An introduction to data privacy and 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 Anonymization is a form of privatising the data before making it public or sending it to the intended audience.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.

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

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.

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

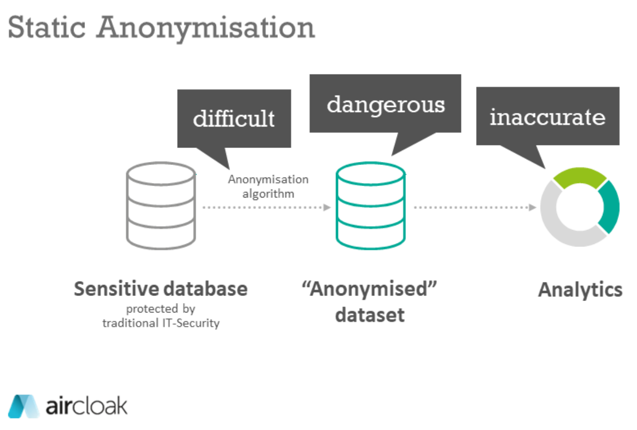
Types of anonymization

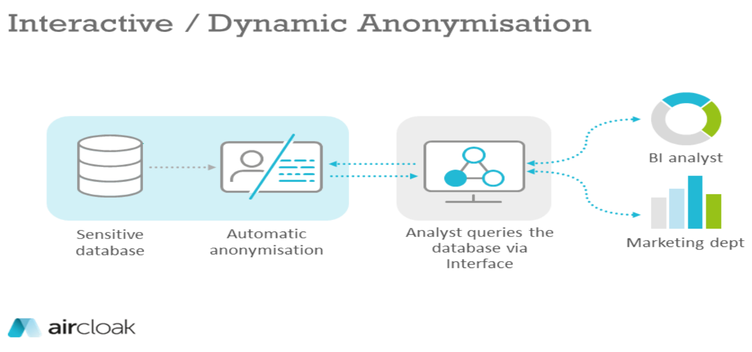
There are two types of anonymization:

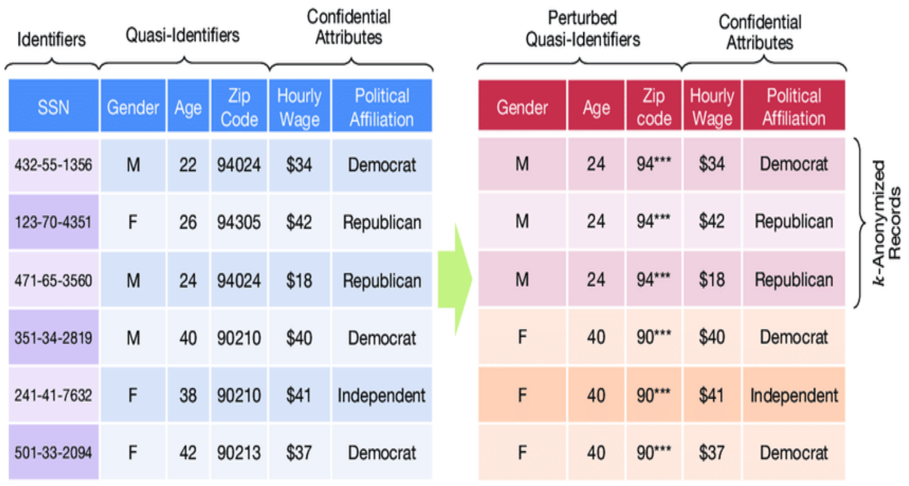
1. Static
2. Dynamic

Static anonymization: Static anonymization refers to the anonymization of the data all at once and then the data is being released to the public or the third party source or vendor. In static anonymization, often a subset of the original data is released after anonymization to the users. Popular static anonymization tools and softwares are ARX, Amnesia etc

Dynamic Anonymization: Dynamic anonymization refers to the anonymization of the data using queries. Often the full dataset is released to the public and the anonymization of the dataset is happening in real-time. Dynamic anonymization is considered more reliable than the static one because static anonymization needs to specify the anonymization techniques very carefully otherwise the subset of the data can be extracted multiple times to paint the picture of the actual data. Some popular dynamic anonymization techniques are: The popular R package diffpriv, Google's RAPPOR etc




Interactive / Dynamic Anonymisation
Sensitive database
Automatic anonymisation
Analyst queries the database via Interface
BI analyst
Marketing dept
aircloak



Identifiers	Quasi-Identifiers			Confidential Attributes	
SSN	Gender	Age	Zip Code	Hourly Wage	Political Affiliation
432-55-1356	M	22	94024	$34	Democrat
123-70-4351	F	26	94305	$42	Republican
471-65-3560	M	24	94024	$18	Republican
351-34-2819	M	40	90210	$40	Democrat
241-41-7632	F	38	90210	$41	Independent
501-33-2094	F	42	90213	$37	Democrat

