anon_jupyter

December 14, 2022

1 Anonymising the customer_information.csv and calculating the k-anonymity of the new dataset

1.0.1 Importing necessary packages

The following packages must be imported (and also installed, if necessary).

[ ]:
```
import pandas as pd
import numpy as np
import hashlib
import re
import os
import country_converter as coco
from geopy.geocoders import Nominatim
from cryptography.fernet import Fernet
```

1.0.2 Helper functions used

The following helper functions are needed -

[ ]:
```
# Helper functions

# Parse country into shortform
def parse_country(country_name):
    country = coco.convert(country_name, to='name_short', include_obsolete=True)
    return country

# The following variable countries were hard-coded to fix unmatched territory␣
 ↪errors
northern_countries = ["Svalbard & Jan Mayen Islands"]
southern_countries = ["Micronesia"]

# Convert country of birth into Hemisphere (Northern or Southern) based on␣
 ↪latitude coordinates
def country_to_hemisphere(country_name):
    try:
        if country_name in southern_countries:
            return "Southern Hemisphere"
        elif country_name in northern_countries:
            return "Northern Hemisphere"
        else:
            return ("Southern" if Nominatim(user_agent="CDM").
 ↪geocode(parse_country(country_name)).latitude < 0 else "Northern") + "␣
 ↪Hemisphere"
    except Exception as e:
        print(e)
        return "Error"

# SHA hash function using a key and salt
def hash(to_hash, key):
    salt = os.urandom(16)
    h = hashlib.sha256()
    h.update(key)
    h.update(salt)
    h.update(to_hash.encode())
    return to_hash, h.hexdigest(), salt.hex()

# To encrypt and save as encrypted file; specify file to encrypt, encrypted␣
 ↪file destination, and destination key location
def encrypt(to_encrypt, file_destination, key_location):
    key = Fernet.generate_key() # AES in CBC mode with a 128-bit key for␣
 ↪encryption
    fernet = Fernet(key)

    with open(key_location, 'wb') as f:
        f.write(key)

    with open(to_encrypt, 'rb') as f:
        plaintext = f.read()

    encrypted = fernet.encrypt(plaintext)
    with open(file_destination, 'wb') as e:
        e.write(encrypted)
```

1.0.3 Loading required data and creating the anonymised dataframe

[ ]:

```
# Read in data to be anonymised
original_data = pd.read_csv("Data/customer_information.csv")

# Reading in postcode_region.csv to map given postcode to countries in the UK -␣
 ↪'England' and 'Other'(includes Wales, Scotland, Northern Ireland)
postcode_dictionary = pd.read_csv('Data/postcode_region.csv')
```

```
# Create anon_data variable as initial data with unneeded direct identifiers␣
 ↪dropped
anon_data = pd.DataFrame()

postcode_dictionary.head()
```

```
[ ]:   Postcode   Region
     0       AB    Other
     1       AL  England
     2        B  England
     3       BA  England
     4       BB  England
```

1.0.4 Adding variables to the anonymised dataset

Assigning gender and case-control status as given

```
[ ]: # Assign gender
     anon_data['Gender'] = original_data['gender']

     # Assign case-control status
     anon_data['CC.Status'] = original_data['cc_status']

     anon_data.head()
```

```
[ ]:   Gender  CC.Status
     0      F          0
     1      M          0
     2      F          0
     3      F          0
     4      F          0
```

