Anonymizing categorical data

DATA PRIVACY AND ANONYMIZATION IN PYTHON

Rebeca Gonzalez
Instructor

datacamp

Generalization

	Age	Gender	Department	Condition
0	30	F	Finance	Anxiety disorders
1	42	M	Production	Bronchitis
2	35	F	Marketing	Dysthymia
3	39	F	Production	Dysthymia
4	40	M	Marketing	Flu

	Age	Gender	Department	Condition
0	<40	F	Finance	Anxiety disorders
1	>=40	M	Production	Bronquitis
2	<40	F	Finance	Dysthymia
3	<40	F	Production	Dysthymia
4	>=40	M	Marketing	Flu

Generalization of categorical data

```
# See the dataset
hr.head()
```

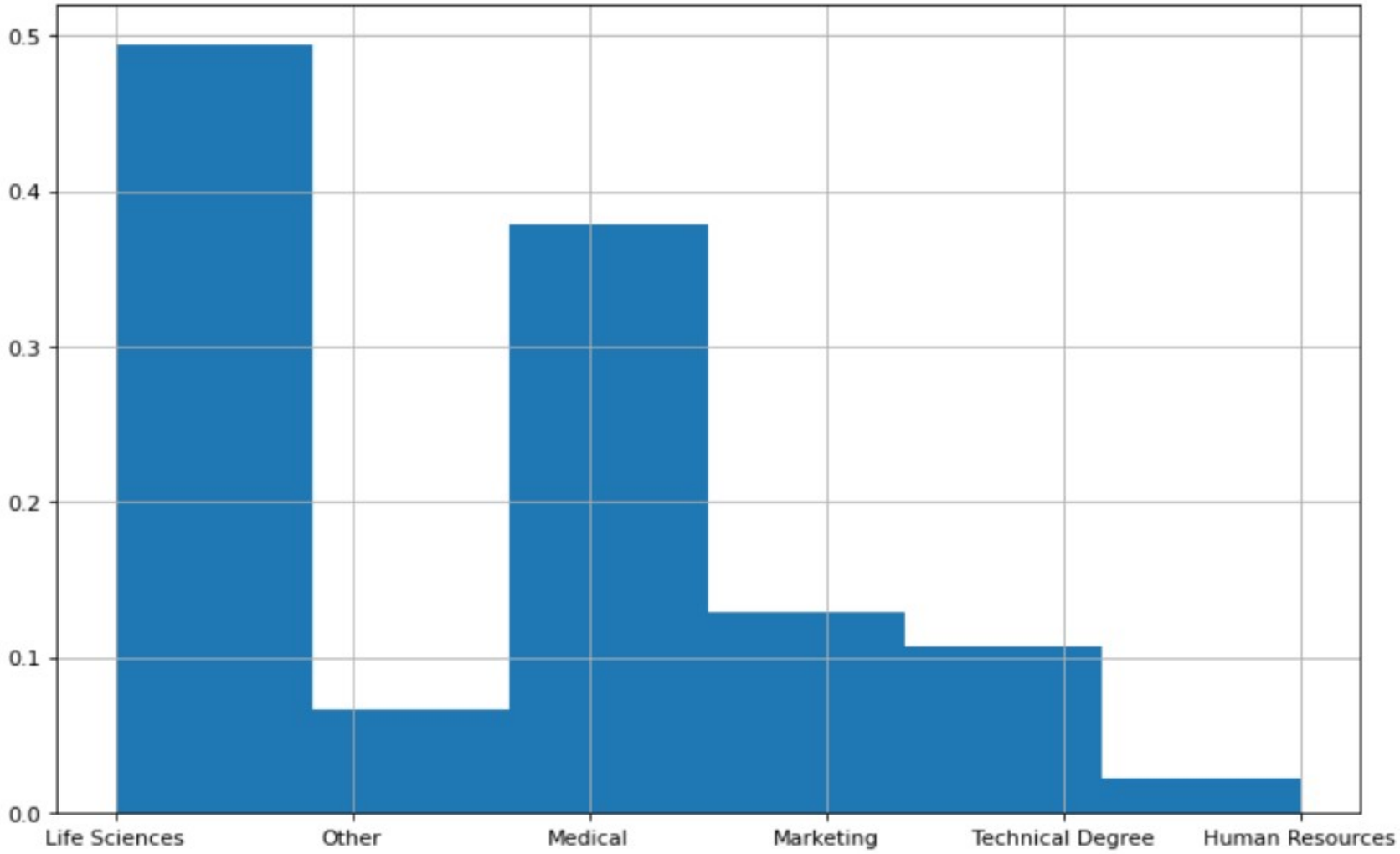
	Age	BusinessTravel	Department	EducationField	EmployeeNumber
0	41	Travel_Rarely	Sales	Life Sciences	1
1	49	Travel_Frequently	Research & Development	Life Sciences	2
2	37	Travel_Rarely	Research & Development	Other	4
3	33	Travel_Frequently	Research & Development	Life Sciences	5
4	27	Travel_Rarely	Research & Development	Medical	7

Categorical data

Limited or fixed number of possible values.

