Privacy-Preserving Data Processing

In this tutorial, you will learn how to apply off-the-shelf anonymization techniques using the most popular libraries: ARX and DiffPrivLib.

Dataset

You will use a dataset reporting various details on the passengers of the Titanic in its first and only trip.

The dataset is taken from this Kaggle competition. Download it using the code we provide you and install dependencies.

In [1]:

```
! curl -L https://www.dropbox.com/s/nnmcywh6ryveit4/titanic_clean.csv?dl=1 > titanic.csv
! pip install --user pandas numpy scikit-learn fastplot
! pip install --user sphinx sphinx_rtd_theme nbsphinx pandoc pytest-cov uplink==0.9.0
! pip install --user pyarxaas --no-deps > /dev/null
! pip install --user --upgrade diffprivlib > /dev/null
```

```
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100   141    0   141    0     0    542      0 --:--:-- --:--:-- --:--:--   544
100   323  100   323    0     0    487      0 --:--:-- --:--:-- --:--:--   487
100 64499  100 64499    0     0  63602      0  0:00:01  0:00:01 --:--:--  417k
Requirement already satisfied: pandas in /opt/conda/lib/python3.9/site-packages (1.3.4)
Requirement already satisfied: numpy in /opt/conda/lib/python3.9/site-packages (1.20.3)
Requirement already satisfied: scikit-learn in /opt/conda/lib/python3.9/site-packages (1.0)
Collecting fastplot
  Downloading fastplot-1.1.0.tar.gz (10 kB)
  Preparing metadata (setup.py) ... done
Requirement already satisfied: python-dateutil>=2.7.3 in /opt/conda/lib/python3.9/site-packages (from pandas) (2.8.2)
Requirement already satisfied: pytz>=2017.3 in /opt/conda/lib/python3.9/site-packages (from pandas) (2021.3)
Requirement already satisfied: scipy>=1.1.0 in /opt/conda/lib/python3.9/site-packages (from scikit-learn) (1.7.1)
Requirement already satisfied: threadpoolctl>=2.0.0 in /opt/conda/lib/python3.9/site-packages (from scikit-learn) (3.0.
0)
Requirement already satisfied: joblib>=0.11 in /opt/conda/lib/python3.9/site-packages (from scikit-learn) (1.1.0)
Requirement already satisfied: matplotlib in /opt/conda/lib/python3.9/site-packages (from fastplot) (3.4.3)
Requirement already satisfied: statsmodels in /opt/conda/lib/python3.9/site-packages (from fastplot) (0.13.0)
Requirement already satisfied: seaborn in /opt/conda/lib/python3.9/site-packages (from fastplot) (0.11.2)
Requirement already satisfied: six>=1.5 in /opt/conda/lib/python3.9/site-packages (from python-dateutil>=2.7.3->pandas)
(1.16.0)
Requirement already satisfied: kiwisolver>=1.0.1 in /opt/conda/lib/python3.9/site-packages (from matplotlib->fastplot)
(1.3.2)
```

```
Requirement already satisfied: pyparsing>=2.2.1 in /opt/conda/lib/python3.9/site-packages (from matplotlib->fastplot)
(2.4.7)
Requirement already satisfied: pillow>=6.2.0 in /opt/conda/lib/python3.9/site-packages (from matplotlib->fastplot) (8.3.
2)
Requirement already satisfied: cycler>=0.10 in /opt/conda/lib/python3.9/site-packages (from matplotlib->fastplot) (0.10.
0)
Requirement already satisfied: patsy>=0.5.2 in /opt/conda/lib/python3.9/site-packages (from statsmodels->fastplot) (0.5.
2)
Building wheels for collected packages: fastplot
  Building wheel for fastplot (setup.py) ... done
  Created wheel for fastplot: filename=fastplot-1.1.0-py3-none-any.whl size=5112 sha256=79ed5c70eec4421a6c6907bdeed2f53c
5b4dd6c0534ce731b758121838ed13ba
  Stored in directory: /home/jovyan/.cache/pip/wheels/22/a8/da/2314af3b8f1ec3e262f17d24c756a5066a1f13d701294907d6
Successfully built fastplot
Installing collected packages: fastplot
Successfully installed fastplot-1.1.0
Collecting sphinx
  Downloading Sphinx-4.2.0-py3-none-any.whl (3.1 MB)
     |████████████████████████████████| 3.1 MB 5.4 MB/s
Collecting sphinx_rtd_theme
  Downloading sphinx_rtd_theme-1.0.0-py2.py3-none-any.whl (2.8 MB)
     |████████████████████████████████| 2.8 MB 50.1 MB/s
Collecting nbsphinx
  Downloading nbsphinx-0.8.7-py3-none-any.whl (25 kB)
Collecting pandoc
  Downloading pandoc-1.1.0-py3-none-any.whl (27 kB)
Collecting pytest-cov
  Downloading pytest_cov-3.0.0-py3-none-any.whl (20 kB)
Collecting uplink==0.9.0
  Downloading uplink-0.9.0-py2.py3-none-any.whl (95 kB)
     |████████████████████████████████| 95 kB 742 kB/s
Collecting uritemplate>=3.0.0
  Downloading uritemplate-4.1.1-py2.py3-none-any.whl (10 kB)
Requirement already satisfied: requests>=2.18.0 in /opt/conda/lib/python3.9/site-packages (from uplink==0.9.0) (2.26.0)
