Merative & Macquarie University Joint Project

# Dynamic Private Record Linkage

User manual

Group 26

**Table of contents**

## 1.1 Introduction of product

This project is dedicated to developing a software program that establishes Dynamic Private Record Linkage. This section will be dedicated to describe how the prototype works and what functions are accomplished, as well as a pre-installation guide for handover requirements.

In this project, we first list the original data from the database sets which involve a primary key and other attributes such as postcode and name, etc. Then we use Bloom filters to encode those attributes together into a long binary array that only consists of hashed values to protect user privacy. The primary attribute is not encrypted as it is strictly used as an evaluation metric to ensure the prototype works. We use the clustering method to compare the hashed records and match those records which have high similarity. Finally, these records with high similarity from different databases are output as the final result in the form of a new CSV file.

## 1.2 Product usage guide

### *For administrator*

The prototype was built using Python and we will be using GitHub as our repository. GitHub is a platform which allows developers and users to interact. An advantage of implementing GitHub as our repository is that version control and ongoing updates would be easy to access for both end-users and administrators. The following demonstration for the product will be operated under a cloud platform named Google Collaboration. The cloud-based platform is convenient for writing Python code smoothly without complicated configuration issue. Therefore, the installation and setup process is simple compared to a traditional coding application like Microsoft Visual Code. Furthermore, as the cloud platform is used for running codes, lowering the system requirement for using the program that we have developed.

### *For end-user*

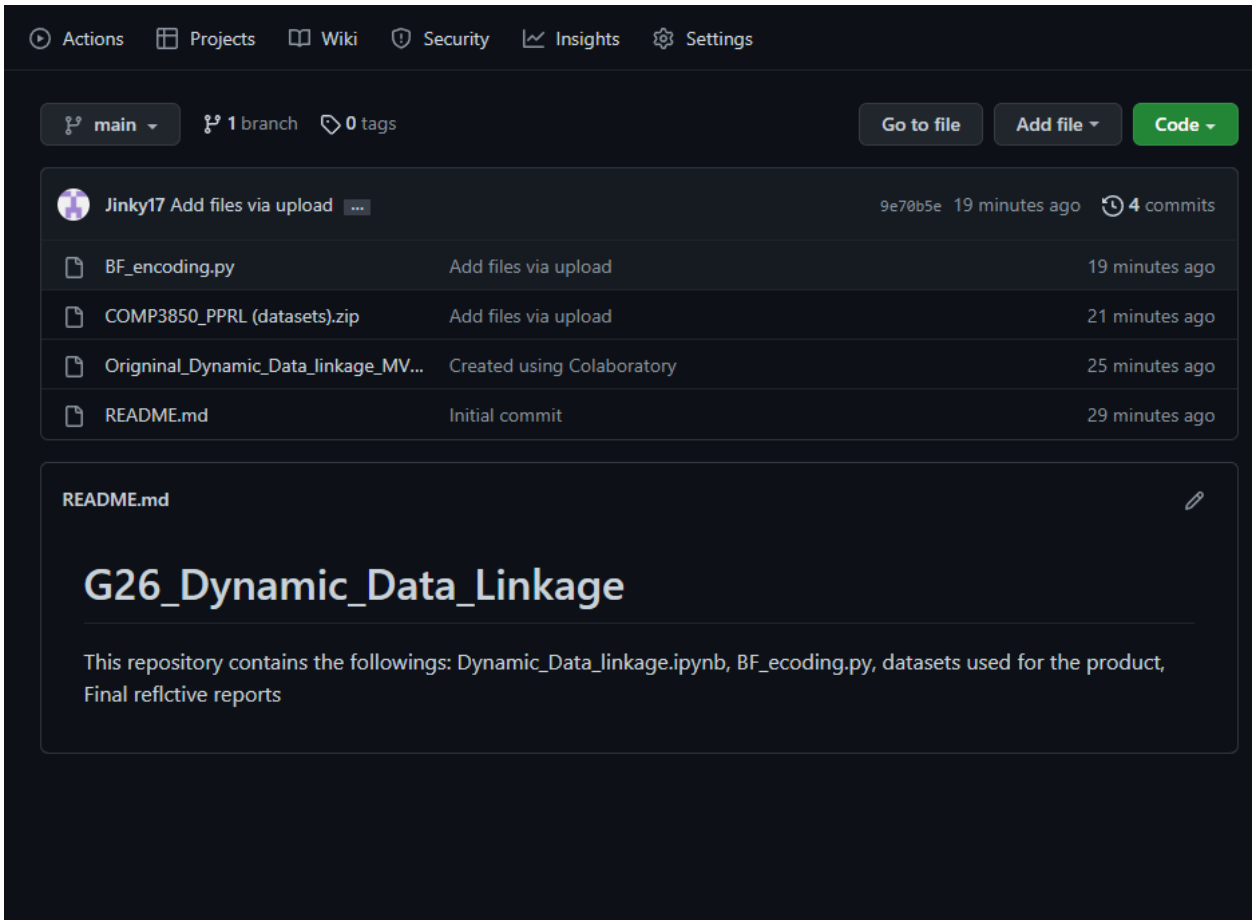
The program was built via google collaboration and the programming language used is python. Hence, it is easy to use and does not require any additional software installed in the users'

computer. The requirements for users to run this program are as follows: A computer with internet connection and a browser.

