CDM_CW2_G2

December 14, 2022

This project addresses the problem below:

Help the CEO of iInsureU123 to anonymise the data and calculate the k-anonymity of the dataset, so that she can share it with the researchers at Imperial and collaborators in the government in a secure and appropriate manner.

The CEO of an insurance company, iInsureU123, wants to understand if she can increase the policy fee for customers with a particular gene variant - the gene DRD4, which is known as the Wanderlust gene. Her hypothesis is that customers with this gene variant travel more and consequently are at greater risk. She has asked some of her former colleagues at Imperial College to help her with this research project.

The government wants to understand if people with this Wanderlust gene have anything in common from an educational or geographical perspective. The data will be made available online for anyone in the public domain to access it. She is not helping the government with their analysis. She is just sharing the data as part of her collaboration contract.

This notebook has five sections:

1. Preparations
2. Dataset for researchers at Imperial
3. Dataset for government collaborators
4. k-anonymity
5. Export data

1 Preparations

1.1 Load packages

First, we import the required packages for anonymisation.

[1]:
```
import pandas as pd
import numpy as np
import random
import pycountry_convert as pc
import json
import secrets
```

1.2 Load data

Then, import data in its raw format from the csv file as a pandas DataFrame.

[2]:
```
# define path
PATH = '../Data/customer_information.csv'

# read csv as dataframe
df = pd.read_csv(PATH)

# explore data
print(df.shape)
df.head()
```

```
(1000, 18)
```

[2]:
```
  given_name   surname gender   birthdate           country_of_birth  \
0   Lorraine      Reed      F  1984-07-05                    Armenia
1     Edward  Williams      M  1997-06-17  Northern Mariana Islands
2     Hannah    Turner      F  1990-06-15                  Venezuela
3  Christine   Osborne      F  2000-07-29                    Eritrea
4  Francesca     Yates      F  1968-11-04                    Ecuador

  current_country     phone_number postcode national_insurance_number  \
0  United Kingdom   (07700) 900876  LS5 8FN             ZZ 19 48 92 T
1  United Kingdom  (07700) 900 877  M0U 1RA               ZZ 753513 T
2  United Kingdom   +447700 900148  S01 8HZ               ZZ 947196 T
3  United Kingdom   +447700 900112  B18 8LW             ZZ 39 69 47 T
4  United Kingdom    07700 900 413  TQ2 6BE             ZZ 30 98 91 T

   bank_account_number  cc_status  weight  height blood_group  \
0             51157818          0    74.2    1.73          B+
1            103328715          0    69.4    1.74          O-
2             69342327          0    98.6    1.88          B+
3             85159170          0    62.0    1.56          O+
4             11399166          0    96.3    1.81          A-

   avg_n_drinks_per_week  avg_n_cigret_per_week education_level  \
0                    6.5                  218.8             phD
1                    0.7                   43.6         primary
2                    7.8                   59.1        bachelor
3                    4.6                  284.2         primary
4                    4.4                  348.8       secondary

   n_countries_visited
0                   48
1                   42
2                    9
```

```
3                   32
4                   34
```

1.3 Sample IDs

A list of unique random numbers is generated as `sid` that distinctively identify each subject in the dataset.

[3]:
```
# set seed
random.seed(23579)

# list of 1000 random 7-digit integers
sid = random.sample(range(1000000, 10000000), 1000)

# attach sample IDs to dataset
df.insert(0, 'sid', sid)

df.head()
```

[3]:
```
       sid given_name   surname gender   birthdate          country_of_birth  \
0  7753338   Lorraine      Reed      F  1984-07-05                   Armenia
1  3293855     Edward  Williams      M  1997-06-17  Northern Mariana Islands
2  3988568     Hannah    Turner      F  1990-06-15                 Venezuela
3  3406793  Christine   Osborne      F  2000-07-29                   Eritrea
4  2729863  Francesca     Yates      F  1968-11-04                   Ecuador

  current_country     phone_number postcode national_insurance_number  \
0  United Kingdom   (07700) 900876  LS5 8FN            ZZ 19 48 92 T
1  United Kingdom  (07700) 900 877  MOU 1RA              ZZ 753513 T
2  United Kingdom   +447700 900148  SO1 8HZ              ZZ 947196 T
3  United Kingdom   +447700 900112  B18 8LW            ZZ 39 69 47 T
4  United Kingdom    07700 900 413  TQ2 6BE            ZZ 30 98 91 T

   bank_account_number  cc_status  weight  height blood_group  \
0             51157818          0    74.2    1.73          B+
1            103328715          0    69.4    1.74          O-
2             69342327          0    98.6    1.88          B+
3             85159170          0    62.0    1.56          O+
4             11399166          0    96.3    1.81          A-

   avg_n_drinks_per_week  avg_n_cigret_per_week education_level  \
0                    6.5                  218.8             phD
1                    0.7                   43.6         primary
2                    7.8                   59.1        bachelor
3                    4.6                  284.2         primary
4                    4.4                  348.8       secondary
```

```
   n_countries_visited
0                   48
1                   42
2                    9
3                   32
4                   34
```