1.0.5 Pseudoanonymisation

Creating the hashed Sample ID Next, a unique Sample ID is created from the National Insurance Number to link the anonymised data with the reference data containing sensitive information.

```
[ ]: # Clean NIN formatting and assign Sample ID as a hashed form of the NIN
     key = os.urandom(16)
     original_data["national_insurance_number"], anon_data['Sample.ID'], salts =␣
      ↪zip(*original_data["national_insurance_number"].apply(
        lambda x: hash(re.sub(r'(.{2})(?!$)','\\1 ', x.replace(' ', '') ), key)))

     anon_data.head()
```

```
[ ]:   Gender  CC.Status                                          Sample.ID
     0      F          0  f62e57d1aa65676b9db9986f73930551d196fa8db9eaa8…
```

```
1      M          0  40a8ffbc83728762c7f690a9537d536f3ba13ec3dacb57…
2      F          0  8d8e6615e9df3aaafb73eaf0a86b329bb7d57e95a4f833…
3      F          0  37035c2be594b97d187b5bec3aeec641ebe519fb374229…
4      F          0  e5ec0f5b36d61c09695a64ed63c509cec2f5b6c0251bd1…
```

```
[ ]: # Create a reference table between NIN and respective hashed NIN
     reference_table = pd.DataFrame()
     reference_table['Hashed.NIN'] = anon_data['Sample.ID']
     reference_table['Salt'] = salts
     reference_table['Key'] = key.hex()
     reference_table['NIN'] = original_data['national_insurance_number']

     reference_table.head()
```

```
[ ]:                                           Hashed.NIN  \
     0  f62e57d1aa65676b9db9986f73930551d196fa8db9eaa8…
     1  40a8ffbc83728762c7f690a9537d536f3ba13ec3dacb57…
     2  8d8e6615e9df3aaafb73eaf0a86b329bb7d57e95a4f833…
     3  37035c2be594b97d187b5bec3aeec641ebe519fb374229…
     4  e5ec0f5b36d61c09695a64ed63c509cec2f5b6c0251bd1…

                                    Salt                               Key  \
     0  7a2b96d32f333ab5b8c7fe0551045b47  1eadb7002887e4569806a37456023fd7
     1  2a8e3151f022d9b4fe57309c5985a8da  1eadb7002887e4569806a37456023fd7
     2  eb117cf898ca22607a0a035b41b6d11f  1eadb7002887e4569806a37456023fd7
     3  f780fe4cf0a955764333c5c2a8f7f7ae  1eadb7002887e4569806a37456023fd7
     4  72126f78bd0d3f62b80baef4b57da6b2  1eadb7002887e4569806a37456023fd7

                  NIN
     0  ZZ 19 48 92 T
     1  ZZ 75 35 13 T
     2  ZZ 94 71 96 T
     3  ZZ 39 69 47 T
     4  ZZ 30 98 91 T
```

1.0.6 Banding - date of birth and education level

```
[ ]: # Banding birth date
     birthyears = pd.DatetimeIndex(original_data['birthdate']).year

     # Band the birth years into 20-year intervals
     anon_data['Birthyear'] = pd.cut(birthyears, np.arange(birthyears.min(),␣
      ↪birthyears.max()+20, 20), right=False)

     anon_data.head()
```

```
[ ]:  Gender  CC.Status                                            Sample.ID  \
     0      F          0  f62e57d1aa65676b9db9986f73930551d196fa8db9eaa8…
     1      M          0  40a8ffbc83728762c7f690a9537d536f3ba13ec3dacb57…
     2      F          0  8d8e6615e9df3aaafb73eaf0a86b329bb7d57e95a4f833…
     3      F          0  37035c2be594b97d187b5bec3aeec641ebe519fb374229…
     4      F          0  e5ec0f5b36d61c09695a64ed63c509cec2f5b6c0251bd1…

           Birthyear
     0  [1975, 1995)
     1  [1995, 2015)
     2  [1975, 1995)
     3  [1995, 2015)
     4  [1955, 1975)
```

1.0.7 Mapping full postcode to countries within the UK using postcode_dictionary

```
[ ]: # Assign UK country derived from postcode
     anon_data['Postcode'] = original_data['postcode'].apply(lambda x: re.
      ↪search('[a-zA-Z]*', x).group(0))
     anon_data = pd.merge(anon_data, postcode_dictionary, on='Postcode', how='left')
     anon_data = anon_data.rename(columns={'Region': 'UK.Country'})

     anon_data.head()
```

```
[ ]:  Gender  CC.Status                                            Sample.ID  \
     0      F          0  f62e57d1aa65676b9db9986f73930551d196fa8db9eaa8…
     1      M          0  40a8ffbc83728762c7f690a9537d536f3ba13ec3dacb57…
     2      F          0  8d8e6615e9df3aaafb73eaf0a86b329bb7d57e95a4f833…
     3      F          0  37035c2be594b97d187b5bec3aeec641ebe519fb374229…
     4      F          0  e5ec0f5b36d61c09695a64ed63c509cec2f5b6c0251bd1…

           Birthyear Postcode UK.Country
     0  [1975, 1995)       LS    England
     1  [1995, 2015)        M    England
     2  [1975, 1995)       SO    England
     3  [1995, 2015)        B    England
     4  [1955, 1975)       TQ    England
```

