# An introduction to data privacy and anonymization:

What is data anonymization

Why do we need it

Techniques

Types of data anonymization

Data privacy is a fundamental right of every individual in this day and age. With people’s online presence increasing, data privacy is important now more than ever as handling large amounts of data and protecting the rights of an individual is a complex task

Data privacy is the combined responsibility of both the users and the other party

Data Anonymization is a form of privatising the data before making it public or sending it to the intended audience.

## Why is data anonymization needed?

One of the biggest examples which states the importance of anonymizing the data is the 2006 AOL data search leak. Here are some interesting statistics about the data leak:

* On 4th of August 2006, search data of approximately 650,000 users along with 20 Million search results were leaked
* The data was removed relatively quickly on 7th of August 2006
* The AOL did not identify the users in the data as the names of the users were not explicitly mentioned in the data
* However, a popular newspaper magazine called New York times were able to identify the users by cross referencing them with other sources like phone book listing

Another popular data breach is that of the Netflix where the data was leaked and the researchers at the university were able to trace it back to the users

A popular data breach at the time of writing this report was that of the UNC chapel hill where the tax from info was sent to wrong people. Tax forms had a lot of sensitive info which created a lot of chaos and the privacy of the users was being sacrificed

Although proper care should be taken that the data does not get leaked in the first place, it is equally necessary to make sure that if the data gets leaked, the sensitive information is not revealed to the users. Data Anonymization ensures privacy of the users and makes sure that the necessary parameters of the data are taken care of

Data anonymization refers to masking sensitive user data in a way that the identity of the user is maintained and if the data is released, the data can not be traced back to the user. Data anonymization is the tradeoff between usability of the data in terms of statistical parameters and privacy of the users and to maintain the same statistical models and domain knowledge are combined to make sure that the anonymization is done properly.

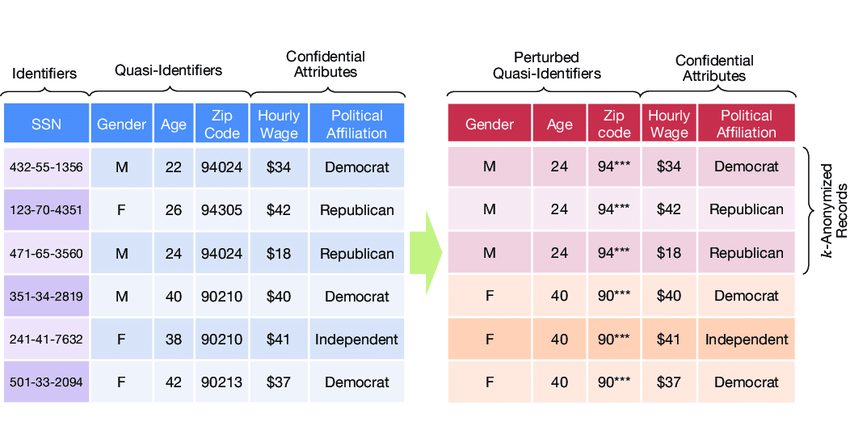
### To ensure the privacy of the data, a five parameter framework is used:

1. Ensure that the data is safe
2. Ensure that the people working on the data are safe
3. Ensure that the scope of the project is viable
4. Ensure that the proper compliant standards are set up to ensure safety in place
5. Disclose of the output data can be monitored to ensure that sensitive data is not leaked

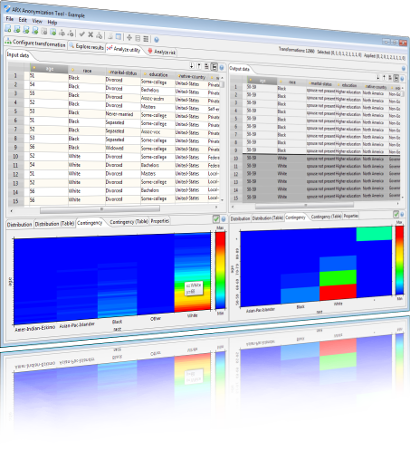
Simply removing the identifying attributes from the data is not enough nor is masking them as they can often be traced back to the original source.