1.4 Direct identifiers

A dataset with `sid` and direct identifiers that explicitly identify a person is created.

[4]:
```python
df_di = df[['sid', 'given_name', 'surname', 'phone_number',␣
 ↪'national_insurance_number', 'bank_account_number']]
```

2 Dataset for researchers at Imperial

We create a dataframe without direct identifiers and carry out anonymisation based on the column order.

[5]:
```python
df_res = df.drop(columns = ['given_name', 'surname', 'phone_number',␣
 ↪'national_insurance_number', 'bank_account_number'])
df_res.head()
```

[5]:
```
       sid gender   birthdate          country_of_birth current_country  \
0  7753338      F  1984-07-05                   Armenia  United Kingdom
1  3293855      M  1997-06-17  Northern Mariana Islands  United Kingdom
2  3988568      F  1990-06-15                 Venezuela  United Kingdom
3  3406793      F  2000-07-29                   Eritrea  United Kingdom
4  2729863      F  1968-11-04                   Ecuador  United Kingdom

  postcode  cc_status  weight  height blood_group  avg_n_drinks_per_week  \
0  LS5 8FN          0    74.2    1.73          B+                    6.5
1  MOU 1RA          0    69.4    1.74          O-                    0.7
2  SO1 8HZ          0    98.6    1.88          B+                    7.8
3  B18 8LW          0    62.0    1.56          O+                    4.6
4  TQ2 6BE          0    96.3    1.81          A-                    4.4

   avg_n_cigret_per_week education_level  n_countries_visited
0                  218.8             phD                   48
1                   43.6         primary                   42
2                   59.1        bachelor                    9
3                  284.2         primary                   32
4                  348.8       secondary                   34
```

2.1 Dictionary

We generate a dictionary which includes the instructions on how to access the dataset and the coding information to decode the attributes.

```
[6]: imp_info = {'instructions': {'Access data file': 'The data file is password␣
     ↪protected, the password can be found in the password.txt file',
                                  'Access original values of attributes':␣
     ↪{'Categorical variables': 'Coding information can be found in this file',
                                                                              ␣
     ↪'Continuous variables': 'Standardised: Mean and standard deviations can be␣
     ↪found in this file to reverse to original values'}
                                  }}
```

2.2 Gender

We convert `gender` into **binary codes** so that researchers can distinguish between genders for analysis but attackers of this dataset would not know whether they are male or female unless they obtain the code information as well.

```
[7]: df_res['gender'] = np.where(df_res['gender'] == 'M', 1, 0)

     # store code info in dictionary
     imp_info['gender'] = {'male': 1, 'female': 0}
```

2.3 Birthdate

`birthdate` is converted to `age` in years at 2022. We divide it into **4 bands** using the quantile-based discretization `qcut()` function.

```
[8]: # retrieve year of birth as int
     birthyear = pd.to_datetime(df_res['birthdate']).dt.year

     # subtract to get age and divide by quartiles
     age = pd.qcut(2022 - birthyear, 4, labels = ['18-32', '33-43', '44-55', '55+'])

     # insert to df
     df_res.insert(2, 'age', age)

     # remove original column
     df_res = df_res.drop(columns = ['birthdate'])
```

2.4 Country of birth

As `country_of_birth` is a quasi-identifier with many unique values, it has to be converted to **continent** (more general grouping) to meet k-anonymity. Nonetheless, continent information is not meaningful for this study, so we **remove** the column.

```
[9]: # descriptive statistics
     df_res['country_of_birth'].describe()
```

```
[9]: count       1000
     unique       237
```

```
top       Korea
freq         12
Name: country_of_birth, dtype: object
```

[10]:
```
df_res = df_res.drop(columns = 'country_of_birth')
```

2.5 Current country

Since all subjects are currently living in the UK, we **remove** the entire column.