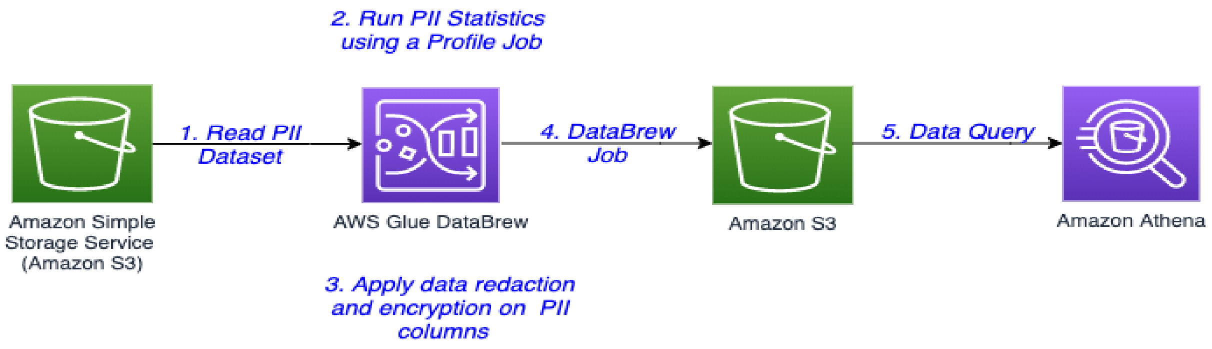
Perturbed Quasi-Identifiers			Confidential Attributes	
Gender	Age	Zip code	Hourly Wage	Political Affiliation
M	24	94***	$34	Democrat
M	24	94***	$42	Republican
M	24	94***	$18	Republican
F	40	90***	$40	Democrat
F	40	90***	$41	Independent
F	40	90***	$37	Democrat

Figure 1 The process of anonymization

Proposed solution 1:

Anonymization of the data using aws databrew

Diagram


2. Run PII Statistics using a Profile Job
Amazon Simple Storage Service (Amazon S3)
1. Read PII Dataset
AWS Glue DataBrew
4. DataBrew Job
Amazon S3
5. Data Query
Amazon Athena
3. Apply data redaction and encryption on PII columns

Components:

1. **AWS S3:** As explained in the previous solution, s3 is an object based service used to store the data
2. **AWS Glue DataBrew:** Is a service by amazon that aids data scientists and machine learning engineers in cleaning and normalising the data.
3. **AWS Athena**: Amazon Athena is a serverless, interactive analytics service built on open-source frameworks, supporting open-table and file formats. Athena provides a simplified, flexible way to analyse petabytes of data where it lives.

Explanation:

the data in question to be anonymized is stored in an s3 bucket. Moreover, macie can be used to detect sensitive data from the already present data. The data from s3 is then loaded into the aws glue data brew which has jobs which mask the sensitive PII column-wise and after the data is masked, the masked data is then stored into s3 which creates a external table on top of athena

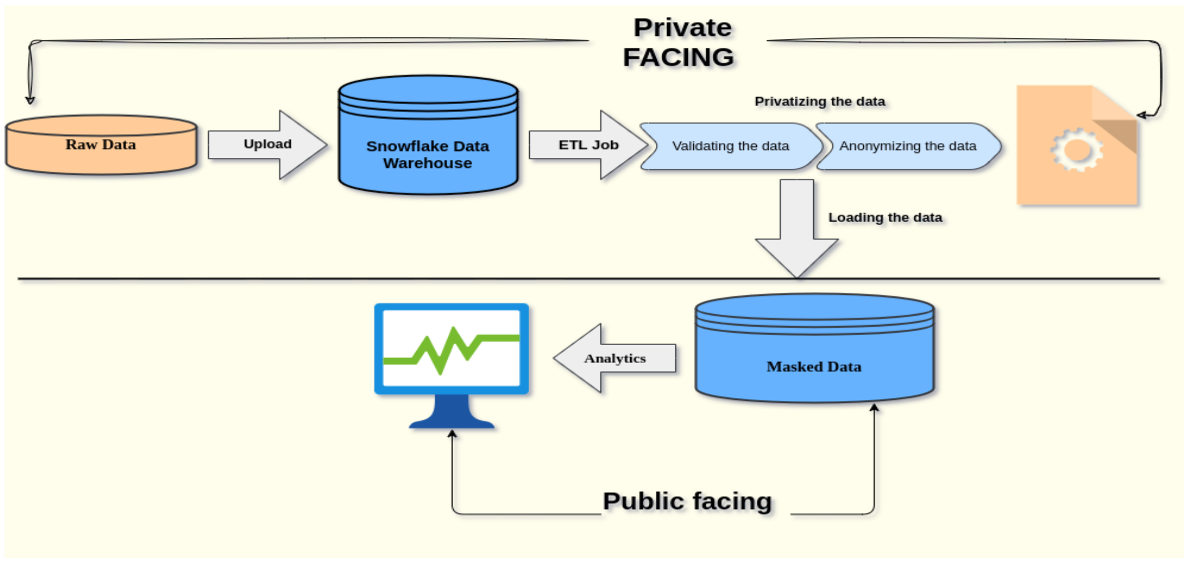
Feedback from the mentor and drawbacks:

Using aws macie to detect sensitive data might work for use cases where the user might have accidentally entered the sensitive data such as credit card information inside the s3 bucket. In our use case, the data that we want to mask is just not sensitive data, but the data that can be used in conjunction with other data to identify the users as well which is something where macie fails. Additionally, the aws data brew has limited support for masking the sensitive data and if the attacker knows the method or the technique using which the data was masked, it might be easy for the attacker to re-identify the data. Overall, we need a solution that is fast, has versioning capability, is able to handle legacy data and also uses a predefined algorithm to anonymize a data with a known schema while being cost optimised

Proposed solution 2

Data Anonymization Pipeline using Snowflake and airflow

Diagram



Components:
Snowflake: Snowflake is a SaaS data warehouse solution designed solely for the cloud, it supports popular cloud providers like gcp, aws and azure. The snowflake can be used to build data warehouse and data lake right from your browser as snowflake is a managed service
PYARXAAS: Is a popular python module which is used as a static anonymization tool. Pyarxaas provides a wrapper to access functions for your local arx instance.
Apache Airflow: Apache airflow is an open source, etl tool that is used to automate the data loading and data transforming pipeline

Explanation:
The following solution provides an end to end pipeline for anonymizing data that gives users the flexibility to scale the solution as per their demand or data. Here is the process of pipeline:

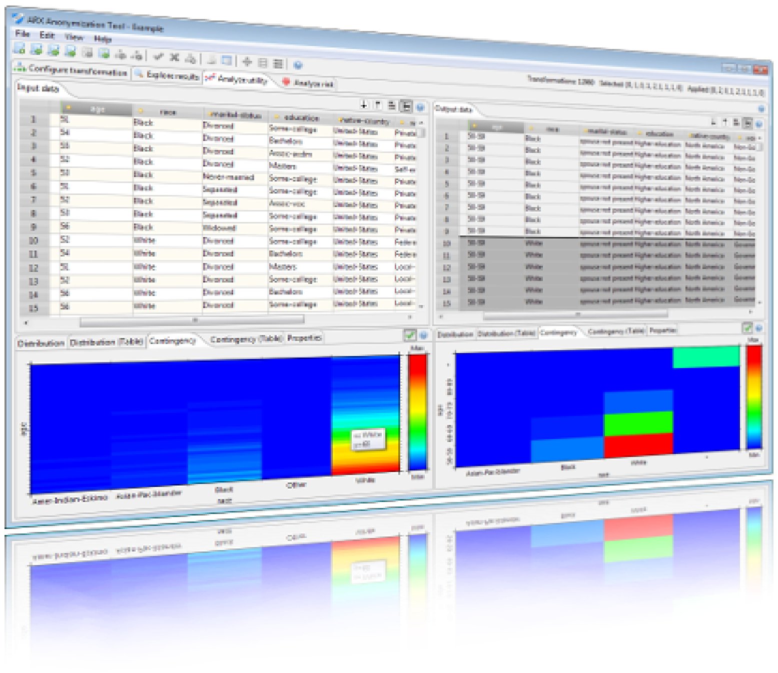
1. Takes in the raw data either from the source or from a sql database. What makes the solution truly customizable is that snowflake can take in data from multiple sources Snowflake natively supports AVRo, Parquet, CSV, JSON and ORC hence data from pretty much any source can be taken

2. The raw data is then loaded into the snowflake to create a data warehouse. The access to the given snowflake is heavily restricted. This step is also customizable and scalable as if you want to give certain users access to the raw data warehouse, you can simply assign their role access and they can be managed using snowflake organisation. Thus, reducing the costs and increasing the flexibility
3. After that, an ETL job is run on the data warehouse which validates the data if need, sets the hierarchy of anonymization for the data and actually anonymizes the data. The anonymization being used here is a function written manually and will be triggered after a certain time when the data is loaded in the data warehouse. Here, since we already know the project columns and all the details about the data being present inside we will use static anonymization instead of dynamic but dynamic anonymization can be used as well
4. The anonymized data is then stored in another snowflake data table or this can be also be connected to the cloud provider of your choice but an interesting thing about snowflake is that it really lets you create stunning visualisations using the snowflake console itself
5. The masked data which is not public facing can be used to generate charts, analysis and visualisation

Comments from the mentor and drawbacks:

One of the biggest advantages of this solution is the scalability in terms of compute and storage. SInce snowflake is a hybrid between the shared nothing and shared everything architecture, it is really easy to scale up or down. The manual data anonymization gives the users greater flexibility to implement their own algorithm. However, one potential challenge to solve is trying to anonymize the data automatically without having the schema of the data. This is still a challenge since pyarxaas is a static anonymization algorithm References: https://aircloak.com/data-anonymisation-software-differences-between-static-and-interactive-anonymisation

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. 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 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

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. 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.
3. 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.