Types of anonymization

There are two types of anonymization: Static and dynamic. Traditionally, static anonymization has been considered a less safer and better option as compared to its querying counterpart but for the use of the project, the dataset comes from the same database and is typically really similar hence using a static anonymization tool such as ARX makes a lot more sense



# What Is ARX:



## History of arx:

Arx is a cloud-based service that provides data anonymization and privacy-enhancing technologies to organisations. The history of ARX can be traced back to the early development of the arx algorithm for cryptography. Arx was originally designed to be a more efficient and secure way of implementing certain cryptographic operations, such as encryption and authentication. However, it was soon recognized that arx could also be used for data anonymization and de-identification, by replacing sensitive data with random or pseudonymous values while preserving the statistical properties of the original data. It uses the arx cryptographic primitive to protect data by replacing sensitive information with random or pseudonymous values while preserving the statistical properties of the original data.

Arx is a method of combining the basic bitwise operations of addition, rotation, and XOR to implement encryption and authentication. It was designed to be a more efficient and secure way of implementing certain cryptographic operations.

Arx module offers a range of anonymization techniques, including k-anonymity, L diversity,T-closeness and differential privacy, which help organisations comply with data protection regulations while preserving data utility. The service is accessed through an API that allows developers to easily integrate the anonymization capabilities of Arxaas into their existing applications. The service is also available as a software but it has limited use cases and cannot be customised. There is also a python module called ‘pyarxaas’ which helps users access the arx service right from their python IDE

Arxaas uses a scalable infrastructure and a cloud-based architecture, which makes it ideal for processing large volumes of data in a distributed manner. The service also provides security and compliance features such as role-based access control and audit logging, to ensure that sensitive data is handled in a secure and compliant manner. However, it’s python module pyarxaas does not support a lot of the additional functionality that the app provides

Overall, Arxaas provides a comprehensive solution for organisations looking to protect the privacy of their data, while also ensuring that they remain compliant with data protection regulations. It is used by different types of organisations, like hospitals, banks, and online stores, to protect their customers' privacy and comply with data protection laws. It is also used widely to make sure that the privacy of their customers is maintained

## Why do we need arx:

ARX might seem like a ‘nice-to-have’ for a lot of normal businesses dealing with the data however, with the concerns of data privacy and ethics rising the applications of the data can be far and wide. Here are some use cases which define the role of arx in an organisation:

1. Compliance: Many organisations are subject to data protection regulations, such as the General Data Protection Regulation (GDPR) and the Health Insurance Portability and Accountability Act (HIPAA), that require them to protect the privacy of their customers' data. Arxaas provides privacy-enhancing technologies that help organisations comply with these regulations.
2. Data sharing: Organisations may need to share data with third parties, such as researchers or business partners, while ensuring that the data is anonymized and does not contain any personally identifiable information. Arxaas can help organisations anonymize their data so that it can be shared safely and securely.
3. Risk management: Organisations may want to reduce the risk of data breaches or other security incidents by de-identifying sensitive data. Arxaas can help organisations minimise the risk of data exposure by anonymizing sensitive data before it is processed or stored.
4. Data analysis: Organisations may want to use data for analysis or research purposes while protecting the privacy of individuals whose data is being used. Arxaas can help organisations preserve the utility of the data while ensuring that the privacy of individuals is protected.

One example of an organisation using Arxaas is the UK's National Health Service (NHS), which has used the service to anonymize patient data for research purposes. Another example is the German National Library of Science and Technology (TIB), which has used Arxaas to anonymize data on digital preservation and research data management. Additionally, several academic institutions have used Arxaas for research purposes, including the University of Copenhagen and the University of Vienna.

In summary, the Arxaas service can help organisations comply with data protection regulations, share data safely and securely, reduce the risk of data breaches, and preserve data utility while protecting individual privacy.

## Flavours of arx

To ensure that organisations and users across all domains can take advantage of their services, arx is available as a variety of services:

**ARXAAS**: is a cloud based service which can handle large amounts of data. Furthermore, the arxaas instance can be run locally using docker.