## ~~1.3 Introduction to User interface ( UI ) of the product~~

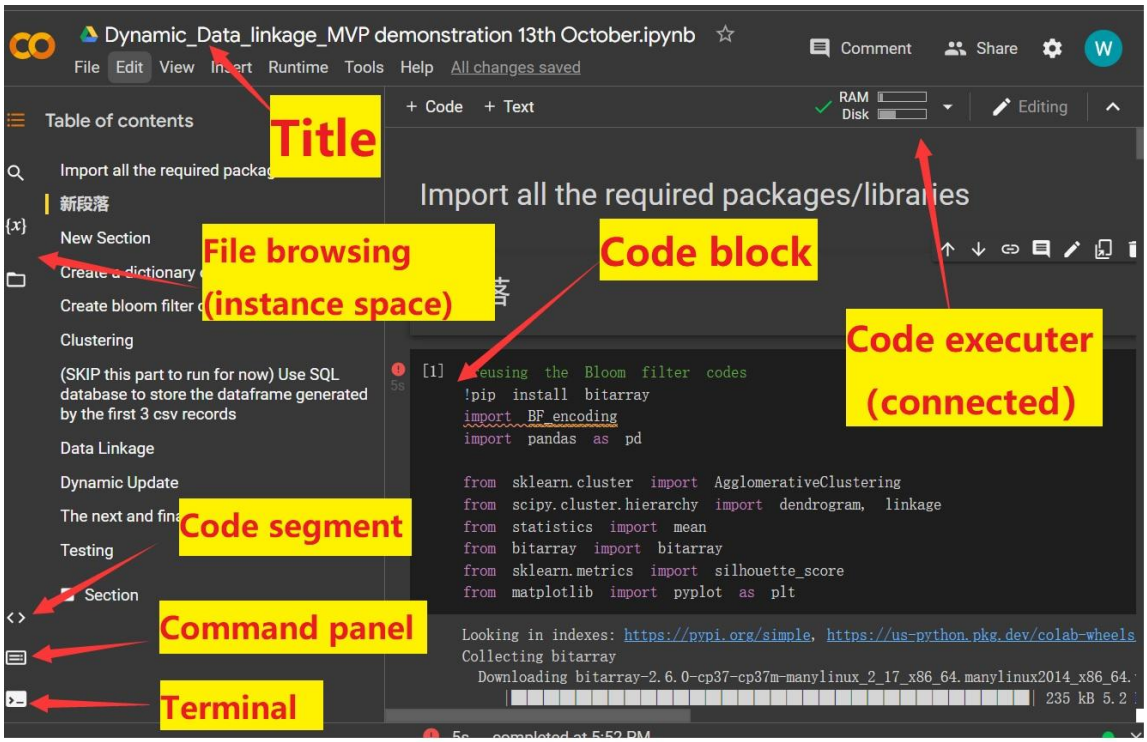
The user interface in software that we have developed is primarily focused on keeping everything as simple as possible so that end-users can understand and complete the required tasks effectively and efficiently.

The diagram below shows the GItHub repository of our product. Which allow users and administrators to provide feedbacks and problems that they might face during the installation process and runtime of the product. We will further explain further explain the installation process with the diagram shown below.



Once users and admin navigated to the GitHub repository with the link provided, they should be getting the same layout as shown in the diagram above. In which the repository will be containing the

The diagram below displays the User interface of a Github repository which includes Google Colaboratory notebook, BF_encoding.py file, Final reflective report, README and User manual. We will further explain the function for each of the buttons and indicators.




**Title :** Shows the file name of the product

**File browser :** Shows the files stored and allows users to navigate through different files

**Code Segments :** Allows users to navigate through different functions of the program

**Command panel :** Allows administrator and users to search for different functions and commands of the program itself and acts as a helper.

**Code Block :** The grouping of source code according to a lexical structure.

**Code executer :** Allows the administrator and user to check RAM and Disk memory used to run the program.

**Terminal :** Allows administrators and users to type commands, manipulate files, execute programs, and open documents.

### 2.1 Pre-installation to run the prototype for *Administrator*

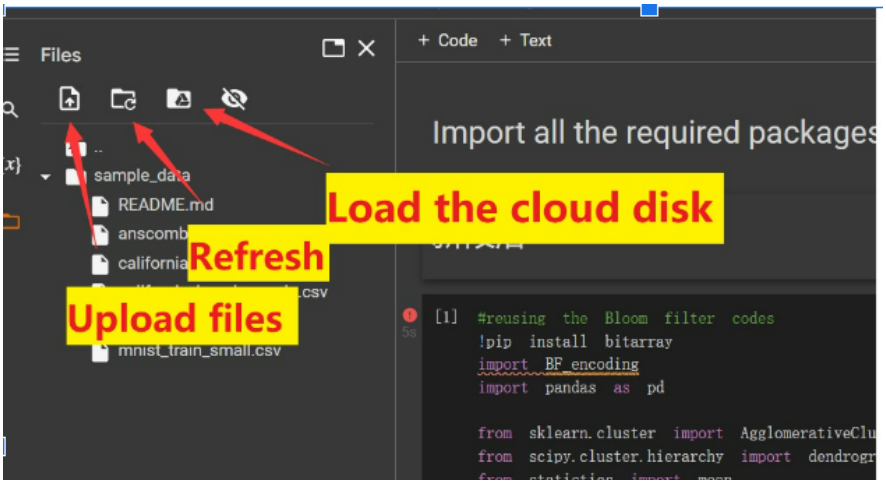
Before running this program, we have to first download the program provided with the file name of "Dynamic_Data_linkage.ipynb". Furthermore, we will be needing to import datasets for the program in order to make it funcional. The data that we were given are synthetic data. Therefore, for real life implementation the admin should import the actual database with the file extension of .csv stored in a folder that has been unzipped. For this scenario we have used the datasets that were given as an example. The steps shown below will guide you through the installation process.