[11]:
```
df_res = df_res.drop(columns = 'current_country')
```

2.6 Postcode

`postcode` is **removed** as it is a quasi-identifier that is not required for this analysis.

[12]:
```
df_res = df_res.drop(columns = 'postcode')
```

2.7 cc status

Since `cc_status` is the **exposure** of our study, we **keep** it in its current format.

2.8 Weight and height

`weight` and `height` are **standardised as Z-score** values. The actual means and standard deviations are stored separately in the dictionary for added layer of security.

[13]:
```
# define standardisation function
def std(x):
    mean = x.mean()
    sd = x.std()
    Z = (x-mean)/sd
    out = [Z, {'mean': mean, 'sd': sd}]
    return out
```

[14]:
```
# apply to weight and height
w = std(df_res['weight'])
h = std(df_res['height'])

# add z-score columns to df
df_res.insert(4, 'weight_std', w[0])
df_res.insert(5, 'height_std', h[0])

# store mean-sd info in dictionary
imp_info['weight'] = w[1]
imp_info['height'] = h[1]

# remove original columns
df_res = df_res.drop(columns = ['weight', 'height'])
```

2.9 Blood group

We **pseudonymise** `blood_group` using alphabet letters so that researchers can distinguish between blood types using coding information provided but attackers of this dataset would not know the exact blood types without obtaining the coding file.

[15]:
```
# Assign unique letter to each group
bg_code = {'B+': 'a', 'O-': 'b', 'O+': 'c', 'A-': 'd', 'A+': 'e', 'AB+': 'f',␣
↪'B-': 'g', 'AB-': 'h'}

# overwrite with codes
df_res['blood_group'] = df_res['blood_group'].replace(bg_code)

# store code info in dictionary
imp_info['blood_group'] = bg_code
```

2.10 Average number of drinks and cigarets per week

Similar to *Section 2.8*, we **standardise** `avg_n_drinks_per_week` and `avg_n_cigret_per_week` then store the actual means and standard deviations in the dictionary.

[16]:
```
# apply standardisation function
d = std(df_res['avg_n_drinks_per_week'])
c = std(df_res['avg_n_cigret_per_week'])

# add z-score columns to df
df_res.insert(7, 'avg_n_drinks_per_week_std', d[0])
df_res.insert(8, 'avg_n_cigret_per_week_std', c[0])

# store mean-sd info in dictionary
imp_info['avg_n_drinks_per_week'] = d[1]
imp_info['avg_n_cigret_per_week'] = c[1]

# remove original columns
df_res = df_res.drop(columns = ['avg_n_drinks_per_week',␣
↪'avg_n_cigret_per_week'])
```

2.11 Education level

Education level is first banded to 3 categories and then, like in *Section 2.9*, **pseudonymised** using alphabet letters so that researchers can distinguish between educational levels but attackers with only the alphabet codes could not re-identify our subjects.

[17]:
```
# banding into 3 groups
df_res['education_level'] = df_res['education_level'].replace({'primary':␣
↪'school',
                                                                'secondary':␣
↪'school',
```

```
                                                                    'bachelor':␣
  ↪'college',
                                                                    'masters':␣
  ↪'college',
                                                                    'phD':␣
  ↪'college'})
# distinct letter for each group
el_code = {'college': 'a', 'school': 'b', 'other': 'c'}

# replace with codes
df_res['education_level'] = df_res['education_level'].replace(el_code)

# store code info in dictionary
imp_info['education_level'] = el_code
```

2.12 Number of countries visited

Similar to *Section 2.8*, we **standardise** `n_countries_visited` and store the actual mean and standard deviation in the dictionary.

[18]:
```
# apply standardisation function
ncv = std(df_res['n_countries_visited'])

# add z-score columns to df
df_res.insert(9, 'n_countries_visited_std', ncv[0])

# store mean-sd info in dictionary
imp_info['n_countries_visited'] = ncv[1]

# remove original columns
df_res = df_res.drop(columns = ['n_countries_visited'])
```