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 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:

All about ARX

```
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

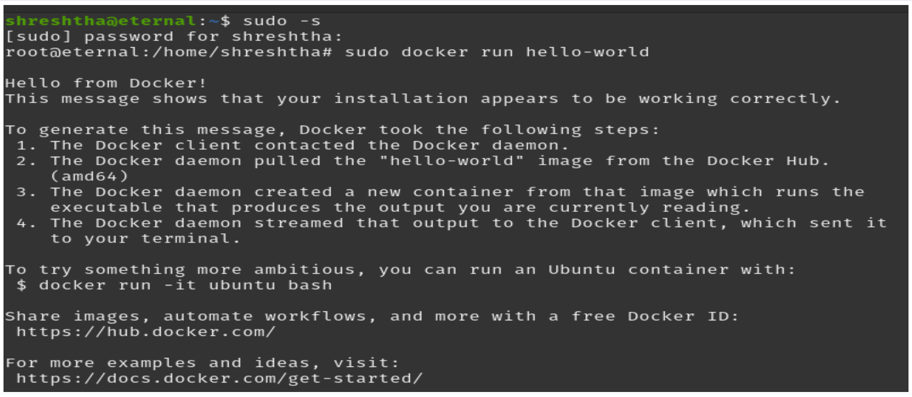
```
echo \
    "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.gpg]
https://download.docker.com/linux/ubuntu \
  $(lsb_release -cs) stable" | sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
```

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
```

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

```
sudo docker run hello-world
```



```
shreshtha@eternal:~$ sudo -s
[sudo] password for shreshtha:
root@eternal:/home/shreshtha# sudo docker run hello-world

Hello from Docker!
This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:
 1. The Docker client contacted the Docker daemon.
 2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
    (amd64)
 3. The Docker daemon created a new container from that image which runs the
    executable that produces the output you are currently reading.
 4. The Docker daemon streamed that output to the Docker client, which sent it
    to your terminal.

To try something more ambitious, you can run an Ubuntu container with:
 $ docker run -it ubuntu bash

Share images, automate workflows, and more with a free Docker ID:
 https://hub.docker.com/

For more examples and ideas, visit:
 https://docs.docker.com/get-started/
```

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
```

Modi Shreshtha (EC)
190130111081

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/
```

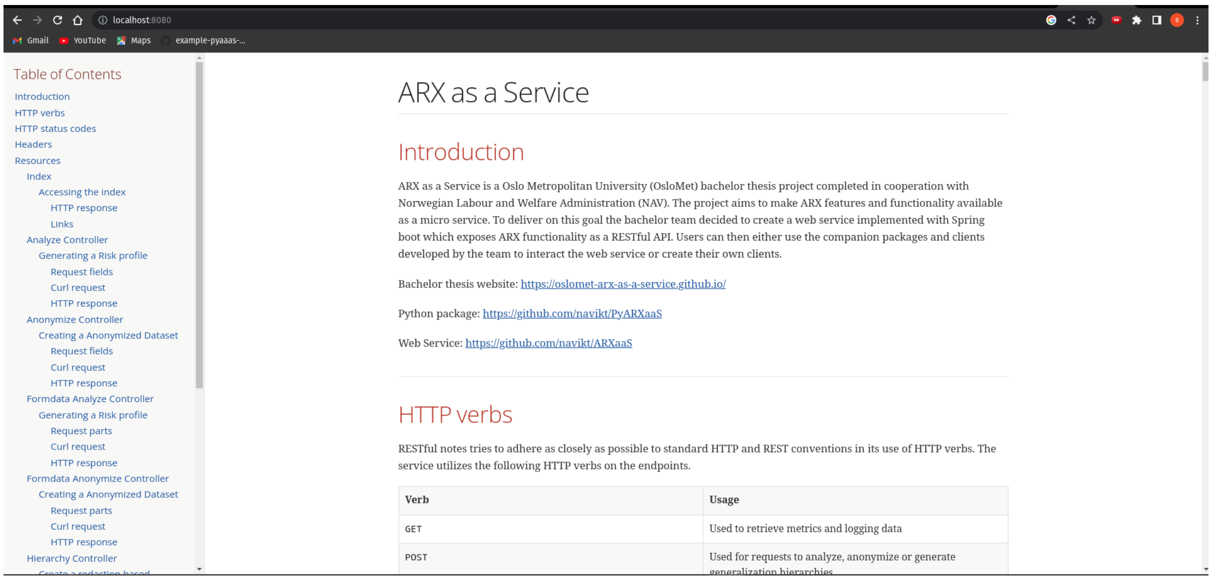


Table of Contents

ARX as a Service

Introduction

ARX as a Service is a Oslo Metropolitan University (OsloMet) bachelor thesis project completed in cooperation with Norwegian Labour and Welfare Administration (NAV). The project aims to make ARX features and functionality available as a micro service. To deliver on this goal the bachelor team decided to create a web service implemented with Spring boot which exposes ARX functionality as a RESTful API. Users can then either use the companion packages and clients developed by the team to interact the web service or create their own clients.