Step 1:
Go to Our GitHub Repository with the link provided below
https://github.com/Jinky17/G26_Dynamic_Data_Linkage

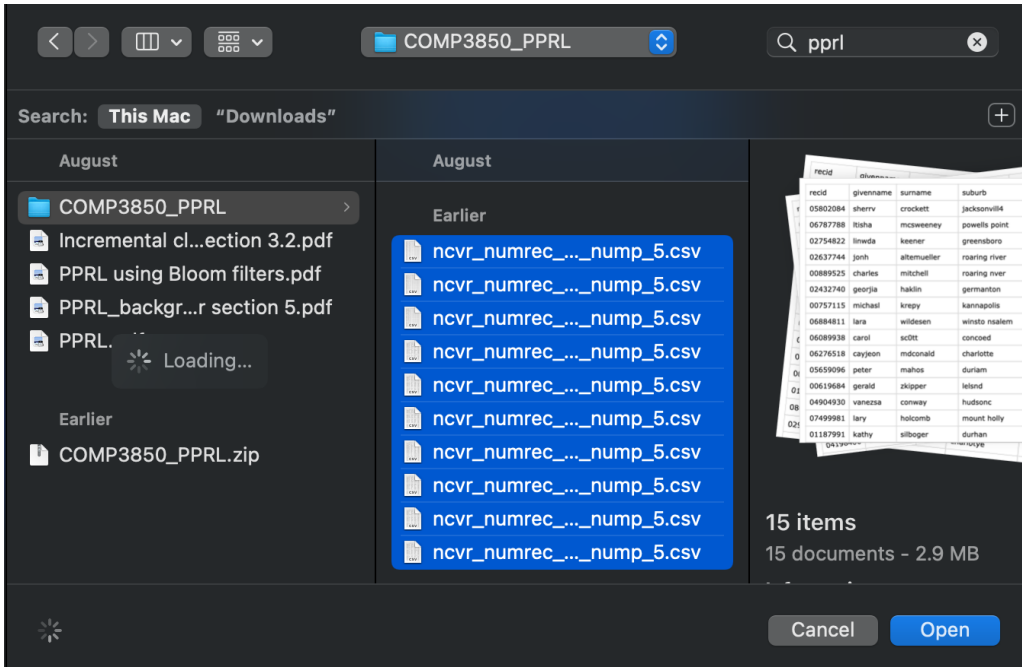
Step 2:
Download all of the files from the repository

Step 3 :
Navigate to Google Colaboratory with the link below
https://colab.research.google.com/

Step 4 :
Select "Upload files" and upload all files downloaded from the GitHub repository as shown below


Files
+ Code
+ Text
Import all the required packages
sample_data
README.md
anscomb
california
mnist_train_small.csv
Load the cloud disk
Refresh
Upload files
[1] #reusing the Bloom filter codes
!pip install bitarray
import BF_encoding
import pandas as pd
from sklearn.cluster import AgglomerativeClu
from scipy.cluster.hierarchy import dendrogr

Step 5 : Upload the datasets as well. Make sure the folder has been unzipped prior.



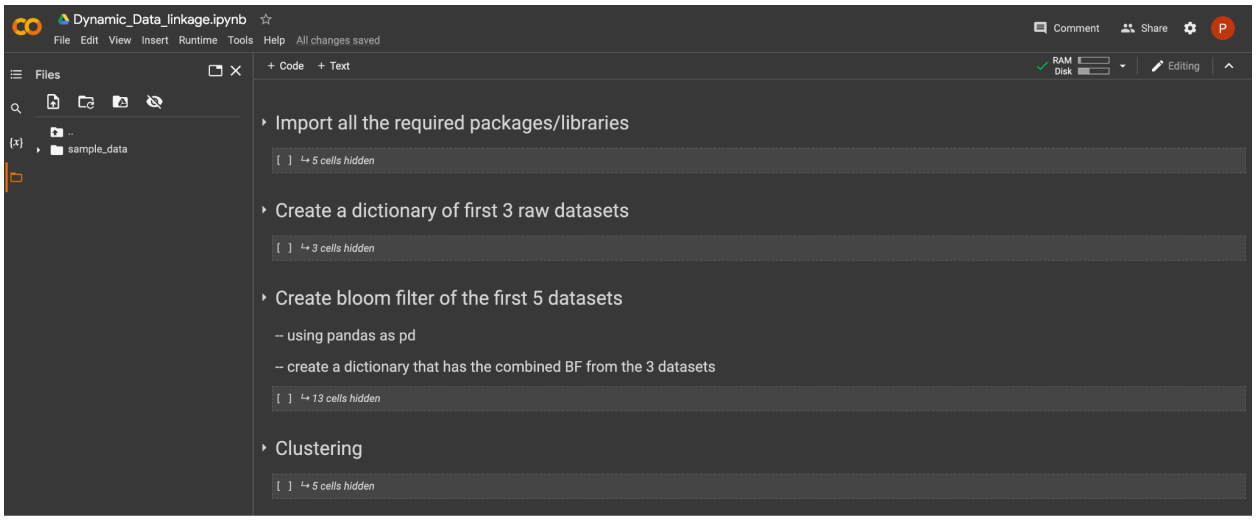

Step 6 :

After uploading the required data and program to Google Colaboratory, the uploaded files should be visible on the Google drive.