- race
- gender
- hometown
- age group
- educational level
- movies they like and preferences

Anonymizing categorical data



Anonymizing categorical data

	Department	EducationField
0	Sales	Life Sciences
1	Research & Development	Life Sciences
2	Research & Development	Other
3	Research & Development	Life Sciences
4	Research & Development	Medical

Original dataset

	Department	EducationField
0	Sales	Medical
1	Research & Development	Marketing
2	Research & Development	Life Sciences
3	Research & Development	Other
4	Research & Development	Life Sciences

Resulting dataset after sampling from the probability distribution of the `educationField` column in the original dataset.

Sample from data

The U.S. Census publicly releases samples of data that they collect about citizens.

Enable calculation of large-scale statistical patterns:

- averages
- variances
- clusters

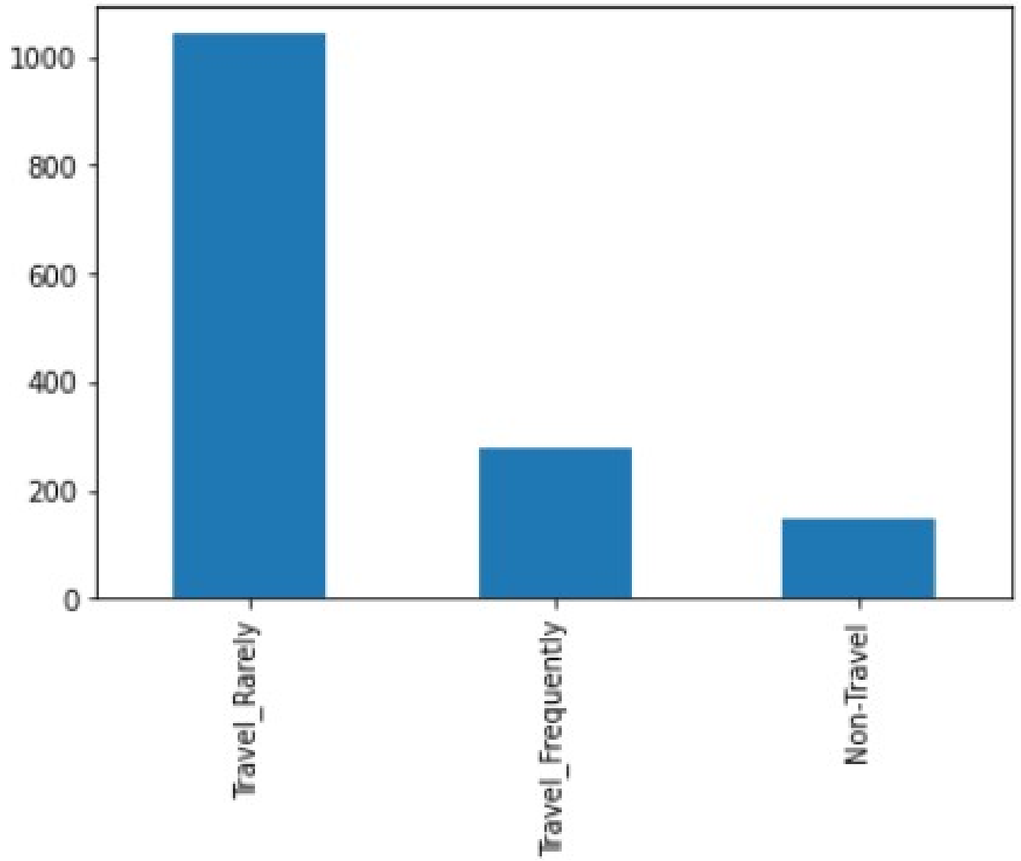
Explore the distribution

```
# Show the absolute frequencies of each unique value
hr['EducationField'].value_counts()
```

```
Life Sciences       606
Medical             464
Marketing           159
Technical Degree    132
Other                82
Human Resources      27
Name: EducationField, dtype: int64
```