2.13 Dataframe for researchers

[19]:
```
df_res
```

[19]:
```
         sid  gender    age  cc_status  weight_std  height_std blood_group  \
0    7753338       0  33-43          0    0.370074    0.188026           a
1    3293855       1  18-32          0    0.118030    0.245055           b
2    3988568       0  18-32          0    1.651298    1.043466           a
3    3406793       0  18-32          0   -0.270538   -0.781473           c
4    2729863       0  44-55          0    1.530527    0.644261           d
..       …       …     …         …          …          …
995  7106972       1    55+          0    1.341494    1.613760           e
996  8081229       1  18-32          0   -0.580342    0.872378           a
997  1922060       0    55+          0    1.457014    1.727818           c
998  5206922       0    55+          0    0.443587   -1.123649           e
```

```
999  2088937       1    55+          0    1.493771   -0.268209           g

     avg_n_drinks_per_week_std  avg_n_cigret_per_week_std  \
0                     0.602188                  -0.170974
1                    -1.411819                  -1.367653
2                     1.053604                  -1.261783
3                    -0.057573                   0.275732
4                    -0.127021                   0.716974
..                         …                        …
995                  -1.029852                   0.126830
996                   1.018879                   0.630911
997                  -1.342371                  -1.285006
998                  -0.022849                   1.275014
999                  -1.411819                  -1.429126

     n_countries_visited_std education_level
0                   1.608008               a
1                   1.167840               b
2                  -1.253086               a
3                   0.434226               b
4                   0.580949               b
..                       …             …
995                -0.372749               b
996                 0.654310               c
997                 0.654310               b
998                 0.654310               a
999                 1.534647               a

[1000 rows x 11 columns]
```

3 Dataset for government collaborators

To suit the government's needs of finding common educational and geographical features and publicising the data, only `sid`, `country_of_birth`, `postcode`, `cc_status` and `education_level` are kept. We extract these columns to a new dataframe for anonymisation using `.copy()` to prevent overwriting the origingal dataset.

[20]:
```python
df_gov = df[['sid', 'country_of_birth', 'postcode', 'cc_status',
 'education_level']].copy()
df_gov.head()
```

[20]:
```
       sid          country_of_birth postcode  cc_status education_level
0  7753338                   Armenia  LS5 8FN          0             phD
1  3293855  Northern Mariana Islands  MOU 1RA          0         primary
2  3988568                 Venezuela  SO1 8HZ          0        bachelor
3  3406793                   Eritrea  B18 8LW          0         primary
4  2729863                   Ecuador  TQ2 6BE          0       secondary
```

3.1 Country of birth

As mentioned in *Section 2.3*, country_of_birth is converted to continent_of_birth as a **more general classification** to prevent re-identification.

[21]:
```
# descriptive statistics
df_gov['country_of_birth'].describe()
```

[21]:
```
count      1000
unique      237
top       Korea
freq         12
Name: country_of_birth, dtype: object
```

[22]:
```
# count for each country
country_n = df_gov.groupby(['country_of_birth']).size().reset_index(name =␣
 ↪'count')
print(country_n)
```

```
      country_of_birth  count
0          Afghanistan      2
1              Albania      3
2              Algeria      6
3       American Samoa      4
4              Andorra      3
..                 …      …
232  Wallis and Futuna      4
233     Western Sahara      6
234              Yemen      4
235             Zambia      2
236           Zimbabwe      5

[237 rows x 2 columns]
```

[23]:
```
# define conversion function
def country_to_continent(country_name):
    if country_name in ['Korea', 'Palestinian Territory', 'Timor-Leste']:
        return 'Asia'
    elif country_name in ['Saint Barthelemy','United States Minor Outlying␣
 ↪Islands']:
        return 'North America'
    elif country_name in ['Saint Helena', 'Reunion', 'Western Sahara', 'Libyan␣
 ↪Arab Jamahiriya', "Cote d'Ivoire"]:
        return 'Africa'
    elif country_name in ['Antarctica (the territory South of 60 deg S)']:
        return 'Antarctica'
    elif country_name == 'Pitcairn Islands':
        return 'Oceania'
```

```
    elif country_name in ['Slovakia (Slovak Republic)', 'Holy See (Vatican City␣
↪State)', 'British Indian Ocean Territory (Chagos Archipelago)', 'Bouvet␣
↪Island (Bouvetoya)', 'Svalbard & Jan Mayen Islands']:
        return 'Europe'
    elif country_name == 'Netherlands Antilles':
        return 'South America'
    else:
        country_alpha2 = pc.country_name_to_country_alpha2(country_name)
        country_continent_code = pc.
↪country_alpha2_to_continent_code(country_alpha2)
        country_continent_name = pc.
↪convert_continent_code_to_continent_name(country_continent_code)
        return country_continent_name

# apply to our data as new column
df_gov.insert(1, 'continent_of_birth', df_gov['country_of_birth'].
↪apply(country_to_continent))

# remove original column
df_gov = df_gov.drop(columns = 'country_of_birth')

# count for each continent
continent_n = df_gov.groupby(['continent_of_birth']).size().reset_index(name =␣
↪'count')
print(continent_n)
```

```
  continent_of_birth  count
0             Africa    225
1         Antarctica      7
2               Asia    204
3             Europe    234
4      North America    146
5            Oceania    117
6      South America     67
```

