Hands-on Lab: Generative AI for Data Anonymization

Estimated Effort: 30 minutes

Introduction

Other than creating synthetic data, which by its base nature takes care of anonymization requirements, you can use generative AI platforms to create scripts that can modify the personally identifiable information (PII) in a given data set. In this lab, you will make use of the generative AI platform to create Python scripts that can be used to apply different anonymization strategies to the PII attributes in the data.

Objectives

In this lab, you will learn how to introduce anonymization in data using techniques like

1. Pseudonymization
2. Redaction
3. Generalization
4. Noise addition

Data set

The data set being used in this lab has been synthetically created using `faker` library in Python. Please note, that none of the entries in the data set have any bearing to any inidividual whatsoever.

The data set is available on the link shared below.
Synthetic data set

Attributes of the data set are:

Attribute	Description
Name	Name of the candidate
Email	Email address of the candidate
Age	Age of the candidate
Contact Number	Contact number of the candidate

Test platform

The code in this lab can be tested in the subsequent lab platform available in the course. Please open that lab in a separate code shell and use that platform to test the Python codes generated in this one.

Pseudonymization

Pseudonymization involves replacing original entries in data with appropriate pseudonyms. This assignment completely removes the original entry and protects the privacy of the user. We can use this technique to convert the names of the candidates into pseudonyms.

You can simply use the GPT model to create a code that will do that for you. Assume that you want to replace the names in the data set with a pseudonym `User_i` where `i` represents the `i th` entry in the data. Consider the following prompt to generate the code that can achieve this.

```
Replace the entries under 'Name' attribute of a dataset into pseudonyms like "User_i" using Python
```

The output generated should look like this.

```
import pandas as pd
# Read the dataset into a pandas DataFrame
df = pd.read_csv('your_dataset.csv')
# Replace the entries under the 'Name' attribute with pseudonyms like "User_i"
df['Name'] = ['User_' + str(i) for i in range(1, len(df) + 1)]
# Save the modified DataFrame back to a new CSV file
df.to_csv('modified_dataset.csv', index=False)
```

You can appropriately modify this code as per requirement (update the filename, file path and destination file path) and try it on the testing interface as described on the introduction page of this lab. The source file location can be used as the URL of the file shared before.

A sample output along with the modifications in the code is shown in the image below.

Code with modification

```
import pandas as pd
# Read the dataset into a pandas DataFrame
df = pd.read_csv('https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBMSkillsNetwork-AI0273EN-SkillsNetwork/labs/v1/m1/data/synthetic_dataset.csv')
print(df.head())
# Replace the entries under the 'Name' attribute with pseudonyms like "User_i"
df['Name'] = ['User_' + str(i) for i in range(1, len(df) + 1)]
# Print the first 5 entries of the modified dataframe
print('Modified dataset')
print(df.head())
```

Ouptut

```
theia@theia-abhishekg1:/home/project$ python3 test_file.py
              Name                      Email  Age  Contact Number
0  Brenda Richards    michelle76@example.org   79      9898586166
1    Antonio Perez    psingleton@example.net   19      9876282758
2     Terry Monroe    edwardross@example.net   30      9782846470
3    Heather Floyd    cookbrooke@example.net   65      9739572462
4    Allen Shelton  craigcollins@example.net   63      9676063153
Modified dataset
     Name                     Email  Age  Contact Number
0  User_1    michelle76@example.org   79      9898586166
1  User_2    psingleton@example.net   19      9876282758
2  User_3    edwardross@example.net   30      9782846470
3  User_4    cookbrooke@example.net   65      9739572462
4  User_5  craigcollins@example.net   63      9676063153
```

Redaction

A common method of anonymization is redacting parts of the information so as to protect the personal information in a data. You can test this out by redacting the email addresses of the candidates in the record such that only the first and last characters of the username and the service providers are visible. The rest of the characters are replaced with the character '*'.

Consider the following prompt to achieve this.

```
Write a python code to redact the entries under the attribute 'Email' in a dataframe such that only the first and last characters of the username and the service providers are visible. Rest all chara
```

The output code generated should look like this.

```
import pandas as pd
# Create a sample dataframe
data = {'Name': ['John', 'Alice', 'Bob'],
        'Email': ['john.doe@example.com', 'alice.smith@example.com', 'bob.jones@example.com']}
df = pd.DataFrame(data)
# Function to redact email addresses
```

```
def redact_email(email):
    username, domain = email.split('@')
    redacted_username = username[0] + '*'*(len(username)-2) + username[-1]
    redacted_domain = domain[0] + '*'*(len(domain)-2) + domain[-1]
    return redacted_username + '@' + redacted_domain
# Redact 'Email' column in the dataframe
df['Email'] = df['Email'].apply(redact_email)
# Display the redacted dataframe
print(df)
```

You can modify this code and use it in the testing environment to confirm that it infact redacts the email addresses in the dataframe. A necessary modification would be using only the function and the function call, and ignoring the import command and data frame creation, since both these steps have already been completed in the previous task. You can append this code to your existing code and see the result of both the processes in a single go.

