Microdata Protection

Privacy Preserving Information Access: Homework 1

Reza Ghasemi

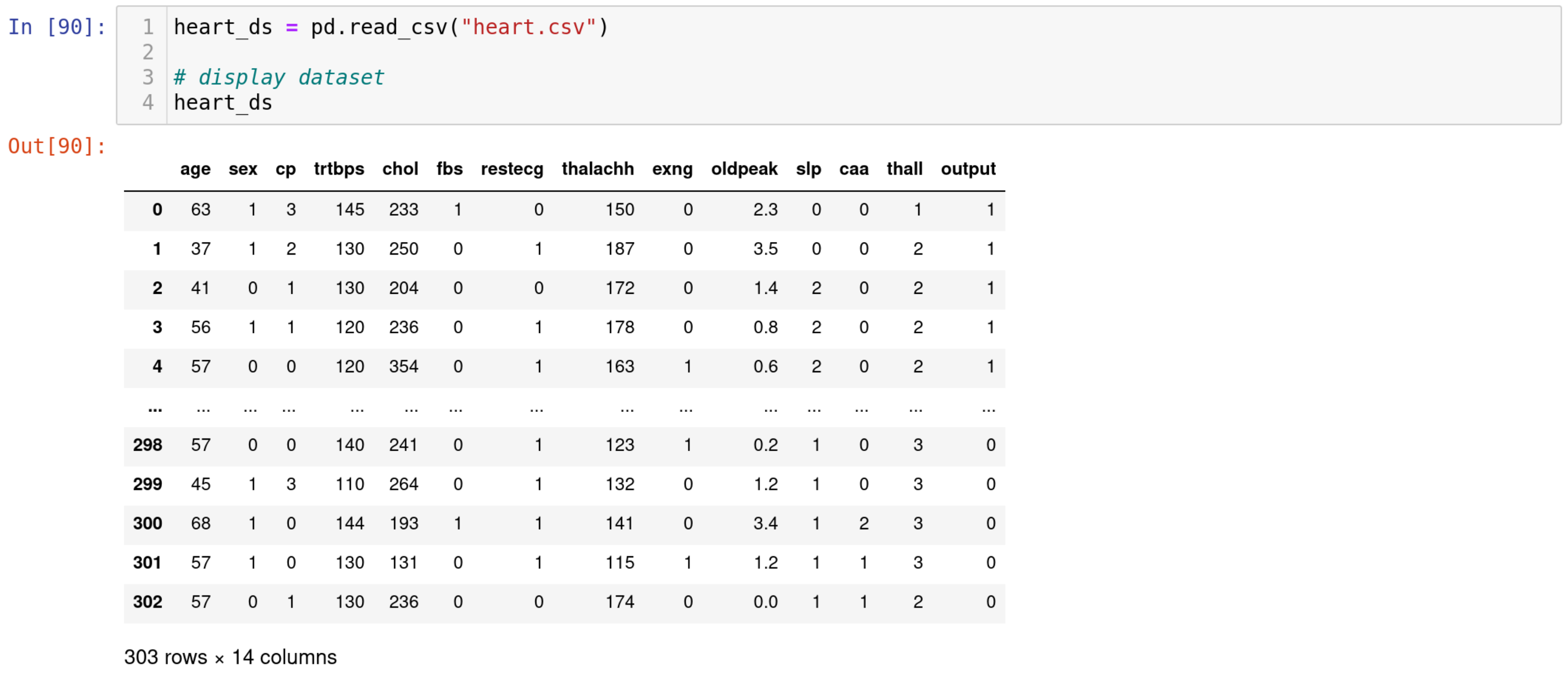
Università degli Studi di Padova

Dipartimento Matematica

A.Y. 2022/2023

Context

Heart Attack Analysis & Prediction Dataset



In [90]:
```
heart_ds = pd.read_csv("heart.csv")

# display dataset
heart_ds
```

Out[90]:

	age	sex	cp	trtbps	chol	fbs	restecg	thalachh	exng	oldpeak	slp	caa	thall	output
0	63	1	3	145	233	1	0	150	0	2.3	0	0	1	1
1	37	1	2	130	250	0	1	187	0	3.5	0	0	2	1
2	41	0	1	130	204	0	0	172	0	1.4	2	0	2	1
3	56	1	1	120	236	0	1	178	0	0.8	2	0	2	1
4	57	0	0	120	354	0	1	163	1	0.6	2	0	2	1
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
298	57	0	0	140	241	0	1	123	1	0.2	1	0	3	0
299	45	1	3	110	264	0	1	132	0	1.2	1	0	3	0
300	68	1	0	144	193	1	1	141	0	3.4	1	2	3	0
301	57	1	0	130	131	0	1	115	1	1.2	1	1	3	0
302	57	0	1	130	236	0	0	174	0	0.0	1	1	2	0

303 rows × 14 columns

Context

- Used in ML to predict whether a patient is going to have heart attack or not.
- Dataset contains the following information regarding patients:

  Age : Age of the patient

  Sex : Sex of the patient

  cp : Chest Pain type

  rtbps : resting blood pressure (in mm Hg)