1.0.8 Data aggregation - grouping education level and country of birth

```
[ ]: # Assign education level as banded education level
     anon_data['Education.Level'] = original_data['education_level'].map(lambda x:␣
      ↪"Higher" if x in ["bachelor", "masters", "phD"] else "BasicOther")

     # Assign hemisphere of birth depending on country of birth
```

```
anon_data['Location.of.Birth'] = original_data['country_of_birth'].apply(lambda␣
 ↪x: country_to_hemisphere(x))

anon_data.head()
```

```
[ ]:   Gender  CC.Status                                          Sample.ID  \
     0      F          0  f62e57d1aa65676b9db9986f73930551d196fa8db9eaa8...
     1      M          0  40a8ffbc83728762c7f690a9537d536f3ba13ec3dacb57...
     2      F          0  8d8e6615e9df3aaafb73eaf0a86b329bb7d57e95a4f833...
     3      F          0  37035c2be594b97d187b5bec3aeec641ebe519fb374229...
     4      F          0  e5ec0f5b36d61c09695a64ed63c509cec2f5b6c0251bd1...

            Birthyear Postcode UK.Country Education.Level    Location.of.Birth
     0  [1975, 1995)       LS    England          Higher  Northern Hemisphere
     1  [1995, 2015)        M    England      BasicOther  Northern Hemisphere
     2  [1975, 1995)       SO    England          Higher  Northern Hemisphere
     3  [1995, 2015)        B    England      BasicOther  Northern Hemisphere
     4  [1955, 1975)       TQ    England      BasicOther  Southern Hemisphere
```

1.0.9 Data perturbation - addition of Gaussian noise

```
[ ]: # Add gaussian noise to weight, height, countries visited, average number of␣
      ↪drinks in alcohol units per week and average cigrettes smoked per week.
     weight_noise = np.random.normal(0,1,1000)*5
     anon_data['Weight'] = round(original_data['weight']+weight_noise, 1)

     height_noise = np.random.normal(0,1,1000)/5
     anon_data['Height'] = round(original_data['height']+height_noise, 2)

     countries_noise = np.random.normal(0,1,1000)*5
     anon_data['Countries.Visited'] =␣
      ↪round(original_data['n_countries_visited']+countries_noise)

     alcohol_noise = np.random.normal(0,1,1000)
     anon_data['Avg.Alcohol'] =␣
      ↪round(original_data['avg_n_drinks_per_week']+alcohol_noise, 1)

     smoking_noise = np.random.normal(0,1,1000)*20
     anon_data['Avg.Cigarettes'] =␣
      ↪round(original_data['avg_n_cigret_per_week']+smoking_noise)

     anon_data.head()
```

```
[ ]:   Gender  CC.Status                                          Sample.ID  \
     0      F          0  f62e57d1aa65676b9db9986f73930551d196fa8db9eaa8...
     1      M          0  40a8ffbc83728762c7f690a9537d536f3ba13ec3dacb57...
```

```
2      F          0  8d8e6615e9df3aaafb73eaf0a86b329bb7d57e95a4f833…
3      F          0  37035c2be594b97d187b5bec3aeec641ebe519fb374229…
4      F          0  e5ec0f5b36d61c09695a64ed63c509cec2f5b6c0251bd1…

      Birthyear Postcode UK.Country Education.Level    Location.of.Birth  \
0  [1975, 1995)       LS    England          Higher  Northern Hemisphere
1  [1995, 2015)        M    England      BasicOther  Northern Hemisphere
2  [1975, 1995)       SO    England          Higher  Northern Hemisphere
3  [1995, 2015)        B    England      BasicOther  Northern Hemisphere
4  [1955, 1975)       TQ    England      BasicOther  Southern Hemisphere

   Weight  Height  Countries.Visited  Avg.Alcohol  Avg.Cigarettes
0    71.8    1.61               43.0          5.8           190.0
1    78.2    1.97               38.0          0.8            19.0
2   104.0    2.12               18.0          7.2            51.0
3    67.7    1.48               35.0          5.5           295.0
4   102.1    2.03               41.0          4.1           356.0
```