Explore the distribution

```
# Generate a bar plot for the categories
df['BusinessTravel'].value_counts().plot(kind='bar')
```



Explore the distribution

```
# Obtain the absolute frequencies of each unique value
counts = hr['EducationField'].value_counts()

# Print the list of indexes
print(counts.index)
```

```
Index(['Life Sciences', 'Medical', 'Marketing',
       'Technical Degree', 'Other', 'Human Resources'],
       dtype='object')
```

Explore the distribution

```
# Probability distributions of each unique value
counts = df['EducationField'].value_counts(normalize=True)
```

```
Life Sciences       0.412245
Medical             0.315646
Marketing           0.108163
Technical Degree    0.089796
Other               0.055782
Human Resources     0.018367
Name: EducationField, dtype: float64
```

Explore the distribution

```
# Values of the frequencies of each unique value
df['EducationField'].value_counts(normalize=True).values
```

```
array([0.4122449 , 0.31564626, 0.10816327, 0.08979592, 0.05578231,
       0.01836735])
```

Sampling from the same distribution

```
# Sample from a probability distribution
hr_sample['EducationField']= np.random.choice(counts.index,
                                              p=counts.values,
                                              size=len(hr))

# See resulting dataset
hr.head()
```

	Age	BusinessTravel	Department	EducationField	EmployeeNumber
0	41	Travel_Rarely	Sales	Life Sciences	1
1	49	Travel_Frequently	Research & Development	Medical	2
2	37	Travel_Rarely	Research & Development	Marketing	4
3	33	Travel_Frequently	Research & Development	Technical Degree	5
4	27	Travel_Rarely	Research & Development	Medical	7

Sampling from the same distribution

```
# Show the absolute frequencies of each category
hr['EducationField'].value_counts()
```

```
Life Sciences       606
Medical             464
Marketing           159
Technical Degree    132
Other                82
Human Resources      27
Name: EducationField, dtype: int64
```

```
# Show the frequencies of the resulting column
hr_sample['EducationField'].value_counts()
```

```
Life Sciences       604
Medical             493
Marketing           158
Technical Degree    120
Other                61
Human Resources      34
Name: EducationField, dtype: int64
```

Let's practice!