Requirement already satisfied: six>=1.12.0 in /opt/conda/lib/python3.9/site-packages (from uplink==0.9.0) (1.16.0)
Requirement already satisfied: setuptools in /opt/conda/lib/python3.9/site-packages (from sphinx) (58.2.0)
Collecting sphinxcontrib-applehelp
  Downloading sphinxcontrib_applehelp-1.0.2-py2.py3-none-any.whl (121 kB)
     |████████████████████████████████| 121 kB 34.4 MB/s
Requirement already satisfied: packaging in /opt/conda/lib/python3.9/site-packages (from sphinx) (21.0)
Collecting sphinxcontrib-serializinghtml>=1.1.5
  Downloading sphinxcontrib_serializinghtml-1.1.5-py2.py3-none-any.whl (94 kB)
     |████████████████████████████████| 94 kB 434 kB/s
Collecting alabaster<0.8,>=0.7
  Downloading alabaster-0.7.12-py2.py3-none-any.whl (14 kB)
```

```
Collecting sphinxcontrib-htmlhelp>=2.0.0
  Downloading sphinxcontrib_htmlhelp-2.0.0-py2.py3-none-any.whl (100 kB)
     |████████████████████████████████| 100 kB 1.5 MB/s
Collecting snowballstemmer>=1.1
  Downloading snowballstemmer-2.1.0-py2.py3-none-any.whl (93 kB)
     |████████████████████████████████| 93 kB 298 kB/s
Requirement already satisfied: Jinja2>=2.3 in /opt/conda/lib/python3.9/site-packages (from sphinx) (3.0.2)
Collecting imagesize
  Downloading imagesize-1.2.0-py2.py3-none-any.whl (4.8 kB)
Collecting sphinxcontrib-devhelp
  Downloading sphinxcontrib_devhelp-1.0.2-py2.py3-none-any.whl (84 kB)
     |████████████████████████████████| 84 kB 149 kB/s
Requirement already satisfied: Pygments>=2.0 in /opt/conda/lib/python3.9/site-packages (from sphinx) (2.10.0)
Requirement already satisfied: babel>=1.3 in /opt/conda/lib/python3.9/site-packages (from sphinx) (2.9.1)
Collecting docutils<0.18,>=0.14
  Downloading docutils-0.17.1-py2.py3-none-any.whl (575 kB)
     |████████████████████████████████| 575 kB 45.1 MB/s
Collecting sphinxcontrib-jsmath
  Downloading sphinxcontrib_jsmath-1.0.1-py2.py3-none-any.whl (5.1 kB)
Collecting sphinxcontrib-qthelp
  Downloading sphinxcontrib_qthelp-1.0.3-py2.py3-none-any.whl (90 kB)
     |████████████████████████████████| 90 kB 547 kB/s
Requirement already satisfied: nbconvert!=5.4 in /opt/conda/lib/python3.9/site-packages (from nbsphinx) (6.2.0)
Requirement already satisfied: traitlets in /opt/conda/lib/python3.9/site-packages (from nbsphinx) (5.1.0)
Requirement already satisfied: nbformat in /opt/conda/lib/python3.9/site-packages (from nbsphinx) (5.1.3)
Collecting ply
  Downloading ply-3.11-py2.py3-none-any.whl (49 kB)
     |████████████████████████████████| 49 kB 608 kB/s
Collecting plumbum
  Downloading plumbum-1.7.0-py2.py3-none-any.whl (116 kB)
     |████████████████████████████████| 116 kB 52.1 MB/s
Collecting pytest>=4.6
  Downloading pytest-6.2.5-py3-none-any.whl (280 kB)
     |████████████████████████████████| 280 kB 49.3 MB/s
Collecting coverage[toml]>=5.2.1
  Downloading coverage-6.1.1-cp39-cp39-manylinux_2_5_x86_64.manylinux1_x86_64.manylinux_2_12_x86_64.manylinux2010_x86_6
4.whl (214 kB)
     |████████████████████████████████| 214 kB 34.8 MB/s
Requirement already satisfied: pytz>=2015.7 in /opt/conda/lib/python3.9/site-packages (from babel>=1.3->sphinx) (2021.3)
Collecting tomli
  Downloading tomli-1.2.2-py3-none-any.whl (12 kB)
Requirement already satisfied: MarkupSafe>=2.0 in /opt/conda/lib/python3.9/site-packages (from Jinja2>=2.3->sphinx) (2.
0.1)
Requirement already satisfied: jupyterlab-pygments in /opt/conda/lib/python3.9/site-packages (from nbconvert!=5.4->nbsph
inx) (0.1.2)
```

```
Requirement already satisfied: bleach in /opt/conda/lib/python3.9/site-packages (from nbconvert!=5.4->nbsphinx) (4.1.0)
Requirement already satisfied: pandocfilters>=1.4.1 in /opt/conda/lib/python3.9/site-packages (from nbconvert!=5.4->nbsp
hinx) (1.5.0)
Requirement already satisfied: testpath in /opt/conda/lib/python3.9/site-packages (from nbconvert!=5.4->nbsphinx) (0.5.
0)
Requirement already satisfied: entrypoints>=0.2.2 in /opt/conda/lib/python3.9/site-packages (from nbconvert!=5.4->nbsphi
nx) (0.3)
Requirement already satisfied: defusedxml in /opt/conda/lib/python3.9/site-packages (from nbconvert!=5.4->nbsphinx) (0.
7.1)
Requirement already satisfied: nbclient<0.6.0,>=0.5.0 in /opt/conda/lib/python3.9/site-packages (from nbconvert!=5.4->nb
sphinx) (0.5.4)
Requirement already satisfied: mistune<2,>=0.8.1 in /opt/conda/lib/python3.9/site-packages (from nbconvert!=5.4->nbsphin