[photo]

Step 7 :

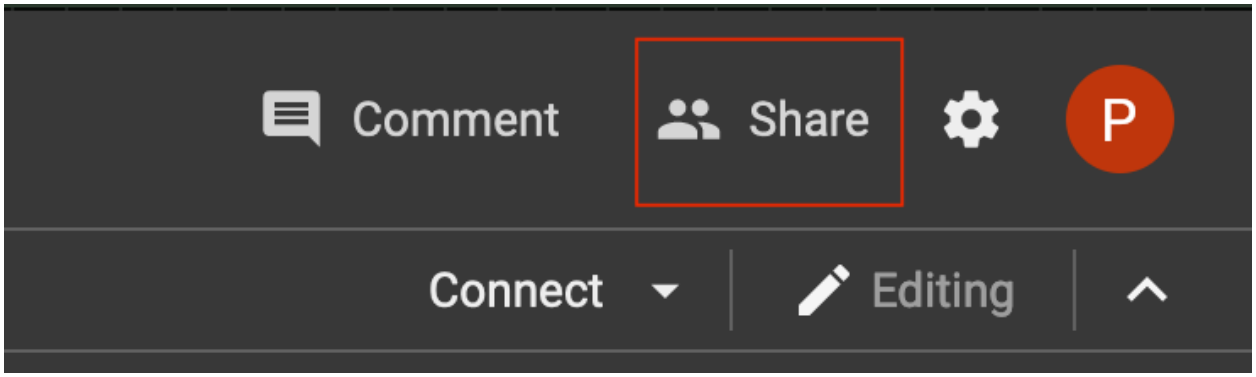
After uploading the required files, open the program with the file name of "Dynamic_Data_linkage.ipynb". The same layout as shown as the Figure below should be displayed.



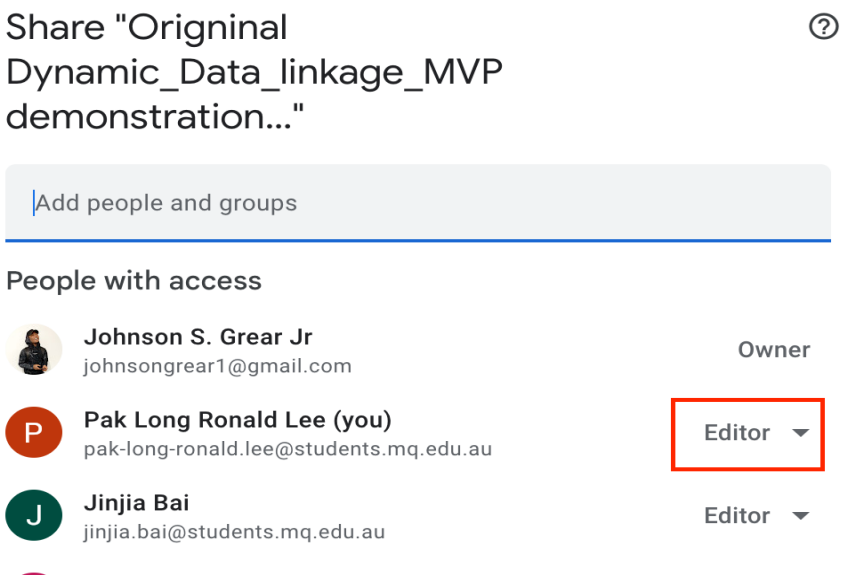

## 2.2 Configurations and settings for Administrator

After completing the steps above, we are almost ready to run the program. But beforehand, there are settings that need to be configured. Since the datasets contain sensitive information such as personal information and addresses, it is important to implement access privileges for the google collaboration folder. This gives rights and access to a specific Authorized User account (e.g., read-only access, write access) which could be done by the steps shown below in order to prevent cyber security threats and vulnerabilities.

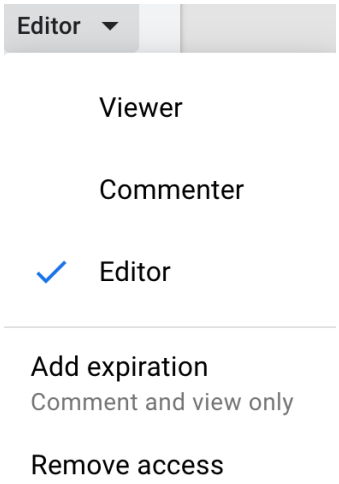
Step 1 : Press the “Share” button on the top right corner



Step 2: Under the tab on the right hand side we can modify the access privileges for each of the end-users and add end-users with their Google email address in this tqab.