1.0.10 Calculating K-anonymity using quasi-identifiers

The following code groups the quasi-identifiers specificied and returns a count of the "unique" rows.

[ ]:

```
# Checking k-anonymity for quasi-identifiers
df_count = anon_data.groupby(['Gender', 'Birthyear', 'Location.of.Birth',
                             'UK.Country', 'Education.Level']).size().
 ↪reset_index(name = 'Count')

# Print rows where k-anonymity is 1
print(df_count[df_count['Count']==1])

# Printing the final grouped output
print("Final grouped output in ascending order of 'Count' - ")
df_count.sort_values("Count")
```

```
Empty DataFrame
Columns: [Gender, Birthyear, Location.of.Birth, UK.Country, Education.Level,
Count]
Index: []
Final grouped output in ascending order of 'Count' -
```

[ ]:

```
   Gender     Birthyear    Location.of.Birth UK.Country Education.Level  Count
22      F  [1995, 2015)  Southern Hemisphere      Other      BasicOther      0
47      M  [1995, 2015)  Southern Hemisphere      Other          Higher      2
31      M  [1955, 1975)  Southern Hemisphere      Other          Higher      2
46      M  [1995, 2015)  Southern Hemisphere      Other      BasicOther      2
19      F  [1995, 2015)  Northern Hemisphere      Other          Higher      2
39      M  [1975, 1995)  Southern Hemisphere      Other          Higher      3
```

```
23       F  [1995, 2015)  Southern Hemisphere      Other         Higher      3
7        F  [1955, 1975)  Southern Hemisphere      Other         Higher      4
30       M  [1955, 1975)  Southern Hemisphere      Other     BasicOther      4
38       M  [1975, 1995)  Southern Hemisphere      Other     BasicOther      4
6        F  [1955, 1975)  Southern Hemisphere      Other     BasicOther      4
45       M  [1995, 2015)  Southern Hemisphere    England         Higher      5
43       M  [1995, 2015)  Northern Hemisphere      Other         Higher      5
15       F  [1975, 1995)  Southern Hemisphere      Other         Higher      5
3        F  [1955, 1975)  Northern Hemisphere      Other         Higher      6
11       F  [1975, 1995)  Northern Hemisphere      Other         Higher      6
14       F  [1975, 1995)  Southern Hemisphere      Other     BasicOther      8
21       F  [1995, 2015)  Southern Hemisphere    England         Higher      8
42       M  [1995, 2015)  Northern Hemisphere      Other     BasicOther      9
20       F  [1995, 2015)  Southern Hemisphere    England     BasicOther      9
29       M  [1955, 1975)  Southern Hemisphere    England         Higher     11
10       F  [1975, 1995)  Northern Hemisphere      Other     BasicOther     12
2        F  [1955, 1975)  Northern Hemisphere      Other     BasicOther     12
44       M  [1995, 2015)  Southern Hemisphere    England     BasicOther     12
35       M  [1975, 1995)  Northern Hemisphere      Other         Higher     13
17       F  [1995, 2015)  Northern Hemisphere    England         Higher     13
18       F  [1995, 2015)  Northern Hemisphere      Other     BasicOther     14
5        F  [1955, 1975)  Southern Hemisphere    England         Higher     14
34       M  [1975, 1995)  Northern Hemisphere      Other     BasicOther     17
27       M  [1955, 1975)  Northern Hemisphere      Other         Higher     17
26       M  [1955, 1975)  Northern Hemisphere      Other     BasicOther     19
13       F  [1975, 1995)  Southern Hemisphere    England         Higher     19
41       M  [1995, 2015)  Northern Hemisphere    England         Higher     19
36       M  [1975, 1995)  Southern Hemisphere    England     BasicOther     21
37       M  [1975, 1995)  Southern Hemisphere    England         Higher     23
28       M  [1955, 1975)  Southern Hemisphere    England     BasicOther     23
12       F  [1975, 1995)  Southern Hemisphere    England     BasicOther     24
40       M  [1995, 2015)  Northern Hemisphere    England     BasicOther     35
4        F  [1955, 1975)  Southern Hemisphere    England     BasicOther     37
16       F  [1995, 2015)  Northern Hemisphere    England     BasicOther     39
25       M  [1955, 1975)  Northern Hemisphere    England         Higher     44
9        F  [1975, 1995)  Northern Hemisphere    England         Higher     46
33       M  [1975, 1995)  Northern Hemisphere    England         Higher     48
1        F  [1955, 1975)  Northern Hemisphere    England         Higher     55
0        F  [1955, 1975)  Northern Hemisphere    England     BasicOther     77
24       M  [1955, 1975)  Northern Hemisphere    England     BasicOther     79
8        F  [1975, 1995)  Northern Hemisphere    England     BasicOther     82
32       M  [1975, 1995)  Northern Hemisphere    England     BasicOther     84
```