x) (0.8.4)
Requirement already satisfied: jupyter-core in /opt/conda/lib/python3.9/site-packages (from nbconvert!=5.4->nbsphinx)
(4.8.1)
Requirement already satisfied: jsonschema!=2.5.0,>=2.4 in /opt/conda/lib/python3.9/site-packages (from nbformat->nbsphin
x) (4.1.0)
Requirement already satisfied: ipython-genutils in /opt/conda/lib/python3.9/site-packages (from nbformat->nbsphinx) (0.
2.0)
Requirement already satisfied: attrs>=19.2.0 in /opt/conda/lib/python3.9/site-packages (from pytest>=4.6->pytest-cov) (2
1.2.0)
Collecting toml
  Downloading toml-0.10.2-py2.py3-none-any.whl (16 kB)
Collecting py>=1.8.2
  Downloading py-1.11.0-py2.py3-none-any.whl (98 kB)
     |████████████████████████████████| 98 kB 178 kB/s
Collecting pluggy<2.0,>=0.12
  Downloading pluggy-1.0.0-py2.py3-none-any.whl (13 kB)
Collecting iniconfig
  Downloading iniconfig-1.1.1-py2.py3-none-any.whl (5.0 kB)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in /opt/conda/lib/python3.9/site-packages (from requests>=2.18.0->u
plink==0.9.0) (1.26.7)
Requirement already satisfied: charset-normalizer~=2.0.0 in /opt/conda/lib/python3.9/site-packages (from requests>=2.18.
0->uplink==0.9.0) (2.0.0)
Requirement already satisfied: idna<4,>=2.5 in /opt/conda/lib/python3.9/site-packages (from requests>=2.18.0->uplink==0.
9.0) (3.1)
Requirement already satisfied: certifi>=2017.4.17 in /opt/conda/lib/python3.9/site-packages (from requests>=2.18.0->upli
nk==0.9.0) (2021.10.8)
Requirement already satisfied: pyparsing>=2.0.2 in /opt/conda/lib/python3.9/site-packages (from packaging->sphinx) (2.4.
7)
Requirement already satisfied: pyrsistent!=0.17.0,!=0.17.1,!=0.17.2,>=0.14.0 in /opt/conda/lib/python3.9/site-packages
(from jsonschema!=2.5.0,>=2.4->nbformat->nbsphinx) (0.17.3)
Requirement already satisfied: jupyter-client>=6.1.5 in /opt/conda/lib/python3.9/site-packages (from nbclient<0.6.0,>=0.
5.0->nbconvert!=5.4->nbsphinx) (7.0.6)
Requirement already satisfied: nest-asyncio in /opt/conda/lib/python3.9/site-packages (from nbclient<0.6.0,>=0.5.0->nbco
```

```
nvert!=5.4->nbsphinx) (1.5.1)
Requirement already satisfied: webencodings in /opt/conda/lib/python3.9/site-packages (from bleach->nbconvert!=5.4->nbsp
hinx) (0.5.1)
Requirement already satisfied: pyzmq>=13 in /opt/conda/lib/python3.9/site-packages (from jupyter-client>=6.1.5->nbclient
<0.6.0,>=0.5.0->nbconvert!=5.4->nbsphinx) (22.3.0)
Requirement already satisfied: python-dateutil>=2.1 in /opt/conda/lib/python3.9/site-packages (from jupyter-client>=6.1.
5->nbclient<0.6.0,>=0.5.0->nbconvert!=5.4->nbsphinx) (2.8.2)
Requirement already satisfied: tornado>=4.1 in /opt/conda/lib/python3.9/site-packages (from jupyter-client>=6.1.5->nbcli
ent<0.6.0,>=0.5.0->nbconvert!=5.4->nbsphinx) (6.1)
Installing collected packages: tomli, toml, sphinxcontrib-serializinghtml, sphinxcontrib-qthelp, sphinxcontrib-jsmath, s
phinxcontrib-htmlhelp, sphinxcontrib-devhelp, sphinxcontrib-applehelp, snowballstemmer, py, pluggy, iniconfig, imagesiz
e, docutils, coverage, alabaster, uritemplate, sphinx, pytest, ply, plumbum, uplink, sphinx-rtd-theme, pytest-cov, pando
c, nbsphinx
  WARNING: The scripts coverage, coverage-3.9 and coverage3 are installed in '/home/jovyan/.local/bin' which is not on P
ATH.
  Consider adding this directory to PATH or, if you prefer to suppress this warning, use --no-warn-script-location.
  WARNING: The scripts sphinx-apidoc, sphinx-autogen, sphinx-build and sphinx-quickstart are installed in '/home/jovya
n/.local/bin' which is not on PATH.
  Consider adding this directory to PATH or, if you prefer to suppress this warning, use --no-warn-script-location.
  WARNING: The scripts py.test and pytest are installed in '/home/jovyan/.local/bin' which is not on PATH.
  Consider adding this directory to PATH or, if you prefer to suppress this warning, use --no-warn-script-location.
Successfully installed alabaster-0.7.12 coverage-6.1.1 docutils-0.17.1 imagesize-1.2.0 iniconfig-1.1.1 nbsphinx-0.8.7 pa
ndoc-1.1.0 pluggy-1.0.0 plumbum-1.7.0 ply-3.11 py-1.11.0 pytest-6.2.5 pytest-cov-3.0.0 snowballstemmer-2.1.0 sphinx-4.2.
0 sphinx-rtd-theme-1.0.0 sphinxcontrib-applehelp-1.0.2 sphinxcontrib-devhelp-1.0.2 sphinxcontrib-htmlhelp-2.0.0 sphinxco
ntrib-jsmath-1.0.1 sphinxcontrib-qthelp-1.0.3 sphinxcontrib-serializinghtml-1.1.5 toml-0.10.2 tomli-1.2.2 uplink-0.9.0 u
ritemplate-4.1.1
```