Step 3 : From this tab we can edit the permissions given to each of the users



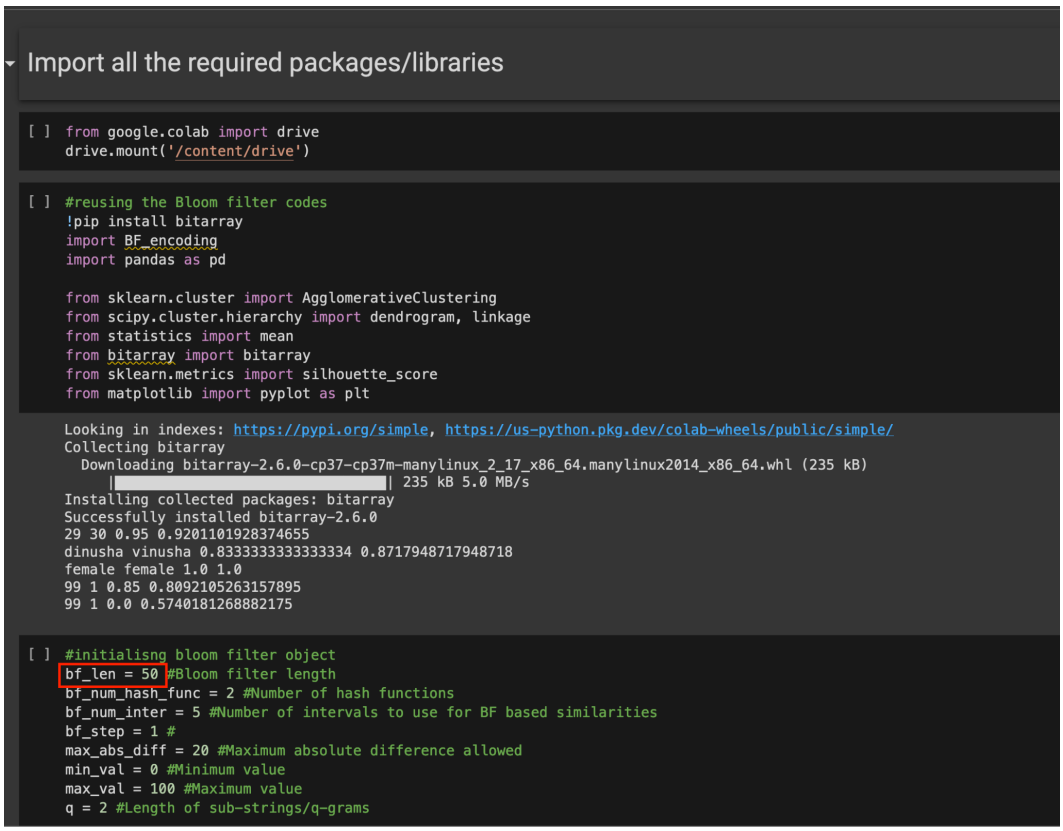

### 2.3 Configuration for the Bloom filter

After configuring the access privileges and adding end-users to the Google Colaboratory. The administrator has to configure the length of the bloom filter for the program. The default setting for the length of bloom filter is set as “50” and the administrator would be able to change. Please note that the length of the bloom filter is proportional to the number of elements being filtered and if the required number of hash functions is higher, the more secure the data would be by turning the raw records into more complex hashes and the runtime of the program will increase at the same time due to more calculations needed. Vise versa, a lower number of the length of the bloom filter will give us a lower security level of encryption but a shorter runtime to calculate..

With this in mind, the length of the bloom filters would affect the level of encryption of the program. The suggested length is “50” which is the default setting and has an optimal level of encryption and acceptable runtime duration. The steps below will guide you through the configuration process for the length of the bloom filter.

**Step 1:**

Under the first section of our program“Dynamic_Data_linkage.ipynb” named “Import all the required packages/libraries. We would be able to notice that there is a variable named “ bf_len” which indicates the length of the bloom filters. Administrators would be able to change it to a minimum number of 1 and there is no limit for the length. The default length is “50”.

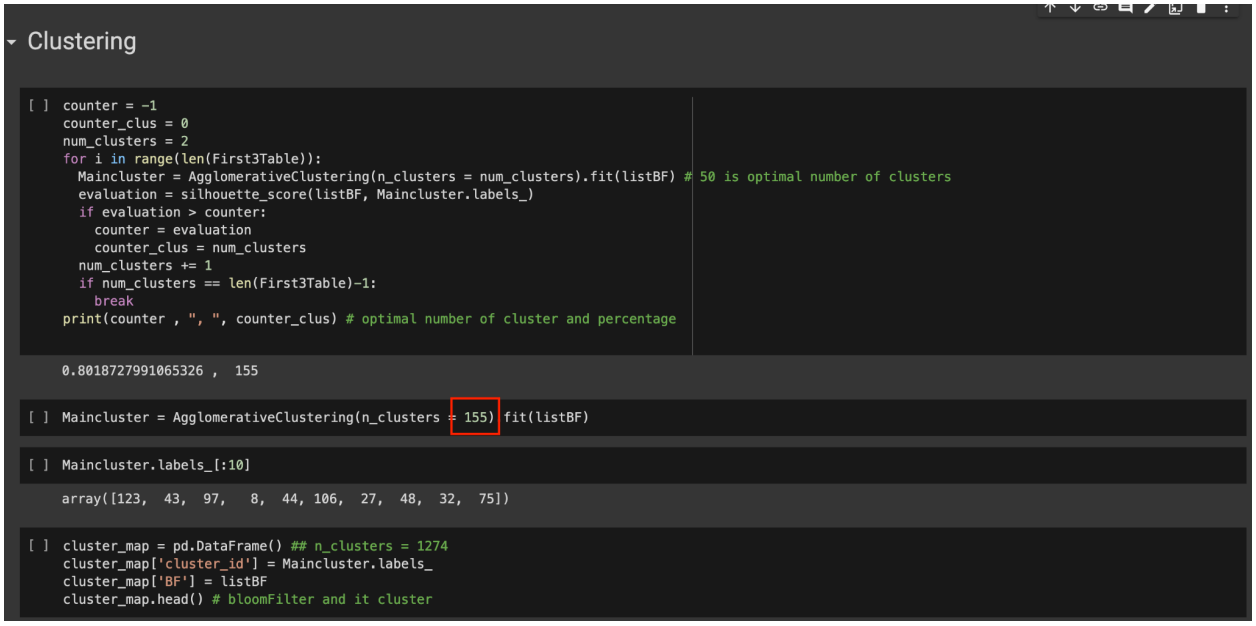


### 2.4 Configuration for number of clusters

From here on, the administrator would have to configure the number of clusters that the program will utilize. The purpose of configuring the number of clusters is important. To determine the number of clusters it varies by the numbers of records within the datasets. To find the number of the optimal number of clusters. The elbow method could be used, in which we have to gather the variance as a function of the number of clusters. The previous number of clusters added to another cluster doesn't give much better modeling of the data is the optimal one. The steps below will guide you through the configuration process for configuring the number of clusters.