**Pyarxaas:** Pyarxaas is a python module for implementing functions for the arx, it supports a limited range of privacy models and does not have the additional functionality that the arx software does but it can be used to connect to the local arx instance and perform functions on the data

**API**: All the functionality of the arx is also available as an API which can be used to integrate to the surface

## Installation guide for arx:

### Installing ARXAAS:

The arxaas can be installed by installing the docker image and your system and running the local docker image

In order to run the local docker image, you have to make sure that the docker and docker desktop are installed.

Before installing docker on the system, the curl and sudo need to be installed or updated on the system

### Installing curl on Ubuntu 22.04

1. Update the system by running the following command:

| sudo apt get update |
| --- |

1. Installing curl on the system:

| sudo apt install curl |
| --- |

1. After the curl has been installed, it can be verified using the curl command

| $curl |
| --- |

### 

### Installing sudo on Ubuntu 22.04

First we update the database using:

| Sudo apt-get update |
| --- |

Then we install or update the sudo using:

| sudo apt-get -y install sudo |
| --- |

Now that the sudo has been installed, we can move on to installing docker for ubuntu

### Installing docker for Ubuntu 22.04

Before we install the docker engine on our system, it is crucial to update our packages and dependencies and that can be done by running the following command:

| sudo apt-get update |
| --- |

After updating the same, we install packages to allow apt to use a repository over HTTPS:

| sudo apt-get install \  ca-certificates \  curl \  gnupg \  lsb-release |
| --- |

Now we add the docker’s official gpg key

| sudo mkdir -m 0755 -p /etc/apt/keyrings curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o /etc/apt/keyrings/docker.gpg |
| --- |

Now we set up the repository

| echo \  "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.gpg] https://download.docker.com/linux/ubuntu \  $(lsb\_release -cs) stable" | sudo tee /etc/apt/sources.list.d/docker.list > /dev/null |
| --- |

After that is done, we move on to installing the docker

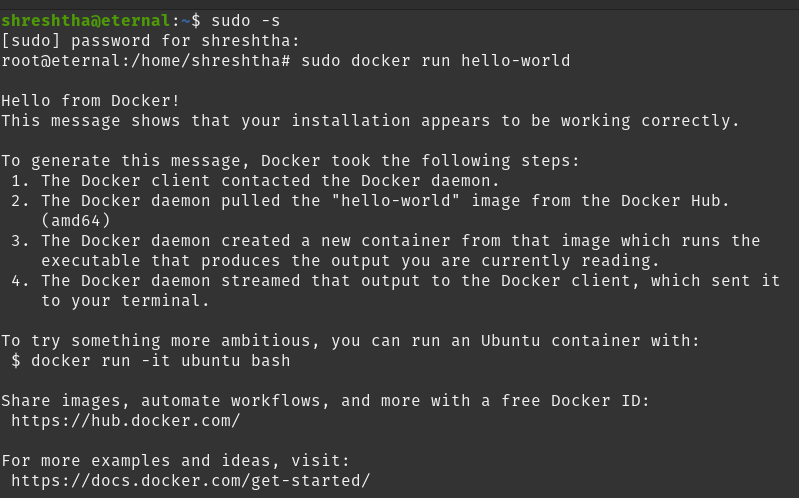


| sudo apt-get install docker-ce docker-ce-cli containerd.io docker-build-plugin docker-compose-plugin |
| --- |

The docker is now installed

We can verify the install by running ‘hello world’ using the command:

| sudo docker run hello-world |
| --- |



This is the output after running the hello world command which indicates that the docker has been running fine

After the docker has been installed, we can install the docker image of the pyarxaas

| docker pull navikt/arxaas |
| --- |

And then running

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

and run the image locally using the port 8080. Simply typing the following url in your browser after running the docker command would run your local docker instance

| http://localhost:8080/ |
| --- |

## 

This is what the arx local docker image looks like

## Installing pyarxaas traditionally:

The pyarxaas can be installed by cloning the github repository

| git clone https://github.com/navikt/pyarxaas |
| --- |

The pyarxaas can also be installed using the package manager pip.