Bachelor thesis website: https://oslomet-arx-as-a-service.github.io/

Python package: https://github.com/navikt/PyARXaaS

Web Service: https://github.com/navikt/ARXaaS

HTTP verbs

RESTful notes tries to adhere as closely as possible to standard HTTP and REST conventions in its use of HTTP verbs. The service utilizes the following HTTP verbs on the endpoints.

Verb	Usage
GET	Used to retrieve metrics and logging data
POST	Used for requests to analyze, anonymize or generate

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.

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
```

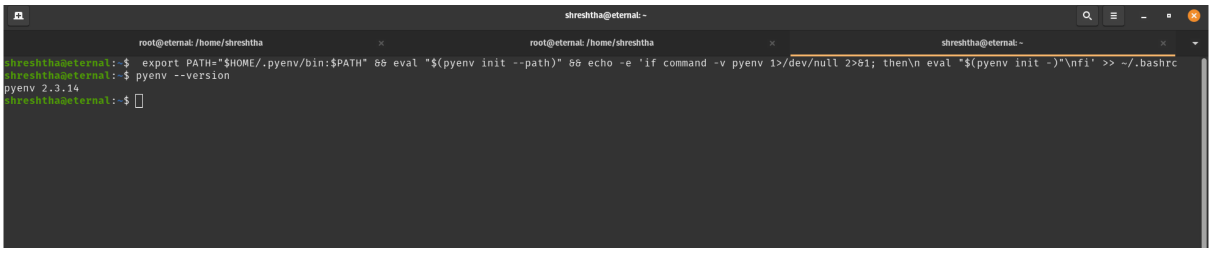
2) After updating the system, we can now download the pyenv script and run it using

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

3) 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



```
shreshtha@eternal:~$  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
shreshtha@eternal:~$ pyenv --version
pyenv 2.3.14
shreshtha@eternal:~$ 
```

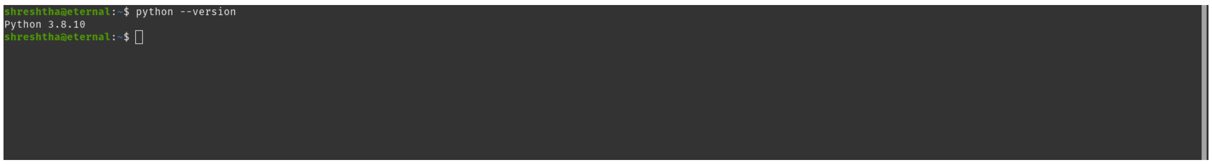
Installing python 3.8.10 using pyenv:

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
```



```
shreshtha@eternal:~$ python --version
Python 3.8.10
shreshtha@eternal:~$ 
```

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
```

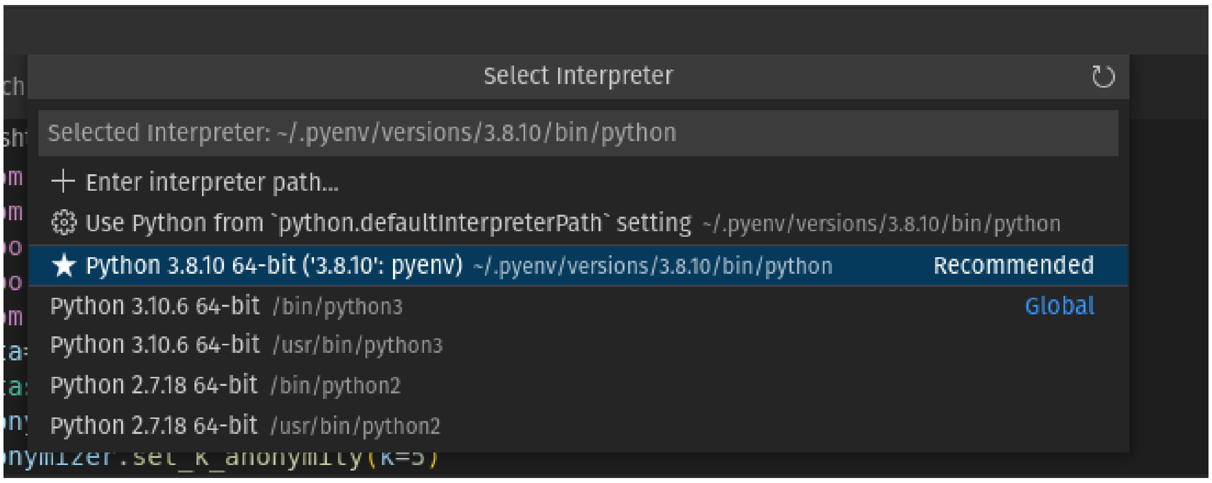
2) After we are done upgrading the system, we can now install pyarxaas using the following command