  chol: cholesterol in mg/dl fetched via BMI sensor

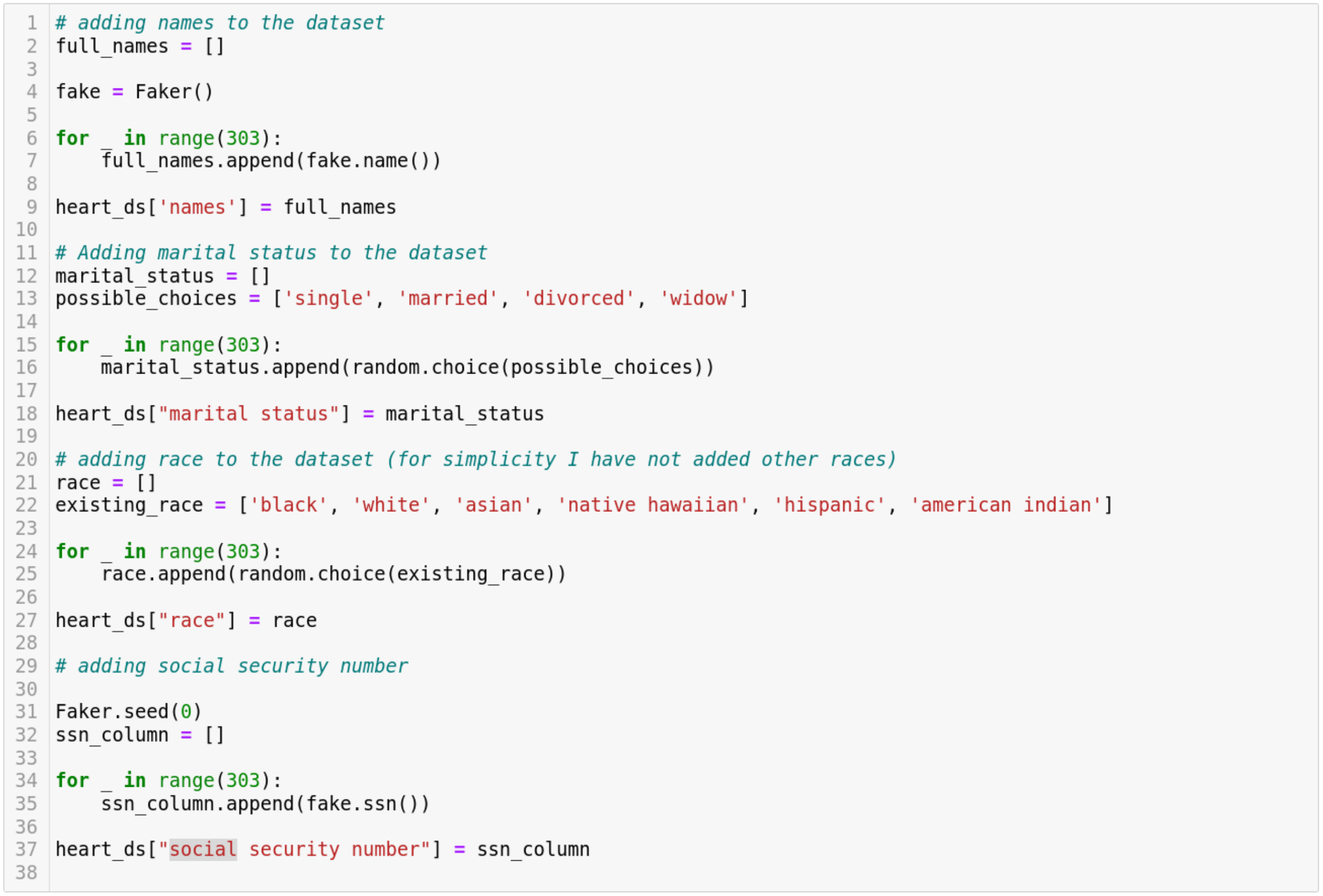
  output: diagnosis of heart disease

  …
- Identifiers have been removed (e.g. name, SSN). Based on the information, we see the likelihood of heart attack as *output* (medical data which must be protected).

DIPARTIMENTO MATEMATICA

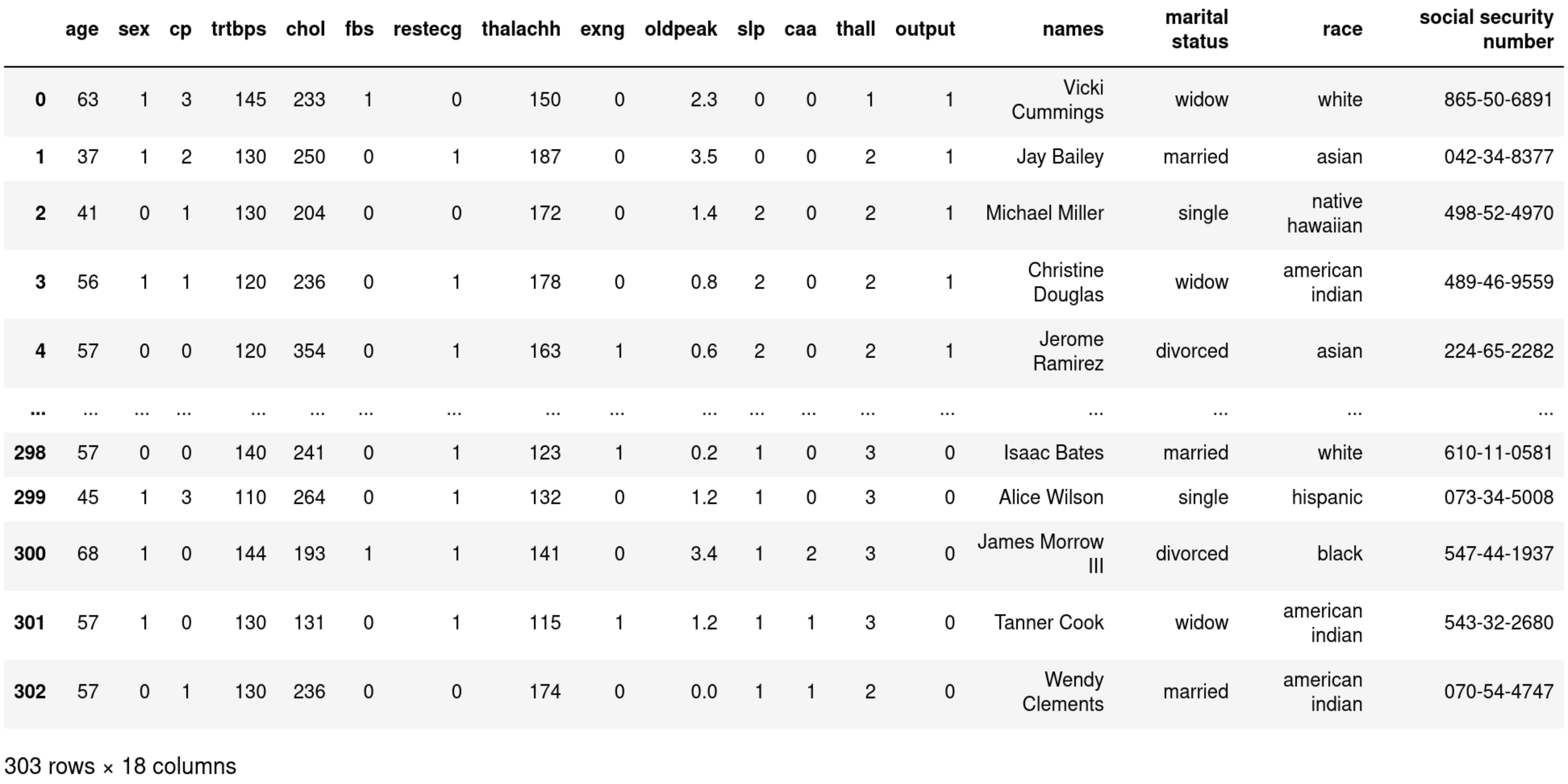
Dataset Expansion

- Fake information is generated and added



```python
# adding names to the dataset
full_names = []

fake = Faker()

for _ in range(303):
    full_names.append(fake.name())

heart_ds['names'] = full_names

# Adding marital status to the dataset
marital_status = []
possible_choices = ['single', 'married', 'divorced', 'widow']

for _ in range(303):
    marital_status.append(random.choice(possible_choices))

heart_ds["marital status"] = marital_status

# adding race to the dataset (for simplicity I have not added other races)
race = []
existing_race = ['black', 'white', 'asian', 'native hawaiian', 'hispanic', 'american indian']

for _ in range(303):
    race.append(random.choice(existing_race))

heart_ds["race"] = race

# adding social security number

Faker.seed(0)
ssn_column = []

for _ in range(303):
    ssn_column.append(fake.ssn())

heart_ds["social security number"] = ssn_column