3.2 Postcode

`postcode` is divided into bands based on **outbound characters**. We then convert it to `UK_region` using a dictionary and combined the overseas territories after checking the counts in each category.

[24]:
```
# define function for finding index of first numerical digit
def find_first_digit(s):
    for i, c in enumerate(s):
        if c.isdigit():
            return i
            break
```

```
# keep characters before first digit
df_gov['postcode'] = df_gov['postcode'].apply(lambda x: x[:find_first_digit(x)])

# count for each area
area_n = df_gov.groupby(['postcode']).size().reset_index(name = 'count')
print(area_n)
```

```
    postcode  count
0         AB      4
1         AL      6
2          B     50
3         BA      4
4         BB      8
..       ...    ...
119       WR      6
120       WS      6
121       WV      3
122       YO      5
123       ZE      9

[124 rows x 2 columns]
```

[25]:
```
# dictionary for conversion
postcode_country = pd.read_csv('postcode_country.csv')
area_to_country = dict(zip(postcode_country['Postcode area'],␣
 ↪postcode_country['Country']))

# convert to UK country and add new column
df_gov.insert(2, 'UK_region', df_gov['postcode'].replace(area_to_country))

# remove original column
df_gov = df_gov.drop(columns = 'postcode')

# count for each UK country
ukcountry_n = df_gov.groupby(['UK_region']).size().reset_index(name = 'count')
print(ukcountry_n)
```

```
          UK_region  count
0   Channel Islands     11
1           England    833
2       Isle of Man      1
3  Northern Ireland      6
4          Scotland    125
5             Wales     24
```

[26]:
```
# combine overseas islands
df_gov['UK_country'] = df_gov['UK_region'].replace({'Channel Islands':␣
 ↪'Overseas territories',
```

```
                                                    'Isle of Man': 'Overseas␣
 ↪territories'})
# count for each category
ukcountry_n = df_gov.groupby(['UK_region']).size().reset_index(name = 'count')
print(ukcountry_n)
```

```
           UK_region  count
0    Channel Islands     11
1            England    833
2        Isle of Man      1
3   Northern Ireland      6
4           Scotland    125
5              Wales     24
```

3.3 cc status

Same as *Section 2.6*, `cc_status` is the **exposure** of the study and so is **kept unchanged**.

3.4 Education level

To reduce the risk of re-identification of subjects with unique values, some `education_level` **groups are combined** to give a broader classification.

[27]:
```
# count for each level
el_n = df_gov.groupby(['education_level']).size().reset_index(name = 'count')
print(el_n)
```

```
  education_level  count
0        bachelor    209
1         masters    112
2           other    108
3             phD     52
4         primary     61
5       secondary    458
```

[28]:
```
# combine categories
df_gov['education_level'] = df_gov['education_level'].replace({'primary':␣
 ↪'school',
                                                                'secondary':␣
 ↪'school',
                                                                'masters':␣
 ↪'postgraduate',
                                                                'phD':␣
 ↪'postgraduate',
                                                                'bachelor':␣
 ↪'undergraduate'})
```

3.5 Dataframe for government collaborators

[29]:
```
df_gov
```

[29]:
```
         sid continent_of_birth UK_region  cc_status education_level  \
0    7753338               Asia   England          0     postgraduate
1    3293855            Oceania   England          0           school
2    3988568      South America   England          0    undergraduate
3    3406793             Africa   England          0           school
4    2729863      South America   England          0           school
..       …                  …       …        …              …
995  7106972               Asia     Wales          0           school
996  8081229             Europe   England          0            other
997  1922060             Africa  Scotland          0           school
998  5206922             Europe   England          0    undergraduate
999  2088937      North America   England          0     postgraduate

    UK_country
0      England
1      England
2      England
3      England
4      England
..         …
995      Wales
996    England
997   Scotland
998    England
999    England

[1000 rows x 6 columns]
```

3.6 Instruction for using the government collaborator's dataset

[30]:
```
instruction1 = "1. The data file is password protected. The password can be␣
↪found in this file."
instruction2 = "2. Please remove 'sid' before publishing this dataset."
notes = "27 records had been removed from the original dataset for security␣
↪reasons."
```

4 k-anonymity

To be a k-anonymised dataset, every combination of quasi-identifier values must occur at least k times. In particular, **k should be at least 2** to ensure individuals cannot be uniquely identified using information of multiple quasi-identifers.