The below block makes your Kernel restart so that you have all dependencies available

In [2]:

```python
import IPython
IPython.Application.instance().kernel.do_shutdown(True)
```

Out[2]:

```
{'status': 'ok', 'restart': True}
```

The dataset includes a line for each passenger, and the columns describe them under various aspects:

- **survival**: Survival 0 = No, 1 = Yes
- **pclass**: Ticket class 1 = 1st, 2 = 2nd, 3 = 3rd
- **sex**: Sex
- **Age**: Age in years
- **sibsp**: # of siblings / spouses aboard the Titanic

- **parch**: # of parents / children aboard the Titanic
- **ticket**: Ticket number
- **fare**: Passenger fare
- **cabin**: Cabin number
- **embarked**: Port of Embarkation C = Cherbourg, Q = Queenstown, S = Southampton
- **deck**: Deck number (A, B, C, D, E, F, G, T, M). M means missing

Load it in a Pandas **DataFrame** and inspect it using the `.head()` method.

In [1]:
```
import pandas as pd

titanic = pd.read_csv("titanic.csv")
titanic.head()
```

Out[1]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked	Deck
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S	M
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S	M
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S	C
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S	M

Characterize the Dataset

As a data curator, you must know the characteristics of your dataset. Thus, let's compute some statistics on the data and make some plots.

Compute the number of people, how many Survived and Dead passengers, how many Male and Female.

In [5]:
```
print ("Number of rows:", len(titanic) )
print ("Number of Survived:", len(titanic[titanic["Survived"] == 1]), "Dead:", len(titanic[titanic["Survived"] == 0]))

#SOLUTION
print ("Number of Male:", len(titanic[titanic["Sex"] == "male"]), "Female:", len(titanic[titanic["Sex"] == "female"]))
```

```
Number of rows: 891
Number of Survived: 342 Dead: 549
Number of Male: 577 Female: 314
```

Which is the average Fare of the tickets? And the average age?

In [6]:

```
# SOLUTION
print("Average Fare:", titanic["Fare"].mean() )
print("Median Age: ", titanic["Age"].median() )
```

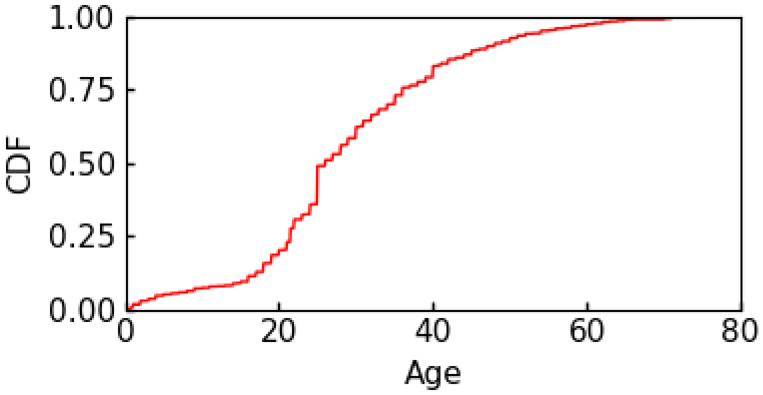
```
Average Fare: 32.2042079685746
Median Age:  26.0
```

Now, you can plot the empirical distribution of the age and fare values

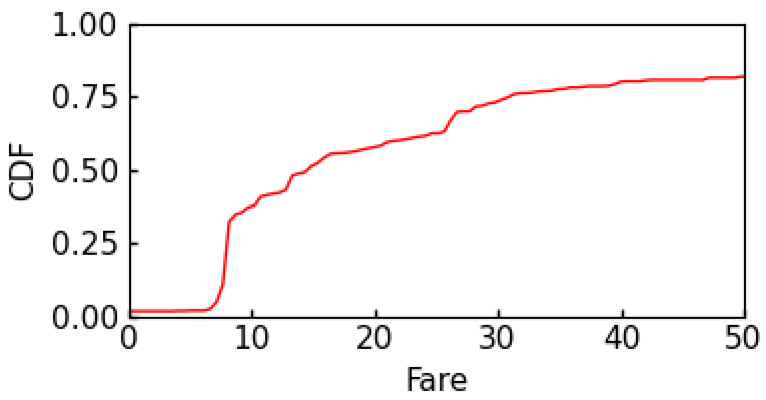
In [30]:

```
import fastplot
%matplotlib inline

fastplot.plot(titanic["Age"].values, None, mode="CDF", xlabel="Age").show()

# SOLUTION
fastplot.plot(titanic["Fare"].values, None, mode="CDF", xlabel="Fare", xlim=(0,50)).show()
```

```
<Figure size 640x480 with 0 Axes>
```




```
<Figure size 640x480 with 0 Axes>
```



Publish the Dataset using the k-Anonymity and l-Diversity

As the Data Curator of the dataset, you want to publish it while at the same time preserving the privacy of users.

You can use the k-anonymity or the l-diversity properties or the to anonymize it. To do so, use the ARX tool.

As ARX is written in Java, to use it in Python, you can use the implemantation available in the Arx As a Service library. The library offers Python the ARX functions and uses a remote ARX server to make the computation.

To start an ARX server, the easiest way is to run:

```
docker run -p 8080:8080 navikt/arxaas
```

on your favorite server.

We already did it for you on `jitsi.polito.it` . So, you can import `pyarxaas` and create your `ARXaaS` object that communicates with the server.

In [8]:
```
from pyarxaas import ARXaaS
arxaas = ARXaaS("http://jitsi.polito.it:8080")
```

Now, convert the `titanic` Pandas dataframe in a pyarxaas `Dataset` .

In this exercize, you want to release a dataset reporting the **Fare** and **Age** of Survived and not Survived passengers. As such, create a dataset with these four columns.

In [9]:

```
from pyarxaas import Dataset

dataset = Dataset.from_pandas(titanic[["Name", "Fare","Age", "Survived"]])
```

You must tell `pyarxaas` which columns are `IDENTIFYING`, `QUASIIDENTIFYING`, `SENSITIVE` and `INSENSITIVE`.

Fare and Age are Quasi Identifiers as can be used to re-identify a person given some domain knowledge. The Name is clearly an identifier. Survived a sensitive attribute, but, since the k-anonymity does not consider sensitive attribute, you shall indicate it as `INSENSITIVE` when passing it to the k-anonymity function of **pyarxaas**

In [10]:

```
from pyarxaas import AttributeType
dataset.set_attribute_type(AttributeType.IDENTIFYING, 'Name')
dataset.set_attribute_type(AttributeType.QUASIIDENTIFYING, 'Fare', 'Age')
dataset.set_attribute_type(AttributeType.INSENSITIVE, 'Survived')
```

It is fundamental that you create **hierarchy** so that pyarxaas knows how to generalize attributes.