```

Expanded dataset

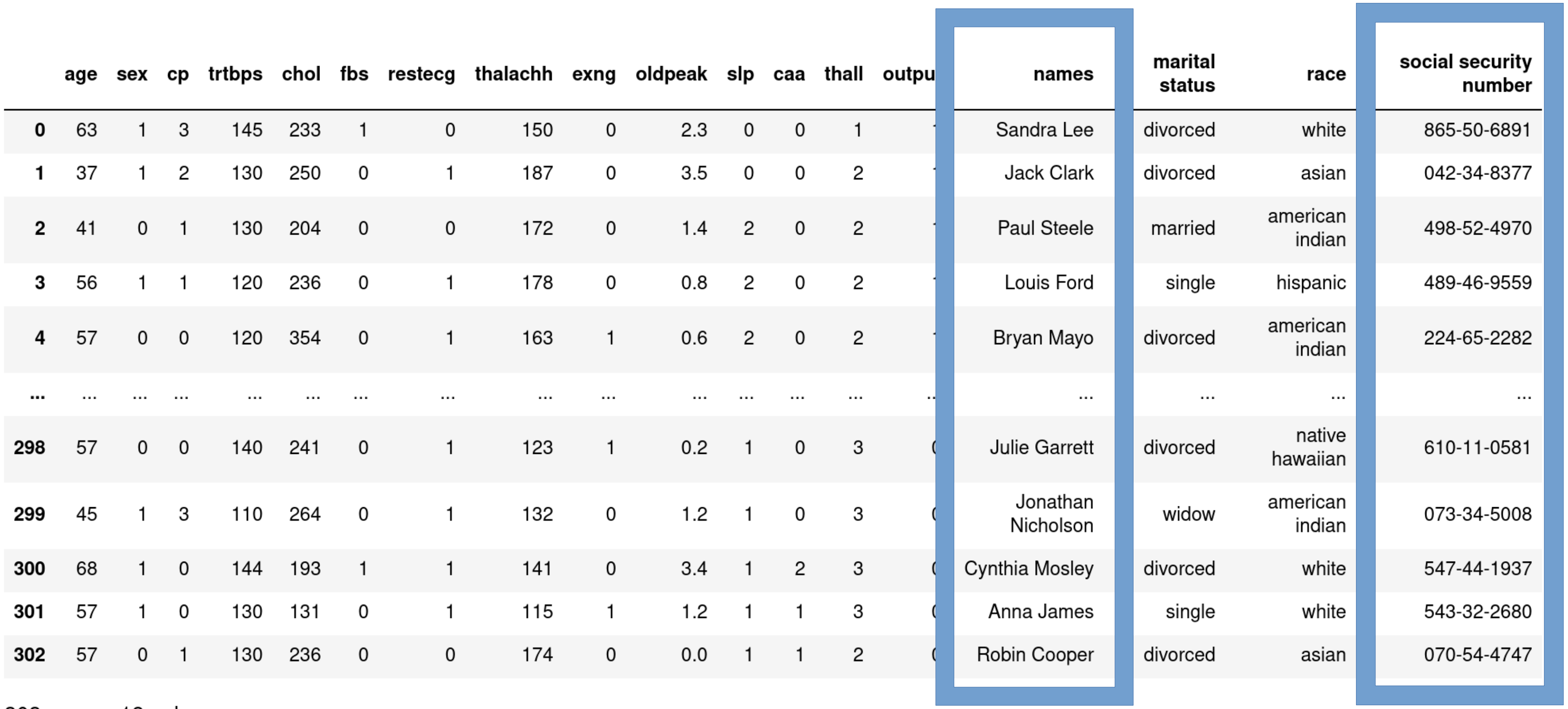


	age	sex	cp	trtbps	chol	fbs	restecg	thalachh	exng	oldpeak	slp	caa	thall	output	names	marital status	race	social security number
0	63	1	3	145	233	1	0	150	0	2.3	0	0	1	1	Vicki Cummings	widow	white	865-50-6891
1	37	1	2	130	250	0	1	187	0	3.5	0	0	2	1	Jay Bailey	married	asian	042-34-8377
2	41	0	1	130	204	0	0	172	0	1.4	2	0	2	1	Michael Miller	single	native hawaiian	498-52-4970
3	56	1	1	120	236	0	1	178	0	0.8	2	0	2	1	Christine Douglas	widow	american indian	489-46-9559
4	57	0	0	120	354	0	1	163	1	0.6	2	0	2	1	Jerome Ramirez	divorced	asian	224-65-2282
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
298	57	0	0	140	241	0	1	123	1	0.2	1	0	3	0	Isaac Bates	married	white	610-11-0581
299	45	1	3	110	264	0	1	132	0	1.2	1	0	3	0	Alice Wilson	single	hispanic	073-34-5008
300	68	1	0	144	193	1	1	141	0	3.4	1	2	3	0	James Morrow III	divorced	black	547-44-1937
301	57	1	0	130	131	0	1	115	1	1.2	1	1	3	0	Tanner Cook	widow	american indian	543-32-2680
302	57	0	1	130	236	0	0	174	0	0.0	1	1	2	0	Wendy Clements	married	american indian	070-54-4747