```
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



```
Select Interpreter
Selected Interpreter: ~/.pyenv/versions/3.8.10/bin/python
+ Enter interpreter path...
Use Python from `python.defaultInterpreterPath` setting ~/.pyenv/versions/3.8.10/bin/python
★ Python 3.8.10 64-bit ('3.8.10': pyenv) ~/.pyenv/versions/3.8.10/bin/python    Recommended
Python 3.10.6 64-bit /bin/python3                                              Global
Python 3.10.6 64-bit /usr/bin/python3
Python 2.7.18 64-bit /bin/python2
Python 2.7.18 64-bit /usr/bin/python2
nymizer.set_k_anonymity(k=5)
```

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

What is pyarxaas

Pyarxaas is the python wrapper for accessing the arx functions on a local arx instance. It can be downloaded via github or using pip.
Functions supported

AttributeType

Attribute type is used to set the attributes to a given dataset. Usually the columns of a dataset fall into either of these categories:
Identifying: A column of a dataset is called identifying when it contains values that can be used to directly identify the attributes of a person. E.g. Name, E-mail, Aadhar card number etc
Quasi Identifying: Quasi Identifying columns are the columns which do not contain the identifying information directly but they can be used in combination with other data and or columns to re identify the person
Let us take an example to understand the quasi identifying columns a little better:
A dataset contains zip code, gender and items purchased. Now, the columns on their own might not be enough to identify the user who purchased the dataset but they can be used in combination. E.g zip code narrows down the area of the user and we can use the gender column to filter out the gender of the user. This dataset in conjunction with customer reviews or lets say order placed can be used to identify the user and hence reveals the sensitive personal information.

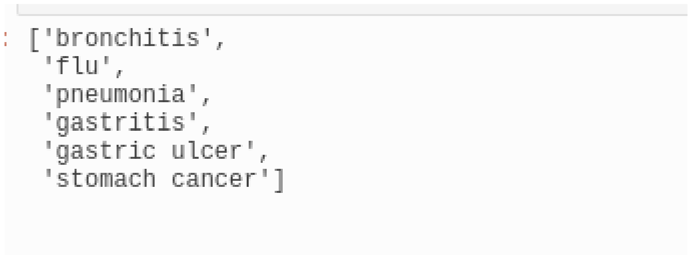
Sensitive: Sensitive attributes are the columns in your data which contain important and crucial information about the user which cannot be reveleade. E.g Bank account number, social security details etc etc. These columns should be removed from the data entirely

Insensitive: Insensitive columns are the columns which do not contain any identifying or sensitive information and they need not be anonymized. E.g when looking at the customer dataset insensitive information could be the brand he rejected or the number of items that he purchased

Types of hierarchy supported

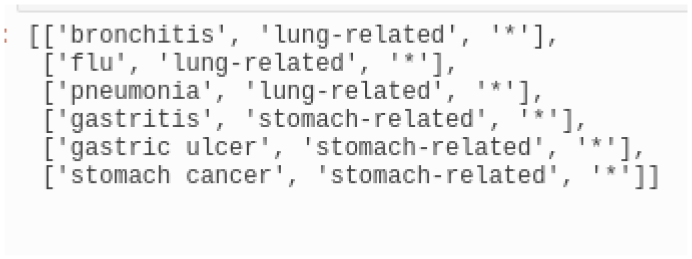
Hierarchy is nothing but the levels of generalisation defined when anonymizing the data.The pyarxaas module supports a variety of hierarchy types. You can either set your own hierarchy for generalisation or create one in pyarxaas. Here are some of the supported hierarchy types in pyarxaas:

Order based Hierarchy: Order based hierarchy are suited for the categorical variables. I.e The value of variables can not be something other than the value from the specified list of values
Here, we have a list of diseases for a patient dataset



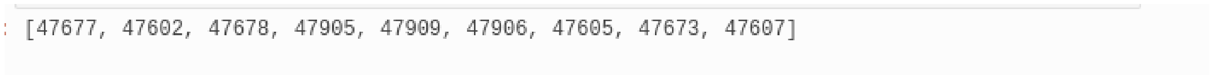
```
: ['bronchitis',
   'flu',
   'pneumonia',
   'gastritis',
   'gastric ulcer',
   'stomach cancer']
```

This is what the hierarchy looks like for the following diseases:

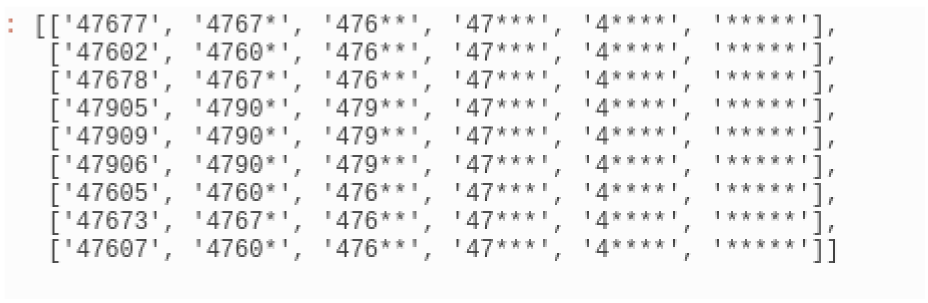


```
: [['bronchitis', 'lung-related', '*'],
   ['flu', 'lung-related', '*'],
   ['pneumonia', 'lung-related', '*'],
   ['gastritis', 'stomach-related', '*'],
   ['gastric ulcer', 'stomach-related', '*'],
   ['stomach cancer', 'stomach-related', '*']]
```

Redaction based hierarchy: Redaction based hierarchy are best suited for numeric but categorical values. E.g phone number or zip code. They take in a list and delete one number at a time from the attribute column until the privacy criteria is met
Example:
Here we have a list of zip codes



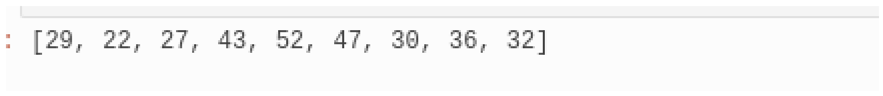
```
: [47677, 47602, 47678, 47905, 47909, 47906, 47605, 47673, 47607]
```



```
: [['47677', '4767*', '476**', '47***', '4****', '*****'],
   ['47602', '4760*', '476**', '47***', '4****', '*****'],
   ['47678', '4767*', '476**', '47***', '4****', '*****'],
   ['47905', '4790*', '479**', '47***', '4****', '*****'],
   ['47909', '4790*', '479**', '47***', '4****', '*****'],
   ['47906', '4790*', '479**', '47***', '4****', '*****'],
   ['47605', '4760*', '476**', '47***', '4****', '*****'],
   ['47673', '4767*', '476**', '47***', '4****', '*****'],
   ['47607', '4760*', '476**', '47***', '4****', '*****']]
```

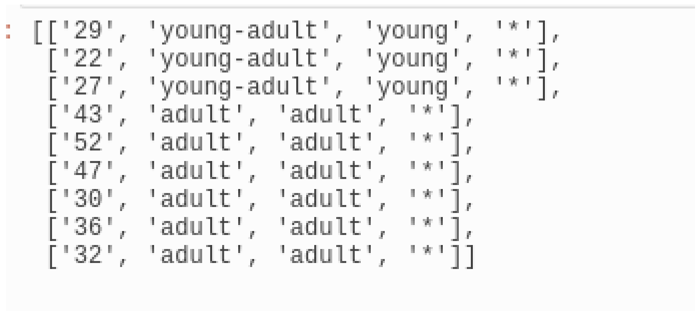
Interval based hierarchy: Interval based hierarchy typically works well for the continuous numeric values such as age, height, weight, credit card number etc. The attribute column here gets divided into generalised level based instead of the actual numbers

Let's say we have a list of the age group for the customer data:



```
: [29, 22, 27, 43, 52, 47, 30, 36, 32]
```

This is what the final hierarchy looks like:



```
: [['29', 'young-adult', 'young', '*'],
   ['22', 'young-adult', 'young', '*'],
   ['27', 'young-adult', 'young', '*'],
   ['43', 'adult', 'adult', '*'],
   ['52', 'adult', 'adult', '*'],
   ['47', 'adult', 'adult', '*'],
   ['30', 'adult', 'adult', '*'],
   ['36', 'adult', 'adult', '*'],
   ['32', 'adult', 'adult', '*']]
```

privacy_models

What are privacy models?

Privacy models refer to a set of techniques, methods, and frameworks that are used to protect the privacy of individuals in a data collection or analysis process. These models provide a structured approach to ensure that sensitive or personally identifiable information is not disclosed, while still allowing useful information to be extracted for research or analysis purposes.

Privacy models often involve a combination of statistical, cryptographic, and computational techniques to achieve their goals. They can be used to enforce different levels of privacy protection, depending on the specific needs and requirements of a particular use case or application.

Privacy Models supported by arx:

ARX supports the following privacy models:

K-anonymity:

K-Anonymity is a privacy model designed to protect the identity of individuals whose personal information is being used in a dataset. This model ensures that an individual cannot be re-identified from a dataset by ensuring that each record in the dataset is indistinguishable from at least k-1 other records.