It is hard to find the correct hierarchy:

- A too fine hierarchy will make the algorithm to delete all the information to anonymize the data.
- A too coarse hierarchy will give poor information to the user of the released data.

In [11]:

```
from pyarxaas.hierarchy import IntervalHierarchyBuilder

# Interval hierarchy for the age
interval_based_age = IntervalHierarchyBuilder()
interval_based_age.add_interval(0.0, 20.0, "0-20")
interval_based_age.add_interval(20.0, 40.0, "20-40")
interval_based_age.add_interval(40.0, 60.0, "40-60")
interval_based_age.add_interval(60.0, 100.0, "60-100")

interval_hierarchy_age = arxaas.hierarchy(interval_based_age, list(titanic['Age'].values) )
dataset.set_hierarchy('Age', interval_hierarchy_age)

# Interval hierarchy for the Fare
# SOLUTION
```

```python
interval_based_fare = IntervalHierarchyBuilder()
interval_based_fare.add_interval(0.0, 30.0, "<=30")
interval_based_fare.add_interval(30.0, 60.0, "30-60")
interval_based_fare.add_interval(60.0, 10000.0, ">60")

interval_hierarchy_fare = arxaas.hierarchy(interval_based_fare, list(titanic['Fare'].values) )
dataset.set_hierarchy('Fare', interval_hierarchy_fare)
```

Anonymize the dataset with the k-anonymity, with $k = 2$.

How the released data look like? How much information is there in your opinion?

Try changing the **hierarchies** and see how the output varies.

In [12]:

```python
from pyarxaas.privacy_models import KAnonymity, LDiversityDistinct

kanon = KAnonymity(2)
anonymize_result = arxaas.anonymize(dataset, [kanon])
titanic_kanon = anonymize_result.dataset.to_dataframe().sort_values(["Fare","Age", "Survived"])
titanic_kanon.head()
```

Out[12]:

	Name	Fare	Age	Survived
50	*	30-60	0-20	0
59	*	30-60	0-20	0
71	*	30-60	0-20	0
86	*	30-60	0-20	0
119	*	30-60	0-20	0

Now anonymize the dataset using the l-diversity.

Note: according to the nature of the property, you must set at least an attribute as `SENSITIVE`, `"Survived"` in this case.

In [13]:

```python
from pyarxaas.privacy_models import KAnonymity, LDiversityDistinct

ldiv = LDiversityDistinct(l=2,  column_name="Survived")
dataset.set_attribute_type(AttributeType.SENSITIVE, 'Survived')
anonymize_result = arxaas.anonymize(dataset, [ldiv])
```

```
#SOLUTION
titanic_ldiv = anonymize_result.dataset.to_dataframe().sort_values(["Fare","Age", "Survived"])
titanic_ldiv
```

Out[13]:

	Name	Fare	Age	Survived
50	*	30-60	0-20	0
59	*	30-60	0-20	0
71	*	30-60	0-20	0
86	*	30-60	0-20	0
119	*	30-60	0-20	0
...	...	...	...	...
745	*	>60	60-100	0
275	*	>60	60-100	1
366	*	>60	60-100	1
587	*	>60	60-100	1
829	*	>60	60-100	1

891 rows × 4 columns

Is there much different compared to the output with k-anonymity? Are these data **useful**?

To understand this, compute the average age of the Survived and Dead passenger on the original dataset and on those anonymized with the k-anonymity or l-diversity.

Do they differ? Do they they use hierarchies differently or similarly?

Hint: as you binned the ages, consider the average age for each bin. E.g., `0-20` becomes 10.

In [14]:

```
#SOLUTION
print ("Original Dataset")
titanic.groupby("Survived")["Age"].mean().reset_index()
```

Original Dataset

Out[14]:

	Survived	Age
0	0	29.737705
1	1	28.108684

In [15]:

```
#SOLUTION
print ("k-anonymized")
titanic_kanon["Age_rebuilt"] = titanic_kanon["Age"].apply(lambda s: {"0-20":10, "20-40":30, "40-60": 50, "60-100":80}[s
titanic_kanon.groupby("Survived")["Age_rebuilt"].mean().reset_index()
```

k-anonymized

Out[15]:

	Survived	Age_rebuilt
0	0	32.240437
1	1	29.853801

Now, you want to release a dataset that indicates whether a passenger survived, indicating the embarkement port and class.

Use the k-anonymity, and set $k = 2$.

What does ARX do? What happens if you increase k? Do you need a hierarchy?

In [31]:

```
dataset = Dataset.from_pandas(titanic[["Pclass","Embarked", "Survived"]])

#SOLUTION
dataset.set_attribute_type(AttributeType.QUASIIDENTIFYING, 'Pclass', 'Embarked')
dataset.set_attribute_type(AttributeType.INSENSITIVE, 'Survived')

kanon = KAnonymity(2)
anonymize_result = arxaas.anonymize(dataset, [kanon])
titanic_kanon = anonymize_result.dataset.to_dataframe().sort_values(["Pclass","Embarked", "Survived"])
titanic_kanon
```

Out[31]:

	Pclass	Embarked	Survived
30	1	C	0
34	1	C	0

	Pclass	Embarked	Survived
54	1	C	0
64	1	C	0
96	1	C	0
...	...	...	...
821	3	S	1
823	3	S	1
838	3	S	1
855	3	S	1
869	3	S	1

891 rows × 3 columns

Differential Privacy

Now you will use the `diffprivlib` IBM library for differential privacy. You will compute some differentially private aggregates from the Titanic dataset.