303 rows × 18 columns

Identifiers

UNIVERSITÀ DEGLI STUDI DI PADOVA

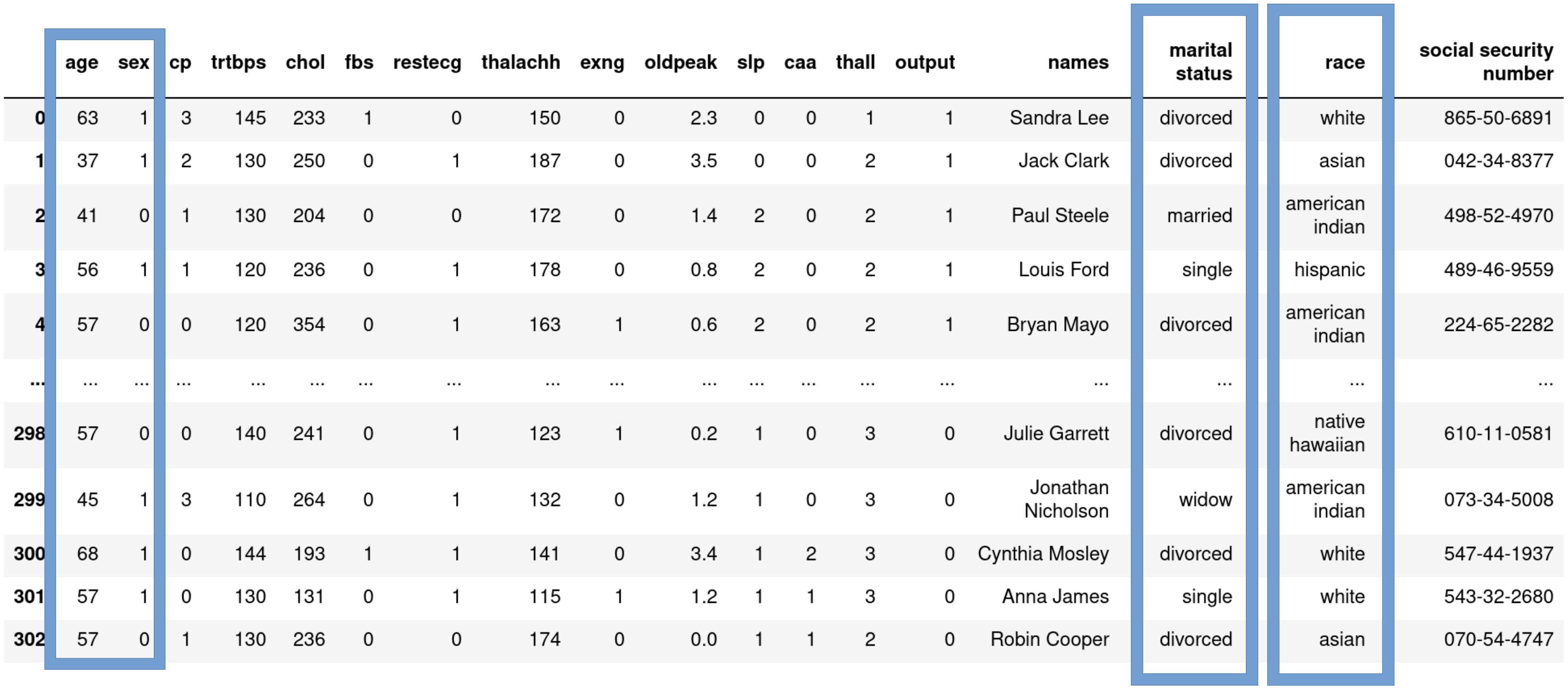


	age	sex	cp	trtbps	chol	fbs	restecg	thalachh	exng	oldpeak	slp	caa	thall	outpu	names	marital status	race	social security number
0	63	1	3	145	233	1	0	150	0	2.3	0	0	1	1	Sandra Lee	divorced	white	865-50-6891
1	37	1	2	130	250	0	1	187	0	3.5	0	0	2	1	Jack Clark	divorced	asian	042-34-8377
2	41	0	1	130	204	0	0	172	0	1.4	2	0	2	1	Paul Steele	married	american indian	498-52-4970
3	56	1	1	120	236	0	1	178	0	0.8	2	0	2	1	Louis Ford	single	hispanic	489-46-9559
4	57	0	0	120	354	0	1	163	1	0.6	2	0	2	1	Bryan Mayo	divorced	american indian	224-65-2282
...	...	...	...	...	...	...	...	...	...	...	...	...	...	..	...	...	...	...
298	57	0	0	140	241	0	1	123	1	0.2	1	0	3	(	Julie Garrett	divorced	native hawaiian	610-11-0581
299	45	1	3	110	264	0	1	132	0	1.2	1	0	3	(	Jonathan Nicholson	widow	american indian	073-34-5008
300	68	1	0	144	193	1	1	141	0	3.4	1	2	3	(	Cynthia Mosley	divorced	white	547-44-1937
301	57	1	0	130	131	0	1	115	1	1.2	1	1	3	(	Anna James	single	white	543-32-2680
302	57	0	1	130	236	0	0	174	0	0.0	1	1	2	(	Robin Cooper	divorced	asian	070-54-4747

303 rows × 18 columns

DIPARTIMENTO MATEMATICA

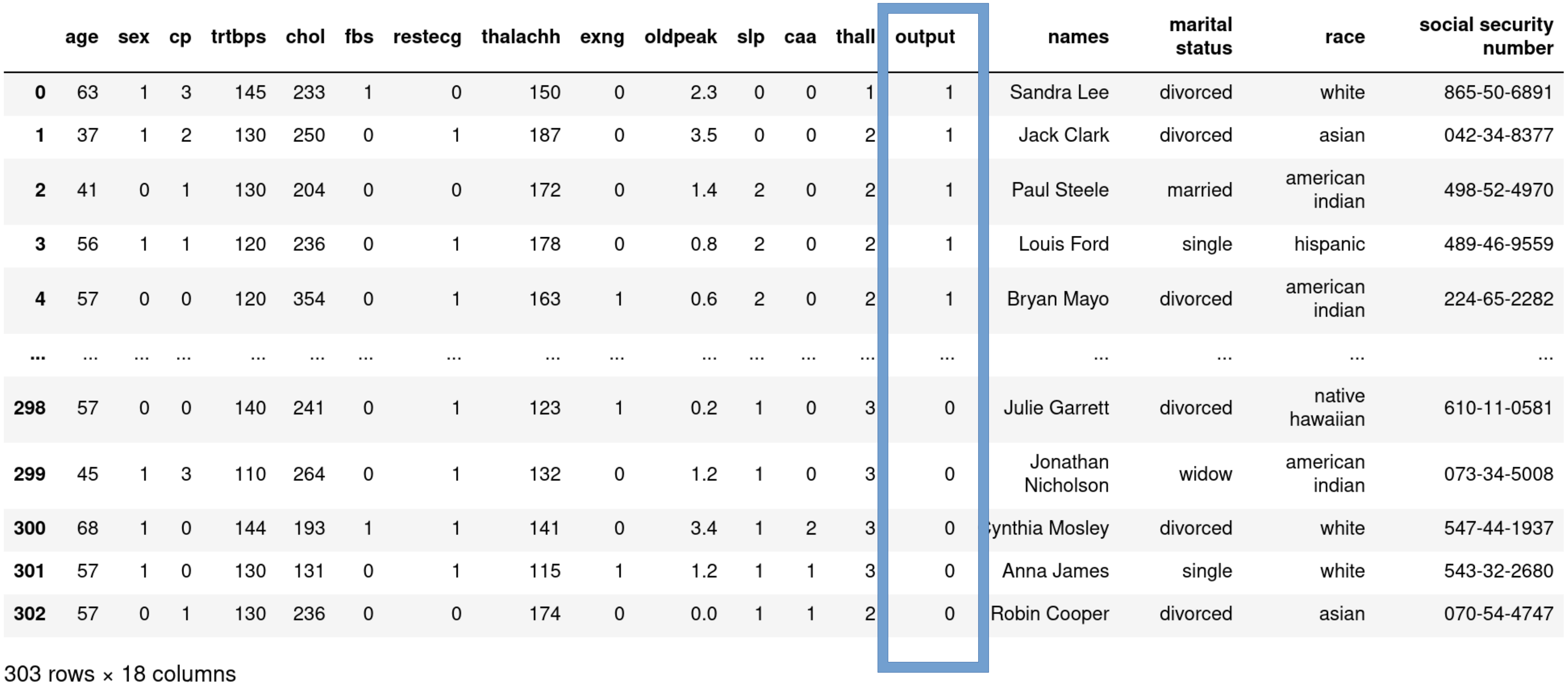
Quasi-identifiers



	age	sex	cp	trtbps	chol	fbs	restecg	thalachh	exng	oldpeak	slp	caa	thall	output	names	marital status	race	social security number
0	63	1	3	145	233	1	0	150	0	2.3	0	0	1	1	Sandra Lee	divorced	white	865-50-6891
1	37	1	2	130	250	0	1	187	0	3.5	0	0	2	1	Jack Clark	divorced	asian	042-34-8377
2	41	0	1	130	204	0	0	172	0	1.4	2	0	2	1	Paul Steele	married	american indian	498-52-4970
3	56	1	1	120	236	0	1	178	0	0.8	2	0	2	1	Louis Ford	single	hispanic	489-46-9559
4	57	0	0	120	354	0	1	163	1	0.6	2	0	2	1	Bryan Mayo	divorced	american indian	224-65-2282
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
298	57	0	0	140	241	0	1	123	1	0.2	1	0	3	0	Julie Garrett	divorced	native hawaiian	610-11-0581
299	45	1	3	110	264	0	1	132	0	1.2	1	0	3	0	Jonathan Nicholson	widow	american indian	073-34-5008
300	68	1	0	144	193	1	1	141	0	3.4	1	2	3	0	Cynthia Mosley	divorced	white	547-44-1937
301	57	1	0	130	131	0	1	115	1	1.2	1	1	3	0	Anna James	single	white	543-32-2680
302	57	0	1	130	236	0	0	174	0	0.0	1	1	2	0	Robin Cooper	divorced	asian	070-54-4747

