and variance proportional to the original column variance

```python
def generate_small_perturbation(variance, size, perturbation_scale=0.1):
    mean_perturbation = 0
    perturbation_variance = variance * perturbation_scale
    perturbation = np.random.normal(mean_perturbation, np.sqrt(perturbation_variance), size)
    return perturbation


def add_perturbation(data, sensitive_data, min_value=0, perturbation_scale=0.1):
    column_name = sensitive_data[0]
    column = data[column_name].values
    mean_original, variance_original = compute_mean_variance(column)
    perturbation = generate_small_perturbation(variance_original, column.shape[0], perturbation_scale)
    perturbed_column = column + perturbation

    mean_perturbed = np.mean(perturbed_column)
    perturbed_column_adjusted = perturbed_column - mean_perturbed + mean_original

    variance_perturbed = np.var(perturbed_column_adjusted, ddof=1)

    std_factor = np.sqrt(variance_original / variance_perturbed)
    perturbed_column_final = (perturbed_column_adjusted - mean_original) * std_factor + mean_original

    perturbed_column_final = np.round(perturbed_column_final).astype(int)  # arrotondamento a numero intero
    perturbed_column_final = np.maximum(perturbed_column_final, min_value)

    data[column_name] = perturbed_column_final

    return data
```

Mapping

Università di Genova

In order to compute statystical analysis as mean, variance on categorical attributes, they must be mapped.

This function maps categorical attributes only if they have a hierarchical order.

```python
def create_mapping(levels, start_value=1):
    mapping = {}
    def recursive_map(levels, current_value):
        for level in levels:
            if 'categories' in level and level['categories']:
                mapping[level['name']] = current_value
                for sub_level in level['categories']:
                    mapping[sub_level['name']] = current_value
                current_value += 1
            else:
                mapping[level['name']] = current_value
                current_value += 1
        return current_value

    recursive_map(levels['categories'], start_value)
    return mapping
```

Agenda

Università di Genova

1 *K-anonymity*

Dataset generation 2

3 *Mondrian Algorithm*

Quality Metrics & Test 4

Original vs Anonymized dataset

Università di Genova

We did some analysis to evaluate the differences between the original dataset and the anonymized dataset. We computed the mean and the standard deviation per each column, analyzing how much the utility has been preserved during the anonymization.

Numerical attribute:

```
Column: age
  Original mean: 54.892
  Anonymized mean: 54.8275
  Original standard deviation: 19.457516052439246
  Anonymized standard deviation: 10.293053030597418
```

Categorical attribute:

```
Column: education
  Original mean: 1.702
  Anonymized mean: 1.1355932203389831
  Original variance: 0.7079039039039037
  Anonymized variance: 0.15196456086286594
Column: gender
  Original mean: 0.537
  Anonymized mean: 0.5769230769230769
  Original variance: 0.2488798798798799
  Anonymized variance: 0.24513040607461206
```

General Purpose Quality Metrics

Università di Genova

The simplest kind of quality measure is based on the size of the equivalence classes E in the dataset. We will see the discernability metric (C_{DM}) and the normalized average equivalence class size metric (C_{AVG}).

The discernability metric (C_{DM}):

$$C_{DM} = \sum_{EquivClasses\ E} |E|^2$$

It assigns to each tuple t a penalty, which is determined by the size of the equivalence class containing t.

The normalized average equivalence class size metric (C_{AVG}):

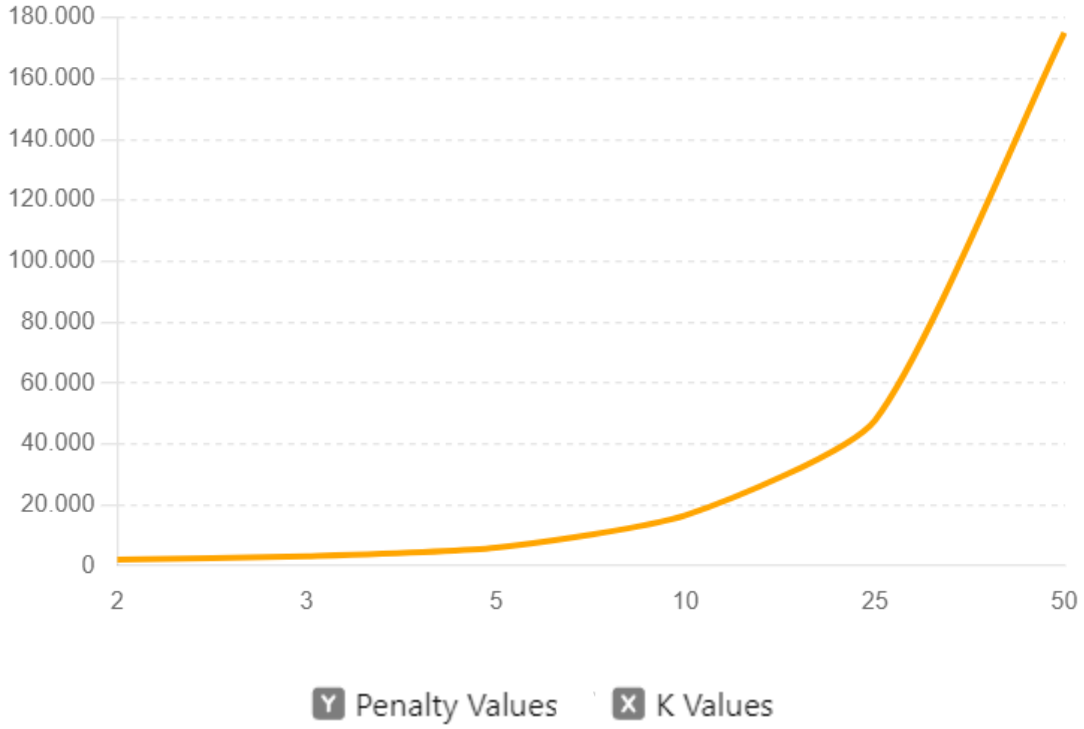
$$C_{AVG} = \left(\frac{total\_records}{total\_equiv\_classes}\right)/(k)$$