Step 1 :

Navigate to the clustering section of the program “Dynamic_Data_linkage.ipynb”. Then expand the section. Under the clustering section you would be able to notice there is a variable name “n_clusters” which stands for number of clusters. The optimal number of clusters for the synthetic datasets are “155”. Please note that the number varies for different datasets imported into the program.

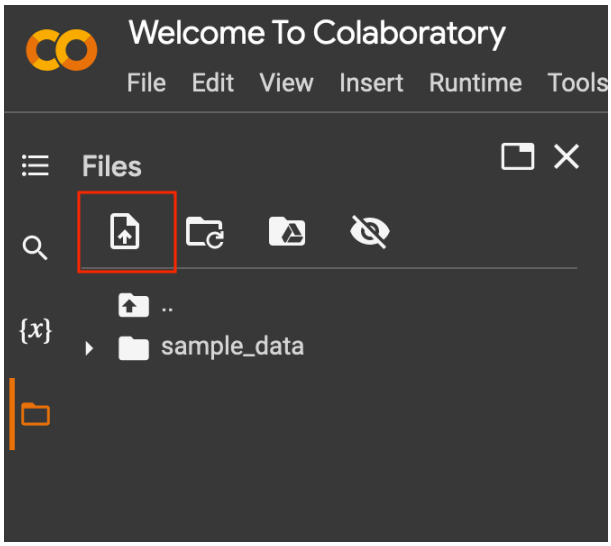


### 3.1 Configurations and settings for End-Users

The end-user of our program will be a public facility such as a hospital or a clinic. The program would allow organizations or companies to interchange information and data which databases have to be connected with each other in order to form a network. For end users, the basic usage steps are pretty simple as the configuration process has been completed by the administrator. To start with, end-users will be focusing on updating their database i.e Dynamic Update function. Whenever the end users get a new database that needs to be updated and uploaded to the database network, end users have to upload the newest file to the Google Colaboratory as shown. For demonstration, the fourth and fifth databases are used as dynamic data to be added in and steps are as follows.

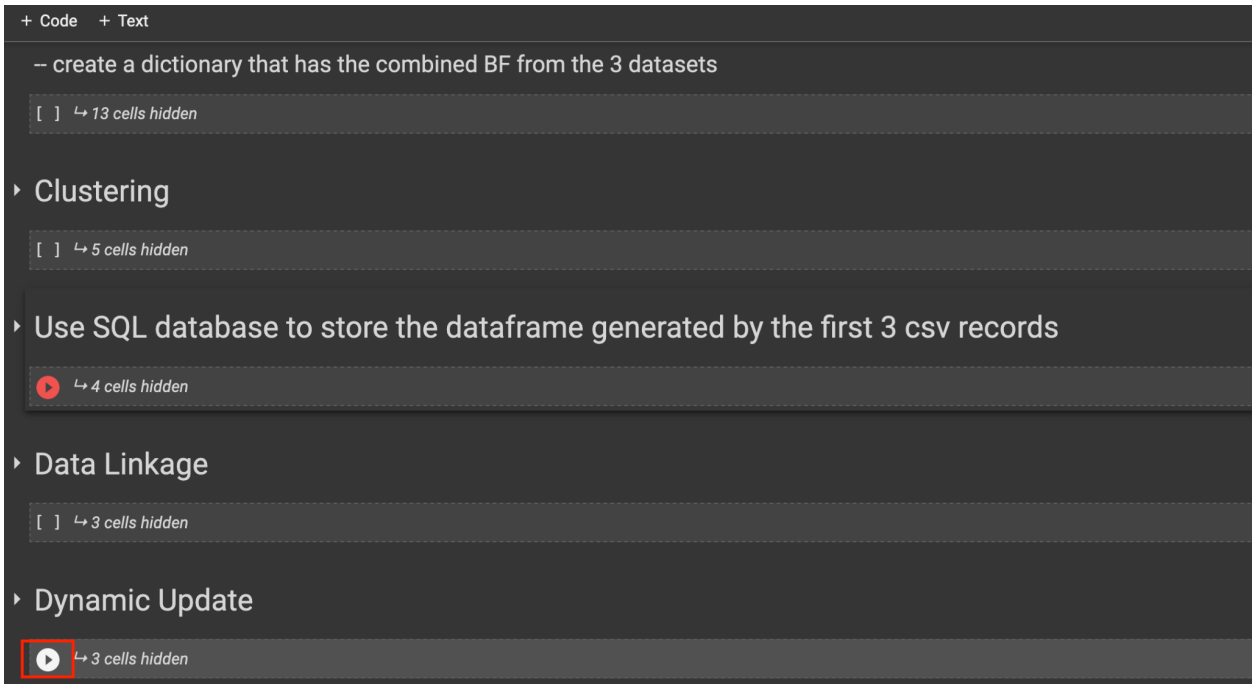
**Step 1 :**

Upload the latest files/database onto Google Colaboratory as shown below using the upload button.



**Step 2 :**

Navigate to the Dynamic Update Section and press the play button, and the program will automatically read the newest file and go through the records within the files imported and added to the existing database.




To illustrate, our product will automatically cluster all the databases (new and existing) and generate the latest results automatically.

here

4.1 Operating the software for end-users

After configuring all of the settings and variables from the steps above, we are now ready to run the software itself. It is important to import the datasets provided from the GitHub repository in order to verify configurations and settings have been done correctly and match the output that will be generated and provided as follows.