### Installing pip for ubuntu:

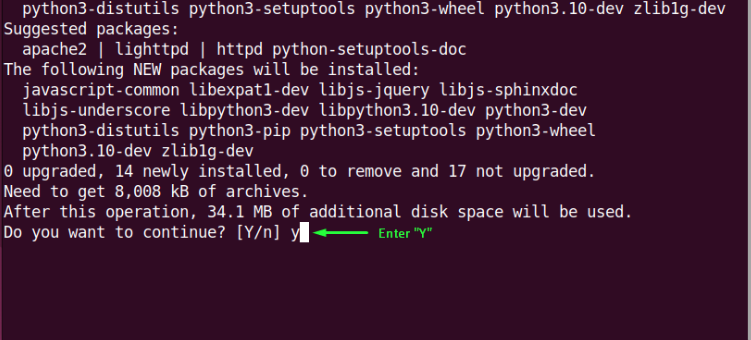
Firstly, run an upgrade on the system using:

| Sudo apt-get upgrade |
| --- |

After running the upgrade, we install the pip by using

| $ sudo apt install python3-pip |
| --- |

When prompted with the question of do you want to continue the installation, please answer yes;

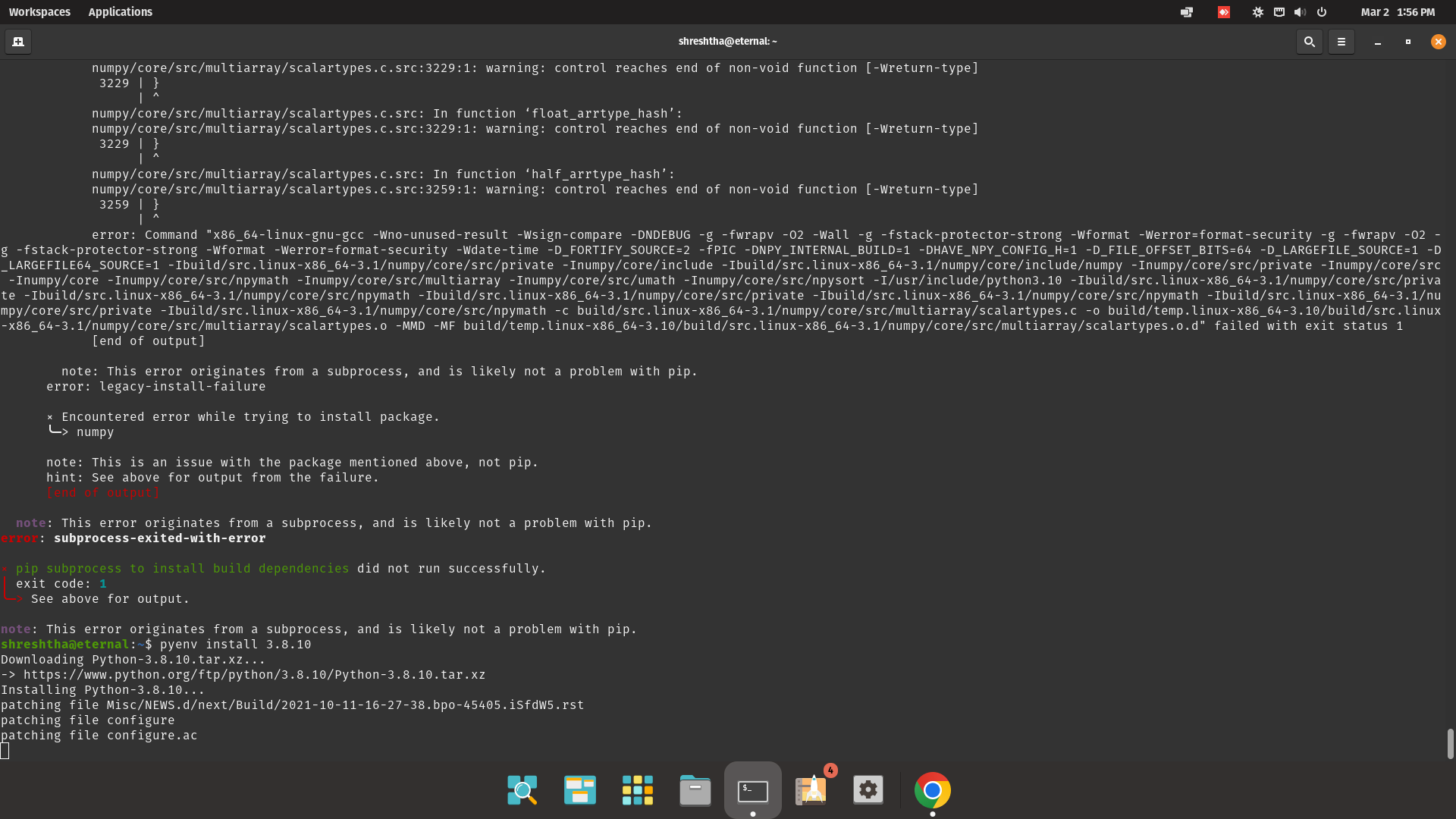


Verify if the pip has been installed successfully by checking the version of the pip

| $ pip3 --version |
| --- |

Now that the pip has been installed, we can start with the installation process of pyarxaas

One thing to note before installing pyarxaas is that without downgrading your system version of python from the latest (python 3.10 or 3.11) to the more older versions of python (3.8.10 or older) pyarxaas might not be installed and you might face errors such as



Hence we need a different environment of python. In order to do that, we can either downgrade the entire system (which is not recommended because it might break other systems dependencies which run on the current version of the python) or you can install multiple versions of python (using a virtual environment) or pyenv

### Installing pyenv for Ubuntu 22.04

1. Update the system and the dependencies using the command

| Sudo apt -get upgrade |
| --- |

1. After updating the system, we can now download the pyenv script and run it using

| $ curl https://pyenv.run | bash |
| --- |

1. After the installation is complete, we add the pyenv variable to the bash file by using the exec command

| export PATH="$HOME/.pyenv/bin:$PATH" && eval "$(pyenv init --path)" && echo -e 'if command -v pyenv 1>/dev/null 2>&1; then\n eval "$(pyenv init -)"\nfi' >> ~/.bashrc |