DATA PRIVACY AND ANONYMIZATION IN PYTHON

Anonymizing continuous data

DATA PRIVACY AND ANONYMIZATION IN PYTHON

Rebeca Gonzalez
Instructor

datacamp

Continuous variables

- age
- height
- weight
- temperature
- date and time

Continuous variables

```
# See the dataset
hr.head()
```

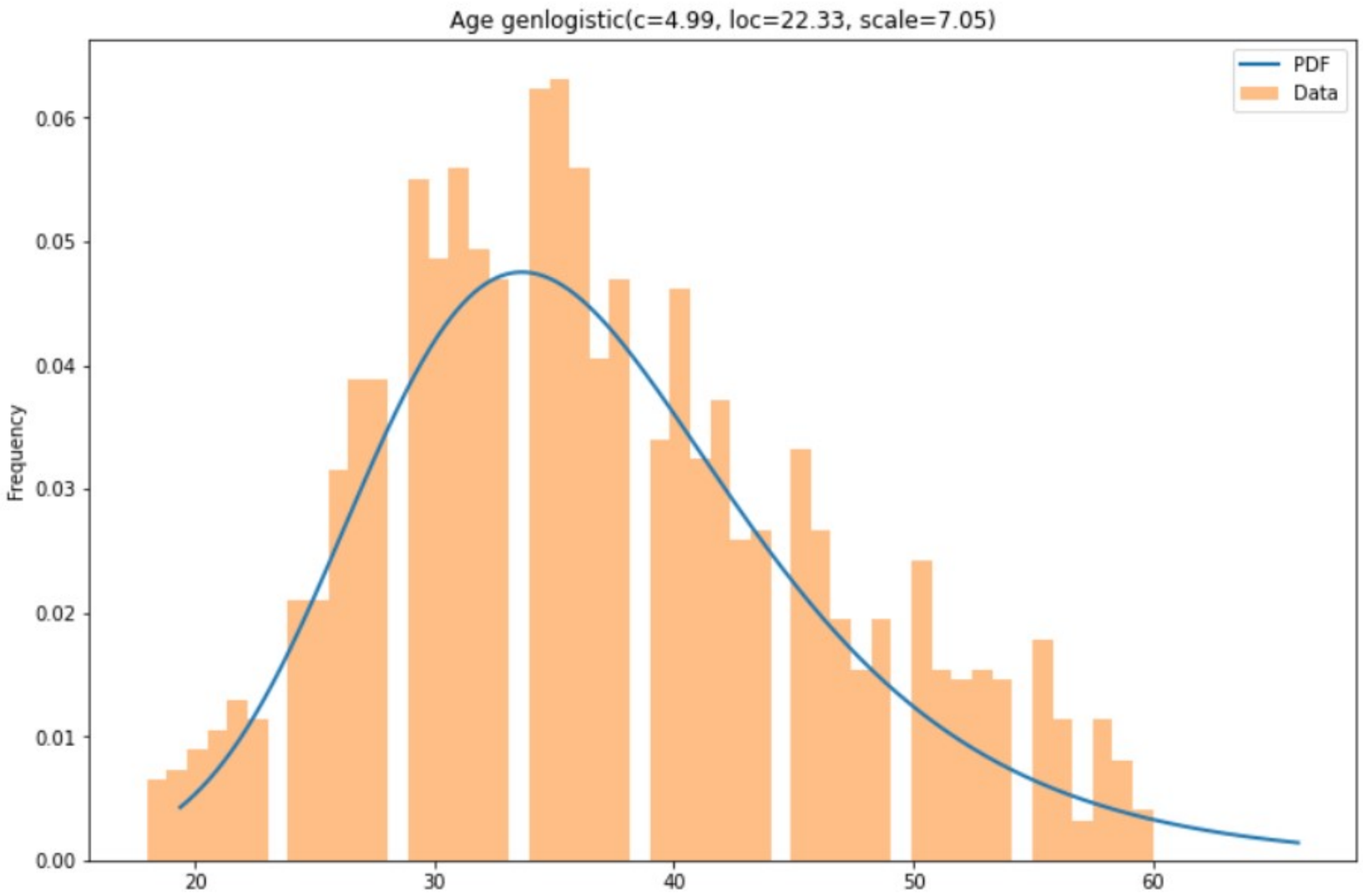
	Age	BusinessTravel	Department	EducationField	EmployeeNumber
0	41	Travel_Rarely	Sales	Life Sciences	1
1	49	Travel_Frequently	Research & Development	Life Sciences	2
2	37	Travel_Rarely	Research & Development	Other	4
3	33	Travel_Frequently	Research & Development	Life Sciences	5
4	27	Travel_Rarely	Research & Development	Medical	7

Continuous distributions

Use the best continuous distribution for our data.

- Create a histogram
- Try continuous functions, fitting them to the histogram.
- Keep the function that has the smallest error between itself and the histogram.

Continuous distribution



Applying a distribution

```
import scipy.stats

# Fit the genlogistic distribution to the continuous variable Age
params = scipy.stats.genlogistic.fit(hr['Age'])

# See the parameters of the continuous functions
print(params)
```

```
(4.9899067653418285, 22.328088531817 44, 7.046590524738551)
```

Sampling from the continuous distribution

```
# Sample from the genlogistic distribution
df['Age'] = scipy.stats.genlogistic.rvs(size=len(df.index), *params)

# See the resulting dataset
df['Age'].head()
```

	Age	BusinessTravel	Department	EducationField	EmployeeNumber
0	40.767259	Travel_Rarely	Sales	Life Sciences	1
1	45.730504	Travel_Frequently	Research & Development	Life Sciences	2
2	41.910050	Travel_Rarely	Research & Development	Other	4
3	35.275320	Travel_Frequently	Research & Development	Life Sciences	5
4	40.198134	Travel_Rarely	Research & Development	Medical	7

Sampling from the continuous distribution

```
# Round the values to obtain discrete values
df['Age'] = df['Age'].round()
```

	Age	BusinessTravel	Department	EducationField	EmployeeNumber
0	41	Travel_Rarely	Sales	Life Sciences	1
1	46	Travel_Frequently	Research & Development	Life Sciences	2
2	42	Travel_Rarely	Research & Development	Other	4
3	35	Travel_Frequently	Research & Development	Life Sciences	5
4	40	Travel_Rarely	Research & Development	Medical	7

Let's practice!