Step 1 :

LEAVE ME ALONE :D

走開 謝謝 :D

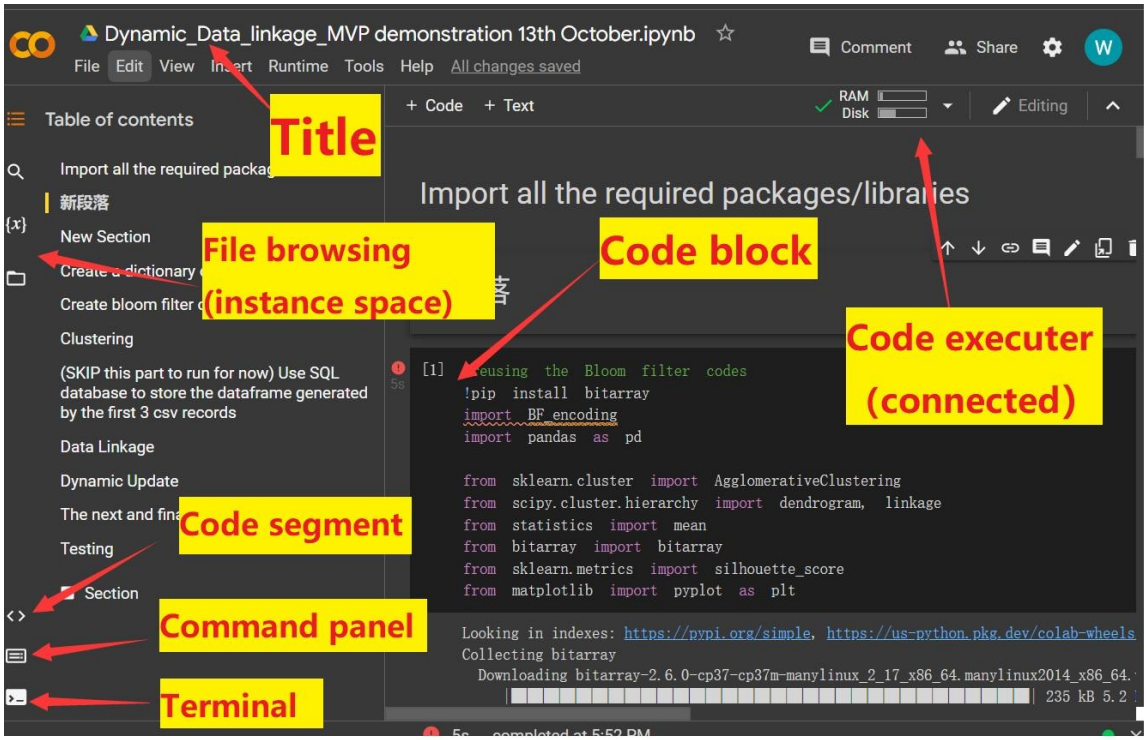
**2.4 Configuration troubleshooting for administrator**

In this section, we have prepared a troubleshooting guide for admin users in case any possible errors occur during the process of running the product developed by Group 26.

Here are the steps for troubleshooting:

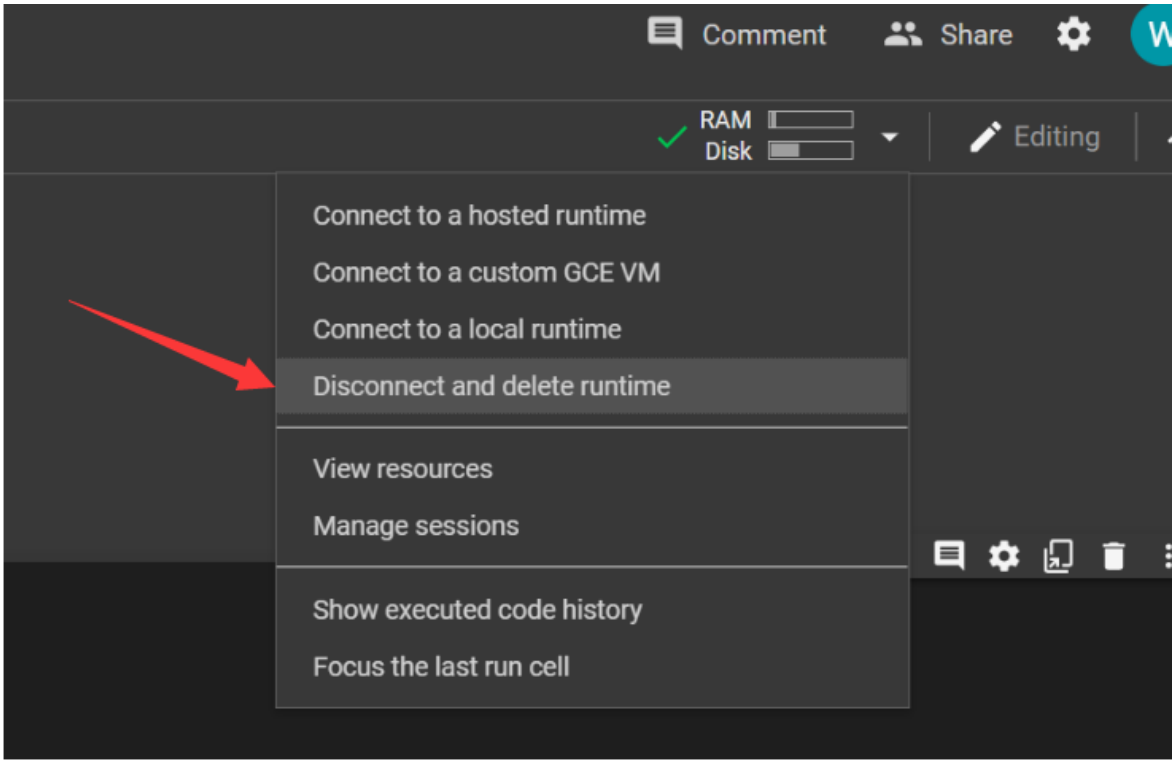
**Scenario 1:**

The prototype was built with Google Colaboratory , the platform offers a cloud environment. In simple terms, users will execute the code from the cloud. Then output will then return to the users' computer.