| --- |

The pyenv is now installed and can be verified by running ‘pyenv- - version’.

Note: In order to see the latest version of the pyenv and the changes that take place, you might need to restart the shell



### Installing python 3.8.10 using pyenv:

1)Installing a specific version of python using pyenv is fairly simple, we can do so using

| pyenv install -v 3.8.10 |
| --- |

2)Now that we have installed the version of the python, we can set it to global using

| Pyenv global 3.8.10  #we can check the version of the python using python —-version |
| --- |

## 

Thus, python 3.8.10 has been installed. With that being installed, we can now install pyarxaas

### Installing Pyarxaas for Ubuntu 22.04

1. Upgrading the dependencies using

| sudo apt-get upgrade |
| --- |

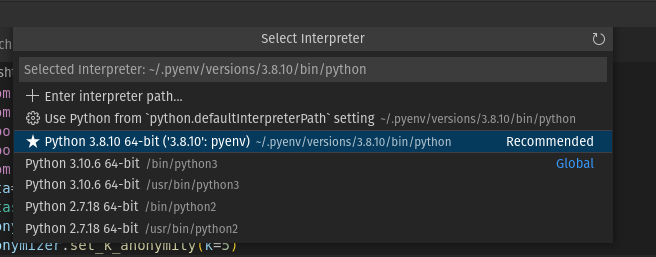
1. After we are done upgrading the system, we can now install pyarxaas using the following command

| Pip install pyarxaas |
| --- |

Upon being asked if you want to install pyarxaas please type yes. Once the arx has been successfully installed, we can verify the install using:

| Pyarxaas -version |
| --- |

After installing the pyarxaas and the python 3.8.10, we still have to change the python version interpreter. This can be done by going into vscode in the prompt and choosing your desired version of python to run



Thus, the kernel of python is ready to run our code

This works perfectly fine. However, please do note that if you choose a version of python that is not 3.8.10 to run the pyarxaas code, it will simply not run because we have installed the pyarxaas on python 3.8.10 version