DATA PRIVACY AND ANONYMIZATION IN PYTHON

Introduction to K-anonymity

DATA PRIVACY AND ANONYMIZATION IN PYTHON

Rebeca Gonzalez
Data engineer

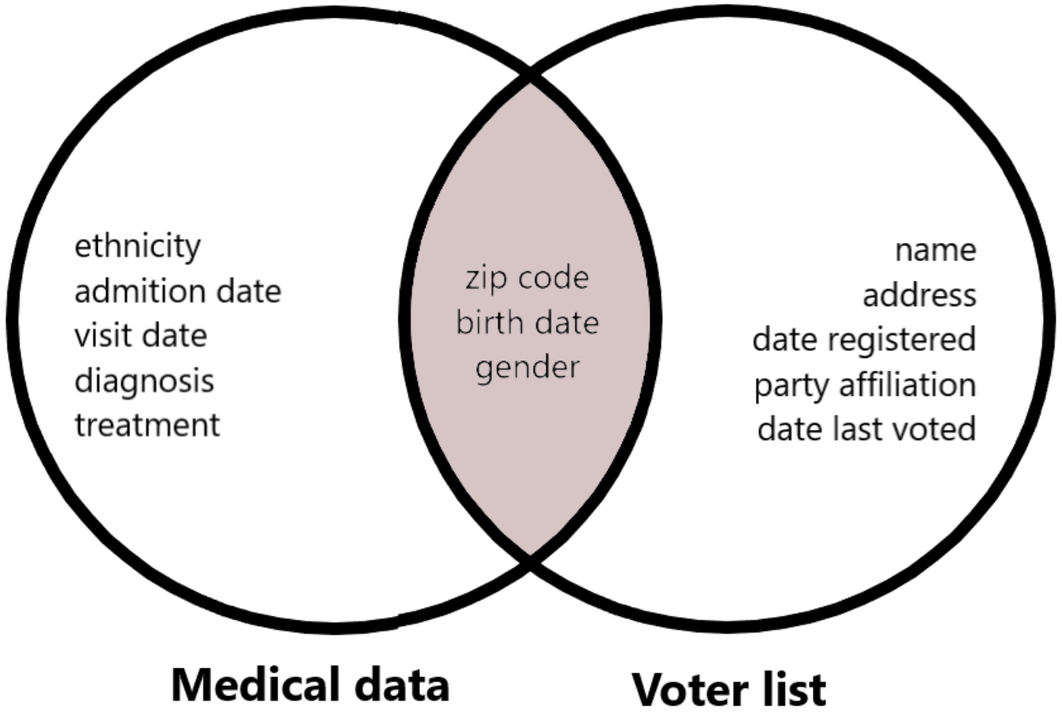
datacamp

Why is k-anonymity important?

Identifying the governor of Massachusetts

Why is it important?

- Six people had his birth date
- Only three were men
- He was the only one in his ZIP code



How to prevent this attack?

Only one record had the demographic values of the governor.

Deleting all the demographic information would leave the data useless for analysis.

What if there is a middle ground to make sure that the demographic values are no longer unique in the dataset?

Definition of k-anonymity

At least k individuals in the dataset share the set of attributes that might become identifying for each individual.

Definition of k-anonymity

K-anonymous dataset

2-anonymous:

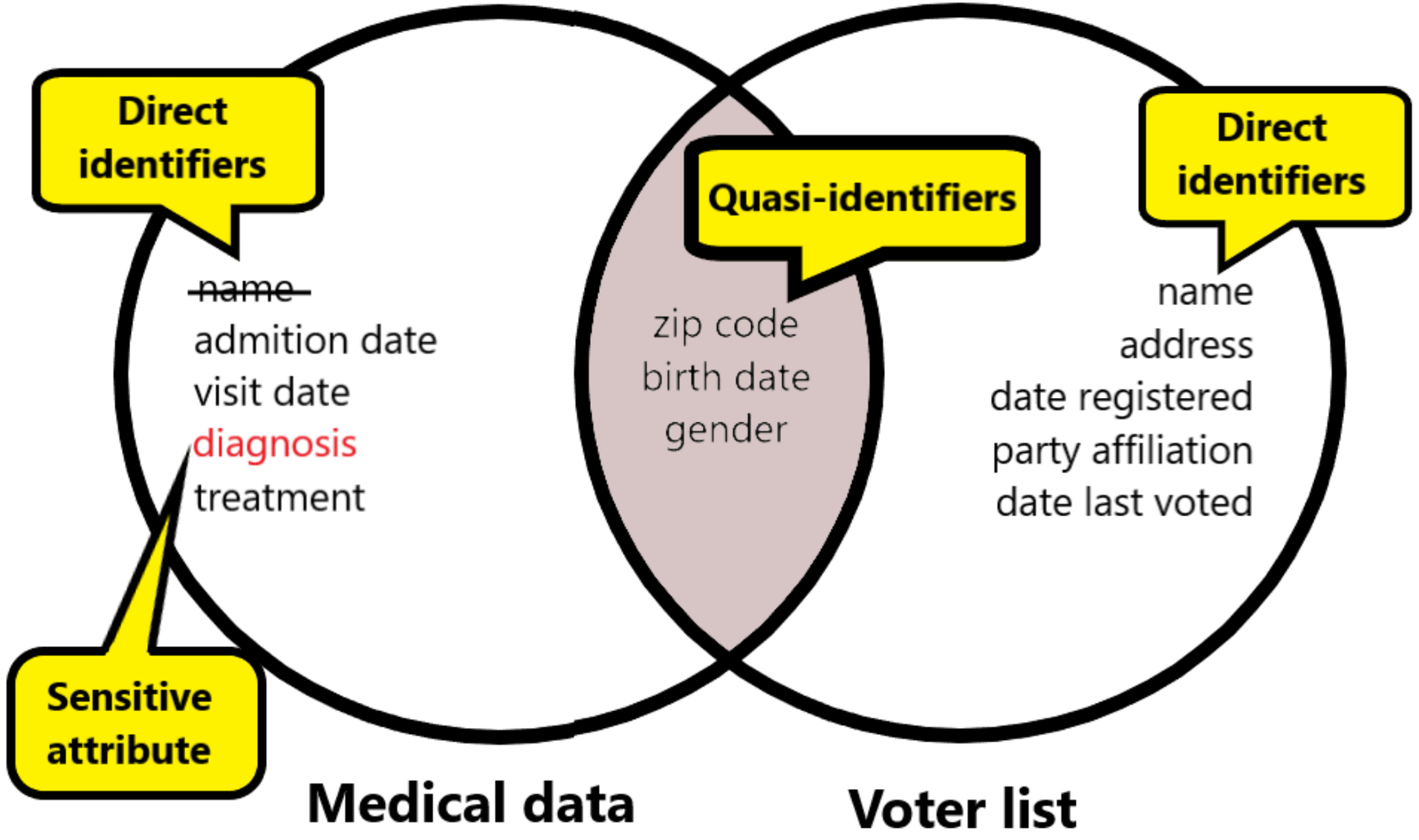
	ZIP code	Age
0	4217	34
1	4217	34
2	1742	77
3	1742	77

Not 2-anonymous:

	ZIP code	Age
0	4217	34
1	4217	34
2	1742	77
3	1743	77

Every combination of values for identifying columns in the dataset appears at least for k different records.

K-anonymity: Terminology



Medical data dataset

```
# Explore DataFrame
medical_df.head()
```

```
    Age    Department   Condition
0    34    Marketing    Anxiety disorders
1    46    Finance      Flu
2    41    Finance      Flu
3    62    Marketing    Anxiety disorders
4    44    Marketing    Anxiety disorders
```

Privacy attributes

Privacy attributes:

- Identifying: Could be SSN and ID numbers.
- Quasi-identifying: Age and department from our dataset
- Sensitives: Medical condition from our dataset

Exploring unique combination of quasi-identifiers

```
# Calculate how many unique combinations are for Age and Department
medical_df.groupby(['Age','Department']).size().reset_index(name='Count')
```

```
    Age    Department    Count
0    30    Production    2
1    31    Marketing     1
2    32    Marketing     1
3    32    Production    1
4    33    Production    1
5    34    Finance       1
6    34    Marketing     1
7    34    Production    1
8    35    Marketing     1
9    36    Finance       2
10    38    Finance      1
11    38    Production   1
```

Approach: generalization

```
# Generalize Age by creating 4 groups of intervals
medical_df['Age_group'] = pd.cut(medical_df['Age'], bins=4)

# Explore the dataset with intervals
medical_df.head()
```

	Age	Department	Condition	Age_group
0	34	Marketing	Anxiety disorders	(29.964, 39.0]
1	46	Finance	Flu	(39.0, 48.0]
2	41	Finance	Flu	(39.0, 48.0]
3	62	Marketing	Anxiety disorders	(57.0, 66.0]
4	44	Marketing	Anxiety disorders	(39.0, 48.0]