303 rows × 18 columns

Confidential Attributes



	age	sex	cp	trtbps	chol	fbs	restecg	thalachh	exng	oldpeak	slp	caa	thall	output	names	marital status	race	social security number
0	63	1	3	145	233	1	0	150	0	2.3	0	0	1	1	Sandra Lee	divorced	white	865-50-6891
1	37	1	2	130	250	0	1	187	0	3.5	0	0	2	1	Jack Clark	divorced	asian	042-34-8377
2	41	0	1	130	204	0	0	172	0	1.4	2	0	2	1	Paul Steele	married	american indian	498-52-4970
3	56	1	1	120	236	0	1	178	0	0.8	2	0	2	1	Louis Ford	single	hispanic	489-46-9559
4	57	0	0	120	354	0	1	163	1	0.6	2	0	2	1	Bryan Mayo	divorced	american indian	224-65-2282
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
298	57	0	0	140	241	0	1	123	1	0.2	1	0	3	0	Julie Garrett	divorced	native hawaiian	610-11-0581
299	45	1	3	110	264	0	1	132	0	1.2	1	0	3	0	Jonathan Nicholson	widow	american indian	073-34-5008
300	68	1	0	144	193	1	1	141	0	3.4	1	2	3	0	ynthia Mosley	divorced	white	547-44-1937
301	57	1	0	130	131	0	1	115	1	1.2	1	1	3	0	Anna James	single	white	543-32-2680
302	57	0	1	130	236	0	0	174	0	0.0	1	1	2	0	Robin Cooper	divorced	asian	070-54-4747

303 rows × 18 columns

Data Release

- **Who releases data?**
  - Government agencies
  - Research institutions
  - Businesses (e.g. Netflix)
- **What harms can be done?**
  - Reputation of business and individual
  - Employability
  - Personal relationships
  - Physical harm

Adversaries

- Malicious actors
- Researchers may single out an individual by accident
- Doxxers may use information for harassment (e.g. sharing home address of target on 4chan)

Generalization

The idea behind generalization is putting multiple categories under one category. (e.g. instead of divorced, widow, single and married, we have “espoused” or “not married”)

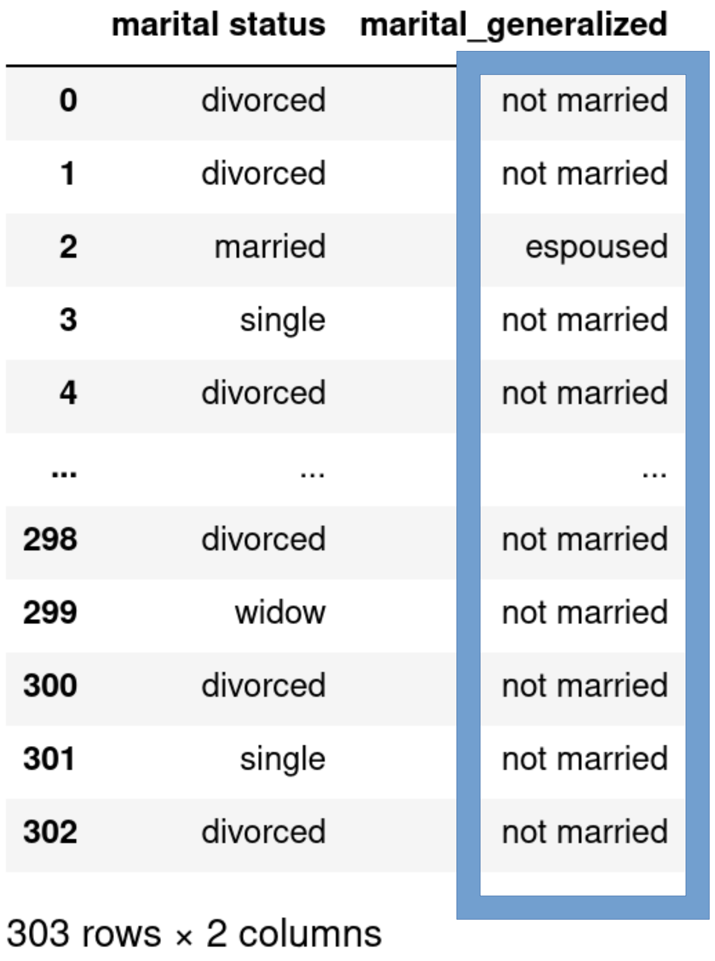
```
heart_ds['marital status'].unique()
```

```
array(['divorced', 'married', 'single', 'widow'], dtype=object)
```

```
# Prepare generalization map
generalization_map = {'married': 'espoused', 'widow': 'not married', 'single': 'not married', 'divorced':'not ma

heart_ds['marital_generalized'] = heart_ds['marital status'].replace(generalization_map)

# Display table
heart_ds[['marital status', 'marital_generalized']]
```

Generalization

Result:



	marital status	marital_generalized
0	divorced	not married
1	divorced	not married
2	married	espoused
3	single	not married
4	divorced	not married
...	...	...
298	divorced	not married
299	widow	not married
300	divorced	not married
301	single	not married
302	divorced	not married