The sample updated code and outputs are shown below.

```
import pandas as pd
# Read the dataset into a pandas DataFrame
df = pd.read_csv('https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBMSkillsNetwork-AI0273EN-SkillsNetwork/labs/v1/m1/data/synthetic_dataset.csv')
print(df.head())
# Replace the entries under the 'Name' attribute with pseudonyms like "User_i"
df['Name'] = ['User_' + str(i) for i in range(1, len(df) + 1)]
# Function to redact email addresses
def redact_email(email):
    username, domain = email.split('@')
    redacted_username = username[0] + '*'*(len(username)-2) + username[-1]
    redacted_domain = domain[0] + '*'*(len(domain)-2) + domain[-1]
    return redacted_username + '@' + redacted_domain
# Redact 'Email' column in the dataframe
df['Email'] = df['Email'].apply(redact_email)
# Print the first 5 entries of the modified dataframe
print('Modified dataset')
print(df.head())
```

Output

```
theia@theia-abhishekg1:/home/project$ python3 test_file.py
             Name                      Email  Age  Contact Number
0  Brenda Richards    michelle76@example.org   79      9898586166
1    Antonio Perez    psingleton@example.net   19      9876282758
2     Terry Monroe    edwardross@example.net   30      9782846470
3    Heather Floyd    cookbrooke@example.net   65      9739572462
4    Allen Shelton  craigcollins@example.net   63      9676063153
Modified dataset
     Name                      Email  Age  Contact Number
0  User_1    m********6@e*********g   79      9898586166
1  User_2    p********n@e*********t   19      9876282758
2  User_3    e********s@e*********t   30      9782846470
3  User_4    c********e@e*********t   65      9739572462
4  User_5  c**********s@e*********t   63      9676063153
```

Generalization

Generalization involves putting specific entries, which may be possible identifiers, into generic groups, such that the personal details in the records are protected. You can apply the generalization logic to the `Age` attribute of the said data set, and convert the specific age of the candidates into generic categories. For example, 28 can become 20s, 36 can become 30s, and so on.

You can create a code for this using the following prompt.

```
Write a python code to generalize the entries under the attribute 'Age' of a data frame such that exact number is converted into a generic range. For example, 28 becomes '20s', 36 becomes '30s', etc.
```

The output code generated should look like this.

```
import pandas as pd
# Create a sample dataframe
data = {'Name': ['John', 'Alice', 'Bob'],
        'Age': [28, 36, 42]}
df = pd.DataFrame(data)
# Function to generalize age
def generalize_age(age):
    age_range = str(age)[0] + '0s'
    return age_range
# Generalize 'Age' column in the dataframe
df['Age'] = df['Age'].apply(generalize_age)
# Display the generalized dataframe
print(df)
```

You can modify this code and use it in the testing environment to confirm that it in fact generalizes the age in the dataframe. A necessary modification would be using only the function and the function call, and ignoring the import command and dataframe creation, since both these steps have already been completed in the first task. You can append this code to your existing code and see the result of all the processes in a single go.

The sample updated code and outputs are shown below.

```
import pandas as pd
# Read the dataset into a pandas DataFrame
df = pd.read_csv('https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBMSkillsNetwork-AI0273EN-SkillsNetwork/labs/v1/m1/data/synthetic_dataset.csv')
print(df.head())
# Replace the entries under the 'Name' attribute with pseudonyms like "User_i"
df['Name'] = ['User_' + str(i) for i in range(1, len(df) + 1)]
# Function to redact email addresses
def redact_email(email):
    username, domain = email.split('@')
    redacted_username = username[0] + '*'*(len(username)-2) + username[-1]
    redacted_domain = domain[0] + '*'*(len(domain)-2) + domain[-1]
    return redacted_username + '@' + redacted_domain
# Redact 'Email' column in the dataframe
df['Email'] = df['Email'].apply(redact_email)
# Function to generalize age
def generalize_age(age):
    age_range = str(age)[0] + '0s'
    return age_range
# Generalize 'Age' column in the dataframe
df['Age'] = df['Age'].apply(generalize_age)
# Print the first 5 entries of the modified dataframe
print('Modified dataset')
print(df.head())
```

Output

```
theia@theia-abhishekg1:/home/project$ python3 test_file.py
             Name                     Email  Age  Contact Number
0  Brenda Richards    michelle76@example.org   79      9898586166
1    Antonio Perez    psingleton@example.net   19      9876282758
2     Terry Monroe    edwardross@example.net   30      9782846470
3    Heather Floyd    cookbrooke@example.net   65      9739572462
4    Allen Shelton  craigcollins@example.net   63      9676063153
Modified dataset
     Name                      Email  Age  Contact Number
0  User_1    m********6@e*********g  70s      9898586166
1  User_2    p********n@e*********t  10s      9876282758
2  User_3    e********s@e*********t  30s      9782846470
3  User_4    c********e@e*********t  60s      9739572462
4  User_5  c**********s@e*********t  60s      9676063153
```

Noise addition

Another way to anonymize the data is to add random noise to it. This converts the original data into unusable garbage data and is effective in protecting the privacy of the candidate. You can apply such random addition to the attribute 'Contact Number' in the data set.