First, compute the average age of passengers, both with and without differential privacy. Set $\epsilon = 0.02$

In [16]:

```
import diffprivlib

print ("Real Mean:", titanic["Age"].mean())
print ("Diff Priv Mean:", diffprivlib.tools.mean(titanic["Age"].values, epsilon=0.02))
```

```
Real Mean: 29.11242424242424
Diff Priv Mean: 22.48590915115323
```

```
/home/jovyan/.local/lib/python3.9/site-packages/diffprivlib/tools/utils.py:259: PrivacyLeakWarning: Bounds have not been
specified and will be calculated on the data provided. This will result in additional privacy leakage. To ensure differe
ntial privacy and no additional privacy leakage, specify bounds for each dimension.
  warnings.warn("Bounds have not been specified and will be calculated on the data provided. This will "
```

With the differential privacy, increasing ϵ you will get more accurate (and less-privacy friendly) results.

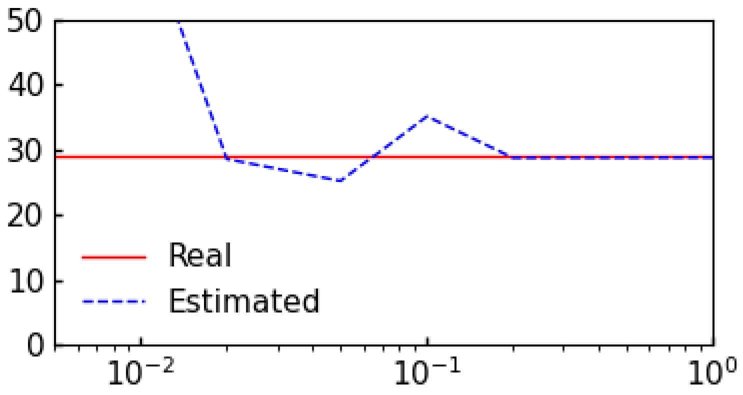
Try to visualize it, running the mean query increasing ϵ. You will notice how it converges towards the real value, nullifying the benefits of effect privacy.

Note: if you plot it, use a logaritmic scale on the x axis.

In [18]:

```
epsilons = [0.005,0.01,0.02,0.05,0.1,0.2,0.5,1]
avgs = []
for eps in epsilons:
    # SOLUTION
    avgs.append(diffprivlib.tools.mean(titanic["Age"].values, epsilon=eps, bounds=(0,100)))

fastplot.plot([ ("Real", ([0,epsilons[-1]],[titanic["Age"].mean(),titanic["Age"].mean()]) ),
                ("Estimated",(epsilons, avgs)) ],
              None, mode="line_multi", xlim=(epsilons[0],epsilons[-1]), xscale="log",
              ylim = (0,50),legend=True).show()
avgs
```

```
<Figure size 640x480 with 0 Axes>
```



Out[18]:

```
[51.13129084950907,
 66.2408903513614,
 28.601569483690945,
 25.226329335051304,
 35.167952415125214,
 28.78388216606397,
 28.802455824987913,
 28.850580443491403]
```

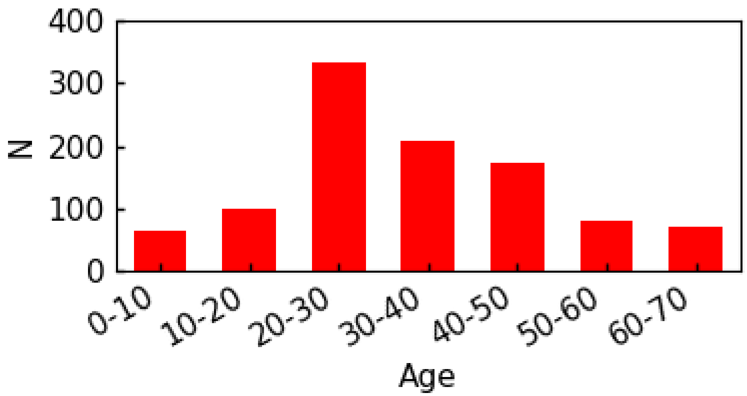
Now, you want to release a differentially private [histogram](https://en.wikipedia.org/wiki/Histogram) of the Age of the Titanic passengers. For each age group spanning 10 years (e.g., `0-10`, `10-20`, `20-30`, etc.)

Try different ϵ, from 0.02 to 2. Plot the histograms and see how they vary.

Imagine you are a Data Analystist. For which ϵ do you get results that radically differ from the original histogram (that you can obtain with numpy)?

In [20]:
```
#h,b = diffprivlib.tools.histogram(titanic["Age"], ...)

# SOLUTION
h,b = diffprivlib.tools.histogram(titanic["Age"], epsilon=0.05,bins=[0,10,20,30,40,50,60,70], bounds=(0,100))
to_plot = list(zip([ f"{b[i]}-{b[i+1]}" for i,v in enumerate(h) ],h ))
fastplot.plot(to_plot , None, mode="bars", ylabel="N", xticks_rotate=30, xlabel="Age" ).show()
```

```
/home/jovyan/.local/lib/python3.9/site-packages/diffprivlib/utils.py:85: DiffprivlibCompatibilityWarning: Parameter 'bou
nds' is not functional in diffprivlib.  Remove this parameter to suppress this warning.
  warnings.warn(f"Parameter '{arg}' is not functional in diffprivlib.  Remove this parameter to suppress this "
/home/jovyan/.local/lib/python3.9/site-packages/diffprivlib/tools/histograms.py:130: PrivacyLeakWarning: Range parameter
has not been specified. Falling back to taking range from the data.
To ensure differential privacy, and no additional privacy leakage, the range must be specified independently of the data
(i.e., using domain knowledge).
  warnings.warn("Range parameter has not been specified. Falling back to taking range from the data.\n"
<Figure size 640x480 with 0 Axes>
```



Now, similarly, compute the 2-dimensional histogram of **Age** and **Survived**. In this way, the published data allow computing the surviving ratio of people with different age.

Note: The Survived column can assume only 0 and 1, meaning Dead or Survived.