Dynamic_Data_linkage_MVP demonstration 13th October.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
W
Table of contents
+ Code + Text
RAM
Disk
Editing
Import all the required packa
新段落
New Section
Create a dictionary
Create bloom filter
Clustering
(SKIP this part to run for now) Use SQL database to store the dataframe generated by the first 3 csv records
Data Linkage
Dynamic Update
The next and fina
Testing
Section
Import all the required packages/libraries
[1] #reusing the Bloom filter codes
!pip install bitarray
import BF_encoding
import pandas as pd
from sklearn.cluster import AgglomerativeClustering
from scipy.cluster.hierarchy import dendrogram, linkage
from statistics import mean
from bitarray import bitarray
from sklearn.metrics import silhouette_score
from matplotlib import pyplot as plt
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels
Collecting bitarray
Downloading bitarray-2.6.0-cp37-cp37m-manylinux_2_17_x86_64.manylinux2014_x86_64.
235 kB 5.2
Title
File browsing (instance space)
Code block
Code executer (connected)
Code segment
Command panel
Terminal

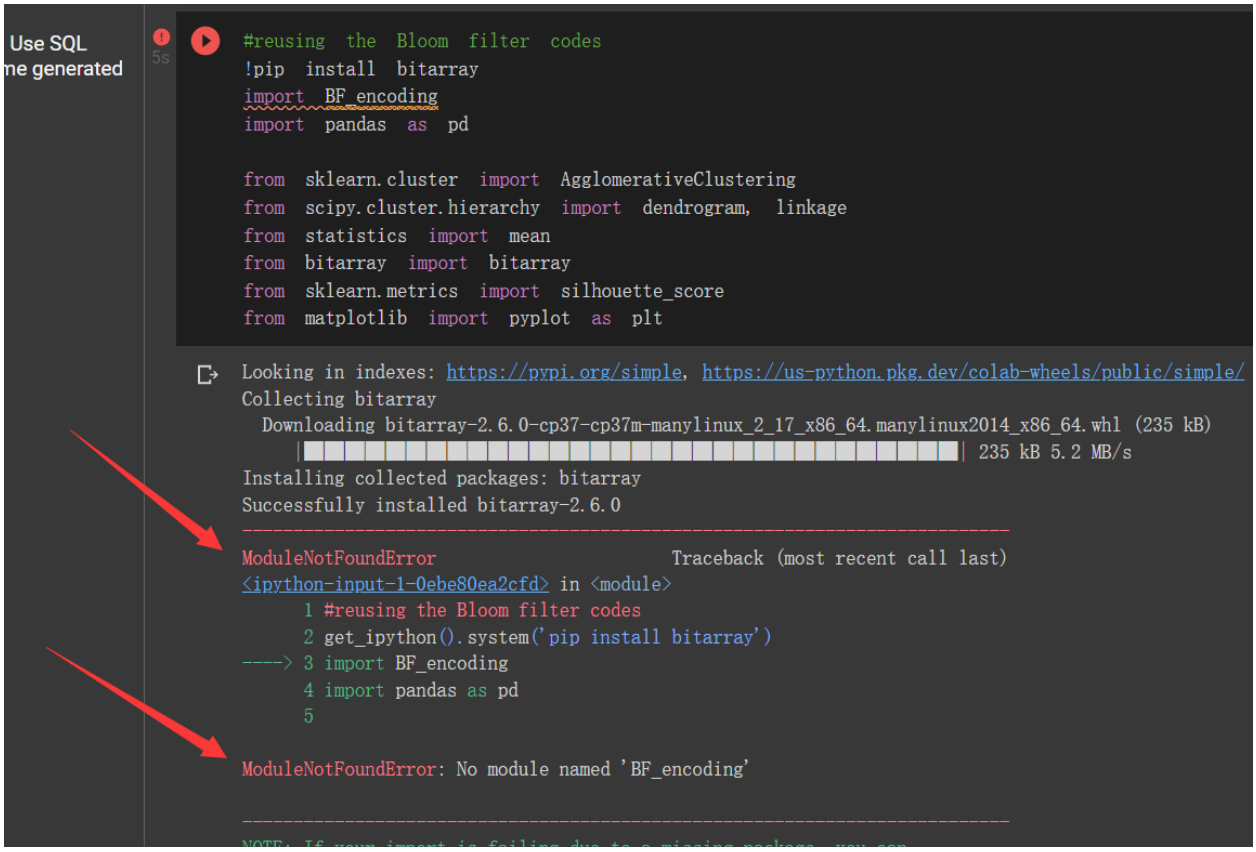
**Scenario 2:**

As Colab executes the code in the cloud server, and sends the results back to the browser. Since that, once it is not running in the proper order or some resources are not responding to the link, users should click “Disconnect and delete runtime” in the top right corner, then wait for the program to relink to the resources stored in the cloud. As shown below.



**Scenario 3:**

When there is a runtime error named ”ModuleNotFoundError” it is unfortunate to happen as shown below. Since this operating environment is a cloud environment, the required files need to be re-uploaded every time when this page is opened on the browser. Otherwise, you will see thai error




When this error happens, users should open the instance space in the top left.

[photo]

Please drag “BF_Encoding” into the instance space, under the “sample.data”.