[31]:
```
# count for each combination
res = df_res.groupby(['gender', 'age', 'education_level']).size().
 ↪reset_index(name = 'count')
res_noel = df_res.groupby(['gender', 'age']).size().reset_index(name = 'count')
gov = df_gov.groupby(['continent_of_birth', 'UK_region', 'education_level']).
 ↪size().reset_index(name = 'count')

# k = minimum count
print('k_res:', min(res['count']),
      'k_res_noel:', min(res_noel['count']),
      'k_gov:', min(gov['count']))
```

```
k_res: 9 k_res_noel: 108 k_gov: 1
```

The `df_res` dataset is **9**-anonymous with `gender`, `age`, `education level` as quasi-identifiers, or **108**-anonymous if coding information for `education level` is not obtained. k of the `df_gov` dataset is **1**, which could be addressed by further generalisation of quasi-identifiers, but this would lead to further loss of information. Thus, in this case, records with unique combinations of quasi-identifier values are removed.

[32]:
```
# combinations that occur only once
gov_1 = gov.loc[gov['count'] == 1]
len(gov_1)
```

[32]:
```
27
```

[33]:
```
# left-join to get sid of records with unique combinations
gov_1 = pd.merge(gov_1,df_gov, how = 'left',
                 left_on = ['continent_of_birth', 'UK_region',␣
 ↪'education_level'],
                 right_on = ['continent_of_birth', 'UK_region',␣
 ↪'education_level'])

# remove records with unique combinations
df_gov = df_gov[df_gov['sid'].isin(gov_1['sid']) == False]

# check k-anonymity
gov = df_gov.groupby(['continent_of_birth', 'UK_region', 'education_level']).
 ↪size().reset_index(name = 'count')
print('k_gov:', min(gov['count']))
```

```
k_gov: 2
```

After removing the 27 records, the dataset is 2-anonymous.

[34]:
```
df_gov
```

[34]:
```
         sid continent_of_birth UK_region  cc_status education_level  \
0    7753338               Asia   England          0    postgraduate
```

```
1    3293855          Oceania   England          0          school
2    3988568    South America   England          0   undergraduate
3    3406793           Africa   England          0          school
4    2729863    South America   England          0          school
..       …                 …        …      …           …
995  7106972             Asia     Wales          0          school
996  8081229           Europe   England          0           other
997  1922060           Africa  Scotland          0          school
998  5206922           Europe   England          0   undergraduate
999  2088937    North America   England          0    postgraduate

    UK_country
0      England
1      England
2      England
3      England
4      England
..         …
995      Wales
996    England
997   Scotland
998    England
999    England

[973 rows x 6 columns]
```

5 Export data

The finalised datasets are saved as **.csv** along with the dictionary as **.json**. To add an extra layer of security, **random passwords** are generated and set.

[35]:
```
# direct identifiers
df_di.to_csv('direct_identifiers.csv', index = False)

# dataset for researchers with dictionary
df_res.to_csv('../Anonymised_data/Imperial_researchers/researchers_dataset.
 ↪csv', index = False)
with open('../Anonymised_data/Imperial_researchers/coding.json', 'w') as fp:
    json.dump(imp_info, fp, indent = 4)

# dataset for government collaobrators
df_gov.to_csv('../Anonymised_data/Government_collaborators/gov_dataset.csv',␣
 ↪index = False)
```

[36]:
```
# set 11-character passwords
password_r = secrets.token_urlsafe(11)
password_g = secrets.token_urlsafe(11)
```

```
with open('../Anonymised_data/Imperial_researchers/password.txt', 'w') as f:
    f.write(password_r)
with open('../Anonymised_data/Government_collaborators/README.txt', 'w') as f:
    f.write('Instructions:' + '\n' + instruction1 + '\n' + instruction2 + '\n' + '\n' +
            'Notes:' + '\n' + notes + '\n' + '\n' +
            'Password: ' + password_g)
```