Compute the surviving ratio for people in separate age groups and compare it with the one you obtain in the original data. Try different ϵ and see the impact.

In [22]:

```
#h,x,y = diffprivlib.tools.histogram2d(titanic["Age"].values, titanic["Survived"].values, ...)
#h,x,y =                 np.histogram2d(titanic["Age"].values, titanic["Survived"].values, ...)

# SOLUTION
import numpy as np
h,x,y = diffprivlib.tools.histogram2d(titanic["Age"].values,titanic["Survived"].values,
                                    epsilon=0.5,
                                    bins=[ [0,10,20,30,40,50,60,70], 2]  )

print("Differentially Private")
for i,t in enumerate(h):
    print(x[i], "-", x[i+1] , int(t[1]/sum(t)*100), "%")

h,x,y = np.histogram2d(titanic["Age"].values,titanic["Survived"].values,
                                    bins=[ [0,10,20,30,40,50,60,70], 2]  )

print("Differentially Private")
for i,t in enumerate(h):
    print(x[i], "-", x[i+1] , int(t[1]/sum(t)*100), "%")
```

```
Differentially Private
0 - 10 57 %
10 - 20 38 %
20 - 30 31 %
30 - 40 46 %
40 - 50 32 %
50 - 60 39 %
60 - 70 26 %
Differentially Private
0 - 10 61 %
10 - 20 40 %
20 - 30 31 %
30 - 40 45 %
40 - 50 35 %
50 - 60 41 %
60 - 70 28 %
/home/jovyan/.local/lib/python3.9/site-packages/diffprivlib/tools/histograms.py:227: PrivacyLeakWarning: Range parameter has not been specified (or has missing elements). Falling back to taking range from the data.
 To ensure differential privacy, and no additional privacy leakage, the range must be specified for each dimension indep
```

```
endently of the data (i.e., using domain knowledge).
  warnings.warn("Range parameter has not been specified (or has missing elements). Falling back to taking "
```

Try to implement yourself a differentially private sum. Use the Laplace mechanism. You must compute the Δf (also called sensitivity).

Which is the maximum variation in the sum if you use a dataset including all people but one?

Compute the sum of the Fares.

In [23]:

```
deltaF = max(titanic["Fare"])
mech = diffprivlib.mechanisms.Laplace(epsilon=1, sensitivity=deltaF)
#SOLUTION
mech = diffprivlib.mechanisms.Laplace(epsilon=1, sensitivity=deltaF)
print ("My Differentially Private Sum of Fares is:", mech.randomise(titanic["Fare"].sum()) )
print ("The DiffPrivLib Sum of Fares is:", diffprivlib.tools.sum(titanic["Fare"].values, epsilon=1))
print ("The real Sum is:", mech.randomise(titanic["Fare"].sum()) )
```

```
My Differentially Private Sum of Fares is: 27854.98059175127
The DiffPrivLib Sum of Fares is: 29526.89605384991
The real Sum is: 30693.80283447178
```

```
/home/jovyan/.local/lib/python3.9/site-packages/diffprivlib/tools/utils.py:666: PrivacyLeakWarning: Bounds have not been
specified and will be calculated on the data provided. This will result in additional privacy leakage. To ensure differe
ntial privacy and no additional privacy leakage, specify bounds for each dimension.
  warnings.warn("Bounds have not been specified and will be calculated on the data provided. This will "
```

Differentially Private Machine Learning

Now, let's have a quick tour on the Differentially Private Machine Learning models available in DiffPrivLib. They are very similar to those implemented in Scikit Learn, but they are differentially private.

We will try a simple classification problem, solving it with both traditional and differentially private classification models, and compare the outcomes.

First, we must create a training and a test set:

In [25]:

```
from sklearn.preprocessing import OneHotEncoder
from sklearn.model_selection import train_test_split
from sklearn.compose import ColumnTransformer

X = titanic[["Sex", "Age", "SibSp", "Parch", "Fare","Embarked", "Deck"]].values
X = ColumnTransformer([("OneHot", OneHotEncoder(),[0,5,6])], remainder="passthrough").fit_transform(X)
```

```python
y = titanic["Survived"].values

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33, random_state=42)
```

Now train a standard scikit learn Random Forest Classifier, and evaluate the performance.

A `RandomForestClassifier` has the `fit()` and `predict()` methods, while you can evaluate performance with the classifaction report.

In [26]:

```python
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report

clf = RandomForestClassifier(n_estimators=10)
#SOLUTION
clf.fit(X_train,y_train)
y_test_pred = clf.predict(X_test)

original_f1 = classification_report(y_test, y_test_pred, output_dict=True)['macro avg']['f1-score']
print(classification_report(y_test, y_test_pred))
```

```
              precision    recall  f1-score   support

           0       0.81      0.83      0.82       175
           1       0.74      0.71      0.72       120

    accuracy                           0.78       295
   macro avg       0.77      0.77      0.77       295
weighted avg       0.78      0.78      0.78       295
```

Now use the differentially private RandomForestClassifier present in DiffPrivLib and evaluate its performance.

In [27]:

```python
clf = diffprivlib.models.RandomForestClassifier(n_estimators=10, n_jobs=-1, epsilon=1)

#SOLUTION
clf.fit(X_train,y_train)
y_test_pred = clf.predict(X_test)

print(classification_report(y_test, y_test_pred))
```

```
/home/jovyan/.local/lib/python3.9/site-packages/diffprivlib/models/forest.py:189: PrivacyLeakWarning: `feature_domains`
parameter hasn't been specified, so falling back to determining domains from the data.
```

```
This may result in additional privacy leakage. To ensure differential privacy with no additional privacy loss, specify `
feature_domains` according to the documentation
  warnings.warn(
              precision    recall  f1-score   support

           0       0.70      0.91      0.79       175
           1       0.78      0.43      0.56       120

    accuracy                           0.72       295
   macro avg       0.74      0.67      0.68       295
weighted avg       0.73      0.72      0.70       295
```

Finally, plot how the F1-Score varies with different ϵ.