Approach: generalization

```
# Calculate how many unique combinations are for Age and Department
medical_df.groupby(['Age_group','Department']).size().reset_index(name='Count')
```

```
     Age_group          Department    Count
0    (29.964, 39.0]     Finance       4
1    (29.964, 39.0]     Marketing     4
2    (29.964, 39.0]     Production    6
3    (39.0, 48.0]       Finance       8
4    (39.0, 48.0]       Marketing     5
5    (39.0, 48.0]       Production    4
6    (48.0, 57.0]       Finance       3
7    (48.0, 57.0]       Marketing     2
8    (48.0, 57.0]       Production    4
```

Approach: generalization

```
# Set k to be 2, for a 2-anonymous dataset
k = 2

# Calculate how many unique combinations are for Age and Department
df_count = medical_df.groupby(['Age_group','Department']).size().reset_index(name='Count')

# Filter the rows that have count less than k
df_count[df_count['Count'] < k]
```

```
    Age_group    Department    Count
```

Let's k-anonymize!

DATA PRIVACY AND ANONYMIZATION IN PYTHON

Generalizing data using hierarchies

DATA PRIVACY AND ANONYMIZATION IN PYTHON

Rebeca Gonzalez
Data engineer

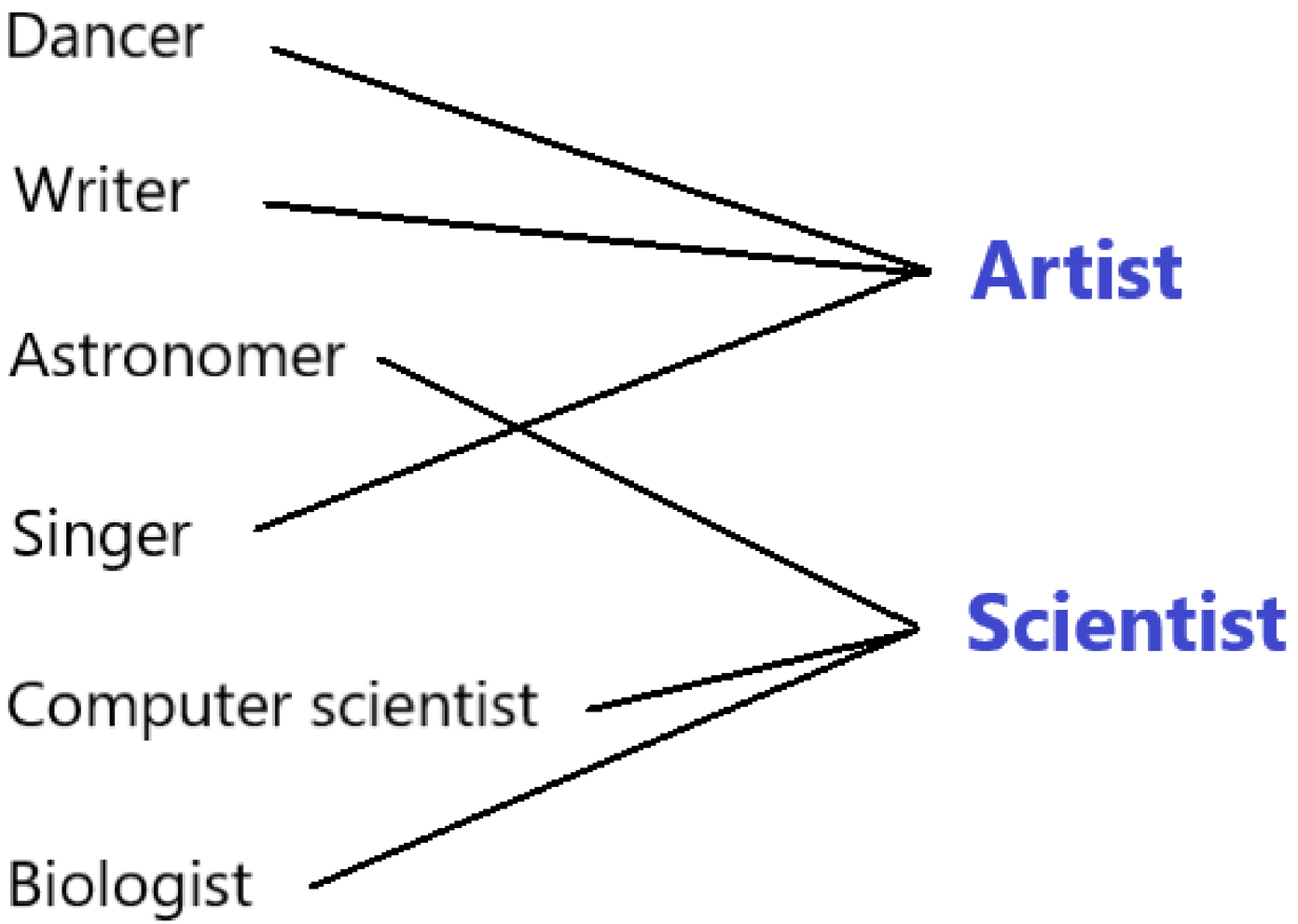
datacamp

Data generalization

Allows you to replace a data value with a less precise value.