303 rows × 2 columns

(n,k) dominance rule

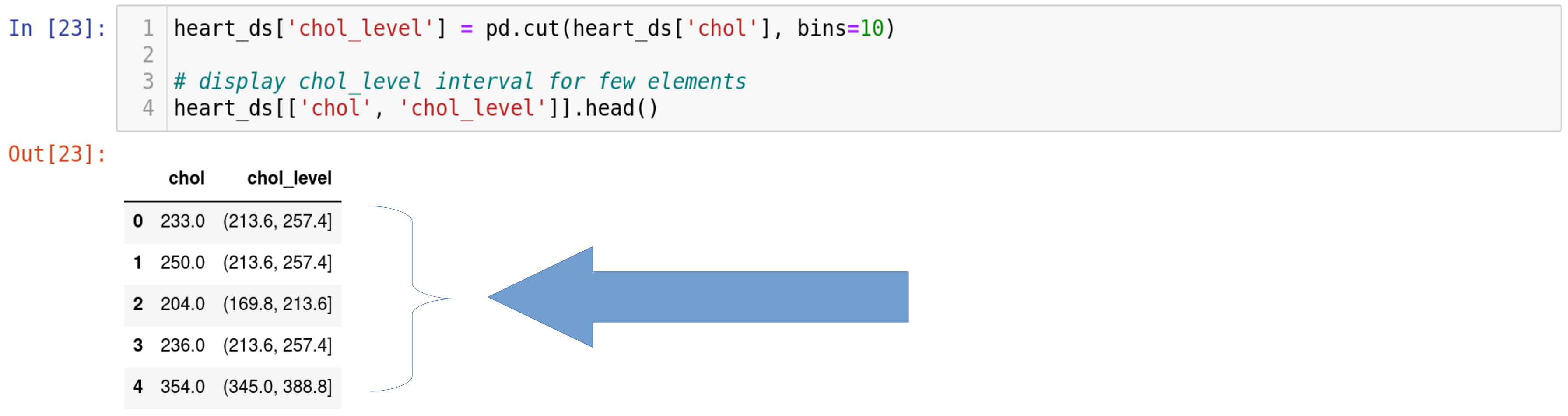
It measures sensitivity. primary suppression should be applied when the sum of the n largest contributors exceeds k% of the cell total.

```python
n = 3
k = 0.2
val = np.array(heart_ds[(heart_ds['race']=='black') &
                        (heart_ds['marital status']=='divorced')]['chol'])

print(f"is the cell sensible according to {n}-{k} rule?: {nk_rule(val, n, k)}\n")

k = 0.04
print(f"is the cell sensible according to {n}-{k} rule?: {nk_rule(val, n, k)}\n")
# not sensitive because we have four individual more than 0.25
```

```
contributions: [0.06864456 0.07998837 0.06573589 0.06806283 0.0744619  0.06573589
 0.0788249  0.06835369 0.0858057  0.0735893  0.05904596 0.0840605
 0.06020942 0.06748109]
number of individuals contributing more than 0.2: 0
is the cell sensible according to 3-0.2 rule?: True

contributions: [0.06864456 0.07998837 0.06573589 0.06806283 0.0744619  0.06573589
 0.0788249  0.06835369 0.0858057  0.0735893  0.05904596 0.0840605
 0.06020942 0.06748109]
number of individuals contributing more than 0.04: 14
is the cell sensible according to 3-0.04 rule?: False
```

Global Recoding

Domain is divided into disjointed intervals



```
In [23]:
heart_ds['chol_level'] = pd.cut(heart_ds['chol'], bins=10)

# display chol_level interval for few elements
heart_ds[['chol', 'chol_level']].head()
```

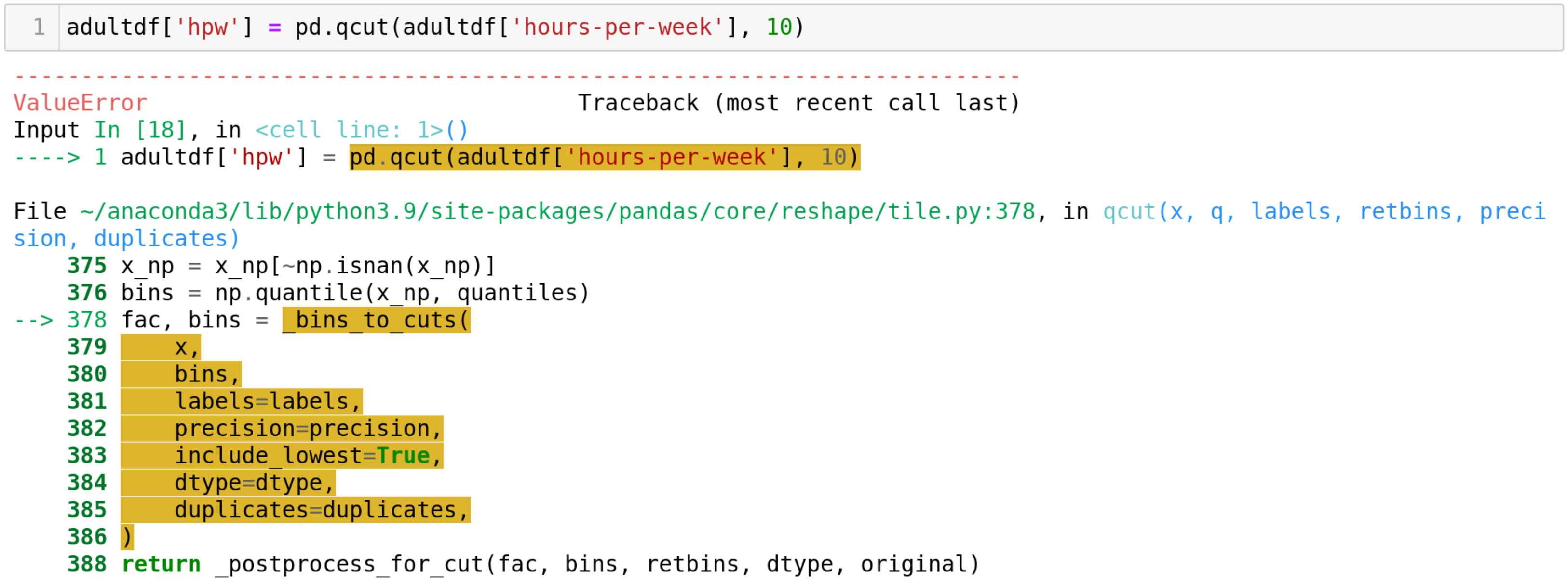
Out[23]:

	chol	chol_level
0	233.0	(213.6, 257.4]
1	250.0	(213.6, 257.4]
2	204.0	(169.8, 213.6]
3	236.0	(213.6, 257.4]
4	354.0	(345.0, 388.8]

… Why not use *qcut* instead?

Global Recoding (cont.)

As we saw in the example, *qcut* may give us errors



```
adultdf['hpw'] = pd.qcut(adultdf['hours-per-week'], 10)
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Input In [18], in <cell line: 1>()
----> 1 adultdf['hpw'] = pd.qcut(adultdf['hours-per-week'], 10)

File ~/anaconda3/lib/python3.9/site-packages/pandas/core/reshape/tile.py:378, in qcut(x, q, labels, retbins, preci
sion, duplicates)
    375 x_np = x_np[~np.isnan(x_np)]
    376 bins = np.quantile(x_np, quantiles)
--> 378 fac, bins = _bins_to_cuts(
    379     x,
    380     bins,
    381     labels=labels,
    382     precision=precision,
    383     include_lowest=True,
    384     dtype=dtype,
    385     duplicates=duplicates,
    386 )
    388 return _postprocess_for_cut(fac, bins, retbins, dtype, original)
```

Solutions?

Global Recoding (cont.)