1.0.11 Viewing the final anonymised dataset

[ ]:
```
# Re-order columns
anon_data = anon_data[['Sample.ID', 'Gender', 'Birthyear', 'Location.of.Birth',␣
 ↪'UK.Country', 'Weight',
                        'Height', 'Education.Level', 'Avg.Alcohol', 'Avg.
 ↪Cigarettes', 'Countries.Visited', 'CC.Status']]

# View the anonymised dataset
anon_data.head()
```

[ ]:
```
                                           Sample.ID Gender     Birthyear  \
0  f62e57d1aa65676b9db9986f73930551d196fa8db9eaa8…      F  [1975, 1995)
1  40a8ffbc83728762c7f690a9537d536f3ba13ec3dacb57…      M  [1995, 2015)
2  8d8e6615e9df3aaafb73eaf0a86b329bb7d57e95a4f833…      F  [1975, 1995)
3  37035c2be594b97d187b5bec3aeec641ebe519fb374229…      F  [1995, 2015)
4  e5ec0f5b36d61c09695a64ed63c509cec2f5b6c0251bd1…      F  [1955, 1975)

    Location.of.Birth UK.Country  Weight  Height Education.Level  \
0  Northern Hemisphere    England    71.8    1.61          Higher
1  Northern Hemisphere    England    78.2    1.97      BasicOther
2  Northern Hemisphere    England   104.0    2.12          Higher
3  Northern Hemisphere    England    67.7    1.48      BasicOther
4  Southern Hemisphere    England   102.1    2.03      BasicOther

   Avg.Alcohol  Avg.Cigarettes  Countries.Visited  CC.Status
0          5.8           190.0               43.0          0
1          0.8            19.0               38.0          0
2          7.2            51.0               18.0          0
3          5.5           295.0               35.0          0
4          4.1           356.0               41.0          0
```

1.0.12 Creating CSV files for the anonymised data and the reference table

[ ]:
```
# Output the files into .csv format
output_name = "anon_dataset"
anon_data.to_csv(output_name + ".csv", sep=",", index=None)

reference_table.to_csv("reference_table.csv", sep=",", index=None)
```

1.0.13 Encrypting and decrypting the dataset

[ ]:
```
# Encrypt csv and delete original file
encrypt(output_name + ".csv", output_name + "_encrypted.csv", "key.key")
os.remove(output_name + ".csv")
```

[ ]:
```
pip freeze > requirements.txt
```

```
Note: you may need to restart the kernel to use updated packages.
```