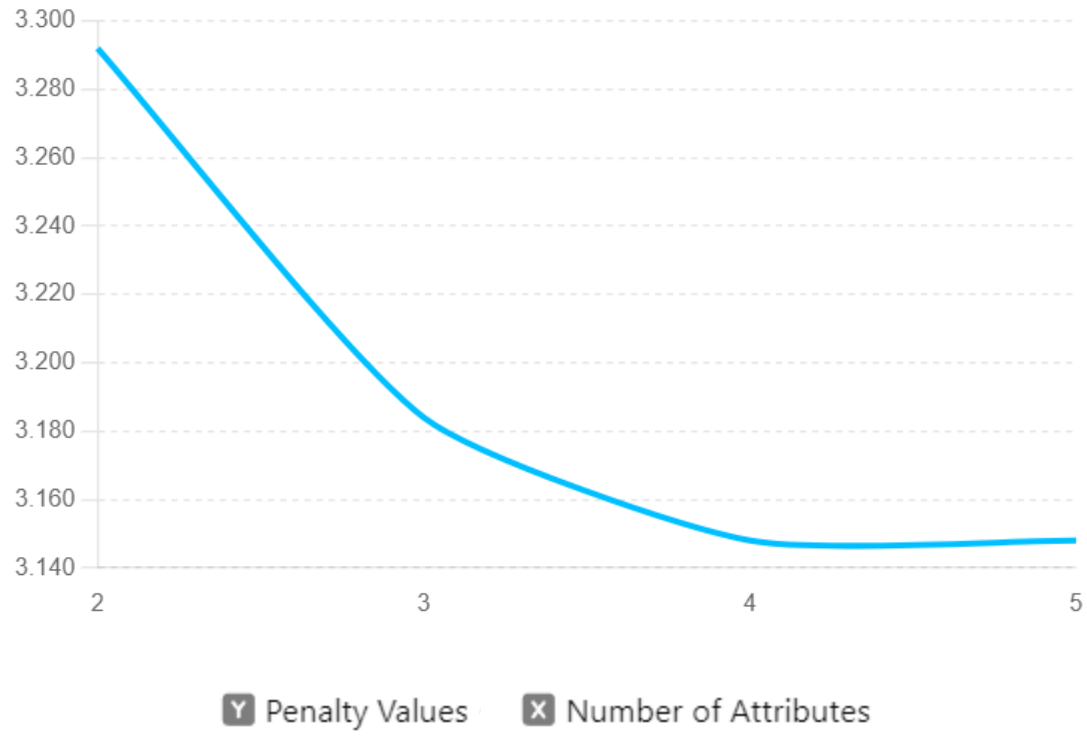
This metric evaluates the average size of equivalence classes relative to the total number of records and the number of equivalence classes, providing insight into the balance between privacy and data utility

General Purpose Quality Metrics

Università di Genova

We evaluated our model by using the discernability penalty metric. The first graph compares the model for varied k. In the second one, we compared the model with different number of attributes.





Random Perturbation

Università di Genova

Sensitive Data should be not anonymized in order to preserve utility, but in some cases, they can be used for re-identification. Hence, we applied a random perturbation technique.

```
Column: annual_income
  Original mean: 36433.602
  Perturbated mean: 36433.606
  Original variance: 266011947.98558158
  Perturbated variance: 266011793.8165806
```

The mean and variance of both columns are very closed.

Università di Genova

Thank you for your attention!

Università
di Genova