- **Possible Solutions**:

  - Use *cut* instead

  - Fixed since panda>=0.20.0 (use duplicates='drop')

  - Use lower number for bins to have less number of same samples

- **Why errors?** Many samples are put in the same category while trying to maintain same quantity.
- **Issues?** outliers are not protected

Top coding and Bottom coding

Top coding:

```python
# top coding
greater_index = heart_ds[heart_ds['chol']>240].index.to_list()
```

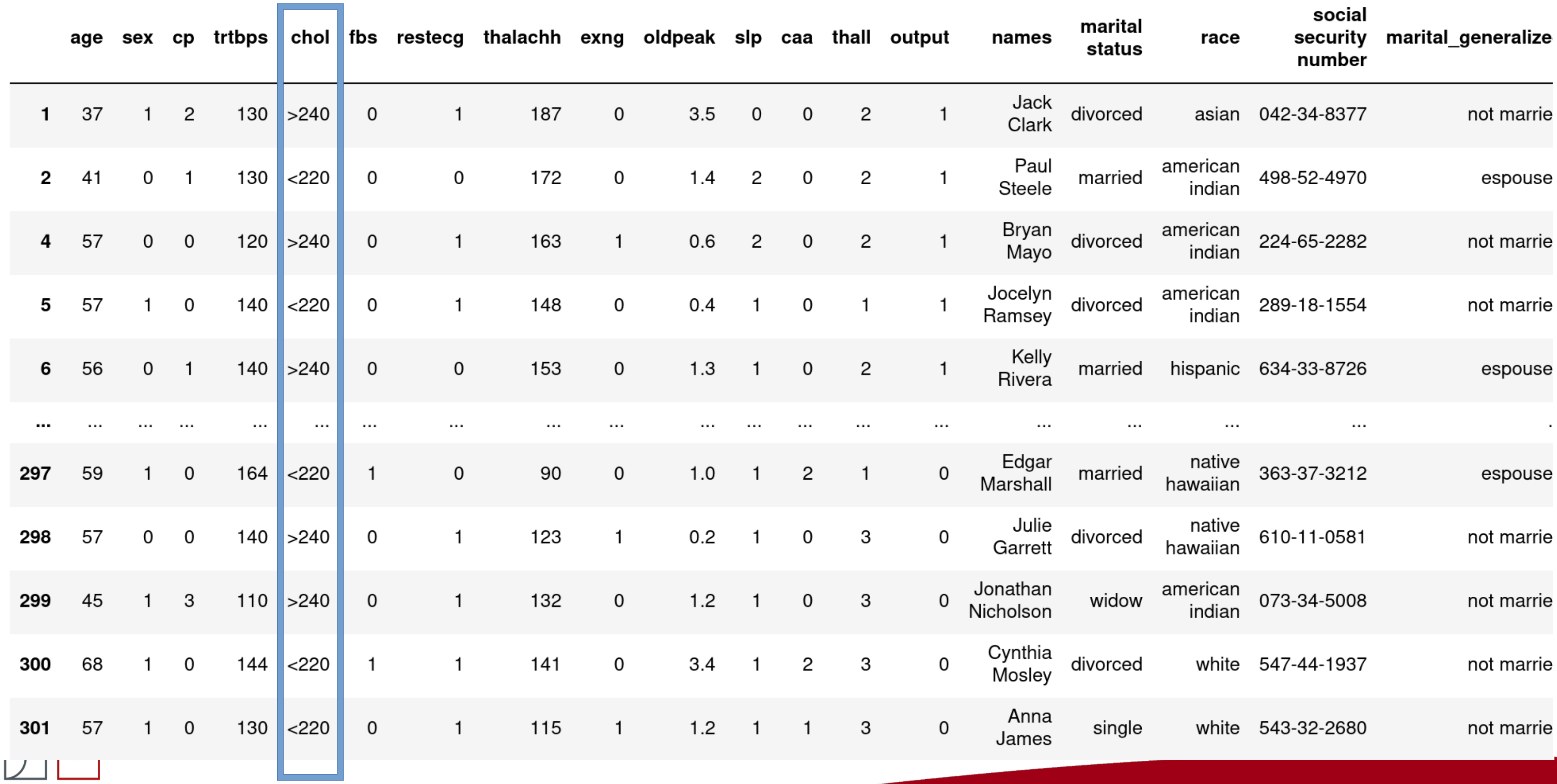
Bottom coding:

```python
# bottom coding
below_index = heart_ds[heart_ds['chol'] < 220].index.to_list()
```

```python
heart_ds.loc[greater_index, 'chol'] = ">240"
heart_ds.loc[below_index, 'chol'] = "<220"
heart_ds[(heart_ds['chol']==">240") | (heart_ds['chol']=="<220")]
```

Top coding and Bottom coding (cont.)

We can use top coding and bottom coding to protect them



	age	sex	cp	trtbps	chol	fbs	restecg	thalachh	exng	oldpeak	slp	caa	thall	output	names	marital status	race	social security number	marital_generalize
1	37	1	2	130	>240	0	1	187	0	3.5	0	0	2	1	Jack Clark	divorced	asian	042-34-8377	not marrie
2	41	0	1	130	<220	0	0	172	0	1.4	2	0	2	1	Paul Steele	married	american indian	498-52-4970	espouse
4	57	0	0	120	>240	0	1	163	1	0.6	2	0	2	1	Bryan Mayo	divorced	american indian	224-65-2282	not marrie
5	57	1	0	140	<220	0	1	148	0	0.4	1	0	1	1	Jocelyn Ramsey	divorced	american indian	289-18-1554	not marrie
6	56	0	1	140	>240	0	0	153	0	1.3	1	0	2	1	Kelly Rivera	married	hispanic	634-33-8726	espouse
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	.
297	59	1	0	164	<220	1	0	90	0	1.0	1	2	1	0	Edgar Marshall	married	native hawaiian	363-37-3212	espouse
298	57	0	0	140	>240	0	1	123	1	0.2	1	0	3	0	Julie Garrett	divorced	native hawaiian	610-11-0581	not marrie
299	45	1	3	110	>240	0	1	132	0	1.2	1	0	3	0	Jonathan Nicholson	widow	american indian	073-34-5008	not marrie
300	68	1	0	144	<220	1	1	141	0	3.4	1	2	3	0	Cynthia Mosley	divorced	white	547-44-1937	not marrie
301	57	1	0	130	<220	0	1	115	1	1.2	1	1	3	0	Anna James	single	white	543-32-2680	not marrie

Local Suppression

A well known technique in which value of sensitive cell is removed.