## Troubleshooting common issues encountered while installing pyarxaas:

* **Curl Error**: While installing docker or pyenv, if your terminal throws in error and doesn’t recognize curl, you can simply update or reinstall the curl by using the command:

| sudo apt install curl/buster-backports |
| --- |

If this still does not work, you can check the path of curl to your system and manually specify the path where your curl is located by using:

| .bashrc  export PATH=/usr/curl/<Directory>/bin:$PATH |
| --- |

Hopefully, this should solve all the errors

* **Sudo error:** If you encounter a sudo error, it is best to install the sudo using sudo apt-get upgrade
* **Numpy/Pandas error:** Since pyarxaas is an older module with discontinued support, the chances of getting this error are high. This error usually occurs when the system has pandas and/or numpy installed which are not supported by pyarxaas. To solve them, simply open the terminal

| Python -version Import numpy Numpy -version #check for the version supported by pyarxaas #downgrade the same using: python -m pip install numpy==x.y.z |
| --- |

* **Pip error**: After installation of pip, pip errors should be unlikely. However, if you still face one of the pip errors, consider changing the version of pip from pip3 to pip in your commands and vice versa. E.g

| pip 3 install python |
| --- |

* **Docker error:** If the docker faces issues during installation, check if your machine supports virtualization by using

| sudo kvm-ok |
| --- |

If your computer supports the virtualization, then you might get an output like:



* **Python error:** For the errors corresponding to python during the installation, please check if the installation of python has been done properly by typing

| Python —-version |
| --- |

If the issue still persists, you might want to point to the path where the python is stored in your machine manually

* **No module named pyarxaas:** Having encountered this issue multiple times, the most common cause for this issue is not using the version of python where the pyarxaas was installed. Simply changing the version of python can resolve this issue

## What is pyarxaas

Functions supported

Identitfers in the dataset

What is a risk profile and how to generate one

Types of hierarchy supported

## Exploring pyarxaas with the titanic dataset

## Creating the fake actual data

## Faker library, installation and common functions

## Creating a dummy dataset using faker

## Writing functions to set the hierarchy and anonymize the dataset

## 

ARX is an opensource tool for data anonymization which is used for static data. ARX

How to install arx:?

ARX can be installed using a browser. For linux-specific flavours, arx can be installed using simply downloading the .deb package from the website and running it using the CLI

ARX also offers an API which is not convenient for our use case. The flavour of arx used here is the popular python module pyarxaas.

Installing Pyarxaas:

Pyarxaas can be installed using



Workaround:

**Part TWO:**

ARX demo:

I tinkered around with the arx module to get a better understanding of the same using a dummy dataset. The end goal here is to understand and configure the module better for the final product.

The dataset being used here is the famous titanic dataset from kaggle which has the information about the name, class, ticket and the fare of the passengers which boarded the titanic. Since the data has a lot of sensitive columns, it is perfect for the analysis.

Understanding the hierarchy of the data and the types of the data

References:

<https://itslinuxfoss.com/install-use-pyenv-ubuntu/>

<https://pyarxaas.readthedocs.io/en/latest/user-guide/connect-to-arxaas.html#hierarchy-generation>

<https://docs.docker.com/engine/install/ubuntu/>

Code:

The code is divided into three parts:

1. Creating the dataset
2. Setting the hierarchies for the dataset
3. Anonymizing the dataset and comparing the risk profile