- In other words, K-Anonymity is a technique that anonymizes data by grouping individuals with similar characteristics into clusters, where each cluster contains at least k individuals. By doing so, it makes it harder for an attacker to identify a particular individual in the dataset.

L-diversity:

- L-Diversity is a privacy model that ensures that sensitive information of individuals in a dataset is not revealed by adding diversity to the dataset. It aims to prevent attackers from identifying individuals by adding enough diversity to the dataset to make it difficult to link specific sensitive attributes to a particular individual.

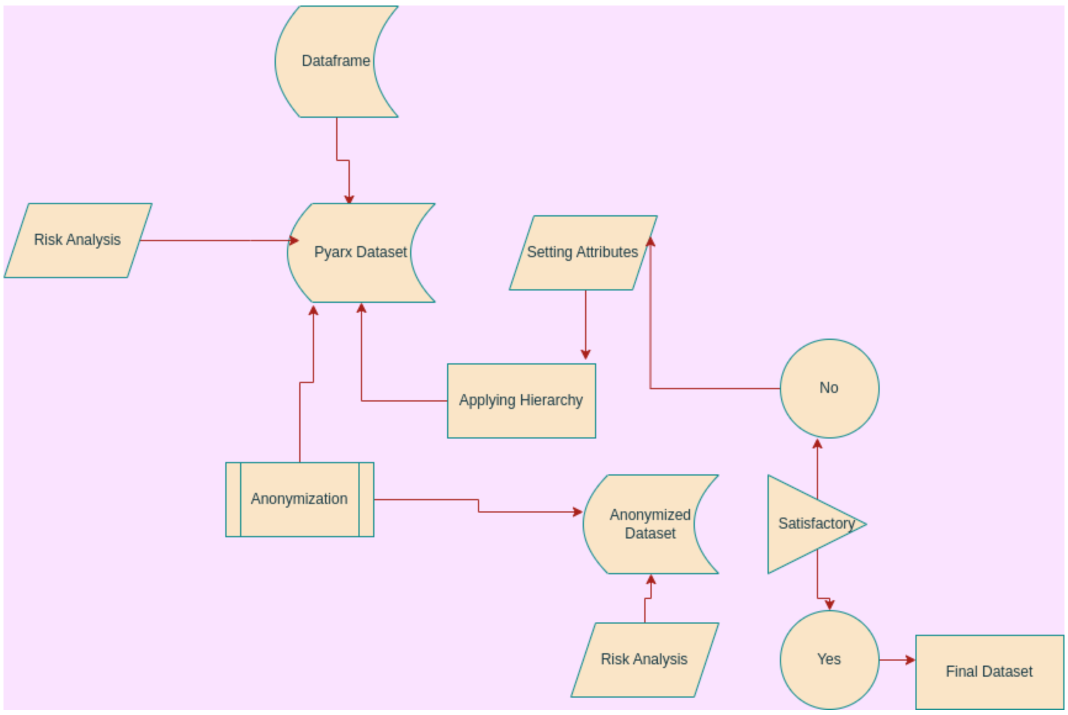
- In other words, L-Diversity ensures that every group of individuals in the dataset is diverse enough in terms of sensitive attributes such as race, religion, or medical condition, so that it is not possible to link these attributes to a specific individual in the group.
- L-Diversity is particularly useful in scenarios where sensitive attributes need to be protected while still allowing data to be used for analysis, research, or other purposes. It is used in many different applications, including healthcare, finance, and social research.

T-closeness:

- T-closeness is a privacy model that measures the degree to which a dataset preserves the privacy of individuals by ensuring that the distribution of sensitive attributes in the dataset is similar to their distribution in the general population. The aim is to prevent attackers from using background knowledge to link specific sensitive attributes to particular individuals in the dataset.
- In other words, T-closeness ensures that the distribution of sensitive attributes (such as age, race, or medical condition) in the dataset is not significantly different from the distribution of those attributes in the general population, to avoid revealing sensitive information about specific individuals.
- T-closeness can be measured using a distance metric that measures the difference between the distribution of a sensitive attribute in the dataset and its distribution in the general population. The goal is to minimise this distance, or "closeness", to protect the privacy of individuals in the dataset.

```
#sample code to set the anonymization for the dataset
kanon = K Anonymity(2)
ldiv = LDiversityDistinct(2, "disease") # in this example the dataset has a disease field
anonymize_result = arxaas.anonymize(dataset, [kanon, ldiv], 0.2)
anonymized_dataset = anonymize_result.dataset
```

End to End process of anonymizing data with pyarxaas:



Creating the fake actual data

The data being used for further analysis from this point on will replicate the real life customer data very closely and will contain columns such as names, gender, email address and zip codes. The dataset is created using a python library called faker. Faker lets you create fake datasets in python with ease

Installing faker:

Faker can be installed using the python library package manager called pip

```
pip install Faker
```

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/

#the faker data will be inserted here