Example: Removing *chol level* higher than 240 for black and divorced individuals.

```
# [243 198 227 265 214 172 254 174 249 231 237]
heart_ds.loc[(heart_ds['race']=='black')
          & (heart_ds['marital status']=='divorced')
          & (heart_ds['chol'] > 240), 'chol']
```

```
11      275
50      256
131     271
140     295
156     253
231     289
Name: chol, dtype: int64
```

DIPARTIMENTO MATEMATICA

Local Suppression (cont.)

```python
heart_ds.loc[(heart_ds['race']=='black')
          & (heart_ds['marital status']=='divorced')
          & (heart_ds['chol'] > 240), 'chol'] = np.NaN
```

```python
# Display

heart_ds.loc[(heart_ds['race']=='black')
          & (heart_ds['marital status']=='divorced')
          & (heart_ds['chol'] > 240), 'chol']
```

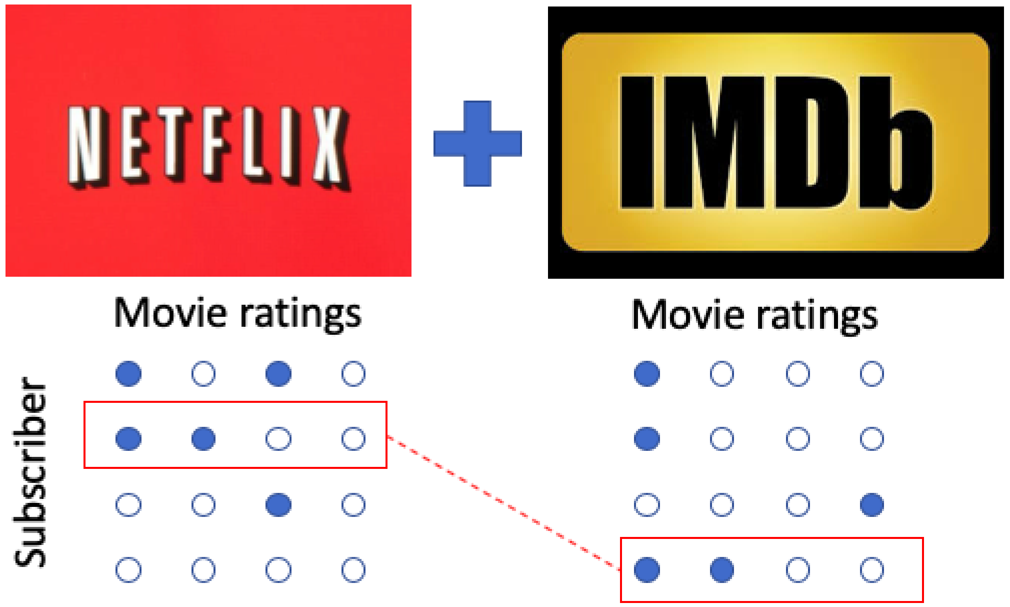
```
Series([], Name: chol, dtype: float64)
```

```python
heart_ds.loc[(heart_ds['race']=='black')
          & (heart_ds['marital status']=='divorced')]
```

	age	sex	cp	trtbps	chol	fbs	restecg	thalachh	exng	oldpeak	slp	caa	thall	output	names	marital status	race	social security number	marital_generalized	cl
3	56	1	1	120	236.0	0	1	178	0	0.8	2	0	2	1	Albert Montes	divorced	black	489-46-9559	not married	
11	48	0	2	130	NaN	0	1	139	0	0.2	2	0	2	1	Daniel Hale	divorced	black	363-56-5181	not married	
17	66	0	3	150	226.0	0	1	114	0	2.6	0	0	2	1	Sergio Wolfe	divorced	black	728-86-0019	not married	
49	53	0	0	138	234.0	0	0	160	0	0.0	2	0	2	1	Heather Valentine	divorced	black	120-91-3613	not married	
50	51	0	2	130	NaN	0	0	149	0	0.5	2	0	2	1	Gabriel Durham	divorced	black	381-22-5449	not married	

Record Linkage

Two sources of information: one anonymized, another public are used for comparison (e.g. Netflix and IMDB)



Record Linkage (cont.)

```python
# We import first dataset
first_dataset = pd.DataFrame(heart_ds)
names_from_first_list = first_d['names'].tolist() # 303 elements in list
```

```python
# We create our second dataset (e.g. database of a bank which was leaked)
def generate_dataset(n):
    output = [{"address":fake.address(),
              "city":fake.city(),
              "iban":fake.iban(),
              "swift":fake.swift(length=8),
              "cc-provider":fake.credit_card_provider()} for x in range(n)]
    return output

second_dataset = pd.DataFrame(generate_dataset(303))

# append names from first list to second list to perform record linkage
second_dataset['names'] = names_from_first_list


# adding race to the dataset (for simplicity I have not added other races)
race_second = []
existing_race_2 = ['black', 'white', 'asian']

for _ in range(303):
    race_second.append(random.choice(existing_race_2))

second_dataset["race"] = race_second

print(second_dataset)
```

Record Linkage (cont.)

Records are compared using *recordlinkage* module

```python
import recordlinkage

indexer = recordlinkage.Index()
indexer.block("names")
candidate_links = indexer.index(first_dataset, second_dataset)
```

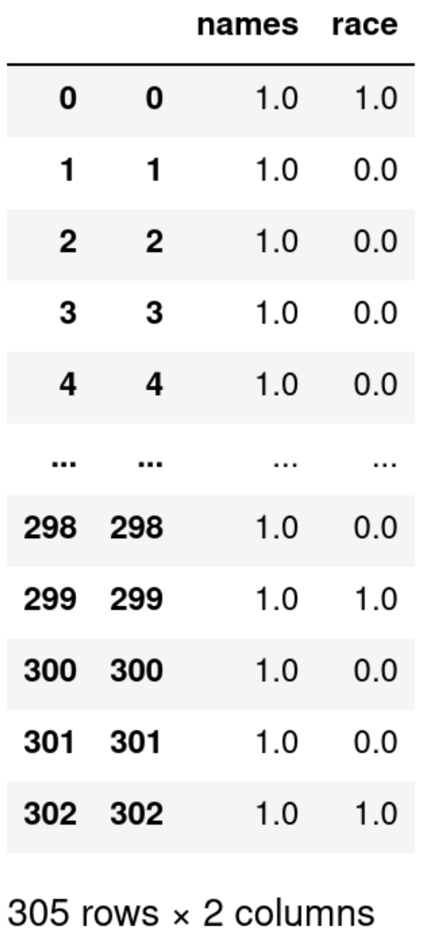
```python
compare_cl = recordlinkage.Compare()


compare_cl.string("names", "names", method="jarowinkler", threshold=0.9, label="names")
compare_cl.string("race", "race", method="jarowinkler", threshold=0.9, label="race")

# Compare the records
features = compare_cl.compute(candidate_links, first_dataset, second_dataset)

# display features
features.describe()
```

```python
features.sum(axis=1).value_counts().sort_index(ascending=False)

features[features.sum(axis=1) >= 0]
```

Record Linkage (cont.)

We used same name for both sets.

name is the same (1.0) while race for some indexes is different.



		names	race
0	**0**	1.0	1.0
1	**1**	1.0	0.0
2	**2**	1.0	0.0
3	**3**	1.0	0.0
4	**4**	1.0	0.0
...	**...**	...	...
298	**298**	1.0	0.0
299	**299**	1.0	1.0
300	**300**	1.0	0.0
301	**301**	1.0	0.0
302	**302**	1.0	1.0

305 rows × 2 columns

In your opinion, what are possible solutions to prevent record linkage?

DIPARTIMENTO MATEMATICA

Conclusions

- Privacy is slowly gaining attention (not just in academia)
- New libraries are beginning to be developed (e.g. PyMASq)
- Many possible attack scenarios and defenses are yet to be discovered.