Assuming that all contact numbers are numerical values of 10 digits length, you need to add a random noise of length five digits to it. To create a Python code that can do this, you can use the following prompt on the GPT system.

```
Write a python code to add random noise of 5 digit length to a numerical attribute 'Contact Number' in a data frame which had all values of length 10 digits.
```

The output code generated should look like this.

```
import pandas as pd
import random
# Create a sample dataframe
data = {'Name': ['John', 'Alice', 'Bob'],
        'Contact Number': [1234567890, 9876543210, 2468101214]}
df = pd.DataFrame(data)
# Function to add random noise
def add_random_noise(contact_number):
    noise = str(random.randint(10000, 99999))
    return str(contact_number)[:-5] + noise
# Add random noise to 'Contact Number' column in the dataframe
df['Contact Number'] = df['Contact Number'].apply(add_random_noise)
# Display the dataframe with added noise
print(df)
```

You can modify this code and use it in the testing environment to confirm that it adds noise to the contact number in the dataframe. A necessary modification would be using only the function and the function call, and ignoring the pandas import command and dataframe creation, since both these steps have already been completed in the first task. The import command for `random` will still be needed for the function to work. You can append this code to your existing code and see the result of all the processes in a single go.

The sample updated code and outputs are shown below.

```
import pandas as pd
import random
# Read the dataset into a pandas DataFrame
df = pd.read_csv('https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBMSkillsNetwork-AI0273EN-SkillsNetwork/labs/v1/m1/data/synthetic_dataset.csv')
print(df.head())
# Replace the entries under the 'Name' attribute with pseudonyms like "User_i"
df['Name'] = ['User_' + str(i) for i in range(1, len(df) + 1)]
# Function to redact email addresses
def redact_email(email):
    username, domain = email.split('@')
    redacted_username = username[0] + '*'*(len(username)-2) + username[-1]
    redacted_domain = domain[0] + '*'*(len(domain)-2) + domain[-1]
    return redacted_username + '@' + redacted_domain
# Redact 'Email' column in the dataframe
df['Email'] = df['Email'].apply(redact_email)
# Function to generalize age
def generalize_age(age):
    age_range = str(age)[0] + '0s'
    return age_range
# Generalize 'Age' column in the dataframe
df['Age'] = df['Age'].apply(generalize_age)
def add_random_noise(contact_number):
    noise = str(random.randint(10000, 99999))
    return str(contact_number)[:-5] + noise
# Add random noise to 'Contact Number' column in the dataframe
df['Contact Number'] = df['Contact Number'].apply(add_random_noise)
# Print the first 5 entries of the modified dataframe
print('Modified dataset')
print(df.head())
```

Output

```
theia@theia-abhishekg1:/home/project$ python3 test_file.py
             Name                       Email  Age  Contact Number
0  Brenda Richards    michelle76@example.org   79      9898586166
1    Antonio Perez    psingleton@example.net   19      9876282758
2     Terry Monroe    edwardross@example.net   30      9782846470
3    Heather Floyd    cookbrooke@example.net   65      9739572462
4    Allen Shelton  craigcollins@example.net   63      9676063153
Modified dataset
     Name                      Email  Age Contact Number
0  User_1    m********6@e********g  70s     9898568297
1  User_2    p********n@e*********t  10s     9876215035
2  User_3    e********s@e*********t  30s     9782863880
3  User_4    c********e@e*********t  60s     9739566757
4  User_5  c**********s@e*********t  60s     9676047700
theia@theia-abhishekg1:/home/project$
```

Practice exercises

Try to create the codes for the following tasks using generative AI model and test them on the testing interface.

1. Redact the `Name` attribute such that only the vowels are visible and rest everything is replaced by the character `#`
2. Assign Pseudonyms in place of the `Email` attribute, such that all email IDs are converted to `user_i@pseudo.com`, where `i` is the `i th` entry in the dataset.
3. Add random noise to the first five numbers of the `Contact Number` instead of the last five.

Conclusion

Congratulations on completing this lab.

By the end of this lab, you are now able to use generative AI to create Python codes that can anonymize data by using the following strategies.

1. Pseudonymization
2. Redaction
3. Generalization
4. Noise addition

Author(s)

Abhishek Gagneja