- "Dancer" -> "Artist"
- "Archaeologist" -> "Scientist"
- "Allergist" -> "Doctor"

Data generalization with hierarchies



Refugee status dataset

```
# Explore dataset
refugee_df.head()
```

```
     Nationality    Gender    RefugeeStatus    Year
0    russian        F         0                2010
1    colombian      M         0                2008
2    vietnamese     F         1                2008
3    korean         M         0                2010
4    ethiopian      F         1                2012
```

Exploring the dataset

```
# Set k value to 4
k = 4

# Calculate how many unique combinations are for nationality and gender
df_count = df.groupby(['Nationality', 'Gender']).size().reset_index(name='Count')

# Filter the rows that have a count value less than k
df_count[df_count['Count'] < k]
```

```
      Nationality   Gender    Count
4     colombian     F         2
7     cuban         M         2
12    liberian      F         2
14    mexican       F         3
15    mexican       M         3
17    russian       M         1
20    ukranian      F         3
21    ukranian      M         1
```

Data generalization with hierarchies

```
# See the combination counts
df['Nationality'].value_counts()
```

```
belarusian    14
korean        14
ethiopian     10
chinese       10
vietnamese     9
taiwanese      9
colombian      6
cuban          6
russian        6
mexican        6
liberian       6
ukranian       4
Name: Nationality, dtype: int64
```

Data generalization with hierarchies

```
# Create hierarchies for origin countries
hierarchies = {'Europe': ['ukranian','russian','belarusian'],
               'America':['mexican','colombian','cuban'],
               'Asia': ['taiwanese','korean','chinese'],
               'Africa': ['ethiopian','liberian']}
```

Data generalization with hierarchies

```
# Creating hierachy father for each contry
origin_hierarchy = {}
for (key, countries) in hierarchies.items():
    for country in countries:
        origin_hierarchy[country] = key
origin_hierarchy
```

```
{'belarusian': 'Europe',
 'chinese': 'Asia',
 'colombian': 'America',
 'cuban': 'America',
 'ethiopian': 'Africa',
 'korean': 'Asia',
 'liberian': 'Africa',
 'mexican': 'America',
 'russian': 'Europe',
 'taiwanese': 'Asia',
 'ukranian': 'Europe'}
```

Data generalization with hierarchies

```
# Apply origin_hierarchy hierarchy generalization to Nationality
df['Nationality_generalized'] = df['Nationality'].map(origin_hierarchy)

# Explore resulting dataset
df.head()
```

	Nationality	Gender	RefugeeStatus	Year	Nationality_generalized
0	korean	M	1	2019	Asia
1	ethiopian	M	1	2003	Africa
2	cuban	M	0	2015	America
3	cuban	F	0	2001	America
4	korean	M	0	2013	Asia

Data generalization with hierarchies

```
# Calculate how many unique combinations are for Nationality_generalized and Gender
df_count = df.groupby(['Nationality_generalized', 'Gender']).size()
                                                            .reset_index(name='Count')

# Filter the combinations that appear less than k times
df_count[df_count['Count'] < k]
```

```
    Nationality_generalized    Gender    Count
```

More on K-anonymity

Types:

- Constraints approach
- Mondrian Multidimensional approach

How safe is K-anonymity?

Bob

Zipcode	*Age*
47678	27

A 3-anonymous patient table

Zipcode	Age	Disease
476**	2*	Heart Disease
476**	2*	Heart Disease
476**	2*	Heart Disease
4790*	≥40	Flu
4790*	≥40	Heart Disease
4790*	≥40	Cancer
476**	3*	Heart Disease
476**	3*	Cancer
476**	3*	Cancer

If we know Bob **is in this table**, he lives in a zip code starting with "**476**...", and he has **27**.
Then he surely has the displayed disease.

Let's practice!

DATA PRIVACY AND ANONYMIZATION IN PYTHON