Notice the trade-off between ϵ and performance.

In [28]:

```
epsilons = [0.01,0.02,0.05,0.1,0.2,0.5,1,2,5,10]
#SOLUTION
f1s = []
for epsilon in epsilons:
    clf = diffprivlib.models.RandomForestClassifier(n_estimators=10, n_jobs=-1, epsilon=epsilon)
    clf.fit(X_train,y_train)
    y_test_pred = clf.predict(X_test)
    f1s.append(classification_report(y_test, y_test_pred, output_dict=True)['macro avg']['f1-score'])
```

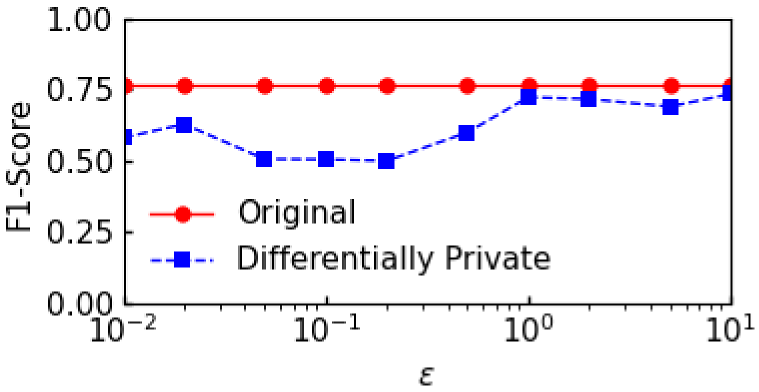
```
/home/jovyan/.local/lib/python3.9/site-packages/diffprivlib/models/forest.py:189: PrivacyLeakWarning: `feature_domains`
parameter hasn't been specified, so falling back to determining domains from the data.
This may result in additional privacy leakage. To ensure differential privacy with no additional privacy loss, specify `
feature_domains` according to the documentation
  warnings.warn(
/home/jovyan/.local/lib/python3.9/site-packages/diffprivlib/models/forest.py:189: PrivacyLeakWarning: `feature_domains`
parameter hasn't been specified, so falling back to determining domains from the data.
This may result in additional privacy leakage. To ensure differential privacy with no additional privacy loss, specify `
feature_domains` according to the documentation
  warnings.warn(
/home/jovyan/.local/lib/python3.9/site-packages/diffprivlib/models/forest.py:189: PrivacyLeakWarning: `feature_domains`
parameter hasn't been specified, so falling back to determining domains from the data.
This may result in additional privacy leakage. To ensure differential privacy with no additional privacy loss, specify `
feature_domains` according to the documentation
  warnings.warn(
/home/jovyan/.local/lib/python3.9/site-packages/diffprivlib/models/forest.py:189: PrivacyLeakWarning: `feature_domains`
parameter hasn't been specified, so falling back to determining domains from the data.
```

```
This may result in additional privacy leakage. To ensure differential privacy with no additional privacy loss, specify `
feature_domains` according to the documentation
  warnings.warn(
/home/jovyan/.local/lib/python3.9/site-packages/diffprivlib/models/forest.py:189: PrivacyLeakWarning: `feature_domains`
parameter hasn't been specified, so falling back to determining domains from the data.
This may result in additional privacy leakage. To ensure differential privacy with no additional privacy loss, specify `
feature_domains` according to the documentation
  warnings.warn(
/home/jovyan/.local/lib/python3.9/site-packages/diffprivlib/models/forest.py:189: PrivacyLeakWarning: `feature_domains`
parameter hasn't been specified, so falling back to determining domains from the data.
This may result in additional privacy leakage. To ensure differential privacy with no additional privacy loss, specify `
feature_domains` according to the documentation
  warnings.warn(
/home/jovyan/.local/lib/python3.9/site-packages/diffprivlib/models/forest.py:189: PrivacyLeakWarning: `feature_domains`
parameter hasn't been specified, so falling back to determining domains from the data.
This may result in additional privacy leakage. To ensure differential privacy with no additional privacy loss, specify `
feature_domains` according to the documentation
  warnings.warn(
/home/jovyan/.local/lib/python3.9/site-packages/diffprivlib/models/forest.py:189: PrivacyLeakWarning: `feature_domains`
parameter hasn't been specified, so falling back to determining domains from the data.
This may result in additional privacy leakage. To ensure differential privacy with no additional privacy loss, specify `
feature_domains` according to the documentation
  warnings.warn(
/home/jovyan/.local/lib/python3.9/site-packages/diffprivlib/models/forest.py:189: PrivacyLeakWarning: `feature_domains`
parameter hasn't been specified, so falling back to determining domains from the data.
This may result in additional privacy leakage. To ensure differential privacy with no additional privacy loss, specify `
feature_domains` according to the documentation
  warnings.warn(
/home/jovyan/.local/lib/python3.9/site-packages/diffprivlib/models/forest.py:189: PrivacyLeakWarning: `feature_domains`
parameter hasn't been specified, so falling back to determining domains from the data.
This may result in additional privacy leakage. To ensure differential privacy with no additional privacy loss, specify `
feature_domains` according to the documentation
  warnings.warn(
```

In [29]:

```
to_plot = [("Original", (epsilons, [original_f1]* len (epsilons) ) ),
            ("Differentially Private", (epsilons, f1s))
           ]

fastplot.plot(to_plot, None, mode="line_multi", xscale = "log", ylim = (0,1),
              xlabel="$\\epsilon$", ylabel = "F1-Score",
              cycler=fastplot.CYCLER_LINESPOINTS, legend=True).show()
```

```
<Figure size 640x480 with 0 Axes>
```


F1-Score
1.00
0.75
0.50
0.25
0.00
10^{-2}
10^{-1}
10^{0}
10^{1}
ε
Original
Differentially Private