| this function sets the hierarchy for the datasetto get it ready for the anonymization #load the data and the neccesary libraries #we will start by importing the neccesarry libraries import pyarxaas from pyarxaas import ARXaaS from pyarxaas.hierarchy import IntervalHierarchyBuilder, OrderHierarchyBuilder,RedactionHierarchyBuilder from pyarxaas import AttributeType from pyarxaas.privacy\_models import KAnonymity from pyarxaas import Dataset import pandas as pd import pandas as pd #instead of reading the csv files multiple times on their own, it makes a lot more sense to store that as pickle objects import pickle from pandas.api.types import CategoricalDtype #this function takes in the dataframe and if you want the risk profile or not and returns the results based on that def set\_hierarchy(data=pd.DataFrame,risk\_profile=False):  # Check if hierarchy files exist and load the neccesarry files  name\_hier\_file = "/home/shreshtha/Hiearchies/namehier.csv"  age\_hier\_file = "/home/shreshtha/Hiearchies/agehier.csv"  mail\_hier\_file = "/home/shreshtha/Hiearchies/emailhier.csv"  gender\_hier\_file = "/home/shreshtha/Hiearchies/gender.csv"  zip\_hier\_file = "/home/shreshtha/Hiearchies/zipcode.csv"  amount\_hier\_file = "/home/shreshtha/Hiearchies/amounthier.csv"    try:  # Load pickled hierarchy files  with open('name\_hier.pkl', 'rb') as f:  name\_hierarchy = pickle.load(f)  with open('age\_hier.pkl', 'rb') as f:  age\_hierarchy = pickle.load(f)  with open('mail\_hier.pkl', 'rb') as f:  mail\_hierarchy = pickle.load(f)  with open('gender\_hier.pkl', 'rb') as f:  gender\_hierarchy = pickle.load(f)  with open('zip\_hier.pkl', 'rb') as f:  zip\_hierarchy = pickle.load(f)  with open('amount\_hier.pkl', 'rb') as f:  amount\_spent\_hierarchy = pickle.load(f)  #exception handling to catch the errors  except FileNotFoundError:  # Load hierarchy files from CSV and pickle for future use  name\_hierarchy = pd.read\_csv(name\_hier\_file)  with open('name\_hier.pkl', 'wb') as f:  pickle.dump(name\_hierarchy, f)  age\_hierarchy = pd.read\_csv(age\_hier\_file)  with open('age\_hier.pkl', 'wb') as f:  pickle.dump(age\_hierarchy, f)  mail\_hierarchy = pd.read\_csv(mail\_hier\_file)  with open('mail\_hier.pkl', 'wb') as f:  pickle.dump(mail\_hierarchy, f)  gender\_hierarchy = pd.read\_csv(gender\_hier\_file)  with open('gender\_hier.pkl', 'wb') as f:  pickle.dump(gender\_hierarchy, f)  zip\_hierarchy = pd.read\_csv(zip\_hier\_file)  with open('zip\_hier.pkl', 'wb') as f:  pickle.dump(zip\_hierarchy, f)  amount\_spent\_hierarchy = pd.read\_csv(amount\_hier\_file)  with open('amount\_hier.pkl', 'wb') as f:  pickle.dump(amount\_spent\_hierarchy, f)   # Set attribute types and hierarchies  dataset = Dataset.from\_pandas(data)  #running the local arx instance  arxaas = ARXaaS("http://localhost:8080/")  #setting the attribute types of arx  dataset.set\_attribute\_type(AttributeType.QUASIIDENTIFYING, 'Age', 'zipcode', 'Amount\_spent', 'Gender')  dataset.set\_attribute\_type(AttributeType.INSENSITIVE,'Name','Email')  if risk\_profile==True:  #generate the risk profile of the data:  risk\_profile = arxaas.risk\_profile(dataset)  #the risk profile has various columns and now we are looking at some of the terms in the risk profile such as reidentification risk  print(risk\_profile.re\_identification\_risk)  #let us now look at the success rate of the attackers  (risk\_profile.attacker\_success\_rate)  #there are several attacker models such as journalist, prosecuter and the arxaas tells us the risk of all of them  #we are now looking at the distribution of the risk for vatious intervals  (risk\_profile.distribution\_of\_risk)  #iterating through the columns of the dataset  for column in ['Age', 'zipcode', 'Amount\_spent', 'Gender']:  #error handling to point out if the data is not proper  if column not in data.columns:  raise ValueError(f"{column} column not found in data")  #setting the hierarchies for the columns of data  elif column == 'Age':  dataset.set\_hierarchy('Age', age\_hierarchy)  elif column == 'Gender':  dataset.set\_hierarchy('Gender', gender\_hierarchy)  elif column=='zipcode':  dataset.set\_hierarchy('zipcode',zip\_hierarchy)  else:  dataset.set\_hierarchy('Amount\_spent',amount\_spent\_hierarchy)  #function ends  return dataset |
| --- |